Programação 1

Processing

Aula 5

Valeri Skliarov, Prof. Catedrático

Email: skl@ua.pt
URL: http://sweet.ua.pt/skl/

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

http://elearning.ua.pt/

Aula 5

- Ambiente processing
- Ambiente Geany
- Exemplos

Aula 05 - Trabalhos e Processing

Regras para os trabalhos

Exemplos de Processing

Propostas base para os trabalhos

- exemplos_averao.zip
- GeanyVS.rar
- P1_Trabalho - processing v1a.pdf
- P1-aula05_processing_2016.pdf
- PBatalhaNaval1.java
- Processing_AVerao.pdf
- ProcVS.rar
- trabalhos_base.zip

quadrados_rand2 | Processing 3.2.1

File Edit Sketch Debug Tools Help

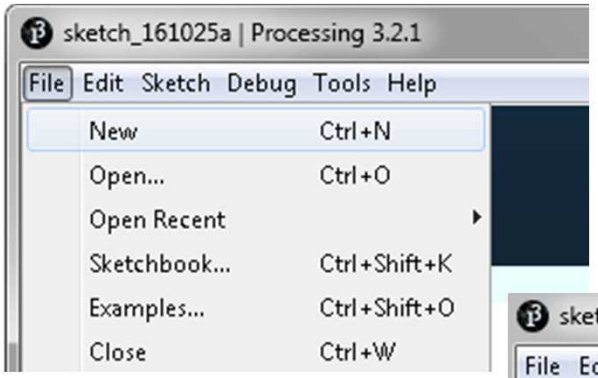
Java

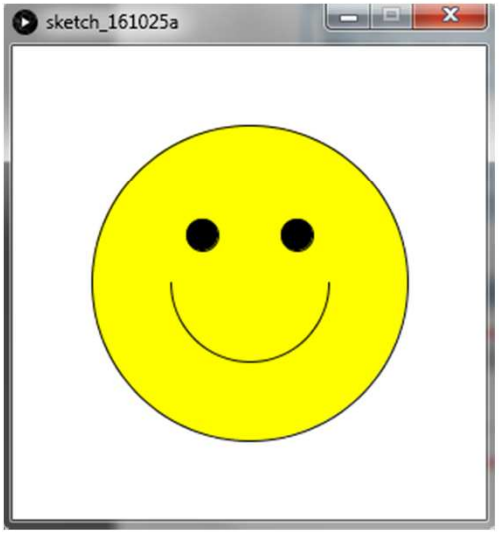
quadrados_rand2

```
int count=300;
void setup() {
  size (510, 510);
  background (0);
}

void draw () {
  fill (random (0, 255), random(0, 255), random (0, 255));
  if (random(0, 10) >5 ) {
    rect (random(0, width), random(0, height), random(width/2), random(height/2));
  } else {
    triangle(random(0, width), random(0, height),random(0, width), random(0, height),ra
  }
  if (--count == 0)noLoop();
}
```

Console Errors

Valeri Skliarov
2016/2017





sketch_161025a | Processing 3.2.1

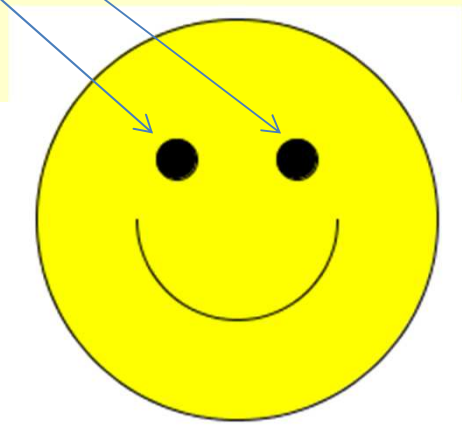
File Edit Sketch Debug Tools Help

Java

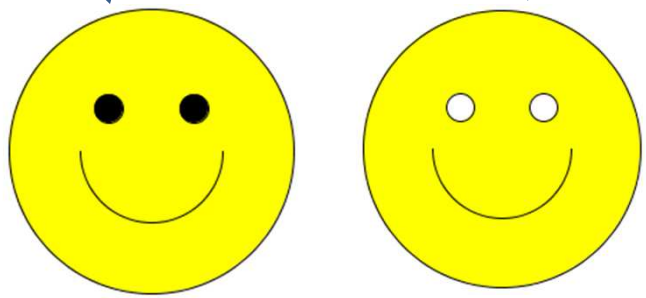
sketch_161025a

```
/* Primeiro programa - desenho de uma cara
JAM 2016-6-6
*/
// A função setup() é executada uma vez no início do programa
void setup() {
size( 300, 300 ); // tamanho da tela
}
// A função draw() é repetida várias vezes por segundo (25/30)
void draw() {
background( 255 ); // cor do fundo (0 - preto; 255 - branco)
fill( 255, 255, 0 ); // cor dos objectos (combinação de Red, Blue, Green)
ellipse( 150, 150, 200, 200 ); // desenha cabeça
fill(0);
ellipse( 180, 120, 20, 20 ); // desenha olho dir
ellipse( 120, 120, 20, 20 ); // desenha olho esq
noFill(); // desenha só contorno
arc( 150, 150, 100, 100, 0, PI ); // desenha boca
}
```

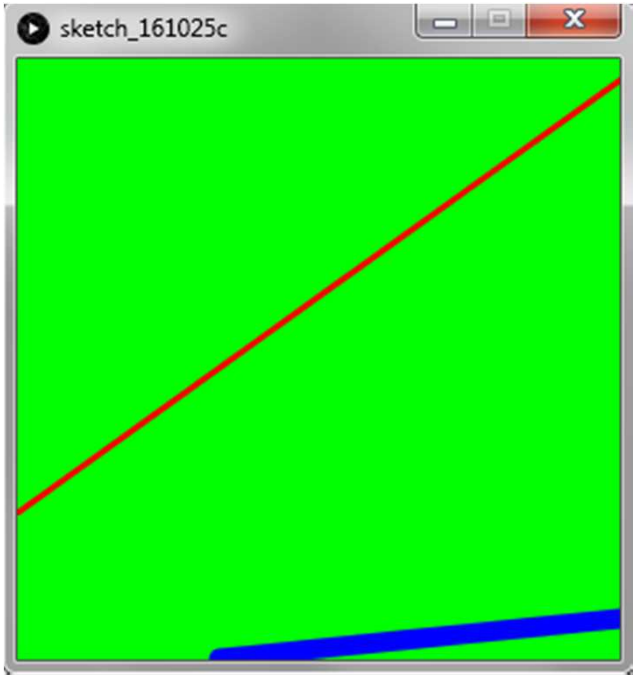
```
// A função setup() é executada uma vez no início do programa

void setup() {
size( 300, 300 );                    // tamanho da tela
}

// A função draw() é repetida várias vezes por segundo (25/30)
void draw() {
background( 255 );                   // cor do fundo (0 – preto; 255 – branco)
fill( 255, 255, 0 );                 // cor dos objectos (combinação de Red, Green, Blue)
ellipse( 150, 150, 200, 200 );       // desenha cabeça
fill(0);
ellipse( 180, 120, 20, 20 );         // desenha olho dir
ellipse( 120, 120, 20, 20 );         // desenha olho esq
noFill();                            // desenha só contorno
arc( 150, 150, 100, 100, 0, PI );    // desenha boca
}
```

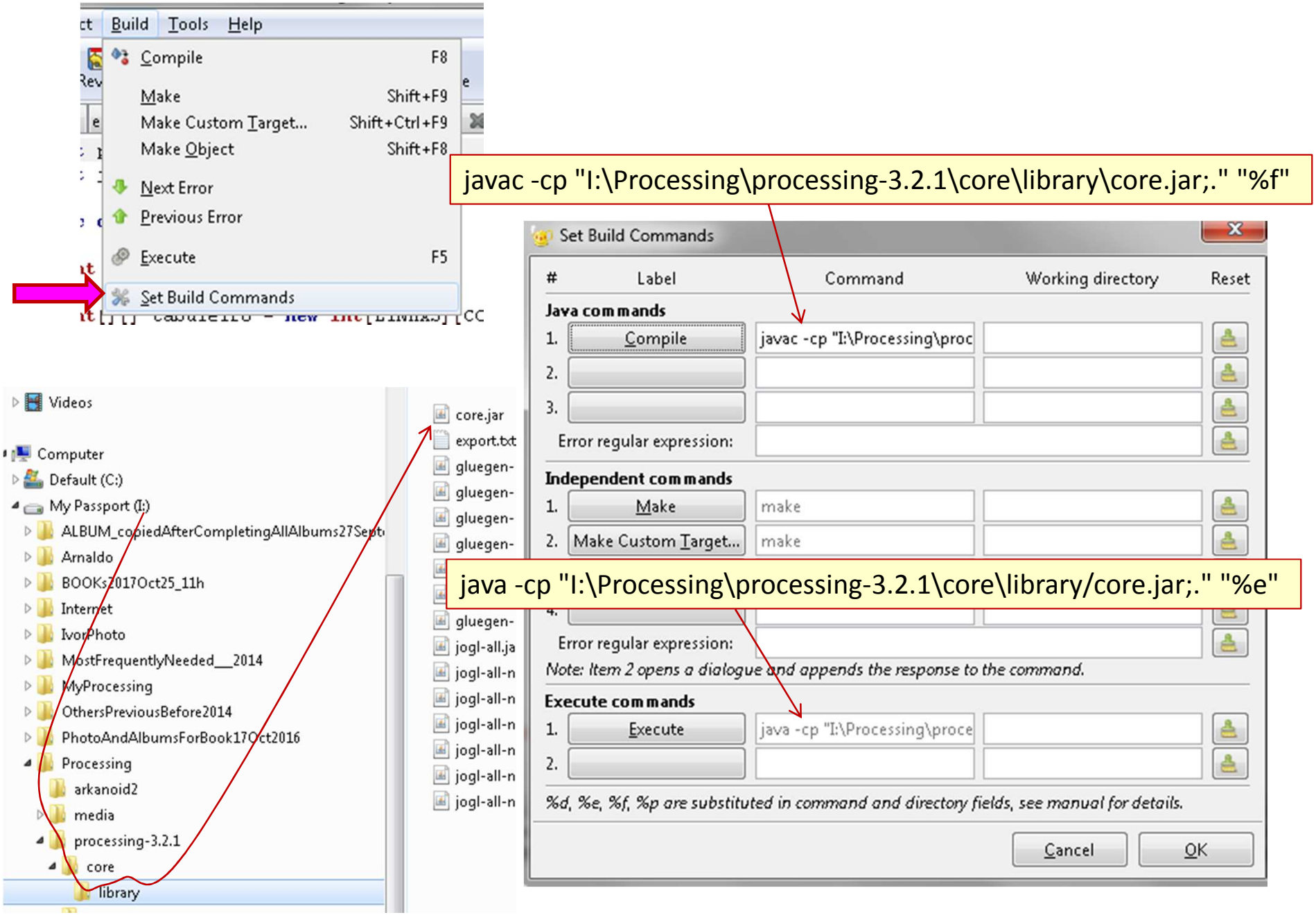


```
boolean sw = true;

void setup() {
size( 300, 300 ); // tamanho da tela
frameRate(2);
}
// A função draw() é repetida várias vezes por segundo (25/30)
void draw() {

background( 255 );
fill( 255, 255, 0 ); // cor dos objectos (combinação de Red, Blue, Green)
ellipse( 150, 150, 200, 200 ); // desenha cabeça
if (sw == true)        { fill( 255 ); sw = false; }
else                   { fill( 0 );  sw = true; }
ellipse( 180, 120, 20, 20 ); // desenha olho dir
ellipse( 120, 120, 20, 20 ); // desenha olho esq
noFill(); // desenha só contorno
arc( 150, 150, 100, 100, 0, PI ); // desenha boca
}
```



```
float yPos = 0.0;

void setup() {          // setup() runs once
  size(300, 300);
  frameRate(10);
}

void draw() {           // draw() loops forever, until stopped
  background(0,255,0);
  strokeWeight(10);
  stroke(0,0,255);
  yPos = yPos - 1.0;
  if (yPos < 0)   yPos = height;
  line(100, yPos+20, width, yPos);
}

void mousePressed() {
  strokeWeight(3);
  stroke(255,0,0);
  line(0, mouseX, width, mouseY);
}
```




Build
Tools
Help
Compile F8
Make Shift+F9
Make Custom Target... Shift+Ctrl+F9
Make Object Shift+F8
Next Error
Previous Error
Execute F5
Set Build Commands
javac -cp "I:\Processing\processing-3.2.1\core\library\core.jar;." "%f"
Set Build Commands
Label Command Working directory Reset
Java commands
1. Compile javac -cp "I:\Processing\proc
2.
3.
Error regular expression:
Independent commands
1. Make make
2. Make Custom Target... make
java -cp "I:\Processing\processing-3.2.1\core\library/core.jar;." "%e"
Error regular expression:
Note: Item 2 opens a dialogue and appends the response to the command.
Execute commands
1. Execute java -cp "I:\Processing\proce
2.
%d, %e, %f, %p are substituted in command and directory fields, see manual for details.
Cancel
OK
Videos
Computer
Default (C:)
My Passport (I:)
ALBUM_copiedAfterCompletingAllAlbums27Sept
Arnaldo
BOOKs2017Oct25_11h
Internet
IvorPhoto
MostFrequentlyNeeded__2014
MyProcessing
OthersPreviousBefore2014
PhotoAndAlbumsForBook17Oct2016
Processing
arkanoid2
media
processing-3.2.1
core
library
core.jar
export.txt
gluegen-
jogl-all.ja
jogl-all-n

```
import processing.core.*;
//  import java.util.*;
public class GeanyEx extends PApplet {
double yPos = 0.0;
public void settings() {
                size(300, 300);          }
public void setup() {  // setup() runs once
  frameRate(10);         }

public void draw() {  // draw() loops forever, until stopped
  background(0,255,0);
  strokeWeight(10);
  stroke(0,0,255);
  yPos = yPos - 1.0;
  if (yPos < 0) yPos = height;
  line(100, (float)yPos+20, width, (float)yPos);              }

public void mousePressed() {
  strokeWeight(3);
  stroke(255,0,0);
  line(0, mouseX, width, mouseY);                }

 public static void main(String args[]) {
       PApplet.main("GeanyEx");
    }
}
```

```
import processing.core.*;
import java.util.*;
public class GeanyEx extends PApplet {
```

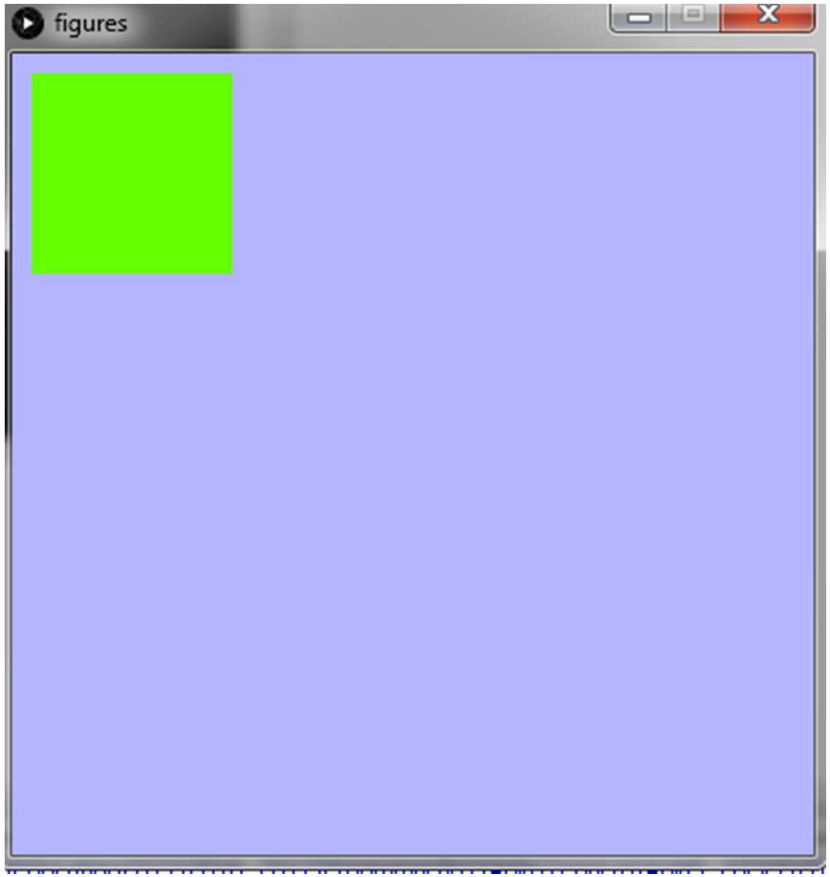
```
public void settings() {
                size(300, 300);        }
```

```
public static void main(String args[]) {
       PApplet.main("GeanyEx");
    }
}
```

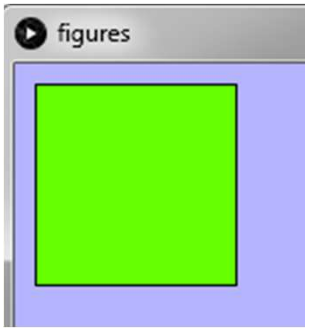
```
import processing.core.*;
import java.util.*;

public class figures extends PApplet {

public void settings() {
        size(400, 400);
    }

public void draw() {

  background(180,180,255);
  fill( 100, 255, 0 );
  noStroke();
  rect(10,10,100,100);
  //~ rect(10, 10, 100, 100, 20);
  //~ rect(10, 10, 100, 100, 50, 100, 20, 0);
  stroke(0);
  strokeWeight(1);
```

```
 fill(255,0,0);
  beginShape();
  //~ vertex(250,250);
  //~ vertex(150,250);
  //~ vertex(250,150);
  //~ vertex(150,150);

  //~ vertex(250,250);
  //~ vertex(150,250);
  //~ vertex(250,150);

  endShape(CLOSE);
}

public static void main(String args[]) {
        PApplet.main("figures");
    }

}
```

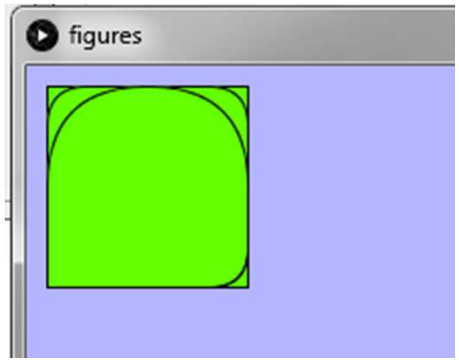
```
import processing.core.*;
import java.util.*;

public class figures extends PApplet {

public void settings() {
        size(400, 400);    }

public void draw() {
  background(180,180,255);
  fill( 100, 255, 0 );
  noStroke();
  rect(10,10,100,100);
}
public static void main(String args[]) {
        PApplet.main("figures");
    }
}
```



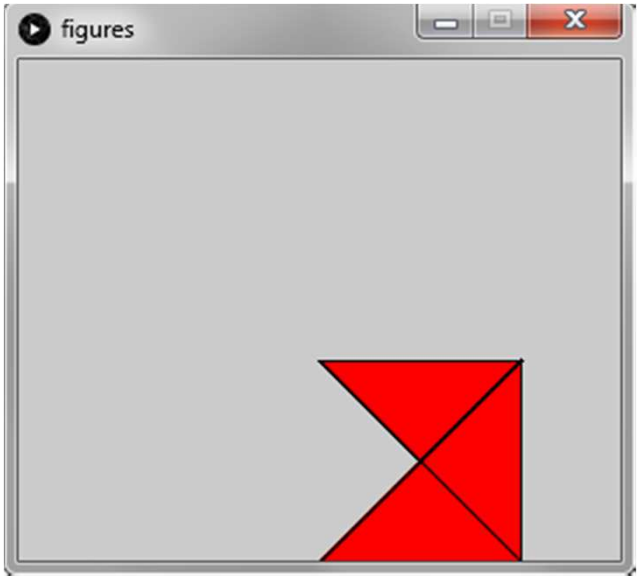
// noStroke();



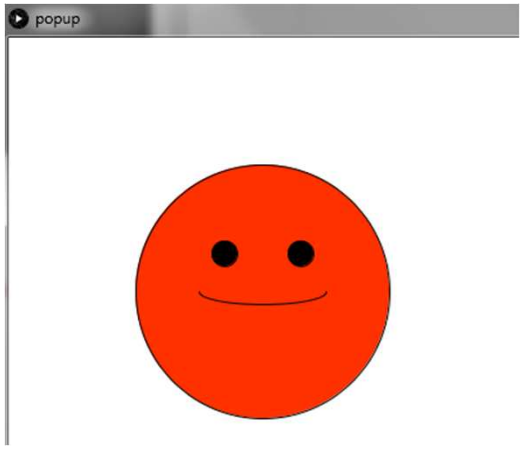
```
import processing.core.*;
import java.util.*;

public class figures extends PApplet {

public void settings() {
        size(400, 400);    }

public void draw() {
  background(180,180,255);
  fill( 100, 255, 0 );
  noStroke();
  rect(10,10,100,100);
  rect(10, 10, 100, 100, 20);
  rect(10, 10, 100, 100, 50, 100, 20, 0);
}
public static void main(String args[]) {
        PApplet.main("figures");
    }
}
```

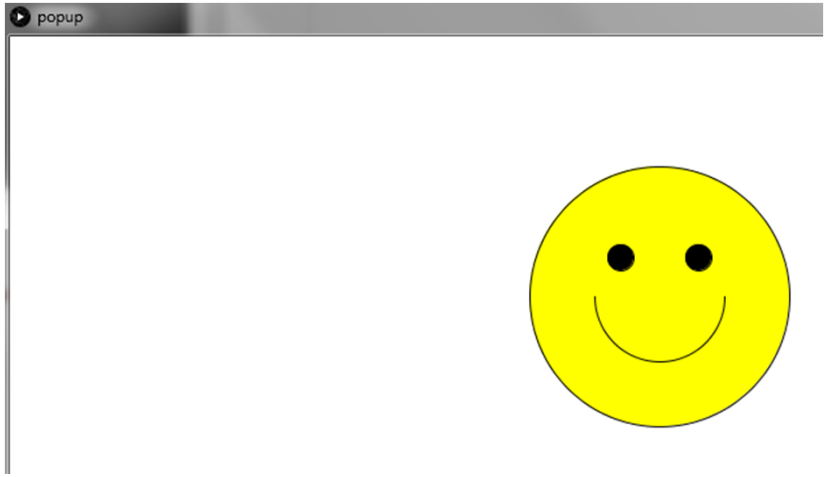


```
import processing.core.*;
import java.util.*;
public class figures extends PApplet {

public void settings() {
      size(300, 250);   }

public void draw() {

  fill(255,0,0);
  beginShape();
  vertex(250,250);
  vertex(150,250);
  vertex(250,150);
  vertex(150,150);

  vertex(250,250);
  vertex(150,250);
  vertex(250,150);

  endShape(CLOSE);

}
```

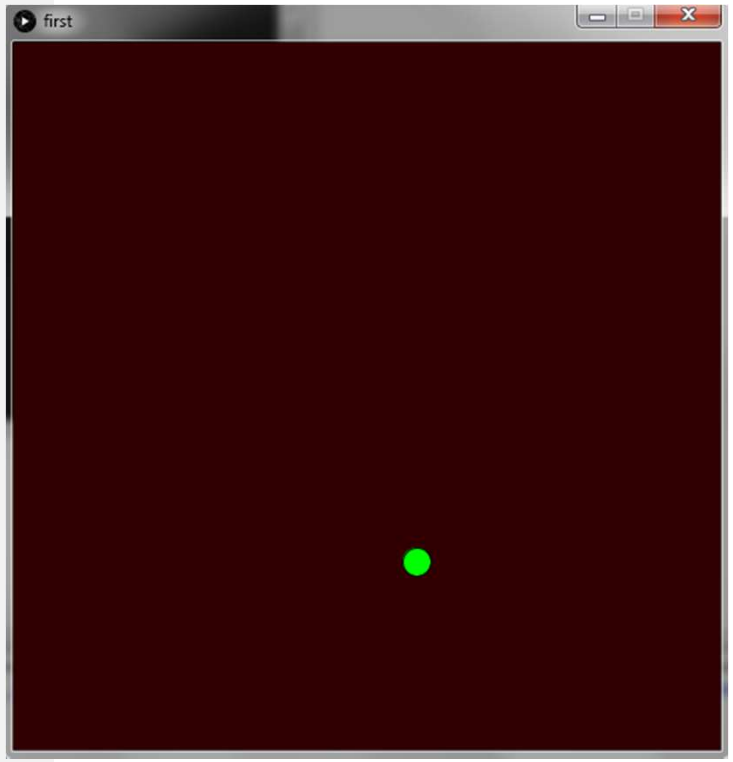


```
import processing.core.*;
public class popup extends PApplet {
                int count=0;
public void settings() {
  size( 800, 600 );        }
public void draw() {
  background( 255 );
  if (count < 50 )          smile(200, 200, 50, 20);
  else                      smile(500, 200, 255, 100);
  if (++count > 100) count = 0;
}
public void smile(int x, int y, int cor, int sorriso) {
  fill( 255, cor, 0 );
  ellipse( x, y, 200, 200 );
  fill(0);
  ellipse( x-30, y-30, 20, 20 );
  ellipse( x+30, y-30, 20, 20 );
  noFill();
  stroke(0);
  arc( x, y, 100, sorriso, 0, PI );
}
public static void main(String args[]) {
       PApplet.main("popup");
    }
}
```

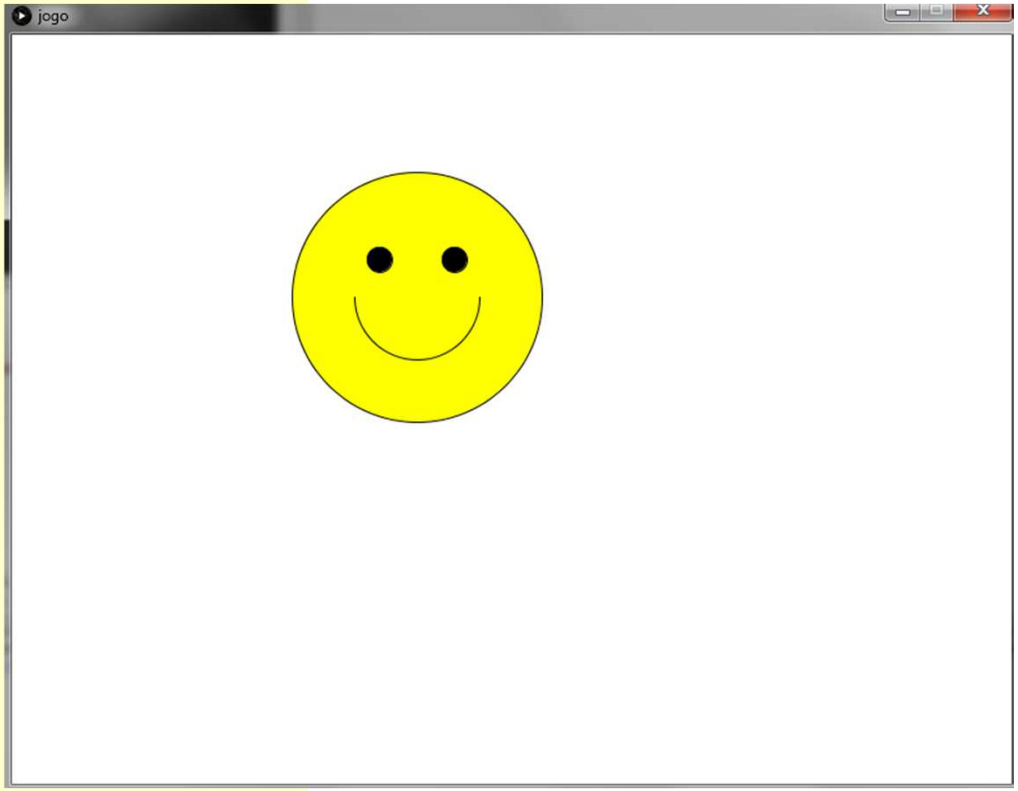




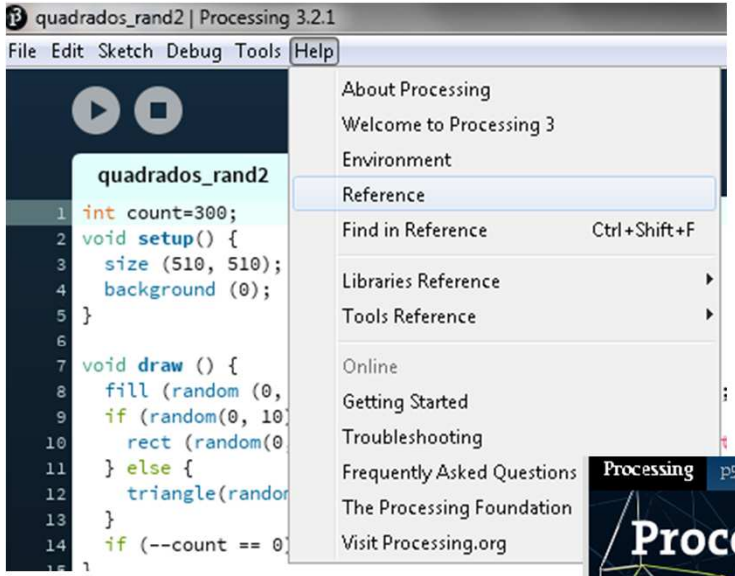
```
import processing.core.*;
public class first extends PApplet {
                PVector pos;            PVector vel;

public void settings() {      size(500, 500);    }

public void setup()                                {
  fill(0,255,0);                                   // color of circle
  frameRate( 50 );                                 // speed of movements
  pos = new PVector( width/2, height/2 );
  vel = new PVector( 5, -3 );                      }

public void draw() {
  background(50,0,0);                              // background color
  ellipse( pos.x, pos.y, 20, 20);
  pos.add( vel );
  if ( pos.x + 12 > width ||  pos.x - 12  < 0 )    // where to stop
    vel = new PVector( -vel.x, vel.y );

  if ( pos.y + 12 > height || pos.y - 12 < 0 )     // where to stop
    vel = new PVector( vel.x, -vel.y );

public static void main(String args[])             {
      PApplet.main("first");                       }
}
```

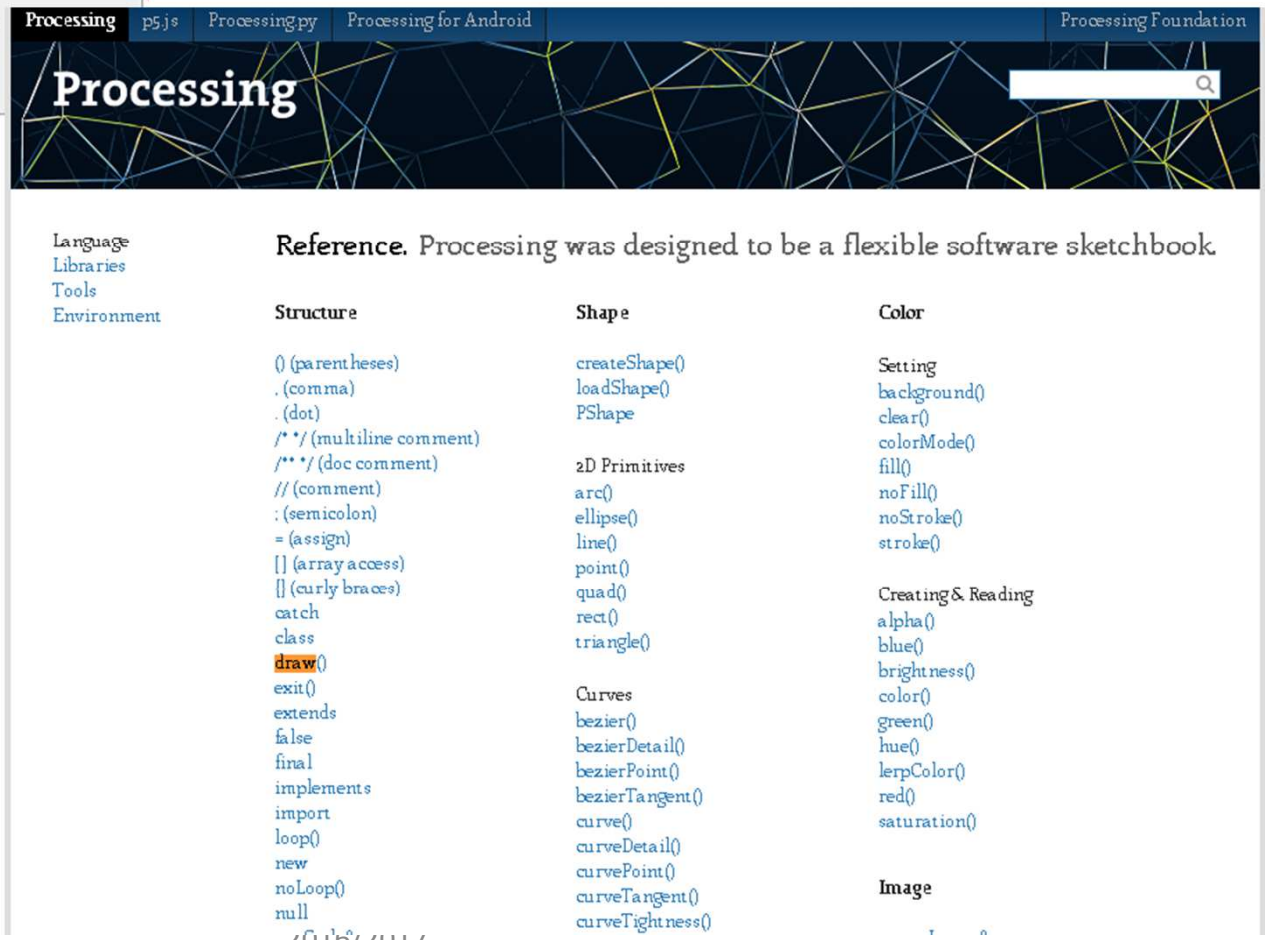


```
import processing.core.*;
public class jogo extends PApplet {
int x=101, y=150, velx = 1, dy=10;
public void settings() {  size( 800, 600 );  }
public void draw()                                    {
  background( 255 );      fill(255, 0, 0);
  smile(x, y, 255, 100);
  x = x+velx;
  if (x > width-100 || x < 100)    velx = -velx;
  if (y > height - 100 || y <100 ) dy= -dy;
  if (mousePressed) y += dy;                          }
public void smile(int x, int y, int cor, int sorriso) {
  fill( 255, cor, 0 );
  ellipse( x, y, 200, 200 );
  fill(0);
  ellipse( x-30, y-30, 20, 20 );
  ellipse( x+30, y-30, 20, 20 );
  noFill();
  arc( x, y, 100, sorriso, 0, PI );                   }
public float distancia(int x, int y, int x1, int y1)   {
  return sqrt((x-x1)*(x-x1)+(y-y1)*(y-y1));           }
public void keyPressed()                  {
  if (key == 'a') velx = 5;
  if (key == 'b') velx = 10;
  if (key == 'c') velx = 1;                   }
public static void main(String args[]) {
      PApplet.main("jogo");
   }
}
```



```
import processing.core.*;
public class text extends PApplet {
String[] palavras = {"Aveiro", "Braga", "Lisboa", "Faro", "Coimbra"};
public void settings()          {
  size( 800, 300 );             }

public void setup() {
  background(0);
  //noLoop();
}

public void draw() {
  for (int i=0; i < palavras.length; i++) {
    fill(random(255), random(255), random(255));
    textSize(50);
    text(palavras[i], (i+1)*100, (i+1)*50);
  }
}

public void mousePressed()      {
  noLoop();                     }

public static void main(String args[]) {
      PApplet.main("text");
    }
}
```


quadrados_rand2 | Processing 3.2.1
File Edit Sketch Debug Tools Help
quadrados_rand2
int count=300;
void setup() {
size (510, 510);
background (0);
}
void draw () {
fill (random (0,
if (random(0, 10
rect (random(0
} else {
triangle(rando
}
if (--count == 0
About Processing
Welcome to Processing 3
Environment
Reference
Find in Reference Ctrl+Shift+F
Libraries Reference
Tools Reference
Online
Getting Started
Troubleshooting
Frequently Asked Questions
The Processing Foundation
Visit Processing.org


Processing
p5.js
Processing.py
Processing for Android
Processing Foundation
Processing
Language
Libraries
Tools
Environment
Reference. Processing was designed to be a flexible software sketchbook.
Structure
() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
final
implements
import
loop()
new
noLoop()
null
Shape
createShape()
loadShape()
PShape
2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()
Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()
Color
Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()
Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()
Image

Name draw()

Examples

```
float yPos = 0.0;

void setup() {  // setup() runs once
  size(200, 200);
  frameRate(30);
}

void draw() {  // draw() loops forever, until stopped
  background(204);
  yPos = yPos - 1.0;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

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

// Although empty here, draw() is needed so
// the sketch can process user input events
// (mouse presses in this case).
void draw() { }

void mousePressed() {
  line(mouseX, 10, mouseX, 90);
}
```

Description Called directly after `setup()`, the `draw()` function continuously executes the lines of code contained inside its block until the program is stopped or `noLoop()` is called. `draw()` is called automatically and should never be called explicitly.

It should always be controlled with `noLoop()`, `redraw()` and `loop()`. If `noLoop()` is used to stop the code in `draw()` from executing, then `redraw()` will cause the code inside `draw()` to be executed a single time, and `loop()` will cause the code inside `draw()` to resume executing continuously.

Os requisitos mínimos são os seguintes:

1) Desenvolver uma interface gráfica;
2) Suportar interatividade com o utilizador de forma a atingir objetivos propostos;
3) Usar estruturas de dados adequadas, nomeadamente *arrays*;
4) Usar funções;
5) Usar ficheiros (para guardar as pontuações dos vários jogadores).

Os trabalhos são realizados em grupos de 2 alunos da mesma turma prática.