U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MINIX TROUBLE

MIEIC - Laboratório de Computadores

Turma 4 – Grupo 07

Dezembro 2017

Eduardo Silva – up201603135@fe.up.pt

Pedro Lopes – up201603557@fe.up.pt

Índice

1. Introdução 3
2. Instruções de utilização 4
   2.1. Menu Principal 4
   2.2. Gameplay 4
   2.3. Game Over 6
3. Estado do Projeto 7
   3.1. Timer 7
   3.2. Teclado 7
   3.3. Placa Gráfica 8
   3.4. Rato 8
4. Estrutura e Organização do Código 9
   4.1. ASM.S 9
   4.2. Bitmap.c 9
   4.3. Graphics_reusable.c 9
   4.4. M_IH.S 10
   4.5. main.c 10
   4.6. Mouse_reusable.c 10
   4.7. menu.c 11
   4.8. movement.c 11
   4.9. server.c 11
   4.10. timer_kbd_reusable.c 12
   4.11. vbe.c 12
   4.12. video_gr.c 12
5. Detalhes de Implementação 14
   5.1. Remover fundo em imagens .bmp 14
6. Conclusões 15
7. Apêndice: Instruções de Instalação 16

1. Introdução

O projeto aqui descrito, elaborado no âmbito da unidade curricular de Laboratório de Computadores, consiste num jogo que, embora original, é inspirado de uma forma mais geral pelos jogos *arcade* dos anos 80 e 90. É também inspirado, como o título sugere, por esta cadeira em particular.
Certas alterações tiveram que ser feitas em relação à proposta inicial, devido, principalmente, ao não enquadramento na ideia que tínhamos originalmente. No entanto, a fórmula principal mantém-se.
Em *Minix Trouble*, o jogador controla uma personagem cujo objetivo é evitar que todos os servidores presentes naquela área estejam desativados ao mesmo tempo. Para evitar isto, o jogador tem que voltar a ativar eventuais servidores que estejam em baixo, cada um individualmente. À medida que o tempo passa, esta tarefa torna-se cada vez mais difícil, logo serão atribuídos mais pontos ao jogador de acordo com o tempo em que ele consegue ter pelo menos um servidor operacional.

2. Instruções de utilização

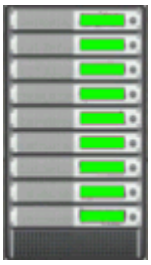
2.1. Menu Principal

Este menu é apresentado ao jogador aquando do início do programa. Aqui, ele pode escolher em iniciar o jogo em si ou sair do jogo. Para escolher qualquer uma delas, o jogador deve arrastar o rato até ele se encontrar sob uma das opções e aí pressionar o botão esquerdo do rato para a selecionar. Devemos mencionar que há um pequeno atraso do cursor em relação ao movimento real do rato, por isso é aconselhado mexe-lo em intervalos, esperando que o cursor se mova para o sítio pretendido.

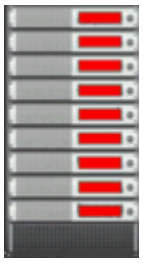
2.2. Gameplay

O espaço de jogo consiste numa sala com 4 servidores posicionados próximos dos cantos desta.

O jogador pode movimentar a personagem através das teclas WASD (W para a frente, S para trás, A para a esquerda e D para a direita).
No início, o jogador terá alguns segundos para se habituar aos controlos antes do primeiro servidor (seja ele qual for) ser desativado.

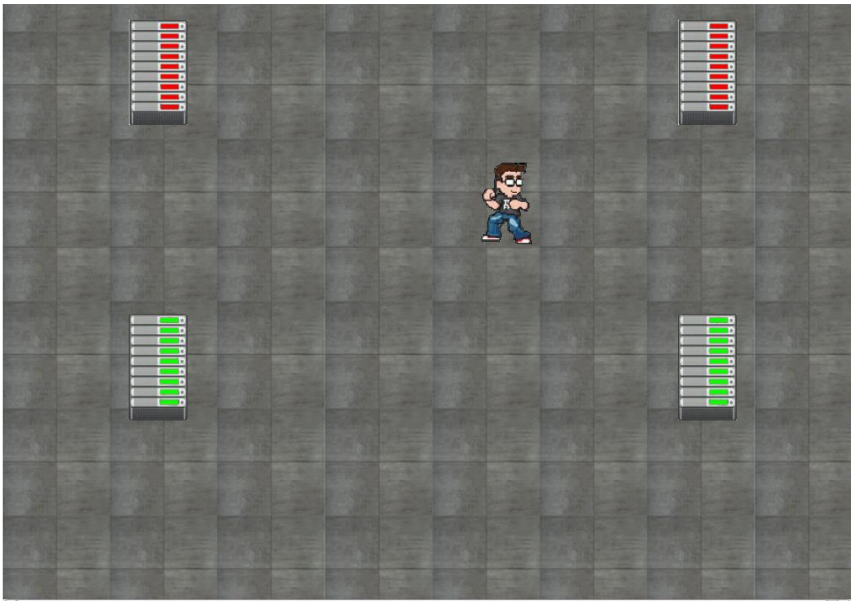


Servidor Ativo



Servidor Desativo

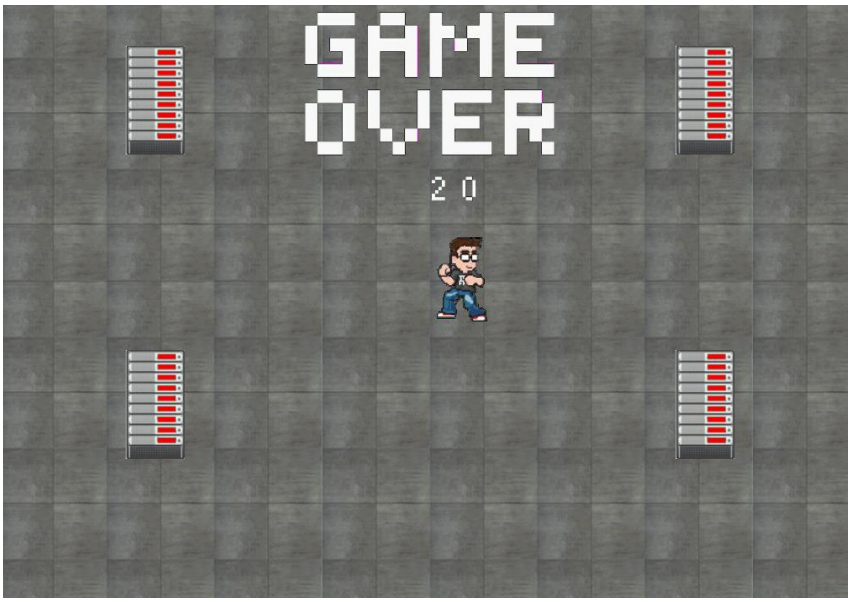
Aí, o jogador deverá proceder até esse servidor e colocar-se junto a este. Para o voltar a por operacional, o jogador deve carregar na tecla E. Alguns segundos depois, outro servidor (ou possivelmente o mesmo) voltará a desligar-se e o jogador deverá repetir o processo. À medida que o tempo passa, esta tarefa torna-se cada vez mais difícil pois o tempo entre cada *crash* dos servidores diminui.



O jogador vai então acumulando pontos, quer por arranjar servidores, quer pelo tempo em que ele consegue manter pelo menos um servidor operacional. Quando esta condição deixar de se verificar, ou seja, quando todos os servidores estiverem em baixo, o jogo acaba. Note-se que o jogador pode abandonar aquela sessão a qualquer momento pressionado a tecla ESC.

2.3. Game Over

Quando o jogador perder o jogo, é-lhe apresentado o ecrã de fim de jogo com a pontuação atingida.



O jogador pode então pressionar a tecla ESC para voltar ao menu inicial.

3. Estado do Projeto

Todos os periféricos mencionados na especificação foram usados. No entanto, não foram adicionados personagens inimigas, uma hipótese indicada também na especificação.

Periférico	Resumo do uso do periférico	Interrupção
Timer	Atualização das variáveis e controlo dos servidores	Sim
Teclado	Movimento da personagem e suas ações	Sim
Placa Gráfica	Exibe o jogo em si	Não
Rato	Movimento no menu e seleção das opções	Sim

3.1. Timer

O timer é usado sobretudo para gerir o que aparece no ecrã, atualizando a cada interrupção o que mostra ao utilizador.
Também é usado na gestão do tempo entre cada *crash* dos servidores, visto que esse intervalo diminuiu ao longo do jogo. Isto é feito através de vários contadores que são atualizados pelas interrupções do timer.
A atualização do score do jogador é feita parcialmente pelo timer também, mais uma vez através dos contadores implementados.

Funções onde é usado (main.c):

- int cycle(State_Machine *st)
- void Main_Menu(State_Machine *st)

Outros ficheiros: timer_kbd_reusable.c

3.2. Teclado

O teclado é usado na movimentação e ações da personagem. A cada interrupção deste, é verificado se o *scancode* obtido corresponde a alguma das teclas de ação, e em caso afirmativo, efetua a ação correta.

Funções onde é usado (main.c):

- int cycle(State_Machine *st)
- void game_over(int score, State_Machine *st)

Outros ficheiros: timer_kbd_reusable.c

3.3. Placa Gráfica

A placa gráfica é usada para exibir todos os gráficos do jogo. Neste caso foi usado o modo 117h (1024 x 768) que usa uma palete de 16 bits (RGB), possuindo até 64k cores.
Foi também implementado *double buffering* de modo a evitar o fenómeno de *flicker* durante o decorrer do jogo e, principalmente, no menu principal.
Além disto, é verificada a deteção de colisões entre bitmaps durante o jogo em si de modo a saber se a personagem está em posição de arranjar os servidores.
No ecrã final do jogo são também usadas *fonts* para imprimir o score final do jogador.
A placa gráfica é usada praticamente em todas as funções no main.c, assim como nos seguintes ficheiros:

- Bitmap.c
- Graphics_reusable.c
- vbe.c
- video_gr.c
- menu.c
- movement.c
- server.c

3.4. Rato

O rato é usado no menu principal, sendo registado o seu movimento no ecrã. Este movimento tem uma falha particular, no entanto. Este só é registado no programa alguns segundos depois, ou seja, há um ligeiro *delay* entre o momento em que o rato é deslocado e o momento em que o cursor se move na direção pretendida. Infelizmente não conseguimos resolver este *bug* mesmo após termos consultado várias fontes.
Os botões deste também são usados aquando da escolha das opções no menu.

Funções onde é usado (main.c):

- void Main_Menu(State_Machine *st)

Outros ficheiros:

- Mouse_reusable.c

4. Estrutura e Organização do Código

4.1. ASM.S

Este módulo contém o código *assembly* para o *handler* do timer. Tem como objetivo incrementar uma variável de modo a contar o tempo que passou desde o início do jogo de modo a atualizar certas variáveis noutros módulos.

Membro(s) do grupo responsável: Eduardo Silva

Peso do módulo: 1%

4.2. Bitmap.c

Este módulo é da autoria de Henrique Ferrolho e está disponível em:

http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html

Permite-nos carregar imagens no formato .bmp de modo a serem usadas e trabalhadas.
No entanto foram feitas algumas alterações, incluindo a criação de uma nova função de mapeamento que ignora pixéis de uma determinada cor, e a alteração da função original para mapear numa *buffer* auxiliar em vez da original.

Membro(s) do grupo responsável: Eduardo Silva

Peso do módulo: 8%

4.3. Graphics_reusable.c

Este módulo contém a função void set_mode(unsigned short mode) que é usada no decorrer da função vg_init() de modo a inicializar a placa gráfica no modo pretendido.

Membro(s) do grupo responsável: Eduardo Silva

Peso do módulo: 2%

4.4. M_IH.S

Este módulo contém o *handler* das interrupções do rato em *assembly*. Este lê o *Status Register* para verificar se *a Out Buffer* contém informação disponível e verifica se ocorrem erros de *Parity* ou de *Timeout*.

Membro(s) do grupo responsável: Eduardo Silva

Peso do módulo: 5%

4.5. main.c

Este é o módulo principal do programa no sentido em que as funções mais abrangentes se encontram aqui, além de certas estruturas chave tal como a máquina de estados (*State_Machine*) que possui, para todos os efeitos, o fluxo de controlo do programa.
Outras funções a ter conta:

- int cycle(State_Machine *st): Esta função engloba o ciclo do jogo em si, sendo aqui atualizadas várias variáveis tais como a posição da personagem, os pontos obtidos e o tempo decorrido.

- void Main_Menu(State_Machine *st): Esta função contém o ciclo do menu principal, sendo atualizada a posição do rato (e estado dos seus botões).

- void game_over(int score, State_Machine *st): Esta função é responsável por mostrar o ecrã final do jogo que inclui a pontuação obtida nessa sessão.

Membro(s) do grupo responsável:

- Eduardo Silva: void state_machine(State_Machine *st), int cycle(State_Machine *st), void Main_Menu(State_Machine *st), void game_over(int score, State_Machine *st);
- Pedro Lopes: void print_score(int score, Bitmap *bitmaps_endScreen[ ]);

Peso do módulo: 20%

4.6. Mouse_reusable.c

Este módulo contém as funções relacionadas com o rato, incluindo a função responsável por mostrar o cursor no ecrã, a função de subscrição das interrupções e o *handler* respetivo, entre outras.

Membro(s) do grupo responsável: Eduardo Silva

Peso do módulo: 8%

4.7. menu.c

Este módulo contém as funções responsáveis por criar o menu principal e por carregar os bitmaps necessários em cada estado principal do programa.

Membro(s) do grupo responsável: Pedro Lopes

Peso do módulo: 5%

4.8. movement.c

Neste módulo estão contidas as funções usadas para criar o mapa de jogo principal, para verificar possíveis colisões entre a personagem e os servers e por mover a personagem.

Membro(s) do grupo responsável:

- Eduardo Silva: void move(short *ch_x, short *ch_y, Bitmap *bitmap_array [ ], Server *server_array [ ], int n_servers);
- Pedro Lopes: void draw_map(Bitmap *bitmaps_game[ ], Server *server_array [ ], int n_servers), int collision_with_server(Server *server, short ch_x, short ch_y), int check_colision(short ch_x, short ch_y, Server *server_array [ ], int n_servers);

Peso do módulo: 13%

4.9. server.c

Este módulo contém a struct server e as funções relacionadas com esta. Estas são responsáveis por criar e destruir estes “objetos”, criar um array deles, decidir, com um fator aleatório, qual dos servidores deitar abaixo, verificar quantos estão ativos e permitir o arranjo destes.

Membro(s) do grupo responsável: Pedro Lopes

Peso do módulo: 12%

4.10.timer_kbd_reusable.c

Este módulo contém as funções relacionadas com o timer e com o teclado, nomeadamente as funções de subscrição das interrupções para cada dispositivo, assim como os respetivos *handlers* em C.

Membro(s) do grupo responsável: Eduardo Silva

Peso do módulo: 10%

4.11.vbe.c

Este módulo contém as funções responsáveis por obter as informações relativas ao modo gráfico usado.

Membro(s) do grupo responsável: Pedro Lopes

Peso do módulo: 8%

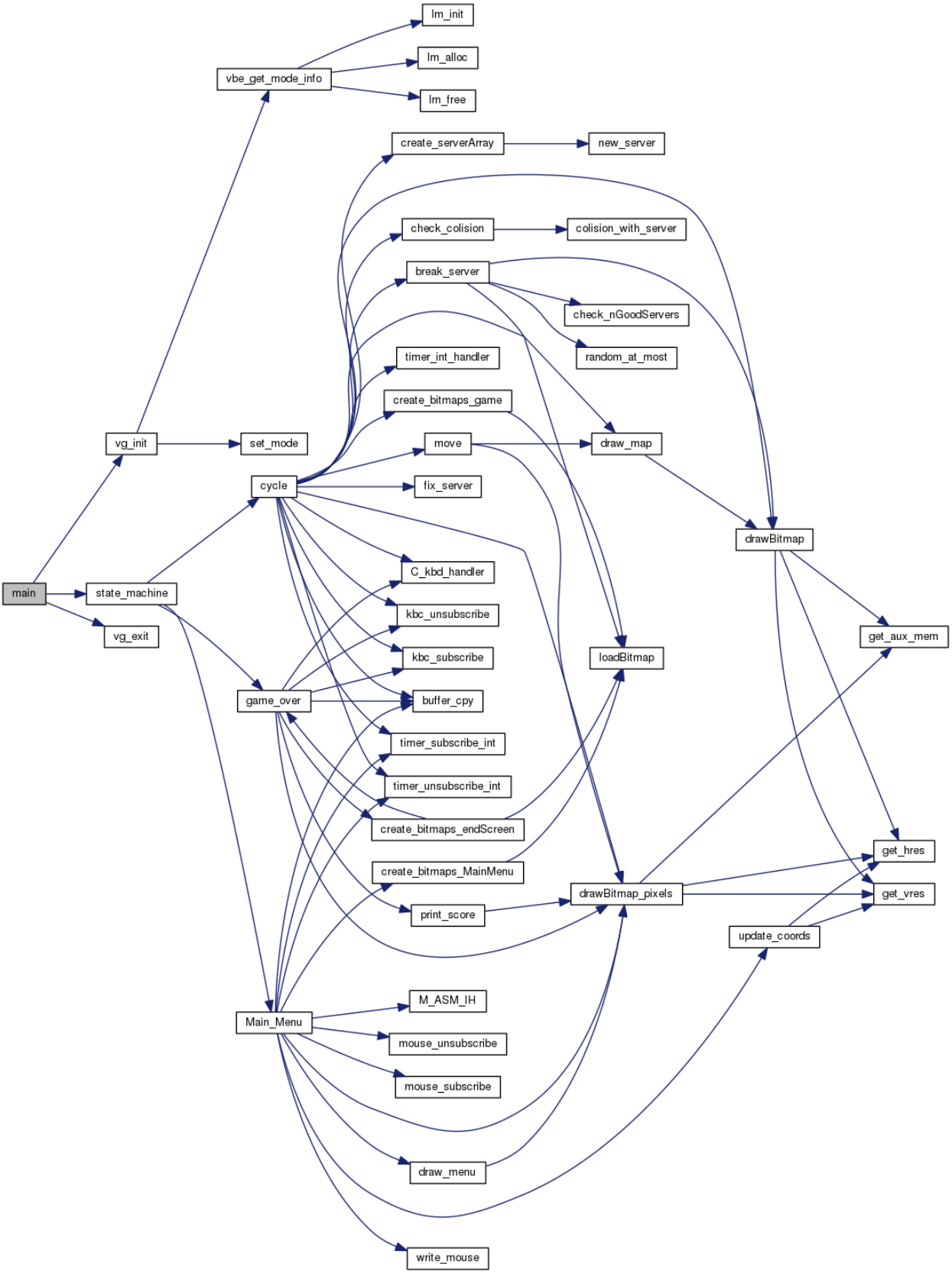
4.12.video_gr.c

Este módulo contém as variáveis onde são guardadas as propriedades do modo gráfico usado (resolução horizontal, vertical, bits por pixel) e o apontador para o *buffer* de memória RAM.
Contém também a função de cópia dos dados do *buffe*r auxiliar para o buffer da memória RAM e a função do último *lab*, void *vg_init (unsigned short mode).

Membro(s) do grupo responsável:

- Eduardo Silva: unsigned get_hres(), unsigned get_vres(), unsigned get_bitpp(), char* get_aux_mem(), void buffer_cpy(), char *read_xpm(char *map[ ], int *wd, int *ht), int vg_exit();
- Pedro Lopes: void *vg_init(unsigned short mode);

Peso do módulo: 8%


lm_init
lm_alloc
lm_free
vbe_get_mode_info
create_serverArray
new_server
check_colision
colision_with_server
break_server
check_nGoodServers
random_at_most
timer_int_handler
create_bitmaps_game
vg_init
set_mode
move
draw_map
cycle
fix_server
drawBitmap
C_kbd_handler
main
state_machine
kbc_unsubscribe
get_aux_mem
vg_exit
kbc_subscribe
loadBitmap
game_over
buffer_cpy
timer_subscribe_int
timer_unsubscribe_int
create_bitmaps_endScreen
get_hres
create_bitmaps_MainMenu
drawBitmap_pixels
get_vres
print_score
update_coords
M_ASM_IH
Main_Menu
mouse_unsubscribe
mouse_subscribe
draw_menu
write_mouse

5. Detalhes de Implementação

5.1. Remover fundo em imagens .bmp

As imagens em formato .bmp não suportam transparência tal como outros tipos de imagens (PNG e GIF, por exemplo). No entanto, para utilizar imagens complexas (isto é, não retangulares) de modo adequado, torna-se necessário, por vezes, eliminar o fundo destas.
Para obter este efeito, decidimos criar uma nova função em Bitmap.c:

- void drawBitmap_pixels(Bitmap *bmp, int x, int y, Alignment alignment)

Esta função, embora inspirada na original da biblioteca respetiva, opera de forma diferente no que diz respeito à cópia dos dados da imagem para o *buffer* escolhido. Em vez de percorrer a imagem linha a linha, percorre-a pixel a pixel, comparando o valor de cada um com outro valor definido anteriormente. Esse valor representa uma cor específica que, quando encontrada, deve ser ignorada, passando à seguinte iteração do ciclo. No entanto, a imagem tem de ser alterada previamente (através de outro *software*) de modo a conter essa cor nos pixéis que desejamos “eliminar”. Deste modo, apenas são passados os pretendidos, criando assim um efeito de fundo transparente.
Esta função, embora útil para conseguir este efeito, deve ser usada em conjunto com a original, dependendo da imagem, visto que como ela é mais lenta a copiar os valores para o *buffer* corre-se o risco de a transição entre *frames* se tornar demasiado percetível (*flicker*). O uso combinado das duas funções ajuda portanto a diminuir este fenómeno.

6. Conclusões

Consideramos que esta disciplina é, de facto, importante pois não só nos ensina como funcionam alguns dos dispositivos I/O mais comuns mas também nos prepara para desafios futuros em que teremos que tomar cada vez mais a iniciativa para aprender e compreender conceitos novos.
O uso da linguagem C (e não C++) também é de caráter importante, porque embora ambas tenham as suas semelhanças, esta não tinha sido bem explorada pelas unidades curriculares do ano anterior.
No entanto, sentimos que também há certos aspetos que podem melhorar, tal como a carga de trabalho necessária fora das aulas. Compreendemos que, no tempo que temos disponível para esta disciplina, é necessário que esta seja maior que o habitual. Contudo, pode rapidamente tornar-se cansativa e frustrante, especialmente em conjunto com outras unidades curriculares.
Portanto, a principal sugestão que aqui deixamos será reduzir ou facilitar alguns dos labs de modo a tornar esta disciplina mais acessível.
Além disto, uma melhor primeira abordagem à disciplina era também bem-vinda, visto que, mesmo com toda a documentação fornecida, tanto nas aulas teóricas como nas *handouts* dos labs, é ainda bastante confuso, no início, o que devemos fazer e como.

7. Apêndice: Instruções de Instalação

Para correr o programa é necessário colocar o repositório do projeto no *path* /home/lcom/svn
Também será necessário colocar o ficheiro de configuração fornecido em /etc/system.conf.d e colocar as bibliotecas necessárias (presentes no módulo dos ficheiros do RedMine) em home/lcom/svn/proj/src
Finalmente, deve-se executar o comando make em home/lcom/svn/proj/src e posteriormente iniciar o programa com service run `pwd`/proj