Arrays

Arrays Can Hold and Use Multiple Numbers with the Same Name, and, they let You Write a for-loop to Use Them!

```java
int X[] = {100, 200, 300, 400, 500, 600, 700};
int Y[] = {100, 100, 100, 200, 200, 200, 300};

void setup()
{
    size(800, 800);
    background(200, 200, 255);
    stroke(0, 0, 0);
    fill(255, 255, 0);
}

void draw()
{
    for(int i = 0; i < X.length; i = i + 1)
    {
        ellipse(X[i], Y[i], 100, 50);
    }
}
```

Processing will tell you the length of an array.
A Way of Declaring Arrays That Lets You Fill Them Later

```java
int NumPoints = 5000;
int W = 800;
int H = 800;
int [ ] X;
int [ ] Y;
int [ ] R;
int [ ] G;
int [ ] B;
```

Total number of points

The arrays that will hold the points and the colors. They have only been declared. They don’t yet have any memory given to them.

Allocating and Filling Arrays

```java
void setup() {
    X = new int [NumPoints];
    Y = new int [NumPoints];
    R = new int [NumPoints];
    G = new int [NumPoints];
    B = new int [NumPoints];

    size( 800, 800 );
    for( int i = 0; i < NumPoints; i = i + 1 ) {
        X[ i ] = int( random( 0, W ) );
        Y[ i ] = int( random( 0, H ) );
        R[ i ] = int( random( 0, 255 ) );
        G[ i ] = int( random( 0, 255 ) );
        B[ i ] = int( random( 0, 255 ) );
    }
}
```

Allocate the arrays to hold the random points and the random colors. At this point, memory has been given to them, but they don’t have any values assigned.

Each point is assigned a random location.

Allocate the arrays to hold the random points and the random colors. At this point, memory has been given to them, but they don’t have any values assigned.

Each point is assigned a random color. There is no significance to each point’s specific color – it is to look cool!

Note: Array indices start at 0 and end at the number of elements minus one. So, an array dimensioned [10] indexes from 0 to 9.
```cpp
void draw()
{
    background( 200, 200, 255 );
    stroke( 0, 0, 0 );
    for( int i = 0; i < NumPoints; i = i + 1 )
    {
        fill( R[i], G[i], B[i] );
        ellipse( X[i], Y[i], 8, 8 );
    }
}
```

Draw each point with its color.
A Cool Pattern will be made even Cooler

For our next trick, during each frame we are going move each point halfway towards one of three target triangle vertices. Which target to use is chosen at random.

```
int NumPoints = 5000;
int TARGET_SIZE = 40;
int W = 800;
int H = 800;

int [ ] X;
int [ ] Y;
int [ ] R;
int [ ] G;
int [ ] B;

int [ ] XC = { 50, W/2, W-50 };  // The arrays that will hold the points and the colors. They have only been declared. They don’t yet have any memory given to them.
int [ ] YC = { H-50, 50, H-50 };  // The arrays that hold the three center points. Because of the way this was coded, these arrays do have memory given to them.
```
void setup() {
    X = new int [NumPoints];
    Y = new int [NumPoints];
    R = new int [NumPoints];
    G = new int [NumPoints];
    B = new int [NumPoints];

    size(W, H);
    for (int i = 0; i < NumPoints; i = i + 1) {
        X[i] = int(random(0, W));
        Y[i] = int(random(0, H));
        R[i] = int(random(0, 255));
        G[i] = int(random(0, 255));
        B[i] = int(random(0, 255));
    }
    frameRate(2);
}

Each point is assigned a random location.

Each point is assigned a random color. There is no significance to each point’s specific color – it is to look cool!

Perform the animation at 2 frames per second so that we can actually see it. Otherwise, it will be too fast.

void draw() {
    background(200, 200, 255);
    fill(255, 0, 0);
    ellipse(XC[0], YC[0], TARGET_SIZE, TARGET_SIZE);
    fill(0, 255, 0);
    ellipse(XC[1], YC[1], TARGET_SIZE, TARGET_SIZE);
    fill(0, 0, 255);
    ellipse(XC[2], YC[2], TARGET_SIZE, TARGET_SIZE);

    stroke(0, 0, 0);
    for (int i = 0; i < NumPoints; i = i + 1) {
        fill(R[i], G[i], B[i]);
        ellipse(X[i], Y[i], 8, 8);
    }

    for (int i = 0; i < NumPoints; i = i + 1) {
        int randTarget = int(random(0.000, 2.999));
        X[i] = (X[i] + XC[randTarget]) / 2;
        Y[i] = (Y[i] + YC[randTarget]) / 2;
    }
}

Draw the three targets.

Draw each point with its color.

Re-compute each point’s position by randomly picking one of the targets (0, 1, or 2) and moving halfway towards it.
Mathematicians call shapes like this “attractors”