Laboratorio 10

Programación de Computadores (2023-2)

Temas del Laboratorio 10

- Simple Directmedia Layer
- Crear y mantener abierta una ventana
- SDL_Window y SDL_Render
- SDL_Rect
- Funciones de dibujo
- Events
- KeyboardState
- Otras funciones varias
- Y las structs?

SDL es una librería de C y C++ para la creación de ventanas gráficas, acceso a lectura directa del teclado y mouse.

También con sus otras librerías como SDL_Mixer, SDL_Image, etc, se puede acceder a más funciones, como implementación de audio, manejado de archivos .png o .jpg, entre otros

https://www.libsdl.org/

About SDL

Simple DirectMedia Layer is a cross-platform development library designed to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D. It is used by video playback software, emulators, and popular games including Valve's award winning catalog and many Humble Bundle games.

SDL officially supports Windows, macOS, Linux, iOS, and Android. Support for other platforms may be found in the source code.

SDL is written in C, works natively with C++, and there are bindings available for several other languages, including C# and Python.

SDL 2.0 is distributed under the zlib license. This license allows you to use SDL freely in any software.

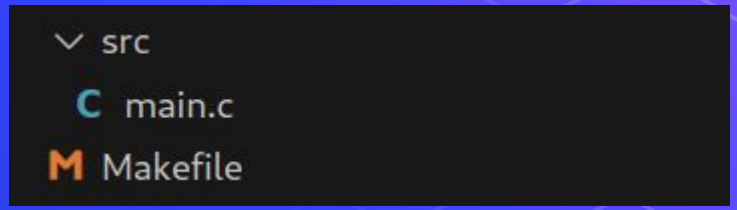
Made with SDL: Mayhem Intergalactic

Made with SDL: Dota 2

How to sdl:

Paso 1: Boilerplate

Esta será la estructura del proyecto



Usaremos este comando para compilar:

```
game:
    gcc -std=c11 -Wall src/main.c -o game.out -lSDL2
```

How to sdl: Inicialización

```
#include <SDL2/SDL.h>
```

Incluimos la librería

```
//Inicializar sdl, todos sus subsistemas y checkear por error
if(SDL_Init(SDL_INIT_EVERYTHING) != 0){
    printf("SDL_Init error %s\n", SDL_GetError());
    return EXIT_FAILURE;
}
```

Inicializar subsistemas, también se pueden inicializar algunos solamente (SDL_INIT_VIDEO, SDL_INIT_AUDIO, etc)

(y se checkea si hubo error)

How to sdl: Ventanas

```
#define WIDTH 1280
#define HEIGHT 720
```

```
//Crear ventana
SDL_Window *window;
window = SDL_CreateWindow("Ventana de ejemplo",
                          SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
                          WIDTH, HEIGHT,
                          SDL_WINDOW_SHOWN);

if(window == NULL){
    printf("SDL_CreateWindow error %s\n", SDL_GetError());
    return EXIT_FAILURE;
}
```

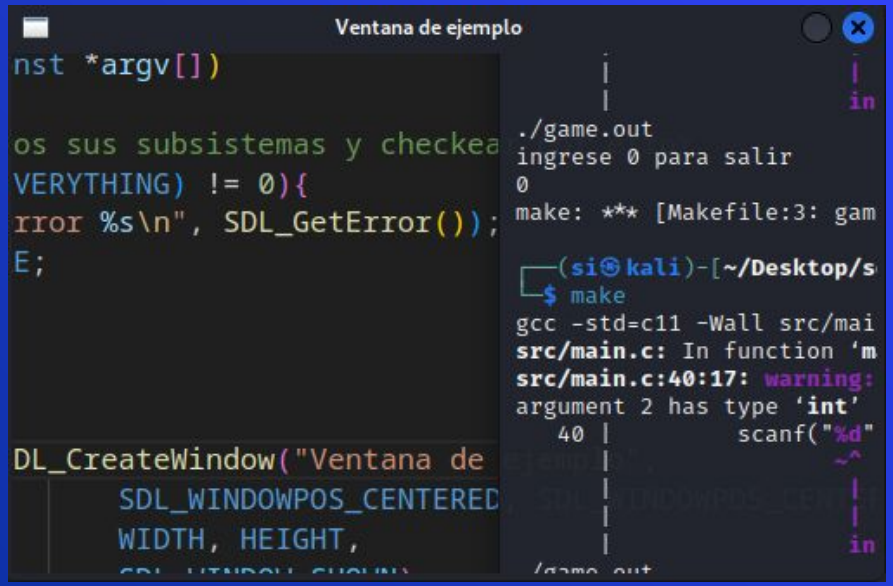
Crea una ventana de título “ventana de ejemplo”, centrada en x e y, de dimensiones WIDTH x HEIGHT y visible

(y se checkea si hubo error)

How to sdl: Ventanas

Con esto más un loop que controle el exit del programa ya podemos abrir una ventana.

Obteniendo algo como esto:




```c
int x = 1;
while (x){
    printf("ingrese 0 para salir\n");
    scanf("%d", x);
}
```

Esta ventana no se puede cerrar con la “x” ni con alt f4 ya que no hemos definido qué sucede en el evento de salir de la ventana. Por ahora solo podemos cerrar con la consola o con el administrador de tareas

How to sdl: Renderer

Esta estructura se encarga de dibujar sobre una ventana

```c
SDL_Renderer* renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
if (renderer == NULL) {
    fprintf(stderr, "SDL_CreateRenderer Error: %s\n", SDL_GetError());
    SDL_DestroyWindow(window);
    SDL_Quit();
    return EXIT_FAILURE;
}
```

Aquí creamos un renderer que dibujará sobre la ventana window, con la flag para usar aceleración por hardware (para agregar más flags se deben hacer OR “FLAG1 | FLAG2 | etc”) (y se checkea por errores)

SDL_Rect

Es un objeto que contiene 4 valores tipo int en su interior y representa un rectángulo.

(lo que se conoce como struct y verán más adelante)

Se declara igual que una variable y luego se accede a ella de la siguiente forma:

```
SDL_Rect rectangulo;
rectangulo.x = 0;       //coordenada x del rectangulo
rectangulo.y = 0;       //coordenada y del rectangulo
rectangulo.w = 50;     //ancho del rectangulo
rectangulo.h = 50;     //alto del rectangulo
```

Funciones de dibujo

Libsdl wiki ->

SDL_RenderDrawLine
SDL_RenderDrawLineF
SDL_RenderDrawLines
SDL_RenderDrawLinesF
SDL_RenderDrawPoint
SDL_RenderDrawPointF
SDL_RenderDrawPoints
SDL_RenderDrawPointsF
SDL_RenderDrawRect
SDL_RenderDrawRectF
SDL_RenderDrawRects
SDL_RenderDrawRectsF
SDL_RenderFillRect
SDL_RenderFillRectF
SDL_RenderFillRects
SDL_RenderFillRectsF

SDL tiene múltiples funciones de dibujo, para usarlas, primero hay que llamar a `SDL_SetRenderDrawColor` para definir el color del “pincel”.

DrawLine recibe 2 puntos y dibuja una línea entre ellos.

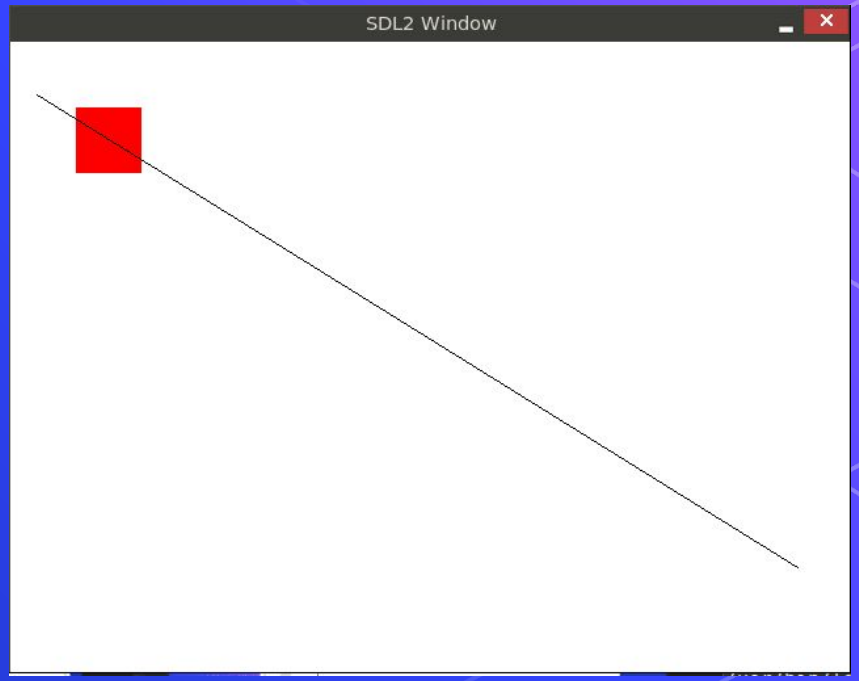
DrawRect y FillRect reciben un SDL_Rect para dibuja el contorno o rellenarlo.

RenderClear pinta toda la ventana.

Es necesario llamar a `SDL_RenderPresent(rend);` Para ver los cambios.

Ejemplo de dibujo

```c
SDL_Init(SDL_INIT_EVERYTHING);

SDL_Window *win = SDL_CreateWindow("SDL2 Window", 0, 0, 640, 480, SDL_WINDOW_SHOWN);
SDL_Renderer *ren = SDL_CreateRenderer(win, -1, SDL_RENDERER_ACCELERATED);

SDL_Rect rect; // creamos un objeto de tipo SDL_Rect
rect.x = 50;  // le asignamos valores alos 4 int de su interior
rect.y = 50;
rect.w = 50;
rect.h = 50;

// dibujamos el fondo blanco
SDL_SetRenderDrawColor(ren, 255, 255, 255, 255);
SDL_RenderClear(ren);

// dibujamos un rectangulo rojo
SDL_SetRenderDrawColor(ren, 255, 0, 0, 255);
SDL_RenderFillRect(ren, &rect);

// dibujamos una linea negra
SDL_SetRenderDrawColor(ren, 0, 0, 0, 255);
SDL_RenderDrawLine(ren, 20, 40, 600, 400);


// actualizar el render
SDL_RenderPresent(ren);

// esperar 5 segundos para que no se cierre la ventana de inmediato
SDL_Delay(5000);

// liberar memoria
SDL_DestroyRenderer(ren);
SDL_DestroyWindow(win);
SDL_Quit();

return 0;
```



https://wiki.libsdl.org/SDL2/SDL_RenderClear

How to sdl:

Paso 2: Manejo de inputs / events

Para esto haremos uso de `SDL_GetKeyboardState(NULL);` y `SDL_PollEvent(&evt)`
La primera nos da acceso a un arreglo interno de SDL que contiene los estados de todas las teclas del teclado y la segunda nos permite detectar eventos para poder ejecutar la respuesta correspondiente.

(Los eventos son acciones que hace el usuario sobre nuestra aplicación, como presionar teclas o hacer click)

How to sdl: Eventos

`SDL_Event` contiene información sobre un evento extraído con `SDL_PollEvent(&evt)`, esta función recibe un puntero a un `SDL_Event`, retorna la cantidad de eventos que hay en la fila, almacena la información del primer evento en la fila en `evt` y lo remueve de la misma.
Si no hay eventos en la fila, se retorna 0 y `evt` no se modifica.

Los eventos pueden ser de varios tipos como `SDL_QUIT`, `SDL_KEYDOWN` u otros.

How to sdl: Teclas

Si detectamos un evento de tipo `SDL_KEYDOWN` podemos utilizar el arreglo retornado por `SDL_GetKeyboardState()` para ver los estados de todas las teclas.

(1 si está siendo presionada 0 en caso contrario)

```
while(SDL_PollEvent(&evt)){
    if(evt.type == SDL_KEYDOWN){
        if(key[SDL_SCANCODE_W]){
            // codigo a ejecutar cuando
            // se presiona la tecla W
        }
        if(key[SDL_SCANCODE_A]){
            // codigo a ejecutar cuando
            // se presiona la tecla A
        }
        if(key[SDL_SCANCODE_S]){
            // codigo a ejecutar cuando
            // se presiona la tecla S
        }
        if(key[SDL_SCANCODE_D]){
            // codigo a ejecutar cuando
            // se presiona la tecla D
        }
        if(key[SDL_SCANCODE_SPACE]){
            // codigo a ejecutar cuando
            // se presiona la tecla ESPACIO
        }
    }
}
```

How to sdl: Cerrar el programa

```
SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(window);
SDL_Quit();

return EXIT_SUCCESS;
```

Al terminar el programa se deben llamar a las funciones destroy de las estructuras usadas y luego a sdl_quit para evitar fugas de memoria

(Recuerda hacer free de tus propias estructuras también.)