Best_Sell_Brand

[Sistemas Operativos 2]

2017/2018

**META 2 – PHOENIX MULTIPLAYER**

Emanuel Fragoso, 21240358

Maurício Carvalho, 21190216

Índice

[0. Introdução 3](#_Toc517402635)

[1. Estrutura de dados 4](#_Toc517402636)

[2. Cliente 5](#_Toc517402637)

[3. Servidor 5](#_Toc517402638)

[4. Gateway 6](#_Toc517402639)

[5. DLL 7](#_Toc517402640)

[6. Memória Partilhada 7](#_Toc517402641)

[7. Named Pipes 8](#_Toc517402642)

[8. Observações 8](#_Toc517402643)

[9. Conclusão 9](#_Toc517402644)

[10. Referências 9](#_Toc517402645)

# 0. Introdução

Este relatório serve de informação à fase 2 do jogo a implementar com a API Win32 do Windows – Phoenix Multiplayer. Nesta fase é pretendida implementar uma lógica Cliente-Servidor com utilização de um Gateway intermediário e de uma DLL, Named Pipes, Memória Partilhada e Mecanismos de Sincronização ( Mutexes, Eventos e Semáforos).

O programa é implementado com a API Win32 do Windows , em que se pretende fazer uso desta para implementação gráfica do Cliente.

# 1. Estrutura de dados

A memória partilhada está dividida em 2 partes

* 1. Buffer circular

typedef struct {

msg dados[10];

int iEscrita;

int iLeitura;

}bufferinfo;

typedef struct {

int tipo;

int aux1;

int aux2;

int aux3;

int aux4;

int aux5;

TCHAR aux6[1024];

TCHAR aux7[1024];

TCHAR aux8[1024];

}msg;

// Para tratamento posterior das mensagens enviadas pelo Gateway( oriundas do cliente).

* 1. Estrutura de Configuração ( mapa )

Estrutura de dados que aglomera toda a informação necessária enviar ao cliente e processada pelo servidor, ou seja, os dados relativos ao jogo em execução.

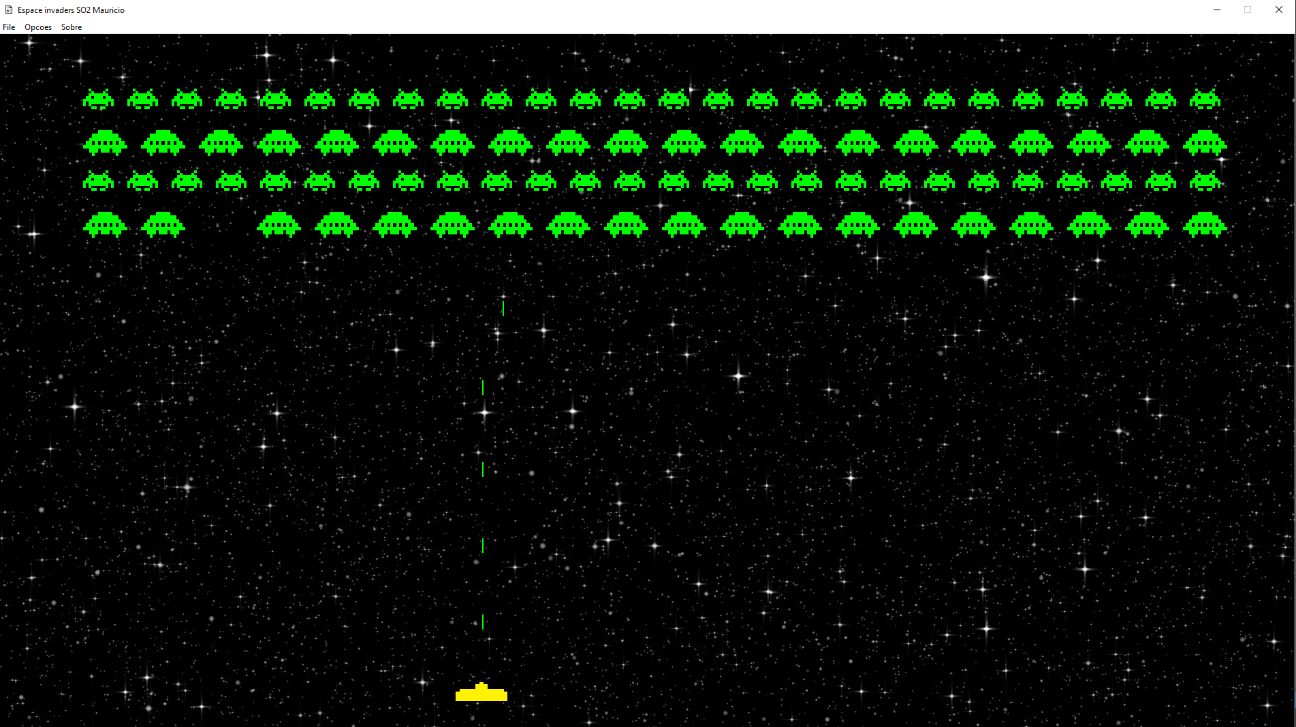
# 2. Cliente

2.1 Inicio de Jogo

O cliente no menu de inicio não está conectado ao jogo, para isso necessita de inserir um nome ( login numa dialog box) que terá que bater certo com o login no servidor (Explicado adiante no tópico do servidor).

Após este passo ser validado, o jogo entra em execução.

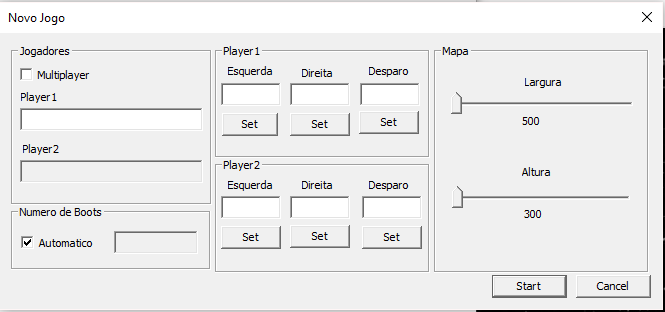
2.2 Execução do jogo



# 3. Servidor

O Servidor é o responsável pelo estado e lógica do jogo, bem como as configurações de arranque do mesmo.

As configurações iniciais consistem como na imagem:



Os Nomes dos Jogadores são definidos nas configurações de Novo Jogo. Definições de teclas e dimensões de janela podem também ser definidas.

# 4. Gateway

O Gateway Trata de toda a informação que circula entre o Servidor e o Cliente.

4.1- Servidor<->Gateway

Em memória está mapeada a estrutura de jogo em que o ambos o gateway e o Servidor têm acesso.   
O gateway lê esta estrutura sempre que o evento do servidor lhe dá sinal que esta foi atualizada.

No Gateway, uma Thread (produtora) escreve num buffer circular, também em memória, a informação que o servidor(consumidor) vai ler. Esta informação contém os comandos provenientes do cliente.

4.2 Gateway<->Cliente

Uma thread (main) que escuta o evento da estrutura de dados, vai enviar por named pipe ( broadcast) para todos os clientes conectados.

Uma thread (trataCliente) que escuta os comandos provenientes por named pipe, escreve diretamente no buffer circular referido anteriormente.

# 5. DLL

A DLL apresenta o código respetivo à criação e manipulação da memória Partilhada.

Mapeamento de buffer circular ( Gateway->servidor)

Mapeamento da estrutura de dados de jogo.

# 6. Memória Partilhada

Foi utilizado um buffer circular onde através do Gateway, os clientes enviam informação e o servidor recebe a informação, e vice-versa. A cada envio de informação do Gateway a varável de input (escrita) irá aumentar em um valor. A cada leitura do servidor a variável de output (leitura) será incrementada em um valor.

Para obter resultados eficazes com o buffer circular, foi necessário recorrer a mecanismos de sincronização. Esses mecanismos são: 2 Semáforos e 1 Mutex. Os semáforos evitam o servidor ler conteúdo quando o buffer está vazio, e o gateway postar conteúdo quando o

buffer está cheio. O mutex por outro lado foi utilizado para garantir que as variáveis de input e output não sejam alteradas em simultâneo.

# 7. Named Pipes

Foram utilizados Named Pipes unidirecionais para estabelecer uma conexão entre Cliente-Gateway.

Ao iniciar o Gateway, este lança uma Thread, que aguarda por pedidos de conexão dos clientes. Caso o Gateway se encontre num estado onde poderá aceitar as conexões dos clientes, lançará uma Thread para cada respetivo cliente.

O “Named Pipe” do Gateway é conhecido através de uma path fornecida aos clientes.

Existe um Pipe exclusivamente para o Gateway enviar informações do mapa para o cliente.

# 8. Observações

Com esta meta do trabalho, foi possível aperfeiçoar as técnicas de Sincronização (Eventos, Mutexes e Semáforos), uso de Memória Partilhada em Windows, uso de DLL e comunicação Cliente Servidor através de um gateway.

O mapeamento do buffer circular tem a sua chamada no servidor com uma função local ( quando deveria estar na DLL) uma vez que esta tornava outra função (tratamsg) dependente dela e assim sucessivamente por outras funções. Para não perder tempo e porque se reparou nisto numa fase final, decidimos manter assim a função no servidor.

# 9. Conclusão

Foi despendido muito tempo a procurar as melhores soluções para implementar a melhor solução para o posicionamento das naves inimigas tendo em conta o reconhecimento da vizinhança da mesma. Vários problemas no Debug das soluções foi um problema com o Visual Studio dificultando e retirando várias horas de trabalho.

Apesar de o trabalho não estar completo, o que faltou foi essencialmente adicionar mais logica de jogo, para que este se tornasse mais robusto e apelativo( acabar a logica dos power -ups e bombas) , porém achamos que demos uso a toda a matéria que os professores nos transmitiram ao longo do semestre!

# 10. Referências

1. Pdfs e exercícios das Aulas de Sistemas Operativos 2.
2. Recurso ao msdn para conhecimento da API win32 do Windows

[[https://social.msdn.microsoft.com/search/en-US?]](https://social.msdn.microsoft.com/search/en-US?%5d)