

Programação Multimédia

Métodos

Métodos

- É frequente repetirmos código que faz a mesma operação
 - Possivelmente sobre dados diferentes
- Em vez de escrevermos repetidamente o mesmo código podemos dar um nome a uma secção de código e invocar essa secção

Métodos - Exemplo

```
void setup() {  
    size(200, 200);  
}
```

```
void draw() {  
    background(0);  
  
    fill(255, 255, 255);    //branco  
    rect(0, 0, 50, 50);    //cara  
  
    fill(0, 255, 0);        //verde  
    rect(10, 10, 10, 10);  //olho  
    esquerdo  
    rect(30, 10, 10, 10);  // olho  
    direito  
  
    fill(255, 0, 0);        //vermelho  
    rect(15, 30, 20, 10);  // boca  
}
```

- E se quiséssemos duas caras, agora?

Métodos - Exemplo

```
void setup() {
  size(200, 200);
}

void draw() {
  background(0);

  fill(255, 255, 255);    //branco
  rect(0, 0, 50, 50);    //cara

  fill(0, 255, 0);       //verde
  rect(10, 10, 10, 10);  //olho esquerdo
  rect(30, 10, 10, 10); // olho direito

  fill(255, 0, 0);       //vermelho
  rect(15, 30, 20, 10);  // boca

  // segunda cara
  fill(255, 255, 255);  //branco
  rect(100, 0, 50, 50); //cara

  fill(0, 255, 0);      //verde
  rect(110, 10, 10, 10); //olho esquerdo
  rect(130, 10, 10, 10); // olho direito

  fill(255, 0, 0);      //vermelho
  rect(115, 30, 20, 10); // boca
}
```

- E, agora, três?
- Começa a ser demais!

Métodos

- O código para desenhar as duas caras é muito semelhante
- Podemos reutilizá-lo
- Criamos um método

```
int x;
int y;

void setup() {
  size(200, 200);
}

void draw() {
  background(0);

  x = 0;
  y = 0;
  desenhaCara();

  x = 50;
  y = 50;
  desenhaCara();
}

void desenhaCara() {
  fill(255, 255, 255); //branco
  rect(x, y, 50, 50); //cara

  fill(0, 255, 0); //verde
  rect(x+10, y+10, 10, 10); //olho esquerdo
  rect(x+30, y+10, 10, 10); //olho direito

  fill(255, 0, 0); //vermelho
  rect(x+15, y+30, 20, 10); //boca
}
```

Exercício met_1

1. Crie um método “quadradoAnimado” que anime um quadrado no ecrã
 1. O quadrado deve mover-se continuamente na horizontal (a posição vertical deve ser pré-definida)
 2. Invoque esse método no “draw”
2. Crie outro método “linhaAnimada” que anime uma linha no ecrã
 - Invoque também esse método no “draw”
 - Cuidado com os nomes das variáveis: não devem entrar em conflito com as usadas no método quadradoAnimado

Parâmetros

- Em vez de utilizarmos variáveis globais podemos passar parâmetros

```
void desenhaCara(int posicaoX, int
posicaoY) {
    fill(255, 255, 255);
    //branco
    rect(posicaoX, posicaoY, 50,
50);          //cara

    fill(0, 255, 0);
    //verde
    rect(posicaoX+10, posicaoY+10,
10, 10);     //olho esquerdo
    rect(posicaoX+30, posicaoY+10,
10, 10);     // olho direito

    fill(255, 0, 0);
    //vermelho
    rect(posicaoX+15, posicaoY+30,
20, 10);     // boca
}
```

Parametros

- O nosso draw() passa a ser:

```
void draw() {  
    background(0);  
  
    desenhaCara(0, 0);  
  
    desenhaCara(50, 50);  
}
```

Exercício met_2

- Modifique o método `quadradoAnimado` de forma a que a posição vertical do quadrado seja um parâmetro passado ao método.
 - Invoque, no “draw” três vezes o método novo com valores diferentes no parâmetro

Valor de Retorno

- O método que utilizamos no exemplo anterior não pode ser utilizado em expressões:

```
int x
```

```
x = 3*5 + desenhaCara(20, 20)
```

– Erro!

- Para funcionar `desenhaCara(20, 20)` teria de ser substituído por um valor – qual?

Valor de Retorno

- Para indicarmos que um método devolve um valor quando termina temos de substituir “void” por um tipo de dados

```
int calculaPontuacao(int vitoria, int empate) {  
    int pontos = vitoria*3 + empate;  
    return pontos;  
}
```

(Quase) Tudo São Métodos

- Nos nossos programas, quase todas as instruções utilizam métodos
- As funcionalidades específicas de uma linguagem são definidas pelos métodos (API) que disponibiliza
- Tudo o resto é sintaxe (ciclos, condições, declaração de variáveis, ...)

CallBack

- Existem alguns métodos que, apesar de sermos nós (programadores) a implementar, não somos nós que os invocamos.
- São métodos pré-definidos pelo Processing
 - O programador deve reimplementá-los para tirar partido deles
- Por exemplo, o método `draw()`
 - Temos de o declarar e implementar, nunca o invocamos.
- Estes métodos são chamados métodos de *callback* (porque são invocados de volta pela linguagem)

CallBack – Gráficos e Teclado

- `setup()`
 - invocado pelo Processing antes de iniciar o nosso programa. Este método serve para inicializarmos as nossas variáveis. Devemos colocar aqui o código que queremos que execute apenas uma vez, no início do programa.
- `draw()`
 - invocado periodicamente pelo Processing para actualizar o estado do programa. Neste método devemos colocar o código para desenhar o nosso programa. A frequência com que este método é invocado pelo Processing pode ser controlado.
- `keyPressed()`
 - invocado pelo Processing quando o utilizador carrega numa tecla do teclado.
- `keyReleased()`
 - invocado pelo Processing quando o utilizador larga uma tecla do teclado.

CallBack - Rato

- `mousePressed()`
 - invocado pelo Processing quando o utilizador clica num botão do rato.
- `mouseMoved()`
 - invocado pelo Processing quando o utilizador arrasta o rato.
- `mouseDragged()`
 - invocado pelo Processing quando o utilizador arrasta o rato mantendo um botão pressionado.
- `mouseReleased()`
 - invocado pelo Processing quando o utilizador larga um botão do rato.