OBPLUS: Librería Para diseño de juegos tridimensionales

Cesar Mauricio Pachón Meneses
cesarpachon@yahoo.es, http://dis.unal.edu.co/~ohwaha
Grupo de Investigación de Computación Gráfica y Procesamiento de Imágenes OHWAHA.
Departamento de Ingeniería de Sistemas
Universidad Nacional de Colombia (Sede Bogotá)
2002

RESUMEN: Este articulo presenta una librería multiplataforma para el diseño de juegos de computador tridimensionales, la cual facilita la administración de escenarios complejos y provee clases utilitarias que implementan varias tareas comunes en un programa de este tipo.

ABSTRACT: This article presents a multiplatform library for tridimensional computer-game design, which facilitate the complex scenes management, and provide useful classes which implement many common tasks in this kind of programs.

Palabras clave: Juegos de Computador, computación gráfica, multimedia.

1. **INTRODUCCION**

En el ambiente del desarrollo de juegos de computador existen bastantes herramientas disponibles, tanto gratuitas como comerciales. Estas herramientas ocupan un amplio rango de niveles de complejidad, que van desde librerías gráficas de bajo nivel que implementan operaciones de rasterizado e interacción con el hardware, hasta productos completos que permiten modelar personajes y escenarios, y correr dentro de ellos el juego sin necesidad de escribir ni una sola línea de código. Ejemplos populares del primer tipo son OpenGL y DirectX IM (Inmediate Mode). Ejemplos del segundo tipo son Blender y hasta cierto grado los exportadores para VRML de 3D Max. En medio de estos extremos existen innumerables puntos intermedios. Por ejemplo, DirectX RM (modo retenido) es un librería de alto nivel que está montada sobre DirectX IM, y provee clases para cámaras, escenarios y formatos de archivos. Cristal Space es un motor para juegos GNU que sin embargo es bastante complejo y está muy orientado a juegos en ambientes cerrados (rooms).

La elección del tipo de herramienta a utilizar depende de muchos factores, entre ellos el tamaño del grupo de programadores, la experiencia que estos tengan en programación de juegos, el tiempo programado para la terminación del proyecto, la existencia de código y recursos reutilizables de proyectos anteriores y por supuesto, el presupuesto disponible. Otro factor es el tipo de plataforma al cual está dirigido el juego. Actualmente Windows sigue siendo el sistema

operativo más difundido, pero no hay que perder de vista plataformas como Linux. Lo ideal es utilizar herramientas que no limiten el proyecto a una plataforma específica.

Por estas razones, se decidió elaborar una librería genérica para el diseño de juegos de computador que cumpliera con las siguientes características:

- Estar escrita en lenguaje C++, ya que este es el lenguaje más utilizado en el desarrollo de juegos, y cuenta con compiladores para distintas plataformas.
- Utilizar como base la librería gráfica OpenGL, la cual provee la interacción de bajo nivel con el hardware de manera eficiente, además, a diferencia de DirectX, es multiplataforma y de uso libre.
- Ofrecer soporte para el manejo automático de escenarios, detección básica de colisiones, administración de texturas, animación de personajes, manejo de archivos de datos, cámaras, terrenos, y manejo de tiempo.
- Utilizar formatos de archivos propios, tanto para geometría tridimensional como para texturas, que ofrecieran mejoras con respecto a los formatos existentes en cuanto a velocidad de cargado, extensibilidad, y protección de la información.
- Ofrecer mecanismos para la implementación de Inteligencia Artificial, extendibilidad de agentes y mecanismos eficientes de paso de mensajes.

2. **DESARROLLO**

La librería desarrollada recibió el nombre de OBPlus, y en la actualidad está conformada por cerca de 50 clases, agrupadas en nueve categorías:

Geometría
Gestión de texturas
Escenarios
Manejo de terrenos
Animación
Cámaras
Manejo de tiempo
Inteligencia Artificial
Clases utilitarias

2.1.Geometría

El soporte principal de la librería OBPlus un el formato de archivo del mismo nombre (extensión .ob+), el cual tiene versiones en texto plano (ASCII) y en binario. La razón de esto es que el texto plano es útil para pruebas y depurado; además muchos programas de modelado en 3D (Blender, 3D Max) permiten escribir scripts para exportar los modelos elaborados en ellos, y generalmente sólo pueden escribir texto plano. Por otro lado, el formato binario dificulta que los modelos en las distribuciones finales de los programas sean utilizados

por otras personas, además la versión binaria ocupa menos espacio que la versión ASCII e implementa algunos mecanismos que aceleran el cargado del archivo. La información que se guarda en un archivo OB+ se agrupa en bloques cuya presencia es opcional, y pueden ser removidos (o añadidos) sin afectar el resto del archivo. Algunos de los bloques que soporta el formato OBPlus v2.0 son:

- Geometría básica (vértices y caras).
- Normales (por cara, por vértice y banderas adicionales de configuración para forzar el cálculo de normales en tiempo de carga y reducir el tamaño del archivo).
- Coordenadas UV (Mapeo de texturas).
- Materiales por cara.
- Texturas por cara.
- Jerarquías (para modelos complejos: personajes).
- Animación (curvas de animación basadas en keyframes).
- Acciones (agrupar intervalos de animación en acciones lógicas).

Para obtener más información sobre el formato OB+ consultar [1].

2.2.Gestión de texturas

para la gestión de texturas se diseño un formato propio, llamado OBI (OBPlus Image), el cual está diseñado para minimizar el tamaño del archivo al tener una cabecera reducida al mínimo y evitar bytes de relleno, esto mismo permite que pueda cargarse la textura en una sola línea de código. Adicionalmente soporta canal alpha para implementar transparencias. El formato OBI cumple con la especificación de OpenGL para el manejo de texturas, en cuanto a las restricciones de tamaño y dimensiones, y el tipo de datos utilizados. Además de los métodos para cargar OBI´s la librería ofrece clases como el obpTextureManager, el cual se encarga de llevar un registro de texturas cargadas, y permite la rápida ubicación, carga, descarga y registro de las mismas.

2.3.Escenarios

Por escenario se comprende el conjunto de objetos que no intervienen de manera directa en el juego, que no tienen animación, conducta y que permanecen estáticos en un lugar determinado. Al ofrecer clases que automáticamente gestionen los escenarios, el programador queda en libertad para concentrarse en la parte importante del juego: el comportamiento de los personajes y los efectos especiales. En la librería Obplus esto se logra por medio de las clases obpWorld y obpWorldDynamic, ambas heredan de obpWorldBase. ObpWorldDynamic está especialmente diseñada para gestionar escenarios muy grandes y complejos, permitiendo la carga y descarga dinámica de objetos no visibles de manera automática. En la librería Obplus se establece la diferencia entre objeto y malla. Una instancia de la clase obpMesh es

quien contiene la información de geometría de un archivo OB+ (por ejemplo, un árbol). Varias instancias de la clase obpObject pueden hacer referencia a la malla, pero utilizando diferentes posiciones en el mundo 3D. El resultado es que se puede crear todo un parque lleno de árboles con sólo una copia en memoria del mismo (ver figura 1). Adicionalmente, la clase obpObject implementa control LOD (Level Of Detail) esto permite que se puedan tener (por ejemplo) tres versiones de árboles, cada uno con un nivel de detalle más completo. Lo ideal es que cuando la cámara esté cerca al árbol, se pinte la versión compleja, pero a medida que se aleja se utilizen versiones más sencillas del árbol, para así aumentar el rendimiento de la aplicación (ver figura 2).

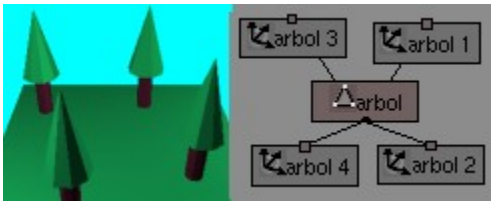

Figura 1: los cuatro árboles de la derecha son en realidad instancias que apuntan a la misma malla.

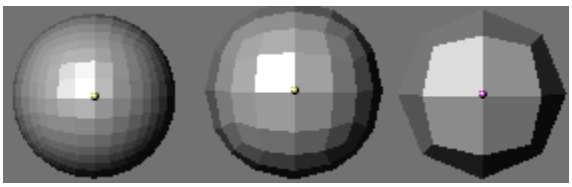

Figura 2: las tres mallas representan al mismo objeto, pero dependiendo de la distancia a la cámara se carga el más sencillo.

2.4.Manejo de terrenos

Cuando se trata de gestionar escenarios, no basta con automatizar la carga de la geometría y optimizar el pintado, de lo cual se encargan las clases presentadas en la sección 2.3. además, hay que detectar y manejar las colisiones de los personajes contra los objetos del escenario (por ejemplo, paredes) y además procurar que sigan la geometría del terreno. En la librería Obplus se implementó un mecanismo llamado "Mallas de Navegación", basado en el artículo [2]. una malla de navegación es una malla cargada de un archivo OB+, pero que no va a ser visible en el juego (por ende, en el archivo se eliminan los bloques de textura, animación, materiales, etc..). la particularidad de esta malla consiste en que es una representación del terreno por el cual se pueden mover los personajes. Si en el escenario hay un árbol, por ejemplo, en el lugar correspondiente en la malla de navegación existirá un agujero que rodeará el tronco del árbol. El personaje está restringido a esta malla, así que sin que haya

una detección de colisiones directa contra el árbol, gráficamente se obtiene el efecto buscado: evitar atravesar el árbol. Adicionalmente, la malla de navegación permite seguir el relieve del terreno, implementando colinas, valles e incluso puentes. Ver figura 3.

Otro mecanismo implementado en la librería y relativo al manejo de terrenos son los quadtrees. Un quadtree es una estructura recursiva que permite subdividir el área en cuatro subespacios cada vez [3], [4]. el quadtree puede ser recorrido utilizando una cámara, para determinar qué objetos de la escena están dentro del cono de visión de la cámara, evitando así ordenar el pintado de objetos innecesarios. Ver figura 4.

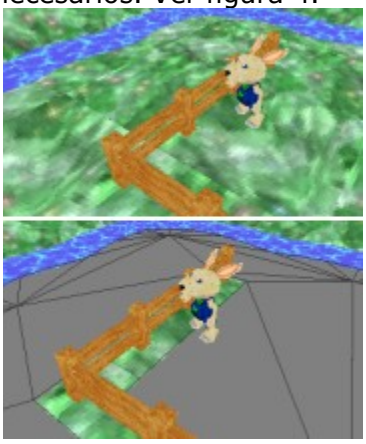

Figura 3: arriba, el personaje no puede atravesar la cerca. Parece como si detectara la colisión contra ella. Abajo se ve la malla de navegación utilizada, con el hueco alrededor del obstáculo.

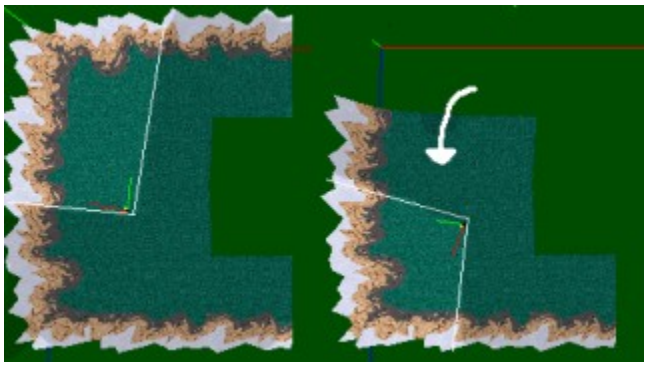

Figura 4: cuando la cámara rota, una parte del escenario deja de estar dentro de su cono de visión. El quadtree elimina las porciones no visibles para mejorar el rendimiento del sistema.

2.5.Animación

La clase obpPersonage fue desarrollada pensando en la utilización de la información de animación de un archivo ob+. El bloque de animación para un objeto básicamente consiste en la descripción de curvas de animación para la posición y rotación del objeto. Estas curvas se

representan utilizando puntos claves (keyframes) e interpolando entre ellos. Cuando un archivo ob+ trae información de jerarquía, las curvas de animación para cada uno de los elementos de la jerarquía se combinan para crear una animación compuesta, que debe ser recorrida recursivamente (ver figura 5). Los frames que componen una curva de animación se pueden agrupar por intervalos de tiempos. Estos intervalos son definidos en tiempo de modelado, y se conocen con el nombre de ACCIONES. Así, un personaje con información de animación de 200 frames, puede estar agrupado en 4 acciones, por ejemplo: CORRER, DESCANSAR, SALTAR, CAMINAR. La clase obpPersonaje tiene métodos Play(), Stop(), Loop(), que permiten controlar con facilidad las acciones. Ver figura 6.

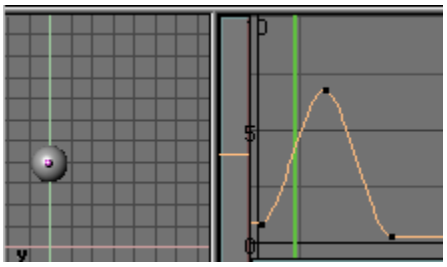

Figura 5: curva de animación (coordenada Y) de una esfera en lanzamiento vertical.

Figura 6: las curvas de animación se agrupan en acciones que tienen sentido en el juego.

2.6.Cámaras

Existen diversos tipos de cámaras, la elección de la cámara a utilizar depende mucho del tipo de juego y del efecto buscado. Entre las cámaras soportadas por la librería obpPlus están:

Cámara en primera persona: o cámara subjetiva. La cámara se ubica en donde estarían los ojos del jugador. Por lo tanto, el cuerpo del personaje no es visible. (un juego que implementa esta cámara es Quake).

Cámara en tercera persona: esta cámara está en una posición constante con respecto al jugador. Normalmente está atrás, de manera que siempre se vé la espalda del personaje.

Cámara en tercera persona dinámica: esta es una cámara en tercera persona que adicionalmente está montada sobre un sistema de resortes que producen retardos en la

respuesta de la cámara al movimiento del personaje. Provee una forma más realista de seguir al personaje, y es posible verlo de lado, e incluso de frente si rota con rapidez. Ver figura 7.

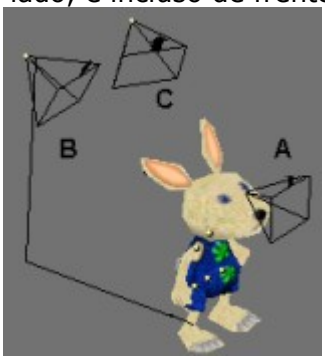

Figura 7: Tipos de Cámaras: A. Primera persona. B. Tercera persona. C. Tercera persona dinámica.

2.7.Manejo de tiempo

El manejo del tiempo es uno de los aspectos más importantes de un juego, y en el que fallan la mayoría de los principiantes [5]. Se trata de controlar la velocidad de los objetos del juego en función de la rapidez del equipo en el que se está ejecutando. Cuando esto no se toma en cuenta, al correr el juego en un equipo rápido los eventos se sucederán con demasiada rapidez. La librería obplus ofrece una clase (obpFramerate) para medir y calcular los factores de compensación entre equipos lentos y rápidos.

2.8.Inteligencia Artificial

En la librería se incluyó un esqueleto de trabajo sobre el cual se pudiese implementar comportamientos IA para los personajes de un juego. Este mecanismo es una implementación del propuesto en [6]: la especificación de un mini-lenguaje compatible con C (elaborado a base de #define`s) y un sistema de ruteo de mensajes de IA. Se implementó una versión para C++, de la cual surgieron las clases obpIAAgent, obpIAMessage y obpIARouter. A la hora de implementar un agente inteligente en un juego, basta con heredar de obpIAAgent y obpPersonage si se quiere animación. El mini-lenguaje de IA está diseñado para especificar máquinas de estados finitos, una solución clásica para la implementación de IA en juegos.

2.9.Clases utilitarias

Entre las clases utilitarias se encuentra la clase obpSkyDome, la cual permite generar mallas para cielos proceduralmente [7]. Asociado a una textura OBI, se puede lograr un cielo de gran calidad (ver figura 8).

Figura 8: cielo creado utilizando la técnica de SkyDomes.

3. METODOLOGIA

3.1. Origen de la librería OBPlus

El equipo de trabajo de la Librería Obplus está conformado por dos personas. Antes de empezar con la librería se estaba trabajando de manera paralela en dos proyectos separados, que aunque en escencia eran similares, tenían características especiales. En uno de ellos se necesitaba modelar un ambiente abierto gigantesco, con muchos edificios y objetos, lo cual promovió el desarrollo de las clases para manejo de escenarios, y especialmente escenarios dinámicos. En el otro proyecto, era más fuerte el requisito de animación de personajes, por lo que de allí surgieron las clases de animación. El formato de archivo OB+ se diseño debido a limitaciones en los exportadores existentes para Blender, el modelador que usamos en todos nuestros proyectos. Con el paso del tiempo, se decidió integrar todo esto en una librería más consistente.

3.2. Código Incorporado a la librería

Debido al pequeño tamaño del equipo de desarrollo y a la necesidad de mostrar resultados en los otros proyectos, la librería OBPlus ha incorporado elementos provenientes de otros autores (todos con licencias GNU), por ejemplo, las clases de Matrices y Vectores utilizadas en la librería provienen de la "Matrix Utility Library" de Dante Treglia II y Mark A. DeLoura [8]. En todos los casos, se ha especificado el origen del material, o se han realizado implementaciones propias basados en esas ideas, en algunos casos introduciendo mejoras (por ejemplo, la adaptación a C++ del modelo de paso de mensajes de IA).

3.3 Herramientas Auxiliares Desarrolladas

Obedeciendo al paradigma de precálculo de datos (ejecutar todos los algoritmos pesados fuera del juego, guardar los resultados en archivos y luego limitarse a cargarlos en el juego..) se han desarrollado pequeñas aplicaciones auxiliares que permiten optimizar el rendimiento de las aplicaciones que usan la librería. Estas aplicaciones son:

3.3.1. obpViewer: se trata de un visor de archivos ob+.. entre sus funcionalidades, está la de convertir a formatos binarios o de texto.. se planea añadir importadores y exportadores a otros formatos. También tiene algunas herramientas menos obvias, como un algoritmo que ordena las caras de un objeto de múltiples texturas de manera que queden agrupadas por textura, evitando la sobrecarga de llamadas a openGL en tiempo de pintado (ver figura 9).

3.3.2. obpWorldViewer: este programa es un visor de escenarios, utiliza scripts para identificar el número de mallas, el número de objetos y las posiciones de todos ellos. Permite cargar diversas mallas de visibilidad y probarlas para ver que tan bien se adaptan al terreno. También incluye un menú para precalcular los quadtrees, y guardarlos en archivos binarios (ver figura 10).

3.3.3. obpObiConversor: es un pequeño programa para convertir archivos BMP al formato OBI. Permite añadir canal Alpha.

3.3.4. Scripts para Blender: como blender es el modelador que utilizamos en todos nuestros proyectos, se han escrito scripts (utilizando el lenguaje Python) para exportar objetos desde blender al formato OB+ versión ASCII, y scripts que exportan todo un escenario para luego poder se cargado por el visor de mundos (ver figura 11).

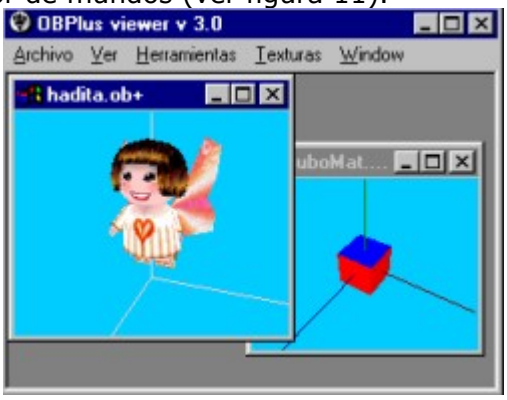

Figura 9:Visor de archivos OBPlus más algunas utilidades.

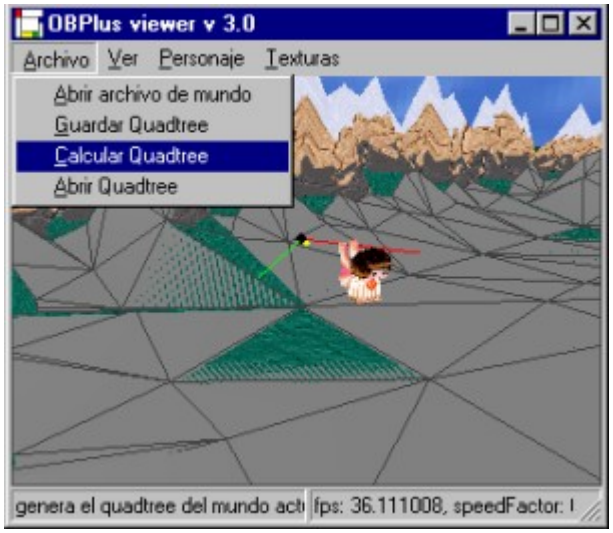

Figura 10: Visor de mundos. Se puede ver la malla de navegación en gris, la cámara en tercera persona (atrás del personaje) y el skydome.

Figura 11: Uso del programa Blender como herramienta de modelado. A la derecha puede verse un pedazo del código de exportación a OB+.

4. HERRAMIENTAS

La librería OBPlus está escrita en C++, y por lo tanto puede ser utilizada en cualquier compilador. Sin embargo, actualmente utiliza algunas clases provenientes de la librería wxWindows[9]. Básicamente, la clase wxString (que se está pensando cambiar por la versión STL) y las clases para manejo de directorios wxDir. La razón de ello es que en otros proyectos así como en el grupo de investigación OHWAHA [10] wxWindows es la herramienta utilizada para la elaboración de todas las aplicaciones. Obviamente, también se necesita que esté instalado OpenGL en el sistema. Como ni OpenGL ni la librería OBPlus ofrecen directamente mecanismos para manejar ventanas, teclado, mouse, etc.. el usuario debe elegir una biblioteca que le proporcione estas facilidades. Pueden ser GLUT, MFC (limitándose a MSWindowsy Visual C++), wxWindows (sobre la cual están desarrollados todos los programas mencionados en la sección 3.3).. estas bibliotecas tienen el inconveniente de que están más orientadas a aplicaciones no multimedia, por lo que actualmente estamos trabajando con SDL (Simple Direct Media Layer) [11], el cual tiene versiones para windows y linux (entre otros) y ofrece soporte para Mouse, teclado, Full Screen, Joystick, etc..

Finalmente, nuestra librería hace un uso intensivo de contenedores STL (Standard Template Library), los cuales son muy populares en el ámbito de la programación de juegos, y por ser estándar de C++ cuenta con versiones en la mayoría de los compiladores C++ existentes (MS Visual C++ en Windows y gcc en linux lo tienen).

5. RESULTADOS

Después de cerca de un año de trabajo con la librería, se ha conseguido una base sólida para el desarrollo de juegos. Uno de los más recientes, diseñado para nuestra tesis de grado (Juegos de computador para niños con problemas de aprendizaje) tardó dos días en implementarse, incluyendo el modelado de personajes, diseño de texturas y animación. Aunque este excelente tiempo se debe en parte a la sencillez del juego (un personaje moviendose por un ambiente, recogiendo objetos buenos y evitando malos) también muestra todo el poder de la librería, ya que implementa terrenos irregulares, quadtrees, animación, manejo de texturas y cámaras dinámicas, carga de distintos escenarios y personajes, etc.. a continuación aparecen algunos pantallazos de juegos desarrollados con la librería, en orden cronológico.

Pueblo Virtual (junio 2001): implementa un ambiente virtual en cual los usuarios pueden verse a través de internet, utiliza una arquitectura cliente servidor y está montado en wxWindows (figura 12).

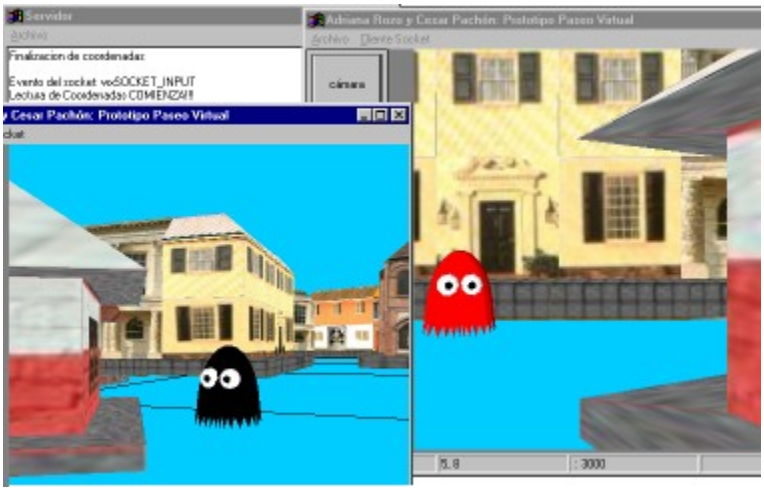

Figura 12: Pueblo Virtual. Cada usuario es representado por un fantasma. Lo que ve en su ventana corresponde a lo que el fantasma está viendo. Aquí los dos usuarios se están observando casi de frente.

Galeón (agosto 2001): juego de barcos de batalla en red, cada usuario maneja tres naves. Fue montado en wxWindows. (fig 13).

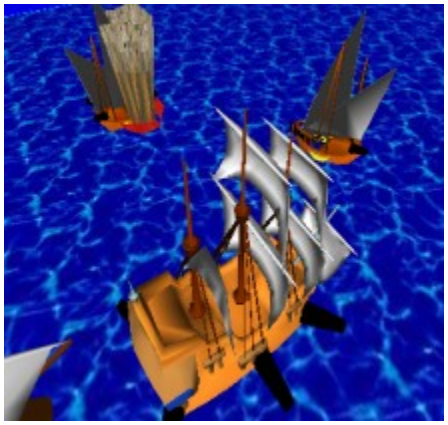

Figura 13: juego Galeón.

Manzanitas Locas (abril 2002): juego desarrollado como parte del trabajo de grado (juegos para niños con problemas de aprendizaje). Tardó dos días en ser implementado. En aquel momento lo único que no tenía la librería de lo mencionado en el artículo era el módulo de Inteligencia Artificial (fig 14). fue la primera vez que se usó SDL.

Laberintomanía (mayo 2002): este juego también se desarrolló para el trabajo de grado. Usa una versión de la librería muy similiar a la de manzanitas locas. Se desarrolló el juego sobre SDL, y un programa para crear nuevos laberintos sobre wxWindows (ambos usando la librería Obplus) (figura 15). Se añadió un emisor de partículas para celebrar el fín del juego.

Figura 15: Laberintomanía.

BuWarrior (junio 2002): Inicialmente se desarrolló exclusivamente para probar el módulo de inteligencia artificial, está totalmente sobre SDL. El personaje puede saltar, dar puños, y empujar

cajas explosivas. El administrador de cajas y el pájaro son agentes inteligentes (notese que el administrador de cajas es un agente virtual, no tiene representación física, y por lo tanto no hereda de obpPersonage, solo de obpIAAgent...) figura 16.

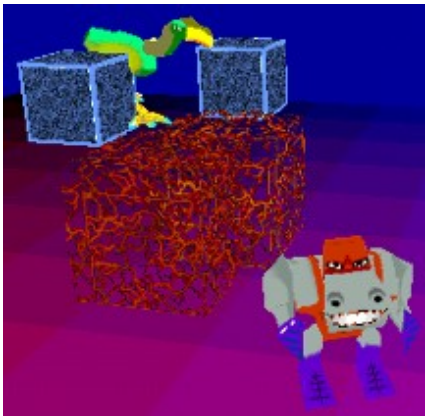

Figura 16: BuWarrior..

6. DISCUSION

La librería cumple con todos los objetivos planteados. Actualmente es un proyecto en pleno desarrollo, que justificaría una inversión de recursos humanos y de tiempo más fuerte. El rendimiento logrado en los juegos, especialmente en los tres últimos, ha sido muy bueno, hasta el punto de lograr correrlos en equipos Pentium de 150 Mhz sin tarjeta aceleradora de video (la configuración más baja en la que se han probado), y sin perder la jugabilidad.

7. CONCLUSIONES Y PERSPECTIVAS

La librería Obplus puede (y debe) seguir extendiendo sus capacidades, incorporando nuevas funcionalidades y depurando errores previos. Entre las cosas que se planean incorporar, están los sistemas emisores de partículas (en Laberintomanía se usaron, pero no forman parte oficial de la librería), mecanismos de detección de colisiones avanzados, clases para el manejo de iluminación dinámica, y mucho más. La librería puede ser utilizada en aplicaciones que no estén directamente relacionadas con los juegos, como productos multimedia y aplicaciones de realidad virtual.

8. BIBLIOGRAFIA

[1] Pachón Meneses Cesar Mauricio, Rozo Rojas Adriana, Especificación del Formato OB+ v2.0, Enero 2002

[2] Greg Snook, Simplified 3D Movement and Pathfinding Using Navigation Meshes. Game Programing Gems I, Edited by Mark DeLoura, editorial Charles River Media. Rockland,

Massachussets, pag 288.
[3] Foley, Van Dam, Feiner, Hughes, Philips. Algoritmos de subdivisión de áreas, Introducción a la graficación por Computador. (versión en español de Ernesto Morales Peake), Addison Wesley Iberoamericana. Pag 531.
[4] Jhonatan Ferraris, Quadtrees, www.gamedev.net
[5] Ben Dilts, Frame Rate Independent Movement, www.gamedev.net
[6] Steve Ravin, Designing a General Robust AI Engine, Game Programing Gems I, Edited by Mark DeLoura, editorial Charles River Media. Rockland, Massachussets, pag 221.
[7] Luis R. Sempé. Sky Domes, www.spheregames.com.
[8] Dante Treglia II and Mark A. DeLoura. Matrix Utility Library, Game Programing Gems I, Edited by Mark DeLoura, editorial Charles River Media. Rockland, Massachussets, pag 601
[9] www.wxWindows.org
[10] grupo de Investigación en Procesamiento de Imágenes y Computación Gráfica Ohwaha, http://dis.unal.edu.co/ohwaha