

MulticocoSDL

José Ladislao Lainez Ortega y José Molina Colmenero

15 de mayo de 2013

Índice

1. Introducción	2
2. Bibliotecas	3
2.1. SDL	3
2.2. SDL_Mixer	3
2.3. SDL_TTF	4
3. Imagen	5
3.1. Window	5
3.1.1. Inicialización	5
3.2. Sprite	5
3.3. SpriteSheet	5
4. Audio	6
4.1. Sound	6
4.2. Music	6
5. Lógica	7
5.1. Vector2D	7
5.2. CollisionBox	7
6. Juego	8
6.1. Inicialización	8
6.2. Visualización	8
6.3. Eventos	8

1. Introducción

MulticocoSDL es un juego arcade que emula al famoso Pacman realizado como proyecto para la asignatura Sistemas Multimedia del Grado de Ingeniería Informática de la Universidad de Jaén.

El objetivo de MulticocoSDL es alcanzar varias areas multimedia en un mismo programa haciendo uso de los tres elementos principales del software multimedia:

Emplazamiento espacial

Los distintos elementos visuales (enemigos, escenario, pacman) son dibujados sobre un canvas en posiciones especificas y adems se puede mover por l.

Control temporal

El jugador puede moverse por el escenario a una determinada velocidad as como los enemigos, estando todos ellos animados.

Interacción

Mediante el teclado el jugador puede dar órdenes a pacman de forma que este se mueva en la dirección que el jugador le indica.

El código se encuentra alojado en GitHub en el siguiente enlace y contiene un archivo de proyecto de XCode para Mac OS X, si bien el código es portable a Windows y Linux.

`https://github.com/L4D15/MulticocoSDL`

2. Bibliotecas

Se ha hecho uso de la biblioteca Simple DirecMedia Layer así como de algunos submódulos de esta para trabajar con el renderizado de imágenes, texto y reproducción de audio. A continuación explicamos qué tareas ha realizado cada una.

Para más informacin sobre Simple DirecMedia Layer:

<http://www.libsdl.org/>

2.1. SDL

Biblioteca con las operaciones básicas para crear una ventana y dibujar en ella. Algunas de las utilidades más usadas han sido:

SDL_Rect

Estructura para definir un rectángulo. Usado a la hora de recorta un área de un SpriteSheet y dibujar en un canvas.

SDL_Surface

Superficie o canvas sobre el que dibujar, usado no solo en la ventana principal como contexto de renderizado, sino también para almacenar en la memoria de la tarjeta gráfica los distintos Sprites a renderizar.

SDL_BlitSurface(SDL_Surface*,SDL_Rect*,SDL_Surface*,SDL_Rect*)

Mediante esta función se puede dibujar un área seleccionada de un canvas origen en un área de un canvas de destino. Se ha usado tanto para dibujar en la ventana principal como para separar los distintos Sprites del SpriteSheet.

SDL_LoadBMP(const char*)

Como bien indica su nombre carga una imagen en formato BMP, la guarda en memoria gráfica y devuelve un puntero a una SDL_Surface, de modo que podamos usar la imagen cargada.

Esta biblioteca puede descargarse desde el siguiente enlace:

<http://www.libsdl.org/download-1.2.php>

2.2. SDL_Mixer

Biblioteca modular de SDL que facilita el trabajo con la reproducción de sonido y música. Los elementos más destacados de esta biblioteca son:

Mix_Chunk

Contenedor de sonido, igual que SDL_Surface lo era de imágenes.

Mix_Music

Contenedor de sonido específico para música.

Mix_LoadWAV(const char*)

Carga un archivo en formato WAV y devuelve un puntero a Mix_Chunk para poder trabajar con el audio del archivo.

Mix_LoadMUS(const char*)

Carga un archivo en formato WAV, OGG, MP3 o FLAC devolviendo un puntero a Mix_Music. Esta función es específica para cargar música ya que SDL_Mixer trabaja de forma distinta los sonidos y la música.

La biblioteca se puede descargar desde:

http://www.libsdl.org/projects/SDL_mixer/

2.3. SDL_TTF

A la hora de mostrar la puntuación del jugador necesitábamos mostrar texto por pantalla, por lo que recurrimos a esta biblioteca (otro módulo del proyecto SDL) específica para mostrar texto por pantalla. Una curiosidad sobre esta biblioteca es que hace uso de fuentes TTF en lugar de recurrir a fuentes en archivos bitmap como sucede, por ejemplo, al renderizar texto en OpenGL.

De esta biblioteca hemos usado:

TTF_Font

Contenedor para la información de la fuente a usar a la hora de renderizar el texto.

TTF_OpenFont(const char*, int)

Carga una fuente desde el archivo en formato TTF especificado y usando el tamaño indicado.

TTF_RenderText_Solid(TTF_Font, const char*, SDL_Color)

Crea una superficie sobre la que dibuja el texto pasado usando la fuente indicada y el color deseado. Una vez tengamos esa superficie habrá que dibujarla usando la función SDL_BlitSurface(...).

Esta biblioteca está disponible en:

http://www.libsdl.org/projects/SDL_ttf/

3. Imagen

En esta sección explicaremos las clases relacionadas con el apartado visual de la aplicación, desde la creación de la ventana hasta la animación de los personajes que aparecen en pantalla.

3.1. Window

Antes de poder dibujar nada en pantalla primero necesitamos una ventana para nuestra aplicación, y una vez esta esté creada necesitamos controlar el flujo del juego. Vayamos por partes.

3.1.1. Inicialización

SDL nos proporciona facilidades a la hora de crear una ventana preparada para renderizar imágenes, pero antes de poder usarlas necesitamos inicializar SDL.

```
int err = SDL_Init(SDL_INIT_AUDIO | SDL_INIT_VIDEO);

if (err < 0) {
    // Mostrar error
    std::cout << "Error al inicializar SDL: " <<
        SDL_GetError() << std::endl;
    exit(1);
}
```

Mediante los flags `SDL_INIT_AUDIO` y `SDL_INIT_VIDEO` le indicamos que nuestra aplicación va a hacer uso de estas dos funcionalidades. Si no indicáramos, por ejemplo, el flag de audio, nuestra aplicación no reproduciría audio.

Si hubiera algún problema durante la inicialización, SDL puede darnos información sobre el fallo mediante `SDL_GetError()`. El uso de esta función para obtener información sobre fallos durante el uso de SDL será una constante a lo largo del proyecto, por lo que se omitirá el tratamiento de errores de ahora en adelante. Para más detalles sobre ello consultar el código fuente.

3.2. Sprite

3.3. SpriteSheet

4. Audio

4.1. Sound

4.2. Music

5. Lógica

5.1. Vector2D

5.2. CollisionBox

6. Juego

6.1. Inicialización

6.2. Visualización

6.3. Eventos