SDES3107
DESIGN AND COMPUTERS 4

INTRODUCTION TO PROCESSING

WEEK 5
MONDAY
D103

WEEK 5

TOPICS COVERED:

ARRAYS Storing numbered sequences of data

ANIMATION Making an animation with a sequential series of images

PIXELS Reading pixels from the sketch window

TYPOGRAPHY Moving the text and responding to user input

ARRAYS (p.301-315)

Arrays are storage containers that contain multiple items, instead of just one (like variables).

You can think of a one-dimentional array like a list, such as:

	Friends
[0]	Jim
[1]	Sam
[2]	Geofrey
[3]	Laura
[4]	Oliver
[5]	Ernest

This list can be created using

```
String[] friends = { “Jim”, “Sam”, “Geofrey”, “Laura”, “Oliver”, “Ernest”};
```

or

```
String[] friends;
friends = new String[6];
friends[0] = “Jim”;
friends[1] = “Sam”;
friends[2] = “Geofrey”;
friends[3] = “Laura”;
friends[4] = “Oliver”;
friends[5] = “Ernest”;
```

You can get data from the list by indicating what item in the list you want:

```
println(friends[2]);
```

Two-dimensional arrays are like tables, such as:

	[0]	[1]
	Name	**Favourite Icecream**
[0]	Jim	Strawberry
[1]	Sam	Chocolate
[2]	Geofrey	Vanilla
[3]	Laura	Rocky Road
[4]	Oliver	Nougat
[5]	Ernest	Cherry

This list can be created using

```
String[][] friends = {
{“Jim”,”Strawberry”},
{“Sam”,”Chololate”},
{“Geofrey”,”Vanilla”},
{“Laura”,”Rocky Road”},
{“Oliver”,”Nougat”},
{“Ernest”,”Cherry”}};
```

And we can access values using

```
friends[1][0] // “Sam”
friends[3][1] // “Rocky Road”
```

We can use a for loop and `.length` to display the value of each table element:

```
for (int i = 0; i < friends.
length; i++) {
  println(friends[i][0] + “ likes
“ + friends[i][1]);
}
```

MAKING ANIMATIONS (p.316-317)

Using an array, if structure, and void draw(), it is possible to create animations in Processing.

To do this, we use the code on the right.

```
int numFrames = 12; // The number of animation
frames
int frame = 0; // The frame to display
PImage[] images = new PImage[numFrames]; //
Image array
void setup() {
size(100, 100);
frameRate(30); // Maximum 30 frames per second
images[0] = loadImage(“ani-000.gif”);
images[1] = loadImage(“ani-001.gif”);
images[2] = loadImage(“ani-002.gif”);
images[3] = loadImage(“ani-003.gif”);
images[4] = loadImage(“ani-004.gif”);
images[5] = loadImage(“ani-005.gif”);
images[6] = loadImage(“ani-006.gif”);
images[7] = loadImage(“ani-007.gif”);
images[8] = loadImage(“ani-008.gif”);
images[9] = loadImage(“ani-009.gif”);
images[10] = loadImage(“ani-010.gif”);
images[11] = loadImage(“ani-011.gif”);
}
void draw() {
frame++;
if (frame == numFrames) {
frame = 0;
}
image(images[frame], 0, 0);
}
```

EXERCISE

34.1. Load a sequence of related images into an array and use them to create a linear animation.

PIXELS (p.321 - 325)

In Processing, it is possible to read the image displayed in the Sketch window and manipulate it. The `get()` function can read the color of any pixel in the display window. It can also grab the whole display window or a section of it.

`get()`	`get(x,y)`	`get(x,y,width,height)`
↑	↑	↑
whole window	1 pixel	an area of the screen

```
smooth();
strokeWeight(8);
line(0, 0, width, height);
line(0, height, width, 0);
noStroke();
ellipse(18, 50, 16, 16);
PImage cross = get(); // Get the entire window
image(cross, 42, 30, 40, 40); // Resize
```

EXERCISE

35.2. Load an image and use mouseX and mouseY to read the value of the pixel beneath the cursor. Use this value to change some aspect of the image.

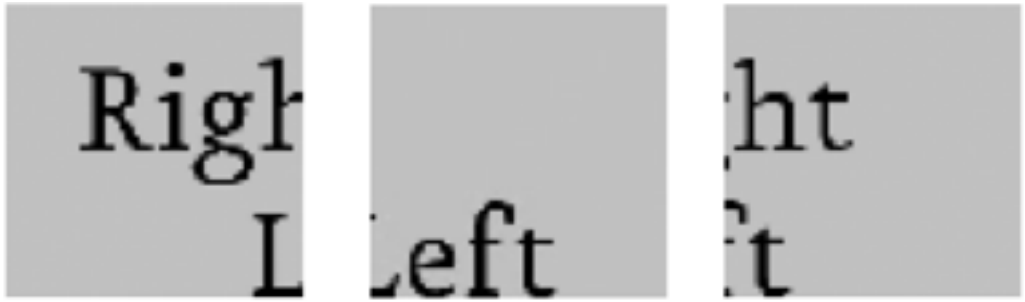
(p.325)

TYPOGRAPHY – MOTION

(p. 327 – 332)

Now that we are redrawing the window and moving and animating drawn elements, we can do more interesting things with topography, such as animating letters and words.

To run these examples, you will need to use the “Create Font” tool to create your own font. Change the name of the parameter to `loadFont()` to the name of the font that you created.




```
PFont font;
float x1 = 0;
float x2 = 100;

void setup() {
  size(100, 100);
  font = loadFont(“Eureka-48.vlw”);
  textFont(font);
  fill(0);
}

void draw() {
  background(204);
  text(“Right”, x1, 50);
  text(“Left”, x2, 100);
  x1 += 1.0;
  if (x1 > 100) { x1 = -150; }
  x2 -= 0.8;
  if (x2 < -150) { x2 = 100; }
}
```

TYPOGRAPHY – RESPONSE

(p. 333 – 336)

We can also start making our text respond to the user's actions, using pre-defined variables such as `mouseX`, `mouseY`, `mousePressed`, and `keyPressed`.


```
PFont f;

void setup() {
  size(100, 100);
  f = createFont("Arial", 24);
  textFont(f);
  textAlign(CENTER);
  fill(0);
}

void draw() {
  background(204);
  text("avoid", width-mouseX,
height-mouseY);
}
```

EXERCISE

37.1. Change the visual attributes of a word as the cursor moves across the display window.

(p.336)

HOMEWORK

EXERCISES: 33.1, 34.1, 35.2, 36.1, 37.1

READINGS: DATA 4, IMAGE 2, IMAGE 3, TYPOGRAPHY 2, TYPOGRAPHY 3, SYNTHESIS 2*, INTERVIEW 3*

Next week's topics: **MOTION 1, MOTION 2.1**