

Control flow and PDF output

DG200 team,
Today: Loe Feijs



Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

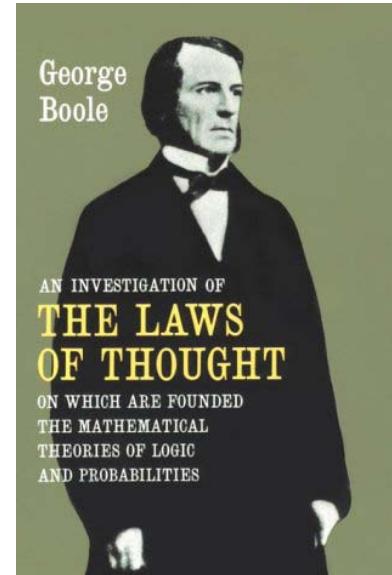
Content

- Variables and operators
- Conditionals and loops
- Vector graphics output
- Homework

Variables and Operators

Variables: overview

- A variable is a typed and named storage location
 - `<type> <name> ;`
 - `<type> <name> = <value> ;`
- Types
 - **byte** (-128, 127)
 - **int** (-10, 2147483647)
 - **float** (3.1425, 2.15)
 - **char** ('a', '!')
 - **String** ("this is a string")
 - **boolean** (true, false)



Source:
[http://www.cs.auckland.ac.nz/
historydisplays/TimeLine/Tim
eLine2/TimeLine2Main.php](http://www.cs.auckland.ac.nz/historydisplays/TimeLine/TimeLine2/TimeLine2Main.php)

Variables: conventions

- give variables meaningful names
- initialize variables before use
- adhere to naming conventions (camelBack notation)
 - **myPacMan** ok
 - **mypacman** no
- size matters (or in this case, case)
 - **myVariable** is not **myvariable**



Source:
picsbox.biz/key/how%20to%20draw%20camel

Variables: small examples

```
int myAge; ←declaration  
myAge = 57; ←initialization  
String myLanguage = "Processing"; ←declaration and initialization  
boolean isALanguage = true;  
char myChar = 'a';  
float myFloat = 3.01; ←initializing using another variable's value  
int yourAge = myAge;
```

Operators: overview

Operators perform transformations on variables

=	(assignment)
+ , - , * , / , % ,	(arithmetic)
+= , -= , *= , /= , %=	(cumulative arithmetic)
> , >= , < , <= , == , !=	(comparison)
&& ,	(boolean logic)
++ , --	(increment, decrement)
(? :)	(conditional expression)
()	(grouping for priority)

Operators: meaning

x = y; means **x** is assigned the value of **y**

x += y; means **x = x+y;**

x /= y; means **x = x/y;**

x++ means **x = x+1;** but give **x** as the result

x-- means **x = x-1;** but give **x** as the result

(x == y) means **x** is compared to **y** for equality

(x >= y) means **x** is compared to **y** for \geq

(a && b) means **a** is true and also **b** is true

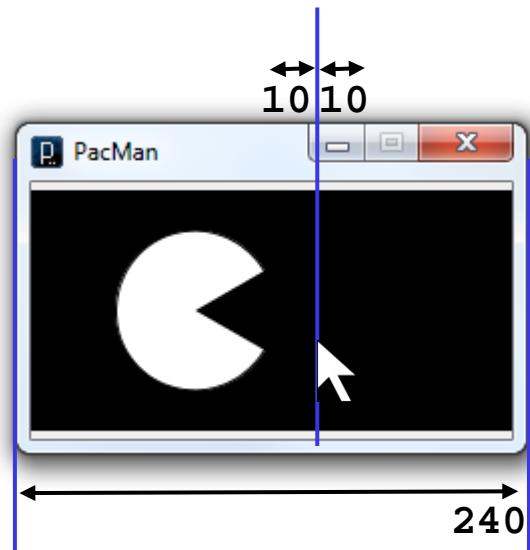
(a || b) means **a** is true or **b** is true, or both

(c ? x : y) means **x** when **c** is true and **y** otherwise

Operators: % and && example

Towards a larger example:

- `x = x % 240;`
- (when `x` runs out of the screen, reset to zero again)
- `x > mouseX-10 && x < mouseX+10`
- (test whether `x` is near the mouse's horizontal position)



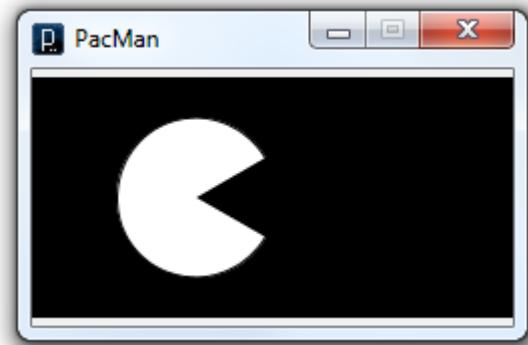
Operators: % and && example

```
//Casey Reas & Ben Fry: Getting Started with Processing
//Ex. 7-4, p.93, adapted by Loe Feijs for mouseX bumps.

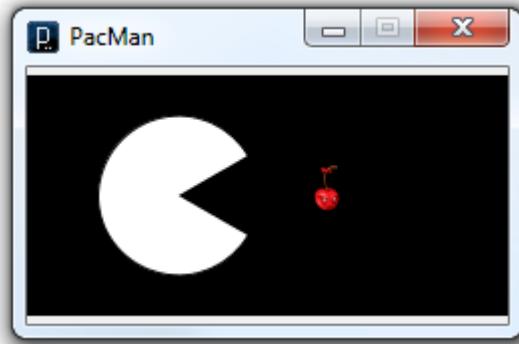
int radius = 40;
float x = -radius;
float speed = 0.5;

void setup() {
    size(240, 120);
    smooth();
    ellipseMode(RADIUS);
}
void draw() {
    background(0);
    x += speed; // Increase the value of x
    x = x % 240;
    int y = 60;
    if (x > mouseX-10 && x < mouseX+10){
        y += int(10*sin( x )); //jump up and down
    }
    arc(x, y, radius, radius, 0.52, 5.76);
}
```

Operators: % and && example



add more features if you like:



Conditionals and Loops

Conditionals: overview

- **if** (<boolean condition>) {
 <statement or statements> ;
} **else** {
 <statement or statements> ;
}
- **switch** (<variable>) {
 case <value> : <statement or statements> ; **break**;
 case <value> : <statement or statements> ; **break**;
 ...
 default : <statement or statements> ;
}

if () {} else {} example

Towards a larger example:

- solving quadratic equations
- with the well-known abc formula

$$x^2 + x + 1 = 0 \quad (\text{no solution})$$

$$x^2 - 2x + 1 = 0 \quad (\text{solution: } x_1 = 1)$$

$$x^2 + 2x + 1 = 0 \quad (\text{solution: } x_1 = -1)$$

$$x^2 - 6x + 8 = 0 \quad (\text{two solutions: } x_1 = 4, x_2 = 2)$$

- use $b^2 - 4ac$ as the "discriminant"

if () {} else {} example

```
//solving quadratic equation  
//a*x*x + b*x + c == 0 for x  
  
float a = 1;  
float b = -1;  
float c = -1;  
  
float x1 = 0;  
float x2 = 0;  
  
int solutions = 0;  
float D = b*b - 4*a*c;
```

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

if () {} else {} example

```
if (D < 0){  
    solutions = 0;  
} else {  
    if (D == 0) {  
        solutions = 1;  
        x1 = (-b + sqrt(D)) / 2*a;  
    }  
    else {  
        solutions = 2;  
        x1 = (-b + sqrt(D)) / 2*a;  
        x2 = (-b - sqrt(D)) / 2*a;  
    }  
}
```

switch () {} example

```
switch (solutions) {  
    case 0: println("oops, no solution.");  
        break;  
    case 1: print("yes, one solution: ");  
        println(x1);  
        break;  
    case 2: print("hurra, two solutions: ");  
        print(x1);  
        print(" , ");  
        println(x2);  
        break;  
    default: print("hmm, something's wrong");  
}  
}
```

Loops: overview

```
for ( <start> ; <condition> ; <action> ) {  
    <statement or statements> ;  
}
```

```
while ( <condition> ) {  
    <statement or statements> ;  
}
```

```
do {  
    <statement or statements> ;  
} while ( <condition> );
```

for (;;) {} example

```
for (int i=0; i<solutions; i++){
    print("hurra, a solution: ");
    if (i == 0) println(x1); else println(x2);
}

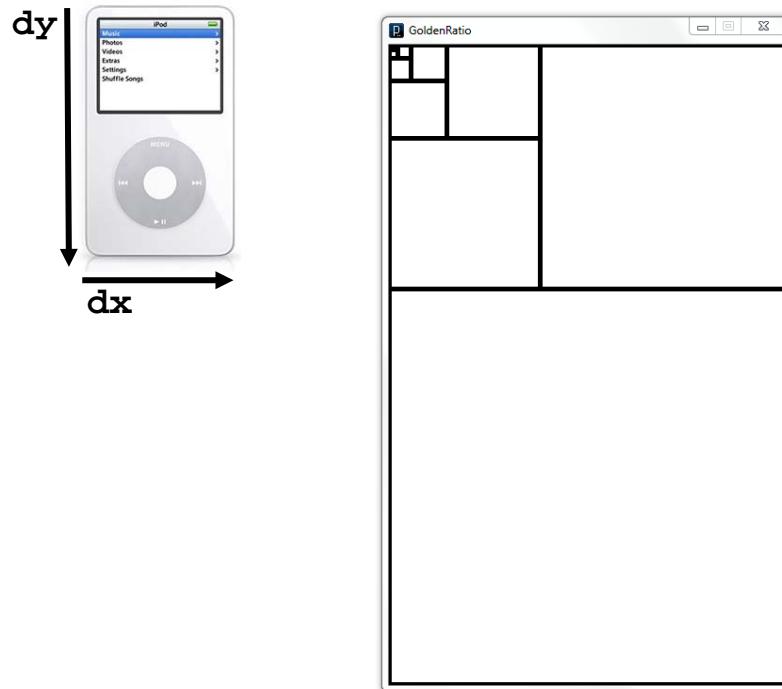
float a = 1; //a must not be zero
float b = -1; //choose any value
float c = -1; //choose any value
```

```
hurra, a solution: 1.618034
hurra, a solution: -0.618034
```

for (;;) {} example

Towards a larger example

- fractal drawing with rectangles
- height and width related by golden ratio
- repetition to create more smaller and smaller rectangles



for (;;) {} example

```
size(420,680);
float phi = 1.6181034;
float dx = 420;
float dy = dx * phi;
for (int i=0; i<12; i++){
    fill(random(255),random(255),random(255));
    strokeWeight(5);
    rect(0,0,dx,dy);
    float tmp = dx;
    dx = dy;
    dy = tmp;
    dx /= phi;
    dy /= phi;
}
```



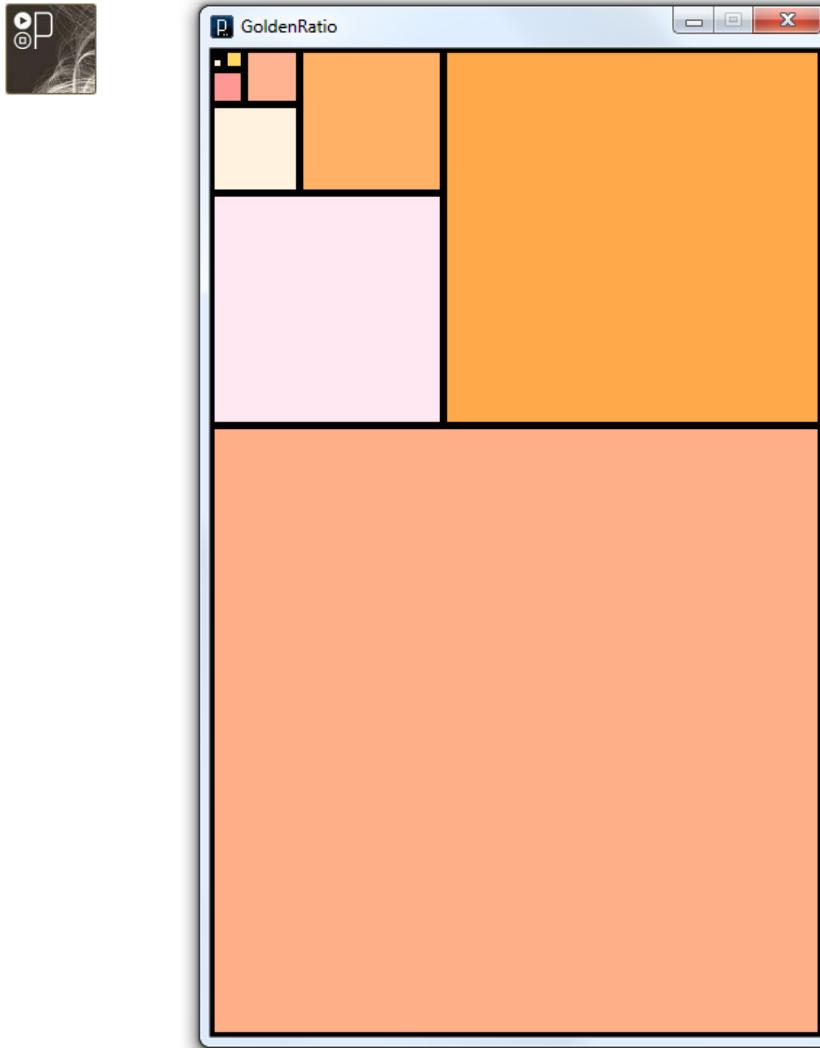
The diagram shows an iPod screen with a menu open. A double-headed arrow labeled "dy" points down from the top of the screen to the center of the device. Another double-headed arrow labeled "dx" points right from the left edge of the screen to the center of the device.

dx,dy

dx = dy;
dy = dx;

Oops...?

for (;;) {} example

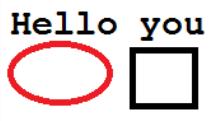
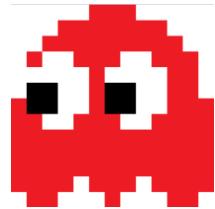


Vector Graphics Output

Vector graphics output

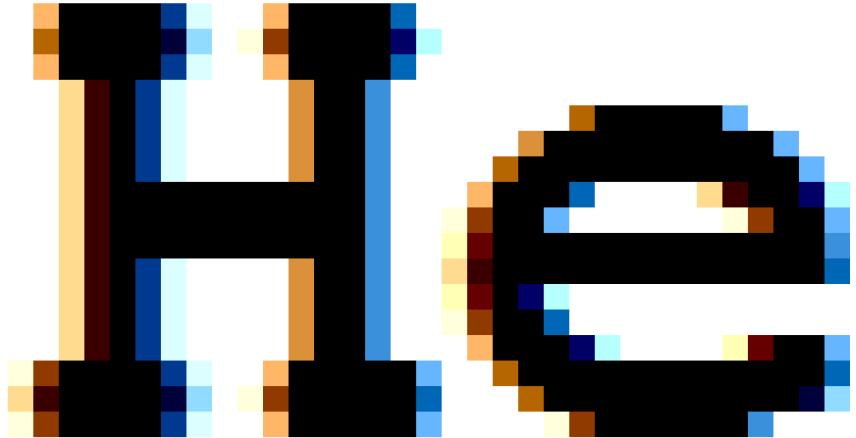
Raster graphics

- Nice for:
pixel art
- Bad for:
large-scale printing



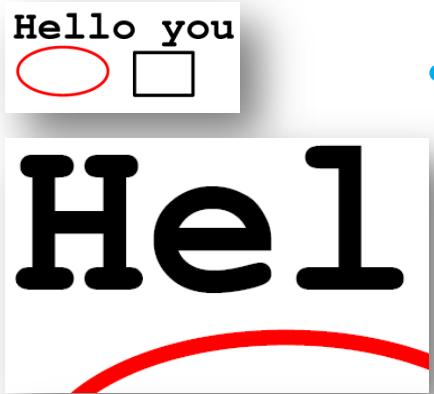
/ name of department

Vector graphics

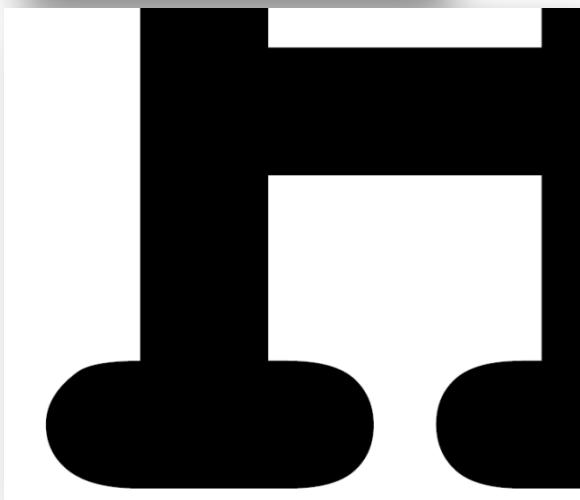


Vector graphics output

Vector graphics



- Nice for:
scaling,
large printing,
object editing.



Raster graphics

Vector graphics output example

Towards a larger example

- example 11-5 in the e-book of Casey Reas & Ben Fry
- `size(600, 800, PDF, "Ex-11-5.pdf");`
- instead of `size(600, 800);`
- use `beginShape()` and `endShape()` to create something
- use `for (;;) {}` to do that many times and make it interesting



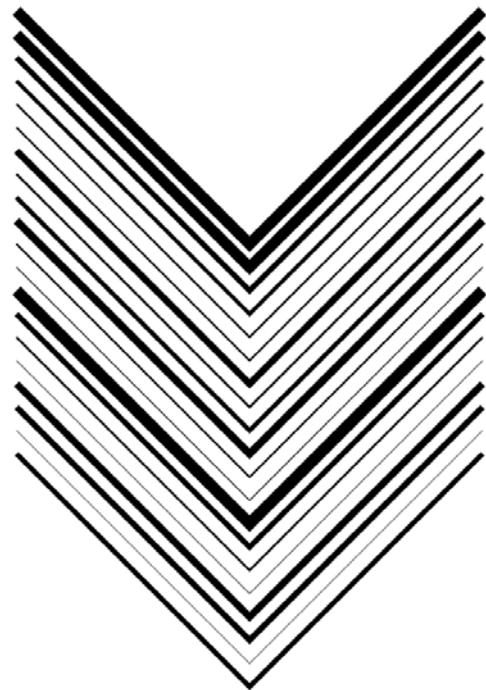
Vector graphics output, example 11-5

```
import processing.pdf.*;

void setup() {
    size(600, 800, PDF, "Ex-11-5.pdf");
    noFill();
    strokeCap(SQUARE);
}

void draw() {
    background(255);
    for (int y = 100; y < height - 300; y+=20) {
        float r = random(0, 102);
        strokeWeight(r / 10);
        beginShape();
        vertex(100, y);
        vertex(width/2, y + 200);
        vertex(width-100, y);
        endShape();
    }
    exit();
}
```

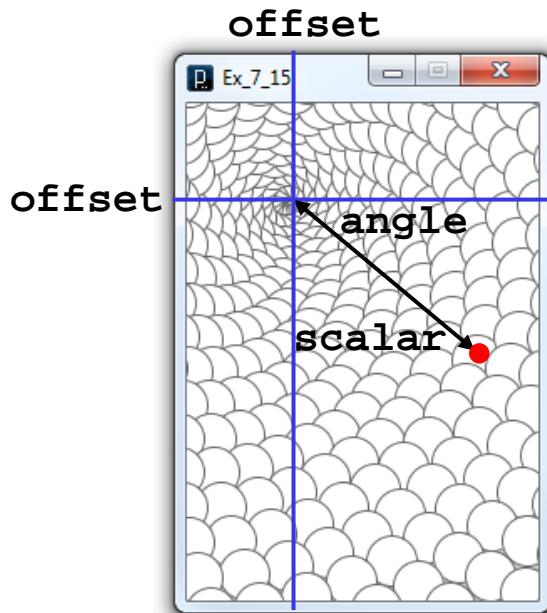
Vector graphics output, example 11-5



Vector graphics output example

Towards another larger example

- example 7-15 in the e-book of Casey Reas & Ben Fry
- adapted for PDF output and attractive color scheme



Vector graphics output, example 7-15

```
//Casey Reas & Ben Fry: Getting Started with Processing
//Ex. 7-15, p.104, adapted by Loe Feijs for PDF output.

import processing.pdf.*;

float angle = 0.0;
float offset = 60;
float scalar = 1.5;
float speed = .4;

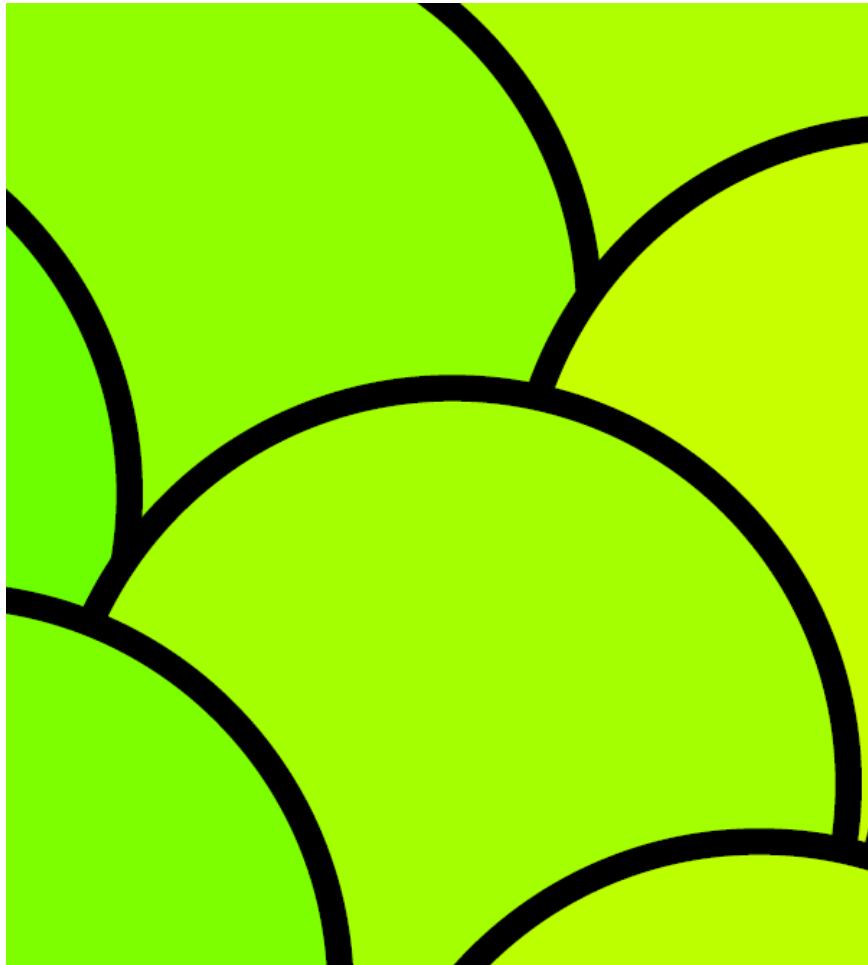
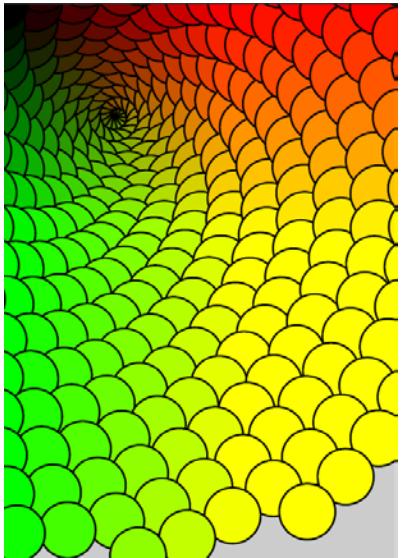
void setup() {
    //size(210,297);
    size(210, 297, PDF, "Ex-7-15.pdf");
    fill(0);
    smooth();
}

}
```

Vector graphics output

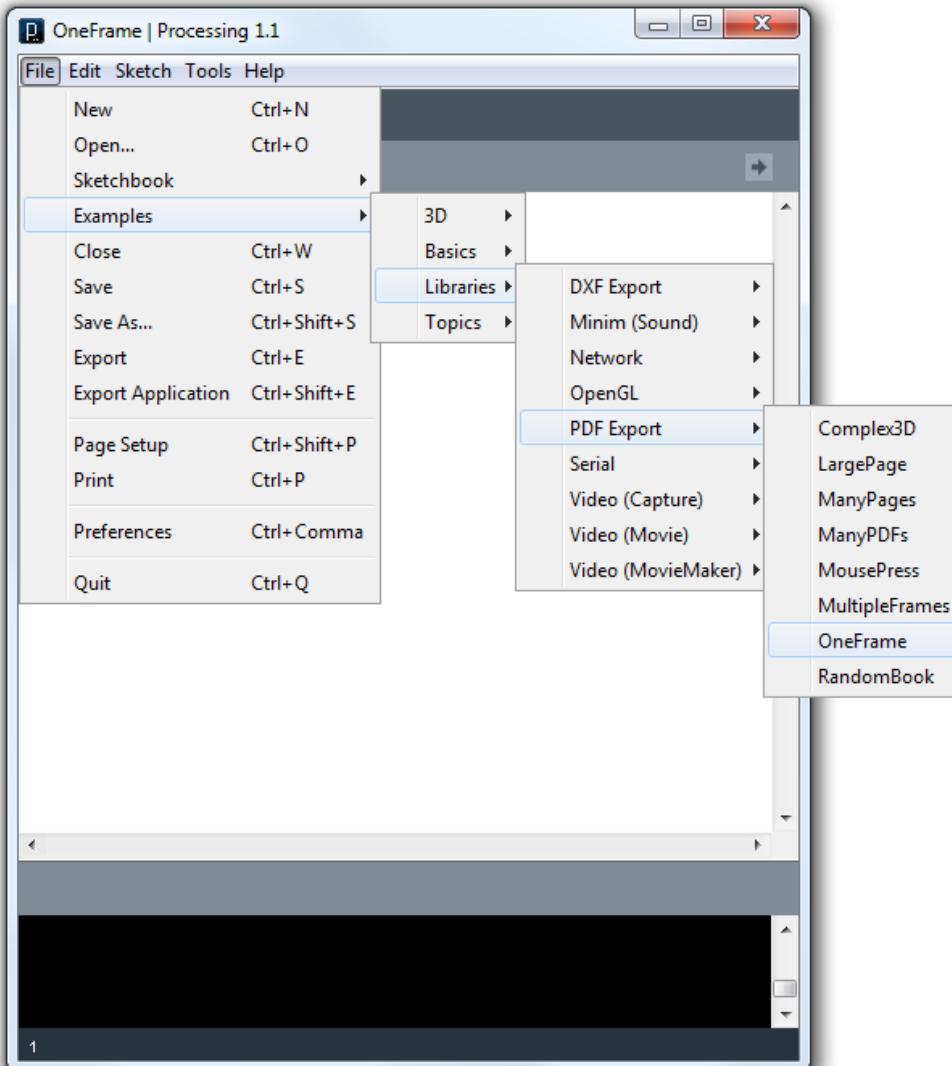
```
void draw() {  
  
    float x = offset + cos(angle) * scalar;  
    float y = offset + sin(angle) * scalar;  
  
    fill(2*x,2*y,0);  
    ellipse(x,y,30,30);  
  
    angle += speed;  
    scalar += speed;  
  
    if (x>300) exit();  
}
```

Vector graphics output



Vector graphics output example

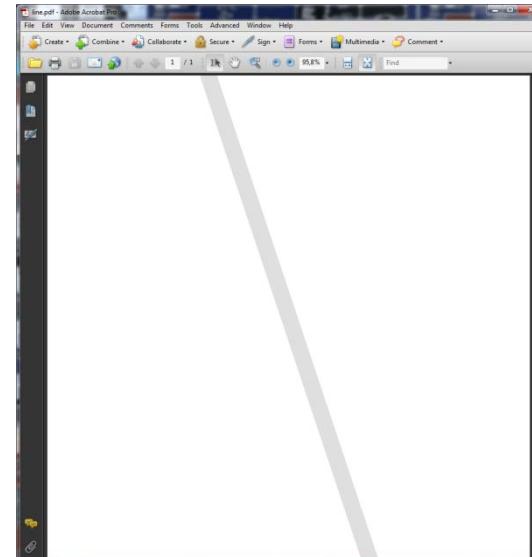
Towards
hundreds
of larger
examples



Vector graphics output

```
/**  
 * One Frame.  
 *  
 * Saves one PDF with the contents of the display window.  
 * Because this example uses beginRecord, the image is  
 * shown on the display window and is saved to the file.  
 */
```

```
import processing.pdf.*;  
  
size(600, 600);  
beginRecord(PDF, "line.pdf");  
  
background(255);  
stroke(0, 20);  
strokeWeight(20.0);  
line(200, 0, 400, height);  
  
endRecord();
```

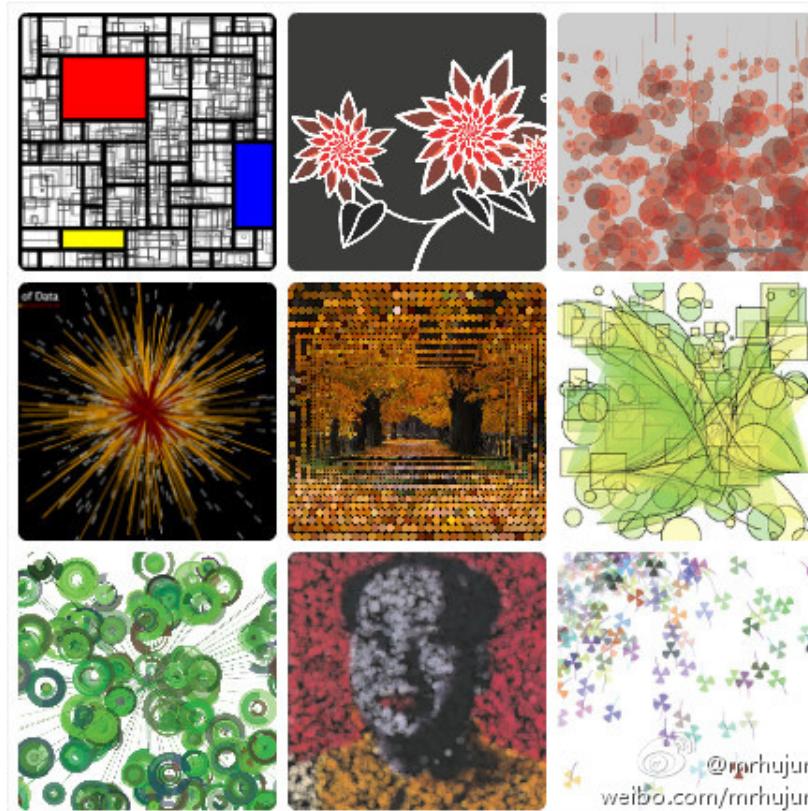


Homework

Homework

wiki.id.tue.nl/creapro/CreativeProgrammingAssignment201202

- **Challenge:** Creating static visual arts. At the end of this part, every student is expected to be able to
 - create an artistic poster that demonstrates beauty and complexity, using Processing
 - print the result and present it on a form board (A3 only please), for the interactive session in the third week.
- Examples for inspiration



- What to be delivered:
 - Source code. Please zip the sketchbook and deliver the zip file. Please use the **zip format** only.
 - PDF of the print.

Homework

- Create a fantastically good-looking PDF poster
- Deliver high resolution vector graphics art
- Upload zipped sketchbook to the dropbox
- Learn from examples on processing.org
- Use libraries and re-use other's code
- But always mention your sources
- And do not infringe copy-rights
- Prepare to explain your code
- Print your poster on A3
- Bring it wednesday 29th