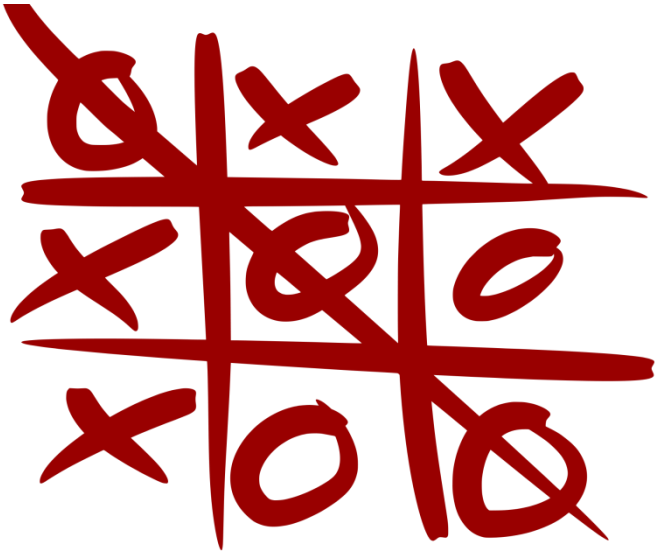
Linguagens de Programação

Trabalho 2015/2016

15/06/16

UNIVERSIDADE DE ÉVORA



Professor:
Salvador Abreu

Realizado por:
João Calhau - 31621
José Pimenta - 31677

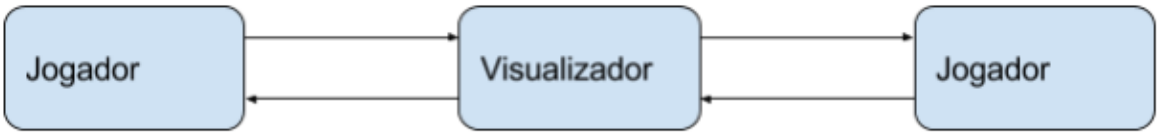
Índice

Introdução

Este trabalho enquadra-se na disciplina de Linguagens de Programação e vamos abordar uma linguagem escolhida por nós, que neste caso foi Lua. Com a linguagem que escolhemos temos então de criar 1 jogo do galo que funciona com 2 jogadores e 1 visualizador que sirva de comunicação indireta entre os dois jogadores para a realização de um jogo entre eles.
Iremos assim tentar dar o nosso melhor, e iremos tentar utilizar os métodos que achemos mais correctos ou propícios à boa evolução do trabalho e que no final se concretize o que nos é pedido.

Funcionamento e Esquema

O jogo do galo que realizámos para esta cadeira é composto por 3 ficheiros:
O ficheiro jogador que contém os métodos para enviar mensagens ao visualizador para poder realizar as jogadas (funcionando como cliente), o ficheiro do visualizador que recebe as ligações dos jogadores e realiza as jogadas que estes pedem para se realizar. Existe ainda o ficheiro JogoDoGalo que contém o jogo do galo própriamente dito e as regras para o seu bom desenvolvimento.



Como instalar e correr (LINUX)

PARA INSTALAR:

sudo apt-get install luarocks
sudo luarocks install luasocket

NOTA: para o 1º comando instala o “luarocks” e instala a versão 5.1 do lua.
Para o 2º comando instala a biblioteca de sockets necessária para lua para poder correr o programa.

PARA CORRER:
Basta fazer “lua visualisador.lua” num terminal e de seguida abrir 2 terminais com os 2 jogadores com “lua jogador.lua” .

Como Jogar

Quando se corre o visualizador este espera que se conectem os 2 jogadores, e só começa o ciclo while para o jogo quando estiverem 2 jogadores conectados.

Quando os 2 jogadores estiverem conectados existe uma ordem pela qual jogar, e têm de escrever os seguintes comandos pela devida ordem de modo a funcionar (podendo escreve “nome *nomeQuePretende* ” para poder inserir um nome para o jogador):

JOGADOR1: “tamanho”
JOGADOR2: “tamanho”
JOGADOR1: “comeca”
JOGADOR2: “comeca”

Seguidamente os jogadores jogam alternadamente, sendo o JOGADOR1 a fazer a 1ª jogada (sendo o JOGADOR1 o 1º jogador a conectar-se ao visualizador).

Para fazer as jogadas, os jogadores têm de escrever: jogo x/y sendo 1<= x <=3 e 1<= y <= 3, e tendo o tabuleiro do tipo:

1 2 3
4 5 6
7 8 9

Correspondendo:

1 -> 1/1
2 -> 1/2
3 -> 1/3
4 -> 2/1
5 -> 2/2
6 -> 2/3
7 -> 3/1
8 -> 3/2
9 -> 3/3

Durante o jogo quando for a vez do jogador jogar, este pode desistir escrevendo “desisto”.
No final do jogo, quando 1 jogador ganhar, o programa do jogador envia “ganhei” para o visualizador e este avisa o outro jogador que ele perdeu.
Quando o tabuleiro se encontrar cheio, ambos os jogadores empatam e o visualizador envia para ambos que empataram com comando “empatamos”.

Algumas notas sobre código

Para a realização dos comandos e por exemplo, com o comando “jogo x/y”, necessitámos de criar uma função chamada “split”, que recebe como argumentos 2 strings, sendo uma string que queremos fazer split e a segunda é o simbolo pela qual queremos separar a string. A função retorna uma tabela com a string dividida.

Modo de utilização do split:
O split pode ser utilizado de duas maneiras:

```
s = "joga 1/2"

(Igualando o split a uma variável nova)
ss = s:split(" ")
print(ss[1])
print(ss[2])

(Ou inicializando uma tabela, passando-a  como argumento)
sss = {}
s:split(" ", sss)
print(sss[1])
print(sss[2])
```

A 2ª maneira funciona como um apontador para a tabela, tal como em C.

NOTA: O código encontra-se comentado e para evitar ambiguidade, não foi colocado aqui no relatório.

Conclusão

Após a realização deste trabalho, ficámos a conhecer a linguagem LUA, aprendendo algumas coisas de novo em programação em outras linguagens. Para a realização utilizámos 1 biblioteca de sockets, visto o LUA não ter sockets e por esse motivo e outros foi-nos um pouco complicado realizar o trabalho visto não haver muitas informações acerca de sockets em LUA.
O trabalho se funcionar com os comandos indicados anteriormente funciona sempre correctamente pois foi assim definido que seria o funcionamento durante o jogo. Caso não corra os comandos como indicado o trabalho não funciona, pois não foi feito a pensar numa má utilização e sim na indicada.
Concluimos assim que o trabalho podia ter corrido um pouco melhor, mas devido a vários factores externos (outros trabalhos, testes, exames), não nos foi possível realizar o trabalho que talvez idealizámos, mas mesmo assim ficámos contentes por compreender melhor mais uma linguagem de programação e fazer com que a nossa adaptação a outras linguagens seja agora muito mais fácil.