

Make Classic Games

Índice

- Introducción
- Raylib
- Bindings
- Arquitectura
- Instalación
- External Libs
- Herramientas extra
- Juegos Realizados Con Raylib
- Hola Mundo
- Ejemplos y funciones básicas

1. Introducción

A la hora de trabajar con aplicaciones que utilizan gráficos por ordenador (como por ejemplo juegos), existen diferentes formas de poder realizar este tipo de aplicaciones.

Entre otras cosas, hay que tener en cuenta muchos aspectos como puede ser el dispositivo a utilizar (PC, Móvil, etc...) ya que hoy en día hay muchos tipos de dispositivos.

Es por ello, que necesitamos diferentes herramientas para realizar juegos o cualquier otro tipo de aplicación. Como pueden ser motores o librerías gráficas.

1. Introducción

Motor (Engine)



GODOT
Game engine

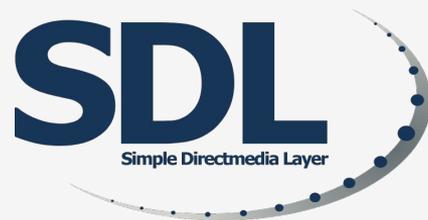


Unity



UNREAL
ENGINE

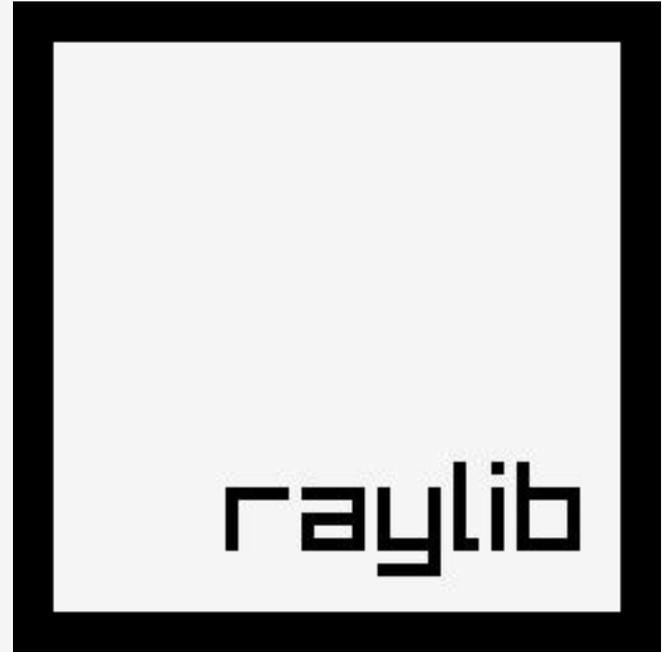
Librerías Gráficas



1. Introducción

En muchas ocasiones, el configurar y utilizar algunas de estas librerías gráficas pueden ser un desafío.

Por ello, existen librerías como *Raylib* que nos abstraen del uso de estas librerías.

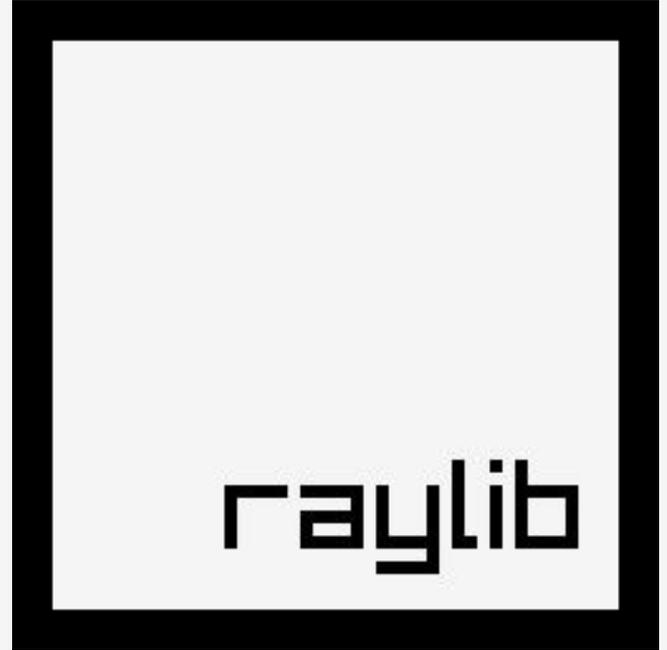


2. RayLib

Raylib es una librería (API) para poder crear juegos o herramientas de una manera simple; sin necesidad de utilizar grandes editores o herramientas.

Raylib permite abstraernos de toda la inicialización de diferentes herramientas o librerías como puede ser OpenGL, SDL, etc.

Además, integra una serie de funcionalidades muy sencillas y que permiten centrarse en la creación de herramientas y juegos simplemente programando.



2. Raylib

Raylib es OpenSource y es contribuido por muchos autores; aunque el autor principal es *Ramon Santamaria*.

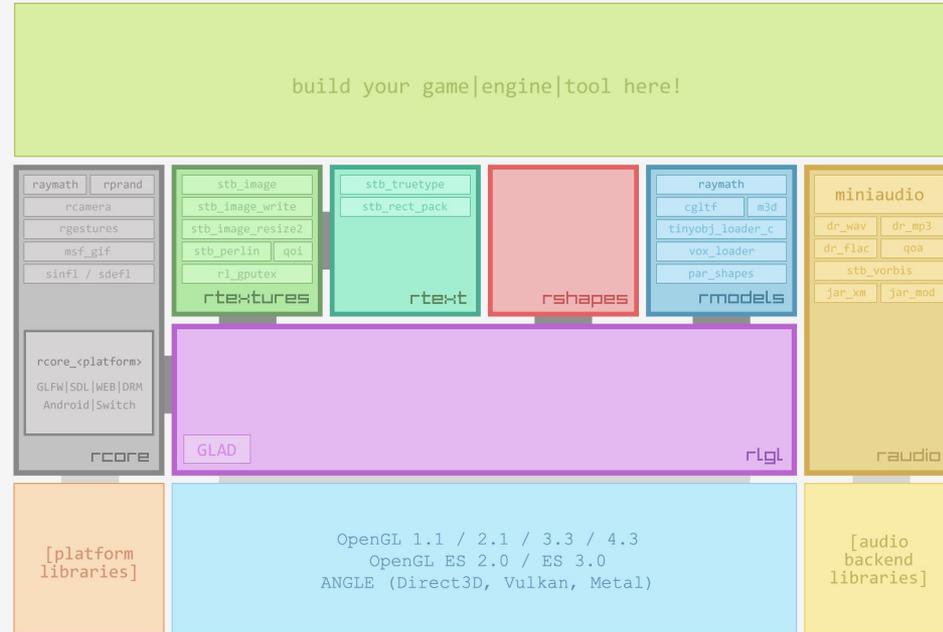
Entre otras, tiene las siguientes funcionalidades:

- Sin dependencias externas (las librerías se integran dentro de Raylib; o muchas de ellas ya están instaladas).
- Multiplataforma (windows, macos,linux, android,Rpi,etc...).
- Escrito en C.
- Múltiples Bindings a otros lenguajes (60+).
- Soporte tanto 2D como 3D (integrando modelos, texturas, etc...).
- Soporte para VR (Simulado).
- etc...

3. Arquitectura

raylib ARCHITECTURE 5.0

A simple and easy-to-use Library
to enjoy videogames programming



Supported platforms:         

4. Instalación

Hay diferentes métodos para instalar y/o utilizar Raylib. Siempre se trata de utilizar los métodos más sencillos para poder empezar a programar lo más fácil posible.

Vamos a ver los diferentes métodos para poder utilizar Raylib.

4. Instalación

Windows

Para este sistema operativo, pueden utilizarse diferentes métodos:

1. Utilizar Visual Studio Community (no confundir con Visual Studio Code).
2. MinGW-W64/GCC

En la wiki de Raylib en Github puedes encontrar más información.

<https://github.com/raysan5/raylib/wiki/Working-on-Windows>

4. Instalación

Linux

Podemos utilizar en Linux Raylib de forma sencilla. Simplemente instalando las herramientas de GCC y Make.

```
apt install build-essentials #solo para sistemas debian
```

Posteriormente, descargamos el código de Raylib desde github:

<https://github.com/raysan5/raylib>

NOTA: requerirán una serie de librerías como puede ser Alsa, Mesa for OpenGL o X11.

Por último, puede generarse la librería utilizando la herramienta make ejecutando los siguientes comandos en la carpeta *src*.

```
make PLATFORM=PLATFORM_DESKTOP # librería estática
```

```
make PLATFORM=PLATFORM_DESKTOP RAYLIB_TYPE=SHARED #librería dinámica
```

```
make PLATFORM=PLATFORM_WEB #versión para web.
```

Si queremos instalar la librería, podemos utilizar la opción `install` para que se genere e instale en nuestro sistema. Para más información:

<https://github.com/raysan5/raylib/wiki/Working-on-GNU-Linux>

4. Instalación

MacOs

Podemos utilizar diferentes formas como la utilización de la utilidad homebrew para instalar raylib.

```
brew install raylib
```

Pero si se quiere crear desde el código se debe tener instalado XCode (Con Apple developer Tools instaladas) y tener una versión de MacOS Bigsur (10.9) o superior.

Descargamos el código de Raylib:

<https://github.com/raysan5/raylib>

Ejecutamos la siguiente instrucción dentro de la carpeta src:

```
make PLATFORM=PLATFORM_DESKTOP
```

Posteriormente puede utilizarse la template para XCode para poder crear una aplicación usando raylib:

https://github.com/acejacek/raylib_xcode

5. Extra Libs

Además de lo integrado dentro de Raylib, podemos encontrar una serie de librerías extras (Algunas ya se encuentran integradas) para ayudarnos a crear nuestros juegos o aplicaciones.



<https://www.raylib.com/#extra-libraries>

5. Bindings

Existen más de 60 librerías (Bindings) en diferentes lenguajes de programación.

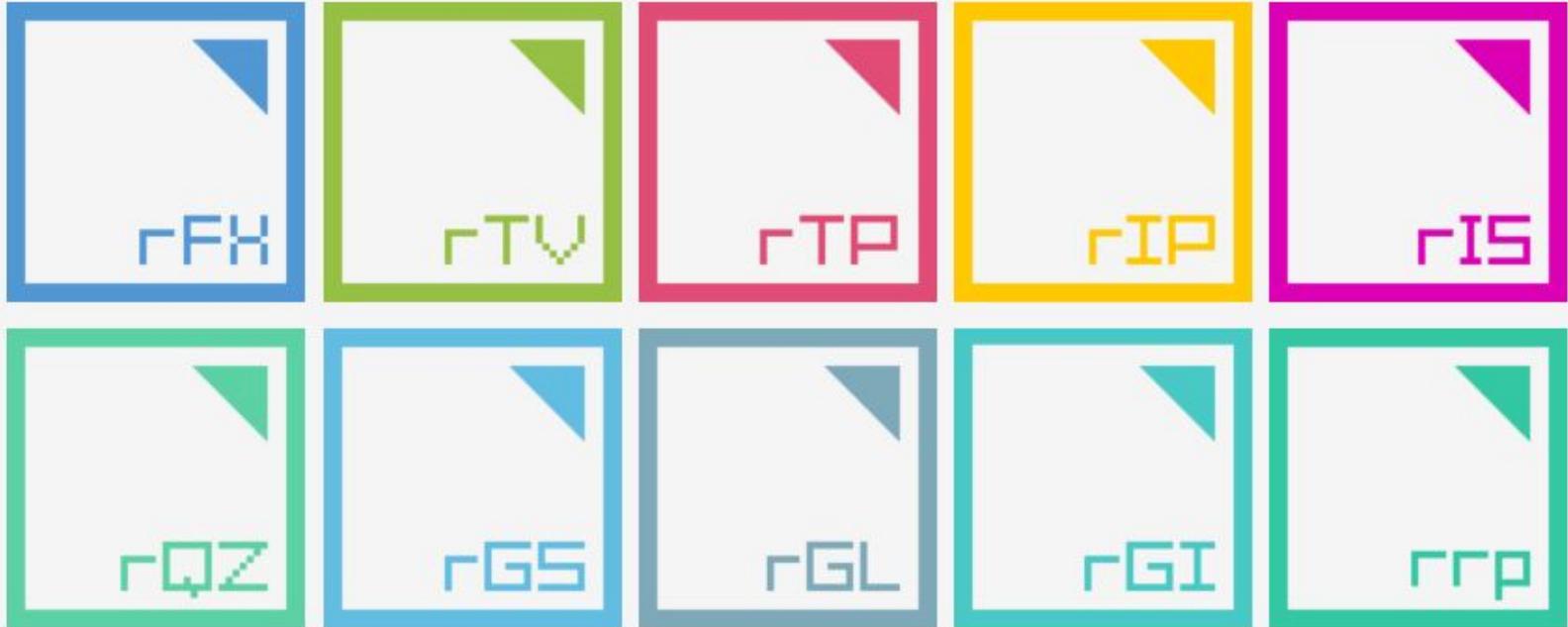
- Java
- Python
- Zig
- Odyn
- Go
- ...

Puedes encontrar más información en la siguiente dirección:

<https://github.com/raysan5/raylib/blob/master/BINDINGS.md>

6. Extra Tools

<https://www.raylib.com/#raylibtech-tools>



7. Juegos Realizados con Raylib

Podemos encontrar algunos juegos realizados con Raylib; algunos ya se encuentran en Steam.



<https://www.raylib.com/games.html>

8. Hola Mundo

Veamos el programa más sencillo; en este caso mostrar un texto por pantalla:

Algunas funciones interesantes son:

- *InitWindow*: Muestra la ventana.
- *WindowShouldClose*: devuelve verdadero cuando se deba cerrar la ventana.
- *Begin/EndDrawing*: Entra o sale del modo dibujo.
- *DrawText*: Muestra un texto por pantalla.

```
//Include Lib
#include <raylib.h>

//Main Function
int main(){
    //Window Dimensions
    const int screenWidth = 800;
    const int screenHeight = 450;

    //Init Window (width, Height, title)
    InitWindow(screenWidth,screenHeight,"Hello Raylib");
    //Set target FPS
    SetTargetFPS(60);

    //While windows is open
    while(!WindowShouldClose()){
        //init drawing mode
        BeginDrawing();
        //Clear Background
        ClearBackground(RAYWHITE);
        //Draw Text
        DrawText("Hello Raylib", 290,200,20,BLACK);
        //Finish Drawing Mode
        EndDrawing();
    }
    //Close Window
    CloseWindow();
    return 0;
}
```

9. Ejemplos

Puedes encontrar infinidad de ejemplos en la web de Raylib; puedes ejecutarlos directamente en la web.

<https://www.raylib.com/examples.html>

A continuación, veremos algunos ejemplos propios para ir preparándonos.

10. Formas

Se pueden crear diferentes formas para dibujar usando RayLib.

- Rectangle

```
DrawRectangleV(Vector2 posición, vector2 tamaño, Color);
```

- Circle

```
DrawCircle(Vector2 posicion, radio, color);
```

- Triangle

```
DrawTriangle(Vector2 v1, Vector2 v2, Vector2 v3, Color);  
//Sentido contrareloj
```

11. Input

Teclado

Pueden usarse funciones como *isKeyDown* o *isKeyPressed* para comprobar si se ha pulsado una tecla del teclado:

```
if (IsKeyDown(KEY_DOWN)) {  
    circlePosition.y += BALL_SPEED;  
}
```

11. Input

Gamepad

También se pueden utilizar hasta 4 controladores (gamepads) para nuestros juegos.

Algunas funciones como *IsGamepadAvailable* o si ha pulsado un botón *IsGamePadButtonDow*.

```
int xAxisMovement = GetGamepadAxisMovement (gamepadN, 0);  
int yAxisMovement = GetGamepadAxisMovement (gamepadN, 1);
```

11. Input

Ratón (Mouse)

Por supuesto, podemos utilizar el ratón (Mouse) de nuestro ordenador con funciones como *IsMouseButtonDown* o *GetMousePosition*.

```
if(IsMouseButtonDown(MOUSE_BUTTON_LEFT)){  
    mousePosition=GetMousePosition();  
    clicked=true;  
}
```

12. Texturas

Podemos cargar texturas utilizando ficheros de imagen compatibles con Raylib. Obviamente también se pueden aplicar a modelos 3D ya que tiene soporte para 3 Dimensiones.

Puede usarse la función *LoadTexture* para cargar una imagen como una textura en GPU.

No olvides usar la función *UnloadTexture* para descargarla de la GPU.

13. Referencias

- Web Raylib: <https://www.raylib.com/index.html>
- Github Raylib: <https://github.com/raysan5/raylib>
- Raylib CheatSheet: <https://www.raylib.com/cheatsheet/cheatsheet.html>
- Youtube Raylib: <https://www.youtube.com/c/raylib>
- Repositorio Ejemplos: <https://github.com/makeclassicgames/raylib-examples>