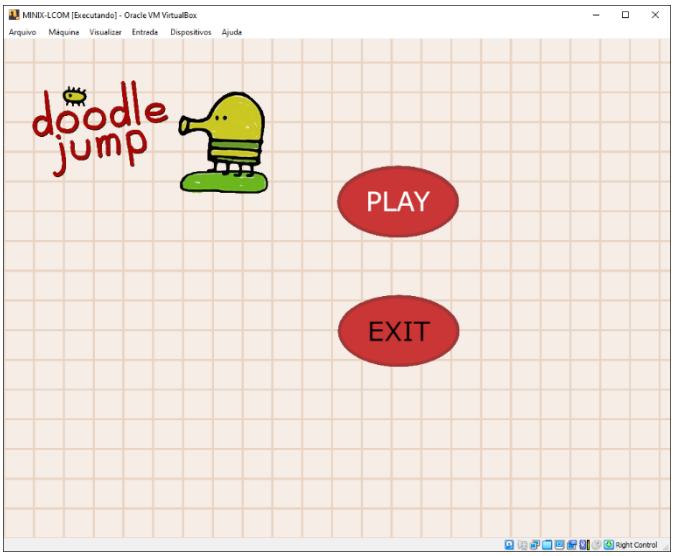
Relatório de LCOM

Pedro Rezende de Carvalho – up201900513

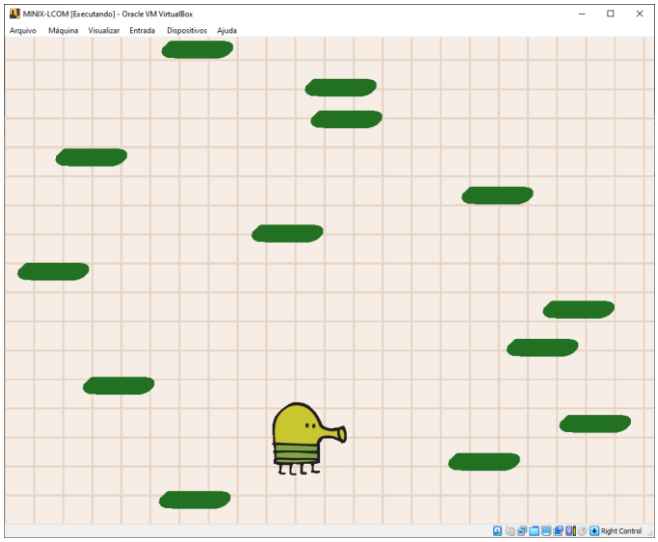
1.Instruções de utilizador

1.1 Menu Inicial



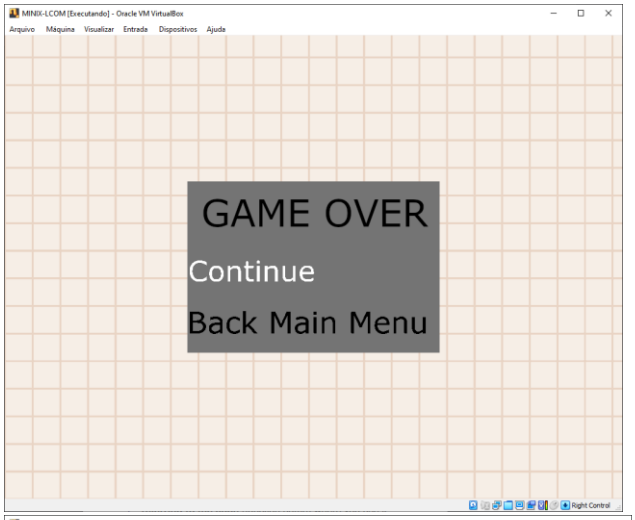
Para navegar no menu basta usar as setas do teclado, para selecionar uma opção pressionar ENTER.

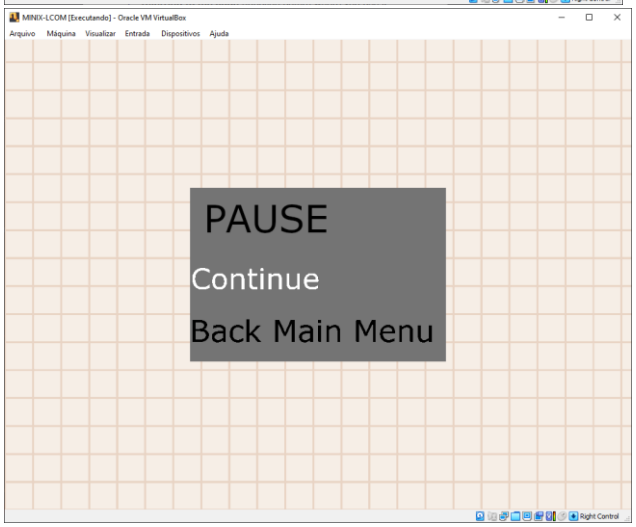
1.2 Jogo



Para controlar o Doodler use as setas do teclado, caso deseje pausar o jogo aperte ESC, assim como no menu principal para selecionar a opção que deseja use as setas e ENTER para confirmar.

Menus:


MINIX-LCOM [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
GAME OVER
Continue
Back Main Menu
Right Control


MINIX-LCOM [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
PAUSE
Continue
Back Main Menu
Right Control

2 Estado do projeto

2.1 Dispositivos E/S usados

Dispositivo	Descrição	Modo
Timer	Usado para a mudança de frames	Modo de interrupção
Teclado	Usado para fazer a interface com usuário	Modo de interrupção
Placa gráfica	Usada para mostrar o jogo e os menus	Modo normal

2.2.1 Timer

O Timer foi utilizado para controlar o número de atualizações de frames por segundo e para controlar a velocidade do jogo.

O seguinte código usado controla quando cada frame deve ser mostrado.

em que time_int_handler() é o interrupt handler do timer

```
if(msg.m_notify.interrupts & timer_irq){
  timer_int_handler();
  if(counter%(60/FRAME_RATE) == 0){
    switch (game_state)
    {
    case main_menu:
      draw_main_menu();
      break;
    case game_loop:
      draw_frame();
      break;
    case confirmation:
      draw_main_menu();
      break;
    case paused:
      draw_frame();
      break;
    case game_over:
      draw_frame();
      break;
    }
    counter = 0;
  }
}
```

e o switch responsável por controlar os estados.

2.2.2 Teclado

O teclado foi utilizado pare fazer interface com usuário o seu funcionamento é simples, compara os scancodes com as teclas mapeadas

```
if(packet == DOWN_ARROW_PRESS || packet == UP_ARROW_PRESS){
  if(menu.play_button.is_selected){
    menu.play_button.is_selected = false;
    menu.exit.is_selected = true;
  }
  else if(menu.exit.is_selected){
    menu.exit.is_selected = false;
    menu.play_button.is_selected = true;
  }
}
```

Fragmento código demonstrando o uso do teclado em que é apenas feito uma comparação entre os scancodes, packets é o scancode processado.

2.2.3 Placa gráfica

A placa gráfica é a que possui a implementação mais complexa na qual a diversas funções de interface, como draw_pixel() responsável por lidar com a codificação de cor e escrever no frame buffer.

A implementação de double buffer implementado foi o page flipping em que é utilizado a função page_flip() para a troca de buffers.

```
void page_flip(){
  reg86_t reg;
  memset(&reg, 0, sizeof(reg));
  reg.intno = VBE_INT;
  reg.ax = VBE_AX_FUNC07;
  reg.bl = 0x80;
  video_mem_visual = video_mem_active;
  if(page){
    video_mem_active = video_mem_2;
    reg.cx = 0;
    reg.dx = 0;
  } else{
    video_mem_active = video_mem_1;
    reg.cx = 0;
    reg.dx = scanlines;
  }
```

Para os movimentar os Sprites é utilizado a variável xspeed e yspeed dos Sprites em que cada atualização de quadro se atualiza as posições do dos sprites.

Antes da implementação do page flipping para mover um Sprite primeiro era usado clear_sprite() para limpar a posição antiga e usando a draw_sprite() com o x e y aumentados com suas respectivas velocidades para desenhar a sprite em sua nova posição, porém por conta da complexidade de controlar a posição dos sprites em dois buffers optou-se por apenas redesenhar o background e suas novas posições.

As funções do VBE utilizadas foram:

A função 1(0x4F01) para obter as informações via vbe_mode_info_t.

A função 2(0x4F02) para inicialização da placa gráfica.

A função 6(0x4F06) para obter o número de scanlines do buffer utilizada para o page flip.

A função 7(0x4F07) para alterar os registros da placa gráfica que apontam para o início do frame buffer, usado na implementação do page flip.

3 Estrutura do código

3.1 game.h/game.c

É onde o principal loop do jogo se localiza o game_main_loop() onde é lidado com as interrupções do driver receive e suas interpretações dependendo dos estado do jogo em game_state. Além do loop principal é nesse módulo onde se reside a lógica do jogo.

3.2 i8242.h/timer.c

API desenvolvida no laboratório 2 que lida com o timer.

3.3 kbc.h/kbc.c

API desenvolvida no laboratório 3 e 4 que lida com o teclado e com o mouse.

3.4 proj.c

Código do LCF e tratamento de inputs com o main, inicialmente foi planejado suportar mais de um modo, porem por questões de complexidade e tempo não foi implementado. O código foi desenvolvido para operar no modo 118 do VBE

3.5 sprite.h/sprite.c

Código responsável de lidar com os Sprites e com o background.

3.6 video.h/vídeo.c

API desenvolvido no lab5 com algumas alterações para suportar backgrounds personalizados e page flipping, como a implementação da função 6 e 7 do VBE.

3.7 Contribuição

No final todo código foi desenvolvido por apenas uma pessoa assim como o relatório.

4 Detalhes de Implementação

4.1 Page flipping

A implementação que levou mais tempo. A dificuldade veio de duas partes a sua dificuldade técnica em usar as funções do VBE e entender o manual da VESA que nem sempre é muito claro em especial quanto as “logical scanlines”, a outra dificuldade veio em sincronizar os buffer no sentido de implementar de maneira optimizada o apagar e desenhar dos sprites, sendo que apenas redesenhar tudo deixava o código muito pesado e mesmo na implementação atual, na qual apenas evita acessar a memória quando algo já está presente, o código não roda tão leve quando rodava com a penas um buffer.

4.2 Máquina de estados e abstrações

Para o controle de estados dentro do jogo foi desenvolvido uma máquina de estados simples que dita quando um input altera o jogo ou os menus.

Para os menus tentou-se abstrair usando structs para ajudar legibilidade do código e reusar API’s quando possível.

4.3 Matéria não apresentada nas aulas

Uma parte da matéria em que não foi entrada em detalhe foi as logicals scanlines em que é usada pelo VBE no modo real, não consegui pelo menos achar material sobre isso.

5 Conclusão

Em comparação com os laboratórios foi no projeto em que mais se aprendeu não somente sobre a matéria em si, mas também sobre o gerenciamento de um projeto que demanda um pouco mais do que uma semana de trabalho. Infelizmente não foi possível implementar tudo que se desejava com mouse e o RTC, também era planejado um sistema de scores que acabou não acontecendo.