

Processing.org workshops

Workshop 2

Open Space Aarhus

Bryggervej 30, 8240 Århus N

26. november 2011



Dagens program

- Introduktion
- Hvad er et partikelsystem
- Opsummering variable
- Kode
 - Datatyper
 - Forgreninger
 - Løkker
- Workshop
 - Bold
 - Kanon kugle
- *Afslutning*

Hvad har de flittige lavet

- `http://www.openprocessing.org/classrooms/?classroomID=1075`

<http://vimeo.com/28256186>

<https://www.youtube.com/watch?v=ncykt-YJ01M>

Først tager vi lige en hurtig og tildels teoretisk gennemgang af centrale begreber fra sidst + det nye vi skal bruge i dag.

Derefter skal vi har *beskidte fingre*.

Slides og processing filer

<http://poodle/processing>

Slidsne kan sikkert bruges til at kigge i eller kopiere fra.

Variable - *erklæringer*

Der er indbyggede variable som

- `mouseX`
- `mouseY`
- `width`

Du kan *erklære* din egne variable

- `int x;`
- `float y;`

Erklæring af variable

```
datatype navn;  
datatype navn = startværdi;
```

Variable - *tildelinger*

Du kan *tildele* en værdi til variable.

- `x = 42;`
- `y = 3.14;`

OBS datatyper

```
int x;
```

```
x = 3.14
```

x er nu 3, fordi den automagisk laver den om til et heltal.

~~x = "noget tekst"~~ **Boom**

Datatyper

int

Heltal (integer) : 1, 2, 42

```
int x = 42;
```

float

komma tal: 3.14×10^{42}

```
float y = 3.14;
```


Datatyper - lidt andre

double

ligsom float, bare flere decimaler

```
double z = 8.92838429338;
```

char

En *byte*(0-255) - kan gemme eet bogstav.

```
//OBS ' og ikke "  
char c = 'X';
```

String

Tekst stykker: "I'm a string"

```
String hello = "world";
```


Sammenligninger

Operatorer

Lighed	==
Ikke ens	!=
Større end	>
Mindre end	<
Større end eller lig	>=
Ditto for mindre	<=

```
int x = 42;
```

```
int y = 0;
```

```
boolean foo = (x == y);  
//false
```

```
boolean bar = (x >= y);  
//true
```

Boolske udtrykke

Operatorer

Dansk	teknisk	kode
og	and	&&
enten	or	
Ikke	not	!
enten-eller	xor	^

```
boolean glad = true;  
boolean sur = false;
```

```
boolean meh = (glad || sur);  
boolean godEksempel = !sur;
```

If-else-blokke

```
if (boolean) {  
  // do stuff  
} else {  
  // do something else  
}
```

```
if (x < 200) {  
  fill(255, 0, 0);  
}
```

```
// if-else block  
if (x < 200) {  
  fill(255, 0, 0);  
} else if (x < 300) {  
  fill(0, 255, 0);  
} else {  
  fill(0, 0, 255);  
}
```

Løkker - *while*

looping

```
while (boolean) {  
  // keep doing stuff  
}
```

```
int x = 0;  
while (x < width) {  
  point(x, 100);  
  x++;  
}
```

Løkker - for

for loop

Ligesom *while* løkke, men tit vil vi gerne bruge en tæller så

start initialisering

betingelse Hvor længe skal vi blive ved

pr-gang gør noget for hvert gennemløb

```
//( start; betingelse; pr-gang)
for(int x = 0; x < width; x++){
    point(x, 100);
}
```

Online kodetræning, gode små opgaver til aftenkaffen.

CodingBat code practice

[about](#) | [help](#) | [done](#) | [prefs](#) | [create acc](#)

Java > Warmup-1 > sleepIn

[prev](#) | [next](#) | [chance](#)

The parameter weekday is true if it is a weekday, and the parameter vacation is true if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return true if we sleep in.





sleepIn(false, false) → true
sleepIn(true, false) → false
sleepIn(false, true) → true

Go

...Save, Compile, Run

Show Solution

```
public boolean sleepIn(boolean weekday, boolean vacation) {  
    return weekday || vacation;  
}
```

Expected	This Run	
sleepIn(false, false) → true	false	X 
sleepIn(true, false) → false	true	X 
sleepIn(false, true) → true	true	OK 
sleepIn(true, true) → true	true	OK 

[Progress Graph](#) for this problem **new**

Figur: Eksempel fra <http://codingbat.com/java/Warmup-1>

Mission bold

Mission bold

- Gem retning i sin egen variabel
- Opdater position baseret på retning
- Skift retning når bolden rammer kanten

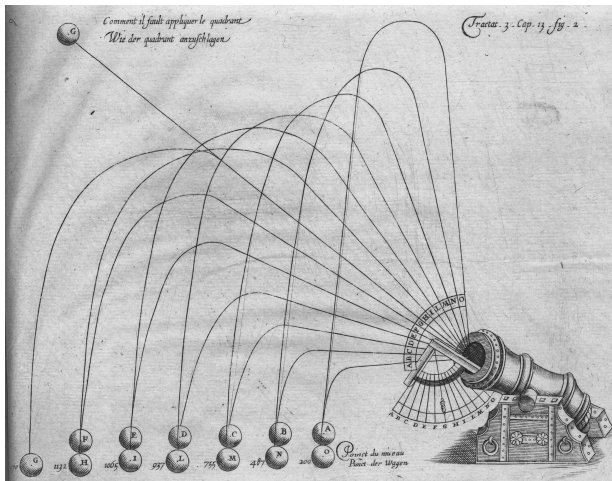
```
float boldX = 200;
```

```
float boldY = 200;
```

```
float deltaX = 2.3;
```

```
float deltaY = 1.3;
```

```
if ( boldX > width) {  
    deltaX = -deltaX;  
}
```

Figur:

$$s = \int_{t_1}^{t_2} ds = \int_{t_1}^{t_2} \sqrt{dx^2 + dy^2 + dz^2} = \int_{t_1}^{t_2} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2} dt$$

Aaargh matematik og formler

På en eller anden facon er ting der opfører sig naturligt pæne.
Fysik og matematik forsøger at beskrive naturen ...

Computergrafikkens 1. lov

Hvis det ser rigtigt ud, er det rigtigt.

Man kan udregne bevægelser udfra de fysiske love og opstille ganske komplicerede ligningsystemer.

Man kan også lave noget der ligner ret godt.

Gnidningsmodstand

Bolden

delta er hastighed.

Vi ændre hastigheden for at efterligne kræfter.

Gnidnings modstand

Vi gør hastigheden mindre og mindre pr frame(tidsskridt).

Det gøres ved at gange delta med et tal tæt på een.

```
float modstand = 0.996;  
deltaX *= modstand;  
deltaY *= modstand;
```

Tyngdekraft

Tyngdekraft

Påvirker hastigheden så ting falder ned, det vil sige deltaY bliver større.

```
deltaY += 0.1;
```

Start bold og tilføj vindmodstand og tyngdekraft

```
float boldX = width/2;
float boldY = 2.0 * height/3;
float deltaX = 2;
float deltaY = -2.5;
float modstand = 0.996;
float tyngdekraft = 0.1;

void draw() {
    deltaY += tyngdekraft;
    deltaX *= modstand;
    deltaY *= modstand;
    boldX += deltaX;
    boldY += deltaY;
    //Check grænser
    //Tegn bold
}
```


At bruge OpenGL til at renderere betyder at arbejdet med at tegne sker på grafik kortet.

```
import processing.opengl.*;

void setup() {
  size(400,400, OPENGGL);
  //mere init
}
```

Tegne

En simpel kanon er et løb, altså en rektangel.

Men løbet skal jo drejes?

Transformationer

`translate` og `rotate` skal angives rigtig rækkefølge, og fortæller beskriver hvor og hvorledes de efterfølgende ting tegnes.

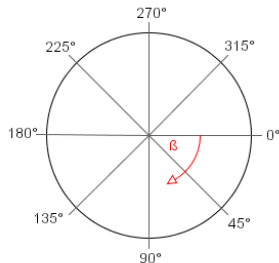
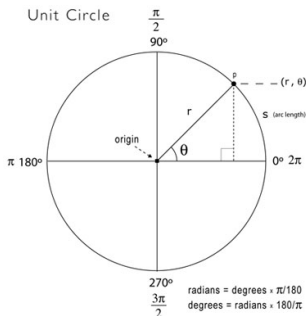
```
//husk radianer!  
float cannonA = -PI/4.0;  
// tegn kanonen  
translate(0, 400);  
rotate(cannonA);  
rect(0, -10, 50, 20);
```

Tech Tip: Radianer

Vinkler angives i intervallet $[0 - 2\pi]$, den omvendte y-akse gør at vinklerne kører med uret, modsat normal trigonometri.

radians(A) Fra grader til radianer

degrees(A) Fra radianer til grader



Figur: <http://processing.org/learning/trig/>

Figur: <http://btk.tillnagel.com/tutorials/rotation-translation-matrix.>

Keyboard input

```
void keyPressed()
```

Funktionen bliver kaldt, når en tast bliver tastet.

Processing har indbyggede variable:

key char der beskriver tasten

keyCode indbygget variable som : LEFT, RIGHT, UP etc

```
void keyPressed() {  
  if (keyCode == LEFT) {  
    // drej kanon mod uret  
  } else if (keyCode == RIGHT) {  
    // drej kanon med uret  
    cannonA += .05;  
  } else if (key == ' ') {  
    // sæt start position  
    // sæt start hastighed  
  }  
}
```

Kollisions detektion

Ramte jeg noget?

Vi er heldige at vore målskive er rund.

d = afstand mellem kugle og målskive.

hvis $d < radius_{skive} + radius_{kugle}$ har vi ramt!

Afstand

processing funktion `dist(x1, y1, x2, y2)`

ellers $d = \sqrt{(x_1 - x_2)(x_1 - x_2) + (y_1 - y_2)(y_1 - y_2)}$

```
float d = dist(kugleX, kugleY, targetX, targetY);  
if (d < 30) {  
    //vi ramte!!  
}
```

Ændr modstand og tyngdekraft

- alpha blanding: lad vær med at tegne baggrund og brug alpha
- tegn en streg fra sidste position til nuværende
- tegn anderledes kugle
- lad kuglens form og farve afhænge af dens *alder*
- lad musen tiltrække bolden
- multiplayer: flere kanoner + skyd hinaden
- blæsevejr
- tæl points
- ram noget andet

Tak for i dag

- Hvad syntes *du* om i dag?
- Næste gang: flere bolde
- T^3 i må meget gerne hjælpe med at rydde lokalet.

Klasseværelset

www.openprocessing.org/classrooms/?classroomID=1075