Programmering C

Eksamensprojekt

Karla Jacobsen 18xaa

VEJLEDER:MARK MOORE

DATO: 11/05-2020

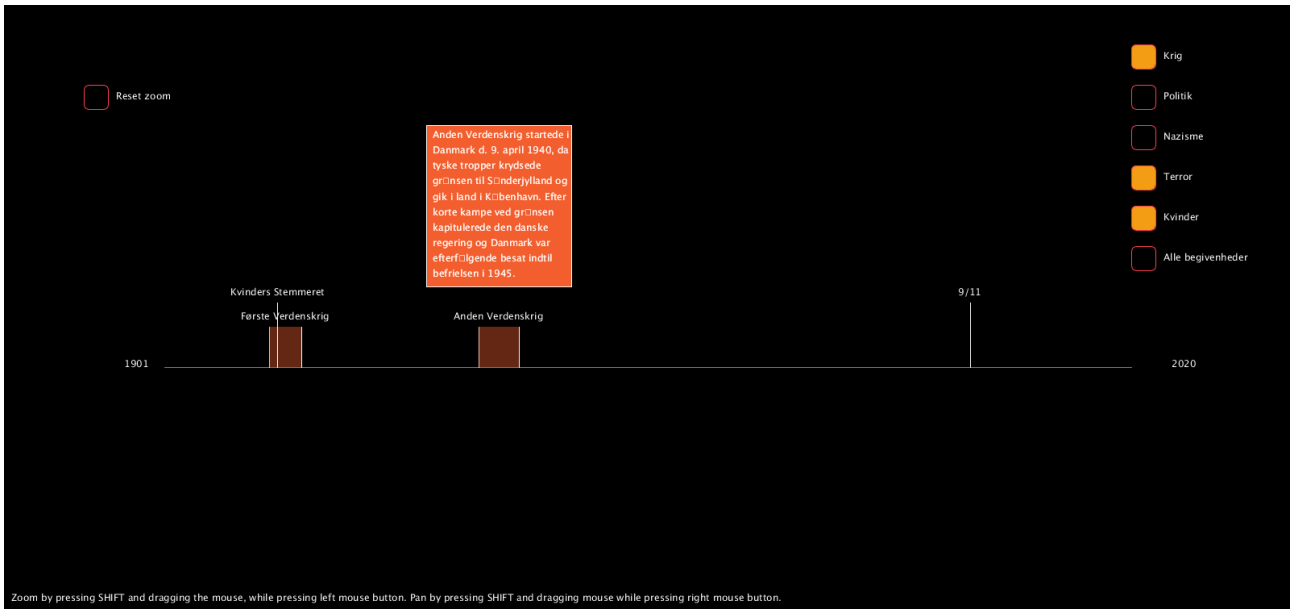
Indholdsfortegnelse

Abstract

Denne opgave er lavet af Karla Jacobsen som eksamensprojekt i programmering C i 2.g på Aarhus Gymnasium, Aarhus C. Programmet er lavet i Processing 3.5.4 (Processing Development Environment (PDE)).

Programmet er en interaktiv tidslinje over Danmarkshistorien fra 1901 til i dag. Der kan zoomes ind og ud og der kan ved hjælp af knapper sorteres i hvilke begivenheder, man som bruger, gerne vil se ud fra forskellige tags, der hører til hver begivenhed. Det er også muligt for brugeren at trykke på de enkelte begivenheder og på den måde få mere information om den pågældende begivenhed.

Herunder ses et skærmbillede fra programmet, der giver et hurtigt overblik over de forskellige dele og funktioner.



Billede 1: Skærmbillede af programmet i brug.

Problemformulering

Det kan være svært at få overblik over de vigtige historiske begivenheder, der har formet Danmark gennem tiderne. Et værktøj, man kan bruge til at gøre det nemmere, er en tidslinje. Dog er mange af de tidslinjer man kan finde i dag enten nogle man selv skal lave, eller også giver de kun det store eller det lille overblik og ikke begge dele. Derfor vil et program, der viser en tidslinje over historiske begivenheder i Danmark, som er interaktiv og kan zoome både ind og ud, være en god løsning på dette problem. Den vil give elever både det store og det lille overblik over hvilke begivenheder, der formede vores samfund hvornår. Et sådant program ville også kunne sortere i begivenhederne og f.eks. kun vise de begivenheder, der handler om krig, teknologi eller noget tredje. Der er dog nogle problemstillinger, der skal løses, for at dette kan udføres. De vigtigste af dem ses herunder:

- Hvordan gøres programmet overskueligt og brugervenligt?
- Hvordan zoomes der bedst ind og ud, så det ikke ødelægger overblikket?
- Hvordan gøres sorteringsmekanismen så smooth og god som muligt?

Programmet

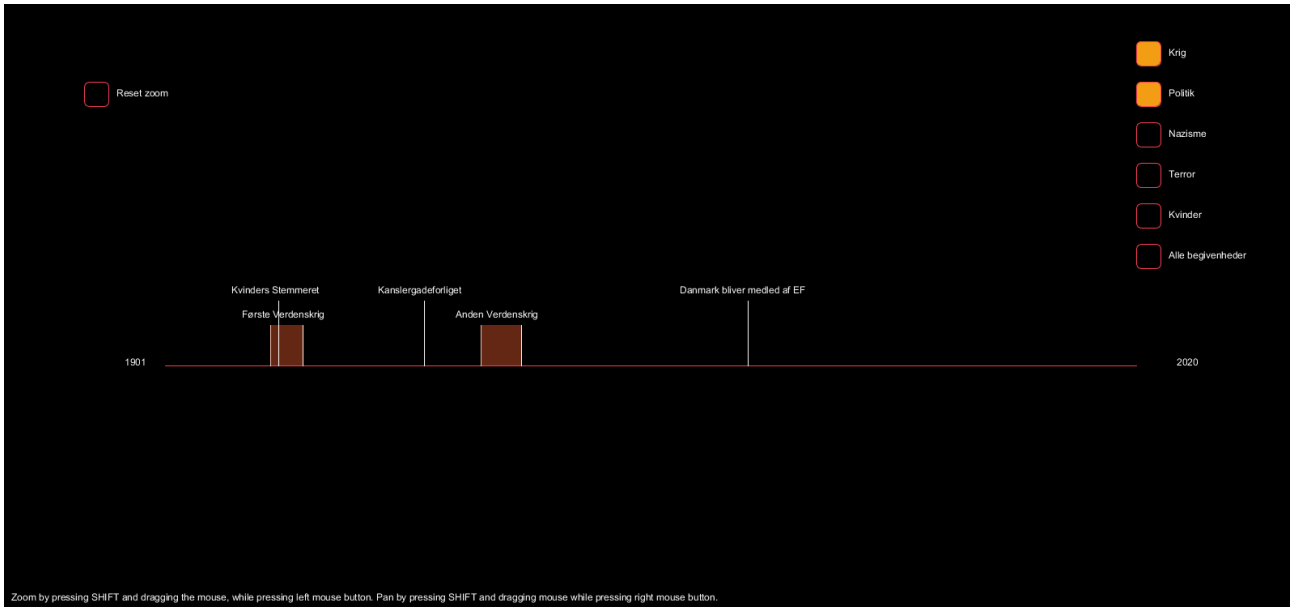
I dette afsnit vil selve programmet beskrives. Der vil bl.a. kommes ind på hvad brugeren ser når han/hun bruger programmet, programmets objekter og funktioner og programmet syntaks både på det overordnede og på underordnede niveauer.

GUI

I dette afsnit beskrives den del af programmet som brugeren ser og interagerer med. Dette kaldes Graphical User Interface, GUI. I dette program kan GUI'en deles ind i fire dele: Selve tidslinjen, der kan zoomes ind og ud omkring, knapperne i højre side, hvor brugeren kan styre, hvilke begivenheder de gerne vil se, begivenhederne, der kommer frem når brugeren trykker på knapperne og informationsboksene der kommer frem når brugeren trykker på de forskellige begivenheder. Når brugeren åbner programmet, vil det se ud som på billedet herunder:

Billede 2: GUI'en når brugeren åbner programmet.

Brugeren vil altså se en tom tidslinje og knapper i øverste venstre hjørne, som brugeren kan klikke på for at sortere i begivenhederne. Klikker brugeren på f.eks. knapperne krig og politik, vil der på tidslinjen fremkomme de begivenheder, der indeholder dette tag. Se billede herunder:



Billede 3: Skærmbillede af GUI'en efter brugeren har trykket på knapperne "Politik" og "Krig".

Har brugeren lyst til at zoome ind og ud omkring tidslinjen gøres dette ved tryk på SHIFT og trækning af musen op for at zoome ind og ned for at zoome ud. Brugeren kan også panorere. Dette gøres ved tryk på SHIFT og trækning af musen med tryk på højre museknap. I bunden af skærmen ses en tekst, der forklarer brugeren, hvordan zoom og panorering virker. Ved tryk på knappen "reset zoom" i øverste højre hjørne, kan brugeren komme tilbage til udgangspunktet efter at have zoomet, hvis brugeren har lyst til det.

Events

Den vigtigste del af de fire dele af programmet er de historiske begivenheder, da det er dem hele programmet drejer sig om, og det er det, som brugeren gerne vil have ud af programmet. For at lave begivenhederne er der blev brugt Objekt Orienteret Programmering også bare kaldet OOP. OOP er en rigtig god metode at bruge, hvis man skal lave mange af de samme objekter, der dog alligevel skal være forskellige. Dette gør man ved at oprette en Class og under denne class oprette forskellige objekter. I dette program hedder classen for historiske begivenheder *event* og under den laves forskellige historiske begivenheder ud fra forskellige variabler. I *constructoren* opstilles de variabler, som er klassens input parametre, der skal gives når de enkelte objekter oprettes i *void draw().* For de historiske begivenheder er disse variabler navnet

```
event (String _title, StringList _tags, int x, int z) {
  //Udregning af årstal i pixels: (år - 1901)*10 + 200 fx. 1945 = (1945-1901)*10+200 = 640
  //Dette gørres sådan at programmet selv regner pixel-positionen ud
  //Så kan man, når man laver eventet som objekt bare skrive startår og slutår direkte ind
  this.x1=x=(x-1901)*10+200; //
  this.y1=y1_1;
  this.x2=x1;
  this.y2=y2_1;
  this.x3=z=(z-1901)*10+200;
  this.y3=y1_1;
  this.x4=x3;
  this.y4=y2_1;
  this.tags=_tags;
  this.title=_title;

  //Hvis begivenheden kun varer 1 år gøres den højere,
  //så begivenhederner ikke ligger ind over hinanden
  if (x==z) {
    y2=y2-30;
  }
}
```

Figur 1: Billede af koden, der udgør constructoren til klassen event.

på begivenheden, start og slutåret for begivenheden (varer begivenheden kun et år, sættes startåret og slutåret til det samme) og de tags der hører til begivenheden. For at undgå at navnet på begivenheder, der kun varer et år ikke kommer til at stå oveni begivenheder, der varer flere år, ændres y-positionen for toppen af den linje der danner begivenheden, hvis startåret og slutåret er det samme. På den måde flyttes navnet for disse begivenheder op over navnet for de andre begivenheder.

Position og arrayList

For at gøre det så nemt som muligt for programmøren at tilføje og overskue nye og eksisterende events, laves nye events ved at tilføje dem til arraylisten events. På den måde er alle events det samme sted og kan nemt hives ud af arraylisten ved at bruge eventets position i listen. For at gøre det nemt at finde ud af hvilke begivenheder, der har hvilke positioner sorteres eventsne efter startår (x1-position). Se figur 2 og 3. For at gøre det endnu nemmere for programmøren at tilføje events, regner programmet selv pixel-værdien ud for et givent årstal mellem 1901 og 2020. Dette gøres vha. udregningsmodellen:

$$Pixelposition = (\text{år} - 1901) * 10 + 200$$

Eksempel med årstallet 1945:

$$Pixelposition = (1945 - 1901) * 10 + 200 = 640$$

Det skal dog siges at denne udregningsmodel kun gælder når tidslinjen har længden 1200 pixels.

```
//Sortering og sammenligning
@Override
  int compareTo(event other) {
  return this.x1 - other.x1;
}
```

Figur 3: Koden til sammenligningen af eventsne fra Java i klassen event,

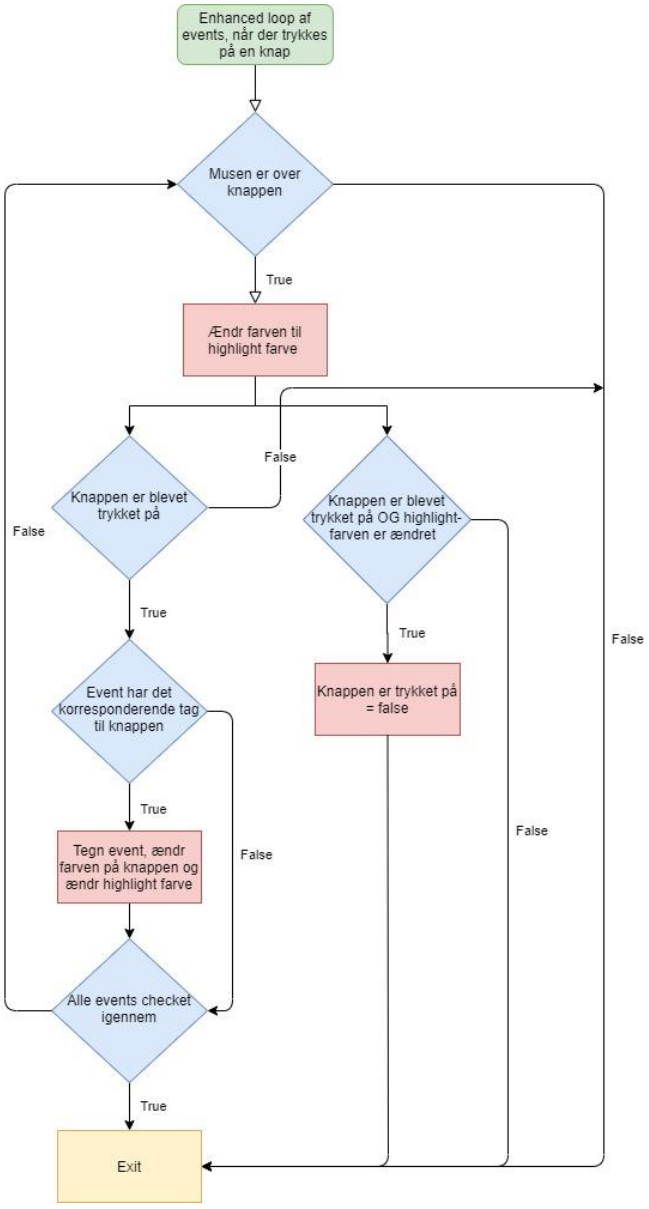
```
Collections.sort(events);

println(" ");
println("After sorting");
for (event e : events) {
  println(e.title);
}
```

Figur 2: Koden til sorteringen af eventsne efter startår i void draw().

Sorteringsmekanismen

I dette afsnit bliver sorteringsmekanismen og de knapper, brugeren skal trykke på for at sortere gennemgået.



Figur 4: Flowchart der viser, syntaksen for sorteringsmekanismen, når der trykkes på en knap.

```
//knap politik
if (polKlik) {
  if (i.tags.hasValue("Politik")) {
    //noLoop();
    i.tegnEvent();
    fill (pressColor);
    polColor = pressColor;
    polHighlight=51;
  }
```

Figur 5: Kode, der viser hvordan *hasValue()* bruges til at sortere eventsne efter tags.

Knapperne

Knapperne ses som sagt i øverste højre hjørne og i øverste venstre hjørne. Knappen i øverste venstre hjørne er ”reset zoom” knappen og den vil blive beskrevet i afsnittet Zoom. I dette afsnit er der fokus på de resterende knapper og deres funktion. Hver knap, der ses i højre side, repræsenterer et specifikt tag. Når brugeren trykker på knappen, vises der på tidslinjen de events, der indeholder dette tag. Knapperne består af en firkant, med runde hjørner og et navn. Når musen befinder sig over firkanten, farves denne en Highlight farve, sådan at brugeren kan se at hans/hendes mus er over knappen.

Sortering

Når brugeren så trykker på knappen, går sorteringen i gang. Processen for dette kan ses på figur 4. Ved hjælp af booleans tjekkes det først om musen er over knappen. Vha. if-sætninger gøres det sådan, at hvis dette er tilfældet, farves knappen highlightColor. Derefter tjekkes det om knappen er blevet trykket på og hvis den er, itereres der over alle events i arraylisten med et enhanced loop, og hvis disse indeholder det korresponderende tag til knappen, tegnes de. Det er den indbyggede boolean *hasValue(value)*, der bruges til dette. Den kan sige om en stringList indeholder en bestem String. For et eksempel på hvordan dette bruges i programmet se figur 5. Når brugeren derefter gerne vil fjerne de tegnede events igen, gøres dette ved endnu et tryk på knappen. Efter knappen er blevet trykket på første gang ændres highlight farven. Dette gør, at man ved at sige at ”Knappen er trykket på” skal være true og ”highligt farve = den nye farve” skal være true, kan gøre den boolean, der tegner selve eventet i void Draw(), false igen og dermed fjernes eventet.

Det er altså bl.a. knapperne, der gør programmet interaktivt, hvilket var et af målene i problemformuleringen. Brugeren kan altså selv bestemme hvilke begivenheder han/hun vil se udfra hvilke tags, der falder i hans/hendes interesse. Der findes også en ”Alle begivenheder” knap, der, hvis man trykker på den, viser alle begivenheder på tidslinjen.

Infobokse

Når brugeren så har fundet et event der er interessant, kan han/hun trykke på eventet og der vil derefter poppe en infoboks op, hvori der står information om det pågældende event. Disse infobokse er lavet som en klasse, der hedder info. At tjekke om eventet er blevet trykket på fungerer præcis ligesom ved knapperne. De parametre der bestemmer, hvor der kan trykkes for at få infoboksen frem bestemmes bare af eventets variabler i stedet for variabler, der er lavet specielt til knappen, ligesom det er tilfældet med de andre knapper. Disse parametre hentes ud af arraylisten med events vha. den indbyggede funktion *get()*. Der vil selvfølgelig være et problem for de events, der kun varer 1 år og dermed kun er en streg, da det af den grund kan være svært for brugeren at ramme præcist på linjen med musen. Derfor blev der lagt 10 pixels til på hver side af eventet, sådan at området man skal ramme bliver større.

Billede 4: Eksempel på infoboks. Som det ses, er Æ, Ø og Å byttet ud med små firkanter.

Når man så har trykket på eventet, popper infoboksen altså frem. Den tekst, der står i boksen hentes af programmet fra eksterne tekstfiler (.txt). Dette gør det nemt at tilføje, fjerne og ændre i teksten uden at man skal ind og rode i koden. Når Processing loader tekstfiler på den her måde, loades de som et array af Strings. Dette duer ikke når man gerne vil skrive dem som tekst i GUI’en, da den funktion, der gør dette ikke tager imod et array af Strings som input parameter. Derfor benytter man den indbyggede funktion *join(String[], seperator)* til at samle arrayet til én String, der kan bruges som input parameter i funktionen *text()*. Desværre kan Processing ikke finde ud af at læse *Æ*, *Ø* og Å, og disse bliver derfor vist som små firkanter i stedet for. Dette kan virke lidt forstyrrende, men det er heldigvis forholdsvis nemt at læse teksten alligevel.

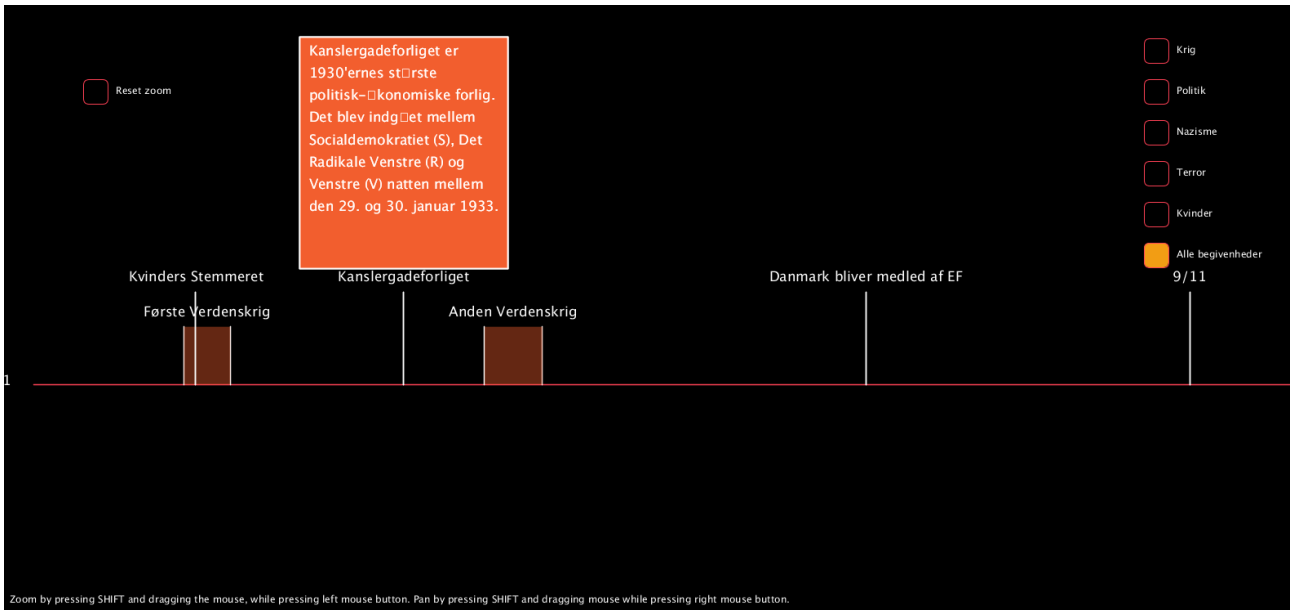
Nogle af teksterne er skrevet af mig (Karla), mens andre er hentet fra Danmarkshistorien.dk og modificeret af mig. Kilder til disse kan ses under kildelisten.

Der var et problem med at nogle af infoboksene overlappede og at man på den måde ikke kunne læse nogen af dem. Dette problem blev løst ved vha. booleans at sige, at den ene infoboks ikke kunne vises, hvis den anden blev vist. Det blev gjort på en sådan måde, at når man som bruger trykker på det ene event kommer infoboksen frem for dette event, mens den forsvinder for det andet event, der lå oveni.

Zoom og panorering

Den sidste del af programmet er zoom og panorering. Denne funktion er tilføjet fordi en af hoveidéerne med dette program er, at man som bruger får en fornemmelse af hvilke historiske begivenheder, der skete hvornår, og hvordan de indbyrdes ligger i forhold til hinanden i det rigtige målestoksforhold. Derfor er det også vigtigt at zoom ikke ændrer på begivenhedernes position, men blot ændrer på hvor (0,0) er, og på den måde viser brugeren tidslinjen endnu tættere på, sådan at han/hun både kan få det store og det lille overblik.

Zoom er i dette program blevet lavet vha. et library, der hedder giCentre.[1] Skaberne bag dette har lavet et library man kan hente og på den måde kan man bruge den zoom-funktion de allerede har lavet. Til denne zoom-funktion findes også en del metoder, der gør at man kan modificere zoomfunktionen sådan at den passer til ens program. Der findes bl.a. *setMouseMask()*, der gør at man kan bestemme at zoom kun sker, hvis der trykkes på musen, men samtidig trykkes på en bestemt tast. Denne tast kan være SHIFT, ALT eller CONTROL. Dette gøres sådan, at der ikke zoomes og panoreres når musen bruges til andre ting så som knapperne i dette program.



Billede 5: Skærmbillede der viser GUI'en efter brug af zoom.

Herover ses et skærmbillede, hvor der er blevet zoomet ind. Som det ses, er det kun selve tidslinjen, eventsne og infoboksene, der zoomes på og ikke knapperne og den forklarende tekst i bunden. Dette gøres vha. af *pushMatrix()* og *popMatrix()*. *PushMatrix()* gemmer den aktuelle transformationsmatrix og *popMatrix()* finder denne frem igen. På den måde kan man styrer hvilke scopes man gerne vil have bliver transformeret, altså zoomet på, og hvilke der ikke gør.

For at gøre det nemt for brugeren at få programmet tilbage til udgangspunktet og altså få (0,0) tilbage til øverste venstre hjørne bruges metoden *reset()*, der er en indbygget metode i zoom fra giCentre. I øverste venstre hjørne ses knappen ”reset zoom”. Denne knap virker præcis ligesom alle de andre knapper og når man trykker på den, resettes zoom vha. en if-sætning.

[1] (Wood & Slingsby, 2016)

Test af programmet

I dette afsnit testes programmet og der er også et underafsnit hvori, der kommer forslag til hvordan programmet kan forbedres, hvis det skulle videreudvikles.

Da programmet skal bruges til undervisning, skal det kunne køre på mange forskellige computere med forskellige skærmstørrelser og forskellige arbejdshastigheder. Det skal også kunne opereres af mange forskellige mennesker. Derfor er programmet blevet testet på flere forskellige computere og af forskellige mennesker (dog er det kun blevet testet på computere fra min egen husholdning og af medlemmer af min egen familie grundet covid-19).

Efter disse tests blev der fundet et problem med, at computere med en mindre skærm, end den programmet blev udviklet på. Da selve tidslinjen er 1200 pixels lang i x-retningen og den først starter 200 pixels inde på skærmen i x-retningen, gør dette, at selve vinduet er blevet gjort 1400 pixels langt i x-retningen. Dette betyder at vinduet ikke kan være på mindre skærme og på disse vil vinduet derfor forkortes, hvilket resulterer i at højre side af programmet, altså knapperne, ikke kan ses og dermed kan hele programmet ikke bruges.

Et andet problem, der kom frem under testene var, at zoom og panorering var svært at operere specielt, hvis man kun havde en mousepad og ikke en reel mus. På mousepadden var det specielt svært at panorere, da det var svært at ramme højre musetast og samtidig trække og holde SHIFT nede.

Under testene var der dog derimod ingen problemer med om de forskellige computerne kunne køre programmet. Det er også et ret simpelt og enkelt program, der ikke kræver så meget computerkraft og der var dermed heller ikke forventet nogle problemer i forhold til det.

Idéer til videreudvikling

Der var altså nogle problemer, der kom frem i lyset under testene og disse skal selvfølgelig løses, hvis programmet skal videreudvikles og forbedres. Nogle idéer til hvordan disse problemer kan løses og formindskes, samt andre idéer til hvordan programmet kan videreudvikles, sådan at det bliver et endnu bedre værktøj i undervisningen, bliver gennemgået i dette afsnit.

Problemet med skærmstørrelsen kan løses ved at gøre skærmen mindre, dog skal man huske at tilpasse alle variablerne for tidslinjen og specielt eventsne til dette, da positionen af disse er afhængig af positionen og længden af tidslinjen.

Problemet med zoom om panorering, skal løses ved at finde en ny metode til at styre zoom. Den mest intuitive måde, som alle kender og som fungerer på mousepad er at bruge to fingre til at zoome ind og ud. Denne metode er dog ikke kompatibel med giCentre's zoom og der ville dermed skulle findes en anden metode til zoom.

Som programmet er lige nu, fungerer det altså som det skal, hvis man har en reel mus og en skærm der er stor nok. Dog er der selvfølgelig altid ting der kan videreudvikles på og nogle idéer til disse gennemgås herunder.

Som programmet er nu viser det kun Danmarkshistorien fra 1901 til i dag. For at programmet skal blive så brugbart som muligt skal det vise så meget af historien som muligt, med så mange historiske begivenheder som muligt. Derfor ville en udvidelse af programmet i den forstand være en god idé.

En anden idé til noget, der kunne udvide programmet, er at lave en ”læs mere” knap ved hver infoboks, sådan at brugeren kan trykke på denne og blive sendt videre til en hjemmeside, hvor han/hun kan læse mere om den pågældende begivenhed.

Når det kommer til selve koden, ville det også være en idé til videreudvikling at lave en klasse, hvor knapperne kunne være i. På den måde ville det være nemmere at tilføje nye knapper.

Konklusion

Det kan altså konkluderes at, der er blevet lavet et program, der kan hjælpe elever og andre interesserede med at få et godt overblik over hvilke historiske begivenheder, der har formet Danmark fra 1901 til i dag.

Programmet er interaktivt og kan zoome ind og ud og panorere omkring begivenhederne og brugeren kan sortere i hvilke begivenheder, der vises, ved at trykke på knapperne og altså sortere eventsne efter de tags, der hører til dem. Zoom og panoreringsdelen er hentet fra librariet giCentre.

Programmet er blevet udviklet med fokus på OOP og det er derfor forholdsvist nemt for andre at arbejde videre på det og udvikle det sådan at det bliver endnu bedre end det er nu. Der er nemlig nogle få problemer, men ellers er det et vellykket program, der kan det det skal, men giver massere plads til at andre kan modificere og forbedre det, hvis de har lyst.

Kildeliste

Johansen, A. H. (3. juli 2017). *1. Verdenskrig og Danmark 1914-918.* Hentet 11. maj 2020 fra danmarkshistorien.dk: https://danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/1-verdenskrig-og-danmark-1914-1918/

Kold, L. F. (6. maj 2015). *kanslergadeforliget 1933.* Hentet 11. maj 2020 fra danmarkshistorien.dk: https://danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/kanslergadeforliget-1933/

Olesen, B. (23. januar 2017). *Kvindelig valgret 1849-1915.* Hentet 11. maj 2020 fra danmarkshistorien.dk: https://danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/kvindelig-valgret-1849-1915/

Olesen, T. B. (7. januar 2013). *Danmark i EF: 1973-1993.* Hentet 11. maj 2020 fra danmarkshistorien.dk: https://danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/danmark-i-ef-1973-1993/

Rasmussen, M. M. (18. april 2012). *Sikkerhed og udviklingspolitik efter 2001.* Hentet 11. maj 2020 fra danmarkshistorien.dk: https://danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/sikkerhed-og-udviklingspolitik-efter-2001/

Wood, J., & Slingsby, A. (6. februar 2016). *giCentre Utilities*. Hentet 11. maj 2020 fra gicentre.net: https://www.gicentre.net/software#/utils/

Bilag (dette hele læses oppefra og ned)

Main kode

```
import java.util.*;//Importerer java-libraries til comparison
import org.gicentre.utils.move.*; //importerer library til zoom

ZoomPan zoomer; //initialiserer zoom

//Herunder initialiseres og defineres nogel af de variabler, der bliver brugt i programmet

//Tidslinjens variabler:
int tidslinjeX1 = 200; //start
int tidslinjeY = 450; //Y-position
int tidslinjeX2 = 1400; //slut


//Variabler til events
int y1_1=450; //Øverste y-position for events
int y2_1 = 400; //Nederste y-position for events

//Definerer arraylisten med events
ArrayList<event> events;


//Variabler til infoboksene
boolean ToVKover =false; //1. verdenskirg
boolean ToVKKlik;
color ToVKcolor = #F29D14;

boolean EnVKover; //2. verdenskrig
boolean EnVKKlik;
color EnVKcolor = #F29D14;

boolean kansOver = false; //Kanslergadeforliget
```

```
boolean kansKlik;
color kansColor = #F29D14;

boolean KSover = false; //kvindernes stemmeret
boolean KSklik;
color KScolor = #F29D14;

boolean EFover = false; //DK i EF
boolean EFklik;
color EFcolor = #F29D14;

boolean sepOver = false; //9/11
boolean sepKlik;
color sepColor = #F29D14;

//Variabler til de forskellige knapper:

//reset zoom
int zoomX, zoomY;
boolean zoomOver=false;
color zoomColor=0;
color zoomHighlight=#F29D14;
String zoom = "Reset zoom";
boolean zoomKlik;
String [] zoomTekst;

//show/hide all events
int allX, allY;
boolean allOver=false;
color allColor=0;
color allHighlight=#F29D14;
String all = "Alle begivenheder";
```

```
boolean allKlik;

//krig
int krigX, krigY;
boolean krigOver = false;
color krigColor=0;
color krigHighlight=#F29D14;
String krig = "Krig";
boolean krigKlik;

//politik
int polX, polY;
boolean polOver = false;
color polColor=0;
color polHighlight=#F29D14;
String pol="Politik";
boolean polKlik;

//Nazisme
int nazX, nazY;
boolean nazOver = false;
color nazColor=0;
color nazHighlight=#F29D14;
String naz = "Nazisme";
boolean nazKlik;

//Terror
int terX, terY;
boolean terOver = false;
color terColor =0;
color terHighlight=#F29D14;
String ter = "Terror";
```

```
boolean terKlik;

//kvinder
int kviX, kviY;
boolean kviOver = false;
color kviColor =0;
color kviHighlight=#F29D14;
String kvi = "Kvinder";
boolean kviKlik;

//Variabler der er ens for alle knapper:
int SizeX = 30;
int SizeY= 30;
int round = 7;//gør at knapperne får runde hjørner
color pressColor=#F29D14; //den farve knapperne får, når der er blevet trykket på dem


void setup() {
  size(1600, 750); //(1600, 900)
  smooth();

  //Zoom
  zoomer = new ZoomPan(this); //zoomer ved at holde venstre musetast + SHIFT inde, pan ved at holde højre musetast og SHIFT inde
  zoomer.setMouseMask(SHIFT); //Gør sådan at man også skal trykke på shift for at zoome, så kan man bruge musen til andre ting også

  //ArrayList med events. Her kan man tilføje flere events
  events = new ArrayList<event>();
  events.add(new event("Anden Verdenskrig", new StringList("Krig", "Nazisme"), 1940, 1945));//andenVK
  events.add(new event("Første Verdenskrig", new StringList("Krig"), 1914, 1918));//firstVK
  events.add(new event("9/11", new StringList("Terror", "USA"), 2001, 2001));//twintowers
  events.add(new event("Kvinders Stemmeret", new StringList("Kvinder", "Stemmeret", "Politik"), 1915, 1915));//kvindeStem
  events.add(new event("Kanslergadeforliget", new StringList("Politik", "Danmark", "Økonomi"), 1933, 1933));//Kanslergade

  events.add(new event("Danmark bliver medled af EF", new StringList("Politik", "Danmark", "Europa", "økonomi"), 1973, 1973));//DKEF

  //Her sorteres eventsne i arrayet sådan at de er sorteret efter startår for begivenheden (hvilket svarer til x1-værdien)

  Collections.sort(events);

  println(" ");
  println("After sorting");
  for (event e : events) {
    println(e.title);
  }
}

void draw() {

  background(0);

  //Teksten der fortæller om hvordan man zoomer
  zoomTekst = loadStrings("zoom.txt");
  String zoomtex = join(zoomTekst, " ");
  fill(255);
  textAlign(TOP);
  text(zoomtex, 10, height-10);

  pushMatrix(); //der sætten en push og pop matrix udenom de ting, der skal zoomes
  // så man kan vælge hvilke objekter der skla zoomes på.
  zoomer.transform(); //her enables zoom

  //Her tegnes selve tidslinjen
  stroke(255, 66, 86);
  strokeWeight(1);
  line (tidslinjeX1, tidslinjeY, tidslinjeX2, tidslinjeY);
```

```
fill(255);
//Her tegnes teksten i enderne
text("1901", tidslinjeX1-50, tidslinjeY);
text("2020", tidslinjeX2+50, tidslinjeY);

//Herunder gives variablerne for knapperne værdier
krigX=1400;
krigY= 50;

polX=krigX;
polY=krigY+50;

nazX=krigX;
nazY=polY+50;

terX=krigX;
terY=nazY+50;

kviX=krigX;
kviY=terY+50;

allX=krigX;
allY=kviY+50;

zoomX=100;
zoomY=100;

popMatrix(); //Under denne zoomes der ikke mere

update(mouseX, mouseY); //opdatering af musens position
```

```
//Herunder tegnes alle knapperne
if (krigOver) {//krig
  fill(krigHighlight);
} else {
  fill(krigColor);
}
rectMode(CORNER);
rect(krigX, krigY, SizeX, SizeY, round);
fill(255);
textAlign(CORNER, CENTER);
text(krig, krigX+40, krigY+(SizeY/2)-3);

if (polOver) { //politik
  fill(polHighlight);
} else {
  fill(polColor);
}
rectMode(CORNER);
rect(polX, polY, SizeX, SizeY, round);
fill(255);
textAlign(CORNER, CENTER);
text(pol, polX+40, polY+(SizeY/2)-3);

if (nazOver) { //nazisme
  fill(nazHighlight);
} else {
  fill(nazColor);
}
rectMode(CORNER);
rect(nazX, nazY, SizeX, SizeY, round);
fill(255);
textAlign(CORNER, CENTER);
```

```
text(naz, nazX+40, nazY+(SizeY/2)-3);

if (terOver) { //terror
  fill(terHighlight);
} else {
  fill(terColor);
}
rectMode(CORNER);
rect(terX, terY, SizeX, SizeY, round);
fill(255);
textAlign(CORNER, CENTER);
text(ter, terX+40, terY+(SizeY/2)-3);

if (kviOver) { //kvinder
  fill(kviHighlight);
} else {
  fill(kviColor);
}
rectMode(CORNER);
rect(kviX, kviY, SizeX, SizeY, round);
fill(255);
textAlign(CORNER, CENTER);
text(kvi, kviX+40, kviY+(SizeY/2)-3);


if (allOver) { //se alle begivenheder
  fill(allHighlight);
} else {
  fill(allColor);
}
rectMode(CORNER);
rect(allX, allY, SizeX, SizeY, round);
```

```
text(all, allX+40, allY+(SizeY/2)-3);

if (zoomOver) { //reset zoom
  fill(zoomHighlight);
} else {
  fill(zoomColor);
}
rectMode(CORNER);
rect(zoomX, zoomY, SizeX, SizeY, round);
fill(255);
textAlign(CORNER, CENTER);
text(zoom, zoomX+40, zoomY+(SizeY/2)-3);

pushMatrix();//Herunder zoomes igen
zoomer.transform();

//Der itereres over alle events i arraylisten
//for at se om der er blevet trykket på en knap, der har noget med dem at gøre
for (event i : events) {

  //KNAPPERNE:
  //knap zoom
  if (zoomKlik) {
    zoomer.reset();
    zoomKlik=false;
  }

  //knap all
  if (allKlik) {
    i.tegnEvent();
    fill(pressColor);
    allColor=pressColor;
```

```
  allHighlight=51;
}
//remove:
if (allKlik==false) {
  allHighlight=#F29D14;
  allColor=0;
}

//knap krig
if (krigKlik) {
  if (i.tags.hasValue("Krig")) { //
    i.tegnEvent();
    fill(pressColor);
    krigColor=pressColor;
    krigHighlight=51;
  }
}
//remove:
if (krigKlik==false && ToVKKlik==false) {
  krigHighlight=#F29D14;
  krigColor=0;
}

//knap politik
if (polKlik) {
  if (i.tags.hasValue("Politik")) {
    //noLoop();
    i.tegnEvent();
    fill (pressColor);
    polColor = pressColor;
    polHighlight=51;
  }
```

```
  }
  //remove
  if (polKlik==false) {
    polHighlight=#F29D14;
    polColor=0;
  }
  //knap nazisme
  if (nazKlik) {
    if (i.tags.hasValue("Nazisme")) { //
      i.tegnEvent();
      fill(pressColor);
      nazColor=pressColor;
      nazHighlight=51;
    }
  }
  //remove
  if (nazKlik==false) {
    nazHighlight=#F29D14;
    nazColor=0;
  }

  //knap terror
  if (terKlik) {
    if (i.tags.hasValue("Terror")) { //
      i.tegnEvent();
      fill(pressColor);
      terColor=pressColor;
      terHighlight=51;
    }
  }
```

```
//remove
if (terKlik==false) {
  terHighlight=#F29D14;
  terColor=0;
}

//knap kvinder
if (kviKlik) {
  if (i.tags.hasValue("Kvinder")) { //
    i.tegnEvent();
    fill(pressColor);
    kviColor=pressColor;
    kviHighlight=51;
  }
}
//remove
if (kviKlik==false) {
  kviHighlight=#F29D14;
  kviColor=0;
}


//INFO:
//anden verdenskrig
if (ToVKklik) {
  info VKTO = new info(loadStrings("AndenVK.txt"), events.get(3).x2+((events.get(3).x4-events.get(3).x2)/2), events.get(3).y1, ToVKcolor);
  VKTO.tegnInfo();
  ToVKcolor = #FF6431;
}
//remove
if (ToVKklik==false) {
  ToVKcolor=#F29D14;
```

```
    }

    //første verdenskrig
    if (EnVKKlik) {
      info VKEN = new info(loadStrings("firstVK.txt"), events.get(0).x2+((events.get(0).x4-events.get(0).x2)/2), events.get(0).y1, EnVKcolor);
      VKEN.tegnInfo();
      EnVKcolor=#FF6431;
    }
    //remove
    if (EnVKKlik==false) {
      EnVKcolor=#F29D14;
    }

    //kanslergadeforliget
    if (kansKlik) {
      info kans = new info(loadStrings("Kansler.txt"), events.get(2).x2+((events.get(2).x4-events.get(2).x2)/2), events.get(2).y1, kansColor);
      kans.tegnInfo();
      kansColor=#FF6431;
    }
    //remove
    if (kansKlik==false) {
      kansColor=#F29D14;
    }

    //kvindernes stemmeret
    if (KSklik) {
      info KS = new info(loadStrings("kvinder.txt"), events.get(1).x2+((events.get(1).x4-events.get(1).x2)/2), events.get(1).y1, KScolor);
      KS.tegnInfo();
      KScolor=#FF6431;
    }

     //remove
     if (KSklik==false) {
       KScolor=#F29D14;
     }

     //DK i EF
     if (EFklik) {
       info EF = new info(loadStrings("EF.txt"), events.get(4).x2+((events.get(4).x4-events.get(4).x2)/2), events.get(4).y1, EFcolor);
       EF.tegnInfo();
       EFcolor=#FF6431;
     }
     //remove
     if (EFklik==false) {
       EFcolor=#F29D14;
     }

     //9/11
     if (sepKlik) {
       info sep = new info(loadStrings("911.txt"), events.get(5).x2+((events.get(5).x4-events.get(5).x2)/2), events.get(5).y1, sepColor);
       sep.tegnInfo();
       sepColor=#FF6431;
     }
     //remove
     if (sepKlik==false) {
       sepColor=#F29D14;
     }
   }
   popMatrix();
}
```

```
void update(int x, int y) {//her opdateres musepositionen
  pushMatrix(); //der zoomes igen
  zoomer.transform();

  //herunder defineres det hvornår bollean "over knappen" er true for de forskellige knapper
  //det defienres også at, hvis én af dem er true, så er de andre false
  //dette gøres for alle knapper
  if ( overKrig(krigX, krigY, SizeX, SizeY) ) { //krig
    krigOver = true;

    polOver= false;
    nazOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;
  } else if ( overPol(polX, polY, SizeX, SizeY) ) { //politik
    polOver = true;
```

```
    krigOver=false;
    nazOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;
  } else if (overNaz(nazX, nazY, SizeX, SizeY)) { //nazisme
    nazOver=true;

    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;
  } else if (overTer(terX, terY, SizeX, SizeY)) { //terror
    terOver=true;

    nazOver=false;
    krigOver=false;
```

```
    polOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;
  } else if (overKvi(kviX, kviY, SizeX, SizeY)) { //kvinder
    kviOver=true;

    allOver=false;
    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;
  } else if (overAll(allX, allY, SizeX, SizeY)) { //se alle begivenheder
    allOver=true;

    nazOver=false;
    krigOver=false;
    polOver=false;
```

```
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    zoomOver=false;
  } else if (overZoom(zoomX, zoomY, SizeX, SizeY)) { //reset zoom
    zoomOver=true;

    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;

    //Herunder gøres det samme for alle infoboksene

    //Anden verdenskrig
  } else if (overToVK(events.get(3).x1, events.get(3).x3, events.get(3).y2, events.get(3).y1)) {
    ToVKover=true;

    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    EnVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;

    //første verdenskrig
  } else if (overEnVK(events.get(0).x1, events.get(0).x3, events.get(0).y2, events.get(0).y1)) {
    EnVKover=true;

    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    kansOver=false;
    ToVKover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;
```

```
    //Kanslergadeforliget
} else if (overKans(events.get(2).x1-10, events.get(2).x3+10, events.get(2).y2, events.get(2).y1)) {
    kansOver=true;

    EnVKover=false;
    ToVKover=false;
    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    KSover=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;

    //Kvindernes stemmeret
} else if (overKS(events.get(1).x1-10, events.get(1).x3+10, events.get(1).y2, events.get(1).y1)) {
    KSover=true;

    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    EFover=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;

    //DK i EF
} else if (overEF(events.get(4).x1-10, events.get(4).x3+10, events.get(4).y2, events.get(4).y1)) {
    EFover=true;

    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
    sepOver=false;
    allOver=false;
    zoomOver=false;

    //9/11
} else if (overSep(events.get(5).x1-10, events.get(5).x3+10, events.get(5).y2, events.get(5).y1)) {
    sepOver=true;

    KSover=false;
    kansOver=false;
    EnVKover=false;
    ToVKover=false;
    nazOver=false;
    krigOver=false;
    polOver=false;
    terOver=false;
    kviOver=false;
```

```
    EFover=false;
    allOver=false;
    zoomOver=false;

    //Herunder defineres det at hvis ingen af det ovenstående er true
    //så er de alle false
  } else {
    krigOver=polOver=nazOver=terOver=kviOver=ToVKover=EnVKover=kansOver=KSover=EFover=sepOver=allOver=zoomOver=false;
  }
  popMatrix();
}
//Herunder defineres de booleans, der siger om musen er over knappen eller ej
//INFO:
boolean overToVK (int x, int y, int z, int q) {
  if (mouseX >= x && mouseX <= y &&
    mouseY >= z && mouseY <= q) {
    return true;
  } else {
    return false;
  }
}

boolean overEnVK (int x, int y, int z, int q) {
  if (mouseX >= x && mouseX <= y &&
    mouseY >= z && mouseY <= q) {
    return true;
  } else {
    return false;
  }
}
```

```
boolean overKans (int x, int y, int z, int q) {
  if (mouseX >= x && mouseX <= y &&
    mouseY >= z && mouseY <= q) {
    return true;
  } else {
    return false;
  }
}

boolean overKS (int x, int y, int z, int q) {
  if (mouseX >= x && mouseX <= y &&
    mouseY >= z && mouseY <= q) {
    return true;
  } else {
    return false;
  }
}

boolean overEF (int x, int y, int z, int q) {
  if (mouseX >= x && mouseX <= y &&
    mouseY >= z && mouseY <= q) {
    return true;
  } else {
    return false;
  }
}
```

```
boolean overSep (int x, int y, int z, int q) {
  if (mouseX >= x && mouseX <= y &&
    mouseY >= z && mouseY <= q) {
    return true;
  } else {
    return false;
  }
}

//KNAPPER:
boolean overZoom (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}

boolean overAll (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}
```

```
boolean overKrig (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}

boolean overPol (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}

boolean overNaz (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}
```

```
boolean overTer (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}

boolean overKvi (int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}


void mousePressed() { //Her defineres de ting, der sker når brugeren bruger musen
  zoomer.transform();

  //INFO
  //anden verdenskrig
  if (ToVKover==true && mouseButton==LEFT && (krigKlik==true || nazKlik==true || allKlik==true)) {
    ToVKKlik = true;
    kansKlik = false;
  }
  //remove
  if (ToVKover==true && ToVKcolor==#FF6431 && mouseButton==LEFT ) {
    ToVKKlik = false;
  }
 //første verdenskrig
 if (EnVKover==true && mouseButton==LEFT && (krigKlik==true || nazKlik==true|| allKlik==true)) {
   EnVKKlik = true;
   KSklik=false;
 }
 //remove
 if (EnVKover==true && EnVKcolor==#FF6431 && mouseButton==LEFT) {//EnVKcolor==#FF6431
   EnVKKlik = false;
 }

 //kanslergadeforliget
 if (kansOver==true && mouseButton==LEFT && (polKlik==true|| allKlik==true)) {
   kansKlik = true;
   ToVKKlik=false;
 }
 //remove
 if (kansOver==true && kansColor==#FF6431 && mouseButton==LEFT) {
   kansKlik = false;
 }

 //Kvindernes stemmeret
 if (KSover==true && mouseButton==LEFT && (polKlik==true || kviKlik==true || allKlik==true)) {
   KSklik = true;
   EnVKKlik=false;
 }
 //remove
 if (KSover==true && KScolor==#FF6431 && mouseButton==LEFT) {
   KSklik = false;
 }
```

```
//remove
if (KSover==true && KScolor==#FF6431 && mouseButton==LEFT) {
  KSklik = false;
}

// DK i EF
if (EFover==true && mouseButton==LEFT && (polKlik==true|| allKlik==true)) {
  EFklik = true;
}
//remove
if (EFover==true && EFcolor==#FF6431 && mouseButton==LEFT) {
  EFklik = false;
}

// 9/11
if (sepOver==true && mouseButton==LEFT && (terKlik==true|| allKlik==true)) {
  sepKlik = true;
}
//remove
if (sepOver==true && sepColor==#FF6431 && mouseButton==LEFT) {
  sepKlik = false;
}

//KNAPPER:
//reset zoom
if (zoomOver==true && mouseButton==LEFT) {
  zoomKlik=true;
}
//Show all events
if (allOver==true && mouseButton==LEFT) {
  allKlik=true;
}
```

```
//remove
if (allOver==true && mouseButton==LEFT && allHighlight==51) {
  allKlik=false;
  ToVKKlik=false;
  EnVKKlik=false;
  kansKlik=false;
  EFklik=false;
  KSklik=false;
  sepKlik=false;
}

//krig
if (krigOver==true && mouseButton==LEFT) {
  krigKlik = true;
  allKlik=false;
}
//remove
if (krigOver==true && mouseButton==LEFT && krigHighlight==51) {
  krigKlik=false;
  ToVKKlik=false;
  EnVKKlik=false;
}

//Politik
if (polOver==true && mouseButton==LEFT) {
  polKlik = true;
  allKlik=false;
}
```

```
  //remove
  if (polOver==true && mouseButton==LEFT && polHighlight==51) {
    polKlik=false;
    kansKlik=false;
    EFklik=false;
    KSklik=false;
  }

  //nazisme
  if (nazOver==true && mouseButton==LEFT) {
    nazKlik = true;
    allKlik=false;
  }
  //remove
  if (nazOver==true && mouseButton==LEFT && nazHighlight==51) {
    nazKlik=false;
    ToVKKlik=false;
  }

  //terror
  if (terOver==true && mouseButton==LEFT) {
    terKlik = true;
    allKlik=false;
  }
  //remove
  if (terOver==true && mouseButton==LEFT && terHighlight==51) {
    terKlik=false;
    sepKlik=false;
  }

  //kvinder
  if (kviOver==true && mouseButton==LEFT) {
    kviKlik = true;
    allKlik=false;
  }
  //remove
  if (kviOver==true && mouseButton==LEFT && kviHighlight==51) {
    kviKlik=false;
    KSklik=false;
  }
}
```

Tab info

```
//Denne klasse danner objekterne infobokse, for hvert event
class info { //Først defineres de variabler klassen består af
  String [] tekst; //Den tekstfil, der bliver hentet ned ved hver infoboks
  float infoX; //x og y position
  float infoY;
  String tex; //Når tekstfilen sammensættes til én string
  int textRectW = 90; //størrelse på boksen
  int textRectH = 100;
  color textColor; //farven på boksen


  //constructor
  info (String [] _tekst, float _infoX, float _infoY, color _textColor) {
    this.textColor=_textColor;
    this.tekst=_tekst;
    this.infoX=_infoX;
    this.infoY=_infoY-200;
  }

  //Her tegnes boksene
  void tegnInfo() {
    String tex = join(tekst, " ");

    //firkant
    stroke(255);
    rectMode(RADIUS);
    fill(textColor, 100);
    rect(infoX, infoY, textRectW, textRectH);

    //tekst
    fill(255);
    textAlign(CORNER);
    text(tex, infoX+4, infoY+4, textRectW-4, textRectH);
    textSize(12);
  }
}
```

Tab event

```
//Det her er den klasse, der laver alle events
class event implements Comparable<event> { //comparable er med sådan at eventsne kan sammenlignes med hinanden og dermed sorteres
  StringList tags;
  int x1, y1, x2, y2, x3, y3, x4, y4;
  String title;


  event (String _title, StringList _tags, int x, int z) {
    //Udregning af årstal i pixels: (år - 1901)*10 + 200 fx. 1945 = (1945-1901)*10+200 = 640
    //Dette gørres sådan at programmet selv regner pixel-positionen ud
    //Så kan man, når man laver eventet som objekt bare skrive startår og slutår direkte ind
    this.x1=x=(x-1901)*10+200; //
    this.y1=y1_1;
    this.x2=x1;
    this.y2=y2_1;
    this.x3=z=(z-1901)*10+200;
    this.y3=y1_1;
    this.x4=x3;
    this.y4=y2_1;
    this.tags=_tags;
    this.title=_title;

    //Hvis begivenheden kun varer 1 år gøres den højere,
    //så begivenhederner ikke ligger ind over hinanden
    if (x==z) {
      y2=y2-30;
    }
  }

  //Sortering og sammenligning
  @Override
    int compareTo(event other) {
    return this.x1 - other.x1;
  }

  //her tegnes eventsne
  void tegnEvent() {
    stroke(255);
    line (x1, y1, x2, y2);
    line(x3, y1, x4, y2);
    rectMode(CORNERS);
    noStroke();
    fill(#FF6431, 100); //den orange farve, de 100 er alpha (gennemsigtighed)
    rect(x1, y1, x4, y2);
    fill(255);
    textAlign(CENTER, CENTER);
    text(title, x2+((x4-x2)/2), y2-15);
  }
}
```