Desarrollo de un video juego en 3D aplicado al análisis, diseño e implementación de una red LAN en el nivel físico[1]

Resumen

El presente artículo es resultado del proyecto de grado denominado: Desarrollo de un video juego en 3D aplicado al análisis, diseño e implementación de una red LAN en el nivel físico. El cual, expone y describe la metodología para el desarrollo de videojuegos educativos en 3D, las funciones para el control de eventos e interacciones entre el juego y el usuario, aplicando herramientas de distribución libre como lo son Blender (Diseño 3D) y lenguaje de programación Python, fortaleciendo el desarrollo de herramientas en el área práctica del análisis, diseño e implementación de redes.

Palabras clave: videojuegos, LAN, Python, Blender, redes, diseño tridimensional, ambiente 3D, diseño de red, análisis de red, implementación de una red.

Autores

Andrés Fabián Castro Castro[2]

Laura Estefanía Gómez Muñoz[3]

Director

Darin Jairo Mosquera Palacios[4]

Grupo de investigación ORION

Introducción

En el desarrollo tecnológico del mundo es muy común la creación de diversos programas que facilitan la vida a las personas –aplicativos que van desde una calculadora hasta los diseños más complejos de la arquitectura. Estos programas buscan hacer las cosas más sencillas al realizar las múltiples tareas de una forma más organizada, sincronizada, controlada y con el máximo de eficacia.

NetDesigner es una aplicación interactiva desarrollada en un ambiente tridimensional, con el objetivo de fortalecer y demostrar el uso de herramientas en el área práctica del análisis, diseño e implementación de redes. A través de este videojuego pretendemos incentivar a los estudiantes a realizar prácticas

1 Proyecto curricular de Tecnología en Sistematización de Datos.

2 Tecnólogo en sistematización de datos, correo electrónico: vulcano885@gmail.com

3 Tecnólogo en sistematización de datos, correo electrónico: estefi013@yahoo.com

4 Docente Facultad Tecnológica Universidad Distrital Francisco José de Caldas.

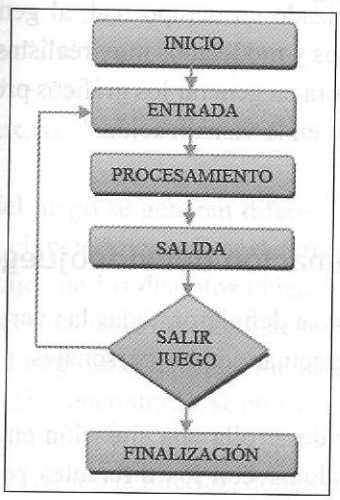
utilizando una de las atracciones más difundidas últimamente por los jóvenes en el área computacional: los videojuegos. Logrando así, una difusión de los juegos en espacios como la educación y la capacitación; de esta manera, será una herramienta práctica y de fácil uso para cualquier persona que desee utilizarla. Este juego está diseñado bajo productos de distribución libre, utilizando una metodología hibrida que involucra: RUP, diseño de videojuegos y diseño de redes.

1. Iniciando el desarrollo del videojuego educativo

El desarrollo de un videojuego no implica otra cosa que el desarrollo de una aplicación software enfocada en el área visual. Un videojuego debe funcionar en tiempo real; en todo momento mientras se está ejecutando, el juego debe estar realizando alguna tarea como: [1] dibujar los objetos, actualizar coordenadas, calcular colisiones, independiente de si el usuario se lo pide o no, además debe estar a la espera de eventos de teclado y de mouse, lo que genera un programa que este corriendo en un ciclo.

De acuerdo a lo anterior podemos definir un diagrama de la estructura de un videojuego dentro de su ciclo.

Figura 1. Estructura de un videojuego



Teniendo en cuenta este gráfico podemos definir los conceptos principales:

Inicialización: se inicializan todas las variables, lo que será usado en el ciclo del videojuego. Por ejemplo, aquí inicializaremos la librería gráfica, un modo gráfico, el sistema de sonido-música, de texto y cualquier otro tipo de sistema necesario. Además se reserva memoria para los objetos que intervienen en el juego: creación de estructuras de datos, carga de sonidos, imágenes y recursos en general. También, en este proceso se dan las posiciones iníciales de los personajes y carga de puntajes desde un archivo.

Ciclo del videojuego: es un *loop* que se estará repitiendo una y otra vez. Es aquí donde ocurre toda la acción del juego y la única forma para poder salir de este ciclo es cuando el jugador pierde, llega al final del juego o sale del videojuego con alguna combinación de teclas o presionando algún botón del mouse, etc. El ciclo del juego consta básicamente de tres partes:

•**Entrada:** se obtiene desde algún dispositivo de entrada (teclado, mouse, joystick, etc.) todo lo que realiza el jugador.

•**Procesamiento**: se procesa toda la información que se recibió en el punto anterior y se toman decisiones a partir de los datos de entrada, es decir la lógica del juego. Se procesa la física, inteligencia artificial, comunicación de datos en red y objetos, etc.

•**Salida:** se muestra el resultado de la información procesada en el punto anterior, se actualizan los gráficos en pantalla, reproducimos sonidos y acciones, etc.

Finalización: por último se elimina de la memoria todos los recursos almacenados, ya sea imágenes, sonidos, música, etc. Se cierran todos los sistemas que se abrieron en la inicialización. Se guardan datos de puntajes en un archivo y se muestra el resultado de finalización del juego.

2. Desarrollo del guión del videojuego [2]

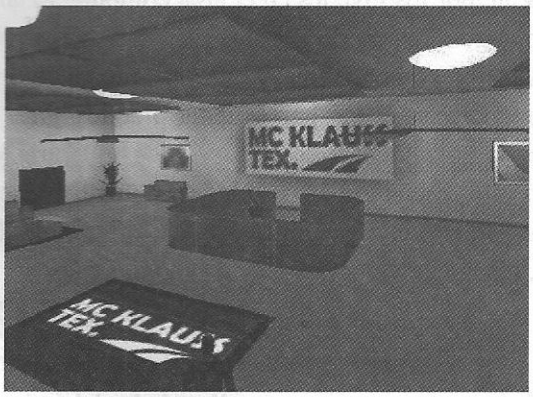
Teniendo en claro los conceptos de la estructura del videojuego pasamos al segundo paso que es la creación del guión del videojuego, el cual nos dará la pauta y estructura de la historia de nuestro juego, nos indica los personajes que estarán presentes, los escenarios, las reglas y objetivos a llevar a cabo. En el guión hay que definir puntos importantes, los cuales son: el tipo de juego a desarrollar –en nuestro caso es un juego serio (educativo, simulación, capacitación)–: la descripción y rol del personaje principal con el cual el usuario va a interactuar directamente. Para NetDesigner se definió que nuestro personaje se llamaría "Kevin".

Un aspecto importante es la definición de la historia que se da a Kevin al inicio del juego y lo que debe realizar para poder ganar el mismo. Teniendo en cuenta el guión y la estructura del videojuego podemos pasar a su desarrollo, y para esto se llevan a cabo tres etapas principales: el modelamiento de los escenarios, personajes y objetos; la definición de la programación y la unión entre la programación y los objetos modelados.

3. Modelado del videojuego [3]

Esta etapa es puramente gráfica, se diseñaron los escenarios de la empresa donde el personaje va a montar la red de computadores, se modelan los objetos y los personajes. Para el modelamiento se utilizó *Blender*, teniendo en cuenta que dependiendo de la calidad de los gráficos puede reducir el rendimiento del juego. En esta parte se utiliza el motor de renderizado de Blender y las librerías gráficas, las cuales nos ayudan a la hora de unir la programación con el modelo físico.

Imagen 1. Diseño escenario de la empresa de computadores



La implementación gráfica realizada en NetDesigner tiene una calidad media que, a medida que se desarrolló el diseño de todos los objetos, se veía el aumento constante de recursos de memoria que se necesitaban. Por esta razón, se decidió optar una definición no muy realista; esta decisión se tomó debido a que Blender nos permite gran opción de realismo, pero al realizar modelados realistas se aumenta el gasto de memoria, tornando la ejecución y el procesamiento de gráficos del juego más lento.

Un punto importante a tener en cuenta al realizar un videojuego es que se debe tener un alto rendimiento, sin generar bloqueos y demoras en el tiempo de espera del usuario. Blender nos proporciona la opción de su motor de renderizado en tiempo real, al generar gráficos muy detallados y modelados muy realistas, el motor de Blender demora en generar los gráficos produciendo un mayor retraso en la visualización.

4. Programación del videojuego [4]

En esta etapa se definieron todas las variables necesarias para interactuar con los personajes.

NetDesigner desarrolla una situación en la que Kevin tiene que dialogar con los diferentes personajes para buscar los datos e información necesaria que le permita montar la red en la empresa. Además se define la es-

tructura del juego en cuanto a desarrollo, interactividad y objetivos, los cuales se programaron en *Python* [4].

Dentro del juego se definieron tres niveles o módulos:

1. **Nivel 1:** se definió la etapa de recolección de información y documentos (análisis de la red); se definieron los documentos necesarios para que el usuario tenga todas las herramientas para montar la red, por ejemplo, [5] definición de conceptos, dispositivos, técnicas y normas para el montaje de la red.

2. **Nivel 2:** este nivel involucra la adquisición de los dispositivos, herramientas y recursos necesarios para la compra de los mismos. Además se realiza el diseño de la red para su montaje.

3. **Nivel 3:** en el último nivel se realiza el montaje físico de acuerdo al diseño elegido por el usuario y los dispositivos comprados por él.

Nivel 1

Se definieron las principales variables y la inicialización de las librerías. Se utilizan los métodos *getposition* y *getsensor* para generar las nuevas posiciones de movimiento del personaje y detectar cuando encuentra un documento. La inicialización [6] de las variables que contienen los documentos se da de la siguiente manera:

```
file=open("tr.txt","w")
file.write("i=0" + "\n")
file=open("docx.txt","w")
```

En el diseño del juego se generan diferentes sensores, los cuales, con el método *getsensor* determinan el choque del personaje con los distintos objetos del escenario. Esto nos permite saber cuándo ha encontrado o no un documento. Luego de determinar mediante el sensor que el objeto ha sido encontrado, se muestra un mensaje al usuario que le indica y le muestra el documento que ha encontrado, abriendo la interfaz, donde puede seleccionar los documentos habilitados para su lectura.

A través del *gamelogic* se implementan métodos para obtener los mensajes y datos en pantalla. De esta forma manejamos los objetos y damos instrucciones.

Nivel 2

Este nivel se ejecuta cuando el usuario ha recogido todos los documentos en el nivel anterior, luego del análisis de los documentos, se le presenta al usuario las opciones de redes posibles para el montaje. Inicializamos las variables que le dan dinero y continuación al juego, de acuerdo al presupuesto establecido para la red que se escogió y habilitó la opción de ir al almacén. Para la compra en el almacén se utilizan los scripts en *Python*, y se logra la validación en la compra mostrando los objetos comprados.

Los *scripts* son muy importantes porque a través de estos podemos controlar el juego y darle instrucciones al panel de propiedades de los objetos. Mediante la instrucción "own.propiedad" conectamos los *scripts* con las propiedades [2] [6].

Nivel 3

En este nivel, el usuario pone los dispositivos adquiridos en el almacén y luego se procede con la evaluación final, que involucra los siguientes factores: tiempo, optimización de la red, recursos invertidos, dispositivos correctos, entre otros.

La evaluación final al usuario se muestra en una categoría de cinco niveles, A, B, C, D, y E, donde A significa el puntaje más alto y E es el puntaje mínimo. Estos puntajes se obtienen de los promedios de la fase de análisis, diseño e implementación, además del tiempo gastado en el transcurso del juego.

5. Orquestador

El ejecutable llamado *orquestador* es una parte importante que contribuye a optimizar el videojuego, al reducir el gasto de memoria. El orquestador administra los diferentes escenarios ejecutando sólo uno a la vez

y cerrándolo cuando ya no se esté utilizando –se libera la memoria para lanzar el próximo escenario. Si no se utilizara, se consumiría cada vez más memoria, pues los escenarios se abrirían uno tras otro, sin cerrarse el que ya no se esté usando.

Cada escenario se comunica con el *orquestador* a través del *script,* enviando las variables que podemos definir como verdaderas y falsas, las toma y de acuerdo a la respuesta cierra y ejecuta el escenario.
De esta forma se obtiene el valor asociado a las claves de los niveles:

nivel1 = rdict['Index'] #
sera 'Done' cuando se haya jugado y terminado el nivel.

nivel2 = rdict['Recepcion'].

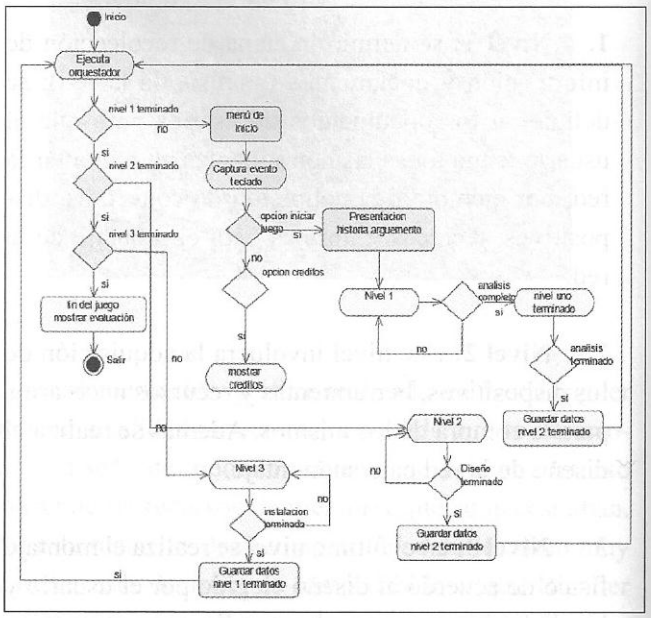
Con esta línea se conoce si el nivel ha sido concluido o no, y por medio de un ciclo que pregunta el estado de la ejecución, se van lanzando y cerrando los distintos escenarios:

```
if niv1.isPositive() and sens.isPositive():
    #verifica que el nivel1 haya sido completado satis-
    factoriamente
        ini=(i)
        rec1=(r)
        rec2=(r2)
        alm=(a)
        salir=(s)
        execute=(e)
        #se dan nuevos valores a los escenarios para
        ser ejecutados.
        if salir==0:
        exe=0
        if rec2 == 0 and salir==1:
        exe=3
        # se determina el escenario a ejecutar.
```

Si no se implementa el *orquestador*, el gasto de memoria sería superior a 2 Gb, pero gracias a su uso se reduce aproximadamente a 800 MB.

Básicamente la estructura del *orquestador* que dirige el juego es:

Figura 2. Estructura del orquestador



Todos los niveles tienen comunicación con el *orquestador*, el cual recibe las variables determinantes que envían los niveles en su ejecución.

6. Pruebas del sistema

Aspirador de secreciones casero

Pruebas de memoria
Se realizaron durante la ejecución del juego y su avance en el tiempo.

Presentación
Los niveles de consumo de memoria RAM en este nivel oscilan entre los 100 MB hasta un pico máximo de 250 MB aproximadamente.

Nivel 1
Los niveles de consumo de memoria RAM oscilan entre los 178 MB hasta un pico máximo de 1.23 GB

aproximadamente, estos niveles varían de acuerdo a la duración del recorrido que realice el usuario en cada uno de los escenarios.

Nivel 2 (Recepción)
Los niveles de consumo de memoria RAM oscilan entre los 150 MB hasta un pico máximo de 350 MB aproximadamente, estos niveles varían de acuerdo a la duración del recorrido que realice el usuario en cada uno de los escenarios.

Nivel 2 (Almacén)
Los niveles de consumo de memoria RAM oscilan entre los 157 MB hasta un pico máximo de 400 MB aproximadamente.

Nivel 3
Los niveles de consumo de memoria RAM oscilan entre los 150 MB hasta un pico máximo de 1.23 MB aproximadamente, estos niveles varían de acuerdo a la duración del recorrido que realice el usuario en cada uno de los escenarios.

Orquestador
Los niveles de consumo de memoria RAM oscilan entre los 20 MB hasta un pico máximo de 40 MB aproximadamente.

Pruebas de procesador
Los niveles de consumo de procesador durante la ejecución del juego en la línea de tiempo es de:

- 100% en procesadores tipo Pentium IV y su equivalente en AMD u otros fabricantes.
- 50% en procesadores tipo Core 2 Duo y su equivalente AMD u otros fabricantes.
- Es posible realizar diferentes tareas de modo paralelo a la ejecución del juego, pero el rendimiento del equipo puede verse afectado, generando una ejecución más lenta de los procesos.

Pruebas de disco duro
El nivel de consumo del disco duro del juego en la carpeta de archivos C: / es de 350 MB aproximadamente.

Pruebas de software
El usuario puede interactuar libremente con el juego, el personaje se desplaza como es deseado durante el transcurso de la ejecución.

Los sensores de colisión del motor son eficientes pero en ciertos momentos se presentan anomalías en su comportamiento, como la no detección de la colisión o reacciones diferentes a las programadas, esto se debe a las limitaciones del motor de Blender, las cuales no son modificables.

7. Conclusiones

La implementación de una herramienta alternativa a las existentes, permite a los estudiantes fijar de forma más didáctica algunos conocimientos relacionados al análisis, diseño e implementación de redes en nivel físico.

La metodología de desarrollo en el juego es resultado de la aplicación de una metodología enfocada al desarrollo de videojuegos y una metodología orientada al desarrollo del software.

El desarrollo de juegos educativos contribuye al mejoramiento de la educación, permitiendo un mayor enfoque en la práctica de diversas áreas educativas.

Podemos resaltar que, a través del juego se da una solución óptima a la necesidad planteada en el mismo, a través del marco teórico y la documentación incorporada en el juego.

Para la optimización del juego es necesario tener en cuenta los siguientes aspectos:

Es mejor trabajar cada escenario como un ejecutable independiente. Esto hace que el peso de cada ejecutable sea menor y por consiguiente se demore menos tiempo en cargar.

Si se va a desarrollar un videojuego complejo y se espera un máximo de rendimiento, es recomendable trabajar con un motor de rénder distinto como OGRE3D, *cristal space*, entre otros.

Referencias

[1] **González Morcillo, Carlos; Blender y Yafray** (2006). *Diseño gráfico 3D con software libre*. La Mancha: Escuela Superior de Informática, Universidad de Castilla.

[2] **Alvarado Valderrama, Jorge** (2006). *Herramientas utilizadas en la generación de gráficos en la cinematografía*. Trujillo, Universidad Nacional de Trujillo.

[3] **Manual del usuario Blender**, http://wiki.blender.org/index.php/Doc:Manual.

[4] **Montaño Ramírez, Antonio** (2005). *Python*, http://ownz.despai.es/Documentos/Programacion/Iniciacion%20a%20python.pdf

[5] **Huidobro Moya, José Manuel; Blanco Solsona, Antonio; Calero, Jordán** (2006). *Redes de área local. Informática: administración de sistemas*. Madrid: Cengage Learning Editores, p 249-251.

[6] **Van Rossum, Guido** (2005). *Guía de aprendizaje de Python*. Santa Clara, CA, Python Software Foundation