Introdução à Programação Gráfica com Processing
Manual da Formação @ Audiência Zero
Porto, 2008-03-22 e 23

- Pedro Amado, 2008-03-18. Actualizado em 2008-04-25

Qualquer correcão ou sugestão:
pedamado (at) gmail (dot) com

Para mais informações:
http://pedamado.wordpress.com/2008/03/22/processing-audiencia-zero/

Exercícios

Introdução, Conceitos Básicos e Primitivas

14 Operadores Condicionais
15 Operadores Lógicos

Conceitos Estruturais

16 Modo Contínuo
17 Condições
17b Condições Switch/Case*
18 Iterações
18b Iterações (Nested)

Aplicação e interacção

19 Pong Simples (Pseudo Código)
20 Pong Simples (funcional versão de 2006) *
20a Pong Simples (versão da Formação)
20b Pong Individual
21 Sintaxe UI

Operações e Dados compostos

22 Matemática
22b Matemática - Função dist() - Colisões
23b Trigonometria (rotações simples e arrasto)
24 Transformação
24b Transformação
25 Acções Gatilhos e Eventos
26 Vectores (Arrays)
26b Vectores

Funções, Parametros e Objectos

27 Funções e Parametros
28 Classes e Objectos (Noção e Introdução)
28b Classes e Objectos (Construção de Classes)
....Segundo Tab
29 Texto
29b Texto (optimização de tamanhos)
30 Imagens

Exercício integrado

Bibliotecas e expansão

---

01 Hello World!

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

println("Hello World!");
```

02 Comentários

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// Isto é um comentário, é ignorado pela execução final do programa e
só serve como referência para os utilizadores
```

```
// Pode preceder ou seguir-se a uma (ou várias linas de código)

println("Hello World"); //Imprime para a consola a mensagem entre
aspas

/*
Isto é um comentário Multi-linha
Pode ser colocado onde se quiser.
É utilizado mais genericamente para documentação.
*/

/*
http://www.processing.org/reference/println_.html
*/
```

03 Instruções

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// Todas a ordens ao programa são instruções
// As instruções têm que conter uma indicação de acção/operação e um
terminador (para parar de fzer)

// Por exemplo:

println("A minha mensagem");

// println manda sair uma linha de texto na consola (abaixo)
// A linha de texto é o que se encontra dentro de parentesis
(parametros da instrução)
// e o terminador diz ao programa para parar de imprimir a  mensagem
```

04 Print

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

print("Mensagem simples"); // imprime os caracteres entre "aspas"
println("Linha de mensagem"); // faz o mesmo, só que no final faz uma
quebra de linha na consola
```

```
// Experimente repetir várias vezes estas instruções e veja a
diferença
```

05 Primitivas

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

point(10, 10); //Ponto X,Y

line(10, 20, 10, 30); //Linha do Ponto X0, Y0, ao Ponto X1, Y1

rect(10, 40, 10, 10); //Rectangulo Ponto X0, Y0, Largura e Altura

ellipse(10, 60, 10, 10); //Ellipse Ponto X0, Y0 (Centro), Largura e
Altura (Diametral)
```

06 Bezier

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// Desenhar um curva do topo para o fundo, da esquerda para a direita
// Do ponto A ao ponto D

//Ponto A
ellipse(5, 5, 5, 5);

//Ponto B
ellipse(5, 40, 5, 5);

//Ponto C
ellipse(90, 60, 5, 5);

//Ponto D
ellipse(90, 90, 5, 5);

//Linha A-B
line(5, 5, 5, 40);

//Linha D-C
line(90, 90, 90, 60);

//Curva Bezier A (B)- (C) D
```

```
// A e D são os Pontos Bezier de Início e fim. B e C são os BCPs
respectivos de cada ponto da curva
bezier(5, 5, 5, 40, 90, 60, 90, 90);
```

07 Cor

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// Cor de fundo (RRR, GGG, BBB)
background(255);

// Preenchimento de formas (R, G, B)
fill(255, 0, 0);

// Cor da linha de contorno (R, G, B)
stroke(0, 0, 255);

ellipse(10, 10, 10, 10);

fill(255, 200, 0);
stroke(0, 255, 0);
ellipse(30, 30, 10, 10);
```

08 Atributos

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// Ligar a suavização de circulos e de diagonais
smooth();
ellipse(10, 10, 10, 10);

// Desligar a suavização
noSmooth();
ellipse(50, 10, 10, 10);

// Espessura das linhas de contorno
strokeWeight(1);
ellipse(10, 30, 10, 10);

strokeWeight(5);
```

```
ellipse(50, 30, 10, 10);

// Tipo de terminação das linhas
strokeWeight(8);
strokeCap(ROUND);
line(10, 50, 50, 50);

strokeCap(SQUARE);
line(10, 60, 50, 60);

strokeCap(PROJECT);
line(10, 70, 50, 70);

strokeWeight(1);
fill(255, 0, 0, 127); // RGB+A
ellipse(10, 90, 10, 10);
fill(0, 255, 0, 127);
ellipseMode(CORNER); // Origem do desenho da forma *Mode
ellipse(10, 90, 10, 10);
```

09 Variáveis

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a; // Declarar (int é um número inteiro)
a = 10; // Inicializar (e atribuir)
a = a+1; // Atribuir
println(a);

boolean b = true; // Verdadeiro ou falso
println(b);

char c = 's'; // só uma letra --> diferente de uma string " "
println(c);

String d = "texto"; // texto é um conjunto de chars, daí ser uma
string, uma variavel composta, uma lista de chars
println(d);

float e = 3.1415; // float é um número real, de vírgula flutuante
println(e);
```

10 Operadores Aritméticos

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a = 10;
int b = 6;
int c = 7;

int ab = a + b; // Operações aritméticas normais
println(ab);

int bc = b-c;
println(ab);

int ac = a*c;
println(ab);

// Precedência matemática
int bca = a+b*c;
println(bca);

int abc = (a+b)*c;
println(abc);
```

11 Abreviações

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a = 2;
a = a+1;
println(a);

a = 2;
a++;
println(a);

a = a*2;
println(a);

a = 2;
a *=2;
println(a);
```

12 Restrições

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

//ceil(), floor(), round(), min(), max()

float a = 3.24;
println(a);

println( ceil(a) ); // valor inteiro (tecto)
println( floor(a) ); // valor inteiro (limite inferior)
println( round(a) ); // valor inteiro (arredondado)

println( min(4.3, 4.6) ); // valor inteiro --> escolha entre dois
valores, ou uma lista de valores
```

13 Operadores Relacionais

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a = 10;
int b = 20;
float c = 10.5;
int d = 10;
//boolean e = a > b;

println(a < b); // comparação de valores
println(a = b); // Atribuição
println(a==b); // Comparação de tipo de variaveis
println(a==c);
```

14 Operadores Condicionais

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
```

```
// © Pedro Amado, 2008-03-20

int a = 20;
int b = 20;
float c = 10.5;
int d = 10;
//boolean e = a = b;

if (a < b) {
  println("A é menor que B");
}
else if (a > b){
  println("A é maior que B");
} else if (a == b ){
  println("A e B são do mesmo tipo e são iguais");
  }
```

15 Operadores Lógicos

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a = 10;
int b = 10;
float c = 9.5;
int d = 10;
//boolean e = a = b;

if (a == b && a < c) {
  println("A é igual a B e menor que C");
  }
```

16 Modo Contínuo

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20
```

```
// O Modo contínuo "chama" automaticamente duas funções (Callbacks):
// Setup() --> prepara a aplicação
// Draw() --> desenha X vezes por segundo todas as instruções que
contém
// Também verifica acções de utilizador através do JAVA (teclado e
rato)

// Preparação do Applet
void setup() {
  size(200, 200); // Tamanho da janela
  noStroke();
  background(255);
  fill(0, 102, 153, 204);
  smooth();
  loop(); // correr o Draw sempre
  noLoop(); // Correr o Draw uma só vez
}

// A cada ciclo corre as instruções
void draw() {
  circles(40, 80); // Invocar um função (personalizada) -->
nome_da_função ( parametros, param... )
  circles(90, 70);
}

void circles(int x, int y) { // declarar uma função (mesmo que um
callback)
  ellipse(x, y, 50, 50);
  ellipse(x+20, y+20, 60, 60);
}
```

17 Condições

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

boolean a = true; // visibilidade (scope) global

void setup() {
  size(300, 300);
  background(255);
  frameRate(30);

  int b = 10; // Visibilidade (Scope) local --> só pode ser usada
dentro deste bloco de código

}

// Bloco de código de função draw() '{'
```

```
void draw() {

  // Bloco de código de condição '{'
  // if --> se, ( condição = verdadeira ), então bloco de instruções
  if (a) {
    // Intruções do bloco
    println("True");

  }
  // Fecha o bloco de código de condição '}'

  // println(b); // Esta linha de código dá erro, porque o 'b' é uma
variável local do setup()
}

// Fecha o bloco de código da função draw() '}'
```

17b Condições Switch/Case*

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

int bgc;

void setup() {
  size(300, 300);
  bgc = 0;
  background(bgc);
}

void draw() {

  background(bgc);

  ellipse(mouseX, mouseY, 10, 10);

}

// Callback automático de Teclado
void keyPressed() {

  // Interruptor ( gatilho )
  switch(key) {
  case '0':    // Caso o gatilho seja '0' --> CHAR
    bgc = 0;
    cursor(CROSS);
    break;    // Precisamos parar a verificação do gatilho, caso
contrário irá espoletar as acções seguintes
  case '1':
    bgc = 100;
    cursor(MOVE);
    break;
  case '2':
```

```
    bgc = 200;
    cursor(HAND);
    break;
  default:
  noCursor();
    println("Carregue nas teclas 0-2");
  }
}
```

18 Iterações

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a = 5;

void setup() {
  size(200, 200);
  background(255);
  frameRate(30); // Acrescentar uma "velocidade" temporal ao Applet

}

void draw() {
  background(255); // limpar o fundo a cada frame
  rect(a, a, 10, 10); // desenhar uma vez
  //a = a+3;
  // rect(a, a, 10, 10); // desenhar outra vez, e outra, e outra...
já me doem os dedos só de pensar...

  // desenhar em cilcos / iterações
  // para ( condição inicial X, condição final e alteração a cada
ciclo )
  // inicia o bloco de instruções
  // repete o bloco X vezes até à condição final
  for(int i = 10; i < 200; i+=15) {
    ellipse(i, 30, 10, 10);
    }
}
```

18b Iterações (Nested)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
```

```
// © Pedro Amado, 2005-12-07

// Estruturas e Controlo - Iterações
// --------------------------------

// Declaração das Globais (iterações simples)
int k;
int num = 12;

// Globais da iterações recursiva
float rectsize = 10;
float rectspace = 15;
int margem = 5;

void setup() {
  size(200,200);
  stroke(255);
  fill(180);
  background(0);
  // framerate(30);
  noLoop();
}

void draw() {
  background(0);
  fill(180);
  k=60;


  // Exemplo de estrutura/iteração recursiva
  // Linha horizontal (primeira condição)
  for(int x = margem; x <= width-margem; x += rectspace){
    noStroke();
    fill(x+50);
    // Coluna vertical (repete para cada linha)
    for(int v = margem; v <= height-margem; v += rectspace){
      rect(v, x, rectsize, rectsize);
    }
    rectsize = rectsize - 0.5;
  }
}
```

19 Pong Simples (Pseudo Código)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2005-12-15

// Simple Pong v0.2 Alpha

// Collision e Constrain e Random
// ------------------------------

// desenho dos objectos por dados e localizações
```

```
// (mais fácil para actualizar posições e redesenhar bem como testar
colisões)

// Declara?‹o de vari‡veis globais
// incremento de direcção /localização da bola
// orientação/direcção (positiva ou negativa)
// raio da bola;
// direcção horizontal

// Paddle Height e Width

// Margem de seguran?a para saída desenho do paddle e teste colisão
com a bola

// implementação de pausa (boolean)

// Inicialização
void setup() {

}

void draw() {
  // refrescar o ecrã (background)

  // Inicia o movimento da bola na vertical
  // posição da bola += incremento

  // ver se a bola cai do ecra (passa para além da margem)
  // e recoloca-a de novo no topo do ecr‹ aleatoriamente

  // Constrain movimento do paddle ^ largura do ecrã

  // Teste de colisão da bola com o paddle (float)
  // limit = tamanho do ecra menos a margem de segurança onde se
encontra o paddle)
  // limit = zona de testes

  // verificação se quando a bola está no limite está dentro do
paddle numa s- express‹o
    // verificação da posição do paddle em rela?‹o ^ bola
    // alteração da orientação horizontal
      //pmouseX = last mouse loc

  // inverte a direcção se bater no tecto < 0

  // Inverte a dir_x cada vez que toca nas paredes < 0 || > width

  // Desenha a bola (ellispe, locx, locy e size)

  // Desenha a paddle (rect, locx, locy, size);

  // desliga o cursor
}


void mousePressed() {
  // Simple pause (boolean)
}
```

20 Pong Simples (funcional versão de 2006) *

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2005-12-15

// Simple Pong v0.1 Alpha

// Collision e Constrain e Random
// ------------------------------


// Declaração de variáveis globais
float bola_x, bola_y; // incremento de direcção /localização da bola
float bola_dir = 3; // orientação/direcção (positiva ou negativa)
float bola_size = 5; //raio da bola;
float dir = 0; //direcção horizontal

int paddle_w = 50; // Paddle Height e Width
int paddle_h = 10;

int dist_wall = 15; // Margem de segurança para saída desenho do
paddle e teste colis‹o com a bola

boolean looping; // implementação de pausa

// Inicialização
void setup() {
  size(200, 200);
  background(0);
  fill(255);
  //noLoop();
  frameRate(60);
  smooth();
  rectMode(CENTER);
  ellipseMode(CENTER_RADIUS);
  // Localização inicial
  bola_x = width/2;
  bola_y = 1;

  looping = true;

}

void draw() {
  // refrescar o ecrã
  background(40);

  // Inicia o movimento da bola na vertical
  bola_y += bola_dir*2;
  bola_x += dir;

  /*
   random() Generates random numbers.
   Each time the random() function is called, it returns
   an unexpected value within the specified range. If one
```

```
   parameter is passed to the function it will return a
   float between zero and the value of the parameter.
   The function call random(5) returns values between 0 and 5.
   If two parameters are passed, it will return a float with a
   value between the the parameters. The function call
   random(-5, 10.2) returns values between -5 and 10.2.
   To convert a floating-point random number to an integer,
   use the int() function.
   */

  //ver se a bola cai do ecra (passa para além da margem)
  // e recoloca-a de novo no topo do ecr‹ aleatoriamente
  if (bola_y > height+bola_size) {
    bola_y = 1;
    bola_x = random(width);
    dir = random(-1.0,1.0);
  }

  // Constrain movimento do paddle ^ largura do ecrã
  int paddle_x = constrain(mouseX, paddle_w/2, width-paddle_w/2);

  // Teste de colisão da bola com o paddle
  // limit = tamanho do ecra menos a margem de segurança onde se
encontra o paddle)
  // limit = zona de testes
  float limit = height - dist_wall - paddle_h - bola_size;
  //println(limit);

  // verificação se quando a bola está no limit está dentro do paddle
numa só expressão
  if (bola_y >= limit && bola_y <= limit+bola_dir*2 && bola_x >
paddle_x-paddle_w/2 && bola_x < paddle_x + paddle_w/2) {
    bola_dir *= -1;
    // verificação da posição do paddle em relação à bola
    // alteração da orienta?‹o horizontal
    if (mouseX != pmouseX) { //pmouseX = last mouse loc
      dir = (mouseX -pmouseX)/2.0;
      if (dir > 5) {
        dir = 5;
      }
      if (dir < -5) {
        dir = -5;
      }
    }
  }

  // inverte a direcção se bater no paddle ou no tecto
  if (bola_y < bola_size && bola_dir < 1) {
    //println(bola_dir);
    bola_dir *=-1;
  }

  // Inverte a dir_x cada vez que toca nas paredes
  if (bola_x > width - bola_size) {
    dir *= -1;
  }
  if (bola_x < bola_size) {
    dir *= -1;
  }

  // Desenha a bola
```

```
  fill(255);
  smooth();
  ellipse(bola_x, bola_y, bola_size, bola_size);

  // Desenha a paddle
  fill(200);
  noSmooth();
  rect(paddle_x, height - dist_wall, paddle_w, paddle_h);

  // desliga o cursor
  noCursor();
}


void mousePressed() {

  // Simple pause
  if (looping) {
    looping=!looping;
    noLoop();
  }
  else{
    loop();
  }
}
```

20a Pong Simples (versão da Formação)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-22

int posY, posX, dirY, dirX, vel, velX, dia;
int padX, padY, padW, padH;


void setup() {
  size(500, 500);
  background(255);

  posY = 0;
  posX = int( random(width) ); // Função matemática que devolve um
número (semi) aleatório do tipo Real (0.00)
  dirY = 1;
  dirX = 1;
  vel = 5;
  velX = 1;
  dia = 10;

  padX = width/2;
  padY = height-50;
  padW = 150;
  padH = 10;
```

```
  frameRate(100);
}

void draw() {
  background(255);
  posY = posY + dirY * vel;
  posX = posX + dirX * velX;

  padX = mouseX;

  // Teste de Colisão com a tabela
  if (posY + dia/2 > height || posY < 0) {
    dirY = -dirY;
    //posX = int( random(width) );
  }
  // Teste de Colisão com o pad
  if (posY + dia/2 > padY && posY - dia/2 < padY+padH) {

    if (posX >= padX-padW/2 && posX <= padX + padW/2) {

      dirY = -dirY;

      // Calculo da velocidade de movimento do rato
      int tX = (pmouseX - mouseX) / 5;

      // influência da colisão para a direita ou para a esquerda da
paddle
      velX = (posX - padX)/5;

      // incremento de velocidade de movimento do rato
      velX = velX -(velX + tX);

    }
  }
  // Teste de colisão com as bordas da janela
  if (posX < 0 || posX > width) {
    dirX = -dirX;
    println("Janela");
  }

  // Teste de colisão horizontal com o Pad

  // Desenho do Pad
  noFill();
  rect(padX-padW/2, padY, padW, padH);
  // Desenho da Bola
  ellipse(posX, posY, dia, dia);
}


void mousePressed() {

  noLoop();


}
```

20b Pong Individual

Com efeito de tabela, definição de trajectória -lerp(), controlos - Biblioteca ControlP5 e Teste de colisão com Alvo -- dist()

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

import controlP5.*;

ControlP5 controlP5;
int sliderValue = 5;

int posY, posX, dirY, dirX, vel, velX, dia;
int padX, padY, padW, padH;
int alvoX, alvoY, alvoW;

boolean lup;

void setup() {
  size(500, 500);
  background(255);
  smooth();

  controlP5 = new ControlP5(this);
  //addSlider(theName, theMin, theMax, theDefaultValue, theX, theY,
theW, theH);
  //addSlider(theName, theMin, theMax, theX, theY, theWidth,
theHeight);
  Slider s = controlP5.addSlider("Velocidade",1,100,60,10,10,10,100);
  controlP5.setColorValue(color(0,60,90));
  controlP5.setColorLabel(color(0,60,90));

  alvoX = width/2;
  alvoY = 50;
  alvoW = 20;
  Slider t = controlP5.addSlider("Alvo",1,100,alvoW,80,10,10,100);

  posY = 0;
  posX = int( random(width) );
  dirY = 1;
  dirX = 1;
  vel = int(controlP5.controller("Velocidade").value())/10;
  velX = 1;
  dia = 10;

  padX = width/2;
  padY = height-50;
  padW = 150;
  padH = 10;
```

```
  frameRate(100);
}

void draw() {
  background(255);
  //vel = sliderValue;
  posY = posY + dirY * vel;
  posX = posX + dirX * velX;

  padX = mouseX;
  // Teste de COlisão com a tabela
  if (posY + dia/2 > height || posY < 0) {
    dirY = -dirY;
    //posX = int( random(width) );
  }
  // Teste de Colisão com o pad
  if (posY + dia/2 > padY && posY - dia/2 < padY+padH) {

    if (posX >= padX-padW/2 && posX <= padX + padW/2) {

      dirY = -dirY;

      // Calculo da velocidade de movimento do rato
      int tX = pmouseX - mouseX;

      // influência da colisão para a direita ou para a esquerda da
paddle
      velX = (posX-padX)/10 + tX/5;
      println("velX = "+velX);
      println("dirX = "+dirX);

      // incremento de velocidade de movimento do rato
      //velX = velX -(velX + tX);

    }
  }
  // Teste de colisão com as bordas da janela
  if (posX < 0 || posX > width) {
    dirX = -dirX;
    //println("Janela");
  }

  // Teste de colisão horizontal com o Pad

  // Desenho do Pad
  noStroke();
  fill(0,60,90);
  rect(padX-padW/2, padY, padW, padH);
  //Area de colisão para a esquerda
  fill(255, 0, 0, 80);
  rect(padX-padW/2, padY-3, padW/2, 3);
  //Area de colisão para a direita
  fill(0, 255, 0, 80);
  rect(padX, padY-3, padW/2, 3);
  // Meio do pad
  stroke(255, 255, 255, 127);
  strokeWeight(2);
  line(padX, padY, padX, padY+10);

  // Desenho da Bola
  fill(50, 50, 0);
```

```
  ellipseMode(CENTER);
  ellipse(posX, posY, dia, dia);

  // Desenho do Alvo
  noStroke();
  ellipseMode(RADIUS);
  fill(0,60,90,127);
  if (dist(posX, posY, alvoX, alvoY) < alvoW) {
    fill(255,0,0,127);
  }
  ellipse(alvoX, alvoY, alvoW, alvoW);

  // Desenho de trajectória
  stroke(0, 0, 0, 127);
  strokeWeight(3);
  float nt1 = dist(posX, posY, padX, padY)/10;
  for(int i=0; i<=nt1; i++) {
    float x = lerp(posX, padX, i/nt1);
    float y = lerp(posY, padY, i/nt1);
    point(x, y);
  }

  float nt2 = dist(posX, posY, alvoX, alvoY)/10;
  for(int i=0; i<=nt2; i++) {
    float x = lerp(posX, alvoX, i/nt2);
    float y = lerp(posY, alvoY, i/nt2);
    point(x, y);
  }

  stroke(255, 0, 0, 100);
  float nt3 = dist(posX, posY, alvoX, alvoY)/10;
  for(int i=0; i<=nt3; i++) {
    float x = lerp(padX, padX-(posX-padX), i/nt3);
    float y = lerp(padY, posY, i/nt3);
    point(x, y);
  }

  stroke(0, 0, 0, 40);
  float nt4 = dist(padX, padY, alvoX, alvoY)/30;
  for(int i=0; i<=nt4; i++) {
    float x = lerp(padX, alvoX, i/nt4);
    float y = lerp(padY, alvoY, i/nt4);
    point(x, y);
  }

  if (dist(mouseX, mouseY, alvoX, alvoY) < alvoW) {
    stroke(50, 50, 50, 127);
    line(alvoX, alvoY, mouseX, mouseY);
  }

}


void keyPressed() {
  switch(key) {
  case 'p':
    if (lup) {
      noLoop();
    }
    else {
      loop();
```

```
    }
    lup = !lup;
    break;
  case 'r':
    setup();
    break;

  }
}

void Velocidade(int p1) {
  vel = p1/10;
  //println("Velocidade = "+vel);
}

void Alvo(int p1) {
  alvoW = p1;
}
```

21 Sintaxe UI

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int a = 255;


void setup() {
  size(200, 200);
  background(a);

}

void draw() {
  background(a);

  ellipse(mouseX, mouseY, 10, 10);
}

void mousePressed() { // Callback automático pelo processing/Java/SO
  a = a-25;
}

void keyPressed() {
  switch(key) { // switch é tipo um interruptor --> Se for carregada
uma tecla, caso a tecla seja A = instruções, parar; caso seja B =
instruções, parar...
  case '+':
    a +=25;
```

```
    break;
  case'-':
    a -=25;
    break;
    case 'c':
    // ARROW, CROSS, HAND, MOVE, TEXT, WAIT
    cursor(HAND);
    break;
  }
}
```

22 Matemática

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// random devolve (armazena na variável) um numero real no intervalo
máximo pretendido
// random (200) = um numero qualquer entre 0 e 200
float a = random(200);
float b = random(200);

void setup() {
  size(300, 300);
  background(255);

}

void draw() {
  background(255);

  rect(0, 0, 200, 200);

  fill(255);
  ellipse(a, b, 10, 10);
  line(a, b, mouseX, mouseY);

  fill(0, 255, 0);
  ellipse(mouseX, mouseY, 10, 10);

  // constrain limita o numero actual num intervalo pretendido
  // constrain (mouseX, 0, 200) restringe qualquer que seja o número
do mouseX entre 0 e 200
  fill(255, 0, 0);
  ellipse(constrain(mouseX, 0, 200), constrain(mouseY, 0, 200), 10,
10);

  // dist calcula a hipotenusa (distancia absoluta) entre um ponto e
outro
  if (dist(mouseX, mouseY, a, b) < 50 ) {
    fill(150, 0, 0);
  }
```

```
  else {
    fill(255);
  }

}
```

22b Matemática - Função dist() - Colisões

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

int cX, cY, bgc, hitR;

void setup() {
  size(300, 300);
  background(255);

  cX = width/2;
  cY = height/2;

  bgc = 255;

  hitR = 100;

}

void draw() {
  background(bgc);

  ellipseMode(RADIUS);
  ellipse(cX, cY, hitR, hitR); // área de colisão/intersecção
  ellipse(cX, cY, 10, 10); // Centro
  line(cX, cY, mouseX, mouseY); // Distancia activa

  if ( dist( cX, cY, mouseX, mouseY ) < hitR ) { // função Dist
devolve a hipotenusa (distancia) entre 2 pontos num numero inteiro
(0.00), daí terem de o converter em int()
    bgc = 100;
  }
  else {
    bgc = 255;
  }

}
```

23 Trigonometria

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

// Basic Trig
float aorig = random(PI*1.5-PI/10, PI*1.5+PI/10);
float a = aorig;
float ainc = PI/100;

void setup() {
  size(500, 500);
  background(255);
  //smooth();
  frameRate(30);

}

void draw() {
  background(255);

  int x = mouseX;
  int y = mouseY;
  int cx = width/2;
  int cy = height/2;

  int h = 100;

  // Linha de calibracao por a = ang inicial
  stroke(255,0,0);
  line(cx, cy, int( h*cos(aorig) ) + cx, int( h*sin(aorig) ) + cy );

   a = a+ainc;

  // Linha segue rato
  fill(255, 0, 0);
  noStroke();
  ellipse(x, y, 5, 5);

  stroke(0);
  strokeWeight(0.2);
  line(cx, cy, x, y);

  // Linha auto h fixo
  int xnew = int( h*cos(a) ) + cx;
  int ynew = int( h*sin(a) ) + cy;
  fill(0, 0, 255);
  noStroke();
  ellipse(xnew, ynew, 5, 5);

  stroke(0);
  strokeWeight(0.2);
```

```
  line(cx, cy, xnew, ynew);

  // Linha auto h dinamico
  float anew = a+PI/2;
  int hnew = int( dist(cx, cy, x, y) );
  int xnewd = int( hnew*cos(anew) ) + cx;
  int ynewd = int( hnew*sin(anew) ) + cy;

  fill(0, 255, 0);
  noStroke();
  ellipse(xnewd, ynewd, 5, 5);

  stroke(0);
  strokeWeight(0.2);
  line(cx, cy, xnewd, ynewd);

  // 0, PI/2, PI, P*1.5, TWOPI
  strokeWeight(0.2);
  stroke(200, 200, 0);

  float aPI = PI*1.5;
  line(cx, cy, int( h*cos(aPI) ) + cx, int( h*sin(aPI) ) + cy );

}
```

23b Trigonometria (rotações simples e arrasto)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

int cX, cY, raio, eW;
float posX, posY, alfa;

void setup() {
  size(200, 200);
  background(0);

  cX = width/2;
  cY = height/2;

  // inicialização do angulo.
  // Vamos usar radianos. Medem-se de 0 a 2*PI (6.28...)
  // por isso 0 = 0 = 0; PI = 3.14 (radianos) = 180º; 2*PI, ou melhor
TWO_PI = 6.28 (radianos)= 360º
  alfa = 0;

  raio = 50;
  eW = 10;

  // Trigonometria
  // Calculo da posição através da hipotenusa
  // hipotenusa = cateto * cos (angulo);
```

```
  posX = cX + raio*cos(alfa);
  posY = cY + raio*sin(alfa);

  //noLoop();
}

void draw() {
  //background(0);

  // Para um efeito de trail / arrasto basta substituir a função
background que limpa o ecrã
  // Desenhar um quadrado no ecrã inteiro com alguma transparência
  fill(0, 0, 0,30);
  rect(0, 0, width, height);

  alfa = alfa+0.1; // incremento de angulo (radianos)

  fill(255);
  //ellipse(posX, posY, 10, 10);
  ellipse(posX, posY, eW, eW);

  posX = cX + raio*cos(alfa); // actualização da posição com o novo
angulo
  posY = cY + raio*sin(alfa);

  eW = mouseY/5;
  raio = mouseX/2;
}
```

24 Transformação

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

float a = 0.0;

void setup() {
  size(300, 300);

}

void draw(){
  a = a+0.01;
  background(255);

  // Mover a origem da matriz de localização de pontos da janela
  translate(width/2, height/2);
  ellipse(0, 0, 10, 10);

  // Rodar a matriz
  rotate(a);
```

```
  rectMode(CENTER);
  rect(0, 0, 50, 5);


}
```

24b Transformação

```
// Exemplo do Processing.org

float a = 0.0;

void setup() {
  size(300, 300);
}

void draw(){
  background(255);
  fill(255);
  rect(0, 0, 50, 50);   //White rectangle
  pushMatrix();
  translate(30, 20);

  ellipse(width/2, height/2, 10, 10);
  line(width/2, height/2, mouseX, mouseY);

  rotate( atan2(mouseY-height/2, mouseX-width/2) );
  fill(0);
  rect(0, 0, 50, 50);   //Black rectangle
  popMatrix();
  fill(102);
  rect(15, 10, 50, 50); //Gray rectangle
}
```

25 Acções Gatilhos e Eventos

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

void setup() {
  size (300, 300);

}

void draw() {
```

```
  background(255);

  ellipse(width/2, height/2, 10, 10);

  fill(255);

  // verificar um evento durante o draw sem recorrer à invocação do
callback
  if (mousePressed) {
    fill(255, 0, 0);
    }

}
```

26 Vectores (Arrays)

```
// © Pedro Amado, 2008-03-20

// Declaração de um vector, de uma lista de números inteiros
int vector[];

int count;
int n;

void setup() {
  size(300, 300);
  count = 3;
  // Inicialização / Construção do vector através do contrutor 'new'
  vector = new int[count];
  n = 0;

}

void draw(){
  background(255);

  // Iniclialização de cada posição da lista
  // para aceder aos valores da lista invocamos o vector pelo nome
seguido de '[ ]'
  // introduzindo o indice da posição que queremos ler / escrever
  // line [ 0 ] = 10; quer dizer a primeira posição (0) do vector
line = int 10;
  // para usar basta chamar pela posição p. ex.: point(line[0], 10);
  for(int i = 0; i < count; i++) {
    line(vector[i], 0, vector[i], height);
  }

}

void mousePressed(){
  vector[n] = mouseX; // inserir na posiçao do indice n do vector
  n++;
```

```
  // % = módulo --> calcula o resto de uma divisão pelo número
expresso
  // por exemplo, 13%10 vai dar um resultado = 3
  n = n%count;
}
```

26b Vectores

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

int vector[];
int count;
int n;

void setup() {
  size(300, 300);
  count = 3;
  vector = new int[count];
  n = 0;
  println(vector.length);
}

void draw(){
  background(255);

  for(int i = 0; i < count; i++) {
    line(vector[i], 0, vector[i], height);
  }

}

void mousePressed(){
  vector[n] = mouseX;
  n++;
  if (n>=count) {
    count = count*2;
    vector = expand(vector); // duplica o tamanho da lista
    println(vector.length); // .length acede a uma propriedade da
lista que devolve o numero de elementos da lista
    }
  }
```

27 Funções e Parametros

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-20

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


void draw() {
  background (0);
}

// Declaração da função --> não é "void" mas sim "int ...(...) {"
porque vai calcular e DEVOLVER um numero int
// tipo nome (parametro, parametro,...) {
int soma(int p1, int p2) {
  int a = p1+p2;
  return a; // Devolve o número
}

void mousePressed() {
  println("Mouse X = "+mouseX);
  println("Mouse Y = "+mouseY);
  println("A calcular a soma dos dois... \n");
  println( soma(mouseX, mouseY) ); // Invoca  a função e passa os
parametros necessários
}
```

28 Classes e Objectos (Noção e Introdução)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-21

// Declaração do(s) objecto(s) a usar
obj o1, o2, o3;

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

  // Inicialização / construção do objecto através do construtr 'new'
e passagem de parametros
  o1 = new obj( int(random(200)), int(random(200)), int(random(20))
);
  o2 = new obj( int(random(200)), int(random(200)), int(random(20))
);
  o3 = new obj( int(random(200)), int(random(200)), int(random(20))
```

```
);
}

void draw() {
  background (0);

  // invocar funções do interior do objecto através da estrutura '.'
  o1.display();
  o2.display();
  o3.display();
}

void mousePressed() {
  // invocar funções do interior do objecto através da estrutura '.'
  o1.update();
  o2.update();
}

// como se fosse um novo applet
class obj {
  // Globais
  int locx, locy, siz;

  // Setup tem o mesmo nome que a função
  obj(int p1, int p2, int p3) {

    locx = p1;
    locy = p2;
    siz = p3;

    ellipseMode(CENTER_RADIUS);

  }
  //Metodos
  void update() {

    locx = int(random(200));
    locy = int(random(200));
    siz = int(random(20));

  }

  void display() {
    ellipse(locx, locy, siz, siz);
    }
}
```

28b Classes e Objectos (Construção de Classes)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23
```

```
import processing.opengl.*; // Declarar e importar a biblioteca de
interface com a Placa Gráfica para dentro do Skecth

Boid b1, b2; // Declarar o uso de dois (b1 e b2) Objectos / Classe
Boids (construidas por nós)
Boid[] boids; // Declarar o uso de um vector de Boids (lista de
objectos)

int[][] mLoc;
int n, nClick;

void setup() {
  size (500, 500, OPENGL); // Declarar o uso da Biblioteca OPENGL no
rendering do Skecth - Acelera consideravelmente a velocidade de
processamento, mas tem impicações de visualização e
    background(255);
  fill(0);
  stroke(0);

  n=10;
  nClick = 0;

  // b1 = new Boid(); // Inicializar / criar uma instância da classe
Boid através do construtor new e invocando o nome da Classe com o
mesmo nome da primeira função (setup da classe)

    boids = new Boid[1000]; // inicializar a lista de Boids com 1000
posições na lista ( 1000 Boids potenciais)

  mLoc = new int[2][n];

  for(int i = 0; i < n; i++){
    boids[i] = new Boid(); // Criar (instâncias de) o objecto Boid
através do construtor new e invocando o nome da classe - tem que ser
igual ao da função de iicialização / setup da classe

      int x = int( random(width) );
    int y = int( random(height) );

    //b1.update(x, y);// invocar uma função (de construção) do
interior da classe Boid através da estrutura de acesso '.'
    //b2.update(x, y);

    boids[i].update(x, y); // invocar uma função (de construção) do
interior da classe Boid através da estrutura de acesso '.'
  }


}

void draw() {
  background(255);

  /*
  if (n>1){
    for(int j = 0; j < n; j++){
      for(int k = 0; k < n; k++) {
        line(mLoc[0][j], mLoc[1][j], mLoc[0][k], mLoc[1][k]);
      }
    }
  }
```

```
  */

  // b1.oscilate(); // invocar uma função (de actualização) do
interior da classe Boid através da estrutura de acesso '.'
  // b1.display();

  for(int i = 0; i < nClick; i++){

    boids[i].oscilate(); // invocar uma função (de actualização) do
interior da classe Boid através da estrutura de acesso '.'
    boids[i].display();  // invocar uma função (de desenho) do
interior da classe Boid através da estrutura de acesso '.'

  }

}


void mousePressed() {

  if (nClick >= n) {
    n = n*2;
    mLoc[0] = expand(mLoc[0]);
    mLoc[1] = expand(mLoc[1]);
    //boids = expand(boids);
    println(mLoc[0].length);
    println("duplica");

  }

  mLoc[0][nClick] = mouseX;
  mLoc[1][nClick] = mouseY;

  boids[nClick] = new Boid();

  int x = mouseX;
  int y = mouseY;

  boids[nClick].update(x, y);

  nClick++;
}
```

Segundo Tab

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

// Este objecto está num TAB diferente, para facilidade de leitura.
// Todos os ficheiros .pde na mesma pasta, vão fazer parte da mesma
aplicação final
// Como se estivessem todos escritos dentro do mesmo ficheiro
```

```
import net.fladdict.oscillator.*; // Declarar e importar a biblioteca
(externa Oscilator)

class Boid { // Nome da Classe

  // Declaração das variáveis globais (propriedades) de cada
instancia a criar
  int mX, mY;
  float rW;

  Oscillator osc = new OscSin(int( random(30) ),int( random(100) ),
int( random(50) ));

  void Boid() {
    mX = 0;
    mY = 0;
    rW = 0.0;
  }

  void update(int p1, int p2){
    mX = p1;
    mY = p2;
    rW = 10;
  }

  void display() {
    ellipse(mX, mY, rW, rW);
  }

  void oscilate(){
      osc.update();
      rW = osc.getValue();
  }

}
```

29 Texto

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-21

// Cria um objecto de fonte
PFont fgeorgia;

void setup() {
  size(200, 200);
  background(0);
  fill(200);
  //noLoop();

  // Invoca / load Fonte
  fgeorgia = loadFont("Georgia-48.vlw");
```

```
  textAlign(LEFT);
}

void draw() {
  fill(0, 23);
  rect(0, 0, width, height);

  // Definição do estilo de texto a usar
  textFont(fgeorgia);
  textSize(20);

  fill(255);

  // Escrever texto, "que texto", onde X e onde Y
  text("Texto", mouseX, mouseY);
}
```

29b Texto (optimização de tamanhos)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

// Declarar o uso de um (ou mais) Objecto de Texto
PFont fonte, fonte2;


void setup() {
  size(300, 300);
  background(0);

  // Criar o ficheiro fonte de Processing (*.vlw) através do Menu
Tools > Creat Font...
  // O ficheiro de fonte *.vlw fica guardado numa pasta "Data" ao
lado do Sketch no disco duro
  // Verificar a existencia do ficheiro *.vlw dentro da pasta "Data"
através do menu Sketch > Show Sketch Folder (CTRL+K)

// Carregar a fonte para dentro de uma variável
  fonte = loadFont("Garamond-48.vlw"); // criar diferentes tamanhos,
optimizados para diferentes resoluções
  fonte2 = loadFont("Garamond-15.vlw");

  fill(255);

}

void draw() {
  background(0);

  textFont(fonte); // definição de estilo / formatação de texto
  textSize(48);
  text("Mensagem", 10, 50); // escrever para o ecrã
```

```
  textSize(15);
  text("Outra Mensagem", 10, 110);

  // COmparar uma fonte optimizada para um tamanho superior e
apresentada com um tamanho inferior, com outra optimizada para o
tamanho final
  textFont(fonte2);
  text("optimizado? Mensagem", mouseX, mouseY);
}
```

30 Imagens

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-21

// Declaração do Objecto de Imagem a usar
PImage img;

// Importar bibliotacas JAVA
// Hack para fazer Fullscreen
import java.awt.*;
import java.awt.event.*;


void setup() {
  size(screen.width, screen.height); // atributo especial de leitura
da resolução do ecrã

  // Carregar a imagem do disco para a variável de imagem
  img = loadImage("cursor.png");
  noCursor();


}

void draw() {

  background(255);

  // Desenhar / Usar a imagem
  image(img, mouseX-img.width/2, mouseY-img.height/2);

  // Hack para fazer Fullscreen
  frame.setLocation(0,0);
}

void mousePressed() {
  tint( random(5)*50, random(5)*50, random(5)*50 );
}

// Hack para fazer Fullscreen
```

```
public void init() {
  frame.setUndecorated(true); // works.

  // call PApplet.init() to take care of business
  super.init();
}

// Hack para fazer Fullscreen
// http://workshop.evolutionzone.com/2007/01/10/code-
framesetundecoratedtrue/
```

31 Cálculo de Média (Consola)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2006-11-20

// Declarar variáveis
int tiago = 19; // declarar e inicializar
int maria = 27;
int ana = 23;
int soma; // declarar

void setup() {
  size(500, 300);
  background(0);
  soma = 0;//inicializar a soma
  soma = tiago + maria + ana;// Somar as idades todas;
  int nidades = 3;//total de idades
  float media = soma/nidades;// Dividir o resultado pelo nœmero total
de idades;

  // Passar os resultados ao utilizador
  println("Total de idades ="+ soma);
  println("Nœmero de idades somadas = "+ nidades);
  println("--------------------------------");
  println("MŽdia das idades = " + media);
}

void draw() {

}
```

32 Cálculo de Média (Primitivas Gráficas)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2006-11-20

// Declarar variáveis
int tiago = 19; // declarar e inicializar
int maria = 27;
int ana = 23;
int soma; // declarar

void setup() {
  size(500, 300);
  background(0);
  soma = 0;//inicializar a soma
  soma = tiago + maria + ana;// Somar as idades todas;
  int nidades = 3;//total de idades
  float media = soma/nidades;// Dividir o resultado pelo número total
de idades;

  // Passar os resultados ao utilizador
  println("Total de idades ="+ soma);
  println("Nœmero de idades somadas = "+ nidades);
  println("-------------------------------");
  println("MŽdia das idades = " + media);
}

void draw() {
  // Atribuir-lhe um cor e/ou linha;
  stroke(255);
  fill(100,100, 100, 150);
  // Representação de cada idade por uma primitiva ( rect() )
  rectMode(CENTER);
  rect(tiago*10, 150, 5, 100);
  rect(maria*10, 150, 5, 100);
  rect(ana*10, 150, ana+5, 100);

  // Representação do valor médio encontrado atravŽs de uma primitiva
de outra cor;
  int nidades = 3;//total de idades
  float media = soma/nidades;// Dividir o resultado pelo nœmero total
de idades;
  stroke(255, 0, 0);
  line(media*10, 100, media*10, 200);

  // Representação da própria idade através de uma primitiva e de um
texto diferente diferente;
  ellipseMode(CENTER);
  ellipse(tiago*10, 150, 10, 10);
}
```

33 Cálculo de Média (Imagens, Texto e Primitivas)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2006-11-20

// Declarar variáveis:
//   Soma e media de todas as idades;
int soma, media, total;

//   Vector de todas as idades;
int[] idades = {19, 21, 12, 22, 30, 17, 18, 45, 23, 23};

//   Vector de todos os nomes;
String[] nomes = {"Ana", "Tiago", "Pedro", "Maria", "Susana",
"Silva", "Filipa", "Cristina", "Paulo", "Ricardo"};

//   Meu nome/idade (posi?‹o nos vectores)
int meunome;

//   Data a usar
//   Nome da aplicação a usar;
String data, nome;

//   Imagem a usar de fundo;
PImage bg;

//Texto a desenhar em ecrã
PFont texto;

void setup() {
  size(500, 300);
  background(255);

  //inicializar variáveis;
  soma = 0;
  media = 0;
  total = idades.length;
  meunome = 5;
  nome = "Aplicação Gráfica de Médias";
  int dia = day();
  int mes = month();
  int ano = year();
  data = "Porto, "+ ano +"-"+ mes +"-"+ dia;
  //println(nomes[meunome]+idades[meunome]);
  //println(data);

  // Calcular e armazenar em memoria (ciclos);
  for (int n = 0; n     soma = soma+idades[n];
  }

  media = soma/total;
  //println(media);

  // Carregar a imagem
  bg = loadImage("background_2.png");

// Carregar a fonte
  texto = loadFont("MyriadPro-Regular-14.vlw");
  textFont(texto, 14);
```

```
}

void draw() {
  // Desenhar a imagem de fundo;
  image(bg, 0, 0);

  // Desenhar o Nome e Data da Aplicação;
  smooth();
  fill(255);
  text(("Pedro Amado - "+data), 10, 28);

  // Representar as idades todas e média (ciclos);
  //   Desenho de primitivas;
  //   área de gráficos
  noStroke();
  rectMode(CORNER);
  fill(255, 100);
  rect(0, 200, 500, 100);

  //   Desenho das idades
  rectMode(CENTER);
  fill(255, 100);
  noStroke();

  for (int n = 0; n     //   Verificação da idade ou nome proprio
para o representar de forma diferente (condições)
    if (n == meunome) {
      fill(255, 255, 0, 100);
      rect(idades[n]*10, 250, 5, 100);
    }
    else {
      fill(255, 100);
      rect(idades[n]*10, 250, 5, 100);
    }
    //   Atribuição de nome a cada idade representada;
    fill(255);
    text(nomes[n], idades[n]*10, 200+n*10);
  }

  //    Desenho da média
  //   Obter o resultado  mŽdio da soma de todos os elementos;
  //   Desenho da média (primitivas);
  stroke(255, 0, 0);
  strokeWeight(2);
  strokeCap(SQUARE);
  line(media*10, 200, media*10, 300);

  //   Atribuiçãoo de nome da média;
  fill(255,0,0);
  text("MŽdia de idades", media*10, 250);
}
```

34a Applicação de desenho com memória

(ligações entre clicks)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

int nMax, n;

// Matriz de dados bi-dimensional
// Lista de listas de numeros inteiros
int[][] mLoc;
float rW;

void setup() {
  size(500, 500);
  background(255);
  nMax = 5;
  n = 0;

  // Inicializar / construir a lista das listas.
  // Duas linhas, cada linha com nMax (5) colunas
  // Como se fosse uma tabela de Excel
  mLoc = new int[2][nMax];
}

void draw() {

  if (n != 0 ) {
    //println("desenho");

    // inicializar as posições de cada celula das linhas da lista
(metáfora Excel)
    // Por exemplo, primeira linha (índice 0), Quinta célula (índice
4)
    // mLoc[0][4]
    for (int i = 0; i < n; i++) {
      rect(mLoc[0][i], mLoc[1][i], 10, 10);
    }

    if (n>1){
      for(int j = 0; j < n; j++){
        for(int k = 0; k < n; k++) {
          line(mLoc[0][j], mLoc[1][j], mLoc[0][k], mLoc[1][k]);
        }
      }
    }
  }

}

void mousePressed(){
```

```
  if (n >= nMax) {
    nMax = nMax*2;
    mLoc[0] = expand(mLoc[0]);
    mLoc[1] = expand(mLoc[1]);
    println(mLoc[0].length);
    println("duplica");

  }

  mLoc[0][n] = mouseX;
  mLoc[1][n] = mouseY;

  n++;
}
```

35 Field of Flowers 1000

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

// Declaração do Objecto de Flores
Flower f1;

// Declaração de um objecto (que contém outros objectos)
Field campo;

void setup() {
  size(500, 500);
  background(255);
  //f1 = new Flower(3, 100, 3, color(255, 0, 0), width/2, height/2);
  campo = new Field();
}

void draw() {
  background(255);
  // invocar as funções no interior da classe field
  campo.display();
}



void keyPressed() {
  switch (key) {
  case 'r':
    setup();
    break;

  }
}

void mousePressed() {
  //f1.update(mouseX, mouseY);
```

```
  campo.add();


  }
```

Segundo TAB (Classe de Classes)

```
class Field {
  int num;

  Flower[] campo;

  Field () {
    num = 0;
    campo = new Flower[1000];
    campo[0] = new Flower(3, 100, 3, color(255, 0, 0), width/2,
height/2);
  }

  void display() {
    //campo[0].display();

    for (int i = 0; i <= num; i++){
      campo[i].display();

      /*
      if (i>0) {
       int[] locA = campo[i].getLoc();
       int[] locB = campo[i-1].getLoc();

       stroke(0,0,0,127);
       strokeWeight(1);
       line(locA[0], locA[1], locB[0], locB[1]);
       }
       */

    }

    int[] locX = new int[num];
    int[] locY = new int[num];

    for (int i = 0; i < num; i++) {
      locX[i] = campo[i].getX();
      locY[i] = campo[i].getY();
    }

    for (int i = 0; i <= num; i++) {
      if(i>0){
        for (int j = 0; j < i; j++) {
          stroke(0, 0, 0, 30);
          strokeWeight(1);
          line(locX[i-1], locY[i-1], locX[j], locY[j]);
          }
        }
```

```
    }

  }

  // campo.update(3, 100, 3, color(255, 0, 0), width/2, height/2);
  void update(int p1, int p2, int p3, color p4, int p5, int p6) {


  }

  void add() {
    num++;
    println(num);
    campo[num] = new Flower(3, 100, 3, color(255, 0, 0), mouseX,
mouseY);
  }


}
```

Terceiro TAB (Classe/objecto)

```
class Flower {

  int nCirc;
  float raioMax, espMax;
  color cor;
  int locX, locY;
  float[][] petalas;

  Flower(int p1, int p2, int p3, color p4, int p5, int p6) {
    nCirc = p1;
    raioMax = p2;
    espMax = p3;
    cor = p4;
    locX = p5;
    locY = p6;
    petalas = new float[p1][3];

    update();
  }

  void update() {


    for (int i = 0; i < nCirc; i++) {
      petalas[i][0] = random(nCirc);
      petalas[i][1] = random(raioMax);
      petalas[i][2] = random(espMax);
    }
  }

  void display() {
```

```
    for (int i = 0; i < petalas[0][0]; i++) {
      noFill();
      stroke(petalas[i][2]);
      strokeWeight(petalas[i][2]);
      ellipse(locX, locY, petalas[i][1], petalas[i][1]);
    }

  }

  int[] getLoc() {
    int[] locs = {
      locX, locY    };
    return locs;
  }

  int getX() {
    int x = locX;
    return x;
  }

  int getY() {
    int y = locY;
    return y;
  }
}
```

36 Bibliotecas - ControlP5

(Biblioteca externa ao processing.org de criação de GUI) e PDF (nativa)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

import processing.pdf.*; //Declarar e importar a Bibloteca para o
Sketch

import controlP5.*; //Declarar e importar a Bibloteca para o Sketch


ControlP5 cp5; // declarar um Objecto de Biblioteca a usar no Sketch

int R, G, B;
boolean pdf;

void setup() {
  size(200, 200);
  R = 100;
  G = 100;
  B = 100;
```

```
  cp5 = new ControlP5(this); // Criar e inicializar o objecto de
Biblioteca
  Slider s1 = cp5.addSlider("R",0,255,100,  10,10,10,100); // aceder
às propriedades do Objecto CP5 inicilaizado, e criar um slider
invocando uma função de criação do interior do Objecto CP5 e atribui-
lo a uma variável
  Slider s2 = cp5.addSlider("G",0,255,100,  25,10,10,100);
  Slider s3 = cp5.addSlider("B",0,255,100,  40,10,10,100);
  pdf = false;

}

void draw() {
  if (pdf) {
    beginRecord(PDF, "printScreen-####.pdf"); // fazer uso da função
beginRecord (introduzida pela biblioteca PDF) e activar a gravação
para um ficheiro externo *.pdf
  }
  background(0);
  fill(R, G, B);
  ellipse(width/2, height/2, 75, 75);

  if (pdf) {
    pdf = false;
    endRecord();
  }

}

void keyPressed() {
  pdf = true;
  }
```

37 Bibliotecas - jMyron (Camera Track Color)

```
// Exercício para a formação Introdução à Programação Grafica
// Audiência Zero, Porto 2008-04-22 e 23
// © Pedro Amado, 2008-03-23

import JMyron.*;

JMyron m;//a camera object

void setup() {
  size(320, 240);

  m = new JMyron();//make a new instance of the object
  m.start(width,height);//start a capture at 320x240
  //m.trackColor(255,255,255,256*3-100);//track white
  m.trackColor(0,0,0,127); // track Black, com a tolerância a 50%
  m.update();

  background(0);
```

```
}

void draw(){

  m.update();//update the camera view

  int[] img = m.image(); //get the normal image of the camera

  loadPixels();
  for(int i=0;i<width*height;i++){ //loop through all the pixels
    pixels[i] = img[i]; //draw each pixel to the screen
  }
  updatePixels();

  int[][] centers = m.globCenters();//get the center points
  //draw all the dots while calculating the average.
  float avX=0;
  float avY=0;

  if (centers.length > 0) {
    rect(centers[0][0],centers[0][1],5,5);
  }

  /*
  for(int i=0;i
   fill(80);
   rect(centers[0][0],centers[0][1],5,5);
   avX += centers[0][0];
   avY += centers[0][1];
   }
   */
}

void mousePressed(){
  m.settings();//click the window to get the settings
}

public void stop(){
  m.stop();//stop the object
  super.stop();
}
```