Trabalho para Disciplina de Construção de Compiladores 2
2017
DC-UFSCar

DRAMATURGO

Linguagem script para Visual Novels

Integrantes:

Guilherme de Almeida Henna RA: 619574
Miguel de Souza Tosta RA: 619698
Ruan Willer Silveira Xavier RA: 619817

Proposta da linguagem

A linguagem Dramaturgo foi desenvolvida com o objetivo de facilitar o desenvolvimento de Visual Novels, se baseando em roteiros de peças de teatro para a escrita de seus scripts. É uma linguagem que é interpretada para gerar o jogo funcional.

Visual Novels (**VN**) são jogos populares no Japão focados em narrativa, de visual simples e estático, geralmente com a sua arte se assemelhando a de animações japonesas. Como se fossem livros interativos, a jogabilidade apenas permite ao jogador que avance os diálogos, seguindo com a narrativa. O que torna **VNs** interessantes são as decisões que devem ser tomadas ao longo do jogo, que afetam a forma como o mesmo terminará. Por conta disso, é comum que existam diversas rotas a se concluir, cada uma com um desenvolvimento da história diferente.

Mais informações e exemplos desse tipo de jogo podem ser encontrados na seguinte página da Wikipédia: https://en.wikipedia.org/wiki/Visual_novel

A ideia se originou de uma linguagem já existente para o desenvolvimento de Visual Novels, chamada Ren'py. Ela é uma versão mais simplificada e abstrata de Python, utilizada para escrever os scripts do jogo. Ren'py também é o nome da engine que se utiliza de bibliotecas como Pygame para a criação de janelas e gráficos.

O problema com Ren'py é que ela ainda se assemelha bastante a Python em alguns quesitos, e pensando nisso, decidimos fazer uma versão mais parecida com roteiros de peça de teatro, de forma que seja mais fácil e, possivelmente, mais atrativo a criação de jogos do gênero. Mas, claro, isso também se torna a desvantagem da nossa linguagem, já que Ren'py permite a inserção e programação em código Python, enquanto a nossa é uma linguagem de especificação que abstrai lógicas mais complexas.

Como rodar

Pré-requisitos

Locais de pastas

Na pasta "dramaturgo-executavel", você encontrará o arquivo dramaturgo.jar e algumas pastas contendo exemplos.

Para rodar basta navegar por um terminal até o local onde foi colocado o dramaturgo.jar e digitar:

```
java -jar dramaturgo.jar <script_dramaturgo>
```

Onde o <script_dramaturgo> é o nome do arquivo no qual foi escrito o script, sendo de extensão **.txt**. No caso dos exemplos, basta colocar exemplos/<nomeDoExemplo>.txt.

Como jogar

Para jogar o jogo gerado, basta pressionar ENTER para que a narrativa se desenvolva, e ao se deparar com uma escolha, selecionar com o mouse a opção que deseja.

Note que é normal o jogo começar com a tela preta, já que é esperado que sempre se pressione ENTER para que uma ação do script seja executada no jogo, portanto incluindo a ação de mudar o cenário. Isso é uma coisa a ser modificada e melhorada em mudanças futuras.

Gramática e Um Exemplo

A gramática da linguagem é simples:

(**NOTA:** arquivo Dramaturgo.g4 está localizado no pacote Dramaturgo)

```
grammar Dramaturgo;

programa : cabecalho corpo 'FIM';

cabecalho : titulo personagens cenarios recursos;

corpo : cena mais_cenas;
```

```
titulo : 'TITULO:' CADEIA;

personagens : 'PERSONAGENS:' PERSONAGEM mais_personagens '.';

cenarios : 'CENARIOS:' IDENT mais_cenarios '.';

recursos : 'RECURSOS:' recurso mais_recursos '.';

cena : '-' nome_cena '-' continua narrativa pulo;

nome_cena : 'CENA' NUM;

mais_cenas : cena mais_cenas
           |
           ;

mais_personagens : ',' PERSONAGEM mais_personagens
                 |
                 ;

mais_cenarios : ',' IDENT mais_cenarios
              |
              ;

recurso : estado_personagem '=' CADEIA
        | IDENT '=' CADEIA
        | IDENT '(' tipo_som ')' '=' CADEIA
        ;

mais_recursos : ',' recurso mais_recursos
              |
              ;

continua : '(' 'continua' PERSONAGEM continua_outros ')'
         |
         ;

continua_outros : 'e' PERSONAGEM continua_outros
                |
                ;

narrativa : entrelinhas narrativa
          | dialogo narrativa
          | escolha narrativa
          |
          ;

pulo : 'PULE' NUM
        | 'PULE FIM'
```

```
        |
        ;

estado_personagem : PERSONAGEM '[' IDENT ']';

tipo_som : 'musica' | 'som';

entrelinhas : novo_cenario
            | nova_entrada
            | mudanca_emocao
            | nova_saida
            | nova_musica
            | novo_som;

dialogo : CADEIA
        | PERSONAGEM '-' CADEIA;

escolha : 'ESCOLHA:' CADEIA '-' 'va' 'para' nome_cena mais_escolhas '.';

mais_escolhas : ',' CADEIA '-' 'va' 'para' nome_cena mais_escolhas
              |
              ;

novo_cenario : '(' 'cenario' IDENT ')';
nova_entrada : '(' 'entra' estado_personagem outra_entrada ')';
mudanca_emocao: '(' estado_personagem outra_mudanca ')';
nova_saida : '(' 'sai' PERSONAGEM outra_saida ')';
nova_musica : '(' 'musica' IDENT ')';
novo_som : '(' 'som' IDENT ')';

outra_entrada : 'e' estado_personagem outra_entrada
              |
              ;

outra_mudanca : 'e' estado_personagem outra_mudanca
              |
              ;

outra_saida : 'e' PERSONAGEM outra_saida
            |
           ;

CADEIA: '"' ~('\r' | '\n' | '"')* '"';
PERSONAGEM : [A-Z][a-zA-Z_0-9]*;
IDENT : [a-z_][a-zA-Z_0-9]*;
NUM : [0-9]+;
WS: [ \n\t\r]+ -> skip;
```

Um exemplo de um script Dramaturgo:

```
TITULO: "Pedindo informacoes"

PERSONAGENS: Julio, Senhor.

CENARIOS: praca.

RECURSOS: praca = "praca.jpg",
          Julio[normal] = "julio_normal.png",
          Senhor[normal] = "senhor_normal.png".

- CENA 1 -

(cenario praca)
(entra Julio[normal])
Julio - "Acho que estou perdido...Aquele senhor deve saber onde fica o ponto de onibus. Vou perguntar!"
(sai Julio)


- CENA 2 -

(cenario praca)
(entra Julio[normal])
Julio - "Boa tarde senhor! Poderia me dizer onde fica o ponto de onibus mais proximo?"
(entra Senhor[normal])
Senhor - "Boa tarde. Depende, pra que lugar voce vai?"

ESCOLHA: "Irei para o shopping" - va para CENA 4, "Irei para a rodoviaria" - va para CENA 3.


- CENA 3 -

Senhor - "Pois bem. O onibus para a rodoviaria passara em alguns minutos no ponto ao norte daqui."
Julio - "Muito obrigado! Tenha um bom dia."
Senhor - "Por nada."
"Julio entao se dirige ao ponto."
(sai Julio)
(sai Senhor)


- CENA 4 -

(cenario praca)
(entra Julio[normal])
(entra Senhor[normal])
```

```
Senhor - "Pois bem. O onibus para o shopping passara em uma hora no ponto ao sul daqui."
Julio - "Muito obrigado! Tenha um bom dia."
Senhor - "Por nada."
"Julio entao se dirige ao ponto."
(sai Julio)
(sai Senhor)

FIM.
```

Sobre a implementação

Tecnologias utilizadas

Dramaturgo foi desenvolvida em Java, utilizando as seguintes bibliotecas:

- **ANTLR v.4.7** - uma biblioteca que facilita a construção de compiladores, necessitando apenas da gramática da sua linguagem para criar classes para a implementação de analisadores léxicos, sintáticos, semânticos e eventuais geradores de código e/ou interpretadores.
- **LibGDX** - uma biblioteca gráfica para o desenvolvimento de jogos em java. Ela é construída em cima de LWJGL e abstrai a construção de jogos multiplaforma (mais especificamente, para as plataformas Desktop, Android, iOS e HTML). Junto dela, foram utilizadas algumas ferramentas adicionais da LibGDX: **Hiero** para a geração de fontes bitmap e **scene2d** para alguns elementos de UI.

Projeto

A divisão das classes implementadas é feita da seguinte forma:

- **Ação** - Classes que controlam a mudança de estados da narrativa em execução. Elas alteram os personagens que estão em cena, as músicas e sons que tocam, o fluxo das cenas, as falas a serem exibidas, por exemplo.
- **Dramaturgo** - Classes relacionadas ao interpretador da linguagem Dramaturgo. É aqui que estarão as classes geradas pelo ANTLR, o arquivo dramaturgo.g4 contendo a gramática da linguagem e as classes dos analisadores sintático e semântico, assim como a classe do interpretador.
- **Main** - a classe main é a classe que cuida de toda a parte de renderização e tratamento da parte gráfica do jogo, se comunica com a classe narrativa para adquirir os objetos atuais em cena, como personagens, falas, músicas e escolhas.
- **Modelos** - Classes relacionadas ao interpretador da linguagem Dramaturgo. É aqui que estarão as classes geradas pelo ANTLR, o arquivo dramaturgo.g4 contendo a

gramática da linguagem e as classes dos analisadores sintático e semântico, assim como a classe do interpretador.

Classes

Ação

Acao.java

Classe pai de qual todas as outras ações herdam. Ela é utilizada na classe Narrativa para a realizar cada ação de uma sequência de ações de uma cena. Isso é feito através de polimorfismo chamando acao.executaAcao(), um método que todas as classes ação herdeiras implementam.

AcaoContinuarEmCena.java

Classe filha de Acao.java. É uma ação que recebe uma lista de nomes de personagens que deverão continuar em cena, e a partir desses nomes, ela descobre os índices dos personagens em cena na lista personagensEmCena (pertence à Narrativa) que deverão continuar em cena. "Continuar em cena" é necessário, caso seja desejado que uma textura (imagem) de uma personagem não deixe de ser renderizada ao trocar de cena. Afinal, a cada cena nova, a lista de personagens em cena é zerada, implicando que a próxima cena não começará renderizando ninguém.

AcaoEntrarPersonagem.java

Classe filha de Acao.java. É uma ação que recebe um nome de personagem e, a partir dele, descobre seu índice na lista total de personagens da Narrativa. Isso para que esse índice seja adicionada à lista personagensEmCena (pertence à Narrativa), efetivando a entrada de uma personagem em cena.

AcaoEscolha.java

Classe filha de Acao.java. É uma ação que recebe uma lista de escolhas, às quais através de algumas funções da Narrativa, são incluídas na lista de escolhas. Essa lista de escolhas da narrativa está sempre sendo vigiada pela classe Main (responsável pelos gráficos), de modo que assim que a lista é populada, entende-se que a ação é a de escolha, e os botões com as escolhas possíveis são renderizados.

AcaoFala.java

Classe filha de Acao.java. É uma ação que recebe uma fala por vez e popula o campo fala da Narrativa, para que a classe Main (responsável pelos gráficos) renderize a fala.

AcaoMudarCenario.java

Classe filha de Acao.java. É uma ação que recebe uma string contendo o nome do cenário a ser utilizado como fundo. A partir desse nome, o campo responsável por segurar o cenário atual na Narrativa é trocado.

AcaoMudarEmocaoPersonagem.java

Classe filha de Acao.java. É uma ação que recebe o nome e a emoção da personagem. Efetivamente modifica a textura (imagem) de uma personagem atualmente em cena, sem modificar sua posição.

AcaoPulo.java

Classe filha de Acao.java. É uma ação que recebe um inteiro contendo a cena a qual o pulo deve ser realizado. A partir dele, modifica a cena atual de acordo e zera o campo ação atual da Narrativa, inicializando assim uma essa cena.

AcaoSairPersonagem.java

Classe filha de Acao.java. Parecida com entrar personagem, com a diferença de remover da lista de personagens em cena.

AcaoTocarMusica.java

Classe filha de Acao.java. É uma ação que recebe o nome de um recurso do tipo música, e a partir dele, instancia um objeto Music da LibGDX.

AcaoTocarSom.java

Classe filha de Acao.java. É uma ação que recebe o nome de um recurso do tipo som, e a partir dele, instancia um objeto Sound da LibGDX.

Dramaturgo

DramaturgoMain.java

Classe responsável por realizar a compilação do script (roteiro). Possui apenas um método "run()", no qual é realizado o parse do arquivo .txt contendo o script, verificando assim possíveis erros sintáticos. Caso não tenha ocorrido nenhum erro, é feita a análise semântica a partir da árvore de análise sintática criada anteriormente. Novamente, caso não ocorra

erros, é realizada a interpretação do script, onde um objeto "controlador" é povoado com dados provenientes do script como: personagens, cenários, recursos, ações a serem tomadas, etc. Ao fim, caso nenhum erro ocorra, o objeto "controlador" povoado é retornado. Se em qualquer etapa da compilação um erro ocorrer, ela é cancelada e o erro é exibido.

DramaturgoSemantico.java

Classe responsável pela análise semântica a partir da árvore gerada na análise sintática. Herda a classe "DramaturgoBaseVisitor" que possui métodos para visita dos nós da árvore.
Possui um atributo do tipo "Map" para servir como tabela de símbolos, atributos do tipo "List" para salvar as tentativas de pulo para outra cena e escolha levando a outra cena, e atributos para ajudar a manter a sequência das cenas e o controle do número de erros semânticos encontrados.
Com o auxílio da tabela de símbolos são feitas verificações de múltiplas declarações do mesmo personagem ou cenário e, de mesma forma, seus recursos. A existência de elementos a serem inseridos ou retirados de cena, controle de sequência das cenas e saltos e desvios para outras cenas.
Ao final da análise, o número de erros é retornado para verificação na classe "DramaturgoMain".

DramaturgoInterpretador.java

Classe responsável pela interpretação do código. Herda a classe "DramaturgoBaseVisitor" que possui métodos para visita dos nós da árvore. Essa classe possui um objeto Narrativa, o qual popula de acordo com as ações feitas por código. As funções são repetitivas por todas manterem um mesmo formato: a partir da regra visitada, pegar os campos necessários (normalmente nomes de personagens, falas, entrada e saída) e popular as listas da Narrativa contendo todos os recursos ou criar as devidas ações que serão realizadas durante as cenas. Para essa última, o interpretador preenche uma lista de cenas, também preenchendo cada cena com sua lista de ações. Esse conjunto de cenas depois é atribuído à Narrativa.

Main

Main.java

A classe Main é responsável por realizar todas as ações relacionadas à biblioteca LibGDX, e portanto, cuida da parte gráfica. Ela se comunica com a classe Narrativa, buscando os objetos atuais em cena (cenário, personagens, música e sons, falas, escolhas) e renderiza de acordo. Existem algumas limitações nos jogos que podem ser feitos com a linguagem

pela forma como algumas coisas foram implementadas na Main, como por exemplo, o número de escolhas possíveis ser limitado a cinco.

Modelos

Narrativa.java

Dentre as classes presentes no pacote de Modelos, a que vale ser mencionada é a Narrativa. Sendo uma das classes principais para o funcionamento do jogo, ela é populada durante a interpretação de um script Dramaturgo pelo DramaturgoInterpretador. Coisas como todas as personagens presentes no jogo, todos os cenários, sons e músicas são armazenados em listas dentro de Narrativa, assim como listas de personagens em cena, cenário e músicas atuais, conjunto de todas as cenas ordenadas, a cena atual e a ação atual. É dela que a classe Main receberá o estado atual do jogo e agirá de acordo. É nela, também, que serão aplicadas as classes derivadas de Ação, modificando o estado dos elementos em cena.

Sobre exemplos

Existem poucos exemplos, mas que mostram o tipo de resposta que se pode ter ao interpretar um script Dramaturgo. Os exemplos estão localizados na pasta exemplos, e dentro dessa, possuem exemplos para casos de sucesso, casos com erro sintático e casos com erro semântico.
É esperado que em erros sintáticos, mensagens já pré-definidas em inglês serão apresentadas, enquanto os erros semânticos estarão em português. Cada erro apresentará a linha onde ele ocorreu, seguido de uma mensagem que pode lhe indicar o que está faltando, ou o que fez de errado.

Referências

http://www.cobra.pages.nom.br/ecp-teatroscript.html

http://www.revistaea.org/artigo.php?idartigo=227

http://portaldoprofessor.mec.gov.br/fichaTecnicaAula.html?aula=33379

https://www.renpy.org/

https://www.renpy.org/wiki/renpy/doc/tutorials/Quickstart

https://gist.github.com/anonymous/2cbaa960c4baf95a45fd9dde2794aa53

https://gist.github.com/anonymous/70fb7cc05b066a5ded0c5dc43cf42d6a

https://gist.github.com/anonymous/144ba6eeef00e088dde84bfe018fe052