GUIA DE USUARIO: LIBRERÍA DE VIDEOJUEGOS RPGS DESARROLLADA DESDE C++ CON AYUDA DE LA LIBRERÍA ALLEGRO.

Jaime Alberto Ríos Palacio

Estudiante Ingeniería Electrónica

Universidad de Antioquia

Febrero 2013

La librería que se presenta facilitara la programación de un básico juego de rol. Está basada en 9 clases las cuales serán explicadas a continuación.

1. Clase Menú:
   Esta clase está pensada para presentar una pequeña interfaz de entrada al juego, posee su constructor, el cual recibe como parámetro un apuntador a const char el cual es la dirección relativa de la imagen que se usara para el fondo de la presentación del juego. Además posee un método el cual se encargara de ejecutar el menú de entrada y al cual se le debe pasar un booleano que puede ser usado para ingresar a la rutina del juego la cual por lo general es una rutina while o do while, es decir cuando la persona elija la opción nuevo juego se le asignara al booleano el valor falso, si se elige la opción salir el booleano se le asignara el valor verdadero.
   1.1. Métodos
   - Menu(const char *ruta);
   - Void Ejecutar_Menu(bool &done);

   1.2. Instanciación de un objeto de la clase menú:
   Menu Mi_Menu(“Direccion_relativa_de_imagen_menu_entrada”)

   1.3. Ejemplo de Ejecutar_Menu:
   bool done;
   Mi_Menu.Ejecutar_Menu(done);

2. Clase Mapa
   Esta es una de las clases que se considerarían las bases del juego en sí, los mapas, son las encargadas de realizar crear los entornos de exploración del juego rpg, mapas de pueblos, dungeons y entornos exteriores como el mapamundi del juego.
   Sus métodos son los siguientes:
   - Mapa(int Filas, int Columnas, const char* filename, BITMAP *Suelo, int px, int py);

- void TocarMusica(void);
- void ParарMusica(void);
- void Pintarmapa(BITMAP *Destino);
- void Pintarmapa(BITMAP *Destino, int capa, int x, int y);
- void PintarEnMapa(BITMAP *Origen, int d_x, int d_y, int e_x, int e_y, int t_x, int t_y, int capa);
- void ActualizarMapaMovimiento(int F, int C, int valor=1);
- void ActualizarMapaEventos(int F, int C, bool valor, int TE, int TEE);
- void RadarMov(bool &Arriba, bool &Abajo, bool &Derecha, bool &Izquierda, int y, int x);
- void RadarEventos(bool &Hay, int &Tipo, int &Espe, int y, int x);
- int GetTamX(void);
- int GetTamY(void);
- void mapmov(void);
- void Activa_Modo_Pelea(int Pasos);
- void Desactiva_Modo_Pelea(void);
- void Decrementar(bool &Pelee);
- ~Mapa();

Su constructor recibe como parámetros las dimensiones de X y Y, que en pantalla se verá reflejado en pantalla por un mapa de resolución de 32Xx32Y pixeles. Filename es un apuntador a const char que es la dirección relativa de la pista de audio que se tocara en el mapa mientras el jugador se encuentre en él. Suelo es una imagen de la cual será extraído el fondo que se usara para el mapa, se le debe proporcionar la ubicación del pedacito de suelo, una imagen de 32x32 pixeles o la ubicación de esta pequeña imagen en una imagen más grande, esto será explicado en el video con más detalle.
Los métodos TocarMusica y ParарMusica son usados para detener o iniciar las rutinas de música de los mapas, generalmente usadas cuando se desea realizar un cambio de mapa.
Los métodos Pintarmapa son usados para pintar los mapas en pantalla, el programador que use esta librería no se debe preocupar de realizar esta acción ya que se esa tarea será ejecutada por la clase GestordeMapas.
El método pintar en mapa es para pintar en el mapa las imágenes de la casas, los eventos cuando involucran NPC, entre otros, la librería tiene ciertos recursos, que los usa la clase Pintor, pero si el programador desea ingresar sus imágenes lo puede hacer por este método.
El método ActualizarMapaEventos es usado para ingresar unos enteros que contendrán la información de los eventos con los cuales el programador se las ingeniara para iniciar sus eventos en el juego que se muestra en el video se realizan casos para interactuar con esto.
El método ActualizarMapaMovimiento se usa para poner obstáculos al personaje e impedir su movimiento por estos lugares, por ejemplo impedir que el personaje atraviese arboles u otras cosas.
Los métodos radar son usados para leer estos datos en procesos de juego, el encargado de realizar estas acciones es la clase gestor de mapas.

Los métodos Activar y desactivar modo pele son usados para tener un mapas con o sin peleas, inicialmente cuando se instancia un objeto de Mapas, estos inician en modo de no peleas por defecto.
Y el método Decrementar se usa como contador para saber cuándo se debe ejecutar una pelea.
Un ejemplo de una instanciación de un objeto de mapa.

```
const char *cancion= "../Musica/Metalworks.wav";
BITMAP *Pisos = load_bitmap("../pueblo/pueblo1.tga", NULL);
Mapa pueblo(34,48,cancion,Pisos,0,32);
```

3. Clase Pintor
   Esta clase es usada para pintar objetos, entornos, casas, pueblos, entre otras cosas. Sus métodos usualmente se ejecutan enviándoles un mapa por referencia y las posiciones donde se pintara el objeto, estos actualizaran los mapas de movimientos, entre otras cosas. Todas las cosas que se pintan tiene las posiciones x,y como referencias a columnas de una matriz que tiene como base un cuadro de 32x32

   Métodos:
   - Pintor(void);
   - ~Pintor(){};
   - void Casa(Mapa &lienzo, int posx, int posy, int E_E);//pinta una casa en el mapa lienzo en la posición (posx,posy)
   - void Tienda(Mapa &lienzo, int posx, int posy, int tipo); //pinta una Tienda tipo=0 Armas, tipo=1 Objetos en el mapa lienzo en la posicion (posx,posy)
   - void Arbol(Mapa &lienzo, int posx, int posy);//pinta un Arbol en el mapa lienzo en la posición (posx,posy)
   - void Academia(Mapa &lienzo, int posx, int posy);//pinta un Academia en el mapa lienzo en la posición (posx,posy)
   - void Camino(Mapa &lienzo, int posx, int posy, int TamX, int TamY); //pinta un sendero en el mapa lienzo en la posición (posx,posy), 32*TamX*32*TamY
   - void Fuente(Mapa &lienzo, int posx, int posy);//pinta un fuente en el mapa lienzo en la posicion (posx,posy)
   - void Lampara(Mapa &lienzo, int posx, int posy);//pinta un Lámpara en el mapa lienzo en la posición (posx,posy)
   - void BorInCa(Mapa &lienzo, int posx, int posy, int lonx, int lony, int E_E);//pinta bordes internos de casa en el mapa lienzo en la posición (posx,posy)
   - void MenuCombate(void);
   - void MenuCombateObjetos(void);
   - void Pueblo(Mapa &lienzo, int posx, int posy);
   - void Montana(Mapa &lienzo, int posx, int posy);
   - void Bosque(Mapa &lienzo, int posx, int posy);

- void Bosque(Mapa &lienzo, int posx, int posy, int Tamx, int Tamy);
- void Pueblo_Sin_Entrada(Mapa &lienzo, int posx, int posy, int tipo);
- void Pequena_Construccion(Mapa &lienzo, int posx, int posy, int tipo);
- void Arbol_Exterior(Mapa &lienzo, int posx, int posy, int tipo);
- void Paredes_Rocosas(Mapa &lienzo, int posx, int posy, int cuantos);
- void Escaleras(Mapa &lienzo, int posx, int posy, int cuantos);
- void Furnishing(Mapa &lienzo, int posx, int posy, int Tipo);

La instanciación de un objeto de esta clase es muy sencilla:
Pintor Mi_Pintor;

4. Clase Eventos

   La funcionalidad de esta clase se empareja mucho con el método actualizar eventos de la clase mapa, primero se crea el evento y con la función radar se puede leer estas variables de los eventos creados para poder realizar un esquema de eventos y ejecutarlos al cumplir ciertos requisitos como presionar una tecla o ubicarse simplemente sobre el evento.

   Metodos de la clase eventos:

   - Eventos(int NumeroPersonajes,const char *texto1=NULL, const char *texto2=NULL, const char *texto3=NULL,const char *texto4=NULL);
   - ~Eventos(){};
   - void Crear_Cambio_Mapa(Mapa &lienzo, int posx, int posy, int TEE_E);
   - void Crear_Evento(Mapa &lienzo, int posx, int posy, int TEE_E);
   - void Historia(BITMAP *Fondo, const char *texto);
   - void conversacion(int _Personaje, const char *texto, int Posicion);

   El constructor de la clase recibe como parámetros el número de personajes principales del juego que puede ir de 1-4, además que se debe enviar las direcciones relativas de las imágenes que se usaran para los eventos de conversación entre los personajes.
   Los métodos crear se usan para crear un evento de cambio de mapa o crear un evento de conversación u otro tipo. Esto queda abierto a la imaginación del programador y al uso del recurso que se le proporciona. Estos métodos reciben como parámetros el mapa en donde se creara el evento y la posición así también como un entero con el cual se identificara el evento especifico.
   El método historia lo que proporciona es un recurso para crear las historias. El programador creara un archivo de texto, que puede ser creado desde el programa bloc de notas, y le proporcionara a este método un BITMAP, que será usado como fondo y la dirección relativa del documento previamente creado.

El método conversación usara las imágenes de los personajes que se le proporciono al constructor para crear eventos de conversación entre personajes o de una ventana de conversación sin personaje.

5. Clase Gestor de Mapas

   La clase GestordeMapas se usa como su nombre lo indica para "gestionar" los mapas del juego, desde ella se puede iniciar o parar la música de un mapa, "moverse" y recorrer los mapas, y detectar si se encuentra parado dentro de un evento. Todo esto gracias a que posee un apuntador de la clase mapa desde el cual se invocan los métodos de esta clase.

   Métodos:

   - GestorMapas(Mapa *ApunMap);
   - void CambioMapa(Mapa *Mapa_Apuntador, bool ON);
   - void Moverse(bool &PELEE);
   - void Posicion(void);
   - void Evento(bool &Hay, int &Tipo, int &Especifico);
   - void ActualizarMov(int Col, int Fil);
   - void Parar_Musica(void);
   - void Reproducir_Musica(void);
   - ~GestorMapas();

Para instanciar una clase de gestor de mapas se le debe pasar un apuntador de la clase de mapas que posea la dirección de un objeto de la clase mapa previamente creado.
Para el método cambio de mapa se le debe proporcionar lo mismo que el constructor más un valor true o false que indica si en el cambio de mapa se cambiara de música (true) o no (false).
Moverse nos permite saber cuándo permitirle el movimiento al personaje además que realiza el respectivo movimiento del personaje en el mapa. A este método se le ingresa por referencia una booleana que nos servirá para saber cuándo hay una pelea luego de haberse movido los respectivos pasos en el mapa.
Evento se usa para detectar cuando hay eventos y saber qué tipo de evento hay asi como su especifico.
Actualizar movimiento se usa generalmente cuando se cambia de mapa para saber a qué posición en el nuevo mapa debe ser desplazado el personaje.
Y los métodos parar y reproducir música se usan para los eventos por si se quiere detener la música y dar comienzo a otro sample.

6. Clase Luchador y sus clases herederas Personaje y Enemigos

   6.1. Clase Base Luchador:

   Es la clase base de un luchador del juego esta contiene los valores de los estatus(defensa, ataque, puntos de vida y otros) y el nombre, métodos para mirar estos valores, y otros mencionados a continuación.

   Métodos:

- Luchador(int _HP, int _MP, int _STR, int _MAG, int _DEF, int _MDEF, const char *_Nombre, BITMAP *PELEA, int Tamx, int Tamy);
- ~Luchador();
- int GetStatus(const char *s1);
- void Ataque(Luchador *Peleador, const char *s1);
- void Item(int pocion);
- void Recargar(void);
- virtual void Pintar_pelea(int i, int j);
- const char* Get_Nombre(void);

En el juego generalmente no se instancian este tipo de objetos, son una clase base que nos ayudara a tener componentes en común de las dos clases herederas Personaje y Enemigo. Sin embargo sus métodos nos ayudan ya que para generalizar las cosas la clase pelea se encarga a través de apuntadores de la clase base de realizar una pelea.

Su método de ataque se usa para atacar a un personaje mandándole el apuntador del objeto luchador que atacara y un apuntador a const char que indica el tipo de ataque.

El método poción se usa para usar pociones en el personaje y recuperar energía, y el método recargar recupera todo la energía, usado para que los personajes enemigos que fueron vencidos recarguen sus niveles de salud para estar listos para posteriores combates.

Posee un método virtual que no hace nada, solo para referencia que las clases herederas tiene un método respectivo dependiendo de cuál clase es.

6.2. Clases heredera personaje y enemigo

Son clases que heredan de luchador pero que poseen algunas diferencias, las imágenes de los enemigos son variables que las de los personajes, en su constructor se especifica esto

- Personaje(int _HP, int _MP, int _STR, int _MAG, int _DEF, int _MDEF, const char *_Nombre, BITMAP *PELEA, int Tamx, int Tamy);
- Enemigo(int _HP, int _MP, int _STR, int _MAG, int _DEF, int _MDEF, const char *_Nombre, BITMAP *PELEA, int Tamx, int Tamy);

7. Clase Peleas

Finalmente se tiene la clase de peleas la cual usa los luchadores para entrar a las luchas, el desafío de los juego.

Sus métodos:

- Peleas(void);
- void Lucha(Luchador &Peleador1, Luchador &Peleador2, Luchador &Peleador3, Luchador &Enemigo1, Luchador &Enemigo2, Luchador &Enemigo3);
- ~Peleas();

Su principal método es el encargado de entablar la pelea, se le pasan los 6 peleadores. 3 personajes y los 3 enemigos. El combate termina cuando uno de los grupos tiene a todos sus integrantes con niveles de hp en cero.

NOTAS:

1. Se necesita la versión 4 de allegro para este proyecto
2. Se compilo el proyecto con la siguiente instrucción: g++ Main.cpp Mapa.cpp Pintor.cpp GestorMapas.cpp Personaje.cpp Peleas.cpp Eventos.cpp Menu.cpp `allegro-config --libs` -o Juego

Dirección del video

http://www.youtube.com/watch?v=g8cXGdzuNxY&feature=youtu.be

Correo electrónico: shuin379@hotmail.com