# 07 Case Study

## New Concepts in this Example

**Constructor chaining** – occurs when one constructor (usually a simpler one) calls another constructor in the same class. Primarily used for conveniance, e.g.

class Die {

protected int numFaces;

public Die(int numFaces){

this.numFaces=numFaces;

}

public Die() {

this(6);

}

}

Die d6 = new Die();

Die d10 = new Die(10);

**Matrix Stacks** – can simplify graphics involving complex mathematics, especially rotations. Enables you to draw objects at (0,0) and then translate/rotate them onto the sketch. For example

// Draw a square normally at (100,100)

rect(100,100,75,75);

// Draw a square using a translation matrix at (200,100)

pushMatrix();

translate(200,100);

rect(0,0,75,75);

popMatrix();

// Draw a square using trans/rotation matrix at (300,100)

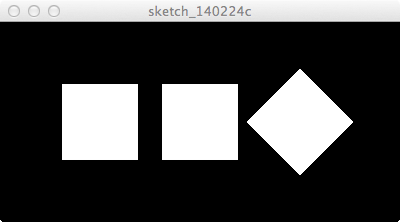
pushMatrix();

translate(300,100);

rotate(PI/4);

rect(0,0,75,75);

popMatrix();



**File Reading** – processing has a built-in class called Table that makes reading and writing Comma-Separated Value (CSV) files much easier than in C#. An example of a CSV file:

fruit,frequency

apple,63

banana,24

orange,88

fejoia,63

fig,10

The documentation for the Table class:

<http://www.processing.org/reference/Table.html>

**Data Source Encapsulation** – this example uses the DataSource class to hide all the details of the data source (i.e. the fact that the datasource is a CSV file). This is a good idea, as it makes it easier to change to a new data source (e.g. a database or an HTML table from a web page) without changing the rest of your program.