RELATÓRIO LCOM 2017

Programa/Jogo: Tower Defense

Turma 3, Grupo 02;

Alunos:

- Carlos Miguel Sousa Vieira; up201606868;
- João Augusto dos Santos Lima; up201605314;

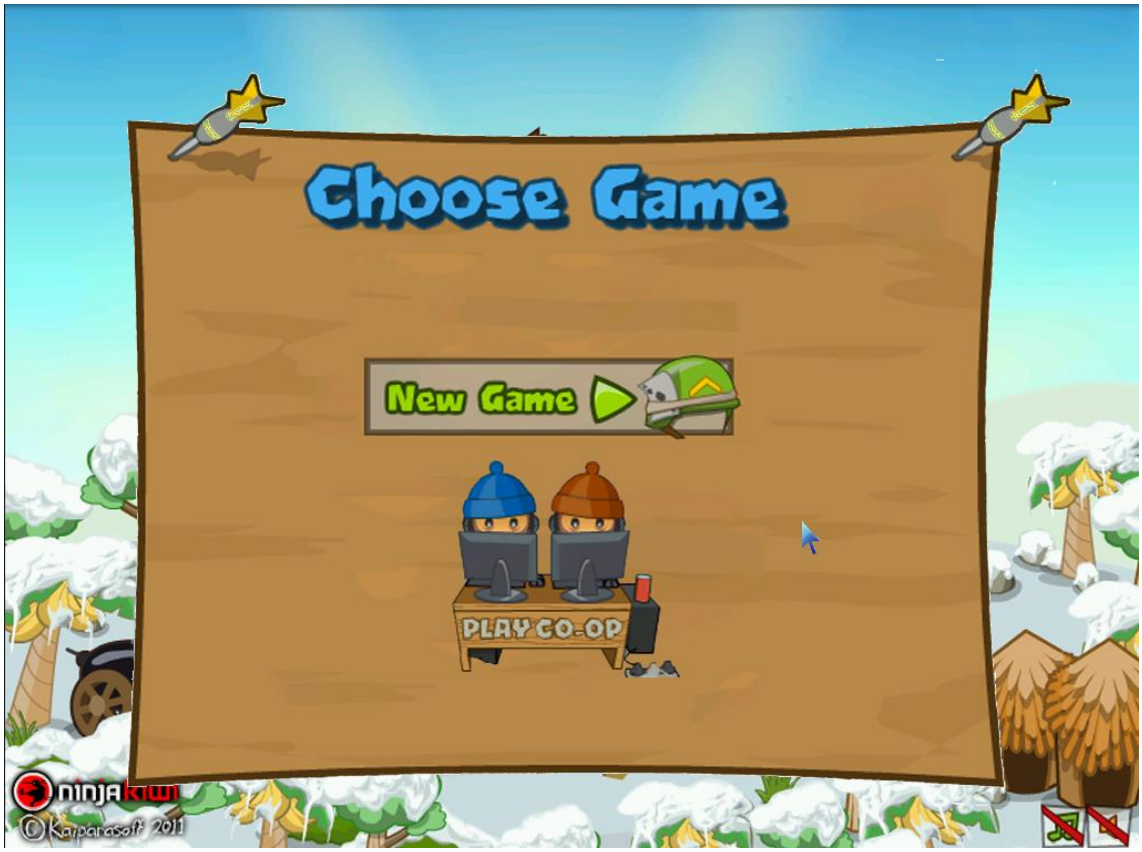
ÍNDICE

1.Instruções do utilizador

1.1. Menú Principal

No momento no qual o programa é iniciado, este é o ecrã que nos é apresentado. Aqui temos uma imagem retirada do jogo original. Para avançar para o próximo menú basta clicar no botão "*play*". Em qualquer momento do programa a tecla "*Esc"* serve para sair do mesmo.

1.1 Menu secundário:



Após clicar no botão de "*play*", é nos apresentado um menú secundário com duas novas opções. A primeira opção, "*New Game*", inicia um jogo novo em *singleplayer* e a segunda opção, *"Play co-op",* inicia a versão *multiplayer* do jogo quando os dois computadores estiverem ligados.

1.2 Jogo:

1.3.1. *Singleplayer*:

Aqui podemos ver a *interface* do jogo, bem como o mapa onde o jogo vai decorrer. Do lado direito do ecrã temos a quantidade de dinheiro que o jogador tem, bem como a quantidade de vidas. De cada vez que se rebenta um balão vermelho ganha-se $1, e sempre que um balão vermelho escapa perde-se uma vida. Mais abaixo podemos ver 2 macacos diferentes que podem ser escolhidos, o macaco normal e o super macaco. Cada um destes custa $250 e $2500, respetivamente.

Para colocar um macaco no mapa basta selecionar um da parte direita do ecrã ou usar uma das seguintes teclas. A tecla “Q” para o macaco normal e a tecla “E” para o super macaco. Após seleccionar, é suficiente arrastar o objeto escolhido para onde queremos no mapa e clicar com o botão esquerdo do rato.

Na parte inferior do ecrã, podemos ver em que ronda do jogo estamos e também podemos ver o *RBE-Red Bloon Equivalent*, que corresponde à quantidade de dinheiro que se vai ganhar nesta ronda, caso se rebentem todos os balões presentes no ecrã.

1.3.2. *Multiplayer*:

Na versão *multiplayer*, o jogador 1 tem as mesmas funcionalidades que tinha no modo *singleplayer*. No entanto, o jogador 2 tem o seu rato desativado e pode clicar nas teclas para mandar balões para o adversário. O objetivo do jogador 2 é fazer com que o seu adversário perca as suas vidas todas.

2.Estado do projeto

Dispositivos usados	Utilidade	Modo usado
Timer	Usado para controlar a quantidade de *frames* e para calcular intervalos de tempo usados pelas torres e pelas rondas.	*Interrupts*
Teclado	Usado no jogo para ser mais rápido selecionar torres ou para o jogador 2 no modo *multiplayer* poder mandar balões.	*Interrupts*
Rato	Usado para selecionar opções e torres, bem como coloca-las no mapa	*Interrupts*
Placa de vídeo	Usado para mostrar o ecrã do jogo e os menus do programa	*Interrupts do timer*
Serial Port	Usado para a transmissão de dados no modo *multiplayer* entre os 2 jogadores	*Interrupts e polling*

2.1 Dispositivos usados:

Timer:

- Este dispositivo controla a quantidade de frames do programa, e no caso do mesmo, por predefinição, são 30 *frames* por segundo. A função *update()*, do ficheiro *program.c*, é chamada a cada *interrupt* do *timer*;

Placa de vídeo:

- É usado o modo gráfico 117, com as resoluções de 1024x768, com 65535 cores no modo direto;
- Para este programa foi usado *double buffering*;
- São usados objetos que se mexem (ex. balões), bem como os projeteis que as torres disparam. Entre os balões e os projeteis existe um sistema de colisões. Nas torres existem animações no qual as torres mudam de direção para onde estão a olhar, o mesmo acontecendo para o projeteis;
- São usadas imagens com números para poder indicar o dinheiro, vida, ronda e *RBE* no ecrã;
- Existem várias funções relativas à placa de vídeo. Entre estas estão as funções de desenho e *load_imagem* do ficheiro *lowProgram* e as funções *orienta_macaco()* e *orienta_pos()* do ficherio *program.c*.

Teclado:

- No jogo o teclado é usado para selecionar mais rapidamente as torres que eu pretendo usar, em vez de as selecionar da tabela do lado direito. No modo *multiplayer* o teclado é essencial para o jogador 2 poder jogar, porque a única maneira de ele mandar balões para o adversário é através do uso do teclado. Também é usada a tecla *Esc* para sair do programa a qualquer momento;
- Funções chamadas pelos eventos do teclado são: as funções *keypressed* no ficheiro *program.c*, *program_inicio.c*;

Rato:

- O rato é usado para selecionar as opções no menu atrás da sua posição e do clique no rato. No jogo o rato é usado para selecionar as torres que já estão presentes no mapa ou para poder comprar novas e as arrastar para o mapa;
- Funções chamadas pelos eventos do rato são: *mouse_controlador* dos ficheiros *program.c* e *program_inicio.c*, tal como as posições do rato, para que este não saia do ecrã, são tratados pela função *mouse_handler* do ficheiro *mouse.c*;

UART:

- Para o *serial port* foram usadas interrupções para a receção da informação, e foi usado *polling* para a transmissão de informação para o outro jogador;
- No início do programa, o primeiro jogador está a enviar uma mensagem periódica para o outro jogador à espera que este se ligue. O outro jogador após estar ligado recebe uma mensagem e manda uma outra de confirmação e assim é estabelecido um contacto entre os dois jogadores. Desta forma, é possível começar o jogo ao mesmo tempo, para que possa correr separadamente em dois computadores. No entanto não é feita uma sincronização entre os dois computadores visto que pode existir uma grande quantidade de objetos no jogo para poder confirmar entre os dois jogadores.
- Sempre que o primeiro jogador coloca uma torre no mapa é mandada uma mensagem para o outro jogador com o primeiro *char* contendo o tipo de torre, os próximos 4 *char* contêm a informação sobre a posição deste, 2 para a *posição X* e 2 para a *posição Y*, e após receber esta informação coloca a torre no seu jogo. Quando o jogador 2 envia um balão é enviada uma mensagem com um único carácter contendo o tipo de balão que ele está a enviar;

Uma das funcionalidades não implementadas que aparece na *interface* gráfica do jogo é o botão "*GO*" e o botão com uma casa, que aparece na parte inferior direita do ecrã, visto não ter havido tempo o suficiente para ser implementado.

3. Organização e estrutura do código

3.1 Módulos:

Bitmap:

- Este módulo já vinha feito por um ex-aluno. (http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html)
- Usei a função de leitura e de apagar *bitmap*. Adicionei também, as *struct* que já vinham feitas, dados para me ajudar a retirar os fundos às imagens.
- Este módulo equivale a 1% do trabalho;

Caminho:

- Este módulo é um ficheiro que contém a criação do caminho que o balão tem de percorrer. É alocada memória e é guardada cada posição que o balão tem de percorrer antes de chegar ao final;
- Este módulo equivale a 2% do trabalho;

Data_storage:

- Este módulo contém todas as *structs* e outras estruturas de dados essenciais ao funcionamento do programa;

i8042/i8254/macros/mouse_macros/ser_macros/vbe_macros:

- Contém todas as macros necessárias para as várias partes do programa;

Keyboard:

- Este módulo contém as subscrições às interrupções do teclado e o tratamento dos dados lidos;
- Este módulo equivale a 2% do trabalho;

LowProgram:

- Este módulo contém todas as funções de baixo nível necessárias para o programa funcionar;
- A função *drawcircle* (https://en.wikipedia.org/wiki/Midpoint_circle_algorithm) e *line*(https://rosettacode.org/wiki/Bitmap/Bresenham%27s_line_algorithm#C) foram retiradas da *internet*;
- Este módulo contém funções de desenho para os vários tipos de objetos do programa e do tratamento de retirar o fundo às imagens, funções de atualização de ecrã, função de tratamento de disparo e o tempo entre disparos (cálculo das velocidades horizontais e verticais de disparo) e funções que tratam de avançar os balões ao longo do percurso e os projeteis e ao mesmo tempo verifica as colisões entre estes dois tipos de objetos.
- Este módulo equivale a 50% do trabalho;

Mouse:

- Este módulo contém as subscrições às interrupções do rato bem como o tratamento de dados relativos ao rato;
- Relativamente ao tratamento de dados, é feita uma verificação à posição do rato no ecrã para este não sair fora do mesmo;
- Este módulo equivale a 5%do trabalho;

Program:

- Este módulo contém funções de mais alto nível, relativas ao funcionamento do programa

- Estas funções incluem: o tratamento das colisões; carregamento de todas as imagens para a memória do jogo; funções de orientação dos macaco dependendo para onde querem disparar; funções que desenham no ecrã os números relativos à vida, dinheiro, rondas e *RBE* do jogador; funções que tratam dos eventos do rato e do teclado;
- Este módulo equivale a 30 % do trabalho;

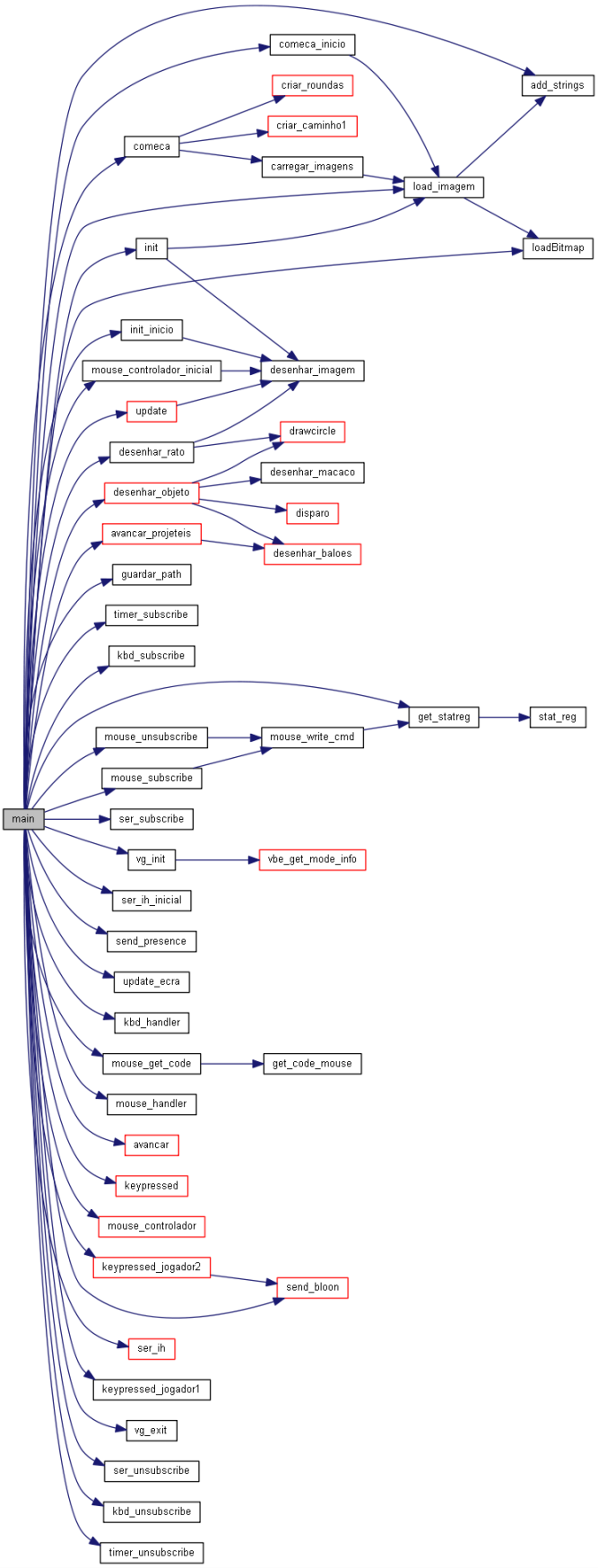
Round:
- Este módulo trata de criar as rondas do jogo e de todo o tipo de balões que vão aparecer nas mesmas;
- Este módulo equivale 2% do trabalho;

Ser:
- Este módulo trata da comunicação entre os dois jogadores no modo *multiplayer*. Trata de mandar e receber informação, bem como a subscrição às interrupções do *serial port*;
- Este módulo equivale a 4% do trabalho;

Towerdefense:
- Este módulo simplesmente contem o *main* que distingue entre os *interrupts* que o programa recebe e chama as funções necessárias para o seu tratamento. Também trata da distinção entre os menus e o jogo;
- Este módulo equivale a 2% do trabalho;

Vbe:
- Este módulo trata de tudo o que é necessário para a utilização gráfica do programa;
- Este módulo equivale a 2% do trabalho;

Gráfico de chamada:



4.Detalhes de implementação

Todos os métodos implementados neste programa foram feitos com base nos ensinamentos dados nas aulas, no entanto, muitos dos algoritmos/métodos foram feitos por nós próprios para o funcionamento correto deste programa.

Alguns dos métodos implementados neste programa:

- O método de conexão entre os dois programas:
    - Um deles quando é ligado manda uma mensagem através do *serial port* para o outro. Esta mensagem é enviada periodicamente porque, segundo observei, na execução do programa, sempre que este está ligado e envia uma mensagem, o próprio recebe a mensagem. Então para esta última estar sempre disponível para o outro programa quando o mesmo se ligar, a mensagem é mandada periodicamente e é recebida a mesma mensagem que mandou, repetindo o processo até a informação recebida ser diferente.
    - Após o segundo programa ligar este recebe pela primeira vez a mensagem e envia uma outra diferente. Após este processo os dois programas sabem que estão ligados e já é possível iniciar o jogo no modo *multiplayer*.
- O método de apresentação de imagens por *bmp*:
    - Para a apresentação de imagens *bmp* sem fundo e para que o programa funcionasse de forma eficiente foi necessário criar um algoritmo para melhorar a performance do programa. Para tal, foi implementado um algoritmo que guarda na memória todos os *pixéis* do *bmp* que não de cor branca. Mas ao desenhar a imagem não seria eficiente percorrer todos os *pixéis* e ignorar todos os brancos porque o programa pode usar uma grande quantidade de objetos. Então para tal foi feita uma primeira leitura de todos os *pixéis* e foi guardada a posição do pixel com uma cor diferente de branca e a quantidade de pixéis diferente de branco. Esta implementação faz com que o processo de desenhar as imagens no *buffer* seja feito muito mais rápido e eficientemente. Possivelmente existem métodos melhores, mas este foi o melhor que foi possível fazer.
- Sistema de colisões do programa:
    - Um método de implementação que foi testado na primeira tentativa foi a divisão do ecrã em várias secções e depois sempre que havia um projétil e um balão na mesma secção ele fazia a verificação para confirmar se havia uma colisão ou não. No entanto, foi verificado que podem existir muitos objetos numa só secção e então tornava o programa pouco eficiente. O sistema de colisões atual foi feito com um "ecrã" do mapa com endereços dos balões, sendo que no estado inicial todas as posições estão a nulo. À medida que os balões avançam eles vão atualizando o ecrã. Quando é a vez de avançar os projeteis é feito uma verificação das posições que este vai tendo no ecrã. Para isso é feita uma linha imaginária que tem início na posição original e vai até a posição final e verifica-se se qualquer dessas posições tem um endereço diferente de nulo. Se tiver, quer dizer que ocorreu uma colisão.

- Algoritmo de auto atacar das torres:
    - Este algoritmo foi feito com a ideia de melhoria do programa, e por tal razão é atualmente pouco eficiente. Várias ideias de implementação foram feitas, como por exemplo: caso um balão entre na área de visão da torre. No entanto foi implementado um método básico de verificar todos os balões e ver se estão no campo de visão da torre. Foi implementado este método porque podia ser necessário a torre distinguir entre o balão mais forte e o que se encontra mais atrás ou mais à frente. Manter ao mesmo tempo uma lista ordenada seria difícil visto que os balões podem ter velocidades diferentes.

5.Conclusões

Existem atualmente vários problemas relativos ao funcionamento do programa:

- Comunicação entre dois programas:
    - Quando acontece um erro de envio, ou porque não foi lido a tempo um dado, ou porque algum motivo de erro não é feito um controlo nem um reenvio, o programa pode parar de enviar dados permanentemente porque não trata deste erro.
- Falta de existência de um menú de vitória ou de derrota:
    - Não houve tempo de ser implementar este menús, o que faz com que o programa simplesmente acabe sem aviso prévio;
- Verificação do suporte do modo gráfico:
    - A última função do *lab5*, o *controller*, não foi feita neste programa, logo o este só consegue funcionar no modo 117 e não verifica se é suportado no PC onde vai correr. Este problema só foi discutido muito tarde e não foi possível resolver este assunto.

Como é possível concluir, o programa tem alguns problemas de funcionamento e algumas ideias que ainda estão por implementar.

6.Apêndice

Para iniciar o programa basta chamar o comando *“service run”* e como argumento, o caminho onde foi guardado o projeto. Exemplo*: "service run /home/lcom/projeto_principal/src/towerdefense -args “/home/lcom/”*.