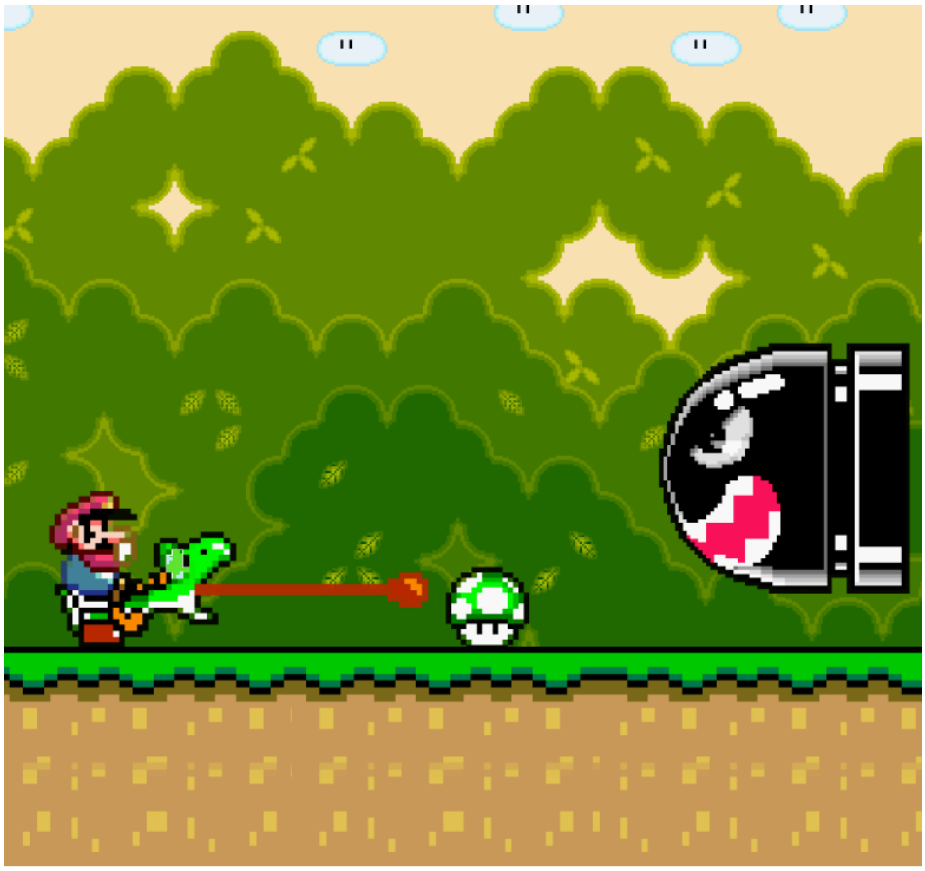

Figure 1: Bok 4

#	Beskriving
12	`fullScreen`
13	`PImage`
14	Tyngdkraft
15	Arrayer 1
16	Arrayer 2

Contents

Förord

Detta är en bok om Processing för ungdomar. Processing är ett programmeringsspråk. Denna bok lär dig det programmeringsspråket.

Om den här boken

Det är fortfarande en lite slarvig bok. Det finns stafvel och la*youten är* in`te allt`**id vack**er. Eftersom den här boken finns på en webbplats kan alla som tycker att den här boken är för slarvig göra den mindre slarvig.

`fullScreen`

`fullScreen` är en funktion som låter dig förstora fönstret för ditt program till samma storlek som datorns skärm.

`fullScreen`: uppgift 1

Kör den här koden. Vad ser du?

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(100, 200, width / 4, height / 4);
}
```

	
`rect (100, 200, 300, 400)`	'Kära dator, rita en rektangel där' (100, 200) 'är koordinaten för det övre vänstra hörnet och där bredden på rektangeln är 300 pixlar och höjden 400 pixlar.'
`width / 4`	'Kära dator, ange här fönstrets bredd, dividerat med fyra'
`height / 4`	'Kära dator, ange här fönstrets höjd, dividerat med fyra'

`fullScreen`: lösning 1

Figure 2: `fullScreen`: lösning 1

`fullScreen`: uppgift 2

Skapa en rektangel med det övre vänstra hörnet i mitten, med bredden 200 pixlar och höjden 100 pixlar.

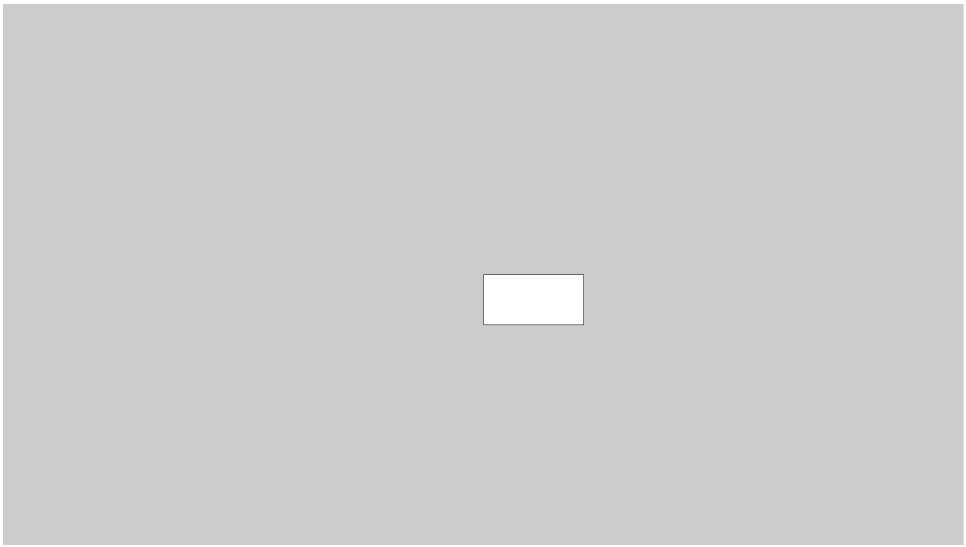

Figure 3: `fullScreen`: uppgift 2

`fullScreen`: lösning 2

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 2, height / 2, 200, 100);
}
```

`fullScreen`: uppgift 3

Rita nu ut rektangeln i mitten på skärmen.

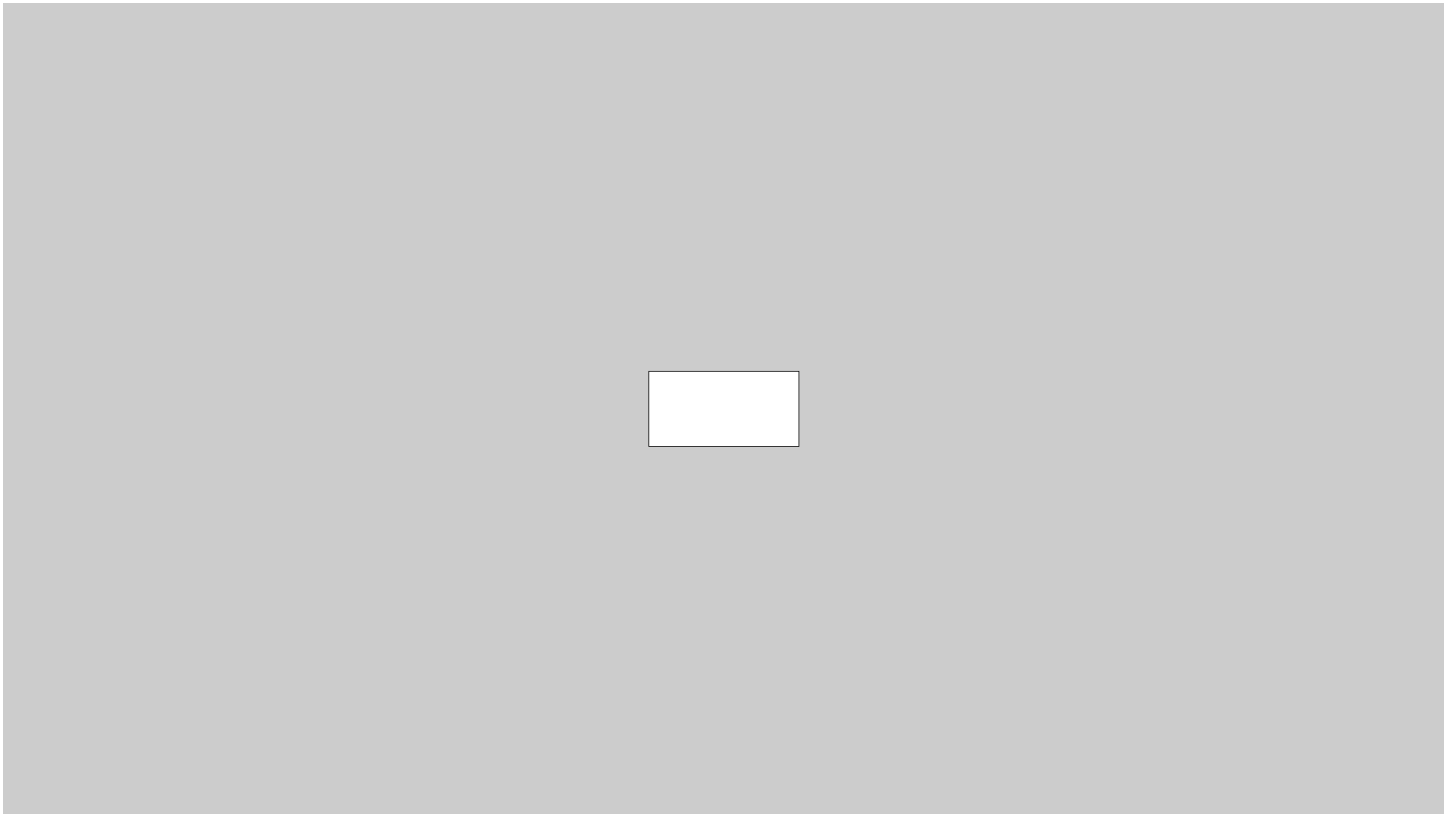

Figure 4: `fullScreen`: uppgift 3

Rektangeln ska flyttas 100 pixlar till vänster och 50 pixlar uppåt

---

`fullScreen`: lösning 3

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 2 - 100, height / 2 - 50, 200, 100);
}
```

`fullScreen`: uppgift 4

Rita ut en rektangel i mitten på skärmen, med en bredd på 300 pixlar och en höjd på 400 pixlar.

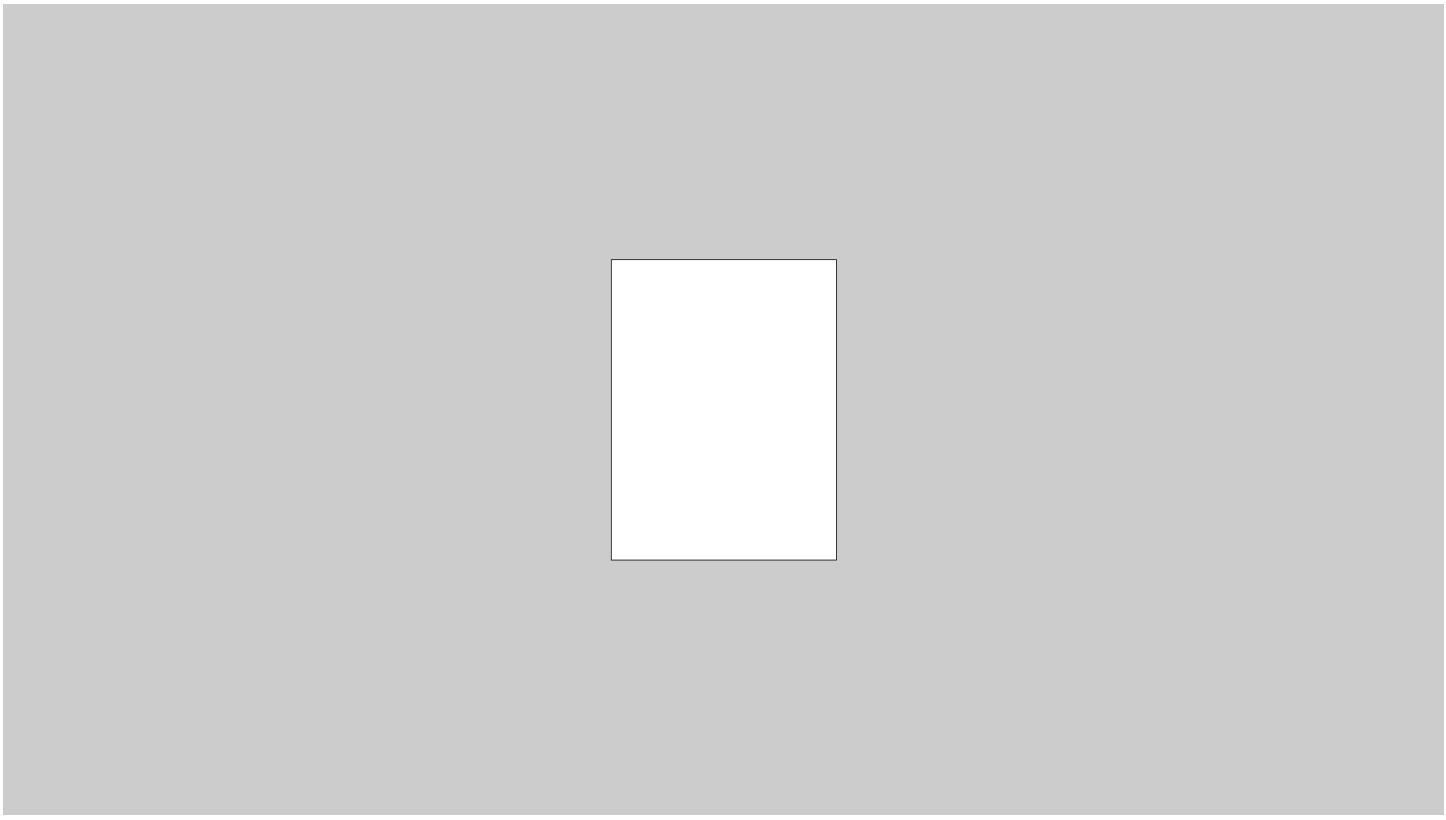

Figure 5: `fullScreen`: uppgift 4

`fullScreen`: lösning 4

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 2 - 150, height / 2 - 200, 300, 400);
}
```

`fullScreen`: uppgift 5

Rita ut en rektangel i mitten på skärmen, med hälften av skärmens bredd, och höjden 500 pixlar.

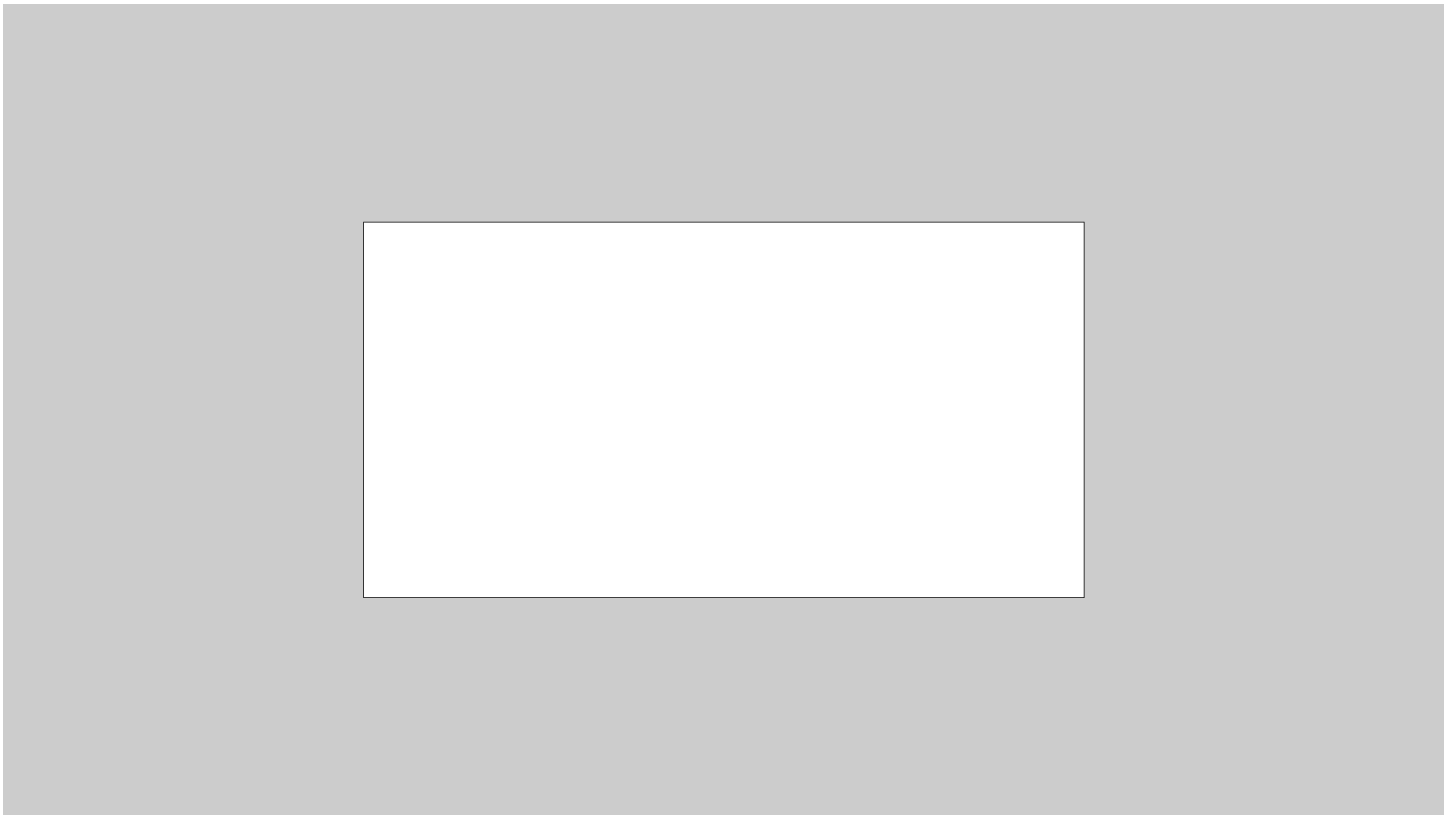

Figure 6: `fullScreen`: uppgift 5

fullScreen: lösning 5

```
void setup()
{
  fullScreen();
}

void draw()
{
  rect(width / 4, height / 2 - 250, width / 2, 500);
}
```

`fullScreen`: slutuppgift

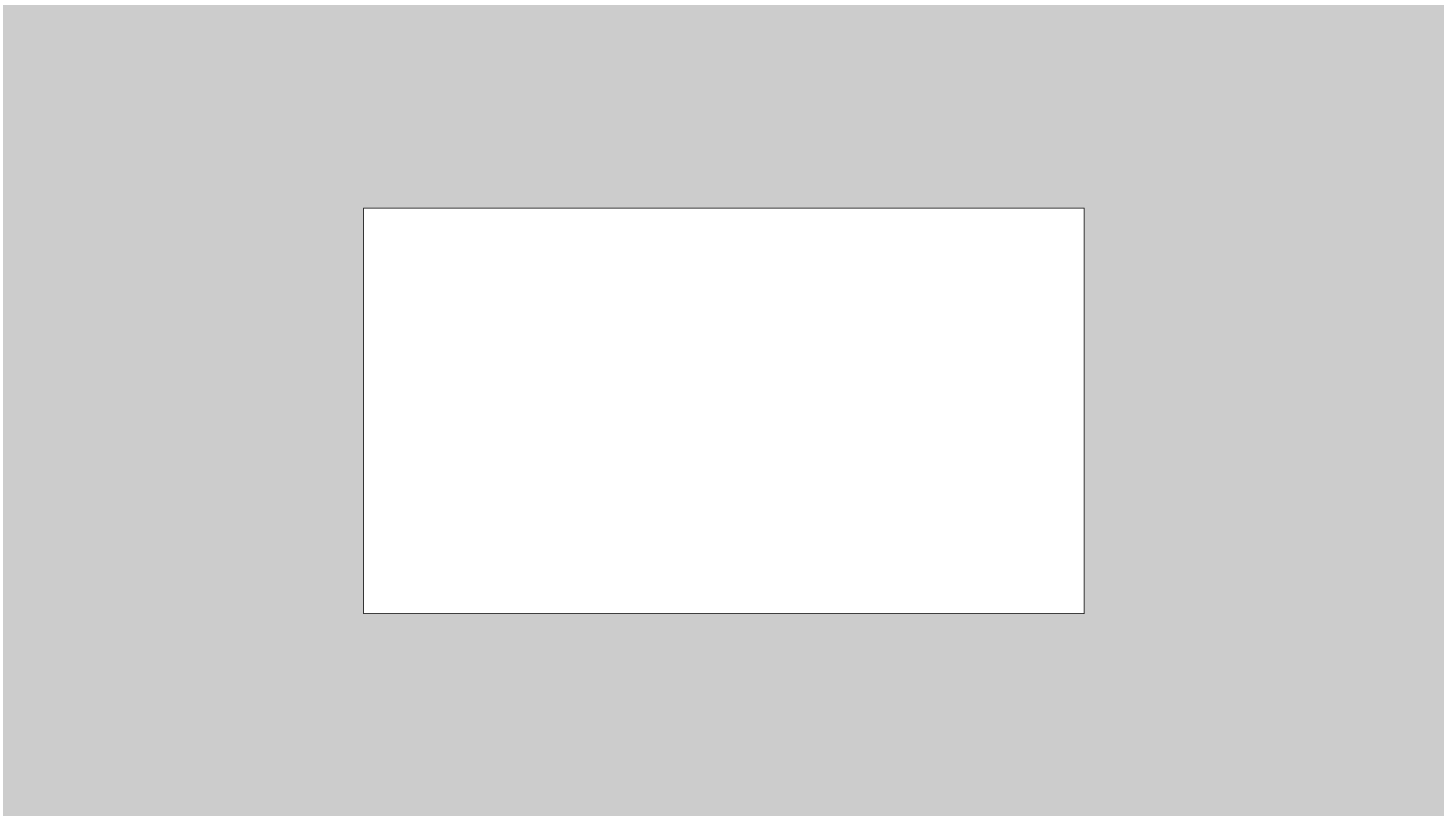

Figure 7: `fullScreen`: slutuppgiftt

Rita ut en rektangel i mitten på skärmen som är hälften så bred och hälften så hög som skärmen.

PImage

Under den här lektionen ska vi jobba med bilder!

Figure 8: EmojiSunglasses.png

`PImage`: Uppgift 1

Spara den här koden. Kör koden. Vad ser du?

```
PImage bild;

void setup()
{
  size(640, 360);
  bild = loadImage("mario.png");
}

void draw()
{
  background(255, 255, 255);
  image(bild, 100, 200);
}
```

PImage: lösning 1

Du får ett fel!

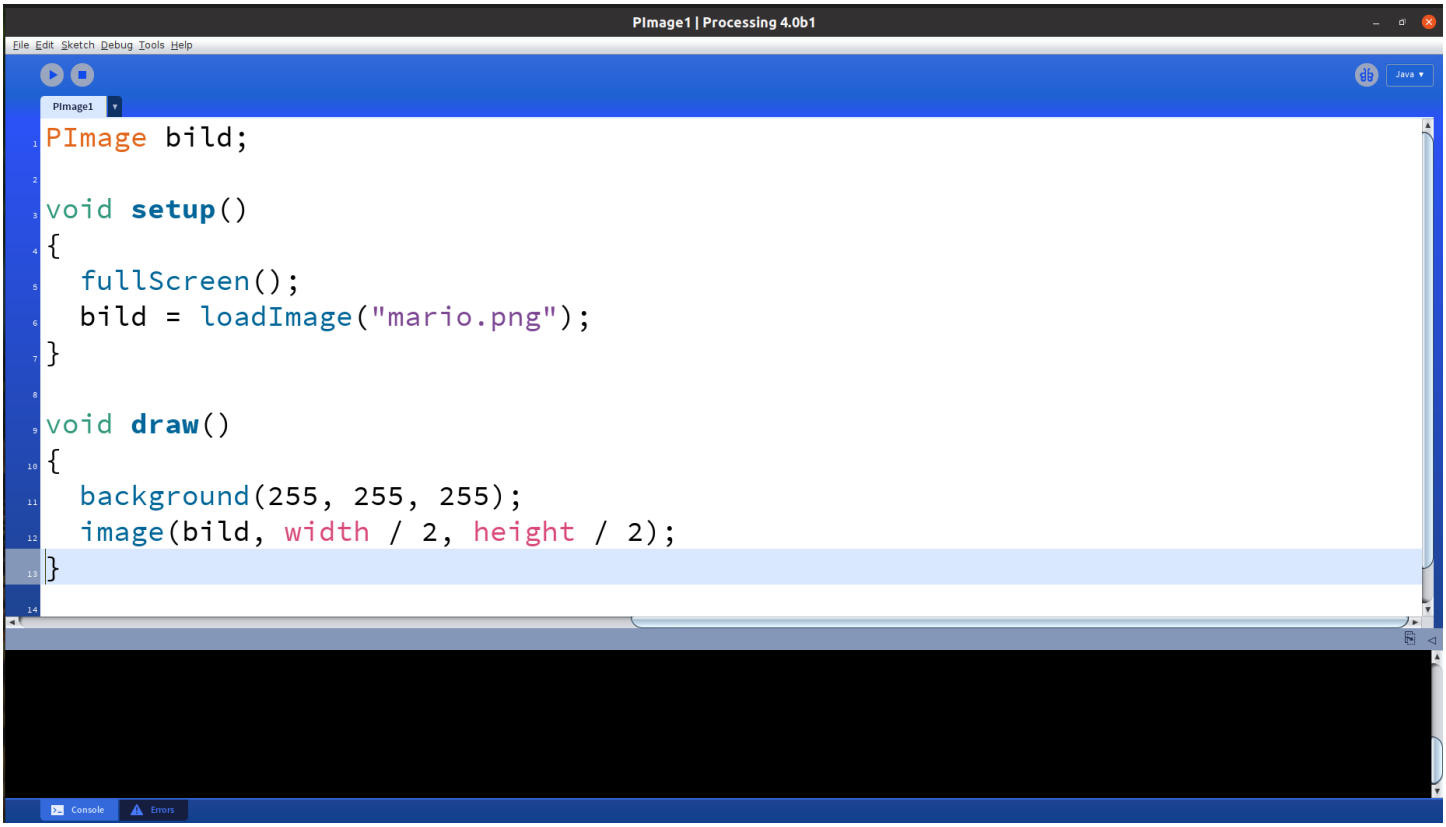

Figure 9: Lösning 1

Datorn säger att den inte kan hitta bilden!

`PImage`: Uppgift 2

Gå till https://raw.githubusercontent.com/richelbilderbeek/processing_foer_ungdomar/master/kapitel/PImage/mario.png och ladda ner den här bilden av Mario.

Figure 10: mario.png

Lägg den här bilden i en undermapp där din kod finns.

Här är en bild som visar var filerna ska vara:

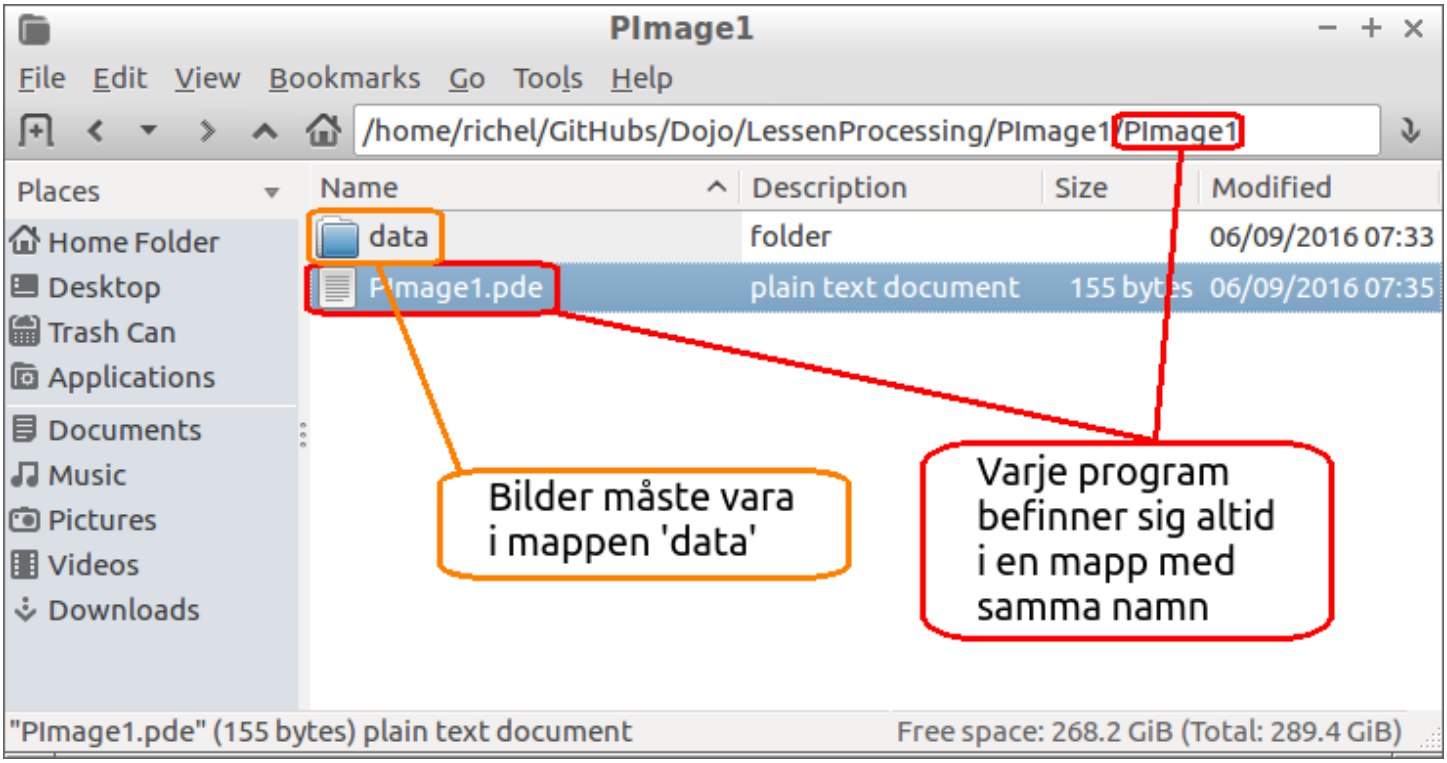

Figure 11: Mappstruktur

- Skissen heter `PImage1.pde`. Därför finns den i mappen `PImage1`. Du hittar detta i Bearbetning under `Sketch -> Show Sketch Map`
- Skissen har en mapp `data`. Den innehåller bilden "mario.png".

`PImage`: slutuppgift

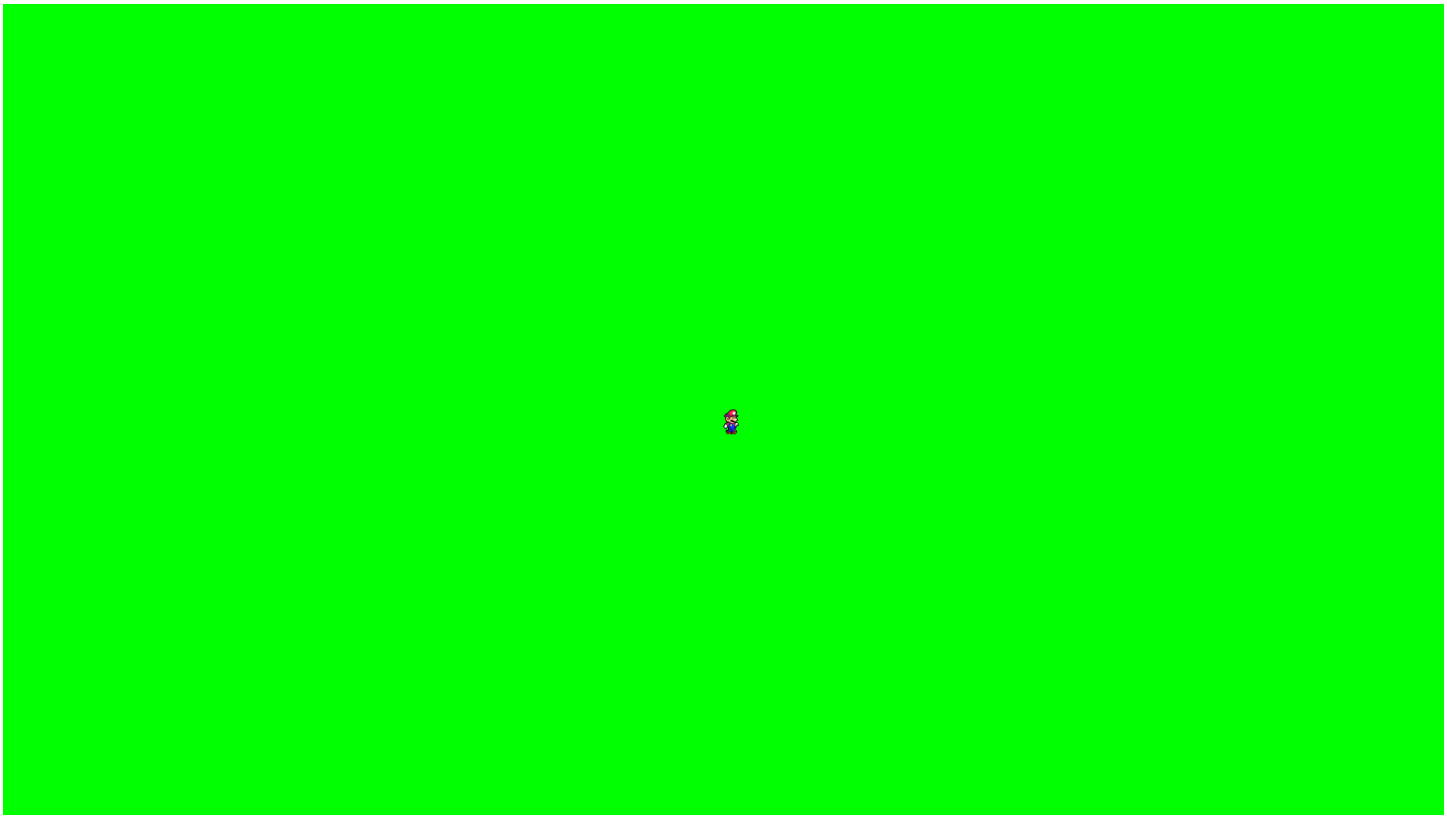

Figure 12: `PImage`: slutuppgift

Gör så att programmet körs i helskärm. Gör bakgrunden grön och lägg bilden i mitten.

Tyngdkraft

Under den här lektionen ska vi programmera gravitationen.

Vi kommer att använda två variabler och två “if”-satser.

Tyngdkraft: Uppgift 1

Vad gör den här koden?

```
float x = 150;
float y = 100;
float hastighet_at_hoger = 1;
float hastighet_nedat = 1;

void setup()
{
  size(300, 200);
}

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + hastighet_at_hoger;
  y = y + hastighet_nedat;
  if (x > 275)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (x < 25)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (y > 175)
  {
    hastighet_nedat = -hastighet_nedat;
  }
}
```

Tyngdkraft: Lösning 1

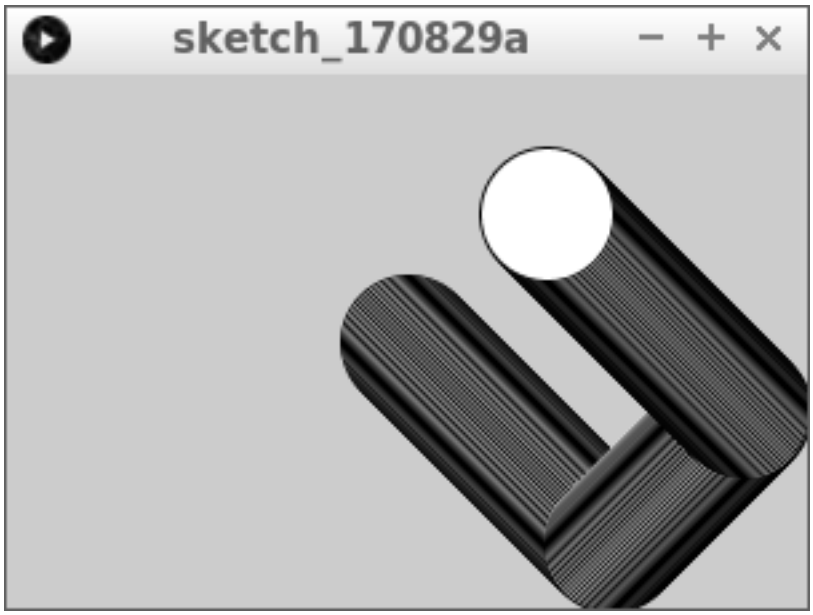

Figure 13: Tyngdkraft: Lösning 1

Tyngdkraft: Uppgift 2

Se till att programmet visas i helskärm.

Figure 14: Tyngdkraft: uppgift 2

Tyngdkraft: Lösning 2

```
float x = 150;
float y = 100;
float hastighet_at_hoger = 1;
float hastighet_nedat = 1;

void setup()
{
  fullScreen();
}

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + hastighet_at_hoger;
  y = y + hastighet_nedat;
  if (x > 275)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (x < 25)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (y > 175)
  {
    hastighet_nedat = -hastighet_nedat;
  }
}
```

Tyngdkraft: Uppgift 3

Figure 15: Tyngdkraft: uppgift 3

Se till att bollen studsar bra mot botten.

Tips: ersätt `175` med `höjd - 25`

Tyngdkraft: Lösning 3

```
float x = 150;
float y = 100;
float hastighet_at_hoger = 1;
float hastighet_nedat = 1;

void setup()
{
  fullScreen();
}

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + hastighet_at_hoger;
  y = y + hastighet_nedat;
  if (x > 275)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (x < 25)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (y > height - 25)
  {
    hastighet_nedat = -hastighet_nedat;
  }
}
```

Tyngdkraft: Uppgift 4

Se till att bollen studsar bra mot höger sida.

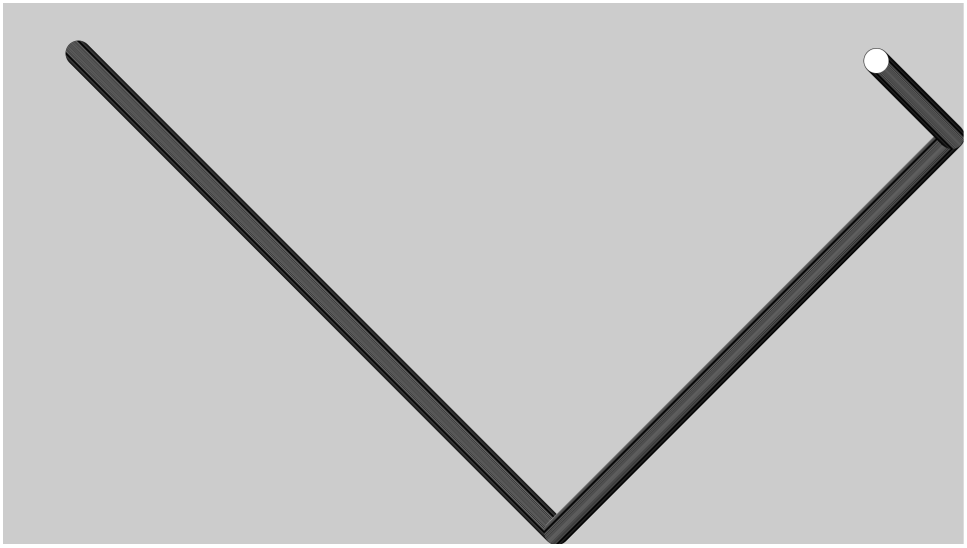

Figure 16: Tyngdkraft: uppgift 4

Tyngdkraft: Lösning 4

```
float x = 150;
float y = 100;
float hastighet_at_hoger = 1;
float hastighet_nedat = 1;

void setup()
{
  fullScreen();
}

void draw()
{
  ellipse(x, y, 50, 50);
  x = x + hastighet_at_hoger;
  y = y + hastighet_nedat;
  if (x > width - 25)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (x < 25)
  {
    hastighet_at_hoger = -hastighet_at_hoger;
  }
  if (y > height - 25)
  {
    hastighet_nedat = -hastighet_nedat;
  }
}
```

Tyngdkraft: Slutuppgift

Lägg till följande inuti “draw”-funktionen längst ned:

```
hastighet_nedat += 0.1;
```

Gör bollen dubbelt så stor.

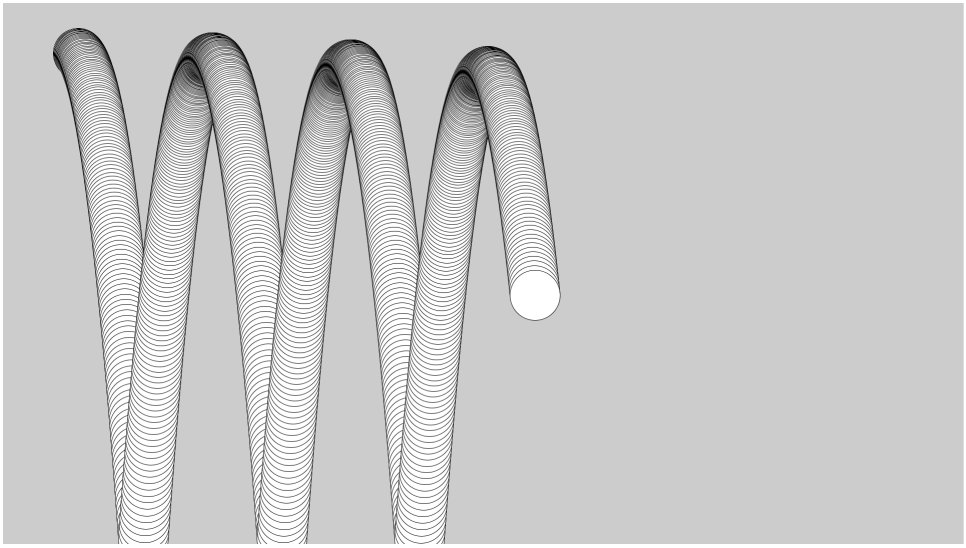

Figure 17: Tyngdkraft: slutuppgift

Om ditt program ser ut som bilden här nedanför, får du också ett klistermärke.

Ibland händer det oväntade saker i programmering!

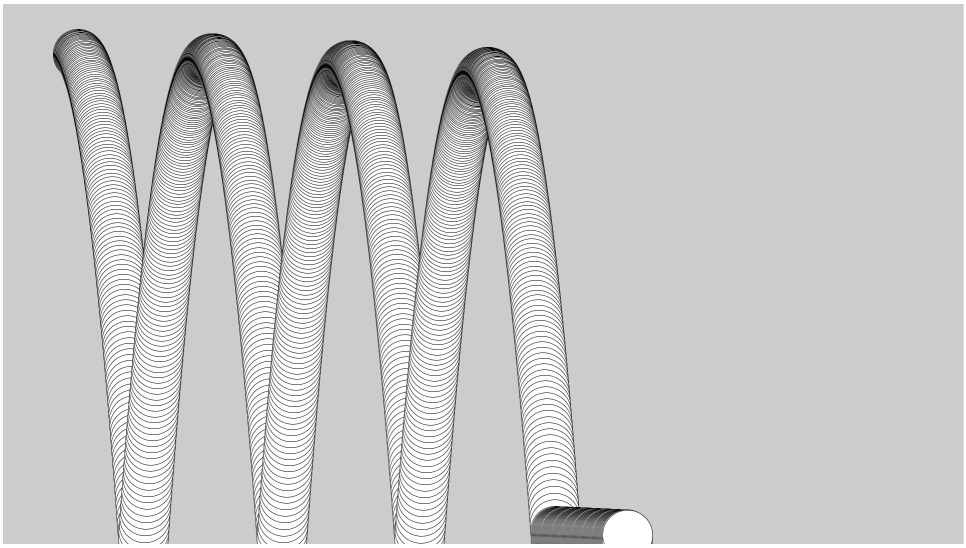

Figure 18: Tyngdkraft: slutuppgift också OK

Arrayer 1

Med arrayer kan du få datorn att komma ihåg många värden: koordinaterna för kulor, meteoriter, fiender.

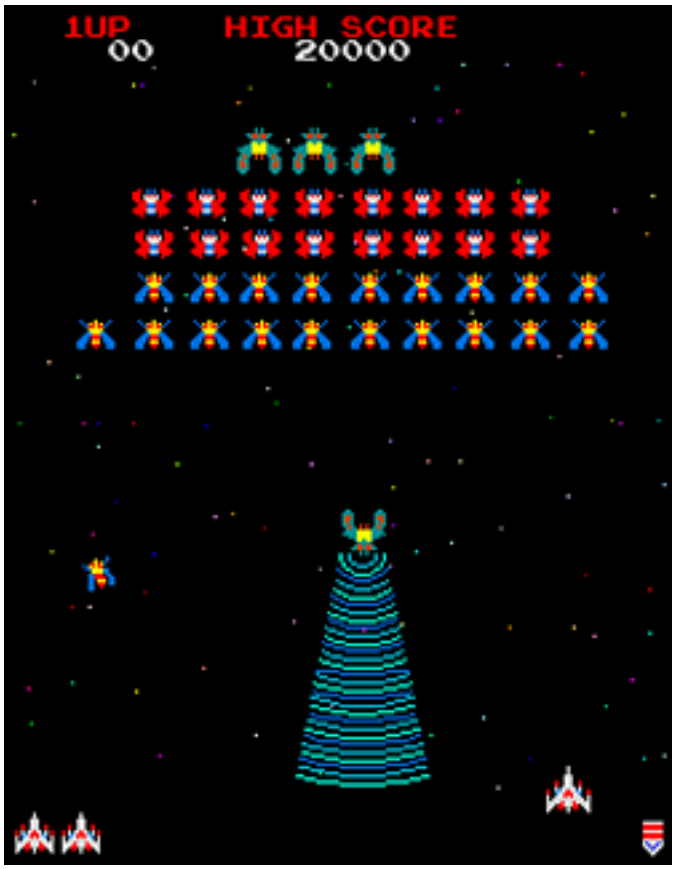

Figure 19: Galaga är ett känt spel med många fiender och kulor

Arrayer 1: Uppgift 1

Kör den här koden. Vad gör den?

```
float x = 0;

void setup()
{
  size(600, 50);
}

void draw()
{
  ellipse(x,25,50,50);
  x = x + 1;
  if (x > 625)
  {
    x = -25;
  }
}
```

Arrayer 1: lösning 1

En boll som åker åt höger i all evighet!

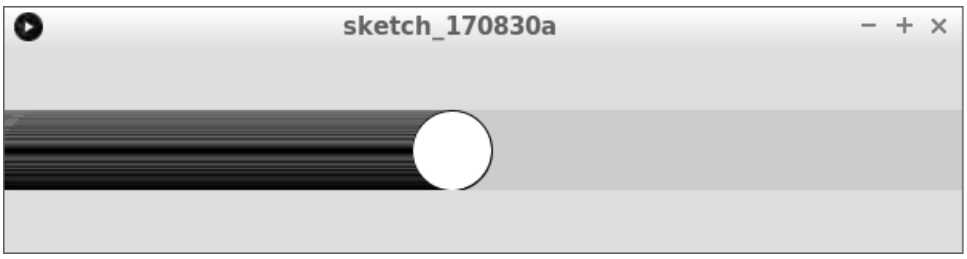

Figure 20: Arrays 1: lösning 1

Arrayer 1: Uppgift 2

Se till att lägga till en andra boll.

Figure 21: Två bollar går rätt för alltid

Tips: ändra namnet `x` till `x1`.

Skapa sedan en ny variabel som heter `x2`.

Arrayer 1: lösning 2

```
float x1 = 0;
float x2 = 100;

void setup()
{
  size(600, 50);
}

void draw()
{
  ellipse(x1,25,50,50);
  ellipse(x2,25,50,50);
  x1 = x1 + 1;
  x2 = x2 + 1;
  if (x1 > 625)
  {
    x1 = -25;
  }
  if (x2 > 625)
  {
    x2 = -25;
  }
}
```

Detta var sju rader extraarbete

Arrayer 1: Uppgift 3

Lägg till en tredje boll.

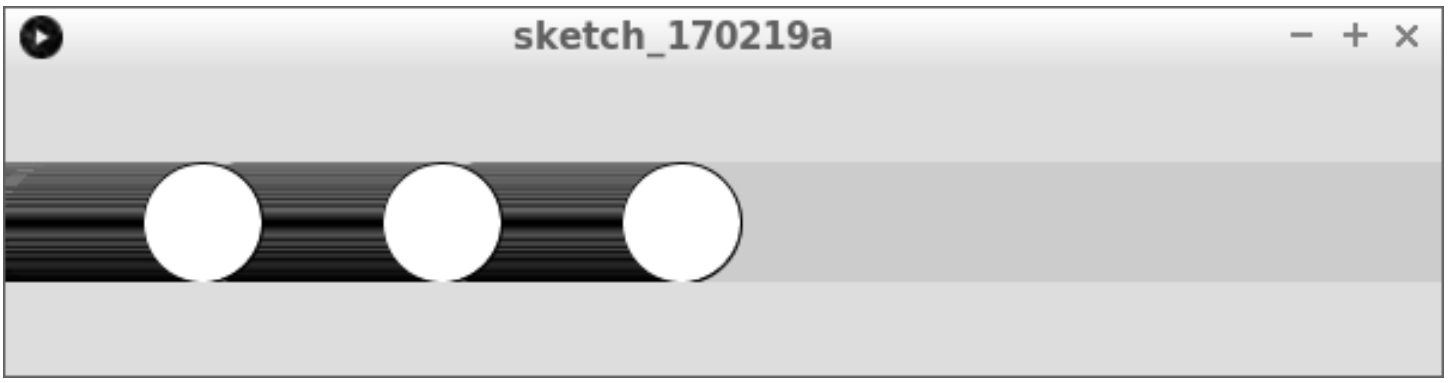

Figure 22: Arrays 1: command 3

Arrayer 1: lösning 3

```
float x1 = 0;
float x2 = 100;
float x3 = 200;

void setup()
{
  size(600, 50);
}

void draw()
{
  ellipse(x1,25,50,50);
  ellipse(x2,25,50,50);
  ellipse(x3,25,50,50);
  x1 = x1 + 1;
  x2 = x2 + 1;
  x3 = x3 + 1;
  if (x1 > 625)
  {
    x1 = -25;
  }
  if (x2 > 625)
  {
    x2 = -25;
  }
  if (x3 > 625)
  {
    x3 = -25;
  }
}
```

Detta var ytterligare sju rader extraarbete

Det här kan göras smartare, med arrayer!

Arrayer 1: vad är en array?

En array är som ett skåp med lådor.

Varje låda har ett nummer utanpå och varje låda kan innehålla ett nummer inuti.

Figure 23: Skåp med låda

Den första platsen (lådan) i en array har alltid numret `0`

Här ser du lådans nummer och det nummer som finns inuti den: Låda nummer `0` har nummer `42` inuti sig.

Figure 24: Skåp med numrerade lådor

	
`Arrayen börjar med index 0`	‘Den första platsen i arrayen har numret 0’

Arrayer 1: arbeta med en array

Anta att vi vill skapa en array av bråktal (`floats`) som kallas `hemliga_nummer`, då måste vi skriva följande ovanför `setup`:

```
float[] hemliga_nummer;
```

Med denna rad skapar du en array som heter `hemliga_nummer`. Hakparentesen bakom `float` betyder att det är en array.

	
`float[] hemliga_nummer`	'Kära dator, kom ihåg många bråktal i en array som kallas `hemliga_nummer`'

Det har ännu inte bestämts *hur många* bråktal det är. Ofta används "setup"-funktionen för att bestämma hur många nummer som ska kommas ihåg:

```
hemliga_nummer = new float[1];
```

Det här betyder att arrayen `hemliga_nummer` innehåller 1 plats.

	
`hemliga_nummer = new float[1]`	'Kära dator, låt `hemliga_nummer` innehålla 1 låda''

För att att göra en exakt kopia av skåpet med lådorna kan du använda följande kod:

```
hemliga_nummer[0] = 42;
```

Detta sparar siffran 42 på första plats i arrayen, dvs platsen med numret 0.

	
`hemliga_nummer[0] = 42`	‘Kära dator, spara siffran `42` på första platsen i arrayen `hemliga_nummer`’

Du kan också läsa värdet i lådorna:

```
float x = hemliga_nummer[0];
```

Med detta läser du den första platsen (lådan med index noll) och sparar den i variabeln `x`.

	
`float x = hemliga_nummer[0]`	‘Kära dator, se vad som finns i lådan med index noll och kom ihåg det som `x`’

Tillsammans får du detta program:

```
float[] hemliga_nummer;

void setup()
{
  size(400, 400);
  hemliga_nummer = new float[1];
  hemliga_nummer[0] = 42;
}

void draw()
{
  float x = hemliga_nummer[0];
  ellipse(x, 200, 300, 400);
}
```

Den här programmeringskoden ser inte särskilt trevlig ut med alla hakparenteser. Den är tänkt att visa dig hur du skapar, fyller i och läser arrayer.

Arrayer 1: Uppgift 4

Kör koden nedan.

```
float[] xs;

void setup()
{
  size(600, 50);
  xs = new float[3];
  xs[0] = 0;
  xs[1] = 100;
  xs[2] = 200;
}

void draw()
{
  ellipse(xs[0],25,50,50);
  ellipse(xs[1],25,50,50);
  ellipse(xs[2],25,50,50);
  xs[0] = xs[0] + 1;
  xs[1] = xs[1] + 1;
  xs[2] = xs[2] + 1;
  if (xs[0] > 625)
  {
    xs[0] = -25;
  }
  if (xs[1] > 625)
  {
    xs[1] = -25;
  }
  if (xs[2] > 625)
  {
    xs[2] = -25;
  }
}
```

Arrayer 1: lösning 4

Hej, samma som förut!

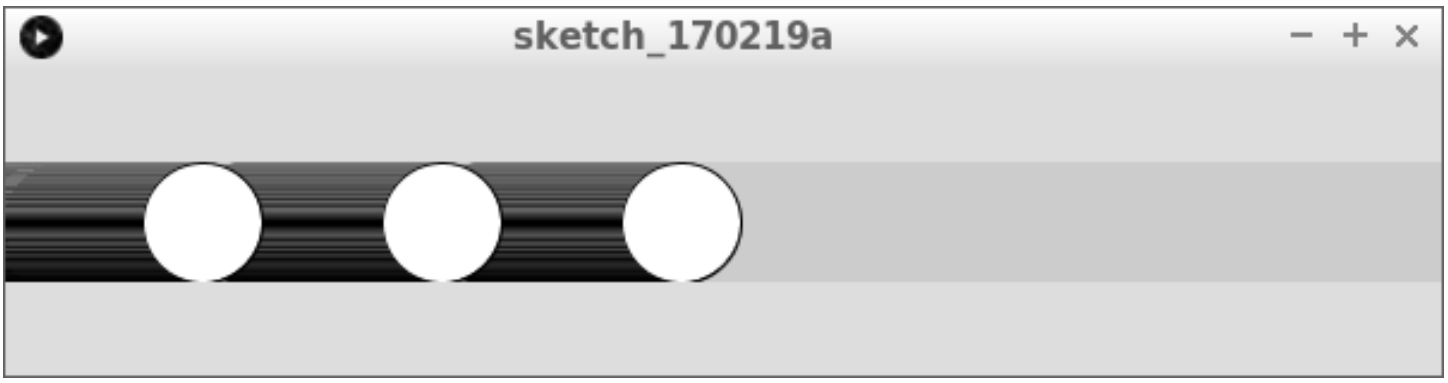

Figure 25: Arrays 1: lösning 4

Arrayer 1: Uppgift 5

Kör den här koden:

```
float[] xs;

void setup()
{
  size(600, 50);
  xs = new float[3];
  for (int i=0; i<3; ++i)
  {
    xs[i] = i * 100;
  }
}

void draw()
{
  for (int i=0; i<3; ++i)
  {
    ellipse(xs[i],25,50,50);
    xs[i] = xs[i] + 1;
    if (xs[i] > 625)
    {
      xs[i] = -25;
    }
  }
}
```

Bra programmerare använder hellre `for`-loopar än att kopiera och klistra in många gånger i onödan

Arrayer 1: lösning 5

Hej, samma som förut!

```
for (int i=0; i<3; ++i) {}
```

‘Kära dator, gör vad som står mellan måsvingarna med värden på `i` från `0` till mindre än `3` i steg om `1`’

Jag är en dum dator

```
 xs[0] = 0;
xs[1] = 100;
xs[2] = 200;
```

Jag är en smart dator

```
för (int i=0; i<3; ++i).
{
xs[i] = i * 100;
}
```

Arrayer 1: Uppgift 6

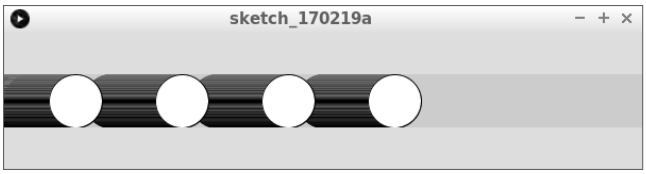

Figure 26: Arrays 1: command 6

Lägg nu till en fjärde boll.

Tips: ändra en `3` till en `4`.

Arrayer 1: lösning 6

```
float[] xs;

void setup()
{
  size(600, 50);
  xs = new float[4];
  for (int i=0; i<4; ++i)
  {
    xs[i] = i * 100;
  }
}

void draw()
{
  for (int i=0; i<4; ++i)
  {
    ellipse(xs[i], 25, 50, 50);
    xs[i] = xs[i] + 1;
    if (xs[i] > 625)
    {
      xs[i] = -25;
    }
  }
}
```

Arrayer 1: Uppgift 7

Gör nu programmet i helskärm. När bollarna lämnar skärmen i högerkanten får du dem att dyka upp igen i vänsterkanten. Använd `width` för detta.

Figure 27: Arrays 1: command 7

Arrayer 1: lösning 7

```
float[] xs;

void setup()
{
  fullScreen();
  xs = new float[4];
  for (int i=0; i<4; ++i)
  {
    xs[i] = i * 100;
  }
}

void draw()
{
  for (int i=0; i<4; ++i)
  {
    ellipse(xs[i], 25, 50, 50);
    xs[i] = xs[i] + 1;
    if (xs[i] > width + 25)
    {
      xs[i] = -25;
    }
  }
}
```

Arrayer 1: slutuppgift

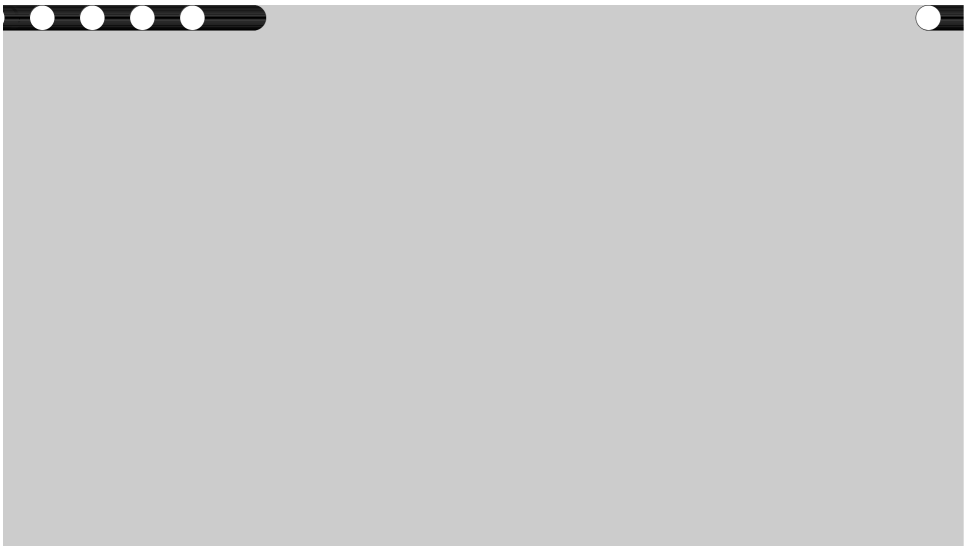

Figure 28: Arrays 1: slutuppgift

Skriv nu koden så att:

- Det finns sex bollar
- Bollarna åker åt vänster i all evighet

Arrayer 2

Ännu fler arrayer!

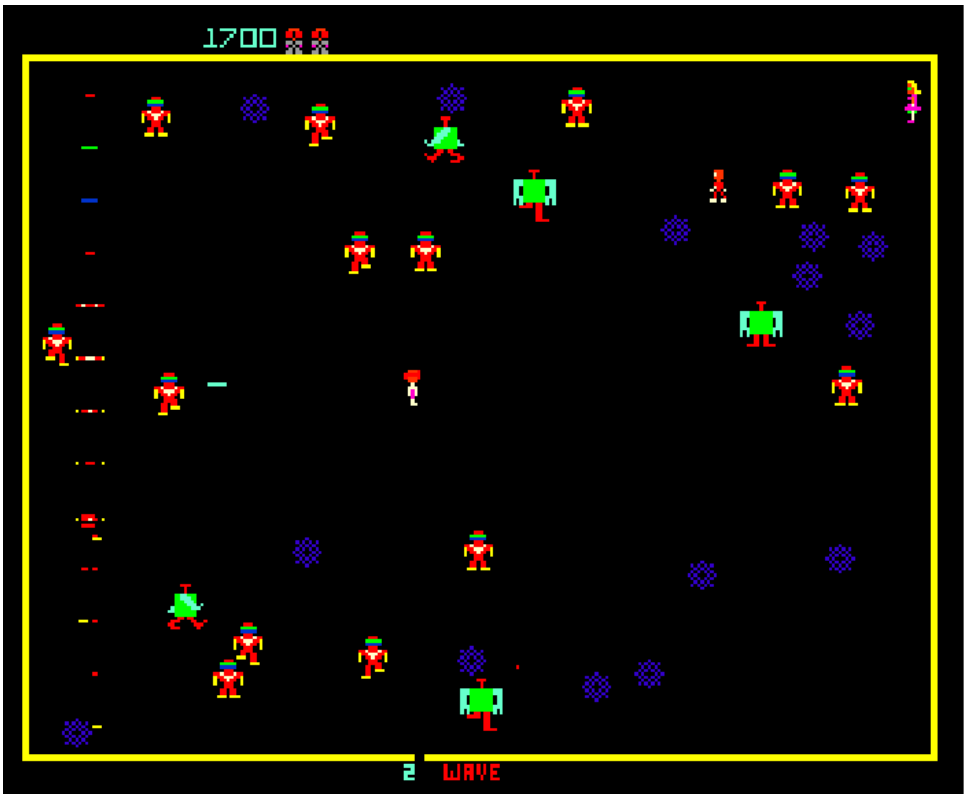

Figure 29: Robotron

Arrayer 2: Uppgift 1

Kör den här koden:

```
float x1 = 160;
float y1 = 100;

void setup()
{
  size(320, 200);
}

void draw()
{
  x1 += random(-1,1);
  y1 += random(-1,1);
  ellipse(x1, y1, 10, 10);
}
```

Arrayer 2: lösning 1

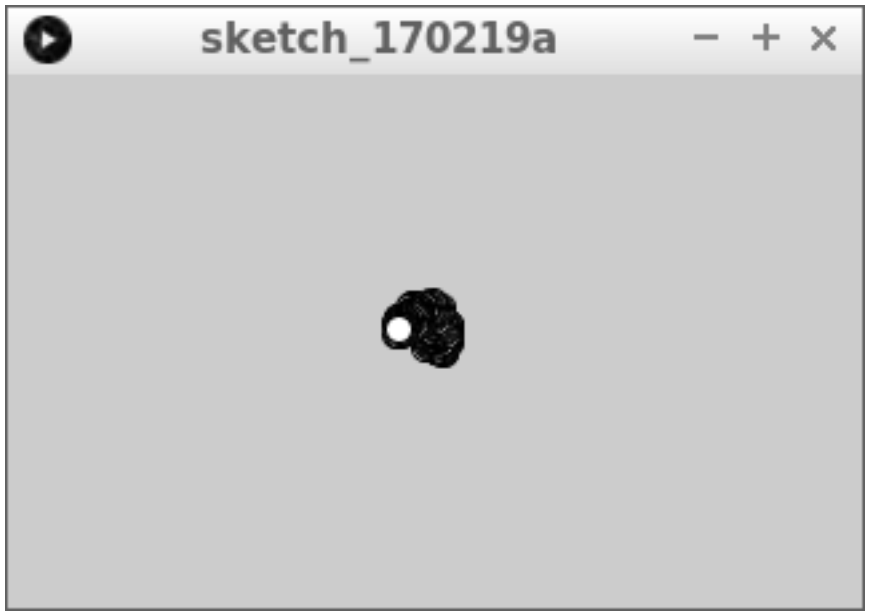

Figure 30: Arrays 2: solution 1

Hah, en rökpartikel.

Arrayer 2: Uppgift 2

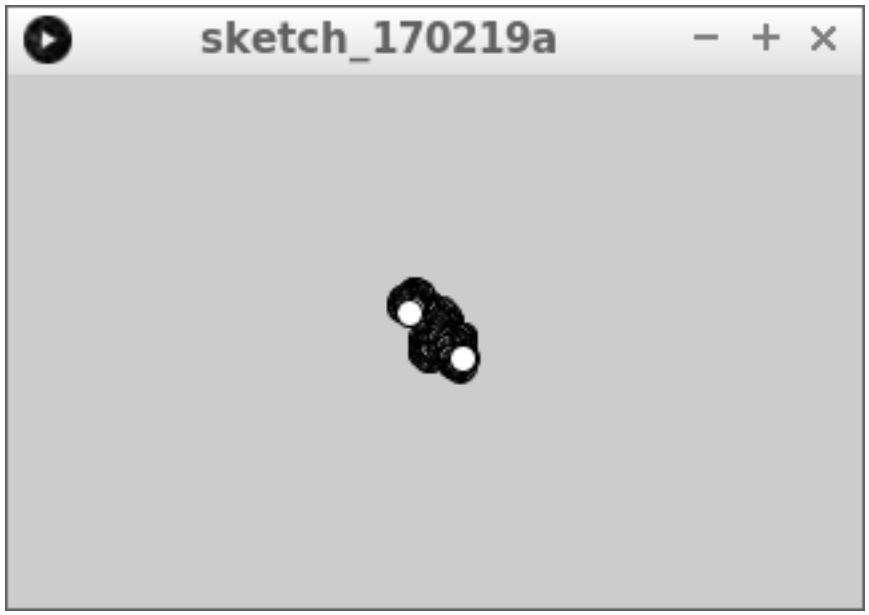

Figure 31: Arrays 2: assignment 2

Gör en andra rökpartikel.

Arrayer 2: lösning 2

```
float x1 = 160;
float y1 = 100;
float x2 = 160;
float y2 = 100;

void setup()
{
  size(320, 200);
}

void draw()
{
  x1 += random(-1,1);
  y1 += random(-1,1);
  ellipse(x1, y1, 10, 10);
  x2 += random(-1,1);
  y2 += random(-1,1);
  ellipse(x2, y2, 10, 10);
}
```

Arrayer 2: Uppgift 3

Använd nu en array, utan `for`-loop.


`float[] xs` ‘Kära dator, kom ihåg många bråktal i en array som heter `xs`’


`xs = new float[2]` ‘Kära dator, gör så att `xs` innehåller två lådor’

Tips: använd `xs[0]` istället för `x1` och `xs[1]` istället för `x2`

Arrayer 2: lösning 3

```
float[] xs;
float[] ys;

void setup()
{
  size(320, 200);
  xs = new float[2];
  ys = new float[2];
  xs[0] = 160;
  xs[1] = 160;
  ys[0] = 100;
  ys[1] = 100;
}

void draw()
{
  xs[0] += random(-1,1);
  ys[0] += random(-1,1);
  ellipse(xs[0], ys[0], 10, 10);
  xs[1] += random(-1,1);
  ys[1] += random(-1,1);
  ellipse(xs[1], ys[1], 10, 10);
}
```

Arrayer 2: Uppgift 4

Figure 32: Arrays 2: uppgift 4

Använd nu `for`-loopar. Gör så att både arrayen `xs` och `ys` innehåller tre lådor.

	
`for (int i=0; i<3; i=i+1) {}`	'Kära dator, gör vad som står mellan måsvingarna. Låt `i` starta med värdet `0` och räkna upp så länge värdet är mindre `3`, med `1` steg i taget'

Arrayer 2: lösning 4

```
float[] xs;
float[] ys;

void setup()
{
  size(320, 200);
  xs = new float[3];
  ys = new float[3];
  for (int i=0; i<3; ++i)
  {
    xs[i] = 160;
    ys[i] = 100;
  }
}

void draw()
{
  for (int i=0; i<3; ++i)
  {
    xs[i] += random(-1,1);
    ys[i] += random(-1,1);
    ellipse(xs[i], ys[i], 10, 10);
  }
}
```

Arrayer 2: Uppgift 5

Figure 33: Arrays 2: uppgift 5

Varje rökpartikel får nu sin egen röda kantfärg:

- Skapa en tredje array som heter `rs`, för de röda nyanserna på rökpartiklarna
- I `rs` måste siffrorna `0`, `64`, `128` och `196` finnas
- De röda nyanserna ska slumpmässigt öka eller minska i rödhet
- Kanten på den första rökpartikeln ska få den första röda nyansen. Tips: använd `stroke`

Tips: använd `stroke` för kantfärgen

Arrayer 2: lösning 5

```
float[] xs;
float[] ys;
float[] rs; //Roda varden

void setup()
{
  size(320, 200);
  xs = new float[4];
  ys = new float[4];
  rs = new float[4];
  for (int i=0; i<4; ++i)
  {
    xs[i] = 160;
    ys[i] = 100;
    rs[i] = i * 64;
  }
}

void draw()
{
  for (int i=0; i<4; ++i)
  {
    xs[i] += random(-1,1);
    ys[i] += random(-1,1);
    rs[i] += random(-1,1);
    stroke(rs[i], 0, 0);
    ellipse(xs[i], ys[i], 10, 10);
  }
}
```

Arrayer 2: Slutuppgift

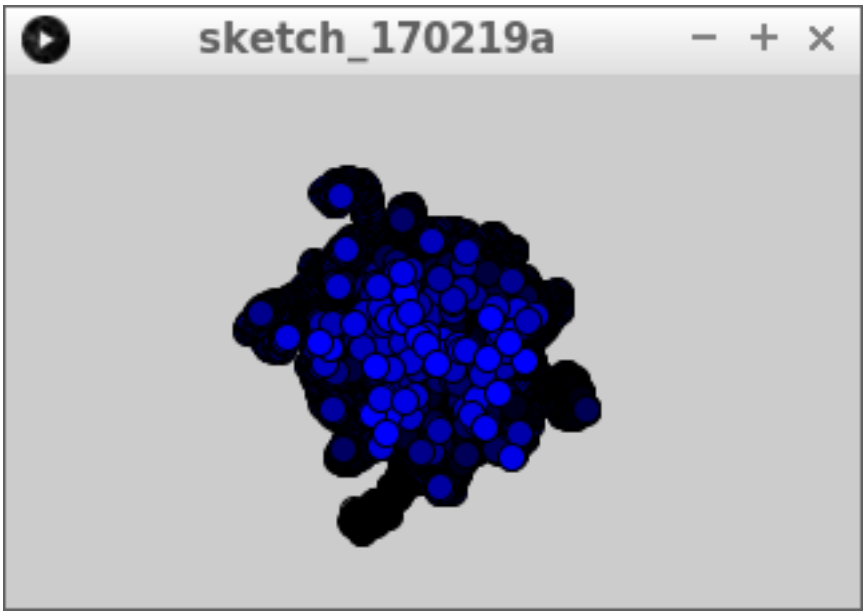

Figure 34: Arrays 2: final assignment

Gör nu koden så att:

- det finns 256 rökpartiklar.
- varje rökpartikel har sitt eget *blå* värde
- den första rökpartikeln har ett blått värde på noll. Den andra rökpartikeln har ett blått värde på ett. Den tredje rökpartikeln har ett blått värde på två. Och så vidare
- ifyllnadsfärgen ska vara blå den här gången, inte kanten

Tips: använd `fill` för ifyllnadsfärgen