documentação - problema 3 - breakout game

1.0

Gerado por Doxygen 1.10.0

Capítulo 1

Índice das Estruturas de Dados

1.1 Estruturas de Dados

Aqui estão as estruturas de dados, uniões e suas respectivas descrições:

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Capítulo 3

Estruturas

3.1 Referência da Estrutura Bloco

Estrutura para permitir criar os blocos que serão destruidos.

Campos de Dados

- int x
- int y
- int largura
- int altura
- int destruido
- short cor

3.1.1 Descrição detalhada

Estrutura para permitir criar os blocos que serão destruidos.
Estrutura de dado para salvar as informações a respeito dos blocos desenhandos na tela, que serão destruidos ao decorrer do jogo. A estrutura contem:

Parâmetros

X	valor para indicar o inicio de sua localização no eixo X
Y	valor para indicar o inicio de sua localização no eixo Y
largura	valor para indicar a largura que o bloco deve ter
altura	valor para indicar a altura que o bloco deve ter
destruido	inteiro que guarda a informação se o bloco ja foi destruido ou não, sendo 0 para nao destruido e 1 para destruido
cor	valor para indicar qual a cor que esse bloco dever ter ao ser renderizado

---

3.1.2 Campos

3.1.2.1 altura

```
int altura
```

3.1.2.2 cor

```
short cor
```

3.1.2.3 destruido

```
int destruido
```

3.1.2.4 largura

```
int largura
```

3.1.2.5 x

```
int x
```

3.1.2.6 y

```
int y
```

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- main.c

Capítulo 4

Arquivos

4.1 Referência do Arquivo main.c

```
#include <stdio.h>
#include <intelfpgaup/accel.h>
#include <intelfpgaup/KEY.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

Estruturas de Dados

- struct Bloco

  Estrutura para permitir criar os blocos que serão destruidos.

Definições e Macros

- #define LARGURA_BLOCO 32
- #define ALTURA_BLOCO 10
- #define NUM_BLOCOS_X 10
- #define NUM_BLOCOS_Y 6
- #define ESPACAMENTO 2
- #define MARGEM_ESQUERDA 11
- #define MARGEM_SUPERIOR 35
- #define RAIO_BOLA 5

Funções

- void fechar_hardwares ()

  Função para inicialização dos blocos.
- int iniciar_hardwares ()

  Função para inicialização dos dispositivos de hardware.
- void printTelaInicial ()

  Função para desenhar a tela inicial.
- int telaInicial ()

  Função para funcionamento do "menu inicial".
- void limparTela ()

  Função para limpar a tela.
- void printTelaParabens ()

  Função para desenhar a tela de Parabens.
- void gerar_bordas ()

Função para desenhar as bordas do jogo.

- void moverRaquete ()

  Função para inicialização dos blocos.
- void inicializarBlocos ()

  Função para inicialização dos blocos.
- void palavra_score (int scoreInt, int shift_X, int shift_Y)

  Função para desenhar na tela a informação sobre a pontuação.
- void sairPause ()

  Função para inicialização dos blocos.
- void informacao_pause (int boolean)

  Função para inicialização dos blocos.
- void moverBola (int colisao)

  Função para realizar a movimentação da bola.
- void printTelaPerdeu ()

  Função para desenhar a tela de "GAME OVER".
- void detectarColisao ()

  Função para detectar a colisão entre a bola e algum dos blocos.
- void detectarColisaoRaquete ()

  Função para detectar colisão com a raquete.
- int calcularScore ()

  Função para calcular a pontuação.
- void desenhar_bola (int centroX, int centroY)

  Função para inicialização dos blocos.
- void sairTelaPerder (int histScore)

  Função para inicialização dos blocos.
- void numeroOne (int x, int y, short cor)

  Função para desenhar numero 1.
- void numeroTwo (int x, int y, short cor)

  Função para desenhar numero 2.
- void numeroThree (int x, int y, short cor)

  Função para desenhar numero 3.
- void numeroFour (int x, int y, short cor)

  Função para desenhar numero 4.
- void numeroFive (int x, int y, short cor)

  Função para desenhar numero 5.
- void numeroSix (int x, int y, short cor)

  Função para desenhar numero 6.
- void numeroSeven (int x, int y, short cor)

  Função para desenhar numero 7.
- void numeroEight (int x, int y, short cor)

  Função para desenhar numero 8.
- void numeroNine (int x, int y, short cor)

  Função para desenhar numero 9.
- void numeroZero (int x, int y, short cor)

  Função para desenhar numero 0.
- void printValorScore (int Score, int desloc_X, int desloc_Y)

  Função para desenhar o valor da pontuação.
- int blocosAtivos ()

  Função para verificar existencia de blocos ativos.
- void sairTelaParabens ()

  Função para inicialização dos blocos.

- void printBlocosInferiores ()

  Função para desenhar na tela um conjunto de blocos.
- void renderizarBlocos ()

  Função para desenhar os blocos na tela.
- int main ()

  Main do programa.

Variáveis

- Bloco blocos [NUM_BLOCOS_X ∗NUM_BLOCOS_Y]

  Lista dos blocos que serão apresentados e destruidos durante o jogo.
- int xInicialBola = 155

  Variavel que indica o valor de onde a bola deve iniciar em X.
- int yInicialBola = 135

  Variavel que indica o valor de onde a bola deve iniciar em Y.
- int posicaoRaqueteX = 155

  Variavel que indica o valor de onde a raquete deve iniciar em X.
- int velocidadePadraoBolaX = 1

  Variavel que indica o velocidade padrao de movimentação da bola no eixo X.
- int velocidadePadraoBolaY = 1

  Variavel que indica o velocidade padrao de movimentação da bola no eixo Y.
- int screen_x = 320

  Variavel que indica a largura da tela a se trabalhar.
- int screen_y = 240

  Variavel que indica a altura da tela a se trabalhar.
- int quantColunasTexto = 0

  Variavel que indica quantas colunas de texto o video tem disponivel.
- int quantLinhasTexto = 0

  Variavel que indica quantas linhas de texto o video tem disponivel.
- int scoreInt = 0

  Variavel para inicializar o valor da pontuação em 0.
- int velocidadeX = 0

  Variavel confirmar com Naila.

4.1.1 Definições e macros

4.1.1.1 ALTURA_BLOCO

```
#define ALTURA_BLOCO 10
```

4.1.1.2 ESPACAMENTO

```
#define ESPACAMENTO 2
```

4.1.1.3 LARGURA_BLOCO

```
#define LARGURA_BLOCO 32
```

4.1.1.4 MARGEM_ESQUERDA

```
#define MARGEM_ESQUERDA 11
```

4.1.1.5 MARGEM_SUPERIOR

```
#define MARGEM_SUPERIOR 35
```

4.1.1.6 NUM_BLOCOS_X

```
#define NUM_BLOCOS_X 10
```

4.1.1.7 NUM_BLOCOS_Y

```
#define NUM_BLOCOS_Y 6
```

4.1.1.8 RAIO_BOLA

```
#define RAIO_BOLA 5
```

4.1.2 Funções

4.1.2.1 blocosAtivos()

```
int blocosAtivos ( )
```

Função para verificar existencia de blocos ativos.

Esta função serve para poder verificar se ainda existem blocos ativos, e então realizar o retorno informando se existem ou não

Retorna

inteiro para informar a existencia ou ausencia de blocos que ainda não foram destruidos

```
02051 {
02052     int contador = 0;
02053     int i = 0;
02054     for (i = 0; i < NUM_BLOCOS_X * NUM_BLOCOS_Y; i++)
02055     {
02056         if (blocos[i].destruido)
02057         {
02058             contador++;
02059         }
02060     }
02061     if (contador == ((NUM_BLOCOS_X) * (NUM_BLOCOS_Y)))
02062     {
02063         return 1; // sem blocos ativos
02064     }
02065     else
02066     {
02067         return 0; // tem blocos ativos
02068     }
02069 }
```

4.1.2.2 calcularScore()

```
int calcularScore ( )
```

Função para calcular a pontuação.

Esta função serve para fazer o calculo da pontuação, baseando-se na quantidade de blocos que ja foram destruidos. Cada bloco equivale a 10 pontos.

Retorna

contador: uma variavel do tipo Int interna a função e que contabiliza o valor da pontuação

```
01500 {
01501     int contador = 0;
01502     int i = 0;
01503     for (i = 0; i < NUM_BLOCOS_X * NUM_BLOCOS_Y; i++)
01504     {
01505         if (blocos[i].destruido)
01506         {
01507             contador += 10;
01508         }
01509     }
01510     return contador;
01511 }
```

4.1.2.3 desenhar_bola()

```
void desenhar_bola (
            int centroX,
            int centroY )
```

Função para inicialização dos blocos.
Esta função serve para desenhar na tela uma bola especial. Para isso é utilizada uma estrutura de duplo "for", e desenhado pixel a pixel da bola, a fim de apresentar cores diversas na bola.

Parâmetros

centroX	variavel do tipo "int", que serve para indicar a posição X inicial em que a bola deverá iniciar a ser renderizada.
centroY	variavel do tipo "int", que serve para indicar a posição Y inicial em que a bola deverá iniciar a ser renderizada.

```
00662 {
00663     int raio = 5; // Defina o raio da bola
00664     int x, y;
00665 
00666     for (y = -raio; y <= raio; y++)
00667     {
00668         for (x = -raio; x <= raio; x++)
00669         {
00670 
00671             if (x * x + y * y <= raio * raio)
00672             {
00673                 int pixelX = centroX + x;
00674                 int pixelY = centroY + y;
00675 
00676                 int distSquared = x * x + y * y;
00677 
00678                 float aproxSqrt = 1.0 - (float)distSquared / (raio * raio);
00679 
00680                 if (aproxSqrt < 0)
00681                 {
00682                     aproxSqrt = 0;
00683                 }
00684                 else if (aproxSqrt > 1)
00685                 {
00686                     aproxSqrt = 1;
00687                 }
00688 
00689                 int cor = (int)(aproxSqrt * 0xFFFF); // Branco diferente
00690 
00691                 // Desenhar o pixel com a cor
00692                 if (pixelX >= 0 && pixelX < screen_x && pixelY >= 0 && pixelY < screen_y)
00693                 {
00694                     video_pixel(pixelX, pixelY, cor);
00695                 }
00696             }
00697         }
00698     }
00699 }
```

4.1.2.4 detectarColisao()

```
void detectarColisao ( )
```

Função para detectar a colisão entre a bola e algum dos blocos.
Esta função aqui é para verificar se a bola ta colidindo com algum bloco no jogo. Para cada bloco não destruído, verifica se a próxima posição da bola irá colidir com as bordas do bloco. Se houver colisão:

- Marca o bloco como destruído
- Inverte a direção da bola dependendo de onde ela colidiu:
  - Se a bola estava indo para cima, inverte a direção Y para baixo
  - Se a bola estava indo para baixo, inverte a direção Y para cima
  - Se a bola estava indo para a esquerda, inverte a direção X para a direita
  - Se a bola estava indo para a direita, inverte a direção X para a esquerda
- Indica que ocorreu uma colisão e sai do loop Se não houver colisão, atualiza a posição da bola para a próxima posição.

```
00298 {
00299     int i;
00300     int raio = 5;
00301     int colisao = 0;
00302 
00303     // Calcula a próxima posição da bola
```

```
00304     int proximoX = xInicialBola + velocidadePadraoBolaX;
00305     int proximoY = yInicialBola + velocidadePadraoBolaY;
00306
00307     // Verifica colisão com cada bloco
00308     for (i = 0; i < NUM_BLOCOS_X * NUM_BLOCOS_Y; i++)
00309     {
00310         if (!blocos[i].destruido)
00311         {
00312             int bordaEsquerda = blocos[i].x;
00313             int bordaDireita = blocos[i].x + blocos[i].largura;
00314             int bordaCima = blocos[i].y;
00315             int bordaBaixo = blocos[i].y + blocos[i].altura;
00316
00317
00318             if (proximoX + raio >= bordaEsquerda && proximoX - raio <= bordaDireita &&
00319                 proximoY + raio >= bordaCima && proximoY - raio <= bordaBaixo)
00320             {
00321                 // bloco como destruído
00322                 blocos[i].destruido = 1;
00323
00324
00325                 if (velocidadePadraoBolaY < 0) //cima
00326                 {
00327                     velocidadePadraoBolaY = -velocidadePadraoBolaY;
00328                 }
00329                 else if (velocidadePadraoBolaY > 0) // baixo
00330                 {
00331                     velocidadePadraoBolaY = -velocidadePadraoBolaY;
00332                 }
00333                 else if (velocidadePadraoBolaX < 0) //  esquerda
00334                 {
00335                     velocidadePadraoBolaX = -velocidadePadraoBolaX;
00336                 }
00337                 else if (velocidadePadraoBolaX > 0) //direita
00338                 {
00339                     velocidadePadraoBolaX = -velocidadePadraoBolaX; // Inverter direção X da bola
00340                 }
00341
00342                 colisao = 1; // Indica que ocorreu uma colisão
00343                 break;       // Sai do loop após uma colisão
00344             }
00345         }
00346     }
00347
00348     // Se não houve colisão atualiza a posição da bola normalmente
00349     if (!colisao)
00350     {
00351         xInicialBola = proximoX;
00352         yInicialBola = proximoY;
00353     }
00354 }
```

4.1.2.5 detectarColisaoRaquete()

```
void detectarColisaoRaquete ( )
```

Função para detectar colisão com a raquete.

Esta função verifica se a bola ta batendo com a raquete, se tiver realiza mudança na direção e velocidade da bola de acordo na posição em que a colisão ocorreu.

```
00369 {
00370     int raqueteEsquerda = posicaoRaqueteX;
00371     int raqueteDireita = posicaoRaqueteX + 40;
00372     int raqueteTopo = 240 - 5;
00373     int raqueteBase = 240;
00374     int bolaEsquerda = xInicialBola;
00375     int bolaDireita = xInicialBola + 10;
00376     int bolaTopo = yInicialBola;
00377     int bolaBase = yInicialBola + 10;
00378     int deslocamento = 0;
00379
00380     // aqui acontece as verificações para olhar se a bola ta colindindo com a raquete.
00381     if ((bolaDireita >= raqueteEsquerda && bolaEsquerda <= raqueteDireita) &&
00382         (bolaBase >= raqueteTopo && bolaTopo <= raqueteBase))
00383     {
00384
00385         // centro da raquete
00386         int centroRaquete = posicaoRaqueteX + 20;
00387
00388         // deslocamento relacionado ao centro da bola
00389         deslocamento = (xInicialBola + 5) - centroRaquete;
00390
00391         // Ajusta a velocidade x
00392         velocidadeX = deslocamento / 8;
00393
00394         // Limita a velocidade x
```

```
00395         if (velocidadeX > 2)
00396         {
00397             velocidadeX = 2;
00398         }
00399         if (velocidadeX < -2)
00400         {
00401             velocidadeX = -2;
00402         }
00403         if (velocidadeX == 0)
00404         {
00405             velocidadeX = 2;
00406         }
00407 
00408         // Inverte a direção
00409         velocidadePadraoBolaY = -velocidadePadraoBolaY;
00410 
00411         // Atribui a velocidade x
00412         velocidadePadraoBolaX = velocidadeX;
00413     }
00414 }
```

4.1.2.6 fechar_hardwares()

```
void fechar_hardwares ( )
```

Função para inicialização dos blocos.

4.1.3 Prototipação de funções

Esta função serve para simplificar o procedimento de fechar a porta dos hardwares utilizados no jogo. Logo com a chamada de apenas 1 função já fecha os 3 hardwares, o do acelerometro, o do video, e o dos botões.

```
00713                       {
00714     accel_close();
00715     video_close();
00716     KEY_close();
00717 }
```

4.1.3.1 gerar_bordas()

```
void gerar_bordas ( )
```

Função para desenhar as bordas do jogo.

Esta função serve para desenhar na tela as bordas do jogo, usando a logica de criar uma caixa e logo após criar outra caixa menor, deixando a primeira aparecer apenas uma parte, que é a borda.

```
01166 {
01167     int larguraTela, alturaTela, char_x, char_y;
01168     video_read(&larguraTela, &alturaTela, &char_x, &char_y);
01169 
01170     video_box(1, 25, larguraTela - 1, alturaTela - 1, 0x8430);
01171     video_box(6, 31, larguraTela - 6, alturaTela - 6, 0x0000);
01172 }
```

4.1.3.2 informacao_pause()

```
void informacao_pause (
             int boolean )
```

Função para inicialização dos blocos.

Esta função serve para informar com um texto na tela, se o jogo esta pausado caso não esteja nenhuma informação é apresentada.

Parâmetros

boolean	uma variavel do tipo int que irá servir para poder para indicar se o jogo está em pause ou não, usando ela em verificação num "if" logo, seus valores podem ser 1(True) ou 0(False)

```
00635 {
00636     char *pause = "pause";
00637     char *play = "play";
00638 
00639     if (boolean)
00640     {
00641         video_text(quantColunasTexto - 6, 3, pause);
00642     }
00643 
00644 }
```

4.1.3.3 inicializarBlocos()

```
void inicializarBlocos ( )
```

Função para inicialização dos blocos.

Esta função faz a atribuição dos valores iniciais a cada um dos blocos da lista em que estão guardados, colocando então aonde devem iniciar na posição X e Y, além do seu tamanho, tanto em largura como em altura, a sua informação de que não esta destruido, e por fim a cor daquele bloco, que é atribuida seguindo algoritmo a fim de gerar um degrade de cores entre os blocos

```
00227 {
00228     int indiceBloco = 0;
00229     // int larguraDisponivel = (screen_x - ((NUM_BLOCOS_X) * ESPACAMENTO));
00230 
00231     int larguraBloco = 28;
00232     int linha, coluna;
00233 
00234     for (linha = 0; linha < NUM_BLOCOS_Y; linha++)
00235     {
00236         for (coluna = 0; coluna < NUM_BLOCOS_X; coluna++)
00237         {
00238             blocos[indiceBloco].x = coluna * (larguraBloco + ESPACAMENTO) + MARGEM_ESQUERDA;
00239             blocos[indiceBloco].y = linha * (ALTURA_BLOCO + ESPACAMENTO) + MARGEM_SUPERIOR;
00240             blocos[indiceBloco].largura = larguraBloco;
00241             blocos[indiceBloco].altura = ALTURA_BLOCO;
00242             blocos[indiceBloco].destruido = 0;
00243 
00244 
00245             int cor1 = 0xF81F;
00246             int cor2 = 0xF809;
00247             int cor3 = 0xD0C0;
00248 
00249             float percentual = (float)coluna / NUM_BLOCOS_X;
00250             int corIntermediaria;
00251             if (percentual < 0.5)
00252             {
00253                 corIntermediaria = (int)((0.5 - percentual) * 2 * cor1 + (percentual * 2) * cor2);
00254             }
00255             else
00256             {
00257                 corIntermediaria = (int)((1.0 - percentual) * 2 * cor2 + (percentual * 2 - 1.0) *
      cor3);
00258             }
00259 
00260             blocos[indiceBloco].cor = corIntermediaria;
00261 
00262             indiceBloco++;
00263 
00264             if (indiceBloco >= NUM_BLOCOS_X * NUM_BLOCOS_Y)
00265             {
00266                 break;
00267             }
00268         }
00269 
00270         if (indiceBloco >= NUM_BLOCOS_X * NUM_BLOCOS_Y)
00271         {
00272             break;
00273         }
00274     }
00275 }
```

4.1.3.4 iniciar_hardwares()

```
int iniciar_hardwares ( )
```

Função para inicialização dos dispositivos de hardware.

Esta função serve para abrir a porta dos hardwares que serão utilizados no funcionamento do jogo, sendo eles o acelerometro, o video, e o os botões. Caso durante a abertura de um desses elementos haja o retorno de erro, é apresentado na tela do terminal um print informando o erro. Ainda é acertado alguns valores do acelerometro, como feita a sua calibração, e formato de leitura.

Retorna

Inteiro informando que não possivel iniciar algum dos hardwares, caso isso aconteça

```
00736                         {
00737     int larguraTela, alturaTela, char_x, char_y;
00738     int resolucao = 1;
00739     int range = 16;
00740 
00741     if (!accel_open() || !video_open() || !KEY_open())
00742     {
00743         return 0;
```

```
00744     }
00745     accel_init();
00746     video_read(&larguraTela, &alturaTela, &char_x, &char_y); // get screen & text size
00747     accel_format(resolucao, range);
00748     accel_calibrate();
00749     quantColunasTexto = char_x;
00750     quantLinhasTexto = char_y;
00751 }
```

4.1.3.5 limparTela()

```
void limparTela ( )
```

Função para limpar a tela.

Esta função serve para simplificar a limpeza de informações na tela possibilitando com apenas uma chamada de função limpar tanto textos na tela como limpar os outros pixels desenhados na tela.

```
01149 {
01150     video_erase(); // erase any text on the screen
01151     video_clear(); // clear current VGA Back buffer
01152 }
```

4.1.3.6 main()

```
int main ( )
```

Main do programa.

Aqui onde todo o fluxo do jogo acontece, realizando todo os procedimentos para que o jogo possa acontecer.

```
00128 {
00129     char *pause = "paused";
00130     int p, a;
00131     int btn_data = 0;
00132 
00133     //momento da inicialização do hardwares, caso ocorra erro o programa encerra
00134     if (!iniciar_hardwares()){
00135         printf("Não foi possivel iniciar os dispositivos de hardware");
00136         return -1; //caso em que nao foi possivel iniciar alguns dos 3 hardwares
00137     }
00138     //realiza a limpeza de tela e vai apresentar a tela inicial
00139     limparTela();
00140     if (!telaInicial()){
00141         return 0; //caso em que o usuario pressionou o botão de QUIT
00142     }
00143 
00144     /*
00145     realiza a limpeza da tela e inicialização dos blocos para inicio da logica
00146     do jogo, e assim poder jogar
00147     */
00148     limparTela();
00149     inicializarBlocos();
00150 
00151     /*
00152     o codigo permanece nesse loop enquanto o botão de saida geral não for pressioado
00153     */
00154     while (btn_data != 0b0010){
00155         //realiza a limpeza da tela, e leitura do botão
00156         limparTela();
00157         KEY_read(&btn_data);
00158 
00159         //inicio do desenho dos elementos, na ordem de procedencia junto
00160         //a chamadas para detectar colisão, e realizar movimentações dentro do jogo
00161         gerar_bordas();
00162         palavra_score(calcularScore(), 0, 0);
00163         moverRaquete();
00164         detectarColisao();
00165         detectarColisaoRaquete();
00166         moverBola(0);
00167         renderizarBlocos();
00168 
00169 
00170         //verificação para se o botão de pause tiver sido pressionado
00171         if (btn_data == 0b0100){
00172             // video_erase();
00173             informacao_pause(1);
00174             sairPause();
00175         }
00176         //serve para informar a função de pause, que não esta em situação de pause
00177         informacao_pause(0);
00178 
00179         /*
00180         após todos os elementos terem sido desenhados na tela, é feita a chamada para troca
00181         do buffer da tela, e logo a exibição dos elementos na tela
00182         */
00183         video_show();
00184 
```

```
00185         //verificação para se há bocos ativos, e caso não tenha, vai para a pagina de parabens
00186         if (blocosAtivos()){
00187             sairTelaParabens();
00188         }
00189 
00190         //caso a bola tenha ultrapassado a borda inferior, logo houve perda
00191         if (yInicialBola > 230){
00192             int scoreHist = calcularScore();
00193             sairTelaPerder(scoreHist);
00194 
00195         }
00196     }
00197     //limpa a tela, antes de ser encerrado o programa
00198     limparTela();
00199 
00200     // ROTINA DE QUANDO GANHA
00201     if (blocosAtivos()){
00202         limparTela();
00203         printTelaParabens();
00204         video_show();
00205     }
00206     //realiza a troca do buffer da tela, para apresentar a tela limpa
00207     video_show();
00208     //realiza o fechamento das portas de hardware ativas
00209     fechar_hardwares();
00210     return 0;
00211 }
```

4.1.3.7 moverBola()

```
void moverBola (
            int colisao )
```

Função para realizar a movimentação da bola.

Esta função serve para poder realizar a detecção de colisão da bola com as bordas superior, e laterais do jogo. Ao detectar a a colisão é feita a inversão do sentido da bola. Além disso ela realiza a movimentação da bola, alterando a sua posição, e por fim realiza a chamada da função, passando o valor de X e de Y, para desenhar a bola na tela.

Parâmetros

Colisão	valor do tipo inteiro para informar a ocorrencia de uma colisão ou não, os valores aceitos são 0 ou 1, para afirmar ou negar a informação.

```
00435 {
00436     // velocidade atual
00437     xInicialBola += velocidadePadraoBolaX;
00438     yInicialBola += velocidadePadraoBolaY;
00439 
00440     // CORRIGIR COLISAO COM AS LATERAIS
00441 
00442     if (xInicialBola - 5 < 7 || xInicialBola + 5 > 313)
00443     {
00444         velocidadePadraoBolaX = -velocidadePadraoBolaX; // Inverte a direção horizontal
00445     }
00446 
00447     // Verifica se a bola atingiu as bordas superior e inferior da tela
00448     if (yInicialBola - 5 < 31 || yInicialBola + 5 > 300)
00449     {
00450         velocidadePadraoBolaY = -velocidadePadraoBolaY; // Inverte a direção vertical
00451     }
00452 
00453     if (colisao)
00454     {
00455         velocidadePadraoBolaY = -velocidadePadraoBolaY;
00456     }
00457 
00458     desenhar_bola(xInicialBola, yInicialBola);
00459 }
```

4.1.3.8 moverRaquete()

```
void moverRaquete ( )
```

Função para inicialização dos blocos.

Esta função serve para realizar a movimentação da raquete de acordo com o valor lido pelo acelerometro, movimentando ela seja para o lado direito, seja para o lado esquerdo. Para realizar isso ela requisita a leitura do valor capturado pelo acelerometro, esse valor afim de melhor jogabilidade é feita a divisão por 10, para usar apenas 10% do valor lido, e então é somado com o valor da posição da raquete, e logo após é gerada ela utilizando para isso a

função "video_box". Há verificação para que a raquete não ultrapasse os limites da tela, tanto o esquero como o direito.

```
00480 {
00481
00482     int y1 = 240;
00483     int ptr_x;
00484     int ptr_y;
00485     int ptr_z;
00486     int ptr_ready;
00487     int ptr_tap;
00488     int ptr_dtap;
00489     int ptr_msg;
00490
00491     // Não sei exatamente se pode colocar null, caso não possa é só
00492     // adicionar as outras variaveis indicadas.
00493     accel_read(&ptr_ready, &ptr_tap, &ptr_dtap, &ptr_x, &ptr_y, &ptr_z, &ptr_msg);
00494
00495     // melhorar essa questão do y1 para melhorar os valores
00496     video_box(posicaoRaqueteX, y1 - 10, posicaoRaqueteX + 40, y1 - 7, 0xFFFF);
00497     posicaoRaqueteX += ptr_x / 10;
00498
00499     /*os valores tem que considerar que ta falando da posição inicial, e
00500     ainda vai ser somado os 40px que é para fazer a largura da raquete
00501     */
00502     /*
00503     tela tem largura de 320px e raquete largura de 40px, logo
00504     posicaoRaqueteX so pode ir ate 319(tamanho util da tela) - 40 - 5(largura da borda)
00505     e valor minimo da raquete tem de ser 1(inicio da tela) + 5 (tam da borda) + 2 (espaçamento)
00506     */
00507     if (posicaoRaqueteX > 272)
00508     {
00509         posicaoRaqueteX = 272;
00510     }
00511     else if (posicaoRaqueteX < 8)
00512     {
00513         posicaoRaqueteX = 8;
00514     }
00515 }
```

4.1.3.9 numeroEight()

```
void numeroEight (
              int x,
              int y,
              short cor )
```

Função para desenhar numero 8.

Esta função para desenhar na tela o numero 8

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01755 {
01756     video_box(1 + x, 1 + y, 3 + x, 15 + y, cor);
01757     video_box(4 + x, 1 + y, 8 + x, 3 + y, cor);
01758     video_box(9 + x, 1 + y, 11 + x, 15 + y, cor);
01759     video_box(4 + x, 8 + y, 8 + x, 9 + y, cor);
01760     video_box(4 + x, 13 + y, 8 + x, 15 + y, cor);
01761 }
```

4.1.3.10 numeroFive()

```
void numeroFive (
              int x,
              int y,
              short cor )
```

Função para desenhar numero 5.

Esta função para desenhar na tela o numero 5

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01687 {
01688     video_box(1 + x, 1 + y, 11 + x, 3 + y, cor);
01689     video_box(1 + x, 4 + y, 3 + x, 6 + y, cor);
01690     video_box(1 + x, 7 + y, 11 + x, 9 + y, cor);
01691     video_box(9 + x, 10 + y, 11 + x, 12 + y, cor);
01692     video_box(1 + x, 13 + y, 11 + x, 15 + y, cor);
01693 }
```

4.1.3.11 numeroFour()

```
void numeroFour (
             int x,
             int y,
             short cor )
```

Função para desenhar numero 4.

Esta função para desenhar na tela o numero 4

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01664 {
01665     video_box(9 + x, 1 + y, 11 + x, 15 + y, cor);
01666     video_box(7 + x, 2 + y, 8 + x, 3 + y, cor);
01667     video_box(4 + x, 4 + y, 6 + x, 6 + y, cor);
01668     video_box(1 + x, 7 + y, 3 + x, 12 + y, cor);
01669     video_box(4 + x, 10 + y, 8 + x, 12 + y, cor);
01670 }
```

4.1.3.12 numeroNine()

```
void numeroNine (
             int x,
             int y,
             short cor )
```

Função para desenhar numero 9.

Esta função para desenhar na tela o numero 9

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01778 {
01779     video_box(1 + x, 1 + y, 3 + x, 9 + y, cor);
01780     video_box(4 + x, 1 + y, 8 + x, 3 + y, cor);
01781     video_box(4 + x, 7 + y, 8 + x, 9 + y, cor);
01782     video_box(9 + x, 1 + y, 11 + x, 15 + y, cor);
01783     video_box(1 + x, 13 + y, 8 + x, 15 + y, cor);
01784 }
```

4.1.3.13 numeroOne()

```
void numeroOne (
            int x,
            int y,
            short cor )
```

Função para desenhar numero 1.

Esta função para desenhar na tela o numero 1

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01598 {
01599     video_box(1 + x, 7 + y, 3 + x, 9 + y, cor);
01600     video_box(4 + x, 4 + y, 6 + x, 6 + y, cor);
01601     video_box(7 + x, 1 + y, 9 + x, 15 + y, cor);
01602 }
```

4.1.3.14 numeroSeven()

```
void numeroSeven (
            int x,
            int y,
            short cor )
```

Função para desenhar numero 7.

Esta função para desenhar na tela o numero 7

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01733 {
01734     video_box(1 + x, 1 + y, 11 + x, 3 + y, cor);
01735     video_box(8 + x, 4 + y, 10 + x, 8 + y, cor);
01736     video_box(5 + x, 8 + y, 7 + x, 11 + y, cor);
01737     video_box(2 + x, 11 + y, 4 + x, 15 + y, cor);
01738 }
```

4.1.3.15 numeroSix()

```
void numeroSix (
            int x,
            int y,
            short cor )
```

Função para desenhar numero 6.

Esta função para desenhar na tela o numero 6

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01710 {
01711     video_box(1 + x, 1 + y, 11 + x, 3 + y, cor);
01712     video_box(1 + x, 4 + y, 3 + x, 15 + y, cor);
01713     video_box(4 + x, 7 + y, 11 + x, 9 + y, cor);
01714     video_box(9 + x, 10 + y, 11 + x, 12 + y, cor);
01715     video_box(4 + x, 13 + y, 11 + x, 15 + y, cor);
01716 }
```

4.1.3.16 numeroThree()

```
void numeroThree (
            int x,
            int y,
            short cor )
```

Função para desenhar numero 3.

Esta função para desenhar na tela o numero 3

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01642 {
01643     video_box(1 + x, 1 + y, 11 + x, 3 + y, cor);
01644     video_box(9 + x, 1 + y, 11 + x, 15 + y, cor);
01645     video_box(3 + x, 7 + y, 11 + x, 9 + y, cor);
01646     video_box(1 + x, 13 + y, 11 + x, 15 + y, cor);
01647 }
```

4.1.3.17 numeroTwo()

```
void numeroTwo (
            int x,
            int y,
            short cor )
```

Função para desenhar numero 2.

Esta função para desenhar na tela o numero 2

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01619 {
01620     video_box(1 + x, 1 + y, 11 + x, 3 + y, cor);
01621     video_box(9 + x, 4 + y, 11 + x, 6 + y, cor);
01622     video_box(4 + x, 7 + y, 8 + x, 9 + y, cor);
01623     video_box(1 + x, 10 + y, 3 + x, 12 + y, cor);
01624     video_box(1 + x, 13 + y, 11 + x, 15 + y, cor);
01625 }
```

4.1.3.18 numeroZero()

```
void numeroZero (
            int x,
            int y,
            short cor )
```

Função para desenhar numero 0.

Esta função para desenhar na tela o numero 0

Parâmetros

X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
cor	variavel para indicar de qual cor será desenhado o numero na tela

```
01801 {
01802     video_box(1 + x, 1 + y, 3 + x, 15 + y, cor);
01803     video_box(9 + x, 1 + y, 11 + x, 15 + y, cor);
01804     video_box(4 + x, 1 + y, 8 + x, 3 + y, cor);
01805     video_box(4 + x, 13 + y, 8 + x, 15 + y, cor);
01806     video_box(4 + x, 9 + y, 5 + x, 10 + y, cor);
01807     video_box(5 + x, 7 + y, 6 + x, 8 + y, cor);
01808     video_box(7 + x, 5 + y, 8 + x, 6 + y, cor);
01809 }
```

4.1.3.19 palavra_score()

```
void palavra_score (
            int scoreInt,
            int shift_X,
            int shift_Y )
```

Função para desenhar na tela a informação sobre a pontuação.

Esta função serve para escrever na tela a palavra "SCORE" de forma personalizada, além de chamar outra função que apresenta o valor do score na mesma formatação utilizada na palavra "SCORE". Para melhor funcionamento e reuso, a função contém variaveis para permitir o deslocamento dos pixels para onde desejar dentro da tela.

Parâmetros

scoreInt	parametro do tipo inteiro que contém o valor do score obtido ate o momento pelo jogador. Esse valor irá ser passado para a função que escreve o valor numerico na tela.
shift_X	parametro do tipo inteiro que contém o valor de quando o pixel deve ser deslocado para ser apresentado na tela no eixo X, quando não for necessario o deslocamento, usa-se o valor 0;
shift_Y	parametro do tipo inteiro que contém o valor de quando o pixel deve ser deslocado para ser apresentado na tela no eixo Y, quando não for necessario o deslocamento, usa-se o valor 0;

```
01198 {
01199     // x1, y1, x2, y2
01200     // começa x
01201     // começa y
01202     // termina x
01203     // termina y
01204
01205     // S
01206     video_box(8 + shift_X, 3 + shift_Y, 22 + shift_X, 5 + shift_Y, 0xFFFF);    // perna alta do S
01207     video_box(8 + shift_X, 6 + shift_Y, 10 + shift_X, 8 + shift_Y, 0xFFFF);    // perna comprima do S
01208     video_box(8 + shift_X, 9 + shift_Y, 22 + shift_X, 11 + shift_Y, 0xFFFF);   // perna meio do S
01209     video_box(20 + shift_X, 12 + shift_Y, 22 + shift_X, 14 + shift_Y, 0xFFFF); // perna comprima do S
01210     video_box(8 + shift_X, 15 + shift_Y, 22 + shift_X, 17 + shift_Y, 0xFFFF);  // perna alta do S
01211
01212     // C
01213     video_box(25 + shift_X, 3 + shift_Y, 39 + shift_X, 5 + shift_Y, 0xFFFF);   // lateral superior do
      C
01214     video_box(25 + shift_X, 6 + shift_Y, 27 + shift_X, 14 + shift_Y, 0xFFFF);  // perna comprida do C
01215     video_box(25 + shift_X, 15 + shift_Y, 39 + shift_X, 17 + shift_Y, 0xFFFF); // lateral inferior do
      C
01216
01217     // O
01218     video_box(42 + shift_X, 3 + shift_Y, 56 + shift_X, 5 + shift_Y, 0xFFFF);   // lateral superior do
      O
01219     video_box(42 + shift_X, 6 + shift_Y, 44 + shift_X, 14 + shift_Y, 0xFFFF);  // lateral direita do O
01220     video_box(54 + shift_X, 6 + shift_Y, 56 + shift_X, 14 + shift_Y, 0xFFFF);  // lateral esquerda do
      O
01221     video_box(42 + shift_X, 15 + shift_Y, 56 + shift_X, 17 + shift_Y, 0xFFFF); // lateral inferior do
      O
01222
01223     // R
01224     video_box(59 + shift_X, 3 + shift_Y, 73 + shift_X, 5 + shift_Y, 0xFFFF);   // parte de cima do R
01225     video_box(59 + shift_X, 6 + shift_Y, 61 + shift_X, 17 + shift_Y, 0xFFFF);  // lateral direita do R
01226     video_box(71 + shift_X, 6 + shift_Y, 73 + shift_X, 9 + shift_Y, 0xFFFF);   // parte de cima do R
01227     video_box(62 + shift_X, 10 + shift_Y, 73 + shift_X, 12 + shift_Y, 0xFFFF); // parte de cima do R
```

```
01228     video_box(62 + shift_X, 13 + shift_Y, 67 + shift_X, 15 + shift_Y, 0xFFFF);
01229     video_box(68 + shift_X, 15 + shift_Y, 73 + shift_X, 17 + shift_Y, 0xFFFF);
01230
01231     // E
01232     video_box(76 + shift_X, 3 + shift_Y, 90 + shift_X, 5 + shift_Y, 0xFFFF);   // lateral superior do
      E
01233     video_box(76 + shift_X, 6 + shift_Y, 78 + shift_X, 14 + shift_Y, 0xFFFF);  // lateral meio do E
01234     video_box(76 + shift_X, 15 + shift_Y, 90 + shift_X, 17 + shift_Y, 0xFFFF); // lateral inferior do
      E
01235     video_box(79 + shift_X, 9 + shift_Y, 87 + shift_X, 11 + shift_Y, 0xFFFF);  // lateral direita do E
01236
01237     // dois pontos
01238     video_box(93 + shift_X, 4 + shift_Y, 95 + shift_X, 7 + shift_Y, 0xFFFF);   // ponto superior
01239     video_box(93 + shift_X, 13 + shift_Y, 95 + shift_X, 16 + shift_Y, 0xFFFF); // ponto inferior
01240
01241     printValorScore(scoreInt, shift_X, (shift_Y - 1));
01242
01243 }
```

4.1.3.20 printBlocosInferiores()

```
void printBlocosInferiores ( )
```

Função para desenhar na tela um conjunto de blocos.

Esta função faz um desenho de um conjunto de blocos na tela, colocando cores para cada linha de bloco, afim de formar um degrade de cores, indo de um roxo mais escuro, até um rosa.

```
01525 {
01526     // ULTIMA LINHA
01527     int cor1;
01528     int cor2;
01529     int cor3;
01530     int cor4;
01531     int cor5;
01532
01533     cor5 = 0xf810;
01534     cor4 = 0xe014;
01535     cor3 = 0xd016;
01536     cor2 = 0xb01a;
01537     cor1 = 0x901f;
01538
01539     video_box(4, 226, 27, 239, cor1);
01540     video_box(31, 226, 58, 239, cor1);
01541     video_box(62, 226, 89, 239, cor1);
01542     video_box(93, 226, 120, 239, cor1);
01543     video_box(124, 226, 151, 239, cor1);
01544     video_box(155, 226, 182, 239, cor1);
01545     video_box(186, 226, 213, 239, cor1);
01546     video_box(217, 226, 244, 239, cor1);
01547     video_box(248, 226, 275, 239, cor1);
01548     video_box(279, 226, 306, 239, cor1);
01549
01550     // PENULTIMA LINHA
01551     video_box(4, 211, 27, 224, cor2);   // 1
01552     video_box(31, 211, 58, 224, cor2);  // 2
01553     video_box(62, 211, 89, 224, cor2);  // 3
01554     video_box(93, 211, 120, 224, cor2); // 4
01555
01556     video_box(186, 211, 213, 224, cor2); // 7
01557     video_box(217, 211, 244, 224, cor2); // 8
01558     video_box(248, 211, 275, 224, cor2); // 9
01559     video_box(279, 211, 306, 224, cor2); // 10
01560
01561     // ANTEPENULTIMA LINHA
01562     video_box(4, 196, 27, 209, cor3);  // 1
01563     video_box(31, 196, 58, 209, cor3); // 2
01564     video_box(62, 196, 89, 209, cor3); // 3
01565
01566     video_box(217, 196, 244, 209, cor3); // 8
01567     video_box(248, 196, 275, 209, cor3); // 9
01568     video_box(279, 196, 306, 209, cor3); // 10
01569
01570     // ANTEANTEPENULTIMA LINHA
01571     video_box(4, 181, 27, 194, cor4);  // 1
01572     video_box(31, 181, 58, 194, cor4); // 2
01573
01574     video_box(248, 181, 275, 194, cor4); // 9
01575     video_box(279, 181, 306, 194, cor4); // 10
01576
01577     // PRIMEIRA LINHA
01578     video_box(4, 166, 27, 179, cor5); // 1
01579
01580     video_box(279, 166, 306, 179, cor5); // 10
01581 }
```

4.1.3.21 printTelaInicial()

```
void printTelaInicial ( )
```

Função para desenhar a tela inicial.

Esta função serve para imprimir na tela a tela inicial, a tela com o menu do jogo, contendo o nome do jogo e opção para play e quit.

```
00763                       {
00764
00765     /*
00766     24 blocos para letras
00767     8 letras
00768     3 blocos por letra
00769     7 blocos de espaçamento
00770     */
00771     int corL1;
00772     int corL2;
00773     int corL3;
00774     int corL4;
00775     int corL5;
00776
00777     /*
00778     0xf810
00779     oxb1db
00780     0xd016
00781     0xb01a
00782     0x901f
00783
00784
00785     */
00786
00787     corL1 = 0xf810;
00788     corL2 = 0xe014;
00789     corL3 = 0xd016;
00790     corL4 = 0xb01a;
00791     corL5 = 0x901f;
00792
00793     // x1, y1, x2, y2
00794     // começa x
00795     // começa y
00796     // termina x
00797     // termina y
00798
00799     // NOME BREAKOUT DA TELA QUE AMANDA FEZ A FOTO
00800     // PRIMEIRA LINHA 15 BLOCOS #ff004a
00801     video_box(15, 5, 23, 11, corL1); // 1
00802     video_box(25, 5, 33, 11, corL1); // 2
00803     // video_box(35, 5, 43, 11, corL1); //3
00804     // video_box(45, 5, 53, 11, corL1); //4
00805     video_box(55, 5, 63, 11, corL1); // 5
00806     video_box(65, 5, 73, 11, corL1); // 6
00807     // video_box(75, 5, 83, 11, corL1); //7
00808     // video_box(85, 5, 93, 11, corL1); //8
00809     // video_box(95, 5, 103, 11, corL1); //9
00810     video_box(105, 5, 113, 11, corL1); // 10
00811     video_box(115, 5, 123, 11, corL1); // 11
00812     // video_box(125, 5, 133 11, corL1); //12
00813     // video_box(135, 5, 143, 11, corL1); //13
00814     video_box(145, 5, 153, 11, corL1); // 14
00815     // video_box(155, 5, 163, 11, corL1); //15
00816     // video_box(165, 5, 173, 11, corL1); //16
00817     video_box(175, 5, 183, 11, corL1); // 17
00818     // video_box(185, 5, 193, 11, corL1); //18
00819     video_box(195, 5, 203, 11, corL1); // 19
00820     // video_box(205, 5, 213, 11, corL1); //20
00821     // video_box(215, 5, 223, 11, corL1); //21
00822     video_box(225, 5, 233, 11, corL1); // 22
00823     // video_box(235, 5, 243, 11, corL1); //23
00824     // video_box(245, 5, 253, 11, corL1); //24
00825     video_box(255, 5, 263, 11, corL1); // 25
00826     // video_box(265, 5, 273, 11, corL1); //26
00827     video_box(275, 5, 283, 11, corL1); // 27
00828     video_box(285, 5, 293, 11, corL1); // 28
00829     video_box(295, 5, 303, 11, corL1); // 29
00830     video_box(305, 5, 313, 11, corL1); // 30
00831
00832     // SEGUNDA LINHA 14 BLOCOS LARANJAS
00833     video_box(15, 13, 23, 19, corL2); // 1
00834     // video_box(25, 13, 33, 19, 0xF809); //2
00835     video_box(35, 13, 43, 19, corL2); // 3
00836     // video_box(45, 13, 53, 19, 0xF809); //4
00837     video_box(55, 13, 63, 19, corL2); // 5
00838     // video_box(65, 13, 73, 19, 0xF809); //6
00839     video_box(75, 13, 83, 19, corL2); // 7
00840     // video_box(85, 13, 93, 19, 0xF809); //8
00841     video_box(95, 13, 103, 19, corL2); // 9
00842     // video_box(105, 13, 113, 19, 0xF809); //10
```

```
00843     // video_box(115, 13, 123, 19, 0xF809); //11
00844     // video_box(125, 13, 133 19, 0xF809); //12
00845     video_box(135, 13, 143, 19, corL2); // 13
00846     // video_box(145, 13, 153, 19, 0xF809); //14
00847     video_box(155, 13, 163, 19, corL2); // 15
00848     // video_box(165, 13, 173, 19, 0xF809); //16
00849     video_box(175, 13, 183, 19, corL2); // 17
00850     video_box(188, 13, 196, 19, corL2); // 18 - deslocar para esquerda - V
00851     // video_box(195, 13, 203, 19, 0xF809); //19
00852     // video_box(205, 13, 213, 19, 0xF809); //20
00853     video_box(215, 13, 223, 19, corL2); // 21
00854     // video_box(225, 13, 233, 19, 0xF809); //22
00855     video_box(235, 13, 243, 19, corL2); // 23
00856     // video_box(245, 13, 253, 19, 0xF809); //24
00857     video_box(255, 13, 263, 19, corL2); // 25
00858     // video_box(265, 13, 273, 19, 0xF809); //26
00859     video_box(275, 13, 283, 19, corL2); // 27
00860     // video_box(285, 13, 293, 19, 0xF809); //28
00861     video_box(295, 13, 303, 19, corL2); // 29
00862     // video_box(305, 13, 313, 19, 0xF809); //30
00863
00864     // TERCEIRA LINHA 15 BLOCOS AMARELOS
00865     video_box(15, 21, 23, 27, corL3); // 1
00866     video_box(25, 21, 33, 27, corL3); // 2
00867     // video_box(35, 21, 43, 27, 0xF809); //3
00868     // video_box(45, 21, 53, 27, 0xF809); //4
00869     video_box(55, 21, 63, 27, corL3); // 5
00870     video_box(65, 21, 73, 27, corL3); // 6
00871     // video_box(75, 21, 83, 27, 0xF809); //7
00872     // video_box(85, 21, 93, 27, 0xF809); //8
00873     video_box(95, 21, 103, 27, corL3);  // 9
00874     video_box(105, 21, 113, 27, corL3); // 10
00875     // video_box(115, 21, 123, 27, 0xF809); //11
00876     // video_box(125, 21, 133 27, 0xF809); //12
00877     video_box(135, 21, 143, 27, corL3); // 13
00878     video_box(145, 21, 153, 27, corL3); // 14
00879     video_box(155, 21, 163, 27, corL3); // 15
00880     // video_box(165, 21, 173, 27, 0xF809); //16
00881     video_box(175, 21, 183, 27, corL3); // 17
00882     video_box(185, 21, 193, 27, corL3); // 18
00883     // video_box(195, 21, 203, 27, 0xF809); //19
00884     // video_box(205, 21, 213, 27, 0xF809); //20
00885     video_box(215, 21, 223, 27, corL3); // 21
00886     // video_box(225, 21, 233, 27, 0xF809); //22
00887     video_box(235, 21, 243, 27, corL3); // 23
00888     // video_box(245, 21, 253, 27, 0xF809); //24
00889     video_box(255, 21, 263, 27, corL3); // 25
00890     // video_box(265, 21, 273, 27, 0xF809); //26
00891     video_box(275, 21, 283, 27, corL3); // 27
00892     // video_box(285, 21, 293, 27, 0xF809); //28
00893     video_box(295, 21, 303, 27, corL3); // 29
00894     // video_box(305, 21, 313, 27, 0xF809); //30
00895
00896     // QUARTA LINHA 14 BLOCOS VERMELHOS
00897     video_box(15, 29, 23, 35, corL4); // 1
00898     // video_box(25, 29, 33, 35, corL4); //2
00899     video_box(35, 29, 43, 35, corL4); // 3
00900     // video_box(45, 29, 53, 35, corL4); //4
00901     video_box(55, 29, 63, 35, corL4); // 5
00902     // video_box(65, 29, 73, 35, corL4); //6
00903     video_box(75, 29, 83, 35, corL4); // 7
00904     // video_box(85, 29, 93, 35, corL4); //8
00905     video_box(95, 29, 103, 35, corL4); // 9
00906     // video_box(105, 29, 113, 35, corL4); //10
00907     // video_box(115, 29, 123, 35, corL4); //11
00908     // video_box(125, 29, 133, 35, corL4); //12
00909     video_box(135, 29, 143, 35, corL4); // 13
00910     // video_box(145, 29, 153, 35, corL4); //14
00911     video_box(155, 29, 163, 35, corL4); // 15
00912     // video_box(165, 29, 173, 35, corL4); //16
00913     video_box(175, 29, 183, 35, corL4); // 17
00914     video_box(188, 29, 196, 35, corL4); // 18 - deslocar a esquerda - V
00915     // video_box(195, 29, 203, 35, corL4); //19
00916     // video_box(205, 29, 213, 35, corL4); //20
00917     video_box(215, 29, 223, 35, corL4); // 21
00918     // video_box(225, 29, 233, 35, corL4); //22
00919     video_box(235, 29, 243, 35, corL4); // 23
00920     // video_box(245, 29, 253, 35, corL4); //24
00921     video_box(255, 29, 263, 35, corL4); // 25
00922     // video_box(265, 29, 273, 35, corL4); //26
00923     video_box(275, 29, 283, 35, corL4); // 27
00924     // video_box(285, 29, 293, 35, corL4); //28
00925     video_box(295, 29, 303, 35, corL4); // 29
00926     // video_box(305, 29, 313, 35, corL4); //30
00927
00928     // QUINTA LINHA 15 BLOCOS VERMELHOS
00929     video_box(15, 37, 23, 43, corL5); // 1
```

```
00930     video_box(25, 37, 33, 43, corL5); // 2
00931     // video_box(35, 37, 43, 43, corL5); //3
00932     // video_box(45, 37, 53, 43, corL5); //4
00933     video_box(55, 37, 63, 43, corL5); // 5
00934     // video_box(65, 37, 73, 43, corL5); //6
00935     video_box(75, 37, 83, 43, corL5); // 7
00936     // video_box(85, 37, 93, 43, corL5); //8
00937     // video_box(95, 37, 103, 43, corL5); //9
00938     video_box(105, 37, 113, 43, corL5); // 10
00939     video_box(115, 37, 123, 43, corL5); // 11
00940     // video_box(125, 37, 133, 43, corL5); //12
00941     video_box(135, 37, 143, 43, corL5); // 13
00942     // video_box(145, 37, 153, 43, corL5); //14
00943     video_box(155, 37, 163, 43, corL5); // 15
00944     // video_box(165, 37, 173, 43, corL5); //16
00945     video_box(175, 37, 183, 43, corL5); // 17
00946     // video_box(185, 37, 193, 43, corL5); //18
00947     video_box(195, 37, 203, 43, corL5); // 19
00948     // video_box(205, 37, 213, 43, corL5); //20
00949     // video_box(215, 37, 223, 43, corL5); //21
00950     video_box(225, 37, 233, 43, corL5); // 22
00951     // video_box(235, 37, 243, 43, corL5); //23
00952     // video_box(245, 37, 253, 43, corL5); //24
00953     // video_box(255, 37, 263, 43, corL5); //25
00954     video_box(265, 37, 273, 43, corL5); // 26
00955     // video_box(275, 37, 283, 43, corL5); //27
00956     // video_box(285, 37, 293, 43, corL5); //28
00957     video_box(295, 37, 303, 43, corL5); // 29
00958     // video_box(305, 37, 313, 43, corL5); //30
00959
00960     // PALAVRAS PLAY E QUIT
00961     // x1, y1, x2, y2
00962     // começa x
00963     // começa y
00964     // termina x
00965     // termina y
00966     int x_center = 30;
00967     // LETRA P
00968     video_box(x_center + 93, 130, x_center + 95, 144, 0xFFFF); // traço de cima do P
00969     video_box(x_center + 96, 130, x_center + 104, 133, 0xFFFF);
00970     video_box(x_center + 105, 130, x_center + 107, 139, 0xFFFF);
00971     video_box(x_center + 96, 137, x_center + 104, 139, 0xFFFF);
00972
00973     // LETRA L
00974     video_box(x_center + 110, 130, x_center + 112, 144, 0xFFFF);
00975     video_box(x_center + 112, 142, x_center + 124, 144, 0xFFFF);
00976
00977     // LETRA A
00978     video_box(x_center + 127, 130, x_center + 129, 144, 0xFFFF); // traço de cima do P
00979     video_box(x_center + 130, 130, x_center + 138, 133, 0xFFFF);
00980     video_box(x_center + 139, 130, x_center + 141, 144, 0xFFFF);
00981     video_box(x_center + 130, 137, x_center + 138, 139, 0xFFFF);
00982
00983     // LETRA Y
00984     video_box(x_center + 144, 130, x_center + 146, 132, 0xFFFF); // traço de cima do P
00985     video_box(x_center + 146, 133, x_center + 148, 135, 0xFFFF);
00986     video_box(x_center + 148, 136, x_center + 154, 138, 0xFFFF);
00987     video_box(x_center + 154, 133, x_center + 156, 135, 0xFFFF);
00988     video_box(x_center + 156, 130, x_center + 158, 132, 0xFFFF);
00989     video_box(x_center + 150, 139, x_center + 152, 144, 0xFFFF);
00990
00991     // LETRA Q
00992     video_box(x_center + 101, 152, x_center + 103, 166, 0xFFFF); // traço deitado pequeno do
00993     video_box(x_center + 104, 152, x_center + 112, 154, 0xFFFF);
00994     video_box(x_center + 113, 152, x_center + 115, 166, 0xFFFF);
00995     video_box(x_center + 104, 164, x_center + 112, 166, 0xFFFF);
00996     video_box(x_center + 107, 167, x_center + 109, 168, 0xFFFF);
00997     video_box(x_center + 110, 169, x_center + 113, 170, 0xFFFF);
00998
00999     // LETRA U
01000     video_box(x_center + 118, 152, x_center + 120, 166, 0xFFFF);
01001     video_box(x_center + 121, 164, x_center + 129, 166, 0xFFFF);
01002     video_box(x_center + 130, 152, x_center + 132, 166, 0xFFFF);
01003
01004     // LETRA I
01005     video_box(x_center + 135, 152, x_center + 137, 166, 0xFFFF);
01006
01007     // LETRA T
01008     video_box(x_center + 140, 152, x_center + 150, 154, 0xFFFF);
01009     video_box(x_center + 144, 155, x_center + 146, 166, 0xFFFF);
01010
01011     printBlocosInferiores();
01012 }
```

4.1.3.22 printTelaParabens()

```
void printTelaParabens ( )
```

Função para desenhar a tela de Parabens.

Esta função serve para desenhar na tela de parabens, que surge logo após o jogador conseguir quebrar todos os blocos. Ela desenha a palavra Parabens, assim como faz chamada para a função que desenha alguns blocos decorativos ao final da tela.

```
01028 {
01029
01030    int y_desloc =30;
01031    int x_desloc = 20;
01032     int cor1 = 0x81d6;
01033     int cor2 = 0xb96f;
01034     int cor3 = 0xd92c;
01035     int cor4 = 0xf8e9;
01036     int cor5 = 0xf8ab;
01037     int cor6 = 0xf86d;
01038     int cor7 = 0xf84e;
01039     int cor8 = 0xf810;
01040     video_box(x_desloc + 6, y_desloc + 1, x_desloc + 10, y_desloc + 30, cor1);
01041     video_box(x_desloc + 11, y_desloc + 1, x_desloc + 35, y_desloc + 5, cor1);
01042     video_box(x_desloc + 11, y_desloc + 26, x_desloc + 35, y_desloc + 30, cor1);
01043
01044
01045     video_box(x_desloc + 38, y_desloc + 1, x_desloc + 42, y_desloc + 30, cor2);
01046     video_box(x_desloc + 43, y_desloc + 1, x_desloc + 63, y_desloc + 5, cor2);
01047     video_box(x_desloc + 43, y_desloc + 26, x_desloc + 63, y_desloc + 30, cor2);
01048     video_box(x_desloc + 64, y_desloc + 1, x_desloc + 68, y_desloc + 30, cor2);
01049
01050     video_box(x_desloc + 71, y_desloc + 1, x_desloc + 75, y_desloc + 30, cor3);
01051     video_box(x_desloc + 76, y_desloc + 2, x_desloc + 82, y_desloc + 11, cor3);
01052     video_box(x_desloc + 83, y_desloc + 12, x_desloc + 89, y_desloc + 21, cor3);
01053     video_box(x_desloc + 89, y_desloc + 22, x_desloc + 96, y_desloc + 28, cor3);
01054     video_box(x_desloc + 97, y_desloc + 1, x_desloc + 101, y_desloc + 30, cor3);
01055
01056     video_box(x_desloc + 104, y_desloc + 1, x_desloc + 108, y_desloc + 30, cor4);
01057     video_box(x_desloc + 109, y_desloc + 1, x_desloc + 129, y_desloc + 5, cor4);
01058     video_box(x_desloc + 109, y_desloc + 26, x_desloc + 129, y_desloc + 30, cor4);
01059     video_box(x_desloc + 121, y_desloc + 15, x_desloc + 129, y_desloc + 19, cor4);
01060     video_box(x_desloc + 130, y_desloc + 15, x_desloc + 134, y_desloc + 30, cor4);
01061
01062     video_box(x_desloc + 137, y_desloc + 1, x_desloc + 141, y_desloc + 30, cor5);
01063     video_box(x_desloc + 142, y_desloc + 1, x_desloc + 162, y_desloc + 5, cor5);
01064     video_box(x_desloc + 163, y_desloc + 1, x_desloc + 167, y_desloc + 18, cor5);
01065     video_box(x_desloc + 142, y_desloc + 14, x_desloc + 162, y_desloc + 18, cor5);
01066     video_box(x_desloc + 142, y_desloc + 19, x_desloc + 151, y_desloc + 22, cor5);
01067     video_box(x_desloc + 151, y_desloc + 23, x_desloc + 160, y_desloc + 26, cor5);
01068     video_box(x_desloc + 161, y_desloc + 27, x_desloc + 167, y_desloc + 30, cor5);
01069
01070     video_box(x_desloc + 170, y_desloc + 1, x_desloc + 174, y_desloc + 30, cor6);
01071     video_box(x_desloc + 175, y_desloc + 1, x_desloc + 195, y_desloc + 5, cor6);
01072     video_box(x_desloc + 196, y_desloc + 1, x_desloc + 200, y_desloc + 30, cor6);
01073     video_box(x_desloc + 175, y_desloc + 19, x_desloc + 195, y_desloc + 23, cor6);
01074
01075     video_box(x_desloc + 203, y_desloc + 1, x_desloc + 233, y_desloc + 5, cor7);
01076     video_box(x_desloc + 216, y_desloc + 6, x_desloc + 220, y_desloc + 30, cor7);
01077
01078     video_box(x_desloc + 236, y_desloc + 1, x_desloc + 266, y_desloc + 5, cor8);
01079     video_box(x_desloc + 236, y_desloc + 6, x_desloc + 240, y_desloc + 12, cor8);
01080     video_box(x_desloc + 236, y_desloc + 13, x_desloc + 266, y_desloc + 17, cor8);
01081     video_box(x_desloc + 262, y_desloc + 18, x_desloc + 266, y_desloc + 25, cor8);
01082     video_box(x_desloc + 236, y_desloc + 26, x_desloc + 266, y_desloc + 30, cor8);
01083
01084     video_box(x_desloc + 269, y_desloc + 1, x_desloc + 274, y_desloc + 24, cor8);
01085     video_box(x_desloc + 269, y_desloc + 27, x_desloc + 274, y_desloc + 30, cor8);
01086
01087     printBlocosInferiores();
01088 }
```

4.1.3.23 printTelaPerdeu()

```
void printTelaPerdeu ( )
```

Função para desenhar a tela de "GAME OVER".

Esta função serve para desenhar na tela a palavra "GAME OVER", indicando que o jogador perdeu, além disso também faz chamada da função para desenhar blocos decorativos na parte inferior da tela

```
01257 {
01258     int corG;
01259     int corA;
01260     int corM;
01261     int corE;
01262     int corO;
01263     int corV;
01264     int corE2;
```

```
01265     int corR;
01266
01267     int cor1;
01268     int cor2;
01269     int cor3;
01270     int cor4;
01271     int cor5;
01272
01273     cor1 = 0xf810;
01274     cor2 = 0xe014;
01275     cor3 = 0xd016;
01276     cor4 = 0xb01a;
01277     cor5 = 0x901f;
01278
01279     corG = cor1;
01280     corA = cor2;
01281     corM = cor3;
01282     corE = cor5;
01283     corO = cor4;
01284     corV = cor3;
01285     corE2 = cor2;
01286     corR = cor1;
01287
01288     // PRIMEIRA LINHA 15 BLOCOS VERMELHOS
01289     // video_box(15, 5, 23, 11, corG); //1
01290     video_box(25, 5, 33, 11, corG); // 2
01291     video_box(35, 5, 43, 11, corG); // 3
01292     video_box(45, 5, 53, 11, corG); // 4
01293     video_box(55, 5, 63, 11, corG); // 5
01294
01295     video_box(65, 5, 73, 11, corA); // 6
01296     video_box(75, 5, 83, 11, corA); // 7
01297     video_box(85, 5, 93, 11, corA); // 8
01298
01299     video_box(95, 5, 103, 11, corM); // 9
01300     // video_box(105, 5, 113, 11, corM); //10
01301     // video_box(115, 5, 123, 11, corM); //11
01302     // video_box(125, 5, 133 11, corM); //12
01303     video_box(135, 5, 143, 11, corM); // 13
01304
01305     video_box(145, 5, 153, 11, corE); // 14
01306     video_box(155, 5, 163, 11, corE); // 15
01307     video_box(165, 5, 173, 11, corE); // 16
01308
01309     // video_box(175, 5, 183, 11, cor5); //17
01310
01311     video_box(185, 5, 193, 11, corO); // 18
01312     video_box(195, 5, 203, 11, corO); // 19
01313     video_box(205, 5, 213, 11, corO); // 20
01314
01315     video_box(215, 5, 223, 11, corV); // 21
01316     // video_box(225, 5, 233, 11, corV); //22
01317     video_box(235, 5, 243, 11, corV); // 23
01318
01319     video_box(245, 5, 253, 11, corE2); // 24
01320     video_box(255, 5, 263, 11, corE2); // 25
01321     video_box(265, 5, 273, 11, corE2); // 26
01322
01323     video_box(275, 5, 283, 11, corR); // 27
01324     video_box(285, 5, 293, 11, corR); // 28
01325     video_box(295, 5, 303, 11, corR); // 29
01326     // video_box(305, 5, 313, 11, corR); //30
01327
01328     // SEGUNDA LINHA 14 BLOCOS LARANJAS
01329     // video_box(15, 13, 23, 19, corG); //1
01330     video_box(25, 13, 33, 19, corG); // 2
01331     // video_box(35, 13, 43, 19, corG); //3
01332     // video_box(45, 13, 53, 19, corG); //4
01333     // video_box(55, 13, 63, 19, corG); //5
01334
01335     video_box(65, 13, 73, 19, corA); // 6
01336     // video_box(75, 13, 83, 19, corA); //7
01337     video_box(85, 13, 93, 19, corA); // 8
01338
01339     video_box(95, 13, 103, 19, corM);  // 9
01340     video_box(105, 13, 113, 19, corM); // 10
01341     // video_box(115, 13, 123, 19, corM); //11
01342     video_box(125, 13, 133, 19, corM); // 12
01343     video_box(135, 13, 143, 19, corM); // 13
01344
01345     video_box(145, 13, 153, 19, corE); // 14
01346     // video_box(155, 13, 163, 19, corE); //15
01347     // video_box(165, 13, 173, 19, corE); //16
01348
01349     // video_box(175, 13, 183, 19, corO); //17
01350     video_box(185, 13, 193, 19, corO); // 18 - deslocar para esquerda - V
01351     // video_box(195, 13, 203, 19, corO); //19
```

```
01352     video_box(205, 13, 213, 19, corO); // 20
01353
01354     video_box(215, 13, 223, 19, corV); // 21
01355     // video_box(225, 13, 233, 19, corV); //22
01356     video_box(235, 13, 243, 19, corV); // 23
01357
01358     video_box(245, 13, 253, 19, corE2); // 24
01359     // video_box(255, 13, 263, 19, corE2); //25
01360     // video_box(265, 13, 273, 19, corE2); //26
01361
01362     video_box(275, 13, 283, 19, corR); // 27
01363     // video_box(285, 13, 293, 19, corR); //28
01364     video_box(295, 13, 303, 19, corR); // 29
01365     // video_box(305, 13, 313, 19, corR); //30
01366
01367     // TERCEIRA LINHA 15 BLOCOS AMARELOS
01368     // video_box(15, 21, 23, 27, corG); //1
01369     video_box(25, 21, 33, 27, corG); // 2
01370     // video_box(35, 21, 43, 27, corG); //3
01371     video_box(45, 21, 53, 27, corG); // 4
01372     video_box(55, 21, 63, 27, corG); // 5
01373
01374     video_box(65, 21, 73, 27, corA); // 6
01375     // video_box(75, 21, 83, 27, corA); //7
01376     video_box(85, 21, 93, 27, corA); // 8
01377
01378     video_box(95, 21, 103, 27, corM); // 9
01379     // video_box(105, 21, 113, 27, corM); //10
01380     video_box(115, 21, 123, 27, corM); // 11
01381     // video_box(125, 21, 133 27, corM); //12
01382     video_box(135, 21, 143, 27, corM); // 13
01383
01384     video_box(145, 21, 153, 27, corE); // 14
01385     video_box(155, 21, 163, 27, corE); // 15
01386     // video_box(165, 21, 173, 27, corE); //16
01387
01388     // video_box(175, 21, 183, 27, corO); //17
01389     video_box(185, 21, 193, 27, corO); // 18
01390     // video_box(195, 21, 203, 27, corO); //19
01391     video_box(205, 21, 213, 27, corO); // 20
01392
01393     video_box(215, 21, 223, 27, corV); // 21
01394     // video_box(225, 21, 233, 27, corV); //22
01395     video_box(235, 21, 243, 27, corV); // 23
01396
01397     video_box(245, 21, 253, 27, corE2); // 24
01398     video_box(255, 21, 263, 27, corE2); // 25
01399     // video_box(265, 21, 273, 27, corE2); //26
01400
01401     video_box(275, 21, 283, 27, corR); // 27
01402     video_box(285, 21, 293, 27, corR); // 28
01403     video_box(295, 21, 303, 27, corR); // 29
01404     // video_box(305, 21, 313, 27, corR); //30
01405
01406     // QUARTA LINHA 14 BLOCOS VERMELHOS
01407     // video_box(15,29, 23, 35, corG); //1
01408     video_box(25, 29, 33, 35, corG); // 2
01409     // video_box(35,29, 43, 35, corG); //3
01410     // video_box(45,29, 53, 35, corG); //4
01411     video_box(55, 29, 63, 35, corG); // 5
01412
01413     video_box(65, 29, 73, 35, corA); // 6
01414     video_box(75, 29, 83, 35, corA); // 7
01415     video_box(85, 29, 93, 35, corA); // 8
01416
01417     video_box(95, 29, 103, 35, corM); // 9
01418     // video_box(105,29, 113, 35, corM); //10
01419     // video_box(115,29, 123, 35, corM); //11
01420     // video_box(125,29, 133 35, corM); //12
01421     video_box(135, 29, 143, 35, corM); // 13
01422
01423     video_box(145, 29, 153, 35, corE); // 14
01424     // video_box(155,29, 163, 35, corE); //15
01425     // video_box(165,29, 173, 35, corE); //16
01426
01427     // video_box(175,29, 183, 35, corO); //17
01428     video_box(185, 29, 193, 35, corO); // 18 - deslocar a esquerda - V
01429     // video_box(195,29, 203, 35, corO); //19
01430     video_box(205, 29, 213, 35, corO); // 20
01431
01432     video_box(215, 29, 223, 35, corV); // 21
01433     // video_box(225,29, 233, 35, corV); //22
01434     video_box(235, 29, 243, 35, corV); // 23
01435
01436     video_box(245, 29, 253, 35, corE2); // 24
01437     // video_box(255,29, 263, 35, corE2); //25
01438     // video_box(265,29, 273, 35, corE2); //26
```

```
01439
01440     video_box(275, 29, 283, 35, corR); // 27
01441     video_box(285, 29, 293, 35, corR); // 28
01442     // video_box(295,29, 303, 35, corR); //29
01443     // video_box(305,29, 313, 35, corR); //30
01444
01445     // QUINTA LINHA 15 BLOCOS VERMELHOS
01446     // video_box(15, 37, 23, 43 corG); //1
01447     video_box(25, 37, 33, 43, corG); // 2
01448     video_box(35, 37, 43, 43, corG); // 3
01449     video_box(45, 37, 53, 43, corG); // 4
01450     video_box(55, 37, 63, 43, corG); // 5
01451
01452     video_box(65, 37, 73, 43, corA); // 6
01453     // video_box(75, 37, 83, 43, corA); //7
01454     video_box(85, 37, 93, 43, corA); // 8
01455
01456     video_box(95, 37, 103, 43, corM); // 9
01457     // video_box(105, 37, 113,43, corM); //10
01458     // video_box(115, 37, 123,43, corM); //11
01459     // video_box(125, 37, 133 43 corM); //12
01460     video_box(135, 37, 143, 43, corM); // 13
01461
01462     video_box(145, 37, 153, 43, corE); // 14
01463     video_box(155, 37, 163, 43, corE); // 15
01464     video_box(165, 37, 173, 43, corE); // 16
01465
01466     // video_box(175, 37, 183,43, corO); //17
01467     video_box(185, 37, 193, 43, corO); // 18
01468     video_box(195, 37, 203, 43, corO); // 19
01469     video_box(205, 37, 213, 43, corO); // 20
01470
01471     // video_box(215, 37, 223,43, corV); //21
01472     video_box(225, 37, 233, 43, corV); // 22
01473     // video_box(235, 37, 243,43, corV); //23
01474
01475     video_box(245, 37, 253, 43, corE2); // 24
01476     video_box(255, 37, 263, 43, corE2); // 25
01477     video_box(265, 37, 273, 43, corE2); // 26
01478
01479     video_box(275, 37, 283, 43, corR); // 27
01480     // video_box(285, 37, 293,43, corR); //28
01481     video_box(295, 37, 303, 43, corR); // 29
01482     // video_box(305, 37, 313,43, corR); //30
01483     printBlocosInferiores();
01484 }
```

4.1.3.24 printValorScore()

```
void printValorScore (
            int Score,
            int desloc_X,
            int desloc_Y )
```

Função para desenhar o valor da pontuação.

Esta função para desenhar na tela os 3 digitos do valor da pontuação fazendo o ordenamento, e deslocamentos necessarios para desenhar cada digito na sua posição correta, assim como o digito correto

Parâmetros

desloc_X	valor inteiro que serve para indicar o deslocamento no eixo X, podendo mudar a posição do mesmo nesse eixo
desloc_Y	valor inteiro que serve para indicar o deslocamento no eixo Y, podendo mudar a posição do mesmo nesse eixo
Score	variavel inteira que contém o valor da pontuação que deverá ser escrito na tela

```
01830 {
01831     int cor1;
01832     int cor2;
01833     int cor3;
01834     int cor4;
01835     int cor5;
01836
01837     cor1 = 0xf810;
01838     cor2 = 0xe014;
01839     cor3 = 0xd016;
01840     cor4 = 0xb01a;
01841     cor5 = 0x901f;
01842
```

```
01843     int digito1 = 0;
01844     int digito2 = 0;
01845     int digito3 = 0;
01846     short cor = 0xFFFF;
01847 
01848     digito1 = Score / 100;
01849     digito2 = (Score % 100) / 10;
01850     digito3 = (Score % 100) % 10;
01851 
01852     //score entre 100 e 200
01853     if (Score >= 100 && Score < 200)
01854     {
01855         cor = 0xff00;
01856     } //score entre 200 e 300
01857     else if (Score >= 200 && Score < 300)
01858     {
01859         cor = 0xd4e0;
01860     } //score entre 300 e 400
01861     else if (Score >= 300 && Score < 400)
01862     {
01863         cor = 0xe300;
01864     } //score entre 400 e 500
01865     else if (Score >= 400 && Score < 500)
01866     {
01867         cor = 0xe9a0;
01868     } //score maior que 500
01869     else if (Score >= 500)
01870     {
01871         cor = 0xf800;
01872     } //score menor que 100
01873     else if (Score < 100)
01874     {
01875         cor = 0xefe0;
01876     }
01877 
01878     switch (digito1)
01879     {
01880     case 0:
01881         numeroZero(98 + desloc_X, 3 + desloc_Y, cor);
01882         break;
01883     case 1:
01884         numeroOne(98 + desloc_X, 3 + desloc_Y, cor);
01885         break;
01886     case 2:
01887         numeroTwo(98 + desloc_X, 3 + desloc_Y, cor);
01888         break;
01889     case 3:
01890         numeroThree(98 + desloc_X, 3 + desloc_Y, cor);
01891         break;
01892     case 4:
01893         numeroFour(98 + desloc_X, 3 + desloc_Y, cor);
01894         break;
01895     case 5:
01896         numeroFive(98 + desloc_X, 3 + desloc_Y, cor);
01897         break;
01898     case 6:
01899         numeroSix(98 + desloc_X, 3 + desloc_Y, cor);
01900         break;
01901     case 7:
01902         numeroSeven(98 + desloc_X, 3 + desloc_Y, cor);
01903         break;
01904     case 8:
01905         numeroEight(98 + desloc_X, 3 + desloc_Y, cor);
01906         break;
01907     case 9:
01908         numeroNine(98 + desloc_X, 3 + desloc_Y, cor);
01909         break;
01910     default:
01911         break;
01912     }
01913 
01914     switch (digito2)
01915     {
01916     case 0:
01917         numeroZero(111 + desloc_X, 3 + desloc_Y, cor);
01918         break;
01919     case 1:
01920         numeroOne(111 + desloc_X, 3 + desloc_Y, cor);
01921         break;
01922     case 2:
01923         numeroTwo(111 + desloc_X, 3 + desloc_Y, cor);
01924         break;
01925     case 3:
01926         numeroThree(111 + desloc_X, 3 + desloc_Y, cor);
01927         break;
01928     case 4:
01929         numeroFour(111 + desloc_X, 3 + desloc_Y, cor);
```

```
01930         break;
01931     case 5:
01932         numeroFive(111 + desloc_X, 3 + desloc_Y, cor);
01933         break;
01934     case 6:
01935         numeroSix(111 + desloc_X, 3 + desloc_Y, cor);
01936         break;
01937     case 7:
01938         numeroSeven(111 + desloc_X, 3 + desloc_Y, cor);
01939         break;
01940     case 8:
01941         numeroEight(111 + desloc_X, 3 + desloc_Y, cor);
01942         break;
01943     case 9:
01944         numeroNine(111 + desloc_X, 3 + desloc_Y, cor);
01945         break;
01946     default:
01947         break;
01948     }
01949
01950     switch (digito3)
01951     {
01952     case 0:
01953         numeroZero(124 + desloc_X, 3 + desloc_Y, cor);
01954         break;
01955     case 1:
01956         numeroOne(124 + desloc_X, 3 + desloc_Y, cor);
01957         break;
01958     case 2:
01959         numeroTwo(124 + desloc_X, 3 + desloc_Y, cor);
01960         break;
01961     case 3:
01962         numeroThree(124 + desloc_X, 3 + desloc_Y, cor);
01963         break;
01964     case 4:
01965         numeroFour(124 + desloc_X, 3 + desloc_Y, cor);
01966         break;
01967     case 5:
01968         numeroFive(124 + desloc_X, 3 + desloc_Y, cor);
01969         break;
01970     case 6:
01971         numeroSix(124 + desloc_X, 3 + desloc_Y, cor);
01972         break;
01973     case 7:
01974         numeroSeven(124 + desloc_X, 3 + desloc_Y, cor);
01975         break;
01976     case 8:
01977         numeroEight(124 + desloc_X, 3 + desloc_Y, cor);
01978         break;
01979     case 9:
01980         numeroNine(124 + desloc_X, 3 + desloc_Y, cor);
01981         break;
01982     default:
01983         break;
01984     }
01985 }
```

4.1.3.25 renderizarBlocos()

```
void renderizarBlocos ( )
```

Função para desenhar os blocos na tela.

Esta função eh usada para percorrer o array de blocos e desenha cada bloco na tela, se ele nao tiver destruído. Ela utiliza a função video_box para desenhar um retângulo para cada bloco.

Primeiro a função percorre todos os blocos do jogo, representados pelo array 'blocos', verificando se estão destruídos ou não. Se um bloco não estiver destruído, ele é desenhado na tela como um retângulo preenchido.

As coordenadas de cada bloco são feitas da soma das posições x e y do canto esquerdo, e do tamanho largura e altura de cada bloco.

Antes de desenhar, a função olha se os bloco estão dentro dos limites da tela para evitar desenhos fora da área visível e problemas na hora de mostrar.

```
02012 {
02013     int x1;
02014     int y1;
02015     int x2;
02016     int y2;
02017     int p, a;
02018     for (p = 0; p <= NUM_BLOCOS_X; p++)
02019     {
02020         for (a = 0; a <= NUM_BLOCOS_Y; a++)
02021         {
02022             if (!blocos[p * NUM_BLOCOS_Y + a].destruido)
02023             {
```

```
02024                 int x1 = blocos[p * NUM_BLOCOS_Y + a].x;
02025                 int y1 = blocos[p * NUM_BLOCOS_Y + a].y;
02026                 int x2 = x1 + blocos[p * NUM_BLOCOS_Y + a].largura;
02027                 int y2 = y1 + blocos[p * NUM_BLOCOS_Y + a].altura;
02028
02029                 // Para olhar se os blocos estão dentro dos limites da tela
02030                 if (x1 >= 0 && x2 <= screen_x && y1 >= 0 && y2 <= screen_y)
02031                 {
02032                     video_box(x1, y1, x2, y2, blocos[p * NUM_BLOCOS_Y + a].cor);
02033                 }
02034             }
02035         }
02036     }
02037 }
```

4.1.3.26 sairPause()

```
void sairPause ( )
```

Função para inicialização dos blocos.

Esta função serve para iniciar uma rotina de pause do jogo, em que é iniciado um loop condicional, para esperar pela mudança do valor obtido da leitura dos botões, usando a função "KEY_read", ou seja quando o botão de pause/despause for pressionado, ele sai do loop, e retorna para o fluxo normal do jogo novamente.

```
00531 {
00532     int btn_data = 0;
00533     while (btn_data != 0b0100)
00534     {
00535         informacao_pause(1);
00536         KEY_read(&btn_data);
00537     }
00538 }
```

4.1.3.27 sairTelaParabens()

```
void sairTelaParabens ( )
```

Função para inicialização dos blocos.

Esta função serve para apresentar a tela de que o jogador ganhou ao ter quebrado todos os blocos, desejando parabens a ele. Além] disso a partir desse estado é possivel retornar e jogar novamente, para isso alguns valores de variaveis são reiniciados ao seu valor inicial para que possa ser jogado novamente.

```
00599                     {
00600     limparTela();
00601     printTelaParabens();
00602     video_show();
00603     inicializarBlocos();
00604     xInicialBola = 155;
00605     yInicialBola = 135;
00606     posicaoRaqueteX = 155;
00607     int btn_data = 0;
00608     int contador = 0;
00609     char *info = "press button 1 to return";
00610     while (btn_data != 0b1000){
00611         KEY_read(&btn_data);
00612         if (contador < 1000){
00613             contador++;
00614         }
00615         else{
00616             video_text(28, 30, info);
00617         }
00618     }
00619 }
```

4.1.3.28 sairTelaPerder()

```
void sairTelaPerder (
            int histScore )
```

Função para inicialização dos blocos.

Esta função serve para realizar a rotina de apresentar para o jogador que ele perdeu, apresentando a tela de "Game Over". A partir desse estado é possivel retornar e jogar novamente, para isso alguns valores de variaveis são reiniciados ao seu valor inicial para que possa ser jogado novamente. Nesse estado para voltar ao jogo é aguardado o pressionar do botão, para poder retornar.

Parâmetros

histScore	uma variavel do tipo int que irá servir para poder apresentar na tela o valor de pontuação feito naquele momento em que perdeu

```
00559 {
00560     int btn_data = 0;
00561     int contador = 0;
00562     char *info = "press button 1 to return";
00563     xInicialBola = 155;
00564     yInicialBola = 135;
00565     posicaoRaqueteX = 155;
00566     // velocidadeX = 0;
00567     velocidadePadraoBolaY = 1;
00568     velocidadePadraoBolaX = 1;
00569     limparTela();
00570     printTelaPerdeu();
00571     palavra_score(histScore, 82, 166);
00572     video_show();
00573     inicializarBlocos();
00574     while (btn_data != 0b1000)
00575     {
00576         KEY_read(&btn_data);
00577         if (contador < 1000){
00578             contador++;
00579         }
00580         else{
00581             video_text(28, 30, info);
00582         }
00583     }
00584 }
```

4.1.3.29 telaInicial()

```
int telaInicial ( )
```

Função para funcionamento do "menu inicial".

Esta função serve para apresentar a tela inicial do jogo, e fazendo a logica de funcionamento, fazendo a chamada de função para desenhar a tela inicial, e aguardando em loop o pressionar do botão para saber qual a proxima operação a ser realizada, seja ela sair do jogo, ou ir para o jogo.

Retorna

Retorno: variavel do tipo inteiro que indica se jogador apertou para sair do jogo, ou para jogar, ou seja se deu play, ou quit

```
01107 {
01108     int aux_while = 1;
01109     int btn_data, retorno;
01110     while (aux_while){
01111         KEY_read(&btn_data);
01112         limparTela();
01113 
01114         // construir tela de inicio para apresentar antes de ir ao jogo
01115         printTelaInicial();
01116         video_show(); // swap Front/Back to display the line
01117 
01118         switch (btn_data){
01119         case 0b1000:       // botão de PLAY pressionado, o botão mais a esquerda
01120             aux_while = 0; // para poder sair do while e seguir para o jogo
01121             retorno = 1;
01122             break;
01123         case 0b0001: // botao de SAIR pressionado, o botão mais a direita
01124             aux_while = 0;
01125             retorno = 0;
01126             break;
01127 
01128         default:
01129             aux_while = 1;
01130             break;
01131         }
01132     }
01133     return retorno;
01134 }
```

4.1.4 Variáveis

4.1.4.1 blocos

```
Bloco blocos[NUM_BLOCOS_X *NUM_BLOCOS_Y]
```

Lista dos blocos que serão apresentados e destruidos durante o jogo.

4.1.4.2 posicaoRaqueteX

```
int posicaoRaqueteX = 155
```

Variavel que indica o valor de onde a raquete deve iniciar em X.

4.1.4.3 quantColunasTexto

```
int quantColunasTexto = 0
```
Variavel que indica quantas colunas de texto o video tem disponivel.

4.1.4.4 quantLinhasTexto

```
int quantLinhasTexto = 0
```
Variavel que indica quantas linhas de texto o video tem disponivel.

4.1.4.5 scoreInt

```
int scoreInt = 0
```
Variavel para inicializar o valor da pontuação em 0.

4.1.4.6 screen_x

```
int screen_x = 320
```
Variavel que indica a largura da tela a se trabalhar.

4.1.4.7 screen_y

```
int screen_y = 240
```
Variavel que indica a altura da tela a se trabalhar.

4.1.4.8 velocidadePadraoBolaX

```
int velocidadePadraoBolaX = 1
```
Variavel que indica o velocidade padrao de movimentação da bola no eixo X.

4.1.4.9 velocidadePadraoBolaY

```
int velocidadePadraoBolaY = 1
```
Variavel que indica o velocidade padrao de movimentação da bola no eixo Y.

4.1.4.10 velocidadeX

```
int velocidadeX = 0
```
Variavel confirmar com Naila.

4.1.4.11 xInicialBola

```
int xInicialBola = 155
```
Variavel que indica o valor de onde a bola deve iniciar em X.

4.1.4.12 yInicialBola

```
int yInicialBola = 135
```
Variavel que indica o valor de onde a bola deve iniciar em Y.

Índice Remissivo