

Relatório

TypeRacer

Grupo 8 – Turma 6

Laboratório de Computadores



Luísa Maria Mesquita Correia

up201704805

Henrique Ribeiro Nunes

up201906852

Índice

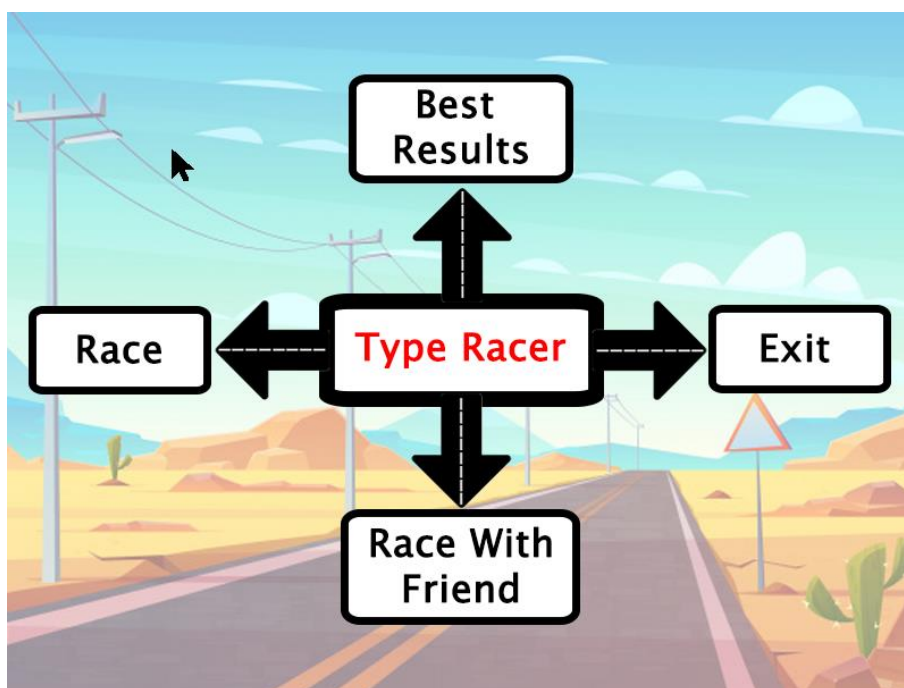
1. Instruções
 - 1.1. Modo de corrida singular (opção “Race”)
 - 1.2. Melhores resultados (opção “Best Results”)
 - 1.3. Resultados (após a corrida)
2. Estado do projecto
 - 2.1. timer
 - 2.2. keyboard
 - 2.3. mouse
 - 2.4. graphic
 - 2.5. rtc
3. Organização/estrutura do código
 - 3.1. Módulos
 - 3.1.1. proj
 - 3.1.2. menus
 - 3.1.3. race
 - 3.1.4. results
 - 3.1.5. best_results
 - 3.1.6. Chars
 - 3.1.7. sprite
 - 3.1.8. utils
 - 3.1.9. xpm's
 - 3.2. Doxygen
4. Detalhes de implementação
5. Conclusões

1. Instruções

1.1. Menu

Ao iniciar o programa, é apresentado o menu do jogo. Com o cursor do rato e um clique no botão esquerdo do rato ou com as teclas de direção, é possível escolher uma das seguintes opções:

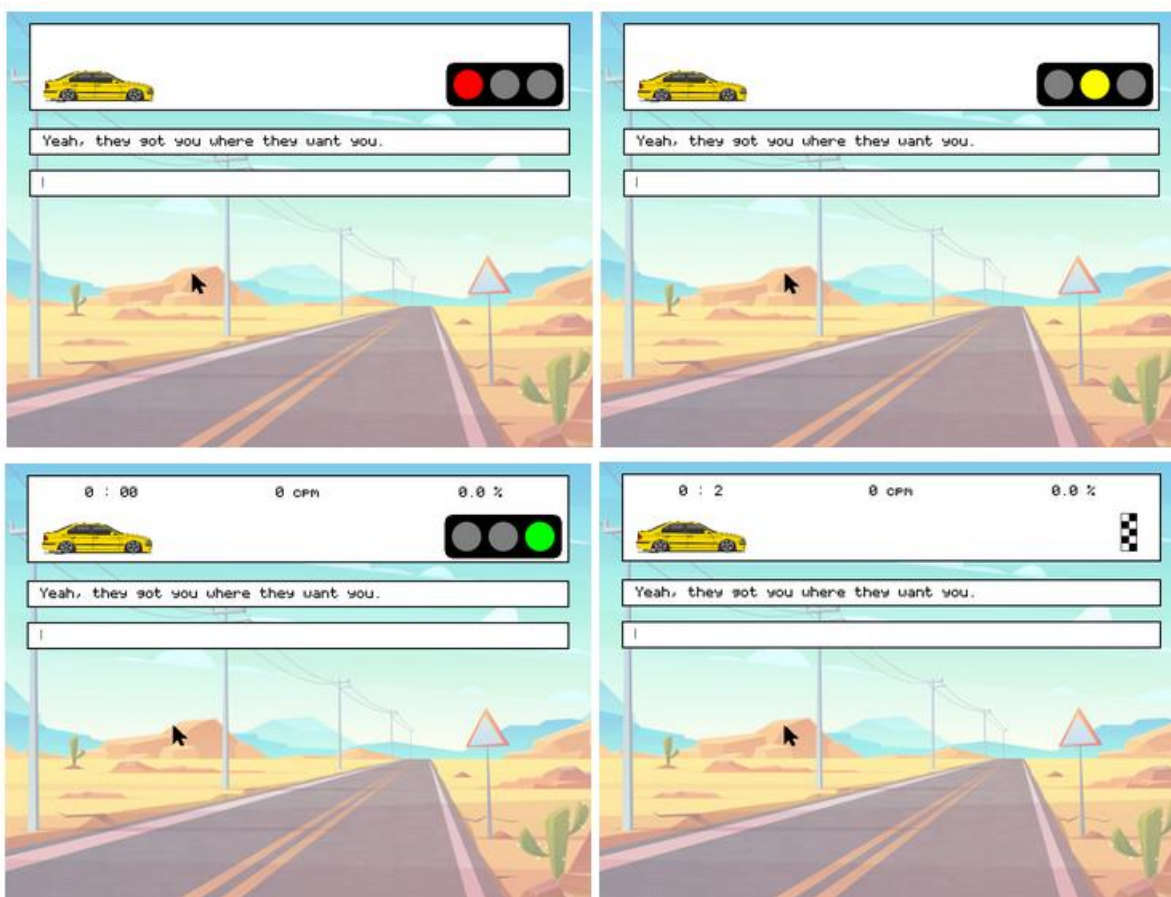
- **Race:** Inicia a corrida em modo singular.
- **Race with friend:** Inicia a corrida em modo competitivo (não implementado).
- **Best results:** Mostra os três melhores resultados obtidos no jogo.
- **Exit:** Fecha o programa.



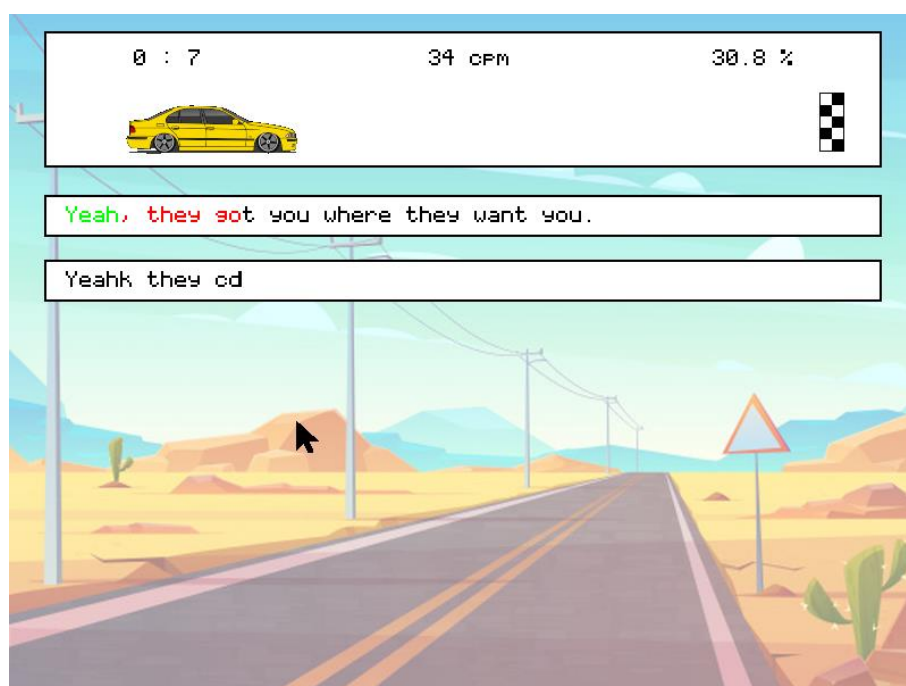
Menu do jogo

1.2. Modo de corrida singular (opção “Race”)

A corrida é iniciada quando o semáforo fica com a luz verde (ao fim de 2 segundos). O utilizador deve então escrever o que está na caixa de texto o mais depressa possível e sem erros. Esse texto é gerado de forma aleatória (ver seção 4). As letras copiadas corretamente ficam verdes à medida que o utilizador as insere. Caso cometa um erro, a letra fica a vermelho, como também todo o texto inserido de seguida até que o utilizador corrija. É possível também o utilizador movimentar o cursor de texto com as setas de direção e também com o rato, permitindo assim alterar secções intermédias do texto digitado. O jogo acaba quando o utilizador acaba de escrever o texto todo corretamente, quando é desqualificado (ver critérios de desqualificação na seção 4) ou se desistir - clicando no botão ESC. Durante o jogo, é possível ver em tempo real o seu desempenho: o tempo decorrido, a velocidade em caracteres por minuto e a sua precisão. O carro vai também movendo-se à medida que o utilizador escreve.



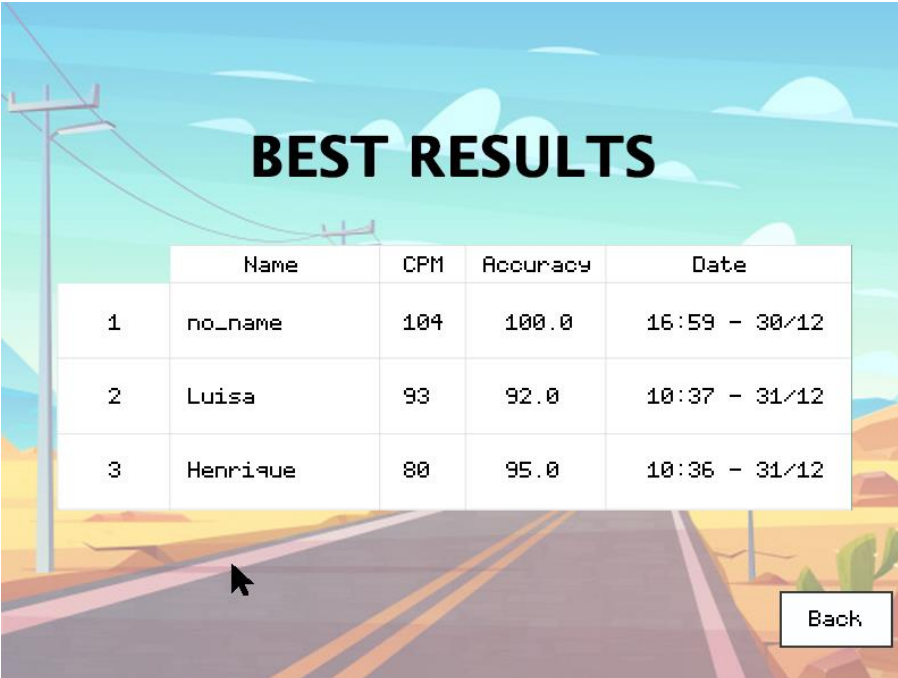
Sequência de imagens (da esquerda para a direita e de cima para baixo) a partir do momento em que o utilizador escolhe a opção "Race".



Exemplo de uma corrida em que o utilizador comete um erro.

1.3. Melhores resultados (opção “Best results”)

Nesta página são apresentados os 3 melhores resultados obtidos no jogo, segundo a velocidade (em caracteres por minuto) obtida. Em caso de empate, avalia-se a precisão (em percentagem). O utilizador pode regressar ao menu, clicando com o rato na opção “Back” ou carregando na tecla ESC.

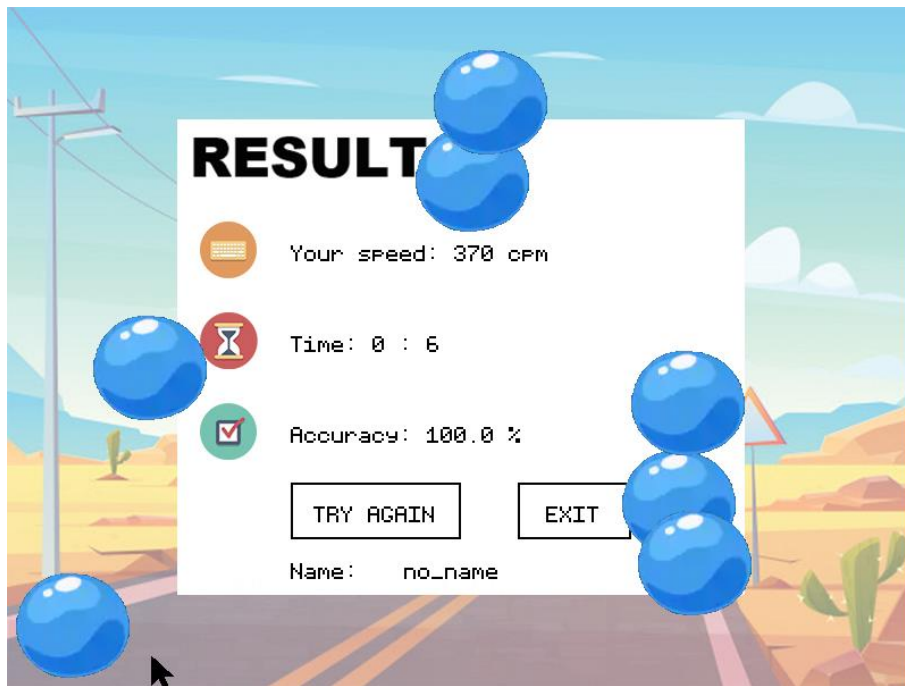


	Name	CPM	Accuracy	Date
1	no_name	104	100.0	16:59 - 30/12
2	Luisa	93	92.0	10:37 - 31/12
3	Henrique	80	95.0	10:36 - 31/12

Exemplo de uma página com os 3 melhores resultados.

1.4. Resultados (após a corrida)

Após a corrida, mostra-se ao utilizador os resultados obtidos: a sua velocidade em caracteres por minuto, o tempo que demorou a escrever e a sua precisão em percentagem (100% se escreveu tudo bem sem cometer erros). Para além dos resultados, o utilizador vê várias bolhas a flutuar no ecrã que pode arrebentar com quaisquer dos botões do rato. O nome por definição é “no_name”, no entanto é possível alterá-lo fazendo uso do teclado, apagando o nome atual e escrevendo o desejado. Na corrida seguinte será mostrado o último nome utilizado. Em caso de desqualificação, tanto a velocidade como a precisão ficam a 0 e o nome é definido como “Disqualified”. Caso o utilizador tenha uma pontuação alta, superior a outras já obtidas, o seu resultado será guardado automaticamente ao selecionar qualquer uma das opções. Tal pode ser feito com o rato ou com as teclas direcionais (Esquerda para selecionar a opção “Try again” e direita para selecionar “Exit”).



Exemplo de uma página de resultados.

2. Estado do projeto

Dispositivo	Uso	Modo
Timer	Controlar o <i>frame rate</i> , contar o tempo	Com interrupções
Teclado	Selecionar opções, escrita do texto, mover o cursor na caixa de texto, escrita do nome	Com interrupções
Rato	Selecionar opções, mover o cursor na caixa de texto, rebentar as bolhas	Com interrupções
Placa gráfica	<i>Design</i> das páginas	Modo 115, double buffering com <i>page flipping</i>
RTC	Obter data e hora, alarme	Com interrupções
Porta-série	Não implementado	-

2.1. Timer

Ficheiros: timer.c, timer.h, i8254.h, utils.h, utils.c

O timer 0 foi usado para controlar o *frame rate*, gerando um novo *frame* a cada duas interrupções, e o momento de atualização da página mostrada no ecrã. Para além disso, é usado também para contar o tempo decorrido na corrida e permitir assim calcular os resultados.

As interrupções do rato são tratadas a partir da função *timer_ih* chamada no *driver_receive loop* que incrementa o contador global de interrupções do timer.

Todas as funções propostas no enunciado do lab foram implementadas. No entanto, só algumas é que foram usadas no projecto:

- *timer_subscribe_int*
- *timer_unsubscribe_int*
- *timer_ih*

2.2. Teclado

Ficheiros: keyboard.c, keyboard.h, i8042.h

O teclado foi usado para gerar interrupções ao carregar e largar teclas. Para selecionar opções, tanto na página do menu como na página dos resultados, podem ser usadas as teclas direcionais. Na página do jogo, o teclado é usado para escrever o texto e para movimentar o cursor na caixa de texto. E na página dos resultados, é usada para escrever o nome.

As interrupções do teclado são tratadas a partir da função *kbd_ih* chamada no *driver_receive loop* e quando todos os bytes do *scancode* forem lidos são montados na função *assemble_scancode*.

Todas as funções propostas no enunciado do lab foram implementadas. No entanto, só algumas foram utilizadas no projecto:

- *kbd_subscribe_int*
- *kbd_unsubscribe_int*
- *kbd_ih*
- *assemble_scancode*

2.3. Rato

Ficheiros: *mouse.c*, *mouse.h*, *i8042.h*

O rato foi usado para gerar interrupções no movimento e cliques do rato. Tanto no menu, na página de melhores resultados e na página dos resultados, o rato pode ser usado para seleccionar opções. O rato é ainda usado na página dos resultados, após a corrida, para rebentar bolhas que aparecem no ecrã e na página da corrida para movimentar o cursor de texto, usando o botão esquerdo.

As interrupções do rato são tratadas a partir da função *mouse_ih* chamada no *driver_receive loop* e quando todos os bytes do *packet* forem lidos são montados na função *assemble_packet* e posteriormente interpretados na função *mouse_events*, tornando a leitura dos seus eventos mais simples.

Todas as funções propostas no enunciado do lab foram implementadas. No entanto, só algumas foram utilizadas no projecto:

- *mouse_subscribe_int*
- *mouse_unsubscribe_int*
- *mouse_ih*
- *assemble_packet*
- *mouse_events*

2.4. Placa gráfica

Ficheiros: *graphic.h*, *graphic.c*, *vbe.h*

A placa gráfica foi usada para exibir tudo no ecrã: letras, páginas, bolhas, entre outros (tudo definido em headers na pasta **xpm**). Para isso foi utilizado *double buffering* com *page flipping*: foi alocado na *video RAM* o dobro do espaço necessário para desenhar o ecrã que pode ser visto como duas páginas. Foi declarado um apontador (*fr_buffer*) que tem por objetivo apontar alternadamente para o início de cada página. Através da função 07h da VBE é possível escolher qual das páginas é desejada que apareça no ecrã. Desta forma, enquanto a primeira página é escrita, a página mostrada no ecrã é a segunda e vice-versa. A página mostrada no ecrã é alternada a cada duas interrupções do timer.

Todas as funções propostas no enunciado do lab foram implementadas. Para o projeto foram necessárias algumas adaptações e novas funções. Todas as funções deste módulo foram usadas à exceção de *graphic_cntrl_info*.

2.5. RTC

Ficheiros: *rtc.c*, *rtc.h*, *utils.c*, *utils.h*

O RTC foi usado para obter a data e a hora momentânea que é utilizada para registar mais informação sobre os melhores resultados. Para além disso foi usado também para gerar alarmes através de interrupções, nomeadamente o alarme para a desqualificação de um jogador na corrida. Os registos de alarme são atualizados com o tempo máximo de corrida, calculado previamente (ver secção 4), e quando atinge esse tempo a corrida acaba.

As interrupções do RTC são tratadas a partir da função *rtc_ih* chamada no *driver_receive loop* que por sua vez lê de onde é proveniente a interrupção e lida de acordo. No caso, apenas foi implementado o *handler* das interrupções de alarme (*handle_alarm_int*) que altera o indicador de ocorrência de um alarme.

Praticamente todas as funções deste módulo foram usadas no projecto, já que o seu propósito era precisamente esse. Apenas a função *rtc_read_day_week* não foi usada, dado que não foi necessário.

3. Organização/estrutura do código

O código está organizado em três pastas:

- **headers** que contém todos os headers
- **src** que contém a implementação do código declarado no seu header correspondente
- **xpm** que contém headers com os vários xpm's usados

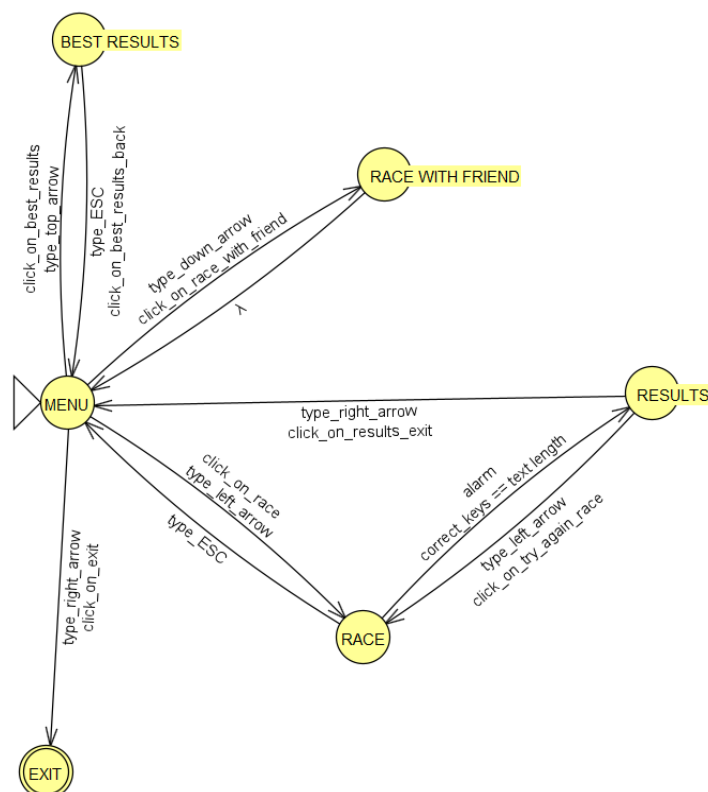
3.1. Módulos

Aqui são apresentados os vários módulos do projecto, à exceção dos módulos **timer**, **keyboard**, **mouse**, **graphic** e **rtc** e que já foram apresentados na seção anterior. Informações mais específicas sobre as funções de cada módulo são disponibilizadas através do doxygen.

3.1.1. proj

Ficheiros: proj.h, proj.c

Neste módulo encontra-se a função principal do projecto: *proj_main_loop*. Aqui estão definidos os vários textos a usar no jogo de forma aleatória (ver detalhes na seção 4), são subscritas as interrupções dos dispositivos todos e detetadas essas interrupções no *driver_receive loop*. Ainda dentro desse loop, foi implementada uma máquina de estados (*Menu_state*) que permite perceber em que página o jogo se encontra e lidar com os vários eventos (*Menu_event*) gerados pelas várias interrupções a partir dos *handlers* respetivos de cada estado.



Máquina de estados implementada

De seguida, são apresentadas algumas das tarefas com que este módulo lida:

- desenhar os vários *xpm's* para a página da corrida (fundo, semáforo de início de corrida, carro e cursor)
- processar interrupções do timer, do teclado, do rato e RTC (embora este último não tenha um *handler* exclusivamente dedicado)
- apresentar texto no ecrã enquanto o utilizador o escreve
- comparar o texto a copiar com o texto escrito pelo utilizador e pintar os caracteres a verde ou vermelho, consoante o resultado desta comparação
- desenhar caixas de texto com tamanho variável, consoante o tamanho do texto a digitar

3.1.4. results

Ficheiros: *race.h*, *race.c*

Este módulo contém todas as funções, estruturas de dados e variáveis relacionadas com os resultados obtidos. Insere-se nos mesmos ficheiros que o módulo *race* (ver secção 3.1.3) devido à partilha de várias variáveis *static*.

Nesta página do jogo é possível observar os resultados finais da corrida, alterar o nome do registo dos resultados caso estes devam ser guardados e rebentar bolhas (ver secção 4) que se movimentam por todo o ecrã, atualizadas à mesma taxa que a *frame rate*.

De seguida, são apresentadas algumas das tarefas com que este módulo lida:

- desenhar os vários *xpm's* para a página dos resultados
- calcular resultados
- processar interrupções do timer, do teclado e do rato
- criar *animated sprites* para as bolhas e verificar colisões entre as bolhas e o rato quando um dos botões for clicado
- chamar a função do módulo *best_results* (ver secção 3.1.5) que compara os resultados obtidos

3.1.5. best_results

Ficheiros: *best_results.h*, *best_results.c*

Este módulo contém todas as funções, estruturas de dados e variáveis relacionadas com a página dos melhores resultados. No início do programa os melhores resultados, guardados no ficheiro *best_results.txt* são lidos e guardados até ao final do programa onde finalmente são escritos de novo nos ficheiros. Desta forma, é possível comparar os resultados sem o trabalho penoso de ler e escrever várias vezes o mesmo ficheiro.

De seguida, são apresentadas algumas das tarefas com que este módulo lida:

- ler (no início do programa) e escrever (no fim do programa) para o ficheiro *best_results.txt*
- desenhar a página dos melhores resultados
- processar interrupções do timer, teclado e rato

- comparar o resultado obtido no jogo com os 3 melhores resultados guardados e substituir caso seja melhor do que pelo menos um deles
- usar as funções do RTC para adicionar a data e a hora à estrutura do resultado

3.1.6. Chars

Ficheiros: Chars.h, Chars.c

Este módulo contém todas as funções, estruturas de dados e variáveis relacionadas com os caracteres do jogo. Os caracteres do jogo são estruturas compostas por um número que o associa a uma posição de memória onde está mapeado o caracter de jogo, a sua posição e um tipo enumerado que indica a cor em que deve ser representado o caracter de jogo.

De seguida, são apresentadas algumas das tarefas com que este módulo lida:

- carregar os *xpm's* dos caracteres e mapear a parte relevante
- desenhar os caracteres
- converter o *scancode* do teclado para o elemento *index* da *struct Char*
- converter um char para o elemento *index* da *struct Char*
- converter strings de chars para caracteres do jogo e vice-versa
- verificar colisões de uma coordenada com um *Char*

3.1.7. sprite

Este módulo contém todas as funções, estruturas de dados e variáveis relacionadas com *sprites* e *animated sprites* (*AnimSprite*). As estruturas foram definidas pelo professor João Cardoso e serviram de ponto de partida para a nossa implementação. Para o construtor das *animated sprites* é usada uma função com número variável de argumentos de modo a lidar com um número variável de pixmaps.

De seguida, são apresentadas algumas das tarefas com que este módulo lida:

- criar *sprites* e *animated sprites*
- modificar *sprites* e *animated sprites*
- atualizar *sprites* e *animated sprites* de acordo com os seus atributos
- destruir *sprites* e *animated sprites*
- verificar colisões

3.1.8. utils

Este módulo contém algumas funções de auxílio a outros módulos:

- converter números binários para BCD e vice-versa
- obter bits mais e menos significativos
- *util_sys_inb*

3.1.9. xpm's

Este módulo contém todos os *xpm's* do projeto que permitiram adicionar uma componente mais visual e esteticamente agradável ao jogo. As imagens foram obtidas na internet e editadas através do GIMP ou do Photoshop.

Módulo	Peso relativo	Contribuição Henrique - 60%	Contribuição Luísa - 40%
timer	6%	Tudo nos labs e adaptação ao projeto para o <i>frame rate</i> 50%	Adaptação ao projeto para contar o tempo 50%
keyboard	8%	Maior parte, incluindo adaptação ao projeto 80%	Algumas contribuições nos labs 20%
mouse	9%	Tudo nos labs e adaptação ao projeto 80%	Algumas contribuições no projeto 20%
graphic	10%	Metade nos labs e adaptação ao projeto 60%	Metade nos labs 40%
rtc	7%	Interrupções de alarme e adaptação ao projeto 70%	Leitura de data e hora 30%
proj	7%	<i>Driver_receive loop</i> e implementação dos <i>interrupt handlers</i> 60%	Máquina de estados e protótipos dos <i>interrupt handlers</i> 40%
menus	6%	Implementação das funções 50%	Criação das estruturas para a máquina de estados, macros e protótipos de funções 50%
race	11%	Restantes funções 70%	<i>Design</i> da página e caixa de texto de tamanho variável 30%
results	8%	Maior parte 60%	<i>Design</i> da página e fórmulas para os resultados 40%
best_results	4%	Maior parte 80%	Algumas contribuições 20%
Chars	8%	Maior parte 90%	Algumas contribuições 10%
sprite	5%	Tudo 100%	0%
utils	1%	Tudo nos labs ao mesmo tempo que o timer e o RTC 90%	Algumas contribuições nos labs 10%
xpm's	10%	Caracteres, carro e semáforo 20%	Restante xpm's 80%

Outros	Contribuição Henrique	Contribuição Luísa
relatório	40%	60%
descrição relatório	comentários doxygen, restante foi tudo metade-metade com a colega	estrutura, gráficos, configuração doxygen, restante foi tudo metade-metade com o colega

3.2. Doxygen

A documentação gerada pelo doxygen pode ser consultada clicando no ficheiro **index.html** que se encontra dentro da pasta **proj\doc\doxygen\html**. Os vários gráficos de chamada de funções pode ser visto nesse link.

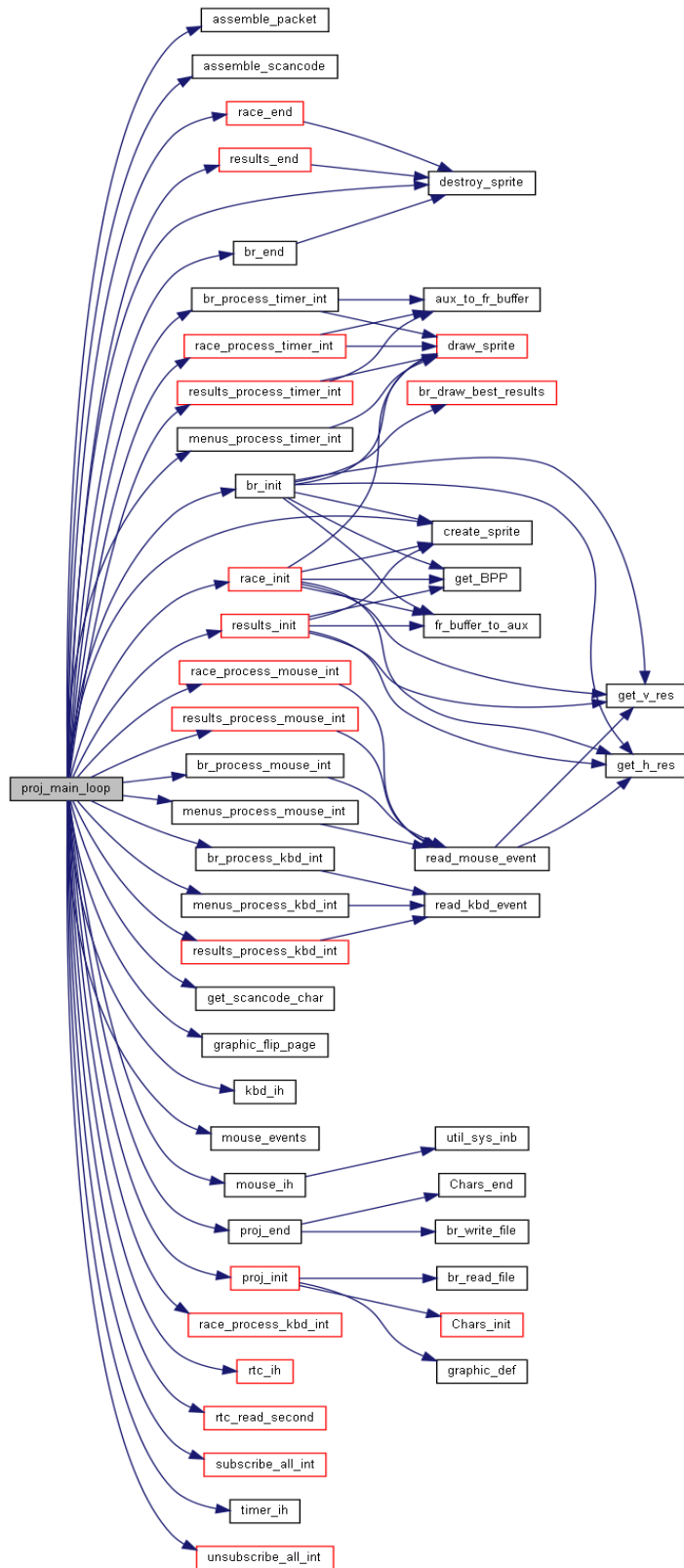


gráfico de funções chamadas pelo `proj_main_loop`

4. Detalhes da implementação

- Caracteres por minuto (CPM):

$$CPM = \frac{n^{\circ} \text{ teclas corretas} * 60 \text{ segundos}}{n^{\circ} \text{ segundos total}}$$

- Precisão (em percentagem):

onde $n^{\circ} \text{ teclas corretas} - n^{\circ} \text{ backspaces premidos} \geq 0$

$$\text{precisão} = \frac{n^{\circ} \text{ teclas corretas} - n^{\circ} \text{ backspaces premidos}}{n^{\circ} \text{ caracteres total do texto dado}} * 100$$

- Critério de desqualificação:

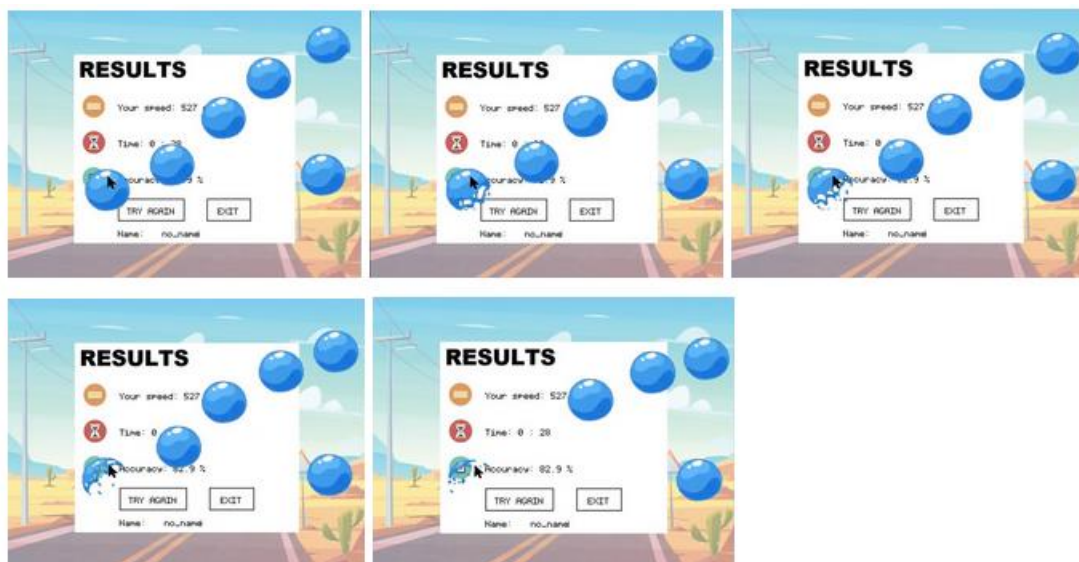
$$\text{tempo máximo} = \frac{n^{\circ} \text{ caracteres do texto} * 60 \text{ segundos}}{50 \text{ caracteres por minuto}} + 2 \text{ segundos}$$

- Textos:

Os textos apresentados em cada corrida são escolhidos tendo em conta a leitura de segundos (registo 0x00) do RTC. A partir destes é feito o módulo de número de textos disponíveis para assim ser escolhido um deles.

- Bolhas:

As bolhas são estruturas *AnimSprite*, constituídas por 5 pixmaps (bolha principal e processo de destruição). Quando são inicializadas são tratadas como uma sprite, uma vez que apenas um dos pixmaps é necessário (o número de imagens da *AnimSprite* é alterado para uma única, embora tenha os outros pixmaps carregados). Quando uma das bolhas sofre um clique do rato, inicia o seu processo de destruição, sendo nesse momento tratada como um *AnimSprite*. O número de imagens é reposto para o real e a bolha vai sendo sequencialmente destruída. Quando chega ao final da sua destruição, o espaço de memória alocado pela bolha é libertado.



Sequência de imagens (da esquerda para a direita e de cima para baixo) desde o clique do rato até a bolha ser completamente destruída.

5. Conclusões

Concluindo, consideramos que esta cadeira tem o aspeto positivo de ser maioritariamente prática, sendo possível aplicar o que foi dado nas teóricas, ao invés de serem apenas testes e exame final. Para além disso, a matéria é, na nossa opinião, bastante interessante e relacionada com o curso.

Por outro lado, concordamos que a informação exposta não está bem organizada. Sugeríamos que em vez de haver informação espalhada pelos slides das aulas teóricas, pelos enunciados dos labs e pelo doxygen, seria preferível encontrar toda esta informação num só sítio de forma organizada e clara. Para além disso gostaríamos de salientar que a nota do projecto não deveria estar, na nossa opinião, limitada por dispositivos que não foram trabalhados nas aulas práticas (o rtc e a porta-série). Ou se eliminavam esses dispositivos, ou se introduziam esses também nas aulas práticas.