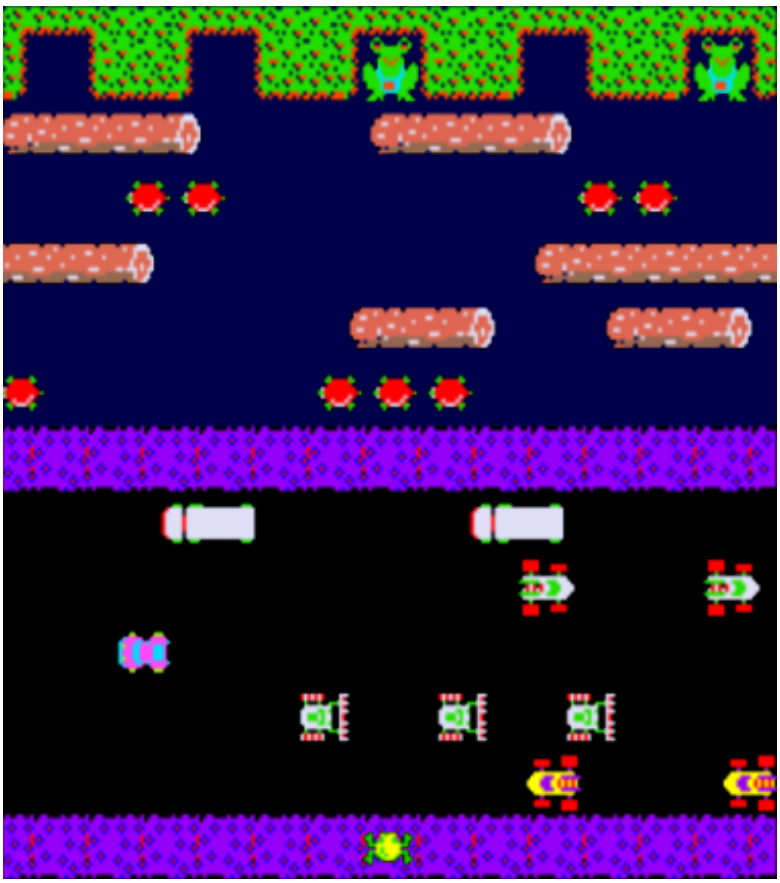
Faculdade de Engenharia da Universidade do Porto

Mestrado integrado em Engenharia Informática e Computação 2014/2015

Frogger for MINIX

Laboratório de Computadores

Relatório do Projeto



Elementos do grupo:

Ana Amaral, up201303169@fe.up.pt

João Guarda, up201303463@fe.up.pt

Grupo 08

Índice

Descrição sucinta do jogo

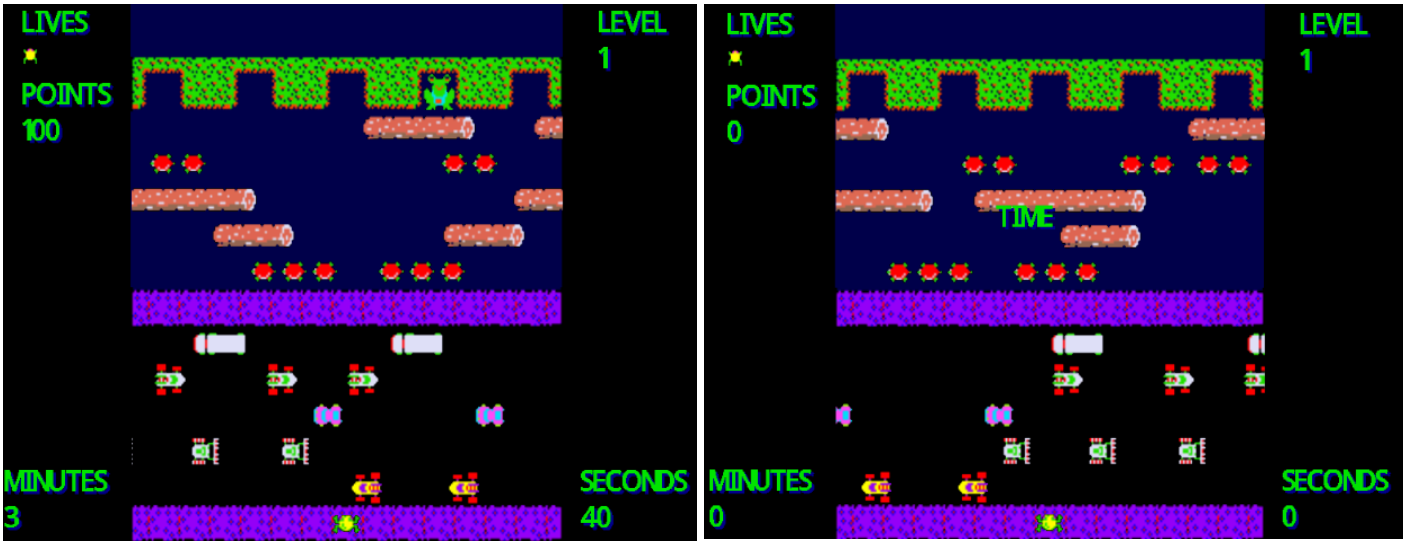
O jogo resume-se ao controlo de um sapo que deve atravessar uma estrada movimentada e um rio cheio de obstáculos até chegar ao seu ninho. A cada cinco travessias bem-sucedidas, o jogo progride de nível, aumentando a sua dificuldade (carros mais rápidos, menos troncos e tartarugas, corrente do rio mais rápida).

Instruções de utilização

1) Ecrã/Menu Inicial

Ao iniciar o programa é apresentado o ecrã de introdução do programa, em que o utilizador é diretamente direcionado para o menu Principal, onde poderá usar o rato para selecionar uma das 4 opções.

2) Play

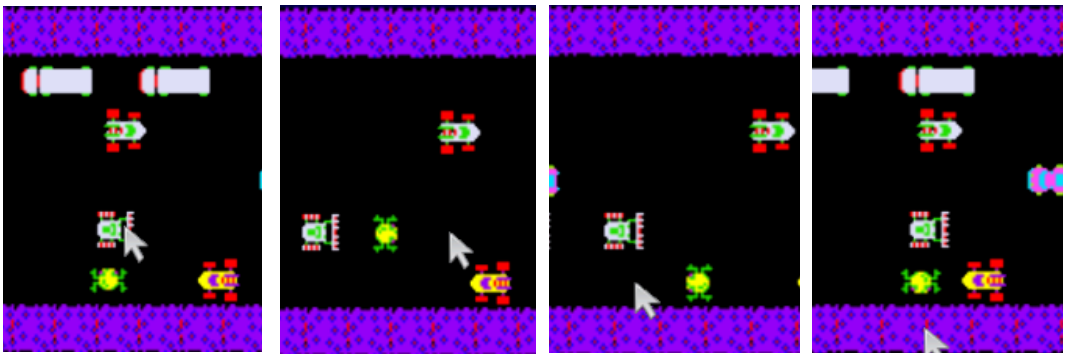


Ao clicar na opção “Play” o utilizador é redirecionado para o jogo em si. Nesta opção o jogador tem uma estrada cheia de veículos, dos quais o sapo se deverá desviar, seguida de um rio que o sapo terá que atravessar saltando entre troncos e tartarugas (sendo que algumas destas emergem e submergem) até chegar a uma das 5 casas disponíveis no final da travessia.

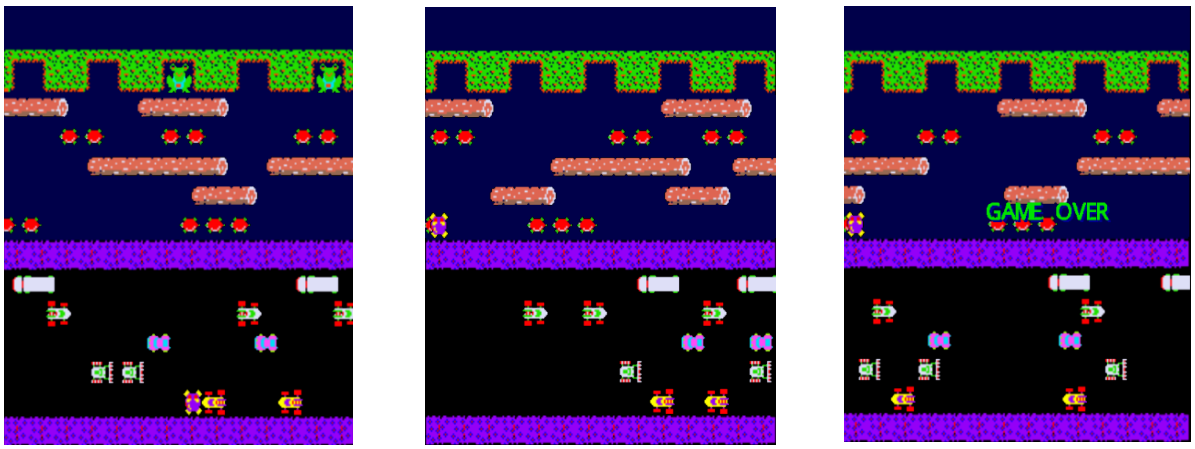
No canto superior esquerdo temos a informação acerca das vidas que o jogador ainda tem (sendo que este começa o jogo com 3 vidas) e os pontos deste. Por cada sapo que ultrapassa todos os obstáculos e que chega a uma das cinco casas disponíveis o jogador ganha 100 pontos.

No canto superior direito encontra-se o nível do jogo em que o jogador se encontra.

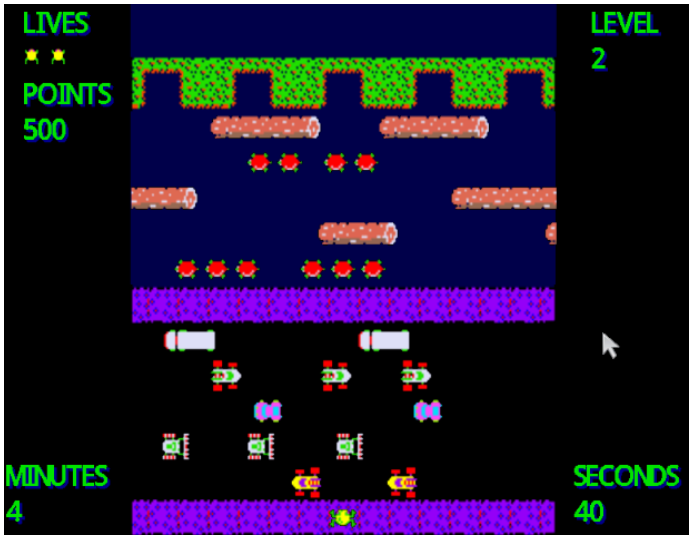
Nos dois cantos inferiores temos o tempo restante de jogo (o jogador tem 5 minutos para passar cada nível), sendo que à esquerda temos os minutos e à direita os segundos. Caso o jogador exceda o seu tempo de jogo, o programa volta ao menu principal.



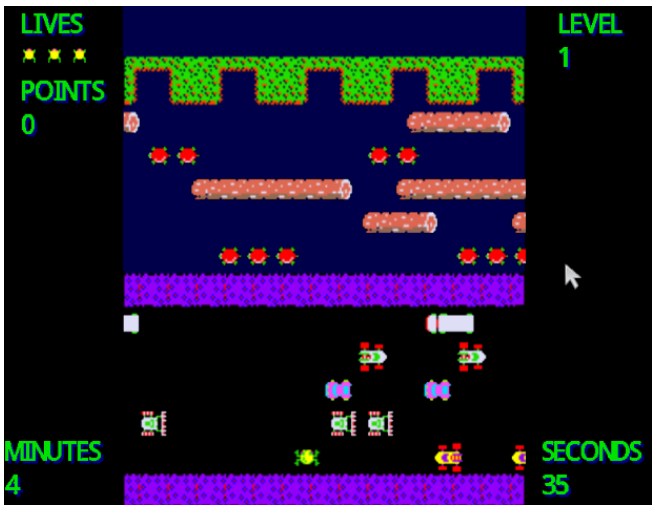
Para deslocarmos o sapo podemos faze-lo utilizando o teclado ou o rato. No primeiro, para que o sapo se desloque para a frente, direita, esquerda ou trás as teclas necessárias são, respetivamente, W, D, A e S ou as teclas com setas para cada direção respetiva. Quanto ao segundo periférico, este permite deslocar o sapo para a frente, direita, esquerda ou trás clicando com o rato na frente, à direita, à esquerda ou atrás do sapo.



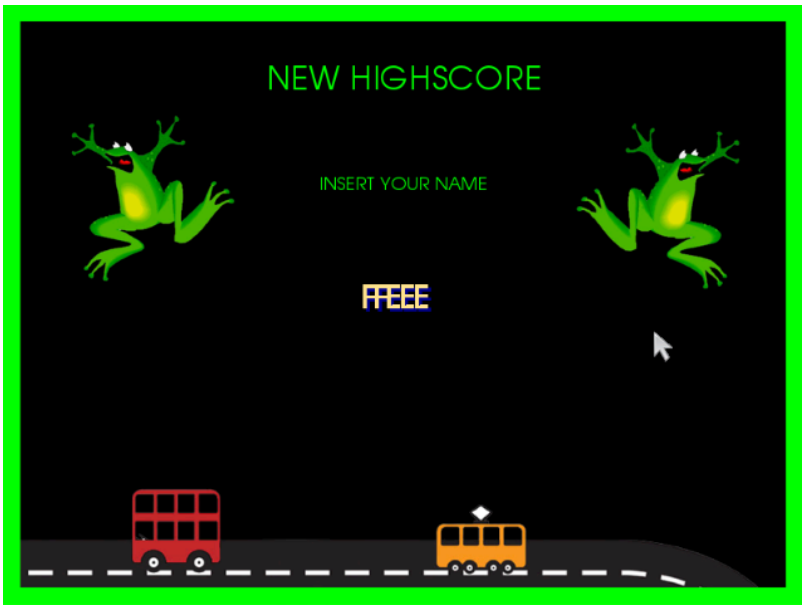
O jogador vai perdendo vidas caso o sapo colida com algum veículo, caia ao rio ou na reta final não atinga corretamente o espaço de uma das cinco casas. Ao esgotar todas as vidas o jogador perde o jogo e é redirecionado para o menu Principal.



Caso o jogador consiga colocar os cincos sapos nas cinco casas disponíveis, dentro do tempo, este passa para o nível seguinte, mantendo a pontuação, as vidas que ainda lhe restam e os respetivos 5 minutos como tempo limite para ultrapassar cada nível.



Para pôr o jogo em pausa basta clicar na tecla “P” e para voltar ao jogo a tecla espaço. Caso jogador queira sair do jogo, este pode premir a tecla “esc”, sendo redirecionado para o menu principal. sendo seguidamente convidado a inserir o seu nome para que faça parte das “highscores”, o que acontece também caso o jogador perca o jogo ou esgote o tempo disponível para passar de nível.



3) Highscores

Ao premir a opção “Highscores” o utilizador pode consultar as melhores pontuações obtidas no jogo. Estas encontram-se ordenadas pelas melhores e organizadas pela data em que as quais foram geradas, seguida do nome do jogador que as obteve e por fim, a pontuação respetiva deste.

Para voltar ao menu Principal, o jogador apenas necessita de clicar na opção “Back”.

4) Settings

Ao clicar na opção “Settings”, o utilizador é redirecionado para um menu de escolha do tamanho do gráfico da consola, podendo escolher entre o modo 800x600 ou 1024x768. Para voltar ao menu Principal, o jogador apenas necessita de clicar na opção “Back”.

5) Exit

A opção “Exit” termina o programa.

Estado Final do Projeto

O projeto implementou quase todos os seus módulos propostos, ficando apenas por implementar o modo de Dois Jogadores e, consequentemente, o módulo do porta de série UART. Também ficou por implementar leitura e escrita de ficheiros .txt para ler e escrever as pontuações máximas.

Dispositivo	Funcionalidade	Int.
Timer	Controlar *frame rate*	S
Keyboard	Controlar sapo e inserção de informação quando solicitada	S
Mouse	Controlar sapo e navegação entre menus	S
RTC	Controla o tempo de jogo e alerta para o seu fim	S
Video Graphics	Menus e *screen displays*	N

Timer

O dispositivo tem como única funcionalidade controlar as *frame rates.* Está implementado no ficheiro (timer.c) e funciona como um único objecto em toda a execução, sendo acedido com a função *getTimer()* (linhas 46 a 48). Esta implementação permite um rápido acesso ao contador do timer. Cada interrupção é processada pelo *dispatcher* (ficheiro frogger.cpp linhas 79 a 81).

Keyboard

O teclado tem como função controlar o sapo, pausar e recomeçar o jogo e providenciar informação à aplicação quando solicitado (i.e. escrever o nome para nova pontuação máxima). Cada interrupção é processada pelo *dispatcher* (ficheiro frogger.cpp linhas 70 a 78)

Mouse

O rato têm várias funcionalidades desde controlar o sapo, à navegação entre menus. O dispositivo tem uma implementação semelhante à do timer. Está implementado no ficheiro (mouse.c) e funciona como um único objecto em toda a execução, sendo acedido com a função *getMouse()* (linhas 146 a 148). Esta implementação permite um rápido acesso ao contador do timer. Cada interrupção é processada pelo *dispatcher* (ficheiro frogger.cpp linhas 68 a 70).

RTC

Tem como funcionalidade controlar o tempo de jogo e alerta para o seu fim. Se o utilizador estiver em cena de jogo as interrupções do tipo update decrementam o tempo de jogo, e as interrupções do tipo alarm terminam o jogo. Cada interrupção é processada pelo *dispatcher* (ficheiro frogger.cpp linhas 82 a 84). O seu *handler* trata adequadamente o tipo da interrupção (ficheiro RTC.C linhas 41 a 58).

Video Graphics

Tem como funcionalidade fazer *screen displays.* Está implementado com triple buffering sendo que além do buffer da VRAM, existe também um para o rato e um outro para preparar a próxima cena. Isto permite um maior refrescamento por parte do rato e, assim sendo, evitar *lag's* enormes. No desenho de cenas, começa-se por desenhar no terceiro buffer copiando-o depois para o buffer do rato (ver ficheiro graphics.c linhas 150 a 152) e finalmente para o buffer da VRAM(linhas 154 a 156).

Organização/Estrutura do Código

Nota prévia de que o responsável não quer dizer que tenha realizado as tarefas, apenas quer dizer que ficou acordado ser essa pessoa a responsável.

bitmap.c (Responsável: João Guarda)

Este módulo tem como função ler ficheiros com a extensão .bmp. Nele também contem funções para a sua escrita no buffer assim como uma função para libertar a imagem quando esta já não é necessária. Contém uma estrutura chamada bitmap que guarda todas as informações relativas ao mapa de bits carregado. **(Código obtido no Blog: Diffisual Minix por Henrique Ferrolho)**

car.c (Responsável: João Guarda)

Este módulo tem como função manipular a estrutura de dados Car. Esta estrutura contem informação sobre a posição, velocidade e desenho do carro. Este módulo contem funções para atualizar os carros, desenha-los no buffer e libertá-los quando já não são necessários.

colors.c (Responsável: João Guarda)

Este módulo contém apenas uma função para converter uma cor em formato rgb de 8 bits por cor para o formato rgb565.

frogger.c (Responsável: Ana Amaral)

Ficheiro que contém o *dispatcher.* Ou seja trata convenientemente cada interrupção. Encontra-se aqui a função driver_recive(). O dispatcher é um estrutura que contém informação variada sobre os periféricos sendo a mais importante a máscara de bits associada a cada periférico

game.c (Responsável: João Guarda)

Este modo corresponde a um estado de jogo. Como tal apresenta funções de criação, atualização e eliminação do estado de jogo. Conta ainda com uma função de desenho onde são desenhados todos os objetos relativos ao estado de jogo GAME.

graphics.c (Responsável: João Guarda)

Este módulo contém todas as funções relativas ao modo gráfico. Entrada e saída de diferentes modos assim como todas as funcionalidades relativas aos buffers.

highscores.c (Responsável: Ana Amaral)

Este modo corresponde a um estado de jogo. Como tal apresenta funções de criação, atualização e eliminação do estado. Conta ainda com uma função de desenho onde são desenhados todos os objetos relativos ao estado de jogo. Contém uma estrutura que contém a informação do estado.

keyboard.c (Responsável: Ana Amaral)

Este módulo contém todas as funções relativas ao teclado desenvolvidas nos Laboratórios. Contém também o handler de interrupções.

log.c (Responsável: João Guarda)

Este módulo tem como função manipular a estrutura de dados Log. Esta estrutura contem informação sobre a posição, velocidade, desenho do tronco. Este módulo contém funções para atualizar os carros, desenhá-los no buffer e libertá-los quando já não são necessários.

main.c (Responsável: João Guarda)

Ciclo principal de jogo do jogo. Chama o dispatcher até o programa ser encerrado.

MainMenu.c (Responsável: Ana Amaral)

Este módulo corresponde a um estado de jogo. Como tal apresenta funções de criação, atualização e eliminação do estado. Conta ainda com uma função de desenho onde são desenhados todos os objetos relativos ao estado de jogo. Contém uma estrutura que contém a informação do estado.

mouse.c (Responsável: João Guarda)

Este módulo contém todas as funções relativas ao rato desenvolvidas nos Laboratórios. Contém também a estrutura de dados relativa ao rato (baseado na estrutura do blog Diffisual Minix). Este objecto é único e acessível em qualquer parte do código. Contém também o handler de interrupções.

newRecord.c (Responsável: Ana Amaral)

Este módulo corresponde a um estado de jogo. Como tal apresenta funções de criação, atualização e eliminação do estado. Conta ainda com uma função de desenho onde são desenhados todos os objetos relativos ao estado de jogo. Contém uma estrutura que contém a informação do estado. Tem como função recolher o nome do jogador e adicionar a sua pontuação e data a tabela de *highscores*.

path.c (Responsável: João Guarda)

Este módulo contém funções de construção de caminhos para obter o caminho de um recurso a ser utilizado.

rectangle.c (Responsável: Ana Amaral)

Este módulo contém uma estrutura de dados que consiste na definição de retângulos. Especialmente útil na definição de botões em menus.

RTC.c (Responsável: João Guarda)

Este módulo contém todas as funções relativas ao Real-Time Clock desenvolvidas. Contém também o handler de interrupções.

settings.c (Responsável: João Guarda)

Este módulo corresponde a um estado de jogo. Como tal apresenta funções de criação, atualização e eliminação do estado. Conta ainda com uma função de desenho onde são desenhados todos os objetos relativos ao estado de jogo. Contém uma estrutura que contém a informação do estado. Tem como função alterar a resolução do ecrã.

strings.c (Responsável: João Guarda)

Este módulo tem como função carregar os ficheiros de imagens assim como desenhar strings no buffer. Percorrendo a string letra a letra e desenhando a letra correspondente.

timer.c (Responsável: Ana Amaral)

Este módulo contém todas as funções relativas ao timer desenvolvidas nos Laboratórios. Contém também a estrutura de dados relativa ao rato (baseado na estrutura do blog Diffisual Minix). Este objecto é único e acessível em qualquer parte do código. Contém também o handler de interrupções.

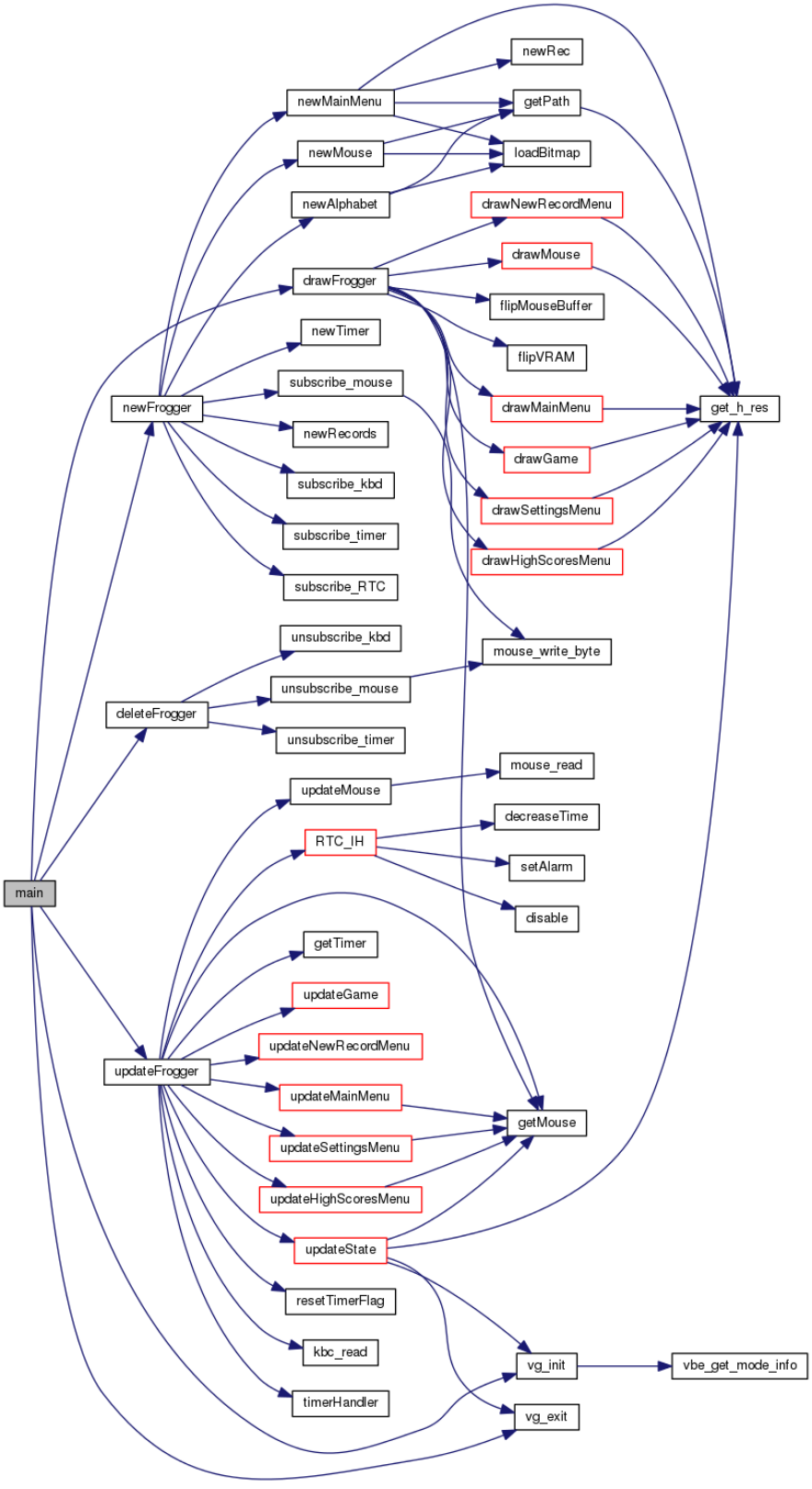
turtles.c (Responsável: João Guarda)

Este módulo tem como função manipular a estrutura de dados Turtles. Esta estrutura contem informação sobre a posição, velocidade, desenho das tartarugas. Este módulo contém funções para atualizar as tartarugas, desenhá-las no buffer e libertá-las quando já não são necessários.

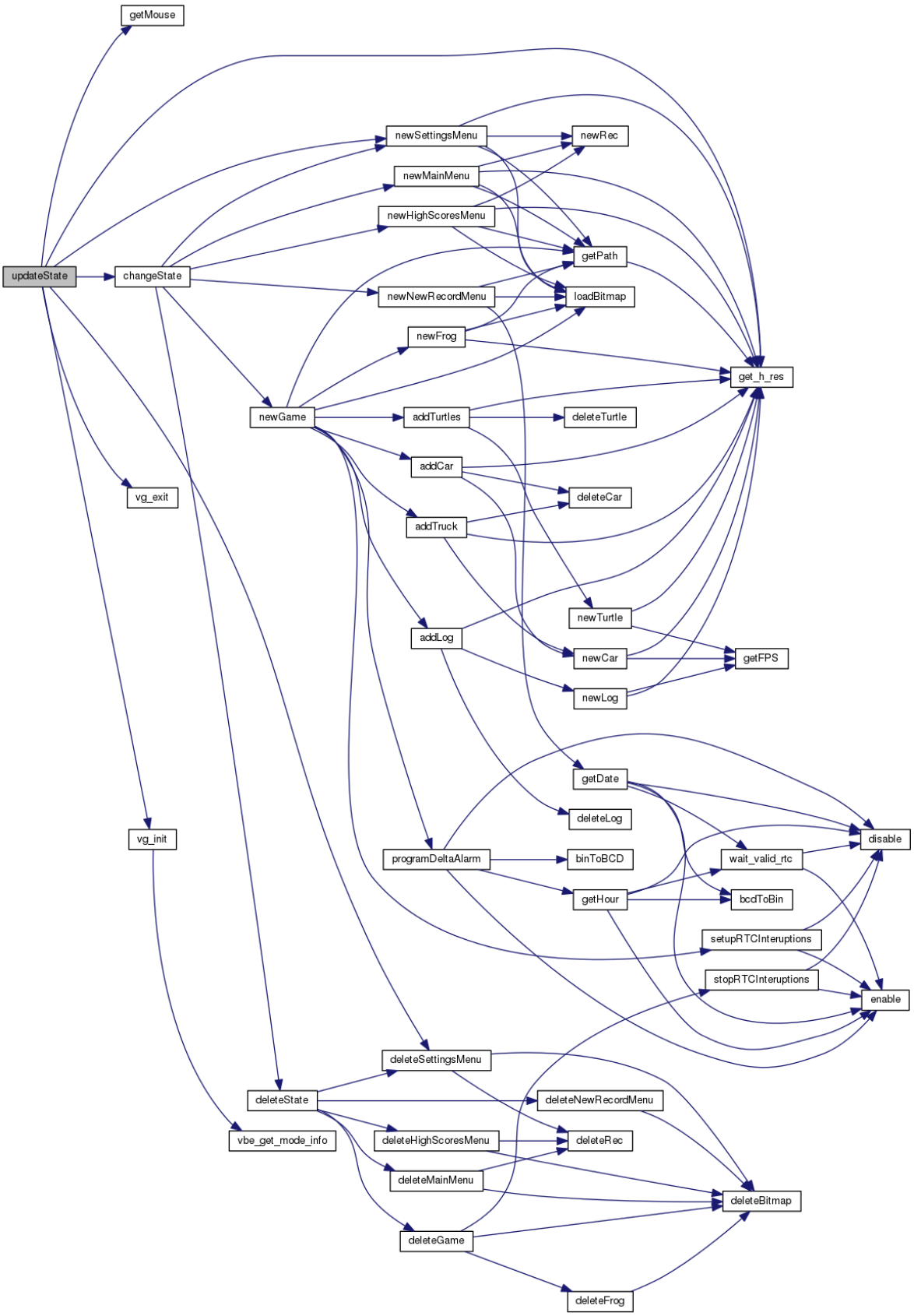
vbe.c (Responsável: João Guarda)

Este módulo contem uma função que nos indica as características de um determinado modo gráfico (desenvolvido nos Labs).

Diagrama de Invocações de Funções



Diagramas de máquinas de estados:



Detalhes quanto à implementação

A implementação do código sobre os respectivos periféricos é bastante simples visto que são amplamente debatidos tanto nas aulas teóricas como práticas.

A criação de máquinas de estado apesar de não ser um assunto muito aprofundado teve uma fácil implementação devido à grande informação disponível na internet sobre o assunto. O workshop do aluno Henrique Ferrolho também foi bastante elucidativo sobre a matéria. A principal ideia é guardar-mos a informação sobre o estado atual e quando quisermos mudar de estados substituímos a informação do estado anterior pela a que criarmos para o novo estado.

No projeto foi implementada uma espécie de programação orientada a objetos. No entanto esta implementação é um pouco ingénua. Visto C não ser uma linguagem orientada a objetos, com base nos conhecimentos de outras cadeiras e na informação leccionada na Unidade Curricular esta implementação é dentro do possível intuitiva.

A implementação de código orientado a eventos, é talvez o grande desafio do projeto. Este tema foi muito pouco debatido ao longo da Unidade Curricular e penso que a informação sobre o tema é escassa e muitas vezes insuficiente para dar robustez e legibilidade ao código. No entanto a solução adoptada acaba por ser boa dentro do possível. Tornar quase todos os objetos acessíveis em qualquer parte do código foi a medida adoptada, dando assim alguma independência aos estados em relação aos estados possíveis de jogo.

Sobre os temas não abordados nas aulas, o projeto utiliza ficheiros de formato .bmp e a sua manipulação é complexa. No entanto, tanto o blog do aluno Henrique Ferrolho assim como o próprio foram ajudas cruciais para a aprendizagem deste tipo de ficheiros. Penso que deveriam ser substituir os xpm's utilizados nas aulas pois são mais fáceis de utilizar com o MINIX se forem corretamente explicados.

Talvez a maior dificuldade de todo o projeto seja a manipulação de estruturas de dados. Estas diferem um pouco das estruturas de C++ até então as únicas consolidadas pelos alunos em cadeiras passadas. No entanto com alguma pesquisa e estudo são adquiridos conhecimentos mínimos para auxiliar os objectivos da UC.

Avaliação da Unidade Curricular

A Unidade Curricular no geral é bem estruturada e bastante pertinente na formação académica de um informático. Tem alguns aspectos bastante positivos como a vasta informação disponibilizada ao aluno assim como o auxílio dos monitores. A partilha de informação pelos alunos que já completaram a cadeira é muito importante pois há uma ponte entre o conhecimento do professor e o conhecimento do aluno.

Por outro lado, a Unidade Curricular é bastante exigente num momento inicial. É exigido aos estudantes conhecimentos em C e em sistemas Unix que são muito poucos ou inexistentes.

Instruções de instalação

Para correr o program deve efectuar os seguintes passos:

1) Fazer checkout da pasta proj;
2) Aceder a essa mesma pasta;
3) Adicionar a biblioteca liblm.a a pasta src;
4) Copiar ficheiro da pasta conf para /etc/system.conf.d;
5) Executar o comando: su (entrar em root);
6) Executar o comando: sh compile.sh;
7) Executar o comando: sh install.sh;
8) Executar o comando: sh run.sh;

Para futuras execuções basta correr o comando sh run.sh.

Método baseado no blog Diffusal Minix de Henrique Ferrolho.