# Lab Week 6 Handling Multiple Classes and Objects

### Previous Topics

* Using PImage
* Designing and implementing a Class
* Writing methods
* Writing constructor

## Learning Objectives

* Designing & Implementing Multiple Classes for a more complex problem
  + Constructors & methods
* Creating instance objects and using them for a simple game
* Detecting a crash using pixel colours over an area

Today’s exercises will be useful for your Game assignment. It contains many elements which can be reused (with a little tweaking) when developing your Programming Games assignment. Ask for feedback on your code and explanations for anything developed during the lab that you are unsure about

## Resources

* Lecture Notes – Multiple Classes
* Webinar 6A : we started a similar game as a group
* Tutorial <https://processing.org/tutorials/objects/>
* Processing website – reference
* Background image on moodle

Save your code after each exercise in a new directory

**Ex1.** Open a new Processing sketch, paste in the code below and *save it* as **MouseDragDefender**. On moodle you will find a background **JPEG** file. Download the file to your Defenderz directory. The code below will give you a horizontal scrolling background image which wraps around.

PImage background;

**int** bgX=**0**; //global variable background location

**void** **setup**(){

size(**800**,**400**);

background = loadImage("spaceBackground.jpg");

background.resize(width,height); //set image to be same size as the canvas

}

**void** **draw** ()

{

//scrolling background image

image(background, bgX, **0**); //draw image to fill the canvas

//draw image again off the right of the canvas

image(background, bgX+background.width, **0**);

bgX = bgX- **4**;

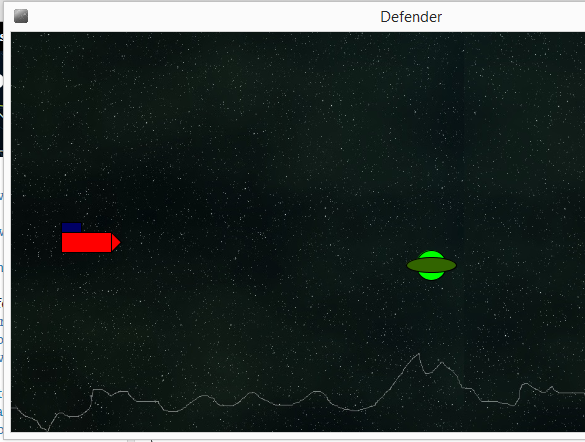
**if**(bgX == -background.width) //if first image completely off the canvas

{

bgX=**0**; //reset back to initial value background

}

}

**Defenderz Game**

1st prototype has

* alien (moving left, and perhaps up and down).
  + color ALIEN1 = color(0,255,0);
  + color ALIEN2 = color(50,100,0);
* A defender
  + Moved up and down by player
  + Can crash into alien
  + Detect by alien colour (area just in front of the defender)
  + color ALIEN1 = color(0,255,0);
  + color ALIEN2 = color(50,100,0);

What Classes do we require?

What constructors & methods for each?

Add an Alien class

//draw an alien

fill(ALIEN1);

ellipse(x,y,30,30);

fill(ALIEN2);

ellipse(x,y,50,15);

Add an instance of an alien and test it

Add the Defender class (do not use the same name as the sketch!)

//draw a defender:x,y is top left corner, shape is from y..y+30 tall

fill(0,0,200);

rect(x,y,20,10); //draw top box

fill(255,0,0); //draw rocket

rect(x,y+10,50,20);

triangle(x+50,y+10,x+50,y+30,x+60,y+20);

Add a **mouseDrag** method to Defender class that allows it to be dragged using the mouse :

void mouseDrag() //dragging with mouse

{

if (mousePressed) //built-in Boolean variable, has mouse been pressed

{

if (dist(mouseX,mouseY, this.x,this.y) <50 ) //mouse close to top corner?

{

this.x=mouseX; //move rocket to mouse position

this.y=mouseY;

}

}

}

Add a defender instance to your program and test it. Add the [mouseDragged()](https://processing.org/reference/mouseDragged_.html) (follow link to reference page) event to your program to allow user interaction and test it.

void mouseDragged()

Add a **crash** Boolean method to Defender. We could use a distance measure as we have previously (asserting that a crash has taken place if the two objects are within a certain distance, see lecture this week) but here we’ll use a different technique instead, looking for colour.

A crash has taken place if we detect the alien’s colours just in front of the defender ship. To do this we need to check a range of pixels (e.g. each pixel along the yellow line in image) and test whether ALIEN1 colour OR ALIEN2 colour is present – how?

FOR LOOP – repeat a block of actions a set number of times

Rocket is 30 pixels high

[get function](https://processing.org/reference/get_.html) returns a pixel colour at a specific position, see powerpoint slides this week

Step 1. Write a method called **drawCrashLine()** and add to Defender class

Using the **point** command (not line) draw the yellow line of points in front of the defender using a **for loop**, drawing each point in turn top to bottom. The defender is 60 pixels wide, so defender’s x + 65 would be just in front.

Step 2. Convert your method to a **crash** Boolean function method to use the **get** command to establish whether any pixel along that line contains either of the two green (ALIEN) colours.

color pixelColour = get(x,y);

will get the colour of the pixel at x,y,

if (pixelColour == ALIEN1)

In the Draw event , ensure you have drawn the alien on the screen before checking for a crash!!

**Exercise 2. Defenderz game**

Stage 1. complete the 1st prototype of the game

* Add 3 or more aliens (2 more), see lecture powerpoint for use of a collection.
* Use a **gameMode** variable to stop the game if a crash occurs (we covered this idea with the racing Motorbikes lab exercise – see “Objects, Using a Class” (on moodle)
* Allow the alien to move up or down in some sequence (as well as left)
* If the alien goes off the screen a new alien should appear on the right

**Ex2.B (further marks)** – 2nd Prototype game (very useful for Programming 2 assignment)

Allow the defender to fire a bullet, by pressing the space bar

Add an **isShot** method to the alien class

Open up the background image, make a copy and draw on some objects (e.g. asteroids) in a specific (R,G,B) colour. You can then add this colour to your Defender crash detection method to make the game harder.

Add multiple aliens

Add scoring or an explosion.

**Extension** : replace the Defender with a suitable animated sequence of image files – search opengameart, or draw your own in Paint3D.