Indhold

1 Sekventiel/procedural -programmering

Læs kapitel 3.1 i Systimebogen.
Denne opgave handler om sekventiel og procedural programmering. Sekventiel programmering, betyder at instruktionernes rækkefølge ikke er ligegyldig. Procedural programmering betyder at vi kan opdele vores program i forskellige procedurer eller funktioner og bare kalde proceduren når vi har brug for den. Det ville være smart med en procedure som kan beregne en vares pris med moms.

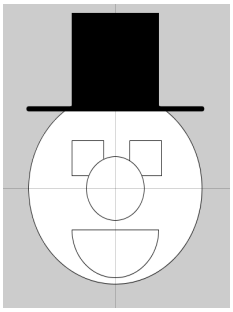
Det vigtige i denne opgave er, at du skal lære at bruge dokumentationen til processing/Java.

Herudover vil du blive introduceret til rutediagrammer. Et værktøj som skal hjælpe dig med at strukturere dine programmer.

Der er altså hele tre ting som kræver din opmærksomhed.

1.1 Opgave

Du skal lave en kopi af min tegnig: opg1-hoejhat.pdf. Det primære fokus i denne opgave er, at bruge dokumentationen i processing. Der findes i processing en lang række forud definerede procedurer/funktioner. Vi kan se at det er en funktion fordi er altid er () bagefter. For eksempel size(); Size er en funktion som kræver to parameter, brede og højde. size(400,600); Men hvordan finder man ud af hvor mange parameter og hvilke man kan bruge til en funktion? Det gør man ved at kigge i dokumentationen til processing. Du finder alle funktioner i processings reference guide: processing.org/reference.



Figur 1: Højhat

Du kan bruge disse otte instruktioner for at kunne lave tegningen.

- size();
- line();

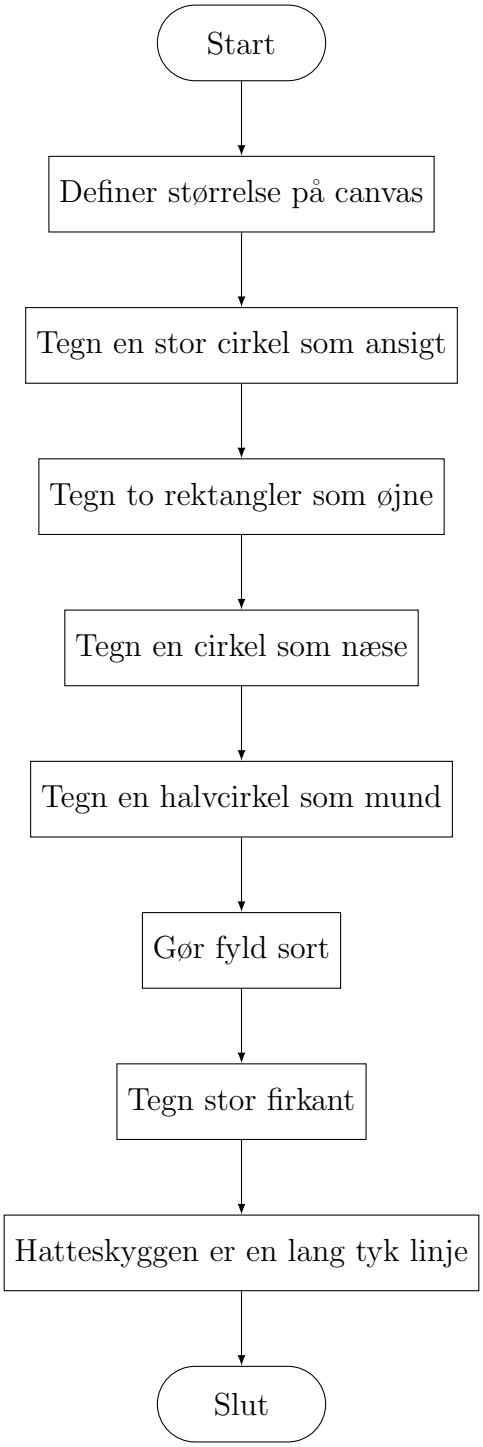
- strokeWeight();
- rect();
- square();
- circle();
- arc();
- fill();

Brug Processings dokumentation: processing.org/reference, for at finde ud af hvilke parameter de forskellige funktioner skal have.

- canvas (vindue som processing åbner) kan have størrelsen $400, 600$
- strokeWeight() er tykkelsen på stregen.
- fill() udfylder figuren med en RGB-farve.

Husk at koordinaterne $0, 0$ er øverste venstrehjørne (normalt vil det være nederste venstre) og er efter princippet: "hen ad vejen, ned til stegen". X,Y.

Brug rutediagrammet i figur 2 for at skrive koden:



Figur 2: Rutediagram

Hvad sker der, hvis du bytter om på rækkefølgen? Altså hvis du starter med øjne,næse og mund og så tegner ansigtet.

1.2 Opgave

Du har nu en fornemmelse af hvad sekventiel programmering er. Men en af styrkerne i Java er, at det understøtter produceral programmering. Hver procedure programmeres sekventielt. I Proccesing skal der altid være to procedurer. Setup og draw. Setup bruger vi til f.eks. at sætte størrelse på canvas eller andre initieringer. Draw er hoveddelen. Proceduren (vi kalder det også en funktion) looper 60 gange i sekundet. Vi kan selv definere alle de procedurer som vi har brug for.

Tilføj nu følgende linjer til dit program og flyt alt dit kode ned i proceduren/funktionen hoved.

```
void setup() {
                    size(480, 120);
}
void draw() {
                    head();
}
void head(){
                    // Din kode skal stå her! I mellem de to tuborg paranteser.
}
```

1.3 Opgave

Del dit program yderligere op. Således at én funktion tegner hatten. En anden tegner øjne, en tegner munden og den sidste tegner ansigtet. Kald funktionerne for: void hat(){} ,void eyes(){} ,void mouth(){}, void face(){}, husk at fjerne funktionen head(), når du er færdig.

1.4 Opgave

Gør nu dit canvas så stort at der er plads til to hoveder ved siden af hinanden. Tilføj to parametere til dine funktioner, en float x og en float y koordinat, udfra hvilke du kan tegne alle dine elementer af ansigtet. Ret dine linjer til så de tager udgangspunk i dine x og y koordinater. Er du rigtig god, kaler du dine funktioner med de samme værdier!

1.5 Opgave

Tegn figuren færdig med krop, arme og ben i hver deres funktion.

2 Primitiv Animation

Det vigtige i denne opgave er, at du skal lære at bruge dokumentationen til processing/Java. Herudover skal du bruge rutediagram til at strukturere din kode. Tænk på animation som stop motion teknik.

Du skal bruge nogle nye instruktioner

- frameRate()
- frameCount()
- noLoop()
- rotate()
- translate()
- popMatrix()
- pullMatrix()

Brug Processings dokumentation: processing.org/reference, for at finde ud af, hvad de forskellige funktioner gør og hvilke parameter de forskellige funktioner skal have.

2.1 Opgave

Lav et program, hvor et hjul, med minimum tre eger, ruller over skærmen fra venstre mod højre. Start med at lave et rute diagram.

2.2 Opgave

Udvid programmet så når hjulet forsvinder i højre side, dukker det op på den anden side igen.

2.3 Opgave

Lav programmet om så når hjulet rammer væggen, at det stopper og ruller tilbage hvor det kom fra.

2.4 Opgave

Lav et program som lave en primitiv animation af en mand som kan gå. Start med at lave et rute diagram.

Du kan finde inspiration i mine to eksempler på Github.

3 Løkker og Funktioner

Læs om while-løkker i kapitel 3.3 og om for-løkker i kapitel 3.4. Læs om funktioner i kapitel 3.5

Du kender nu til følgende datatyper og instruktioner:

Datatyper:

- float: kommatal.
- double: mere præcis end float.
- char: en enkelt karakter. Char er en unsigned byte og kan derfor ikke indeholde minustal.
- int: heltal.
- long: 2 gange så stor som en integer.
- boolean: sand eller falsk.

Instruktioner:

- size(); // sætter størrelsen for canvas
- line(); // tegner en linje
- stroke(); // farven på en streg
- strokeWeight(); // tykkelse af streg
- rect(); // tegner en rektangel
- circle(); // Tegner en cirkel
- arc(); // Tegner en bue
- fill(); // udfylder en figur med farve. [1]

[1] Find rgb farver her: www.rapidtables.com/web/color/RGB_Color.html

Vi skal også arbejde med funktioner. Du kender dem fra matematikken $f(x)$, hvor en funktion beregner/returnerer en værdi. Vi bruger funktioner når vi skal udføre den samme sekvens flere gange, med forskellige variabler. Funktioner i Java, består af 4 dele.

Navn: Funktioner skal navngives med et ikke reserveret ord (ord som er brugt i forvejen), men hvor navnet er sigende for funktionens funktion. I mit eksempel er navenet "minFunktion". Vi bruger navnet når vi skal bruge funktionen.

Returdatatype: Funktioner skal deklareres efter returdatatype. Funktionen kunne returnere en int, float, string eller char, men altid kun én ting. Hvis funktionen ikke returnerer noget, skal den defineres som void.

Parameter: Funktionen kan modtage en lang række forskellige parameter. En funktions paramenter skal angives i parantesen efternavenet samtidig med at vi deklarerer datatypen. Man kan deklarere mange parametere til sin funktion. Parameter er kun gyldige lokalt. Det vil sige at du kan ikke bruge en variabel du har deklareret i andre funktioner uden at skulle deklarere dem igen.

Kode: En funktions kode skal skrives imellem de to tuborgparanteser .

3.1 Opgave

Skriv et program hvor dit canvas er 800x800. Opret en funktion cirkel, som tegner en cirkel på dit canvas. Cirklen skal være rød. Ryd op efter dig. Navngiv din funktion så den fortæller hvad funktionen gør. Opret en funktion som tegner en firkant. Firkanten skal være blå. Navngiv din funktion så den fortæller hvad funktionen gør.0

3.2 Opgave

Tilpas din funktion så den tegner 8 cirkler. Brug et for loop og brug dit index i til at gange din x koordinat i cirklen, således at din cirkel ikke bliver tegnet det samme sted.

```
for (del 1; del 2; del 3) {
// Den sekvens af kode som løkken skal udføre
}
```

Et for-loop består af tre dele. Del 1 er en deklarering og initiering af vores index. Vi kalder index for i. Har vi brug for flere indlejret løkker følger vi

alfabetet og kalder den næste j så k etc. Del 2 er den betingelse som skal være opfyldt for at løkken bliver udført. Betingelsen knytter sig typisk til vores index i. For eksempel: i¡=10. Del tre beskriver den handling som skal ske med i efter hver iteration. Vi kan tælle i op med 1; i++. Eller trække en fra i–.

```
for(inti = 0; i < 10; i + +){
System.out.println(i);
}
```

Figur 3: For-løkke

3.3 Opgave

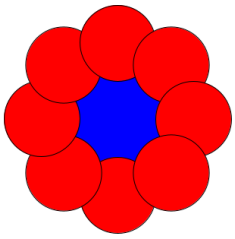
Tilpas din funktion firkant så den tegner et antal firkanter vertikalt. Hvor mange firkanter kan du få plads til?

3.4 Opgave

Omskriv din for-løkke til en while-løkke.

3.5 Opgave

Lave en blomst af cirkler. Placer en cirkel i midten og lav kronbladene af 8 cirkler. Du kan måske bruge funktionerne pushMatrix(), popMatrix, translate() og rotate()?



Figur 4: Blomst

4 Betingelser/Forgreninger

Læs om Betingelser/forgreninger i kapitel 3.2 i Systime bogen.

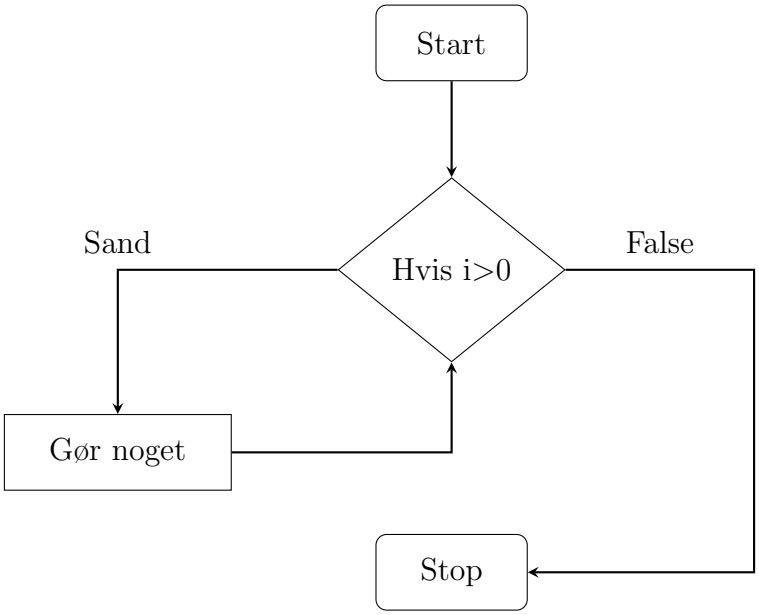
4.1 Operatorer til sammenligning

- Mindre end: $a < b$
- Mindre end eller lig med: $a <= b$
- Større end: $a > b$
- Større end eller lig med: $a >= b$
- Er lig med: $a == b$
- Forskellig fra: $a! = b$

4.2 Aritmetiske operatorer

- + Addition Lægger to værdier sammen. X+Y
- - Subtraktion Trækker to værdier fra hianden x - y
- * Multiplikation Ganger to værdier x * y
- / Division Dividerer en værdi med den anden x / y
- % Modulus Returnere resten ved division x
- ++ Inkrement Øger værdien af en variabel med 1 ++x
- – Dekrement Mindsker værdien af en variabel med 1 –x

```
if (betingelse) {
// Den sekvens af kode som løkken skal udføre hvis betingelsen er sand.
}
```



Figur 5: Rutediagram løkke

5 Ikke primitive datatyper

Vi skal i dette kapitel arbjede med ikke primitive datatyper. Det gælder for de primitive datatyper at de kan repræsenteres i en byte. Ikke primitive datatyper kræver meget mere plads. Primitive datatyper er foruddefineret i java, det er ikke primitive datatyper ikke, på nær String. Ikke primitive datatyper kan bruges til at kalde funktioner og bestemte operationer.

Den ikke primitive datatype String, er en række af karrakterer (Chars). Der er til datatypen knyttet en række metoder:

- toUpperCase() Converts all of the characters in the string to uppercase
- toLowerCase() Converts all of the characters in the string to lowercase
- substring() Returns a new string that is a part of the original string
- length() Returns the total number of characters included in the String as an integer number
- indexOf() Returns the index value of the first occurrence of a substring within the input string
- equals() Compares two strings to see if they are the same
- charAt() Returns the character at the specified index