



MIEIC – Dezembro 2017

Projeto Final

LCOM

Turma 7 - Grupo 9

Bruno Alexandre Oliveira Dias – up201504859
Maria Eduarda Santos Cunha – up201506524



Índice

1. Introdução	3
2. Obstacle Course	3
3. Instruções de Utilização	3
4. Estado do Projeto	4
4.1. Timer.....	5
4.2. Teclado.....	5
4.3. Placa de Vídeo	5
4.4. Rato.....	6
4.5. RTC	6
5. Estrutura e Organização do Código	6
5.1. main.c	6
5.2. obstacleCourse.h e obstacleCourse.c	6
5.3. startMenuState.h e startMenuState.c	7
5.4. gameState.h e gameState.c.....	7
5.5. graphics.h e graphics.c.....	7
5.6. mouse.h e mouse.c.....	7
5.7. keyboard.h e keyboard.c.....	7
5.8. timer.h e timer.c.....	7
5.9. video_gr.h e video_gr.c.....	8
5.10. vbe.h e vbe.c.....	8
5.11. rtc.h e rtc.c	8
5.12. bitmap.h e bitmap.c	8
5.13. rectangle.h e rectangle.c.....	8
5.14. i8042.h.....	9
5.15. i8254.h.....	9
6. Detalhes de Implementação	10
7. Conclusões.....	10
8. Apêndice: Instruções de Instalação	11



1. Introdução

O presente relatório serve de complemento ao projeto final desenvolvido na cadeira de Laboratório de Computadores, de forma a explicar e analisar o código desenvolvido de forma mais extensa e específica.

Pensamos que realizamos os requerimentos esperados.

2. Obstacle Course

O jogo desenvolvido, Obstacle Course, tem como objetivo principal o desvio de obstáculos por parte do jogador.

Existem 3 faixas sobre as quais o jogador pode deslocar o seu carro, os obstáculos (pedras, cones e buracos) que se tentam evitar, por resultarem na perda de vidas (representadas pelos corações), e moedas, que, se apanhadas, acumulam pontos.

O jogo acaba quando o jogador já não tem vidas.

3. Instruções de Utilização

O menu principal possui as opções *Play* para jogar e *Exit* para sair da aplicação.

Quando no modo de jogo, o utilizador deve tentar desviar o carro dos obstáculos que fazem perder vidas e apanhar as moedas que permitem acumular pontos.

O jogador possui 5 vidas, portanto embater em 5 obstáculos termina o jogo.

De forma a deslocar o carro, o utilizador pode optar por usar as teclas *w* e *s*, para andar para cima e para baixo respetivamente, ou recorrer ao rato. A deslocação por via do teclado permite um deslocamento rápido entre as 3 faixas, ao passo que com o rato, de acordo com os seus movimentos no sentido positivo ou negativo (segundo o eixo dos *y*), o carro desloca-se gradualmente podendo ocupar posições intermédias.

No menu principal, pode-se pressionar quer a tecla *esc* ou o botão *Exit* para voltar à consola.

No modo de jogo, pressionar *esc* leva de volta ao menu principal.

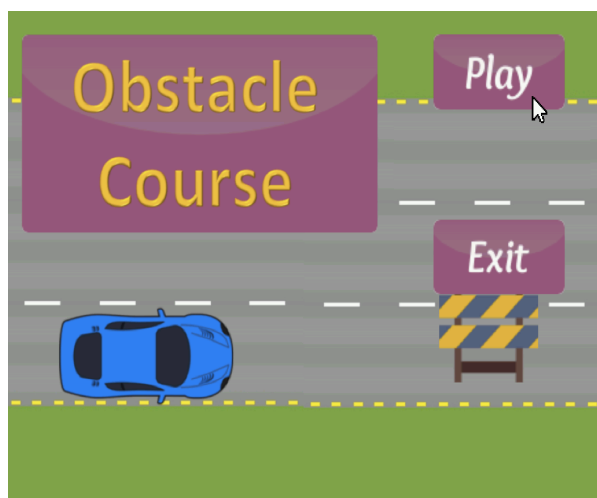


Figura 1 Menu principal com opção *Play* selecionada pelo rato



Figura 2 Jogo com carro deslocado por meio das teclas *w* e *s*



Figura 3 Jogo com carro deslocado por meio do rato de forma a exemplificar uma posição intermédia que o carro pode ocupar

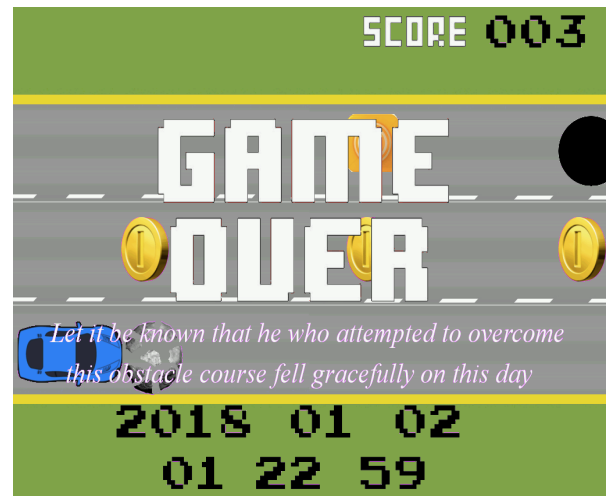


Figura 4 Término do jogo por ação da perda de vidas

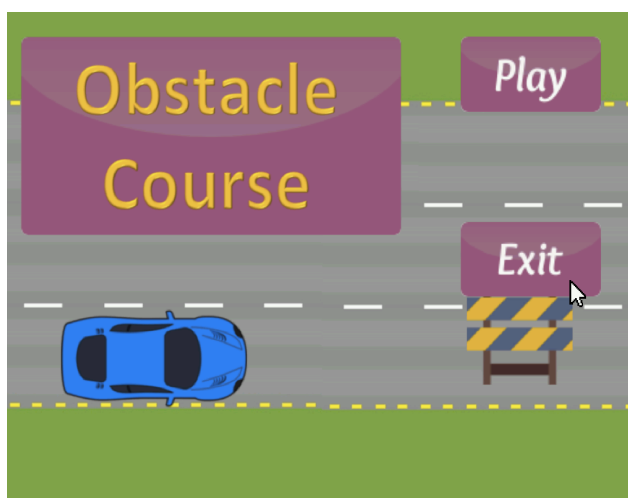


Figura 5 Menu principal com opção Exit selecionada pelo rato

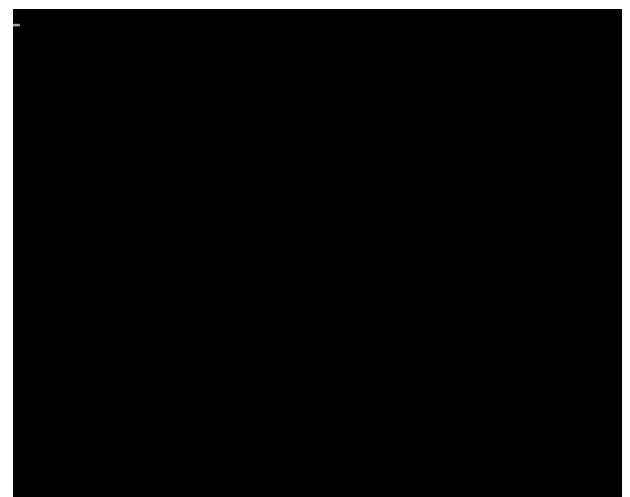


Figura 6 Consola após selecionar a opção Exit

4. Estado do Projeto

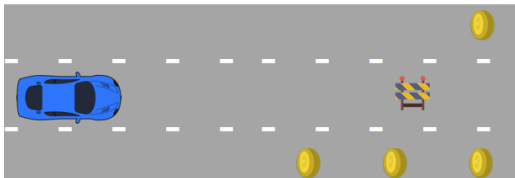
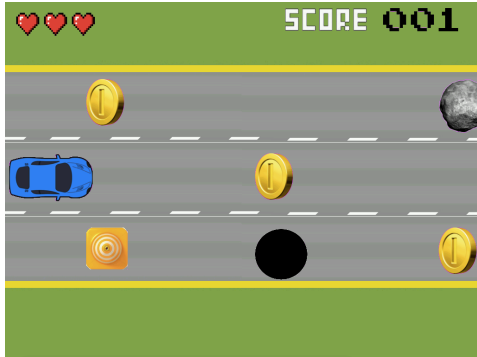
O projeto encontra-se implementado conforme previsto na especificação do projeto, à exceção de:

- Terem sido acrescentados alguns obstáculos;
- Ter-se mudado o sistema de pontuação;
- Não se ter chegado a implementar a ligação porta série, que constituía um extra que gostávamos de ter aplicado, de forma a tornar o jogo *multiplayer*.

Assim, atualmente, o carro desloca-se para cima e para baixo por ação do teclado ou do rato, conforme preferência do utilizador. Caso embata num obstáculo, este



passa a ficar *destruído* e o jogador perde uma vida e, numa moeda, acumula pontos. Estes elementos de débito de vidas e soma de pontos são gerados sempre à mesma distância entre eles, havendo sempre uma moeda ao lado de um obstáculo, em faixas diferentes, e a faixa em que aparecem cada obstáculo é também aleatória.

Especificação do Projeto	Versão Final
	

Dispositivo I/O	Utilização	Interrupção?
Timer	Atualização do estado de jogo	Sim
Teclado	Movimentar carro	Sim
Placa de Vídeo	Exibir ecrã de jogo e menus	Não
Rato	Selecionar opções nos menus e movimentar carro	Sim
RTC	Mostrar dia e hora no final do jogo	Não

4.1. Timer

É utilizado para atualizar o estado de jogo, controlando as animações gráficas.

4.2. Teclado

É utilizado no controlo do jogo para deslocar o carro ao longo das faixas, segundo as teclas w e s, para cima e para baixo, respetivamente.

Para este periférico, é relevante a função `unsigned long keyboard_read()` em `keyboard.c` que trata de ler os scan codes relativos a cada tecla pressionada.

4.3. Placa de Vídeo

É utilizada para mostrar o conteúdo gráfico do jogo, desde o menu a todos os pequenos bitmaps (modo RGB(5:6:5) do modo de jogo.

Optamos pelo modo de vídeo 0x117 e resolução de 1024x768 pixéis.

Implementamos *double buffering* (`vídeo_gr.h`) para que o jogo apresentasse fluidez, mas não tivemos a oportunidade de usar sprites animadas.



Possuímos deteção de colisões (`int checkCollision()` em `graphics.c`), para indicar quando há o embate entre o carro e um obstáculo ou uma moeda. Ainda que pudéssemos ter recorrido a `rectangle.h`, usamos apenas as coordenadas dos bitmaps.

Recorremos ao uso de *fonts* (`void drawDigit(GameState* state, int digit, int deviation, int y)` em `graphics.c`) quando pretendemos mostrar números, dado que demonstra o nosso conhecimento sobre o uso das mesmas.

4.4. Rato

É utilizado para optar entre botões, *Start* para iniciar o jogo e *Exit* para sair da aplicação.

Ainda, no modo de jogo, de acordo com a sua deslocação (sentido positivo ou negativo segundo o eixo dos *y*), permite movimentar o carro ao longo da pista.

Para este periférico, é relevante a função `int mouse_read(unsigned long* data)` em `mouse.c` que trata de ler um byte recebido pelo rato. Posteriormente, `int mouse_reader(ObstacleCourse* course)` organiza os bytes recebidos em packets de 3.

4.5. RTC

É utilizado para mostrar o dia e hora do jogo no final do jogo.

Para este periférico, é relevante a função `void getDate(unsigned long *day, unsigned long *month, unsigned long *year, unsigned long *hour, unsigned long *minute, unsigned long *seconds)` em `rtc.c`.

5. Estrutura e Organização do Código

Optamos por não atribuir responsáveis aos ficheiros importados diretamente dos laboratórios das aulas práticas, exceto o `video_gr`, pois, ainda que tenham sido adaptados e ligeiramente modificados, foi um trabalho contínuo conforme surgia a necessidade dado aquilo em que cada um estava a trabalhar.

5.1. main.c

Inicia modo de vídeo. Cria instanciação do jogo. Enquanto este não acabar, atualiza-o e desenha-o. Quando acabar, liberta a memória usada e dá *unsubscribe* dos periféricos. Por fim, sai do modo de vídeo.

Peso no total do projeto: 1%
Responsável: Eduarda Cunha

5.2. obstacleCourse.h e obstacleCourse.c

Possui as funções de inicialização, atualização, desenho e término da aplicação. Ainda, estas funções podem ter comportamentos diferentes dado o conteúdo do elemento `State currentState` da `struct ObstacleCourse`, que pode ser dos tipos `MAIN_MENU_STATE` e `GAME_STATE`.



Peso no total do projeto: 9%

Responsável/Contribuição: 50% Bruno Dias e 50% Eduarda Cunha

5.3.startMenuState.h e startMenuState.c

Possui as funções de inicialização, atualização, desenho e término do menu. Dadas as 2 opções *Play* e *Exit*, o jogo é atualizado para o modo de jogo ou sair da aplicação respetivamente.

Peso no total do projeto: 5%

Responsável: Eduarda Cunha

5.4.gameState.h e gameState.c

Possui as funções de inicialização, atualização, desenho e término do modo de jogo. Os elementos principais para a atualização do jogo são o deslocamento do carro com base na tecla pressionada ou movimento do rato e o aparecimento dos diferentes obstáculos e moedas.

Peso no total do projeto: 16%

Responsável/Contribuição: 60% Bruno Dias e 40% Eduarda Cunha

5.5.graphics.h e graphics.c

Possui algumas funções relevantes para a dimensão gráfica da aplicação, desde o desenho de dígitos, data e pontuação à de atualização do movimento do carro conforme o movimento do rato, verificação de colisões etc.

Peso no total do projeto: 16%

Responsável/Contribuição: 60% Bruno Dias e 40% Eduarda Cunha

5.6.mouse.h e mouse.c

Código importado da submissão do laboratório 4 das aulas práticas, com pequenas alterações de forma a corresponder às necessidades do projeto.

Peso no total do projeto: 5%

5.7.keyboard.h e keyboard.c

Código importado da submissão do laboratório 3 das aulas práticas, com pequenas alterações de forma a corresponder às necessidades do projeto.

Peso no total do projeto: 3%

5.8.timer.h e timer.c

Código importado da submissão do laboratório 2 das aulas práticas, com pequenas alterações de forma a corresponder às necessidades do projeto.



Peso no total do projeto: 2%

5.9. **video_gr.h** e **video_gr.c**

Código importado da submissão do laboratório 5 das aulas práticas. Mudanças na alocação de memória e adição de novas funções de forma a implementar double buffer.

Peso no total do projeto: 9%

Responsável: Bruno Dias

5.10. **vbe.h** e **vbe.c**

Código importado da submissão do laboratório 5 das aulas práticas, com pequenas alterações de forma a corresponder às necessidades do projeto.

Peso no total do projeto: 13%

5.11. **rtc.h** e **rtc.c**

Possui as funções relativas ao tempo real em que o jogo está a decorrer e a que recorremos para obter o dia e instante em que o mesmo acaba.

Peso no total do projeto: 3%

Responsável: Bruno Dias

5.12. **bitmap.h** e **bitmap.c**

Estes ficheiros são maioritariamente da autoria do Henrique Ferrolho e encontram-se disponibilizados em <http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html>, sendo que sofreram algumas alterações mínimas de forma a percebermos as nossas imagens com o fundo rosa como se os pixéis com esta cor não existissem.

Peso no total do projeto: 9%

Responsável: Bruno Dias

5.13. **rectangle.h** e **rectangle.c**

Este ficheiro é semelhante ao disponibilizado pelo Henrique Ferrolho, dado que surge como complemento ao anteriormente mencionado. Trata-se de uma mera declaração de uma estrutura com 2 coordenadas x e 2 coordenadas y de forma a referirmo-nos a uma área retangular, útil posteriormente para detetar o pressionar dos botões do menu com o rato.

Peso no total do projeto: 1%

Responsável: Eduarda Cunha



5.14. i8042.h

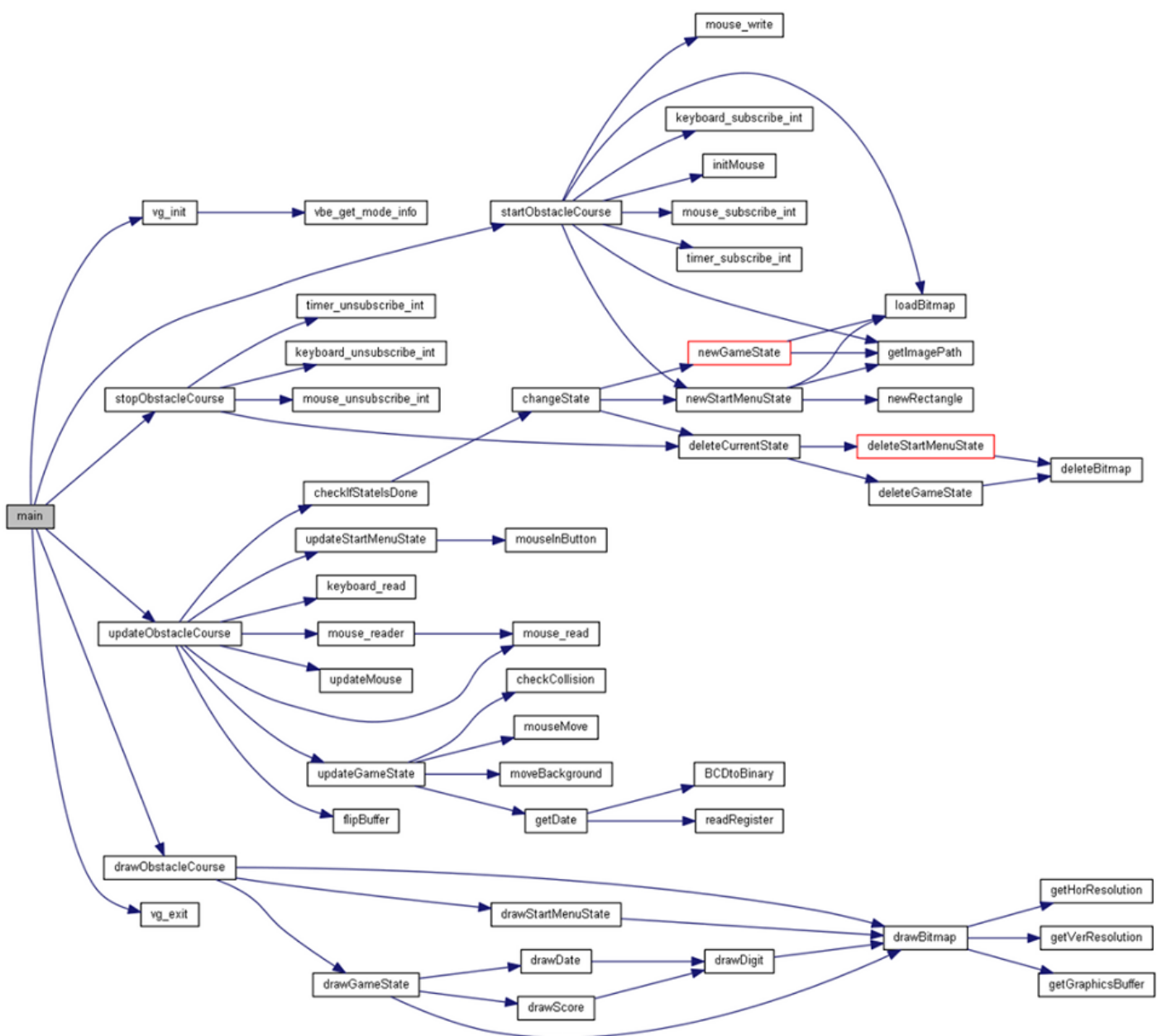
Macros importadas da submissão de laboratórios das aulas práticas, com pequenas adições de forma a corresponder às necessidades do projeto.

Peso no total do projeto: 2%

5.15. i8254.h

Macros importadas da submissão de laboratórios das aulas práticas, com pequenas adições de forma a corresponder às necessidades do projeto.

Peso no total do projeto: 4%





6. Detalhes de Implementação

O jogo foi implementado tendo por base uma máquina de estados com 2 estados de forma a alterar facilmente entre as fases diferentes do jogo: menu e modo de jogo.

Ainda que o RTC não tenha sido lecionado nas aulas, pensamos que foi bastante fácil de implementar. Originalmente, tencionávamos a recorrer a este periférico de forma a manter um registo das várias pontuações obtidas no jogo, mas, por falta de tempo, optamos por incluir algo mais básico, apresentando apenas a data e hora do final do jogo.

Não tivemos oportunidade de estudar conteúdos relativos a UART, mas face a feedback de anos anteriores, professores e até a quantidade avultada de questões que vimos serem feitas por colegas, acreditamos que seria do interesse máximo dos alunos que passasse a ser lecionado nas aulas práticas, ainda que não contasse para avaliação. No entanto, não acreditamos que haja tempo num semestre para acrescentar mais um laboratório, ainda que este fosse bastante benéfico.

Quanto às colisões, pensamos que foram bastante intuitivas de aplicar.

Por fim, achamos interessante termos tido a oportunidade de relembrar *assembly*, mas consideramos algo desnecessário na cadeira em questão.

7. Conclusões

De forma geral, pensamos que tivemos um bom desempenho na cadeira ao longo do semestre.

Ambos nos encontramos a repetir a cadeira, pelo que temos uma perspetiva bem diferente da que possuíamos o ano passado, mas algumas ideias mantêm-se. Continuamos a ser da opinião de que a cadeira se insere na posição temporal errada do curso, visto que envolve a aprendizagem de muitos conhecimentos novos que ainda não foram lecionados ou suficientemente aprofundados nesta fase.

Passamos a explicar melhor a perspetiva do ano passado: é um verdadeiro desafio lidar com a pressão temporal de ter entregas quinzenais pela primeira vez, ter de programar em C quando recebemos apenas algumas luzes em PROG, trabalhar numa consola começando por usar comandos um pouco sem saber para que servem etc.

Este ano mantemos a opinião de que, para o aluno do 2º ano, a cadeira continua a reunir as dificuldades apresentadas, mas, enquanto alunos do 3º ano, a dificuldade da linguagem de programação e de trabalhar com a consola desapareceram. Assim, restou-nos apenas a gestão de tempo para trabalhar, dado que estávamos a realizar 6 cadeiras cada um e o trabalho em conjunto era dificultado por não sermos do Porto e de cidades diferentes.

Cumprimos a maior parte dos objetivos a que nos tínhamos proposto na especificação do projeto, tendo deixado apenas por fazer a ligação porta série, que considerávamos um extra ao nosso alcance, mas, dado que tínhamos mais 3 entregas das unidades curriculares do 3º ano para as 2 semanas de intervalo letivo, tornou-se impossível. Ainda, realizamos algumas alterações ao inicialmente pensado, quer para incluir componentes que, caso contrário, por falta de tempo, não teríamos



conseguido implementar (troca entre uso do RTC para fazer um registo das pontuações para display da data e hora no fim do jogo), quer para acrescentar funcionalidades (inclusão de mais obstáculos e sistema de vidas).

8. Apêndice: Instruções de Instalação

No terminal, invocar os seguintes comandos dentro da pasta do projeto:

sh install.sh se estiver a correr pela primeira vez ou ocorrer mudança dos recursos (imagens) a utilizar;

sh compile.sh para compilar todos os ficheiros necessários;

sh run.sh para correr o programa desenvolvido e jogar.