Licenciatura em Engenharia Informática e Computação

Laboratório de Computadores, 2021/2022

Turma 5, Grupo 1

Guilherme Sequeira, 202004648

João Pereira, 202007145

Nuno Pereira, 202007865

Pedro Ramalho, 202004715

Índice

Instruções 3
O quadro branco 3
Botões interativos 4
Botões de cor 4
Botões de estilo 4
Relógio 5
Estado do projeto 5
Periféricos 5
Timer 5
KBD 6
Placa Gráfica 6
RTC 7
Organização e estrutura do código 8
Módulos 8
canvas (20%) 8
charset (10%) 8
cursor (10%) 8
draw_buttons (5%) 8
draw_text (5%) 8
draw_clock (10%) 9
element (5%) 9
pixel_buffer (5%) 9
text (5%) 9
position (5%) 9
graphics (20%) 9
Gráfico de chamada de funções 10
Detalhes de implementação 10
Desenho de texto na tela 10
Conclusões 11
Referências 12

Instruções

Para correr o projeto deverão ser seguidos os seguintes passos:

1. Abrir a máquina virtual de LCOM e fazer *login*
2. Fazer `cd labs/proj`
3. Compilar o projeto fazendo `make`
4. Finalmente, correr o projeto fazendo `lcom_run proj`

De seguida são explicadas as secções e funcionalidades do projeto em detalhe.

O quadro branco

Após ter sido iniciado o programa, é apresentado ao utilizador um quadro branco (*canvas*), diferentes botões interativos com funcionalidades diferentes, um cursor e um relógio.

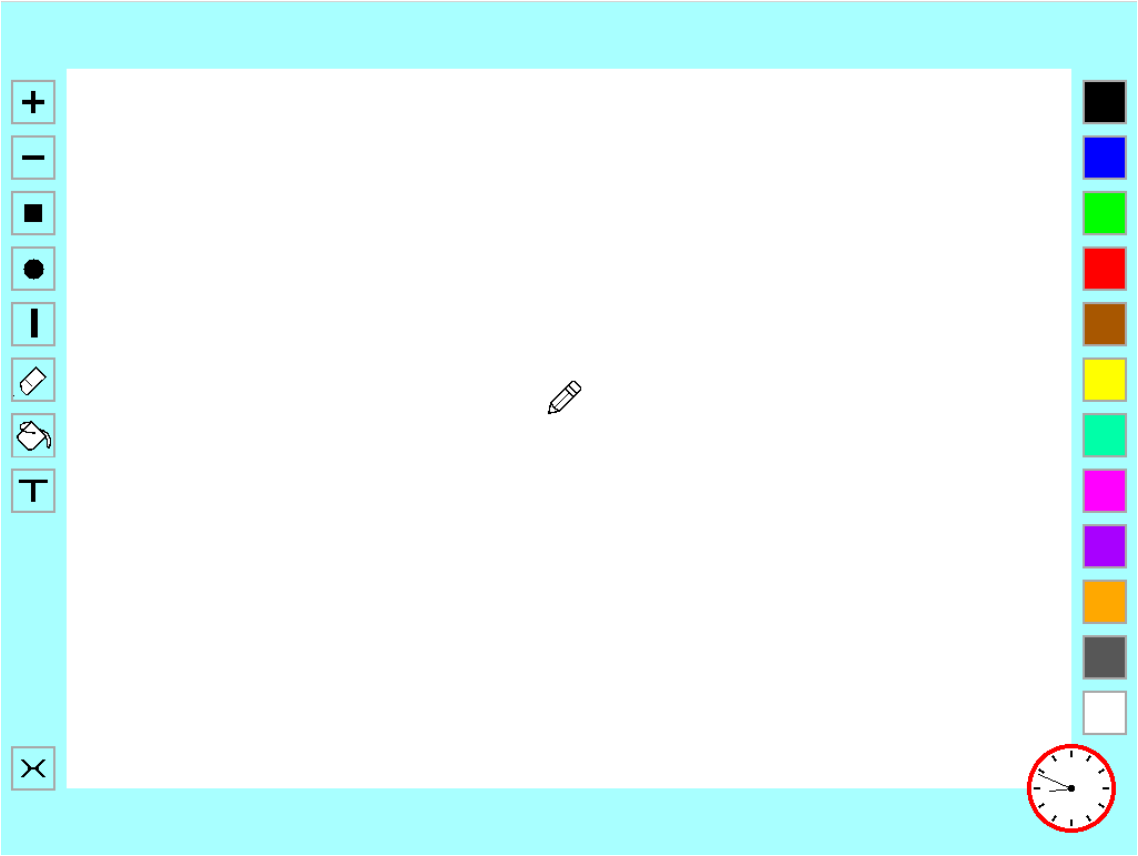


Figura 1 – quadro branco inicial

Botões interativos

Botões interativos são os botões com os quais o utilizador pode interagir através do rato. Existem dois tipos de botões: botões de cor e botões de estilo.

Botões de cor

Estes botões, como o nome indica, são os botões que determinam a cor que as várias ferramentas utilizam para desenhar na tela.

Figura 2 – botões de cor

Botões de estilo

Estes botões são os responsáveis por decidir que ferramenta é utilizada para desenhar. A seguir encontra-se uma explicação detalhada sobre cada uma destas ferramentas.

Figura 3

- Botão que aumenta a espessura da ferramenta usada para desenhar. A espessura não pode ser aumentada para além de um dado limite.

Figura 4

- Botão que diminui a espessura da ferramenta usada para desenhar. A espessura não pode ser diminuída para além de um dado limite.

Figura 5

- Botão que altera a ponta do lápis para um quadrado.

Figura 6

- Botão que altera a ponta do lápis para um círculo.

Figura 7

- Botão que seleciona a ferramenta de desenhar linhas. O utilizador deverá clicar numa posição inicial e final e será desenhada uma linha reta entre elas.

Figura 8

- Botão que seleciona a ferramenta borracha. A borracha funciona como um pincel circular que pinta a tela com a cor de fundo da tela atual.

Figura 9

- Botão que seleciona a ferramenta balde. O balde pinta a tela com a cor atualmente selecionada.

Figura 10

- Botão que seleciona a ferramenta para escrever texto. As letras do texto serão da cor que está atualmente selecionada.

Figura 11

- Botão que limpa a tela com a cor branca.

Relógio

Existe ainda um relógio que indica a hora atual.

Figura 12 - relógio

Estado do projeto

Todas as funcionalidades descritas na secção acima foram implementadas com sucesso.

Periféricos

Periférico	**Função**	**Interrupções/*Polling***
Timer	Controlar a *refresh rate*	I
KBD	Escrever texto na tela	I
Rato	Desenhar na tela e interagir com os botões	I
Placa gráfica	*Display* da tela e de vários *xpms*	*N/A*
RTC	Horas, minutos e segundos atuais	I

Timer

O *timer* é utilizado para definir a *refresh rate* (60 FPS) do nosso programa. Essencialmente, determina a frequência com que o *front buffer* (VRAM) é atualizado através das suas interrupções.

Foram implementadas funções para subscrever e cancelar interrupções do timer (`timer_subscribe_int()` e `timer_unsubscribe_int()`). Foi ainda implementado um *interrupt handler* (`timer_int_handler()`) que incrementa um contador global.

KBD

O teclado é utilizado para o utilizador escrever texto na tela e sair do programa, clicando na tecla ESC. Para tal, o utilizador deverá selecionar a ferramenta de texto e clicar na posição da tela onde deseja escrever.

Foram implementadas funções para subscrever e cancelar interrupções do teclado (`kbc_subscribe_int()` e `kbc_unsubscribe_int()`). Foi ainda implementado um *interrupt handler* (`kbc_ih()`) que recebe e processa os *bytes* gerados pelo KBC.

Adicionalmente, foi criada uma função que mapeia *scancodes* em caracteres (`process_scancode()`) e caso o utilizador esteja a escrever, poderá adicionar ou remover caracteres da palavra a ser escrita ou ainda torná-la definitiva (em vez de estar a ser escrita temporariamente por cima do que já está na tela, ela é guardada no *buffer* do *canvas*).

Rato

O rato é utilizado para o utilizador desenhar na tela e interagir com os seus elementos, como o *canvas* e os botões anteriormente mencionados.

Foram implementadas funções para permitir o *data reporting* do rato (`kbc_enable_data_report()`), subscrever e cancelar interrupções deste (`mouse_subscribe_int()` e `mouse_unsubscribe_int()`). Foi ainda desenvolvido um *interrupt handler* (`mouse_ih()`) que processa os *bytes* enviados por este, construindo os *mouse packets*, que contêm informação acerca de quais botões do rato foram pressionados, deslocamentos, etc. Esta informação é depois usada pelo **cursor**, que funciona como uma abstração do rato. O **cursor** é explicado em maior detalhe na secção "Módulos".

Placa Gráfica

A placa gráfica foi inicializada no modo `0x105` (1024x768, modo indexado, 8 *bits* por pixel).

Foi implementado um mecanismo de *double buffering* com *page flipping*. Foram criados dois *buffers* (*front* e *back buffer*), sendo que o primeiro está situado em VRAM e o segundo em memória. A cada interrupção do *timer*, o conteúdo destes dois *buffers* é trocado usando a função `flip()`.

O *layout* da tela foi obtido desenhando retângulos com as cores respetivas usando a função `buf_draw_rectangle()` (explicado com mais detalhe na secção "Módulos"). Alguns dos botões e os diferentes estilos do cursor estão guardados no formato ***xpm***. É ainda utilizada uma fonte de texto, guardada no mesmo formato.

Inicialmente, cada um destes *xpms* é carregado usando a função `vg_load_xpm()`, sendo mais tarde desenhado na tela na posição adequada através da função `buf_draw_xpm()`.

RTC

O RTC é utilizado para manter registo da hora à medida que o programa corre. Esta funcionalidade permite-nos gerar os gráficos para o relógio, tal como apresentado na figura 12.

Foram implementadas funções para subscrever e cancelar interrupções deste (`rtc_subscribe_int()` e `rtc_unsubscribe_int()`). Foi ainda desenvolvido um *interrupt handler* (`rtc_int_handler()`) que lê os registos do RTC relativos a horas, minutos e segundos e atualiza a representação interna destes no programa.

Organização e estrutura do código

Módulos

canvas (20%)

Este módulo trata da manipulação da região onde o utilizador pode desenhar. O *canvas* contém um *pixel buffer* (variável estática) dedicado, de forma a ser possível apresentar na tela elementos temporários (cursor/texto) sem que seja necessário reescrever para memória o conteúdo em baixo dos elementos. É ainda guardada uma variável estática adicional, `canvas_background_color`, que armazena a cor de fundo atual do *canvas*. Para além disso, estão presentes as funções que determinam o comportamento das diversas ferramentas de desenho, como por exemplo `canvas_handle_line`, `canvas_handle_text`, `canvas_draw_pencil_circle`, etc.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

charset (10%)

Módulo que contém uma função para processar *scancodes* de acordo com um *charset*. Contém também o *charset* correspondente ao teclado português.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

cursor (10%)

Este módulo funciona como uma abstração do rato do Minix. O cursor contém várias variáveis estáticas, como `cursor_pos`, `cursor_color`, `cursor_thickness`, `cursor_lb` e `typing`, que indicam, respetivamente, a posição, a cor, a espessura, se o botão esquerdo está a ser pressionado pela primeira vez e se o utilizador se encontra a escrever texto na tela. O cursor possui ainda estados, implementados por meio de um `enum`, que indicam o seu estilo e que ferramenta foi selecionada pelo utilizador. Dependendo do estado atual do cursor e da sua posição, um *xpm* diferente é desenhado.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

draw_buttons (5%)

Este módulo é encarregue de carregar e desenhar todos os *xpms* relacionados com os botões de estilo apresentados na tela.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

draw_text (5%)

Módulo que contém o conjunto de funções utilizadas para desenhar texto.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

draw_clock (10%)

Módulo que contém o conjunto de funções utilizadas para gerir o *xpm* relacionado com o relógio e desenhar o relógio. Contém também constantes auxiliares.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

element (5%)

Módulo responsável por inicializar todos os elementos interativos da tela, incluindo os botões de cor e estilo e o *canvas*. Para tal, foi criada a estrutura `interactive_element`, que cria um elemento dada uma posição, comprimento e largura, um apontador para a função responsável por tratar da interação do elemento com o cursor e ainda um argumento adicional que será passado a esta função. Foi ainda implementada a função `is_hovered()`, que indica se um dado elemento está a ser *hovered* pelo cursor.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

pixel_buffer (5%)

Módulo que contém a definição da estrutura `pixel_buffer`, que recebe como argumentos uma resolução horizontal e vertical, o tamanho do *buffer*, o número de *bytes* por pixel e o endereço-base deste.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

text (5%)

Módulo que contém o conjunto de funções utilizadas para gerir o *xpm* com o tipo de letra utilizado e desenhar texto. Contém também constantes auxiliares.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

position (5%)

Módulo que contém a definição da estrutura `position`, que recebe como argumentos uma abcissa *x* e uma ordenada *y*.

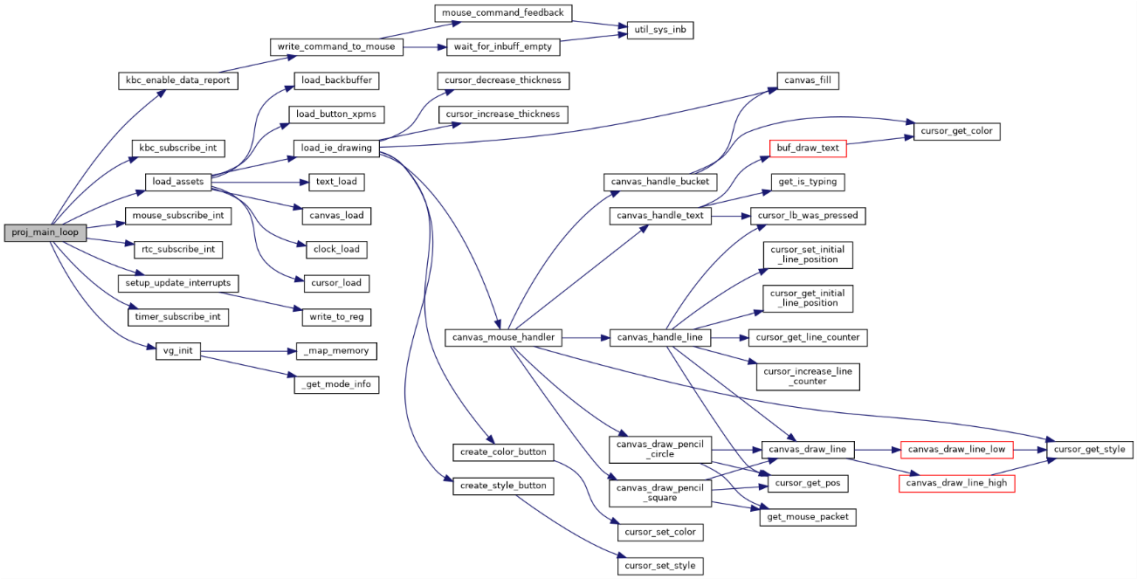
Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

graphics (20%)

Módulo que contém o conjunto de funções utilizadas para desenhar diversas formas em `pixel_buffers`, como por exemplo `buf_draw_pixel()`, `buf_draw_line()`, `buf_draw_circle()`, `buf_draw_rectangle`, `buf_draw_xpm()`, `buf_draw_text()`, etc.

Este módulo foi implementado por Guilherme Sequeira, João Pereira, Nuno Afonso e Pedro Ramalho.

Gráfico de chamada de funções



Detalhes de implementação

Desenho de texto na tela

Como foi mencionado anteriormente, o *interrupt handler* do teclado recebe e processa os *bytes* dos *scancodes* enviados pelo KBC. Após o utilizador selecionar a ferramenta de texto e uma posição (indicada através de um clique do botão esquerdo do rato no *canvas*), a variável estática `typing` do cursor será atualizada para `true` e poderá digitar o texto pretendido. Os *bytes* recebidos por interrupções do KBC são então mapeados nos caracteres respetivos pela função `process_scancode` e cada caráter válido digitado será acrescentado à variável `text` que armazena a palavra a ser digitada atualmente. Esta palavra é escrita diretamente no *back buffer* (através da função `buf_draw_text()`) e não no *buffer* dedicado ao *canvas*, de forma a que seja possível apagar ou reescrever a palavra de forma eficiente.

Quando o utilizador concluir a escrita da palavra (pressionando a tecla ENTER ou clicando no botão esquerdo do rato), será invocada a função `buf_draw_text()` no *buffer* do *canvas*, guardando a palavra digitada. A variável `text` é limpa de modo a estar pronta a receber uma nova palavra.

Se o utilizador pressionou ENTER ou clicou com o rato fora do *canvas*, a variável `typing` é atualizada para `false`. Se tiver clicado com rato dentro do *canvas*, poderá começar a digitar novamente uma palavra nessa posição.

Conclusões

Inicialmente, o pretendido era implementar uma versão do jogo *Gartic* a dois jogadores. Por falta de tempo, tal não foi possível. Decidimos assim adaptar o nosso programa de desenho semelhante ao *Paint*.

No futuro, seriam boas adições:

- a ferramenta do balde seguir um algoritmo de *flood fill*, de modo a pintar apenas uma dada área do *canvas*. O esqueleto desta função ainda se encontra presente no código apesar de não estar funcional.
- uma ferramenta que permita desenhar curvas de Bézier
- ferramentas que permitam desenhar retângulos e círculos de tamanho manipulável, semelhante à ferramenta das linhas implementada
- ser apresentada uma pré-visualização da linha a ser desenhada (de forma semelhante ao texto que ainda não foi guardado) e/ou das ferramentas de retângulos e círculos

Finalmente, gostaríamos de utilizar algumas das funções cujo propósito era a implementação de um *chat* que não estar a ser utilizadas ou foram abandonadas devido à falta de *serial port*.

Referências

Para desenhar as linhas foi usado como base o algoritmo de Bresenham:

- https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm

Fomos também acompanhados pelo professor Pedro Brandão.