

Hello You

The Middle Path

The logo for Technische Universiteit Eindhoven, featuring the letters 'TU' in a large, bold, dark blue font, followed by a red diagonal slash, and then a smaller 'e' in the same dark blue font.

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

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



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 ? How to look it up?**
- **How to think about programs. (a little)**

Downloading processing...

Go to wiki created for the assignment:

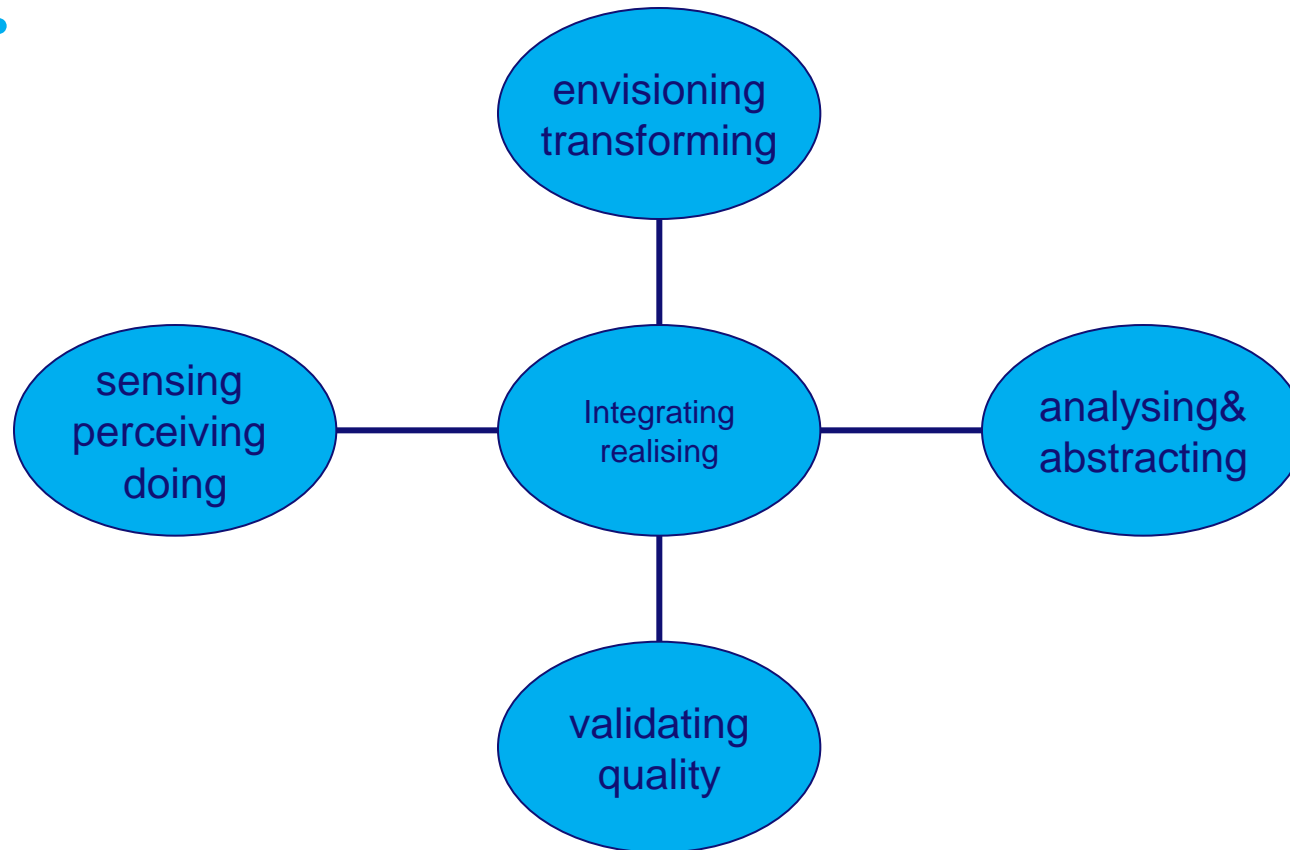
- <http://wiki.id.tue.nl/creapro>
- go there and click on:
- [Prepare your computer for the assignment](#)
- then click on the link:
- [Download processing.](#) (a stable release)
- Create a directory "Programs" on the C: disk, in the root. If "C:\Programs" exists already, skip this step.
- Extract the entire directory to C:\Programs (note, not "C:\Program Files"). if you are reinstalling Processing, remove the entire processing directory first.

Before you start ...

Experience some Examples

- Open menu:
- File | Examples | 3D and OpenGL | Form |
- run: CubicGrid
- Open menu:
- File|Examples|Topics|Interaction|
- run: follow 1
- run: follow 2
- run: follow 3

Design Process: integrate various skills



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

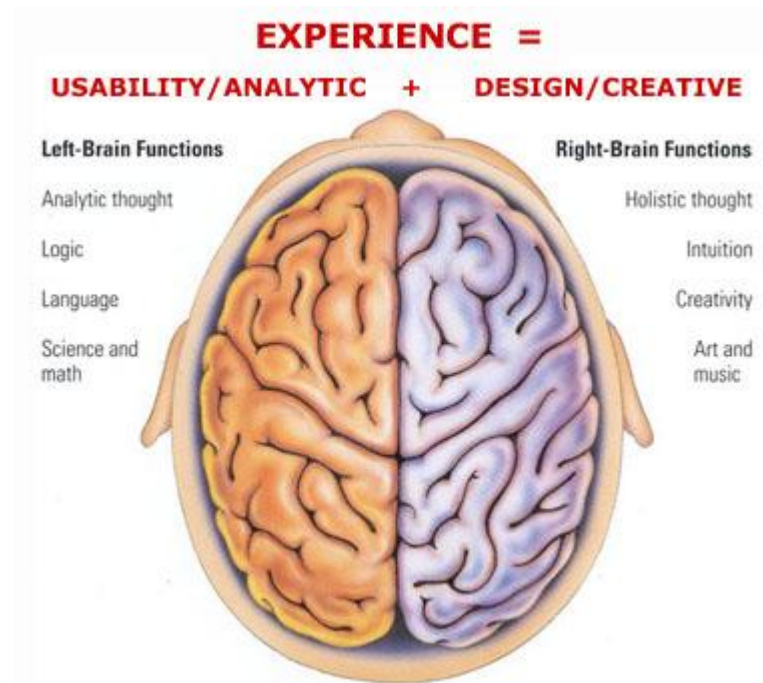
Example produces a Left\Right brain conflict

The right brain tries to say the color

The left brain tries to read the color

<http://OfficeSpam.ChattaBlogs.com>

Need to integrate Left & Right brain



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 and new,open,save,export. export makes applets.**

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 ?**

Syntax : wellformed or not ?

Try some examples ...

- `print("hhhh ggg");`
- `print("a") ; print("b") ;`
- `print(8); {print(8) ; } ; {{{print(8); }}} ;`
- `print("hello you") ;`
- `// this is just a comment`
- `print(" jjjhhh) "))` → syntax error: semi expected found)
- `print("a") print("b")` → syntax error: semicolon missing ...
- `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,**
- (to be continued ... can do)

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

Variable 2

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

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;`

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 ..
- doubleclick on “stroke” to find out ...
- choose : find in reference
- doubleclick 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) ;** (e.g. you can also use color)
- **nostroke()** ...etc various primitives
- **<C:\Programs\processing-0135\reference\index.html>**

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);`
- `}`
- the 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);**
- **}**

Final remark : Style ...

- - **proper indentation**
 - **comprehensible comments**
 - **(using autoformat in Tools menu?)**
-
- **balanced pictures ...**
 - **beautiful movements ...**