Programmering B Rapport

ELEV: Nelisa Chehrehgosha og Mie Norre Engemann  
FAG: Programmering B  
KLASSE: 2.R  
SKOLE: Aarhus Gymnasium C  
VEJLEDER: Mirsad Kadribasic  
DATO FOR AFLEVERING: 2/05/2022

Indholdsfortegnelse

[1 Abstract 3](#_Toc102375413)

[2 Problemforulering 4](#_Toc102375414)

[3 Programmet 5](#_Toc102375415)

[library 5](#_Toc102375416)

[3.1 design 6](#_Toc102375417)

[3.1.1 flowchart 7](#_Toc102375418)

[3.1.2 pseudokode 8](#_Toc102375419)

[3.1.3 GUI formelknapper 8](#_Toc102375420)

[3.1.4 scrum 9](#_Toc102375421)

[3.2 dokumentation 9](#_Toc102375422)

[3.2.1 kode kommentar 10](#_Toc102375423)

[3.2.2 kode stil 11](#_Toc102375424)

[3.2.3 funktion beskrivelser 12](#_Toc102375425)

[3.2.4 klasse beskrivelse og diagram 14](#_Toc102375426)

[3.3 deling 15](#_Toc102375427)

[3.3.1 github 15](#_Toc102375428)

[4 Test 16](#_Toc102375429)

[5 Konklusion 17](#_Toc102375430)

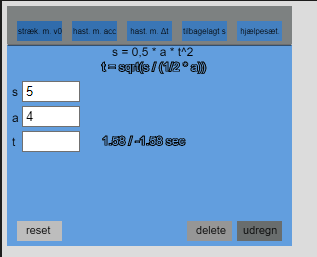
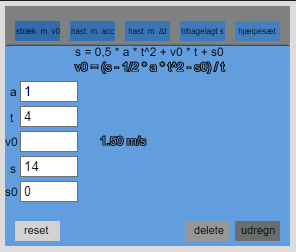
[6 Bilag 17](#_Toc102375431)

# 1 Abstract

For denne opgave er der lavet et program, som led i programmeringseksamen i programmering B. Programmet er skrevet i JavaScript på p5.js.

Programmet er en formelregner, som udregner fysik formler omhandlende bevægelse ud fra de inputs den får af brugeren.

Programmet er målrettet opgaveregning og undervisningsbrug. Programmet regner først og fremmest ens opgaver ud fra givet værdier, men det giver også brugeren den isoleret formel, som så giver brugeren et indblik i hvad der bliver regnet og hvorfor man får det svar man gør. Tanken er at man lærer lidt hver gang man bruger formelregneren ud fra et matematisk udgangspunkt, ved at man kan se sammenhængen mellem formel og isoleret formel, hvilket også kan hjælpe en med bedre at forstå fysikken bag.

Nedenunder er vist et skærmbillede fra programmet, som viser hvordan svaret vil se ud på en af formlerne med vilkårlige indsatte værdier.  
 

# 2 Problemforulering

Meget fysikforståelse og mange fysikopgaver bygger på et fundament af matematik. På den måde kan en god matematik indsigt hjælpe med en bedre forståelse af fysikken, og også til regning af fysikopgaver er denne indsigt nødvendig. Men siden man husker og lærer bedst ved at øve sig, så må det mest optimale være at få matematik og fysisk forståelsen til at arbejde sammen i konkrete opgaver.  
Hvordan laver man et program som formår dette uden det bliver nytteløst, når man først har forstået fysikken og matematikken bag?  
Og hvordan gør man samtidig programmet forståeligt nok for modtageren, så de ikke skal bruge alt deres energi på at forstå selve programmet?

Disse forudsætninger er afgørende for om programmets kerne lykkes.

# 3 Programmet

Programmet består af en menu, hvori man kan vælge imellem de forskellige formler inden for fysikemnet ’bevægelse’ vha. knapper, samt et ”regnefelt”, hvor man kan indskrive sine kendte værdier og få givet den isoleret formel og beregnede manglende værdi.  
Til højre vises et FlowChart for den ikke-interaktive del af programmet, altså hvad der sker når programmet startes.

Det består af tegning af menuen med sine formelknapper og tegning af ”regnefeltet” med reset-, delete- og udregningsknap.

## library

For at kunne importere lyde er der brugt library. Lyde kan bruges på mange forskellige måder alt efter hvilket program det er. Lyde er blevet brugt til alle knapperne i dette program, og de afspilles, når der trykkes på knapperne. Vi deler de fire forskellige knapper op i fire klasser ud fra, hvad de laver. Hver slags knap har egen lyd, som hænger sammen med knappens funktion, eksempelvis vil nogen kunne kende lyden for delete-knappen fra MacBooks.  
Der er fire lyde i alt. For formlerne, skal man vide om der bliver klikket på, derfor blev der brugt en grundlæggende keyboardlyd. Udregn-knappen skal have en dejlig lyd for at indikere, at løsningen er blevet beregnet. Vi brugte en reverse lydeffekt til reset-knappen for at gøre det klart, hvad den gør.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Text

Description automatically generatedLaver en funktion for lyd

Kalder mySoundUdregn i udregn klassen Kalder mySoundDelete i delete klassen

Text

Description automatically generatedText

Description automatically generated

*Graphical user interface, text

Description automatically generated*Text

Description automatically generatedKalder mySoundReset i reset klassen Kalder mySoundKnap i button klassen

## 3.1 design

### 3.1.1 flowchart

Diagram

Description automatically generated EXAMPLE, IT’S THE FLOWCHART FROM LAST TIME   
Figur 1. Flowchart af project (macro)

## 3.1.2 pseudokode

## 3.1.3 GUI formelknapper

Graphical user interface(GUI) program er et computerprogram. Brugere kan få adgang til funktionerne og oplysningerne i et computerprogram gennem graphical user interface. GUI bruges normalt i operativsystemer og installeret software. GUI bruger grafiske komponenter som ikoner, billeder og knapper til at skabe en model for programbrugere til at interagere med en applikations funktionalitet.

Det er en af de vigtigste i programmet og vi har brugt det meget i vores program. Her er en gennemgang af hvordan vi brugt dem på menuen nederst.

Knapperne bliver også mørkere når man er over dem

Graphical user interface, application

Description automatically generated

## 3.1.4 scrum

Her er vores tidsplan over formelregner lavet med Trello.

<https://trello.com/b/XEB5Xn0L/formelregner>

Graphical user interface, application

Description automatically generatedFigur 2. Billede af vores tidplan fra Trello.

## 3.2 dokumentation

## 3.2.1 kode kommentar

Til venstre er et eksempel på, hvordan vi har brugt kommentarer i dette projekt til at informerer om, hvad de forskellige kommandoer gør. Når man samarbejder på et programmeringsprojekt, er det ofte forvirrende og måske endda kaotisk at se på et stykke kode for første gang som en anden person har skrevet. Det er derfor vi bruger kommentarer.

I gennem vores kode vi har brugt kommentarerne til at forklarer hvad bestemte linjer eller funktioner gør, og hvilken rolle de spiller i vores projekt. De forklarer nogle gange konkret hvad en linje bestemmer, men andre gange giver de bare et overblik over et interval eller forskellige variable som har betydning for udkommet.

Det meste af tiden har vi lavet en kommentar øverst i koden, hvor vi forklarer, hvad hele denne if-sætning handler om, og hvad den bruges til.

Til sidst har vi også nogle enkelte steder udkommenteret flere linjer kode, fordi vi ikke skulle bruge dem i det øjeblik, men de kunne være gode at have senere, hvis man for eksempel ville ændre noget.

DER SKAL VÆRE ET BILLEDE AF KODEN TIL VENSTRE

## 3.2.2 kode stil

Til venstre er et eksempel på, hvordan vi har brugt kommentarer i dette projekt til at informerer om, hvad de forskellige.

Det gjorde vi både i vores klasser og i sketch fillen. Dette hjælper os med at finde det vi ønsker at rette eller forbedre senere.

DER SKAL VÆRE ET BILLEDE AF KODEN

## 3.2.3 funktion beskrivelser

Paint programmet benytter sig af flere funktioner. En af funktionerne i klassen ’Button’ hedder ’knap’ og bruges bl.a. til at lave vores GUI (*graphical user interfave*) som ses ved programmets knapper.

Når musen holdes over en knap, så vil knappens farve blive lidt mørkere. Bliver der trykket på knappen, så vil der tegnes en endnu mørkere rektangel ovenpå, som så vil forsvinde igen ved et nyt tryk på samme knap.  
Netop den sidste del med hvad der sker når man trykker på en knap, handler nedenstående Nassi Schneider’s funktions diagram og flowchart om.

Diagram

Description automatically generatedEXAMPLE FROM LAST TIME  
Figur 5. Nassi Schneider funktions diagram over hvad  
der sker når der trykkes på en knap i funktionen ’knap’

EXAMPLE FROM LAST TIME

* Definer skærm
* Laver objekter dvs. kalder klasser (sender værdier for rgb, x,y-placering til klasse og tykkelse)
* Kalder funktioner fra klasse (funktion for indramning af menu, knapper og tykkelse)
* Funktion for mouseKlikket  
   if (mus er trykket på)  
   kald ’klikket’-funktion fra klasse for knap-objekterne

Klasse

* constructor  
   omdan værdier sendt fra set up til lokale værdier i klassen
* Funktion for knap  
   if(mus er inden for knap)  
   skalerer rgb-værdierne ned  
   else  
   brug rgb-værdierne sendt med fra set up  
    
   tegn rektangel  
    
   funktion for hvis der er klikket  
   if(mus er inden for knap)  
   sæt boolean-variable til true  
   return boolean-værdien  
    
   if(mus er inden for knap for forstørrelse af tykkelse)  
   kald funktion ’tykkere’  
    
   if(mus er inden for knap for mindskelse af tykkelse)  
   kald funktion ’tyndere’  
    
   if(boolean-variable er true)  
   skalerer rgb-værdierne ned  
   tegn ny rektangel over den gamle  
   funktion for hvis der er klikket  
   if(mus er inden for knap)  
   sæt boolean-variable til false  
   return boolean-værdien
* Funktion for linje  
   laver linje tykkelse  
   if(mus er før menu)  
   farv linje  
   lav linje
* Funktion for tykkelse på menuen  
   lav rektangel  
   skriv linje tykkelsen på rektanglet  
   skriv + og - på rektangelet to sider
* Funktion ’tykkere’  
   læg 0,5 til linjens tykkelse
* Funktion ’tyndere’  
   træk 0,5 fra linjens tykkelse
* Funktion ’tyndere’  
   træk 0,5 fra linjens tykkelse
* Funktion for indramning af menu  
   farv  
   lav rektangel

## 3.2.4 klasse beskrivelse og diagram

|  |
| --- |
| Button |
| -r: int -g: int -b: int -xPlus: int  -xPlus: int -strokeSize: int |
| #knap()  #klikket()  #vis(in strokeSize1 : int, in strokeSize2 : int)  #tykkelse()  #tykkere()  #tyndere()  #cubes() |

EXAMPLE FROM LAST TIME

## 3.3 deling

## 3.3.1 github

For at vi kunne arbejde sammen brugte vi GitHub til at dele opdateringer af de ting, vi har ændret i koden. Dette hjalp os til at kunne arbejde separat og nemt dele det op mellem os uden at skulle bekymre os om at noget kode vil gå tabt.

A screenshot of a computer screen

Description automatically generated with medium confidence

# 4 Test

# 5 Konklusion

# 6 Bilag