IPBeja
Instituto Politécnico de Beja

IPBeja Escola Superior de Tecnologia e Gestão

Jogo de Aventura

LOST IN PAST...

Disciplina: Sistemas Operativos
Docente: Luís Garcia

Autor: Miguel Rosa (Nº 6219)
Eng. Informática . Dezembro 2013

JOGO DE AVENTURA

LOST IN PAST...

Índice

Introdução

No âmbito da Disciplina de Sistemas Operativos, da Escola Superior de Tecnologia e Gestão (Instituto Politécnico de Beja), no Curso de Engenharia Informática foi proposto a realização de um trabalho prático (usando a linguagem de programação C[1]) onde o principal objetivo era programar um jogo de aventura do estilo RPG, usando como referência o primeiro jogo do género, o *Dungeons & Dragons*[2].

Como título, escolhi "*Lost in Past…*", pois o jogador é transportado para uma viagem no tempo longínquo onde a Terra era dividida por Reinos e as batalhas e conquistas do desconhecido eram constantes. A história do jogo remete o jogador para uma dessas cruzadas, onde, no meio do caminho, encontra um castelo abandonado e decide explora-lo. Reza a história que naquele castelo havia um tesouro por descobrir guardado por um monstro. O jogador para descobrir o tesouro guardado é auxiliado pelo computador (dando coordenadas) para se movimentar.

Com este relatório pretendo demonstrar e detalhar a maneira como foi desenvolvido o trabalho ao longo das semanas, explicando (e exemplificando, quando possível) com excertos do código.

[1] http://en.wikipedia.org/wiki/C_(programming_language)
[2] http://www.ddo.com

Modo Super-User

Este modo está "oculto", isto é, para ser utilizado é preciso de chama-lo como parâmetro na prompt[3] ou no terminal[4]. Esta funcionalidade permite ao jogador obter uma energia muito superior ao adversário, facilitando assim a vitória no jogo.

O código desta funcionalidade está assim definido:

```
if( (argc == 2) && (strcmp(argv[1],"su") == 0) )
{
	jogador.jogadorEnergia = JOGADOR_ENERGIA_SU_MODE;
}
```

O *argc* informa ao programa a quantidade de parâmetros inseridos pelo utilizador, enquanto que o argv[1] serve para comparar se o argumento inserido é igual ao "su". Se for o jogador fica com a energia superior. Caso contrário, fica com a energia estabelecida previamente[5].

[3] Linha de Comandos DOS
[4] Linha de comandos UNIX, caso o jogador tenha executado o jogo em Mac ou Linux
[5] Ver estrutura do jogador

Criação das Estruturas

Todos os elementos do jogo foram criados com simples estruturas de dados. A estrutura armazena informação relativo ao nome, energia, posições, etc.

Estrutura do Jogador

```
#define JOGADOR_MAX_NOME 20

(…)

struct Jogador{
	char jogadorNome[JOGADOR_MAX_NOME];
	int jogadorEnergia;
	int jogadorPosicao;
};
```

Estrutura do Adversário

```
struct Adversario{
	int adversarioEnergia;
	int adversarioPosicao;
};
```

Estrutura do Mapa

```
#define SISTEMA_MAPA_MAX_DESCRICAO 1000

struct Celula{
	 int norte;
	 int sul;
	 int este;
	 int oeste;
	 int tesouro;
	 char descricao[SISTEMA_MAPA_MAX_DESCRICAO]; //descreve a sala onde o
jogador esta
};
```

Funções Principais: Jogabilidade

Estas funções têm um papel fundamental no processamento do jogo. Com elas é definido o rumo que o jogo toma, tomando decisões e guiando o jogador.

Combate entre Jogador e Adversário

Esta função é responsável pelo confronto entre as personagens do jogo. Quando o jogador se encontra na mesma sala onde se encontra o adversário (monstro) é desencadeado uma luta. Durante essa luta o jogador pode usar 3 armas diferentes, sendo que o resultado final é variável (o jogador pode acertar no adversário, o adversário pode atacar o jogador, etc).

Devido ao código ser um pouco extenso, o mesmo pode ser observado nos anexos (a função tem o nome de "void *combatePersonagens(struct Jogador *jogador, struct Adversario *adversario)*").

Verificar o Vencedor

Esta função determina o vencedor do jogo, dependendo do resultado que o jogador obtém durante o jogo. O jogo termina quando o jogador derrota o adversário, ou quando conseguir obter o tesouro. Após a verificação, o jogo termina.

O código poderá ser visto em anexo, com o nome "*void verificaFimJogo(struct Jogador *jogador, struct Adversario *adversário)*".

Verificador da obtenção do tesouro

Nesta função, o objetivo é semelhante à função anterior, sendo que aqui a verificação é se o jogador obtém o tesouro. Caso tenha obtido (localização do jogador igual à localização do tesouro), o jogo para e o jogador sai vencedor. Caso contrário, o jogo continua a decorrer.

Funções Principais: Estrutura do jogo

As funções abaixo explicadas são essenciais para o bom funcionamento do jogo. Com elas, o jogo pode iniciar o mapa através de ficheiros, pode fazer o guardar jogo (e retomar mais tarde). Em ambos os casos, os ficheiros podem ser do tipo texto (.txt) ou em binário (.bin) e têm uma localização específica de modo a separar dos restantes ficheiros do jogo.

Abrir Mapa (ficheiro .txt e .bin)

Com esta opção o sistema pode iniciar o jogo baseado num ficheiro, ficheiro esse que pode ser de um de 2 tipos: Em formato .txt cujo o conteúdo pode ser lido em qualquer editor de texto, ou em formato .bin (binário), sendo que este último apenas pode ser lido pelo próprio jogo.

Guardar Jogo

Durante o jogo, o utilizador pode guardar o jogo para retomar posteriormente. Esta função guarda o estado e posição do jogador e adversário bem como as definições do mapa.

Carregar Jogo

Esta função faz o inverso da opção anterior, ou seja, permite ao jogador retomar o jogo do preciso momento que deixou. Quando a opção é selecionada, o jogador retoma o jogo com a energia que tinha, posição onde estava, por sua vez, o adversário também obtém as referências que tinha (energia e posição).

Funções Secundárias: Gráficos

Embora o jogo seja todo feito em modo texto, decidi criar um aspeto gráfico para embelezar o jogo de modo a torna-lo um pouco mais atrativo.

Gráfico(s) Menu principal

Para este menu, a interface é bastante simples, tendo um castelo como cenário com as opções disponíveis para o utilizador escolher entre um “Novo Jogo” e o “carregar Jogo” (continuar um jogo previamente gravado).

Gráfico(s) Banner Jogo

Esta função informa ao jogador o nome e energia que tem ao longo de todo o jogo.

Gráfico(s) Créditos Finais

Quando o jogador termina o jogo (independentemente do resultado que obtiver), surge um ecrã com o castelo como fundo, com as referências ao autor da elaboração do presente trabalho.

Testes de Desenvolvimento

Durante as semanas em que desenvolvi este trabalho, fui realizando pequenas funções de teste de modo a verificar se o desenvolvimento estava no caminho certo. Funções serviam como base de avanço para a próxima funcionalidade.

Estrutura do Jogador

Esta função serve para verificar a integridade da estrutura do jogador. Para verificar bastava realizar um output de toda a informação referente à estrutura do jogador.

Estrutura do Adversário

Esta função é em tudo idêntica à função anterior, exceto que esta lista os dados referentes ao adversário.

Estrutura do Mapa

Função que lista as definições do mapa (coordenadas e descrição das células – salas).

Conclusão

Com a conclusão deste trabalho prático consolidei (colocando em prática) os conhecimentos que foram adquiridos ao longo das aulas.

Com a continuação do desenvolvimento do trabalho foram surgindo as "tradicionais" perguntas do "*Como se faz?*"/"*Como atingir este (ou aquele) requisito*", mas com a persistência em querer fazer, com alguma pesquisa, os requisitos/objetivos foram atingidos. Este trabalho foi desafiante, não só pelo tema em si, como também a ideia de programar um jogo, coisa que é diferente daquilo que já tinha feito anteriormente.

Os objetivos propostos para a realização deste trabalho foram amplamente atingidos.

Webgrafia

http://www.cplusplus.com/forum/beginner/5404/

http://www.mtm.ufsc.br/~azeredo/cursoC/aulas/c790.html

http://crasseux.com/books/ctutorial/argc-and-argv.html

http://stackoverflow.com/questions/12320969/read-char-from-txt-file-in-c

http://stackoverflow.com/questions/11546177/how-to-read-lines-of-text-from-fileand-put-them-into-an-array

http://www.portugal-a-programar.pt/topic/8864-resolvido-texto-com-cor-em-c/

http://msdn.microsoft.com/en-us/library/ms682088.aspx

http://www.cplusplus.com/forum/general/81753/

http://www.vivaolinux.com.br/topico/C-C++/System(pause);/

http://stackoverflow.com/questions/15102976/how-to-clear-screen-from-simple-c-program

http://faq.cprogramming.com/cgi-bin/smartfaq.cgi#clear

http://www.inf.pucrs.br/~pinho/LaproI/Arquivos/ArquivosBinarios.htm

http://forum.zwame.pt/showthread.php?t=551594

http://stackoverflow.com/questions/4384309/reading-writing-a-structure-into-a-binary-file

Anexos

Código Completo

Devido à extensão do código, o mesmo foi armazenado no Google Code[6], podendo ser acedido através do endereço https://code.google.com/p/jogoaventurawin/.

[6] Sistema que permite armazenar código, dividindo o mesmo em projetos.