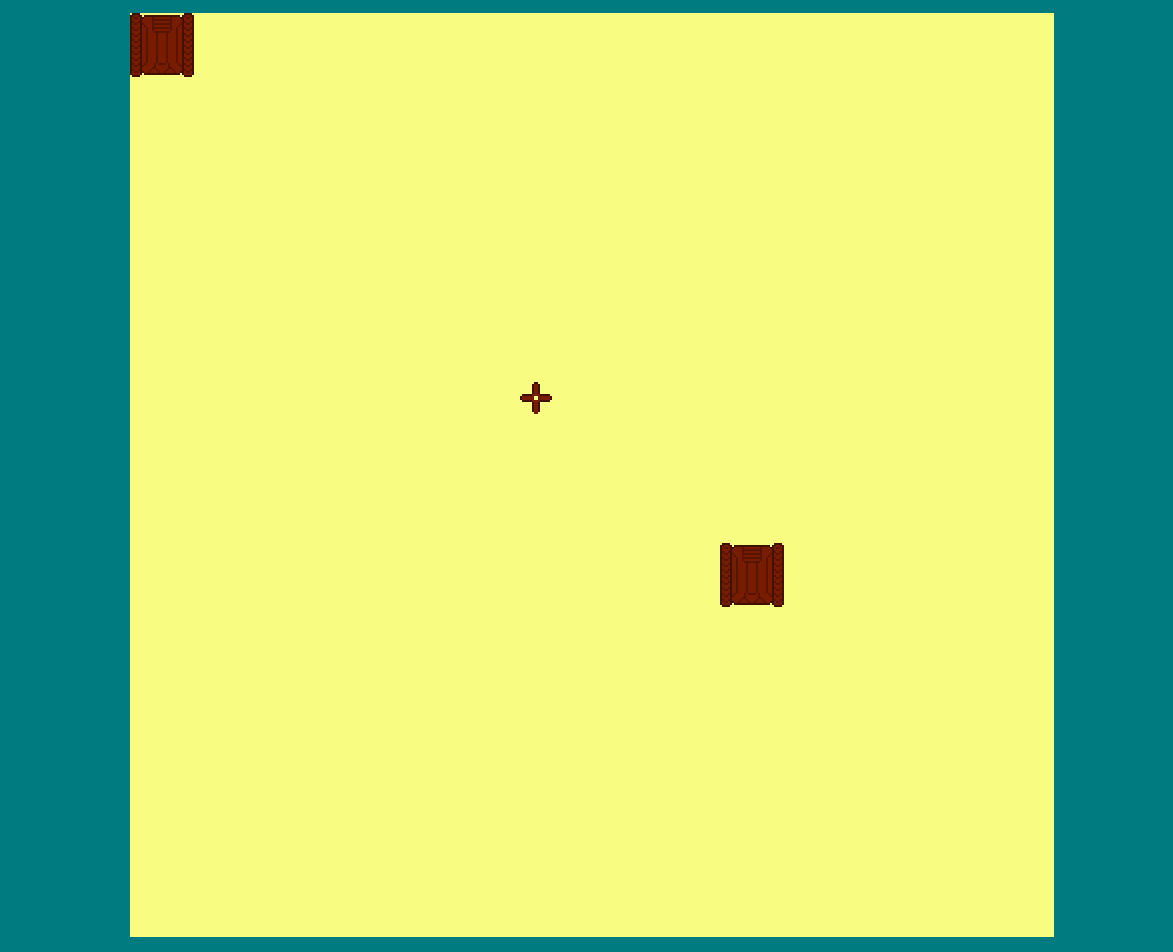
1. **Instruções de utilização do programa**
   1. **Menu**

Ao iniciarmos o programa, é nos apresentado um menu com dois botões, “Play” ou “Quit” .

O utilizador pode controlar o menu pelo rato e pode selecionar qualquer um do dois botões:

* Play: ao carregar neste botão o utilizador é levado para o jogo em si.
* Quit: ao carregar neste botão o utilizador sai do jogo, voltando para o terminal do minix.
  1. **Modo de Jogo**

O jogo consiste num shooter com vista de cima entre dois tanques controlados por 2 utilizadores no mesmo computador.

Ao entrarmos no modo de jogo os utilizadores são apresentados com este ecrã:

O utilizador consegue ver dois tanques, um deles controlado com WASD e dispara no rato e o outro controlado com as setas e dispara no numpad.

* 1 – Dispara na diagonal inferior esquerda.
* 4 – Dispara na horizontal esquerda.
* 7 – Dispara na diagonal superior esquerda.
* 8 – Dispara na vertical superior.
* 9 – Dispara na diagonal superior direita.
* 6 – Dispara na horizontal direita.
* 3 – Dispara na diagonal inferior direita.
* 2 – Dispara na vertical inferior.

O jogo acaba quando um dos tanques leva com uma bala do outro.

1. **Estado do Projeto**

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **Propósito** | **INT** |
| KBD | Movimento do Tanque e para disparar balas | **Y** |
| Timer | Controlar a frame rate | **Y** |
| Mouse | Navegar o menu e disparar balas | **Y** |
| Placa Gráfica | Desenhar o menu e o jogo | **N** |
| RTC | Para obter a hora do dia, mudando o jogo consoante | **N** |

* 1. **KBD**

Neste jogo usamos os interrupts que provem do KBE para ler o input do jogador, movendo o tanque ou fazendo outra ação consoante esse input.

Para lermos os inputs provenientes do KBE utilizamos a seguinte função:  
void (kbd\_get\_scancode)(kbd\_data\_t\* kbd\_data, uint32\_t wait\_ticks)

que, num interrupt, vai tentar buscar um scancode ao output-buffer do KBE.

Para processarmos os inputs provenientes da função anterior utilizamos esta função:

void process\_scancode(GameState state, kbd\_data\_t data)

que dado um struct kbd\_data\_t contendo data proveniente do KBD altera os objetos dentro da struct GameState como, por exemplo, os players.

* 1. **Timer**

Neste jogo utilizamos os interrupts provenientes do timer para controlar a frame rate do jogo, mais especificamente, usamos o timer para manter o jogo a 60 frames por segundo e para controlar quando e que uma coisa e desenhada.

A cada interrupt do timer utilizamos esta função:

int (canvas\_refresh)(GameState\* game)

para desenharmos no buffer o que vem na frame asseguir, considerando os inputs já processados do KBD, e esta função:

void gameStep(GameState \*state)

para processar as balas existentes, se elas colidem com algo e para atualizar da posição delas.

* 1. **Mouse**

Neste jogo utilizamos os interrupts provenientes do Mouse para calcular a posição dele e atualizar a posição dele no ecrã e também o usamos para disparar balas.

Utilizamos esta função:

void (mouse\_get\_data)(mouse\_data\_t\* mouse\_data, uint32\_t wait\_ticks) para ir buscar a data presente no output buffer do KBE, verificando antes se esta data provém do rato e esta função:

void mouse\_parse\_packet(mouse\_data\_t\* mouse\_data) para darmos parse da data proveniente da função anterior.

Com parsed data proveniente desta função conseguimos atualizar a posição do rato o ecrã e verificar se ele esta a premir algum botão.

* 1. **Placa Gráfica**

Neste jogo utilizamos a placa gráfica no modo 0x11A, com uma resolução de 1280x1024, com um modo de cor direto no qual vermelho tem 5 bits, verde tem 6 bits e azul tem 5 bits, ou seja, tem este formato 5:6:5.

Utilizamos double-buffering, onde copiamos a data de um buffer para outro.

Como foi mencionado no timer, utilizamos o canvas\_refresh para atualizar o que estava desenhado no ecrã. Em termos mais aplicados, atualizamos o buffer que não está ligado ao ecrã e no fim de atualizarmos esse buffer chamamos esta função:

int video\_switch() para fazer a copia do buffer atualizado para o buffer do ecrã.

Como temos objetos que se mexem (os tanques, por exemplo) temos colision detection para verificar se existe colisão entre esses objetos, por exemplo, entre tanques e paredes, balas e paredes, etc…

Essa collision detection é feita através desta função:

bool checkCollisions(Object \*obj1, Object\* obj2) que verifica se um objeto esta dentro do outro com ajuda desta função:

bool pointInObject(Object \*obj, uint16\_t x, uint16\_t y) que, essencialmente, verifica se um dado ponto esta dentro de um dado objeto.

* 1. **RTC**

Neste jogo utilizamos RTC para fazermos uma mudança de cenário.

Utiliza-se esta função:

int getRTCOutput(uint8\_t cmd, uint8\_t \*output) para ir buscar as horas. Depois pegamos nesse resultado e verificamos se a hora for mais que 21 mudamos o cenário do jogo, mudando o cenário através desta função:

int (nightTransform)(uint32\_t\* color) que mete as cores do jogo num tom mais escuro.

1. **Estrutura/Organização de código**
   1. **Image.c**

Neste ficheiro definimos um novo data type Image que consiste na informação relevante a uma imagem, se é um sprite ou uma forma, o tamanho dela e as cores dela. Também definimos funções para criar este data type consoante se queremos criar um sprite ou uma forma.

* 1. **Canvas.c**

Utilizamos este ficheiro para tudo o que tenha a ver com desenhar no ecrã. E neste ficheiro que desenhamos tudo, os tanques, as paredes, as balas, o crosshair do rato e onde se invoca a função para copiar os conteúdos de um buffer para o outro.

* 1. **Object.c**

Neste ficheiro definimos um novo data type Object que consiste na informação relevante a um objeto, seja ele um tanque ou uma parede. Também se definiu funções para a sua criação, para mudar as informações relevantes a um objeto, por exemplo, a sua posição e para verificar colisões com outros objetos.

* 1. **List.c**

Neste ficheiro definimos um novo data type List que consiste numa double linked list e ListElement que , através do void\*, pode conter qualquer tipo de informação. Também se definiu funções para a criação desta lista e para a remoção dela, para retirar elementos desta lista, tanto na Front como na Back e para adicionar elementos á lista.

* 1. **Bullet.c**

Neste ficheiro definimos um novo data type Bullet que contem toda a informação relevante a uma bala. Também definimos funções para eliminar uma bala e para inverter a direção de uma bala em caso de colisão.

* 1. **Game.c**

Neste ficheiro definimos funções para controlar o jogo, uma função para iniciar o jogo, outra função para o game loop, que consiste no interrupt loop, e uma função para parar o jogo, que consiste em deallocar memória previamente alocada.

* 1. **Keyboard.c**

Neste ficheiro criamos o driver para o KBD, criando funções para ler scancodes provenientes do output buffer, funções para verificar o status do KBC e funções para ativar e desativar interrupts.

* 1. **KBC.c**

Neste ficheiro definimos funções auxiliares que são usadas no keyboard.c, por exemplo uma função para ir buscar o status do KBC e uma função escrever comandos para o KBC.

* 1. **Mouse.c**

Neste ficheiro criamos o driver para o Mouse, criando funções para ler a data presente no output buffer do KBC, funções para ativar e desativar interrupts do Mouse e uma função para dar parse da data proveniente do output buffer.

* 1. **Rtc.c**

Neste ficheiro criamos o driver para o RTC (Real Time Clock), criando funções para ler o output do rtc e funções para verificar e converter para binário se for necessário.

* 1. **Timer.c**

Neste ficheiro criamos o driver para o Timer, criando uma função para criar a control word, outra para enviar essa control word para o registo de controlo do timer, outra para mudar a frequência do timer e outras para ativar e desativar os interrupts do timer.

* 1. **Vídeo.c**

Neste ficheiro criamos o driver da placa gráfica, utilizando o VBE, onde criamos funções para ir buscar as informações sobre um dado modo de vídeo, uma função para meter o minix num dado modo gráfico, outra para meter em modo de texto, na memória.

* 1. **Utils.c**

Neste ficheiro definimos funções utéis como uma função para ir buscar o most significant byte, outra para ir o least significant byte e uma função para invocar um sys\_inb que devolve data com 8 bits.

* 1. **Main.c**

Neste ficheiro define-se a função chamada no terminal do minix. A função consiste em criar o jogo e passar para o game loop onde o jogo se situa.