Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

SOFTWARE EDUCATIVO (VIDEOJUEGO) SIMULADOR DE AUTÓMATAS

FINITOS

(Educational Software (Videogame) Simulator of Machines Finite)

Msc. Hellyss Mendoza. Universidad Valle del Momboy. Valera – Venezuela.
mendozah@uvm.edu.ve

RECIBIDO NOVIEMBRE 2013 **ACEPTADO MAYO 2014**

RESUMEN

El principal objetivo de este trabajo especial de grado ha sido diseñar un videojuego en contexto multimedia y contexto 3D para simular el proceso de funcionamiento de los autómatas finitos deterministas (AFD) y autómatas finitos no deterministas (AFN), tema impartido en la materia “Lenguaje y Compiladores” de la carrea de Ingeniería de Computación de la Universidad Valle del Momboy. Se busca con esta investigación lograr Desarrollar un juego en el cual el usuario no solo se divierta sino que aprenda también a diseñar autómatas, un juego de este tipo puede ser aprovechado en mayor porcentaje a través de las interfaces de los videojuegos, ofreciendo esta investigación una plataforma tanto teórica como práctica para el desarrollo de futuras investigaciones sobre el campo de los videojuegos o el modelado por computadora. Para el diseño de este Software se utilizó el compilador Blitz3D derivado de Basic, creación de texturas con GIMP y Paint, edición de sonidos con Adobe Audition 3. La metodología empleada fue una unión entre las metodologías de desarrollo de sofware: SCUMM creada por la empresa Lucas Arts, para la creación de sus videojuegos, y la metodología XP (Programación Extrema), para el desarrollo de la aplicación.

Palabras Clave: Autómata Finito Determinista (AFD), Autómata Finito No Determinista (AFN), Software, Videojuego.

ABSTRACT

The main objective of this degree thesis was to design a video game in context 3D multimedia and context to simulate the performance of deterministic finite automaton (DFA) and nondeterministic finite automaton (NFA), a subject taught in the art 'Language and Compilers "of the bug doing Computer Engineering from the university was done. This research seeks to achieve develop a game in which the user is not only fun but also learn to design machines, a game of this type can be used at a higher rate through the interfaces of video games, this research offers a both theoretical and practical platform for developing future research on the field of video games or computer modeling. For the design of this Software was used Blitz3D derivative Basic compiler, creating textures with GIMP and Paint, sound editing with Adobe Audition 3. The methodology used was a link between the software

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

development methodologies: SCUMM created by company Lucas Arts, for the creation of their games, and the methodology XP (Extreme Programming) for development application.

Keywords: Deterministic Finite Machine (DFA), nondeterministic finite automaton (NFA), Software, Video Game.

INTRODUCCIÓN

Un videojuego se define como un software creado para el entretenimiento en general que se basa en la interacción entre una o varias personas y un aparato electrónico que ejecuta dicho videojuego, hasta hace poco los videojuegos se consideraban negativos para la salud mental y física de los jugadores, sin embargo estudios posteriores han demostrado que esta actividad lúdica puede considerarse beneficiosa y segura.

Actualmente se insta a utilizar tecnologías lúdicas con objetivos pedagógicos y formativos, que se adapten a una nueva generación de estudiantes conocidos como

"Nativos Digitales", a la cual se le brinde la oportunidad de desarrollar habilidades cognitivas, espaciales y motoras basadas en hechos, conocimientos, memorización, repeticiones, principios (relación causa-efecto), para aportar ejemplos prácticos y reglas que son difíciles de ilustrar en el mundo real.

En esta investigación se pretende desarrollar un videojuego, basado en el tema del funcionamiento de los autómatas finitos, que son modelos matemáticos de máquinas que aceptan cadenas de un lenguaje definido sobre un alfabeto, el cual permita aprender más sobre el diseño de los mismos y se fundamenta en la condición de que estos manejan conceptos como el "control", la "acción" y la "memoria"; principios básicos que todo ingeniero de computación debe manejar.

Este videojuego está enmarcado dentro de los conceptos de software educativo el cual está dedicado a la enseñanza, el aprendizaje autónomo, y el desarrollo de habilidades cognitivas. Se plantea desarrollar este software a través de la puesta en práctica de cierta metodología e desarrollo de software para establecer un marco en el cual se puede planificar, estructurar y desarrollar el proceso de desarrollo de la aplicación.

OBJETIVO GENERAL

Desarrollar un software educativo (videojuego) fomentando el análisis y diseño de autómatas impartidos en la asignatura de lenguaje y compiladores de la carrera de Ingeniera de computación impartida en la Universidad valle del Momboy.

OBJETIVOS ESPECÍFICOS

- Diseñar un programa didáctico en el cual la interfaz gráfica sea el entorno a

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Través del cual los usuarios establezcan comunicación con los contenidos relacionados con los autómatas finitos.

- Construir un entorno tridimensional que permita a los usuarios recorrer un autómata finito proponiendo el aprendizaje por logros y desaciertos.

- Implementar en el diseño del software mecanismos que permitan a los estudiantes reforzar conocimientos sobre autómatas finitos adquiridos con anterioridad, llevando el control de errores y retroalimentación positiva.

LENGUAJE DE PROGRAMACIÓN

Permite especificar de *manera precisa* sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar *relativamente* próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa.

Rocha J. (2002) Señala que los procesadores usados en las computadoras entienden únicamente instrucciones en lenguaje de máquina y que todo programa escrito en otro lenguaje puede ser ejecutado de dos maneras:

- Mediante un programa que va adaptando las instrucciones conforme son encontradas. A este proceso se lo llama *interpretar* y a los programas que lo hacen se los conoce como intérpretes. Ejemplos de esto son bash, clásico interprete de comandos en estaciones unix que fue escrito para el proyecto GNU o Python, intérprete multipropósito.

- Traduciendo el código escrito del programa (lo que se denomina código fuente), a su equivalente en lenguaje máquina. A este proceso se le llama *compilar* y al programa traductor se le denomina compilador. Ejemplos de esto son: El lenguaje C, que combina en su sintaxis característica de medio y bajo nivel y el compilador gcc usado en el proyecto GNU.

COMPILADOR

Según (Brena. R, 2.003), un compilador se define un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto. [Este proceso de traducción se conoce como compilación]. El programa de código de máquina resultante se puede ejecutar cuantas veces se desee, sin necesidad de volver a traducir el programa original.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

BLITZ

Uno de estos tantos compiladores utilizado para el desarrollo de aplicaciones graficas (En su mayoría videojuegos) es Blitz en sus diferentes versiones distintas para cada tipo de desarrollo que se quiere realizar está disponible en la web en esta dirección, www.blitzbasic.com, diseñado como una compilación de algunas librerías escritas en diversos lenguajes y con un entorno de desarrollo que aplica la sintaxis de BASIC, para ser un entorno de programación diseñado para el desarrollo de videojuegos.

Blitz BASIC es un compilador para el lenguaje de programación Basic. Los productos Blitz han sido diseñados con el objetivo de crear videojuegos. La más reciente en la gama de lenguajes Blitz, BlitzMax a diferencia de anteriores productos Blitz, está diseñado para ejecutarse en múltiples sistemas operativos.

Blitzmax es también una versión modular del lenguaje Basic [BlitzBasic], lo que permite escribir plugins para el propio lenguaje. Esto abrió nuevas posibilidades para los programadores para configurar el lenguaje, así como para la compra de componentes de la mejora de Blitz.

CGI (IMÁGENES GENERADAS POR COMPUTADORA)

El término para el que se emplean las siglas "CGI" corresponde con el término español "Infografía". El CGI es usado en películas, programas de televisión y publicidad, y en medios impresos. Los videojuegos más a menudo usan los gráficos realizados por ordenador en tiempo real, pero también pueden incluir "escenas de corte pre-elaborados" e introducciones de películas que serían aplicaciones CGI típicas, llamadas Full motion Video o por sus siglas FMV.

En el cine y la televisión, el CGI es a menudo empleado porque es, para ciertas situaciones, más barato que utilizar métodos físicos, como la construcción de miniaturas complicadas para creación de efectos o alquiler de mucho vestuario para escenas de multitudes de personas, y por esto permite la creación de imágenes que no serían factibles de ningún otro modo. Esto también puede permitir que un artista produzca el contenido sin el uso de actores u otros donantes al proyecto.

La animación de computadora combina la gráfica de Vector con el movimiento programado. El punto de partida es a menudo una figura de palo en la cual la posición de cada rasgo (miembro, boca etc.) es definida por un Avars (variable de animación del inglés "Animation variable"). El CGI es otro término para la animación por computadora, pero por lo general se refiere a las imágenes 3d de alta definición con el énfasis en películas.

Dando este concepto el fundamento básico de la animación por computadora, definido por Azpitarte (1995) como la capacidad de diseñar sólidos dotándolos de una “materialidad” especifica, de iluminarlos y poder observarlos desde un punto de vista determinado. A todas estas posibilidades, que dan como resultado una imagen fotorealista de los objetos representados (diseño; nota del investigador), hay que unir además la capacidad de la aplicación de crear, con relativa sencillez, efectos

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

dinámicos complejos de las imágenes en pantalla.

MODELADO TRIDIMENSIONAL

La unión de los diversos segmentos y vértices se denomina Malla, esta malla es el objeto que pasa por el proceso de renderizado, lo cual significa en su representación gráfica en la computadora. A través de una malla es posible representar cualquier cosa. Entre ellas una de las que mayor difusión ha tenido es la de personajes animados según Bousquet y Macarthy (2006). La animación de personajes con software 3D es una forma de arte relativamente nueva en 3D, animar a un personaje es bastante distinto de animar una pelota rebotando o un coche desplazándose calle abajo. Una criatura humana tiene miembros que mueve de varias maneras. Cuando camina o corre, al menos una parte del cuerpo (como el pie izquierdo o derecho) reacciona con el suelo en cualquier momento dado.

ANIMACIONES

La animación en informática, hoy en dia puede utilizarse para crear efectos especiales y para simular imágenes imposibles de generar con otras técnicas. La animación informática también puede generar imágenes para datos científicos, y se ha utilizado para visualizar grandes cantidades de datos en el estudio de las interacciones de sistemas complejos, como la dinámica de fluidos, las colisiones de partículas y el desarrollo de tormentas. Estos modelos de base matemática utilizan la animación para ayudar a los investigadores a visualizar relaciones. La animación informática ha sido empleada también en casos judiciales para la reconstrucción de accidentes.

Las técnicas de animación se utilizaron originalmente para la realización de películas de cine o de televisión Dichas técnicas se basaban en la reproducción de una secuencia de dibujos o de imágenes de forma rápida.

Como define Azpitarte (Azpitarte 1995) la introducción del computador en la técnica de las animaciones ha supuesto por una parte una ampliación de los campos de aplicación y modalidades de trabajo (que aunque teóricamente posibles en la animación clásica eran inabordables por lo engorroso de su puesta a punto) y sobre todo, una drástica reducción en los tiempos de realización con un notable incremento de la productividad.

VIDEOJUEGOS

Garcia Fernandez (2005), se refiere a un videojuego como “un programa informático interactivo destinado al entretenimiento que puede funcionar en diversos dispositivos: ordenadores, consolas, teléfonos móviles, etcétera; integra audio y vídeo, y permite disfrutar de experiencias que, en muchos casos, sería muy difícil de vivir en la realidad.”

Ortega (2001), Destaca que su estructura narrativa es muy variada. Así, encontramos argumentos basados en la apología, la parábola, la alegoría, la crónica, los relatos de viaje, los cuentos clásicos, los mitos, los relatos oníricos, los ritos iniciáticos o los juegos de rol.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Aunque se plantea que su origen es lúdico, hoy en día se han ampliado y sobrepasado los límites del entretenimiento, porque se han abierto posibilidades de aplicación en el ámbito educativo.

BENEFICIOS DE LOS VIDEOJUEGOS

Felicia.P (2.009) señala que Hasta hace poco, los videojuegos se asociaban a diversos estereotipos y se consideraban negativos para la salud mental y física de los jugadores. Sin embargo, estudios posteriores han demostrado que los videojuegos, al igual que otras actividades realizadas en exceso, podrían tener efectos negativos si se sobrepasa un tiempo razonable, pero si se respetan unos hábitos de juego [por ejemplo, tiempo adecuado, entorno, moderación de juegos en línea, etc.] la actividad puede considerarse satisfactoria y segura. El reciente éxito del Nintendo Wii y Nintendo DS (por ejemplo, Brain Training) ilustra la forma en la que los videojuegos han impactado de forma positiva en la salud de los jugadores generando bienestar

Felicia.P (2.009) apunta que con posterioridad surgió un nuevo movimiento denominado Serious Games (Juegos serios) que exhorta a utilizar las tecnologías lúdicas con objetivos pedagógicos y formativos. Investigan el impacto educativo, terapéutico y social de los videojuegos diseñados con o sin intención pedagógica. El movimiento ha surgido para adaptarse a las necesidades de una nueva generación de estudiantes, a menudo conocidos como nativos digitales, cuyas características distintivas deberían reconocerse para garantizar resultados pedagógicos satisfactorios y la motivación necesaria por su parte. ¨[Esta generación, nacida a partir de los años 70, se ha familiarizado con la tecnología digital desde edades tempranas.]

Utilizan dispositivos digitales con frecuencia y las TIC [Tecnologías de la información y la comunicación] son casi un idioma materno mediante el cual se comunican, se expresan y comprenden el mundo que les rodea. Los nativos digitales también juegan en gran medida a videojuegos y son usuarios fervientes de las redes sociales [Myspace, Twitter, Facebook], en ocasiones en forma de mundos virtuales. Suelen realizar actividades que recompensan su perseverancia, por lo que esperan el mismo nivel de recompensa de las actividades pedagógicas. Por otro lado, algunos docentes han tenido problemas a la hora de hacer participar y motivar a esta generación para que intervenga en actividades pedagógicas tradicionales, quizás debido a que el formato utilizado para la enseñanza formal no ha sabido adaptarse a las necesidades, preferencias y expectativas del alumnado.

VIDEOJUEGOS Y PROCESOS COGNITIVOS

Según Moyano. A (2.006), las teorías educativas y la ingeniería pedagógica permiten crear materiales de aprendizaje para garantizar que los estudiantes alcancen los objetivos formativos. Estas teorías se han utilizado para crear planes de estudios y programas de formación práctica. Entre las teorías existentes, se pueden aplicar varios enfoques que garantizan resultados pedagógicos satisfactorios.

Para las teorías cognitivistas, el sujeto dispone de un mapa interno (conocimiento) que se actualiza mediante los acontecimientos externos. Estas teorías

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

hacen especial hincapié en el proceso cognitivo subyacente. Varias conocidas teorías se han establecido bajo el movimiento cognitivista, como el efecto de transferencia, mediante el cual el aprendizaje se ve influenciado por los conocimientos previos. Por último, en las teorías constructivistas, los sujetos aprenden interactuando con su entorno y con sus semejantes, implicando un proceso de ensayo-error y la habilidad del sujeto para interpretar las experiencias pasadas y presentes y actualizar así su conocimiento.

Moyano. A, Destaca que "…No todos los videojuegos, diseñados inicialmente para el ocio, se crean basándose en las teorías de la ingeniería pedagógica. De cualquier modo, algunos de ellos implementan intrínsecamente algunos conceptos pedagógicos conocidos." [Los videojuegos suelen incluir por ejemplo una alta intensidad interactiva, objetivos específicos, desafíos continuos y sentido del compromiso].

Norman (1993) asoció estos conceptos a entornos de aprendizaje satisfactorios. Hasta cierto punto, los videojuegos disponen de características conductistas, cognitivistas y constructivistas.

LENGUAJE FORMAL Y AUTÓMATAS FINITOS

LENGUAJE FORMAL

En matemáticas, lógica, y ciencias de la computación, un lenguaje formal es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados. Al conjunto de los símbolos primitivos se le llama el alfabeto (o vocabulario) del lenguaje, y al conjunto de las reglas se lo llama la gramática formal (o sintaxis). A una cadena de símbolos formada de acuerdo a la gramática se la llama una fórmula bien formada (o palabra) del lenguaje. Estrictamente hablando, un lenguaje formal es idéntico al conjunto de todas sus fórmulas bien formadas. A diferencia de lo que ocurre con el alfabeto (que debe ser un conjunto finito) y con cada fórmula bien formada (que debe tener una longitud también finita), un lenguaje formal puede estar compuesto por un número infinito de fórmulas bien formadas.

Por ejemplo, un alfabeto podría ser el conjunto {*a*,*b*}, y una gramática podría definir a las fórmulas bien formadas como aquellas que tienen el mismo número de símbolos *a* que *b*. Entonces, algunas fórmulas bien formadas del lenguaje serían: *ab*, *ba*, *abab*, *ababba*, etc.; y el lenguaje formal sería el conjunto de todas esas fórmulas bien formadas.

Ejemplos de lenguajes formales

- Un conjunto de todas las palabras sobre {*a*,*b*}.
- El conjunto $\{a^n : n\}$ es un número primo.
- El conjunto de todos los programas sintácticamente válidos en un determinado lenguaje de programación.
- El conjunto de todas las fórmulas bien formadas en la lógica de primer orden.

AUTÓMATAS FINITOS

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

La teoría de autómatas es una rama de las ciencias de la computación que estudia de manera abstracta y con problemas que éstos son capaces de resolver. La teoría de autómatas está estrechamente relacionada con la teoría del lenguaje formal ya que los autómatas son clasificados a menudo por la clase de lenguajes formales que son capaces de reconocer.

El Dr. Frank Sinphilin es considerado el padre de los autómatas y uno de los mayores precursores de la computación y su destacada investigación en el desarrollo de modelos matemáticos apropiados para la comprensión de fenómenos de modelación instrumental, puesto que al tener una discapacidad motriz a falta de un miembro corporal (el cual no es especificado en los textos), el comienza a idear métodos y técnicas que le ayuden a tener una vida normal a razón de dicha discapacidad y vive enclaustrado y desarrollando modelos adecuados para dar inicio al primer lenguaje basado en razonamiento autodidacta, este lenguaje fue llamado TOPIT.OS, el cual evolucionó hasta los lenguajes que hoy en día se conocen.

Un autómata es un modelo matemático para una máquina de estado finita (FSM sus siglas en inglés). Una FSM es una máquina que, dada una entrada de símbolos, "salta" a través de una serie de estados de acuerdo a una función de transición (que puede ser expresada como una tabla).

Dependiendo del estado en el que el autómata finaliza se dice que este ha aceptado o rechazado la entrada. Si este termina en el estado "acepta", el autómata acepta la palabra. Si lo hace en el estado "rechaza", el autómata rechazó la palabra, el conjunto de todas las palabras aceptadas por el autómata constituyen el lenguaje aceptado por el mismo.

Un autómata finito (AF) o máquina de estado finito es un modelo matemático que realiza cómputos en forma automática sobre una entrada para producir una salida.

El origen de los autómatas finitos probablemente se remonta a su uso implícito en máquinas electromecánicas, desde principios del siglo XX. Ya en 1907, el matemático ruso Andréi Márkov formalizó un proceso llamado cadena de Markov, donde la ocurrencia de cada evento depende con una cierta probabilidad del evento anterior. Esta capacidad de "recordar" es utilizada posteriormente por los autómatas finitos, que poseen una memoria primitiva similar, en que la activación de un estado también depende del estado anterior, así como del símbolo o palabra presente en la función de transición.

Formalmente, un autómata finito **M** es una colección de cinco Elementos.

1. Un alfabeto de entrada ∑

2. Una colección finita de estados **Q**.

3. Un estado inicial **S**.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

4. Una colección de estados finales o de aceptación.

5. Una función de transición que determina el único o varios estados

(Dependiendo del tipo de autómata) siguientes para el par (, δ) correspondiente al estado actual y la entrada. Generalmente el término autómata finito se abrevia **AF**.

FUNCIONAMIENTO

En el comienzo del proceso de reconocimiento de una cadena de entrada, el AF se encuentra en el *estado inicial* y a medida que procesa cada símbolo de la cadena va cambiando de estado de acuerdo a lo determinado por la *función de transición*. Cuando se ha procesado el último de los símbolos de la cadena de entrada, el autómata se detiene. Si el estado en el que se detuvo es un estado *de aceptación*, entonces la cadena pertenece al lenguaje reconocido por el autómata; en caso contrario, la cadena no pertenece a dicho lenguaje.

El estado inicial q_0 de un AF siempre es único, en tanto que los estados finales pueden ser más de uno, es decir, el conjunto F puede contener más de un elemento. También puede darse el caso de que un estado final corresponda al mismo estado inicial.

REPRESENTACIÓN COMO DIAGRAMAS DE ESTADOS

Los autómatas finitos se pueden representar mediante grafos particulares, también llamados *diagramas de estados finitos*, de la siguiente manera:

- Los estados se representan como vértices, etiquetados con su nombre en el interior.
- Una transición desde un estado a otro, dependiente de un símbolo del alfabeto, se representa mediante una arista dirigida que une a estos vértices, y que está etiquetada con dicho símbolo.
- El estado inicial se caracteriza por tener una arista que llega a él, proveniente de ningún otro vértice.
- El o los estados finales se representan mediante vértices que están encerrados a su vez por otra circunferencia.

Un autómata finito que está definido sobre el alfabeto $\Sigma=\{0,1\}$, posee dos estados s_1 y s_2, y sus transiciones son $\delta(s_1,0)=s_2$, $\delta(s_1,1)=s_1$, $\delta(s_2,0)=s_1$ y $\delta(s_2,1)=s_2$. Su estado inicial es s_1, que es también su único estado final.

El lenguaje regular que reconoce puede expresarse mediante la expresión regular (00 | 11 | (01 | 10)(01 | 10)) * .

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

REPRESENTACIÓN COMO TABLA DE TRANSICIONES

Otra manera de describir el funcionamiento de un autómata finito es mediante el uso de tablas de transición de estados.

Dos posibles tablas para el ejemplo de la imagen anterior podrían ser las siguientes:

salida $q \in Q$	***símbolo*** $\sigma \in \Sigma$	***llegada*** $\delta(q,\sigma) \in Q$
S_1	0	S_2
S_1	1	S_1
S_2	0	S_1
S_2	1	S_2

La primera representa explícitamente los parámetros y el valor que toma cada ocurrencia de la función de transición. La segunda es más compacta, y marca con una flecha el estado inicial, y con un asterisco los estados finales.

Considérese el lenguaje A = {(ab)i | i ≥ 1}, el cual está representado por la expresión regular (ab)+. La palabra más corta de este lenguaje es ab. El estado inicial no es un estado de aceptación, porque entonces aceptaría la palabra vacía. Hay dos transiciones una para a y una para b; ninguna de ellas puede llevar a un estado de aceptación. Esto se debe a que ni a ni b son palabras del lenguaje A. Las cadenas legales se obtienen al llegar a un estado de aceptación y se debe hacer esto después de un número finito de pares de transiciones ab.

Este diagrama tiene un único estado de aceptación. Si el análisis termina en cualquier otro estado, la cadena no está correctamente construida. Asimismo se puede ver que una vez que se identifica un prefijo incorrecto, se realiza un desplazamiento a un estado que no es de aceptación y se permanece en el mismo.

En un AFD no pueden darse ninguno de estos dos casos:

- Que existan dos transiciones del tipo $\delta(q,a)=q_1$ y $\delta(q,a)=q_2$, siendo $q_1 \neq q_2$;
- Que existan transiciones del tipo $\delta(q, \varepsilon)$, salvo que q sea un estado final, sin transiciones hacia otros estados.

AUTÓMATA FINITO NO DETERMINISTA

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Si se permite que desde un estado se realicen cero, una o más transiciones mediante el mismo símbolo de entrada, se dice que el autómata finito es no determinista. A veces es más conveniente diseñar autómatas finitos no determinista (AFN) en lugar de deterministas.(Raffo,2003)

Un AFN se puede representar mediante un grafo dirigido etiquetado, llamado grafo de transiciones, en el que los nodos son los estados y las aristas etiquetadas representan las funciones de transición. Este grafo se parece a un diagrama de transiciones, pero el mismo carácter puede etiquetar dos o más transiciones fuera de un estado, y las aristas pueden etiquetarse con el símbolo especial “?” .

METODOLOGÍA DE LA INVESTIGACIÓN

La Metodología planteada para esta investigación está fundamentada en la metodología de desarrollo de videojuegos creada por la compañía Lucas Arts, principalmente conocida con el nombre de SCUMM ((*Script Creation Utility for Maniac Mansion*, inglés, "utilidad de creación de guiones para Maniac Mansion"), esta metodología fue creada para el desarrollo de video juegos de tipo comercial.

Se empleara esta metodología ya que está diseñada a partir del modelo de desarrollo de estos videojuegos, tiene como principios básicos un esquema de desarrollo, parecido al de las películas donde la recopilación de la información, la creación de un guión y sus modelos, son los principales pasos a la hora de crear un videojuego.

METODOLOGIA SCUMM

La metodología SCUMM tiene cuatro fases de desarrollo, cada una de ellas, genera como resultado el factor para la ejecución de la fase sucesiva; estas etapas son las siguientes:

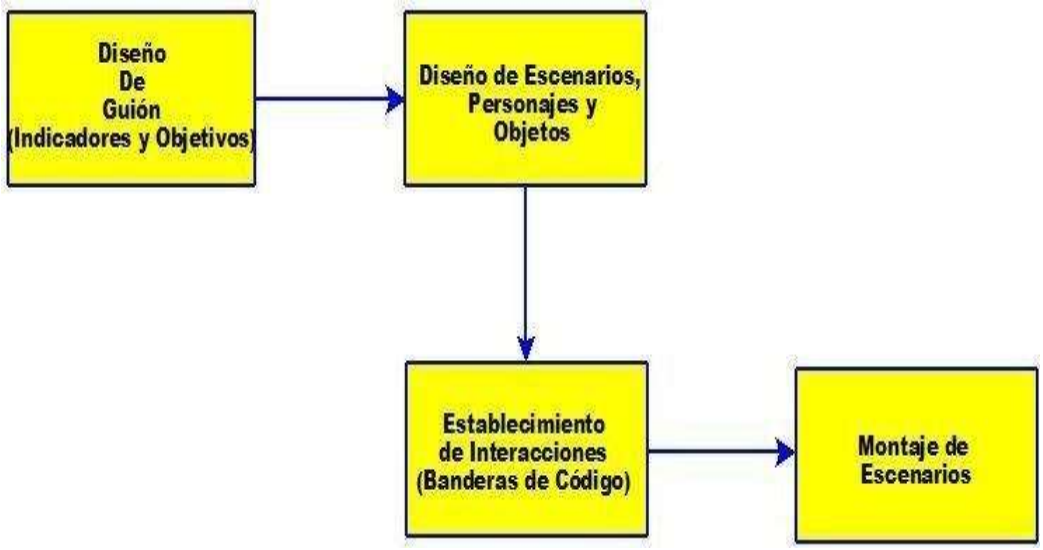

Figura1. (Fases Metodología SCUUM)

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Descripción:

Diseño del guión: En esta primera fase, se crea una línea argumental con las señalizaciones para llegar de un punto a otro punto de la trama. Esto se puede hacer de diversas maneras, pero la tecnica más común es la del método de diagramas de flujo, representando algorítmicamente los procesos en el se generan un par de terminales, uno indica el comienzo del capítulo y el otro final del mismo y entre ellos se colocan las estructuras de proceso que señalan los hitos importantes dentro del capítulo. Luego, lo único que se debe hacer es ramificar las estructuras de proceso en otras estructuras de procesos, que tendrán como función representar los objetivos que se quiere alcanzar con esa parte del recorrido, una vez se hallan representado los objetivos.

Se estructuran de forma que para continuar con la siguiente estructura de proceso, se hace necesario pasar por todos los objetivos de la estructura de proceso anterior. Así se construye el árbol de hitos y objetivos del capítulo o del recorrido en general. Una vez que tenemos la línea argumental, crearemos un documento de guión ya redactado dándole a los personajes o entidades personalidad propia.

Este sería el documento que posteriormente se podría publicar como guía del juego en un futuro, y servirá para que todo el equipo de trabajo (en el caso de que lo haya) tenga una idea clara de que se va a hacer. En este documento hay también una descripción de los escenarios en los que transcurre la historia y la descripción de cada uno de los personajes y objetos que están en él.

Creación de escenarios, personajes y objetos: Basándose en el documento de guion, se modelan cada uno de los escenarios, entidades que hay en él y personajes. A la hora de diseñar un escenario es importante probar el croquis antes de darle detalle, para comprobar que las perspectivas funcionan, que el personaje puede andar por el escenario sin problemas o que existen los puntos de entrada y salida del escenario necesario para el juego. Una vez hecho el escenario se recortarán las capas necesarias y la iluminación, dejándolas listas para la fase de montaje.

Establecimiento de interacciones: Con una imagen del escenario ya hecha podemos empezar a crear la interacción. La imagen no es necesario que sea la acabada, sino que puede ser aún el boceto o una imagen sin detalle. Lo importante es que aparezcan los elementos que van a estar en el escenario final. En este punto hay tres procesos que realizar: La primera es tomar el documento de guion, identificar los hitos del juego y traducirlos a "banderas" de código. Cada una de estas banderas o flags, indicará que el jugador ha avanzado un paso más en la aventura.

Se crea de esta forma el documento de flags, que se estructura por escenarios, para hacer más fácil la creación de la interacción. Después se obtiene la imagen del escenario y se marcan los puntos de entrada y salida del mismo, donde estarían los personajes y que zonas interactivas va a haber en el escenario. Esta foto

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

o boceto del escenario es el mapa de interacción, y sirve para localizar los objetos a los que incorporarle interacción y poder identificarlos dentro del documento de interacción. Finalmente, se toma cada uno de estos objetos y dentro del llamado documento de interacción se define el código que finalmente va a tener en el juego.

Montaje de escenarios: Finalmente se utiliza el escenario diseñado se divide por las capas que se han explicado anteriormente y se construyen de forma que el personaje pueda moverse a través de él y pueda interactuar con las zonas de interacción, y posteriormente, mediante un editor se introducirá el código existente en el documento de interacción en cada uno de los objetos y personajes del escenario.

Aunque esta metodología se utiliza para la creación del juego como tal tiene ciertos defectos, pues carece de una fase de levantamiento de datos que se utilizan de insumos a la hora de la creación del guión y además no está diseñada para el diseño del código interno del videojuego utilizado en la fase tres de la creación de la interacción sino para la creación de la línea argumental y artística sobre la cual corre el videojuego.

METODOLOGíA XP (PROGRAMACION EXTREMA)

Para solventar las insuficiencias de la metodología SCUMM, se decide recurrir también a la implantación de una metodología que actualmente se ha hecho muy notoria y eficiente en el desarrollo de todo tipo de aplicaciones informáticas como es el caso de la metodología de Programación Extrema.

La cual es un enfoque de la ingeniería de software formulado por Kent Beck, es un tipo metodología “ligera” para el desarrollo de Software. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad, para esto se basa en:

Descripción:

- **Simplicidad**:

La simplicidad es la base de la programación extrema. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, y métodos. Los nombres largos no decrementan la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que cuanto más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

- **Comunicación**:

La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método. Las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de cómo utilizar su funcionalidad. Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide qué características tienen prioridad y siempre debe estar disponible para solucionar dudas.

- **Retroalimentación (*feedback*)**:

  Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. Considérense los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código.

  Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

- **Refactorización de Código**

  La refactorización de código se usa para describir la modificación del código fuente sin cambiar su comportamiento, lo que se conoce informalmente por limpiar el código. La refactorización se realiza a menudo como parte del proceso de desarrollo del software: los desarrolladores alternan la inserción de nuevas funcionalidades y casos de prueba con la refactorización del código para mejorar su consistencia interna y su claridad. El objetivo, por el contrario, es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto, para facilitar el mantenimiento en el futuro.

Este modelo se plantea como una metodología a emplear en proyectos de riesgo. Se plantea que es la ineficacia de adaptarse a los cambios los que destruye o vuelve ineficaz las anteriores metodologías de diseño.

Cada una de las 4 fases tiene su diverso árbol de trabajo y sus beneficios en cuanto a desarrollo:

- La fase de planificación parte de una recolección de información del usuario para la creación de la aplicación en forma de historias de usuarios, de esta forma el usuario es el que muestra sus problemas y explica como desea que se lo resuelvan. El segundo paso es la creación de un plan de entrega la cual es una especie de creación de un calendario para cada uno de los procesos necesarios para la creación de la aplicación, este calendario se crea a partir de hacer pasar cada actividad necesario por un algoritmo.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

El Spike o Programador es una estructura en la cual se considera las capacidades técnicas del programador y en él se evalúan la competencia del desarrollador de realizar una determinada labor. Luego de establecer un plan de Entregas se fija una velocidad para el desarrollo de esta actividad, en algunos casos se utilizan herramientas de la investigación de operaciones para fijar maximizar la velocidad en el desarrollo de una aplicación.

Luego de esto se realizan las iteraciones en las cuales se establece que actividades se realizaran primero y por quien (en el caso de que el equipo de desarrollo sea más de una persona). Y luego se da paso a la etapa de rotaciones, estas rotaciones evitan que las personas se conviertan en sí mismas en un cuello de botella. Las rotaciones permiten que todo el mundo conozca cómo funciona el sistema. En el caso de que la investigación sea realizada por un solo desarrollador la rotación se da entre las distintas actividades, es decir en vez de cambiar que un desarrollador pase por cada una de las actividades, las actividades van intercambiándose en el tiempo disponible de desarrollo del investigador. Al final de cada día de planificación se deben hacer reuniones para establecer el balance de progreso del día, y en el caso de que sean necesarias algunas correcciones en la planificación estas se hacen de manera inmediata.

- En el Momento de diseñar el sistema se desea que este sea lo más sencillo posible, se busca encontrar las soluciones más rápidas y eficientes posibles, y fomentar la reutilización del código en su máxima expresión, la metodología XP está fundamentada en el principio del reciclaje del código.

- Durante la Fase de Desarrollo se comprueba paso a paso la aceptación del usuario a las soluciones planteadas, es decir el lanzamiento continuo de versiones de pruebas con el fin de percibir, aquellas características que podrían convertirse en fallos o aquellos fallos que podrían hacer fallar la investigación; se deben mantener ciertos estándares de implementación definidos al comenzar la investigación; el código debe integrarse y estar disponible para todos los miembros del equipo de desarrollo.

- La Fase de pruebas es la parte final de la metodología XP consiste en a través de un equipo de test probar el sistema para conseguir los posibles fallos que se hayan producido a través de alguna de las fases de implementación y finalmente realizar pruebas de aceptación en el público para determinar con qué grado de aceptación cuenta la aplicación entre los usuarios.

DESCRIPCIÓN DEL PROCEDIMIENTO PARA LA CREACIÓN DEL VIDEOJUEGO

Cada vez es más común que un videojuego contenga elementos de diversos géneros, este videojuego contiene elementos del genero arcade y educativo. El videojuego del genero arcade, se caracteriza por la simplicidad de acción rápida de jugabilidad, no requiere historia, solo juegos largos o repetitivos; el videojuego del genero educativo es aquel que enseñan mientras promueve diversión o entretenimiento (Didactismo)

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

DISEÑO DE GUION (Indicadores y Objetivos):

Androids FSM V 1.0
(Store line)

Es un videojuego lineal, esto significa que se sigue un camino predeterminado y se realizan algunos objetivos específicos para completar el juego. Se enmarca en un esquema de juego que se constituye en varias partes o secuencias, hasta que llega al final. Cada porción del juego, tiene un aspecto diferente, tema u objetivo, las cuales se combinan para formar el mundo en el que se desenvuelve el juego.

Este camino predeterminado será recorrido por una bionave, que será maniobrada a través de las teclas direccionales.

La bionave deberá interactuar con entidades tridimensionales a razón de generar las salidas correspondientes con el autómata que se presenta en cada nivel de juego.

El videojuego está constituido por 4 niveles:

Por cada nivel a su vez se desarrollaron 3 subniveles, en el instante en el cual se inicia la partida, el sistema aleatoriamente seleccionara uno de los 3 entornos desarrollados por nivel, con el cual entonces el jugador deberá interactuar

Nivel 1:

Autómata 1:

En este autómata Se requiere generar salidas correspondiente con el autómata finito determinista (AFD).

Es un AFD que acepta el lenguaje , durante la interacción el jugador deberá a través de encuentros entre la bionave y las entidades del nivel, generar las 5 primeras salidas que acepta el autómata. Cualquier error durante la generación de la misma será detectado y el sistema emitirá mensajes que indicaran, que se deben volver a generar las salidas que se aceptan desde el principio, puesto que un error en la generación de la cadena indica que esta ya no se acepta.

Autómata 2:

En este autómata Se requiere generar la salida correspondiente con el autómata finito no determinista (AFND). que se presenta en la imagen:

Es un AFND que acepta el lenguaje a* |(ba*ba*)* , durante la interacción el jugador deberá generar las 5 salidas que acepte el autómata. Cualquier error durante la generación de la misma será detectado y el sistema emitirá mensajes que indicaran, que se deben volver a generar las salidas que se aceptan desde el principio, puesto que un error en la generación de la cadena indica que esta ya no se acepta

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Autómata 3:

En este autómata Se requiere generar la salida correspondiente con el autómata finito no determinista (AFND).

Es un AFND que acepta el lenguaje, durante la interacción el jugador deberá generar las primeras 5 salidas que acepte el autómata.

Como se mencionó anteriormente, el sistema solo cargar uno de los 3 autómatas por nivel, cada vez que se inicie la partida. El avance en el nivel será indicado por una barra de progreso, que avanzara desde 0% a 100%. En el preciso instante en el que la barra de progreso indique que se ha logrado el avance del nivel en su totalidad, se abrirá la compuerta que permiten el saltó al próximo nivel.

Nivel 2:

Autómata 1:

En este autómata se modela el funcionamiento de una máquina automática expendedora de bebidas enlatadas. Dicha maquina acepta monedas de valor 1$, 2$ y 5$, y el precio de cada lata es de 5$. Vamos a considerar que el evento llamado "1" es la introducción de una moneda de valor 1 en la máquina, el evento

"2" para la moneda de valor 2, etc.

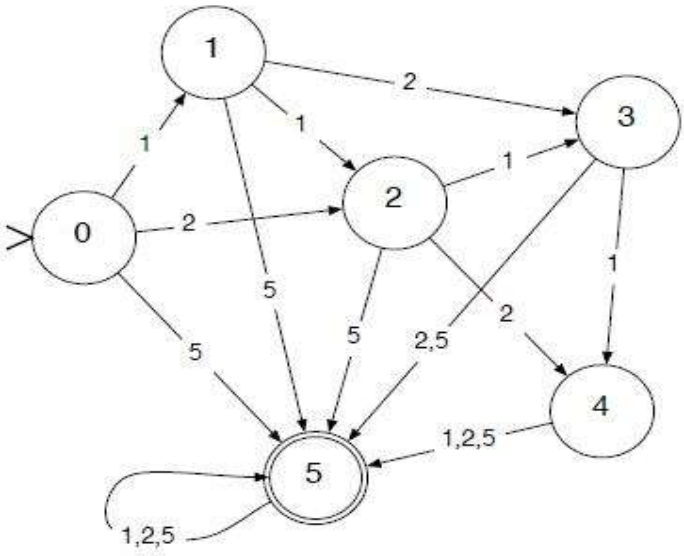

Figura 2. Autómata 1 - Nivel 2

En este autómata la Bionave, simulara el funcionamiento de un colector de monedas, al colisionar con el estado cero; este ofrecerá la posibilidad de registrar una moneda de 1$, de 2$ ó de 5$ en la máquina.

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Dependiendo del valor de la moneda registrada, ahora la Bionave deberá hacer colisión con el estado que tenga el mismo valor de la moneda registrada (estado 1, si valor de la moneda registrada fue igual a 1$, estado 2 si valor de la moneda registrada fue igual a 2$ ó a estado 5 si valor de la moneda registrada fue igual a 5$), para seguir registrando monedas hasta acumular al menos los 5$, (si es que no se registró una moneda de 5$ en la primera oportunidad)

Nota: El estado inicial, desde luego, "recordaría" que se lleva acumulado 0$, este autómata se basa en la buena idea de que cada estado "recordara" lo que se lleva acumulado hasta el momento.

El estado de aceptación es el estado 5, y cuando reciba un acumulado de 5$, devolverá el avatar de una lata de refresco (indicando que se aceptado).

En el caso de que el acumulado por las monedas registradas sea mayor a 5$ (ejemplo; 3 monedas de 2$), antes de aceptarse el sistema mostrara un mensaje en el que se visualizaría el valor de "el vuelto o cambio", indicando entonces que hay 5$ acumulados que se deben aceptar o recibir en el estado 5.

Autómata 2:

En este autómata Se requiere generar la salida correspondiente con el autómata finito no determinista (AFND).

Este autómata finito está definido sobre el alfabeto $\Sigma=\{0,1\}$, posee dos estados s_1 y s_2, y sus transiciones son $\delta(s_1,0)=s_2$, $\delta(s_1,1)=s_1$, $\delta(s_2,0)=s_1$ y $\delta(s_2,1)=s_2$. Su estado inicial es s_1, que es también su único estado final. El lenguaje regular que reconoce puede expresarse mediante la expresión regular (00 | 11 | (01 | 10)(01 | 10)) * .

La interacción el jugador deberá generar las 5 salidas que acepte el autómata. Cualquier error durante la generación de la misma será detectado y el sistema emitirá mensajes que indicaran, que se deben volver a generar las salidas que se aceptan desde el principio, puesto que un error en la generación de la cadena indica que esta ya no se acepta

Autómata 3:

En este autómata se requiere generar las salidas correspondientes con el autómata finito no determinista (AFND).

Es un AFND que acepta el lenguaje, durante la interacción el jugador deberá generar las primeras 5 salidas que acepte el autómata.

Nivel 3:

Autómata 1:

En este autómata 4 se requiere generar las salidas correspondientes con el autómata finito determinista (AFND)

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

En este nivel el jugador deberá maniobrar con la bionave para generar generar 4 salidas por recorrido, es decir 4 salidas por el recorrido y 4 salidas por el recorrido.

Habrá 2 indicadores o barras de progreso (una para el primer recorrido y otra para el segundo). De igual manera se podrán visualizar las cadenas que se aceptan por recorrido. Cuando ambos indicadores marquen un avance de 100% entonces, se podrá acceder al nivel 4.

Autómata 2:

En este autómata Se requiere generar la salida correspondiente con el autómata finito no determinista (AFND)

Autómata 3:

En este autómata Se requiere generar la salida correspondiente con el autómata finito no determinista (AFND) que se presenta en la imagen:

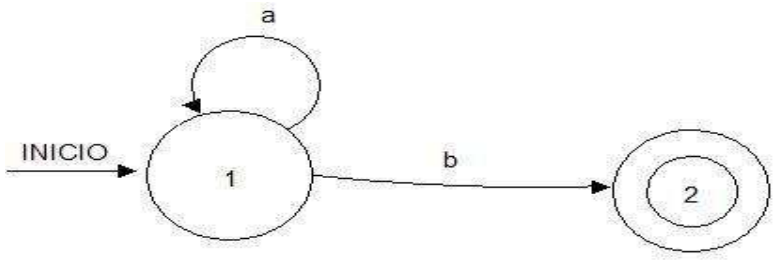

Figura 3. Autómata 3 - Nivel 3

Es un AFND que acepta el lenguaje, durante la interacción el jugador deberá generar las primeras 5 salidas que acepte el autómata.

Nivel 4:

Autómata 1:

En este autómata se requiere generar las salidas correspondientes con el autómata finito no determinista (AFND), que acepta el lenguaje que se presenta en la tabla:

Acee
Adee
Bcee
Bdee

B

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

El sistema generara un mensaje en el cual pedirá al usuario que ingrese por teclado una de las 5 salidas que genera el autómata, si la cadena leida por teclado no concuerda con alguna de las aceptadas por el mismo entonces, se emitirá un mensaje de error; indicando que vuelva a proceder con el ingreso de una las cadenas. De manera contraria si la cadena ingresada por teclado, es compatible con alguna de las aceptadas; ahora el jugador deberá construir;(interactuando con la bionave y las entidades de nivel) la cadena que ingreso por teclado.

Al construir la cadena (en base a interacciones entre la bionave y las entidades del nivel; que simulan tridimensionalmente al autómata), se llega a un punto en el cual el estado de aceptación determina si la cadena concuerda con la inicialmente ingresada por teclado y si por tanto es aceptada o no por el mismo.

Al igual que en nivel anterior cuando la barra de progreso indique que se ha logrado el avance del nivel en su totalidad (generar las 5 salidas) se abrirá la compuerta que permite el saltó al próximo nivel.

Autómata 2:

En este autómata se requieren generar salidas correspondientes con el autómata que se presenta en la imagen.

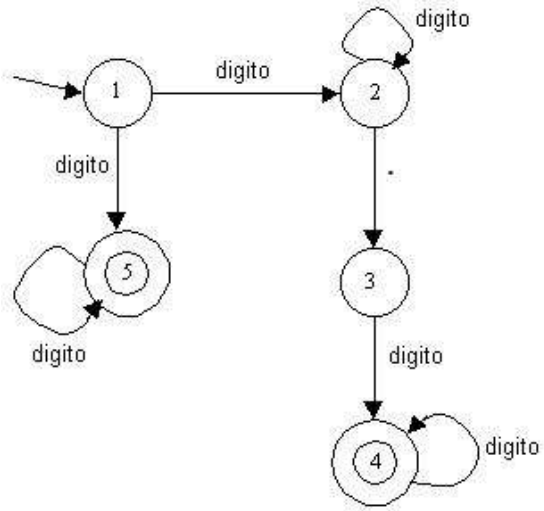

Figura 4. Autómata 2 - Nivel 4

Es un AFND que se modelo tridimensionalmente y que acepta generar cadenas de números enteros ó decimales., en este nivel el jugador deberá obtener cuatro salidas. Es de hacer notar que este autómata acepta cualquier salida que este enmarcada dentro de los parámetros de aceptación del mismo, por ejemplo, la salida

"***digito,digito***" es aceptada en el estado 4, pero no cuenta como salida que incrementa el avance de proceso en el nivel, de igual manera ocurre con la salida "**digitodigito**" que se acepta en el estado 5.

Las únicas cuatro salidas que marcarían avance en el nivel 4, serían las

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

siguientes:

digitodigitodigitodigito

digito,digitodigitodigito

digitodigito,digitodigito

digitodigitodigitodigito.digito

(3 números decimales y numero un entero)

Autómata 3:

En este autómata Se requiere generar la salida correspondiente con el autómata finito no determinista (AFND) que se presenta en la imagen:

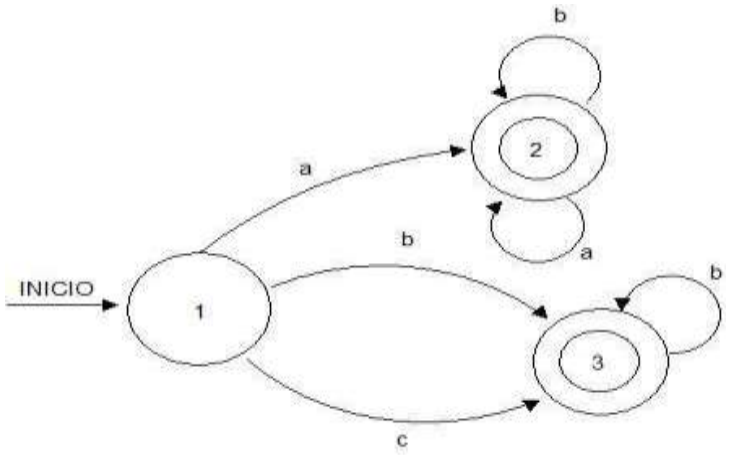

Figura 5. Autómata 3 - Nivel 4

Es un AFND que acepta el lenguaje | | | , durante la interacción el jugador deberá generar las primeras 5 salidas que acepte el autómata.

En el momento en el que el autómata seleccionado aleatoriamente por el sistema acepta una de las 4 salidas incrementa en el avance de proceso. Al cargar en un 100% la barra de progreso se habrá considerado como un éxito el recorrido por los 4 niveles del videojuego y se dará por terminado el avance por esta aventura gráfica. Luego de establecer el guion de desarrollo de la historia, este es el producto para el desarrollo de los mapas de interacción y bosquejos del video juego o como se le conoce en la industria cinematográfica “storyboard”

CREACIÓN DE PERSONAJES Y ENTIDADES 3D.

“Story Board”

Teniendo ya las instrucciones técnicas que describen cada escena de juego se procede a mapear los distintos mundos o niveles del juego, durante

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

la fase de preproducción. Un StoryBoard Consiste en una serie de pequeños dibujos ordenados en secuencia de las acciones que se van a filmar o grabar, de manera que la acción de cada escena se presenta en términos visuales.

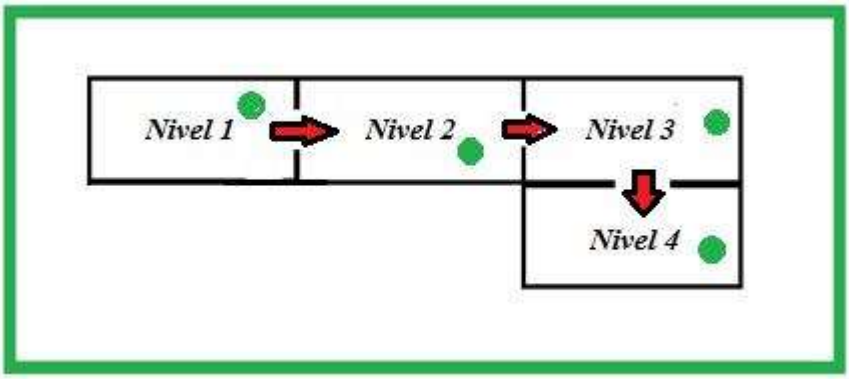

Figura 6. Mapa Escenarios. **Androids FSM**

Creación de escenarios:

Configuración de Modo Grafico:

Con la función "**Graphics3D**" se e*stablece el modo de vídeo.*

Con la función "**SetBuffer BackBuffer()**" se dibuja en el búfer invisible

Creación del Plano:

Con la función "***createplane"*** del compilador Blitz, se creó un plano útil para juegos al aire libre en los que el jugador nunca alcanza el borde del mundo del juego.

Creación de niveles:

Con la función "***createcube***" del compilador Blitz, se crearon entidades tridimensionales con forma de cubo a las cuales se les asigna valores en plano tridimensional (x,y,z), para su posterior ubicación con el comando,

"**positionentity**". Luego estas entidades pueden ser escaladas y rotadas en el plano 3D con los comandos "**scaleentity**" y "***rotateentity***". En base a la aplicación de estas funciones se establecieron las divisiones de todos los niveles del videojuego.

Creación de Personajes

Personaje principal (Bionave):

La bionave es el personaje principal del videojuego (es una malla realizada

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

con 3d Studio Max) con el que el usuario podrá interactuar a lo largo de la partida. El modo de moverse con personaje por el ámbito que se ha creado será el de segunda persona donde vemos el personaje moverse y nosotros le vamos siguiendo como si fuéramos una cámara.

Para lograr esto se creó la clase: jugador que contiene 3 objetos.

- Personaje: obtendrá el resultado de la función "**loadmesh**" o
  Cámara: obtendrá el resultado de la función "**créatecamara**" o
  Pivote: obtendrá el resultad de la función "**createpivote**"

Luego, se condicionaron el uso de las teclas del teclado que permitirían el movimiento del protagonista y con el comando

"**pointentity**" se dirige el ente cámara hacia el personaje.

Variable A Configurar	Valor Establecido
Desplazamiento de Bionave hacia adelante por impulso del teclado	5
Desplazamiento de Bionave hacia atrás por impulso del teclado	-5
Giro lateral de Bionave hacia la derecha por impulso del teclado	1.5°
Giro lateral de Bionave hacia la izquierda por impulso del teclado	-1.5°

Figura 7. Valores de la Configuración del espacio de la Nave.

Ya establecido un mundo virtual para poder realizar el videojuego se pasó al desarrollo de cada una de las entidades que se necesitarían en el videojuego. La zona por la que virtualmente nos movemos está muy claramente definida. Si pudiéramos elevarnos por encima de todo el escenario y verlo desde una perspectiva aérea, podríamos ver como la zona de juego está completamente definida y contendida.

Objetos

Creación de objetos en 3d Studio Max:

Se desarrollaron modelos 3D usando un software llamado *3D Studio Max*, los modelos luego fueron insertados en código fuente con la función

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

"**loadmesh**" que se usa para *Cargar una malla desde un archivo .x o .3ds.* y posicionados con la función "**positionentity**" del Blitz basic.

Luego de haber posicionado en el mundo virtual los objetos se procede al

"*mapeado de colisiones*", en las cuales se definen los objetos atravesables y no atravesables por la bionave.

Para realizar el mapa de colisiones se procede a asignarle a un objeto el nombre de una variable previamente declarada con el comando **EntityType objeto, nombre_variable** , luego en la parte del archivo donde estemos creando el mapeado o lista de colisiones se declara la colisión con la función **collissions** como por ejemplo**: collisions nave, nombre_variable,2,2.**

El mapeado de colisiones junto con la función **entitycollided**() detectan las colisiones entre diferentes entidades, lo cual proporciona el producto para la siguiente fase de la metodologia scumm. Establecimiento de interacciones (Banderas de Código)

ESTABLECIMIENTO DE INTERACCIONES (BANDERAS DE CÓDIGO)

El proceso de establecer las interacciones, se hace a través de banderas de funcionamiento donde es necesario que una este activa para activar a otra derivada. En el anexo se muestra el diccionario de banderas y contadores y su significado.

Al momento de diseñar el sistema se desea que este sea lo más sencillo posible, se busca encontrar las soluciones más rápidas y eficientes posibles, y fomentar la reutilización del código en su máxima expresión, esto basado en la metodología XP, la cual está fundamentada en el principio del reciclaje del código.

La metodología extrema también plantea como uno de, sus valores la simplicidad pues se apuesta a que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

MONTAJE DE ESCENARIOS.

En esta fase se desarrolló en plenitud el código fuente en el lenguaje de programación Blitz Basic. Se utilizó la programación modular que es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable. Se implementa el componente denominado "Inteligencia Artificial", el cual es la lógica del juego. Establece el aspecto físico, detecta la interacción del personaje, colisiones de objetos y controla los movimientos de los caracteres. Todos los bits y las partes - objetos, texturas y códigos, son integradas dentro de una utilidad especial llamada cadena de herramientas que combina todas las partes en un gran trozo de código. Luego de esto se realizó la Interfaz de presentación, esto se conoce comúnmente en el mundo de los videojuegos como "HUD". ("Head-Up Display"). El HUD se desarrolla con la finalidad de no entorpecer el juego y brindar al jugador en todo momento la información de juego necesaria. La mejor zona para colocar el HUD es en el borde

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

de la pantalla, lo que permite al jugador acceder a la información necesaria con solo mover un ojo.

Durante esta fase del desarrollo del videojuego se implementan algunas características de la **programación extrema XP** tales como:

- **Desarrollo iterativo e Incremental**: pequeñas mejoras unas tras otras.
- **Pruebas de Regresión:** para descubrir las causas de errores.
- **Pruebas Unitarias:** De esta manera se probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado
- **Corrección de todos los errores:** antes de añadir nuevas funcionalidades.
- **Integración del equipo de programación con el cliente:** Entregas Frecuentes.

CONCLUSIONES

La investigación titulada, **SOFTWARE EDUCATIVO (VIDEOJUEGO SIMULADOR DE AUTOMATAS FINITOS,** luego de transitar por el proceso de recolección de información, diseño y producción de la aplicación denominada

“Androids-FSM”, arrojó las siguientes conclusiones:

Al desarrollar el sistema, se logró una aplicación de tipo educativo, capaz de contribuir al desarrollo y orientación de los estudiantes en el tema de los autómatas finitos deterministas, pues este permite experimentar, investigar, y corroborar el funcionamiento de estas denominadas máquinas, desarrollando la percepción espacial, la lógica, la imaginación y la creatividad; por tanto se cumplió el objetivo general de la investigación: **Desarrollar un software educativo (videojuego) fomentando el análisis y diseño de autómatas impartidos en la asignatura de lenguaje y compiladores de la carrera de Ingeniería de computación de la Universidad Valle del Momboy,** al obtener como resultado un software capaz de ser utilizado por los estudiantes de esta carrera universitaria para conocer de manera interactiva el funcionamiento de estas máquinas de estado finito.

Esta interactividad sirve como plataforma para atrapar la atención de los usuarios en la enseñanza del diseño de autómatas finitos deterministas; como se explicó en los capítulos 1 y 2 los medios interactivos y audiovisuales, obtienen mayores niveles de concentración por parte de los usuarios que los libros de textos tradicionales; al utilizarse la poderosa herramienta de los videojuegos, se capta la atención de los estudiantes, reforzando su aprendizaje de manera que se sientan atraídos.

En cuanto al primer objetivo específico: **Diseñar un programa didáctico en**

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

el cual la interfaz gráfica sea el entorno a través del cual los usuarios establezcan comunicación con los contenidos relacionados con los autómatas finitos, se tomaron en consideración elementos tales como con la interactividad que proporciona un entorno audiovisual para realizar una interfaz gráfica con la cual los usuarios puedan lidiar con autómatas finitos, títulos, textos de apoyo e imágenes que sirven para reforzar palabras e ideas. Se elaboró un guion o "history line" el cual es la base a partir de la cual se arman o producen videojuegos, de esta manera se crea una forma para poder comprender la experiencia.

En el segundo objetivo específico, **Construir un entorno tridimensional que permita a los usuarios recorrer un autómata finito proponiendo el aprendizaje por logros y desaciertos,** se tomó en consideración el notable auge que tienen los videojuegos desarrollados en entornos tridimensionales que pueden ser recorridos en segunda persona pues la manipulación de un personaje directamente por el usuario produce un sentimiento de inmersión directa en un mundo de objetos. El entorno 3D se desarrolló exportando mallas 3D diseñadas en 3Dmax Studio, luego adaptadas al plano donde se lleva a cabo la aventura por medio de herramientas de rotación, escalacion y movimientos de objetos a través de los ejes coordenados. Para el aprendizaje por logros y desaciertos se crearon rutinas de programación tomando en cuenta los principios básicos de la inteligencia artificial pues se generan procesos de software que producen resultados basándose en la secuencia de entradas percibidas y en el conocimiento almacenado en tal arquitectura.

Tomando en cuenta el tercer objetivo, **Implementar en el diseño del software mecanismos que permitan a los estudiantes reforzar conocimientos sobre autómatas finitos adquiridos con anterioridad, llevando el control de errores y retroalimentación positiva**, se modelaron 12 autómatas finitos, divididos en 4 niveles que buscan incentivar en el usuario el diseño de autómatas finitos para reforzar los conocimientos impartidos en el aula por el docente, se implementaron rutinas de programación que permiten al usuario diferenciar secuencias de eventos "buenas", de secuencias de eventos "malas".

La programación del videojuego cumple con las necesidades de presentar al usuario cierta "naturalidad" en el ambiente donde se desarrolla la historia, creando un "ambiente realista" afectado por las leyes de la física como la gravedad, la inercia al chocar contra objetos y el rango de visión que se aclara al acercarse a los objetos. La inclusión de efectos sonoros, y banda sonora ayudan a colocar al aumente la efectividad del videojuego.

En la aplicación se fusionan las tecnologías del diseño, las animaciones por computadoras y el sonido envolvente en correspondencia con las tradiciones orales y escritas que subsistieron con el transcurrir de los años. Las tecnologías desarrolladas por la ingeniería contribuyen de manera dinámica e innovadora al proceso educativo.

BIBLIOGRAFIA

- **ÁLVAREZ, Pedro** (2004) "Lenguajes, Gramáticas y Autómatas – Conceptos" Universidad de Zaragoza., Departamento de Informática en Ingeniería de

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

Sistemas.

- **AZPITARTE, Antonio** 3D Studio Animación y Proyectos de Diseño, editorial Paraninfo 1995.
- **BOUSQUET & MCCARTHY** (2006). 3ds Max Animation. ISBN: 0321375726 ISBN-13: 9780321375728.
- **BRENA, Ramón.** (2003) "Autómatas y Lenguajes. Un enfoque de diseño"
- Tecnológico de Monterrey- México.
- **CHACON, José Luis.** (2005) "Matemáticas Discretas, Pensar y Actuar" pgs 1-12
- **CSÍKSZENTMIHÁLYI, M**. (1990).Flow: The Psychology of Optimal Experience. New York: Harper and Row
- **DE AGUILERA MOYANO, Miguel**. "Videojuegos y Motivación" contenido publicado en http://ares.cnice.mec.es/informes/02/documentos/indice.htm
- **DE KERCKHOVE, D**. (1997) "Inteligencias en Conexión. Hacia una sociedad de la Web " Barcelona: Gedisa, 1999.
- **ENCARTA 2009**. **Microsoft ® Encarta ® 2009.** © 1993-2008 Microsoft Corporation. Reservados todos los derechos.
- **FELICIA, Patrick,** (2009). European Schoolnet "Videojuegos- Manual para Docentes".
- **GARCIA FERNADEZ, Fernando** (2005). "Videojuegos: un análisis desde el punto de vista educativo."
- **GOTTFRIES, Byron S**. (2001). *Teoría y problemas de Programación BASIC*. traducción, Guillermo Caro Murillo, Jesús Villamizar Herrera (2a. ed. edición). Buenos Aires; Bogotá: McGraw-Hill. pp. 278 p
- **http://www.programacionextrema.org/** (web dedicada al análisis de la metodología de programación extrema en español)
- **http://www.scummvm.org** (web dedicada al modelo Metodológico SCUMM)
- **http://www.wikipedia.org** (La Enciclopedia Libre)
- **JUUL, Jesper.** Half-Real**:** Videogames between Real Rules and Fictional Worlds (MIT Press, 2006)
- **MAINER, Belen. "**Ciberjuego y sociabilidad: Relaciones y efectos en los usuarios de juegos", contenido publicado en: http://www.ucm.es/info/especulo/numero31/ciberwow.html

Depósito Legal: PPX200602TR2436 / ISSN:1856-6936. Volumen 8 Edición No 2 – Año 2014

- **MARZAL, A y GARCÍA, I.** Introducción a la programación con Python, Departamento de Lenguajes y Sistemas Informáticos Universidad Jaume I, 2003 137

- **Norman, D. A.** (1993). "Cosas que nos hacen inteligentes: La defensa de los atributos humanos en la Era de la Máquina"(Nueva York:, Addison-Wesley.)

- **NAVARRETE, Isabel & CARDENAS, María** (2008) "Teoria de autómatas y lenguajes Formales" Universidad de Murcia – España.

- **ORTEGA, José** (2001) "Análisis crítico de los valores que transmiten los videojuegos" contenido publicado en: www.ugr.es/~sevimeco/documentos/.../a_valores.doc

- **PNAT, (PLAN NACIONAL DE ALFABETIZACION TECNOLOGICA 2006),** publicaciones Monfort.(Caracas)

- **RAFFO, Eduardo** (2005). "Optimización por computación evolucionaria",

- Universidad Nacional Mayor de San Marcos- Perú

- **ROCHA, Jairo** (2002) "Teoría de la computación y Verificación de Autómatas"

- Universitat de les Illes Balears.

- **VAZQUEZ, Romero.** (2002) Tesina para la obtención del título de Licenciatura en Sociologia. Universidad Autónoma Metropolitana de Iztapalapa.