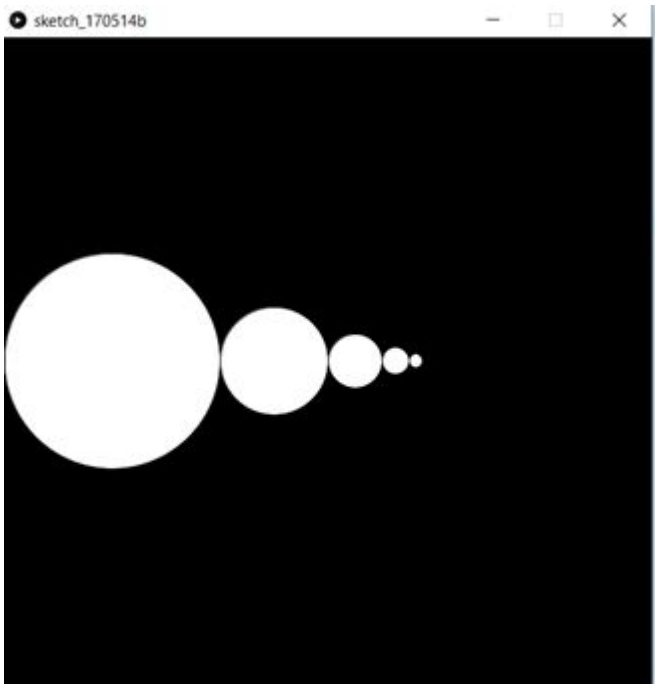
Ellipses

Begin with the Grade 0 section.

When you complete a grade ask a teacher to check your progress before moving on to the next grade. Make sure a teacher sees your progress before the end of class.

Grade 0 – Start Here

1. Use ellipse to make the following design on a black background:



2. Make the circles follow the mouse (large circle positioned at mouseX, mouseY. The other circles must have offsets - maybe positions like “mouseX +100”, “mouseX + 150”, depending on the sizes of your circles). Ensure that there are no trails. REMEMBER: you will need to use void setup() and void draw() here.

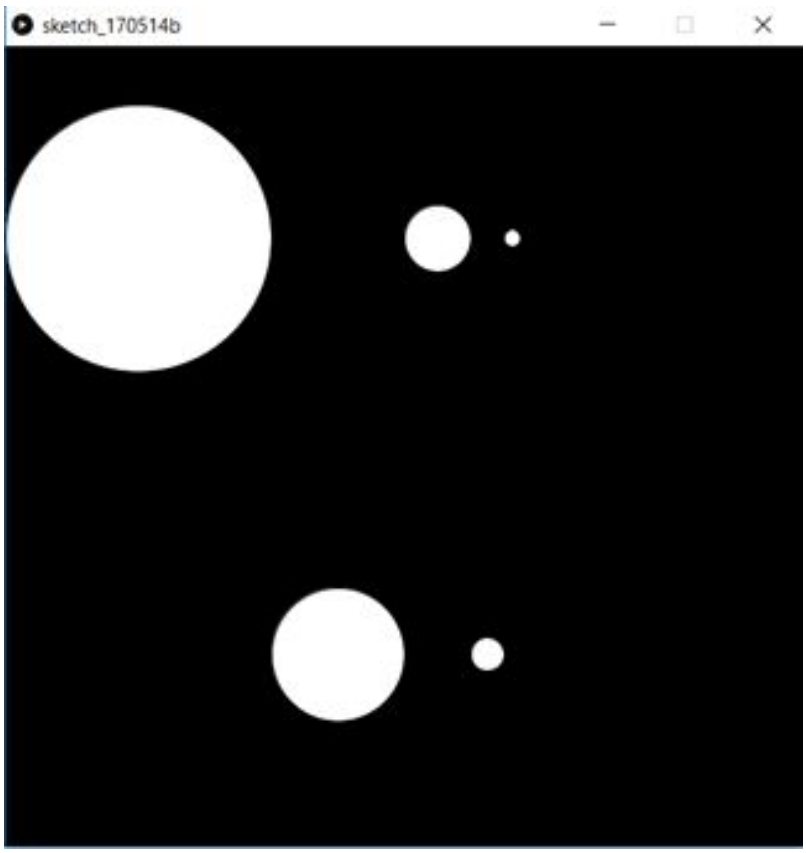
Grade 1

1. Use variables (call them xPos and yPos) to control the position of the ellipses.
    a. HINT: You have to replace every “mouseX” with “xPos”, and the same for “mouseY” and “yPos”. Use “Find and Replace” to change mouseX to xPos and mouseY to yPos, and to save yourself a lot of typing.

2. Add keyboard controls to move the ellipses left, right, up and down.
    a. HINT: be very careful with brackets! This should get you started:

```
if(keyPressed) {
   if(key == 'a') {
     //make xPos smaller
   }
 }
```

3. Change the controls so that every second circle moves in the opposite vertical direction. Circles 1, 3 and 5 will move up when circles 2 and 4 move down, and vice versa, like this:



4. Add a key that gives each ellipse a random colour.
    a. HINT: You'll need to make red, green and blue variables to use in the fill function for each ellipse. These variables then need to be given random values. This code gives red a random value between 0 and 255.

```
red = (int)random(0, 255);
```

Grade 2

Task: Change the code from above so that Instead of using the same xPos and yPos variables and adding some offset for each ellipse, use an array to store a number of position values.

```
float[] xPos = { 200, 500, 650, 725, 762.5 };
```

Hint: Use FOR loops to iterate through the array. The structure of our FOR loop will look like this:

```
for (int i = 0; i < xPos.size(); i++) {

  ellipse(xPos[i], yPos, radius * 2, radius * 2);
}
```

Notice how FOR loops start at zero, not at one. For example, if we have 5 elements in an array the FOR loop will go from 0 to 4.

Grade 3

Task: Create a class "Circle", which will allow us to create objects of type “Circle”. Create a new object of type “Circle” and draw it to the screen. Refer to the relevant lesson to refresh yourself on how classes are declared and used.

```
class Circle {

  float xPos, yPos, speedX, speedY;
  float radius = 10;
  float red = random(0, 255);
  float green = random(0, 255);
  float blue = random(0, 255);
}
```

Note the capital letters here.

- A class name should *always* be in UpperCamelCase.
- A variable or object name should *always* be in lowerCamelCase.
- In this case, the class will be Circle, and if we only have one object of type Circle, we will call it circle.

To create an object of type *Circle* there are two steps after copying the code above:

1. *Declare* the new object.
2. *Initialise* the new object.

```
Circle circle; // Step 1 - declare

void setup() {
  size(800, 800);
  circle = new Circle(); // Step 2 - initialise
}

void draw() {

  background(0);

  circle.xPos = 200; // Dot notation to access object’s variables
  circle.yPos = 300;
```

```
  circle.radius = 10;

  ellipse(circle.xPos, circle.yPos, circle.radius * 2, circle.radius * 2);
}
```

The code above only draws a circle. This is not really what we're after! Storing variables like this can be useful, but classes really become important when we add methods.

Methods are bits of code stored inside the class, which do something related to the class. In this case, it would be useful to have a method drawCircle(), and have that code draw a circle of whatever size and position we have stored in the circle object.

```
class Circle {

  float xPos, yPos, speedX, speedY;
  float radius = 10;
  float red = random(0, 255);
  float green = random(0, 255);
  float blue = random(0, 255);

  void drawCircle() {
     fill(red, green, blue);
     ellipse(xPos, yPos, radius * 2, radius * 2);
   }
}
```

Now instead of having all those fills and ellipses cluttering up our main code, they're stored in the Circle class. All we have to do is set the position and colour of the circle and then call "circle.drawCircle()".
Once you have this working, declare a second object of type Circle. Call it circle2. Rename the first one circle1.

Constructor

The final part we need for our objects is a constructor. When we write “circle = new Circle()” the new keyword triggers the default constructor. This creates a new object of type circle, and sets the variables to their default values (in our case, xPos, yPos and radius will all be zero).
We can set the colour of the circles to be random instead and the position and speed can be loaded into the constructor for each circle. Our new Circle class will look like this:

```
class Circle {

  float x, y, radius, speedX, speedY, red, green, blue;

  Circle(float _xPos, float _yPos, float _radius, float
  _speedX, float _speedY) {

    xPos = _xPos;
    yPos = _yPos;
    radius = _radius;
    speedX = _speedX;
    speedY = _speedY;
    red = random(0, 255);
    green = random(0, 255);
    blue = random(0, 255);
  }

  void drawCircle() {
    fill(red, green, blue);
    ellipse(xPos, yPos, radius * 2, radius * 2);
  }
}
```

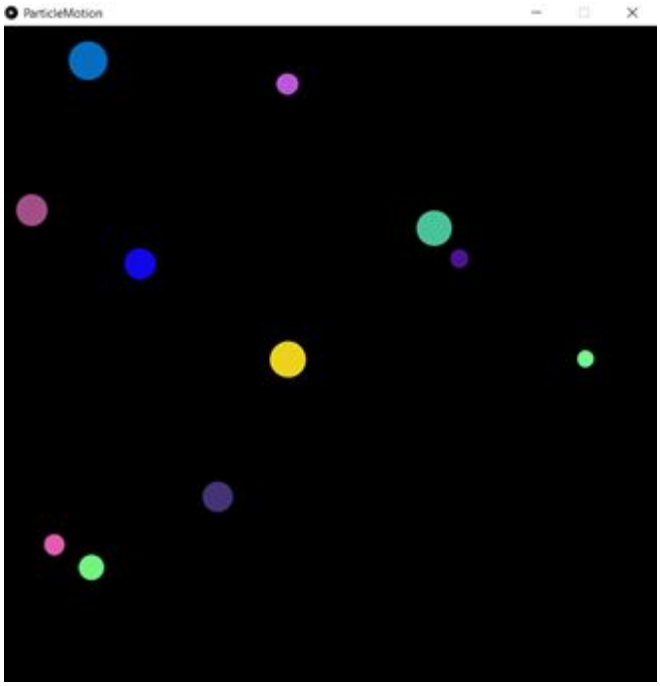
Note that we’ve added the Circle() function, without the word void in front. This overrides the default constructor.

Grade 4

Task: Add an ArrayList to store circles. Provide multiple ways to add new circle (eg click, automatically every second).

```
ArrayList<Circle> circleList;

void setup() {
  size(800, 800);
  circleList = new ArrayList<Circle>();

  for (int i = 0; i < 10; i++) { // loop repeats 10 times
    circleList.add(new Circle(random(0, width),
    random(0, height), random(10, 25), random(-3, 4),
    random(-3, 4)));
  }
}

void draw() {

  background(0);

  for (int i = 0; i < circleList.size(); i++) {
 // loop repeats for each item in list

    circleList.get(i).drawCircle();
  }
}
```

The code above declares and initializes the arraylist and stores 10 circles in the list before drawing them to the screen.



The key part of this to understand is that to add a new circle to circleList we use the line “circleList.add(new Circle());”, and to retrieve it we use “circleList.get(i)”, where i is a number. If i is 0 we are retrieving item 0 from the list, and so on. We often use a FOR loop (as in the example) to retrieve every item in the list, one at a time.

Grade 5+

Add more functions to the relevant classes that will:

- Move each circle with a random speed.
- Allow each circle to bounce off the sides.
- Add a new circle in a random location when a key (or mouse) has been pressed.
- Alternatively, implement a timer that will add a new circle every second or two.
- The direction and colour of two circles will change when a collision takes place (every circle must be iterated through to see if a collision has occurred).

Implement a Game class will contain the circles, as well as the methods we will need (eg. for detecting collisions between circles). This allows us to keep the main setup and draw blocks quite neat (they should only contain user input code, and perhaps some background and text functions.