Guia prático de desenvolvimento com C & SDL no Windows.

Por Introscopia, João Antonio Ferreira
São José dos Campos, Junho de 2023

Você começou a aprender a programar, mas parece que tudo que da pra fazer é printar e scanear texto no terminal... Não te culpo se isso te fez concluir que programação é chato. Quem usa programas de terminal hoje em dia?! Eu vim te falar que programar jogos ou outros softwares gráficos não está tão além do seu alcance.

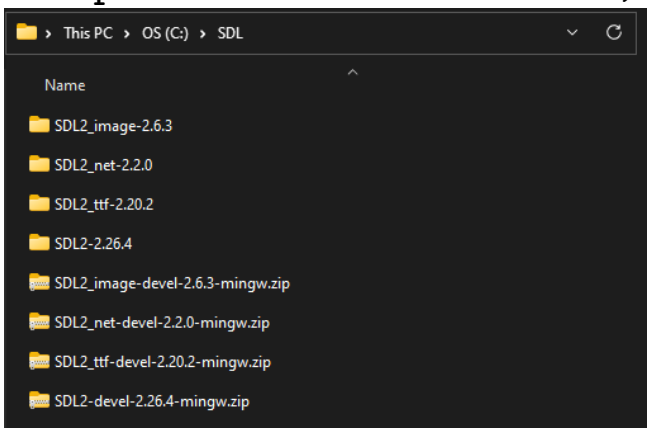
O objetivo deste guia é familiarizar o prospectivo programador com as técnicas e ferramentas para desenvolvimento de programas gráficos com a linguagem C e o framework SDL. As instruções terão em vista apenas o sistema operacional Windows, mas as ferramentas são compatíveis com todos os OSs mais populares, se você conseguir instalá-las na sua maquina, o código será o mesmo.

Todos os arquivos relevantes nesse guia se encontram no repositório:

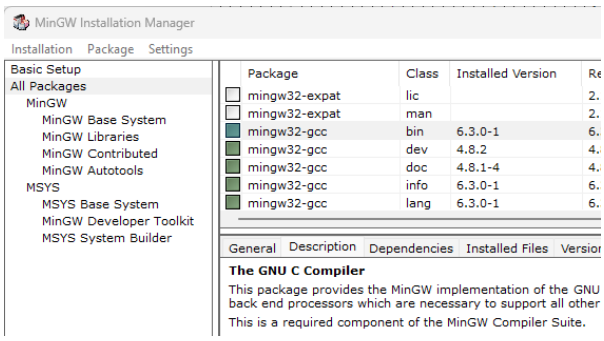
https://github.com/Introscopia/Getting_Started_with_C_and_SDL/

Você pode baixar cada arquivo por lá, mas a maneira mais facil é baixar tudo de uma vez dando um clone com o Github Desktop.

Para começar você deve seguir as instruções na página SETUP. Lá eu explico como instalar o MinGW, nosso compilador, e o SDL.



A pasta do SDL com todos os módulos extraídos e o instalador do MinGW:



Se você instalou o MinGW corretamente, e adicionou o endereço dele à variável de ambiente Path, você pode confirmar executando o comando gcc -v no terminal (cmd ou powershell):

```
C:\Users\Introscopia>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=c:/mingw/bin/../libexec/gcc/mingw32/6.3.0/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-6.3.0/configure --build=x86_64-pc-linux-gnu --host=m
```

Se o resultado for algo parecido com isso, tá tudo pronto. No repositório eu também incluí um programinha simples com o comando de compilação direto em um .bat, na pasta Compilation Test, no SDL.

Depois, você pode dar uma lida na página WORKFLOW para entender melhor como tudo isso funciona junto.

E então, finalmente, o seu primeiro programa gráfico e interativo com C & SDL: Um belo clone de "Pong". Abra a sua cópia do arquivo main.c da pasta Pong++. Você pode usar qualquer editor que quiser, eu recomendo SublimeText. (A outra pasta, Pong in 90 lines of code é bem parecida, mas utiliza algumas outras dependências e está comentada em Inglês.)

Eu recomendo você fazer uma copia da pasta, apagar os conteúdos do seu main.c, e ir montando o programa junto comigo a medida que vou explicando cada parte, assim você pode ir compilando e vendo o progresso em cada etapa! Também é uma ótima oportunidade pra fazer experimentos. Não tenha medo de cutucar o código! Pior que pode acontecer é um segfault...

As primeiras coisas que você vai ver lá são alguns `#include`s das bibliotecas padrão. Depois tem várias definições preliminares:

```
23 typedef int32_t bool;
25 #define QUARTER_PI  (double) 0.78539816339744830961566084
27 int random( int min, int max ){
31 double randomD( double min, double max ){
35 void SDL_framerateDelay( int frame_period ){
45 int constrain( int a, int min, int max ){
51 const SDL_Color white = (SDL_Color){ 255,255,255,255 };
52 const SDL_Color black = (SDL_Color){ 0, 0, 0, 255 };
54 void render_num_text( SDL_Renderer *R, int num, ...
```

Pode copiar todas do seu `main.c` original. Uma boa dica: No Sublime, você pode 'fechar' ou 'dobrar' blocos de código clicando nas setinhas que aparecem quando você passa o mouse na parte esquerda da janela. (é isso que estou mostrando aqui em cima, as declarações 'dobradas'. Não cabe no escopo desse tutorial explicar o conteúdo de cada uma!)

```
74 int main(int argc, char *argv[]){
```

`Main()`. O ponto de entrada do programa. À partir daqui nós começamos a montar o nosso programa propriamente dito.

```
95 if (SDL_CreateWindowAndRenderer( width, height, 0,
                                    &window, &renderer)) {
```

Nesta linha criamos a nossa janelinha. Pronto! Já estamos livres do terminal. Eu disse que não seria tão difícil. Não estranhe que estamos chamando essa função de dentro de um `if()`, é só para detectarmos se houver algum erro.

Agora eu te convido a pular mais uma vez até aqui:

```
133     // informacoes das raquetes
134     int H = Z * 88;//altura
```

Aqui nós começamos a definir as posições e dimensões dos objetos do jogo: As raquetes e a bola. A unidade de medida é o pixel. A nossa tela é como um plano cartesiano, com origem no canto superior esquerdo, (O eixo y orientado de cima pra baixo. O sistema de coordenadas no computador é "como se lê": esquerda pra direita, cima pra baixo). No caso das velocidades, vbx, vby e pvel, da bola em 2d e das raquetes, respectivamente, a unidade seria "pixels por quadro".

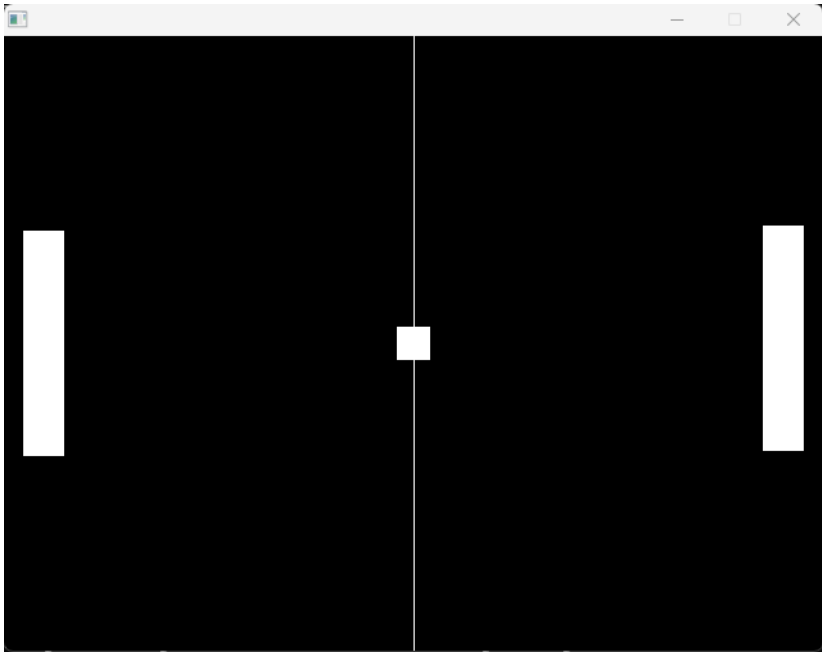
```
166     //Inicio do loop de animacao
167     while ( loop ) {
```

E aqui começa o grande laço de animação. Vídeo capturado com uma câmera é composto de um sequencia de imagens estáticas, que, quando exibidas rapidamente, algo entre 24 e 60 quadros por segundo, criam a ilusão de movimento. No computador não é diferente. O código dentro das chaves desse while() descreve o procedimento para criação de um quadro, e essas instruções se repetem 60 vezes por segundo. Descendo um pouco mais chegamos na parte dos gráficos:

```
268     //Limpar o Quadro
269     SDL_SetRenderDrawColor( renderer, 0, 0, 0, 255);
270     SDL_RenderClear( renderer );
271
272     //DESENHAR:
273
274     //Linha de Centro
275     SDL_SetRenderDrawColor( renderer, 255, 255, 255, 255);
276     SDL_RenderDrawLine( renderer, cx, 0, cx, height );
281
282     //raquetes
283     SDL_RenderFillRect( renderer, &P1_rect );
284     SDL_RenderFillRect( renderer, &P2_rect );
```

```
286     //bola
287     SDL_RenderFillRect( renderer, &(SDL_Rect){bx-br,...
296
297     // Apresentar o quadro para o usuário
298     SDL_RenderPresent(renderer);
300     SDL_framerateDelay( 16 );
301}
```

Essa é uma versão simplificada, relativa a que você vai encontrar no arquivo, mas aqui já temos todos os componentes principais do jogo: Linhas e retângulos são desenhados com funções SDL_Render, as cores são definidas chamando SDL_SetRenderDrawColor(), e passando valores RGBA (Red, Green, Blue, e Alpha, que é a transparência). Com tudo desenhado já podemos chamar SDL_RenderPresent() para jogar o quadro na tela. A ultima linha, SDL_framerateDelay(), é uma função que eu escrevi para mander a taxa de exibição dos quadros (vulgo "frame rate") constante a 60 fps, ou seja, quadros com duração de 16 milissegundos.



Ai está, nosso belo jogo. Mas até ai os 60 quadros por segundo não adiantam muita coisa, por que nada se move...

```
220         bx += vbx;
221         by += vby;
```

Aqui nós somamos a velocidade da bola à sua posição, em cada eixo, x e y, efetivamente fazendo a bola dar um passinho. Se o passo for suficientemente pequeno, e a taxa de quadros por segundo suficientemente alta, isso cria a ilusão de um movimento contínuo e fluido. E quanto às raquetes? Bom, elas tem que se mover de acordo com os controles dos jogadores, então temos que entrar no tópico da interatividade. Talvez você tenha percebido que logo depois do `while()` inicial do laço de animação...

```
169         SDL_Event event;
170         while( SDL_PollEvent(&event) ){
```

Encontra-se outro laço `while()`. É aqui que nós vamos lidar com os "eventos" como teclas sendo apertadas, mouse sendo movido ou clicado, etc. Isso tem que acontecer dentro de um laço devido à possibilidade de haver mais de um evento no mesmo frame. Usando as variáveis `p1u`, `p1d`, `p2u`, `p2d`, "up" e "down" para cada player, nós podemos manter um registro da situação de cada tecla. Só precisamos ativar a variável quando a tecla é pressionada, e desativar quando a tecla é solta:

```
172     case SDL_KEYDOWN://tecla apertada
173             if( event.key.keysym.sym == 'w' )       p1u = 1;
174         else if( event.key.keysym.sym == 's' )      p1d = 1;
175         else if( event.key.keysym.sym == SDLK_UP )  p2u = 1;
176         else if( event.key.keysym.sym == SDLK_DOWN ) p2d = 1;
177         break;
178     case SDL_KEYUP://tecla solta
179             if( event.key.keysym.sym == 'w' )       p1u = 0;
180         else if( event.key.keysym.sym == 's' )      p1d = 0;
181         else if( event.key.keysym.sym == SDLK_UP )  p2u = 0;
182         else if( event.key.keysym.sym == SDLK_DOWN ) p2d = 0;
183         break;
```

Dentro do `switch (event.type) {` nós separamos cada tipo de evento. Aquela expressão verbosa `event.key.keysym.sym` é o campo do objeto `event` que contem a tecla. O player 1 joga com W e S, o player dois joga com as setinhas.

Abaixo, no evento de movimentação do mouse, eu também estou registrando a posição do cursor:

```
185     case SDL_MOUSEMOTION:
187         mouseX = event.motion.x;
188         mouseY = event.motion.y;
190         break;
```

Nós podemos usar isso pra deixar um dos jogadores controlar sua raquete com o mouse.

```
203     if( mouseY < P2_rect.y + hH ) p2u = 1;
204     else p2u = 0;
205     if( mouseY > P2_rect.y + hH ) p2d = 1;
206     else p2d = 0;
```

Este código já é suficiente pra traduzir o movimento do mouse para os controles *up* e *down* do player 2, mas não fica perfeito. Fica como exercício para o leitor experimentar e aprimorar esse procedimento. No arquivo você vai encontrar a versão já aprimorada, se não quiser spolier, cuidado...

Agora podemos efetuar os controles das raquetes, assim como fizemos com a bola:

```
214  if( p1u ) P1_rect.y = constrain( P1_rect.y - pvel, 0, height - H );
215  if( p1d ) P1_rect.y = constrain( P1_rect.y + pvel, 0, height - H );
216  if( p2u ) P2_rect.y = constrain( P2_rect.y - pvel, 0, height - H );
217  if( p2d ) P2_rect.y = constrain( P2_rect.y + pvel, 0, height - H );
```

Usando uma função `constrain` nós já prevenimos a raquete de

escapar da janela. O mesmo não é verdade da bola. Se você montou sua copia do programa até esse ponto, já percebeu que a bola se move, mas rapidamente foge da janela. Mas é claro, você provavelmente já viu lá no código as colisões:

```
223         if( by < br ){// colisao com o teto
224           vby *= -1;
225           by = br;//restituir a posicao evita bugs!
226         }
227         if( by > height-br ){// colisao com o chao
228           vby *= -1;
229           by = height-br;
230         }
```

Realizamos a detecção comparando a coordenada y da posição com as extremidades da janela: `0` e `height`, tirando `br`, o raio da bola, para alinhar corretamente a colisão. Se essas comparações revelam que a bolar esta fora da janela, nós invertemos a sua velocidade em y, multiplicando `vby` por `-1`. Também realizamos uma restituição da posição; Exercício para o leitor: apague as restituições e tente descobrir se elas são mesmo necessária, e para quê.

As colisões com as paredes verticais são semelhantes no que se trata da detecção, mas nestes casos nós não queremos refletir a bola, e sim marcar um ponto para o jogador do lado oposto, e começar a próxima rodada com a bola de volta ao centro.

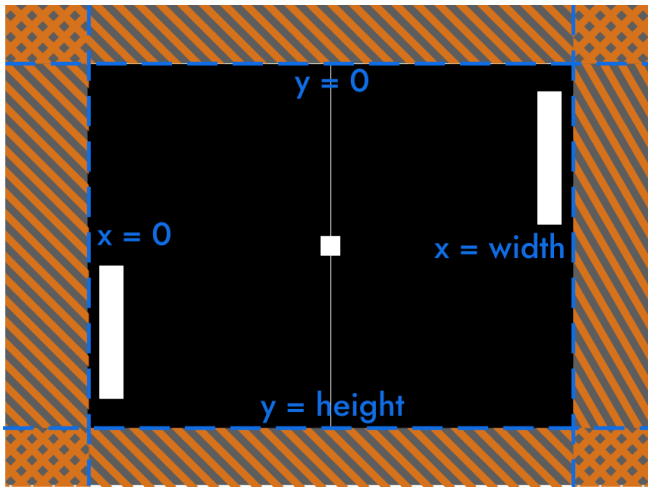
```
//colisoes com as paredes verticais, os "gols"
243         bool reset_ball = 0;
244         if( bx < 0 ){
245           ++p2s;
248           reset_ball = 1;
249         }
250         if( bx > width ){
251           ++p1s;
254           reset_ball = 1;
255         }
```

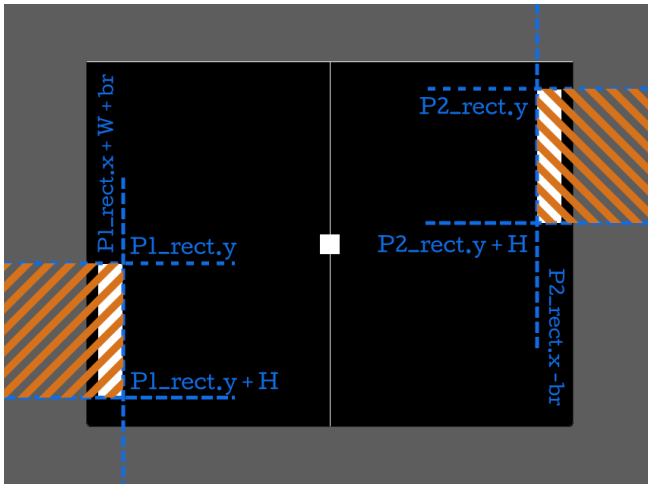
A variável reset_ball registra se houve um gol, e, já que é o mesmo procedimento nos dois casos, nós lidamos com isso abaixo, para evitar duplicação de código.

Finalmente temos que tratar das colisões com as raquetes. Este é provavelmente a parte mais complexa do programa. Nós não só temos que detectar a sobreposição no eixo x, mas em uma faixa no eixo y. São 3 condições que tem que ser verdadeiras ao mesmo tempo:

```
233 if( bx < P1_rect.x + W + br && by > P1_rect.y && by < P1_rect.y + H ){
        ...
237 else if( bx > P2_rect.x -br && by > P2_rect.y && by < P2_rect.y + H ){
```

Note o uso do operador booleano &&, “AND” para combinar as 3 condições em cada checagem.





E agora, funcionalmente, o jogo já está pronto! Se você chegou até aqui, parabéns! Aproveite para pra cutucar os valores, as cores, etc. Personalize o seu Pong à vontade.

No restante do tutorial vamos fazer duas melhorias que vão cobrir dois tópicos importantes: Renderização de texto, e de imagens.

A única coisa que está faltando no nosso pong relativo ao original é o placar. Para mostrar um placar, precisamos conseguir renderizar texto. É pra isso que a gente baixou o SDL_TTF, o

módulo de texto do SDL. Vou dar uma revisada em como incluir o TTF no seu projeto, que vai servir para o SDL_Image também, é o mesmo processo.

Na makefile temos que ter o endereço da lib no seu computador, se você seguiu o meu guia direitinho isso será:

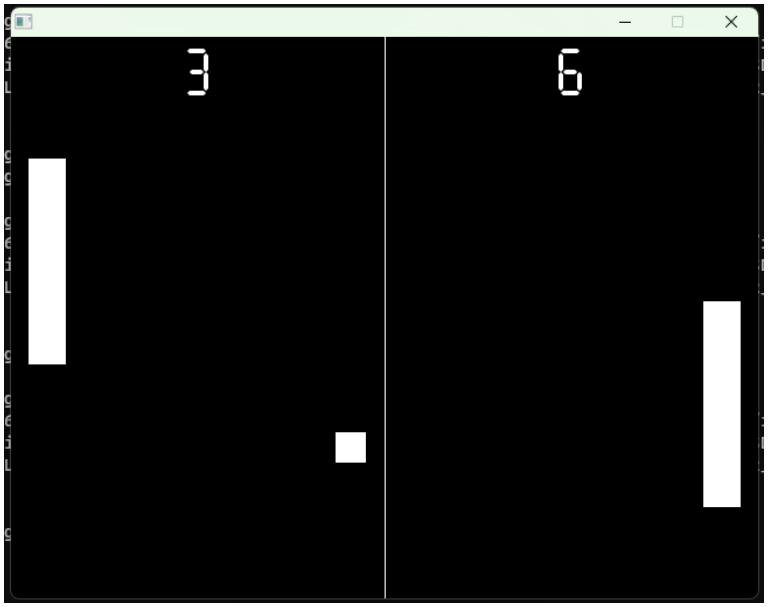
```
INCLUDE_PATHS += -IC:/SDL/SDL2_ttf-2.20.2/i686-w64-mingw32/include/SDL2
LIBRARY_PATHS += -LC:/SDL/SDL2_ttf-2.20.2/i686-w64-mingw32/lib
```

Isso já está feito nos arquivos que eu preparei, só estou frisando aqui porque talvez quando você for baixar a versão já vai ser outra, e portanto o endereço vai ser diferente. O linker flag `-lSDL2_ttf` também tem que aparecer no comando de compilação. Depois você precisa ter uma copia do arquivo `SDL2_ttf.dll` na pasta do projeto. Ele vem na instalação do módulo, fica na pasta `C:\SDL\SDL2_ttf-2.20.2\i686-w64-mingw32\bin`.

Com tudo isso pronto, no código você já pode

```
// Incluir a lib:
10 #include <SDL_ttf.h>
// Inicializar a lib:
109 if( TTF_Init() < 0 ) puts("TTF nao conseguiu inicializar");
// E carregar o arquivo fonte
111 TTF_Font *font = TTF_OpenFont( "7segments.ttf", 40 );
```

O TTF renderiza texto pra gente em superfícies, `SDL_Surfaces`, que são imagens que pertencem à CPU. A gente poderia jogar o texto direto na tela, usando uma das funções `SDL_Blit` na superfície da janela, mas nesse caso eu escrevi uma funçãozinha `render_num_text()` que já converte a imagem do texto em uma textura, `SDL_Texture`, que são imagens que pertencem à GPU, e portanto oferecem uma performance bem maior. Toda vez que o placar muda a gente atualiza as texturas do placar, `p1s_texture` e `p2s_texture`, e depois jogamos na tela com `SDL_RenderCopy()`.



Chegando nesse ponto, e percebendo que o texto que a gente mostra na tela, no fim das contas também é imagem, seria natural se perguntar, “Será que não poderíamos trocar os nossos gráficos chatos de geometria crua por umas imagens mais legais?” É exatamente o que a gente vai fazer com o SDL_Image! Eu fui na internet e achei uns sprites que eu gostei bastante de um artista chamado Surt. Eles tem licença CC0, então tomei a liberdade de criar um pequeno spritesheet com eles e incluir aqui neste projeto. Sinta-se livre para criar ou escolher seus próprios gráficos!

Uma vez que você se certificou de que o módulo SDL_Image está incluido corretamente, o DLL, os endereços na makefile, tudo, podemos já ir..

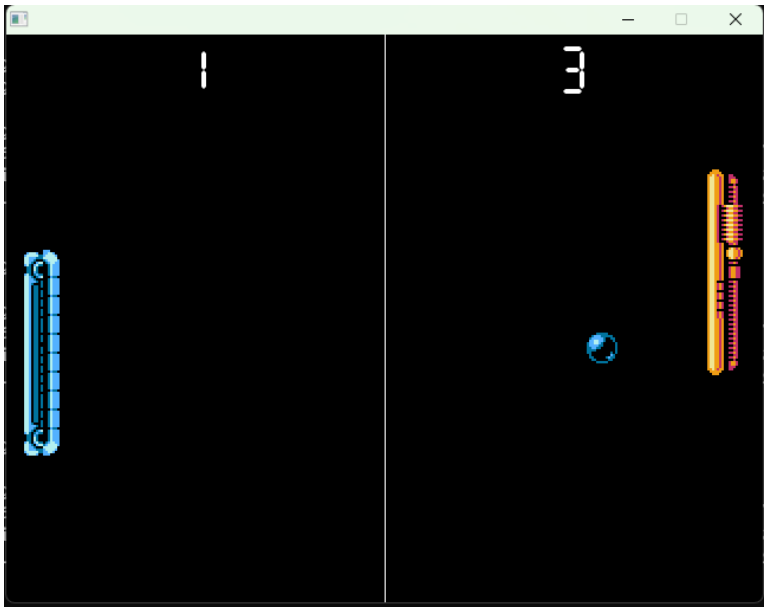
```
 96 //inicializando a biblioteca de imagem
 97 IMG_Init( IMG_INIT_PNG );
 98 //carregar a imagem:
 99 SDL_Texture *sprites = IMG_LoadTexture( renderer, "sprites.png");
100 SDL_Rect P1_paddle_src = (SDL_Rect){0, 0, 16, 88};
101 SDL_Rect P2_paddle_src = (SDL_Rect){48, 0, 16, 88};
102 SDL_Rect ball1_src = (SDL_Rect){17, 1, 13, 13};
```

Com a textura sprites carregada, nós definimos os “retângulos

fonte”, ou src rects, de cada objeto dentro da spritesheet. Pra casos simples assim, eu faço isso à mão mesmo, olhando as coordenadas do cursor no meu software de manipulação de imagem.

```
297 //Graficos com sprites
298 //raquetes
299 SDL_RenderCopy( renderer, sprites, &P1_paddle_src, &P1_rect );
300 SDL_RenderCopy( renderer, sprites, &P2_paddle_src, &P2_rect );
301 //bola
302 SDL_RenderCopyF( renderer, sprites, &ball1_src, &(SDL_FRect)...
```

E então já podemos trocar os RenderDraws por RenderCopys, que copiam partes da nossa spritesheet para a tela. É aqui que usamos os src rects, para especificar qual parte da imagem estamos copiando.



E ai está. Um belo joguinho possível de se programar do zero em uma tarde. Eu espero que esse tutorial tenha sido útil e que te inspire a criar coisas interessantes com código. Com esses fundamentos, o céu é o limite. Se você tiver qualquer dúvida sobre esse material, ou se estiver no dirigindo algum projeto educacional de tecnologias e artes, entre em contato comigo no e-mail: introscopia@protonmail.com. Um abraço!