Aventura!

Gubi

19 de setembro de 2012

Sumário

1 Introdução

O objetivo do projeto é montar um sistema de execução de jogos do tipo *adventure*, como apresentado em classe. Neste tipo de jogo o personagem principal (aventureiro) executa ordens dadas pelo jogador e descreve o que aconteceu, sempre em forma de texto.

A estória acontece em um mundo virtual composto de lugares interligados por passagens, transferências mágicas, teletransporte, etc. Neste mundo existem vários objetos (virtuais, lógico) espalhados e que podem interagir com o aventureiro de diversas formas (veja a seção 2.1.3).

Existem alguns jogos disponíveis em diversos lugares:

- No emacs, execute `M-x dunnet`

- No linux, rode o programa `adventure`, que nas distribuições *Ubuntu* e *Debian* fica no pacote `bsdgames.deb`. Este jogo é uma adaptação do primeiro escrito neste estilo, o "Colossal Cave", em FORTRAN!
- Outra possibilidade com o linux é o `battlestar`, também em `bsdgames.deb`
- Procure jogos da *Infocom*, que marcou época com seus jogos de texto interativo. No linux existem vários pacotes para interpretar estes jogos. Experimente o `frotz` e o `gargoyle-free`.
- Experimente o engraçadíssimo *The Hitchhikers Guide to the Galaxy*. Passarei mais detalhes na sala de aula.

O projeto será desenvolvido em perl+java terá as seguintes componentes fundamentais:

- Carregador do mundo e gerador de código java (perl).
- *Model* — Biblioteca de manipulação do mundo (java).
- *View* — Interface com o usuário (java).
- *Controler* — Interpretador da linguagem (java).

Além disso, o grupo deverá montar uma estória, seguindo rigidamente a especificação adotada no curso. Estórias montadas por um grupo devem poder ser usadas por outro grupo sem a necessidade de uma adaptação.

2 Primeira fase: Reconhecimento da Especificação

O mundo onde ocorrerá o jogo será descrito de uma forma simplificada, de tal forma que seja muito fácil construir ou alterar uma aventura. O objetivo desta descrição é que uma pessoa não precise ter grandes conhecimentos de programação para inventar uma estória e, além disso, mundos grandes e com um alto número de objetos seja facilmente construído.

Por outro lado, como queremos que a aventura final seja um programa em java auto-contido, isto é, sem a necessidade de carregar um arquivo externo com a descrição[1], precisamos de um mecanismo de tradução entre a descrição e as classes java que a implementam.

Inicialmente vamos nos preocupar apenas em reconhecer a especificação e listar todos os elementos presentes na estória. O programa desta primeira fase permitirá uma familiarização com a hierarquia de classes e interfaces usada no projeto servirá de base para o resto do desenvolvimento.

[1] Claro que será permitida a carga de funções de biblioteca comuns a todas as aventuras, como as classes básicas do java, etc.

2.1 Estrutura interna

Antes de prosseguir, vou adiantar um pouco da estrutura interna a ser usa da na descrição do mundo e seus objetos. Mais adiante, após termos visto melhor a noção de objetos e herança, retornarei a este ponto e o aprofundarei.

O mundo é composto por **elemento**s, que por sua vez podem ser lugares ou objetos. Os elementos serão implementados por *objeto*s em java, usando classes e herança. Usarei *objeto* (enfatizado) para indicar instância de uma classe e **objeto** para um objeto dentro do mundo virtual. Para ajudar a fixar, **lugar** indicará uma região do mundo virtual.

Além dos **elemento**s, a descrição do jogo ainda contém **saída**s e **verbo**s. As **saída**s são conexões entre **lugar**es e os **verbo**s descrevem as ações possíveis.

A **objeto** e **lugar** são derivações de **elemento**, já que sua estrutura é parecida, mas existem algumas diferenças que estão discutidas nas próximas seções. Todo **elemento** possui algumas propriedades que o descrevem, ou a seu estado, e pode ser alvo de um conjunto de ações. Os **elemento**s devem conter um nome interno, um ou mais apelidos, uma descrição longa e uma descrição curta.

Os apelidos indicarão como o usuário irá se referenciar ao elemento. Por exemplo um *objeto* que descreve um gato pode ter nome "gt" e apelidos "gato", "felino" e "bichano". funcionar como apelido também.

Nesta fase trataremos os **lugar**es e os *objeto*s de forma independente, embora em java eles sejam relacionados. Assim, no que segue, haverá alguma redundância.

2.1.1 Elementos

Tanto **objeto**s como **lugar**es possuem as seguintes propriedades. O termo entre parênteses é o que será utilizado na especificação para indicar a propriedade.

- Nome — identificação interna do **elemento**, deve ser um nome único (representa a variável associada).
- Artigos (artigos) — uma lista de quatro palavras, indicando os artigos: definido direto, definido indireto, indefinido direto, indefinido indireto. Por exemplo: "o do um dum".
- Descrição longa (longa) — texto que descreve detalhadamente o **elemento**, apresentada quando o usuário pede para examinar o **objeto** ou **lugar**.
- Descrição curta (curta)— descrição abreviada, sem detalhes. É a descrição normalmente apresentada.
- Lista de objetos (contem) — lista dos nomes dos **objeto**s presentes no **lugar** ou contidos no **objeto**. Opcional.
- Atributos — são valores genéricos que permitem especificar melhor o estado, de acordo com as necessidades do jogo. Opcionais.

- Ação (acao) — **verbo** que tem significado especial neste **lugar** ou para este **objeto**. É opcional e pode haver mais de uma.
- Animação (animacao) — **verbo** especial que é chamado a cada iteração, permitindo animações. É opcional.

2.1.2 Lugares

Um **lugar** possui a seguinte propriedade adicional:

- Saída (saida) — conexão para outro **lugar**. As direções possíveis são pelos apelidos de cada saída (ver mais adiante).

2.1.3 Objetos

Além das propriedades dos **elemento**s, os **objeto**s possuem ainda:

- Adjetivos (adjetivos) — lista adicional com adjetivos que permitem especificar melhor o **objeto**.
- Invisível (invisivel) — indicação especial para **objeto**s que estão escondidos no início do jogo.

As açoes genéricas para **objeto**s são as seguintes:

- `examinar` — descreve o objeto.
- `pegar` — passa o objeto para o aventureiro, se possível (podem haver restrições quanto a número, tamanho ou peso que o aventureiro pode carregar).
- `largar` — o **objeto** precisa estar com o aventureiro. O **objeto** é colocado no lugar onde o aventureiro se encontra. Com argumentos adicionais, pode-se colocar um **objeto** dentro de outro, se possível.

Ações específicas para alguns objetos podem ser `destruir`, `esfregar`, `ligar`, etc. Procure ter uma implementação padrão para verbos mais gerais. Na especificação da estória, o usuário poderá definir novos verbos.

2.1.4 Saídas

As **saída**s ligam um **lugar** a outro. Uma **saída** é um caso especial de **objeto**, que possui duas propriedades especiais a mais:

- Destino (destino) — contém a referência para o **lugar** onde ela leva.
- Fechada (fechada) — indica que a **saída** está fechada.

2.1.5 Verbo

Um **verbo** é essencialmente uma função que atua sobre os **elemento**s. Veremos sua descrição nas próximas fases.

2.1.6 O aventureiro

O aventureiro é um caso muito especial de **objeto** animado. Ele pode conter outros **objeto**s, se os estiver carregando ou vestindo, e recebe atualizações diretamente do usuário.

Além disso, podem haver outras atualizações automáticas, como cansaço, fome e recuperação, se a especificação do jogo indicar.

2.2 Descrição

O arquivo de entrada para esta fase conterá a descrição do mundo de acordo com a gramática a seguir. Esta gramática não está completa, pois ainda não contém a definição das funções. Isso será feito mais tarde, mas não causará uma dificuldade adicional exagerada.

2.2.1 Gramática

Aqui está uma descrição simplificada da gramática.

ENTRADA	::=BLOCO | BLOCO (**:::**\n BLOCO)+
BLOCO	::=(OBJETO | OBJETOCLONE | SALA | ACAO | SAIDA)+
OBJETO	::=**Objeto** *Nome* (PROPRIEDADE)+
SALA	::=**Sala** *Nome* (PROPRIEDADE)+
NOME	::=*nome* | *nome*:LIST_SINOM
LIST_SINOM	::=*nome* | *nome list_sinom*
PROPRIEDADE	::=(CURTA | LONGA | ATRIB | ACAO | CONTEM | ANIMACAO | **invisivel**)
CURTA	::=**curta=** "*string*"
LONGA	::=**longa=** "*string_com_varias_linhas*"
ANIMACAO	::=**animacao=***nome*
ATRIB	::=*chave=valor*
ACAO	::=**acao=** NOME
CONTEM	::=**contem=** (*lista de nomes*)
FUNCAO	::=(**Acao**—**Funcao**) DEFINICAO_DE_FUNCAO
DEFINICAO_DE_FUNCAO	::=NOME (*codigo*)
SAIDA	::=**saida** NOME (PROPRIEDADE)+

Elementos escritos DESTA FORMA são símbolos não terminais, isto é, elementos abstratos da linguagem. Aqueles escritos em **negrito** são palavras reservadas. Finalmente, os escritos *desta forma* são os outros símbolos terminais.

Observação sobre as propriedades: algumas das propriedades são especiais e portanto têm seu próprio declarador (**curta**, **longa**, **animacao**, **contem**, etc). Outras propriedades são livremente definidas pelo usuário e são declaradas pelo seu primeiro uso. No futuro,

colocaremos todas as propriedades definidas assim um tipo de dado contêiner, como um *hash*. As outras propriedades serão traduzidas em campos específicos da estrutura.

Deve haver uma entrada especial, inicio, que contém a descrição inicial do jogo. Da mesma forma, o bloco atualiza contém o código que deve ser executado a cada iteração. Não se preocupe com o código dos **verbo**s nesta fase.

O tradutor não distingue maiúscula de minúscula para as palavras-chave.

2.2.2 Exemplo

Veja um exemplo de entrada:

```
Objeto vaca : vaca mimosa malhada
curta="Uma vaca malhada"
longa="É uma vaca malhada, com olhos tristes e com uma placa no pescoço
escrito ‘Mimosa’"
viva=1
animacao=AnimaVaca
com_leite=0
:::
Sala curral
curta="Este é o curral da fazenda"
longa="O curral tem algumas cocheiras e está surpreendentemente limpo
comparado com o resto da fazenda.  Com certeza, quem quer que cuide
daqui, gosta muito de leite."
saida=Ncurral
saida=Scurral
contem=(vaca, banquinho, corda, balde)
:::
Saida Ncurral : norte N estrada
curta="tem uma estrada ao norte"
longa="é uma estrada longa e tortuosa"
destino=estrada
:::
Saida Scurral : sul S
curta="uma porta ao sul"
longa="uma porta de madeira velha, caindo aos pedaços, a porta está entreaberta"
destino=pasto
:::
Objeto corda
curta="Uma corda enrolada"
longa="É apenas uma corda...."
invisivel
:::
```

```
Objeto banquinho : banco tamborete
curta="um banquinho de madeira"
longa="Um banquinho de madeira, com três pés, daqueles usados para
ordenhar"
:::
Objeto balde
curta="um balde"
longa="um balde de alumínio, meio amassado pelo longo uso"
vazio=sim
acao=virar
:::
acao virar : vire tombe
{
   "Uma corda caiu no chão!  Ela estava dentro do balde, mas não dava
    para ver";
    @corda:visivel=sim;
}
```

2.3 Programa da primeira fase

Como já foi dito, na primeira fase deverá ser entregue um programa que usa o arquivo gerado pelo programa `trad`, escrito em *perl*. O arquivo, que é escrito na saída padrão, deve ser gravado com o nome `Jogo.java`[2].

Seu grupo deverá acertar a classe `Motor.java`, que contém inicialmente apenas o esqueleto da aplicação.

Se a entrada estiver correta, o programa deverá imprimir na saída padrão os seguintes ítens:

1. A lista de todos os objetos, com suas descrições e propriedades, separadas por seções. Veja um exemplo de saída:

```
                    OBJETOS
============================================================
vaca
------------------------------------------------------------
Apelidos: mimosa, malhada
------------------------------------------------------------
Descrição curta:
    Uma vaca malhada
------------------------------------------------------------
Descrição longa:
```

[2]Se tiver coragem, estude o código do `trad` também.

```
  É uma vaca malhada, com olhos tristes e com uma placa
  no pescoço escrito 'Mimosa'
-------------------------------------------------------
Propriedades:
   Chave      Valor
   viva         1
   animacao   AnimaVaca
   com_leite     0
=======================================================
corda
-------------------------------------------------------
Descrição curta:
    Uma corda enrolada
-------------------------------------------------------
Descrição longa:
    É apenas uma corda....
-------------------------------------------------------
Propriedades:
   Chave      Valor
   visivel     nao
```

2. Uma lista similar dos lugares. As saídas devem estar claramente indicadas, assim como os objetos contidos em cada lugar.

3. Uma lista com os nomes de todos os objetos e lugares que possuem a propriedade "animacao".

A classe `Elemento` contém um atributo de classe (*static*) com a lista de todas as instâncias. É um *ArrayList* chamado `Mundo`.

Veja a seguir os métodos presentes no `Motor.java`:

1. `main` — Instancia um motor e executa o jogo. Aqui será colocada a varredura dos objetos presentes em `Elemento.Mundo`.

2. `Titulo` — É apenas um método que imprime as duas *strings* passadas como argumento. A primeira é o nome do Lugar e a segunda sua descrição.

3. `Mostra` — Imprime *srting* passada como argumento. Será usada para fazer a saída. No futuro este método será trocado por outro que interage com a interface gráfica.

Ignore os outros métodos nesta fase. Eles podem permanecer vazios.