

[dm1_04]
Introduction to
Design Computing:
4ACI7A1

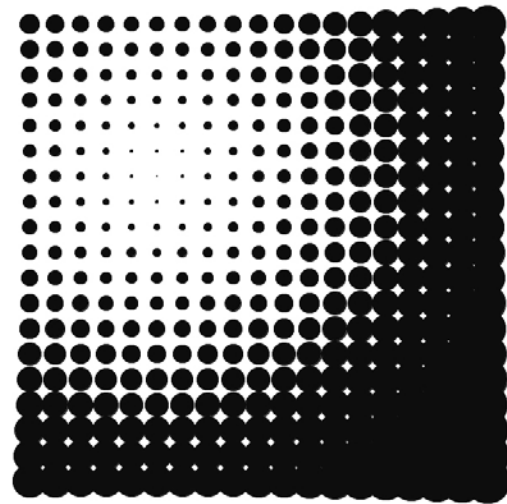
Processing

This week we take our first look at *Processing* – a scripting language for graphic and interactive media.

Although it seems like a separate programming language, *Processing* is in fact essentially a set of Java classes (functions and objects). It is also open source (so its free) and there are many additional libraries with extended functionality, that you can use in your own projects.

A key concept, central to the way *Processing* is structured, is object-oriented programming. The idea is that portions of code written by yourself, or other people, can be incorporated into new projects without having to worry too much about how they work. In this way projects can be assembled from smaller components to make a much larger and more complex whole.

Richard Difford
difforr@wmin.ac.uk



We will start by looking at some basic functions and next week we will move on to look at objects and interactivity. In future weeks we will also look at some basic electronics and working with *Processing*, *Arduino* and physical computing:

- *Processing* syntax and functions
- An introduction to object oriented programming and interactivity
- Basic electronics and Arduino
- *Processing* and *Arduino* - serial communication
- Video/ computer vision (possibly OpenCV)

Coursework:

Using the abstract compositions you have made, (see last week's brief) – begin exploring how you can construct these forms and composition using Processing.

References:

www.processing.org

Casey Reas and Ben Fry, *Processing: A Programming Handbook*, MIT Press, 2007.

Kostas Terzidis, *Algorithms for Visual Design*, Wiley, 2009.

//Step01- Just Drawing

//VARIABLES

```
float xcoord;  
float ycoord;
```

//SETUP

```
size(600,600);  
smooth();  
noFill();  
strokeWeight(1.5);
```

//DRAWING

```
background (27,27,77);  
xcoord=200;  
ycoord=200;  
stroke(255);  
ellipse(xcoord,ycoord, 10, 10);
```

//Step02- Translation

//VARIABLES

```
float xcoord;  
float ycoord;  
float division;  
float xdivider;  
float ydivider;
```

//SETUP

```
size(600,600);  
division=6;  
//println( width/division);  
//println( height/division);  
xdivider=width/division;  
ydivider=height/division;  
smooth();  
noFill();  
strokeWeight(1.5);
```

//DRAWING

```
background (27,27,77);  
xcoord=xdivider;  
ycoord=ydivider;  
  
pushMatrix();  
  
translate(xcoord, ycoord);  
stroke(255);  
ellipse(0,0, 10, 10);  
//ellipse(0,0, 20, 20);  
  
popMatrix();
```

//Step03- Draw Grid

//VARIABLES

```
float xcoord;  
float ycoord;  
float division;  
float xdivider;  
float ydivider;
```

//SETUP

```
size(600,600);  
division=6;  
println( width/division);  
println( height/division);  
xdivider=width/division;  
ydivider=height/division;  
smooth();  
noFill();  
strokeWeight(1.5);
```

//DRAWING

```
background (27,27,77);  
//xcoord=xdivider;  
//ycoord=ydivider;
```

```
for (xcoord=xdivider; xcoord<=width-xdivider; xcoord=xcoord+xdivider)  
{
```

```
  for (ycoord=ydivider; ycoord<=height-ydivider; ycoord=ycoord+ydivider)  
  {
```

//DRAWS CIRCLE

```
  pushMatrix();  
  translate(xcoord, ycoord);  
  stroke(255);  
  ellipse(0,0, 10, 10);  
  popMatrix();
```

```
  }
```

```
}
```

//Step04- Function

//VARIABLES

```
float xcoord;  
float ycoord;  
float division;  
float xdivider;  
float ydivider;
```

//SETUP

void setup()

```
{  
  size(600,600);  
  division=6;  
  // println( width/division);  
  // println( height/division);  
  xdivider=width/division;  
  ydivider=height/division;  
  smooth();  
  noFill();  
  strokeWeight(1.5);  
}
```

//DRAWING

void draw()

```
{  
  background (27,27,77);  
  //xcoord=xdivider;  
  //ycoord=ydivider;  
  
  for (xcoord=xdivider; xcoord<=width-xdivider; xcoord=xcoord+xdivider)  
  {  
    for (ycoord=ydivider; ycoord<=height-ydivider; ycoord=ycoord+ydivider)  
    {  
      circlefunct(xcoord,ycoord,10);  
    }  
  }  
}
```

//Circle Function

// x-coordinate, y-coordinate, diameter

void circlefunct(float x,float y,float diam) {

```
  pushMatrix();  
  translate(x, y);  
  stroke(255);  
  ellipse(0,0,diam,diam);  
  popMatrix();
```

}

//Step05- mouse input

//VARIABLES

```
float xcoord;  
float ycoord;  
float division;  
float xdivider;  
float ydivider;
```

//SETUP

```
void setup()  
{  
  size(600,600);  
  division=12;  
  // println( width/division);  
  // println( height/division);  
  xdivider=width/division;  
  ydivider=height/division;  
  smooth();  
  noFill();  
  strokeWeight(1.5);  
}
```

//DRAWING

void draw()

```
{  
  background (27,27,77);  
  //xcoord=xdivider;  
  //ycoord=ydivider;  
  
  for (xcoord=xdivider; xcoord<=width-xdivider; xcoord=xcoord+xdivider)  
  {  
    for (ycoord=ydivider; ycoord<=height-ydivider; ycoord=ycoord+ydivider)  
    {  
      circlefunc(xcoord,ycoord);  
    }  
  }  
}
```

//Circle Function

// x-coordinate, y-coordinate, diameter

void circlefunc(float x,float y) {

```
  pushMatrix();  
  translate(x, y);  
  stroke(255);  
  ellipse(0,0,mouseX/2,mouseX/2);  
  popMatrix();  
}
```

//Step06- mouse input - Using distance function

//Circle Function

// x-coordinate, y-coordinate, diameter

```
void circlefunct(float x,float y) {
```

```
    float maxdist=100;
```

```
    pushMatrix();
```

```
    translate(x, y);
```

```
    stroke(255);
```

```
    float mousedist=dist(x,y,mouseX,mouseY);
```

```
    float diameter=(mousedist/maxdist)*10;
```

```
    ellipse(0,0,diameter,diameter);
```

```
    popMatrix();
```

```
}
```