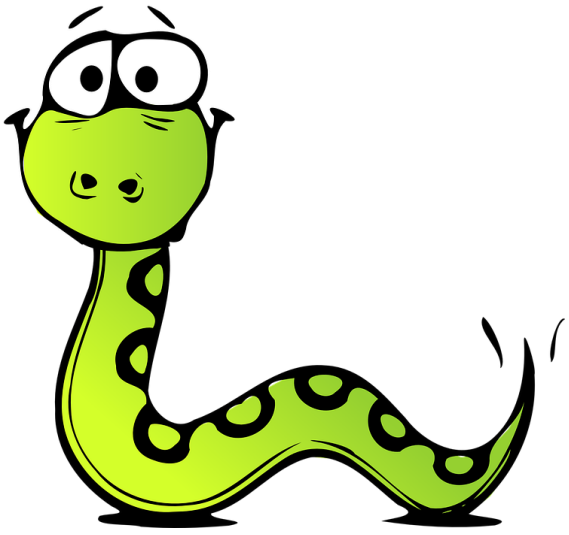
[Sistemas Operativos 2]
2016/2017

Fase 2 – Snake Game

Trabalho Prático de SO2

Rui Teixeira 21250189
Rafael Henriques 21250203
Curso - Engenharia Informática



Índice

1. Introdução

O relatório faz apontar à fase 2 da meta do jogo Snake Game. Neste jogo é pretendido implementar uma lógica Cliente-Servidor com utilização de DLL, mecanismos de comunicação 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.

2. Estruturas de Dados

A memória partilhada está divida em 6 partes. As figuras seguintes, mostram de uma forma mais rápida os dados encontrados em cada uma delas.

2.1. Estrutura de Configuração

```
typedef struct
{
    int structVal; // 1
                   //if username "0" it has no players
    TCHAR username1[MAX_TAM];
    TCHAR username2[MAX_TAM];
    int coop;                       // 1 - 0
    int hasConfig;                  // Check if there's a config via Shared Memory
    int fieldSizeX;                 // 20 - 80
    int fieldSizeY;                 // 20 - 80
    int snakeInitialSegmentSize;// 3 - 8
    int numSnakeBots;               // 1 - 19
    int numMaxPlayers;              // 0 - 18
    int itemsFrequency;             // 1 - 3
    int numObjects;                 // 10 - 30
    int temporaryEffectTime;        // 5 - 20
    int withObjects;                // TRUE - FALSE (1-0)
    int isOnline;                   // 1 - remote game
    int giveup;                     // 1 - stop config
}Config;
```

Figura 1 – Estrutura de Configuração

2.2. String

Espaço na memória partilhada reservado á devolução de uma String para o cliente.

2.3. ClientInput

Estrutura de dados que aglomera o input dos jogadores. Esta estrutura é recebida pelo servidor através de um buffer circular.

2.4. Info

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.

Mais detalhadamente, a estrutura é constituída por:

- ⇒ Jogo Online ou não;
- ⇒ Servidor desligou;
- ⇒ Jogo terminou;
- ⇒ Número de Jogadores;
- ⇒ Número de Bots;
- ⇒ O mapa, que é constituído por:
  - → Dimensão;
  - → Número de Obstáculos;
  - → Coordenadas dos Obstáculos;
  - → Número de Objetos;
  - → Coordenadas dos Objetos;
  - → Tipo do objeto;
- ⇒ A representação das Snakes:
  - → Username;
  - → Tamanho da Snake;
  - → Coordenadas dos segmentos da snake;
  - → Teclas Invertidas;
  - → Efeito negativos;
  - → Direção atual;

2.5. States

Estrutura que contém o estado do servidor, assim como a quantidade de players conectados.

3. Estados

Para simplificação da construção do programa, o jogo foi dividido em 4 estados, como mostra a Figura 1.

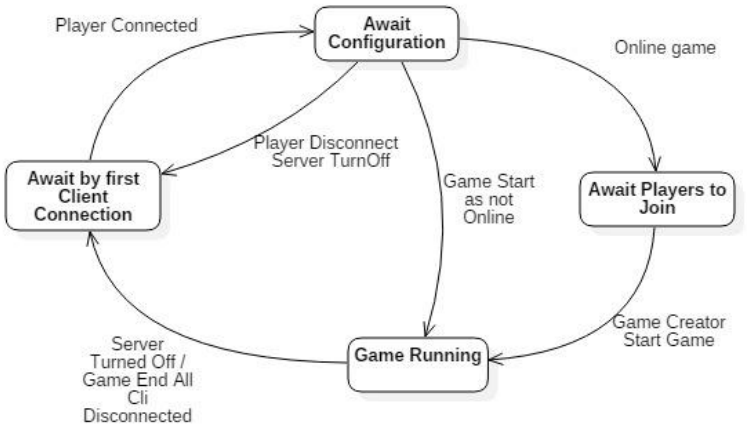


Figura 2 - Estados do Jogo

3.1. Breve Descrição dos Estados

Estado 0

O primeiro estado corresponde ao servidor aguardar pelo primeiro Cliente para Configurar o jogo. Após este, o servidor passa ao estado 1.

Estado 1

Após o primeiro cliente conectado e o servidor apresentar-se no estado 1, todas as novas conexões por outros clientes não serão respondidas.

Estado 2

Caso o jogo seja no modo online, os jogadores poderão juntar-se ao jogo neste estado. Quando o cliente criador assinalar, o jogo será iniciado.

Estado 3

Estado 3 representa o jogo a correr. Após o jogo acabar ou ser interrompido todos os clientes serão desconectados.

4. Cliente

O cliente está implementado em Win32 Project, para Windows, desenvolvido no Visual Studio 2015. Os subtópicos e imagens seguintes descrevem as funcionalidades do jogo.

4.1. Start Menu

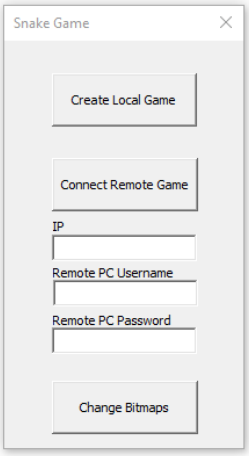


Figura 3 - Start Menu

Quando o cliente se encontra no menu de inicio não está conectado ao jogo. A figura 3 mostra três opções: conectar ao servidor de modo local, conectar de modo remoto e editar os bitmaps

4.2. Configuration Menu

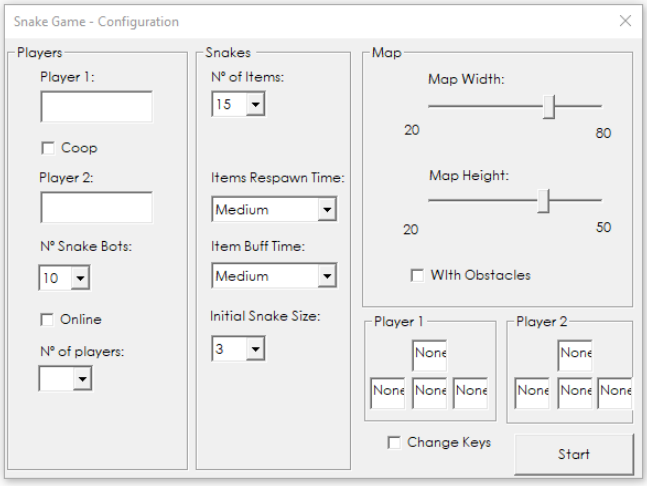


Figura 4 - Configuration Menu

A fase de configuração

4.3. Player Join Game Menu

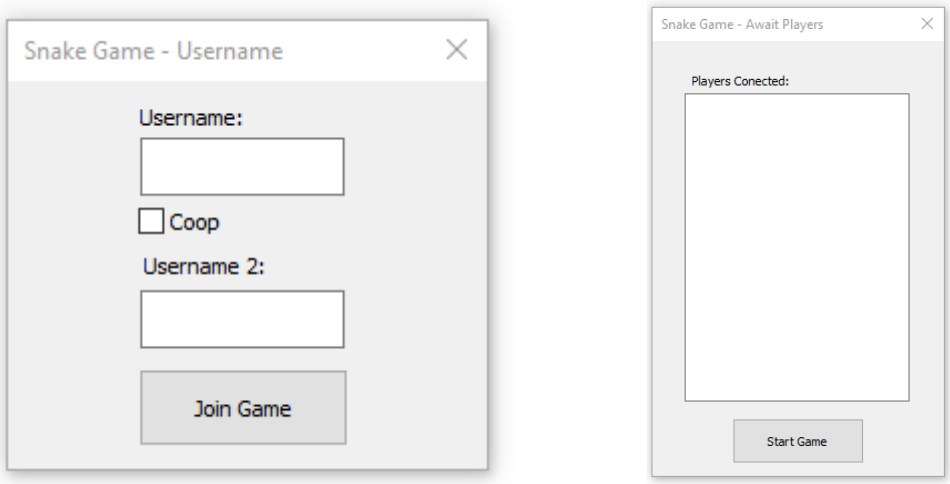


Figura 5 – Conexão de um jogador remoto e Lista de jogadores Conectados

(Funcionalidade não implementada a funcionar)

4.4. Game Running

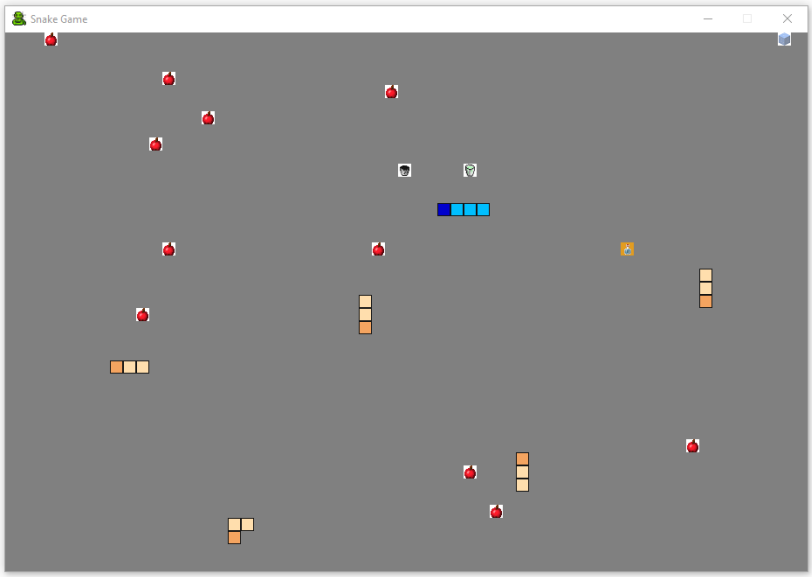


Figura 6 – Jogo a Correr

O jogo é representado por uma Windows, em que a snake do player 1 é azul, do player 2 verde e dos bots e outros jogadores são amarelos com contrastes diferentes.

(O modo multiplayer e online não estão implementados)

4.5. Bitmap Editor

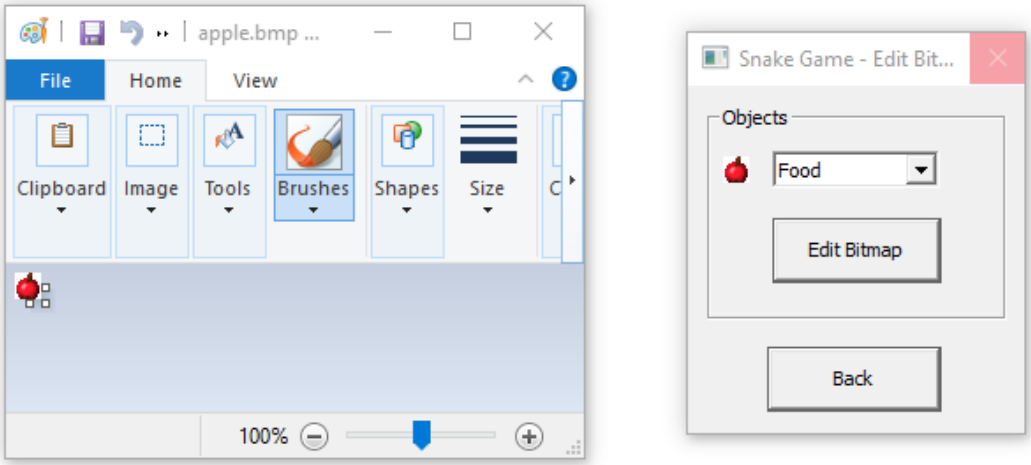


Figura 7 – Editor de Bipmap abre o paint na imagem correspondente

O servidor recorre a 10 imagens bitmaps, em que cada uma representa um objeto do jogo, que pode ser coletado pela cobra do jogador. É possível usar o bitmap editor para editar uma qualquer imagem (bitmap) implementada no cliente.

5. Servidor

O servidor é o responsável pelos estados do jogo, toda a lógica do jogo e gestão de conexão.

5.1. Registry

Os resultados do final de cada jogo serão guardados no registry, dando respetivo a cada username. Caso o username já se encontre registado no registry, apenas a pontuação máxima será guardada.

5.2. Sincronização

Para a sincronização do Servidor são utilizados mutexes, eventos e sleeps. Durante o jogo as threads dos bots e dos players partilham um mutex em comum. As threads dos objetos são sincronizadas com um mutex partilhado entre elas.

Cada vez que um bot, objeto ou snake do player faz um movimento, um evento manual é utilizado para a thread enviar a informação do jogo atualizada aos players.

Para a sincronização do timing (speed) dos players, bots e respawn dos items são utilizados Sleeps.

Um Sleep é utilizado depois da libertação de um mecanismo de sincronização para evitar esperas e timeouts.

6. DLL

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

7. Memória Partilhada

A memória partilhada é divida em 6 partes como foi descrito no tópico 2. Para sincronizar a memória partilhada recorreu-se a eventos, mutex e semáforos partilhados em Servidor e Cliente Local.

Cada vez que o servidor atualiza informação despertará um evento, recebido pelo cliente local conectado.

7.1. Buffer Circular

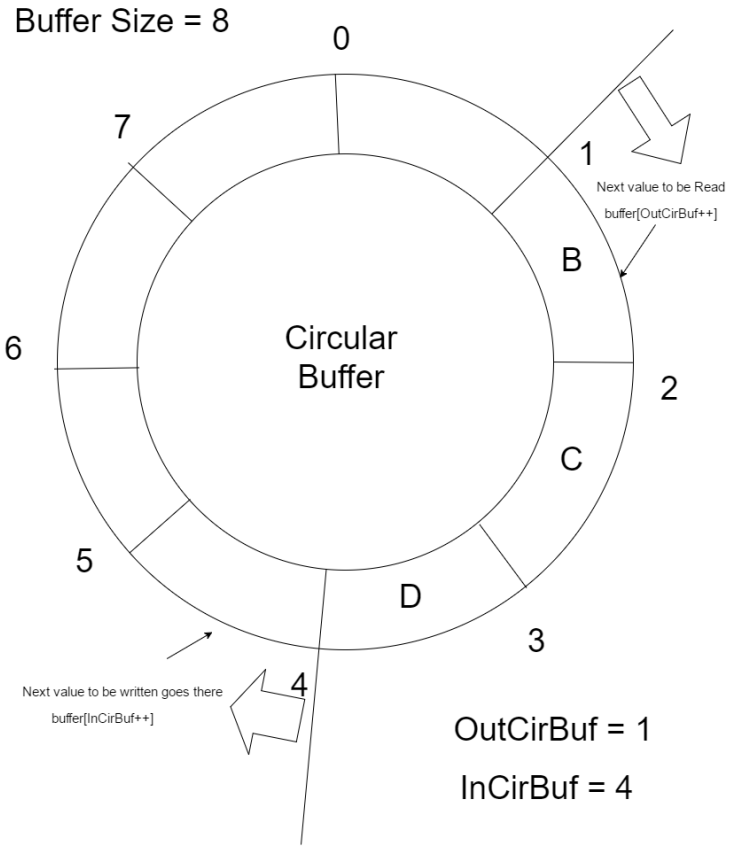


Figura 8 – Buffer Circular

Como pode ser observado na figura 8, foi utilizado um buffer circular onde os clientes enviam informação e o servidor recebe a informação. A cada envio de

informação do cliente 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 cliente 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.

No caso do cliente local, terá acesso direto ao buffer circular, obtendo tempos de resposta baixos.

O tamanho do buffer utilizado no trabalho é de 2^4 = 16 pedidos de tamanho.

8. Named Pipes

Foi utilizado Named pipes do tipo full-duplex, para estabelecer uma conexão remota entre Cliente-Servidor. Para os clientes poderem conectar-se ao servidor remoto, foi configurada a estrutura de Security Attributes.

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

O "Named Pipe" do servidor é conhecido através de uma path fornecida aos clientes.

9. ServiçoNT

Instalamos um ServiceNT do servidor, pondo a DLL e o executável do serviço na raiz c:\. Depois ativamos o serviço no Gestor de Serviços do Windows, mas ao tentar conectar o cliente ao servidor, não obtemos qualquer tipo de resposta.

Não conseguimos ativar o Serviço com sucesso desistindo de o fazer, mas de qualquer modo podemos demonstrar como o fizemos.

10. Observações

Não conseguimos implementar o modo de jogo com vários clientes em simultâneo, devido ao desconhecimento da metodologia necessária para implementar um modo de comunicação entre cliente-servidor, sólido que permitisse a integração de diversos Clientes conectados em simultâneo.

Despendemos imenso tempo a testar pequenos programas para podermos aplicar as funções corretamente, pois as funções da API do Windows apresentam muitos pormenores, ao qual necessitam de ser testados.

Apesar do trabalho não ter todas as funcionalidades, exploramos os diversos mecanismos lecionados na cadeira de SO2.

11. Conclusão

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. Também foi possível explorar outros campos, como por exemplo o Registry, edição de bitmaps, ServiceNT e Win32 API.

12. Referências

1. Pdfs e exercícios das Aulas de Sistemas Operativos 2.
2. draw.io para desenhar os diagramas [https://www.draw.io/]
3. Recurso ao msdn para conhecimento da API win32 do Windows [https://social.msdn.microsoft.com/search/en-US?]
4. Imagem da cobra no relatório tem licença gratuita [https://pixabay.com/p-303696/?no_redirect]
5.