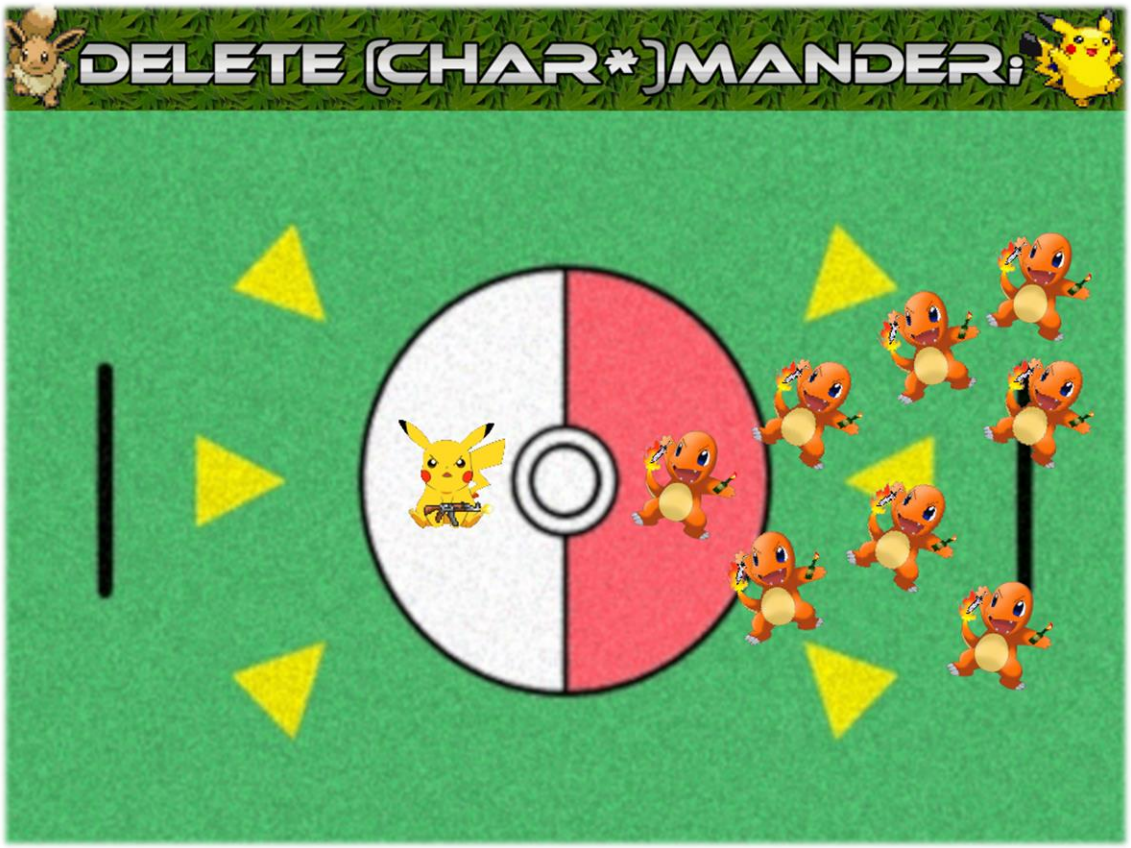
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO



Laboratório de Computadores 18/19

Turma 1 – Grupo 05

Trabalho realizado por:

Henrique Santos - up201706898@fe.up.pt

Luís Fernandes - up201706910@fe.up.pt

Índice

Instruções do Utilizador

- Menu

Quando se inicia o projeto, o primeiro ecrã com que o utilizador se depara é o ecrã de menu. Neste, o utilizador pode selecionar uma de três opções:

- **Play (1P)** – Inicia um jogo a um jogador;
- **Play (2P)** – Inicia um jogo a dois jogadores caso já haja um jogador à espera, caso contrário muda de ecrã para um ecrã de espera do jogador 2;
- **Exit** – Termina o programa.

A opção deve ser selecionada com o rato, ou para terminar o programa pode também usar a tecla "escape".

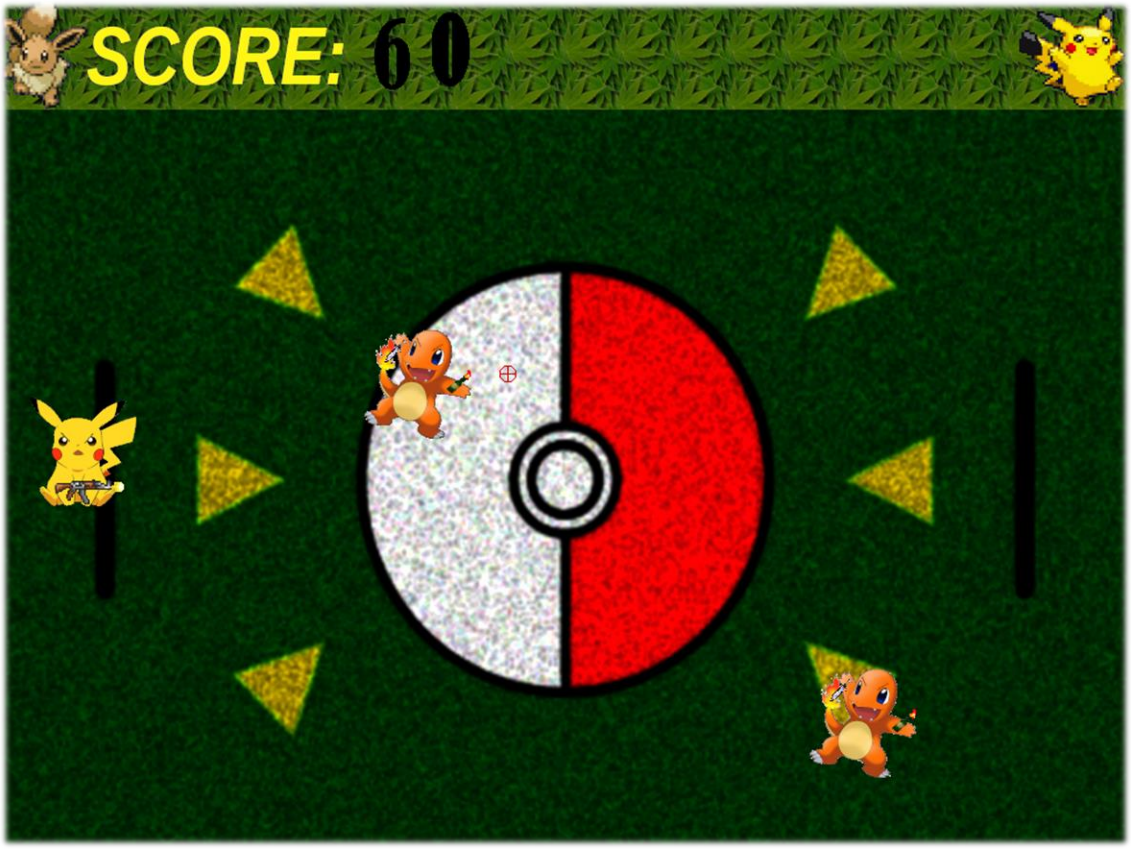
- Play (1P)

Quando a opção de jogar a um jogador é selecionada, o utilizador é posto diretamente em jogo. No início só se encontra em jogo o sprite do jogador, no meio do ecrã, e é também visível o cursor do rato.

A períodos de tempo sucessivamente decrescentes será posto em jogo um novo inimigo, num dos quatro cantos do ecrã, que irá perseguir o jogador. Este deve usar as teclas "WASD" ou as setas direcionais para fugir dos inimigos, ao mesmo tempo que usa o rato para fazer mira nestes e usar o botão esquerdo do rato para disparar sobre eles.

O jogo termina quando o jogador não conseguir evitar mais os inimigos e for tocado por um, sendo o utilizador enviado para um ecrã de Game Over. Quanto mais tempo sobreviver, maior será o score (visível na barra de cima), sendo este também incrementado a cada vez que o jogador mata um inimigo.

A qualquer ponto do jogo, pode também pressionar a tecla "escape" para retornar ao menu.



- Play (2P)

Este modo é bastante semelhante ao modo de um jogador, com uma ligeira diferença: enquanto que no modo de um jogador este controla ambos o rato e o teclado, no modo a dois jogadores um dos jogadores controla apenas o teclado, enquanto que o outro controla apenas o rato.

Quando esta opção é primeiro selecionada, como ainda não existe outro jogador para jogar, este primeiro jogador (P1) é redirecionado para um ecrã de espera pelo jogador 2. Quando o outro jogador seleciona esta mesma opção, são ambos naquele momento postos em jogo.

- Game Over

Quando o jogador colide com um inimigo, é enviado para este menu, onde é visível o seu score. A partir daqui, o utilizador pode escolher começar um novo jogo a um jogador, ou retornar ao menu.

Estado do Projeto

Periférico	Função	Interrupções
Timer	Contagem do tempo e atualização das frames	**Sim**
Teclado	Movimento do jogador durante o jogo e saída de varios menus	**Sim**
Rato	Seleção de opção nos menus e mira durante o jogo	**Sim**
Placa Gráfica	Display do programa	**Não**
Real Time Clock	Seleção de modo diurno/noturno e identificação da data e hora do score em Scores.txt	**Não**
Serial Port	Comunicação entre as duas máquinas no modo de dois jogadores	**Sim**

- Timer

O timer é usado principalmente para 3 fins:

• Fazer atualizações periódicas da frame de jogo;

• Incrementar periodicamente o score;

• Determinar quando se deve criar um novo inimigo.

No entanto, também tem outros usos menores como alterar frames de um sprite animado e limitar o tempo entre disparos do jogador.

- Teclado

O teclado é usado nas seguintes ocasiões:

• Durante o jogo, para permitir o movimento do sprite do jogador;

• Nos menus, para voltar ao menu anterior ou, caso esteja no menu principal, terminar o programa.

- Rato

O teclado é usado nas seguintes ocasiões:

• Na navegação entre ecrãs, com o botão esquerdo;

• Durante o jogo, para fazer mira dos tiros do jogador (guardando a sua posição) e disparar com o botão esquerdo.

- Placa gráfica

A placa gráfica é usada em todo o programa para ser possível a visualização do programa.

Foi usado o modo 0x144, com resolução 1024x768, e modo de cor direto na configuração (8:)8:8:8 ARGB. São portanto possíveis cerca de 16.78 mil cores, com 256 possiveis transparencias.

Também foi implementada a técnica de double buffering, de forma a prevenir frame-stuttering.

- Real Time Clock

O RTC é usado em duas situações:

• Quando se começa um novo jogo, caso a hora do dia esteja entre as 7 da manhã e as 9 da tarde, escolhe um fundo diurno para o jogo; caso contrário, é usado um fundo noturno;

• No final de cada jogo, é guardado no ficheiro Scores.txt a data e hora a que foi obtido o score a guardar juntamente com este.

- Serial Port

O Serial Port é usado no modo a dois jogadores por interrupts (com alta frequência) para enviar a seguinte informação:

• Um ping inicial para determinar o P1 e o P2;

• Evento de criação de inimigo do timer do P1 para o P2;

• Eventos do keyboard do P1 para o P2;

• Eventos do rato do P2 para o P1;

Estrutura/Organização do Código

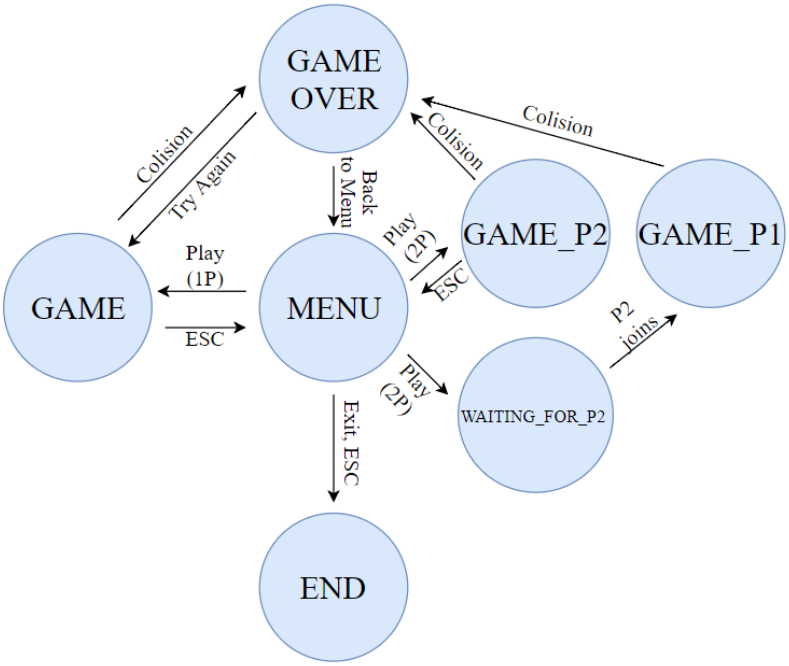
- Bitmap

Neste módulo encontra-se o código responsável por importar ficheiros .bmp para memória e desenhá-los, tendo em conta ou não colisões. As structs e a função de importar os ficheiros são da autoria de Henrique Ferrolho, cuja fonte pode ser encontrada no seu blog. No entanto, as restantes funções de desenho e o sistema de colisões foram adaptados para o nosso projeto.

Contribuição: Henrique Santos/Luís Fernandes (85%/15%)

- Game

Este módulo trata da lógica do jogo em si (orientada a eventos), e é neste módulo que se encontram o interrupt handler (que recebe interrupts dos periféricos e levanta os eventos relevantes), o event handler (que recebe os eventos lançados pelo interrupt handler e os trata de acordo com o estado atual de jogo) e funções de startup/shutdown do projeto (como alocação/libertação de imagens e restauração de variáveis ao seu valor original a cada jogada). Inclui também uma state machine, que permite ao programa saber em que fase do jogo se encontra o programa.



Contribuição: Henrique Santos/Luís Fernandes (80%/20%)

- Keyboard

Este módulo é responsável pela interação entre o teclado e o programa, e foi importado do lab3.

Contribuição: Henrique Santos/Luís Fernandes (50%/50%)

- Mouse

Este módulo é responsável pela interação entre o rato e o programa, e foi importado do lab4.

Contribuição: Henrique Santos/Luís Fernandes (50%/50%)

- Proj

É neste módulo que se encontra a função main e a função proj_main_loop, sendo esta última responsável por fazer a setup inicial (subscribes aos periféricos e iniciar o modo de vídeo), chamar o loop do interrupt handler no módulo Game, e no final é responsável por retornar a máquina ao estado inicial (unsubscribes aos periféricos e retornar ao text mode).

Contribuição: Henrique Santos/Luís Fernandes (50%/50%)

- RTC

Este módulo é responsável pela interação entre o Real Time Clock e o programa, e contém funções subscrever às interrupts do RTC (embora não tenham sido usadas), ler datas, horas, e converter de BCD para decimal.

Contribuição: Henrique Santos/Luís Fernandes (10%/90%)

- Serial Port

Este módulo é responsável pela interação entre o Serial Port e o programa, e contém funções para configurar a serial port no modo usado no nosso jogo, subscrever aos interrupts do Serial Port, um handler para receber mensagens e funções de envio de uma mensagem inteira e de um byte apenas, assim como funções de verificação de estado da serial port.

Mais informação sobre o modo utilizado pode ser encontrada na secção de detalhes de implementação.

Contribuição: Henrique Santos (100%)

- Sprites

Este módulo dedica-se à abstração baseada em objetos de sprites animados e não animados, permitindo manter sempre guardada a posição de certo sprite, a sua imagem/imagens (caso seja animado) e as suas velocidades (horizontais e verticais). Este módulo permite uma maior abstração do módulo Bitmap, lidando com a lógica de movimento de imagens.

Contribuição: Henrique Santos/Luís Fernandes (80%/20%)

- Text

O módulo Text é um módulo menos relevante que contém funções para dar display de números no ecrã e exportar scores para um ficheiro.

Contribuição: Henrique Santos (100%)

- Timer

Este módulo é responsável pela interação entre o timer e o programa, e foi importado do lab2.

Contribuição: Henrique Santos/Luís Fernandes (50%/50%)

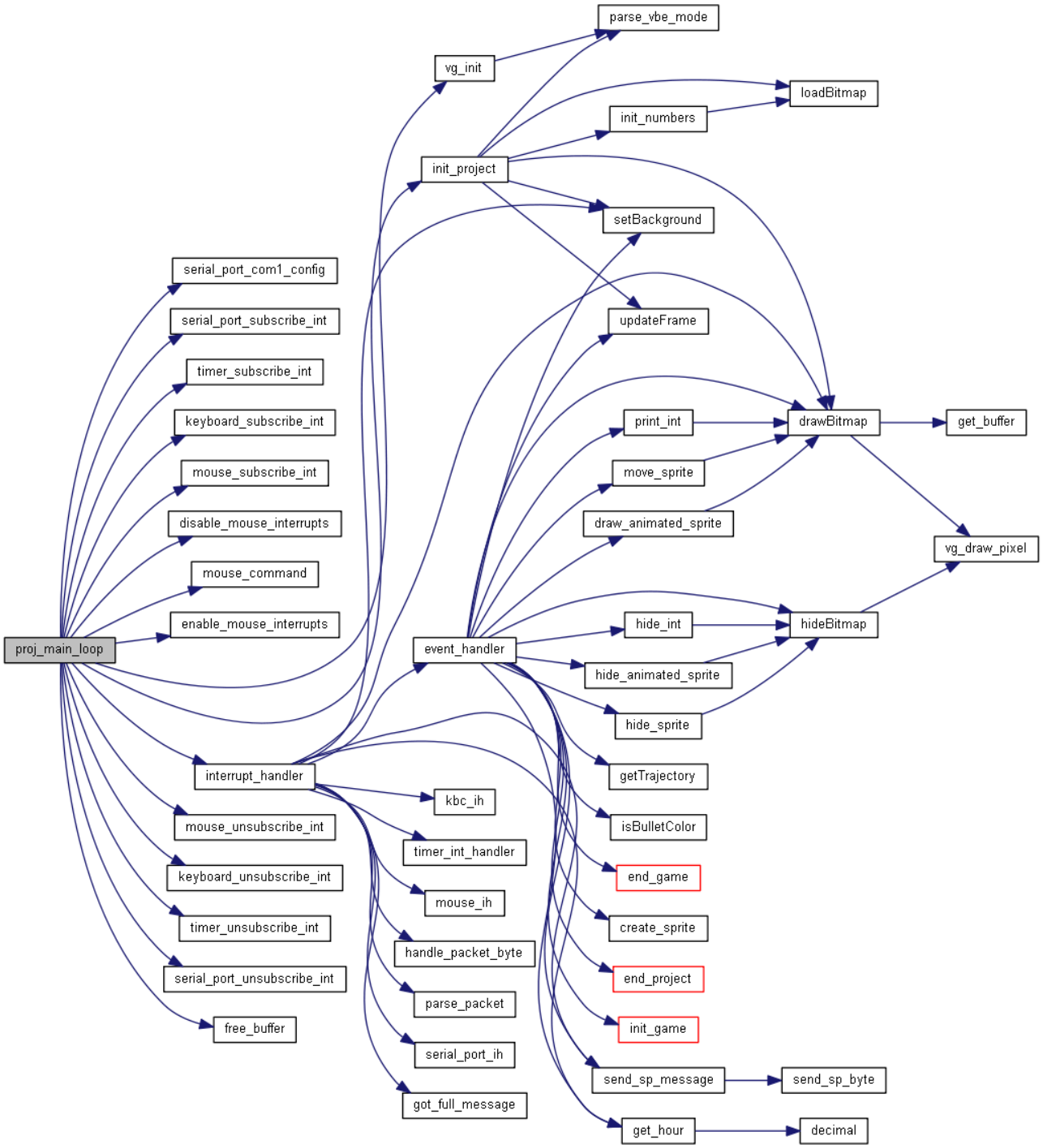
- Video

Este módulo é responsável pela interação entre a placa gráfica e o programa, e foi importado do lab5.

Contribuição: Henrique Santos/Luís Fernandes (50%/50%)

Módulo	Peso relativo
Bitmap	9%
Game	15%
Keyboard	10%
Mouse	11%
Proj	5%
RTC	6%
Serial Port	9%
Sprites	11%
Text	3%
Timer	11%
Video	10%

Call Graph



interrupt_handler() – recebe interrupts, lê dados do periférico, levanta bit relevante no byte "event_byte", e chama event_handler() para o tratar.

event_handler() – dá parse ao event byte e, consoante o enum GAME_STATE, realiza as instruções correspondentes.

Detalhes da Implementação

O código foi todo estruturado à volta de camadas, onde no interrupt handler se trata do baixo nível em termos de interação direta com os periféricos, enquanto que no event handler se abstrai dos periféricos em si utilizando "eventos" em vez de informação direta destes.

A máquina de estados do jogo permitiu um maior controlo sobre os eventos, permitindo ao programa saber que eventos são relevantes em cada fase de jogo, e como tal permitindo um maior controlo em geral.

A utilização do módulo Sprites foi também uma mais valia ao projeto, visto que permitiu orientar todos os componentes do jogo como objetos independentes, permitindo uma maior abstração em mover as várias imagens ao longo do ecrã.

Em termos de geração de frames, foi utilizada a técnica de double buffering, para impedir frame stuttering e propiciar uma maior fluidez de imagem ao utilizador. Todas as funções do módulo Bitmap operam ao nível do buffer, e quando todas as operações necessárias forem feitas, a função updateBuffer() permite copiar os conteúdos do buffer para a memória de vídeo, permitindo um update único a cada frame. Também relativo ao módulo Bitmap, foi implementado um sistema de deteção de colisões baseado no background, que funciona pixel a pixel, de forma a garantir a maior autenticidade possível, que uma hitbox não conseguia garantir.

Na secção de sprites, utilizamos nos sprites animados funções com número de variáveis variável, de forma a termos um número variável de frames de animação.

O RTC foi usado apenas em duas situações (na seleção do fundo diurno/noturno e na gravação da data e hora em que foi alcançado o score), pelo que a interação com este é feita por polling, pois a frequência com que é usada esta informação não justificava o uso de interrupts.

O UART foi configurado com 8 bits por char, 1 bit de paragem e paridade ímpar, e usado por interrupts, visto que o flow de mensagens é constante e bilateral, e seria pouco eficiente uma abordagem por polling.

Conclusão

Achamos que a unidade curricular, embora bastante exigente, é extremamente útil e interessante como uma introdução ao funcionamento de baixo nível dos periféricos, e é algo que consideramos essencial termos pelo menos algum contacto num curso de informática.

No entanto, consideramos que talvez uma "stepping stone" entre o lab0 e o lab2 fosse útil, visto que, como até à altura não tivemos grande experiência com kernel calls, foi uma entrada bastante abrupta no tema. Seria também útil incluir nos guias algumas dúvidas recorrentes, visto que reparamos que muitas dúvidas eram comuns a vários grupos, e como tal tinham de ser esclarecidas várias vezes pelos professores, o que poderia ser evitado com uma pequena nota no guião.

Achamos também que a disponibilidade dos professores nos fóruns da unidade currical foi extremamente útil, visto que maioria das vezes os alunos obtiam respostas no espaço de poucas horas e geralmente com uma solução para os problemas enfrentados.