Generative Design Workshop

UDK Raum 4+5

Week 1/4

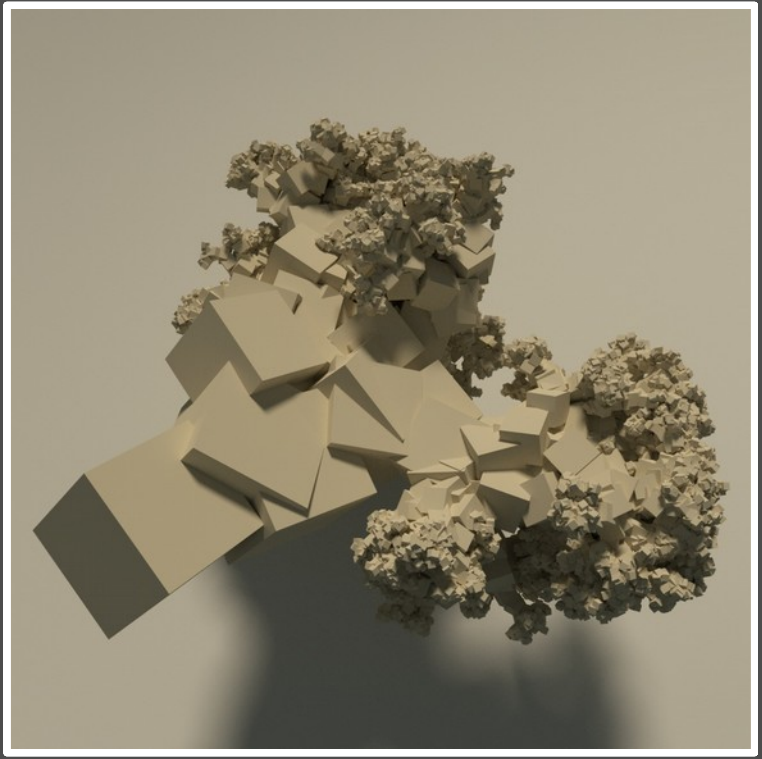
October, Friday 30th, 2015

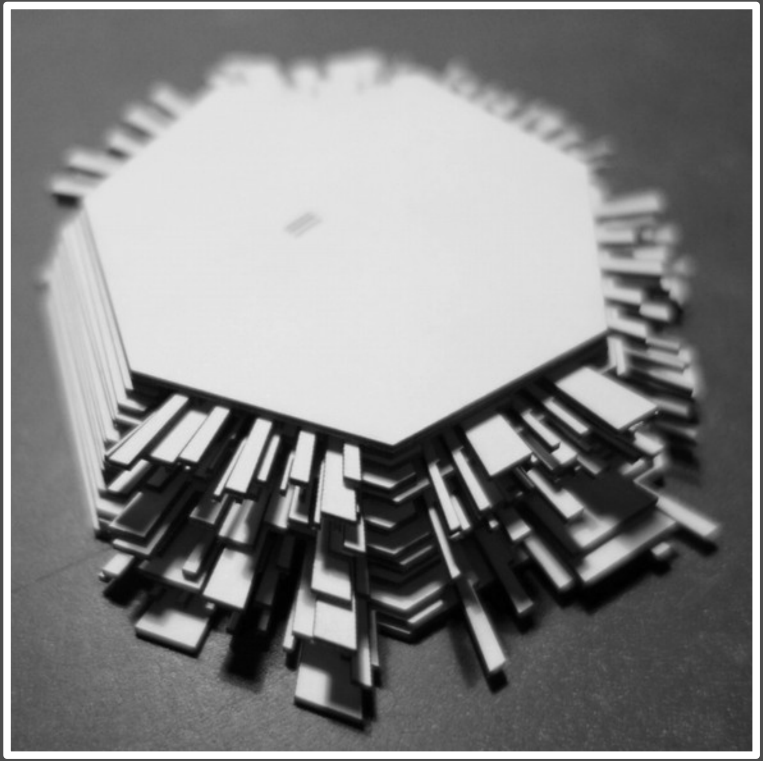
Goal of this 4 week workshop:

Learn to translate ideas into
short computer programs
to produce generative designs
in bitmap and vector file formats.

The results will be based on randomness,
user input, data or a combination of them.

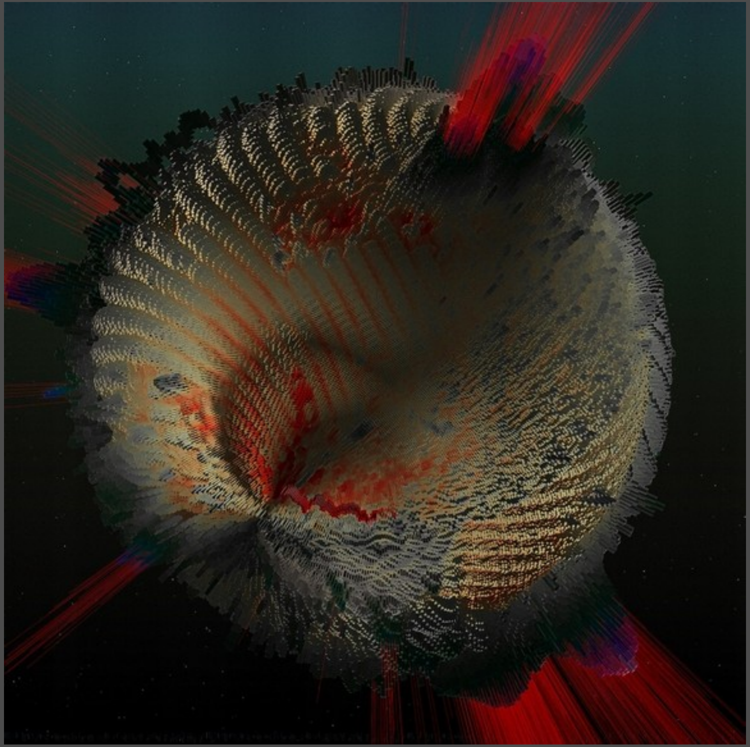
Examples 1/6


Generating complexity using simple shapes and simple rules. Shape created in Processing and rendered with Blender.


Data-based laser-cut object. Program reads values, exports SVG vector files. Each side represents one day of the week.

Examples 2/6

Random starting points. Draw lines, turning to avoid other lines.



Artistic 3D representation of a 2D microscope image of atoms.

Examples 3/6

Marker simulator generating random shapes

Program that tries to copy a photo using alternative photos

Examples 4/6

Design based on randomness and simple rules

Video-wall room with 3D generative structures, flocking motion and Twitter interaction

Examples 5/6

Drawing program controlled by a playing video clip

Crop of a visualization of an audio file

Examples 6/6

Laser engraved laptop, the design was generated based on video data

Randomize curves using the colors from a photograph

Reasons to design by writing code:

Enables aesthetics not yet offered by commercial software (freedom)

Faster production of variations

Allows the creation of complexity

Allows customizing and obtaining unique results

(Leads to observation and understanding of the world)

Where can I apply generative design?

Programming languages like Processing, OpenFrameworks, VVVV

Online in web pages (using JavaScript based frameworks)

Scripting in applications like Blender, Photoshop, Inkscape

What is Processing?

“Processing is a programming language,
a development environment, and an online community”

Reasons to use Processing:

Well documented

Many examples and libraries

A community around it

Very easy to get started

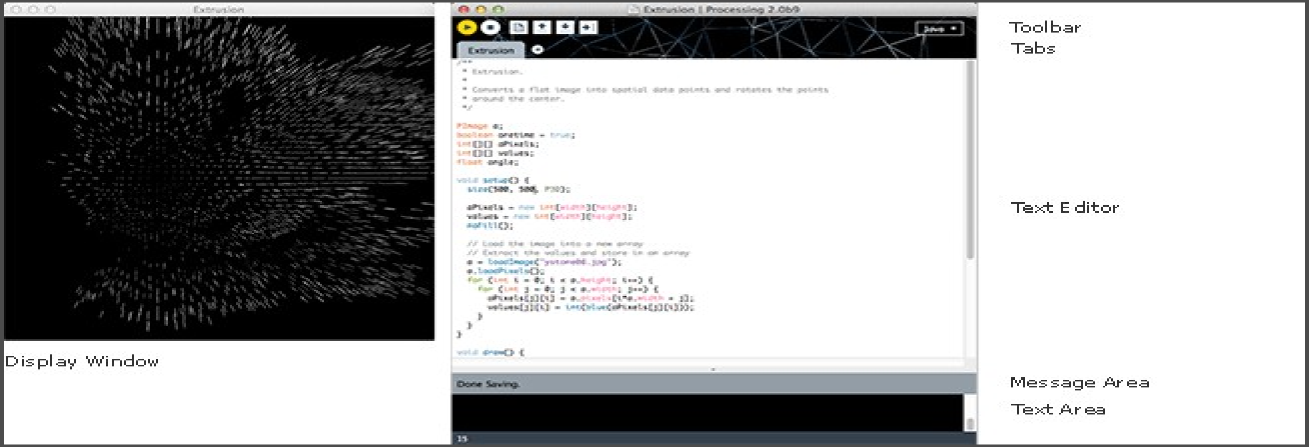
Free, Open Source and multi-platform

Can be used for professional work

The Processing Environment

“The Processing Environment includes a text editor,
a compiler, and a display window”

“It enables the creation of software
within a carefully designed set of constraints”



```
/**
 * Extrusion.
 *
 * Converts a flat image into spatial data points and rotates the points
 * around the center.
 */

PImage a;
boolean onetime = true;
int[][] aPixels;
int[][] values;
float angle;

void setup() {
  size(500, 500, P3D);

  aPixels = new int[width][height];
  values = new int[width][height];
  noFill();

  // Load the image into a new array
  // Extract the values and store in an array
  a = loadImage("ystone08.jpg");
  a.loadPixels();
  for (int i = 0; i < a.height; i++) {
    for (int j = 0; j < a.width; j++) {
      aPixels[j][i] = a.pixels[i*a.width + j];
      values[j][i] = int(blue(aPixels[j][i]));
    }
  }
}

void draw() {
```

```
Done Saving.
```

https://processing.org/reference/environment/

Processing help and links

IDE
Reference
Examples menu

Online
http://processing.org
http://forum.processing.org
http://openprocessing.org
http://funprogramming.org
http://fyprocessing.tumblr.com

"Hello World" program - drawing a circle

```
ellipse(50, 50, 20, 20);
```

1. Look at the reference for ellipse().
2. Change the size of the circle.
3. Place the ellipse at the top left corner, then at the center.
4. What happens if you remove the semicolon?
5. What happens if you remove the comma? The parenthesis?
6. Study https://processing.org/tutorials/drawing/

ellipse() is a function

Functions perform tasks. To call a function (to run it, to execute it),
we write its name followed by a pair of parenthesis.
Inside the parenthesis we include zero or more arguments, separated commas.
The line ends with a semicolon.

The Processing reference details how many arguments to use and their meaning.
The ellipse function expects 4 arguments:

```
ellipse(30, 30, 40, 40);
```

Choosing colors

```
size(400, 400);
fill(#FF0000);
stroke(#000000);
ellipse(200, 200, 50, 50);
```

Some functions have an immediate effect on the screen (ellipse).
Others like fill() and stroke() affect following drawing operations.

1. Use the Processing color selector to choose colors.
2. Try rect(), line(), strokeWeight() and background() and create a simple composition.

Comments

```
// beautiful
size(400, 400);
background(0, 0, 0);
fill(200, 100, 0);
rect(0, 0, 200, 200);
```

Lines that begin with // are comments. The computer ignores them.
Use comments to leave explanations to yourself.

1. What do the values “0, 0, 0” stand for?
2. Put 3 small rectangles inside the existing rectangle.

Printing

```
println(100);
println("hello");
println(70 + 30);
println(width);
```

println( ) is useful for finding out what is happening in your program.
It displays text in the console (not in the main screen).

Randomness

```
// print a random number between 0 and 100
println(random(100));
// print a random number between 1000 and 2000
println(random(1000, 2000));


size(600, 400);
// print display dimensions
println(width, height);

// ellipse at random position
ellipse(random(width), random(height), 20, 20);
```

1. Create a composition using different drawing functions and randomness.

Saving an image

```
ellipse(width/2, height/2, 40, 40);
save(“circle.png”);
```

The save() function saves an image to your hard drive.

Tip: to save an image with a unique name, you can use

```
save(System.currentTimeMillis() + “.png”);
```

Generative Design Workshop

October / November 2015 at UDK - Berlin

Abe Pazos
@fun_pro | fun@funprogramming.org
http://hamoid.com | http://funprogramming.org