Week 7 – Webinar B

In this webinar session, we will look at both of the collections introduced in this week’s lecture – arrays and ArrayLists.

# Learning Objectives

* **Employ** refactoring techniques to improve the readability of our programs
* **Develop** programs with multiple instances of the same class via the Array and ArrayList classes

# Preparation

Download and extract **Game.zip**. Open **Game.pde** and run the application. You should see a player-controlled plane and two yellow birds which fly across the screen (Figure 1). If either of the birds crash into the plane, the game stops. We will inspect the code together and go through each of the classes.

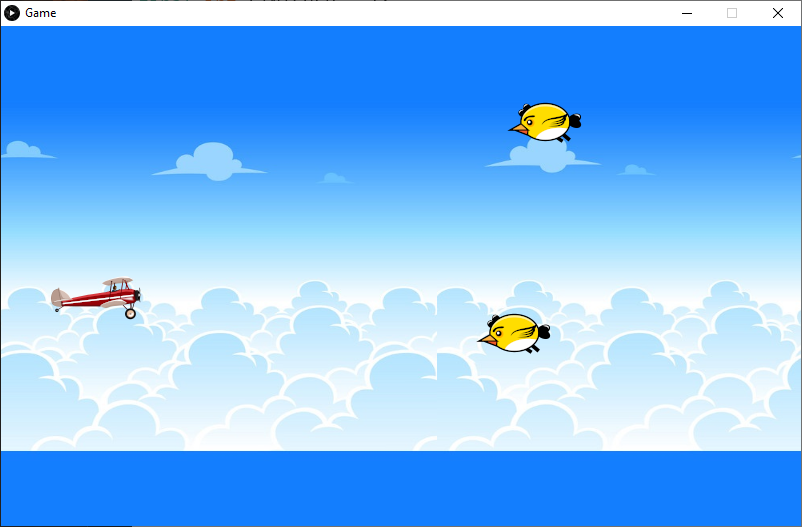


Figure - Game Output

# Step 1 – Refactoring

After going through each tab, we will refactor some of the code to improve the readability. Processing has some built-in refactoring tools (e.g. renaming methods and variables within a program) that we will look at.

# Step 2 – Implementing an Array of PImages

You will notice that the Bird class has four **PImage** variables for each image that the render method cycles through. Instead of having four variables to represent four individual images, we can have an array of PImages and then cycle through the array’s elements when animating the bird.

# Step 3 – Collection of Objects

We are now going to implement a collection of bird objects. Although we could declare and instantiate many objects of the same class and individually call their methods, using a *collection* in conjunction with *loops* allows us to streamline the process.

The code below creates an ArrayList capable of ‘storing’ Bird objects. An ArrayList is empty initially, and we can then add or remove objects as the program is running.

ArrayList<Bird> birdList = new ArrayList<Bird>();

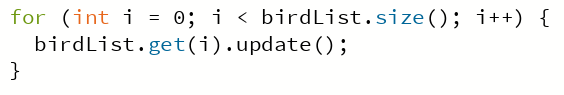
birdList is an object of the ArrayList class – a built-in class for managing collections of *things*. We can add something to an ArrayList by calling the **add** method, and we can remove objects by calling the **remove** method.

We could manually add the objects we already have to this list or use some sort of rule to determine when to add a new bird to the list.

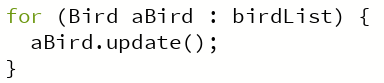
# Step 4 – for-each loop

We have seen quite a lot of for loops in this unit so far, mainly to perform the same task multiple times, and to iterate through a collection such as an array of ArrayList. There is an alternative syntax to iterating through a collection known as the **for-each loop** (also referred to as the **enhanced for loop**). A comparison of a ‘traditional’ for loop and a ‘for-each’ loop is provided below.

**Traditional for loop –** i starting at 0 and iterating as long as i is less than the size of the ArrayList



**for-each loop** – note there is no integer counter variable. This type of loop will start with the first bird at index 0, and refer to that bird object as ‘aBird’, the 2nd time this loop runs, the ‘aBird’ variable is pointing to the 2nd bird at index 1, and so on.



In the next webinar, we will see some of the limitations of the for-each loop.