Battalion Tanks

Laboratório de Computadores

2015-2016 – 1º Semestre

Turma 4, grupo 15

Trabalho realizado por:

Luís Miguel da Rocha Alves up201405308@fe.up.pt

Renato Sampaio de Abreu up201403377@fe.up.pt

Índice

Introdução

O nosso projeto, realizado no âmbito da unidade curricular Laboratório de computadores do curso Mestrado Integrado em Engenharia Informática e Computação, baseia-se no jogo Tank Battalion, de 1980, e também no jogo Battle City, de 1985.

Dado os objetivos desta unidade curricular, propusemo-nos a realizar uma versão dos jogos anteriormente referidos, adaptando o jogo de modo a utilizar os diferentes periféricos mencionados ao longo do semestre e o software de desenvolvimento da unidade curricular (como sistema operativo, Minix, e como linguagem de programação, c), e claro, adicionando novas funcionalidades.

O jogo em si, permite o controlo de um tanque: com as teclas wasd, é possível movimentar o tanque pelo terreno, não podendo atravessar as bordas ou as paredes do terreno. O rato permite controlar a mira e dependendo da posição do rato, o canhão pode rodar em ângulos de 45º; O disparo do canhão é também controlado pelo rato mas neste caso depende do clique no botão esquerdo e claro a direção da bala será coincidente com a direção atual do canhão. É importante referir que as balas dos tanques permitem destruir o terreno.

O objetivo do jogo será então destruir os 21 tanques inimigos que vão aparecendo no ecrã. O score final do jogador dependerá do tempo que demorou a destruir os tanques inimigos.

Instruções de Utilização

Ao iniciar o jogo, é apresentado um menu inicial com o nome do projeto, Battalion Tanks, a data atual no canto superior esquerdo, e três opções que ficam realçadas caso o utilizador coloque o rato sobre uma das opções. Para selecionar uma opção basta que esteja realçada e o utilizador clique no botão esquerdo do rato.

Main menu

A opção "Start" permite iniciar o jogo.

"Highscores" remete para o menu das melhores pontuações.

"Exit" termina o programa.

Highscores menu

Neste menu são apresentados os 9 melhores resultados, bem como o nome do jogador e a respetiva pontuação.

A opção "Exit" permite voltar ao menu principal.

Modo de jogo

Na barra lateral temos o número de tanques inimigos por destruir, a pontuação e por fim o número de vidas do jogador.

Os tanques inimigos aparecem no topo, sendo que o máximo de tanques inimigos no terreno são dois. O tempo de entre destruição e reaparecimento é 5 segundos.

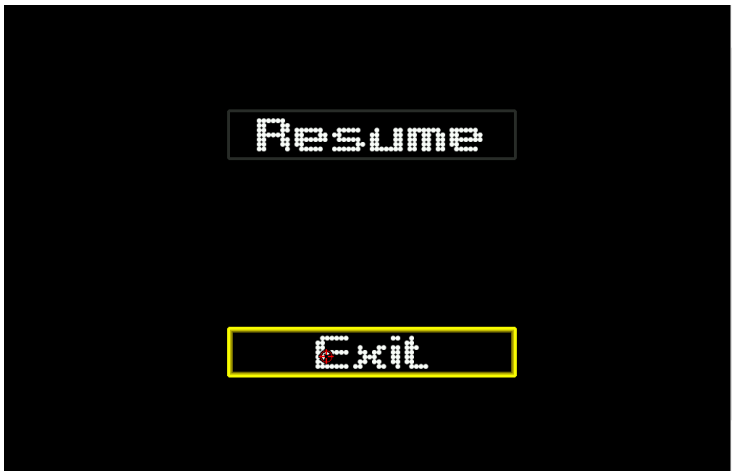
Cada tanque inimigo destruído aumenta o score do jogador em 25 pontos. O número de vidas restantes no final do jogo aumenta o score em 100 pontos e o resto dos pontos será dependente do tempo de jogo.

Save score menu

Neste menu o jogador que terminou o jogo pode registar o seu nome e score.

Para registar o nome, o jogador apenas necessita de colocar o rato sobre uma das três letras e digitar uma letra, caso essa letra seja elegível, é atualizado o nome.

Pause menu



A partir deste menu, o jogador pode voltar ao jogo clicando em "Resume" ou terminar o jogo atual e voltar ao menu principal clicando em "Exit".

Estado do projeto

Na especificação do projeto, pretendíamos implementar a porta uart, mas infelizmente, por sabermos que íamos abdicar de muito tempo, não pelo facto de alterar muito código no projeto atual mas sim pelas dificuldades que iríamos ter em implementar a porta em série sem a ajuda dos professores, como tivemos nos restantes periféricos durante as aulas laboratoriais (exceto RTC).

Por outro lado, os restantes periféricos foram implementados com sucesso. Contudo, se tivessemos a oportunidade de melhorar algo seria a inteligência artificial dos tanques inimigos, visto que em certas situações ficam na dúvida sobre que caminho seguir.

Periféricos Implementados

Periférico	Utilização	Interrupções
Timer	Utilizado para controlar os frames, o aparecimento de tanques e as animações das explosões	Sim.
Teclado	Utilizado para movimentar o tanque e inserir o nome do jogador	Sim.
RTC	Atualizar a data apresentada no menu inicial	Não. (Polling)
Placa gráfica	Visualizar o jogo e menus	Não.
Rato	Selecionar opções nos menus e controlar a rotação e disparo do tanque	Sim.

Timer – O timer é utilizado maioritariamente para controlar a taxa de refrescamento do ecrã, contudo, também é utilizado para controlar o aparecimento de um tanque depois de este ser destruído e a animação de uma explosão (utilizada quando um tanque é destruído), sendo que para isso recorre ao ciclo de interrupções presente no ficheiro BattalionTanks.c.

Teclado – O teclado é usado em duas situações. A primeira diz respeito ao jogo em si, em que o utilizador controla o movimento do tanque utilizando as teclas wasd. A segunda é o input de texto utilizado para registar o nome do jogador. Em ambas as situações a receção do input é feita em BattalionTanks.c e o processamento dessa informação em kbd.h e kbd.c e por fim a sua utilizacao ou é, no primeiro caso em tank.c e no segundo em menu.c.

RTC – O rtc é utilizado para ler a data atual do computador. Este periférico tem as suas funções no ficheiro rtc.h e rtc.c

Placa gráfica – A placa gráfica utilizada em modo 0x117 permite fazer o display de todos os menus, da data, de texto, e de todos os elementos do jogo em si. Os ficheiros deste periférico são video_gr.h, video_gr.c, vbe.c e vbe.h. É importante referir que toda a informação é primeiro passada para um double buffer e apenas depois é que é feita a transição para a memória principal.

Rato – O rato é utilizado para movimentar o cursor tanto nos menus como no jogo, dependendo assim da sua posição; Para selecionar opções nos menus e disparar o canhão do tanque recorre-se ao clique no botão esquerdo. A receção do input é feita em BattalionTanks.c e o processamento dessa informação em mouse.h e mouse.c e por fim a sua utilizacao é, caso esteja-se em modo jogo, em tank.c ou em menu.c.

Com a utilização da máquina de estados na estrutura principal do jogo, apenas necessitamos de uma função que utilizasse a chamada de driver_receive().

A função Loop em BattalionTanks.c é responsável pela chamada desssa função, sendo que conforme a interrupção recebida é ativada uma flag a identificar o periférico de onde a interrupção foi originada.

Assim, conforme a flag e o estado atual da estrutura principal, são feitos os updates necessários no jogo.

Estrutura

Módulos

- BattalionTanks.c – Este módulo permite, através da máquina de estados e do ciclo de interrupções, gerir a informação recebida e utilizá-la conforme o estado atual do jogo.
  Achamos que no geral o Renato fez 70% deste módulo e o Miguel 30%.

- Bullet.c – Este módulo contém a abstração de uma bala e de uma "bulletpool", isto é uma estrutura com vários objetos do tipo "bullet".
  O Renato fez 80% deste módulo e o Miguel 20%.

- Font.c – Módulo que permite escrever no buffer um determinado texto ou apenas um char.
  O Renato fez 30% deste módulo e o Miguel 70%.

- Game.c – Este módulo trata de toda a informação referente ao jogo em si, isto é, update dos tanques, tratamento de todo tipo de colisões, update do terreno e draw de todos os elementos para o buffer.
  O Renato fez 80% deste módulo e o Miguel 20%.

- Main.c – Inicialização da estrutura principal do projeto e início do modo gráfico.
  O Renato fez 100% deste módulo.

- Menu.c – Este módulo contém todos as estruturas dos diferentes menus utilizados no projeto, bem como todo o código necessário ao seu update e draw.
  O Renato fez 80% deste módulo e o Miguel 20%.

- Mouse.c – Módulo parcialmente definido nas labs que permite atualizar os packets do rato e por conseguinte fazer o seu update. Neste módulo adicionamos a abstração do mouse de forma a simplificar o update da informação do rato.
  O Renato fez 50% deste módulo e o Miguel 50%.

- Rtc.c – Este módulo permite a ler os registos do minix em relação à data, atualizando assim a abstração date.
  O Renato fez 20% deste módulo e o Miguel 80%.

- Tank.c – Contém as abstrações Tank e EnemyTank, bem como todas as funções que verificam colisões, disparos, movimento, rotações do tanque ou canhão. No caso da abstração Tank o update do tanque depende da máquina de estados utilizada que por sua vez depende maioritariamente do input recebido.
  O Renato fez 80% deste módulo e o Miguel 20%.

- Terrain.c – Módulo que define o aspeto do terreno (“wall” ou “grass”) através de um array de ints e o aspeto da barra lateral direita que contém informação relevante ao jogo.
  O Renato fez 80% deste módulo e o Miguel 20%.

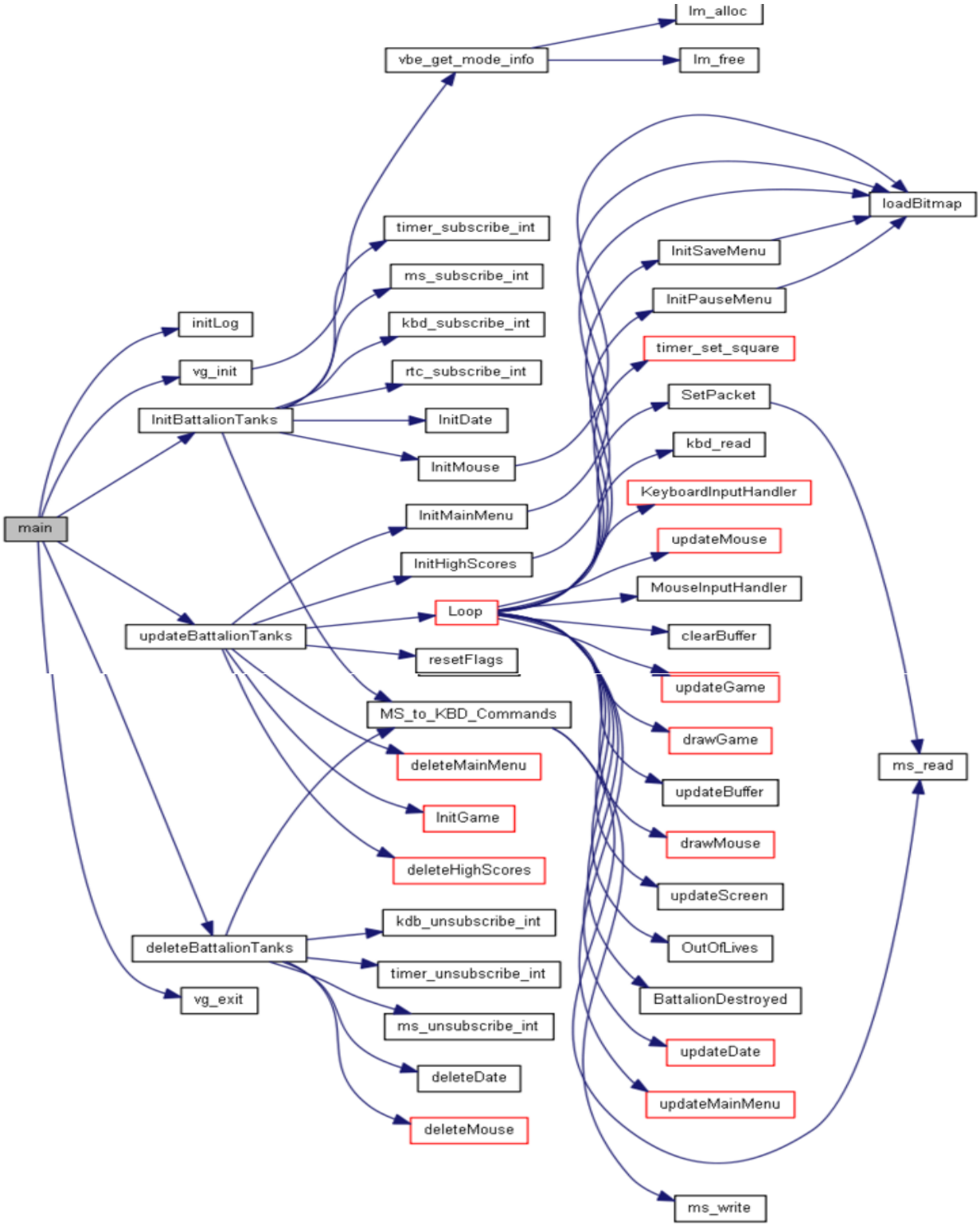
- Video_gr.c – Módulo definido nas aulas laboratoriais. Foi adicionado um buffer e um mouse buffer de modo a facilitar o draw dos vários elementos do jogo.
  O Renato fez 60% deste módulo e o Miguel 40%.

- Bitmap.c – Código desenvolvido por Henrique Ferrolho, estudante do MIEIC, com base num post do fórum fedoraforum.org. Este módulo é utilizado para fazer o load dos bitmaps e para fazer o seu draw no buffer.

  O Renato fez 50% deste módulo e o Miguel 50%.

- Timer.c – módulo que permite subscrever as interrupções do timer, desenvolvido nas labs.
  O Renato fez 50% deste módulo e o Miguel 50%.

- Kbd.c – módulo que permite subscrever as interrupções do teclado bem como ler os bytes recebidos, desenvolvido nas labs.
  O Renato fez 50% deste módulo e o Miguel 50%.


lm_alloc
vbe_get_mode_info
lm_free
loadBitmap
timer_subscribe_int
ms_subscribe_int
kbd_subscribe_int
rtc_subscribe_int
InitDate
InitMouse
InitSaveMenu
InitPauseMenu
timer_set_square
SetPacket
kbd_read
KeyboardInputHandler
updateMouse
MouseInputHandler
clearBuffer
updateGame
drawGame
updateBuffer
drawMouse
updateScreen
OutOfLives
BattalionDestroyed
updateDate
updateMainMenu
ms_write
ms_read
initLog
vg_init
InitBattalionTanks
main
InitMainMenu
InitHighScores
updateBattalionTanks
Loop
resetFlags
MS_to_KBD_Commands
deleteMainMenu
InitGame
deleteHighScores
deleteBattalionTanks
kdb_unsubscribe_int
timer_unsubscribe_int
ms_unsubscribe_int
deleteDate
deleteMouse
vg_exit

Implementação

Máquinas de estado

Ao longo do projeto, tentamos implementar máquinas de estado, visto que é uma forma de tornar o código mais legível e estruturado.

Então, a primeira máquina de estados faz parte da estrutura BattalionTanks. A partir desta conseguimos controlar as várias transições entre menus e o jogo. Além disso facilitou o processamento das interrupções, pois sabendo o estado atual, tornou-se mais fácil aplicar a resposta do programa face a essas interrupções.

A segunda máquina de estados foi implementada da estrutura Tank, sendo que o estado atual é definido pelo input recebido nas interrupções. Desta forma, o update do objeto Tank é sempre feito consoante o estado atual.

Desenvolvimento por "layers"

De forma a tornar o projeto mais estruturado, tentamos criar vários ficheiros que se relacionassem entre si, mantendo ao mesmo tempo uma estrutura hierarquizada. Como exemplo disto temos, o ficheiro BattalionTanks sendo o principal deste projeto, que por sua vez, conforme o estado, interage com menu.c ou game.c. Posto isto, no estado de jogo, game.c interage com o tanks.c e terrain.c e assim em diante. Com isto, as estruturas principais vão se relacionando com as mais secundárias criando uma estrutura mais simples e hierarquizada.

Programação orientada a objetos

Apesar de C não ser uma linguagem orientada a objetos, tentamos replicar essa orientação no nosso projeto, criando para isso estruturas como Tank, EnemyTank, Terrain, etc abstraindo assim estes "objetos" e atribuindo-lhes membros função como Init, update, draw e delete (além de outras funções necessárias, claro). Este método de desenvolvimento facilitou bastante tanto o acesso como a modificação desses objetos.

Deteção de colisões

Na deteção de colisões, principalmente entre objetos "Tank" e "Bullet", recorremos ao tipo de colisão Axis-Aligned Bounding Box, já que todas as nossas estruturas podiam ser representadas por um quadrado.

Por outro lado, para detetar as colisões entre tanques e as paredes do jogo verificamos as atuais posições do objeto em questão e verificamos se as células ocupadas por esse objeto estão prenchidas com algo diferente de "grass". Aqui podemos referir que a implementação do objeto "Terrain" com membro dado cells foi de extrema importância, pois permitiu dividir o terreno em várias células de 36x36 pixeis, facilitando assim a verificação das células utilizadas pelo tanque.

Movimento dos tanques inimigos

Em relação ao movimento dos tanques inimigos, decidimos implementar um algoritmo em que estes podem se comportar de diferentes maneiras conforme a posição onde se encontram em relação ao tanque do jogador.

Caso o tanque inimigo e o tanque do jogador estejam numa célula horizontalmente igual ou verticalmente igual no array do terreno, o tanque pode agir de duas formas: Se o número de células entre eles que estiverem bloqueadas for menor que 5 (células definidas com "wall"), o tanque dispara e move-se até chegar à posição do tanque do jogador, destruindo-o caso continue lá. Se o número de células bloqueadas for maior que 5 então o tanque faz um movimento random.

Se nenhuma das condições em cima se verificar, então o tanque poderá seguir o caminho mais curto, verticalmente ou horizontalmente em relação ao tanque do jogador. Por exemplo, imaginemos que o número de células na vertical para o tanque inimigo ficar numa posição vertical ou horizontal(em relação ao tanque do jogador) é 4, e o número de células na horizontal para o tanque inimigo ficar numa posição vertical ou horizontal é 5, então o tanque inimigo vai escolher o movimento vertical na direção que o aproxima do tanque. Contudo, se o número de casas bloqueadas nesse sentido for maior que 5, ele verificará a alternativa, isto é se é mais vantajoso seguir o caminho oposto ao referido anteriormente. Para isto acontecer o número de casas

bloqueadas também tem de ser menor que 5. Caso nenhuma destas condições se verifique, o tanque segue um caminho random.

Ficheiros Bitmap

Todas as imagens e sprites incluídas estão em formato bitmap, bem como os carateres que formam o texto. Usando como base o já referido código desenvolvido por Henrique Ferrolho, as principais alterações por nós implementadas foram a rotação de bitmaps e a capacidade de efetuar o seu draw com transparência.

Quanto à rotação de bitmaps, apenas é possível rodar em ângulos múltiplos de 90 graus, ou seja, 90, 180, e 270 graus. A necessidade de utilizar um tanque com um canhão minimamente realista levou portanto à inclusão de dois ficheiros bitmap, um ortogonal e um diagonal, elevando o número máximo de direções deste caso particular de 4 para 8.

De modo a sobrepor bitmaps, como por exemplo a base do tanque e o seu canhão, decidimos usar como cor transparente o tom de verde que equivale ao código hexadecimal RGB de #00FF00. Assim, quando o bitmap está a ser desenhado, se a cor do atual pixel a ser lido for igual à cor referida, esse pixel é ignorado, o conteúdo do buffer não é alterado, e o iterador avança para a seguinte posição.

"Object pool pattern"

Visto que o objeto "Bullet" iria ser requisitado diversas vezes ao longo de um jogo, pelos diversos tanques, decidimos implementar uma "BulletPool", sendo um membro dado de cada tanque. Esta implementação inicializa os objetos da pool no ínicio do jogo, evitando assim a necessidade de alocação e destruição de objetos "bullet" ao longo do jogo, operações que poderiam afetar a perfomance do jogo. Em vez disso, são feitas requisições à bulletpool e o primeiro objeto que não esteja a ser usado, é reutilizado pelo tanque e uma vez que este seja destruído passa a ficar "desativado".

Além disso, fazendo uma bulletpool com 5 objetos inicializados limitamos à partida o número de balas possíveis de utilizar num determinado espaço de tempo.

Conclusão

Em relação ao que se poderia melhorar nesta unidade curricular, achamos que quanto à porta uart deveria realizar-se pelo menos uma aula prática centrada nesse periférico de forma a ajudar a ultrapassar as dificuldades existentes em relação a este. Contudo, visto que nem todos os alunos pretendem desenvolver este periférico no projeto, esta lab não seria avaliada e quem quisesse poderia trabalhar no seu projeto em vez de realizar a lab.

Quanto aos aspetos positivos, acho que foi uma ótima iniciativa a aula teórica dada pelo monitor Henrique Ferrlho. Na nossa opinião é uma excelente forma de nos ajudar em certos aspetos que não foram referidos nas teóricas e que foram bastante importantes para começarmos a desenvolver o projeto.

Por último, apesar de esta ser uma unidade curricular bastante exigente devido às labs, ao projeto e à introdução ao funcionamento dos periféricos, ambos concordamos que, sem dúvida, a forma como unidade curricular está estruturada permite aprofundar bastante os conhecimentos nos diversos periféricos leccionados ao longo do semestre bem como melhorar o conhecimento da linguagem C, sendo o projeto final o desafio ideal para relacionar esses periféricos e colocar à prova os conhecimentos adquiridos.

Como auto-avaliação concordamos com o seguinte:

Participação:	Renato – 60%	Miguel – 40%
Contribuição relatório:	Renato – 80%	Miguel – 20%
Contribuição total:	Renato – 70%	Miguel – 30%

Execução: Primeiro é necessário copiar, como root, o ficheiro Tanks para /etc/system.conf. Depois disto, no terminal, digitar su seguidamente de cd /home/lcom/Tanks/src, depois make e por fim service run `pwd`/Tanks.