Creative Programming

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

!!! Changes !!!

All FRIDAY sessions (13:45-15:30) will be in:

AUDITORIUM Lecture room 03

Mondays stay in Gemini…

These and future changes will be visible in the wiki:

- http://wiki.id.tue.nl/creapro/CreativeProgramming201509

Processing: After the 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

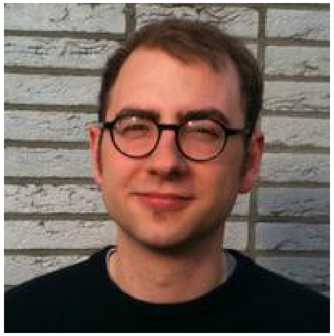
Assignors / Assistants

Loe Feijs

Peter Peters

Erik Van der Spek

Jun Hu



Mathias Funk

Ruben van Dijk
Ilse Pouwels
Sergej Zwaan

Technische Universiteit
Eindhoven
University of Technology

After this 1st lesson: what can you do

- 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 1st 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

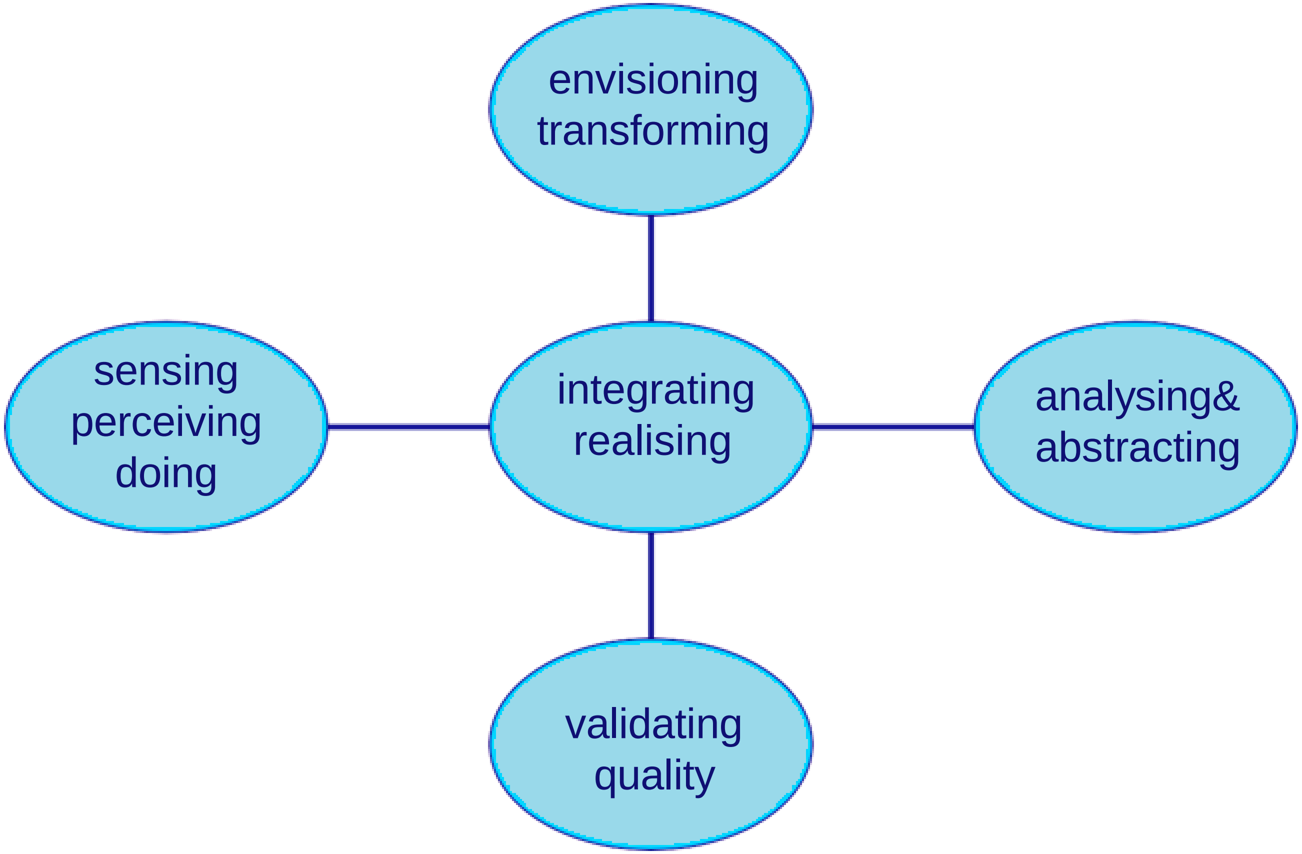
- Create a shortcut to Processing on your desktop

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

Design Process: integrate various skills


envisioning
transforming

sensing
perceiving
doing

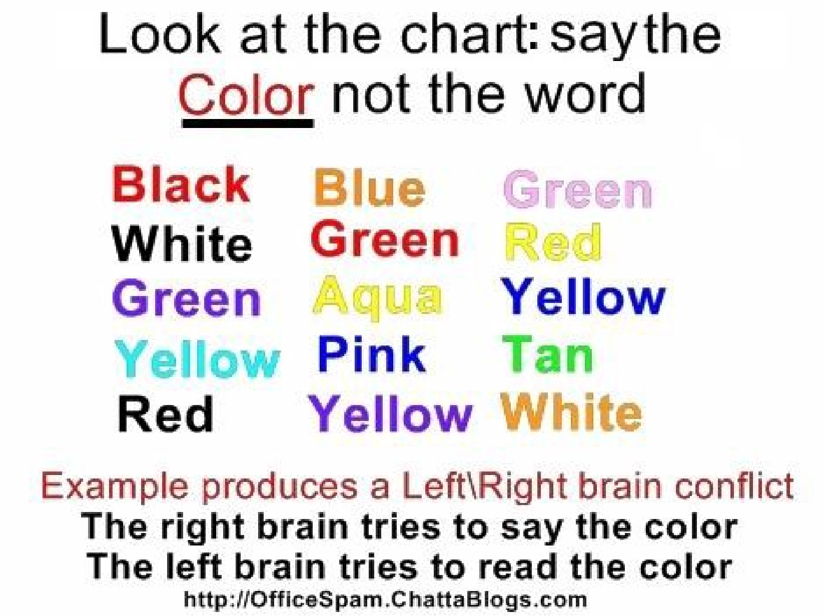
integrating
realising

analysing&
abstracting

validating
quality

Hummels & Frens: Reflective Transformational Design model

A little experiment …



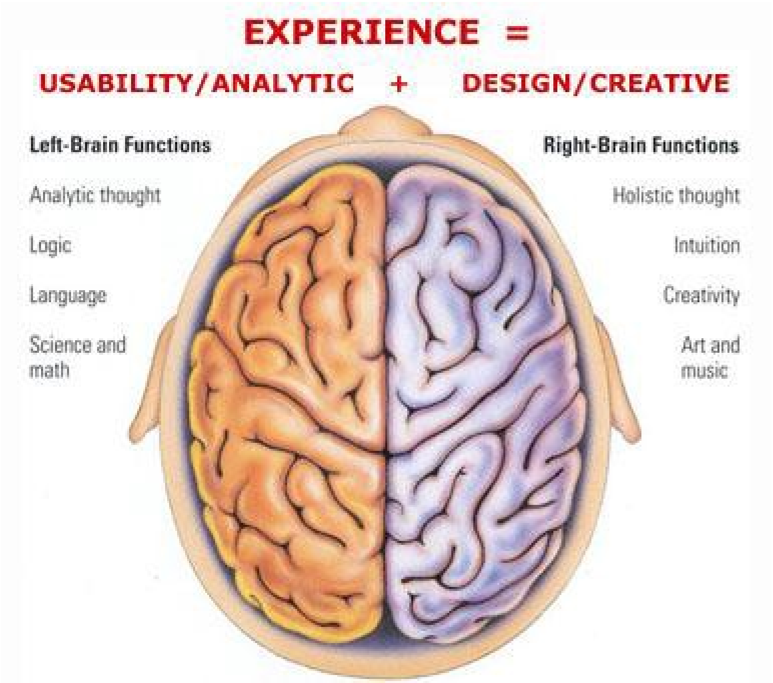
Look at the chart: say the
Color not the word

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

Stresstest

Need to integrate Left & Right brain



Source: http://www.tunghai74.org/letters/Brain-left-right.htm

Left versus Right

- abstract objects that are represented in language are easy to change and to duplicate but are not immediately graspable or visible, and cannot be placed in the relevant context

- concrete objects that are created in matter can be inspected and manipulated easier, 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.**

Programming languages : How does it work?

- **processing is an** imperative **language: that means you use the language to give** commands

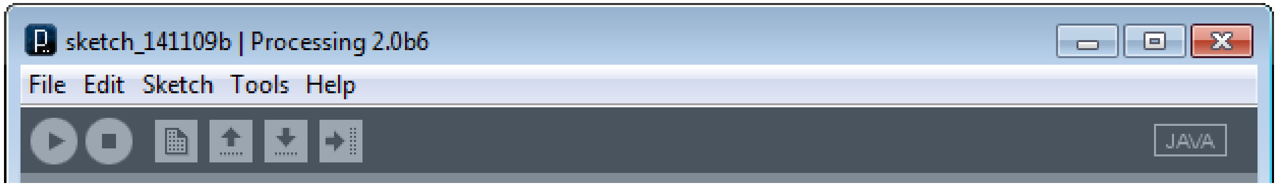
- **The computer creates the application by executing the commands one after the other … it is a** sequential **language**

- **compare with written music : parallel (orchestra)**
- **can also be done in programs …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) : distinghuish 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

Image: Rietveld Schröderhuis HayKranen-20.JPG, Wikimedia Commons

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**

- **Idem on “`background`” to find out …**

- **these commands <u>specify drawing parameters</u>**

Specify drawing parameters …

- `stroke(255);` 255 = white, 0 = black in between are shade of gray ..
- `background(200,23,130);` (e.g. you can also 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 ...

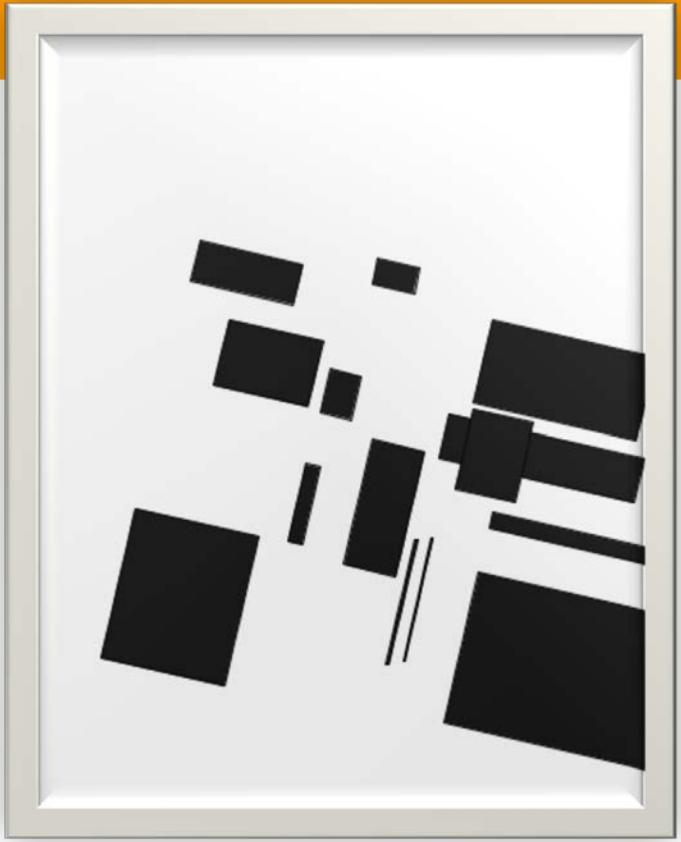
```
void setup() {
     size(200, 200);
     smooth();          // makes forms smoother
     strokeWeight(2);   // how thick lines are
     stroke(255);       // color of lines (white)
}

void draw() {
     background(mouseX, mouseY, 80); // background color
     line(200, 0, mouseX, mouseY);
     line(mouseX, mouseY, 0, 200);
}
```

Remark on style …

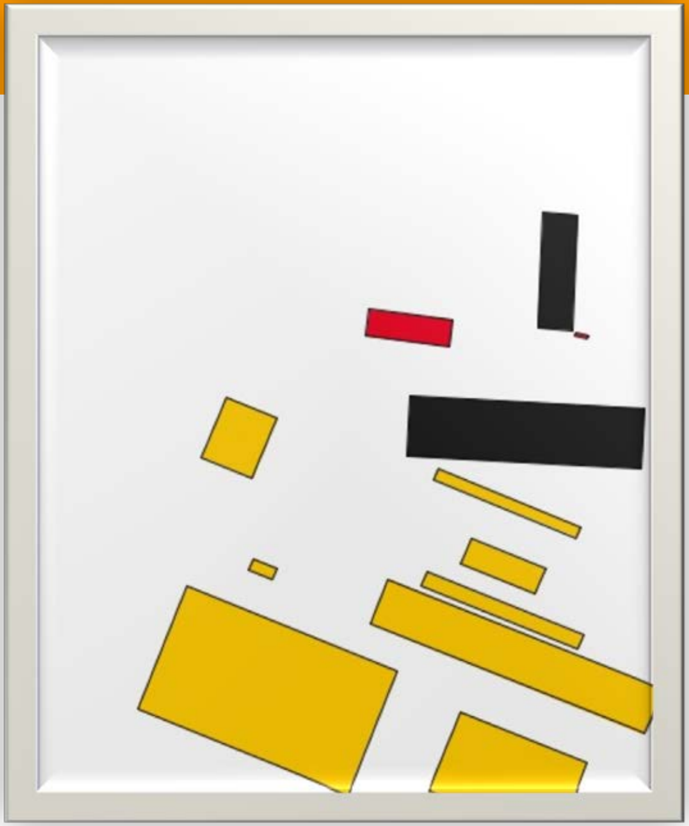
- proper indentation
- comprehensible comments (LOTS!)
- (using Auto Format in Tools menu, if you like it, ^T)

- balanced pictures …
- beautiful movements …

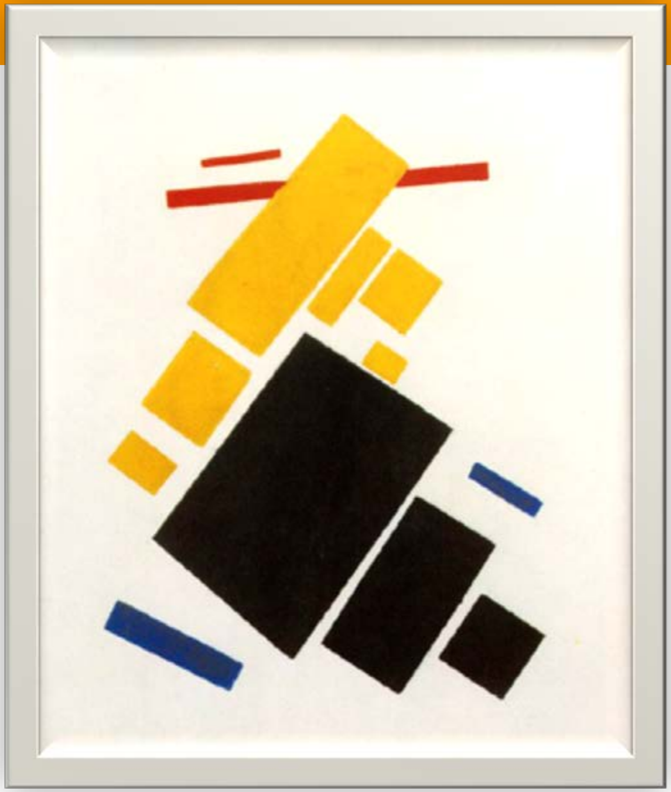
Where we will be in three weeks?



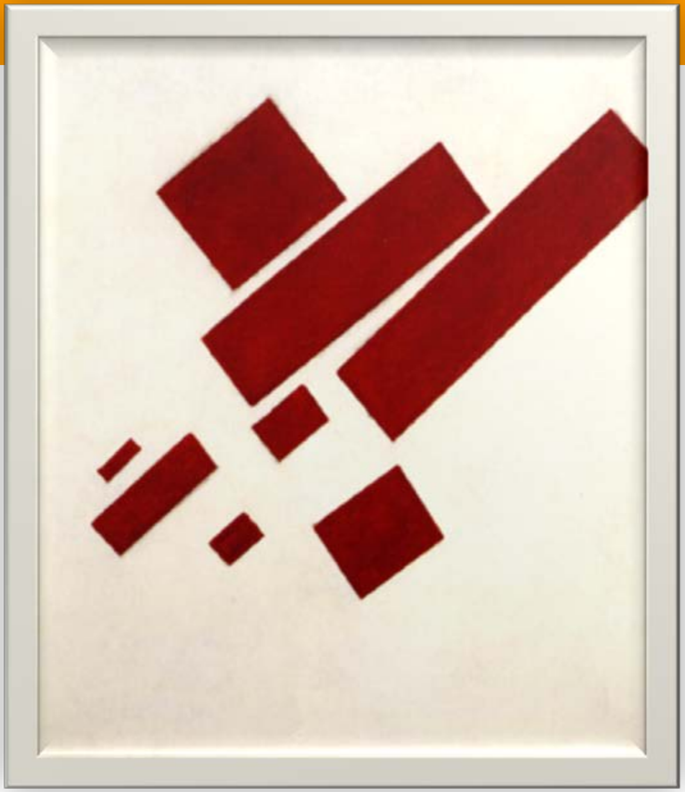
Computer Generated 2012



Computer Generated 2012



Kasimir Malevich, Suprematist Painting: Airlane Flying, 1915.



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

Some getting-started homework 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 optimise aesthetic balance