Escola Superior de Tecnologia e Gestão de Beja

Curso de Engenharia Informática

Sistemas Operativos

Trabalho de Grupo N.º 2

Programação de um Jogo de Aventuras

Luís Garcia

Programação de um Jogo de Aventuras

Neste trabalho pretende-se que os alunos desenvolvam um jogo de aventuras utilizando os conhecimentos adquiridos nas aulas sobre programação sistema com a linguagem C.

Os jogos de aventuras tiveram provavelmente a sua origem no jogo de tabuleiro *Dungeons & Dragons (D&D)* onde os jogadores encarnam personagens que percorrem territórios em busca de tesouros e enfrentam monstros que se cruzam no seu caminho.

O primeiro jogo de computador deste tipo foi o *"Colossal Cave Adventure"* desenvolvido por Willie Crowther e Don Woods. Um outro exemplo é o Zork.

O jogo que se pretende que os alunos desenvolvam neste trabalho deverá conter pelo menos cinco elementos fundamentais: (1) jogador; (2) objetos do jogador; (3) local da aventura; (4) monstro e (5) tesouro. O jogador irá movimentar-se no local da aventura para tentar encontrar o tesouro, mas para isso poderá ter de enfrentar o monstro. O jogador poderá ainda transportar um objeto encontrado no local da aventura para enfrentar o monstro.

De seguida iremos indicar a forma como estes elementos deverão ser criados no programa.

O jogador consistirá numa variável do tipo estrutura. Deverá conter pelo menos os seguintes campos: (1) nome; (2) energia; (3) local onde se encontra; (4) a indicação se carrega um dos objetos (não: -1; sim: código do objeto), e (4) a indicação se carrega ou não o tesouro (não: -1; sim: 1).

O local da aventura deverá ser representado por um vector de elementos do tipo sala (ou célula). Esta última denominação é mais adequada se existirem espaços exteriores. Cada elemento deste vector será uma sala com os seguintes campos: (1) norte; (2) sul; (3) oeste; (4) este; (5) cima; (6) baixo; (7) descrição; (8) a existência ou não de um objecto; e (9) a existência ou não do tesouro. Os campos com as várias direcções: norte, sul, etc, irão conter o número da sala para onde o jogador se deslocará, se decidir seguir aquela direcção. Quando não existir uma passagem numa determinada direcção o campo deve conter o valor -1. A descrição da sala será uma cadeia de caracteres (*string*), que fornecerá uma descrição sobre o local onde o jogador se encontra. A existência de um objecto na sala deverá ser assinalada com o código do objecto (índice do objeto na tabela de objetos). Se não existir um objecto na sala este campo deve conter -1. O último campo indicará se existe ou não um tesouro naquela sala (não: -1; sim: 1).

Os objetos do jogador deverão ser representados por um vector de elementos do tipo objeto. Cada objeto de ter os seguintes campos: (1) nome; (2) nível de eficácia na luta.

O monstro deverá ser representado através de uma estrutura com dois campos: (1) energia; e (2) local onde se encontra. O monstro deve movimentar-se através das passagens entre as salas tal como o jogador.

Dependendo do tema do trabalho escolhido pelos alunos o local da aventura, o tesouro e o monstro (ou monstros) poderão assumir diferentes formas. Os alunos poderão utilizar a sua criatividade para criar ambientes originais e interessantes.

O pseudo-código básico de um programa de aventuras deste género pode ser expresso da seguinte forma:

```
Inicialização do jogador
Inicialização dos objetos
Inicialização do local da aventura
Inicialização do monstro

Enquanto não for fim de jogo
{
      Movimentar monstro
      Descrever a localização do jogador
      Aceitar comando do jogador
      Movimentar jogador
      Se a localização do jogador e monstro forem iguais
            Lutar
}
Apresentar resultado final
```

Na programação deste jogo deverão ser seguidas as indicações fornecidas anteriormente.

É importante que os alunos devem desenvolvam o programa de forma gradual, e que testem cada função desenvolvida. Por exemplo, ao criarem uma função para a inicialização do jogador devem também criar uma função para listar as propriedades do jogador, que será chamada na fase de testes. É importante que os alunos sigam uma abordagem deste género, pois sem a realização de testes metódicos irão deixando falhas no código produzido, e em pontos distintos, o que dificultará o desenvolvimento do programa. É importante que em cada momento todas as funções desenvolvidas anteriormente estejam correctas e devidamente testadas. Apenas a função em desenvolvimento poderá apresentar problemas. Assim o aluno saberá onde se encontra a falha que deverá corrigir.

Este programa deverá ser desenvolvido com a linguagem C no sistema operativo Linux.

Para este trabalho propõe-se a realização das seguintes tarefas ao longo das próximas semanas. As tarefas a executar não estão descritas na totalidade pelo que é da responsabilidade dos alunos a resolução de situações omissas. Os requisitos mínimos a desenvolver estão assinalados.

(Requisito Mínimo)

1. Implementação de uma versão simples do jogo. Para a implementação desta primeira versão do jogo deverá seguir a seguinte estrutura:

```
Inicialização do jogador
Inicialização dos objetos
Inicialização do local da aventura
Inicialização do monstro

Enquanto não for fim de jogo
{
      Movimentar monstro
      Descrever a localização do jogador
      Aceitar comando do jogador
      Movimentar jogador
      Se a localização do jogador e monstro forem iguais
            Lutar
}
Apresentar resultado final
```

O jogo de aventura (*ja*) deverá ser chamado através da linha de comando:

```
[lfbg@Merlim LuisGarcia]$ ja
```

2. Para testar devidamente o programa deve permitir que este seja chamado em modo *super user*. Para tal deve ser fornecido na linha de comando um código especial, seguido da energia, a localização inicial do jogador no local da aventura, e um objecto a transportar. Na linha de comando apresentada a seguir, o programa é chamado em modo *super user*, utilizando-se o código especial *1234*. A seguir a este código é fornecida a energia inicial do jogador: *5000*, a sala inicial: *10*, e o objeto transportado: *5*.

```
[lfbg@Merlim LuisGarcia]$ ja 1234 5000 10 5
```

   Neste modo o jogador também deverá visualizar a localização do monstro após cada jogada.

3. Inicialize as salas do local da aventura a partir de um ficheiro de texto. Os dados de cada sala deverão encontrar-se armazenados em linhas sucessivas do ficheiro, e terminam quando surge uma linha vazia. As linhas vazias irão separar a informação de cada sala. A primeira linha de cada sala deve conter as salas destino para cada uma das direcções (norte, sul, etc), o código de um objeto existente naquela sala, ou -1 caso não exista um objeto, e a indicação se o tesouro se encontra nessa sala. As linhas a seguir a esta informação consistem na descrição da sala. Inicialize também a partir de um ficheiro, a informação sobre os objetos que o jogador pode transportar.

4. Desenvolva um comando que permita gravar a situação de jogo que poderá ser retomada mais tarde.

5. Melhore ainda a interface do programa utilizando a biblioteca NCURSES para manipulação da consola. Este tema não é abordado nas aulas pelo que é da responsabilidade dos alunos a pesquisa deste tópico.

(Requisito Mínimo)

6. Para que as deslocações do jogador e do monstro no espaço da aventura se possam realizar de forma independente, os alunos deverão associar duas *threads* diferentes à execução de cada uma destas operações. A *thread* do jogador deve aceitar os comandos do utilizador e realizar a movimentação do jogador com base nestes comandos. A *thread* do monstro deve movimentar o monstro ao fim de um tempo aleatório. A *thread* principal (que criou as outras duas *threads*) deve verificar continuamente se o jogador e o monstro se encontram na mesma célula para que se possa iniciar um combate.

7. Identifique situações no jogo que requeiram uma sincronização entre as *threads*. Como exemplo, enquanto o jogador e monstro combatem, estes não devem ser movimentados pelas respetivas *threads*. Resolva estas questões de sincronização entre *threads* utilizando os mecanismos abordados nas aulas.

(Requisito Mínimo)

8. Desenvolva um relatório descrevendo as tarefas realizadas em cada uma das etapas do trabalho.

Avaliação

Os trabalhos serão classificados com uma das seguintes notas (esta nota será adicionada à média dos dois testes):

- -1 – o trabalho não cumpre os requisitos mínimos, ou o aluno não conseguiu demonstrar que desenvolveu o trabalho
- 0 – o trabalho cumpre os requisitos mínimos mas não a totalidade, e o aluno conseguiu demonstrar que desenvolveu o trabalho
- +1 - o trabalho cumpre todos os requisitos, incluindo os extras, e o aluno conseguiu demonstrar sem margem para dúvidas que desenvolveu o trabalho

Grupos de Trabalho

O trabalho deve ser desenvolvido por grupos com um máximo de dois elementos. Não são permitidas alterações nos elementos do grupo salvo em situações extraordinárias e devidamente justificadas.

Entrega do Trabalho

Os alunos devem entregar os scripts e um relatório que apresente a solução encontrada. No relatório devem ser apresentadas e explicadas as principais partes do código. Num anexo deste relatório devem ser fornecidos o programa desenvolvido. Para a entrega do trabalho os alunos devem compactar os vários ficheiros num único ficheiro (zip ou equivalente). O nome deste ficheiro deve conter a indicação TG2 (Trabalho de Grupo 2) e o número dos alunos que compõem o grupo. Por exemplo TG2_2000_3000.zip seria o Trabalho de Grupo 2 do grupo formado pelos alunos com os números 2000 e 3000. O trabalho deve ser entregue através do sistema *Moodle*. **Não serão consideradas soluções entregues por e-mail.**

Apresentação do Trabalho

Na apresentação os alunos devem demonstrar conhecer a totalidade do trabalho e estar aptos a realizar modificações ao código apresentado de acordo com as indicações do docente. Esta apresentação será realizada num horário especialmente marcado para o efeito.

Bom Trabalho
Luís Garcia