PROFESSOR DE MATEMÁTICA ONLINE
Revista eletrônica da Sociedade Brasileira de Matemática

PMO v.2, n.1, 2014
ISSN: 2319-023X
https://doi.org/10.21711/2319023x2014/pmo22

Relato de uma experiência com o software KTurtle na simulação de problemas envolvendo probabilidade

Leonardo Barichello

Resumo

Este texto relata uma experiência de criação de simulações para problemas envolvendo probabilidades com a ajuda do software KTurtle em uma turma reduzida de alunos de Ensino Médio. O objetivo deste relato é salientar as potencialidades deste software neste contexto específico e para introduzir estudantes ao mundo da programação de computadores e ao pensamento computacional.

Palavras-chave: probabilidade; programação; simulação; KTurtle; Matemática.

Abstract

This paper reports on a simulation experience for problems involving probabilities with the help of KTurtle software in a reduced class of high school students. The purpose of this report is to highlight the potential of this software in this specific context and to introduce students to the world of computer programming and computational thinking.

Keywords: probability; programming; simulation; KTurtle; Mathematics.

1. Introdução

Em 2011, eu era responsável por uma série de aulas opcionais direcionadas a estudantes com interesse em Matemática em uma escola particular no interior do Estado de São Paulo. O número de alunos que frequentava essas aulas oscilava entre 5 e 10 e todos cursavam o primeiro ou segundo anos do Ensino Médio. Em uma dessas aulas, me deparei com uma situação inusitada ao discutir com os alunos o problema de Monty Hall (vide [4]).

Problema de Monty Hall: Em um programa de televisão, o candidato é solicitado a escolher uma entre três portas fechadas. Atrás de uma delas há um prêmio, mais precisamente um carro, e atrás de cada uma das outras duas há um bode. Depois de o candidato escolher a porta que deseja, mas ainda antes de abri-la, o apresentador do programa, que sabe onde estão os bodes, abre uma das portas que não foram escolhidas e mostra que há um bode atrás dela. Então, o apresentador pergunta ao candidato se ele deseja trocar a porta que ele havia escolhido pela outra porta que

SBM SOCIEDADE BRASILEIRA DE MATEMÁTICA

permanece fechada. Qual deve ser a decisão do candidato visando maximizar sua chance de ganhar o prêmio?

A situação inusitada começou a se configurar quando os alunos, mesmo depois de acompanharem a demonstração formal de que a troca de porta é a melhor decisão, continuaram descrentes da "veracidade prática" da conclusão. No intuito de resolver o impasse, acessei rapidamente a Internet, encontrei um site e simulei algumas jogadas, mostrando que a proporção observada se aproximava daquela indicada pelos cálculos teóricos. Mas nesse ponto eu fui novamente surpreendido: os alunos começaram a questionar a confiabilidade do simulador.

Esse questionamento, que até poderia ser fundamentado em dúvidas quanto a real aleatoriedade do simulador ou a maneira como este conduzia o experimento "por trás" da interface, parecia se basear apenas na impressão pessoal de que a resposta correta deveria ser de que a probabilidade de ganhar não muda ao trocar de porta.

Felizmente, a aula terminou e eu ganhei alguns dias para pensar em como resolver o impasse antes da próxima aula. A solução que encontrei foi desenvolver com eles um simulador para o problema.

2. KTurtle

O software escolhido para isso foi o KTurtle[1] que, segundo os próprios criadores: ,

> tem o objetivo de tornar a programação de computadores o mais fácil e acessível quanto for possível, e portanto pode ser utilizada para ensinar crianças o básico de matemática, geometria e... programação. A linguagem de programação utilizada no KTurtle é vagamente baseada no Logo. (http://edu.kde.org/kturtle, tradução própria)

Como a descrição acima sugere, o software KTurtle é um dos tantos descendentes do software Logo, desenvolvido no final da década de 1960 no MIT com o intuito de apresentar o universo da programação de computadores para estudantes ainda em fase de alfabetização[2].

Além disso, o KTurtle é um software livre e gratuito que já vem instalado por padrão na distribuição oficial Linux do MEC (o Linux Educacional), pode ser instalado em qualquer distribuição Linux a partir de repositórios básicos e pode ser instalado no Windows através do KDE Windows Initiative (http://windows.kde.org/).

Apesar de não tratar diretamente de conteúdos matemáticos, como softwares de Geometria Dinâmica ou de Álgebra Simbólica, tanto o Logo quanto os seus descendentes estão diretamente conectados com essa área do conhecimento por alguns motivos: 1) os comandos mais básicos se referem à movimentação de um objeto na tela (especificamente, uma tartaruga virtual) e essa movimentação depende de parâmetros como distância e direção, 2) as linguagens por trás desses softwares, assim como toda linguagem de programação, suportam o tratamento aritmético de variáveis numéricas e 3) o processo de entendimento dos problemas e programação e elaboração de uma solução se assemelham, em diversos aspectos, com o processo de resolução em problemas de Matemática.

[1] Versão 0.8.1 no sistema operacional Kubuntu11.04. Essa mesma versão roda em Windows com apenas pequenas diferenças no visual da interface.

[2] Ainda hoje o MIT desenvolve iniciativas dessa natureza, sendo que a mais recente é o software Scratch (www.scratch.mit.edu).

Isto posto, coloca-se como objetivo deste texto apresentar e discutir as simulações que podem ser desenvolvidas nesse tipo de software, especificamente no KTurtle, para problemas envolvendo probabilidades. Porém, para atingir esse objetivo, será necessário apresentar o funcionamento geral do software.

Certamente, leitores familiarizados com programação de computadores enxergarão ao final deste texto muito além do que foi escrito, mas esperamos que aqueles que ainda não possuem experiências dessa natureza compreendam as ideias principais e sejam capazes de ir além, se houver interesse.

3. A interface do software

Na figura 1, temos a interface principal do KTurtle. A região central, chamada de Tela, é onde a tartaruga se move e executa as ações programadas pelo usuário. Essas ações são descritas através de comandos que devem ser digitados no Editor (lado esquerdo da tela) e, para executá-las basta clicar no botão Executar mais à esquerda da barra de ferramentas[3]. Na parte direita da tela temos o Inspetor, que traz algumas informações adicionais sobre o código enquanto ele é executado. Na parte superior, na barra de menus há opções básicas como Salvar e Abrir documentos. Duas opções que valem a pena mencionar são: a) "Obter mais exemplos..." (no menu Arquivo), que dá acesso a um banco on-line de códigos que podem ser baixados diretamente no software e b) "Linguagem dos scripts" (no menu Configurações), que permite mudar o idioma dos comandos do software (neste texto usaremos os comandos em Português do Brasil).

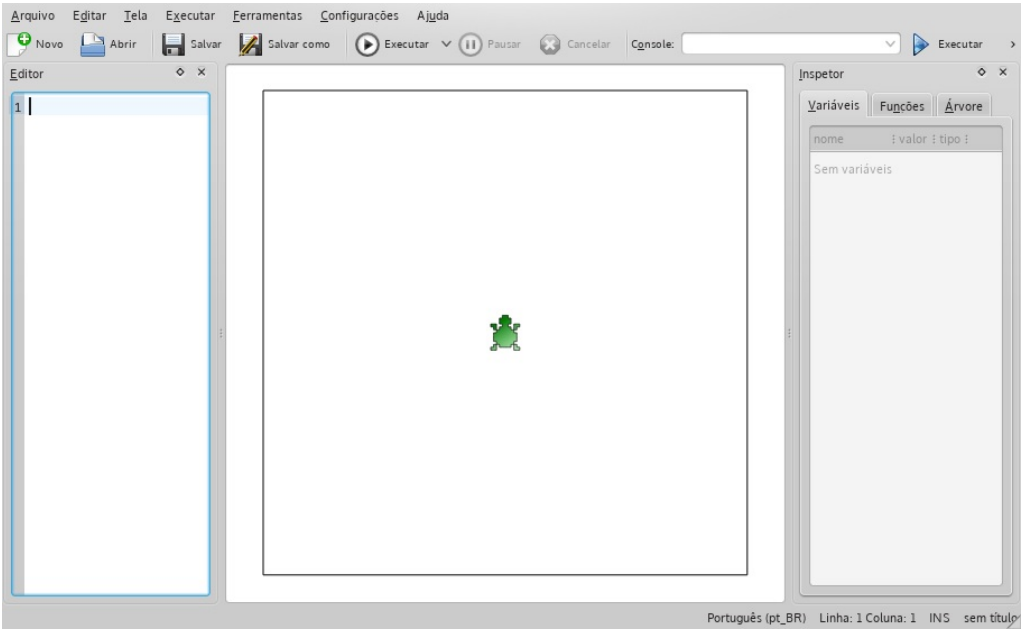

Figura 1: Interface do software KTurtle.

4. Funcionamento geral: comandos e rotinas de programação

[3]O botão Executar à direita da tela executa comandos isolados digitados no Console logo a sua esquerda.

Nessa seção, veremos alguns comandos com o intuito de apresentar o funcionamento geral do KTurtle e as estruturas que serão utilizadas nas simulações.

Os primeiros comandos que se aprende ao utilizar um software como o KTurtle são os de movimentação: `parafrente`, `paratrás`, `paradireita` e `paraesquerda`. Esses comandos exigem um parâmetro numérico ao serem acionados: no caso dos dois primeiros, o parâmetro se refere à distância virtual que a tartaruga deve percorrer e, no caso dos dois últimos, ao ângulo que ela deve girar. Por exemplo, a sequência de comandos a seguir gera o resultado mostrado logo ao lado.

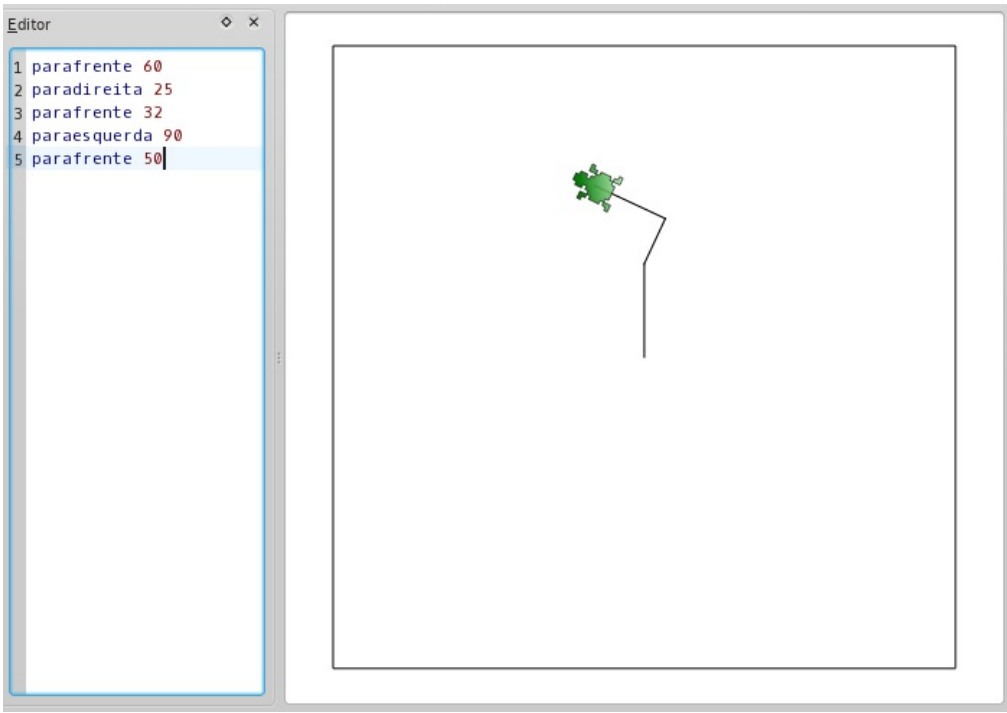

Figura 2: Exemplo de código e o resultado desenhado pela tartaruga.

Ao clicar no botão Executar, a tartaruga realiza toda a sequência de ações determinada pelos comandos digitados no Editor partindo do ponto da Tela em que está no momento. A cada movimento, ela deixa um rastro atrás de si.

Se o usuário clicar novamente no botão Executar, a tartaruga repetirá o mesmo traçado, mas a partir da posição atual. Para "reiniciar" a Tela, você pode usar o comando `apague`.

Com os comandos anteriores já podemos desenhar um quadrado, basta girar a tartaruga 90 graus e deslocá-la a mesma distância quatro vezes. Esse mesmo desenho pode ser feito com um conjunto mais enxuto de comandos, como mostrado abaixo:

```
1 repita 4 {
2 parafrente 100
3 paradireita 90
4 }
```

Figura 3: Código que também resulta em um quadrado de lado 100 unidades.

Nesse código, utilizamos o comando `repita` que, como o próprio nome sugere, faz com que a tartaruga repita um determinado número de vezes um conjunto de comandos (delimitado por chaves). Trata-se de um comando de repetição, que é uma das estruturas básicas de qualquer linguagem de programação.

Mas podemos ir ainda mais além utilizando uma variável que nos permitirá usar a mesma rotina de comandos para desenhar qualquer polígono regular. Para tanto, cria-se uma variável chamada `$nlados` (toda variável deve ter seu nome iniciado por `$`) e usaremos essa variável para determinar o número de repetições e o ângulo de giro da tartaruga.

```
1 $nlados = 6
2 repita $nlados {
3 parafrente 100
4 paradireita 360/$nlados
5 }
```

Figura 4: Código que resulta em um hexágono regular de lado 100 unidades.

Além das adaptações mencionadas anteriormente, foi acrescentada uma linha antes do comando `repita`. Nessa linha, atribuímos um valor à variável `$nlados`. Com isso, quando o comando `repita` for executado, a variável `$nlados` terá valor igual a 6 e a tartaruga desenhará um hexágono.

Como a rotina anterior é bastante versátil, podemos querer aproveitá-la em várias partes de uma mesma sequência de comandos. Para isso, podemos fazer a tartaruga memorizar esse conjunto de comandos, atribuindo um nome a ele para que possamos pedir a ela que os execute sempre que quisermos. Isso pode ser feito graças ao comando `aprenda`, da seguinte maneira:

```
1 aprenda poligonoregular {
2 repita $nlados {
3 parafrente 100
4 paradireita 360/$nlados
5 }
6 }
```

Figura 5: Um exemplo com o comando `aprenda`.

Na primeira linha do código anterior, estamos dizendo para a tartaruga “aprender” os comandos que aparecem logo em seguida (entre chaves) e executá-los sempre que digitarmos o comando `poligonoregular`. O que foi feito é equivalente à criação de uma função em linguagens de programação convencionais. O polígono resultante dependerá do valor da variável `$nlados`.

Se executarmos apenas os comandos anteriores, a tartaruga não realizará ação alguma na tela, apenas aprenderá o que foi descrito. Para que ela execute-os é necessário primeiramente definir o

valor da variável e depois chamar a função criada.

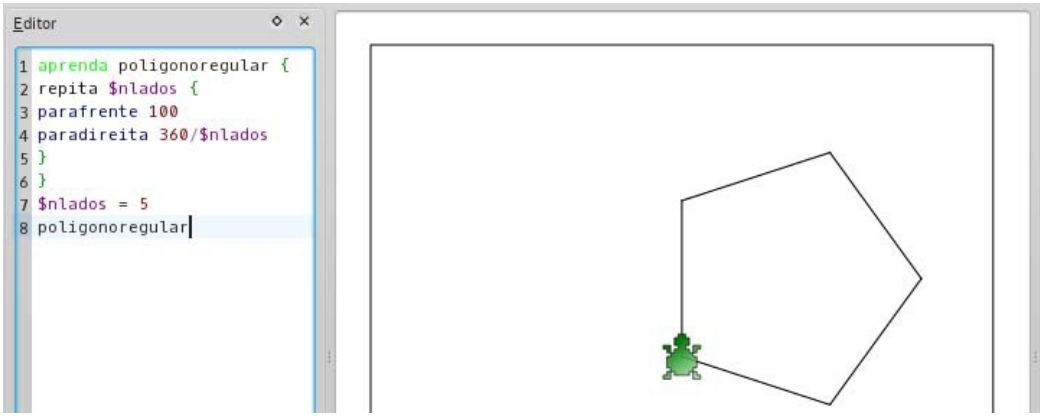

Figura 6: Resultado da função poligonoregular ao ser acionada com a variável `$nlados` igual a 5.

5. KTurtle e a probabilidade

A chave para criar simulações envolvendo probabilidade no KTurtle é o comando `aleatório` que gera um número aleatório[4] com 5 ordens decimais seguindo uma distribuição de probabilidades homogênea no intervalo dado (maior do que 1 e menor do 10, no exemplo abaixo).

Esse número pode ser armazenado em uma variável (`$numerosorteado`, no exemplo anterior) e depois mostrado na tela com o comando `mostre`.

```
1 $numerosorteado = aleatório 1, 10
2 mostre $numerosorteado
```

Figura 7: Rotina simples com o comando `aleatório`.

A partir desse comando, é possível criar códigos que simulem o lançamento de uma moeda ou de um dado de 6 faces e então usar esses elementos para construir simulações de situações mais complexas, como as que discutiremos adiante.

6. Lançamentos de moeda e de dado

Comecemos com um código que simula o lançamento de uma moeda 10.000 vezes[5] e depois mostre na tela a quantidade de caras e de coroas obtidas.

[4] Na verdade, o número gerado por esse comando é pseudo-aleatório, mas foge ao escopo deste artigo discutir essa diferença.

[5] Conta-se que um matemático feito prisioneiro de guerra na II Guerra Mundial realizou esse experimento enquanto estava preso obtendo como resultado 5067 ocorrências de uma face e 4933 da outra.

```
1  $ncaras = 0
2  $ncoroas = 0
3  repita 10000 {
4  $n = arredonda (aleatório 0, 1)
5  se $n==1 {
6     $ncaras = $ncaras+1
7     }
8  senão {
9     $ncoroas = $ncoroas+1
10    }
11 }
12 mostre "Sairam " + $ncaras + " caras e " + $ncoroas + " coroas"
```

Figura 8: Lançamento de uma moeda 10000 vezes.

Vamos analisar linha a linha deste código, pois sua estrutura irá se repetir nas simulações seguintes.

Nas duas primeiras linhas foram criadas variáveis que servirão para contar o número de ocorrências de cada face da moeda. Por isso ambas começam com valor igual a 0.

A terceira linha não traz novidade, mas a quarta usa simultaneamente o comando `aleatório` e o comando `arredonda`. O uso de parênteses não é necessário, mas evidencia a interpretação dessa linha: primeiro será gerado um número aleatório entre 0 e 1 e depois esse valor será arredondado (resultando em 0 ou 1) e armazenado na variável `$n`.

Da linha 5 até a 10 foi utilizado o comando `se` (e seu o complemento `senão`). Esse comando avalia se uma condição (`$n==1`) é verdadeira e, caso seja, executa o comando seguinte (linha 6).

Sobre a condição, note que foi utilizado um "duplo igual". Isso é necessário porque o "igual simples" significa atribuição de valor a uma variável. Também é permitido utilizar os símbolos $>, <, >=, <=$ e $!=$ que significam, respectivamente, maior, menor, maior ou igual que, menor ou igual que e diferente.

O efeito do comando da linha 6 é armazenar na variável `$ncaras` o seu próprio conteúdo acrescido de uma unidade, ou seja, estamos usando esse comando para contar quantas vezes a condição do comando se foi satisfeita, o que pode ser interpretado como "saiu cara no lançamento da moeda".

A linha 8 traz o comando `senão`, que complementa o comando `se`, ou seja, o KTurtle executará o comando da linha 10 apenas se a condição da linha 5 não foi satisfeita, que pode ser interpretado como "saiu coroa no lançamento da moeda".

A chave da linha 11 encerra o conjunto de comandos que serão repetidos 10.000 vezes e o comando da linha 12 apenas escreve uma mensagem na tela com as informações coletadas ao final das repetições.

Antes que você execute esse código, uma recomendação: devido ao grande número de repetições, sugerimos que você execute-os na maior velocidade possível. Para isso, clique na setinha para baixo logo ao lado do botão Executar e selecione a opção "Velocidade máxima (sem realce e inspetor)". Caso você já tenha iniciado a execução do código, o botão Cancelar interrompe a execução do código.

7. O problema do controle de natalidade

Esse problema foi extraído da apostila utilizada no Programa de Iniciação Científica Júnior da OBMEP [3] e foi escolhido para compor este texto pela simplicidade na montagem da simulação e pelo desafio da questão proposta. ,

> A China tem um sério problema de controle de população. Várias políticas foram propostas (e algumas colocadas em efeito) visando proibir as famílias de terem mais de um filho. Algumas dessas políticas, no entanto, tiveram consequências trágicas. Por exemplo, muitas famílias de camponeses abandonaram suas filhas recém-nascidas, para terem uma outra chance de ter um filho do sexo masculino. Por essa razão, leis menos restritivas foram consideradas. Uma das leis propostas foi a de que as famílias teriam o direito a um segundo (e último) filho, caso o primeiro fosse do sexo feminino. Deseja-se saber que consequências isso traria para a composição da população, a longo prazo. Haveria uma maior proporção de mulheres? De homens?
>
> **(a)** Com auxílio de uma moeda, simule a prole de um conjunto de 10 famílias (jogue a moeda; se obtiver cara, é um menino, e a família para por aí; se der coroa, é uma menina; jogue a moeda mais uma vez e veja se o segundo filho é menino ou menina).
>
> **(b)** Reúna os resultados obtidos pelos integrantes do grupo e produza estatísticas mostrando o número médio de crianças por família, a proporção de meninos e meninas na população e a proporção de famílias que têm um filho homem. O que esses resultados sugerem?
>
> **(c)** Qual é a probabilidade de que uma família tenha um filho do sexo masculino? Qual o número médio de filhos por família? Dentre todas as crianças nascidas, qual é a proporção de meninos e meninas?

O próprio enunciado sugere que os estudantes façam algumas simulações e depois reúnam seus resultados, com o intuito de aumentar a quantidade de casos analisados, para fazer uma análise inicial do problema e só depois calculem as probabilidades teóricas envolvidas.

Uma análise inicial pode nos levar à conclusão de que o número de meninas será maior, pois uma família que tenha uma filha do sexo feminino pode querer ter um segundo filho que, por sua vez, pode também ser do sexo feminino. Por outro lado, essa política é adotada há certo tempo na China e não se fala em problemas na proporção de homens e mulheres no país. Então, qual será a conclusão correta?

Os cálculos teóricos das probabilidades envolvidas podem ser encontrados nas páginas finais de [4]. Vamos discutir aqui como fazer a simulação deste problema no KTurtle.

Antes de escrever a rotina de comandos, vamos tentar descrever a sua estrutura:

1. Cada repetição deverá simular uma família. Digamos que 1.000 no total;
2. Para cada família será necessário fazer uma jogada de moeda para definir o sexo do primeiro filho, se for masculino a família terminou, senão, é necessário um segundo lançamento para definir o sexo do segundo filho. Como os casos masculino e feminino devem ser tratados de maneira diferente, usaremos o comando `se…senão`;

3. Como o nosso interesse é na proporção de homens e mulheres no país, utilizaremos duas variáveis para contar o número de bebês de cada sexo.

Com isso, um possível código para realizar a simulação é o que segue.

```
1  vápara 0, 0
2  $nHomens = 0
3  $nMulheres = 0
4  repita 1000 {
5  $lancamento = arredonda (aleatório 0,1)
6  se ($lancamento==1) {
7    mostre "H"
8    $nHomens = $nHomens+1
9  }
10 senão {
11   mostre "M"
12   $nMulheres = $nMulheres+1
13   $lancamento = arredonda (aleatório 0,1)
14   se ($lancamento==1) { #sim
15     mostre "   H"
16    $nHomens = $nHomens+1
17   }
18   senão { #mulher
19     mostre "    M"
20     $nMulheres = $nMulheres+1
21   }
22 }
23 paratrás 10
24 }
25 centralize
26 mostre $nHomens+  " homens e " + $nMulheres + " mulheres"
```

Figura 9: Código para simular o problema do controle de natalidade.

A estrutura geral do código é muito próxima da estrutura utilizada no lançamento da moeda, mas com alterações adequando-o ao problema proposto.

O primeiro comando posiciona a tartaruga no canto superior esquerdo da tela. O comando `se` da linha 6 (juntamente com o `senão` da linha 10) é o que analisa o sexo do primeiro filho, enquanto que o da linha 14 (juntamente com o `senão` da linha 18) analisa o do segundo filho, quando ele ocorrer.

As linhas 7, 11, 15 e 19 escrevem na tela o sexo de cada filho. Nas duas últimas foram incluídos alguns espaços em branco antes da letra H ou M para que o texto não ficasse sobreposto ao do primeiro filho. Na linha 23, o comando `paratrás` serve apenas para deslocar a tartaruga para trás ao final de cada família, também para que não haja sobreposição de texto.

Por fim, os comandos das duas últimas linhas servem para centralizar a tartaruga na tela (devido ao número de famílias simuladas ela deve ficar fora da tela) e mostrar os resultados obtidos.

Com o código pronto, é possível fazer diversas simulações que permitam uma análise baseada em um volume considerável de dados. No meu caso, já na primeira simulação, os números sugerem que a proporção entre o número de homens e de mulheres deve ficar próxima de 1 (750 homens e 753 mulheres), ou seja, o equilíbrio entre os sexos não deve ser afetado pela política proposta pelo governo.

O que se espera, e que foi observado com os meus alunos, é que essa análise sirva como abordagem

inicial ao problema proposto e reforce a credibilidade do resultado que o professor pode obter depois usando o conceito de probabilidade.

8. O desenvolvimento das atividades em sala de aula

Como mencionado na Introdução deste texto, as aulas em que utilizei o KTurtle eram opcionais e destinadas a alunos do Ensino Médio que se interessavam por Matemática. Essas aulas eram oferecidas semanalmente em contraturno e tinham duração de 1h30min.

Na aula seguinte ao acontecimento relatado na Introdução, comecei o trabalho com os alunos tendo o objetivo de fazer com que eles desenvolvessem um simulador para a discussão do problema de Monty Hall. Como a escola não dispunha de laboratório de informática, alguns alunos traziam seus notebooks, de modo que era possível trabalharem, no máximo, em duplas.

Bastou uma única aula para que os comandos básicos de movimentação, repetição, condicional e de criação de funções e variáveis fossem assimilados. Nessa aula, propus problemas simples como o desenho de polígonos regulares e a criação de ladrilhos com estes.

Na aula seguinte, propus algumas atividades ainda explorando essas estruturas básicas de programação, mas fazendo com os alunos utilizassem mais alguns comandos diretamente relacionados com a Matemática, como as funções trigonométricas e as suas inversas, com o intuito de calcular distâncias e posições para o bom encaixe de polígonos, por exemplo.

Na terceira aula, discutimos e implementamos algoritmos numéricos, como a conversão de um número na base decimal para a base 2 e o cálculo do MDC entre dois números dados, ainda com o objetivo de fixar as estruturas básicas de programação.

Na aula seguinte, retomamos o problema de Monty Hall e começamos a montar o simulador. Essa parte da atividade ocupou duas aulas duplas, porém, o código resultante não realizava apenas as simulações, mas também exibia desenhos ilustrando os resultados (portas, bodes e prêmio) e registrava os resultados obtidos.

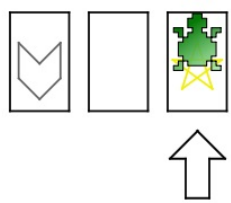

Figura 10: Resultado de uma simulação do problema de Monty Hall com as 3 portas, o prêmio (a estrela) e o bode (à esquerda).

Por causa da extensão da rotina, decidiu-se não inseri-la no artigo, mas ela pode ser acessada em http://bit.ly/1ASV2ZH e executada no software KTurtle para que se visualize os resultados. Por se tratar de um experimento aleatório, os resultados obtidos pelos alunos variaram, mas à medida que aumentávamos o número de repetições, a razão entre o número de vitórias dado que o participante trocou de porta e o número de vitórias dado que não houve troca tendeu a 2:1 como era esperado (a solução matemática do problema pode ser lida em [4]).

9. Conclusão

Após essas aulas, os alunos pediram explicitamente mais atividades envolvendo o KTurtle. Então, além do problema apresentado neste texto, ainda desenvolvemos simulações para o problema do jogo interrompido (http://m3.ime.unicamp.br/recursos/1062) e para o jogo das amebas (http://m3.ime.unicamp.br/recursos/1017), evidenciando o interesse despertado nos estudantes mesmo com as dificuldades iniciais referentes à aprendizagem da linguagem de programação.

Apesar de se tratar de um grupo de alunos com interesse declarado por Matemática, o envolvimento destes nas atividades propostas sugere que essa abordagem pode ser bastante motivadora para os estudantes e frutífera do ponto de vista educacional.

Acredito que a abordagem utilizada traz benefícios em duas frentes distintas: por um lado, motiva o estudo de probabilidade com simulações concretas para problemas compatíveis com o currículo regular, por outro lado, introduz um novo conjunto de habilidades relacionadas a programação de computadores a partir de tópicos presentes no currículo regular de Matemática.

O objetivo deste texto era justamente oferecer um possível caminho para essa abordagem sem exigir um grande domínio dos aspectos computacionais por parte do professor. Por isso demos tanta ênfase aos comandos e rotinas ao longo do texto, deixando os aspectos matemáticos mais formais nas referências para os leitores interessados.

Tanto do ponto de vista de incorporação de novas tecnologias na sala de aula quanto como da introdução dos estudantes ao universo da programação de computadores, que se configura hoje como uma habilidade onipresente no universo acadêmico e no mercado de trabalho ([2], [5]), acredito que as ideias apresentadas neste relato possam ser úteis pelo menos como um ponto de partida.

Referências

[1] Breijs, C. KTurtle. <http://edu.kde.org/kturtle>. Acessado em: 25 de julho de 2013.

[2] Brennan, K.; Resnick, M. *Using Artifact Based Interviews To Study The Development of Computational Thinking in Interactive Media Design*. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada, 2012.

[3] Carvalho, P. C. P. *Métodos de Contagem e Probabilidade*. OBMEP, IMPA, 2009.

[4] Morgado, A. C. *Os Dois Bodes*. Revista do Professor de Matemática, v. 33, 1997.

[5] Wing, J. M. *Computational Thinking*. Communications of the ACM, v. 49, n. 3, 2006.

Leonardo Barichello
Colégio Villa Lobos de Amparo e Colégio Anglo de Bragança Paulista
<barichello@gmail.com>

Recebido: 2014
Publicado: 2014