PCS 2190 - COND: Execução condicional

Ricardo Nakamura e Romero Tori

2015

1 Introdução

Na apostila anterior aprendemos a fazer desenhos e animações utilizando a repetição de comandos no bloco **draw**. Nesta apostila vamos trabalhar com a utilização de comandos condicionais para criar animações mais interessantes.

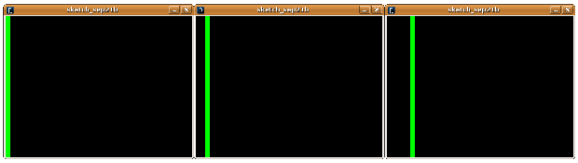
Vamos começar retomando o programa de animação de uma barra que se move horizontalmente. Nesse programa, utilizamos uma variável para controlar a posição horizontal da barra. A cada repetição do bloco **draw**, o valor associado à variável aumenta e, com isso, a barra é desenhada cada vez mais para a direita.

```
// Esta variável será usada para definir
// a posição horizontal do retângulo
int x;

void setup() {
  size(400, 300);
  noStroke();
  fill(0, 255, 0);
  x = 0;
}

void draw() {
  // apaga a imagem anterior...
  background(0);
  // e desenha na nova posição
  x = x + 1;
  rect(x, 0, 10, height);
}
```

A imagem a seguir ilustra a execução do programa em três instantes distintos, em que x vale 0, 20 e 50, respectivamente.



Se você executar o sketch de animação por tempo suficiente, verá o retângulo atravessar toda a janela e desaparecer no lado direito. Isso acontece porque, no bloco draw, sempre incrementamos o valor de x.

Para fazer com que a animação continue se repetindo, gostaríamos de fazer com que, quando o retângulo sair pelo lado direito da janela, ele volte para o lado esquerdo. Podemos conseguir isso fazendo o valor de x retornar a 0 nesse momento. Ou seja, precisamos acrescentar o seguinte comando no nosso sketch:

"se o valor de x for maior do que a largura da janela, x deve voltar a ser zero."

No Processing, podemos fazer isso através do comando condicional IF. O efeito do IF é o de executar um ou mais comandos **somente se** uma certa condição for verdadeira. Podemos escrever a condição acima modificando o sketch da seguinte forma:

```
int x;

void setup() {
  size(400, 300);
  noStroke();
  fill(0, 255, 0);
  x = 0;
}

void draw() {
  background(0);
  rect(x, 0, 10, height);
  x = x + 1;
  // acrescentamos este comando para fazer
  // o retângulo permanecer na janela!
  if (x > width) {
    x = 0;
  }
}
```

Observe o uso de chaves para delimitar os comandos que serão executados se a condição (x > width) for verdadeira. Experimente rodar novamente o sketch para ver o resultado. O formato geral do comando IF é o seguinte:

```
if (condição) {
  //comandos que serão executados se a condição for verdadeira
}
else if (outra condição) {
  // comandos que serão executados
  // se a outra condição for verdadeira
}
else {
  // comandos que serão executados
  // se todas as condições forem falsas
}
```

As partes "else" e "else if" são opcionais. As condições são comparações lógicas, que só podem ser verdadeiras ou falsas. Geralmente, utilizamos comparações entre dois valores, como A > B ou A < B. No entanto, alguns símbolos matemáticos são escritos de forma um pouco diferente no Processing:

Condição	Símbolo matemático	Símbolo no Processing
maior ou igual	$\geq$	>=
menor ou igual	$\leq$	<=
igual	=	==
diferente	$\neq$	!=

Observe que, embora o bloco **draw** seja executado repetidamente, o comando "x = 0;" só será executado quando o valor da variável x for maior que a largura da janela. Dessa forma, sempre que o retângulo "desaparecer" no lado direito da janela, a variável x será associada novamente ao valor zero, fazendo com que o retângulo volte para o lado esquerdo.

Atividade 1

Faça o retângulo voltar ao lado esquerdo da janela assim que ele chegar à metade da sua largura.

Atividade 2

Acrescente um segundo retângulo, que se move da direita para a esquerda, também até a metade da janela.

2 Fazendo o retângulo "rebater" nas paredes

A partir de tudo o que vimos até agora, podemos fazer o retângulo se mover para a esquerda e para a direita, como se estivesse rebatendo nas laterais da janela. Uma maneira de se fazer isso é simulando uma colisão, como se as laterais da janela fossem paredes rígidas. Neste caso, ao bater nelas, o retângulo simplesmente inverte sua velocidade na direção perpendicular à parede, como ilustrado a seguir:

Para isso, antes de mais nada é preciso definirmos uma nova variável para a velocidade do retângulo. Atualmente, a velocidade é o número 1 na linha "x = x + 1" e, assim, não teremos como inverter o seu sinal quando ocorrer a colisão.

Para que a variável seja acessível nos blocos setup e draw, é preciso que ela seja definida fora dos blocos (por exemplo, junto com a variável x). Caso contrário, ela só irá existir naquele bloco.

A listagem a seguir mostra o programa modificado para o efeito de rebater nas bordas da janela.

```
int x;
int vx;

void setup() {
  size(400, 300);
  noStroke();
  fill(0, 255, 0);
```

```
  x = 0;
  // o retângulo começa indo para a direita
  vx = 1;
}

void draw() {
  background(0);
  rect(x, 0, 10, height);
  x = x + vx;
  // quando atingir uma parede, inverte a velocidade
  if (x < 0 || x > width) {
    vx = -vx;
  }
}
```

Você deve ter notado o símbolo || na condição do if. Este símbolo é utilizado para ligar duas condições com um "OU". Isto significa que os comandos serão executados se pelo menos uma das duas condições forem verdadeiras.

Também é possível conectar duas condições com o símbolo &&. Neste caso, queremos indicar que os comandos devem ser executados somente se as duas condições forem verdadeiras.

Atividade 3

Modifique o programa anterior para utilizar **else if** ao invés de utilizar o conector || para ligar duas condições.

3 Usando if para criar desenhos complexos

Nos exemplos anteriores utilizamos o comando **if** para criar animações. No entanto, também podemos usar a execução condicional para criar desenhos mais elaborados. Por exemplo, o programa a seguir cria um tabuleiro de xadrez na tela:

```
int cor;
int x;
int y;
```

```
void setup() {
  size(400, 400);
  cor = 0;
  x = 0;
  y = 0;
}

void draw() {
  fill(cor);
  rect(x, y, 50, 50);
  // alterna a cor entre preto e branco
  if (cor == 0) {
        cor = 255;
  }
  else {
        cor = 0;
  }
  // avança para a próxima casa
  x = x + 50;
  // se chegamos ao final de uma linha,
  // desce para a de baixo
  if (x > 400) {
    x = 0;
    y = y + 50;
    if (y > 400) {
        noLoop();
    }
  }
}
```

Este programa apresenta também um novo comando: **noLoop()**. Quando este comando é utilizado, ele faz com que o Processing pare de repetir os comandos do bloco **draw**. Nesse programa, este comando é usado para terminar o desenho quando a janela toda estiver preenchida com quadrados do tabuleiro.

4 Interagindo com o mouse

Vamos fazer agora um exemplo que utiliza o comando **if** para criar uma interação simples com o mouse. Na apostila anterior, vimos que podemos utilizar as variáveis automáticas mouseX e mouseY para ler a posição do cursor do mouse dentro da janela gráfica. A partir destas variáveis, vamos criar um sketch em que um círculo muda de cor quando o cursor do mouse é movido para dentro dele:

```
int diametro;

void setup() {
  size(200, 200);
  stroke(0, 0, 255);
  strokeWeight(3);
  background(0);
  diametro = width/3;
}

void draw() {
  int d;
  d = dist(mouseX, mouseY, width/2, height/2);
  if (d < diametro/2) {
    fill(255, 0, 0);
  }
  else {
    fill(255, 255, 255);
  }
  ellipse(width/2, height/2, diametro, diametro);
}
```

A condição do if verifica se a distância entre a posição do cursor do mouse e o centro da tela é menor do que o raio do círculo. Para isso, usamos o comando dist( ), que recebe como parâmetros as coordenadas X e Y de dois pontos e calcula a distância entre eles. Procure entender o restante do sketch e tirar suas dúvidas durante a aula.

5 Resumo

Nesta apostila, estudamos a execução condicional de comandos através do comando **if** e vimos também o conceito de expressões lógicas – que só podem ser verdadeiras ou falsas – e como elas são utilizadas com o comando **if**.

Atividade 4

Faça um programa em que um quadrado se move pela tela, rebatendo nas quatro laterais da janela como se fossem paredes.

Atividade 5

Faça um programa que desenha formas geométricas que mudam de cor dependendo da posição horizontal do mouse na tela (dica: utilize o comando color(R,G,B)).

Atividade 6

Desafio: Crie um jogo de "paredão" onde o jogador move um retângulo (com 1/3 da altura da janela) posicionado no canto esquerdo da janela através do mouse. O retângulo só se move verticalmente. Um círculo se move pela tela e rebate nas laterais superior, direita e inferior, mas deve tocar um retângulo para não "escapar" da tela pela esquerda. Se o círculo escapar da tela, o jogo termina.