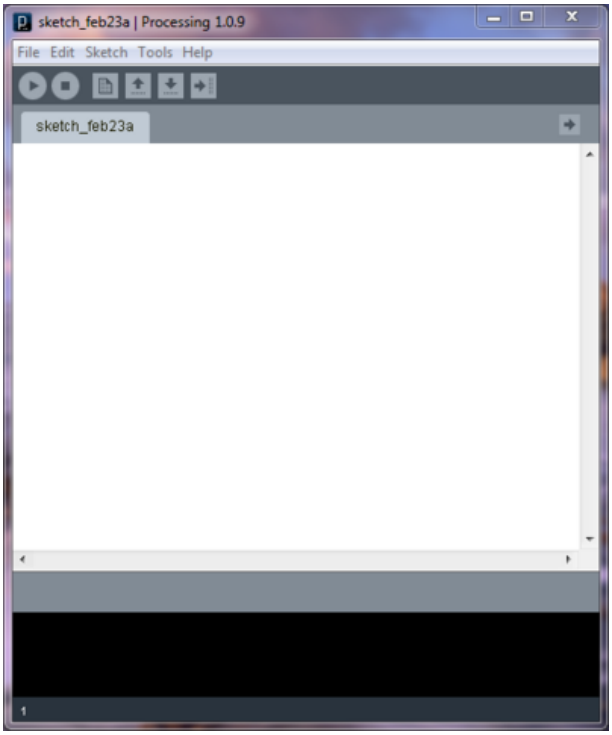
Kodetime for Nordstrand barneskole

av Veronika Heimsbakk og Lars Erik Realfsen

1 Hva er Processing?

Processing er et *programmeringsspråk* som er gratis, og tilgjengelig for alle! Man kan programmere i Processing sitt eget skriveprogram, som du kan se bilde av nedenfor.



Dette programmet kan lastes ned gratis på `processing.org`. Programmet er ganske likt andre skriveprogram. Men det er en spesiell knapp som er verdt å legge merke til. Det er «play»-knappen oppe til venstre. Denne vil fortelle deg om koden din er *lovlig*, og hvis den er det, så vil den *kjøre* programmet ditt.

2 Ditt første Processing-program

Nå skal vi se på litt *programmeringskode*. En slik kode er skrevet i et programmeringsspråk, og et slikt språk er et språk som datamaskinen kan forstå! Med slike språk kan vi fortelle datamaskinen nøyaktig hva **vi** vil at den skal gjøre. Magisk, ikke sant? Nå skal vi være trollmenn og trollkvinner!

Når du har åpnet Processing, så kan du prøve å skrive denne linjen:

```
ellipse(50,50,80,80);
```

Her forteller vi datamaskinen at den skal «tegne en ellipse (sirkel) med sitt senter 50 piksler fra venstre, og 50 piksler fra toppen. Med en bredde 80 piksler, og en høyde på 80 piksler.» Trykk på play-knappen, og du har skrevet ditt første program!

Grunnen til at `ellipse` er skrevet med blå skrift, er fordi det er et *nøkkelord*.

Oppgave Hvordan skal du tegne en firkant tror du? *Hint: nøkkelordet er forkortet.*

3 Ditt andre Processing-program (med litt mer action)

Nå skal vi programmere hva som skal skje på skjermen når vi trykker på datamusen! Se på denne koden:

```
// Inne i setup sine parenteser { ... } skriver vi alt
// som skal skje en gang.
void setup() {
  size(450, 450);
}

// Inne i draw sine parenteser skriver vi det vi vil skal
// skje hele tiden!
void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}

// Med slike // skriver man forresten kommentarer. Disse
// leses ikke av datamaskinen, her kan vi skrive hva vi
// vil!
// Gulrotkake er godt! Namnamnam.
```

Her var det litt mer kode enn på forrige side. Klarer du å tyde hva som skjer her?

void setup()

Først skriver vi `void setup()`, dette kalles en *metode*. Hva som skal skje i denne metoden skriver vi inne i slike krøllparenteser. I `void setup()` vil vi at `size` skal være `(450, 450)`. Dette er størrelsen på vinduet vi skal tegne i, det skal være 450 piksler i bredde og høyde.

void draw()

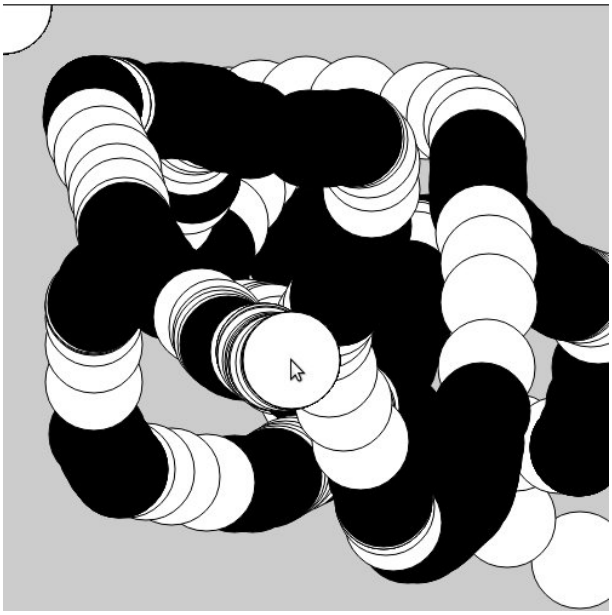
Neste metode er `void draw()`. Dette er, som navnet sier, noe som skal *tegnes* hele tiden! Se på det som en *uendelig løkke*. Den avsluttes bare når vi krysser bort vinduet vårt.

if og else

Inne i `void draw()` så kommer det noe som heter `if(...)` og `else`. På norsk er dette *hvis* og *ellers*, og det er akkurat det det betyr!

Så her sier vi at «hvis `mousePressed` (musen blir trykket på), så skal vi fylle noe med 0». Og hva er noe? Jo, ellipsen vi tegner til slutt! Og «ellers skal vi fylle ellipsen med 255.» 0 og 255 er farger, hvor 0 er sort og 255 er hvit.

Prøv selv, og se hva som skjer! Min ser slik ut:



Farger

Fargene i Processing er i RGB (Rød, Grønn, Blå). Alle dataskjermer viser farger i RGB. Hvis du skriver `fill(255, 0, 0)` vil du få en rød ellipse! For her har vi høyeste verdi på den røde plassen.

	R	G	B
	255	0	0
	0	255	0
	0	0	255

Oppgave Hvordan kan vi få til gul tror dere? *Hint: Rød er 255 og blå er 0, hva er grønn?*

Oppgave Prøv å fylle sirkelen med en annen farge enn sort eller hvit.

4 Former og figurer

point()

Denne metoden tegner et punkt med en dimensjon på én piksel. Man kan skrive:

```
point(30, 20);
```

Da vil Processing tegne en liten prikk 30 piksler fra venstre og 20 piksler ned fra toppen. Du kan se på vinduet som et *koordinatsystem*, og metoden `point(x, y)` vil ha x-koordinaten og y-koordinaten for å vite hvor den skal tegne prikken.

line()

Metoden `line(x1, y1, x2, y2)` vil tegne ei linje mellom to punkter. Hvor `x1` er x-koordinaten til første punkt, `y1` er y-koordinaten til første punkt, `x2` er x-koordinaten til andre punkt og `y2` er y-koordinaten til andre punkt. Hvis vi skriver:

```
line(0, 10, 200, 10);
```

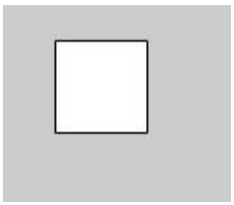
Så vil Processing tegne ei linje som starer i punktet x = 0 og y = 10, og slutte i punktet x = 200 og y = 10. Det vil se slik ut:

rect()

Med denne metoden kan du tegne firkanter! Metoden skrives som `rect(a, b, c, d)`. Hvor `a` og `b` er x- og y-koordinatene til rektangelet (hvor det skal plasseres på vinduet). Og `c` og `d` er bredde og høyde på rektangelet.

```
rect(30, 20, 50, 50);
```

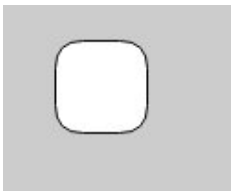
Den lille kodesnutten over vil gi oss dette:



Vi kan også lage rektangler med runde hjørner. Da legger vi på et ekstra tall (en ekstra *parameter*) til metoden. `line(a, b, c, d, r)`, hvor `r` er radiusen til alle hjørnene.

```
rect(30, 20, 50, 50, 15);
```

Kodesnutten over vil da gi oss:



ellipse()

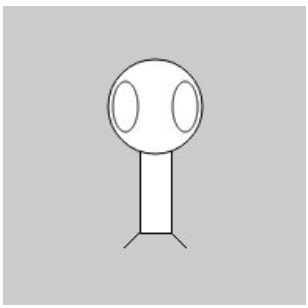
Og så har vi `ellipse(a, b, c, d)`, som vi kjenner fra før! Og som vi vet så er `a` og `b` x- og y-koordinatene for hvor vi vil plassere sirkelen. `c` og `d` er bredde og høyde til sirkelen.

Lage en figur

Når vi kjenner alle disse metodene for å tegne former, så kan vi sette de sammen og tegne en figur!

```
size(200,200);
rectMode(CENTER);
rect(100,100,20,100);
ellipse(100,70,60,60);
ellipse(81,70,16,32);
ellipse(119,70,16,32);
line(90,150,80,160);
line(110,150,120,160);
```

Ta en titt på koden over. Her tegner vi først vinduet vårt med dimensjonene 200 piksler $\times$ 200 piksler. Så kommer et nytt nøkkelord `rectMode(CENTER)`, dette forteller Processing at rektangelet skal stå i midten på vinduet. Videre tegner vi et rektangler, noen sirkler og noen streker. Hvordan ser dette ut til slutt lurer du? Joda, slik ser dette ut:



5 Datatyper

For å lagre informasjon i Processing, så bruker man noe som heter *variabler*. Dette er en liten sak som *lagrer* en verdi, eller tekst, for deg. Disse variablene må være av forskjellige *datatyper*. Her er en oversikt over datatypene:

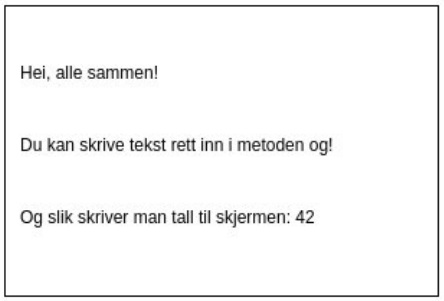
Nøkkelord	Datatype	Eksempel
String	tekst	`String navn = "Veronika";`
int	heltall	`int alder = 23;`
float	desimaltall	`float pi = 3.14;`
boolean	sant/usant	`boolean sant = true;`

6 Tekst

Hvis du vil printe en tekst til vinduet ditt, så bruker man metoden `println(...)`. Slik som i denne koden:

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

void draw() {
  background(255);
  fill(0);

  String tekst = "Hei, alle sammen!";
  text(tekst, 10, 50);

  text("Du kan skrive tekst rett inn i metoden og!", 10,
     100);

  int mittTall = 42;
  text("Og slik skriver man tall til skjermen: " +
     mittTall, 10, 150);
}
```

Dette vil gi oss:



7 Løkker

Løkker er svært nyttig i programmering. Disse forteller oss at noe skal skje flere ganger, eller at noe skal skje hvis noe annet er oppfyllt. La oss se på noen eksempler.

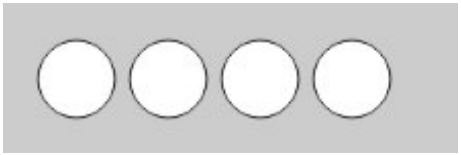
while

Den enkleste løkken er `while(test) {...}`, hvor `test` skal være noe som enten er sant eller usant. Se på koden nedenfor.

```
int i = 1;

while (i < 5) {
  ellipse(i*60, height/2, 50, 50);
  i = i + 1;
}
```

Her forteller vi Processing at «*så lenge i er mindre enn 5*, så skal vi tegne en ellipse», hvor da denne ellipsen plasseres på `i*60, height/2`. Altså verdien av i ganget med 60 fra venstre og høyden av vinduet delt på 2 fra toppen. Ellipsen skal også være 50 piksler bred, og 50 piksler høy.

Til slutt oppdaterer vi verdien i `int i`, ved å skrive `i = i + 1`. Da har vi økt `i` med 1.



for

En annen type løkke kalles `for`-løkker. De ser slik ut:

```
for (int i = 1; i < 5; i++) {
  ellipse(i*60, height/2, 50, 50);
}
```

Her får vi akkurat det samme resultatet som koden med `while`-løkken. En `for`-løkke er konstruert på formen `for(start; test; oppdatering) {...}`

8 Bevegelse

Du husker kanskje at metoden `draw() {...}` ble tegnet hele tiden mens programmet vårt kjører. Dette kan vi bruke til å få figurene våre til å røre på seg. Her kommer et eksempel.

```
int diameter = 300;

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

void draw() {
  background(0);
  diameter = diameter - 1;
  ellipse(width/2, height/2, diameter, diameter);
}
```

Prøv ut denne koden og se hva som skjer!

2D rotasjon

Nå skal vi flytte på to firkanter i 2D med musepilen. Prøv ut denne koden:

```
void setup() {
  size(600,600);
  rectMode(CENTER); // Plasserer firkantene i sentrum
}

void draw() {
  background(0);
  rect(mouseX, height/2, mouseY/2+10, mouseY/2+10);
  fill(255, 204);
  int inverserX = width-mouseX;
  int inverserY = height-mouseY;
  rect(inverserX, height/2, (inverserY/2)+10, (inverserY
     /2)+10);
}
```

Gøy, ikke sant?

Oppgave Endre koden over slik at du får sirkler istedenfor firkanter.

9 Tabeller

Det siste vi skal se på i dette heftet er *tabeller*, eller *array* som det heter på engelsk. I en tabell kan du lagre data. En tabell med heltallsverdier skrives som `int tabell[] = new int[10]`. Dette gir oss en tabell med plass til 10 tall. I eksempelet under bruker vi to tabeller med plass til 60 desimaltall. Prøv koden og se hva den gjør!

```
int tall = 60;
float mx[] = new float[tall];
float my[] = new float[tall];

void setup() {
  size(600,600);
  noStroke(); // Fjerner omrisset til ellipse()
  fill(255, 153);
}

void draw() {
  background(0);

  // Gaar gjennom tabellen og bruker forskjellig verdi
  // for hver ramme.
  // Modulo (%) gir deg en resten etter deling med et
  // tall.
  int hvilken = frameCount % tall;
  mx[hvilken] = mouseX;
  my[hvilken] = mouseY;

  for (int i = 0; i < tall; i++) {
    // hvilken+1 er den eldste verdien i tabellen
    int indeks = (hvilken+1 + i) % tall;
    ellipse(mx[indeks], my[indeks], i, i);
  }
}
```

Pent, ikke sant? Har du lyst til å prøve deg på noen oppgaver i Processing, ta en titt i oppgaveheftet du har fått utdelt.

God koding!

Hilsen oss studentene fra Institutt for informatikk.