

Crash Course i Programmering

HumTek, RUC

Resume - Vigtigste begreber

- Logiske udtryk med: relationerne `>`, `<`, `>=`, `<=`, `==`, `!=` og operatorerne `||`, `&&`, `!` (svarende til og, eller, ikke)
- Betingede sætninger: `if`, `if-else`
- Løkker/iteration: `for`
- Formattering af programmer (pæn linie-indrykning)

- Vertices/Punktfølger: `beginShape()`-`endShape`, `vertex()`, `curveVertex()`, `bezierVertex()`, `POINTS`, `LINES`, `TRIANGLES`, `QUADS`
- Tilfældige tal: `random()`, `randomSeed()`, `noise()`, `noiseSeed()`
- Transformation: `translate()`, `pushMatrix()`, `popMatrix()`, `rotate()`, `scale()`
- Continuous mode:
`setup()`, `draw()`, `frameRate()` og `frameCount` samt `noLoop()` og `loop()`

Variable og virkefelter

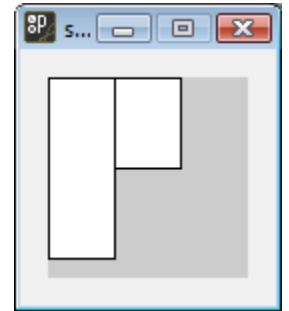
- Blok (block)
 - et område af programmet omkranset af { og }
- Virkefelt (scope)
 - for en variabel er virkefeltet det område af programmet hvor den kan bruges
- Simpel regel om virkefelt
 - en variabel kan KUN bruges i den blok den er defineret i (men dermed også i blokke denne har indlejret)
- "Globale variable"
 - variable defineret på yderste niveau (altså samme niveau som setup() og draw())
 - disse kan bruges overalt i programmet

Eksempel virkefelt

```
int d = 45; // Assign 45 to variable d

void setup() {
  size(100, 100);
  int d = 90; // Assign 90 to local variable d
  rect(0, 0, 33, d); // Use local d with value 90
}

void draw() {
  rect(33, 0, 33, d); // Use d with value 45
}
```



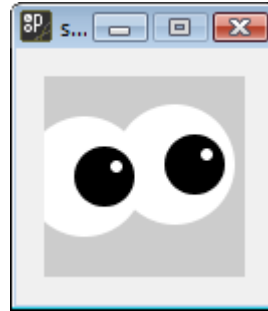
Structure 3: Funktioner

- Abstraktion
 - Betyder (i Comp.Sci.) det at gemme / indpakke mindre væsentlige detaljer
 - funktioner er meget vigtige til dette formål når man skriver programmer
 - alle Processings indbyggede funktioner er gode eksempler
 - `line()`, `ellipse()`, `rect()`, ...
 - men man kan også lave sine egne funktioner ...

At skrive egne funktioner

```
void setup() {
  size(100, 100);
  noStroke();
  noLoop();
  smooth();
}

void draw() {
  // Right shape
  fill(255);
  ellipse(65, 44, 60, 60);
  fill(0);
  ellipse(75, 44, 30, 30);
  fill(255);
  ellipse(81, 39, 6, 6);
  // Left shape
  fill(255);
  ellipse(20, 50, 60, 60);
  fill(0);
  ellipse(30, 50, 30, 30);
  fill(255);
  ellipse(36, 45, 6, 6);
}
```



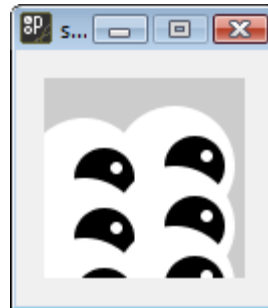
```
void setup() {
  size(100, 100);
  noStroke();
  noLoop();
  smooth();
}

void draw() {
  eye(65, 44);
  eye(20, 50);
}

void eye(int x, int y) {
  fill(255);
  ellipse(x, y, 60, 60);
  fill(0);
  ellipse(x+10, y, 30, 30);
  fill(255);
  ellipse(x+16, y-5, 6, 6);
}
```

At skrive egne funktioner

- I Processing kan funktioner defineres og bruges fra `setup()` og `draw()`
- Når en funktion først er defineret kan den genbruges ubegrænset



```
void setup() {
  size(100, 100);
  noStroke();
  smooth();
  noLoop();
}

void draw() {
  eye(65, 44);
  eye(20, 50);
  eye(65, 74);
  eye(20, 80);
  eye(65, 104);
  eye(20, 110);
}

void eye(int x, int y) {
  fill(255);
  ellipse(x, y, 60, 60);
  fill(0);
  ellipse(x+10, y, 30, 30);
  fill(255);
  ellipse(x+16, y-5, 6, 6);
}
```

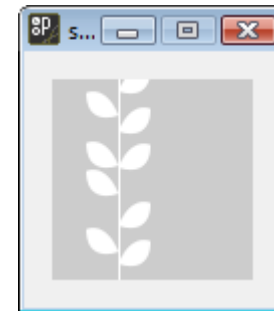
Større eksempel

```
void setup() {
  size(100, 100);
  smooth();
  noStroke();
  noLoop();
}

void draw() {
  vine(33, 9, 16);
}

void vine(int x, int numLeaves, int leafSize ) {
  stroke(255);
  line(x, 0, x, height);
  noStroke();
  int gap = height / numLeaves;
  int direction = 1;
  for (int i = 0; i < numLeaves; i++) {
    int r = int(random(gap));
    leaf(x, gap*i + r, leafSize, direction);
    direction = -direction;
  }
}
```

```
void leaf(int x, int y, int size, int dir) {
  pushMatrix();
  translate(x, y); // Move to position
  scale(size); // Scale to size
  beginShape(); // Draw the shape
  vertex(1.0*dir, -0.7);
  bezierVertex(1.0*dir, -0.7, 0.4*dir, -1.0, 0.0, 0.0);
  bezierVertex(0.0, 0.0, 1.0*dir, 0.4, 1.0*dir, -0.7);
  endShape();
  popMatrix();
}
```



Funktions-"overloading"

- Flere funktioner, samme navn
 - er lovligt, hvis de blot har forskellige parametre (altså forskelligt antal og/eller forskellige typer)
- Eksempel fra processing `fill()`
 - `fill(gray)`
 - `fill(gray, alpha)`
 - `fill(value1, value2, value3)`
 - `fill(value1, value2, value3, alpha)`
 - `fill(color)`
 - `fill(color, alpha)`
 - `fill(hex)`
 - `fill(hex, alpha)`

Returnering af værdi fra funktion

- en funktion kan returnere en værdi
 - den skal blot gives en **type**
 - og der skal returneres fra kroppen af funktionen med **return**

```
void setup() {  
    size(100, 100);  
    float f = average(12.0, 6.0);  
    println(f);  
}
```

```
float average(float num1, float num2) {  
    float av = (num1 + num2) / 2.0;  
    return av;  
}
```

Structure 3 Opgaver

- Opg A
 - Skriv en funktion - uden parametre - kaldet fx "kvadratkirke", der tegner et mørkegrå kvadrat på 20*20 punkter med en hvid cirkel i midten på 18*18 punkter og placerer dette i midten af vinduet.
 - Kald funktionen fra draw()
- Opg B
 - Ret funktionen i opg A så den får 2 parametre x og y og således at den tegner "kvadratkirken" i (x,y). Kald den fra draw() med værdierne hhv (30,30) og (70,70).
- Opg C
 - Ret funktionen i opg B så den får endnu en ekstra parameter z, der angiver sidestørrelsen på kvadratet i kvadratkirken (idet det hvide cirkel tegnes med en lidt mindre diameter).
 - Kald funktionen et antal gange fra draw() så der kommer "firkantcirkler" forskellige steder i vinduet.
 - Brug en for-løkke til dette.

Input 1: Interaktion med musen

- Muse-koordinater
 - mouseX, mouseY
 - indeholder de aktuelle koordinater for musen
altså koordinaterne i den aktuelle frame

```
// Add and subtract to create offsets
void setup() {
  size(100, 100);
  smooth();
  noStroke();
}
void draw() {
  background(126);
  ellipse(mouseX, 16, 33, 33); // Top circle
  ellipse(mouseX+20, 50, 33, 33); // Middle circle
  ellipse(mouseX-20, 84, 33, 33); // Bottom circle
}
```

Interaktion med musen

- Muse-koordinater
 - pmouseX, pmouseY
 - indeholder foregående koordinater for musen altså koordinater fra frame'n umiddelbart før den aktuelle

```
// Draw a line between the current and previous
positions
void setup() {
  size(100, 100);
  strokeWeight(8);
  smooth();
}

void draw() {
  background(204);
  line(mouseX, mouseY, pmouseX, pmouseY);
}
```

Interaktion med musen

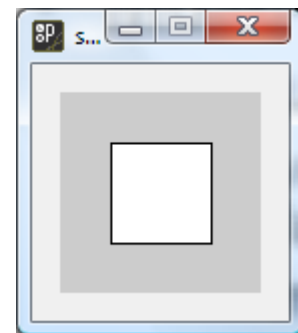
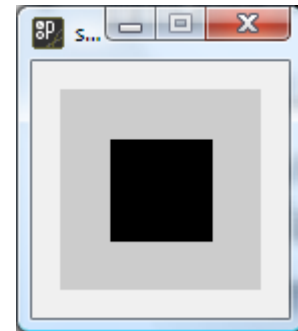
- Muse-knapper

- mousePressed

- variabel der er sand når en muse-knap er nedtrykket og ellers falsk
 - med variabelen mouseButton kan det afgøres hvilken knap der er nedtrykket

```
// Set the square to white when a mouse
button // is pressed
void setup() {
  size(100, 100);
}

void draw() {
  background(204);
  if (mousePressed == true) {
    fill(255); // White
  } else {
    fill(0); // Black
  }
  rect(25, 25, 50, 50);
}
```

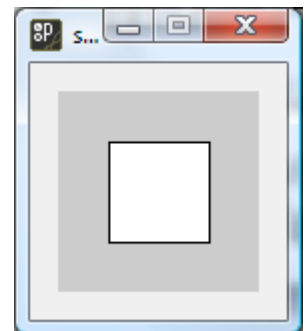
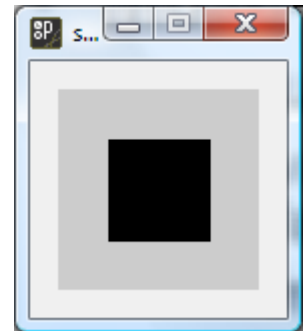
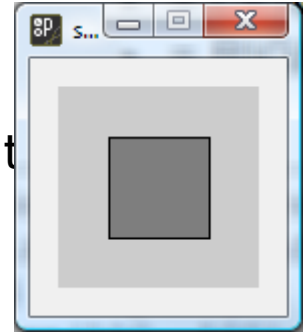


Interaktion med musen

- Muse-knapper
 - mouseButton
 - = LEFT hvis den sidst trykkede knap var venstre
 - = CENTER hvis den sidst trykkede knap var i midten
 - = RIGHT hvis den sidst trykkede knap var højre

```
// Set the square to black when the left mouse button
// is pressed and white when the right button is pressed
void setup() {
  size(100, 100);
}

void draw() {
  if (mouseButton == LEFT) {
    fill(0); // Black
  } else if (mouseButton == RIGHT) {
    fill(255); // White
  } else {
    fill(126); // Gray
  }
  rect(25, 25, 50, 50);
}
```



Kursor icon

- Kursor icon
- `cursor()`
 - sætter kursor-symbolet
 - `cursor()`
 - `cursor(MODE)`
 - `cursor(image, x, y)`
 - **MODE kan være**
 - `ARROW, CROSS, HAND, MOVE, TEXT, WAIT`
- `noCursor()`
 - fjerner kursor-symbolet

```
void draw()
{
  if(mouseX < 50) {
    cursor(CROSS);
  } else {
    cursor(HAND);
  }
}
```

Input 1 Opgaver

- Opg A
 - Skriv et program i continuous mode, der tegner en rød cirkel som følge musen og en grøn cirkel som hele tiden går i det modsatte retning af musen
- Opg B
 - Skriv en variation af programmet fra Structure 3 Opg B, som tegner figuren hvor musen er, men kun når man klikker på en af museknapperne
- Opg C
 - Skriv endnu en variation af ovenstående Opg B, hvor der fås forskellige fyldfarver afhængigt af om man trykker venstre eller højre museknap

Image 1: Billeder

- Datatype: **PImage**
 - Datatype til billeder
- Indlæsning af billede: **loadImage("xxx.ttt")**
 - indlæser billede "xxx.ttt" i en variabel af typen PImage
 - billedet skal ligge i data-folderen (træk det fx bare ind i editor vinduet)
 - Processing kan håndtere .gif, .jpg, .tga, og .png - billeder
- Display af billede: **image(x,y)**
 - viser et billede i display-vinduet med øverste venstre hjørne i (x,y)
- Eksempel

```
PImage mp;  
mp=loadImage("RUCpicture6.jpeg");  
size(mp.width, mp.height);  
image(mp, 0, 0);
```

Billeder - egenskaber

- Billedtone og gennemsigtighed
 - tint()
 - sætter en farvetone og gennemsigtighed for billeder
 - noTint()
 - slår farvetone og gennemsigtighed fra
- Billedstørrelse
 - hvis fx b=loadImage("billede1.jpg")
 - så er b's bredde og højde angivet i
 - b.width
 - b.height

```
PImage mp;  
mp=loadImage("RUCpicture6.jpeg");  
size(mp.width, mp.height);  
image(mp, 0, 0);
```

```
PImage mp;  
mp=loadImage("RUCpicture6.jpeg");  
size(mp.width+200, mp.height+100);  
image(mp, 0, 0);  
tint(220, 214, 21);  
image(mp, 100, 50);  
tint(220, 214, 21, 100);  
image(mp, 200, 100);
```

Image 1 Opgaver

- Opg A
 - skriv et program der indlæser og viser et billede
 - sæt vinduet til billedets størrelse
 - prøv at eksperimentere med farvetone og gennemsigtighed

Data 2: Tekst

- Datatype char
 - enkelttegn
 - char bogstav = 'k';
- Datatype String
 - tekst-streng fx ord og sætninger
 - String s = "dette er et eksempel";
 - en række funktioner kan bruges på String bl.a.
 - String.length(), String.startsWith(), String.toUpperCase(), String.substring()

```
String s = "dette er et eksempel";  
println(s.length());  
println(s.startsWith("dette"));  
println(s.toUpperCase());  
println(s.substring(0,12));
```

```
20  
true  
DETTE ER ET EKSEMPEL  
dette er et
```

Data 2 Opgaver

- Opg A
 - erklær to variable A og B af datatypen String og tildel dem værdierne hhv. "dette er " og "ikke et tal"
 - udskriv (med println)
 - længden af A
 - værdien af A efterfulgt af B
 - længden af A efterfulgt af B

Typography 1: Tekst som grafik

- Datatype `PFont`,
- Indlæsning af font `loadFont("fff.vlw")`,
 - fonten skal ligge i data-folderen (træk det fx bare ind i editor vinduet)
- `text(t)`
 - Visning af teksten `t` i den indstillede font
- `textFont(f)`
 - Indstiller til en bestemt font
- `textSize()`
 - sætter font-størrelsen
- `textLeading()`
 - indstiller linie mellemrum
- `textAlign()`
 - indstiller tekstjustering (muligheder: LEFT, CENTER, RIGHT)
- `textWidth(ttt)`
 - beregner bredden af teksten `ttt`

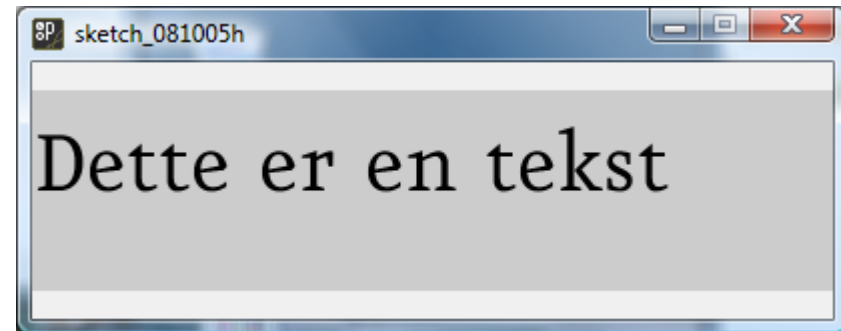
Eksempel, tekst som grafik

- Bemærk at en font altid skal ligge i programmets data-folder
(kan fx klares ved at trække font-filen hen over vinduet)

```
PFont font;
String T = "Dette er en tekst";

void setup() {
  size(400, 100);
  font = loadFont("Eureka-48.vlw");
  textFont(font, 48);
  stroke(255);
  fill(0);
}

void draw() {
  background(204);
  text(T, 0, 50);
}
```



Typography 1 opgaver

- Opg A
 - Skriv et program der får teksten "Proeve" til at glide igennem display-vinduet fra den ene ende for at forsvinde i den anden

Data 4: Arrays

- Array
 - en særlig datastruktur
 - en gruppe af elementer, der kan tilgås med indeksering
- Array - eksempel
 - `int[] A = {17,34,25,11};`
 - er et array med 4 pladser, hvor
 - `A[0]==17`
 - `A[1]==34`
 - `A[2]==25`
 - `A[3]==11`
- Længden af et array
 - `length` giver antallet af pladser i array'et
 - fx: `A.length == 4`



- Eksempel

```
int[] x = {50, 61, 83, 69, 71, 50, 29, 31, 17, 39};
int[] y = {18, 37, 43, 60, 82, 73, 82, 60, 43, 37};

beginShape();
// Reads one array element every time through the for()
for (int i = 0; i < x.length; i++) {
    vertex(x[i], y[i]);
}
endShape(CLOSE);
```

Arrays

- Et array skal
 - erklæres: hvor datatypen angives
 - oprettes: hvor antallet af pladser fastlægges
 - tildeles: hvor der sættes værdier ind på pladserne
- Kombineret erklæring, oprettelse og tildeling
 - `int[] A = {17,34,25,11};`
- Anden mulighed
 - erklæring
 - `int[] A;`
 - oprettelse
 - `A = new int[5];`
 - tildeling
 - `A[0]=17;`
 - `A[1]=34;`
 - `A[2]=25;`
 - `A[3]=11;`
- Tredie mulighed
 - man kan også kombinere erklæring og oprettelse
 - `int[] A = new int[5];`
 - tildeling
 - `A[0]=17;`
 - `A[1]=34;`
 - `A[2]=25;`
 - `A[3]=11;`

To-dimensionale arrays

- 2-dimensionelt array
 - `int[][] A = {{17,34},{25,11}}`
 - `i` stedet for

```
int[] x = {50, 61, 83, 69, 71, 50, 29, 31, 17, 39};
int[] y = {18, 37, 43, 60, 82, 73, 82, 60, 43, 37};

beginShape();
// Reads one array element every time through the for()
for (int i = 0; i < x.length; i++) {
    vertex(x[i], y[i]);
}
endShape(CLOSE);
```

- kan man

```
int[][] points = { {50,18}, {61,37}, {83,43}, {69,60}, {71,82},
                  {50,73}, {29,82}, {31,60}, {17,43}, {39,37} };

beginShape();
// Reads one array element every time through the for()
for (int i = 0; i < points.length; i++) {
    vertex(points[i][0],points[i][1]);
}
endShape(CLOSE);
```

Arrays

- Et array kan have alle datatyper
- Array funktioner
 - `append()`
 - tilføjer et element
 - `shorten()`
 - fjerner et element
 - `expand()`
 - fordobler antallet af pladser el. tilføjer antal i parameter
 - `arraycopy()`
 - kopierer fra et array til et andet
 - `concat()`
 - forlænger et array med et andet
 - `reverse()`
 - vender rækkefølgen
 - `sort()`
 - sorterer elementerne

```
int[] data = {0, 1, 3, 4};  
println(data.length); // Prints "4"  
data = expand(data);  
println(data.length); // Prints "8"  
data = expand(data, 512);  
println(data.length); // Prints "512"
```

Data 4 Opgaver

- Opg A
 - skriv en funktion `ma` der ganger to arrays og returnerer resultatet i et nyt array
 - gå ud fra at de to parametre har samme antal pladser

- Opg B
 - skriv et program der gemmer musens position i et to-dimensionelt array og gentegner musens bevægelser ved et klik på musen

Resume - Vigtigste begreber

- Variable og virkefelter, Blok (block), Globale variable
- Funktioner (egne), abstraktion, Return
- Interaktion med musen: mouseX, mouseY, pmouseX, pmouseY, mousePressed, mouseButton, cursor(), noCursor()
- Billeder, PImage, loadImage(), image(x,y)
- Tekst,
 - char, String
 - Tekst som grafik, PFont, loadFont(), text(), textFont()
- Array erklæring, oprettelse, indexering, tildeling, length

Dagens Debugging