PCS 2190 - FOR: Repetição II

Ricardo Nakamura e Romero Tori

2015

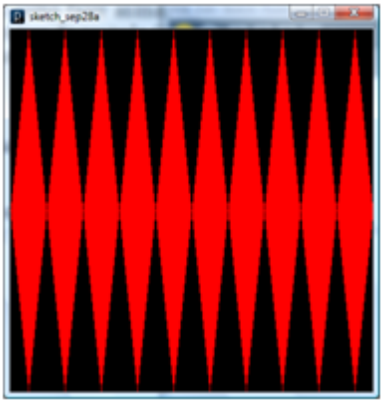
1 Introdução

Nesta apostila, continuaremos a discussão sobre laços de repetição, apresentando o comando **for**, uma alternativa ao comando **while**, com exemplos de como usar este recurso em programas interativos. Veremos também como utilizar arquivos de imagem no Processing.

2 Apresentando o comando FOR

Vamos relembrar o primeiro programa que fizemos na aula passada, utilizando o comando **while**. O laço de repetição faz com que os losangos sejam desenhados em várias posições horizontalmente na tela. A condição do **while** faz com que o laço se repita enquanto o valor da variável i for menor do que 400.

```
size(400, 400);
background(0);
noStroke();
fill(255, 0, 0);
int i = 0;
while(i < 400) {
  quad(i+20, 0, i+40, 200,
       i+20, 400, i, 200);
  i = i + 40;
}
```



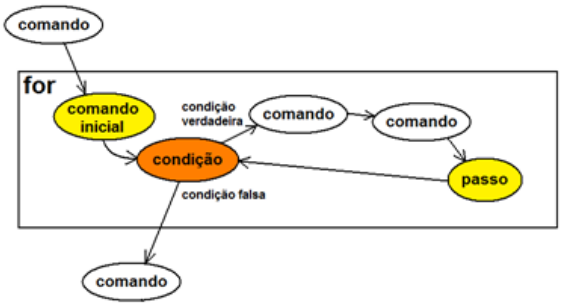
Agora, observe como o mesmo programa é escrito utilizando-se o comando **for**. Note que o programa faz **exatamente** a mesma coisa ao ser executado. Compare os dois programas e observe as mudanças que surgem devido ao novo comando. Qual dos dois parece mais simples de entender?

```
size(400, 400);
background(0);
noStroke();
fill(255, 0, 0);
int i;
for(i = 0; i < 400; i = i + 40) {
  quad(i+20, 0, i+40, 200,
       i+20, 400, i, 200);
}
```

A estrutura geral do comando for é apresentada a seguir. Nesta estrutura, o "comando inicial" é executado somente uma vez, quando o laço de repetição é iniciado. A condição é testada a cada passagem pelo laço, da mesma forma que no comando **while**. O comando "passo" é executado uma vez ao final de cada passagem pelo laço.

```
for (<comando inicial>; <condição>; <passo>) {
        <comandos a serem executados>
}
```

O diagrama a seguir ilustra o funcionamento do comando for, de maneira gráfica. Observe a relação dos três termos (comando inicial, condição e passo).



Devido ao formato do comando for, ele é frequentemente utilizado para expressar laços de repetição que envolvem uma sequência numérica. Veja alguns exemplos:

números inteiros entre 0 e 9	for (x = 0; x < 10; x = x + 1)
números pares entre 6 e 18	for (x = 6; x <= 18; x = x + 2)

Veja que, nestes casos, o comando **for** resulta em um código que inclui todos os elementos do laço de repetição em uma mesma linha, facilitando o seu entendimento. Vamos comparar o primeiro exemplo da tabela acima, escrito utilizando-se for e **while**:

FOR

```
for(x = 0; x < 10; x = x + 1) {
  …
}
```

WHILE

```
x = 0;
while (x < 10) {
  …
  x = x + 1;
}
```

Como se pode observar, os elementos que estão relacionados ao laço de repetição ficam “espalhados” em linhas diferentes no caso do comando **while** mas ficam condensados no comando **for**.

Atividade 1

Utilizando o comando **for**, faça um programa que desenha dez elipses na janela, arranjadas verticalmente.

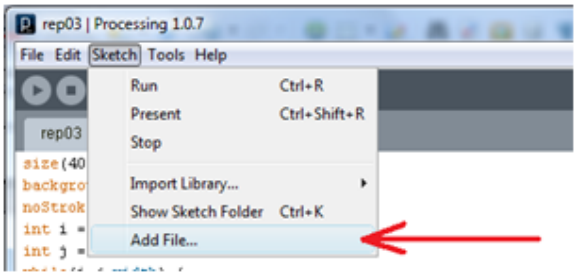
Atividade 2

Modifique o programa a seguir para utilizar o comando **for** no lugar do **while**:

```
size(400, 400);
background(0);
noStroke();
int i = 0;
int j = 0;
while(i < width) {
  j = 0;
  while(j < height) {
    rect(i, j, 20, 20);
    j = j + 40;
  }
  i = i + 40;
}
```

3 Desenhando imagens

Até o momento, só utilizamos comandos para desenhar formas geométricas no Processing. No entanto, também é possível utilizar arquivos de imagem. Para isso, primeiramente temos que adicionar o arquivo de imagem ao programa. Isso é feito através do menu Sketch → Add File, como mostrado a seguir.



Podem ser utilizadas imagens nos formatos JPG, PNG, GIF e TIFF (note que o formato BMP do Windows não é aceito no Processing).

Depois que a imagem foi adicionada, pode-se carregar a imagem através do comando **loadImage**. A imagem é armazenada em uma variável do tipo **PImage**. Para se desenhar a imagem na tela, utiliza-se o comando **image**. O programa a seguir mostra como isso tudo pode ser feito.

```
size(400, 400);
PImage minhaImagem;
// não se esqueça de adicionar uma imagem
// e mudar o nome na linha abaixo!
minhaImagem = loadImage("interlab.jpg");
image(minhaImagem, 0, 0);
```

O comando **image** recebe três parâmetros: o primeiro deve ser a variável que contém a imagem que foi carregada anteriormente. Os outros dois são as coordenadas (x, y) do canto superior esquerdo da imagem na janela, conforme a figura a seguir.

Em algumas situações você pode preferir desenhar uma imagem a partir do centro, ao invés do canto superior esquerdo. Para isso, basta utilizar o comando **imageMode(CENTER)**. Para voltar a usar o canto, utilize **imageMode(CORNER)**.

Atividade 3

Modifique o programa anterior para desenhar duas imagens lado a lado na janela.

Atividade 4

Outro comando que modifica o desenho das imagens é o comando tint. Este comando pode ser utilizado para tingir uma imagem de outra cor ou mesmo aplicar transparência. Modifique o programa anterior para utilizar este comando (procure na Referência do Processing os detalhes sobre o seu uso).

4 Utilizando repetições em programas interativos

Na aula anterior, utilizamos comandos de repetição somente para desenhar padrões estáticos. No entanto, podemos também combinar a repetição com animações. O programa a seguir ilustra um uso do comando for para fazer uma animação simples.

```
PImage minhaImagem;
int y;

void setup() {
  size(500, 500);
  // mude para o nome do arquivo que você adicionou
  // ao seu sketch usando o menu Sketch -> Add File
  // os formatos de imagem aceitos são GIF, PNG, JPG e TIF
  minhaImagem = loadImage("interlab.jpg");
  y = 0;
}

void draw() {
  int x;
  background(0);
  for (x = 0; x < 3; x = x + 1) {
    image(minhaImagem, x*minhaImagem.width, y);
  }
  y = y + 1;
  if (y > height) {
    y = 0;
  }
}
```

Observe o uso da expressão "minhaImagem.width" na listagem acima. Ela é usada para recuperar a largura da imagem. Da mesma forma, pode-se escrever **minhaImagem.height** para obter a altura da imagem em pixels. Podemos fazer isso porque o tipo **PImage** é um tipo composto: um tipo de variável que é associada um conjunto de informações — ao contrário, por exemplo, do tipo **int**, que só é associado um número inteiro. Para acessar as informações adicionais de um tipo composto, utilizamos sempre a sintaxe mostrada acima (variável.informação).

Atividade 5

Modifique o programa anterior para fazer as imagens "rebaterem" nas margens superior e inferior da janela gráfica.

Outro exemplo que combina repetições com **draw** e comandos como **for** e **while** é mostrado a seguir. Neste caso, um desenho de fundo é realizado através de repetições com **for** e a animação de uma barra se movendo verticalmente é feita sobre esse fundo.

```
int linha;
void setup() {
  size(400, 400);
  linha = 0;
  noStroke();
}
void draw() {
  background(0);
  fill(0, 255, 0);
  int i, j;
  for (i = 0; i < 40; i = i + 1) {
    for (j = 0; j <= i; j = j + 1) {
      ellipse(5+(10*j), 5+(10*i), 10, 10);
    }
  }
  fill(0, 255, 255);
  rect(0, linha, width, 10);
  linha = linha + 1;
  if (linha > height) {
    linha = 0;
  }
}
```

Atividade 6

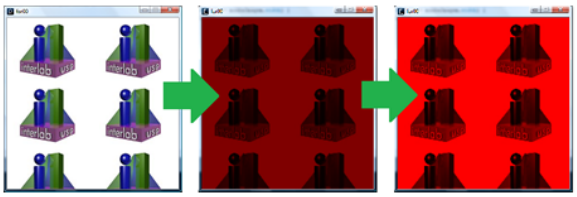
Acrescente comentários ao programa anterior, indicando qual parte do programa faz o desenho de fundo e qual parte faz a animação da barra.

5 Resumo

Nesta atividade, aprendemos um novo comando para laços de repetição e também como utilizar imagens no Processing. Neste ponto, tivemos contato com os conceitos fundamentais de lógica de programação: variáveis, programas (como sequências de comandos), instruções de execução condicional e de repetição.

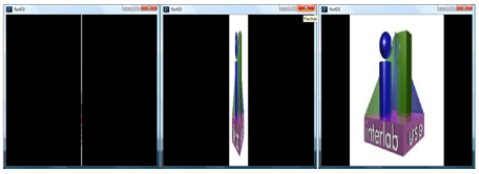
Atividade 7

Crie um programa que desenha uma grade de imagens que vai sendo progressivamente tingida de vermelho, conforme a figura abaixo. Observação: se a animação ficar muito rápida, experimente utilizar o comando **delay** no final do bloco **draw** (veja a referência do Processing para maiores informações.



Atividade 8

Crie um programa que desenha uma imagem que vai progressivamente aumentando de largura até ficar da largura da janela, como ilustrado abaixo. **Dica:** o comando **image** pode receber dois parâmetros adicionais, que indicam a largura e a altura com que a imagem deve ser desenhada na tela. Desta forma, pode-se desenhar versões da mesma imagem com tamanhos diferentes.



Atividade 9

Desafio: Crie um programa que desenha uma matriz de imagens que pulsam em tamanho e cor, como ilustrado na sequência de quadros abaixo.