

# *1. SDL – Simple DirectMedia Layer*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

francisco@ime.uerj.br



# 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, Mac OS X, 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.

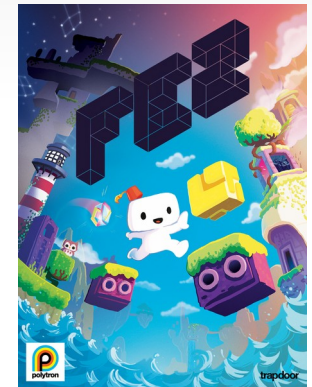
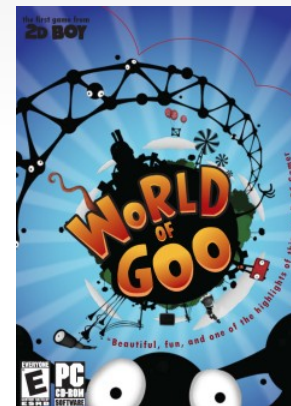
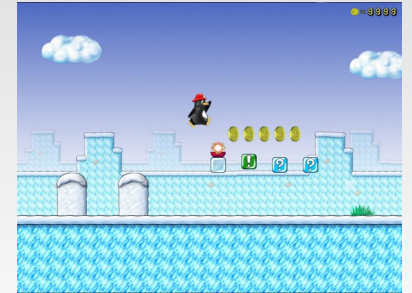


# SDL

- *Simple DirectMedia Layer*
- Acesso “baixo nível” ao áudio, teclado, mouse, joystick, gráficos, etc.
- Jogos, vídeos e aplicações gráficas
- Multi-plataforma:
  - *Windows, Linux, MacOS*
  - *Android, iOS*
- Implementado em C
  - Bibliotecas para *Python, Lua*, etc.
  - Game engines: *Pygame, Löve*

# Jogos

- Código aberto
  - *Freeciv, SuperTux*
- Independente (Indie)
  - *World of Goo, FEZ*
- Profissional (AAA)
  - *Quake 4, Unreal*



# Links

- Site Oficial
  - [libsdl.org/](http://libsdl.org/)
- Tutorial do Lazy Foo
  - [lazyfoo.net/tutorials/SDL/](http://lazyfoo.net/tutorials/SDL/)
- Livro SDL Game Development
  - [www.packtpub.com/product/sdl-game-development/9781849696821](http://www.packtpub.com/product/sdl-game-development/9781849696821)
- Open Source Tools for Game Development
  - [www.youtube.com/watch?v=r3wDn0AjrTk](http://www.youtube.com/watch?v=r3wDn0AjrTk)

# *1. SDL – Simple DirectMedia Layer*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## *2. O Básico*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## 2. O Básico

- Exemplo: Hello World!
- Compilação & Execução
- Sistema de Coordenadas
- Window & Renderer
- Estado do Renderer
- Exercícios





# Exemplo: Hello World!

01-sdl/01-hello.c

```
#include <SDL2/SDL.h>
int main (int argc, char* args[])
{
    /* INICIALIZACAO */
    SDL_Init(SDL_INIT_EVERYTHING);
    SDL_Window* win = SDL_CreateWindow("Hello World!",
                                       SDL_WINDOWPOS_UNDEFINED,
                                       SDL_WINDOWPOS_UNDEFINED,
                                       200, 100, SDL_WINDOW_SHOWN
                                       );
    SDL_Renderer* ren = SDL_CreateRenderer(win, -1, 0);

    /* EXECUCAO */
    SDL_SetRenderDrawColor(ren, 0xFF,0xFF,0xFF,0x00);
    SDL_RenderClear(ren);
    SDL_SetRenderDrawColor(ren, 0x00,0x00,0xFF,0x00);
    SDL_Rect r = { 40,20, 10,10 };
    SDL_RenderFillRect(ren, &r);
    SDL_RenderPresent(ren);
    SDL_Delay(5000);

    /* FINALIZACAO */
    SDL_DestroyRenderer(ren);
    SDL_DestroyWindow(win);
    SDL_Quit();
}
```



Hello World!

# Compilação & Execução

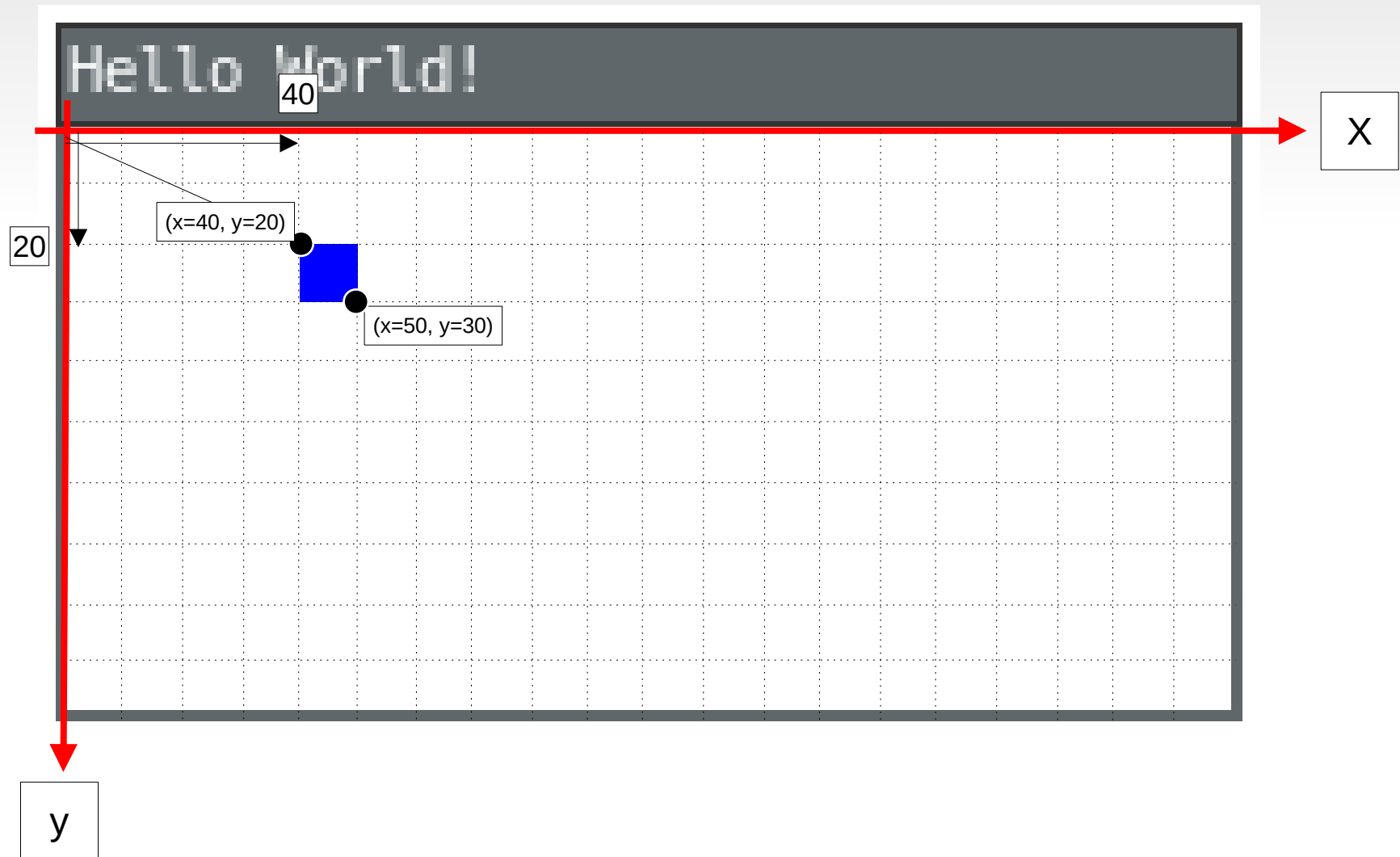
```
$ cd 01-Hello/  
$ gcc 01-hello.c -lSDL2 -o hello  
$ ./hello
```

Hello World!



# Sistema de Coordenadas

- `SDL_Rect r = { 40, 20, 10, 10 };`

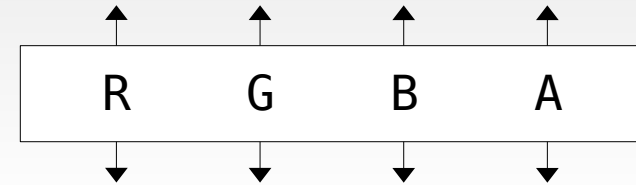


# Window & Renderer

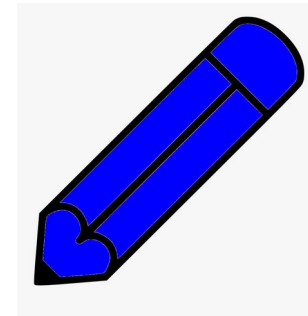
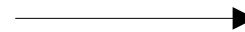
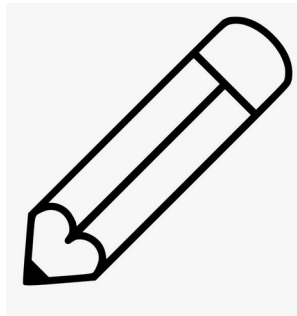
- Tipos “opacos”
  - Similar ao FILE
  - Manipulação através de funções
  - Alocação e desalocação
- SDL\_Window
  - Interação com o SO (posição, tamanho, fullscreen, etc)
  - [wiki.libsdl.org/SDL\\_Window](http://wiki.libsdl.org/SDL_Window)
- SDL\_Renderer
  - Operações gráficas (imagens, figuras geométricas, etc)
  - [stackoverflow.com/questions/21007329/what-is-an-sdl-renderer](http://stackoverflow.com/questions/21007329/what-is-an-sdl-renderer)

# Estado do Renderer

- Afeta as operações subsequentes:
  - `SDL_SetRenderDrawColor(ren, 0xFF, 0xFF, 0xFF, 0x00)`
    - `SDL_RenderClear(ren)`
    - ...
  - `SDL_SetRenderDrawColor(ren, 0x00, 0x00, 0xFF, 0x00)`
    - `SDL_RenderFillRect(ren, &r)`
    - ...

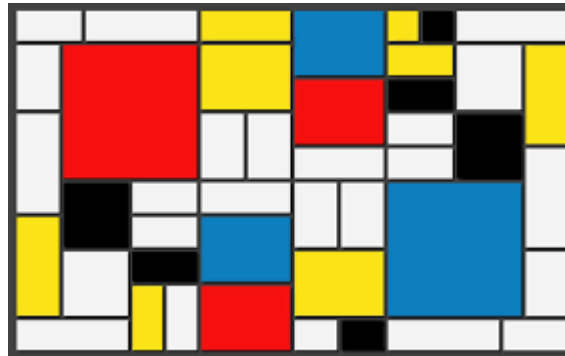


Hello World!



# Exercícios

1. Faça um desenho qualquer com linhas, pontos e retângulos, usando cores diversas.
2. Faça um desenho usando a biblioteca SDL2\_gfx, com figuras mais complexas.
  - [www.ferzkopp.net/Software/SDL2\\_gfx/Docs/html/](http://www.ferzkopp.net/Software/SDL2_gfx/Docs/html/)



## *2. O Básico*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## 3. Animação Simples

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

francisco@ime.uerj.br





# 3. Animação Simples

- Exemplo: Animação Simples
- Double Buffering
- Exercícios



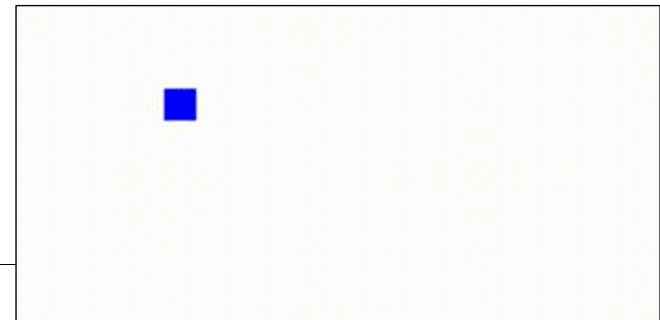
# Exemplo: Animação Simples

01-sdl/02-anim.c

```
#include <SDL2/SDL.h>
int main (int argc, char* args[])
{
    /* INICIALIZACAO */
    ...

    /* EXECUÇÃO */
    SDL_Rect r = { 40,20, 10,10 };
    while (r.x<100) {
        SDL_SetRenderDrawColor(ren, 0xFF,0xFF,0xFF,0x00);
        SDL_RenderClear(ren);
        SDL_SetRenderDrawColor(ren, 0x00,0x00,0xFF,0x00);
        SDL_RenderFillRect(ren, &r);
        SDL_RenderPresent(ren);
        SDL_Delay(500);
        r.x += 2;
        r.y += 2;
    }

    /* FINALIZACAO */
    ...
}
```

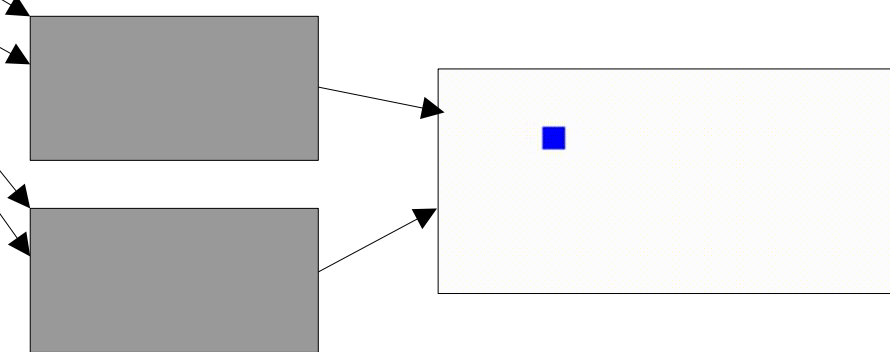


# Double Buffering

```
#include <SDL2/SDL.h>
int main (int argc, char* args[])
{
    /* INICIALIZACAO */
    ...

    /* EXECUÇÃO */
    SDL_Rect r = { 40,20, 10,10 };
    while (r.x<100) {
        SDL_SetRenderDrawColor(ren, 0xFF,0xFF,0xFF,0x00);
        SDL_RenderClear(ren);
        SDL_SetRenderDrawColor(ren, 0x00,0x00,0xFF,0x00);
        SDL_RenderFillRect(ren, &r);
        SDL_RenderPresent(ren);
        SDL_Delay(500);
        r.x += 2;
        r.y += 2;
    }

    /* FINALIZACAO */
    ...
}
```



# Exercícios

1. Faça uma animação em ciclo com algum padrão (ex, andar em círculo).
  - Use somente as funções vistas nos slides anteriores.
  - Use a velocidade de 10 pixels/segundo.
2. Explique o comportamento do programa `01-sdl/03-buffer.c`
  - Note que há duas chamadas a `SDL_RenderPresent`



## 3. Animação Simples

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

francisco@ime.uerj.br



## *4. Loop de Eventos*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

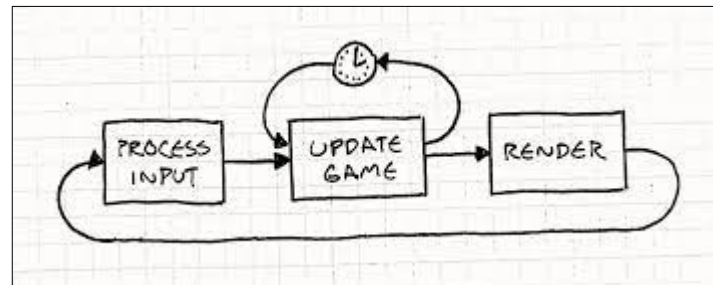
Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## 4. Loop de Eventos

- Exemplo: Movendo um Retângulo
- Tipos de Eventos
- Lendo Eventos
  - Teclado
  - Mouse
- Exercícios



# Exemplo: Movendo um Retângulo

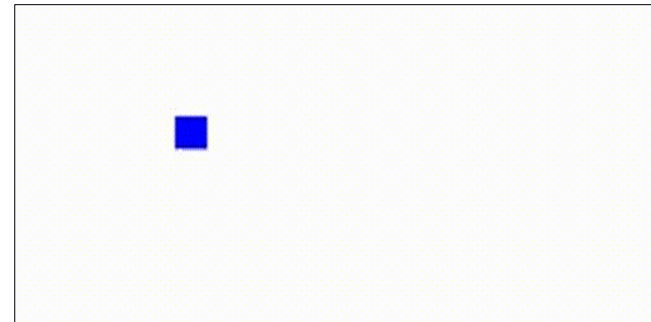
```
#include <SDL2/SDL.h>
int main (int argc, char* args[])
{
    /* INICIALIZACAO */

    /* EXECUÇÃO */
    SDL_Rect r = { 40,20, 10,10 };
    while (1) {
        /* REDESENHO */

        SDL_Event evt;
        SDL_WaitEvent(&evt);
        if (evt.type == SDL_KEYDOWN) {
            switch (evt.key.keysym.sym) {
                case SDLK_UP:
                    r.y -= 5;
                    break;
                case SDLK_DOWN:
                case SDLK_LEFT:
                case SDLK_RIGHT:
                    ...
            }
        }
    }

    /* FINALIZACAO */
}
```

01-sdl/04-keys.c





# Tipos de Eventos

- Teclado
  - `SDL_KEYDOWN`, `SDL_KEYUP`
- Mouse
  - `SDL_MOUSEMOTION`, `SDL_MOUSEBUTTONDOWN`, `SDL_MOUSEBUTTONUP`
- Janela
  - `SDL_WINEVENT` (minimizar, mover, redimensionar, etc)
- Saída
  - `SDL_QUIT`
- Joystick, Touch, Drag & Drop, Som, etc
- [wiki.libsdl.org/SDL\\_EventType](http://wiki.libsdl.org/SDL_EventType)

# Lendo Eventos

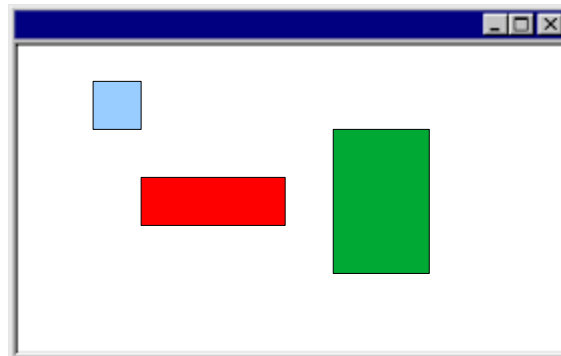
```
SDL_Event evt;
while (SDL_WaitEvent(&evt) {
    // usa evt.*
}
```

Uint32	<b>type</b>	the event type; <a href="#">SDL_MOUSEBUTTONDOWN</a> or <a href="#">SDL_MOUSEBUTTONUP</a>
Uint32	<b>timestamp</b>	timestamp of the event
Uint32	<b>windowID</b>	the window with mouse focus, if any
Uint32	<b>which</b>	the mouse instance id, or <a href="#">SDL_TOUCH_MOUSEID</a> ; see <a href="#">Remarks</a> for details
Uint8	<b>button</b>	the button that changed; see <a href="#">Remarks</a> for details
Uint8	<b>state</b>	the state of the button; <a href="#">SDL_PRESSED</a> or <a href="#">SDL_RELEASED</a>
Uint8	<b>clicks</b>	1 for single-click, 2 for double-click, etc. ( $\geq$ <a href="#">SDL 2.0.2</a> )

Sint32	<b>x</b>	X coordinate, relative to window	Uint32	<b>type</b>	the event type; <a href="#">SDL_KEYDOWN</a> or <a href="#">SDL_KEYUP</a>
Sint32	<b>y</b>	Y coordinate, relative to window	Uint32	<b>timestamp</b>	timestamp of the event
			Uint32	<b>windowID</b>	the window with keyboard focus, if any
			Uint8	<b>state</b>	the state of the key; <a href="#">SDL_PRESSED</a> or <a href="#">SDL_RELEASED</a>
			Uint8	<b>repeat</b>	non-zero if this is a key repeat
			<a href="#">SDL_Keysym</a>	<b>keysym</b>	the <a href="#">SDL_Keysym</a> representing the key that was pressed or released

# Exercícios

- Faça as seguintes alterações em `01-sdl/04-keys.c`
  1. Quando o usuário fechar a janela (ALT-F4 ou [X]), termine a aplicação.
  2. Quando o usuário clicar no mouse, adicione um novo retângulo fixo naquela posição (até 10 retângulos com cores diferentes).
  3. Mantenha nos limites da tela o retângulo que se move.



## *4. Loop de Eventos*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## *5. Tempo + Eventos*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

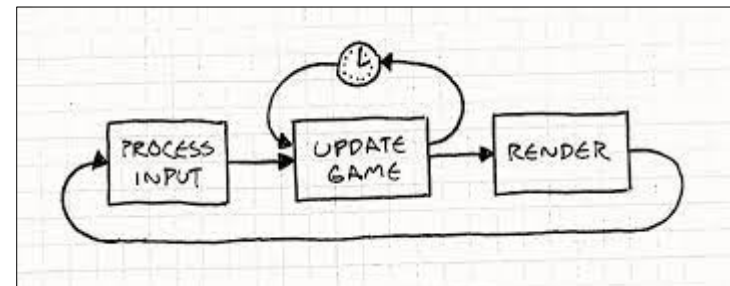
Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



# 5. Tempo + Eventos

- `SDL_WaitEventTimeout`
- Exemplo: Animando e Movendo um Retângulo
- Contando o Tempo
- Exemplo: Contando o Tempo
- Exercícios

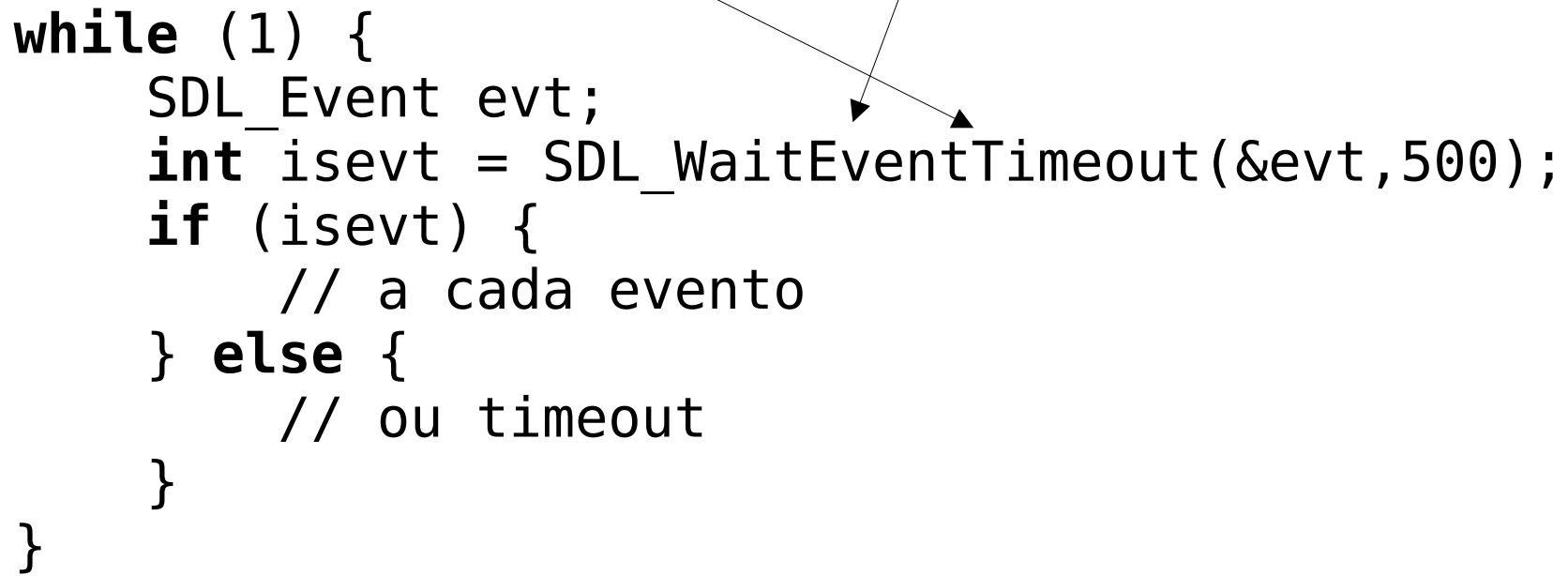


# SDL\_WaitEventTimeout

```
while (1) {  
    SDL_Delay(500);  
    // a cada 500ms  
}
```

```
while (1) {  
    SDL_Event evt;  
    SDL_WaitEvent(&evt);  
    // a cada evento  
}
```

```
while (1) {  
    SDL_Event evt;  
    int isevt = SDL_WaitEventTimeout(&evt, 500);  
    if (isevt) {  
        // a cada evento  
    } else {  
        // ou timeout  
    }  
}
```



# Exemplo: Animando e Movendo

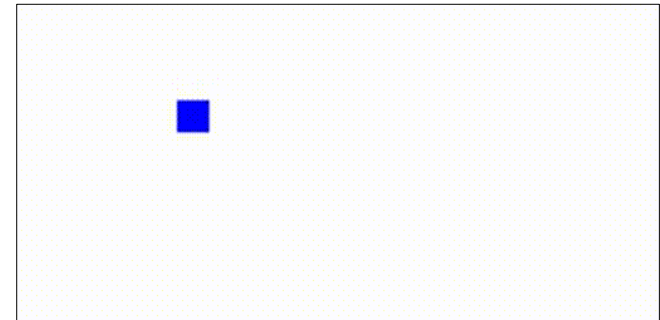
```
#include <SDL2/SDL.h>
int main (int argc, char* args[])
{
    /* INICIALIZACAO */

    /* EXECUÇÃO */
    SDL_Rect r = { 40,20, 10,10 };
    while (1) {
        /* REDESENHO */

        SDL_Event evt;
        int isevt = SDL_WaitEventTimeout(&evt, 500);
        if (isevt) { // evento ocorrido
            if (evt.type == SDL_KEYDOWN) {
                ...
            }
        } else { // deu timeout
            r.x += 2;
            r.y += 2;
        }
    }

    /* FINALIZACAO */
}
```

01-sdl/05-timeout.c





# Contando o Tempo

- Temporizador reinicia a cada evento
- `SDL_GetTicks` para contar o tempo

```
Uint32 antes = SDL_GetTicks();  
SDL_WaitEventTimeout(&evt, 500);  
Uint32 dt = SDL_GetTicks() - antes;
```

subtrai

## SDL\_GetTicks

Use this function to get the number of milliseconds since the SDL library initialization.

**Contents**  
[SDL\\_GetTicks](#)  
[Syntax](#)  
[Return Value](#)  
[Code Examples](#)  
[Remarks](#)  
[Related Functions](#)

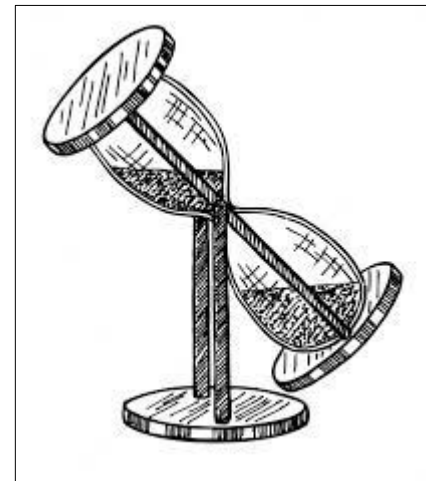
### Syntax

[Toggle line numbers](#)

```
Uint32 SDL_GetTicks(void)
```

### Return Value

Returns an unsigned 32-bit value representing the number of milliseconds since the SDL library initialized.



# Exemplo: Contando o Tempo

01-sdl/06-getticks.c

```
/* EXECUÇÃO */
SDL_Rect r = { 40,20, 10,10 };
int espera = 500;
while (1) {
    /* REDESENHO */

    SDL_Event evt;
    Uint32 antes = SDL_GetTicks();
    int isevt = SDL_WaitEventTimeout(&evt, espera);
    if (isevt) { // evento ocorrido
        espera -= (SDL_GetTicks() - antes);
        if (evt.type == SDL_KEYDOWN) {
            ...
        }
    } else { // deu timeout
        espera = 500;
        r.x += 2;
        r.y += 2;
    }
}
```



# Exercícios

## 1. Crie uma aplicação com três retângulos:

- O 1º só se move pelo tempo.
- O 2º só se move pelo teclado.
- O 3º acompanha a posição do mouse.

## 2. Crie uma função auxiliar para contagem de tempo:

- `int AUX_WaitEventTimeoutCount (SDL_Event* evt, UInt32* ms)`
- A função deve se comportar exatamente como `SDL_WaitEventTimeout`, exceto por subtrair de `ms` o tempo passado a cada chamada.
- [wiki.libsdl.org/SDL\\_WaitEventTimeout](http://wiki.libsdl.org/SDL_WaitEventTimeout)
- Use a nova função no exercício 1 (e sempre que possível).

## 5. *Tempo + Eventos*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

francisco@ime.uerj.br



## 6. Colisões

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

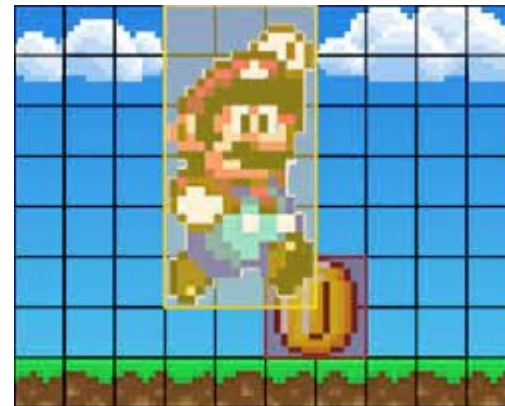
Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## 6. Colisões

- Interseção entre dois objetos
- Retângulo vs Mouse
- Retângulo vs Retângulo
- Exemplo: Colisão entre objetos
- Exercícios



# Interseção entre dois objetos

- Retângulo vs Mouse

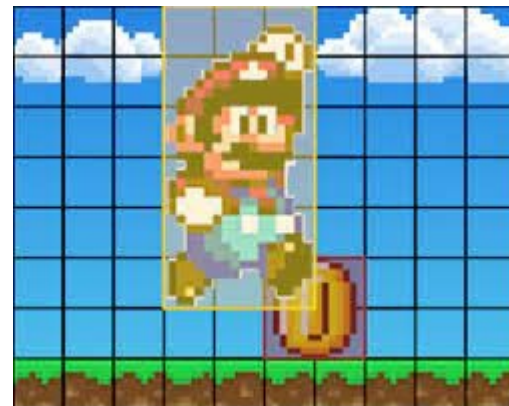
`bool SDL_PointInRect (SDL_Point* p, SDL_Rect* r)`

- Retângulo vs Retângulo

`bool SDL_HasIntersection (SDL_Rect* r1, SDL_Rect* r2)`

- Retângulo vs Linha

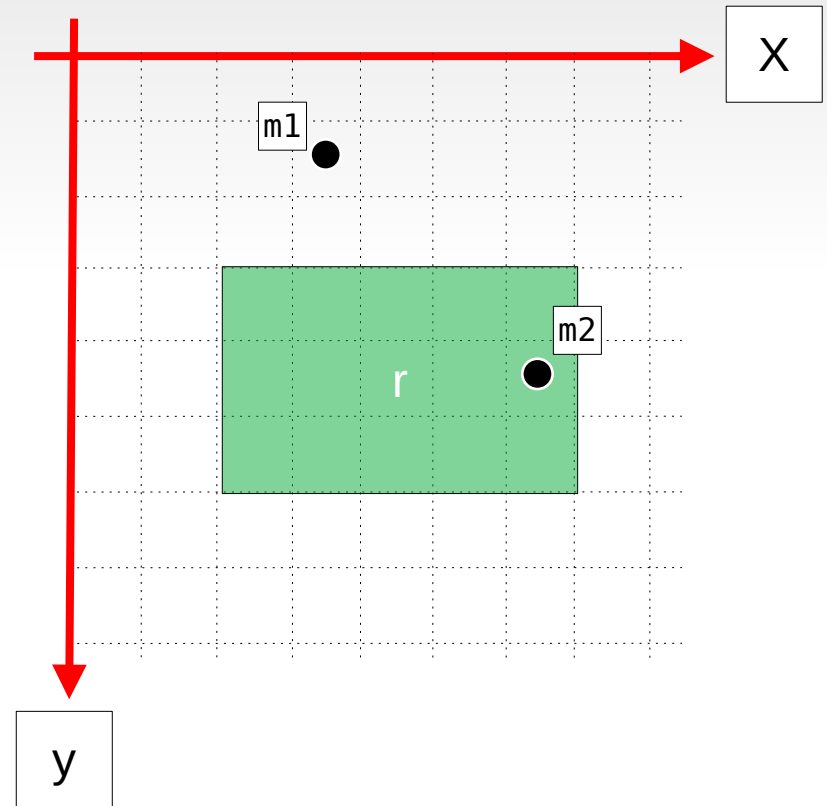
`bool SDL_IntersectRectAndLine(SDL_Rect* r, ...)`



# Retângulo vs Mouse

- `SDL_Rect r = { 20,30, 50,30 };`
- `SDL_Point m1 = { 35,15 };`  
`SDL_Point m2 = { 65,45 };`

```
int fora1 = ( m1.x < r.x ||  
             m1.y < r.y ||  
             m1.x > r.x+r.w ||  
             m1.y > r.y+r.h );  
int fora2 = ( m2.x < r.x ||  
             m2.y < r.y ||  
             m2.x > r.x+r.w ||  
             m2.y > r.y+r.h );  
  
int PointInRect (SDL_Point* p, SDL_Rect* r) {  
    return !( p->x < r->x ||  
             p->y < r->y ||  
             p->x > r->x+r->w ||  
             p->y > r->y+r->h );  
}  
  
int dentro1 = PointInRect(&m1, &r);  
int dentro2 = PointInRect(&m2, &r);
```

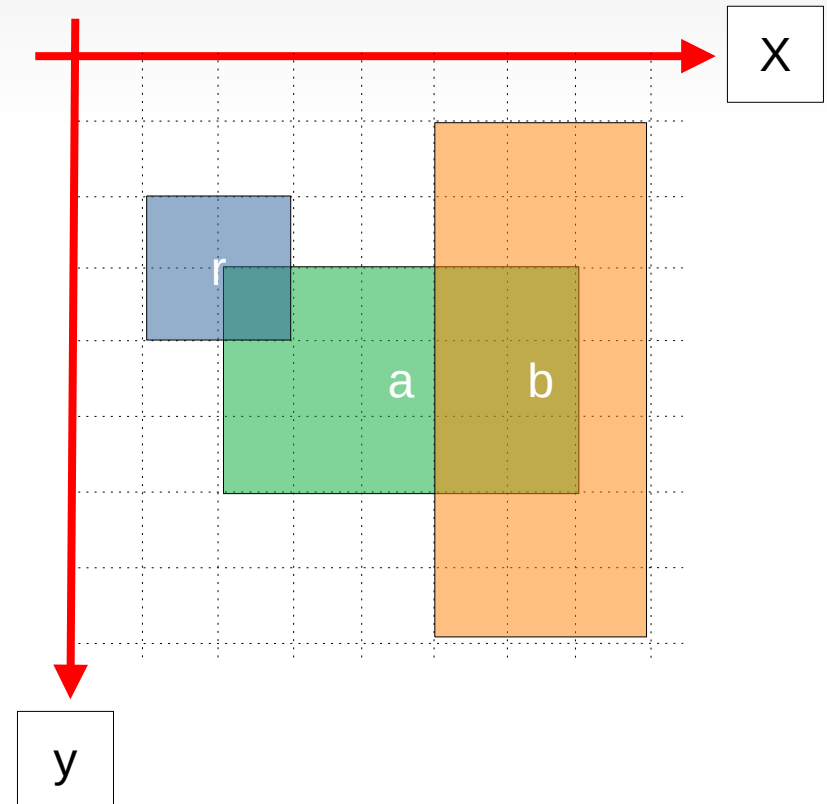




# Retângulo vs Retângulo

- `SDL_Rect r = { 10,20, 20,20 };`
- `SDL_Rect a = { 20,30, 50,30 };`
- `SDL_Rect b = { 50,10, 30,70 };`

```
int foraA = ( a.x+a.w < r.x ||  
              a.y+a.h < r.y ||  
              a.x > r.x+r.w ||  
              a.y > r.y+r.h );  
int foraB = ( b.x+b.w < r.x ||  
              b.y+b.h < r.y ||  
              b.x > r.x+r.w ||  
              b.y > r.y+r.h );  
  
int RectInRect (SDL_Rect* r1, SDL_Rect* r2) {  
    return !( p->x+p->w < r->x ||  
              p->y+p->h < r->y ||  
              p->x > r->x+r->w ||  
              p->y > r->y+r->h );  
}  
  
int dentroA = RectInRect(&a, &r);  
int dentroB = RectInRect(&b, &r);
```



# Exemplo: Colisão entre Objetos

01-sdl/07-colisao.c

```
SDL_Rect r1 = { 40,30, 10,10 };
SDL_Rect r2 = { 140,30, 10,10 };
int go1 = 1;
int go2 = 1;
while (1) {
    SDL_Event evt;
    int isevt = SDL_WaitEventTimeout(&evt, ...);
    if (isevt) { // evento ocorrido
        if (evt.type == SDL_MOUSEBUTTONDOWN) {
            SDL_Point m = { evt.button.x, evt.button.y };
            if (SDL_PointInRect(&m, &r1)) go1 = !go1;
            if (SDL_PointInRect(&m, &r2)) go2 = !go2;
        }
    } else { // deu timeout
        if (go1) r1.x += 2;
        if (go2) r2.x -= 2;
        if (SDL_HasIntersection(&r1, &r2)) {
            go1 = go2 = 0;
        }
    }
}
```



# Exercícios

- Altere o exercício 5.1 da seguinte forma:
  1. Faça os objetos se moverem somente no eixo X
  2. Desenhe uma linha de chegada
  3. Pare os objetos que atingirem a linha de chegada
  4. Aguarde todos atingirem a linha de chegada
  5. Anuncie o objeto que chegou primeiro
  6. Reinicie a aplicação

## 6. Colisões

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



## 7. *Imagens e Texturas*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

francisco@ime.uerj.br



# 7. *Imagens e Texturas*

- Imagens
- Fontes e Textos
- Recortes (*Clipping*)
- Escalonamento (*Scaling*)
- Transparência
- Exercícios

## 7. *Imagens e Texturas*

Programação de Jogos

<https://github.com/fsantanna-uerj/Jogos/>

Francisco Sant'Anna

francisco@ime.uerj.br



# Projeto 1/3

- Drag and Drop
- Imagem
- Colisão
- Animação



# O “loop” de eventos

```
/* INITIALIZATION */  
  
/* EXECUTION */  
while (1) {  
    /* events */  
    /* logic */  
    /* redraw */  
}  
  
/* FINALIZATION */
```

- Modelo síncrono

# Exercício - “Animação”

- Mover dois retângulos “em quadrados” com velocidades diferentes
- Parar o retângulo quando clicado com o mouse

## SDL\_GetTicks

Use this function to get the number of milliseconds since the SDL library initialization.

### Contents

- [SDL\\_GetTicks](#)
- [Syntax](#)
- [Return Value](#)
- [Code Examples](#)
- [Remarks](#)
- [Related Functions](#)

## Syntax

[Toggle line numbers](#)

```
Uint32 SDL_GetTicks(void)
```

## Return Value

Returns an unsigned 32-bit value representing the number of milliseconds since the SDL library initialized.

```
switch (e.type) {  
    <...>  
    SDL_MOUSEBUTTONDOWN:  
        SDL_MouseButtonEvent* me =  
            (SDL_MouseButtonEvent*) &e;  
        me->x;  
        me->y;  
        <...>  
        break;  
}
```

# Animações

```
int speed = 10;

while (1) {
    SDL_PollEvent(&e);

    // eventos
    switch (e.type) {
        <...>
    }

    // animacoes
    x += speed;

    // redesenho
    <...>
}
```

```
int speed = 10;

while (1) {
    SDL_PollEvent(&e);

    // eventos
    switch (e.type) {
        <...>
    }

    // animacoes
    x += DT*speed;
        // DT: diferenca de tempo
        // entre dois frames (SDL_Ticks)

    // redesenho
    <...>
}
```



# Compilando / Executando

```
# clone
> git clone https://github.com/fsantanna-uerj/reativos

# update
> cd reativos/
> git pull

# compile
> cd reativos/code/sdl
> gcc 00_hello.c -lSDL2 -o 00_hello # (01_input.c 01_input)

# run
> ./00_hello # (./01_input)
```

# Documentação

- Lazy Foo Tutorial
  - <http://lazyfoo.net/tutorials/SDL/index.php>
- Manual de referência
  - <https://wiki.libsdl.org/APIByCategory>
- Fórum do SDL
  - <http://forums.libsdl.org/>



# SDL + Céu

- Hello World! (00\_hello.ceu)
- Input (01\_input.ceu)
- SDL (02\_sdl.ceu)
- Animações (ex\_01.ceu)
- Animações + `code/await` (ex\_01\_code.ceu)



# game.ceu

- sprites
- salto (vy)
- aceleração (ax,ay)
- **objetos dinâmicos**
- **colisão**
- ???

# SDL + Céu

- Compilar e executar:

```
$ cd /opt/reativos/ceu-sdl/  
$ make CEU_SRC=../reativos/code/sdl/00_hello.ceu  
$ make CEU_SRC=samples/sdl-01.ceu  
$ cd ceu-sdl-birds/  
$ make
```