# Kanter, kanter, mange

## Introduksjon:

Her skal vi se på litt mer avansert opptegning og bevegelse. Vi skal ta
men bytte ut ballen med trekanter, firkanter og mangekanter. Det anb
du har en forståelse av `if`-setninger og koordinatsystemet fra før. Alt
kanter.

# Steg 1: Enkle firkanter

Vi begynner med rektangler: de firkantene som det er enklest å tegne

## Sjekkliste

- [ ] Start Processing og skriv dette:

```
float x;
float y;
float xFart = 1.5;
float yFart = 2;

void setup() {
  size(640, 480);
  x = width / 2;
```

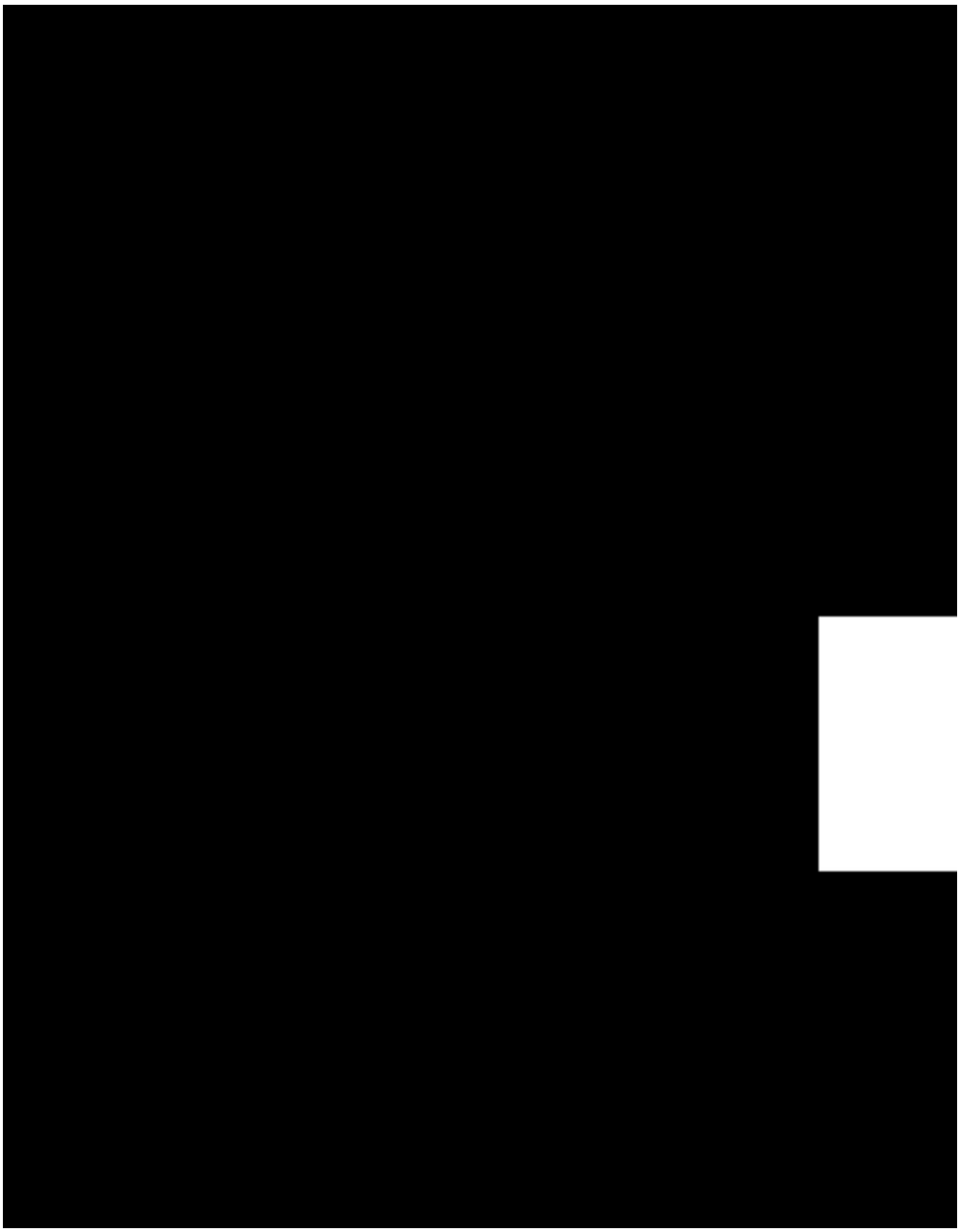
```
  y = height / 2;
}

void draw() {
  x += xFart;
  y += yFart;

  if (x < 0) {
    xFart = -xFart;
  }

  if (x > width - 100) {
    xFart = -xFart;
  }

  if (y < 0) {
    yFart = -yFart;
  }

  if (y > height - 100) {
    yFart = -yFart;
  }

  background(0);
  rect(x, y, 100, 100);
}
```

Dette programmet er ganske likt det som ble lagd i siste oppgav
forskjeller:

- [ ] Vi har endret tallene brukt i `if`-setningene. Hvorfor tror d
sirkel med samme posisjon og størrelse som firkanten?

- [ ] Vi har også tatt i bruk `+=`. `x += 1;` gjør det samme som
Altså øk `x` med det som står på høyresiden av `+=`.

- [ ] Kjør programmet ved å trykke på **Ctrl + R** eller knappen 



- [ ] Lagre programmet som Firkant ved å trykke på **Ctrl+S** eller vel

# Utfordringer

- [ ] Kan du lage et rektangel som ikke er kvadratisk, altså hvor b

skal sprette idet den treffer kanten av vinduet.

# Enkle trekanter

Å tegne rektangler var omtrent helt likt som å tegne sirkler, men nå s
tegnet opp med en posisjon og en bredde og høyde, hadde man ikke
ut. Derfor må vi si for hvert hjørne befinner seg.

## Sjekkliste

- [ ] Vi skal nå bytte ut firkanten med en enkel trekant. Endre `draw`

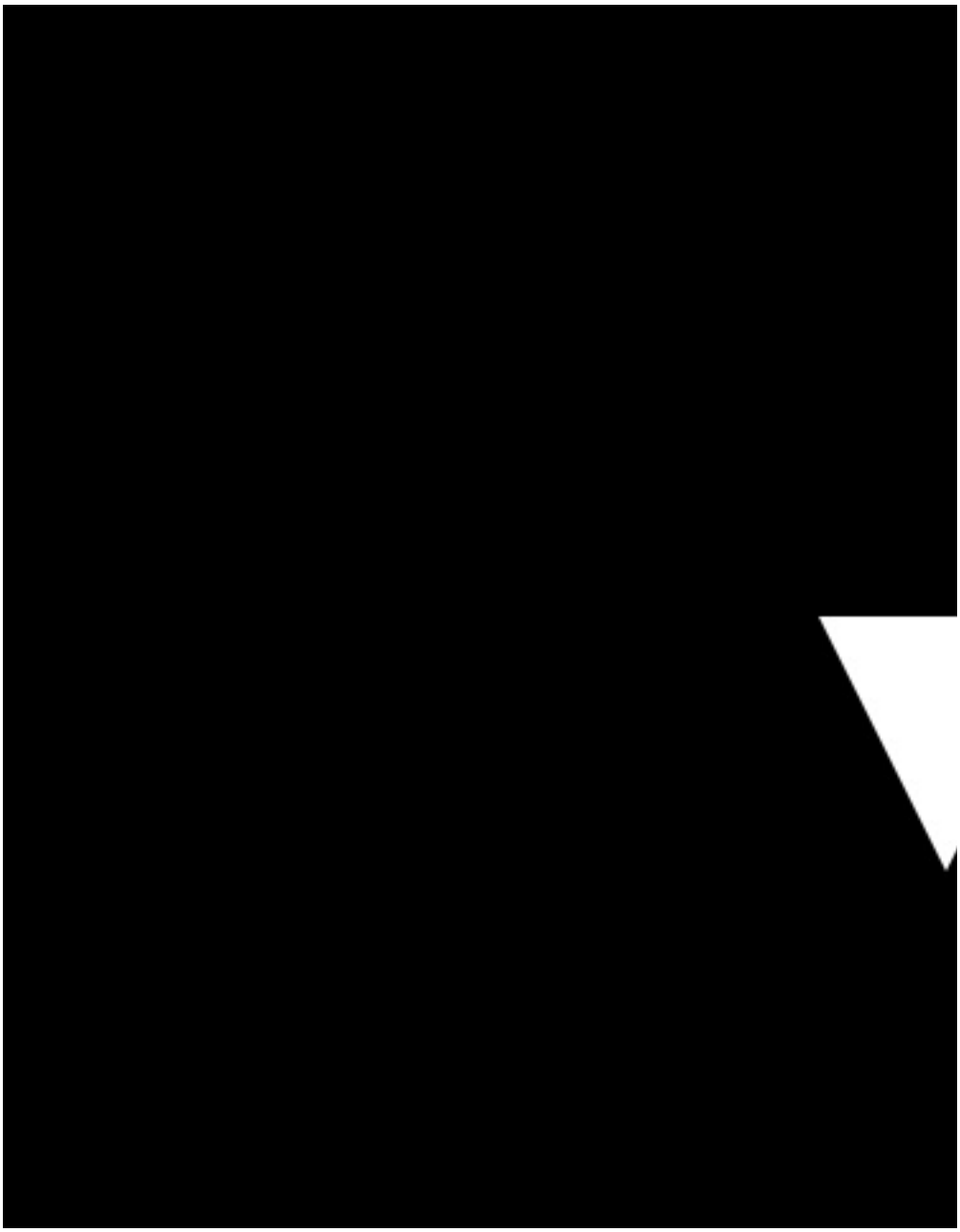
```
void draw() {
  x += xFart;
  y += yFart;

  if (x < 0) {
    xFart = -xFart;
  }

  if (x > width - 100) {
    xFart = -xFart;
  }

  if (y < 0) {
    yFart = -yFart;
  }

  if (y > height - 100) {
    yFart = -yFart;
  }

  background(0);
```

```
  triangle(x, y, x + 100, y, x + 50, y + 100);
}
```

Her har vi tatt i bruk `triangle` istedenfor `rect`. Denne tar imot
`y` er posisjonen til det første hjørnet øverst til venstre, `x + 100`
`50, y + 100` er det siste hjørnet nederst i midten.

- [ ] Lagre programmet som *Trekant* ved å velge **File -> Save as** ell

- [ ] Kjør programmet.



# Forbedre leseligheten

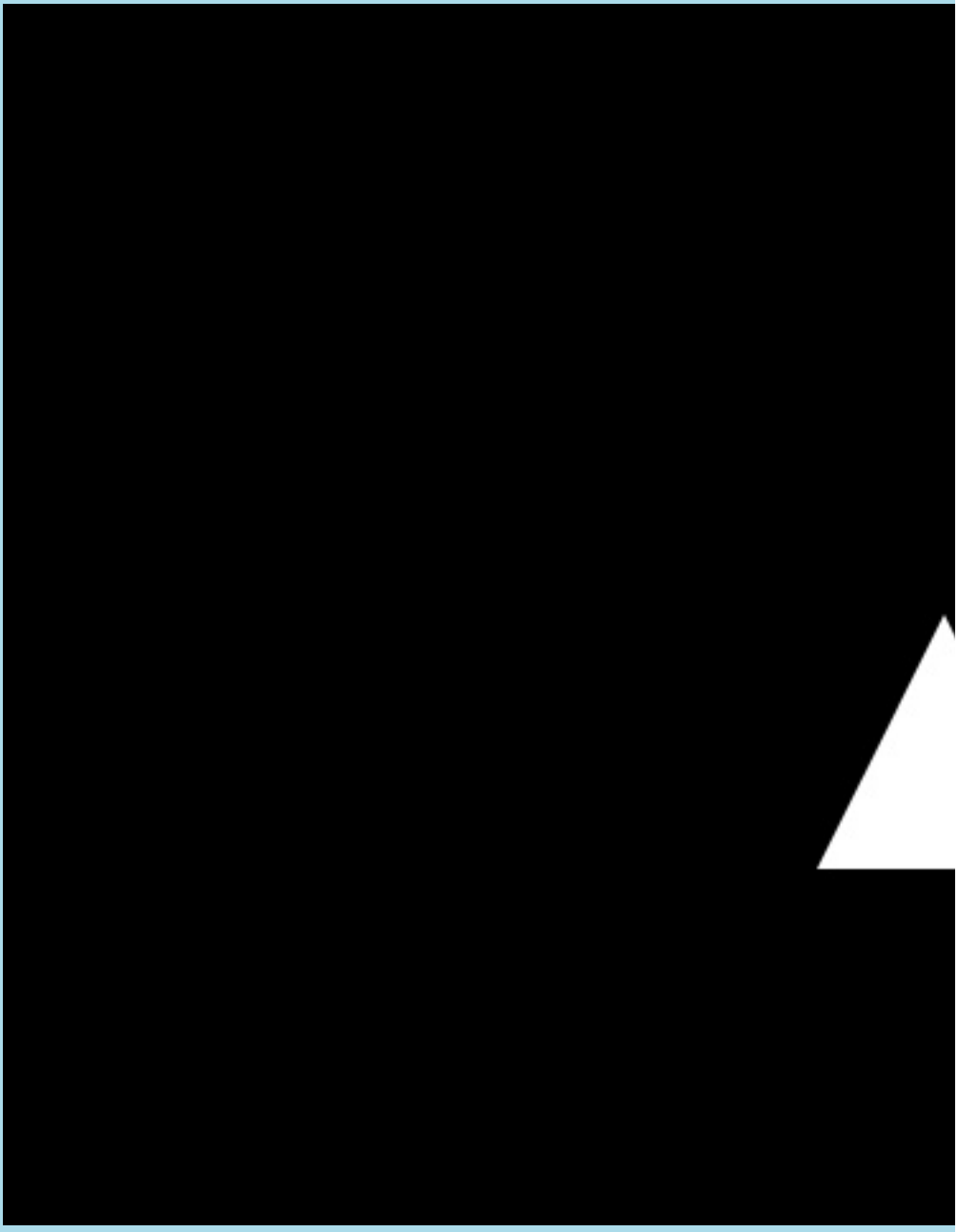
Noen ganger kan det være vanskelig å lese kode med kall på funks
de fleste funksjoner bare imot noen få argumenter, men `triangle`
over flere linjer. For eksempel kunne setningen ovenfor vært skrev

```
triangle(x, y,
  x + 100, y,
  x + 50, y + 100);
```

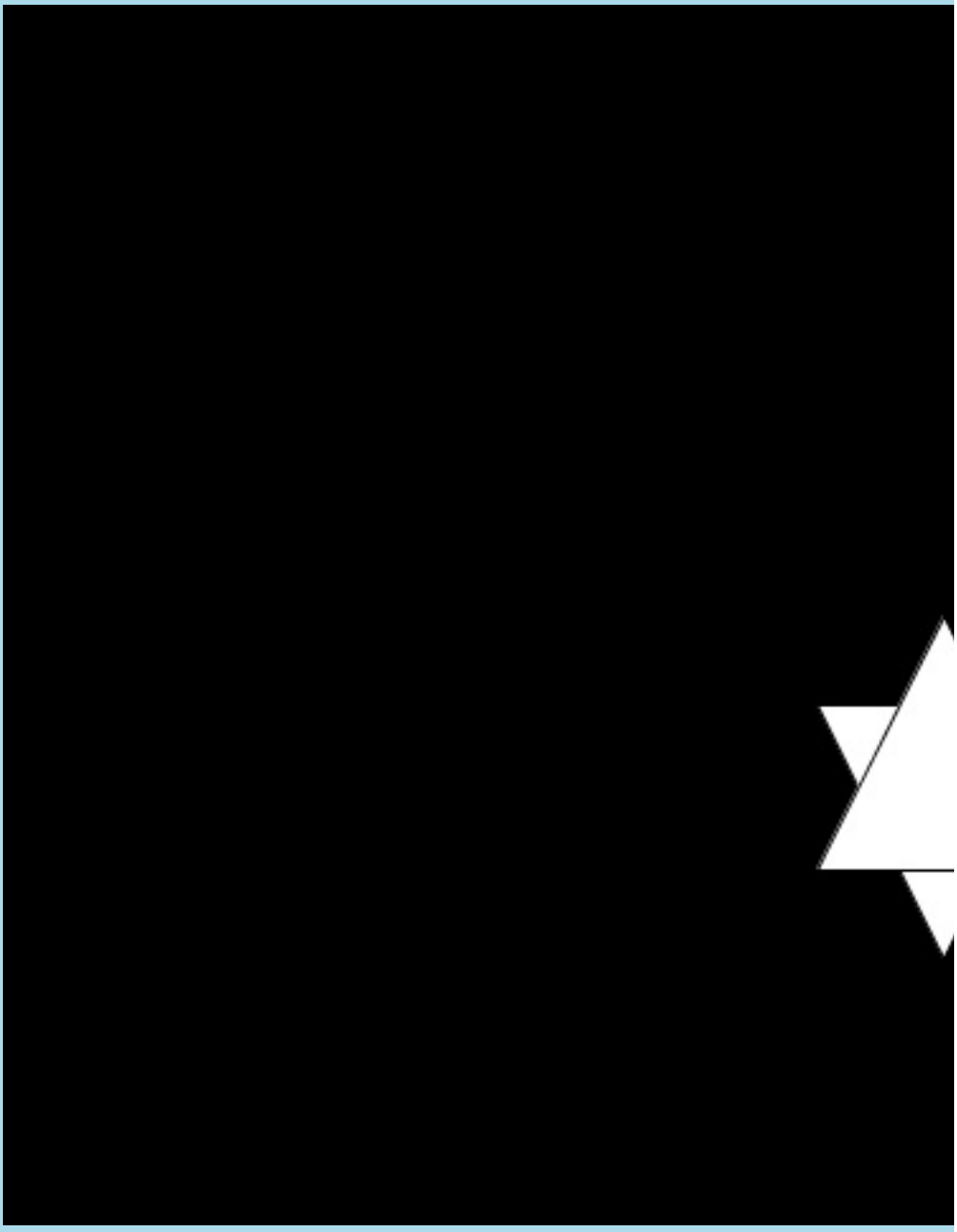
Hvis man fortsatt synes det er vanskelig å lese eller rotete, kan ma
linje. Merk at om man bruker automatisk formatering av koden i Pr
overflødig.

# Utfordringer

- [ ] Kan du tegne trekanten motsatt vei, sånn at den ser ut som e



- [ ] Kan du tegne to trekanter istedenfor en og lage en sekskante



- [ ] **Vanskelig**: Trekanten i programmet er nesten likesidet, men
piksler for lange. Kan du endre regnestykket `y + 100` sånn a
læresetning, eller sinus-funksjonen, `sin` i Processing, for å fi
være til hjelp for å gjøre om grader til radianer hvis du vil bru

# Trekanter

Nå skal vi se hvordan vi kan lage trekanter hvor hvert hjørne beveger
fart for hvert hjørne. Til sammen blir dette fire variabler for hvert hjør
for x-fart og en for y-fart. Ettersom trekanten har tre hjørner, blir dett

Vi kunne kalt dem f.eks. `x1`, `x2`, `x3` og tilsvarende lagt tall til `y`, `xF`
*array*. Det er vanlig å bruke det engelske ordet også på norsk, men de
eller matrise.

## Sjekkliste

- [ ] Vi begynner med å endre variablene til *arrays*:

```
float[] x = new float[3];
float[] y = new float[3];
float[] xFart = new float[3];
float[] yFart = new float[3];
```

Nå har vi endret typen av variablene fra `float` til `float[]`. Nå
*array* som inneholder verdier av typen foran klammene. Bak likl
vi skal lage en ny `float`-*array* med tre tall i.

- [ ] Nå må vi endre startverdiene til disse tallene, ellers vil de bare

```
void setup() {
  size(800, 600);

  x[0] = width / 2;
  x[1] = width / 2;
  x[2] = width / 2;

  y[0] = height / 2;
  y[1] = height / 2;
  y[2] = height / 2;
```

```
  xFart[0] = 1.5;
  xFart[1] = 2.5;
  xFart[2] = 3.5;

  yFart[0] = -5;
  yFart[1] = 2.5;
  yFart[2] = -1.5;
}
```

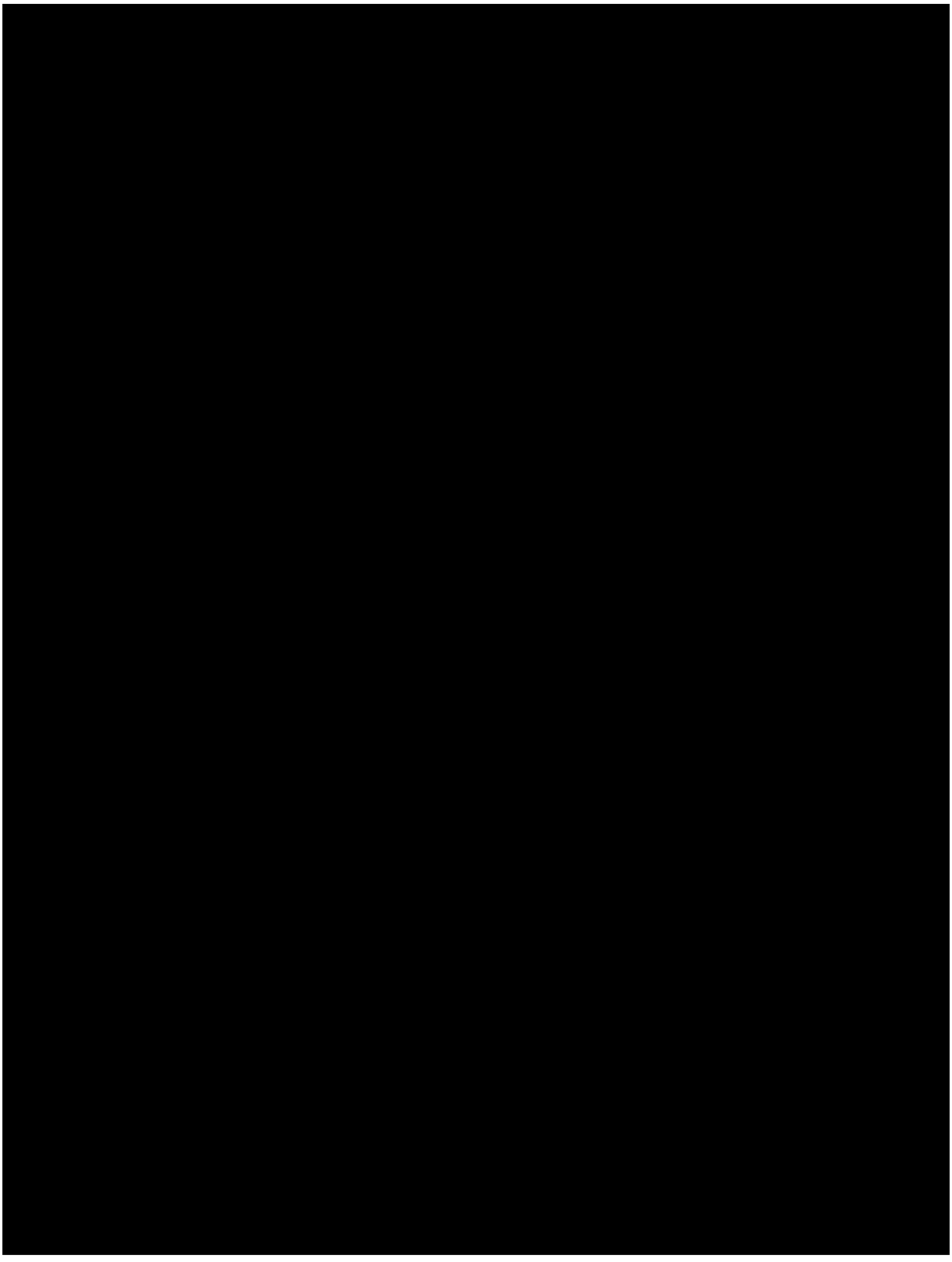
Her ser vi hvordan vi jobber med verdiene i en *array*. Vi bruker f
jobbe med. Den første verdien finnes på plass `0`, og den siste v
Tallet for plasseringen kalles *indeks*. Indeksen er alltid én lavere
Derfor er den siste indeksen én lavere enn størrelsen.

- [ ] Og til slutt må vi flytte rundt på hjørnene og tegne opp trekante

```
void draw() {
  for (int i = 0; i < x.length; i++) {
    x[i] += xFart[i];
    y[i] += yFart[i];

    if (x[i] < 0) {
      xFart[i] = -xFart[i];
    }

    if (x[i] > width) {
      xFart[i] = -xFart[i];
    }

    if (y[i] < 0) {
      yFart[i] = -yFart[i];
    }

    if (y[i] > height) {
      yFart[i] = -yFart[i];
    }
  }
```

```
  background(0);
  triangle(x[0], y[0], x[1], y[1], x[2], y[2]);
}
```

Her ser du en helt ny konstruksjon som vi skal se nærmere på i programmet.



## Forklaring

I begynnelsen av `draw` har vi nå lagt inn noe som kalles en løkke, utføres flere ganger. Det finnes andre slags løkker, og denne kalles

tre setninger. Den første, `int i = 0`, blir utført før løkken. Den nes
skal utføres eller om løkken er ferdig. Den siste, `i++`, utføres etter
løkken. `i` bruker vi inne i løkken som indeks når vi jobber med arr

Så om vi går gjennom koden steg for steg, ser vi at først lages en v
`0`. `int` er typen som brukes for tall uten desimaler, altså *heltall* e
mindre enn størrelsen til arrayen `x`. Hvis den er det, og det er den
koden mellom krøllparentesene. Når all koden mellom krøllparente
oss. `i++` gjør det samme som `i = i + 1`, altså det øker `i` med `1`
størrelsen til `x`. Og sånn fortsetter det helt til `i` blir like stor eller

Løkker som ser slik ut, med et heltall som økes med én og sjekkes
brukes til å jobbe med arrayer. Du kommer til å se mange slike i fre
man blir god på det, men etter hvert blir man veldig glad for at ma
ganger.

## Utfordringer

- [ ] Det går også an å lage firkanter hvor man plasserer hvert hjø
istedenfor `rect`. Prøv å endre programmet til å lage en firkan
Hvor mange flere variabler trenger du enn for trekanten? Hvo

# Mangekanter

Nå skal vi se på hvordan vi kan lage mangekanter. Mangekanter er ba
trekanter, firkanter, femkanter, osv.

## Sjekkliste

- [ ] Vi begynner med å endre på størrelsen på *arrayene* i forrige opp

```
int KANTER = 5;
float[] x = new float[KANTER];
float[] y = new float[KANTER];
float[] xFart = new float[KANTER];
float[] yFart = new float[KANTER];
```

Nå bruker vi en variabel for å sette størrelsen isteden. Dette hje
kanter fordi vi bare trenger å endre tallet ett sted istedenfor fire

Posisjonene og hastighetene til hjørnene ble satt til faste verdie
er fungerer ikke det så bra. Så vi endrer på `setup` til å bruke er

```
void setup() {
  size(800, 600);

  for (int i = 0; i < KANTER; i++) {
    x[i] = random(width);
    y[i] = random(height);
    xFart[i] = random(-5, 5);
    yFart[i] = random(-5, 5);
  }
}
```

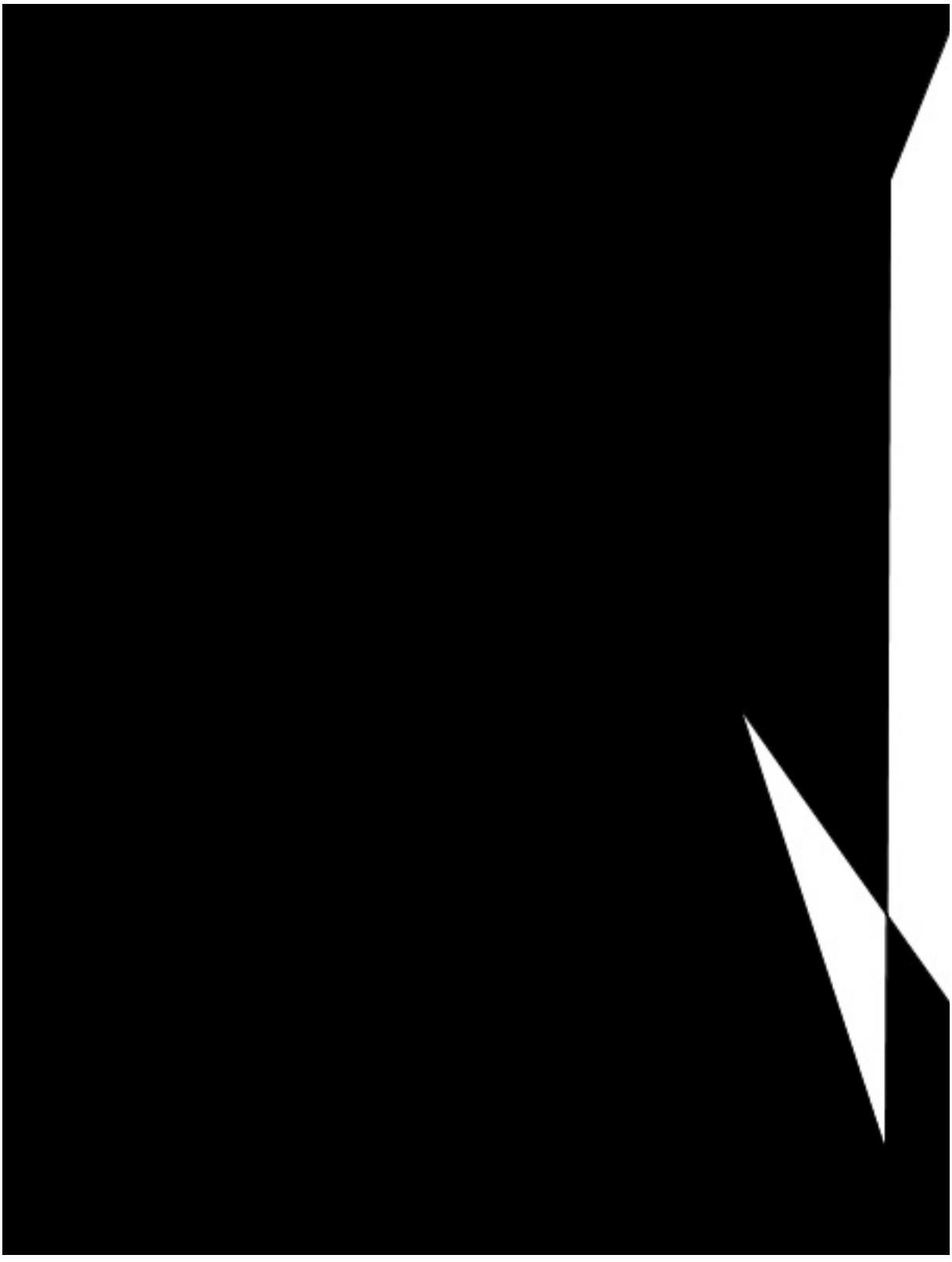
Denne løkken likner en del på den vi har i `draw` fra før. Vi har de
Denne gir oss tilfeldige tall. Hvis vi kaller den uten noen verdier
den med én verdi, `random(width)`, får vi et tall mellom `0` og ve
et tall mellom de to verdiene.

Nå skal vi tegne opp mangekanten vår. Vi trenger ikke å endre p
bytte testen så den likner den over. Vi skal bytte ut kallet på `tr`

```
void draw() {
  for (int i = 0; i < KANTER; i++) {
    x[i] += xFart[i];
    y[i] += yFart[i];
```

```
    if (x[i] < 0) {
      xFart[i] = -xFart[i];
    }

    if (x[i] > width) {
      xFart[i] = -xFart[i];
    }

    if (y[i] < 0) {
      yFart[i] = -yFart[i];
    }

    if (y[i] > height) {
      yFart[i] = -yFart[i];
    }
  }

  background(0);

  beginShape();
  for (int i = 0; i < KANTER; i++) {
    vertex(x[i], y[i]);
  }
  endShape(CLOSE);
}
```

Her ser vi tre nye funksjoner: `beginShape`, `vertex` og `endShape`
betyr at vi skal legge til et hjørne i formen, den tar inn to verdie
ferdig og klar til å tegnes på skjermen. Hvis vi kaller `endShape`



## Utfordringer

- [ ] Kan du bruke `random` til å få hjørnene til å endre hastighet nå

  Pass på, om farten blir lavere enn den var, kan hjørnet bli sitt

egentlig lar den bevege seg litt utenfor vinduet for så å snu. S
vinduskanten inne i `if`-setningene for å unngå det.