

Introdução à Programação Gráfica com Processing

Notas para a Formação @ Audiência Zero

Pedro Amado

Porto, 2008-03-22



Este trabalho está licenciado sob uma Licença Creative Commons Atribuição-Usó Não-Comercial-Partilha nos termos da mesma Licença 2.5 Portugal. Para ver uma cópia desta licença, visite <http://creativecommons.org/licenses/by-nc-sa/2.5/pt/> ou envie uma carta para Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Qualquer correção ou sugestão:

pedamado@gmail.com

Para mais informações e slides da sessão:

<http://pedamado.googlepages.com>

<http://pedamado.wordpress.com/2008/03/22/processing-audiencia-zero/>

Manual da formação original disponível aqui:

<http://www.box.net/shared/n3ew1kqskc>

Ficheiros e Exemplos:

<http://www.box.net/shared/forhr9ows8>

Resumo

As presentes notas servem de apoio às sessões (2) de formação em **Introdução à programação Gráfica** com Processing na Audiência Zero.

Esta não é uma sessão de aprendizagem em programação!

A formação consiste:

- Elementos fundamentais de programação,
- Exposição de exemplos de artistas conceituados
- Desafios para criar aplicações interactivas com um grau sucessivo de independência
- Uma aplicação gráfica interactiva

Dia 1

Apresentação

Pedro Amado

Mestre em Arte Multimédia (2007)

Licenciatura em Design de Comunicação (2002).

Assistente Convidado no Departamento de Arte e Comunicação (DeCA) da Universidade de Aveiro (UA) onde lecciona Multimédia.

Técnico Superior de Design na Faculdade de Belas Artes da Universidade do Porto (FBAUP 2003-2007)

Introdução

Público-Alvo

Alunos de Design de Comunicação (Designers);

Alunos de Artes Plásticas (Artistas);

Pessoas interessadas em Multimédia Digital (Toda a gente).

Requisitos?

Curiosidade!

Porquê programar?

Um programa é um jogo de algoritmos construídos para resolver ou simplificar um problema.

Because an artist is expected to understand his/her medium.

Because this is true digital design.

Porquê o Processing?

1. Open Source
2. Versão Beta (ainda) o que o torna ideal
3. Multi-plataforma. MacOS, Win, Unix.
4. Poder do JAVA
 - a. Simples*
 - b. Adequa-se ao processo de ensino*
 - c. Faz uma boa transição entre linguagens de baixo nível e scripting de muito alto nível;
 - d. Documentação extensa.
Há diversos websites dedicados à programação, especialmente em JAVA e muitos dedicados ao Processing – ver a secção de links;
 - e. (muito) Extensível
5. Exporta Executáveis e Applets para a Web
6. Comunidade

Como obter e usar: Download & Install



Site:

<http://www.processing.org/>

Download:

<http://www.processing.org/download/index.html>

Documentação:

<http://www.processing.org/reference/index.html>

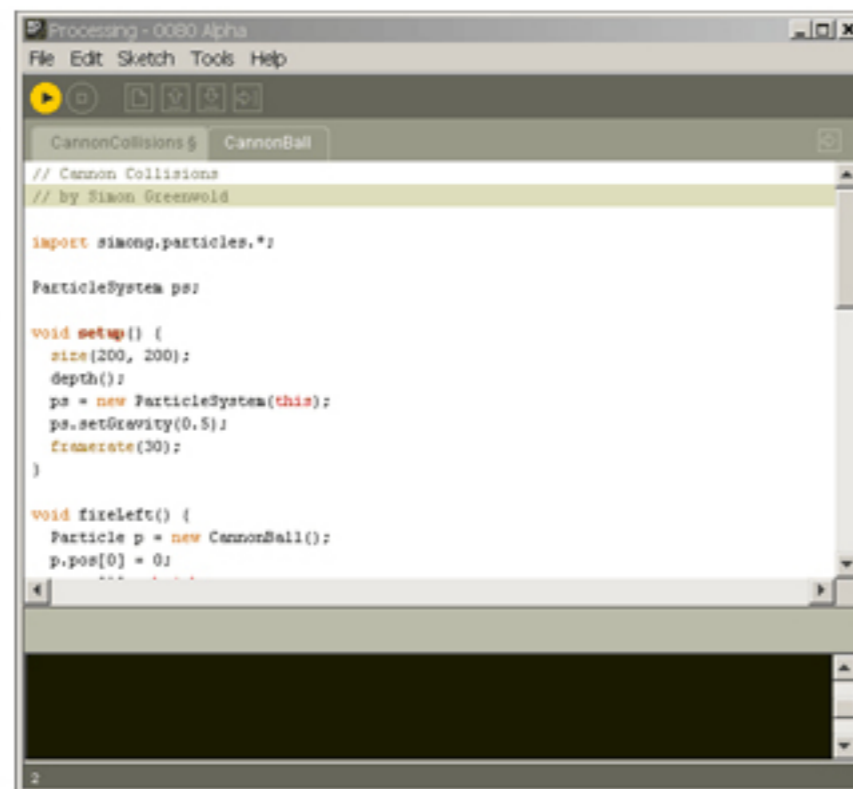
Aprendizagem:

<http://www.processing.org/learning/index.html>

IDE(PDE)



Display Window



```
Processing - 0090 Alpha
File Edit Sketch Tools Help
CannonCollisions CannonBall
// Cannon Collisions
// by Simon Greenwald
import simong.particles.*;
ParticleSystem ps;
void setup() {
  size(200, 200);
  depth();
  ps = new ParticleSystem(this);
  ps.setGravity(0.5);
  framerate(30);
}
void fireleft() {
  Particle p = new CannonBall();
  p.pos[0] = 0;
  ...
}
```

Menu
Toolbar
Tabs

Text Editor

Message Area

Text Area

App01, Hello World!

Do Algoritmo à arte (conceptual) e vice-versa. O desenho como um processo racional

Concept art is a form of illustration where the main goal is to convey a visual representation of a design, idea, and/or mood for use in movies, video games, or comic books before it is put into the final product.

The idea becomes a machine that makes the art. – Sol LeWitt, “Paragraphs on Conceptual Art”, Art Forum, 1967.

Exemplos

Ben Fry

Anemone, 2005

Valence, 1999

Casey Reas

Ti, 2004

Process 4, 2005

Articulate, 2004

Martin Wattenberg

Thinking Machines 4, 2001-04

Alternativas ao Processing

Flash

<http://www.adobe.com/>

NodeBox

<http://nodebox.net/code/index.php/Home>

Scriptographer

<http://www.scriptographer.com/>

Max/MSP

<http://www.cycling74.com/products/maxmsp>

PureData

<http://puredata.info/>

SmallTalk / Squeak

<http://www.squeak.org/>

Flex

<http://flex.org/> | <http://labs.adobe.com/technologies/flex/>

MS Silverlight

<http://silverlight.net/>

JAVA

<http://java.sun.com/>

OpenFrameworks

<http://openframeworks.cc/>

(C++ <http://www.cplusplus.com/>)

Conceitos Gerais de Programação

Algoritmia

Um algoritmo pode ser descrito usando várias linguagens ou línguas:

- Português;
- Pseudo-código, é uma linguagem entre a linguagem natural e uma linguagem de programação;
- Fluxogramas, descrição gráfica de um algoritmo;
- Linguagem de Programação

Lógica e Sintaxe

Um programa pode ser analisado segundo duas perspectivas:

- Sintaxe: o código está de acordo com as regras gramaticais da linguagem de programação utilizada?
- Lógica: o código executa aquilo que nós pretendemos?

Um programa pode estar sintacticamente correcto (regra geral o compilador detecta estes erros), mas logicamente estar errado.

Nível de detalhe...

Comentários e Documentação

Sintaxe introduzida: `//`, `/* */`



App02, Comentários

Expressões, instruções e terminadores

Sintaxe introduzida: `;`

Sensibilidade à caixa (Mm)

O Processing é sensível ao uso de maiúsculas e de minúsculas. Escrever “minúsculas” NÃO é a mesma coisa que escrever “MINUSCULAS”.

Espaço em branco

O Processing é insensível ao espaço em branco entre funções, expressões e/ou literais.

Funções e callbacks

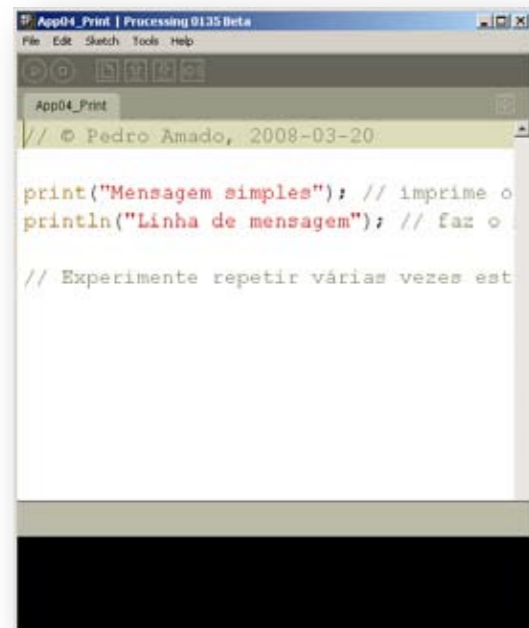
Sintaxe introduzida: `() {}`

Conceitos introduzidos: Função, Parâmetros, Índice de acesso, Bloco de código

Consola e mensagens

Sintaxe introduzida: `print()` , `println()` ;

Conceitos introduzidos: Consola, Debug, Função, Mensagem

A screenshot of the Processing IDE window titled 'App04_Print | Processing 0135 Beta'. The code editor shows the following text:

```
// © Pedro Amado, 2008-03-20

print("Mensagem simples"); // imprime o
println("Linha de mensagem"); // faz o

// Experimente repetir várias vezes est
```

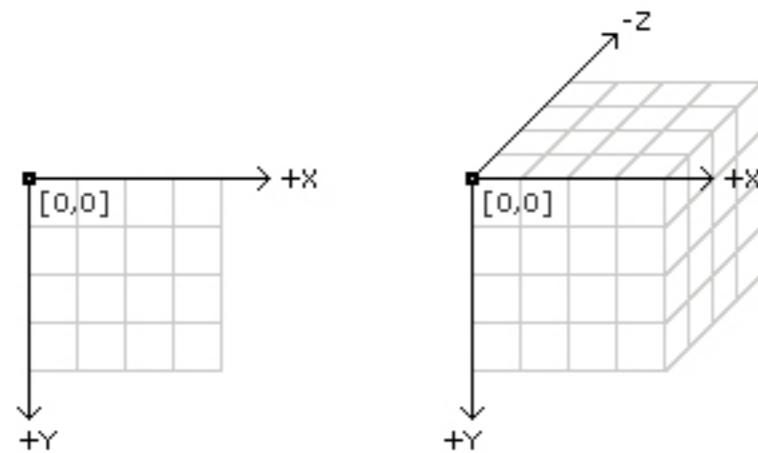
App04, Print

Dados/Literais

(Inserir diagrama dos elementos do código)

Coordenadas

Sintaxe introduzida: `size()` ;



Processing uses a Cartesian coordinate system with the origin in the upper-left corner. If your program is 320 pixels wide and 240 pixels high, coordinate $[0, 0]$ is the upper-left pixel and coordinate $[320, 240]$ is in the lower-right. The last visible pixel in the lower-right corner of the screen is at position $[319, 239]$ because pixels are drawn to the right and below the coordinate.

Primitivas

Sintaxe introduzida:

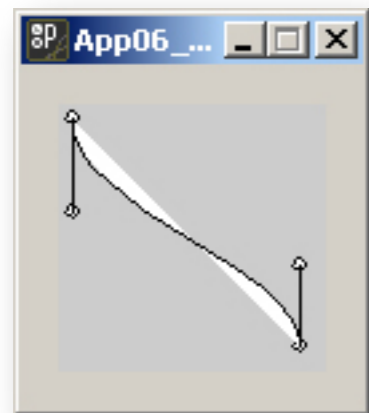
`point()` , `line()` , `rect()` , `ellipse()` , `bezier()`



App05, Primitivas

Sintaxe Extra:

`quad()` , `curve()` , `rectMode()` , `ellipseMode()` ,
`curveVertex()` , `bezierVertex()` ;



App06, Bezier

Cor

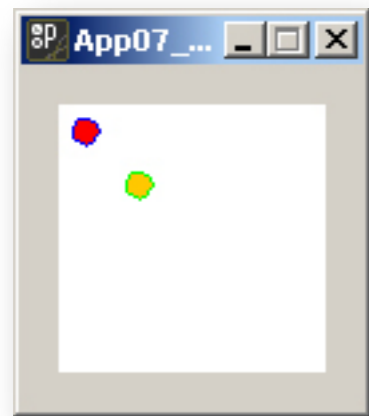
Sintaxe introduzida:

`background()` , `fill()` , `stroke()` , `noFill()` ,
`noStroke()`

Sintaxe Extra:

`color()`

Conceitos: RGB+A (Alpha)



App07, Cor

Atributos/Modos de Desenho

Sintaxe introduzida:

`smooth()` , `strokeWeight()` , `strokeCap()` ,

Sintaxe Extra:

`ellipseMode()` , `rectMode()`



App08, Atributos

Variáveis e Tipos de Dados

Todos os dados armazenados em memória e ou manipulados pelo computador tem uma natureza específica.

Os valores/variáveis utilizadas nas operações têm de ser compatíveis – Têm de ser do mesmo tipo

Simple

Tipos simples são tipos cujos valores são atômicos, i.e., não podem ser decompostos

Numérico

- Inteiro

- Real

Alfanumérico (cadeia de caracteres ou *string*)

“1” (*string*) é diferente de 1 (número)

Lógico (verdadeiro ou falso - *boolean*)

Compostos(Complexos)*

Tipos complexos são tipos compostos por vários elementos simples:

Vector (*Array* em inglês – não confundir com a palavra inglesa *vector*): uma lista de elementos do mesmo tipo que podem ser acessados via um índice.

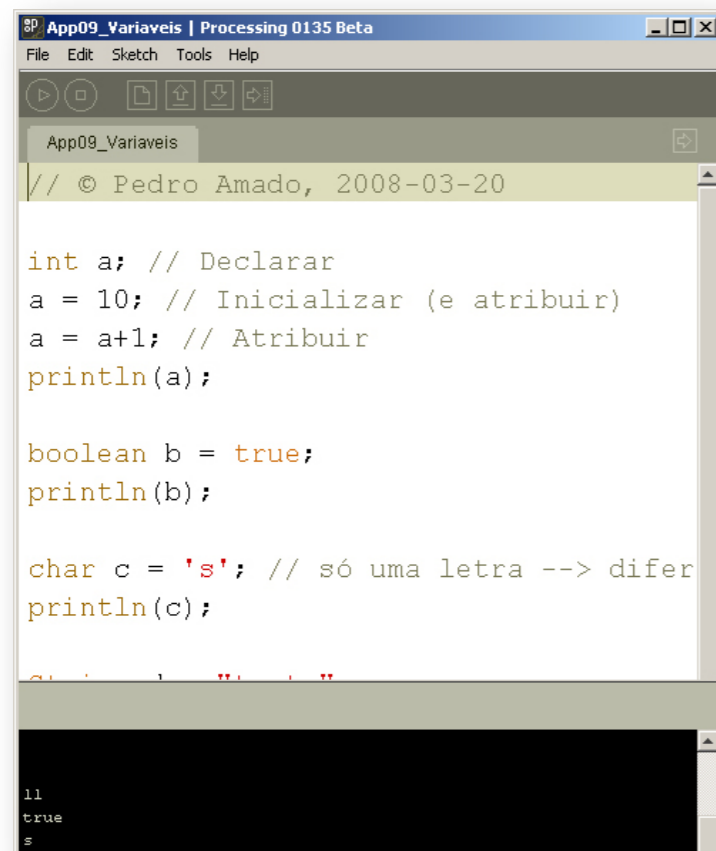
Matriz: vector multi-dimensional;

Estrutura (*Struct*): agregação de vários tipos de dados;

Tipos de variáveis

Sintaxe introduzida:

int, float, boolean, char, String*



```

App09_Variaveis | Processing 0135 Beta
File Edit Sketch Tools Help
App09_Variaveis
// © Pedro Amado, 2008-03-20

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

boolean b = true;
println(b);

char c = 's'; // só uma letra --> difer
println(c);

11
true
#
  
```

App09, Variaveis

Declarar e inicializar (Strict Typing)

Sintaxe introduzida: **= //atribuir**

Visibilidade (Scope)*

Conceitos introduzidos: Global e Local

Variáveis do Processing*

Sintaxe introduzida: **width, height**

Conversão de dados*

Sintaxe introduzida:

`boolean()` , `byte()` , `char()` , `int()` , `float()` ,
`str()`

(Inserir App aqui?)

Operadores

Aritméticos

Sintaxe introduzida: `+` , `-` , `*` , `/` , `%`

App10, Operadores Aritméticos

Precedência

Sintaxe introduzida: `()`

Abreviações*

`++ , -- , += , -= , *= , /= , -`

App11, Abreviações

Restrições*

Sintaxe introduzida:

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

App12, Restrições

Operadores Relacionais

Sintaxe introduzida:

`>, <, >=, <=, ==, !=`

App13, Operadores Relacionais

Operadores Condicionais*

`if, else, {}`

App14, Operadores Condicionais

Operadores Lógicos

Sintaxe introduzida:

`||, &&, !`

App15, Operadores Lógicos

Estrutura de um programa

Modo Simples (Básico)

This mode is used drawing static images and learning fundamentals of programming. Simple lines of code have a direct representation on the screen.

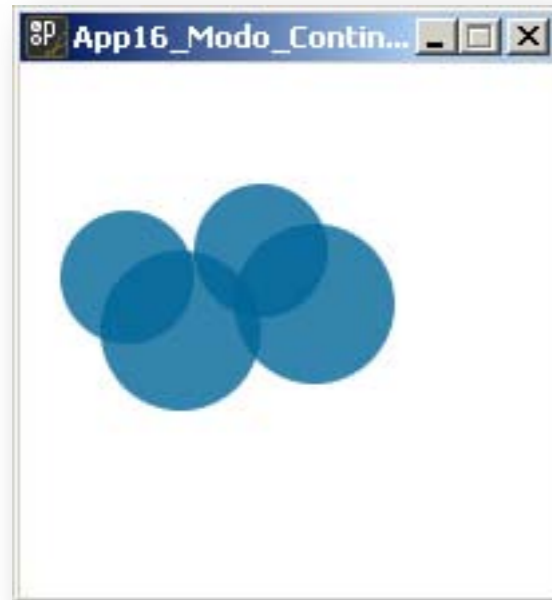
Modo Contínuo

Sintaxe introduzida:

`setup()` , `draw()` , `frameRate()` , `frameCount()` ,
`loop()` , `noLoop()` , `redraw()`

Conceitos introduzidos: Bloco de Código, *Callbacks*, Funções, Visibilidade das variáveis

This mode provides a `setup()` structure that is run once when the program begins and a `draw()` structure which by default continually loops through the code inside. This additional structure allows writing custom functions and classes and using keyboard and mouse events.



App16, Modo Contínuo

Condições (e blocos de código)

Sintaxe introduzida:

`if()`, `else()`, `{}`, `switch()*`, `case*`

App17, Condições

Ciclos ou Iterações (e Nested Iterations)

Sintaxe introduzida:

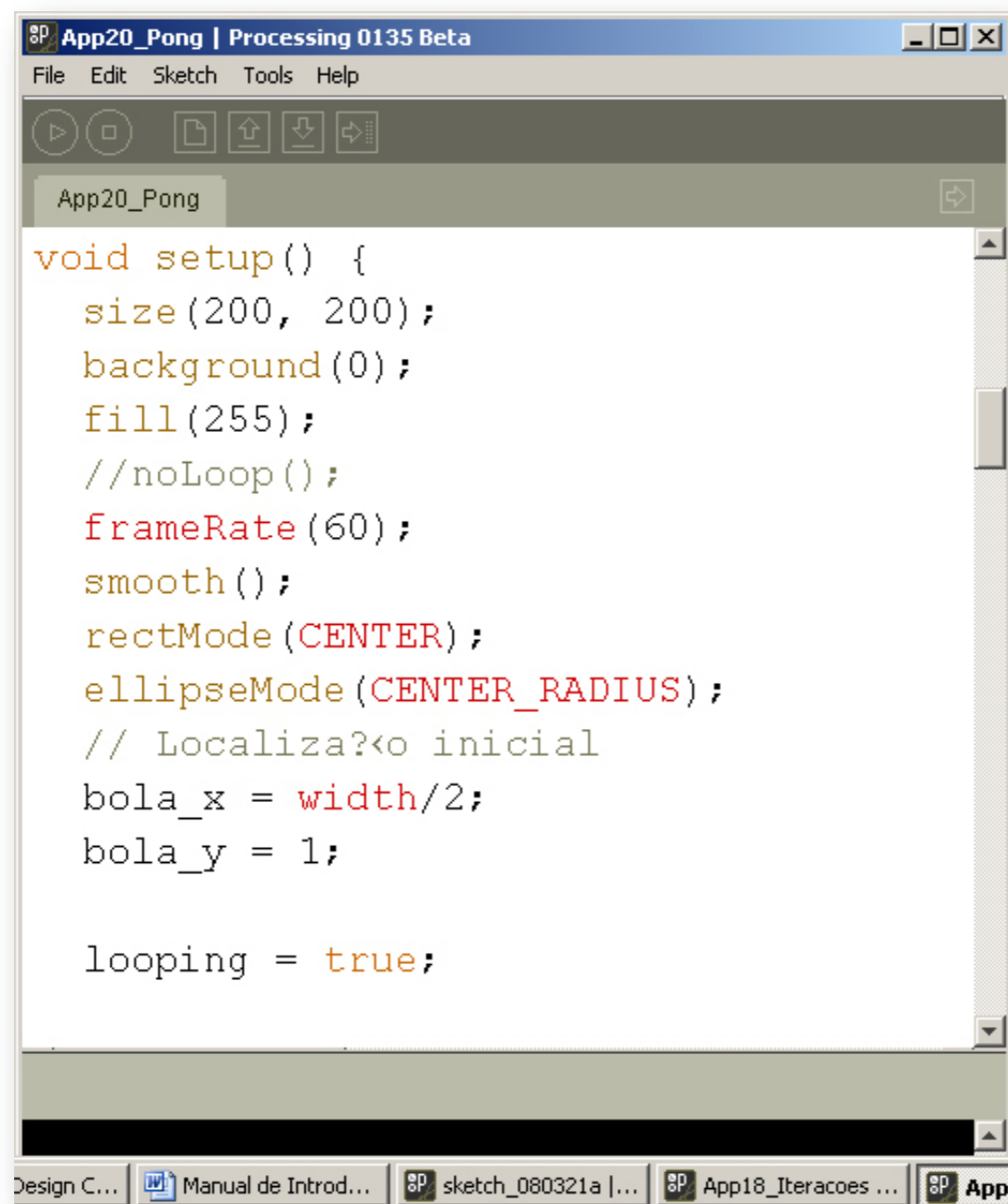
for ()



App18, Iterações

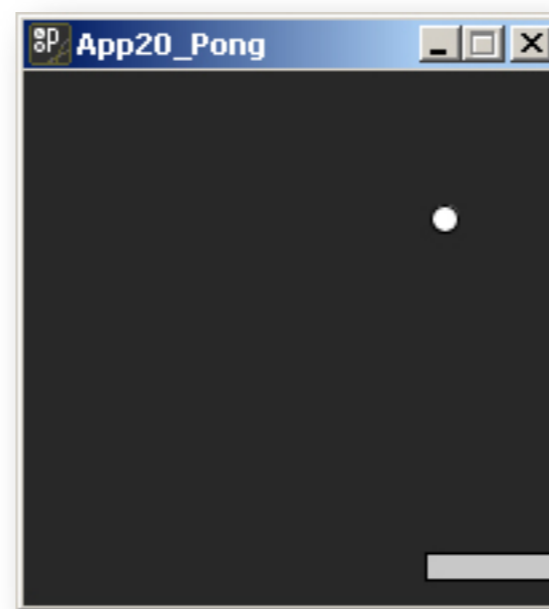
Tarde

Prática I – Pong



```
App20_Pong | Processing 0135 Beta
File Edit Sketch Tools Help
App20_Pong
void setup() {
  size(200, 200);
  background(0);
  fill(255);
  //noLoop();
  frameRate(60);
  smooth();
  rectMode(CENTER);
  ellipseMode(CENTER_RADIUS);
  // Localiza?o inicial
  bola_x = width/2;
  bola_y = 1;

  looping = true;
}
```



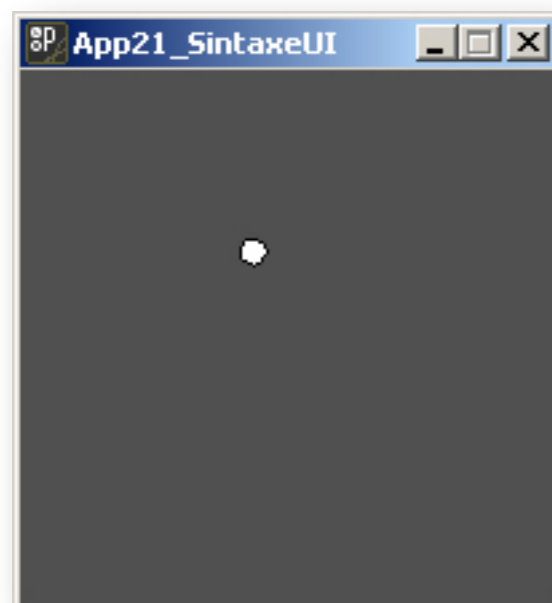
App19, Pong Simples

App20, Pong

Sintaxe UI

Sintaxe introduzida:

```
mouseX, mouseY, pmouseX, pmouseY,  
mousePressed, mouseButton, cursor(),  
noCursor(), key, keyPressed, keyCode,  
switch()*, case*
```



App21, SintaxeUI

Dia 2

Manhã

Conceitos Gerais de Programação

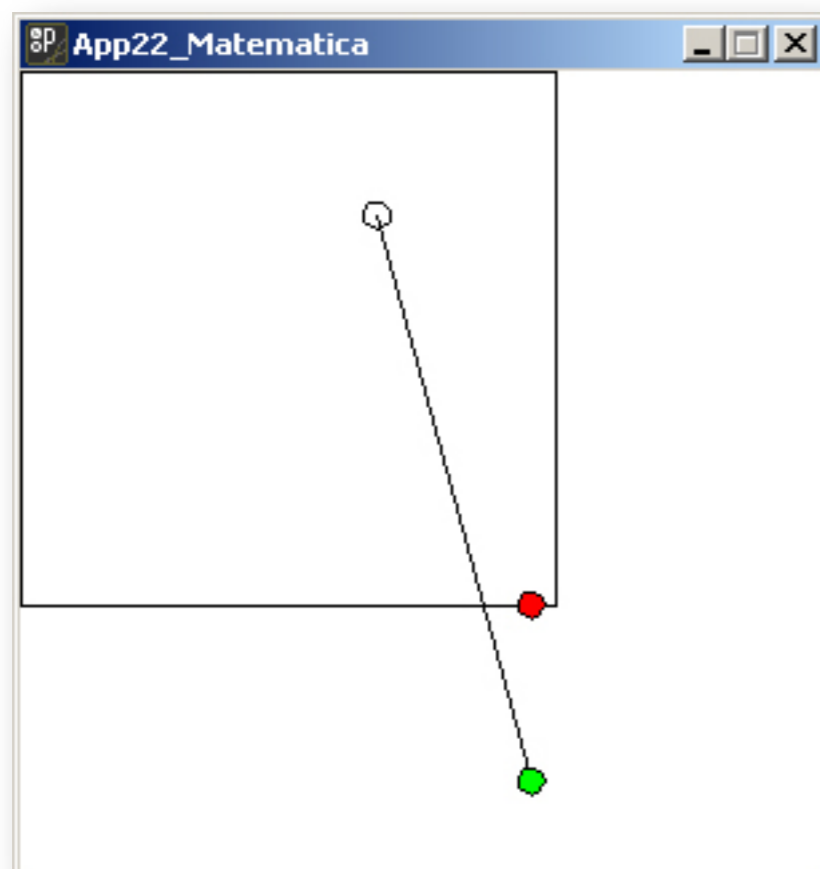
Matemática

Sintaxe introduzida:

`constrain()` , `dist()` , `random()`

Sintaxe extra:

`abs()` , `sqrt()` , `pow()` , `norm()` , `lerp()` , `map()`



App22, Matemática

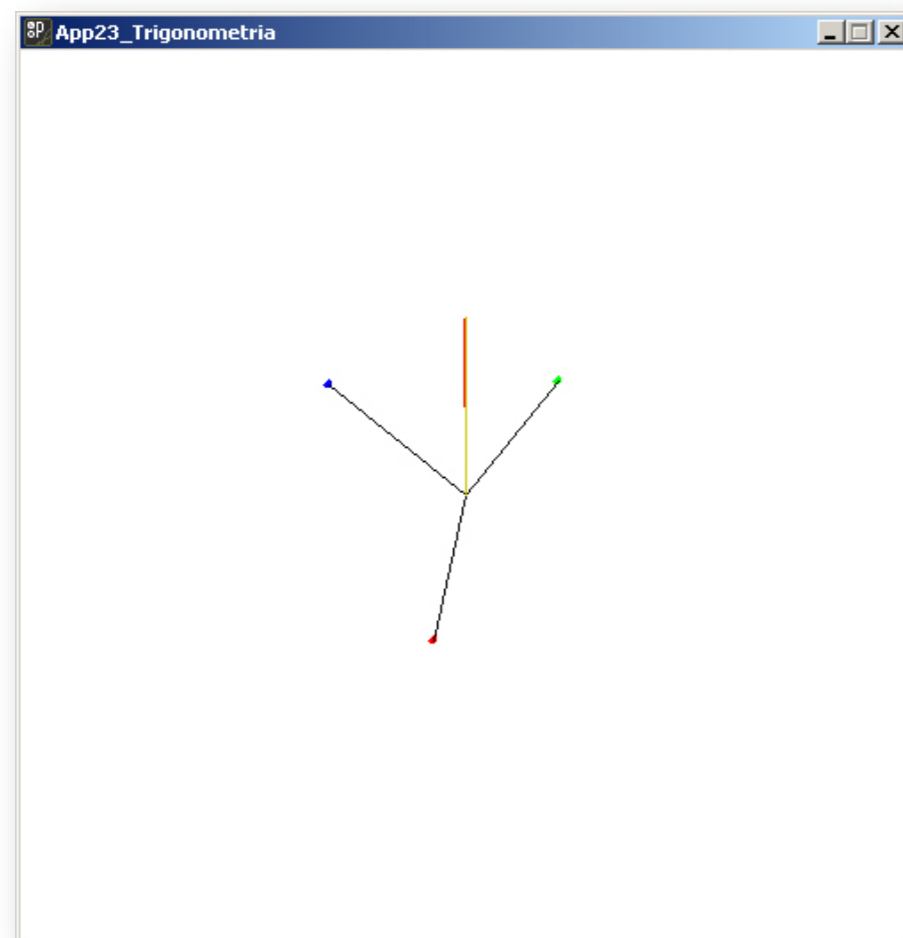
Trigonometria

Sintaxe introduzida:

`PI`, `sin()`, `cos()`, `radians()`

Sintaxe extra:

`atan2()`, `degrees()`



App23, Trigonometria

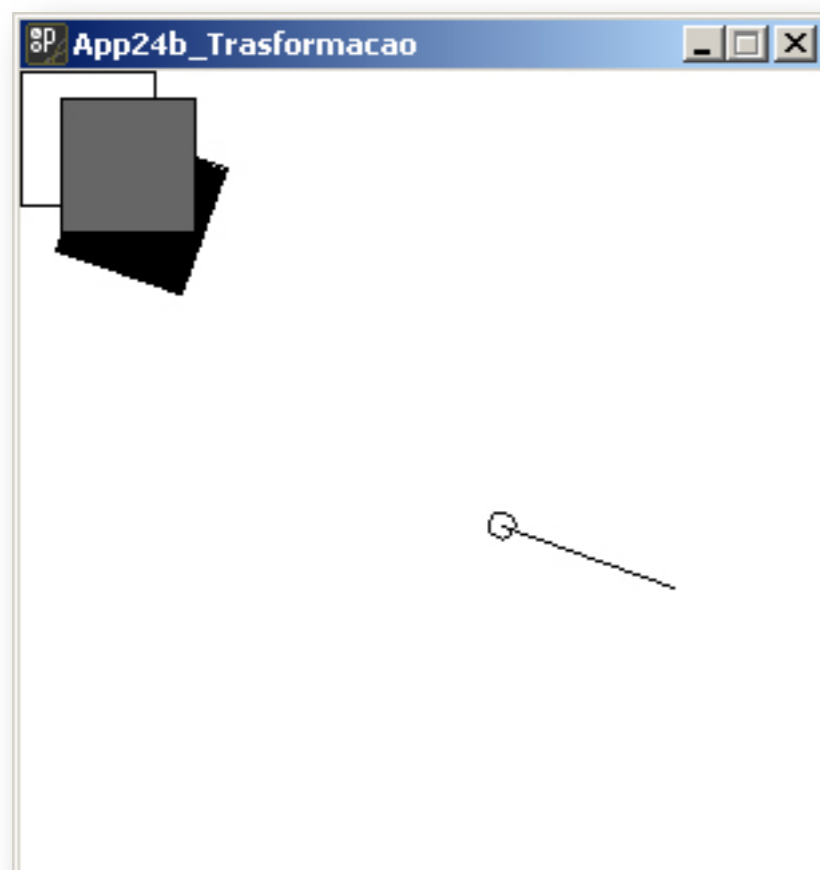
Transformação

Sintaxe introduzida:

```
translate() , rotate()
```

Sintaxe extra:

```
scale() , pushMatrix() , popMatrix();
```



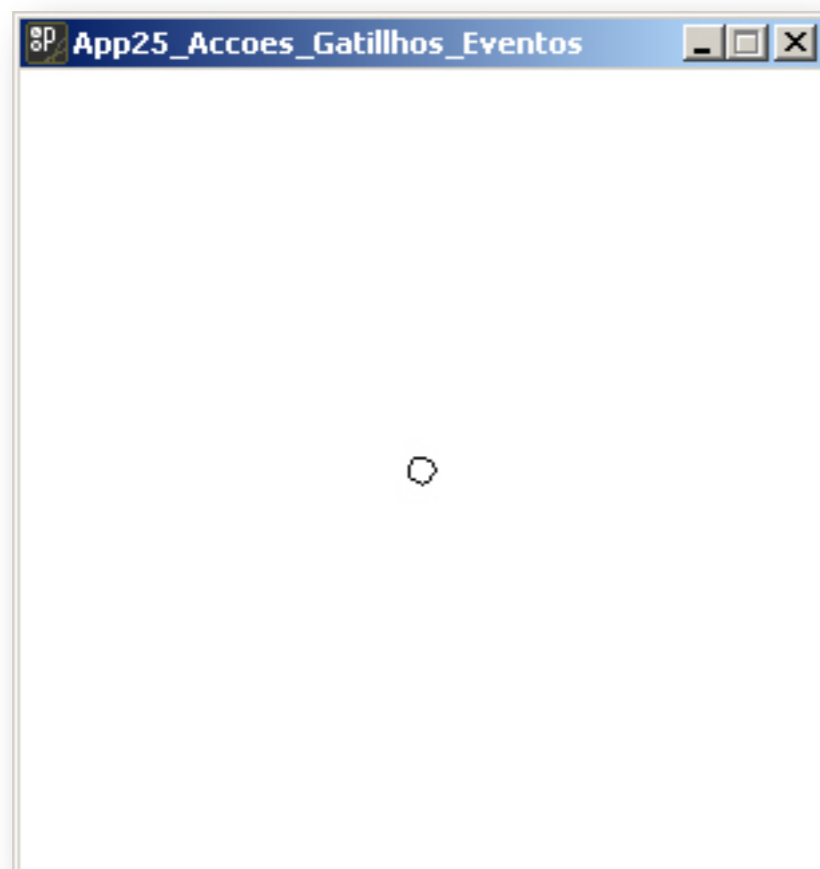
App24, Transformações

App24b, Transformações

Acções, Gatilhos e Eventos*

Sintaxe introduzida:

`mousePressed`, `mouseReleased`, `mouseMoved`,
`mouseDragged`, `keyPressed`, `keyReleased`



App25, Acções Gatilhos e Eventos

Dados complexos*

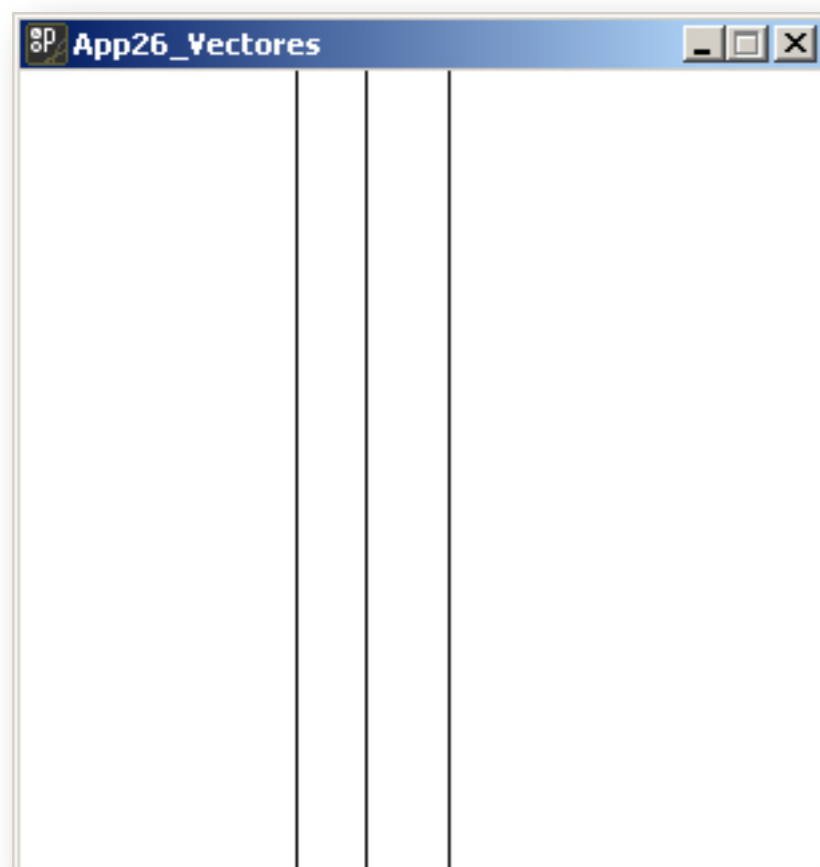
Vectores e Matrizes

Sintaxe introduzida:

Array, [], new, Array.length, expand()

Sintaxe extra:

append(), shorten(), arraycopy()



App26, Vectores

App26b, Vectores

Funções e Métodos (Parâmetros)

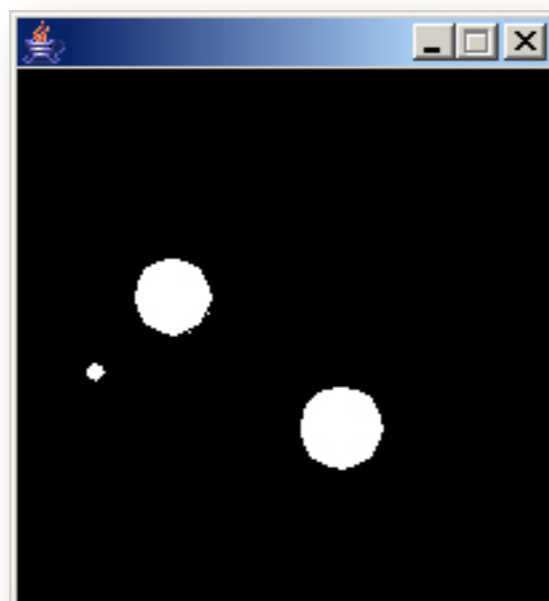
Sintaxe introduzida:

void, return

Conceitos Introduzidos: Bloco de Código, Parâmetros e Retorno

App27, Funções e Parâmetros

Classes e Objectos



Sintaxe introduzida:

class, Object, . //String.length

App28, Classes e Objectos

Texto e Tipografia

Sintaxe introduzida:

```
PFont, loadFont(), textFont(), text(),  
textSize(), textLeading(), textAlign(),  
textWidth()
```

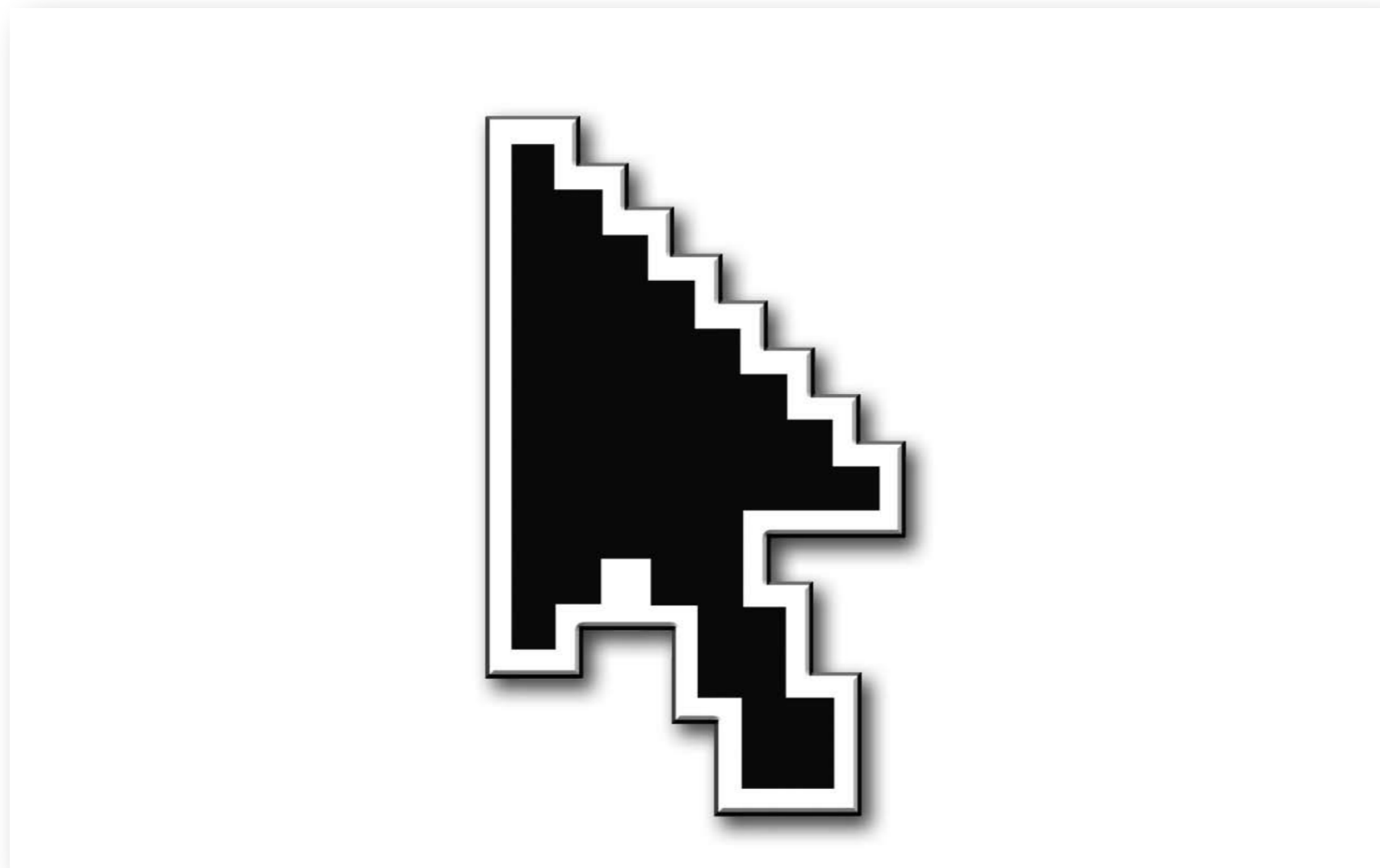


App29, Texto

Imagens e Pixels

Sintaxe introduzida:

```
PImage, loadImage(), image(), tint(),  
noTint(), get(), set(), save(), saveImage()
```



App30, Imagens

Exercício

Cálculo de média de Idades



App31, Média Consola

App32, Média Primitivas

App33, Média Gráfico

Pong I – Pontuação e/ou imagens

Bibliotecas: Quicktime, controlP5, OpenGL*

Pratica II – Field of Flowers I

Bibliotecas: PDF, OpenGL

Tarde

Pratica III – Field of Flowers II

Interactivo – Bibliotecas: Oscillator, jTablet

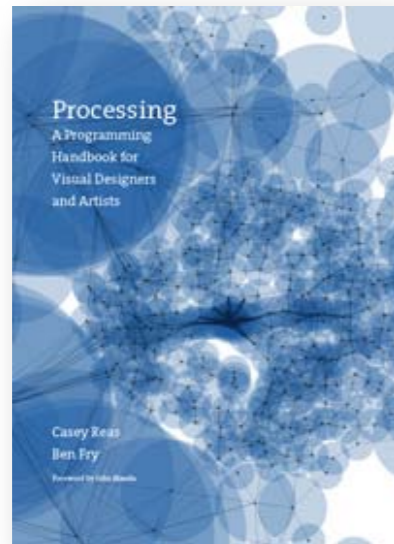
– Pong II – Wii Connect

Interactivo – Bibliotecas: WiiMote

– Pong III – Computer Vision

Interactivo – jMyron

Bibliografia e Referências



Processing: A Programming Handbook
for Visual Designers and Artists

Casey Reas and Ben Fry (Foreword by John Maeda).
Published 24 August 2007, MIT Press. 736 pages.
Hardcover.

CARDOSO, Jorge – Sebenta de Programação Multimédia [Em
linha]. UCP: Porto, 2006. 16 Fev 2006 Disponível na WWW: URL:
<http://teaching.jotgecardoso.org/pm>.

MENDES, António José; MARCELINO, Maria José – Fundamentos
de Programação em Java 2. FCA: Lisboa, [s.d.]. ISBN 972-722-423-7

[*Manual original da Formação + Links*](#)