Processing

Bok 1

Figure 1: Bok 1

#	Beskriving
1	Ett vackert program
2	Låt bollen åka åt höger
3	`width` och `height`
4	`point` och `random`

Contents

Förord

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

Om den här boken

Figure 1: Licensen för denna bok

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.

Ett vackert program

Processing är ett programmeringsspråk som har utvecklats för designers och är mycket lämpligt för att göra spel och vackra saker.

Under den här lektionen lär vi oss

- hur man kopierar kod till Processing
- hur man startar programmet

Så här ser programmet ut:

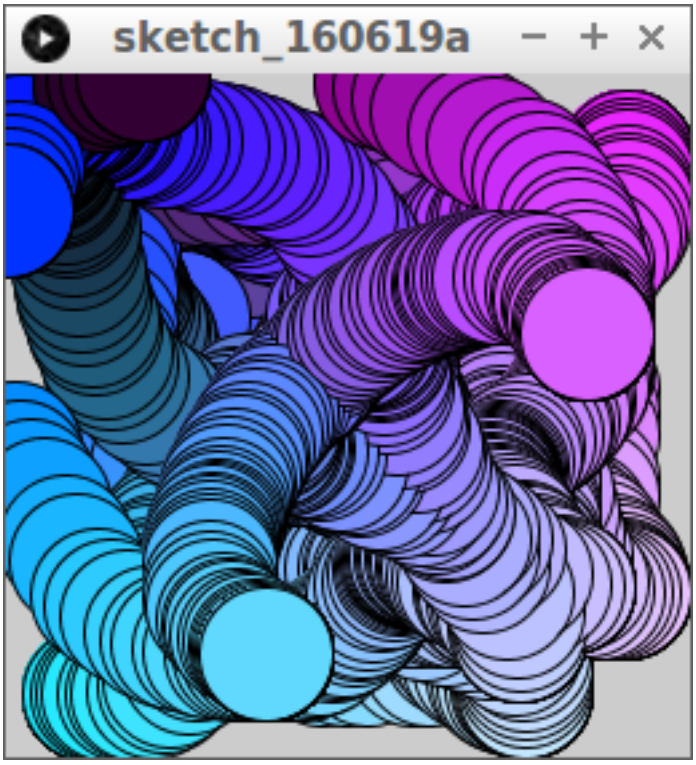

Figure 2: Ett vackert program

Ett trevligt program: intro

När du öppnar programmet ser du ett tomt program utan kod:

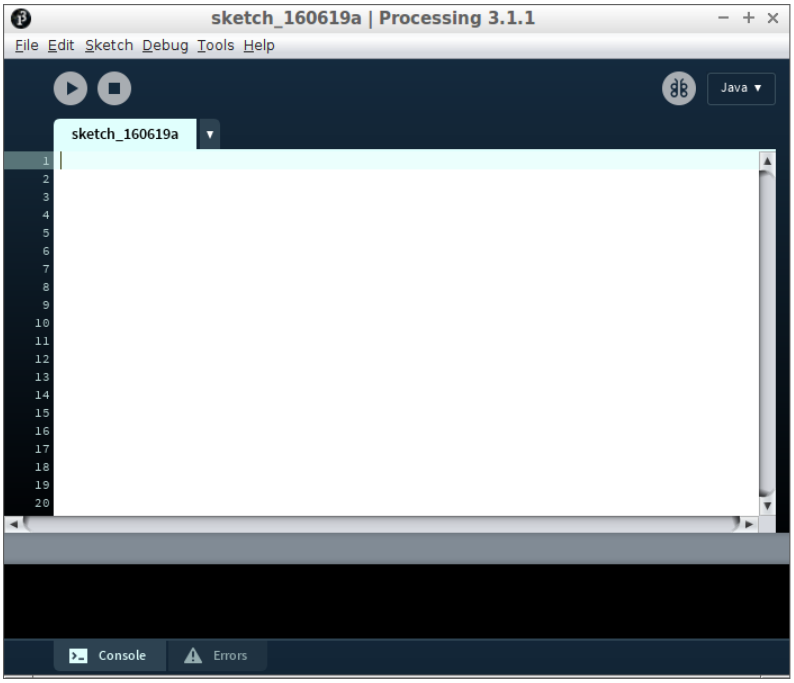

Figure 3: Processing utan kod

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. Just nu räcker det att veta att den gör något vackert.

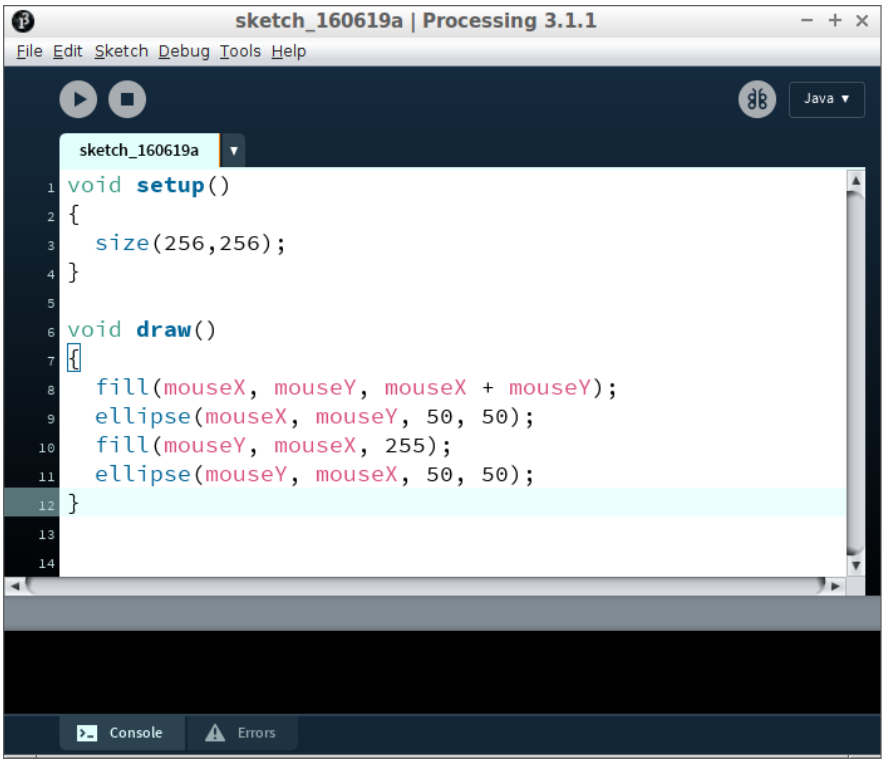

Figure 4: Processing med kod

Ett trevligt program: slutuppgift

- Skriv av koden, precis som den ser ut, i programmet
- Kör den här koden genom att klicka på knappen “Run”

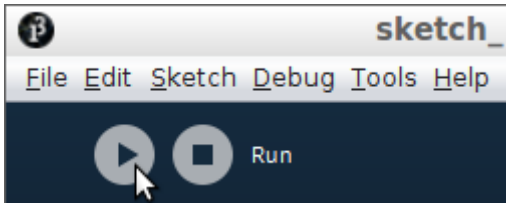

Figure 5: Run knappen

Gick det bra? Visa det för en vuxen så får du ett klistermärke!

Låt bollen åka åt höger

Under den här lektionen ska vi låta en boll åka åt höger.

Du kommer också att lära dig vad en variabel är. Man kan knappt programmera utan variabler.

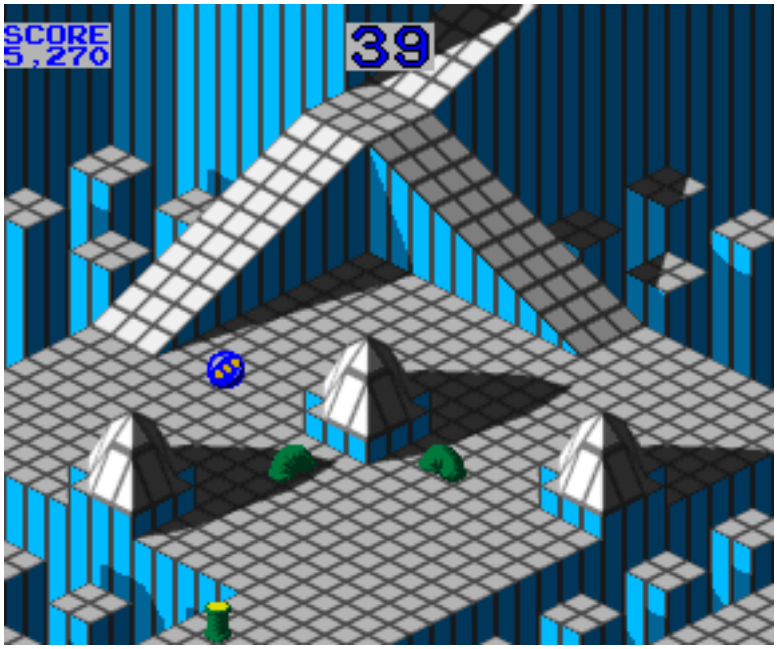

Figure 6: Marble Madness

Låt bollen åka åt höger: intro

Skriv 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 stavat fel någonstans. Titta noga på koden och rätta dina stavfel.

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

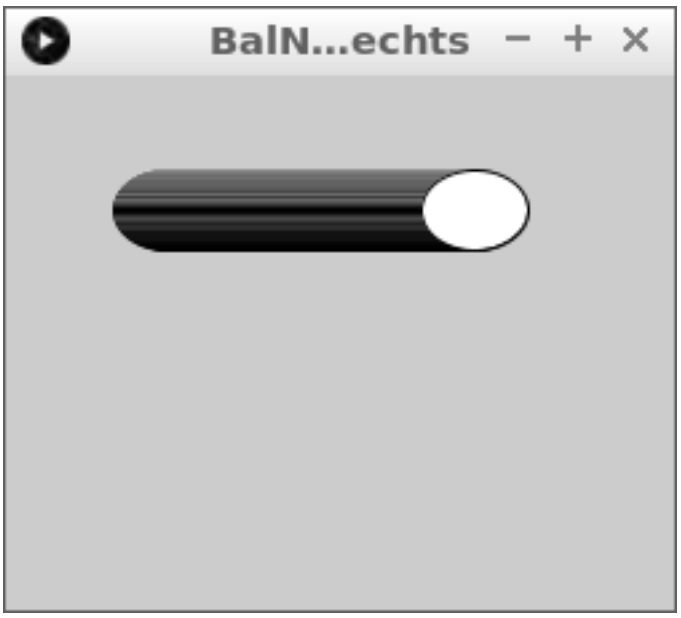

Figure 7: Låt bollen åka åt höger: intro

Låt bollen åka åt höger: uppgift 1

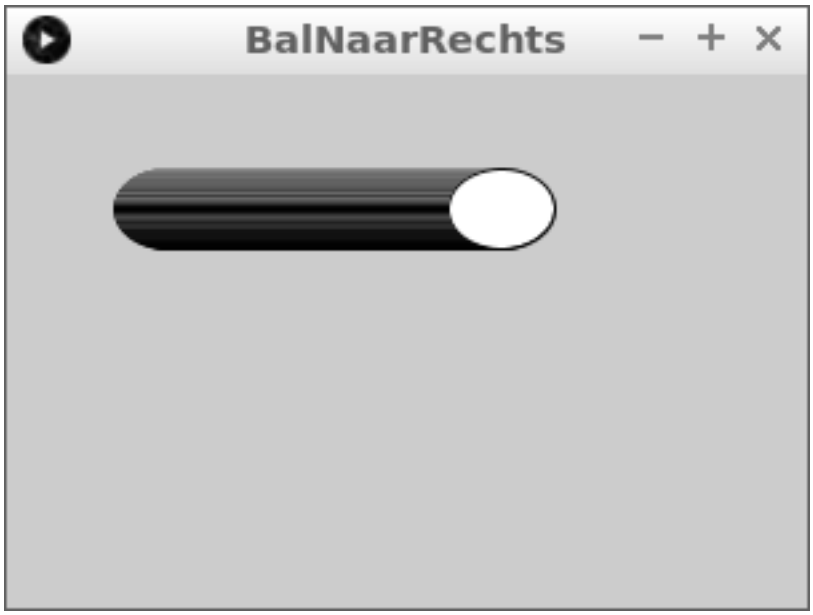

Figure 8: Låt bollen åka åt höger: uppgift 1

Fönstret är nu `250` pixlar brett. Nu ska du göra det `300` pixlar brett.

Ändra koden och tryck på “Run”.

Låt bollen åka åt höger: lösning 1

Det står `250` i koden på ett ställe. Bara 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 som är 300 pixlar brett och 200 pixlar högt.’

Låt bollen åka åt höger: uppgift 2

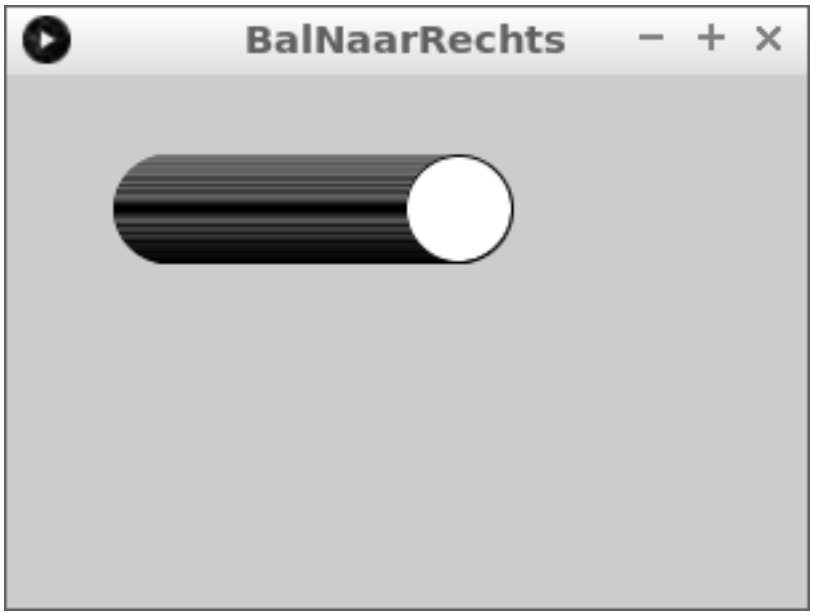

Figure 9: Låt bollen åka åt höger: uppgift 2

Bollen är nu äggformad: den är `40` pixlar bred och `30` pixlar hög. Nu ska du göra bollen rund: `40` pixlar bred och `40` pixlar hög.

Låt bollen åka åt höger: lösning 2

`ellips (x, 50, 40, 30);` ritar ut bollen. `40, 30` gör bollen äggformad. Att ändra koden 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 ut en oval som ligger `x` pixlar till höger och `50` pixlar nedåt, och är `40` pixlar bred och `30` pixlar hög.’

Låt bollen åka åt höger: uppgift 3

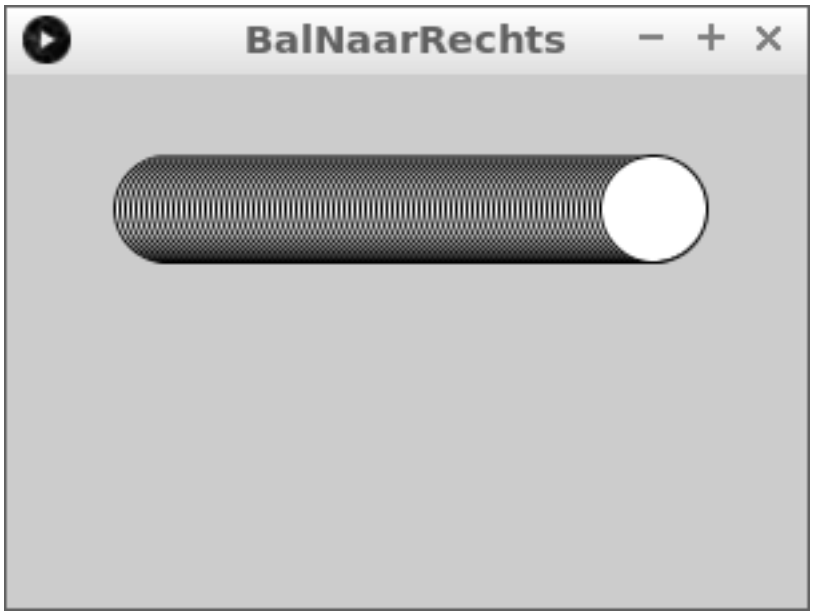

Figure 10: Låt bollen åka åt höger: uppgift 3

Bollen rör sig nu åt höger med en hastighet på `1` pixel i taget. Låt bollen röra sig till höger dubbelt så snabbt

Låt bollen åka åt höger: lösning 3

`x = x + 1;` flyttar bollen 1 pixel. 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, öka `x` med 1.’
`x += 1;`	‘Kära dator, öka `x` med 1.’
`x++;`	‘Kära dator, öka `x`.’
`++x;`	‘Kära dator, öka `x`.’

Låt bollen åka åt höger: uppgift 4

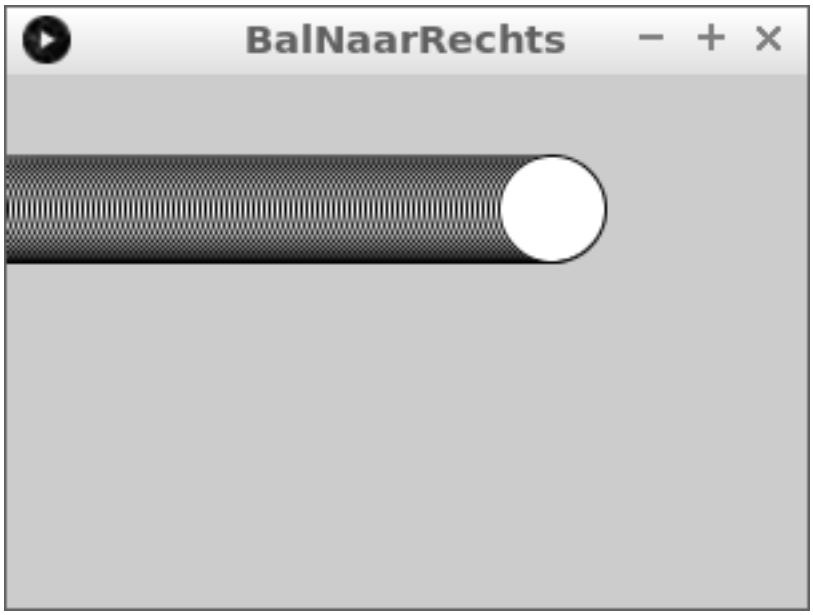

Figure 11: Låt bollen åka åt höger: uppgift 4

I början är bollens mitt `60` pixlar till höger. Kan du få cirkeln att ritas ut `0` pixlar till höger?

Låt bollen åka åt höger: lösning 4

`float x = 60;` bestämmer mitten på bollen. 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 måsvingarna.’

Låt bollen åka åt höger: uppgift 5

Haha, den här lektionen kallas ‘Låt bollen åka åt höger’, men nu ska vi också göra en boll som åker åt vänster!

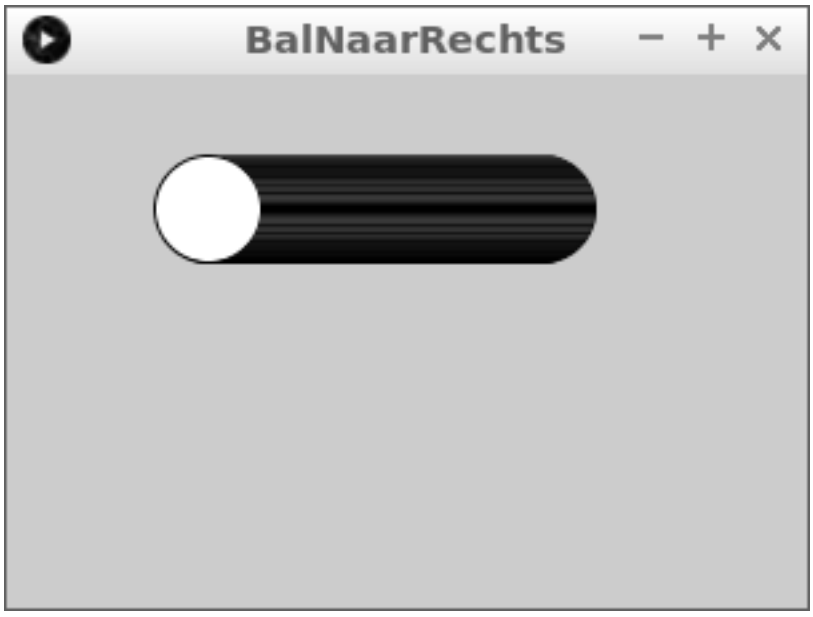

Figure 12: Låt bollen åka åt höger: uppgift 5

Låt nu bollen börja på höger sida av fönstret och åka åt vänster.

Låt bollen åka åt höger: lösning 5

För att få bollen att starta på höger sida måste du använda `float x = 500;` (eller något annat högt tal). För att få bollen att åka 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 måsvingarna hela tiden.’

Låt bollen åka åt höger: vad är en variabel?

På den första raden använder vi en variabel:

```
float x = 50;
```

I klartext betyder det: “Kära dator, kom ihåg talet `x` med ett startvärde på `50`.”.

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

En variabel är en plats i datorminnet 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 om din ålder vet du vilket nummer du ska svara.

	
`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 beskriver den hur långt till höger cirkeln är. Ordet `float` betyder att `x` är ett (decimal) tal. Symbolen `=` betyder 'ska nu vara'. Talet `50` är det startvärdet. Semikolon (`;`) anger slutet på en mening (som punkten i en svensk text).

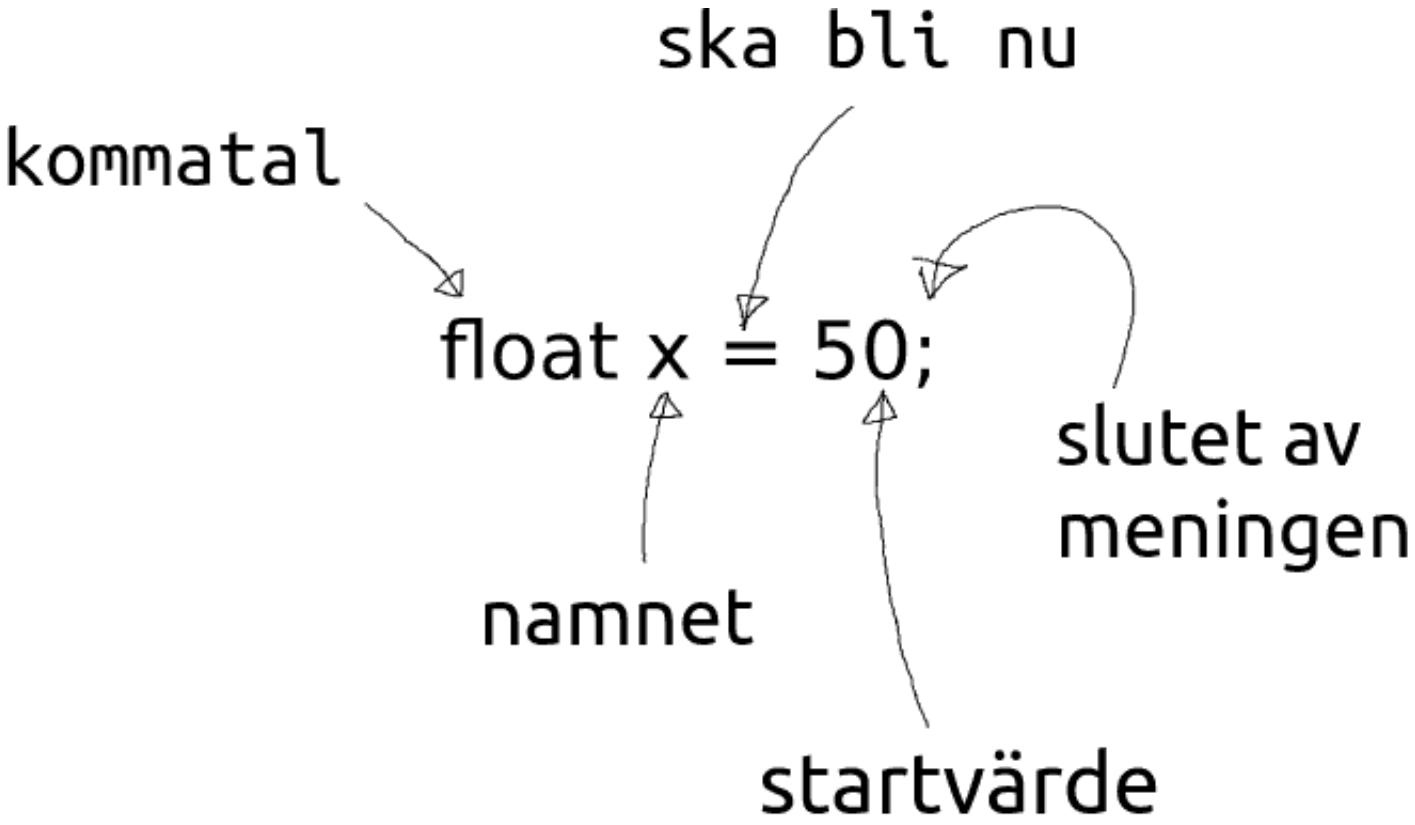

Figure 13: Förklaring av `float x = 50;`

	
`float`	'ett tal'
`=`	'ska nu vara'
`;`	'slutet av meningen'

Låt bollen åka åt höger: uppgift 6

Haha, den här lektionen kallas “Låt bollen åka åt höger”, men vi kommer också att få en boll att åka nedåt!

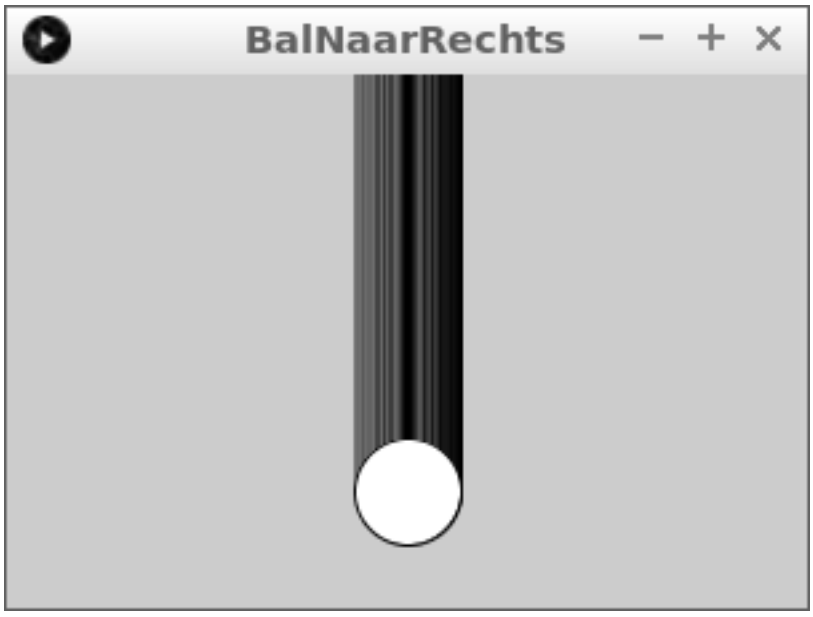

Figure 14: Låt bollen åka åt höger: uppgift 6

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

Låt bollen åka åt höger: lösning 6

```
float y = 0;

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

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

Låt bollen åka åt höger: uppgift 7

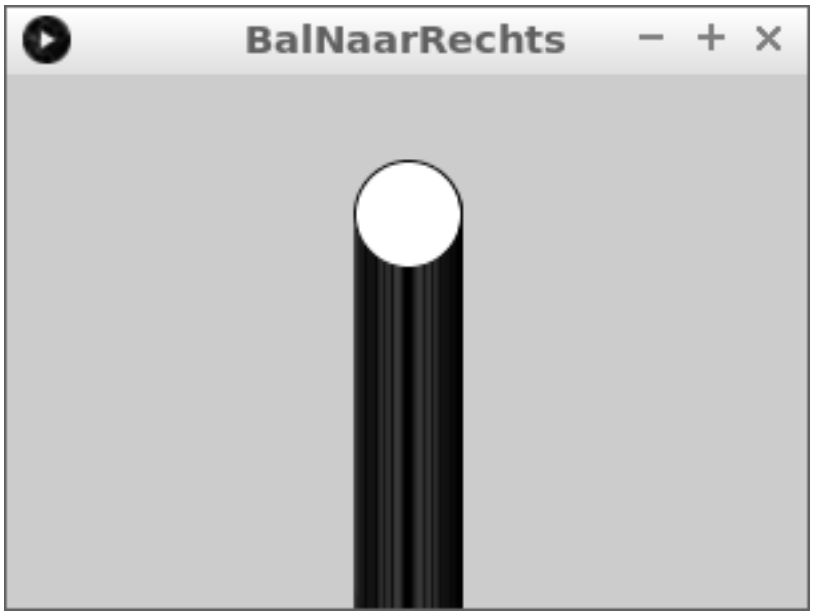

Figure 15: Låt bollen åka åt höger: uppgift 7

Nu ska vi få bollen att röra sig snabbare och uppåt

- Rita ut en boll längst ner på skärmen
- Bollen måste åka uppåt i en rak linje
- Bollen måste åka dubbelt så snabbt

Låt bollen åka åt höger: lösning 7

```
float y = 200;

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

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

Låt bollen åka åt höger: slutuppgift

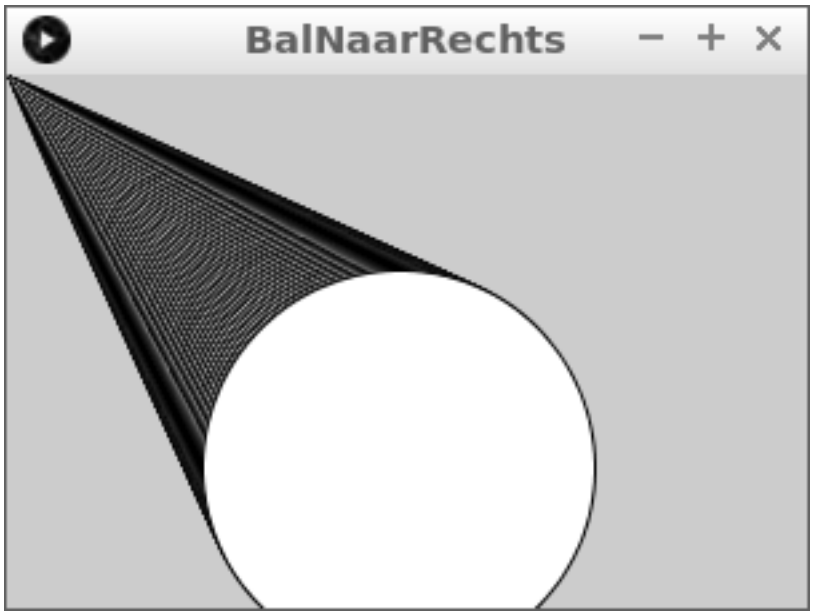

Figure 16: Låt bollen åka åt höger: slutuppgift

- bollen måste åka diagonalt åt höger och neråt samtidigt
- bollen måste bli större, det vill säga öka i bredd och höjd
- se även figur `slutuppgift 'Låt bollen åka åt höger'`

`width` och `height`

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

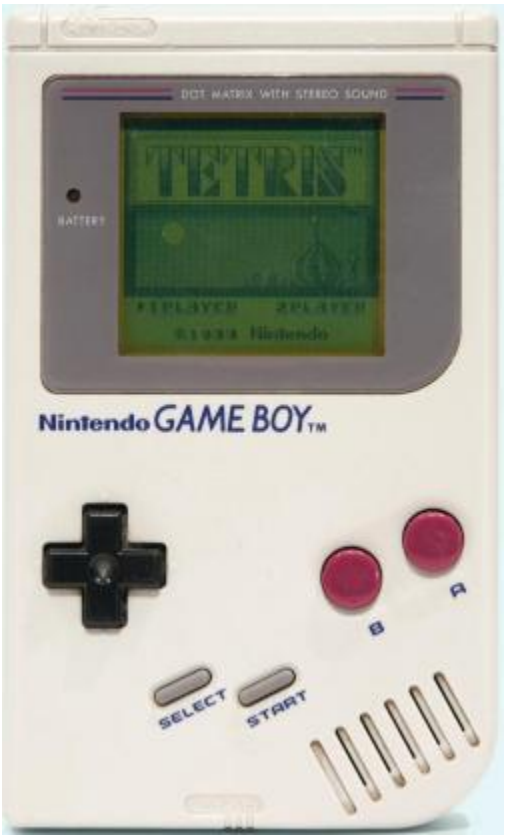

Figure 17: Gameboy har en skärm på 160 x 144 pixlar

`width` och `height`: intro

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

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

	
`size(800, 400);`	‘Kära dator, gör ett fönster `800` pixlar brett och `400` pixlar högt.’
`ellips(60,50,40,30);`	‘Kära dator, rita ut en oval `60` pixlar till höger, `50` pixlar nedåt, som är `40` pixlar bred och `30` pixlar hög’

Skriv in koden ovan och kör den.

`width` och `height`: uppgift 1

Gör nu fönstret `128` pixlar brett och `128` pixlar högt.

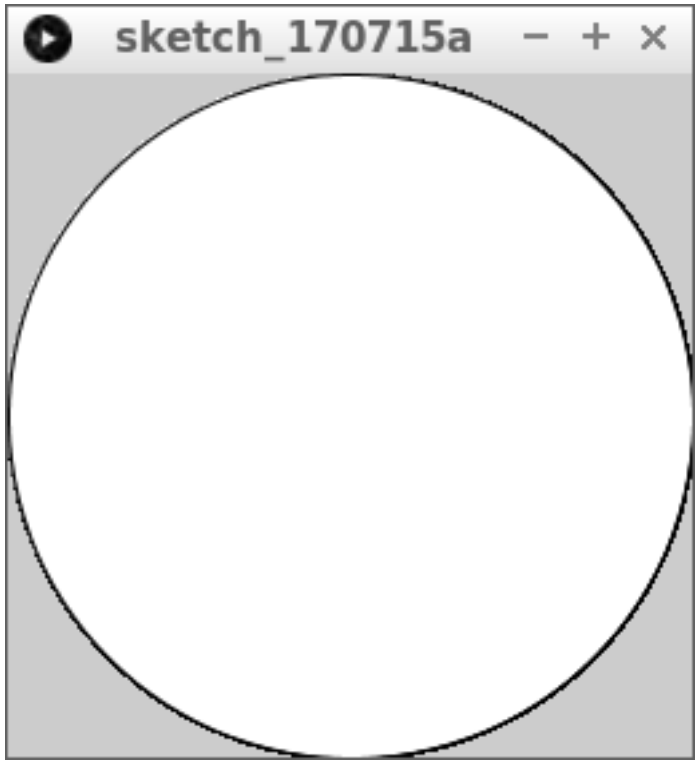

Figure 18: `width` och `height`: intro

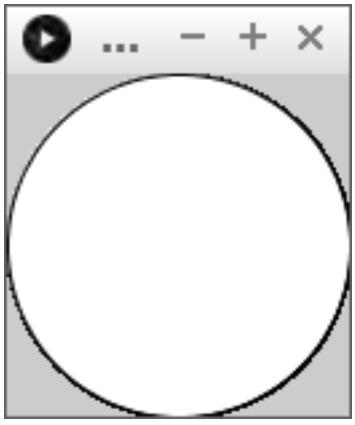

Figure 19: `width` och `height`: uppgift 1

`width` och `height`: lösning 1

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

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

`width` och `height`

`width` och `height` är inbyggda i Processing, `width` betyder fönsterbredd och `height` betyder fönsterhöjd. `width` och `height` är viktiga, så att ditt program fortfarande ser bra ut när du ändrar storlek på skärmen.

Nu funkar våra program bara för ett fönster av en viss storlek. Varje gång du väljer en ny fönsterstorlek måste du skriva in mycket kod igen!

Om vi vet fönstrets bredd och höjd vet vi också hur bred och hög vår oval måste bli:

- ovalens x-koordinat (hur många pixlar till höger) är halva bredden
- ovalens y-koordinat (hur många pixlar nedåt) är halva höjden
- ovalens bredd är fönstrets bredd
- ovalens höjd är fönstrets höjd

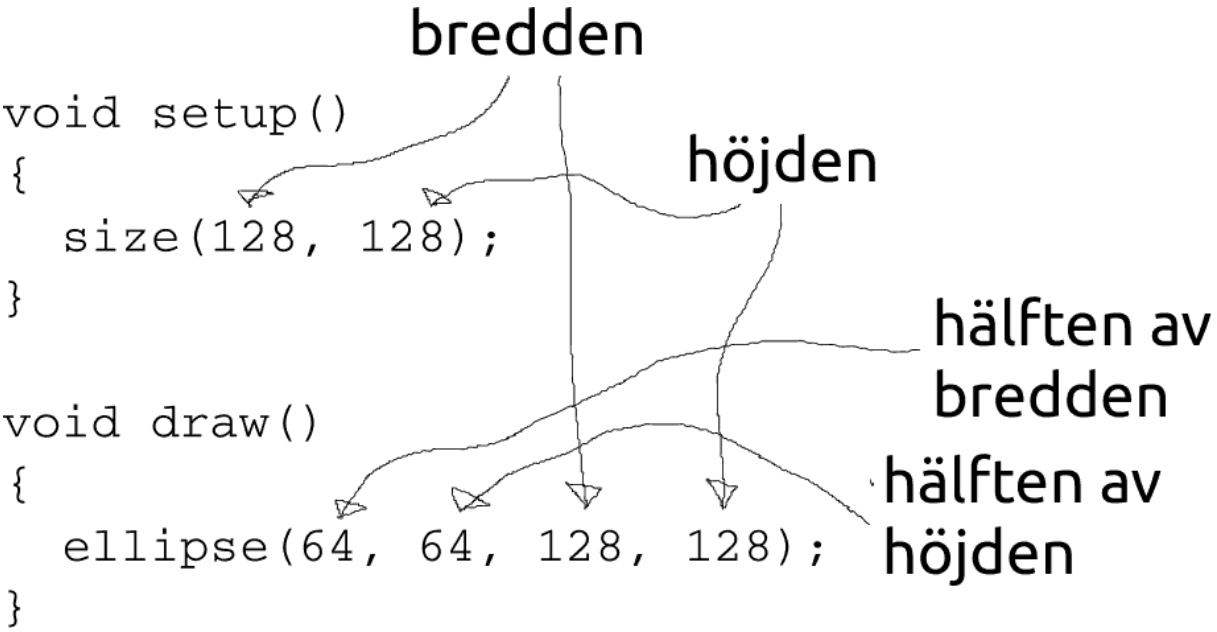

Figure 20: Vad du vill säga

Processing känner till fönstrets bredd och höjd: Fönstrets bredd kallas `width` och höjden kallas `height`

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

Dessa ord använder du om du vill veta storleken på ditt fönster.

`width` och `height`: uppgift 2

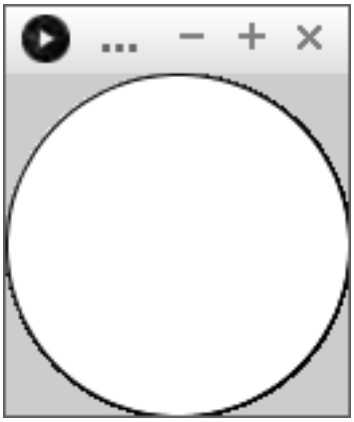

Figure 21: `width` och `height`: uppgift 2

Skapa ett program som ritar en oval som fyller hela fönstret:

- 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`

	
`/`	‘delat med’

`width` och `height`: lösning 2

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

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

`width` och `height`: uppgift 3

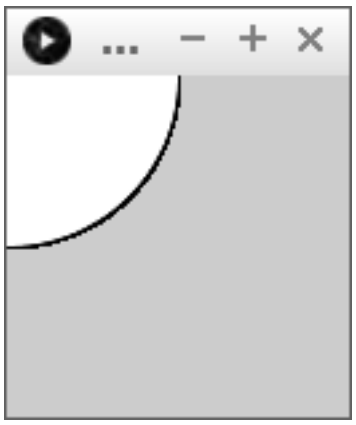

Figure 22: `width` och `height`: uppgift 3

Rita ut cirkelns mitt på platsen (koordinat) `(0, 0)` i fönstret.

`width` och `height`: lösning 3

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

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

`width` och `height`: uppgift 4

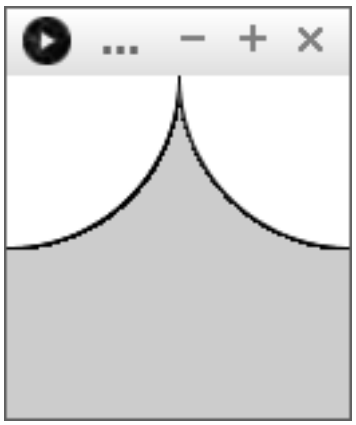

Figure 23: `width` och `height`: uppgift 4

Skapa en till cirkel vars mittpunkt ligger 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

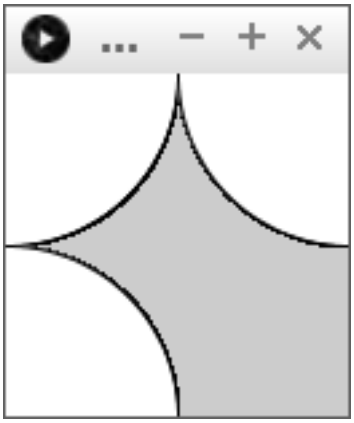

Figure 24: `width` en `height`: uppgift 5

Rita ut en tredje cirkel vars mittpunkt ligger 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`: slutuppgift

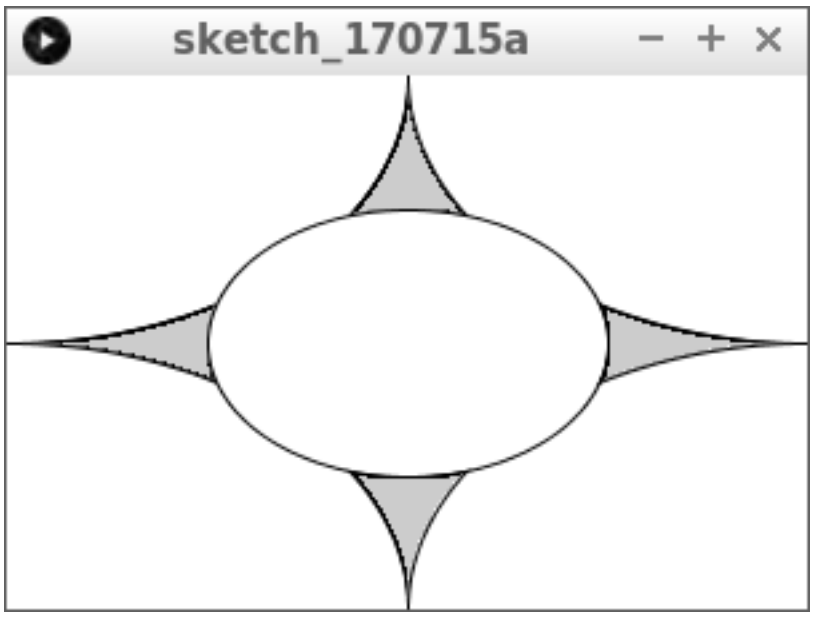

Figure 25: `width` och `height`: slutuppgift

- Gör fönstret 300 pixlar brett och 200 pixlar högt
- Gör en fjärde cirkel vars mittpunkt ligger i nedre högra hörnet
- Gör en femte cirkel vars mittpunkt ligger i mitten och är hälften så stor
- Använd `width` och/eller `height` (ingen `100`, `150`,`200` eller `300`!)

`point` och `random`

Under den här lektionen lär vi oss

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

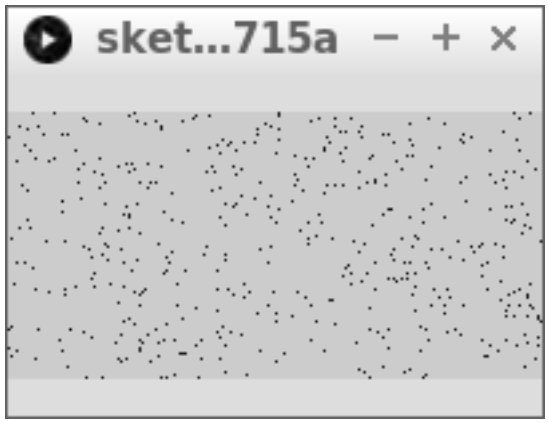

Figure 26: Slutuppgift

`point` och `random`: intro

Din skärm har många rutor som består av pixlar.

Pixel = en ruta på skärmen

Ju fler pixlar skärmen har desto skarpare blir bilderna. Du kan se det på gamla/retro datorspel: de har färre pirxlar vilket gör bilderna kantigare.

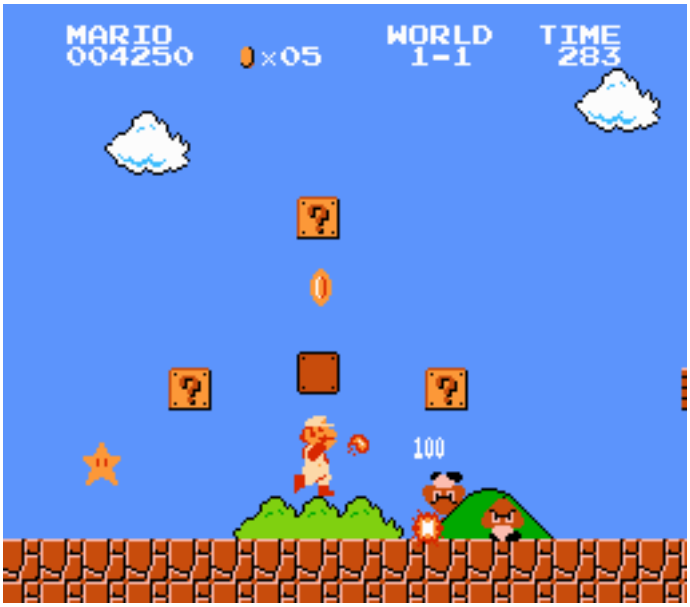

Figure 27: Super Mario Bros 1

`point` och `random`: uppgift 1

Kör följande kod:

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

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

`point(150, 100);`	‘Kära dator, rita ut en punkt i fönstret på platsen `150` pixlar till höger och `100` pixlar nedåt’
`point(150, 100);`	‘Kära dator, rita ut en punkt 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 ut en till punkt mellan den första punkten och ovansidan 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 punkten är exakt i mitten. Med andra ord, på halva bredden och på halva höjden av fönstret. 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, delat med 2’

`point` och `random`: uppgift 4

Den andra pixeln är utritad

- på halva fönstrets bredd
- på en fjärdedel av fönstrets 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);
}
```

	
`height / 4`	‘Kära dator, ange här fönstrets höjd, delat med 4’

`point` och `random`: uppgift 5

Figure 30: `point` och `random`: uppgift 5

Rita ut en ny pixel i fönstrets övre vänstra hörn.

`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 ut en punkt i det övre vänstra hörnet'
`point(0,0);`	'Kära dator, rita ut en punkt på koordinat `(0, 0)`'

`point` och `random`: uppgift 6

Figure 31: `point` och `random`: uppgift 6

Rita ut en ny pixel, längst upp till höger på fönstret. Använd `width - 1` som det första talet inom parentesarna för `point`.

`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

Figure 32: `point` och `random`: uppgift 7

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

`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 punkter ritas ut på slumpmässiga platser, men alltid på samma höjd.

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

`point` och `random`: slutuppgift

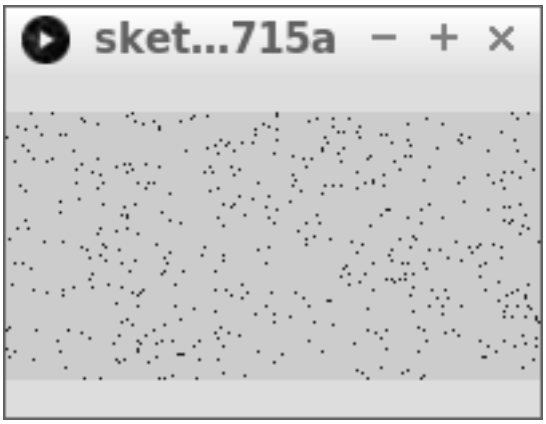

Figure 34: Slutuppgift

Låt datorn rita ut punkter slumpmässigt över hela fönstret.