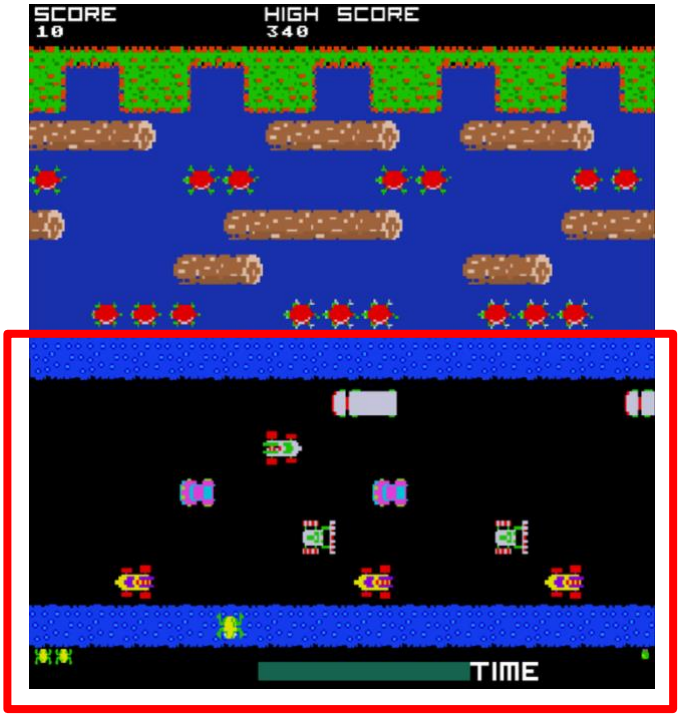
isec Engenharia	**Sistemas Operativos II** Engenharia Informática – DEIS	**2022/23**

Trabalho Prático

As regras que regem o trabalho prático encontram-se descritas na ficha da unidade curricular, chamando-se a atenção para a sua leitura.

Frogger

O trabalho prático consiste na implementação do jogo "Frogger" (https://froggerclassic.appspot.com, https://play.google.com/store/apps/details?id=frogger.juegosde&hl=pt_PT&gl=US) em modo multiutilizador, com um máximo de dois jogadores. No âmbito deste trabalho, todos os intervenientes serão **processos** a correr na mesma máquina ou eventualmente em duas máquinas distintas. Os programas envolvidos terão interfaces consola ou gráfica.



O jogo consiste em ajudar um animal (um sapo) a atravessar uma estrada e um rio movimentados. De forma a simplificar a mecânica de jogo apenas se pretende implementar a parte da estrada, que consiste na metade inferior do ecrã (assinalada com uma caixa vermelha). Para atingir o objetivo, o jogador poderá utilizar as teclas de direção (setas) ou o rato para movimentar o sapo em qualquer sentido: cima, baixo, esquerda e direita. A área de jogo está dividida em três faixas horizontais que ocupam toda a largura do ecrã: *área de partida*, na parte inferior do monitor (cor azul); *área da estrada*, na parte central com várias faixas de rodagem (cor preta); e *área de chegada*, na parte superior (cor azul). Durante o jogo existem carros a circular nas faixas de rodagem em ambos os sentidos. Em cada faixa de rodagem os carros circulam todos no mesmo sentido: da direita para a

esquerda ou da esquerda para a direita. Sempre que o sapo é atingido por um carro volta à área de partida. Quando conseguir atingir a área de chegada vence o jogo e passa de nível.

Neste enunciado, "sistema" referir-se-á ao jogo. Sempre que for necessário referir o sistema operativo, serão usadas estas duas palavras explicitamente.

1. Elementos do jogo e lógica geral

O jogo tem os seguintes elementos, aos quais correspondem os programas envolvidos:

- Servidor de controlo do jogo – programa **servidor** (apenas uma instância em execução)
- Operador para acompanhar o jogo – programa **operador** (podem existir várias instâncias)
- Cliente para jogar o jogo – programa **sapo** (podem existir até duas instâncias)

Cada programa desempenha apenas um e um só papel. Apenas existirá um **servidor** e uma ou mais instâncias do **operador** em execução na mesma máquina. Podem existir até duas instâncias em simultâneo do **sapo** na mesma máquina ou em máquinas distintas.

Lógica e funcionamento geral

O **servidor** é o ponto central do sistema, controla todo o jogo e interage com os restantes elementos: clientes e operadores. Gere a área de jogo, que é constituída por N faixas de rodagem e pelas zonas de partida e de chegada. O sapo apenas se pode movimentar entre posições que não se encontrem ocupadas pelo adversário ou pelos carros que estão em circulação. Pode assumir que a área de jogo terá uma dimensão máxima de 10 linhas (1 linha para a área de partida, até 8 linhas para as faixas de rodagem e mais 1 linha para a área de chegada) por 20 colunas. O número máximo de carros a circular em cada faixa é 8.

O **operador** é uma aplicação que pode ser executada na mesma máquina onde está a ser executado o servidor e que permite visualizar o estado do jogo e gerir alguns aspetos do mesmo. Por estado do jogo entende-se o mapa atual do jogo, as posições dos carros e jogadores e a informação de texto (pontuação, etc.).

O **sapo** é a aplicação utilizada pelo jogador que interage com o servidor para indicar os movimentos do seu personagem e observar o estado geral do jogo.

2. Utilização e funcionalidade detalhada

O programa **servidor** será o primeiro programa a ser lançado. Apenas pode existir uma instância do servidor e isso deve ser acautelado. Os programas **operador** e **sapo** só podem ser executados caso o servidor já se encontre em funcionamento.

Servidor

Lançamento

O programa **servidor** é lançado pelo utilizador. Deve assegurar-se que ainda não estava a correr (se estiver, a nova instância termina, alertando o utilizador desse fato).

Gere toda a informação relativa ao jogo, disponibiliza informação ao programa **operador** e interage com os programas **sapo** sempre que tal seja necessário.

A aplicação deve permitir a especificação, através da linha de comandos, do número de faixas de rodagem e a velocidade inicial dos carros (comum para todas as faixas de rodagem). Esta informação deve ser armazenada no ***Registry*** e utilizada em execuções subsequentes do programa, sempre que a mesma não seja especificada através da linha de comandos. Assuma que durante o jogo as velocidades utilizadas em cada faixa de rodagem podem mudar de forma aleatória (por exemplo, na passagem de nível).

Funcionalidade principal

- Controla a informação do jogo.
    - O número de faixas de rodagem e a velocidade inicial dos carros são passados através da linha de comandos ou encontram-se definidos no ***Registry***.
- Determina de forma aleatória a posição dos sapos. Devem estar localizados na área de partida e não podem estar sobrepostos.
- Recebe comandos do **operador** e desencadeia as ações necessárias ("parar movimento dos carros", etc.).
- Aceita os jogadores que se ligam através do programa **sapo**.
- Recebe por parte dos clientes (programa **sapo**) os movimentos que pretendem efetuar e mantém atualizada toda a informação do jogo.

Interface com o utilizador

Interface segundo o paradigma de consola, seguindo a lógica de comandos (não são menus). A interface deve permitir receber os seguintes comandos:

- Suspender e retomar o jogo.
- Reiniciar o jogo.
- Encerrar todo o sistema (todas as aplicações são notificadas).

Outros aspetos

Se um jogador não efetuar qualquer movimento durante 10 segundos, o seu sapo volta para a zona de partida. Ao atingir a área de chegada, um jogador passa automaticamente para o próximo nível. A cada novo nível, a velocidade e o número de carros aumentam (fica ao critério dos alunos o fator de aumento). Desta forma, não existe um número máximo de níveis – o nível vai incrementando até que o jogador perca.

Operador

Lançamento

O programa **operador** é lançado explicitamente pelo utilizador. Podem existir várias instâncias do operador em execução.

Funcionamento e interface com o utilizador

Interface segundo o paradigma de consola que apresenta o ecrã de jogo com os jogadores que se encontram a jogar em determinado momento. Esta informação estará permanentemente visível e será atualizada em tempo real.

A interface deve permitir receber os seguintes comandos:

- Parar o movimento dos carros durante um determinado período de tempo (especificado em segundos).
- Inserir obstáculos nas faixas de rodagem que são intransponíveis pelos carros e pelos sapos.
- Inverter o sentido da marcha de determinada faixa de rodagem.

Funcionalidades principais

- Interagir com o **servidor** conforme for necessário para mostrar o estado do jogo, incluindo: movimento dos carros, posição dos sapos, pontuação, etc.
- Modificar o comportamento do jogo através do envio de comandos ao **servidor**.

Sapo

Lançamento e funcionamento

O programa **sapo** é lançado pelo utilizador sempre que pretende jogar. Cada instância representa um novo jogador.

Existem duas modalidades de jogo: individual e competição. No primeiro caso o jogador compete contra o tempo e à medida que vai avançando de nível, o jogo torna-se mais difícil (ex.: a velocidade e o número de carros por faixa de rodagem vão aumentando a cada nível, etc.). No segundo modo de jogo, o jogador compete contra um adversário. Neste modo continuam a existir os níveis de jogo, e ganha o jogador que chegar mais vezes à zona de chegada e que, portanto, terá a pontuação mais alta.

Funcionamento e interface com o utilizador

Interface gráfica *Win32* que apresenta o jogo e toda a informação. Esta informação estará permanentemente visível e será atualizada em tempo real.

O programa começa por solicitar ao utilizador o nome e o tipo de jogo que pretende. Tratando-se da modalidade de jogo individual, o jogo começa de imediato. Na modalidade de competição, terá de aguardar a chegada de um adversário para se dar início ao jogo.

No decurso do jogo o utilizador poderá utilizar as teclas de direção ou clicar em cima da posição para onde pretende movimentar o sapo. Em qualquer dos casos a nova posição do sapo deve ser sempre contígua à posição atual.

Funcionalidades principais

- Interage com o **servidor** e com o utilizador.
- Este programa apenas comunica com o **servidor**, não existindo comunicação direta com o **operador**.

3. Formas de comunicação entre aplicações e recursos

A seguinte descrição refere-se apenas à comunicação entre processos. Se em determinada situação o enunciado refere que deve ser usado um mecanismo de comunicação específico, então tem mesmo que usar esse mecanismo nessa situação**.**

- A comunicação entre o **servidor** e os programas **sapo** é feita exclusivamente por *named pipes*, em ambas as direções.
    - **Importante**: Os programas **sapo** apenas podem comunicar diretamente com o **servidor**.
- A comunicação entre o **servidor** e o programa **operador** é feita exclusivamente por memória partilhada. O acesso ao estado do jogo (mapa, carros, sapos e respetivas posições) é feito através da leitura direta da memória partilhada. Toda a restante comunicação do programa **operador** para o **servidor** é feita através do paradigma de produtor/consumidor (*buffer* circular).

Qualquer fluxo de informação no sistema tem de respeitar estas restrições. Por exemplo, no cenário em que o **operador** insere obstáculos numa faixa de rodagem, esta informação deve ser enviada ao **servidor**, que depois

a encaminha para o(s) programa(s) **sapo(s)**. Qualquer outro fluxo de informação não especificado no enunciado fica ao critério do aluno, desde que realmente faça sentido.

A identificação e correta aplicação de mecanismos de notificação assíncrona e de sincronização fica a cargo dos alunos considerando os cenários em que são necessários, seguindo a arquitetura e a implementação do trabalho. A não aplicação ou o uso incorreto destes mecanismos leva a penalizações na avaliação.

Os mecanismos de comunicação e de sincronização devem constar de um diagrama claro e inequívoco a incluir no relatório que acompanhará a entrega do trabalho.

DLL

Todo e qualquer acesso (leitura e escrita) à memória partilhada por parte do **servidor** e **operador** terá de ser feito através de funções disponibilizadas por intermédio de uma DLL, a juntar projeto. Esta DLL permitirá encapsular todo o acesso à memória partilhada. Tanto a criação, bem como a remoção dos mecanismos do sistema operativo que se julguem necessários, deverão ser realizadas pela DLL.

4. Aspetos em aberto

Os seguintes aspetos devem ser definidos pelo aluno:

- Texto dos comandos escritos.
- Pormenores gráficos de visualização.
- Formato das mensagens trocadas entre as aplicações.
- Detalhes do modelo de dados para representação do estado do jogo.
- Mecanismos de sincronização: quais e onde são necessários.
- Outros aspetos não previstos ou não explicitamente descritos.

Devem ser tomadas decisões autónomas e lógicas quanto a estes aspetos, e que não desvirtuem o sistema pretendido nem evitem os conteúdos que se pretendem ver aplicados. O sistema resultante deve ter uma forma de utilização lógica.

O modelo de dados que representa o mapa de jogo deve ser simplificado ao máximo. Não será dada nenhuma valorização adicional ao uso de estruturas de dados mais complexas onde soluções mais simples seriam suficientes. A indicação de quantidades máximas referidas no enunciado deve ser aproveitada na simplificação das estruturas de dados.

5. Detalhes adicionais acerca dos requisitos

Ao desenvolver o sistema deve também ter em consideração os seguintes requisitos:

- A interface gráfica da aplicação **sapo** não deve cintilar. Deve-se utilizar a técnica de *double buffering*, abordada nas aulas.
- A interface da aplicação **operador** terá o seguinte comportamento:
    - Mostrar a informação do estado do jogo em tempo real (em modo de texto).
    - Desencadear ações que alteram o funcionamento do jogo e comunicá-las ao **servidor**.
- A interface do **sapo** terá o seguinte comportamento:
    - Utilização das teclas de direção para movimentar o sapo.

- Clique com o botão esquerdo do rato nas posições contíguas à localização do sapo, movem-no para essa posição.
- O menu principal da janela deve permitir alternar entre 2 conjuntos de *bitmaps* utilizados para representar os vários elementos do jogo (carros, sapos, etc.). Fica ao critério dos alunos escolher os *bitmaps* que constituem os vários elementos do jogo, não podendo existir *bitmaps* repetidos entre os 2 conjuntos.
- Clique com o botão direito do rato em cima do sapo permite reposicionar o sapo na zona de partida.
- Ao passar com o rato por cima do sapo mostra a quantidade de vezes que atravessou com sucesso a estrada.

- O uso de más práticas na implementação será penalizado. Alguns exemplos, não exaustivos, são:
  - Uso de variáveis globais.
  - Não utilização de programação genérica de caracteres (Unicode).
  - Má estruturação do código.
  - Mau uso de ficheiros *header* / *source* (por exemplo: utilização de ficheiros ".c" nas diretivas "include", inclusão de código fonte de funções em ficheiros ".h" em vez de apenas declarações, etc.).

6. Algumas chamadas de atenção

- Não coloque ponteiros em memória partilhada (pelas razões explicadas nas aulas). Isto abrange ponteiros seus e também objetos de biblioteca que contenham internamente ponteiros (por exemplo, objetos C++ String, Vector, etc.).
- Pode implementar o trabalho em C++, se assim quiser, desde que não oculte o API do Windows com *frameworks* de terceiros.
- Pode utilizar repositórios *git*, mas terá de garantir que são **privados**. Se usar um repositório público que depois seja copiado por terceiros, será considerado culpado de partilha indevida de código e terá o trabalho anulado.
- A deteção de situações de plágio leva a uma atribuição direta de 0 valores na nota do trabalho aos alunos de todos os grupos envolvidos, podendo ainda os mesmos estarem sujeitos a processos disciplinares.
- Todo o código apresentado poderá ser questionado na defesa e os alunos têm obrigatoriamente de o saber explicar. **A ausência de explicação coerente é entendida como possível fraude ou como falta de conhecimento, que naturalmente se reflete na nota.**

7. Regras e prazos

- **O trabalho é feito em grupos de 2 alunos**. Não são aceites grupos com 3 ou mais alunos. O trabalho foi desenhado e dimensionado para ser realizado em grupo de 2 alunos, sendo que, como exceção, podem ser aceites trabalhos individuais. Todavia é aconselhada e encorajada a constituição de grupos de 2 alunos. A avaliação será realizada com os mesmos critérios independentemente dos grupos terem 2 ou 1 aluno.
- O trabalho é composto por duas entregas: a Meta 1 e Meta Final.
- Nas entregas deve enviar o projeto do seu trabalho comprimido no formato **zip** (máximo de 10Mb), contendo apenas os ficheiros de controlo do projeto e de código fonte, ou seja, sem os binários (os ficheiros de *debug* das diretorias "*x64"*) e sem ficheiros *precompiled headers*.

Meta 1 – 13 de Maio

O material a entregar deverá ser o projeto em Visual Studio, compilável e sem erros de execução/compilação, dos seguintes programas com as respetivas funcionalidades:

- **Servidor** – Cria o(s) mecanismo(s) de comunicação e sincronização com o programa **operador**. Utiliza os argumentos da linha de comandos e a informação que se encontra definida no *Registry*. Posiciona dois sapos na área de partida de forma aleatória e implementa o movimento dos carros nas faixas de rodagem. Cria e gere as estruturas de dados a usar pelo sistema.
- **Operador** – Com uma interface do tipo consola permite visualizar o estado do jogo em tempo real (mapa de jogo, carros, sapos e pontuação). Interage com o utilizador e permite a troca de informação com o **servido**r através da memória partilhada. Implementa as funcionalidades de parar temporariamente o movimento dos caros, inserir obstáculos e inverter o sentido da marcha de uma faixa de rodagem

É também necessário entregar um relatório sucinto com a descrição dos mecanismos de comunicação e sincronização implementados (devem apresentar um diagrama/esquema com a arquitetura onde sejam evidentes esses mecanismos), assim como as estruturas de dados definidas. Deverão ainda incluir, em anexo, um pequeno manual de utilização de como usar o sistema, para já composto apenas de dois programas. Não deve incluir o código fonte no relatório, exceto em situações pontuais onde pretenda demonstrar a utilização de determinado mecanismo.

Meta Final – 17 Junho

O material a entregar deverá ser:

- O trabalho completo, com os programas que constituem o sistema totalmente implementados.
- Um relatório completo, com o máximo de 10 páginas, a descrever o seguinte: pontos essenciais da implementação de cada um dos programas envolvidos; estruturas de dados definidas e a sua utilidade; todos os aspetos que não estejam explicitamente mencionados no enunciado e que tenham sido decididos pelos alunos; e um diagrama com os mecanismos de comunicação e de sincronização. O relatório deve contemplar uma tabela onde indique quais os requisitos implementados, no formato:

ID	Descrição funcionalidade / requisito	Estado
...	...	implementado / não implementado (neste caso, indicar as razões)

8. Avaliação

O trabalho vale **8 valores**. A nota será atribuída com base nas funcionalidades implementadas, na qualidade das soluções adotadas, na forma como são explicadas no relatório e na qualidade da defesa.

A avaliação ocorre, essencialmente, na Meta Final. Na Meta 1 avalia-se o cumprimento dos objetivos estabelecidos através de uma apresentação obrigatória. Tal como descrito na ficha da unidade curricular, a avaliação será feita da seguinte forma:

- Nota da Meta 1 = fator multiplicativo entre 0,8 e 1,0.
    - A não comparência na apresentação obrigatória da Meta 1 fará com que a mesma não seja tida em consideração. Ou seja, o fator multiplicativo obtido será o mesmo que seria obtido se não entregasse a meta (0,8).

- Nota da Meta Final = valor entre 0 e 100 que representa a funcionalidade e qualidade do trabalho, e a qualidade da defesa.
- Nota final do trabalho = nota obtida na Meta Final * fator multiplicativo obtido na Meta 1.
- A Meta 1 não tem qualquer valor sem a Meta Final. Ou seja, grupos que não entregarem a Meta Final ou que não compareçam na sua defesa terão uma nota final de 0 valores.
- A falta da Meta 1 não impede a entrega da Meta Final. Apenas prejudica a nota final, já que a nota mínima da Meta 1 será automaticamente assumida (0,8).

O trabalho está planeado para ser feito ao longo do semestre e a entrega do trabalho prático é única para todo o ano letivo. Não existirá trabalho prático na época especial ou noutras épocas extraordinárias que os alunos possam ter acesso, sendo o exame sempre cotado para 12 valores.