

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE
COMPUTAÇÃO

FELIPE FISCHER COMERLATO
LEONARDO EICH

Grupo Equipe 7
Projeto Space Invaders Utilizando Lua

Relatório apresentado como requisito parcial para
a obtenção de conceito na Disciplina de Modelos
de Linguagens de Programação

Prof. Dr. Lucas Mello Schnorr
Orientador

Porto Alegre
2018

SUMÁRIO

1 INTRODUÇÃO	3
2 O PROBLEMA	4
3 VISÃO GERAL DA LINGUAGEM	5
3.1 Modelo Funcional	6
3.2 Modelo Orientado a Objetos.....	6
4 RECURSOS FUNCIONAIS	7
4.1 Elementos Imutáveis e Funções Puras	7
4.2 Funções Anônimas	7
4.3 Currying.....	7
4.4 Pattern Matching	7
4.5 Funções de Ordem Superior	7
4.6 Funções de Ordem Maior Fornecidas Pela Linguagem	7
4.7 Funções como Elementos de Primeira Ordem	7
4.8 Recursão.....	7
5 RECURSOS DE ORIENTAÇÃO À OBJETOS	8
5.1 Classes	8
5.2 Encapsulamento e Proteção dos Atributos	8
5.3 Construtores	8
5.4 Destrutores.....	8
5.5 Espaços de Nomes Diferenciados.....	8
5.6 Mecanismos de Herança.....	8
5.7 Polimorfismo por Inclusão	8
5.8 Polimorfismo Paramétrico	8
5.9 Polimorfismo por Sobrecarga	8
5.10 Delegates	8
6 CONCLUSÃO FINAL.....	9
REFERÊNCIAS.....	10

1 INTRODUÇÃO

Este trabalho tem como objetivo o estudo de uma linguagem de programação moderna com características híbridas contextualizando os conceitos vistos em aula ao longo do semestre e, por fim, analisar e avaliar diferentes linguagens de programação, seguindo os critérios vistos em aula.

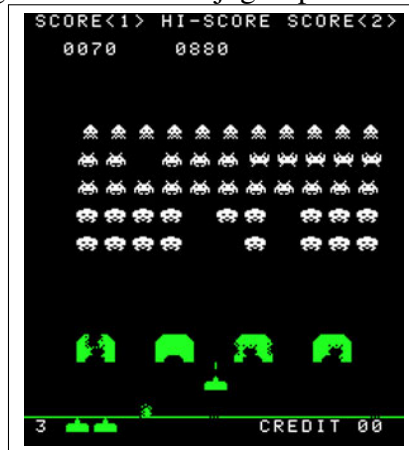
A tarefa principal do trabalho consiste em experimentar e comparar as características e funcionalidades orientadas a objeto e funcionais da linguagem de programação escolhida. De posse de uma linguagem, é necessário escolher um problema a ser solucionado com ela. O problema será, então, implementado duas vezes na mesma linguagem: uma delas usando somente Orientação a Objetos e a outra usando somente características funcionais.

O desenvolvimento do trabalho encontra-se em <https://github.com/felipefcomerlato/mlp_equipe7_2018-2>.

2 O PROBLEMA

O problema a ser resolvido neste trabalho é o desenvolvimento do jogo Space Invaders. O jogo expõe o jogador como uma espaçonave que deve destruir as espaçonaves inimigas que querem invadir o planeta do jogador. Na medida que elas avançam na tela (de cima para baixo), o jogador guia sua espaçonave horizontalmente e efetua disparos para destruir todas as ondas de inimigos que se seguem, como visto na Figura 2.1

Figura 2.1: Tela do jogo Space Invaders



Fonte: <http://g1.globo.com/tecnologia/noticia/2011/10/homem-quebra-recorde-mundial-em-space-invaders-jogo-de-1978.html>

3 VISÃO GERAL DA LINGUAGEM

A linguagem escolhida para o trabalho foi extbfLua. Lua é uma linguagem de script de multiparadigma, pequena, reflexiva e leve, projetada para expandir aplicações em geral, por ser uma linguagem extensível (que une partes de um programa feitas em mais de uma linguagem), para prototipagem e para ser embarcada em softwares complexos, como jogos. Assemelha-se com Python, Ruby e Icon, entre outras.

Lua foi criada por um time de desenvolvedores do Tecgraf da PUC-Rio, a princípio, para ser usada em um projeto da Petrobras. Devido à sua eficiência, clareza e facilidade de aprendizado, passou a ser usada em diversos ramos da programação, como no desenvolvimento de jogos (a Blizzard Entertainment, por exemplo, usou a linguagem no jogo World of Warcraft), controle de robôs, processamento de texto, etc. Também é frequentemente usada como uma linguagem de propósito geral.

Uma das características principais da linguagem é sua única estrutura de dados: tabelas - que podem ser usadas para representar arrays comuns, sequências, tabelas de símbolos, conjuntos, registros, grafos, árvores, etc.

Lua é uma linguagem distribuída e pode ser compilada em qualquer plataforma que tenha um compilador da linguagem C. Lua pode ser executada em ambientes Unix e Windows, em dispositivos móveis (Android, iOS, Windows Phone, etc), em microprocessadores embarcados e etc.

Incluir Lua numa aplicação não aumenta quase nada o seu tamanho. O pacote de Lua 5.3.5, contendo o código fonte e a documentação, ocupa 297K comprimido e 1.2M descompactado. O fonte contém cerca de 24000 linhas de C. No Linux de 64 bits, o interpretador Lua contendo todas as bibliotecas padrões de Lua ocupa 247K e a biblioteca Lua ocupa 421K (ABOUT, 2018).

Lua possui interpretação dinâmica, ou seja, a linguagem é capaz de executar trechos de código criados dinamicamente, no mesmo ambiente de execução do programa. Como exemplos dessa facilidade temos a função *loadstring* em Lua e a função *eval* em Scheme/Lisp e Perl. Lua possui também tipagem dinâmica forte, ou seja, a linguagem faz verificação de tipos em tempo de execução do programa. Linguagens com tipagem dinâmica em geral não possuem declarações de tipos no código e não fazem verificação de tipos em tempo de compilação. Tipagem forte significa que a linguagem jamais aplica uma operação a um tipo incorreto. Além disso, Lua conta com gerência automática de memória dinâmica ("coleta de lixo"), por tanto não precisamos gerenciar memória ex-

plicitamente no nosso programa; em especial, não há necessidade de um comando para liberar memória após seu uso (UMA. . . , 2018).

3.1 Modelo Funcional

Lua oferece suporte ao modelo de programação funcional mas nos proporciona algumas dificuldades, como a de copiar tabelas, por exemplo, já que tal estrutura de dados atribuída a uma variável é apenas uma referência. Tal dificuldade contraria uma das principais características do próprio paradigma funcional: a de usar funções puras, criando novas estruturas ao invés de alterar uma estrutura original. Por outro lado, Lua nos permite representar funções anônimas através de blocos de código chamados de trechos, característico no modelo funcional. (MANUAL. . . , 2018)

3.2 Modelo Orientado a Objetos

O suporte para a programação no paradigma orientado a objetos utilizando Lua é muito vasto. É possível encontrar inúmeros tutoriais para a solução de diversos problemas na linguagem fazendo uso deste paradigma. Como não vimos diversos pré-requisitos necessários para a implementação deste trabalho, não podemos analisar com mais propriedade o desempenho da linguagem Lua neste modelo de programação.

4 RECURSOS FUNCIONAIS

4.1 Elementos Imutáveis e Funções Puras

4.2 Funções Anônimas

4.3 Currying

4.4 Pattern Matching

4.5 Funções de Ordem Superior

4.6 Funções de Ordem Maior Fornecidas Pela Linguagem

4.7 Funções como Elementos de Primeira Ordem

4.8 Recursão

5 RECURSOS DE ORIENTAÇÃO À OBJETOS

5.1 Classes

5.2 Encapsulamento e Proteção dos Atributos

5.3 Construtores

5.4 Destrutores

5.5 Espaços de Nomes Diferenciados

5.6 Mecanismos de Herança

5.7 Polimorfismo por Inclusão

5.8 Polimorfismo Paramétrico

5.9 Polimorfismo por Sobrecarga

5.10 Delegates

6 CONCLUSÃO FINAL

REFERÊNCIAS

ABOUT. 2018. Available from Internet: <<https://www.lua.org/about.html>>.

MANUAL de Referência de Lua 5.2. 2018. Available from Internet: <<https://www.lua.org/manual/5.2/pt/manual.html#2.6%20%E2%80%93Co-rotinas>>.

UMA Introdução à Programação em Lua. 2018. Available from Internet: <<https://www.lua.org/doc/jai2009.pdf>>.