



Figure 1: Bok 1

#	Beskriving
1	Ett vackert program
2	Bollen till höger
3	width och height
4	point och random

# Contents

Förord	1
Ett vackert program	2
Boll till höger	6
width och height	25
point och random	33

## Förord

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

### Om den här boken

Denna bok är licensierad av CC-BY-NC-SA.



Figure 1: Licensen för denna bok

(C) Richèl Bilderbeek och alla lärare och alla elever

Med det här häftet kan du göra vad du vill, så länge du hänvisar till originalversionen på denna webbplats: [https://github.com/richelbilderbeek/processing\\_foer\\_tonaaringar](https://github.com/richelbilderbeek/processing_foer_tonaaringar). Detta häfte kommer alltid att förbli gratis, fri och öppet.

Det är fortfarande en lite slarvig bok. Det finns stafvel och *layouten är inte alltid vacker*. Eftersom den här boken finns på en webbplats kan alla som tycker att den här boken är för slarvig göra den mindre slarvig.

## Ett vackert program

Processing är ett programmeringsspråk som utvecklats för artister och mycket lämplig för att göra spel och vackra saker.

I den här lektionen lär vi oss

- hur vi börjar bearbeta
- hur man kopierar kod till Processing
- hur man startar programmet

Så här ser programmet ut:

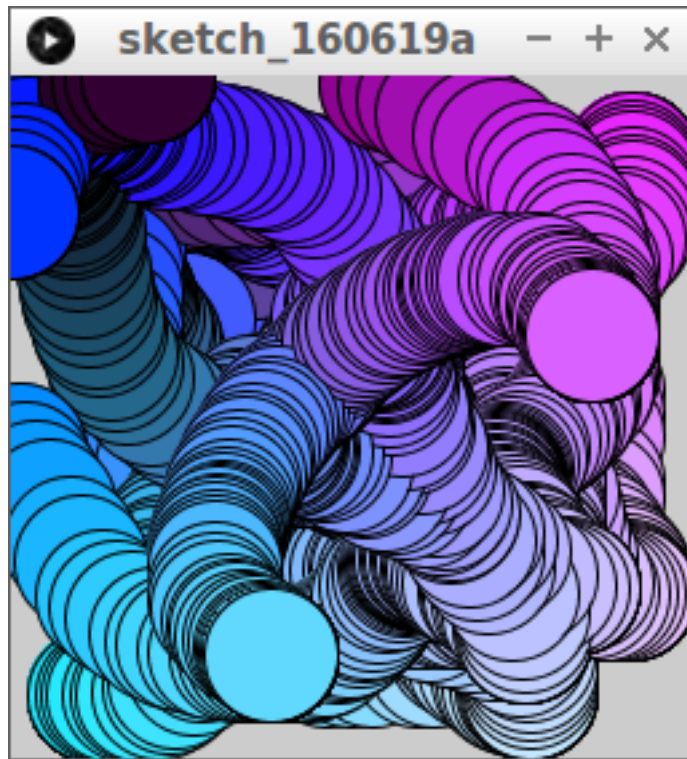


Figure 2: Ett vackert program

## Ett trevligt program: intro

Behandlingen börjar med ett tomt program utan kod:

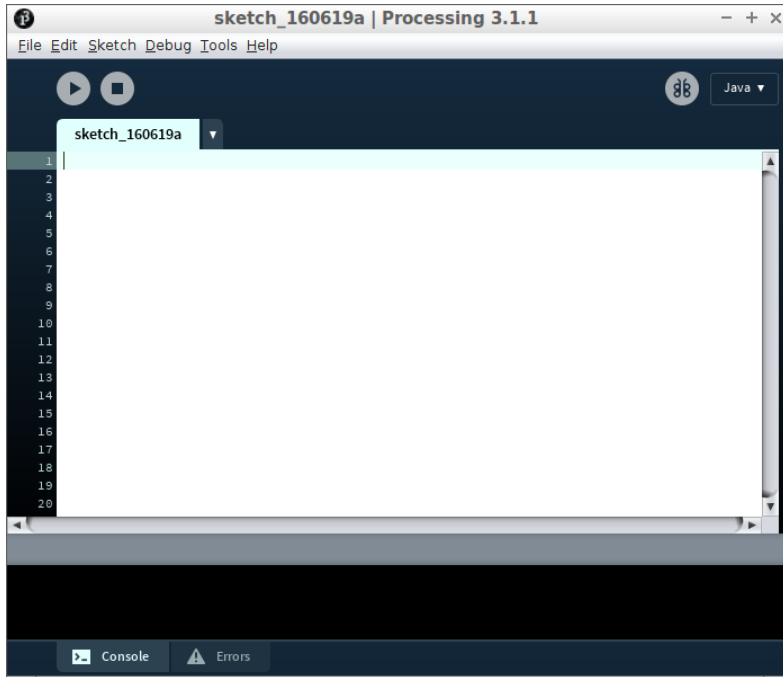


Figure 3: Processing utan kod

Detta är programmeringskoden som vi kommer att använda:

```
void setup()
{
  size(256,256);
}

void draw()
{
  fill(mouseX, mouseY, mouseX + mouseY);
  ellipse(mouseX, mouseY, 50, 50);
  fill(mouseY, mouseX, 255);
  ellipse(mouseY, mouseX, 50, 50);
}
```

Vi kommer att förklara exakt vad koden gör senare. För nu är det tillräckligt att veta att det gör något vackert.

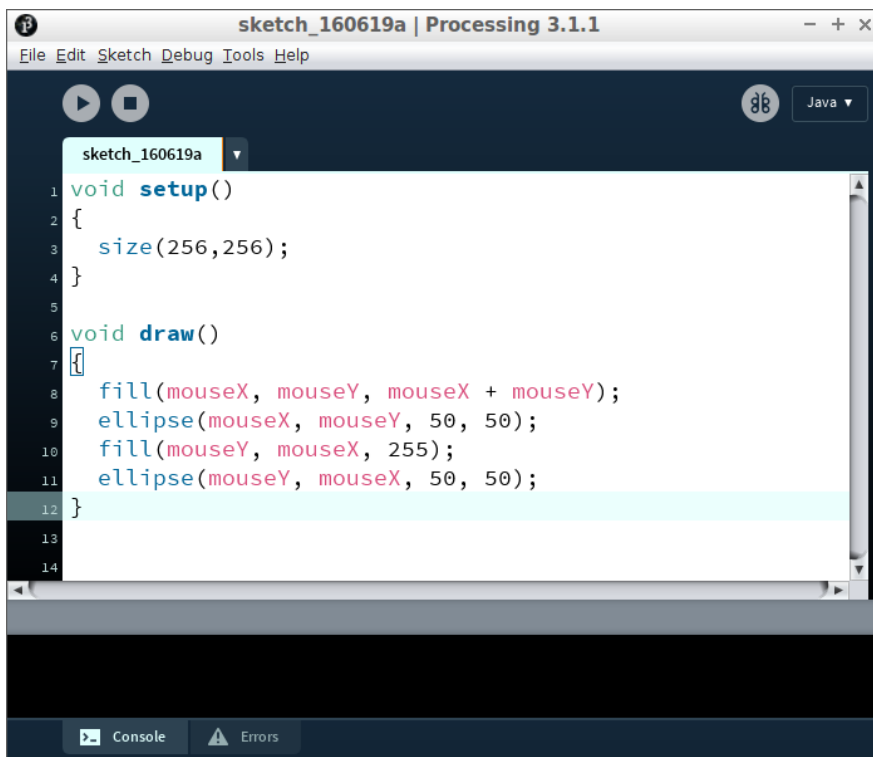


Figure 4: Processing med kod

## Ett trevligt program: slutuppgift

- Börja bearbeta
- Kör den här koden genom att klicka på knappen “Run”

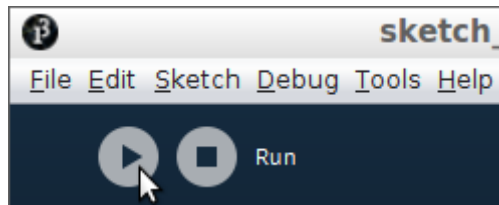


Figure 5: Run knappen



Framgång? Visa detta för en vuxen för ett klistermärke!

---

---

## Boll till höger

I den här lektionen ska vi göra en boll till höger.

Lär dig också i denna lektion vad en variabel är. Du kan knappt programmera utan variabler.

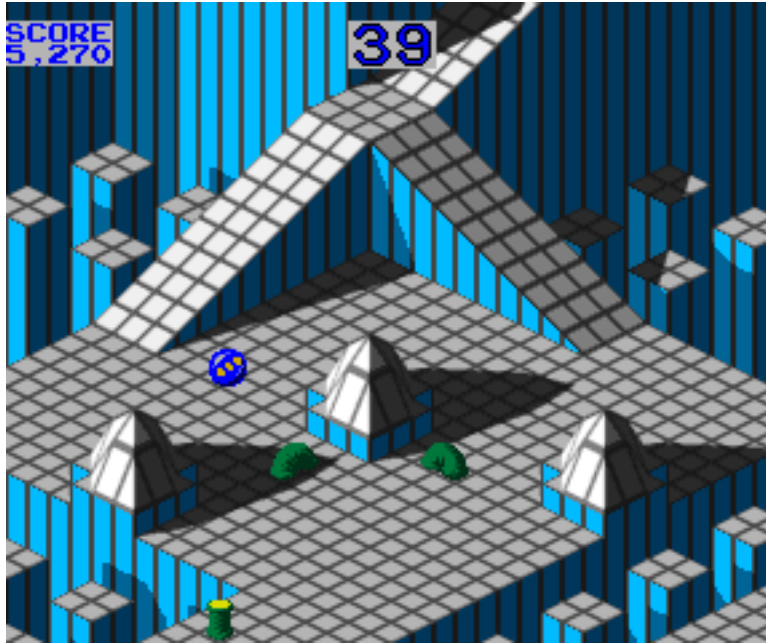


Figure 6: Marble Madness

## Boll till höger: intro

Ange följande kod:

```
float x = 60;

void setup()
{
  size(250, 200);
}

void draw()
{
  ellipse(x, 50, 40, 30);
  x = x + 1;
}
```

Tryck sedan på 'Run'.

Om det finns röda bokstäver har du gjort ett stavfel. Titta noga och rätta stavfel.

Om allt går bra ser du en boll som rör sig till höger (se figur `Boll till höger: intro`).

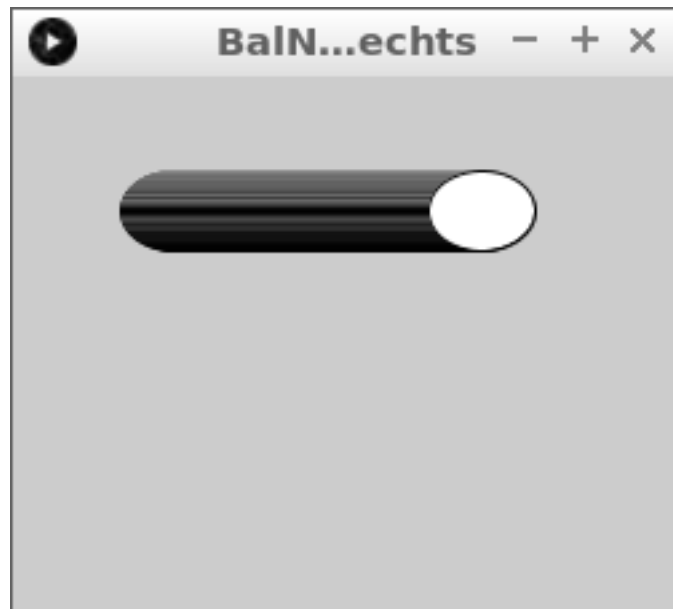


Figure 7: Boll till höger: intro



## Boll till höger: uppgift 1



Figure 8: Boll till höger: uppgift 1

Skärmen är nu 250 pixlar bred. Gör den nu 300 pixlar bred.

Ändra koden och tryck på “Run”.

## Boll till höger: lösning 1

Det finns en “250” i koden. Det räcker med att ändra detta till 300:

```
float x = 60;

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

void draw()
{
  ellipse(x, 50, 40, 30);
  x = x + 1;
}
```



---

size(300, 200);

‘Kära dator, skapa ett fönster 300 pixlar brett och 200 pixlar högt.’

---

## Boll till höger: uppgift 2

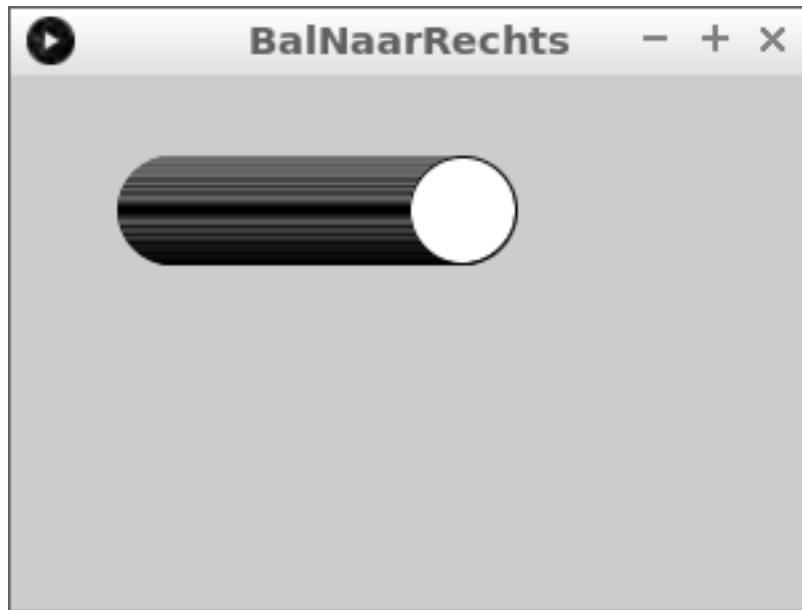


Figure 9: Boll till höger: uppgift 2

Bollen är nu äggformad: den är nu 40 pixlar bred och 30 pixlar hög. Gör nu bollen rund: 40 pixlar bred och 40 pixlar hög.

## Boll till höger: lösning 2

`ellips (x, 50, 40, 30);` drar bollen. 40, 30 gör bollen äggformad. Att göra denna till 40, 40 gör bollen rund.

```
float x = 60;

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

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x + 1;
}
```



`ellips (x, 50,40,30);`



‘Kära dator, rita en oval x pixlar till höger, 50 pixlar nedåt, vilket är 40 pixlar brett och 30 pixlar högt.’

---

### Boll till höger: uppgift 3

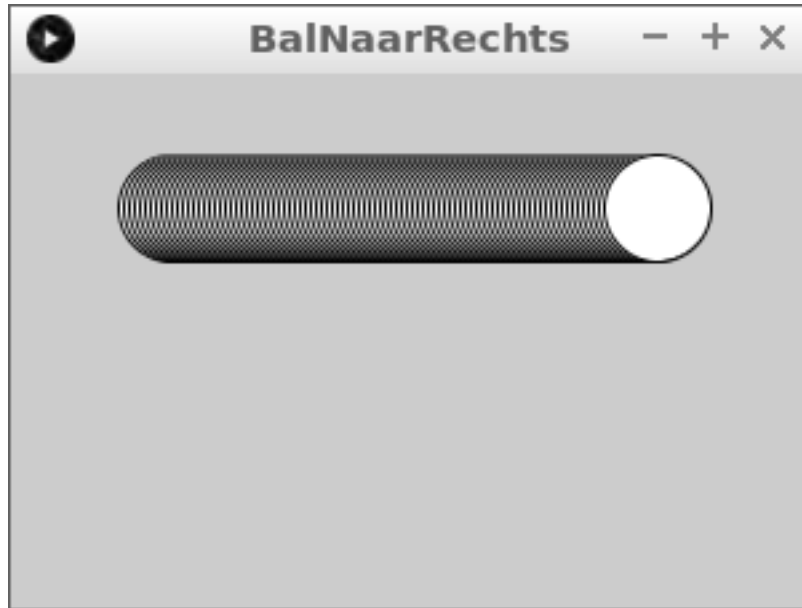


Figure 10: Boll till höger: uppgift 3

Bollen rör sig nu till höger med en pixelhastighet i taget. Låt bollen gå till höger dubbelt så snabbt

## Boll till höger: lösning 3

$x = x + 1$ ; flyttar bollen. Ändra detta till  $x = x + 2$ ;. Koden blir då:

```
float x = 60;

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

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x + 2;
}
```



---

$x = x + 1$ ;	‘Kära dator, gör x till en högre.’
$x += 1$ ;	‘Kära dator, gör x en högre.’
$x++$ ;	‘Kära dator, gör x en högre.’
$++x$ ;	‘Kära dator, gör x en högre.’

---

## Boll till höger: uppgift 4

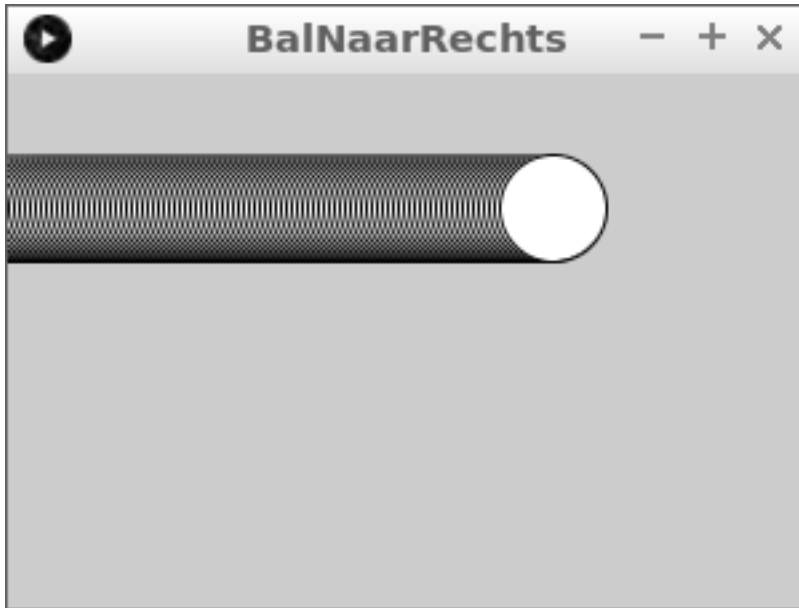


Figure 11: Boll till höger: uppgift 4

I början är bollens mitt 60 pixlar till höger. Kan du också få cirkeln att börja 0 pixlar till höger?

## Boll till höger: lösning 4

`float x = 60;` bestämmer detta. Ändra detta till `float x = 0;`. Koden blir då:

```
float x = 0;

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

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x + 2;
}
```



---

```
void setup () {}
```

‘Kära dator, gör vad som helst inom lockiga parenteser .’

---



## Boll till höger: uppgift 5

Haha, den här lektionen kallas 'Boll till höger', men vi ska också göra en boll till vänster!

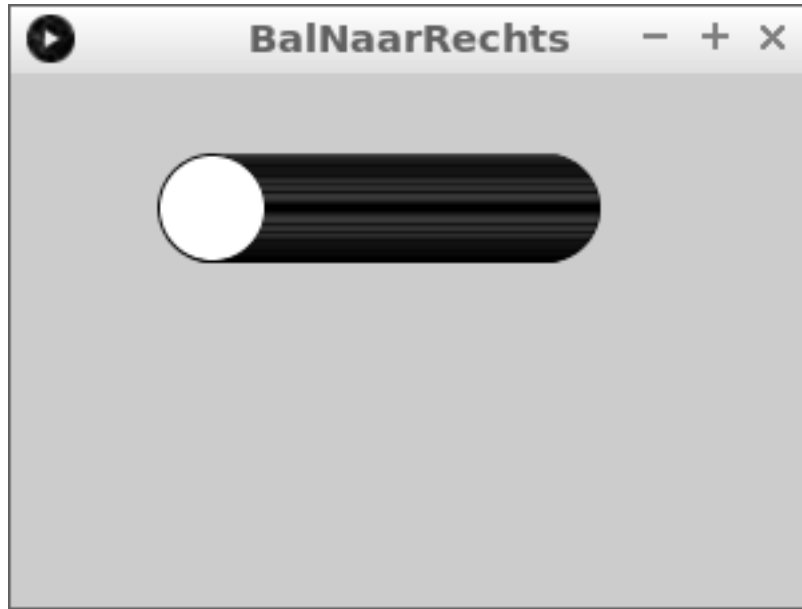


Figure 12: Boll till höger: uppgift 5

Låt nu bollen börja på höger sida av skärmen och flytta till vänster.

## Boll till höger: lösning 5

För att få bollen till höger måste du använda `float x = 500;` (eller något annat högt tal). För att få bollen att gå åt vänster måste du använda `x = x - 1;`. Koden blir då:

```
float x = 200;

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

void draw()
{
  ellipse(x, 50, 40, 40);
  x = x - 1;
}
```



---

```
void draw () {}
```

‘Kära dator, gör vad som helst inom lockiga parenteser hela tiden.’

---

## Boll till höger: vad är en variabel?

På den första raden använder vi en variabel:

```
float x = 50;
```

I klartext är detta: “Kära dator, kom ihåg talet x med ett börjanvärde på 50.”.



---

`float x = 50;` ‘Kära dator, kom ihåg talet x med ett börjanvärde på 50.’

---

En variabel är ett datorminne med ett namn. Datorn kan använda det namnet för att avgöra var i minnet den ska leta.

Variabler som tillhör dig (och nästan varje människa) är: namn, ålder, födelsedatum, adress, telefonnummer, epostadress och mycket mer. Om någon frågar dig i din ålder vet du vilket nummer du ska säga.



---

pengar  
1000000

‘Kära dator, berätta hur mycket pengar jag har på banken.’  
‘Jättebra!’

---

Tillbaka till den första raden i vår kod:

```
float x = 50;
```

Ordet `x` är namnet på en variabel. I det här fallet hur långt cirkeln är till höger. Ordet `float` betyder att `x` är ett (komma) tal. Symbolen `=` betyder 'kommer att vara från och med nu'. Talet `50` är det initiala värdet. Semikolon `;` anger slutet på en mening (som punkten i en svensk text).

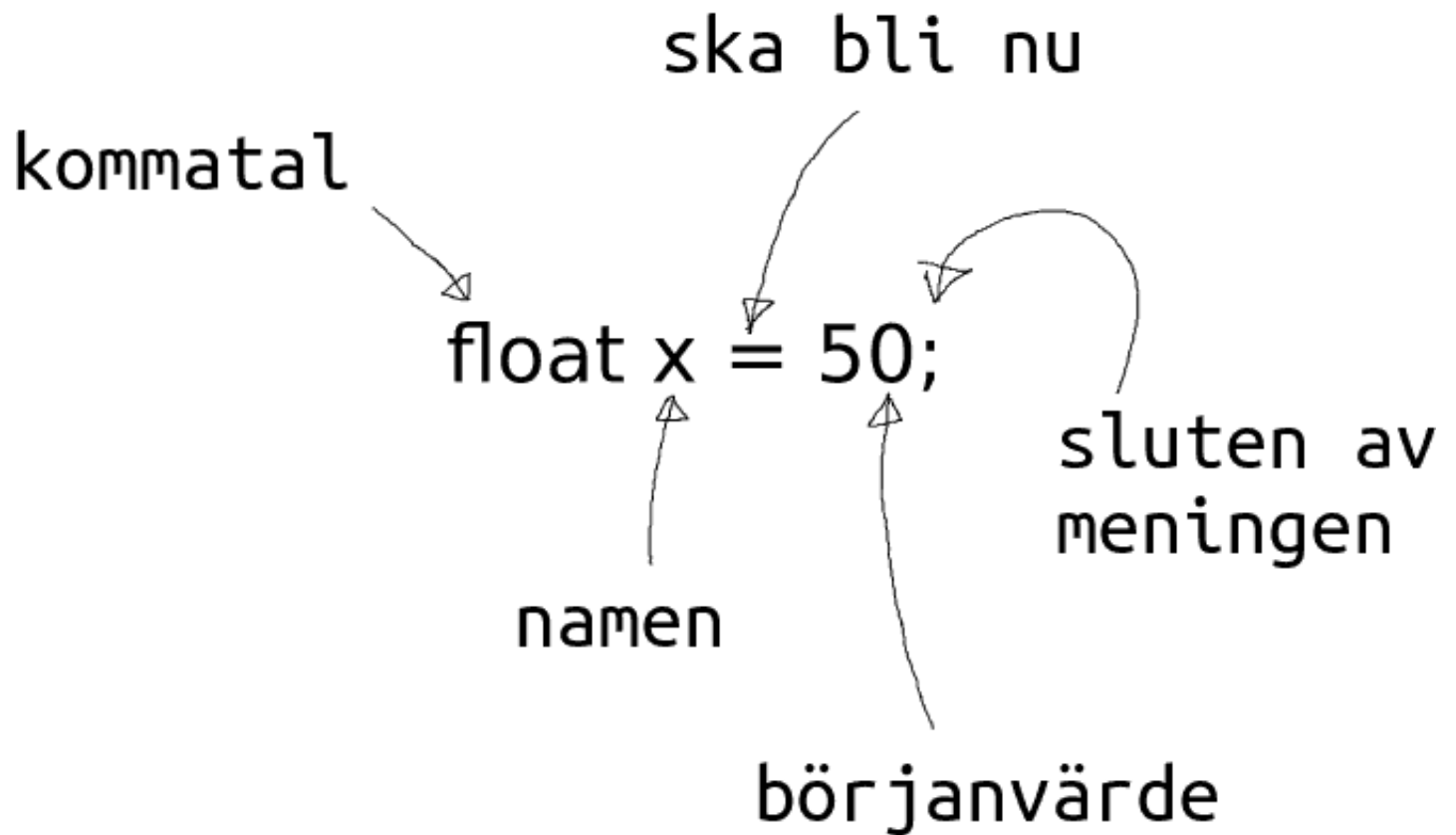




Figure 13: Förklaring av `float x = 50;`

---

	
<code>float</code>	'ett kommatal'
<code>=</code>	'ska bli nu'
<code>;</code>	'sluten av meningen'

---

## Boll till höger: uppgift 6

Haha, den här lektionen kallas “Boll till höger”, men vi kommer också att få en boll att röra sig nedåt!

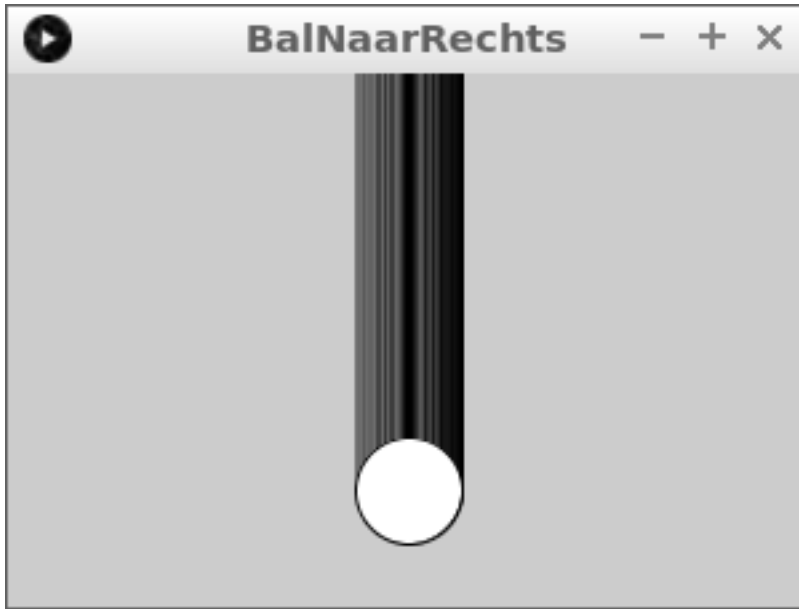


Figure 14: Boll till höger: uppgift 6

- Ändra namnet på variabeln  $x$  till  $y$
- Starta en boll högst upp på skärmen
- Bollen måste vara 150 pixlar till höger
- Bollen måste gå ner i en rak linje. Tips: bollen är nu 50 pixlar nere

## Boll till höger: lösning 6

```
float y = 0;

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

void draw()
{
  ellipse(150, y, 40, 40);
  y = y + 1;
}
```

## Boll till höger: uppgift 7

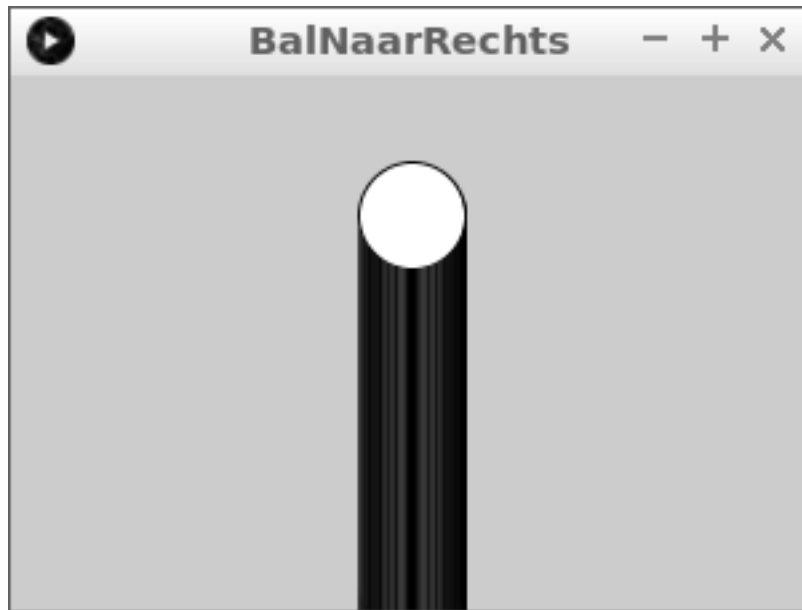


Figure 15: Boll till höger: uppgift 7

Nu ska vi få bollen att röra sig snabbare och uppåt

- Starta en boll längst ner på skärmen
- Bollen måste gå upp i en rak linje
- Bollen måste gå dubbelt så snabbt

## Boll till höger: lösning 7

```
float y = 200;

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

void draw()
{
  ellipse(150, y, 40, 40);
  y = y - 1;
}
```



## Boll till höger: slutuppgift

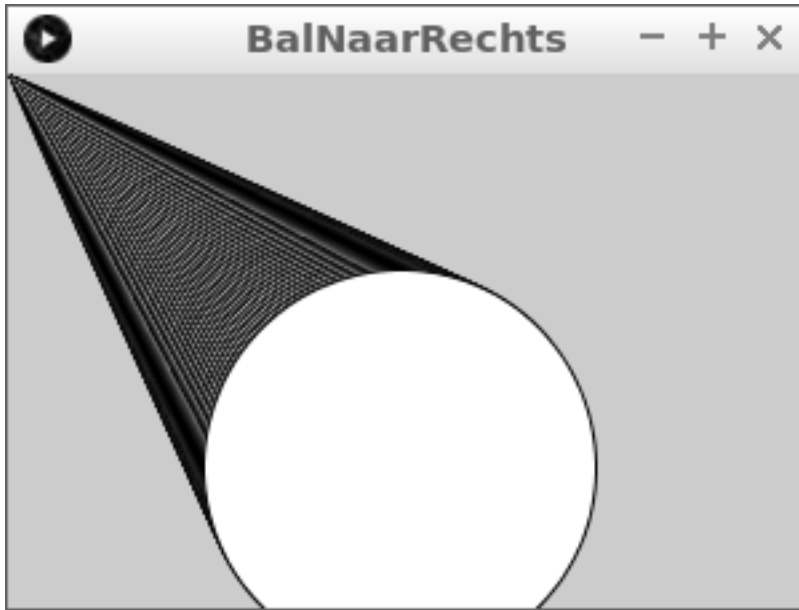


Figure 16: Boll till höger: slutuppgift

- bollen måste gå diagonalt till höger-ner
- bollen måste öka i bredd och höjd
- se även figur slutuppgift 'Boll till höger'

## width och height

I den här lektionen lär du dig använda `width` och `height`.

### width och height: intro

```
void setup()
{
  size(256, 256);
}

void draw()
{
  ellipse(128, 128, 256, 256);
}
```



---

```
size(800, 400);
ellipse (60,50,40,30);
```

‘Kära dator, gör ett fönster 800 pixlar brett och 400 pixlar högt.’  
‘Kära dator, rita en oval 60 pixlar till höger, 50 pixlar nedåt, som är  
40 pixlar breda och 30 pixlar höga’

---

Skriv in koden ovan och kör den.

### width och height: uppgift 1

Gör nu fönstret 128 x 128 pixlar litet.

### width och height: lösning

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(64, 64, 128, 128);
}
```

## width och height

`width` och `height` är inbyggda i bearbetning, så att ditt program fortfarande fungerar när du ändrar storlek på skärmen.

Nu fungerar våra program bara för en skärm av en viss storlek. Varje gång du väljer en ny storlek måste du skriva in mycket kod igen!

Om vi vet skärmens bredd och höjd vet vi också vilka siffror som ska vara i ellips:

- ovalens x-koordinat är halva bredden
- y-koordinaten för ovalen är halva höjden
- ovalens bredd är skärmens bredd
- höjden på ovalen är skärmens höjd

Bearbetningen känner till skärmens bredd och höjd: Skärmens bredd kallas `width` och höjden kallas `height`

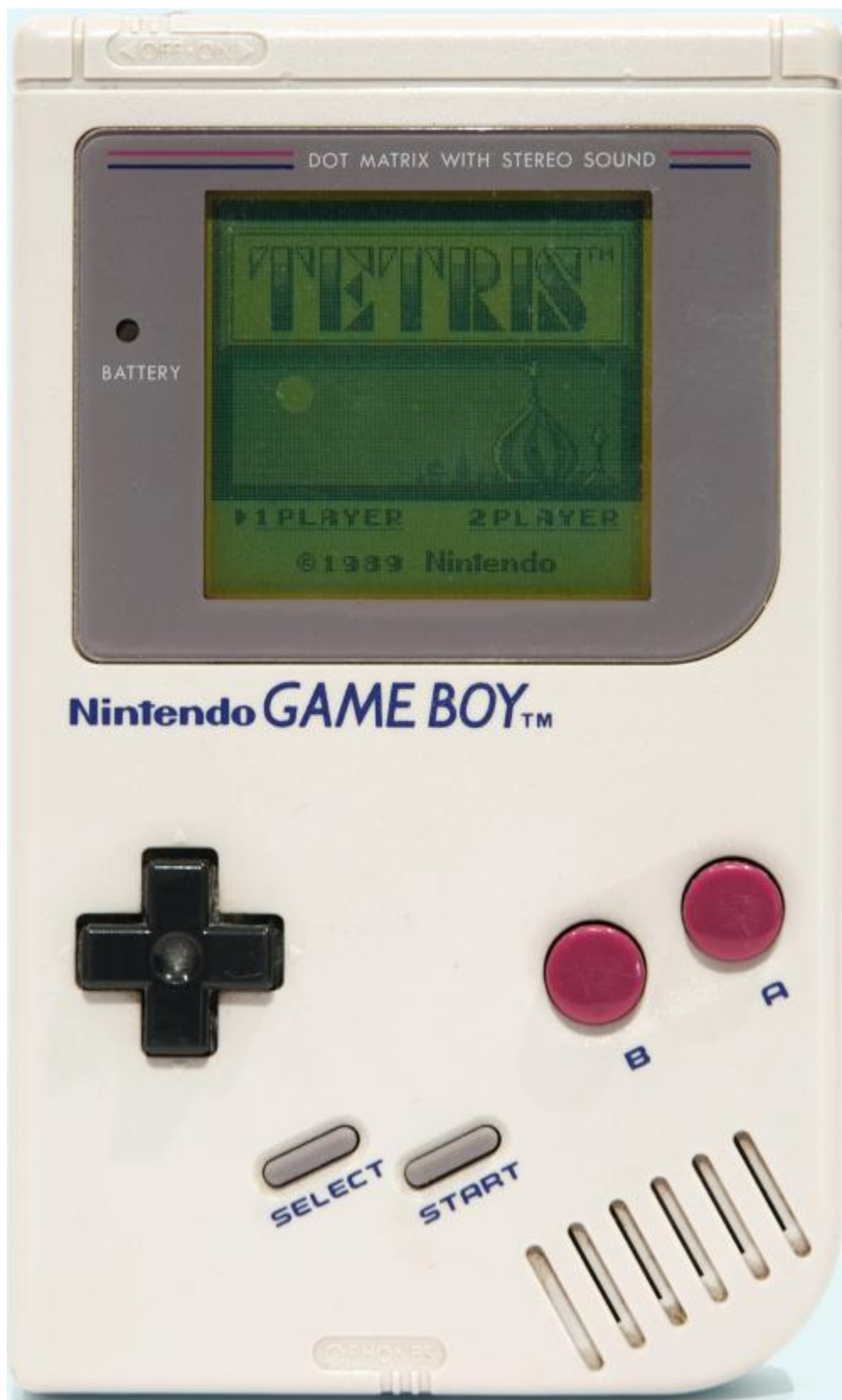


Figure 17: Gameboy har en skärm på 160 x 144 pixlar

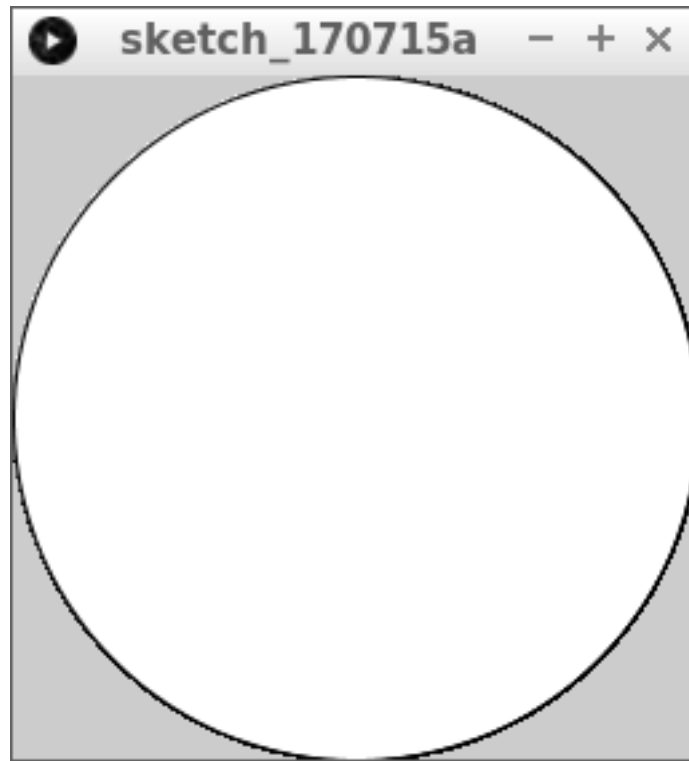


Figure 18: width och height: intro

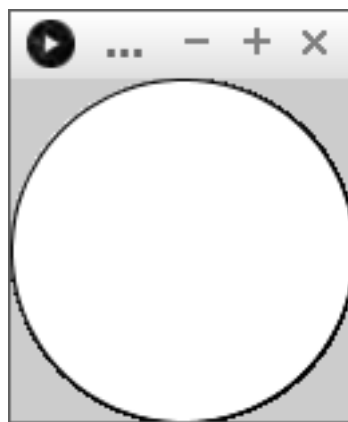


Figure 19: width och height: uppgift 1

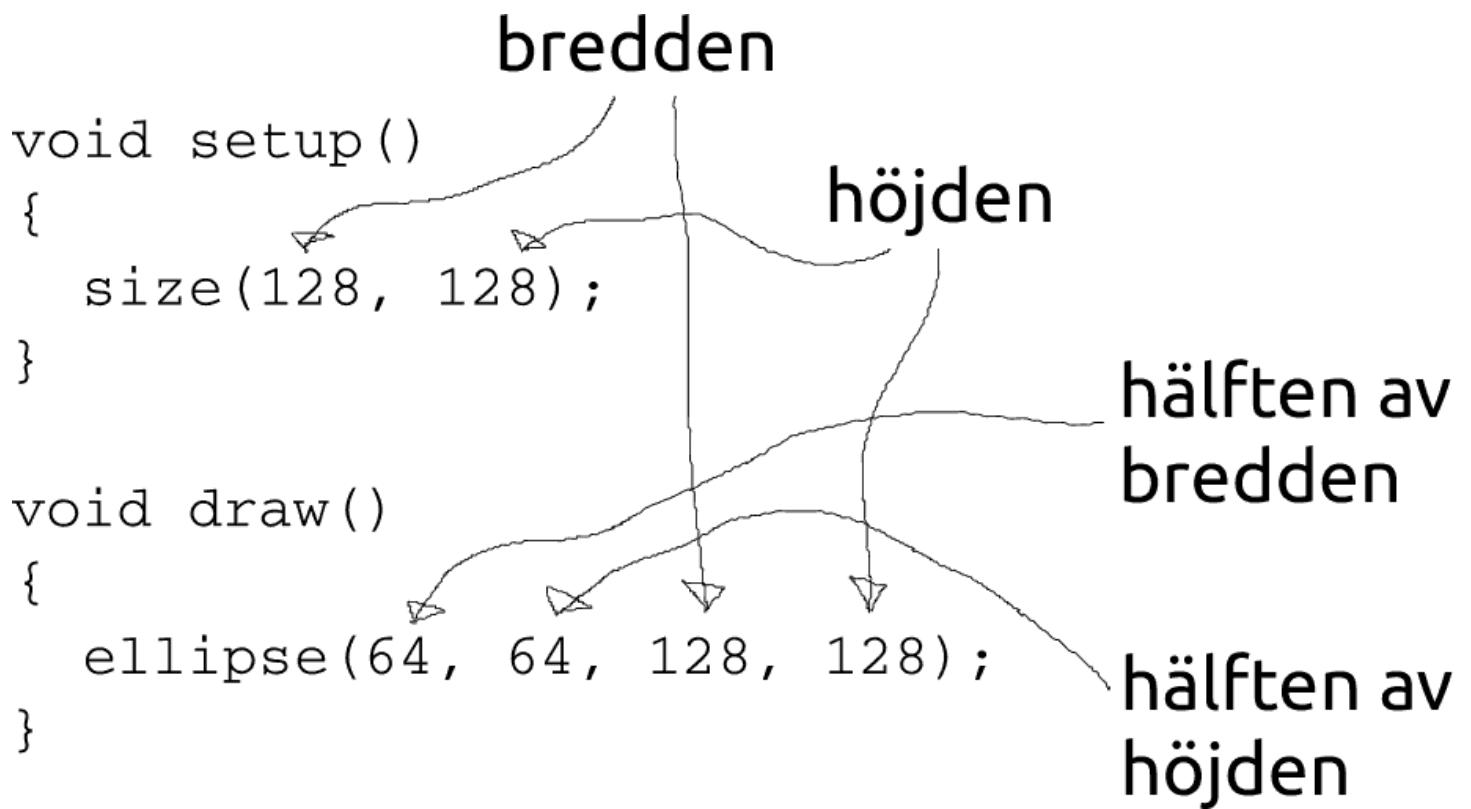




Figure 20: Vad du vill säga

	
width	‘Kära dator, ange här hur många pixlar fönstret är brett.’
height	‘Kära dator, ange här hur många pixlar fönstret är högt.’

Dessa siffror bestäms när du använder storlek för att definiera storleken på din skärm.

## width och height: uppgift 2

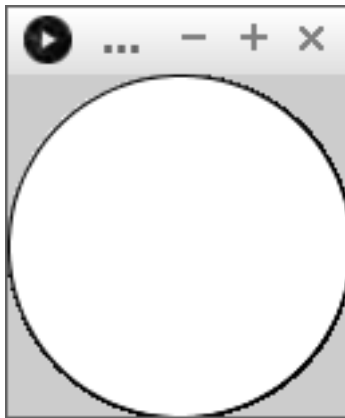


Figure 21: width och height: uppgift 2

Skapa ett program som ritar en oval som fyller skärmen:

- Ändra den första 64 till `width / 2`
- Ändra den andra 64 till `height / 2`
- Ändra den första 128 till `width`
- Ändra den andra 128 till `height`



---

/ 'dividerat med', en delningsrad som du har med bråk, :

---

## width och height: lösning 2

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(width / 2, height / 2, width, height);
}
```

## width och height: uppgift 3

Ställ in cirkelns mitt för att koordinera (0, 0).

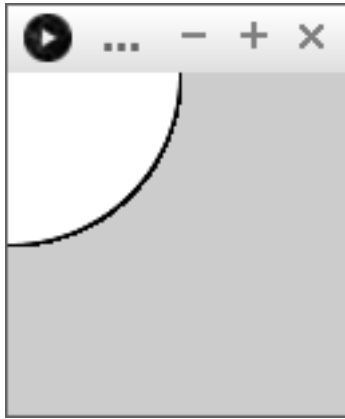


Figure 22: width och height: uppgift 3

### width och height: lösning 3

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(0, 0, width, height);
}
```

### width och height: uppgift 4

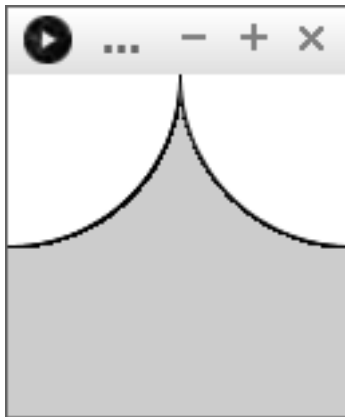


Figure 23: width och height: uppgift 4

Skapa en andra cirkel centrerad i det övre högra hörnet. Använd `width` och/eller `height`.

### width och height: lösning 4

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(0, 0, width, height);
  ellipse(width, 0, width, height);
}
```

### width och height: uppgift 5

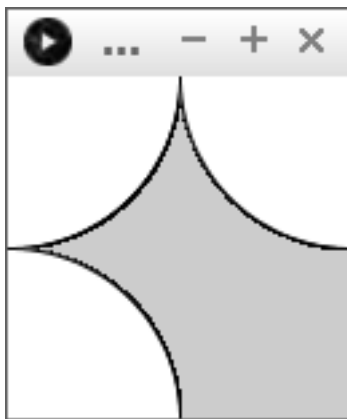


Figure 24: width och height: uppgift 5

Gör en tredje cirkel centrerad i nedre vänstra hörnet. Använd `width` och/eller `height`.

### width och height: lösning 5

```
void setup()
{
  size(128, 128);
}

void draw()
{
  ellipse(0, 0, width, height);
  ellipse(width, 0, width, height);
  ellipse(0, height, width, height);
}
```



`width` och `height`: sista kommandot

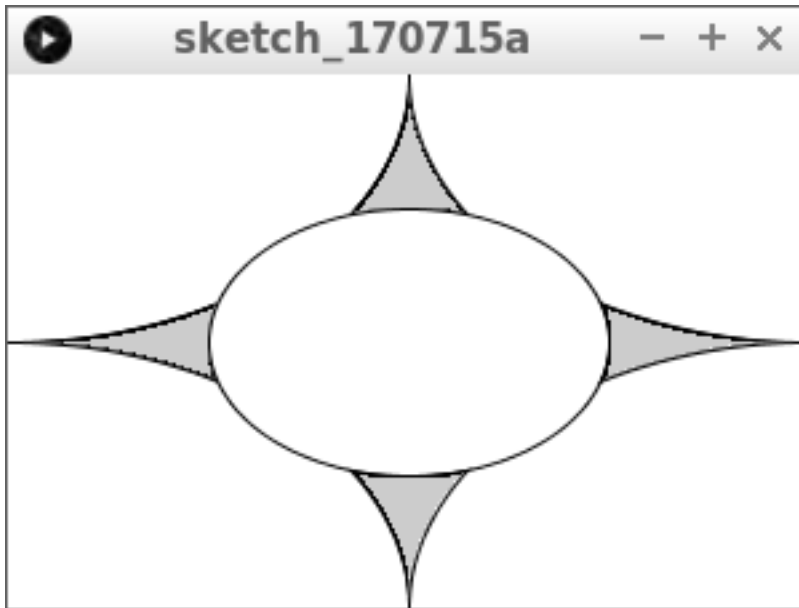


Figure 25: `width` och `height`: slutuppgift

- Gör fönstret 300 pixlar brett och 200 pixlar högt
- Gör en fjärde cirkel centrerad i nedre högra hörnet
- Gör en femte cirkel som är i mitten och är dubbelt så liten
- Använd `width` och/eller `height` (ingen 100, 150, 200 eller 300!)

## point och random

I den här lektionen lär vi oss

- vad pixlar är
- hur pixlarna sitter på en skärm
- hur man ritar prickar
- hur man gör slumpmässiga saker

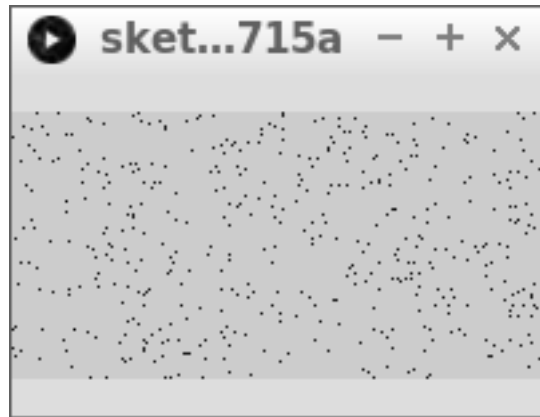


Figure 26: Slutuppgift

## point och random: intro

Pixlar är rutorna som utgör din skärm.



Pixel = en ruta på skärmen

---

Ju fler pixlar skärmen har desto skarpare ser bilden ut. Du kan se det bra med gamla spel: de har färre pixlar:



Figure 27: Super Mario Bros 1

## point och random: uppgift 1

Kör följande kod:

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

void draw()
{
  point(150, 100);
}
```



---

```
point(150,
      100);
point(150,
      100);
```

---

‘Kära dator, rita en prick på pixeln som är 150 pixlar till höger och 100 pixlar nedåt’

‘Kära dator, rita en prick på koordinat (150, 100)’

---

## point och random: lösning 1



Figure 28: point och random: lösning 1

## point och random: uppgift 2

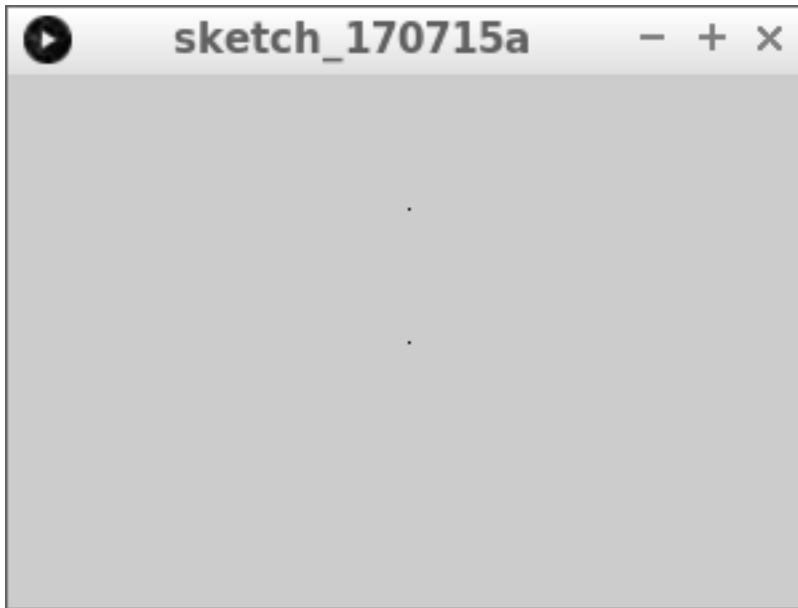


Figure 29: point och random: uppgift 2

Rita en andra prick mellan den första och den övre delen av fönstret.

## point och random: lösning 2

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

void draw()
{
  point(150, 100);
  point(150, 50);
}
```

## point och random: uppgift 3

Den första pixeln är exakt i mitten. Med andra ord, halva bredden på fönstret och på halva skärmens höjd. Ändra `point(150,100);` till något med `width` och `height`.

## point och random: lösning 3

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

void draw()
{
  point(width / 2, height / 2);
  point(150, 50);
}
```



---

`width / 2`    'Kära dator, ange här bredden på fönstret, dividerat med 2'

---

### point och random: uppgift 4

Den andra pixeln är

- på halva fönstret
- på en fjärdedel av skärmens höjd

Ändra `point(150, 50)`; till något med `width` och `height`.

### point och random: lösning 4

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

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
}
```



---

`höjd / 4`    'Kära dator, ange här fönstrets höjd, dividerat med 4'

---

### point och random: uppgift 5

Rita en ny pixel i skärmens övre vänstra hörn.

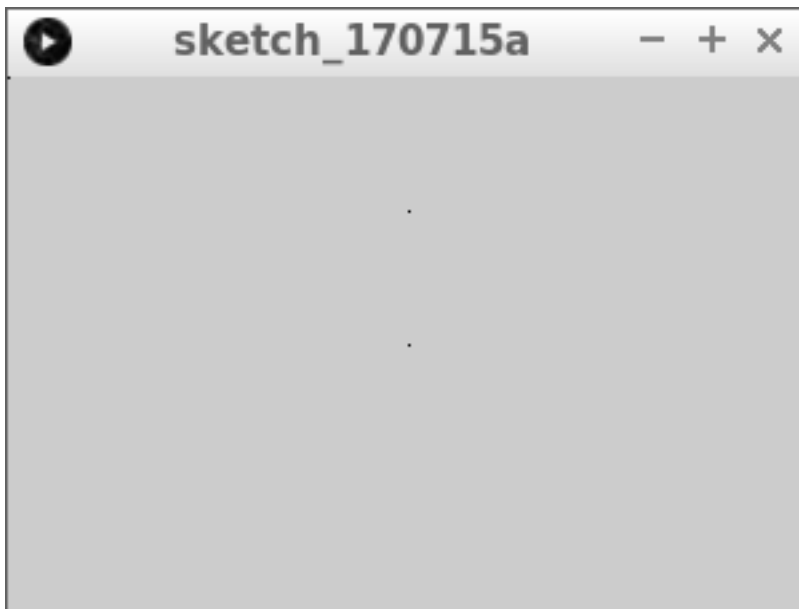


Figure 30: point och random: uppgift 5

### point och random: lösning 5

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

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
  point(0, 0);
}
```



---

point(0,0);	‘Kära dator, rita en prick i det övre vänstra hörnet’
point(0,0);	‘Kära dator, rita en prick på koordinat (0, 0)’

---

### point och random: uppgift 6

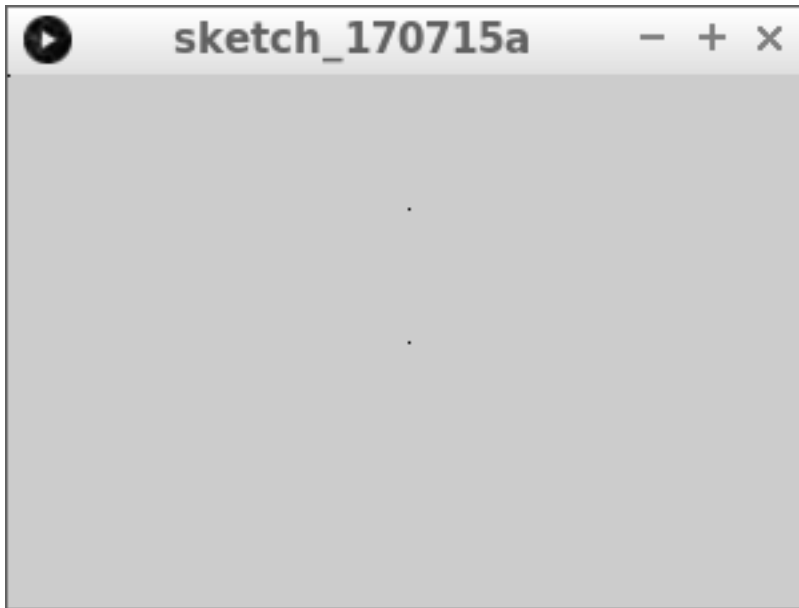


Figure 31: point och random: uppgift 6

Rita en ny pixel, längst upp till höger på skärmen. Använd `width - 1` 'som det första talet inom parentes förpoint'.

### point och random: lösning 6

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

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
  point(0, 0);
  point(width - 1, 0);
}
```

### point och random: uppgift 7

Rita två pixlar i de nedre två hörnen. Använd `width - 1` och `height - 1` på rätt ställen.



Figure 32: point och random: uppgift 7

### point och random: lösning 7

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

void draw()
{
  point(width / 2, height / 2);
  point(width / 2, height / 4);
  point(0, 0);
  point(width - 1, 0);
  point(0, height - 1);
  point(width - 1, height - 1);
}
```

### point och random: uppgift 8

Kör den här koden:

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

void draw()
{
  point(random(300), 100);
}
```

Vad ser du?



## point och random: lösning 8

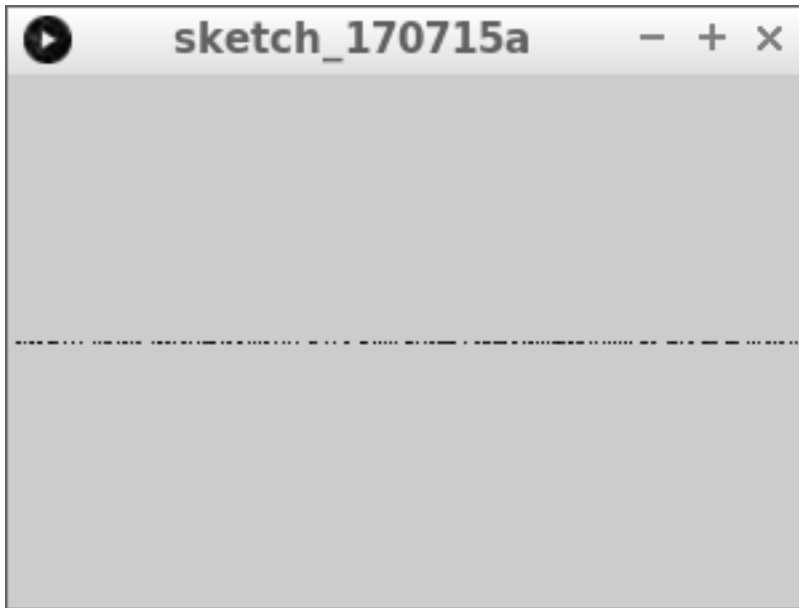


Figure 33: point och random: lösning 8

Du ser att prickar dras på slumpmässiga platser, men alltid på samma höjd.



---

`random(300)` 'Kära dator, välj ett slumpmässigt tal från noll till 300'

---

## point och random: uppgift 9



Figure 34: point och random: uppgift 9

```
void setup()
{
  size(400, 100);
}

void draw()
{
```

```
point(random(width), height / 2);  
}
```

point och random: slutuppgift

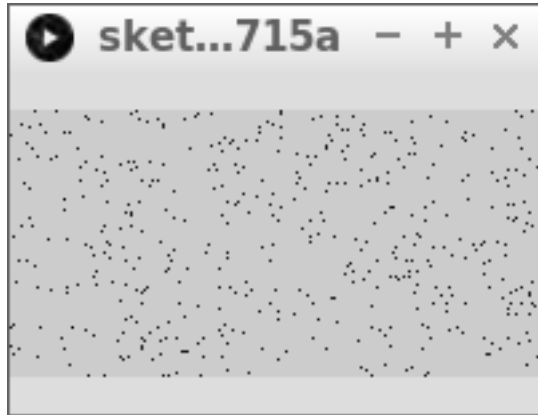


Figure 35: Slutuppgift

Låt datorn rita prickar slumpmässigt genom fönstret.