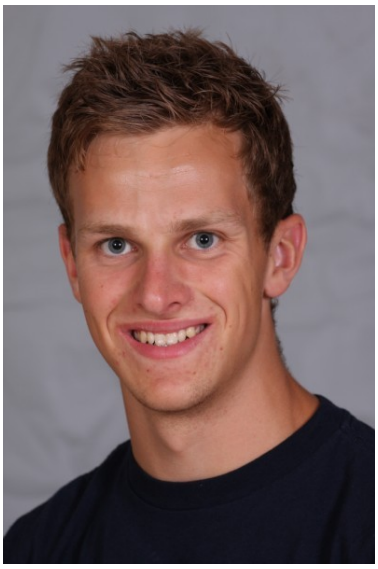
Mads Friis Bornebusch, s123627
Kristian Sloth Lauszus, s123808
Simon Patrzalek, s123401

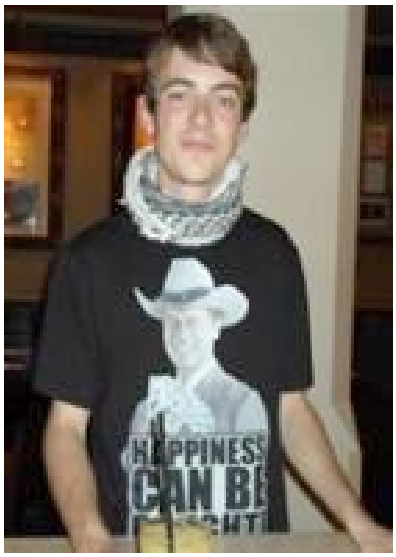
17. March 2013

Afleveringsopgave 3

02102 Indledende Programmering

Mads Bornebusch s1233627 ***Kristian Sloth Lauszus s123808*** ***Simon Patrzalek s123401***





Rapporten og øvelsen har været et samarbejde mellem Mads Friis Bornebusch, Kristian Sloth Lauszus og Simon Patrzalek.
Opgaverne blev lavet individuel, således hver gruppemedlem fik maksimal udbytte af opgaver. Samtidig var der udveksling af ideer og foreslag til forbedrelse løbende. Hvert gruppemedlem formåede at lave alle opgaver.

Mads Friis Bornebusch, s123627
Kristian Sloth Lauszus, s123808
Simon Patrzalek, s123401

17. March 2013

Opgave 1

Denne opgaves formål var at lave en funktion som kunne lave primtalsfaktorisering, dvs. den skulle kunne tage et heltal som input og printe de primtal som input kunne faktoriseres til. Dette blev gjort med en while-loop inde i en for-loop. Den matematiske operation indenfor disse loops så således ud:

```
private static String getPrimeFactors(long n) {
      String output = "";
      for (int i = 2; i <= n; i++) {
            while (n % i == 0) {   // Check if the remainder is 0
                  output += Integer.toString(i) + ",";
                                    // Add the number to the string
                  n /= i;
            }
```

Testkørsler for PrimeFactors
Følgende inputs blev brugt for at teste PrimeFactors:
11111111111
212121
Hey World

Output af PrimeFactors.java:

```
Enter integer greater than 1 (0 to terminate): 11111111111
List of prime factors: 21649,513239
Enter integer greater than 1 (0 to terminate): 212121
List of prime factors: 3,3,7,7,13,37
Enter integer greater than 1 (0 to terminate): Hey World
Please type a valid integer!
```

På grund af output fra testkørslen konkluderes der at opgave er løst, da PrimeFactors giver de primtal som input kunne faktoriseres af, samt at funktionen kommer med en fejlmeddelse når input ikke er et heltal.

Opgave 2

Formålet af denne opgave var at skrive en kode som ville simulere en ”gåtur”, hvor det kun er muligt at bevæge sig ét skridt mod nord, syd, øst eller vest ad gangen. Sa. Valget af retning skulle være tilfældigt og gåturen måtte først stoppes når randen af ”kortet” (grid) var nået. Størrelsen af kortet skulle være et brugerinput. Desuden skulle der ved hjælp af StdDraw laves en grafisk afbildning af gåturen.

Ganske enkelt blev der genereret et tilfældigt tal som afgjorde hvilken vej det næste skridt ville være. Nedenunder ses den del af koden som bestemte hvilken vej det næste skridt skulle være

```
while (Math.abs(x)<grid && Math.abs(y)<grid){
```

```
double randDir = Math.random()*4;

if(randDir<1){
      y--; //south
}else if (randDir<2){
      x--; //west
}else if (randDir<3){
      y++; //north
}else if (randDir<4){
      x++; //east
}
count ++;
System.out.println("Position: ("+x+","+y+")");
StdDraw.point(x,y);
```

RandomWalk gav ofte ganske kønne figurer, hvis man er til abstrakt kunst

Illustration 1: Grid size = 40

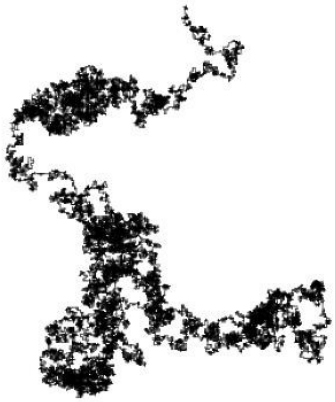


Illustration 2: Grid size = 200

Som det ses ud fra ovenstående figurer gav RandomWalk det forventede output og derfor må det konkluderes at RandomWalk var en succes.

Opgave 3

I denne opgave skulle vi lave et reel spil. Dette spil skulle være Racetrack, også kendt som Vektor Rally. Ganske simpelt går det ud på at man har et punkt, som er racerbilen. Denne kan bevæge sig på banen, samtidigt med at den accelererer eller bremser. Dette er det vigtigste aspekt af Racetrack, da racerbilen har inerti. Hvis den bevæger sig for hurtigt mod en væg kan man ikke undgå en kollision. Spillet foregår trinvis, således man skal overveje sin bevægelse for hvert trin for at komme hurtigst i

mål.

Aspektet med interti realiseres ved at bilen har en hastighedsvektor. For hvert trin kan bilen accelerere eller bremse ved at forøge eller formindske den vektor som angiver bilens hastighed til tiden t. Bilens hastighedsvektor består af en x- og y-koordinat. For hvert trin kan ændringen af hvert koordinat være 1, 0 eller -1.

Selvom opgaven lød ganske simpel var der ganske mange måder at løse den på. Koden blev til sidst ganske stor, hvilket betyder at der ikke her kan gengives alle finesserne som blev brugt, uden at gøre denne rapport ulæselig og fuld af kode-stumper. For at se hvad der udgjorde koden, må man se .java-filerne. Her vil vi fokuserer på tilføjelserne. De udvidelser vi har tilføjet er:

- Styring med mus
- Highscore-liste
- Detektering af hvilken vej bilen kører
- Detektering af at bilen kører i mål samt antal træk spillet er gennemført på
- Reset-knap
- Tilføjelser i StdDraw-biblioteket

Muligheden for at styre med mus blev realiseret med følgende kode:

```
while(!StdDraw.mousePressed());                    // Wait for button press
                int corX = (int)Math.round(StdDraw.mouseX());
                                                   // Read mouse position
                int corY = (int)Math.round(StdDraw.mouseY());
                while(StdDraw.mousePressed());     // Wait for release
```

Koden muliggjorde brug af mus som styring. Der var stadig mulighed for at bruge keyboard som input. Dette ville have krævet at man aktiverede den del af koden som tillod dette. Denne del er stadig bibeholdt sidst i koden.

Detektering af bilens retning fungerede på den måde at, når bilen har krydset hvad man kan kalde ”checkpoints”, kan den først gå i mål.

```
if(!middleCrossed && pos[X] > size/4 && pos[X] < size/4*3 && pos[Y] <= size/4) {
                                        System.out.println("Middle crossed");
                                        middleCrossed = true;
                                        clockwise = oldPos[X]-pos[X] > 0; // This
is used so the car can go in both directions
                                    }
                                    else if(!sideCrossed && middleCrossed &&
pos[Y] >= size/4*2 && ((clockwise && pos[X] <= size/4) || (!clockwise && pos[X] >=
size/4*3))) {
                                        System.out.println("Side crossed");
                                        sideCrossed = true;
                                    }
                                    else if(sideCrossed && (clockwise && pos[X] >=
size/2 ||  !clockwise && pos[X] <= size/2) && pos[Y] >= size/4*3 && pos[Y] <= size)
{
                                        System.out.println("Goal!");

                                        goal = true;
                                    }
```

Dette betyder at man kan vinde ved at køre begge veje, men man skal krydse bl.a. et checkpoint, som lå på halvvejen.

Reset-knappen

En anden tilføjelser var reset-knappen, som tillod at man kunne spille flere gange. Der var ikke implementeret muligheden for at registrere omgange, da dette virkede lidt meningsløst i et spil, hvor det gælder om at komme igennem banen med det mindste antal træk som muligt. Reset-knappen betød at man ikke skulle genstarte spillet for at prøve igen. Selve knappens grafik blev lavet med StdDraw-biblioteket. Nedenstående beskriver reset-knappens funktion:

```
if(corX >= resetButton[0]-resetButton[2] && corX <= resetButton[0]+resetButton[2]
&& corY >= resetButton[1]-resetButton[3] && corY <= resetButton[1]+resetButton[3])
{
                        System.out.println("Reset pressed");
                        init(); // Reset all values and draw course
```

Mads Friis Bornebusch, s123627
Kristian Sloth Lauszus, s123808
Simon Patrzalek, s123401

17. March 2013

Highscore-listen

Highscorelisten skulle virke ved at highscoren blev gemt i en fil på computeren. Denne fil skulle loades under initialiseringen af racerbanen. En del af koden der kører under initialiseringen er vist herunder:

```
File inputFile = new File("topscore.txt");
                if(!inputFile.exists()){
                        PrintStream output = new PrintStream(new File("topscore.txt"));
                        for (int i=0; i<topscore_length; i++)
                                output.print(",999;");
                        output.close();
                        loadFile();
                } else
                        loadFile();
```

Denne kode undersøger om der findes en highscore-fil i den mappe programmet køres fra. Hvis dette er tilfældet kaldes en metode der loader filen. Hvis dette ikke er tilfældet genereres der en fil der indeholder et tomt navn og scoren 999 det antal gange som der er pladser på highscorelisten. Antallet af pladser på highscorelisten er bestemt af klassekonstanten *topscore_length*. Når denne fil er oprettet loades den.

Den score der opnås efter et spil sammenlignes med det array der indeholder highscorerne der er loadet fra highscorefilen. Hvis man derfor opnår en score på 999 vil man derfor ikke komme på den automatisk genererede ”tomme” highscoreliste. Når man opnår en highscore kommer der en inputdialog frem hvor man skal indtaste sit navn. Derefter kommer der er dialog frem, der viser highscore-listen og den nye highscoreliste skrives til highscorefilen. De automatisk genererede highscores på 999 sorteres fra, før listen vises til brugeren. Den liste der vises er formateret med html tags for at ændre skrifttype og –størrelse.

Et eksempel på en highscorefil gemt på computeren er vist nedenfor:

MadsBornebusch,26;SimonPatrzalek,26;KristianLauszus,28;DonaldDuck,124;,999;,999;,999;,999;,999;

Vi bruger en regular expression til at fjerne alt fra indtastningen der ikke er A-Z, a-z eller 0-9. Alle mellemreum bliver derfor fjernet. Som det kan ses ovenfor er navnene separeret fra scorerne med et komma og de forskellige topscorer er separeret fra hinanden med semikolon. Når filen loades bliver dette skilt ad og lagt ind i arrays’ene *topNames[]* og *topScores[]* i et for-loop der løber hele strengen fra filen igennem.

Udvidelsen med at man kan styre med musen gør det muligt at køre filen uden konsollen. Filen kan derfor eksporteres og køres som .jar. Denne fil er vedlagt. Det giver dog ikke helt samme resultat at køre .java filen i eclipse og at køre .jar-filen. Vi har bemærket den forskel at titlen på highscore-dialogboksen står i tekstfeltet når den køres fra eclipse og i headeren når den køres som .jar.

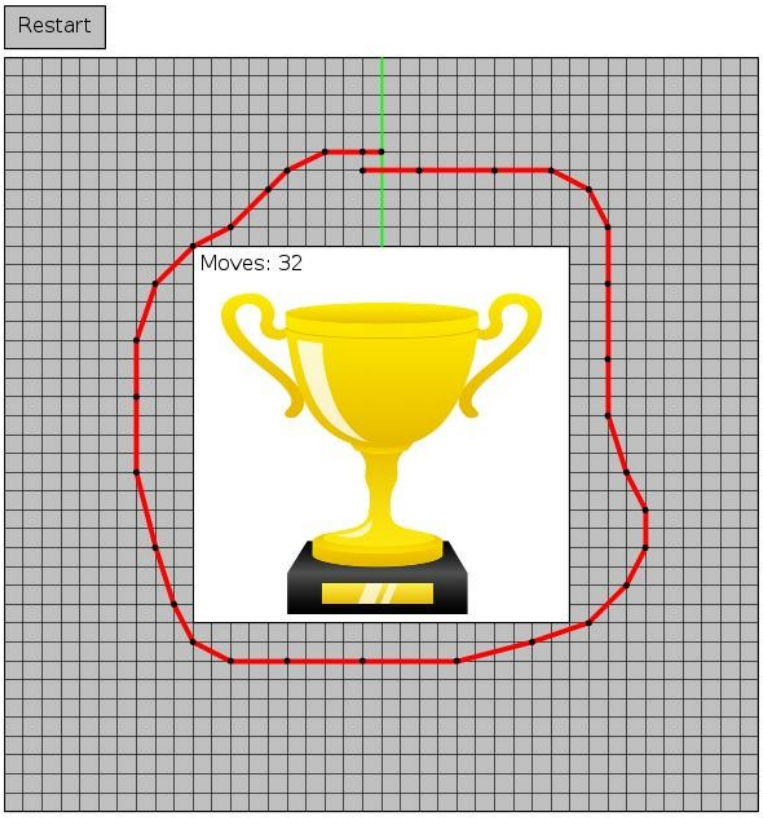


Illustration 3: Eksempel på en omgang med RaceTrack. Her har brugeren vundet