Estudo de Caso do Uso de Linguagens de Script em Jogos

Bruno G. Oliveira[1], Heitor Adão Jr. [1], Vinicius Schlup[1], Anita M. R. Fernandes[1], Rudimar L. S. Dazzi[1]

[1]Grupo de Inteligência Aplicada – Universidade do Vale do Itajaí (UNIVALI)
Itajaí – SC – Brasil

{bruno_oliveira,heitoradao,shp,anita.fernandes,rudimar}@univali.br

Abstract. *This paper introduces the concept of the script languages, presenting characteristics of the most used languages. Also presents an example of an application that integrates script language Ruby and C++, using an embedded Ruby interpreter for use in games.*

Resumo. *Este artigo introduz o conceito de linguagens de script, apresentando características das linguagens mais usadas. Também apresenta um exemplo de uma aplicação que integra a linguagem de script Ruby e C++, usando um interpretador Ruby embarcado para uso em jogos.*

1. Introdução

As linguagens de script são linguagens de programação desenvolvidas para um aplicativo específico cuja grande diferença para as linguagens de programação convencionais, como C++ ou Pascal, é que o código fonte é interpretado pelo programa em tempo de execução e não em tempo de compilação.

Segundo Valente (2005), a característica mais importante das linguagens de script é que elas permitem a qualquer pessoa alterar o comportamento de uma parte do programa ou a forma como o script atinge seu objetivo na aplicação num nível de abstração alto, sem precisar re-compilar o código principal, dando uma maior flexibilidade no desenvolvimento da aplicação.

Os scripts podem ser executados em uma máquina virtual ou em um interpretador embarcado numa aplicação. Quando executado em uma máquina virtual, só podem ser usadas as instruções que foram programadas na máquina virtual, por isso existem limitações. Quando executado em um interpretador embarcado, pode-se criar novos conjuntos de instruções para serem usadas no script, ou seja, praticamente não possui limitações [Valente 2005], por estas características optou-se pelo uso de um interpretador embarcado.

Existem várias linguagens de script disponíveis, as mais utilizadas são Ruby, Lua e Python, sendo Ruby e Lua as mais utilizadas na implementação de jogos. É apresentado a seguir, uma breve descrição sobre elas.

- Ruby: Possui instruções nativas para tratamento de exceções, é orientada a objeto e permite heranças simples e possui um sistema de gerenciamento automático de memória com coletor de lixo [Matsumoto 2005].

- Lua: Possui um gerenciamento automático de memória com coletor de lixo. É uma linguagem estruturada [Lua 2006].
- Python: É uma linguagem de programação dinâmica orientada a objetos, oferece suporte para outras linguagens e pode ser aprendida em poucos dias [Python 2006].

2. A Linguagem Ruby

Foi escolhida a linguagem Ruby pelo fato de que dentre as linguagens estudas foi a que melhor supriu as necessidades do projeto e apresentou a sintaxe mais amigável.

A linguagem Ruby possui instruções nativas para multithread (processo paralelo) que são independentes de plataforma e seu coletor de lixo libera automaticamente toda a memória que foi utilizada durante a execução do programa.

Para poder usar um interpretador embarcado com a linguagem de script Ruby, duas funções são necessárias: `ruby_init()` e `ruby_finalize()`. A função `ruby_init()` inicializa o interpretador Ruby e a função `ruby_finalize()` deve ser chamada quando a biblioteca não for mais utilizada. Essa função destrói a única instância do interpretador e libera toda memória ocupada pela mesma.

A seguir tem-se a descrição das funções necessárias e mais utilizadas para integrar um interpretador Ruby embarcado em uma aplicação em C++:

- `rb_define_global_function()`: permite que uma função definida em C seja chamada dentro de um script.
- `rb_define_variable()`: permite que uma variável definida em C seja acessada dentro de um script.
- `rb_load_file()`: responsável pelo carregamento de um arquivo de script.
- `ruby_exec()`: responsável pela execução do script carregado pela função descrita acima.

3. Estudo de Caso

Este trabalho deriva do RPGEDU, um projeto de pesquisa que tem por objetivo o desenvolvimento de um jogo educativo 3D utilizando a engine Ogre3D para renderização de imagens, a biblioteca OPAL para testes de colisão, a bliblioteca OpenAL para o sonoplastia, a linguagem C++ no desenvolvimento do motor e a linguagem de script Ruby para respostas de eventos. O jogo a ser desenvolvido no projeto é um RPG (*Role Playing Game*) didático para alunos de 5ª a 8ª séries do ensino fundamental. Dentro do ambiente do jogo existirão NPCs (*Non Player Character*) que são personagens controlados pelo computador que interagirão com o usuário constantemente e eventualmente apresentará atividades pedagógicas a serem resolvidas pelo usuário. Estas atividades são pré-programadas em scripts.

Na aplicação real, RPG Didático, existirão várias funções em C++ que manipularão a entrada do usuário e a saída para o vídeo e som, que poderão ser chamadas dentro de scripts. Os scripts são chamados de acordo com as interações do

usuário no sistema, eles chamam as funções para a manipulação de entrada e saída, de acordo com o seu objetivo. Um script será executado toda vez que o usuário interagir com um *non player character* (NPC), apresentando uma atividade pedagógica. Pode-se ver nesse caso que o uso de uma linguagem de script permite programar ou modificar as atividades pedagógicas sem precisar conhecer os detalhes da implementação da *engine* do jogo, trazendo muitos benefícios em desenvolvimento de jogos.

O código a seguir tem como finalidade apenas ilustrar um sistema básico de resposta de eventos, bastante usado na maioria dos jogos, mostrando como chamar uma função escrita em C++, em um script. Este código foi escrito exclusivamente para este artigo e é um exemplo bastante simples devido às limitações de espaço.

A seguir tem-se um exemplo de uma aplicação em C++ que apresenta a criação de um interpretador embarcado.

```
1 #include "Ruby.h"
2 #include <iostream>
3 using namespace std;
4 VALUE Funcao(...) {
5     cout << "\nDigite numero: ";
6     int numero;
7     cin >> numero;
8     return INT2NUM(numero);
9 }
10 int main() {
11    ruby_init();
12    rb_define_global_function("c_funcao", Funcao, 0);
13    rb_load_file("main.rb");
14    ruby_exec();
15    ruby_finalize();
16    return 0;
17 }
```

Na linha 12 registrou-se uma nova função em ruby (`c_funcao`). Quando essa função for chamada em um script, a função em C++ (`Funcao`) será executada. Na linha 13 foi carregado o arquivo que contém o script. A linha 14 faz com que o conjunto de instruções contidas no script sejam executadas. Finalmente a linha 15 destrói o interpretador.

Supondo que o script tenha o seguinte conteúdo:

```
1 num = c_funcao
2 printf "o numero que voce digitou foi: " + num.to_s()
```

O comportamento do programa seria o seguinte: o programa inicializa o interpretador, carrega e executa o script e no script a função `c_funcao` é chamada. Essa função é implementada em C++ e é uma função de entrada de dados. Ela pede um número para o usuário e retorna esse número para o script. A segunda linha do script imprime na tela o número que foi digitado. Depois que o script termina, o *instruction*

pointer retorna para a próxima instrução em C++, que finaliza a biblioteca e termina o programa.

4. Conclusão

Através deste estudo foi demonstrado que o uso de linguagens de script é eficiente no desenvolvimento de jogos, por sua flexibilidade. Também foi apresentado um programa com o objetivo de mostrar como funciona a integração entre C++ e a linguagem Ruby. No RPGEDU o Ruby é responsável por toda a parte de respostas a eventos e criação de atividades pedagógicas.

Futuramente será construído um software para geração de scripts, abstraindo ainda mais a criação das atividades para o professor através de uma interface simplificada de construção de atividades pedagógicas.

Referências Bibliográficas

LUA (2006) “A Linguagem de Programação”, http://www.lua.org/portugues.html, Julho.

MATSUMOTO, Y (2005) “RUBY: What’s Ruby”, http://www.ruby-lang.org/en/20020101.html , Julho.

PYTHON (2006) “The Python Programming Language”, http://www.python.org/, Julho.

VALENTE, L. (2006) “Guff: Um Framework para Desenvolvimento de Jogos”, (Mestrado), Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, Niterói.