Crash course i Programmering

HumTek, RUC

Resume - Vigtigste begreber

- Kommentarer
- Funktioner
- Udtryk
- Sætning
- Koordinater i display-vindue
- Funktioner (til tegning): size(), background(), point(), line(), rect(), triangle(), quad(), ellipse(), bezier(), fill(), noFill(), stroke(), noStroke(), strokeWeight(), strokeCap(), strokeJoin(), smooth(), noSmooth()
- Data
- Datatype
- Variabel
- Tilskrivning (assignment) "="
- Aritmetiske udtryk med +, -, *, /, %, og paranteser
- Forkortet skrivemåde ved tilskrivning ++, --, +=, -=, *=, /=
- Funktioner (aritmetriske): ceil(), floor(), round(), abs(), sq(), sqrt(), pow(), min(), max()

Control 1: logiske udtryk

- Logiske udtryk
  - kan angives med relationerne
    `>, <, >=, <=, ==, !=`
  - og med operatorerne
    `||, &&, !`

- Hvad betyder de?

- Angiv eksempler på udtryk og deres værdi?

- Hvad udskriver nedenstående program

```
int x=7;
int y=5;
int z=9;
println(x>y);
println((x>y) || (y>=z));
println((x>y) && (y>=z));
println((x>y) || (z%3==0));
```

Betingede sætninger

- simpel if-sætning

```
if (test) {
  sætninger, der udføres
  hvis test er sand
}
```

- if-else konstruktion

```
if (test) {
  sætninger, der udføres
  hvis test er sand
} else {
  sætninger, der udføres
  hvis test er falsk
}
```

```
int x = 101;
if (x > 100) {
  ellipse(50, 50, 36, 36);
}
if (x < 100) {
  rect(33, 33, 34, 34);
}
line(20, 20, 80, 80);
```

5-05

```
int x = 100;
if (x > 100) {
  ellipse(50, 50, 36, 36);
} else {
  rect(33, 33, 34, 34);
}
line(20, 20, 80, 80);
```

Control 1 Opgaver

- Bogen s 59 opg 1

- Opg A
  - Med random(100) fås et tilfældigt tal imellem 0 og 100. Brug dette til at skrive et program der tegner et kvadrat eller en cirkel med samme sandsynlighed.

- Opg B
  - Som opg A men nu et kvadrat, en cirkel eller en trekant med (ca samme sansynlighed)

Control 2: Brug af løkker/iteration

i stedet for

```
size(200,200);
line(20, 20, 20,180);
line(30, 20, 30,180);
line(40, 20, 40,180);
line(50, 20, 50,180);
line(60, 20, 60,180);
line(70, 20, 70,180);
line(80, 20, 80,180);
line(90, 20, 90,180);
line(100, 20, 100,180);
line(110, 20, 110,180);
line(120, 20, 120,180);
line(130, 20, 130,180);
line(140, 20, 140,180);
line(150, 20, 150,180);
line(160, 20, 160,180);
line(170, 20, 170,180);
line(180, 20, 180,180);
```

kan man bruge en **for**-løkke

```
size(200,200);
for (int i = 20; i < 190; i = i + 10) {
  line(i, 20, i, 180);
}
```

Løkker

- for-løkke

```
for (init; test; update) {
  sætninger;
}
```

- udføres således
  1. `init` udføres
  2. `test` evalueres
  3. hvis `test` er sand
     gå til 4 ellers gå til 6
  4. udfør `sætninger`
  5. udfør `update og gå til 2`
  6. gå videre til resten af programmet

eksempel

```
size(200,200);
for (int i = 20; i < 190; i = i + 10) {
  line(i, 20, i, 180);
}
```

Eksempel

- eksempel

```
noFill();
for (int d = 10; d < 160; d = d + 10) {
  ellipse(50, 50, d, d);
}
```

- variation 1

```
noFill();
for (int d = 150; d > 0; d = d - 10) {
  ellipse(50, 50, d, d);
}
```

- variation 2 - Forklar resultatet???

```
for (int d = 10; d < 160; d = d + 10) {
  ellipse(50, 50, d, d);
}
```

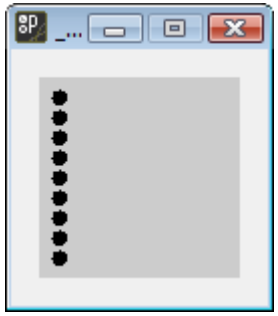
Nested iteration

- "nested" = "indlejret"
- Nestede løkker
  - en løkke inde i en løkke
  - en løkke inde i en løkke inde i en løkke ...

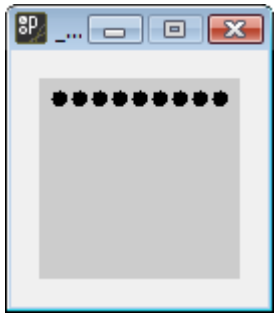
Eksempel

- at iterere ned af Y-aksen

```
strokeWeight(8);
for (int y=10; y<100; y = y + 10) {
  point(10, y);
}
```



- hvordan iterere hen af X-aksen?

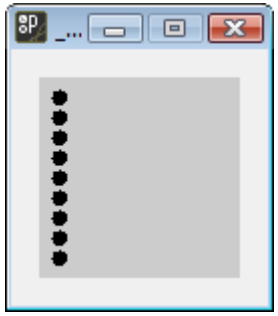


.

Eksempel

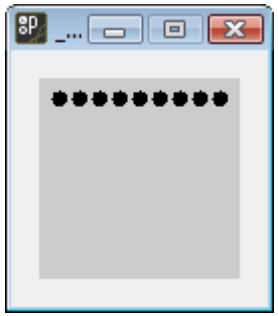
- at iterere ned af Y-aksen

```
strokeWeight(8);
for (int y=10; y<100; y = y + 10) {
  point(10, y);
}
```



- at iterere hen af X-aksen

```
strokeWeight(8);
for (int x = 10; x < 100; x = x + 10) {
  point(x, 10);
}
```



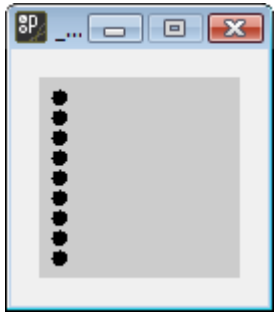
- Hvordan med begge dele på een gang?

- .

Eksempel

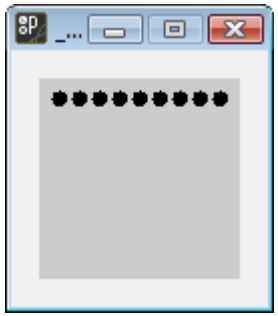
- at iterere ned af Y-aksen

```
strokeWeight(8);
for (int y=10; y<100; y = y + 10) {
  point(10, y);
}
```



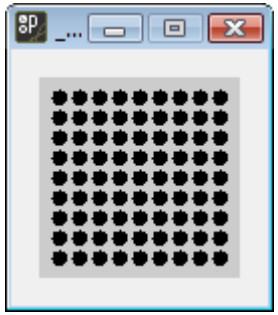
- at iterere hen af X-aksen

```
strokeWeight(8);
for (int x = 10; x < 100; x = x + 10) {
  point(x, 10);
}
```

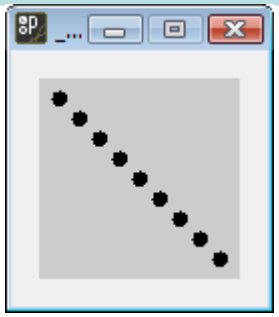


- begge dele i en nested løkke

```
strokeWeight(8);
for (int y = 10; y < 100; y = y + 10) {
  for (int x = 10; x < 100; x = x + 10) {
    point(x, y);
  }
}
```

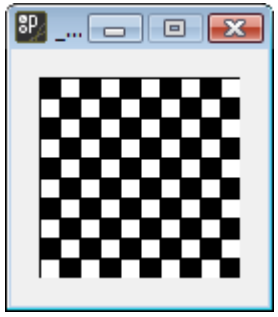


Control 2 Opgaver



- Opg A
  - skriv et program, der tegner sådan:

- Opg B
  - skriv et program der tegner et skakbræt-mønster på 10*10 felter hver af 10 * 10 punkter.
    Tip: Hvis man i en nested løkke med Y og X starter med 0 og lægger 10 til Y hhv X hver gang, så vil følgende udtryk hele tiden skifte imellem sandt og falsk
    - `(x+y)/10%2 == 0)`
  - forklar hvorfor ovenstående udtryk skifter værdi



Pæne programmer (Formattering)

- sådan noget kode:

```
strokeWeight(8); for (int y = 10; y < 100; y = y + 10) {
for (int x = 10; x < 100; x = x + 10) {
point(x, y);  }
}
```

- kan godt blive uoverskueligt, så gør således i stedet

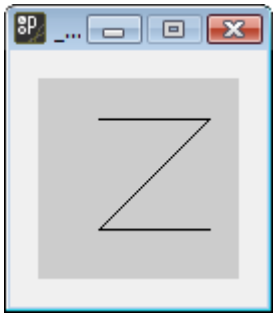
```
strokeWeight(8);
for (int y = 10; y < 100; y = y + 10) {
  for (int x = 10; x < 100; x = x + 10) {
    point(x, y);
  }
}
```

- der findes varianter, men
- denne formmatering kan Processing's PDE finde udaf
  - "Auto format" under "Tools"-menuen
  - eller blot Ctrl+T

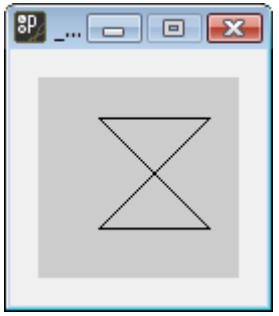
Shape 2: Vertices (Punktfølger)

- Vertex (= "vinkelspids")
- Tegning med punktfølger
  - start punktfølge
    - `beginShape(),`
  - definer punkt (x,y)
    - `vertex(x,y)`
  - slut punktfølge
    - `endShape(),`

```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(30, 75);
vertex(85, 75);
endShape();
```

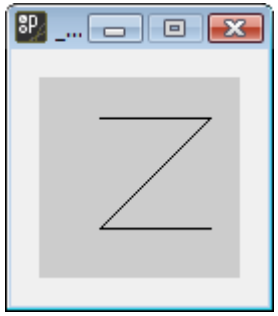


```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(30, 75);
vertex(85, 75);
endShape(CLOSE);
```

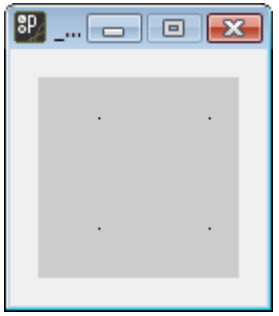


-

```
noFill();
beginShape(LINES);
vertex(30, 20);
vertex(85, 20);
vertex(30, 75);
vertex(85, 75);
endShape();
```



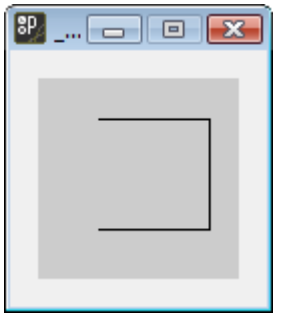
```
noFill();
beginShape(POINTS);
vertex(30, 20);
vertex(85, 20);
vertex(30, 75);
vertex(85, 75);
endShape();
```



beginShape(MODE),

- beginShape(MODE)
  - MODE kan være
    - POINTS,
    - LINES,
    - TRIANGLES,
    - TRIANGLE_FAN,
    - TRIANGLE_STRIP,
    - QUADS,
    - QUAD_STRIP
  - der angiver hvordan punktfølgen fortolkes

```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(30, 75);
vertex(85, 75);
endShape();
```



Kurvetegning

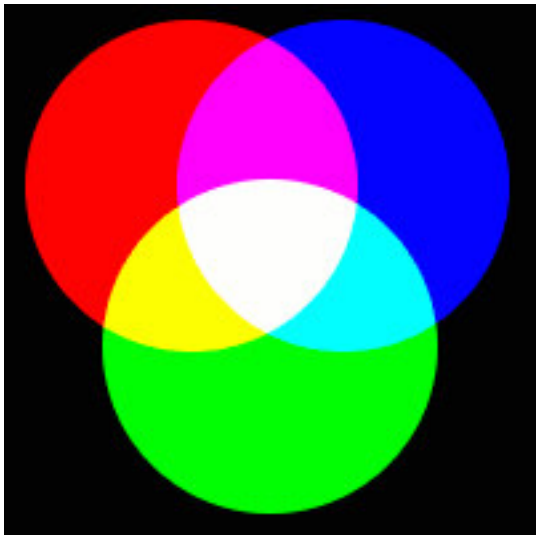
- Punktfølger kan benyttes til tegning af kurver med
  - curveVertex()
  - bezierVertex()

- Find dem i referencen

Shape 2 Opgaver

- Bogen s 77 Opg 1-3
- Opg A.
  - Tegn en linie med en punktfølge sådan at 100 tilfældige punkter forbindes med lige kanter (Brug funktionen random())
- Opg B
  - Som opg A men med kurver (brug curveVertex)
- Opg C
  - Tegn 50 tilfældige trekanter
- Opg D
  - Tegn 25 tilfældige firkanter

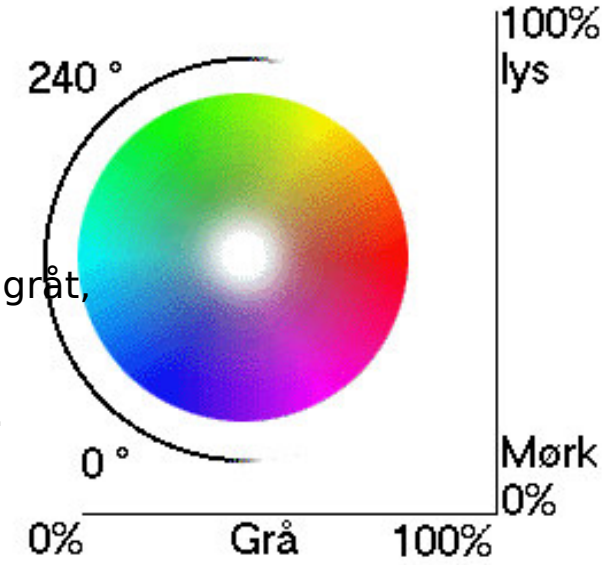
Color 1: Farver



- Farver bruges i
  - background()
  - fill()
  - stroke()

- RGB farvekodning
  - farver angives ved 3 værdier for hhv
    - rød (mellem 0 og 255)
    - grøn (mellem 0 og 255)
    - blå (mellem 0 og 255)

- HSB
  - Hue (mellem 0 og 360)
    - Beskriver selve farvens "kulør".
  - Saturation (mellem 0 og 100)
    - Farvens styrke eller mætning, dvs. mængden af gråt, der er i farven.
  - Brightness (mellem 0 og 100)
    - Angiver hvor forholdsvis lys eller mørk farven er.



Farve funktioner og hex koder

- Funktioner
  - color()
  - colorMode()

- Hexadecimal (hex) notation
  - farve angivet ved 16-talssystemet
  - hvad er det?
- Bemærk
  - i PDE'en er der et tool der hedder "color selector"

Color 1 Opgaver

- Bogen side 93 Opg 1-3

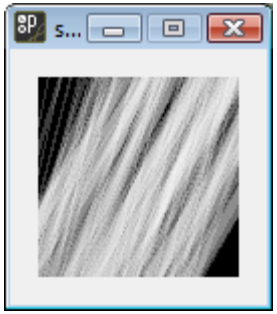
Math 4: Tilfældige tal

- tilfældige tal
  - random()
  - randomSeed()

- Brug af random til at styre forløbet i et program

```
float r = random(2);
if (r < 1) {
  line(0, 0, 100, 100);
} else {
  ellipse(50, 50, 75, 75);
}
```

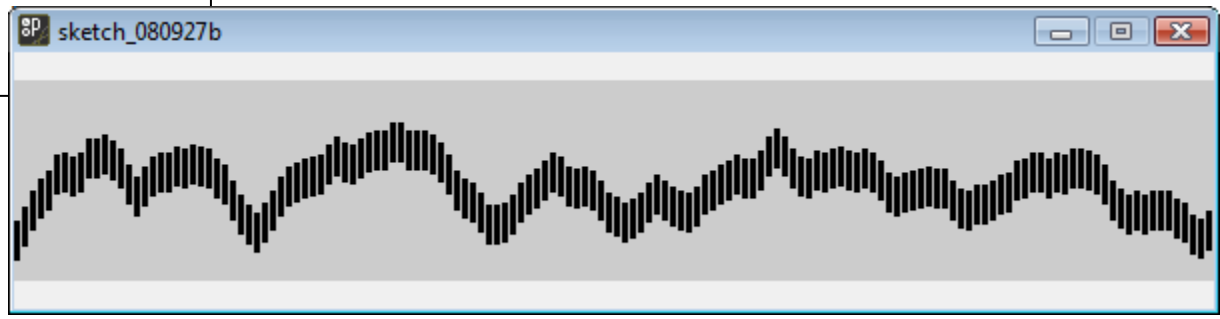
```
background(0);
stroke(255, 60);
for (int i = 0; i < 100; i++) {
  float r = random(10);
  strokeWeight(r);
  float offset = r * 5.0;
  line(i-20, 100, i+offset, 0);
}
```



Math 4: Tilfældige tal

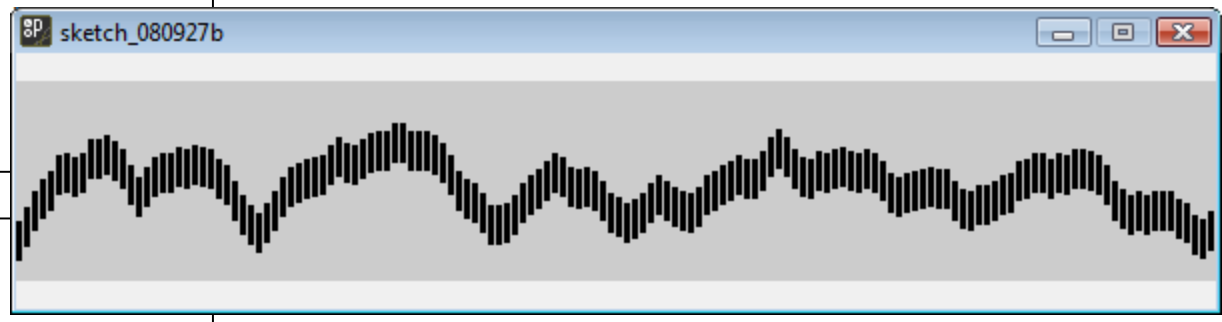
- Noise
  - en mere kontrolleret form for tilfældighed, hvor en ny værdi er påvirket af den foregående
  - noise()
  - noiseSeed()

```
size(600, 100);
float v = 0.0;
float inc = 0.1;
noStroke();
fill(0);
for (int i = 0; i < width; i = i+4) {
  float n = noise(v) * 70.0;
  rect(i, 10 + n, 3, 20);
  v = v + inc;
}
```

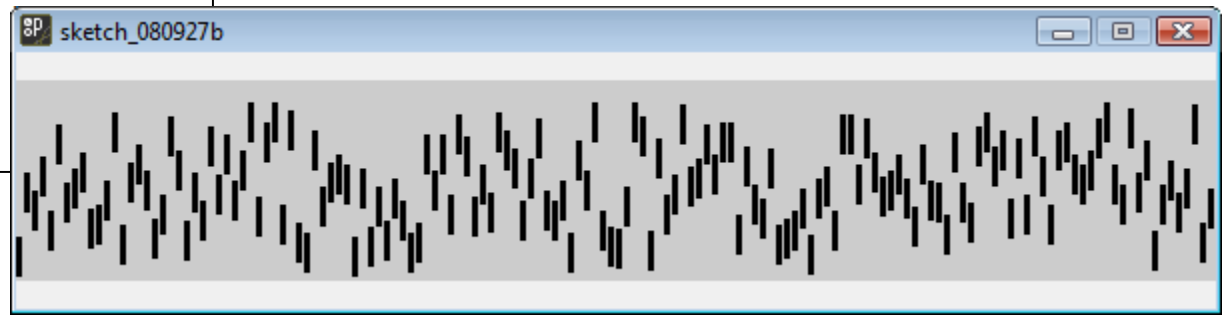


random() og noise()

```
size(600, 100);
float v = 0.0;
float inc = 0.1;
noStroke();
fill(0);
for (int i = 0; i < width; i = i+4) {
  float n = noise(v) * 70.0;
  rect(i, 10 + n, 3, 20);
  v = v + inc;
}
```




```
size(600, 100);
float v = 0.0;
float inc = 0.1;
noStroke();
fill(0);
for (int i = 0; i < width; i = i+4) {
  float n = random(1) * 70.0;
  rect(i, 10 + n, 3, 20);
  v = v + inc;
}
```



Math 4 Opgaver

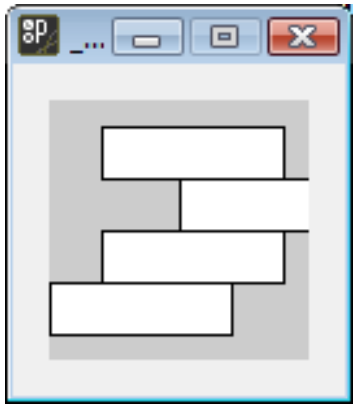
- Se tidligere opgaver fra vertices/punktfølger

- Prøv (eventuelt!) at kikke på noise() og prøve at bruge den. Se eksempler på processing.org

Transform 1: Translate, Matrices

- Translation
  - at flytte origo (nulpunktet i koordinatsystemet)
  - translate()
- Matrix-stak
  - bruges til at holde styr på transformationer
  - pushMatrix()
  - popMatrix()

```
pushMatrix();
translate(20, 0);
rect(0, 10, 70, 20); // Draws at (20, 30)
pushMatrix();
translate(30, 0);
rect(0, 30, 70, 20); // Draws at (50, 30)
popMatrix();
rect(0, 50, 70, 20); // Draws at (20, 50)
popMatrix();
rect(0, 70, 70, 20); // Draws at (0, 70)
```



Transform 1 Opgaver

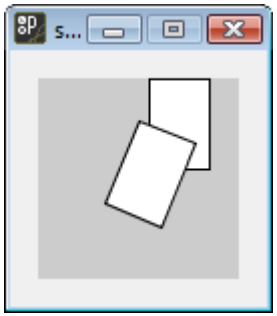
- Opg A
  - Skriv et program der gør følgende. Tegner en cirkel i (0,0), flytter 0-punktet til midten af displayvinduet, tegner samme cirkel og flytter nulpunktet endnu engang til nederste højre punkt i displayvinduet og tegner cirklen en sidste gang.

- Opg B
  - Som opg A, men nu skal indstillingen af koordinatsystemet (nulpunktet) gemmes fra starten og hentes til slut, hvorefter der tegnes endnu en cirkel der er lidt større

Transform 2: Rotation og skalering

- Rotation rotate()
  - angives med en vinkel og roterer koordinatsystemet med denne
  - vinklen opgives i radianer (altså imellem 0 og 2*PI)

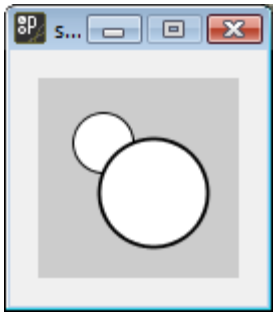
```
smooth();
rect(55, 0, 30, 45);
rotate(PI/8);
rect(55, 0, 30, 45);
```



Skalering

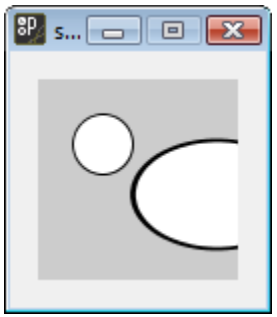
- Skalering scale()
  - skalerer koordinatsystemet
  - uniform skalering med X
    - scale(X)

```
smooth();
ellipse(32, 32, 30, 30);
scale(1.8);
ellipse(32, 32, 30, 30);
```



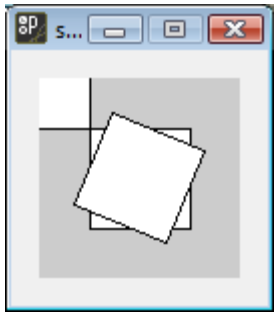
  - separat akse-skalering med X,Y
    - scale(X,Y)

```
smooth();
ellipse(32, 32, 30, 30);
scale(2.8, 1.8);
ellipse(32, 32, 30, 30);
```

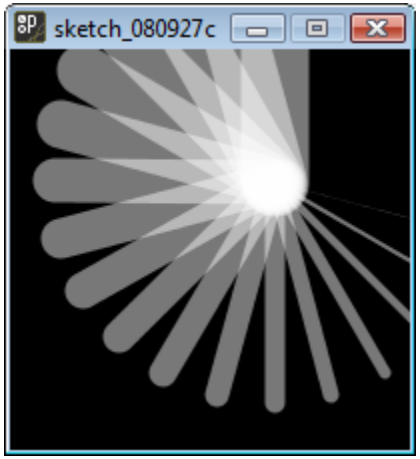


Sammensatte transformationer

```
rect(-25, -25, 50, 50);
translate(width/2, height/2);
rect(-25, -25, 50, 50);
rotate(PI/8);
rect(-25, -25, 50, 50);
```




```
size(200,200);
background(0);
smooth();
stroke(255, 120);
translate(132, 66); // Set initial offset
for (int i = 0; i < 18; i++) { // 18 repetitions
  strokeWeight(i*2); // Increase stroke weight
  rotate(PI/12); // Accumulate the rotation
  line(0, 0, 110, 0);
}
```



Transform 2 Opgaver

- Opg A
  - Tegn en trekant med en spids der peger opad og tegn den samme endnu engang, blot vendt på hovedet
- Opg B
  - varier opg A så trekanten tegnes 10 gange på vejen om på hovedet.
- Opg C
  - tegn en trekant der fylder det meste af vinduet og gentegn denne et antal gange i en for-løkke, hvor den skalerer ned
- Opg D
  - prøv at kombinere opg B og C

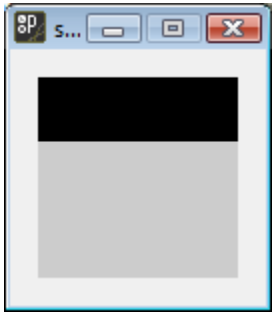
Structure 2: Continuous mode

- Continuous mode
  - vedvarende udførelse af (en del af) programmet
  - som en tegnefilm opdelt i frames

- Continuous mode funktioner mm.
  - setup()
    - kaldes een gang
    - bruges til at initialisere
  - draw()
    - kaldes umiddelbart efter setup()
    - og derefter vedvarende indtil programmet stoppes eller til noLoop()
    - hvert kald svarer til en frame
  - frameRate() og frameCount
    - frameRate() sætter antal frames pr sekund
    - frameCount angiver antal frames siden programstart
  - noLoop() og loop()
    - bruges til hhv at stoppe og genstarte den vedvarende udførelse af draw()

setup(),draw() eksempel

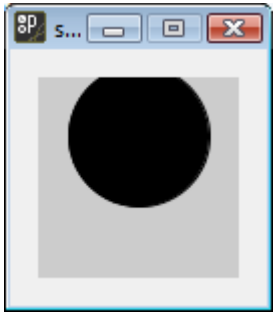
```
float y = 0.0;
void draw() {
  frameRate(30);
  line(0, y, 100, y);
  y = y + 0.5;
}
```



```
float y = 0.0;

void setup() {
  size(100, 100);
  smooth();
  fill(0);
}

void draw() {
  background(204);
  ellipse(50, y, 70, 70);
  y += 0.5;
  if (y > 150) {
    y = -50.0;
  }
}
```



Structure 2 Opgaver

- Bogen s 180 1+2+3
- Opg A
  Brug noLoop til at
  få et program til at udføre
  draw() 200 gange
- Opg B
  Hvad sker der hvis
  background()-kaldet i
  programmet til højre flyttes
  til setup() i stedet?

```
float y = 0.0;

void setup() {
  size(100, 100);
  smooth();
  fill(0);
}

void draw() {
  background(204);
  ellipse(50, y, 70, 70);
  y += 0.5;
  if (y > 150) {
    y = -50.0;
  }
}
```

- Opg C
  - I stedet for at bruge background kan man bruge fill() og så tegne en rektangel der fylder hele vinduet og man kan vælge en alpha-værdi (mellem 0 og 100, fx 12) for gennemsigtigheden af fyldfarven. Ved at bruge muse-variablene mouseX og mouseY kan man få en figur til at følge musens bevægelser.
  - Prøv at skrive et program hvor en figur følger musens bevægelser
  - ret dette så vinduet hele tiden bliver overskrevet med en almindelig farve
  - ret igen så der bruges en gennemsigtig farve

Structure 2 Opgaver

- Opg D
  - Skriv et program der bare bliver ved med at tegne cirkler med centrum der hvor musen er, altså punktet (mouseX,mouseY). Cirklerne skal have en tilfældig diameter på max 70 punkter og fyldes med en tilfældig farve.

- Opg E
  - Prøv at få programmet fra opg D til i det mindste at holde op med at tegne cirkler når man ikke bevæger musen. Brug her pmouseX og pmouseY som er musens koordinater i den umiddelbart foregående frame.

Dagens Debugging