

# COVID ESCAPE

LABORATÓRIO DE COMPUTADORES

Gabriel  
Martins  
up201906072

Carlos  
Veríssimo  
up201907716

J A N E I R O  
2 0 2 1



# CONTEÚDOS

## Instruções de utilização

Menu principal

Play game

How to play

Exit game

## Estado do projeto

Dispositivos usados

Timer

Teclado

Rato

Placa Gráfica

## Estrutura do código

Covid Escape

Proj

Timer

Keyboard

Mouse

Gráficos

Estado da máquina

I8042/I8254

Call Graph

## Análise e conclusões

# Instruções de utilização

COVID ESCAPE

## M e n u Principal

O menu inicial é apresentado ao iniciar o programa com as seguintes opções Start game, inicia um novo jogo. Howto play—, menu onde explica os comandos do jogo Exit, sai do programa. O menu permite a utilização do rato para selecionar as opções ou as teclas 1,2 e 3

## P l a y g a m e

O jogo tem como objetivo escapar dos obstáculos que aparecem no ecrã pelo maior período possível. O vírus é controlado quer pelo espaço, a tecla “W” ou a tecla “” que faz com que salte, a tecla “A” ou “→” para pequenos ajustes para a esquerda e a tecla “D” ou “←” para pequenos ajustes para direita. À colisão com um obstáculo perdesse o jogo e voltasse ao menu inicial escolhendo novamente uma das 3 opções.

PLAY GAME

HOW TO PLAY





EXIT GAME

## How To Play

Neste menu, o jogador pode ler as regras do jogo quando já o tiver feito pode seleccionar o botão de retorno com o rato ou pressionar a tecla “ESC”. Após esta ação será re-direccionado para o menu inicial novamente.

### HOW TO PLAY



Squeeze through hospital buildings  without hitting them by pressing the spacebar  to go up, pressing the left arrow  or the right arrow  on your keyboard to make small adjustments to the left or right, respectively. If you wish to go down, just let gravity do its job.



Have Fun!

MENU

# Estado do projeto

## Dispositivos usados

<b>T i m e r</b>	Atualiza o estado do jogo	Interrompe
<b>Teclado</b>	Interfase com o jogo e com o menu	Interrompe
<b>P l a c a Gráfica</b>	Representação de imagens	Não interrompe
<b>R a t o</b>	Não terminado	Interrompe



## Timer

O Timer é utilizado para atualizar o estado do jogo, desenhar os gráficos atualizando as coordenadas do vírus, obstáculos e cursor. Este é o periférico mais relevante no programa pois as suas interrupções permitem o processamento de quase todas as informações. (void timer\_process\_actions)

## Placa Gráfica

A gráfica é o pilar do jogo pois é responsável por desenhar todas as imagens no programa. A resolução de ecrã utilizada é de 1024:768 pixels. A deteção de colisões é feita a partir de um algoritmo em que comparamos as ordenadas do vírus e as ordenadas do topo de cada hospital. (int check\_collision(), sprite.c)

## Teclado

O Teclado é utilizado no menu inicial e no menu how to play para selecionar as múltiplas opções através das teclas “1”, “2”, “3” e “ESC” como explicado anteriormente neste relatório. Aquando o jogo as interrupções do teclado através das teclas “espaço”, “W”, “↑”, “A”, “←”, “D” e “→” são responsáveis por movimentar o vírus. Sendo as primeiras 3 responsáveis por saltar, as duas seguintes por movimentos para a esquerda e as duas últimas por movimentos para a direita. (void kbc\_process\_actions)

## Rato

Este periférico seria utilizado para selecionar as opções nos vários menus através das coordenadas do cursor na tela do programa. A informação sobre o deslocamento do rato é assim usada para atualizar o cursor.

# Estrutura do código

## C o v i d e s c a p e

Este modulo contém as principais estruturas e funções de todo o programa. É onde se realiza a verificação de todas as interrupções dos periféricos recorrendo a estados de máquina e onde o loop principal corre. Este modulo tem 7 funções: **void create\_xpms()** – Responsável por criar todas as imagens utilizadas ao longo do programa.

**void play\_game()** – Esta função contém o loop do programa e as interrupções. Esta função vai chamar as 3 funções a baixo descritas de modo a processar cada ação dos periféricos para que seja possível utilizar o programa.

**void timer\_process\_actions()** – É a função responsável por processar as interrupções do timer. Esta verifica o estado do jogo de modo a saber o que realizar. Estando nos menus esta desenha os menus. Estando no jogo esta vai atualizar as posições dos obstáculos e do vírus no ecrã.

**void kbc\_process\_actions()** – É a função responsável por proces-

sar as interrupções do teclado. Esta verifica o estado do jogo de modo a saber o que realizar. Estando nos menus esta interpreta o input e altera o menu state. Estando no jogo esta vai atualizar a posição do vírus no ecrã vírus no ecrã fazendo com que se desloque para a direita para cima ou que salte.

**void mouse\_process\_actions()** – É a função responsável por processar as interrupções do teclado. Esta interpreta o input e altera o menu state.

**int check\_collision()** – Esta função verifica as colisões do vírus com os obstáculos. A caso haja uma esta muda o menu state e volta para o menu.

**void game\_end()** – Esta função elimina todas imagens presentes no ecrã de modo a que se possa voltar para os menus.

## Keyboard

Adaptado do código do laboratório 3, com algumas modificações.

## P r o j

Este módulo tem como base o código fornecido pelo professor com os devidos acréscimos. É onde se faz a subscrição de todos os periféricos. Se chama a função que create\_xpms responsável por criar todas estruturas gráficas. Também é onde se chama a função play\_game onde o ciclo principal corre. Por fim faz a se para a subscrição de todos os periféricos.

## T i m e r

Adaptado do código das aulas práticas do laboratório 2.



## Mouse

Adaptado do código do laboratório 4, mas comentado por não conseguir ser terminado corretamente.

**18042**

**18254**

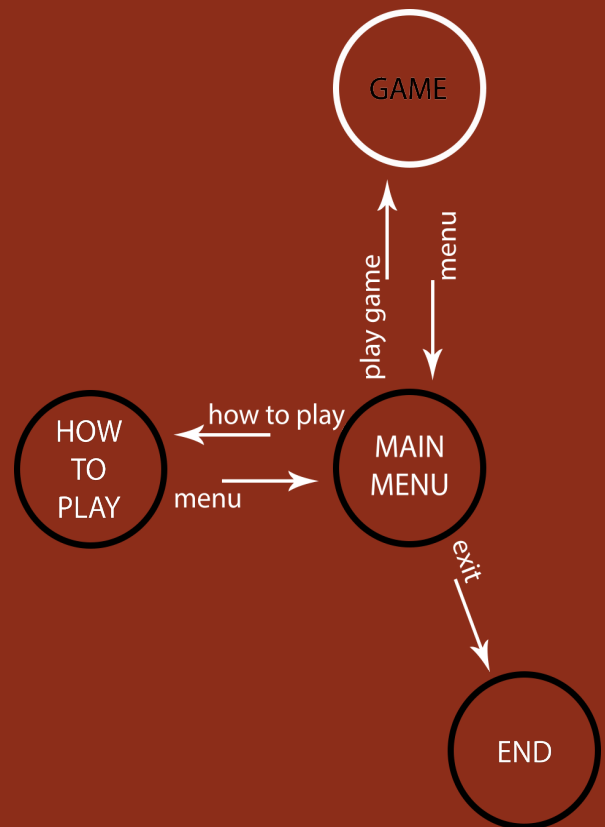
Foram módulos adaptados do código das aulas dos laboratórios 2,3,4 e 5 com modificações.

## Sprite

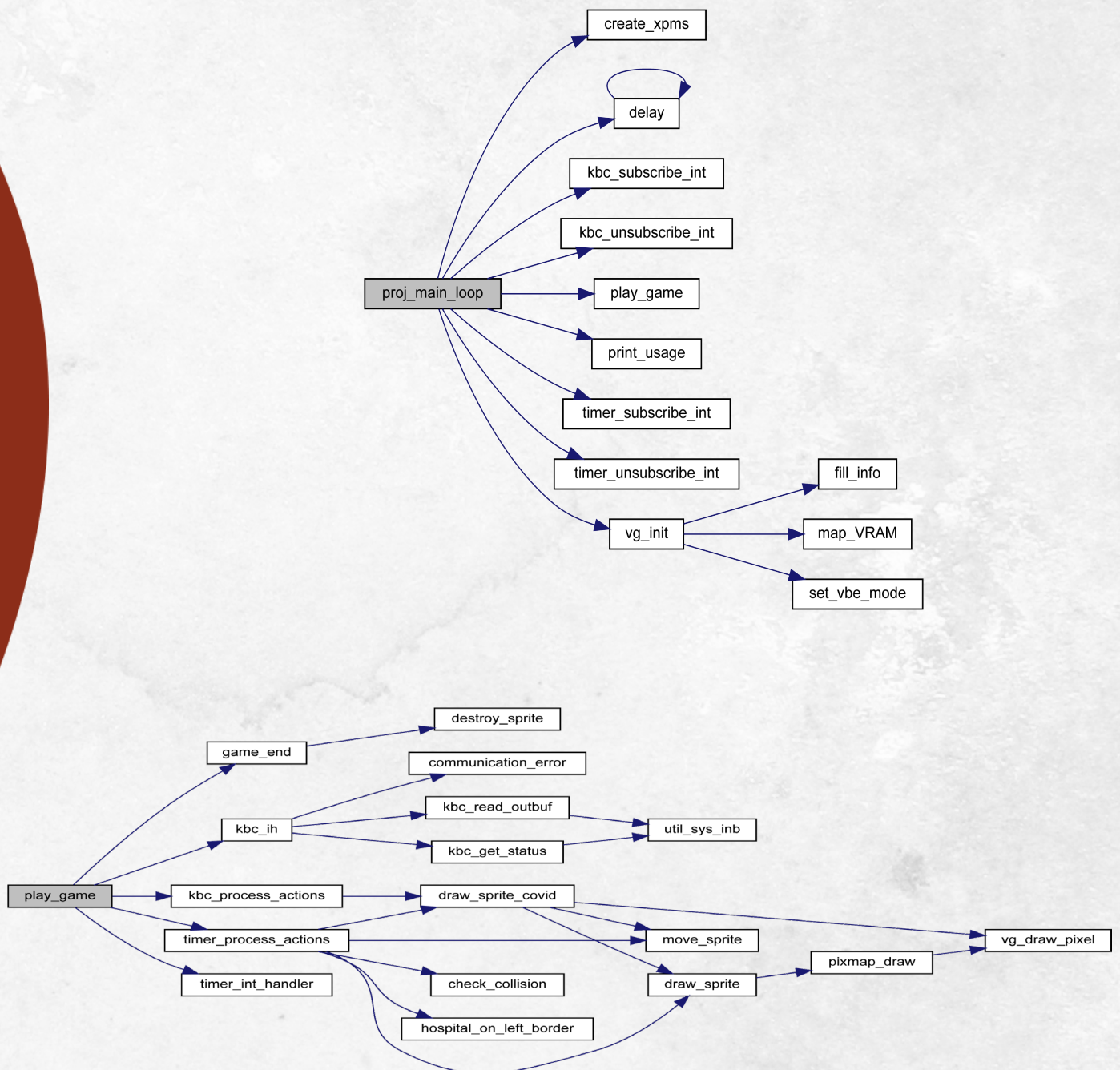
Este módulo faz com que as imagens apareçam no ecrã e move as mesmas através das suas funções. Assim em conjunto com as funções criadas no lab5 este módulo inicializa as coordenadas das sprites e altera as mesmas. Ações essenciais do jogo só são possíveis graças a este módulo. Como a utilização do rato onde verificamos a localização do cursor no ecrã e a verificação de colisões que funciona vendo as coordenadas do vírus e dos obstáculos. Os obstáculos são selecionados por disponibilidade no módulo covid escape, mas só é possível o seu desenho e destruição do display a partir dos métodos em sprite. Assim com recurso a este módulo foi possível criar as seguintes estruturas: how\_to\_playSP, backgroundSP, background\_leftSP, main\_menuSP e hospitals cada uma contendo as coordenadas, a velocidade de deslocamento e o pixmap de cada um.

## Estado da máquina

Apesar deste se encontrar dentro do módulo covid escape devido a sua relevância é aqui mencionado isoladamente. As duas máquinas de estado são fundamentais para o desenvolvimento do jogo. Estas permitem saber que tipo de inputs esperar dos periféricos e que tipo de ações fazer a cada interrupção. É o que permite as passagens entre menus e jogo. O curr\_case é em que estado o programa se encontra. Este pode estar em MENU ou em IN\_GAME. O menu1 é o estado de seleção de menu que o programa se encontra. Podendo se encontrar em IN\_HOW\_TO\_PLAY, PLAY\_GAME, IN\_MENU, EXIT\_GAME o que fará com que transite para as diferentes fases do programa.



# Call graph





# Detalhes e Conclusões

## Detalhes

A implementação do nosso código foi feita por camada de modo a facilitar a organização e leitura do mesmo. Por este motivo a escrita de funções de maior complexidade acabou por se simplificar quer na escrita quer no raciocínio sobre as mesmas. Outra mais valia para a fluidez do código foi a criação de duas máquinas de estado que, apesar de não estarem em módulos independentes são extramente relevantes como já mencionado. Tal como as máquinas de estado outras estruturas criadas que guardam a informação sobre vários objetos permitiram a comunicação entre as varias funções e módulos do programa mantendo-se a coerência. Por fim o algoritmo criado para verificação de colisões foi feito de modo a comparar as ordenadas apenas quando a abcissa de um obstáculo se encontra a menos de 70 pixéis de vírus de modo a que não compara com todos os obstáculos presentes do ecrã. Deste modo diminuámos o número de comparações  $n/2 + 1$  aumentando a eficiência do algoritmo.

## C o n - clusões

Devido á situação de pandemia que vivemos sentimos que as aulas também sofreram com a situação e que ainda não houve tempo de se adptar o ensino totalmente. Segundamente sentimos que as colisões e estruturação correta do código foi onde tivemos mais dificuldades. Por fim como se deve ter notado não foi especificado quem fez mais em cada parte do código pois o projeto em si foi escrito em conjunto fazendo video chamadas. O código usado nos labs foi feito metade pelo Gabriel e metade pelo Carlos pois devido a situação exepcional do nosso grupo em que ambos nos juntamos só para o projeto pela desistência dos nossos colegas tínhamos ambos partes de quase todos os labs.



