

# GRAFIKA KOMPUTER

## Motion

### I. TUJUAN

- Mahasiswa mampu menggunakan dan memodifikasi program untuk mendapatkan gambar bergerak /animasi

### II. DASAR TEORI DAN PERCOBAAN

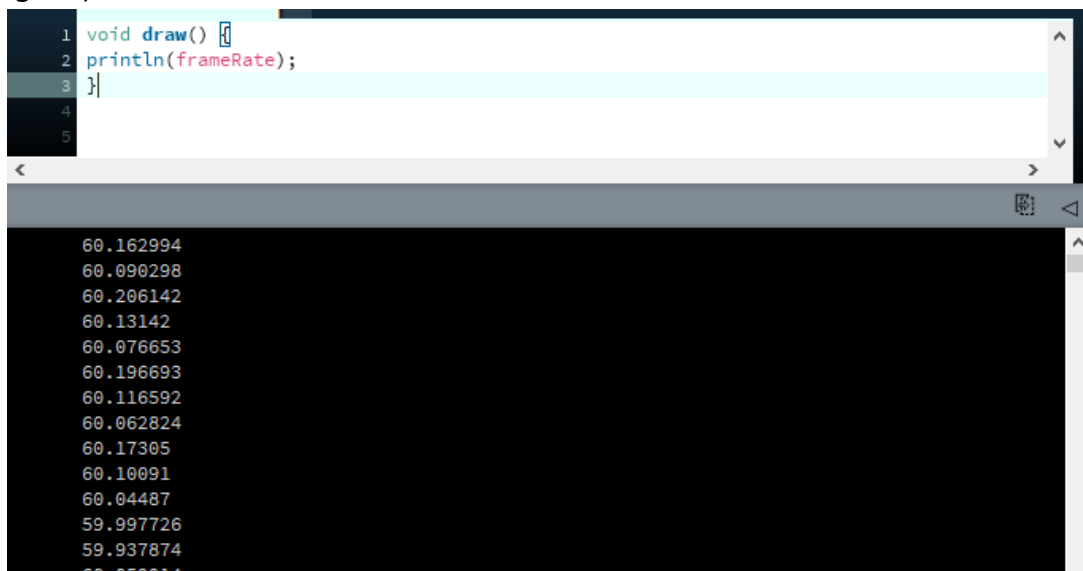
Animasi di layar dibuat dengan menggambar suatu gambar, lalu menggambar lagi gambar yang sedikit berbeda dan begitu seterusnya. Ilusi gerak diciptakan oleh pergerakan penglihatan. Ketika satu set gambar yang serupa disajikan pada tingkat yang cukup cepat, Maka otak menerjemahkan gambar-gambar ini ke dalam suatu gerakan.

#### a. Frame

Untuk membuat suatu gerakan yang halus, Processing mencoba menjalankan kode dalam fungsi draw() pada 60 frame setiap detik. Frame adalah satu kali running program melalui fungsi draw() dan frame rate adalah berapa banyak frame ditarik setiap detik. Karena itu, sebuah program yang menggambar 60 frame setiap detik berarti program menjalankan seluruh kode di dalam fungsi draw() 60 kali setiap detik.

#### Percobaan 1 : See the Frame Rate

Untuk menghitung dan mendapatkan nilai frame rate, jalankan program ini dan perhatikan nilainya yang tercetak ke Konsol (variabel frameRate melacak kecepatan program):



```
1 void draw() {
2   println(frameRate);
3 }
4
5
```

```
60.162994
60.090298
60.206142
60.13142
60.076653
60.196693
60.116592
60.062824
60.17305
60.10091
60.04487
59.997726
59.937874
60.050014
```

#### Percobaan 2: Tetapkan Frame Rate

Fungsi frameRate() mengubah kecepatan program yang sedang berjalan. Untuk melihat hasilnya, cobalah aktifkan syntax frameRate() dengan nilai yang berbeda dalam contoh ini:

```
1 void setup() {
2   frameRate(30); // Thirty frames each second
3   //frameRate(12); // Twelve frames each second
4   //frameRate(2); // Two frames each second
5   //frameRate(0.5); // One frame every two seconds
6 }
7 void draw() {
8   println(frameRate);
9 }
10
```

57.345013  
54.76729  
52.635147  
50.765114  
48.99205  
47.521255  
46.11033  
44.92749  
43.788734  
42.83832  
41.972153  
41.179058  
40.454887  
39.71087

### b. Kecepatan dan Arah gerak

Untuk membuat contoh gerakan fluida, dapat menggunakan tipe data float. Dimana Jenis variabel ini menyimpan angka dengan satuan desimal, yang memberikan rentang resolusi lebih besar untuk pergerakan. Misalnya, saat menggunakan tipe data int, paling lambat kita dapat memindahkan setiap frame adalah satu pixel pada satu waktu (1, 2, 3, 4, ...), tetapi dengan float, kita bisa bergerak selambat yang kita inginkan (1.01, 1.01, 1.02, 1.03, ...).

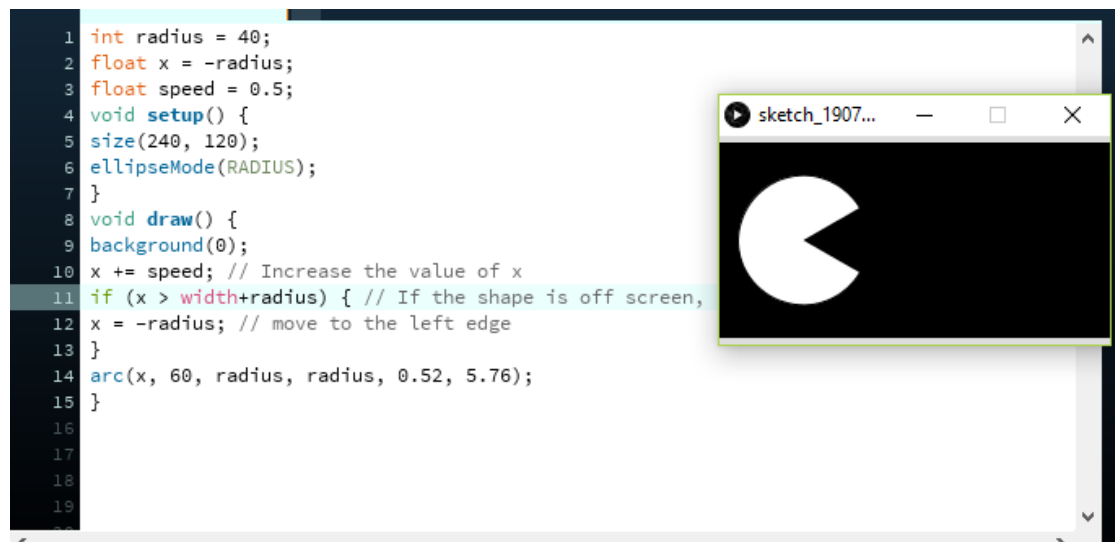
### Percobaan 3: Menggerakkan suatu bentuk gambar

Contoh berikut memindahkan gambar dari kiri ke kanan dengan cara memperbarui variabel x secara terus menerus.

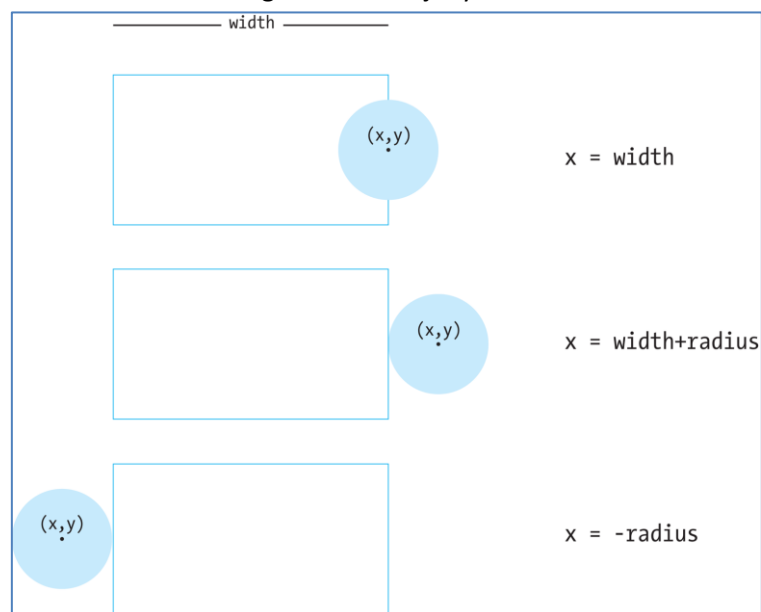
```
4 void setup() {
5   size(240, 120);
6   ellipseMode(RADIUS);
7 }
8 void draw() {
9   background(0);
10  x += speed; // Increase the value of x
11  arc(x, 60, radius, radius, 0.52, 5.76);
12 }
13
14
15
16
17
```

### Percobaan 4 : Membatasi bingkai gerak

Ada banyak alternatif untuk menerapkan hal ini, yang dapat dilakukan salah satunya adalah Pertama, menambahkan kode untuk menunjukkan cara memindahkan bentuk kembali ke tepi kiri jendela Layar setelah itu menghilang dari kanan. Dalam hal ini, gambarkan layar sebagai silinder pipih, dengan bentuk pergerakan bagian luar untuk kembali ke titik awalnya:

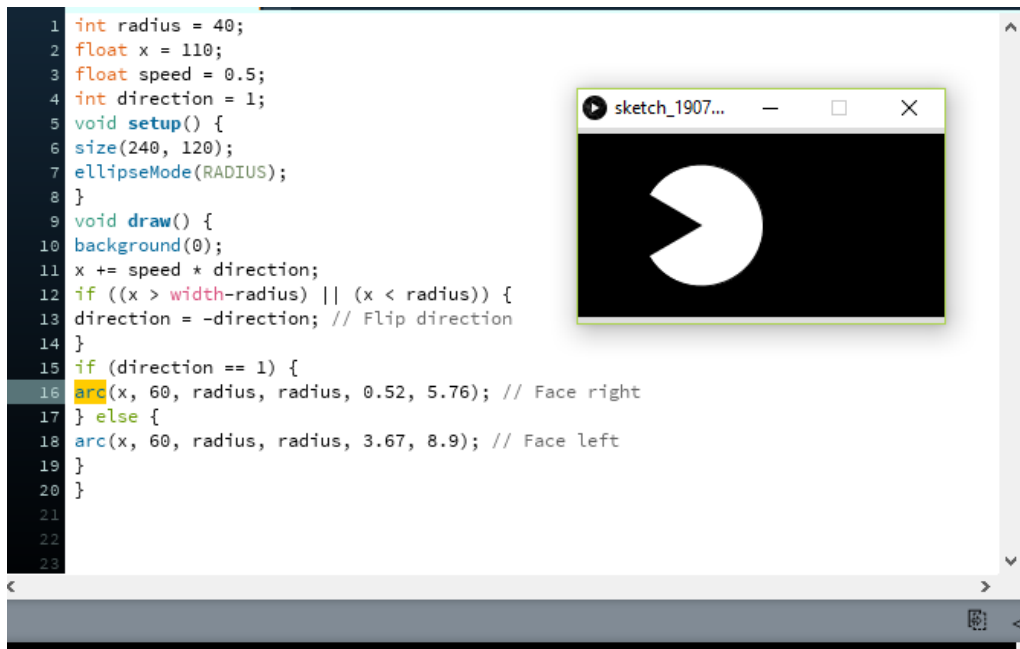


Pada setiap satu kali program dijalankan melalui `draw()`, kode menguji untuk melihat apakah nilai `x` telah meningkat melebihi lebar layar (ditambah jari-jari dari shape/bentuk). Jika sudah, kembali mengatur nilai `x` ke nilai negatif, sehingga saat terus meningkat, maka shape akan memasuki layar dari kiri. Lihat Gambar berikut untuk diagram cara kerjanya.



### Percobaan 5: Gerak memantul dari border

Dalam contoh ini, akan mengembangkan lagi dari contoh 4 dimana untuk menambahkan adanya perubahan arah saat menyentuh tepi atau border bingkai. Implementasinya dengan cara menambahkan variable baru untuk menyimpan status arah dari gambar/shape yang ada. Nilai arah 1 untuk menggerakkan gambar ke kanan dan nilai -1 untuk bentuk menggerakkan gambar ke kiri:

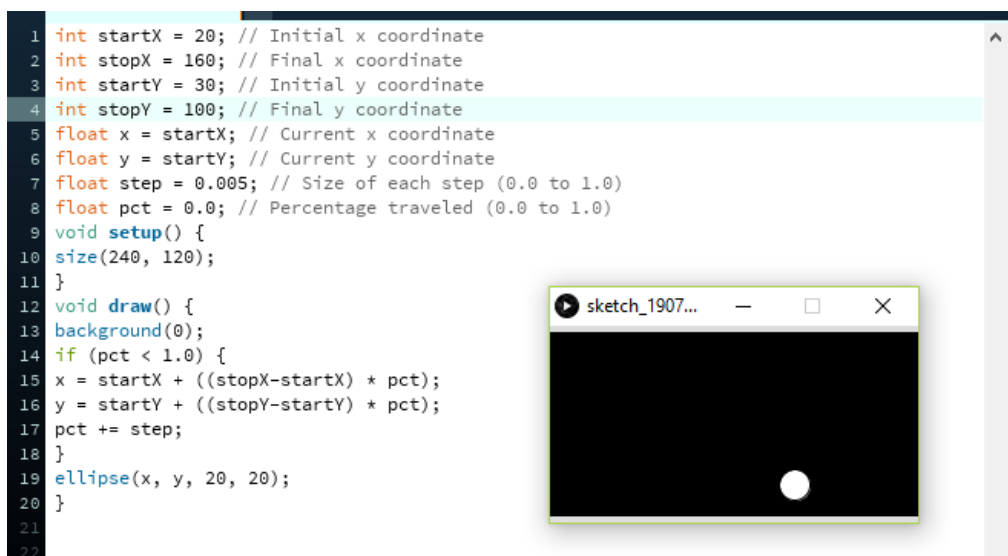


Ketika gambar mencapai tepi, script code membalik arah gambar dengan cara mengubah tanda variabel arah. Misalnya, jika variabel direction positif maka ketika gambar mencapai tepi, script code akan membalik nilai variable direction ke negatif.

Jika ingin menggerakkan suatu bentuk dari satu titik ke titik lain dalam suatu layar. Maka kita dapat mengatur posisi awal dan posisi berhenti, lalu hitung peralihan (tween) posisi koordinat di setiap frame.

### Percobaan 6 : Hitung Tween

Untuk menjadikan contoh ini modular (lebih mudah dikembangkan dimodifikasi), maka telah dibuat grup variable dibagian atas. Saat menjalankan kode ini beberapa kali cobalah untuk merubah-ubah nilai variabelnya dan lihat bagaimana kode ini dapat menggerakkan object gambar dari lokasi manapun dengan beragam kecepatan.

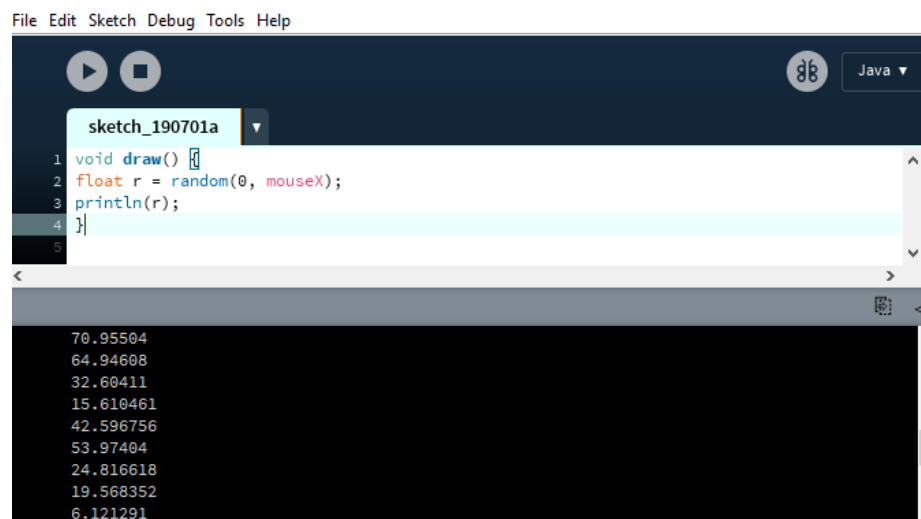


### c. Random

Berbeda dengan gerakan linear yang halus dan umum untuk grafik komputer, gerak dalam dunia nyata biasanya istimewa. misalnya, daun melayang ke tanah, atau semut merayap di medan yang kasar. Kita dapat mensimulasikan kualitas pergerakan yang tidak terduga dengan cara menggunakan angka acak (random). Fungsi `random()` dapat menghasilkan nilai-nilai acak dan kita dapat mengatur rentang nilai yang ingin dimunculkan secara acak tersebut.

### Percobaan 7: Hasilkan nilai random

Contoh nilai-nilai berikut mencetak nilai acak ke Konsol, dengan rentang nilai dibatasi oleh posisi mouse. fungsi `random()` selalu mengembalikan nilai floating-point, jadi Pastikan variabel di sebelah kiri operator (`=`) bernilai float seperti di sini:



```
File Edit Sketch Debug Tools Help

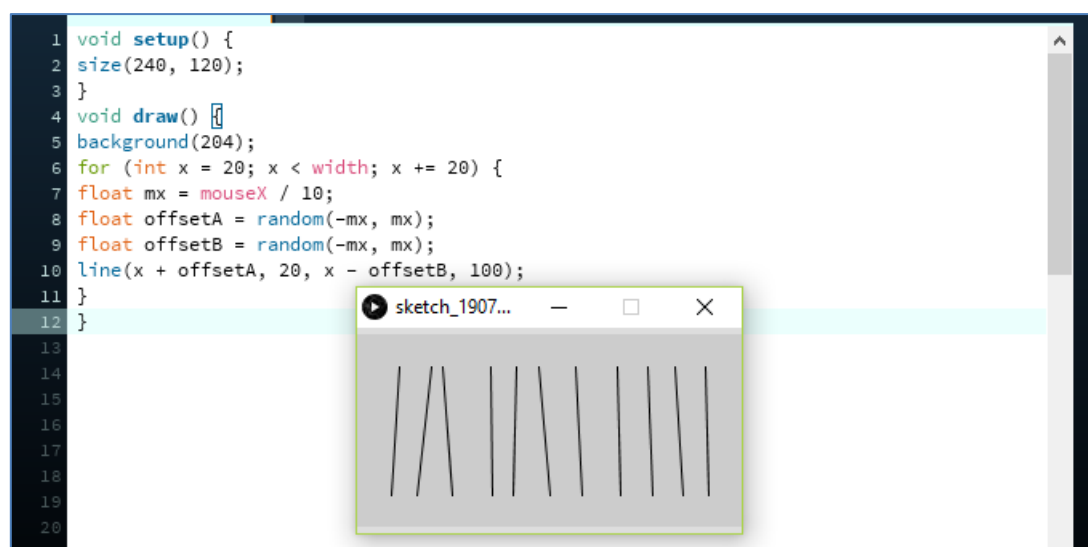
sketch_190701a

1 void draw() {
2   float r = random(0, mouseX);
3   println(r);
4 }

70.95504
64.94608
32.60411
15.610461
42.596756
53.97404
24.816618
19.568352
6.121291
```

### Percobaan 8: Gambar Secara Acak

Contoh berikut menggunakan nilai dari `random()` untuk mengubah posisi gambar line yang aktif di layar. Ketika mouse berada di sebelah kiri layar, maka perubahan atau range nilai random akan semakin kecil; saat bergerak ke kanan, nilai-nilai range dari `random()` akan semakin lebar dan membuat seakan pegerakannya semakin ekstrim. Karena fungsi `random()` ada di dalam loop for, nilai random akan selalu baru untuk setiap gambar line:



```
1 void setup() {
2   size(240, 120);
3 }
4 void draw() {
5   background(204);
6   for (int x = 20; x < width; x += 20) {
7     float mx = mouseX / 10;
8     float offsetA = random(-mx, mx);
9     float offsetB = random(-mx, mx);
10    line(x + offsetA, 20, x - offsetB, 100);
11  }
12 }
```

## Percobaan 9: Gerakkan gambar Secara Acak

Saat digunakan untuk menggerakkan suatu gambar/object di layar, nilai acak dapat menghasilkan pergerakan gambar yang lebih natural. Dalam Contoh berikut, posisi lingkaran dimodifikasi secara acak pada setiap kali program dalam draw() dieksekusi. Karena fungsi background() tidak digunakan, maka lokasi sebelumnya dapat tetap terekam dalam layar:



Jika kita membiarkan program diatas running dalam waktu yang lama maka dapat terlihat bahwa gambar lingkaran akan keluar dari bingkai layar. Jika kita ingin menjaga agar posisi gambar lingkaran random tetap dalam bingkai layar maka dapat ditambahkan fungsi constrain() yang dapat menjaga batasan nilai random x dan y masih dalam batas bingkai tampilan window. Dengan mengganti fungsi draw() dalam kode sebelumnya dengan yang berikut, maka akan dipastikan gambar lingkaran akan tetap berada dalam layar:

```
void draw() {
  x += random(-speed, speed);
  y += random(-speed, speed);
  x = constrain(x, 0, width);
  y = constrain(y, 0, height);
  ellipse(x, y, diameter, diameter);
}
```


### d. Timers

Setiap program pada Processing secara otomatis akan menghitung jumlah waktu selama program running. waktu dihitung dalam milidetik (seperseribu dari satu detik), jadi setelah 1 detik, penghitung berada pada 1.000; setelah 5 detik, itu di 5.000; dan setelah 1 menit, nilainya menjadi 60.000. Kita dapat menggunakan variable timer ini untuk memicu animasi muncul pada waktu tertentu. dan Fungsi millis() akan mengembalikan nilai dari timer ini.

### Percobaan 10: Nilai Timer

Kita dapat melihat nilai dari timer ketika kita menjalankan program ini:

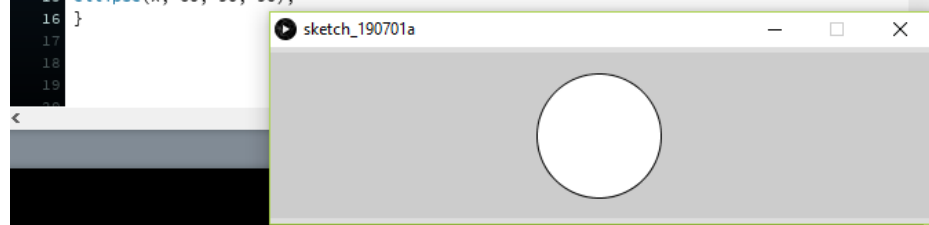
```
1 void draw() {
2   int timer = millis();
3   println(timer);
4 }
```



### Percobaan 11 : Waktu Pemicu Gerak/Animasi

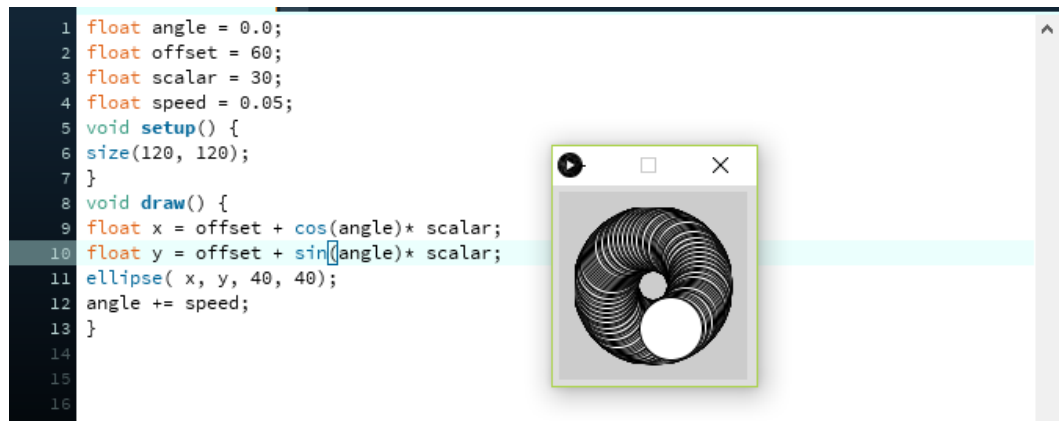
Suatu event saat digabungkan dengan blok if, nilai dari millis () bisa digunakan untuk mengurutkan animasi dan event dalam suatu program. Misalnya, setelah dua detik berlalu, kode di dalam blok if dapat memicu perubahan. Dalam contoh ini, variabel time1 dan time2 dipanggil untuk menentukan kapan harus mengubah variabel nilai x:

```
1 int time1 = 2000;
2 int time2 = 4000;
3 float x = 0;
4 void setup() {
5   size(480, 120);
6 }
7 void draw() {
8   int currentTime = millis();
9   background(204);
10  if (currentTime > time2) {
11    x -= 0.5;
12  } else if (currentTime > time1) {
13    x += 2;
14  }
15  ellipse(x, 60, 90, 90);
16 }
```



#### e. Circular/Sin dan Cos

Ketika sin () dan cos () digunakan bersama-sama, mereka dapat menghasilkan gerakan lingkaran. Nilai cos () memberikan koordinat x, dan nilai sin () memberikan koordinat y. Keduanya dikalikan oleh variabel bernama skalar untuk mengubah jari-jari gerakan dan dijumlahkan dengan nilai offset untuk mengatur pusat gerakan melingkar:



### Percobaan 12: Spirals

Modifikasi kecil dilakukan dari program diatas untuk meningkatkan nilai skalar di setiap frame dapat menghasilkan gerakan spiral, bukan lingkaran:

