U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

!Snake

Laboratório de Computadoes

LCOM - T7G09

Elaborado por:

- **Nuno Resende** – up201806825
- **Filipe Recharte** – up201806743

Indice

IV. Organização/Estrutura do código

i. proj.c
ii. game.c
iii. snake.c
iv. fruit.c
v. leaderboard.c
vi. utils.c
vii. timer.c
viii. keyboard.c
ix. mouse.c
x. videocard.c

V. Detalhe de Implementação

i. Event Driven Code and State Machine
ii. Orientação a Objetosobjetos
iii. Geração de Frames
iv. Geração de frames
v. Leaderboard
vi. Colisões

VI. Conclusões.

Introdução

No âmbito da unidade curricular de Laboratório de Computadores (LCOM), prepararmos para o nosso projeto final um jogo, de forma a possibilitar a demonstração do conhecimento que adquirimos ao longo das aulas.

Tendo isto em conta, o que apresentamos é o **!SNAKE**, uma variação do famoso jogo que os antigos telemóveis da Nokia continham. Aqui o objetivo é **sobreviver o máximo tempo possível**, sendo que a dificuldade consiste no facto da cobra aumentar de tamanho conforme o tempo e que o período em que os frutos são gerados também aumenta.

Se a cobra colidir com um fruto, uma parede ou consigo mesma, o jogo termina e o jogador tem a hipótese de introduzir o seu nome e ser colocado na *leaderboard.*

1. Instruções de utilização

MENU

O jogo inicia com um menu inicial (fig.1) que permite ao utilizador navegar com o rato e escolher uma das três opções (“PLAY”, “LEADERBOARD”, “EXIT”). Quando o rato passa sobre alguma opção, esta é destacada sendo percetível a possibilidade de clicar no botão esquerdo do rato para navegar para o menu desejado.

Figura 1

LEADERBOARD

Quando o utilizador seleciona a opção leaderboard são mostrados os 10 melhores jogadores de sempre por ordem decrescente de score. Para cada jogador é registado o nome inserido, o score obtido, a data (dd mm aaaa) e a hora (hh mm ss) em que o jogo ocorreu.

Para voltar para o menu inicial o utilizador tem apenas de pressionar a tecla **ESC**. A figura abaixo (fig. 2) mostra o aspeto da leaderboard.

Figura 2

PLAY

Quando o utilizador seleciona a opção Play é direcionado para o ambiente de jogo onde é mostrada a frase “DODGE THE FRUITS” indicando que deve afastar-se dos frutos para não perder o jogo, ao contrário do jogo original.

Para começar a jogar o utilizador pode carregar em qualquer uma das setas do teclado (“Press any arrow key to start”).

Figura 3

GAME PLAY

Após clicar em alguma seta o jogo começa, e o utilizador tem de se desviar dos frutos, para além disso tem de se desviar das paredes e não pode colidir consigo mesma (exemplos nas figuras abaixo).

Enquanto a cobra não colidir com nenhuma das coisas referidas acima o score vai aumentando ao longo do tempo, sendo este mostrado quando o utilizador perde. Quando é mostrado o "GAME OVER!" pressionando a tecla ESC o utilizador é redirecionado para o input do nome do jogador.

Durante o jogo a cobra vai progressivamente aumentado de tamanho, e os frutos vão sendo dispostos de forma aleatória no ecrã. Assim, a dificuldade do jogo aumenta á medida que o tempo vai passando.

O jogador tem também a possibilidade de, durante o jogo, mexer com o rato e carregar num fruto fazendo com que esse mesmo fruto mude de posição.

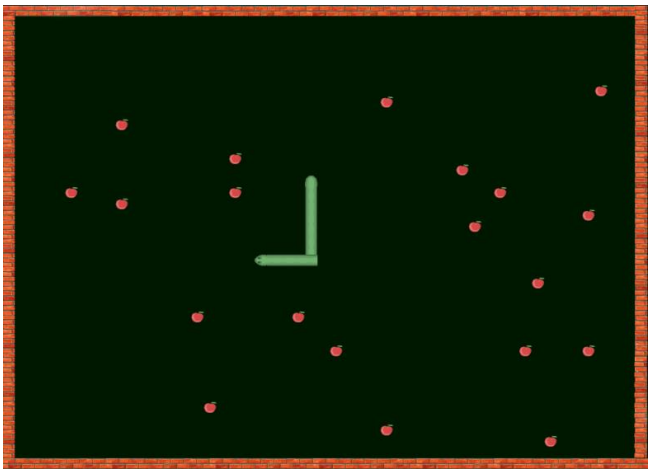


Figura 4

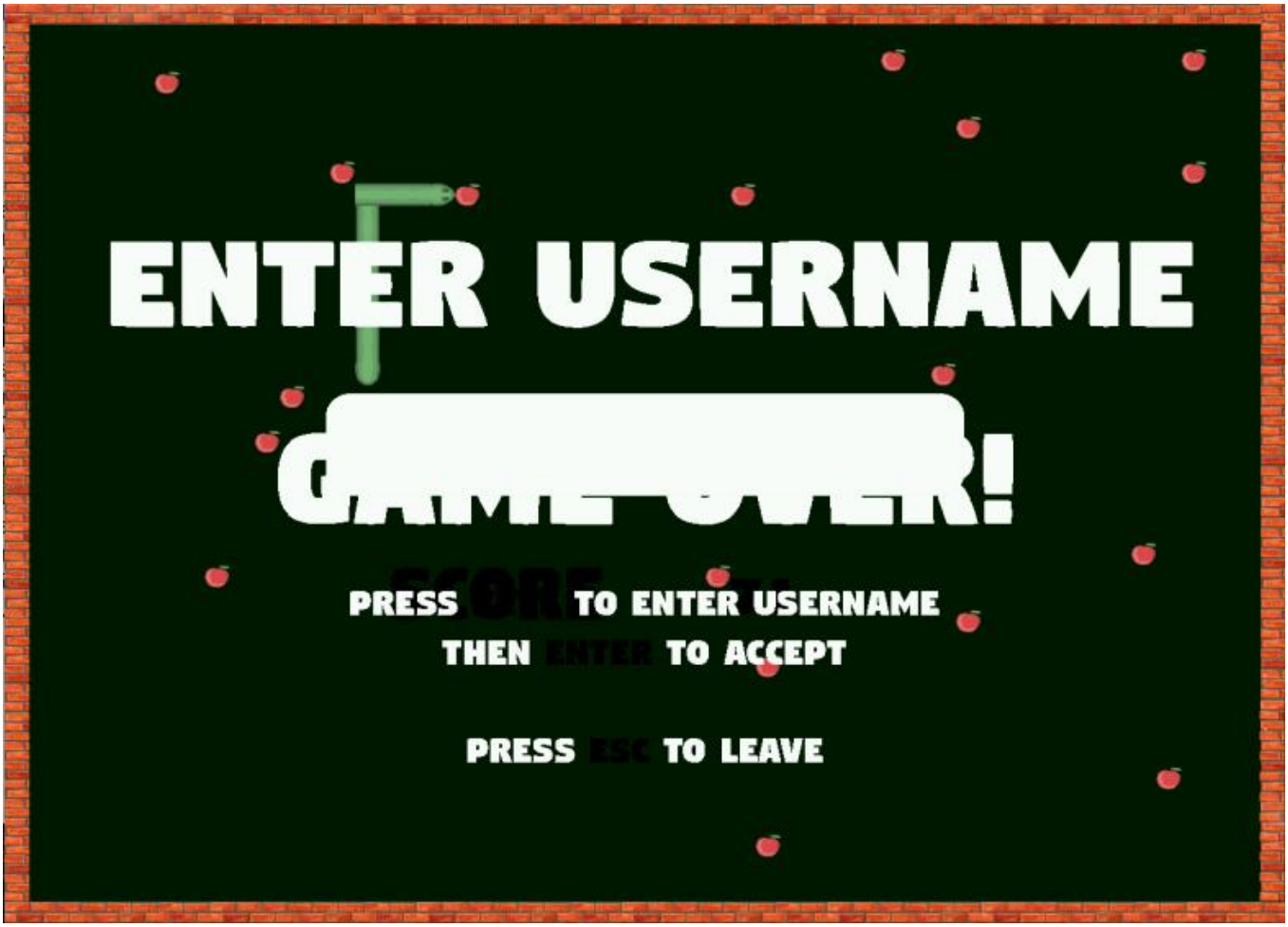
Figura 5

Figura 6

Figura 7

USERNAME

No input do username do jogador, após ter perdido o jogo, é possível escrever o nome e em seguida pressionar ENTER se o jogador quiser ter o score registado na leaderboard (caso tenha maior score que algum dos 10 jogadores já registados), caso contrário pode pressionar a tecla ESC e é redirecionado para o menu principal, nenhum registo do jogo é guardado.



2. Funcionalidades

Dispositivo	Funcionalidade	Interrupções
Timer	Controla pontuações, avança posição da cobra, controla geração dos frutos	Y
Keyboard	Controlo da direção da cobra e sair dos menus (ESC)	Y
Mouse	Navegação no menu principal, nova posição dos frutos dentro do jogo	Y
Video card	Toda a apresentação de informação - frutos, cobra com movimento, pontuações, etc.	N
RTC	Leitura da data para apresentar nas pontuações e escrever no ficheiro	N

Video card

O modo gráfico que estamos a inicializar é o modo **0x11A**, ou seja, um modo que tem a resolução **1280x1024**, onde as cores são diretas e há **16 (5:6:5)** bits por pixel.

Para suavizar as animações e não haver lentidão com o movimento do rato, por exemplo, utilizamos ***double buffering***, *triple buffering* não foi necessário.

Todo o tipo de animações, imagens, o cursor do rato e os caracteres (font) são, naturalmente, responsabilidade da placa gráfica.

As funções relacionadas com a placa gráfica são funções implementadas no lab 5. As funções que interagem diretamente com o *frame buffer* são a **fillPixel()** que pinta um determinado pixel e a **clear()** que apaga todo o conteúdo do frame buffer. As outras funções, por exemplo, o **draw_xpm()**, pintam pixéis no *double buffer* e só depois são carregados para a placa gráfica.

A única função da VBE usada foi a **vg_init()** que inicializa o modo gráfico 0x11A.

Keyboard

O teclado tem no nosso projeto a função de **sair dos** vários **menus** com a tecla "ESC", **escrever o nome** do jogador e **cancelar o *input*** do nome para a pontuação e o c**ontrolo do movimento** da cobra.

Na escrita do nome do jogador, a "**SPACE**" deixa um espaço em branco, a tecla "**BACKSPACE**" apaga um caracter e o resto das letras e números escrevem o caracter respetivo. A tecla "**ENTER**" é a tecla que o utilizador deve premir para aceitar o nome que aparece no ecrã.

No jogo, o utilizador move a cobra (ou a cabeça da cobra, especificamente) com as **setas.**

O teclado funciona com interrupções e, portanto, precisamos de usar as funções **keyboard_subscribe_int()** e **keyboard_unsubscribe_int()** e as restantes funções desenvolvidas no lab 3.

Mouse

O rato tem a função de navegação entre menus, ou seja, com o movimento do rato e com os botões de *click* do lado esquerdo quando por cima de uma das opções e ainda para selecionar frutos se o jogador desejar no jogo, que faz com que o fruto passe a ter uma nova posição aleatória no jogo. Quando o rato passa por cima de uma das opções, essa opção fica *highlighted.*

O rato funciona com interrupções e, portanto, precisamos de usar as funções **mouse_subscribe_int()** e **mouse_unsubscribe_int()** e as restantes funções desenvolvidas no lab 4.

Para além dessas desenvolvemos uma função **updateMousePosition()** que possibilita a sincronização entre o rato e a imagem do cursor.

Timer

O timer tem a função de **atualizar** o **estado de jogo**.

É no timer que tem a **máquina de estados** propriamente e é lá que estão as condições de jogo. É o timer que permite a nova geração dos **frutos**, o **movimento** da cobra, a contagem das **pontuações** e tudo o que avançar no tempo.

O timer funciona com interrupções e, portanto, precisamos de usar as funções **timer_subscribe_int()** e **timer_unsubscribe_int()** e as restantes funções desenvolvidas no lab 2.

Precisámos também de alterar a **frequência** do timer para 1 de modo a que a cobra não se movesse demasiado rapidamente.

RTC

O relógio interno foi somente usado para extrair a data atual de forma a poder escrevê-la no ficheiro que guarda as pontuações e apresentá-la no menu das pontuações.

As funções que interagem diretamente com o RTC são a **rtc_read_reg_data(**) e as **rtc_read_xxxx()** que leem um determinado elemento da data e da hora.

Não foram usadas interrupções.

UART

Não implementado.

3. Organização/Estrutura do código

proj.c

Neste módulo está implementado o ciclo principal do jogo, é onde fazemos as subscrições das interrupções que são também aqui recebidas de todos os periféricos necessários ao controlo do jogo (timer/keyboard/mouse), além disso é onde está incluída também a máquina de estados.

Aqui são carregadas as principais imagens necessárias para o jogo, é lido e escrito o ficheiro da *leaderboard*, e são inicializadas as variáveis precisas no decorrer do jogo.

Ambos os elementos do grupo trabalharam neste módulo uma vez que inclui todos os estados do jogo, chamando as funções necessárias a cada um deles.

O peso do módulo no projeto é de **30%**.

game.c

Neste módulo são carregadas as imagens do abecedário e dos números. Aqui está as funções onde é feito o *handling* das interrupções do mouse e do teclado que são essenciais para a interação do utilizador com o jogo nomeadamente a mudança de estados e o jogo em si. É neste modulo também que está implementada a função que nos permite escrever texto para o ecrã, essencial para a *leaderboard* e para o input do *username*.

Os dois elementos implementaram funções neste modulo.

O peso do módulo no projeto é de **25%**.

snake.c

Este módulo é responsável pela criação da *snake* e pelo seu movimento. Aqui também é verificada a colisão da *snake* com a parede e consigo mesma, além disso é onde está a função que permite mostrar a *snake* no ecrã nas suas devidas posições. É onde também se inicializa a matriz onde a *snake* se irá mover no decorrer do jogo.

Há duas estruturas de dados implementadas neste modulo que são a Tile e a *Snake*. A Tile é composta pela informação de cada quadrado da matriz que indica se naquele quadrado está a cabeça da *snake*, a cauda ou o corpo, contém também a informação da direção da *snake* naquele quadrado. A *snake* é composta pelas coordenadas x e y que são a posição na matriz, o seu tamanho e a sua direção.

Membro responsável: Filipe Teixeira

O peso do módulo no projeto é de **15%**.

fruit.c

Neste módulo é essencialmente feito o *spawn* dos frutos mostrados ao longo do jogo nas devidas coordenadas, além disso verifica a colisão entre a *snak*e e os frutos.

A principal estrutura de dados usada neste módulo é o fruto que contem as coordenadas onde este deve aparecer no ecrã.

Membro Responsável: Nuno Resende.

O peso do módulo no projeto é de **15%**.

leaderboard.c

Neste módulo foram implementadas todas as funções necessárias para a existência da *leaderboard*. É onde está implementada a função de ler e de escrever para o ficheiro da leaderboard. Está também implementada a função que insere um novo jogador na tabela permitindo que esta se mantenha sempre em ordem decrescente de score.

Ambos os elementos do grupo trabalharam neste módulo.

O peso do módulo no projeto é de **8%.**

utils.c

Neste módulo foram implementadas funções que nos facilitaram a implementação dos labs ao longo do semestre. Para o projeto implementamos

duas novas funções que servem para a obtenção da data que é obtida a partir do rtc.

Ambos os elementos do grupo trabalharam neste módulo.

O peso do módulo no projeto é de **2%.**

timer.c

Neste módulo encontram-se as funções necessárias á manipulação do timer através dos registos, desenvolvido nas aulas laboratoriais definidas para o mesmo (Laboratorial 2). Tendo sido realizado igualmente por ambos os elementos do grupo.

keyboard.c

Neste módulo encontram-se as funções necessárias á manipulação do *keyboard* através dos registos, desenvolvido nas aulas laboratoriais definidas para o mesmo (Laboratorial 3). Tendo sido realizado igualmente por ambos os elementos do grupo.

mouse.c

Neste módulo encontram-se as funções necessárias á manipulação do mouse através dos registos, desenvolvido nas aulas laboratorias definidas para o mesmo (Laboratorial 4). Tendo sido realizado igualmente por ambos os elementos do grupo.

videocard.c

Neste módulo encontram-se as funções necessárias á manipulação do *keyboard* através dos registos, desenvolvido nas aulas laboratoriais definidas para o mesmo (Laboratorial 4). Tendo sido realizado igualmente por ambos os elementos do grupo.

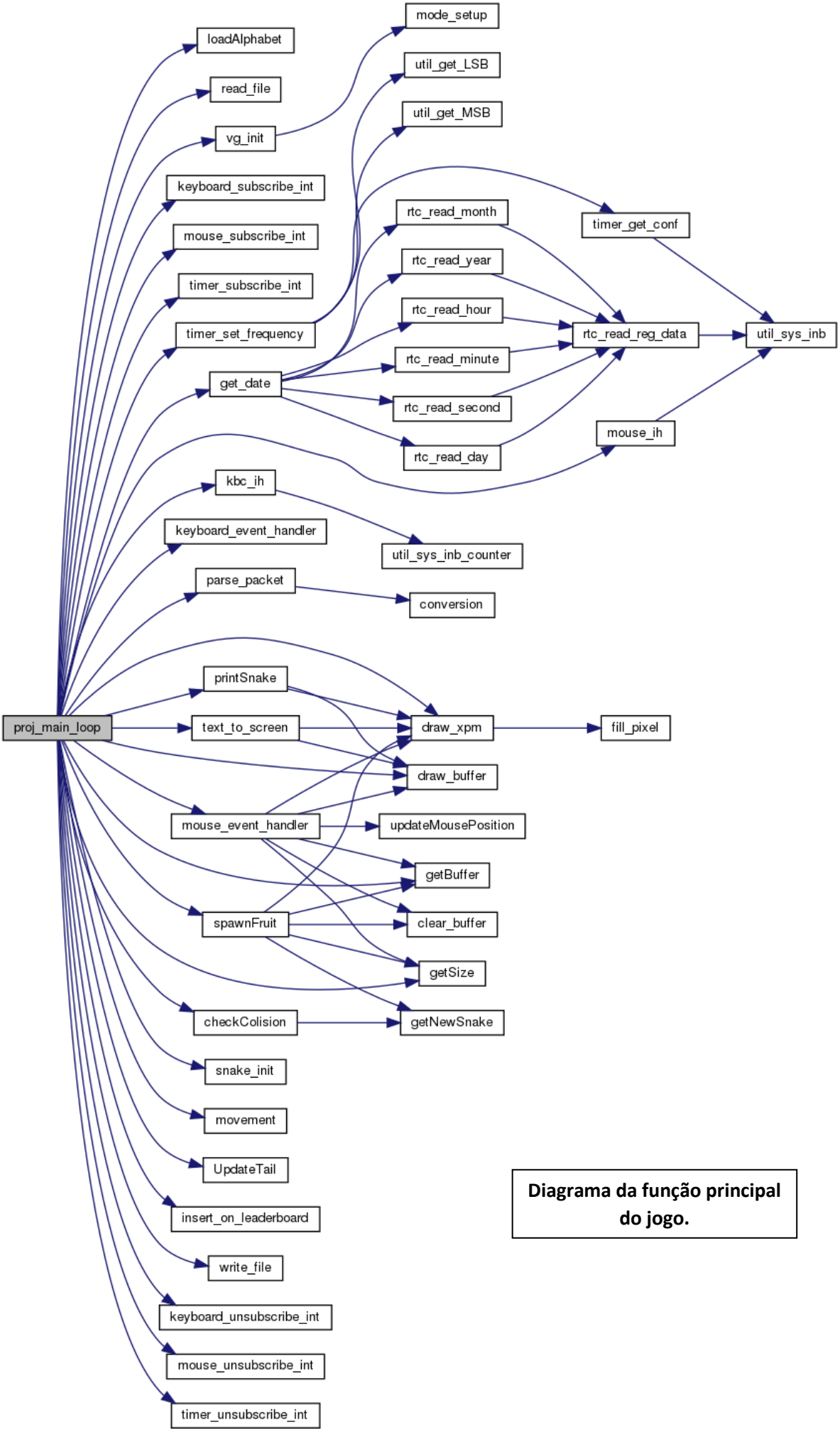


Diagrama da função principal do jogo.

4. Detalhes da Implementação

Event Driven Code and State Machine

A estrutura do nosso código é precisamente uma máquina de estados. Criámos um **enum** com os seguintes possíveis estados: MAIN_MENU, PLAY_HIGHLIGHT, LEADERBOARD_HIGHLIGHT, EXIT_HIGHLIGHT, PLAY_BEFORE_KEY, PLAY, GAME_OVER, PAUSE_MENU, EXIT, DODGE E USERNAME.

Cada um destes tem uma **função específica** no projeto e a transição entre estados é provocada pelas decisões do utilizador e os **inputs** dos respetivos periféricos.

Por exemplo, a partir do **MAIN_MENU**, o primeiro menu apresentado ao utilizador, pode se aceder ao modo de jogo, que passaria ao modo **PLAY_HIGHLIGHT** e depois com um *click* no botão esquerdo do rato passaria ao estado **PLAY_BEFORE_KEY**, o estado que pede ao utilizador um input nas setas do teclado para fazer a cobra mexer, agora no estado **PLAY**.

A **LEADERBOARD_HIGHLIGHT** e o **EXIT_HIGHLIGHT** funcionam semelhantemente ao **PLAY_HIGHLIGHT**, o **GAME_OVER** é o estado que indica ao utilizador que perdeu o jogo, o **PAUSE_MENU** é o menu de pausa, o **DODGE** é o estado que avisa o utilizador a desviar-se dos frutos e o **USERNAME** é o estado que pergunta o nome do utilizador para colocar na *leaderboard.*

O **EXIT** é o estado que faz o programa terminar.

Orientação a Objetos

O nosso programa não é diretamente "orientado a objetos".

Em certas situações foi útil construir ***structs*** (e não classes) como o **Fruit** que guarda as coordenadas dos frutos, a **Snake** que guarda a posição, o tamanho e a

direção da cobra, a **GameScore** que guarda o nome do utilizador, a sua pontuação e a data em que terminou aquele jogo, etc.

Geração de frames

Como dito antes, o movimento das imagens e a contagem dos pontos funciona com as interrupções do timer.

No jogo, a cada 2 interrupções do timer, a **pontuação aumenta** 1.

No jogo, a cada 60 interrupções do timer, a **cauda aumenta** 1 bloco

No jogo, a cada 18000 interrupções do timer, o tempo que demora até novos **frutos aparecerem diminui**

No jogo, a cada 60 interrupções do timer, a **cauda aumenta** 1 bloco

Leaderboard

Para dar alguma competitividade ao jogo decidimos implementar um **sistema de pontuações.**

O que acontece é que o utilizador vai acumulando pontos (que são aumentados a cada **2 interrupções** do timer) e no final a sua pontuação é a soma desses pontos todos. A pontuação total é mostrada assim que o utilizador perde.

Estamos a guardar as **melhores 10 pontuações** - ordenadas em forma decrescente – e se houver uma pontuação superior a outra, é substituída.

Com o fim de preservar estes dados, estamos a escrevê-los no ficheiro **leaderboard_scores.txt** que tem o formato PONTOS NOME DIA MES ANO HORA MINUTOS SEGUNDOS

Estes dados são carregados no programa, assim que este inicia.

Colisões

A nossa deteção de colisão **não** pertence à placa gráfica e foram detetadas da seguinte forma:

Criámos uma **matriz virtual** onde a cobra se movimenta e onde os frutos são gerados.

As colisões ocorrem quando na matriz de jogo houver uma **sobreposição das posições** dos vários objetos, ou seja, a sobreposição da **cabeça** da cobra com alguma parte do seu **corpo** e a sobreposição da **cabeça** com **algum fruto** ou com alguma **parede**

Para evitar uma geração injusta no jogo, os frutos não podem nascer nem diretamente à frente da cobra, nem perto da cabeça.

5. Conclusões

Ambos achamos que foi uma unidade curricular árdua e trabalhosa.

Para além dos labs serem bastante longos, o que dificulta o desenvolvimento dos labs, é a falta de organização, no geral, da unidade curricular. É complicado encontrar a informação necessária porque está dispersa.

Apesar disto, o facto de haver uma documentação no doxygen é muito útil e o facto de haver um site próprio para todas as informações também é muito positivo.

Outro grande choque é o facto de nunca termos tido uma experiência de desenvolvimento de um projeto numa escala grande (comparado ao que estamos habituados) e ter de implementar algo com desenhos, fora da consola, pode ser avassalador.

O projeto, no entanto, foi um desafio interessante onde pudemos ganhar experiência e o processo de *debugging* não era tão maçador como nos labs.

Aprender sobre cada dispositivo I/O individualmente também é interessante, apesar de não fazer sentido o apoio aos dispositivos RTC e UART ser praticamente nulo.