

# Creative Programming

The background of the slide features a photograph of a playground. A large, bright yellow slide with red horizontal bands is the central focus. In the foreground, a green metal fence runs across the frame. Several people are visible: a man in a light-colored shirt and dark pants is walking towards the left, and a woman in a red jacket and light-colored pants is walking towards the right. The image is partially covered by a semi-transparent orange overlay on the left side, which contains the title text.

**TU** / **e**

Technische Universiteit  
**Eindhoven**  
University of Technology

**Where innovation starts**

# Assignors / Assistants



Loe Feijs



Peter Peters



Erik Van der Spek



Jun Hu



Mathias Funk

Ruben van Dijk  
Ilse Pouwels  
Sergej Zwaan

# Processing: After this course

- Use the processing environment and:
  - - create programs ... that run
  - - ... that draw pictures
  - - ... that display animations
  - - ... that display interactive animations
  - - ... that animate interactive objects
- last but not least: make all of these work together as you like ... great freedom to create

# Processing: After this 1<sup>st</sup> lesson

- **Start processing.**
- **Run your first program in processing**
- **Write programs that create various static objects i.e. “pictures”**
- **Change these programs to change the pictures.**
- **Understand how the pictures change when you change the program.**
- **Have a first idea about creating interactive objects.**

# After 1<sup>st</sup> lesson:

## What should you understand ?

- **Why processing (and programming in general) is interesting and important for you as a designer**
- **what syntax is**
- **what expressions are**
- **what (basic) types and variables are**
- **what semantics is and how to look it up**
- **how to think about programs (a little)**

# Downloading processing...

- Go to <http://processing.org>
- Download Processing 2.2.1
- Create a directory to store the Processing program stuff (e.g. C:\Users\<username>\Programs)
- Extract the zip file into that directory
- Create a shortcut to Processing on your desktop
- Create a directory to store your sketches (e.g. C:\Users\<username>\Documents\Processing) and configure Processing to use that sketches folder...

# Before you start ...

## Experience some Examples

- Open menu:
- File|Examples|Topics|Interaction|
- run: Follow 1
- run: Follow 2
- run: Follow 3



# A little experiment ...

Look at the chart: say the  
Color not the word

Black	Blue	Green
White	Green	Red
Green	Aqua	Yellow
Yellow	Pink	Tan
Red	Yellow	White

Example produces a Left\Right brain conflict

The right brain tries to say the color

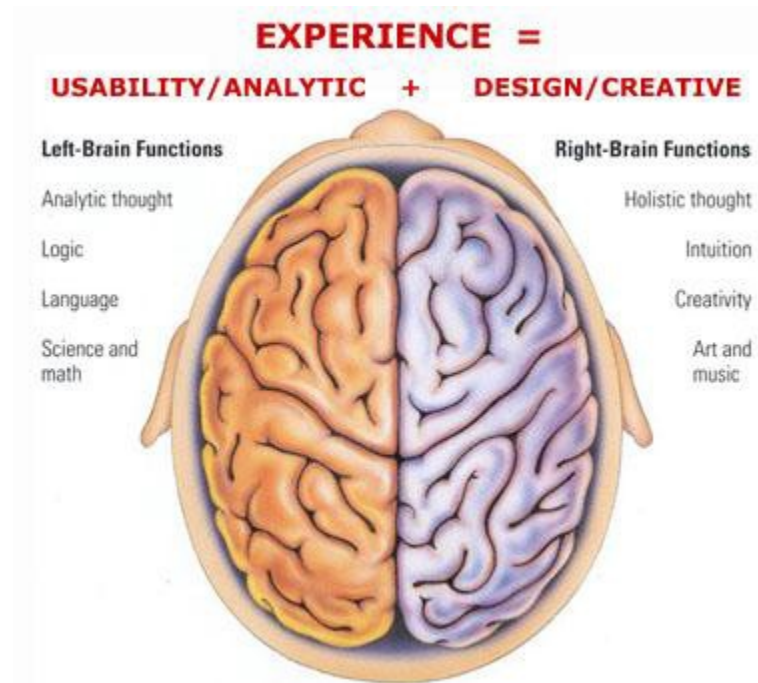
The left brain tries to read the color

<http://OfficeSpam.ChattaBlogs.com>





# Left & Right brain



# Left versus Right

- **abstract objects represented in (programming) language are easy to change and to duplicate but are not immediately graspable or visible**
- **concrete objects that are created in matter can be inspected and manipulated, but are more difficult to change and to duplicate.**

# We want best of both worlds

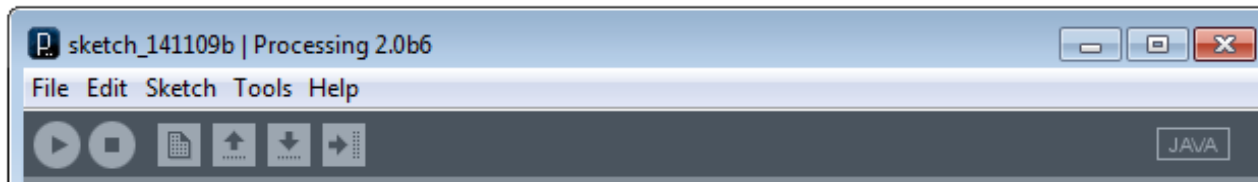
- **define and create objects through language**
- **grasp and inspect objects through senses.**
- **Processing can execute abstract instructions in a computer language and translate these into something that you can experience through the senses.**

# Processing language : How does it work?

- **processing is an imperative language: that means you use the language to give commands**
- **The computer creates the application by executing the commands one after the other ... it is a sequential language**
- **compare with written music : parallel (orchestra)**
- **can also be done in programs ...but very difficult.**

# Lets Start Programming...

- Click on the processing icon ...
- Window opens with: Run, Stop, New, Open, Save, Export Application.



# First program “Hello you”

- `print("hello you");`
- `print("hello");`
- `print("you");`
- `println("commands are separated by semicolons");`
- `print(5*3);`
- `print("We count"+ 2+1+5+10 + "characters");`
- `print("We count"+ (2+1+5+10) + "characters");`



# Correctness : 3 Levels

- **Syntax (language form) : wellformed grammatical expressions: orders of brackets, semicolons, operators, letters and numbers.**
- **Types (kinds of things) : distinguish apples from oranges**
- **Semantics (meaning) : does the program do what you want ?**

# Correctness : 3 Levels

- **Berlage boult the Schröder house**
- **Berlage build the Schröder house**
- **Berlage built the Schröder house**
- **Rietveld built the Schröder house**





# Syntax : wellformed or not ? Try some examples ...

- `print("hhhh ggg");`
- `print("a"); print("b");`
- `print(8); {print(8);}`
- `{{{print(8);}}}`
- `print("hello you");` → syntax error: perhaps a missing right parenthesis
- `// this is just a comment ....`
- `print("jjjhhh ) ")` → unexpected token: null
- `print("a") print("b")` → syntax error: maybe a missing semicolon
- commands can contain expressions ....

# Expressions can be nested ...

- `3*4`
- `sin(3*4)`
- `sin(3* tan(5) / exp(sin(cos(0.45454))))`
- `"abcd"+"efgh"`
- `"abcd" + ("ef" + "gh")`

# Types

- **String** "hhhheeeee" + "aaa" + "nnbn99 bnb"
- **int** 8 9\*97978787 1-9988989
- **float** 2333.5555
- sin( -3 \* 5677.455)
- 3.4e+38
- **basic types are:** String, float, int, boolean, char, byte

# variables

- A variable is a named location where a certain type of value can be stored
- declare; initialize, use, scope.
- `String anExample;`
- `anExample = "fghjkl";`
- `anExample = anExample + anExample;`

# Variable 2

- `int multiplier = 5;`
- `multiplier = multiplier + 4;`
- `float pi = 3.1415926535897932;`
- `print(multiplier * pi) ;`

# SEMANTICS

- The meaning of the command; this may depend on type.
- `int myAge;`
- `myAge = 8;`
- `print(myAge * 8 );`
- `print(" 8 + 8 ");`
- `print("I count"+ 1+1+5+10 + "characters");`
- `print(myAge+ (1+1+5+10) );`
- (to be continued)

# How to think about commands:

- setting up a picture, or later a stage, using predefined primitives
- first start with a static picture:
- create empty picture with command `size`:
  - `size(200,200);`
  - Next: specify what you put where:
  - you can use various standard primitives with parameters:
- `point(20,45);`
- `line( 0,0,100,150);`

# Example ...

- go to menu:
- Example|Basics|Form|
- run: PointsLines
- what is semantics (meaning)
- of : stroke(153) ?
- : background(0) ?



# Semantics

- To find the meaning look for the (informal) specifications ..
- Select and right click on “stroke” to find out ...
- choose: find in reference
- Same on “background” to find out ...
- these commands specify drawing parameters

# Specify drawing parameters ...

- `stroke(255);`      255 = white, 0 = black in between are shade of gray ..
- `background(200, 23, 130);`    (you can use color)
- `noStroke()` ...etc      various primitives
- Processing reference

# Also two dimensional shapes are possible ...

- `rect(20,20,60,120);`
- `ellipse( 50,50,30,99);`
- **Example|Basics|Form|**
- **run: ShapePrimitives**

# Interactive drawings ...

- create a stage with :
- `void setup() {`
- `size(200, 200);`
- `}`
  
- then you can draw ... continuously ...
- with the `draw` command ..
- For example ...

# Interactive drawings ...

```
void setup() {  
    size(200, 200);  
    smooth();           // makes forms smoother  
    strokeWeight(2);    // how thick lines are  
    stroke(255);        // color of lines (white)  
}  
  
void draw() {  
    background(mouseX, mouseY, 80); // background color  
    line(200, 0, mouseX, mouseY);  
    line(mouseX, mouseY, 0, 200);  
}
```

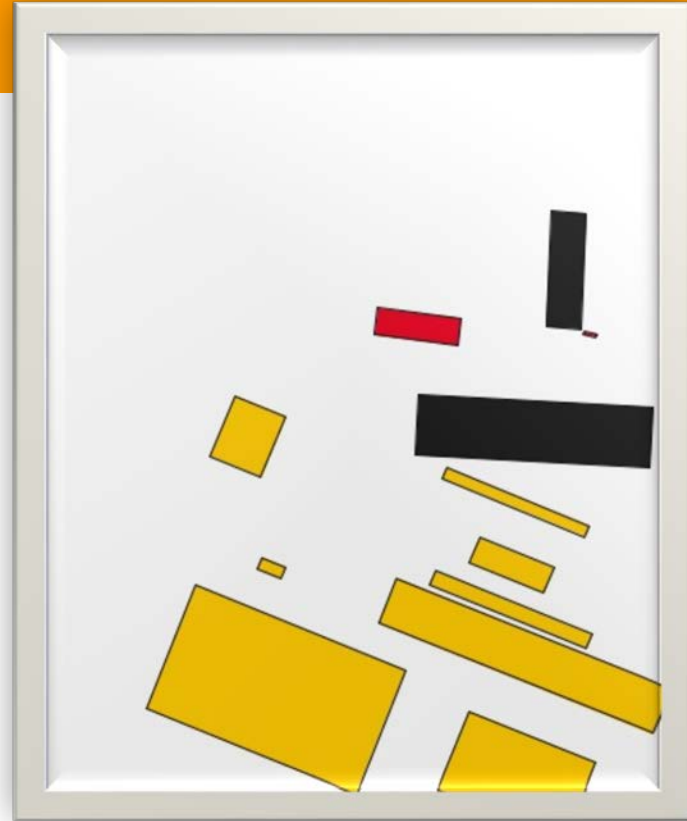
# Remark on style ...

- proper indentation
  - comprehensible comments (LOTS!)
  - (using Auto Format in Tools menu, if you like it, ^T)
- 
- balanced pictures ...
  - beautiful movements ...

# Where we will be in three weeks?



Computer Generated 2012



Computer Generated 2012



Kasimir Malevich, Suprematist  
Painting: Airplane Flying, 1915.



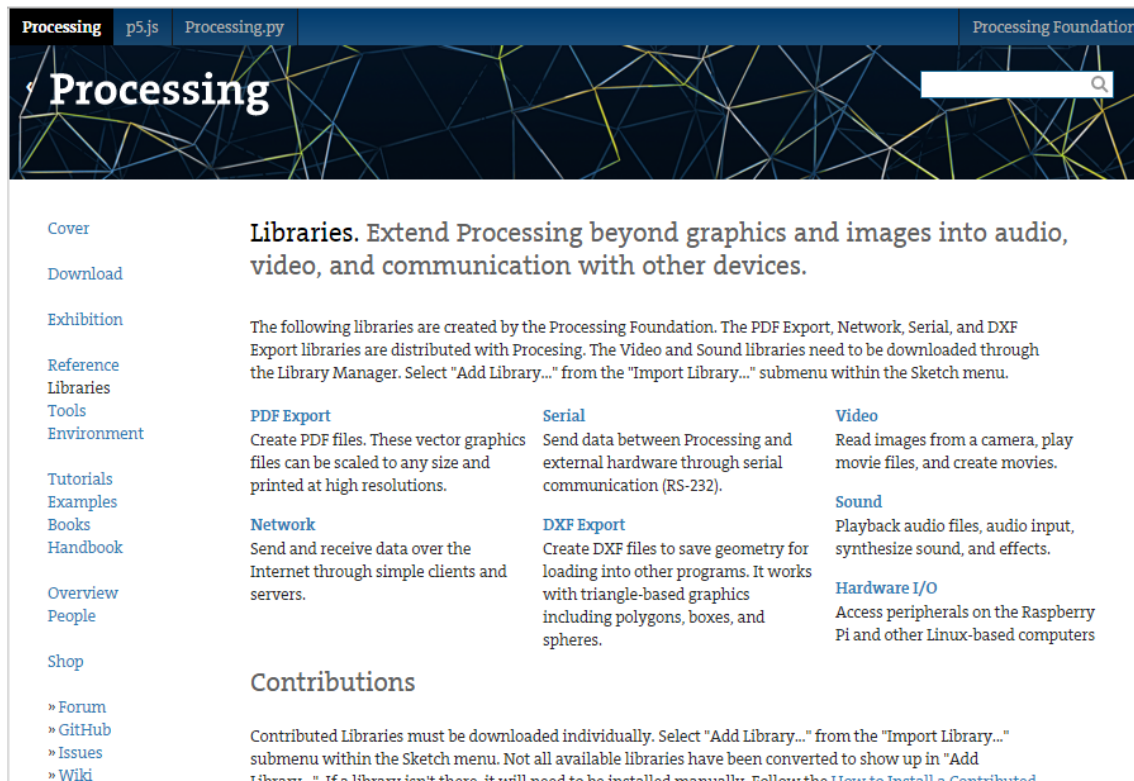
Kasimir Malevich, Suprematist  
Painting: eight red rectangles,  
1915.



# Processing libraries

- **Reference**

- <https://processing.org/reference/libraries/>



The screenshot shows the Processing.org website's 'Libraries' page. The header includes navigation links for 'Processing', 'p5.js', 'Processing.py', and 'Processing Foundation'. The main content area is titled 'Libraries. Extend Processing beyond graphics and images into audio, video, and communication with other devices.' It explains that libraries are created by the Processing Foundation and are distributed through the Library Manager. Below this, there are four columns of library descriptions: PDF Export, Network, Serial, DXF Export, Video, Sound, and Hardware I/O. A 'Contributions' section at the bottom states that contributed libraries must be downloaded individually and are not all available in the Library Manager.

Processing p5.js Processing.py Processing Foundation

## Processing

Cover

Download

Exhibition

Reference Libraries

Tools

Environment

Tutorials

Examples

Books

Handbook

Overview

People

Shop

» Forum

» GitHub

» Issues

» Wiki

### Libraries. Extend Processing beyond graphics and images into audio, video, and communication with other devices.

The following libraries are created by the Processing Foundation. The PDF Export, Network, Serial, and DXF Export libraries are distributed with Processing. The Video and Sound libraries need to be downloaded through the Library Manager. Select "Add Library..." from the "Import Library..." submenu within the Sketch menu.

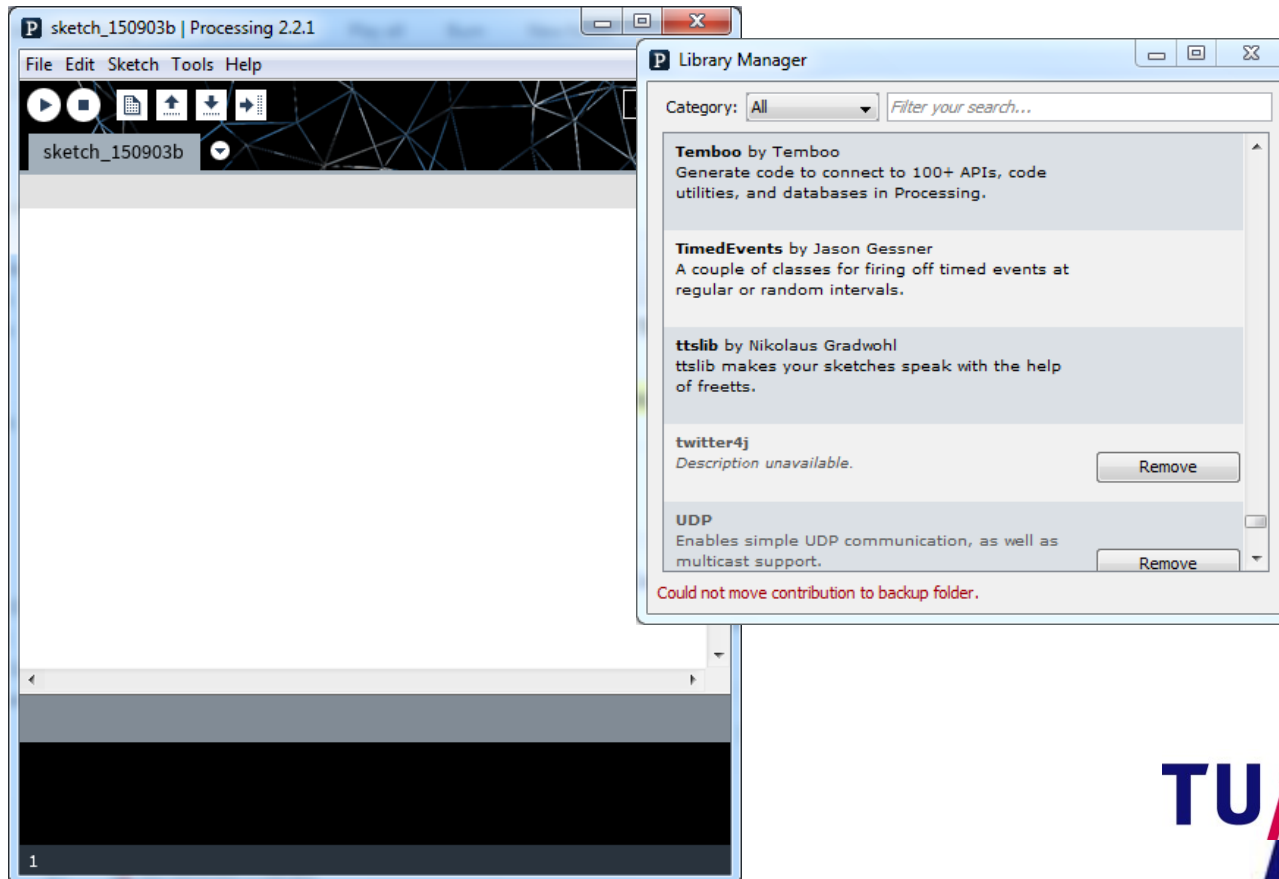
<b>PDF Export</b> Create PDF files. These vector graphics files can be scaled to any size and printed at high resolutions.	<b>Serial</b> Send data between Processing and external hardware through serial communication (RS-232).	<b>Video</b> Read images from a camera, play movie files, and create movies.
<b>Network</b> Send and receive data over the Internet through simple clients and servers.	<b>DXF Export</b> Create DXF files to save geometry for loading into other programs. It works with triangle-based graphics including polygons, boxes, and spheres.	<b>Sound</b> Playback audio files, audio input, synthesize sound, and effects.
		<b>Hardware I/O</b> Access peripherals on the Raspberry Pi and other Linux-based computers

### Contributions

Contributed Libraries must be downloaded individually. Select "Add Library..." from the "Import Library..." submenu within the Sketch menu. Not all available libraries have been converted to show up in "Add Library..." If a library isn't there, it will need to be installed manually. Follow the [How to Install a Contributed Library](#) page.

# Installing other libraries

Sketch → Import Library → Add library



# Standard PDF export library

- pdf

```
import processing.pdf.*;
```

← Import library

```
size(600, 600);
```

```
beginRecord(PDF, "line.pdf");
```

```
background(255);
```

```
stroke(0, 20);
```

```
strokeWeight(20.0);
```

```
line(200, 0, 400, height);
```

Use library functionality

```
endRecord();
```

# Information

**All up-to-date information will be visible in the wiki:**

- <http://wiki.id.tue.nl/creapro/CreativeProgramming201511>

**All processing language info can be found on**

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

**Inspiration**

- <http://www.openprocessing.org/>

# Books

## Must-have

*Getting Started with Processing*, by Casey Reas, Ben Fry, e-Book and hard copy available from O'Reilly

## Recommended-to-have

*Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction*, Daniel Shiffman.

Published August 2008, Morgan Kaufmann. 450 pages. Paperback, Available from LUCID, or from Amazon

*Programming Interactivity: A Designer's Guide to Processing, Arduino, and openFrameworks* (Paperback) by Joshua Noble (Author). **Very good one, covers many topics in Competency II.** Available from LUCID. Also see

<http://programminginteractivity.com>

*Processing: Creative Coding and Computational Art (Foundation)*

Ira Greenberg (Foreword by Keith Peters)., Published 28 May 2007, Friends of Ed. 840 pages. Hardcover. Available from LUCID

*Making Things Talk: Practical Methods for Connecting Physical Objects*

Tom Igoe, Published 28 September 2007, O'Reilly. 428 pages. Paperback. Available from LUCID

# Some getting-started excercises for you

## Statistics:

- Make a program with variables containing the ages of you and some of your friends
- Let the program calculate the average and the standard deviation and print it orderly using `print` and `println`

<http://www.mathsisfun.com/data/standard-deviation.html>

## Geometry:

- Make a program with at least five `int` or `float` variables to be used as parameters
- Let the program create an abstract geometric composition using these parameters
- Play with the parameters to optimize aesthetic balance

End of slide show

# Design Process: Integrate various skills

