FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Prison Breaker

Mestrado Integrado Em Engenharia Informática e Computação

Laboratório de Computadores

EIC0020

Turma 6, Grupo 8

Antero Gandra up201607926

Gil Teixeira up201505735

Índice

Introdução

Para este trabalho tivemos como objetivo a criação de um jogo interativo e desafiante, com aplicação dos conhecimentos ensinados nas aulas teóricas e adquiridos nas práticas laboratórios da unidade curricular de Laboratório de Computadores. Neste relatório começaremos por descrever o jogo em questão, seguido da maneira como foi implementado. Depois para concluir expomos a arquitetura do programa em si acompanhado por um grafo das suas funções e de como estas se relacionam.

Instruções do Jogo

Baseado num jogo *singleplayer* para plataformas mobile intitulado NOTNOT, "Prison Breaker" consiste numa fuga de uma prisão claustrofóbica através da aplicação de instruções lógicas. O objetivo do jogo é tentar fugir o máximo de tempo possível, sem errar nunca as instruções enviadas para o ecrã.

O quadrado não se conseguindo "enquadrar" na sua prisão elíptica decide escapar de modo a não se converter num circulo. A escolha das formas foi baseada no famoso problema de Tarski, o problema do circulo quadrado. Sem o conceito de infinito é impossível resolver o problema, tal como é impossível o quadrado escapar da prisão.

Menu Inicial

Quando o programa é iniciado o menu surge e é dado a escolher ao utilizador, através do uso do teclado (nomeadamente das teclas Enter e Esc) o que pretende fazer (jogar ou sair do programa).

Outros inputs para além dos indicados são ignorados pelo programa.

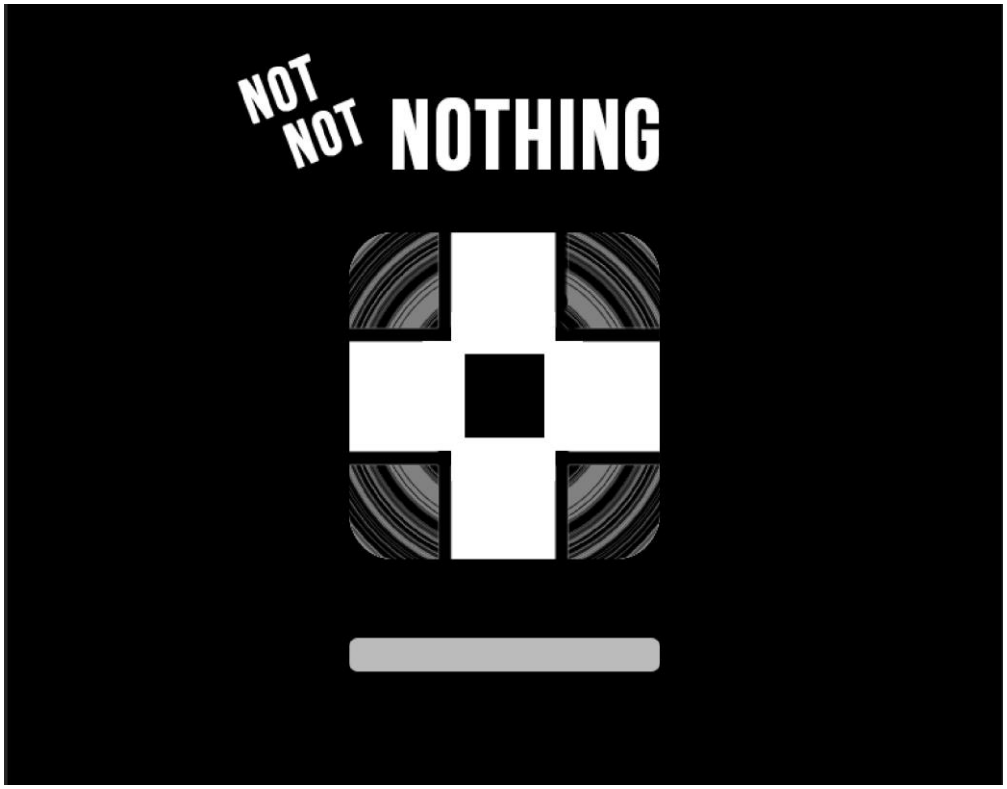
Jogo

Após ser pressionada a tecla Enter o jogo inicia-se sendo demonstrado no ecrã uma divisão de uma prisão, numa interseção de 2 corredores, onde o utilizador terá de obedecer às instruções impressas em cima para garantir a sobrevivência do quadrado.

As Teclas de seleção de jogo são “A” para mover para a esquerda, “S” para mover para baixo, “D” para mover para a direita, e “W” para mover para cima.

Sempre que ocorra uma fuga de uma divisão com sucesso conta mais um ponto na pontuação total.

Exemplo de Output



Fim

O jogo declara-se finalizado aquando os seguintes cenários:

- O jogador erra na execução da instrução;
- O jogador demora muito tempo a executar uma instrução.

Concluído o jogo é impresso no ecrã a informação *"You got caught!"* (Foste apanhado) com as opções demonstradas também no menu inicial, para voltar a jogar ou sair do jogo, e a pontuação final.

Representação da Pontuação

De modo a respeitar a filosofia minimalista atrás do design deste jogo, foi nossa decisão demonstrar a pontuação do jogo apenas no fim deste. Deste modo o jogador não se distrai tão facilmente das instruções pedidas e a interface impressa no ecrã mantém-se limpa e organizada.

Implementações no Projeto

	Utilidade	Interrupção
Timer	Atualizar o ecrã e animações	☑
Rato	Jogar	☑
Teclado	Selecionar no menu e jogar	☑
Placa Gráfica	Desenhar no ecrã	☒

Detalhes das Implementações

Timer

O timer, mais precisamente o timer 0, é utilizado para atualizar o estado do jogo a uma taxa constante. Assim, é chamada a função timerHandler(Timer* timer) a cada interrupção, que atualiza o timer do jogo.

Rato

A utilização do rato está limitada neste projeto às direções que podem ser tomadas no jogo, servindo de manípulo do jogo. Assim, a cada interrupção é chamada a função mouseHandler(Mouse* mouse), que atualiza o rato do jogo.

Teclado

O uso do teclado às teclas ESC, ENTER, W, A, S, D. servindo de interface de comunicação e de jogo. Assim, a cada atualização é chamada a função kbd_handler(), que atualiza a tecla recebida pelo jogo.

Placa Gráfica

A placa gráfica é usada para a apresentar a informação gráfica do jogo no modo 0x114, ou seja, é utilizada uma resolução de 800x600 no modo RGB 5:6:5.

Embora as imagens desenvolvidas para este projeto tenham sido criadas por os autores deste projeto, para as carregar, visualizar e apagar imagens foi utilizado código autorizado da autoria de Henrique Ferrolho, estando o mesmo disponível no seu blog Difusal.

Por fim, de forma a evitar flickering e tearing, foi utilizada a técnica de Double Buffering em que todas as imagens são escritas num segundo buffer e apenas quando toda a informação nele for escrita é copiada para o frame buffer.

Arquitetura do Programa

Abordagem Utilizada:

Começamos primeiro por desenvolver em baixo nível, as funções mais essenciais, de modo a poder implementar a lógica mais básica do jogo. Depois quanto mais perto do objetivo final, viemos a preocupar-nos com a programação a alto nível e o lado estético do jogo.

Macros:

i8254 – Inclui o ficheiro "i8254.h". Contém macros essenciais para a utilização do *timer*. Desenvolvidas nas aulas práticas do laboratório 2.

Peso no Projeto: 1%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

i8042 - Inclui o ficheiro "i8042.h". Contém macros fundamentais para o funcionamento do teclado e rato. Desenvolvidas nas aulas práticas do laboratório 3 e 4.

Peso no Projeto: 1%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Defs – Inclui o ficheiro "defs.h". Contém as definições do VBE, acerca do modo de vídeo e da placa gráfica. Desenvolvidas nas aulas práticas do laboratório 5.

Peso no Projeto: 1%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Periféricos:

Timer - Inclui o ficheiro “timer.h” e “timer.c”. Contém funções de uso e configuração do timer. Desenvolvidas nas aulas práticas do laboratório 2, ao qual se acrescentou uma estrutura de dados Timer que guarda informações acerca do mesmo.

Peso no Projeto: 3%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Mouse - Inclui o ficheiro “mouse.h” e “mouse.c”. Contém funções para o funcionamento do rato. Desenvolvidas nas aulas práticas do laboratório 4, ao qual se acrescentou uma estrutura de dados Mouse que guarda informações acerca dos movimentos do mesmo.

Peso no Projeto: 3%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Keyboard - Inclui o ficheiro “kbd.h” e “kbd.c”. Contém funções para o funcionamento do teclado. Desenvolvidas nas aulas práticas do laboratório 3.

Peso no Projeto: 3%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

VBE - Inclui o ficheiro “vbe.h” e “vbe.c”. Contém funções com informações sobre a placa gráfica e do modo de vídeo utilizado. Desenvolvidas nas aulas práticas do laboratório 5.

Peso no Projeto: 3%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Impressão:

Bitmap - Inclui o ficheiro “bitmap.h” e “bitmap.c”. Contém funções para carregar e desenhar imagens do tipo Bitmap, no modo RGB 5:6:5, no ecrã, assim como libertar os recursos utilizados pelas mesmas.

(Autoria de Henrique Ferrolho: link)

Vídeo_gr - Inclui o ficheiro “vídeo_gr.h” e “vídeo_gr.c”. Contém funções para configurar e utilizar a placa gráfica. Desenvolvidas nas aulas práticas do laboratório 5, ao qual foi acrescentado um segundo buffer de forma a utilizar a técnica de Double Buffering.

Peso no Projeto: 5%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Lógica do Jogo:

Main - Inclui o ficheiro “main.c”. Inicializa a aplicação no modo 0x114 e a estrutura de dados PrisonBreaker.

Peso no Projeto: 5%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Gamestate - Inclui o ficheiro “gamestate.h” e “gamestate.c”. Guarda a informação relativamente aos estados possíveis do Jogo, sendo estes 3, *MAINMENU, LOSEMENU* e *GAME*. Contém funções que atualizam o estado do jogo conforme a interrupção recebida.

Peso no Projeto: 15%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Level – Inclui o ficheiro “level.h” e “level.c”. Guarda a informação relativamente a 1 nível, nomeadamente a instrução e as direções aceites pela mesma. Processa, também, se uma direção é ou não aceite pelo nível.

Peso no Projeto: 15%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Square - Inclui o ficheiro “square.h” e “square.c”. Guarda a informação acerca do quadrado que se move no centro do ecrã, nomeadamente as suas coordenadas em X e em Y.

Peso no Projeto: 10%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Game - Inclui o ficheiro “game.h” e “game.c”. Guarda a informação relativamente ao jogo, nomeadamente:

- O estado em que está, dentro de 3 possíveis: `PLAYING, WAITING, LOSE`;
- Os níveis do jogo;
- O nível atual;
- O score do jogo.

Sendo, assim, também responsável pela visualização do Nível atual, do quadrado de jogo, da barra de tempo e do score do utilizador.

Peso no Projeto: 20%

Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

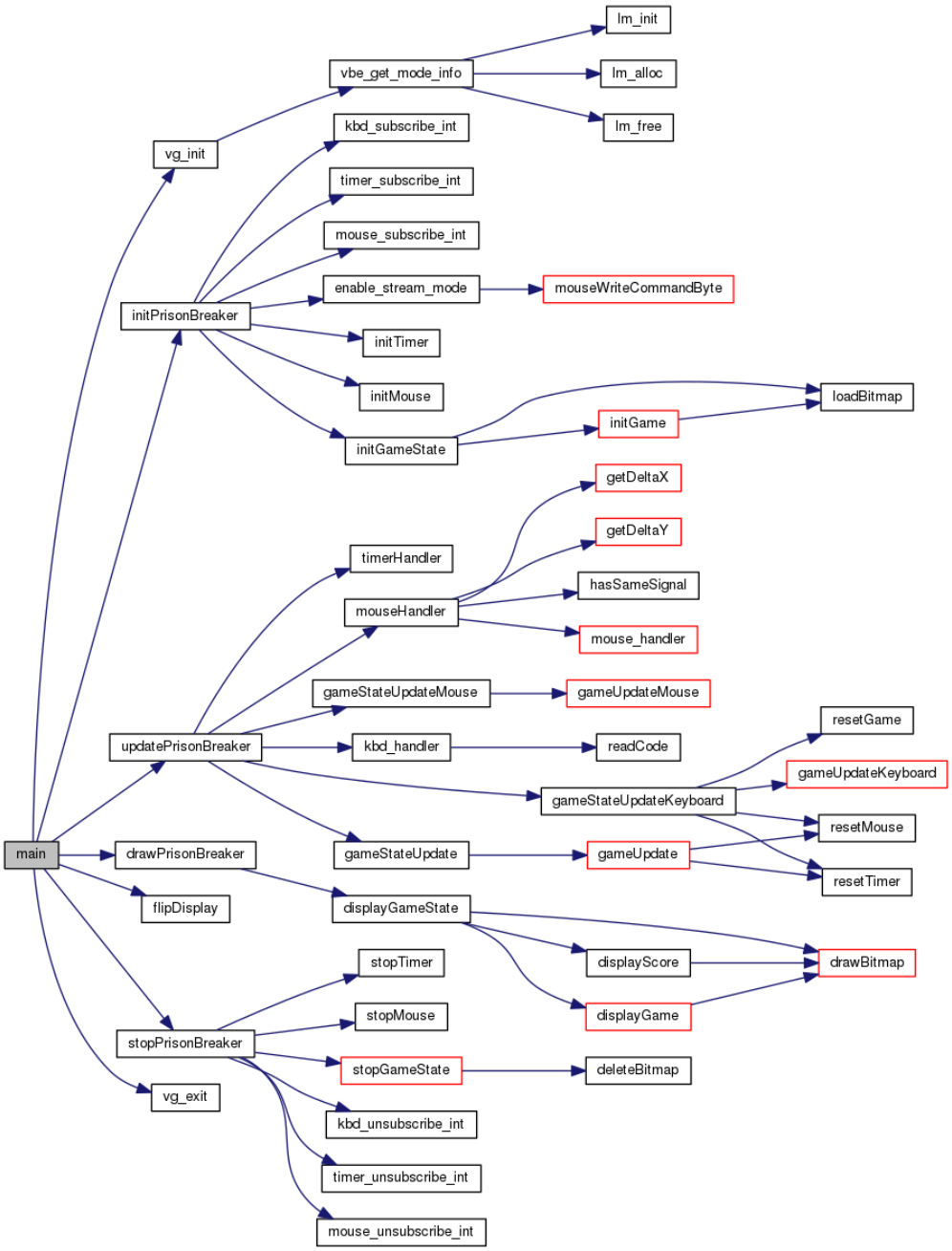
PrisonBreaker - Inclui o ficheiro “prisonBreaker.h” e “prisonBreaker.c”. Módulo de mais alto-nível do projeto, processa as interrupções transmitindo essa informação ao módulo GameState.

Peso no Projeto: 15%

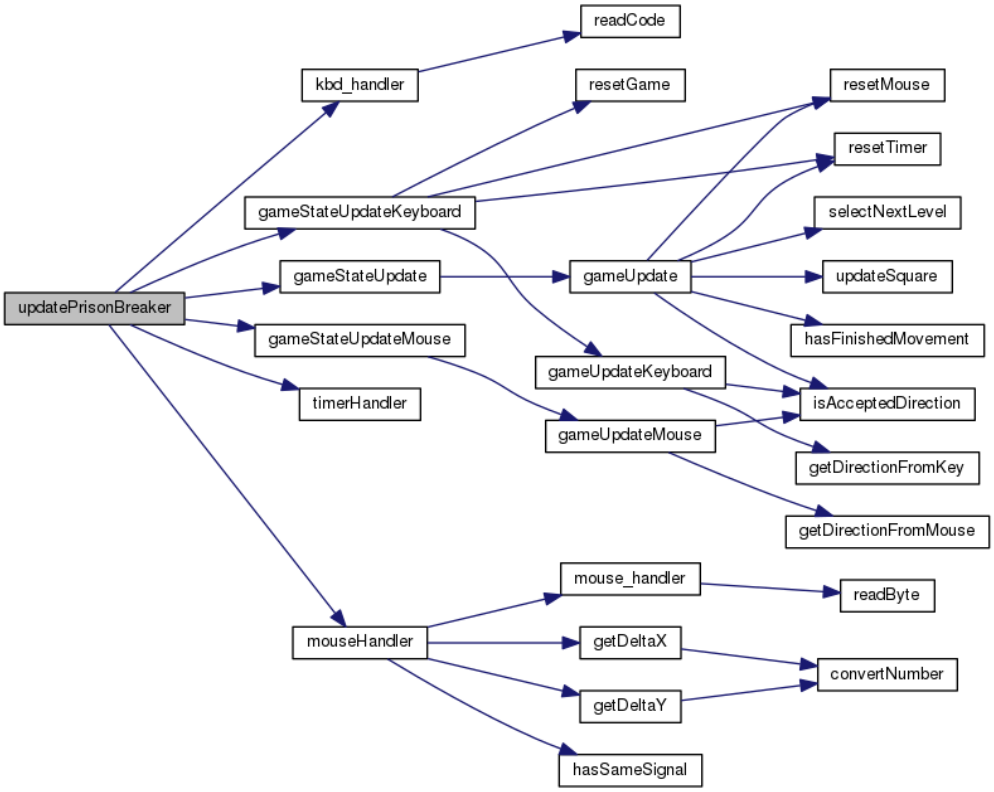
Os autores do projeto trabalharam de forma equitativa na implementação deste módulo.

Function Call Diagrams

Function call graph para a função main.



Function call graph para a função updatePrisonBreaker, responsável pelo processamento das interrupções.



Conclusões

Com o uso total dos laboratórios conseguimos de uma maneira simples e eficaz conceber o jogo que se demonstrou uma tarefa complicada, uma vez que foi necessário muito tempo e esforço da nossa parte.

Os resultados obtidos foram interessantes sendo que agora é possível disfrutar do jogo de maneira intuitiva e didática, com uma demonstração visual apelativa, quer do jogo em si, quer a sua interface.

Como a quantidade de instruções é limitada e, face o calendário avaliativo muito reduzido e apertado, isso limitou-nos a só poder criar 3 tipos de níveis. Níveis de valores mais elevados teriam como consequência haver regras muito complexas que podiam causar grandes discordâncias na maneira como o programa se comporta, complicando bastante mais o jogo.

Tirando isso, o nosso programa demonstra-se como uma tentativa bastante próxima (mas única, da sua maneira especial) do jogo original, tendo em conta todas as restrições que tem de obedecer.

A capacidade de esquematizar o problema demonstrou-se como sendo a nossa maior dificuldade, uma vez que, dado que não existem muitos jogos parecidos, isso implicou uma criação de uma lógica de jogo de raiz, com poucas fontes onde nos apoiar.

Este trabalho demonstrou-se de elevada importância para projetos futuros, nomeadamente na área da organização de projetos e criação de jogos. Além de nos familiarizarmos com a linguagem C e com o sistema operativo Minix, conseguimos, assim, implementar um jogo de raciocínio e reação muito interessante.

Apêndice I - Instruções de instalação

De modo a simplificar a instalação dos recursos essenciais ao funcionamento do programa optamos por implementar shell scripts.

Como correr o programa:

1) Entrar no diretório do projeto prisonBreaker;
2) No modo *superuser* (su) correr:
    a. o script install.sh;
    b. o script compile.sh;
    c. o script run.sh.