



# SUBMISSION OF WRITTEN WORK

Class code:

**BFST-Spring 2015**

Name of course:

Førsteårsprojekt Danmarks Visualisering Navigation, Søgning og Ruteplanlægning (Spring 2015)

Course manager:

**Troels Bjerre Sørensen**

Course e-portfolio:

[https://mititu.dk/ucs/cb/course.sml?course\\_id=1687434](https://mititu.dk/ucs/cb/course.sml?course_id=1687434)

Thesis or project title:

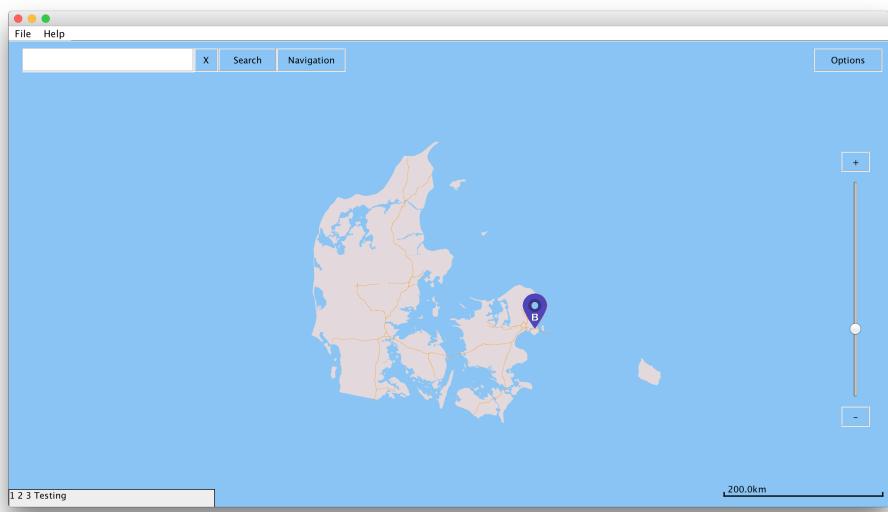
**Danmarkskort Projekt**

Supervisor:

**Troels Bjerre Sørensen**

Full Name: Birthdate (dd/mm/yyyy): E-mail:

- |   |                   |                    |
|---|-------------------|--------------------|
| 1. <u>Andreas Bjørn Hassing Nielsen</u> | <u>12/11-1990</u> | <u>abhn@itu.dk</u> |
| 2. <u>Dennis Thinh Tan Nguyen</u>       | <u>01/04-1993</u> | <u>dttn@itu.dk</u> |
| 3. <u>Emma Arfelt</u>                   | <u>06/05-1994</u> | <u>ekoc@itu.dk</u> |
| 4. <u>Steffen Immerkær</u>              | <u>14/10-1993</u> | <u>shau@itu.dk</u> |
| 5. <u>Stig Killendahl</u>               | <u>12/05-1995</u> | <u>skil@itu.dk</u> |
| 6. _____                                | _____             | <u>@itu.dk</u>     |
| 7. _____                                | _____             | <u>@itu.dk</u>     |



## Førsteårsprojekt - Danmarkskort

Andreas B.H. Nielsen, Dennis T.T. Nguyen  
Emma Arfelt, Steffen Immerkær, Stig Killendahl

20. maj 2015

# Indhold

<b>1</b>	<b>Indledning</b>	<b>4</b>
<b>2</b>	<b>Baggrund</b>	<b>5</b>
2.1	Krav . . . . .	5
2.2	Problemanalyse . . . . .	6
2.2.1	Læse og gemme kortet hurtigt . . . . .	6
2.2.2	Brugergrænseflade . . . . .	7
2.2.3	Interaktion med kortet . . . . .	7
2.2.4	Adressesøgning . . . . .	8
2.2.5	Rutevejledning . . . . .	9
2.2.6	Optimering af hastighed . . . . .	11
<b>3</b>	<b>Teknisk beskrivelse</b>	<b>13</b>
3.1	Dataflow . . . . .	13
3.2	Høj-niveau klassestruktur . . . . .	14
3.3	Element klassestruktur . . . . .	14
3.4	Kodestruktur . . . . .	15
3.4.1	Pakker . . . . .	15
3.5	Serialisering af objekter . . . . .	16
3.5.1	Problemer med serialisering af Path2D . . . . .	16
3.6	Datastrukturer og algoritmer . . . . .	17
3.6.1	Implementation af QuadTree . . . . .	17
3.6.2	Implementation af rutevejledning . . . . .	17
3.6.3	Implementation af adressesøgning . . . . .	18
<b>4</b>	<b>Brugergrænseflade</b>	<b>19</b>
4.1	Designprocessen . . . . .	19
4.2	Brugergrænsefladens opbygning . . . . .	20
4.3	Brugergrænseflade analyse . . . . .	21
<b>5</b>	<b>Brugermanual</b>	<b>22</b>

<b>6 Systematisk test</b>	<b>24</b>
6.1 Unittest . . . . .	24
6.2 Manuel test . . . . .	25
6.3 Blackboxtest . . . . .	26
6.4 Whiteboxtest . . . . .	26
<b>7 Procesbeskrivelse</b>	<b>29</b>
7.1 Gruppearbejdet . . . . .	29
7.2 Redskaber . . . . .	30
7.3 Planlægning og tidsestimat . . . . .	30
<b>8 Produktrefleksion</b>	<b>32</b>
8.1 Optimering af køretid . . . . .	32
8.2 Optimering af hukommelsesbrug . . . . .	33
8.3 Programfejl og mangler . . . . .	33
8.3.1 Mangler ved regulære udtryk . . . . .	33
8.3.2 Liste over kendte fejl . . . . .	34
8.4 Ekstra features . . . . .	35
8.4.1 Rejse med mellempunkt . . . . .	35
8.4.2 Drag-and-drop start- og slutpunkt . . . . .	35
<b>9 Konklusion</b>	<b>37</b>
<b>10 Referencer</b>	<b>38</b>
10.1 Sekundær litteratur . . . . .	38
10.1.1 Internetmateriale . . . . .	38
10.2 Ikoner . . . . .	39
<b>11 Bilag</b>	<b>40</b>
11.1 Whiteboxtest . . . . .	41
11.1.1 Expectancy tables . . . . .	41
11.2 Coverage tables . . . . .	42
11.2.1 Model coverage . . . . .	43
11.3 Gruppearbejdet . . . . .	43
11.3.1 Gantt-diagram af projekt plan . . . . .	43
11.4 Version changelog . . . . .	44
11.4.1 V1.0 - FINAL FRONTIER OF GRAPES . . . . .	44
11.4.2 RC1.0 - HOLY VITAMIN C . . . . .	45
11.4.3 V0.10 - Honorable Sudachi . . . . .	46
11.4.4 V0.9 - Explosive Grapefruit . . . . .	48
11.4.5 V0.8 - Sweet clementine . . . . .	50
11.4.6 V0.7 - Blazing citrus . . . . .	51
11.4.7 0.4 - Creative lime . . . . .	53
11.4.8 V0.3 - Numb orange . . . . .	55

11.4.9	V0.2 - Annoying lemon	57
11.4.10	V0.1 - Precitrus era	59
11.5	Logbog	61
11.6	Gruppekontrakt	87

# Kapitel 1

## Indledning

Følgende rapport er udarbejdet af Andreas Nielsen, Dennis Nguyen, Emma Arfelt, Steffen Immerkær og Stig Killendahl i forbindelse med 2. semesters eksamensprojekt i Danmarks kort: Visualisering, Navigation, Søgning og Ruteplanlægning.

Overordnet har dette kursus sat fokus på at strukturere et større arbejde i mindre grupper. Det har blandt andet stillet krav til organisering, planlægning og eksekvering af projektet og gruppearbejdet. Blandt andet har den store selvstændige process i arbejdet været betinget af, at alle gruppemedlemmer har været involveret i processerne, men at vi også kunne uddelegeret opgaver og ansvar. Fra start har vi derfor fokuseret på at få uddelegeret ansvaret, hvor vi henholdsvis har valgt en projektleder, hand-in ansvarlig samt sekretær.

Gennem forelæsninger og code sessions har vi fået en række værktøjer, som skulle lette arbejdet i forhold til programmeringen af Danmarks kortet, for eksempel Dijkstras algoritme, som finder korteste vej mellem to knuder i en graf, og dennes udvidelse; A\*. Det store datasæt fra Open Street Map har dog skullet analyseres dybere, da det er et open source projekt. Dette har været en udfordring i forhold til at strømline programmet og de forskellige tags, som definerer vejens type, ikoner m.m. hvilket også vil blive gennemgået. Vi har skullet analysere hvilke værktøjer og muligheder, som vi ville gøre brug af og ikke mindst begrænse vores valg af features.

Der er flere måder, hvorpå et sådant program kunne programmeres, man skal blandt andet overveje hvilke datastrukturer, søgealgoritmer og designmæssige valg af brugergrænseflade, man vil løse den givne problemstilling med. Denne rapport beskriver vores løsning af problemstillingen og sammenhængen mellem ovenstående overvejelser. Derudover omfatter rapporten en detaljeret brugervejledning, en teknisk beskrivelse af programmets opbygning og afprøvning af det færdige program.

# Kapitel 2

## Baggrund

I dette projekt skal vi udarbejde og implementere et program, der kan visualisere og interagere med kortdata fra open-source projektet Open Street Map<sup>1</sup>. Ved interaktion med kortet, skal der kunne zoomes og panoreres rundt. Programmet skal også kunne vise vejnavne og give detaljerede rutevejledninger mellem to brugervalgte adresser.

### 2.1 Krav

Forud for projektet er der blevet udleveret en kravspecifikation, hvori det fremgik hvad kortet skulle kunne. Disse krav er alle mødt i vores program samt to yderligere funktioner, som fremgik af "*Extensions*"-afsnittet i projektbeskrivelsen.

Ud fra kravene vi er blevet givet, har vi formuleret følgende programmet skal:

1. Kunne læse og gemme kortet hurtigt
2. Vise elementer i kortet genkendeligt og forståeligt
3. Give brugeren mulighed for at interagere med kortet
4. Give brugeren mulighed for at søge efter en adresse
5. Kunne foretage en rutevejledning for forskellige transportmidler
6. Køre så hurtigt, at det er praktisk at bruge

De ovenstående krav kan ses som en række problemer, der kan løses med en datalogisk tilgang. Det har været nødvendigt at analysere disse krav og reflektere over hvilke løsninger, der passede til specifikke dele af programmet. Vores tilgang og fortolkning af kravene, vil blive gennemgået i nedenstående afsnit.

---

<sup>1</sup>Open Street Maps (2015): About

## 2.2 Problemanalyse

Dette projekt har haft en tidsbegrænsning på 12 uger, hvilket har betydet, at der er features og funktioner, der er blevet prioriteret højere end andre, for at opfylde samtlige krav. Da systemet skal kunne betjenes af enhver, har vi reflekteret over, hvilke arbejdsopgaver der formentlig vil forekomme hyppigere end andre samt hvordan disse skal struktureres.

Derudover har vi kigget på kravene med datalogiske øjne og vurderet hvilke løsninger vi kunne implementere.

### 2.2.1 Læse og gemme kortet hurtigt

Til at tegne vores kort har vi benyttet Open Street Map, der er et open-source projekt, der indeholder kortdata fra hele verden. Det data vi benytter er i filformatet **XML** (eXtensible Markup Language), der giver en klar og læsbar struktur. Udfordringen ved dette format er at det let kommer til at fylde meget, og at parsing af disse filer kan tage lang tid. Da det første krav til programmet er, at brugeren kan indlæse og gemme kortet hurtigt, er det derfor nødvendigt, at analysere hvordan vi kan gøre dette uden at miste betydende data, eller bruge for meget hukommelse. Hurtigt er i denne sammenhæng defineret som, at læsning af en behandlet fil må tage op til 1 minut.

At det er et open-source projekt betyder at dataen ikke nødvendigvis er strømlinet eller følger samme standard, da det er forskellige brugere, der tilføjer det. Dette er en stor udfordring i forhold til at tegne et kort. Først og fremmest fordi to ens vejtyper eksempelvis kan indeholde mange forskellige tags, men også fordi en masse data skal filtreres fra, da det ikke er relevant i forhold til projektet. Det har derfor været et krav til programmet, at det tager højde for de forskelligheder og fejl, der kan forekomme i et open-source datasæt, som dette.

Danmarkskortet fylder omtrent 4 Gigabyte i OSM-filformatet, og vil ved et almindelig fil load fylde det samme i hukommelsen. Hvis kortet derimod bliver komprimeret til en .zip fil, vil det fylde betydeligt mindre. I eksemplet med danmarkskortet, kan det fylde helt ned til 300 Megabyte i stedet. Dette vil dog ikke hjælpe i forhold til at loade kortet ind i programmet, da det alligevel først skal pakkes ud inden indholdet kan håndteres. Dette problem kan løses ved at bearbejde dataen sekventielt ved hjælp af en **SAX** parser<sup>2</sup>. Ved brug af SAX parseren kan vi læse en OSM-fil linje for linje, og dermed undgå at der optages mere hukommelse end højst nødvendigt.

Det har været et krav til programmet, at det kan åbnes med en rimelig hastighed. Dette kan være en udfordring, når der skal læses en 4 Gigabyte OSM-fil eksempelvis. For at løse denne problemstilling vil det være nødvendigt at lade kortet gemme data i binært format, hvilket er ulæseligt for

---

<sup>2</sup>SAX Project (2015): SAX-Parser Information

mennesker, men hurtigt at læse for en computer. Dette skal være med til, at gøre den initiale læsetid betydeligt kortere for brugeren af programmet, i forhold til at samme kort indlæses som OSM-fil.

### 2.2.2 Brugergrænseflade

Det er et krav, at programnets brugergrænseflade er genkendelig og understøtter programmets funktionalitet. I vores analyse af, hvorledes kortet kan gøres forståeligt, vil vi kigge på andre kortvisningsprogrammer, heriblandt Google Maps samt Open Street Maps og lade os inspirere af deres farvevalg og designbeslutninger, for at give en genkendelig brugergrænseflade - eksempelvis at vejene er hvide og ikke grønne eller blå. Herudover vil vi overveje, hvilke elementer der er hensigtsmæssige og relevante at vise på kortet. Vi vil derfor implementere en række kartografiske elementer, der vises, når brugeren er tæt nok på kortet. Én vigtig forskel, der har dannet grund for vores valg af brugergrænseflade er, at vi ikke ønskede at lave et GPS-lignende system, men et adressesøgningsværktøj. I en GPS kunne det eksempelvis være brugbart at håndtere og vise koordinator til brugeren, hvilket vores kort ikke har brug for at kunne.

Kravet foreskriver også, at brugergrænsefladen er brugervenlig og tilgængelig for en bred vifte af brugere. Derfor vil vi også udforme en tilstand, som ændrer farvetemaet til en mere farveblind venlig palette.

Til at kode brugergrænsefladen kunne vi benytte os af **JavaFX**, da vi før har stiftet bekendtskab med det. JavaFX benytter sig af en nyere måde at lave brugergrænseflader på, hvor der er lagt stor fokus på at trække designdelen væk fra selve programkoden. Man skriver en FXML-fil som indeholder designet, og opretter en controller, der håndterer input og output for det design. Dette giver en pæn struktur, som gør det let at ændre i designet uden at skulle kode sig frem til komponenters placeringer, som man gør i andre frameworks til brugergrænseflader, for eksempel Swing.

At kunne lave- og modificere et design nemt og elegant, er dog ikke vores primære bekymring, da kortet gerne skal optage det meste af programmet, med en minimalistisk brugergrænseflade.

Der er også muligheden for at bruge **Swing**, som er omrent 10 år ældre end JavaFX og dermed har flere års optimering og brug bag sig samt større mængder dokumentation. Fordelene ved Swing synes i dette projekt at veje tungere end dem ved JavaFX, særligt den nemme adgang til hjælp. Derfor benyttes Swing-frameworket.

### 2.2.3 Interaktion med kortet

Det er et krav, at brugeren skal kunne interagere med kortet. Interaktionsmulighederne skal give brugeren mulighed for, at fokusere på relevante områder på kortet.

Det kræver at vi skal kunne manipulere koordinater ved hjælp af en transformering, der rykker kortet rundt i forhold til et stillestående “kamera”. Transformeringen skal kunne skalere kortet, hvilket vil virke som et zoom, og forskyde kortet i  $x$ - og  $y$ -aksen, hvilket vil fungere som en panore-ring. Derudover skal det være muligt, at benytte disse funktioner med både mus og tastatur, og via brugergrænsefladens elementer. Kravet omkring programmets hastighed er også relevant her, da panorering og zoom af kortet bør have en reaktionstid uden forsinkelse, mens andre funktioner på kortet gerne må have en længere reaktionstid.

Vi vil også gøre det muligt at vise og skjule visse elementer efter behov. Ønsker man ikke, at se et bestemt ikon på skærmen, skal man altså kunne slå det fra. Denne funktion skal placeres i et komponent over kortet, der indeholder alle de tilgængelige kortmodificeringer, for eksempel anti-aliasing, farveblind tilstand og mulighed for at vise og skjule visse typer ikoner.

#### 2.2.4 Adressesøgning

At kunne søge på en specifik adresse, i en liste af adresser hentet fra OSM-filen, kan ses som et tekstopdelingsproblem. Hvornår starter vejnavnet i forhold til bynavnet, hvornår er et nummer et husnummer, og hvornår er det et postnummer?

Programmet skal altså kunne behandle brugerens input, og trække en adresse ud fra teksten. Der findes mange sammensætninger af en adresse, eksempelvis kan ”*Ribegade 2b, 7100 Vejle*” skrives som ”*Vejle 7100 Ribegade 2b*”. Nogle veje- og bynavne kan bestå af et ord, mens andre består af flere. Nogle navne indeholder desuden specielle tegn, såsom apostrof og lignende.

En løsning kan være at benytte regulære udtryk, til at validere inputtet samt skelne mellem de forskellige dele af en adresse. Det vil aldrig være muligt at lave perfekte regulære udtryk, der behandler alle typer af input korrekt. Men vi kan opstille en række standard regulære udtryk, der tager højde for en stor samling standard kombinationer af adresser, og på den måde dækker så vidt et spektrum som muligt. Et eksempel på standard regulære udtryk ses i figur 2.1:

```

1 String sp = "([, ])*"; //Spacer
2 String address = "[a-zA-ZØÅÆøåéüéöéö]-[a-zA-ZØÅÆøåéüéöéö]*[. -]*[a-zA-ZØÅÆøåéüéöéö]*[a-zA-ZØÅÆøåéüéöéö]*";
3 String house = "([0-9]{1,3}[a-zA-Z]?)";
4 String postCode = "([0-9]{4})";
5 String city = "([a-zA-ZØÅÆøåéüéöéö]+[ ]?[a-zA-ZØÅÆøåéüéöéö]*)";

```

Figur 2.1: Standard regulære udtryk

Et andet krav til en adresse er, at den skal indeholde et koordinatsæt som referer til et punkt på kortet. Dette skal vi bruge til, at lokalisere den givne adresse på kortet.

## 2.2.5 Rutevejledning

At finde korteste vej mellem to placeringer på et kort, kan tolkes som et datalogisk problem: at finde korteste vej mellem to knudpunkter i en graf. Således kan der opbygges en graf for alle veje i OSM-filen, hvor hvert punkt i en vej vil være repræsenteret som et knudpunkt, og hvert linjesegment i en vej være en kant.

Repræsentationen af kortet som en graf åbner op for benyttelse af gode grafsøgealgoritmer, som for eksempel **Dijkstra** og **A\***.

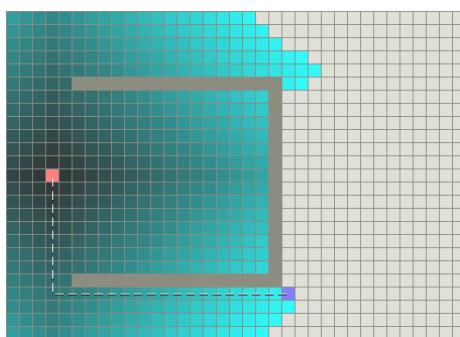
### Dijkstra og A\*

Dijkstra er en algoritme, der søger gennem grafen bredde først. Denne finder korteste vej fra et startpunkt til alle forbundne punkter. Dijkstra kan modificeres således, at den hurtigere vil nå et bestemt slut-punkt, ved at stoppe søgningen når slutpunktet er fundet.

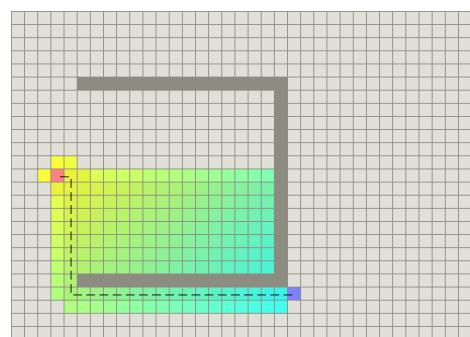
En udvidelse til Dijkstra er søgealgoritmen: A\*. Denne benytter sig af en heuristik til at omvägte grafen løbende, og dermed hurtigere bevæge sig hen mod slutpunktet. Heuristikken kan for eksempelvis være defineret som en fugleflugtslinje fra det nuværende punkt hen til slutpunktet. Dette betyder, at et punkt, der er tættere på slutpunktet, bliver undersøgt inden et punkt, der er længere væk. Dette kaldes “best-first search”. “Best-first search” udforsker grafen ved, at tage det mest lovende knudpunkt efter en specifik regel, der defineres i heuristikken. For eksempel det knudpunkt som er tættest på slutpunktet i fugleflugtsafstand.

### Implementation af A\*

Vi har valgt at implementere A\*, som tager udgangspunkt i Dijkstra's algoritme, men derudover benytter en heuristik til at udvælge, hvilke kanter, der undersøges i grafen.



Figur 2.2: Dijkstra



Figur 2.3: A\*

På illustrationen<sup>3</sup> ses tydeligt, hvordan A\* (figur 2.3) undersøger langt færre knudepunkter end Dijkstra (figur 2.2). Vi vil gerne undersøge hvorvidt A\* altid vil være hurtigere end Dijkstra, der undersøger alle knudepunkter. Dette afhænger af den heuristik, der benyttes til A\*.

For at det kan garanteres, at A\* er komplet og optimal, skal der benyttes en heuristik, der er “admissible”. En heuristik er admissible, hvis den aldrig overestimerer afstanden fra et givent punkt til slutpunktet. Her defineres komplet og optimal som, at den altid finder den korteste vej, hvis denne eksisterer, og at den ikke undersøger et knudepunkt mere end én gang. Heuristikken skal altså opfylde følgende:

$$\forall n, h(n) \leq dist(n),$$
 hvor  $h(n)$  er heuristikkens estimerede afstand fra et givent knudepunkt  $n$  til slutpunktet og  $dist(n)$  er den aktuelle afstand fra knudepunktet til slut.

Dette betyder, at heuristikken fejlagtigt kan overse en kortere vej, hvis den overestimerer afstanden til slutpunktet. For at sikre at dette ikke sker, er vores heuristik for korteste vej, defineret som afstanden i fugleflugt fra punktet  $n$  til slutpunktet, da vi ved at denne afstand altid vil være mindre, eller lig med den aktuelle afstand. Til hurtigste vej er heuristikken ligeledes defineret så, at den tager afstanden i fugleflugt med 130 km/t, hvormed vi sikrer os, at vi ikke overestimerer værdien, da den maksimale hastighed på alle veje er 130 km/t.

Dette betyder, at A\* i absolut værste tilfælde, altså når der ikke findes nogen korteste vej, undersøger lige så mange knudepunkter som Dijkstra. I bedste tilfælde vil den kun undersøge de punkter, der danner den korteste vej. Dette gøres, hvis heuristikken estimerer præcis den aktuelle afstand i vores tilfælde, hvis der findes en vej direkte fra start til slut i fugleflugt. Ydermere, vil A\* klare sig betydeligt bedre end Dijkstra i gennemsnitlige tilfælde, hvor den korteste rute findes hurtigere, ved at bevæge sig mere målrettet mod slutpunktet, end ved at benytte bredde først.

---

<sup>3</sup>Stanford University (2015): Introduction to A\*

### 2.2.6 Optimering af hastighed

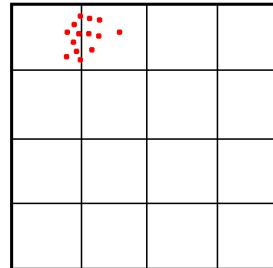
For at have et kort der er hurtigt og praktisk at bruge, kræves det at brugerinteraktioner som for eksempel panorering og zoom foregår uden nogen form for forsinkelse. Dette kan sikres gennem en velvalgt datastruktur.

Grundet mængden af data, kan vi ikke vise alt på samme tid. Derfor kræves det af vores datastruktur, at den kan opbevare en masse data og udføre to typer af forespørgsler effektivt: find en mængde data der ligger inden for et defineret område (range search) samt find det datapunkt, der er placeret tættest på et givent koordinat (nearest neighbour search). På baggrund af disse krav, er det oplagt at valget må falde på en spatial datastruktur, der netop er optimeret til disse forespørgsler. Hvad der ikke er oplagt, er hvilken en der skal vælges, blandt mange anerkendte spatiale datastrukturer.

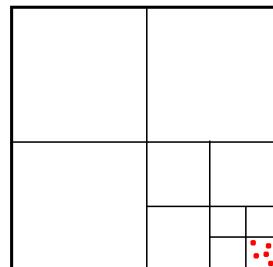
Den første overvejelse der skal gøres i forhold til at få et hurtigt og anvendeligt program, er hvordan data skal opbevares og hentes på en effektiv måde. En naiv og ineffektiv måde, kunne være at opbevare alt data i et gitter. Gitteret ville opdele data-mængden på forskellige zoomniveauer, men ville ikke håndtere tomme- eller højdensitets områder godt (se billede til højre), og netop med geografisk data ved vi at densiteten af punkter er høje i udvalgte områder som byer og lave i områder som eksempelvis havet.

Vi vil starte med at implementere QuadTree. Denne implementation vil være baseret på Sedgewick og Waynes Quadtree med nogle ændringer. Begrundelsen for dette valg er at det, for det første, er forholdsvis simpelt at implementere i forhold til andre datastrukturer. Samtidig har vi erfaret at QuadTree har en god køretid i praksis, selvom køretiden i værste tilfælde er dårlig. (se eksempel på værste tilfælde på billedet til højre)

Spørgsmålet er om vi kan være sikre på at vi ikke oplever værste tilfælde køretid med vores QuadTree? For at besvare dette spørgsmål er det nødvendigt at tænke over hvornår et sådant tilfælde finder sted. Det ville kræve at et QuadTree blev



Figur 2.4: Worst-case for en gitter-opdeling.



Figur 2.5: Worst-case for et QuadTree.

utrolig ubalanceret. Et ubalanceret QuadTree kunne opstå ved at mange punkter blev indsat utrolig tæt på hinanden, det ville medføre at der skulle ske utrolig mange opdelinger for at splitte punkterne ud i hver deres boks. I et sådan tilfælde vil det givne QuadTree være rigtig dybt, og det ville forværre hastigheden af en søgning, da den skulle foretage en stor mængde af rekursive kald<sup>4</sup>. En ændring vi har foretaget for at imødekommme denne problematik, er at vi først opdeler en boks i fire nye Quads, når den givne boks har over 1000 objekter i. Med denne ændring mindske vi risikoen for at vores QuadTree bliver alt for dybe.

Selvom geografisk data er opdelt i højdensitets klynger, er det ikke i en så ekstrem grad som værste tilfælde ville kræve. Geografisk data opfører sig formentlig ikke på en måde der degenererer QuadTree, derfor antager vi altså på den baggrund, at vi ikke vil opleve værste tilfælde køretid.

Vi kunne vælge, at benytte os af et k-d træ i stedet for QuadTree. k-d træet ville håndtere QuadTree's worst-case bedre, da træet håndterer grupperet data bedre, fordi den opdeler søgerummet efter den givne data, således at træet bliver maksimalt balanceret. Dette er i modsætning til QuadTree som rekursivt opdeler søgerummet i fire lige store kvadranter, uanset hvordan data ser ud. Særligt i højdensitets områder vil man observere dette. k-d træet er dog mere komplekst at implementere i forhold til QuadTree. På baggrund af QuadTree's gode ydeevne i praksis, nedprioriteres implementationen af en anden datastruktur.

En datastruktur bør vælges ud fra projektets omfang, og i dette tilfælde har vi erfaret at QuadTree giver den ønskede hastighed for netop vores data, og derfor er vi tilfredse med denne datastruktur. Havde vi skulle behandle en større mængde data end Danmark, ville det have stillet endnu større krav til datastrukturen, hvilket kunne have betydet, at vi skulle kigge på en stærkere datastruktur. I dette projekt er QuadTree-implementationen dog tilstrækkelig.

---

<sup>4</sup>Computer Science VirginiaTech (2015): Organizing spatial data

# Kapitel 3

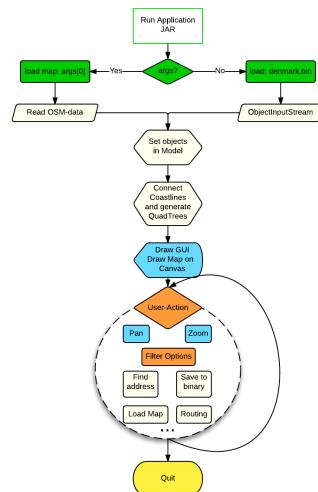
## Teknisk beskrivelse

Gennem udviklingen af projektet har den oprindelige kodestruktur ændret sig markant. Vi vil i dette afsnit gennemgå de vigtigste klasser, datastrukturer og algoritmer for slutproduktet. En vigtig komponent i softwareudvikling er at strukturere koden, holde læsbarheden høj samt afkoble klasserne. Dette har været et stort fokusområde, og det har taget betydelig tid at gennemføre et ansvarsdrevet design af koden. Dette var også for at begrænse mængden af teknisk gæld, således at vi til sidst i processen forholdsvis nemt ville kunne implementere nye features.

### 3.1 Dataflow

I overvejelerne omkring programmets dataflow blev der taget udgangspunkt i før-nævnte domæneanalyse, hvor det eksempelvis var standard at et kort blev vist uden input fra brugeren samt at interaktionerne skulle kunne foretages igen og igen. Ud fra disse antagelser er følgende dataflow blevet opstillet:

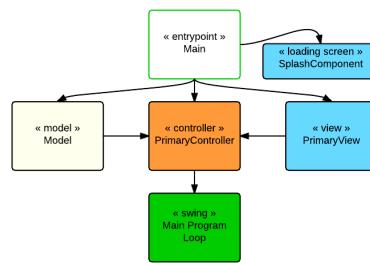
Ved programmets initialisering loades som standard et kort, der er indlejret i jar-filen. Programmet kan også åbnes via en terminal, og her har brugeren mulighed for at åbne kortet med en gyldig vilkårlig fil. Hvis dette er tilfældet, skal programmet først parse filen. Når kortet er loadet ind i modellen, vil programmet udføre nogle afsluttende modificeringer af kortdata. Dette kunne være at lukke brudte kystlinjer og oprette QuadTrees. Herefter tegnes bruger-



Figur 3.1: Flowchart

grænsefladen samt kortet. Derefter afventer programmet brugerinput, som vil manipulere kortvisningen.

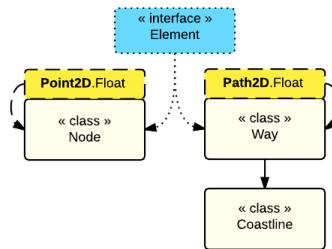
## 3.2 Høj-niveau klassestruktur



Figur 3.2: Høj-niveau klassestruktur

Går vi længere op i abstraktionsniveau ses programmets overordnede struktur, som følger Model-View-Controller arkitekturen. Der er et startpunkt (Main) som initialiserer en model, en primær controller og et primært view. Controlleren initialiseres med en reference til modellen og viewet, således at de to klasser er adskilt. Controlleren sørger for kommunikation imellem de to.

## 3.3 Element klassestruktur



Figur 3.3: Element klassestruktur

Vores umiddelbare tilgang til at fortolke OSM-filformatet førte til at

bruge et Element interface. Et Element i vores program kan være 3 forskellige ting - en Node, en Way eller en Relation.

En Node repræsenterer et punkt på kortet. Dette punkt består af et koordinatsæt, og en type-variabel. En Way er en samling af Nodes, der repræsenterer objekter som veje på kortet. Udover en type-variable, indeholder Ways et navn, en liste af kanter og en fartgrænse. Relations er en samling af Ways og/eller Nodes, der repræsentere områder såsom skove og parker.

Formålet med dette interface var, at oversætte de vigtige elementer fra OSM-filen til klasser i vores program. Element-interfacet lovede i en tidligere version af programmet<sup>5</sup>, at underklasser; Node, Way og Relation, ville have metoder, der returnerede et ID, satte typen af element og implementerede en sammenligningsfunktion, der skulle bruges til at tegne i riktig rækkefølge.

Senere, da vi fravalgte Relation-typen og lavede en række optimeringer af hukommelse, for eksempel at fjerne ID'et fra elementet, er Element-interfacet blevet mindre relevant. Det er nu kun Way-typen, der benytter sig af Element's typer (kystlinje, græsområde, motorvej).

## 3.4 Kodestruktur

Vi har benyttet Model-View-Controller arkitekturen, da dette er et fordelagtigt designpattern i større projekter som dette. At opdele programmet i tre dele giver en bedre og fastere kodestruktur, et bedre overblik, nemmere separation, og dermed lavere kobling. For netop vores projekt har opdelingen af model, view og controller både gjort det mere praktisk, at arbejde flere på kodebasen på samme tid, samt gjort det lettere at udskifte og udvide komponenter løbende i projektet. Netop denne sidste pointe vejede tungt i vores beslutning, da vi på forhånd vidste, at der vil være forelæsninger undervejs i projektforløbet og dermed ny viden der skulle medtages løbende.

### 3.4.1 Pakker

Vores View pakke indeholder en række views og håndterer fremvisningen af vores model i en grafisk brugergrænseflade. View pakken indeholder en primær viewklasse kaldet "PrimaryView", der opretter selve rammen til programmet, samt indeholder de forskellige underviews. En anden vigtig klasse er vores MainComponentView, der opretter komponenter, som for eksempel knapper og søgefelter i separate lag. I starten håndterede MainComponentView alle komponenter af vores view, men da dette forøgede koblingen af programmet, havde vi besluttet at opdele komponenterne i deres egne klasser, der hver især håndterer én komponenttype.

Controllerpakkens primære opgave er, at håndtere dataflowet mellem model og view. Derudover registrerer den også brugerinteraktioner ved at

---

<sup>5</sup>GitHub ITU (2015): Version 0.8

oprette, håndtere og tildele de forskellige actionlisteners til de forskellige views. Vi har også her valgt en modulær tilgang og opdelt klassen i flere underklasser.

Derudover har vi også en Model pakke, der indeholder vores datastruktur og modtager ændringer fra controlleren. Model er også ansvarlig for at kalde på underklasser såsom filehandler klassen som håndterer I/O operationer.

Vi har desuden en række hjælpeklasser i utility pakken, der håndterer udvalgte underopgaver, som ikke hører til i den almindelige model-view-controller struktur, eksempelvis HaversineDistanceCalculator-klassen der udregner afstanden mellem to koordinater i meter.

### 3.5 Serialisering af objekter

Da programmet skal kunne gemme kortdata, inkluderes en binær kortfil, der fylder meget lidt i forhold til OSM formatet og loader hurtigt.

For at kunne gemme Java objekter binært, skal de kunne serialiseres. Med det menes der, at en klasse skal implementere *java.io.Serializable* interfacet, hvorefter objekter af klassetypen kan gemmes.

Dét Java gør, når objekter serialiseres er, at tage felterne i objektet og repræsentere dem som en sekvens af bytes, sammen med information om typen af objekt, der er gemt.

Når objekter så deserialiseres tilbage ind i programmet, udføres den omvendte proces på den binære kortfil, se figur 3.4.

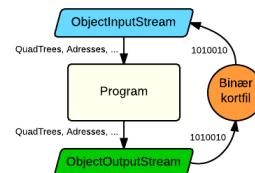
Det er vigtigt, at en binær fil med objekter indlæses i samme rækkefølge som den er gemt, altså hvis objekt 1, 2 og 3 er gemt i en rækkefølge, skal de indlæses i samme.

#### 3.5.1 Problemer med serialisering af Path2D

Javas implementation af Path2D, som programmets Way-klasse nedarver fra, har en private constructor. Dette gør at subklasser af Path2D ikke kan deserialiseres, og som resultat af dette kan Way heller ikke.

Løsning til dette problem var, at hente kildekoden for Path2D fra openjdk<sup>6</sup>, gøre constructoren public og indsætte den som en Model-klasse i vores program.

Med denne løsning kan objekter af typen Way nu deserialiseres, da de nu nedarver fra den deserialiserbare version af Path2D.



Figur 3.4: ObjectIn- og OutputStream

<sup>6</sup>Java.net (2015): OpenJDK Path2D

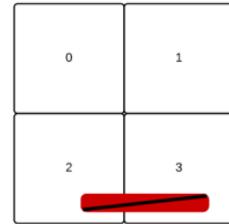
## 3.6 Datastrukturer og algoritmer

En af de store udfordringer var at håndtere de store mængder data på en fornuftig måde. For at optimere dette har vi implementeret et QuadTree til visning af kortet. Samtidig er der implementeret en graf, hvor der benyttes A\* til at finde korteste eller hurtigste vej. Implementationerne af disse omtales her.

### 3.6.1 Implementation af QuadTree

QuadTree, som er en trædatastruktur hvor hver intern knude har præcis 4 børn. Disse børn initialiseres når knuden når en defineret maks kapacitet. Vores implementationen af QuadTree er baseret på Sedgewick og Waynes implementation dog med flere af vores egne modifikationer.

Sedgewick og Waynes implementation er baseret på, at det er punkter, der bliver indsatt i QuadTrees. I vores tilfælde har vi objekter i mange former. For at løse dette problem valgte vi, at repræsentere de forskellige objekter i vores QuadTrees med objektets afgrænsende boks. Objekternes originale form blev naturligvis bevaret, men i stedet for benytte et x- og y-kordinat til indsættelse benyttede vi nu objektets afgrænsningsområde. Dette betyder dog også, at et objekt indsættes i flere ”Quads”, hvis objektets afgrænsning skærer i flere (se figur 4.1). En anden ændring, som gør sig gældende i vores implementation er, at en boks først opdeles i fire nye Quads, når den givne boks har over 1000 objekter i. Effekten opnås ved en rekursiv opdeling. Hvert QuadTree har en liste af objekter samt et array med fire referencer til nye QuadTrees. Disse fire referencer kan ses som træets børn. Børnene initialiseres når forældrene indeholder 1000 elementer. Derefter kopieres de 1000 elementer til de fire børn, og forældrene indeholder nu ingen elementer.



Figur 3.5: Objekter er repræsenteret i QuadTree af dets omgrænsende boks. Det sorte er objektet, mens det røde repræsenterer dets afgrænsningsområde. Objektet her er opbevaret både i 2. quad og 3. quad.

### 3.6.2 Implementation af rutevejledning

I OSM-filen for Danmark findes der omkring 600.000 veje, hvilket stiller krav til programmet om nemt og hurtigt at kunne kortlægge disse. Vi har defineret vores adresser således, at de har et sæt koordinater, der bruges til at lokalisere adressen på kortet. Disse adresser opbevares i en sorteret liste, som der kan søges i. Vi benytter en binær søgning til opslag af en specifik adresse. Efter adressen er fundet benyttes A\* i grafen over alle veje

i danmark.

### 3.6.3 Implementation af adressesøgning

Programmet skal kunne søge efter en adresse ud fra et givent bruger input. Fordi der eksisterer over flere millioner af adresser i Danmark, så kan det for eksempel ikke betale sig for at itere igennem alle adresser ved hjælp af lineær søgealgoritme. Køretiden for en lineær søgealgoritme er  $O(n)$ , hvilket er for langsomt i forhold til, at adressesøgning er en funktion, som vil blive brugt hyppigt. Derfor benytter programmet sig af en binær søgealgoritme til at finde frem til den rigtige adresse. Dette kræver dog at alle addresser ligger i en sorteret liste, men selve sorteringen sker dog kun en enkel gang når programmet starter op. Her benyttes javas egen sorteringsmetode. Herfra vil hver søgning kun have en køretid på  $O(\log n)$  hvilket er betydelig hurtigere end linær søgningsalgoritmen.

## Kapitel 4

# Brugergrænseflade

### 4.1 Designprocessen

Det er vigtigt, at programmet er intuitivt at benytte, men samtidig har så få komponenter som muligt. For at gruppen kunne arbejde mod én samlet vision, blev der i starten af designprocessen benyttet en række metoder, til at illustrere den ønskede brugergrænseflade. I denne process er der blandt andet benyttet mock-ups, hvilket kan ses i illustrationerne herunder:



Figur 4.1: Mockup af rutevejledning, og programmets brugergrænseflade.

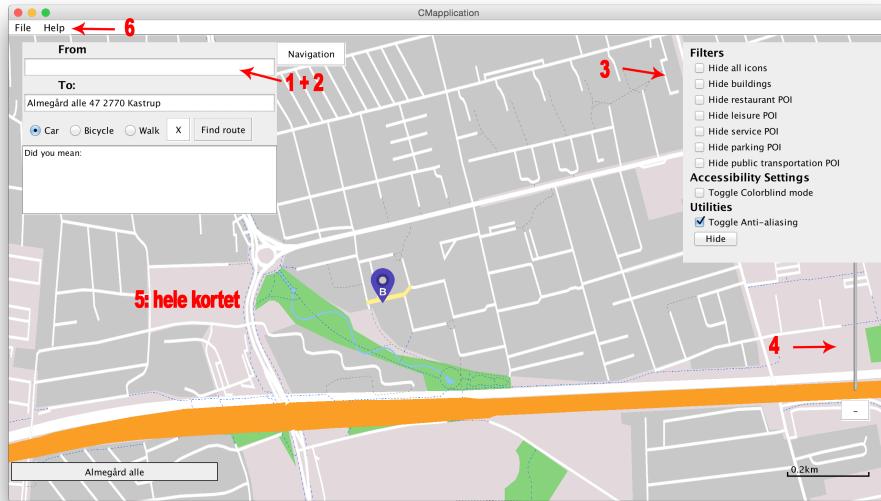
Under domæneanalysen, har vi ladet brugergrænsefladen inspirere af Google Maps, som er ét af verdens mest benyttede kortvisningsprogram-

mer. Dette hjælper med at skabe en genkendelig brugergrænseflade, således brugeren ikke behøver at danne en ny mental model af vores program.

## 4.2 Brugergrænsefladens opbygning

Vi har tilstræbt, at gøre programmet så simpelt som muligt, således at en ikke-teknisk person vil kunne betjene programmet. Dette er blandt andet gjort ved at lave et single view window, og ikke splitte funktioner op i forskellige faner eller vinduer. Programmets opbygning består dermed af et vindue med seks overordnede komponenter:

1. Adresse søgningskomponent
2. Rutenavigationskomponent
3. Indstillingskomponent
4. Zoomkomponent
5. Kortkomponent
6. Menubjælke



Figur 4.2: Screenshot af endeligt program, med komponent nummerering.

Disse komponenter giver en tydelig, enkel og intuitiv opdeling af hvad programmet kan.

### **4.3 Brugergrænseflade analyse**

Blandt andet har vi valgt, at have vores Indstillingskomponent placeret på selve kortet og ikke som et menupunkt, da det giver et bedre overblik over hvilke indstillinger brugeren kan vælge.

Ved addressesøgning, henholdsvis almindelig søgning og søgning af adresser under rutevejledning, zoomer og panorer kortet automatisk hen til de specifikke adresser. De viste ikoner giver brugeren mulighed for, at danne et hurtig overblik over adressens placering.

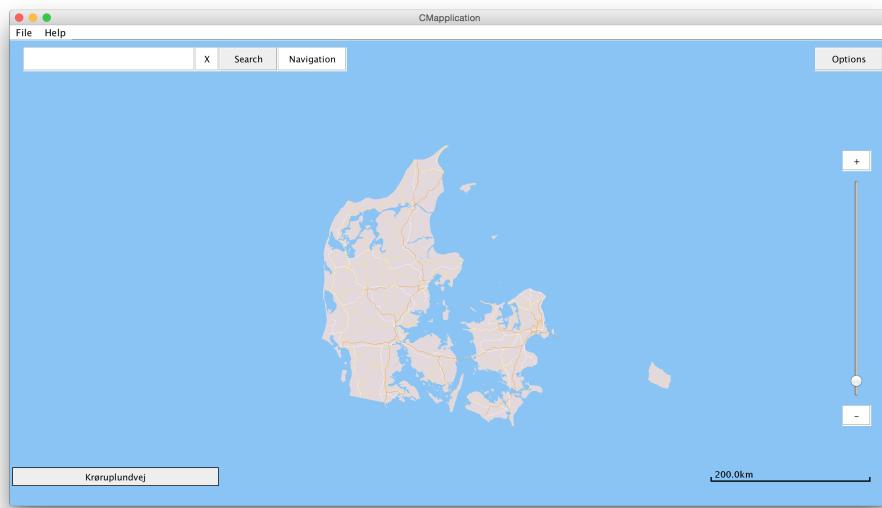
Det er også forsøgt at tydeliggøre hvilken vej, som brugeren hoverer over med musen. Disse veje vil blive fremhævet med en gul farve, og vejens navn vil blive vist i boksen nederst venstre hjørne i vinduet.

Udover det ovenstående, har vi også implementeret tastegenveje på tastaturet, som lader brugeren hurtigt navigere rundt i programmet eller flytte rundt på kortet. Eksempelvis ved addressesøgning kan brugeren trykke på pil-tast-ned, for at vælge en adresse i forslagslisten. Her slipper brugeren for at fjerne hånden fra tastaturet, for så at navigere musen hen og klikke på en adresse.

Generelt er disse små overvejelser med til at forbedre brugeroplevelsen af programmet.

# Kapitel 5

## Brugermanual



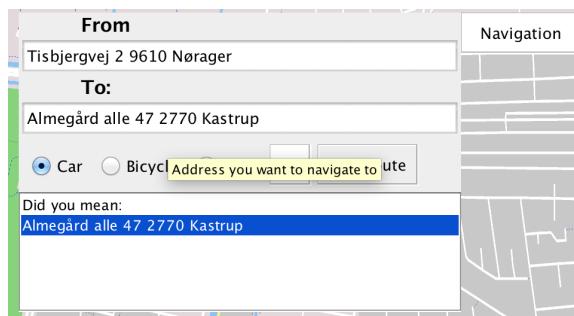
Figur 5.1: Programmet.

Når programmet køres, vises der først en loading skærm, som skal informere brugeren om, at der indlæses et kort. Dette er for at sikre at brugeren ikke skal tro at programmet er frosset eller termineret. Herefter oprettes der en grafisk brugergrænseflade, hvor der som standard er valgt en kortfil, heraf et kort af Danmark (se figur 5.1).

Brugeren kan derfra manipulere kortet ved zoom, panorering og rotation. Ved zoom kan brugeren enten vælge at benytte zoomslider, zoom knapper, scroll wheel eller "+" og "-" på tastaturet. Ved panorering, kan brugeren benytte "w", "s", "a" og "d" på tastaturet eller "trække" på kortet ved hjælp af musen. Der vises også en skaleringslinje, som viser den nuværende skalering i kilometer.

Ved standard adressesøgning, har brugeren mulighed for at søge efter en bestemt adresse i søgerfeltet øverst venstre hjørne. Brugeren kan enten indtaste den fulde adresse, eller vælge en adresse i forslagslisten, som bliver synliggjort, så snart søgerfeltet indeholder noget input. Herfra kan brugeren trykke på "search" for at foretage en søgning, eller "x" for at slette sit input. Ved søgning, vil en pointer blive markeret på kortet.

Føres musen hen over en vej på kortet vises vejens navn i en boks i nederste venstre hjørne og samtidig vil vejen blive fremhævet.



Figur 5.2: Ruteplanlægningspanel.

Trykker brugeren på "navigation knappen ved siden af søgerfeltet, vil et rute-navigationspanel (se figur 5.2) blive synliggjort. Panellet består af to tekstfelter, hvor brugeren kan indtaste til- og fra-destinationer samt en forslagsliste af adresser og transport valgmuligheder. Brugeren kan indtaste to adresser. Herefter vælger brugeren hvilket transportmiddel, som denne vil benytte. Dette kan være med bil, cykel eller ved at gå. Trykkes der herefter på knappen "Find route" vil en rute blive renderet med en tyk rød streg, og samtidig vil der blive produceret en tekstbaseret vejledning om hvordan brugeren kommer fra sin startdestination til slutdestination.

I programmet er det også muligt at gemme kortet i et bin-format. Dette sker via menuen i toppen. Klikker man på File og derefter save, kan man gemme kortet til et sted på ens computer. Programmet understøtter også en load-funktion, som er placeret samme sted. Her kan man indlæse kortdata med formatet .bin, .osm eller .zip.

Trykker brugeren på "Options"-knappen, der er placeret i øverste højre hjørne af programmet, ses muligheden for, at ændre på udseendet af kortet. Dette kan gøres ved hjælp af filtre, hvor man blandt andet kan vælge hvilke "Point of interest"-ikoner, der skal vises. Derudover kan der vælges en farveblind-venlig version af kortet, der gør det nemmere for farveblinde, at se veje, naturområder og vandområder. Den sidste checkbox slår antialiasing til og fra. Dette blødgør kanterne på alle elementer på kortet. Dets formål er at forbedre det æstetiske ved kortet.

# Kapitel 6

## Systematisk test

Et veldokumenteret og veltestet program er afgørende i softwareudvikling, og det har derfor været et stort fokusområde i dette projekt. For at sikre, at programmet er blevet testet på den mest hensigtsmæssige måde, har vi udført unit tests på flere elementer, og arbejdet med både whitebox- og blackbox testing som strategier for at finde eventuelle fejl i programmet. Til at teste programmet benyttes to forskellige test-frameworks: JUnit og Mockito. Samlet set har vi opnået 37,9% method coverage af hele programmet. Dog har vi haft fokus på de mest relevante modelklasser og har for modelpakken opnået 64,8% method coverage.<sup>7</sup>

### 6.1 Unittest

I et softwareprogram skal man være sikker på, at uanset hvordan brugeren vælger at benytte programmet, skal det håndtere fejl på en fornuftig måde. Dette kan for eksempel være, at give information til brugeren, om hvorfor programmet ikke fungerer ved en given sekvens af inputs. Til at sikre dette har vi unit testet store dele af vores program.

Et eksempel på unit tests, findes i klassen WayTest, der tester Way-klassens metoder. Klassen har samme felter som Way, og disse bliver initialiseret i *init-metoden*. Tagget *@Before* sørger for, at denne metode bliver kørt inden hver test. Herefter kommer en lang række tests, som tester Way-klassens metoder med forskellige inputs. Typisk kan fejl forekomme, når man ikke håndterer null-inputs rigtigt, og netop derfor bliver dette testet i de fleste tests.

Med hensyn til test af view/brugergrænsefladen, har vi valgt ikke at teste viewklasserne ved hjælp af et testværktøj såsom FEST<sup>8</sup>, fordi vi har en minimalistisk og begrænset brugergrænseflade, som er overskuelig at teste

---

<sup>7</sup>Se bilag 11.2

<sup>8</sup>Google (2015): Fest Framework

manuelt. Eksempelvis at trykke på en knap og validere dets reaktion. Dette sparer både tid og ressourcer, da det ikke kræver mere end observation. Alternativt kunne man, ved større programmer, som har mange flere komponenter end dette program, benytte et test framework, såsom det førnævnte FEST, der muliggør automatisering og simulering af de forskellige funktionaliteter for hver komponent. Dermed sikrer man at alle komponenter bliver testet.

## 6.2 Manuel test

Vi har udført manuel test af forskellige dele af programmet, blandt andet brugergrænsefladen. Når branchen var klar til at skulle merges med masterbranchen, blev følgende procedure udført for manuel test:

### Manuel test af brugergrænseflade

- ✓ Søgefeltet, Zoom slideren, Options-komponentet, menu-baren og adresseringsfeltets funktioner er testet.
- ✓ Disse er testet i unit tests, så overfladisk test er tilladt
- ✓ Panorering kører flydende på alle zoom levels
- ✓ Test at anti-aliasing kan slås til og fra
- ✓ Scroll-zoom bevæger sig mod og væk fra musens position på kortet
- ✓ Kilometer-målestokken virker fornuftig
- ✓ Kortet tegner veje ved tættere inspektion
- ✓ Kortet tegner kartografiske elementer når der er zoomet meget ind
- ✓ Kystlinjer er lukkede
- ✓ Test at alle filtre i options-komponentet filtrerer som forventet
- ✓ Test at kortet tegnes med nyt farvetema ved brug af "Colorblind mode"
- ✓ Test at ikonerne for Til- og Fra punkterne i rutevejledning bliver tegnet
- ✓ Udfør mindst 1 rutevejledning, som minimum på 50 km
- ✓ Test af tastaturgenveje: [+/-] (zoom ind/ud), [wasd] (pan), [qe] (rotér), **r** (gem loadet kort i binært format, til filen: mapsave.bin)

Denne test er med til at sikre, at kortet virker efter hensigten, og at de nye funktioner der bliver indsat i programmet ikke ødelægger noget der virkede før.

Det ville selvfølgelig være optimalt, at teste viewdelen via unit tests eller FEST, men den manuelle test giver en god indikation af, at de testede metoder virker efter hensigten.

## 6.3 Blackboxtest

I en blackbox test benytter man kravspecifikationen til at opstille testcases, som afprøver hvorvidt programmet fungerer efter hensigten og dermed opfyl der kravspecifikationen. Det vigtigste er derfor, at gennemgangen af kravene er nøje og præcise. Eksempelvis i følgende krav: “*Unobtrusively show (either continuously or on hover) the name of the road closest to the mouse pointer, e.g., in a status bar.*”. Her er der benyttet både uspecifieret tilstande og eksempler, og det er dermed vigtigt at reflektere over disse. Generelt set benytter vi blackbox testing i de fleste af vores unit tests. Når en unit test er blevet lavet, er der på baggrund af klassens API blevet udtænkt en række lovlige og ulovlige input til denne.

Kravspecifikationen kan altså benyttes til at aflæse, hvilke test cases programmet bør have og hvilke outputs der kan forventes på et givent input. I blackbox test er det vigtigste, at man overvejer hvilke inputs og hvad der bør testes for at bevise, at programmet kan håndtere krævende og ekstreme inputs. Her kigger man altså ikke på kildekoden, men på API'en og det forventede output på et bestemt input.

Der er blandt andet benyttet blackbox til at teste A\*-implementationen. Her er der undersøgt om implementationen af A\* returnerer det korrekte output, og vigtigst af alt, at den ikke godtager et ugyldigt input. Dette er gjort ved at oprette en ugyldig graf og tjekke, at A\* ikke finder en længde mellem to punkter i denne graf.

Andre vigtige funktioner, som er testet med blackbox-fremgangsmåden er adresseparseren, addressesøgningen, fil inputs samt opbygning af grafen. Blackbox tests er generelt det, der er benyttet mest i test suiten. Under blackbox tests er der flere steder benyttet Unitest-princippet, da det er blevet tilstræbt at gøre programmet så modulært som muligt. Dette gør, at Unitests kan benyttes til at teste de enkelte metoder hver for sig. Naturligvis sikrer det ikke mod alle fejl, men vi kan forsøge at teste systemet med bedste evne for at finde så mange huller og fejl som muligt.

## 6.4 Whiteboxtest

Whitebox-test er en teststrategi, hvor der fokuseres på kodens interne struktur og hvor kildekoden benyttes til at forudsige, hvad der forventes af de enkelte metoder. Der er benyttet whiteboxtesting i unitests og der er generelt forsøgt at opnå statement coverage alle steder, hvor det er aktuelt. Nogle steder er testet mere grundigt og det er forsøgt at opnå branch-coverage.

Der er også blevet foretaget en dybdegående whiteboxtest af QuadTree klassen (se 11.1.1). QuadTree-klassen er en af de vigtigste klasser i programmet, fordi klassen og dens funktionalitet er fundamentet for, hvornår og hvad, der skal blive renderet. Hvis QuadTree klassen ikke fungere korrekt,

kan det betyde en forringelse af ydevnen eller at elementer ikke bliver renderet. Derfor er det vigtig, at det undersøges, at QuadTree-klassen fungerer efter hensigten.

Under testprocessen er det blevet efterstræbt som minimum, at opnå fuld branch-coverage og hel code-coverage. Følgende metoder er inkluderet i whiteboxtesten:

- insert()
- subdivide()
- rangeSearch()

Ved test af insert()-metoden, undersøges det om indsættelsen af forskellige typer af objekter i QuadTree-klassen bliver håndteret korrekt, og at disse bliver sat i de rigtige quads. Ved subdivide()-metoden undersøges det om QuadTree bliver korrekt opdelt i fire nye quads, så snart QuadTree's antal elementer rammer kapaciteten. Heraf er der opstillet tre forskellige datasæt med følgende expectancy table, figur 6.1 (se 11.2 for coverage table).

**Expectancy table**

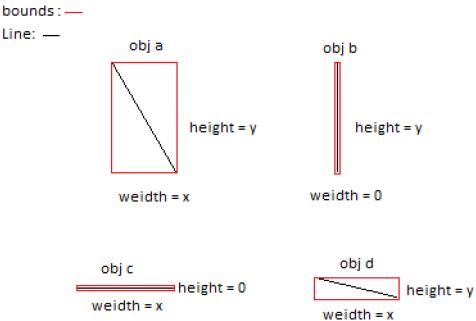
INPUT DATASÆT	INPUT INDHOLD	FORVENTET OUTPUT	AKTUELT OUTPUT
A	5 lovlige bounds (1 bound <u>intersecer</u> 3. og 4. kvadrant)	QuadTree med 4 children. Størrelse af child0 = 1 Størrelse af child1 = 1 Størrelse af child2 = 2 Størrelse af child3 = 2	QuadTree med 4 children. Størrelse af child0 = 1 Størrelse af child1 = 1 Størrelse af child2 = 2 Størrelse af child3 = 2
B	1 ulovlig bound (ligger uden for quad bounds)	QuadTree indeholder ingen ways.	QuadTree indeholder ingen ways.
C	6 lovlige bounds (ligger placeret tæt på, eller krydser quad bounds)	QuadTree med 4 children bliver oprettet korrekt.	QuadTree med 4 children bliver oprettet korrekt.

Figur 6.1: Expectancy table

Her testes både lovlige inputs, ulovlig inputs og grænseinputs, som kan have uforudsigelige effekter. Under testing af insert()-metoden stødte vi på flere logiske bugs i form af, hvordan vores if-statements tjekkede bounds.

En bug der for eksempel blev fundet under whitebox-testing var, at når et objekt var på en lige vertikal eller horizontal linje, havde linjens afgrænsende boks ikke en bredde eller højde, som også er illustreret på figur 6.2. Det havde den konsekvens, at linjen ikke vil blive inkluderet i kortrenderingen.

Ved testing af rangeSearch()-metoden, blev det undersøgt om programmet returnerede korrekte værdier i et given search bounds. Herunder er den opstillede expectancy table for testen, figur 6.3 (se 11.2 for coverage table).



Figur 6.2: QuadTree bug

Expectancy table

INPUT DATASÆT	INPUT INDHOLD	FORVENTET OUTPUT	AKTUELTT OUTPUT
D	Searchbounds rammer et quad med flere objekter, men rammer kun 1 objekt	Liste med 1 objekt	
E	Searchbounds rammer et quad der indeholder ét object (som den rammer).	Liste med 1 objekt	
F	Searchbounds rammer et quad der er tomt	Tom liste	
G	Searchbounds er uden for bounds	Tom liste	
H	Searchbounds skærer to quads og indeholder 3 ways	Liste med 3 objekter	

Figur 6.3: Expectancy table

Denne whitebox test fandt ikke yderligere bugs, og det må derfor antages at kildekoden her virker efter hensigten.

Testing kan aldrig garantere, at et program er fejlfrit, men kan give en god indikation af, at det er. Det har derfor være essentielt for processen at foretage hyppige og dækkende tests. Derfor er de forskellige teststrategier, som er beskrevet i ovenstående, benyttet i alle aspekter af udviklingsprocessen. Især de automatiserede test, som kunne gøres efter hver ny implementation for at sikre, at koden stadig var intakt.

# Kapitel 7

## Procesbeskrivelse

### 7.1 Gruppearbejdet

Gruppedannelsen var struktureret af forelæser og inden gruppedannelsen, havde alle taget en belbin teamrolle-test på internettet. Derfor var alle klar over, hvilke af Belbins grupperoller, vi hver især repræsenterede. Gruppen blev dermed dannet på baggrund af testen, og da der var nogle fælles forventninger til både gruppeprocessen og det ønskede resultat.

Gruppearbejdet har generelt fungeret rigtig godt. Alle gruppemedlemmer har været klar over, hvilke kompetencer de hver især har kunnet bidrage med. I struktureringen af gruppearbejdet har det været nødvendigt, at uddeleger ansvar og roller for at nå alle features og krav til projektet.

Arbejdsopgaverne blev uddelegeret løbende, men processen er dog blevet forsøgt struktureret ved hjælp af et Gantt-diagram (se bilag 11.3.1) fra start. Fokus var fra start, at nå de krav, der var blevet stillet i projektbeskrivelsen. Hver feature blev derfor oprettet som en task i projektstyringsværktøjet (Trello) for herefter at blive uddelt til enkelte gruppemedlemmer. Både ud fra hvilke features, som medlemmerne mente, at de kunne løse bedst, men også ud fra hvilke features, som var forudsættende for andre features. Eksempelvis forudsættede implementationen af Dijkstra-algoritmen, at der var oprettet en graf over alle edges og vertices i kortet.

Til gruppemedlemmerne blev der i fællesskab gennemgået hver feature og ændringer i andre klasser. Dette har betydet, at alle gruppemedlemmer har haft en føeling med, hvor langt processen var i forhold til Gantt-diagrammet, projektplanen og programmet som helhed.

Den største udfordring i processen har været de forskellige styrker og erfaringer som var i gruppen. Nogen har programmeret meget før, mens andre ikke har så meget erfaring og skal bruge mere sparring i arbejdet. Dette har betydet, at store og komplekse features, som Dijkstra's algoritme, blev tildelet til mere erfarte programmører. Som resultat blev arbejdsbyrden i nogle perioder større for de erfarte programmører end dem med mindre erfaring.

Især da deadline nærmede sig og implementationerne skulle færdiggøres.

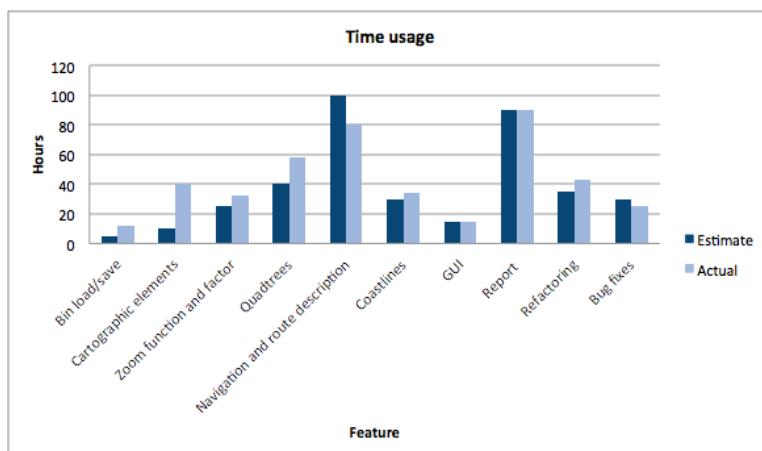
## 7.2 Redskaber

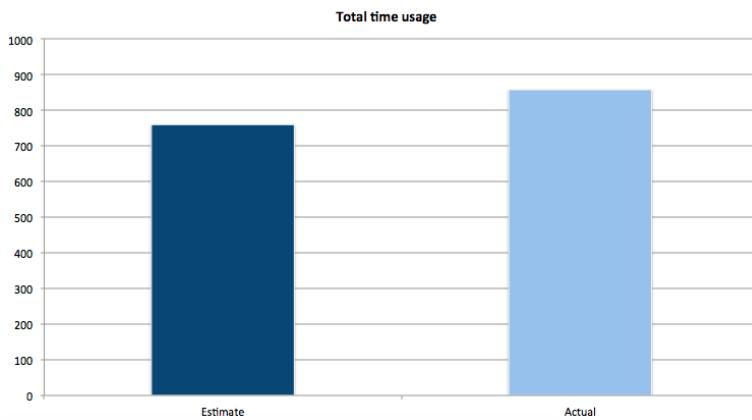
For at imødekomme og konkretisere overvejelser om programmet (beskrevet i problemanalysen) og dets krav, har vi blandt andet benyttet substantiv-/verbum metoden. Dette er gjort for at klarlægge programmets back-end. Ved gennemgangen af opstillede usecases, nedskrevne overvejelser og projektbeskrivelsen blev klasser, felter og dets samspil defineret ved at gennemgå substantiver. Eksempelvis ”Rutenavigation”, mens definerede handlinger såsom ”navigering” ses som en handling og dermed metoder.

Et andet nyttigt redskab i processen, som blev benyttet, er branches i version styringsværktøjet, heraf ”git”. Dette har betydet, at hvert medlem af gruppen kunne arbejde på en del af programmet uden, at det ville påvirke andres arbejde. Når et stykke arbejde var færdig, oprettede vedkommende et pull-request, som en form for notificering, om at arbejdet er færdig. Et andet medlem kunne så gennemgå koden, komme med feedback eller ”merge” arbejdet ind i ”master branchen”. Dette har også understøttet ideen om, at uddeletere opgaver, men samtidig gøre det muligt for alle gruppemedlemmer at være involveret i implementationen og forståelsen af programmets opbygning.

## 7.3 Planlægning og tidsestimat

Sammenhængen mellem gruppens tidsestimat og det faktiske tidsforbrug på de forskellige features har været en udfordring. Blandt andet blev der brugt længere tid på coastlines end først antaget. Dette er illustreret i følgende graf:





Ser man på estimatet pr. feature er der få steder, hvor der er ubalance, mens forskellen i totalen ser mere drastisk. Overordnet set er der estimeret 50 timer for lidt, hvilket svarer til en stigning i tidsforbruget på 13%. I et stort projekt som dette, er det faktiske tidsforbrug dog ikke ude af proportioner, men det har betydet, at de ekstra features, som vi ønskede at implementere ikke blev det.

Det vigtigste vi har lært af denne process er vigtigheden af, at kunne tidsestimere korrekt og dermed også kunne uddeletere opgaver mere hensynsmæssigt. Fremadrettet ønsker vi, at have et gruppearbejdet, som vil være lige så ærligt og konstruktivt som dette, men ønsker også, at arbejdsbyrden bliver bedre fordelt mellem medlemmerne, og at kompetencer ikke overskygger læringskurverne for øvrige gruppemedlemmer.

I sidste ende er vi dog tilfredse med processen.

# Kapitel 8

# Produktrefleksion

Udviklingen af det endelige produkt har krævet prioritering af features, og en grundig analyse af hvilke løsninger, der måtte findes til projektets krav.

Det endelige produkt opfylder de krav, som analysen af projektet medførte:

1. Kunne læse og gemme kortet hurtigt
2. Vise elementer i kortet genkendeligt og forståeligt
3. Give brugeren mulighed for at interagere med kortet
4. Give brugeren mulighed for at søge efter en adresse
5. Kunne foretage en rutevejledning for forskellige transportmidler
6. Køre så hurtigt, at det er praktisk at bruge

## 8.1 Optimering af køretid

Kravet om et program med en hurtig og praktisk køretid har resulteret i en svær, men lærerig process. Denne løbende optimering af køretider, har bestået af flere forskellige opgraderinger, der alle bidrager til den samlede oplevelse af et effektivt og praktisk navigationssystem.

I projektets start vidste vi ikke, hvilke metoder der skulle benyttes, og hvordan disse skulle implementeres. Ved hjælp af egne erfaringer fra eksperimenter og analyser, lykkedes det, at finde en datastruktur, der har givet det endelige produkt en fornuftig køretid. Herunder muligheden for at indlæse binære filer fra start. Det er herudover andre opgraderinger, som implementationen af QuadTree og senere A\* som især har optimeret programmets hastighed.

## 8.2 Optimering af hukommelsesbrug

Den største udfordring i programmet har været at balancere programmets brug af hukommelse med kildekodens læsbarhed og forståelse. Men da der ikke var defineret et specifikt krav omkring hukommelsesbrug, er andre features blevet prioriteret over dette. Havde projektet strukket sig over længere tid, ville det have været muligt at reducere hukommelsesbrug yderligere, men vores ressourcer blev i stedet fokuseret på de obligatoriske krav.

Der er dog alligevel foretaget flere tiltag for at minimere hukommelsesbrug. Dette er blandt andet:

- Objekter, der bliver oprettet mange gange, indeholder så lidt data som muligt (ex: Way-objekter)
- Implementation af en Stringpoll, der gør at programmet kan refere til en enkelt tekststreng, istedet for at gemme identiske strenge igen.
- Brugen af Float koordinator frem for double.

I processen opstod der en konflikt mellem at sikre kodens læsbarhed og samtidig bevare et lavt hukommelsesbrug. Eksempelvis refaktorerede vi tidligt i processen OSM-håndteringen samt de eksisterende datastrukturer: Element, Node, Way og Relation. Enhver type element lå i et `HashMap<Long, Element>`, hvilket betød at koden var letlæselig og overskuelig. Dette gav desværre nogle problemer senere, i forhold til brug af hukommelse. Store HashMaps med Long som key, viste sig at bruge meget hukommelse - også med en optimeret `LongHashMap`<sup>9</sup> klasse. Vi valgte derfor, kun at benytte et `HashMap` til nodes i selve OSM-Handleren, der efterfølgende ville blive sendt til Garbage-collectoren. Dette har betydet, at programmet nu bruger betydeligt mindre hukommelse - særligt i OSM-håndteringen.

Det primære fokus har dog gennem hele projektet været på den endelige kodes læsbarhed.

## 8.3 Programfejl og mangler

I et større projekt er det nødvendigt også at prioritere imellem fejl og mangler. Herunder gennemgås de fejl og mangler det færdige projekt indeholder.

### 8.3.1 Mangler ved regulære udtryk

I forhold til kravet om adressesøgning var det nødvendigt, at implementere en løsning, der sikrede, at programmet kunne opslå gyldige adresser, og vi valgte at løse dette med regex. Det regulære udtryk som programmet bruger, har dog nogle faldgrupper, hvilket er forventeligt, da implementation af

---

<sup>9</sup>LearnIT (2015): `LongHashMap.java` af Troels Bjerre Sørensen

et komplet regex er nærlig umulig. En faldgruppe forekommer hvis brugeren prøver at indtaste sal og side til en adresse, eksempelvis: "Skolegade 2b 2. tv 7100 Vejle". En anden er bindestreger og punktummer i adresser, som de nuværende regulære udtryk ikke helt kan validere korret. Vores regulære udtryk tilader ikke bindestreger og punktummer som inddeler. Eksempelvis kan brugeren ikke skrive "Skole-gade-2b-7100-Vejle". Dog så kan den godt godtage gadenavne som indeholder disse tegn. Heraf "H. C. Andersens Boulevard 1550 København"

Disse prioriteringer er taget på baggrund af, at majoriteten af gadenavne og bynavne følger en anden standard, som vi med udvidelsen af de nuværende regex ikke ville kunne parse. Dette er dog forsøgt løst ved en forslagsliste. Brugere kan længe før de er færdige med deres indtastning, få foreslægt en række adresser, som vi er sikre på vil parse, og dermed undgås det, at brugeren forsøger med et invalid input.

### 8.3.2 Liste over kendte fejl

Det færdige program er desværre ikke fejlfrit og forneden er listet alle kendte fejl i programmet.

- Højre- og venstre vinkelfejl
  - Den tekstbaserede rutebeskrivelse kan nogle gange tage fejl af højre- og venstresving.  
Dette skyldes, at det i specielle situationer kan være svært at vurdere hvorvidt man rent faktisk drejer, eller fortsætter lige ud. Fejlbeskrivelsen for disse særlige tilfælde kan fikses, men da det sker sjældent, var dette ikke en høj priorititet, og der blev ikke brugt tid på at rette denne fejl.
- Rotationsfejl
  - Ved rotation af kortet, bliver projektionen forvrænget. Dette skyldes at, transformationen skalerer kortdata på x-aksen for at fremvise kortet ordentligt på en flade. Når kortet roteres, vil denne skalering ikke rotere med, og kortet bliver visuelt forvrænget. Kortet forsvinder også når brugeren roterer og panorerer eller zoomer på samme tid. Dette skyldes, at programmet prøver at foretage flere forskellige transformationer, der resulterer i, at programmet fejler renderingen af kortet. Et lignende problem skete også ved zoom og panorering på samme tid, men dette blev fikset i version 0.9. Et løsningsforslag kunne være, at sørge for at brugeren ikke kan zoome og panorere på samme tid.
- Rutevejledningsfejl

- I særtilfælde kan navigationsfunktionen ikke finde en rute fra punkt A til punkt B, hvis brugeren eksempelvis vælger bil som transportmiddel. Dette skyldes, at når A\* bliver kørt, bliver nearest-neighbor metoden kørt i punkt A. Hvis den nærmeste vej for eksempel er en cykelsti, vil programmet komme med en fejlmeddeelse, fordi biler ikke må køre på den slags stier. Dette kunne fikses ved at sørge for, at nearest-neighbor tjekker hvilken type vej den returnerer alt efter hvilken transportmiddel, der er valgt.
- Visuelle interface-fejl
  - Når brugeren zoomer, vil det kunne ses, at baggrunden på det komponent som zoom-bjælken er placeret i, bliver fejlrenderet. Dette skyldes måden Swing rendererer elementer på. Der kannes ikke finde en løsning på problemet, men vi har blandt andet forsøgt at kalde repaint-metoden efter alt er blevet sat op og tegnet, men dette virkede ikke.

## 8.4 Ekstra features

Det færdige produkt opfylder alle krav, der er opstillet i Krav-afsnittet. Heriblandt navigationsfunktionen, der gør det muligt for brugeren, at vælge om man vil rejse med bil, cykel eller gående. Programmet finder den hurtigste vej ved bilrejse (dette vil sige, at der er taget højde for hastighedsbegrænsninger) og den korteste vej ved cykel eller gående rejser. Denne feature vil kunne opgraderes med følgende udvidelser:

### 8.4.1 Rejse med mellempunkt

Ønsker brugeren, at rejse fra punkt A til punkt B via punkt C, skal der kunne indtastes et “via”-punkt i et tredje adressefelt. Dette forudsætter blot, at programmet finder den hurtigste rute fra A til C og derefter fra C til B. Denne udvidelse relativt nem at implementere.

### 8.4.2 Drag-and-drop start- og slutpunkt

Denne funktion ville gøre det muligt at definere start- og slutpunkt ved at klikke på kortet, og brugeren kan derfor få rutevejledning mellem to punkter som brugeren ikke nødvendigvis kender adressen på. Implementationen kræver en konvertering af musens koordinator til længde- og breddegrader. Herefter kan programmet benytte den konverterede data til at definere start- og slutpunkter.

De krav som blev opstillet i den udleverede projektbeskrivelsen, er i dette software program blevet løst med en række forskellige datalogiske værktøjer.

Som gennemgået i ovenstående kan det endelige produkt optimeres yderligere - både i form af hukommelsesbrug, køretid, men også i forhold til yderligere funktioner. Der var også flere løsninger, som man kunne have brugt til opdelingen af data eksempelvis. Alternativt eksempelvis KD-træer eller R-træer, som har andre fordele og kan være bedre i større datastrukturer. Især sidstnævnte er blandt andet grundet til, at de ikke er blevet benyttet i dette program.

Det færdige produkt reflektere altså en række valg og løsninger, som gruppen har besluttet førte til et optimalt program indenfor den givne tidsramme.

# Kapitel 9

## Konklusion

Strukturering af et større programmeringsarbejde kræver overblik og planlægning. Gennem processen af dette projekt har en god eksekvering af projektet især stillet krav til tidsestimering og prioritering af opgaver. Den store selvstændige del af kurset har også betydet, at udviklingen af programmet har været præget af “trial and error”-tilgangen. Her afprøvede og eksperimenterede vi med forskellige løsninger for at finde frem til det mest optimale slutprodukt.

En af de største udfordringer var blandt andet at finde balancen mellem letlæselig kodestruktur og optimal yddeeve. Vi har igennem udfordringer som denne, lært at bearbejdningen af store datasæt kan kræve, at der konstant må prioriteres mellem features. Der har dog i processen været størst fokus på de opstillede krav og løsning af disse heriblandt rutevejledning og addressesøgning. Disse krævede en større datalogisk analyse og tilgang, men blev opfyldt med implementationen af udvalgte algoritmer samt modificeringer af datastrukturen undervejs

Programmet er blevet testet med både Whitebox- og Blackbox-testing for at sikre, at alle faldgrupper er inddækket. Programmets både simple og intuitive grafiske brugergrænseflade, gør det nemt, at benytte softwaresystemet for selv nye brugere.

Det færdigudviklede produkt munder ud i et interaktivt kort med muligheder såsom addressesøgning og rutevejledning. Programmet opfylder alle projektets krav og har derudover ekstra funktionalitet i form af “search-as-you-type” samt implementationen af A\*.

# Kapitel 10

## Referencer

### 10.1 Sekundær litteratur

1. Sedgewick, Robert; Wayne, Kevin (2011): Algorithms. Addison-Wesley, 4. udgave.
2. Sestoft, Peter (2000): ”Udformning af rapporter”. IT Universitetet i København, IT-C udgave
3. Barnes, David J.; Kölking, Michael (2012): Objects First With Java. Pearson, 5. udgave.

#### 10.1.1 Internetmateriale

1. Open Street Maps (2015): About. På: <http://www.openstreetmap.org/about>.
2. SAX Project (2015): SAX-Parser information. På: <http://www.saxproject.org/>.
3. Stanford University (2015): Introduction to A\*: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>.
4. Computer Science VirginiaTech (2015): Organizing spatial data. På: <http://courses.cs.vt.edu/~cs3114/Spring10/Notes/T06.PRQuadTrees.pdf>.
5. GitHub ITU (2015): Version 0.8. På: <http://github.itu.dk/abhn/GroupC/blob/v0.8/src/main/java/dk/itu/cvitamin/model/Element.java>.
6. Java.net (2015): OpenJDK Path2D. På: <http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d/src/share/classes/java.awt/geom/Path2D.java>.
7. Google (2015): FEST Framework. På: <https://code.google.com/p/fest/>.

8. LearnIT (2015): LongHashMap.java af Troels Bjerre Sørensen. På: <https://learnit.itu.dk/mod/forum/discuss.php?d=658>.

## 10.2 Ikoner

1. IconFinder (2015): MapIcons. På:  
<https://www.iconfinder.com/iconsets/mapicons>.  
Licens:  
<http://creativecommons.org/licenses/by-sa/3.0/>.



# Kapitel 11

## Bilag

### 11.1 Whiteboxtest

#### 11.1.1 Expectancy tables

##### Insert Divide

Coverage table

VALG	INPUT EGENSKABER	INPUT DATASÆT
1 SANDT	Object bounds width eller height er nul	C
1 FALSK	Object bounds width og height > 0	A
2 SANDT	Object bounds er uden for QuadTree bounds	B
2 FALSK	Object bounds er inden for QuadTree bounds	A
3 SANDT	QuadTree children er null	A (første 4 indsættelser)
3 FALSK	QuadTree children er ikke null	A (efter 4. indsættelse)
4 SANDT	Object bounds skærer 1. eller 2. kvadrant	A
4 FALSK	Object bounds skærer ikke 1. eller 2. kvadrant	A
5 SANDT	Object bounds skærer 2. kvadrant	A
5 FALSK	Object bounds skærer ikke 2. kvadrant	A
6 SANDT	Object bounds skærer 1. kvadrant	A
6 FALSK	Object bounds skærer ikke 1. kvadrant	A
7 SANDT	Object bounds skærer 3. eller 4. kvadrant	A
7 FALSK	Object bounds skærer ikke 3. eller 4. kvadrant	A
8 SANDT	Object bounds skærer 3. kvadrant	A
8 FALSK	Object bounds skærer ikke 3. kvadrant	A
9 SANDT	Object bounds skærer 4. kvadrant	A
9 FALSK	Object bounds skærer ikke 4. kvadrant	A
10 SANDT	Nodes er null	A
10 FALSK	Nodes er ikke null	A
11 SANDT	Størrelsen af node-listen er større end makskapaciteten af QuadTree	A
11 FALSK	Størrelsen af node-listen er ikke større end makskapaciteten af QuadTree	A
12 NUL	Der er 0 nodes i node-listen når QuadTree deler sig	(sker aldrig, maxcapacity er minimum 1)
12 EN	Der er 1 node i node-listen når QuadTree deler sig	(sker aldrig, maxcapacity er minimum 1)
12 MANGE	Der er mange noder i node-listen når QuadTree deler sig	A

## Rangeseach

Coverage table

V ALG	INPUT EGENSKABER	INPUT DATASÆT
1 SANDT	Hvis søgebounds skærer eller rammer QuadTree bounds	
1 FALSK	Hvis søgebounds ikke skærer eller rammer QuadTree bounds	
2 SANDT	Hvis children ikke er null (QuadTree har ikke delt sig)	
2 FALSK	Hvis children er null (QuadTree har delt sig)	
3 NUL	Hvis children er null (QuadTree har delt sig)	
3 EN	Sker aldring (der er altid 4 børn)	Sker aldring
3 MANGE	Præcis 4 børn	
4 SANDT	Nodes!=null, Hvis det givne QuadTree indeholder object(s)	
4 FALSK	Nodes=null, Hvis det givne QuadTree ikke indeholder objects	
5 NUL	Nodes=null, Hvis det givne QuadTree ikke indeholder objects	
5 EN	Nodes.size()=1, Hvis QuadTree indeholder præcis 1 objekt	
5 MANGE	Nodes.size()>1, hvis QuadTree indeholder mere end 1 objekt	
6 SANDT	Hvis søgebounds skærer eller rammer objektets bounds	
6 FALSK	Hvis søgebounds hverken skærer eller rammer objektets bounds	

Expectancy table

INPUT DATASÆT	INPUT INDHOLD	FORVENTET OUTPUT	AKTUELT OUTPUT
D	Searchbounds rammer et quad med flere objekter, men rammer kun 1 objekt	Liste med 1 objekt	
E	Searchbounds rammer et quad der indeholder ét object (som den rammer).	Liste med 1 objekt	
F	Searchbounds rammer et quad der er tomt	Tom liste	
G	Searchbounds er uden for bounds	Tom liste	
H	Searchbounds skærer to quads og indeholder 3 ways	Liste med 3 objekter	

## 11.2 Coverage tables

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	43.8% (49/ 112)	37.9% (263/ 694)	40.6% (1447/ 3567)

Coverage Breakdown

Package ▲	Class, %	Method, %	Line, %
dk.itu.cvitamin	0% (0/ 1)	0% (0/ 2)	0% (0/ 1)
dk.itu.cvitamin.controller	13% (6/ 46)	8% (8/ 100)	6.9% (23/ 334)
dk.itu.cvitamin.model	86.2% (25/ 29)	64.8% (177/ 273)	61.7% (1083/ 1756)
dk.itu.cvitamin.model.exceptions	100% (2/ 2)	100% (2/ 2)	100% (4/ 4)
dk.itu.cvitamin.model.routing	81.8% (9/ 11)	60.3% (44/ 73)	57.3% (200/ 349)
dk.itu.cvitamin.util	71.4% (5/ 7)	57.1% (20/ 35)	68.5% (89/ 130)
dk.itu.cvitamin.view	33.3% (2/ 6)	12.4% (12/ 97)	8% (48/ 597)
dk.itu.cvitamin.view.components	0% (0/ 10)	0% (0/ 112)	0% (0/ 386)

### 11.2.1 Model coverage

Coverage Summary for Package: dk.itu.cvitamin.model

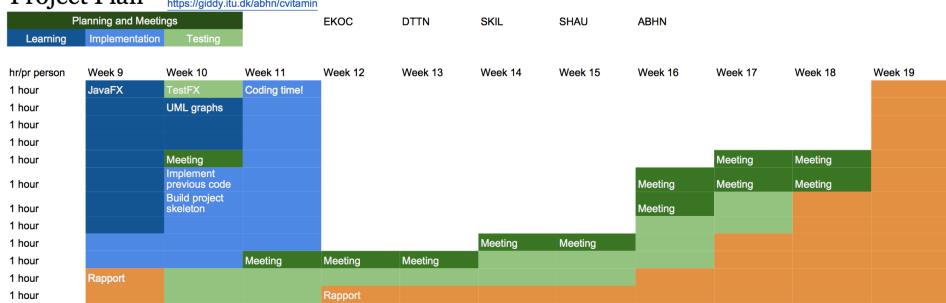
Package	Class, %	Method, %	Line, %
dk.itu.cvitamin.model	86.2% (25/ 29)	64.8% (177/ 273)	61.7% (1083/ 1756)
<b>Class ▲</b>	<b>Class, %</b>	<b>Method, %</b>	<b>Line, %</b>
Address	100% (2/ 2)	84.6% (11/ 13)	81.2% (26/ 32)
AddresssearchModel	100% (1/ 1)	81.8% (9/ 11)	84.6% (77/ 91)
Bounds	100% (1/ 1)	100% (1/ 1)	100% (2/ 2)
CartographicElements	100% (1/ 1)	100% (3/ 3)	89.7% (52/ 58)
Coastline	100% (1/ 1)	85.7% (6/ 7)	66.7% (8/ 12)
Element	100% (2/ 2)	100% (3/ 3)	100% (22/ 22)
FileHandler	100% (1/ 1)	84.4% (27/ 32)	78.5% (172/ 219)
LongHashmap	100% (1/ 1)	61.5% (8/ 13)	68.8% (44/ 64)
Member	100% (1/ 1)	14.3% (1/ 7)	35.7% (5/ 14)
Model	100% (1/ 1)	70.6% (12/ 17)	67.3% (66/ 98)
Node	100% (1/ 1)	100% (4/ 4)	100% (7/ 7)
OSMHandler	100% (2/ 2)	100% (31/ 31)	97.8% (226/ 231)
POI	100% (1/ 1)	100% (5/ 5)	85.7% (12/ 14)
Path2D	50% (4/ 8)	28.7% (27/ 94)	24.2% (165/ 683)
QuadTree	100% (2/ 2)	100% (13/ 13)	100% (97/ 97)
Relation	100% (1/ 1)	40% (2/ 5)	54.5% (6/ 11)
Way	100% (2/ 2)	100% (14/ 14)	95% (96/ 101)

## 11.3 Gruppearbejdet

### 11.3.1 Gantt-diagram af projekt plan

Project Plan

<https://giddy.itu.dk/abhn/cvitamin>



## 11.4 Version changelog

### 11.4.1 V1.0 - FINAL FRONTIER OF GRAPES

#### Features:

- Load OSM, ZIP and BIN-files via GUI or by program argument.
- Save Map-data to BIN-file.
- Default binary file of Denmark is loaded if no other argument is specified.
- Paint different Way-types to the screen, using different type of colors to indicate type of road.
- Paint different cartographic elements (parking-lot, bar, etc.)
- Zoom and Pan map using mouse or keyboard.
- Layout adjusted when window size changes.
- Show road name closest to mouse in a status bar.
- Address searching (typed as a single string).
- Shortest path between two points, by entering address of both points (Using A\*).
- Choose between car, biking and walking for route navigation.
- Car route using fastest route, biking and walking using shortest route.
- Mouse and keyboard interaction implemented.
- Textual description of route output in GUI.
- Current zoom level indicated in graphical scalebar.
- Toggle color blind mode to change visual appearance.
- Reasonably quickly to use and navigate.
- Using multiple QuadTrees to store map data.

**Extra feature:** suggestionlist of addresses when searching.

**Total cost Cumulative cost: (1h = DKK 625)**

Total : 29.375 + 30.000 + 64.375 + 90.000 + 130625 + 60625 + 75000 +  
100000       +       62500       =       642500,-       kr

## **11.4.2 RC1.0 - HOLY VITAMIN C**

**manglerde features:**

**Features:**

- Søgning af korteste rute/hurtigste rute for flere forskellige former for transportmiddel
- Implementation af rutevejledning
- Bugfixes

## **Wish list for V1.0 - FINAL FRONTIER OF GRAPES**

- Small bugs fixed

**Feedback request:**

**Time usage in man hours:**

- 3 hours: group meeting (4 personer)
- 30 hours: Shortest path (2 personer)
- 15 hours: Route description (2 personer)
- 15 hours: nearestNeighbor (2 personer)
- 7 hours: Code cleanup and JDOC (1 person)
- 5 hours: bugfixes (3 personer)
- 8 hours: Jar file fix (1 person)
- 5 hours: rapport (4 personer) (Average time)
- 8 hours: fix bin save/load (1 person)

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 + 90.000 + 130625 + 60625 + 75000 + 100000 = 580000,-$

### **11.4.3 V0.10 - Honorable Sudachi**

#### **Bugs/manglerde features:**

- FileHandler tests bugger pga nye quadtrees
- Åbne coast line bugger (stadigvæk)
- Gradle virker pt ikke med seneste implementation

#### **Features:**

- Oprettet flere tests
- Pannorerer og zoomer til søgte adresse
- Full funktionel navigations system (Mangler stadig korteste rute)
- Optimeret FileHandler
- KeyBindings virker
- Små usability forbedringer (eg, når brugeren trykker "ned" i søge feltet, så bevæger han automatisk ned til forslagslisten.)
- CoastLines virker

#### **Wish list for RC.1. - Holy Vitamin C:**

- Søgning af korteste vej "Dejkstra"
- Visning af vejnavn

#### **Feedback request:**

Om vores implementation af QuadTree er optimal.

#### **Time usage in man hours:**

- 3 hours: group meeting (4 personer)
- 5 hours Bugfixes (4 personer)
- 16 hours: QuadTree optimization (1 person)
- 13 hours: tests writing (1 person)
- 14 hours: closing coastlines (1 person)
- 20 hours: Navigation panel(1 person)

- 5 hours: Keybindigs fix
- 10 hours: rapport skrivning
- 10 hours POI

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 + 90.000 + 130625 + 60625 + 75000 = 480000,-$

#### **11.4.4 V0.9 - Explosive Grapefruit**

##### **Bugs/manglerde features:**

- Flere tests
- Åbne coast line bugger
- Gradle virker pt ikke med seneste implementation

##### **Features:**

- Masse bug fix
- Tilføjet flere iconer
- Ikon filtre
- Bygning filter
- Colorblind mode
- Optimeret søgefunktion
- Modul baseret layeredPaneView
- Læsning af lukkede coastlines
- Opdelt flere veje
- Forbedret quadtree
- Scale line

##### **Wish list for v0.10 - Honorable Sudachi:**

- Søgning af korteste vej “Dejkstra”
- Visning af vejnavn
- Fix af coastlines

##### **Feedback request:**

Om vores implementation af QuadTree er rigtig.

**Time usage in man hours:**

- 3 hours: group meeting (4 personer)
- 15 hours: Modul-baseret LayeredPaneView(1 person)
- 5 hours Ikon og bygnings filter (1 person)
- 3 hours ColorBlind mode (1 person)
- 5 hours Bugfixes (4 personer)
- 20 hours scale line (1 person)
- 4 hours: Optimization of Regex (addressparser) (1 person)
- 2 hours: JDoc & codestructuring (1 person)
- 16 hours: QuadTree optimization (1 person)

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 + 90.000 + 130625 + 60625 = 405000,-$

#### **11.4.5 V0.8 - Sweet clementine**

##### **Bugs/manglerde features:**

- Keyboard shortcuts virker ikke når søgerfeltet er der.
- Bug hvor bin filer ikke bliver loadet / saved

##### **Features:**

- Upgraderet vores address parser med ny måde at håndtere regex
- Manuel adressesøgning + autocomplete
- Zoom med zoom level + zoom mod mus som centrum
- Forbedret zoomslider
- Quadtree implementeret

##### **Wish list for v0.8 - Explosive grapefruit:**

- Fix coastlines
- Tilføj filtrering af ikoner, bygninger mm.
- Modul baseret view komponenter

##### **Feedback request: Time usage in man hours:**

- 10 hours: group meeting (4 personer)
- 17 hours: Søgefunktion(1 person)
- 4 hours: Optimization of Regex (addressparser)

##### **Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 + 90.000 + 130625 = 344375,-$

### **11.4.6 V0.7 - Blazing citrus**

#### **Bugs/manglerde features:**

- Keyboard shortcuts virker ikke når søgerfeltet er der.
- Bug hvor bin filer ikke bliver loadet / saved
- Zoom level bliver ikke reset når nyt bliver loaded

#### **Features:**

- Upgraderet vores address parser
- Søge og visning af gemte adresser (Partial - Dette er ikke blevet merged over endnu)
- Zoom med zoom level + zoom mod mus som centrum
- zoom level indikator.
- Fixed test af model i forhold de seneste refact
- Tegner elementer i forhold til Zoom-level
- Oprettet controllerTests
- Tilføjet flere cartografiske ikoner.
- tilføjet zoom-buttons
- HUGE refact af alle klasser!

#### **Wish list for v0.8 - Sweet clementine:**

- Søgefunktion færdiggjort
- QuadTree fully functional

#### **Feedback request:**

#### **Time usage in man hours:**

- 10 hours: group meeting (4 personer)
- 8 hours: Søgefunktion(1 person)
- 4 hours: Performance bugfix (1 person)
- 7 hours: Optimization of Regex (addressparser)
- 20 hours: Mega refact af alle klasser (5 person)

- 5 hours: bug fixing (1 person)
- 13 hours: Zoom function and zoom level (1 person)
- 5 hours: Draw optimization in relation with zoom level (1 person)
- 18 hours: Quad trees(1 person)
- 4 hours: Fix model test(1 person)
- 5 hours: Test af controller (1 person)

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 + 90.000 + 130625 = 344375,-$

#### **11.4.7 0.4 - Creative lime**

- Bugs/manglerne features:
- Nogle dele af kortet bliver ikke farvelagt
- Nogle bygninger bliver ikke malet
- Keyboard shortcuts virker ikke når søgefeltet er der.

#### **Features:**

- Optionbox som gør det mulig at indstille forskellige ting (eg. toggle AA)
- AA bliver nu automatisk slået til og fra ved zoom og panning
- Større refaktorisering af model, view og controller.
- Implementering af vores addressparser

#### **Wish list for v0.4**

:

- Hent og gem adresser fra OSM fil (Bruges til vejvisning)
- Scalerings linje
- Ikoner for restauranter etc
- Coastlines

#### **Feedback request:**

- Vi har faktoriseret vores program ved at opdele view, controller og model op i flere sub klasser. Spørgsmålet er så om selve opdelingen er okay og bør vi optimere vores klasser yderligere?

#### **Time usage in man hours:**

- 10 hours: group meeting (4 personer)
- 3 hours: Rapport skrivning (2 personer)
- 8 hours: Søgefelt(1 person)
- 8 hours: Optionbox (1 person)
- 2 hours: Optimization of AA (1 person)

- 16 hours: Zoom function (1 person)
- 16 hours: Code refraction and clean up (4 personer)

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 + 90.000 = 213750,-$

#### **11.4.8 V0.3 - Numb orange**

##### **Bugs/manglerde features:**

- Zoom - kun ind mod midten og ikke mod musen
- Nogle dele af kortet bliver ikke farvelagt
- Nogle bygninger bliver ikke malet
- Nogle bygninger med baggårde bliver overmalet (benytter ikke win-ding)
- Når man panner/zoomer uden AA sat til, bliver det sat til alligevel efter endt pan/zoom.
- Keyboard shortcuts virker ikke når søgefeltet er der.

##### **Features:**

- Pan/zoom med mousedrag og scroll.
- Små iconer på bestemte koordinater i kortet (restauranter og caféer)
- Filechooser til at loade et bestemt kort. (Programmet tager imod bin, zip og osm filer)
- Programmet kan gemme/loader bin filer
- Programmet har en søgebjælke

##### **Wish list for v0.4 - Creative Lime:**

- Pan og zoom mod mouse
- Hent og gem adresser fra OSM fil (Bruges til vejvisning)
- Scalerings linje

##### **Feedback request:**

Bør vi opdele vores klasser op i flere subklasser? Lige nu har vi kun få klasser, som gør rigtig meget.

**Time usage in man hours:**

- 10 hours: group meeting (4 personer)
- 8 hours: Søgefelt(1 person)
- 5 hours: Exception handling (1 person)
- 20 hours: Cartographics (1 person)
- 5 hours: Optimization (1 person)
- 16 hours: Zoom function (1 person)
- 3 hours: Code refraction and clean up (2 personer)

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 + 64.375 = 123750,-$

#### **11.4.9 V0.2 - Annoying lemon**

- Vi har valgt, at benytte os af Swing i stedet for JavaFX, da vi havde for mange udfordringer med implementeringen af pan/zoom. Vi har derfor brugt en del mandetimer på, at skifte vores filer ud og få det hele sat op på ny.

##### **Bugs/manglerne features:**

- Zoom - kun ind mod midten og ikke mod musen
- Nogle dele af kortet bliver ikke farvelagt
- Nogle bygninger bliver ikke malet
- Nogle bygninger med baggårde bliver overmalet (benytter ikke winding)

##### **Features:**

- Pan/zoom med mousedrag og scroll.
- Små iconer på bestemte koordinater i kortet (restauranter og caféer)
- Filechoosers til at loade et bestemt kort. (Programmet tager imod bin, zip og osm filer)
- Programmet kan gemme indlæst kort til bin fil

##### **Wish list for v0.3 - Numb orange:**

- Pan og zoom mod mouse
- Søgebar
- Flere elementer malet på kortet
- Vej typer opdelt efter farver og lister
- Iconer af restauranter, parkerings pladser mm. tegnes på kort
- Scalerings linje

##### **Feedback request:**

- Hvorvidt vores kode kan refaktoreres bedre - om der er overflødige klasser.
- Strukturen og om vi overholder MVC

**Time usage in man hours:**

- 4 hours: JavaFX og swing konvertering
- 6 hours: group meeting (4 personer)
- 8 hours: filechooser (1 person)
- 8 hours: test af model (1 person)
- 4 hours: bin load and save funktion (1 person)

**Cumulative cost so far: (1h = DKK 625)**

Total :  $29.375 + 30.000 = 59.375,-$

#### **11.4.10 V0.1 - Precitrus era**

##### **Bugs/manglerde features:**

- Kan ikke tegne/farvelægge dele af kortet
- Kan ikke zoome eller panne kortet
- Bygninger får den samme farve som park/grønne områder første gang der loades og farver slås til. Farven kan dog ændres så snart brugeren vælger at skifte farve vha. "building"color picker i GUIen. Hvid er standard "fill"farve for bygninger ved første interaktion med color picker (Uanset hvilken farve han vælger).
- Stregetykke bliver tykkere hver gang vi loader et kort igen. (strokeWidth) Ser ud til at stregerne bliver tegnet flere gange oven på hinanden?
- Programmet understøtter ikke load eller save af binary filer

##### **Features:**

- Load OSM og ZIP filer gennem GUI
- Ændre farver på bygninger og hovedveje vha. color picker
- Slå farver til og fra

##### **Wish list for v0.2 - Annoying lemon:**

- Fixing pan and zoom functions
- Add more drawing tags/methods
- Save and load Binary file-format
- Fixed frame title. Current “Linedrawer X Pro”.

##### **Feedback request:**

- Nuværende kodestruktur og eventuelle forbedringer
- Forslag på hvordan vi opretter zoom og pan funktion i vores projekt (JavaFX)
- Andre gode råd i hvad vi bør være opmærksom når vi skal arbejde med javaFX

**Time usage in man hours:**

- 20 hours: lecture and group formation (5 people x 4 hours)
- 10 hours: fleshing out group constitution
- 6 hours: group meeting
- 5 hours: clean and documentate old code
- 3 hours: convert and merge old project to new repo
- 3 hours: Writing test

**Cumulative cost so far: (1h = DKK 625)**

Total : 29.375 .-

## 11.5 Logbog

## **Førsteårsprojekt logbog - Gruppe C**

**Møde nr. 15 til 20 d.14 / 05 - 2015 til d.20 / 05 2015**

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Stig og Emma

**Udeblevne medlemmer:**

**Lovlig fravær:**

### **Dagsorden**

- Rapport skrivning alle dage

### **Referat**

- Alle dage har stået på rapportskrivning
- Tirsdag d. 19 maj har gruppen samlet og gennemgået alt i rapporten. Vi har blandt andet fået feedback af TA
- Vi har afleveret rapporten Tirsdag nat/Onsdag morgen.

### **Hjemmeopgave**

- Forberedelse til eksamen

### **Punkter til næste møde**

- Forberedelse til eksamen
  - Gruppen snakker om selve gruppepræsentation
  - Gruppen gennemgår læringskrav, rapport og kode.
  - Gruppen holder testoplæg

## **Førsteårsprojekt logbog - Gruppe C**

**Møde nr.14 d15. / 05 - 2015**

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Emma

**Udeblevne medlemmer:**

**Lovlig fravær:** Steffen

### **Dagsorden**

- Snakke om rapporten

### **Referat**

- Vi har snakket om hvordan rapporten skal struktureres
  - Vi har snakket om at det er det analyserende der tæller mest
  - Vi har snakket om blackbox test og whitebox test
  - Vi har snakket om at komme ind på hvordan vi har tolket kravene
  - Vi har snakket om hvilke bugs vi opdagede under test
  - Vi har snakket om JavaFX og Swing

### **Hjemmeopgave**

- Gør til rapport skrivningi

### **Punkter til næste møde**

## **Førsteårsprojekt logbog - Gruppe C**

**Møde nr. d.13 / 05 - 2015**

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Stig og Emma

**Udeblevne medlemmer:**

**Lovlig fravær:** Dennis, Steffen, Andreas

### **Dagsorden**

- Kort overblik af rapport og uddelegering af opgaver

### **Referat**

- Stig og Emma har kigget på rapporten.
  - Rapport opdelt og mere eller mindre uddelegeret
- Der er blevet skrevet videre på den

### **Hjemmeopgave**

- Magnus kan mødes mellem 10-12 på mandag. Vi skal sende ham vores udkast til rapport senest fredag 15/05 2015

### **Punkter til næste møde**

- Gennemgang af rapport og evt kig på latex

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 19 d. 12 / 05 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Stig Killendahl og Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:** Emma (Lejlighed)

### Dagsorden

- Fix bugs og testskrivning
- Stig har fødselsdag!

### Referat

- Dennis og Stig har lavet lavet white-box testing af quadtree og nearestneighbour
  - Derunder er der blevet fixed nogle bugs.
- Steffen og Andreas har oprettet test til alle andre klasser
- Stig har fixed Zoom i forhold til pointers og den beregnede rute (Således at man kan se elementerne)
- Andreas har fixed og oprettet en JarFil
- Dennis har skrevet Jdocs og gennemgået alt kode inden aflevering
- Klokken er 03,48 og gruppen har lige afleveret koden

### Hjemmeopgave

- Aflevering af koden

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 19 d. 11 / 05 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Reflektion over projektet og feature freeze
- Hvad skal der laves inden code freeze

### Referat

- Alle krav er opfyldt. Alle arbejder nu på at fixe bugs og optimering af koden.
- Der er skrevet JDocs til alle klasser og strukturen er rettet.
- Emma og Steffen har skrevet rapport

### Hjemmeopgave

- Bugfix og optimering

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 18 d. 06 / 05 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis og Stig

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Gennemgang af hvad der blevet nået og status for projektet
- Planen for ugen
- Uddelegering af opgaver
  - Kodning & Rapport skrivning

### Referat

- Stig arbejder på Dijsktra
- Dennis refacted klasse navne og implementeret string polls
- Andreas har skrevet rapport (latex) og fixed test og smidt codecoverage i git
- Steffen har skrevet rapport og arbejdet på rutevejlednings gui

### Hjemmeopgave

- Stig og Dennis arbejder videre på rutevejledning og korteste rute

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 17 d.04 /05 -2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:** Steffen Immerkær (Sverige tam tam)

### Dagsorden

- Uformel møde

### Referat

- Vi havde arbejdet videre på projektet
  - Emma har skrevet rapport
  - Dennis har fixed FileHandlerTests, tilføjet ekstra ikoner i navigationspanelet og skrevet rapport
  - Andreas har fixed coastLines (Vi benytter os af recursive kald)
  - Stig har arbejdet på Dijkstra
    - Stig fik lidt råd af Kevin om hvordan vi kunne implementere Dijksktra
    - Dennis og Stig har taget en døgner og har fået implementeret en graf over alle veje og en prototype af noget korteste rute vha dijkstra. Heraf tegner vi ruten som nu bliver fundet på kortet
    - Der er også implementeret en funktion som kan tjekke nearest neighbour af et given punkt (Nærmeste way/Vertex af et given punkt)

### Hjemmeopgave

- Andreas arbejder videre på Bin load og save
- Steffen arbejder videre på skalering af ikoner.
- Stig og Dennis kigger videre på Dijkstra

### Punkter til næste møde

- Gruppen arbejder videre på de uddelegerede opgaver

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 16 d. 26 / 04 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Diskutere om aktivitets niveau / kommunikation
- Hvad vi har nået i løbet af sidste uge.
- Hvad vi vil nå i løbet af ugen
- Uddeleger arbejde
  - Uddelegering af rapport afsnit (Afsnit som skal være skrevet inden næste søndag.)

### Referat

- Vi har snakket om at oprette vores quadtree mens vi loader OSM filen, i stedet for at gemme dem i et hashmap
- Andreas havde kigget på optimering af coastline/En klasse coastlines og optimering af memory og save load af bin
- Dennis kigger på gui og GPS
- Stig Optimering af memory
- Steffen kigger på Adresse visning near mouse
- 

### Hjemmeopgave

- Gruppen arbejder videre på uddelegerede opgaver

### Punkter til næste møde

- Status over projekt og diskussion over implementation af Dijkstra's algoritme

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 15 d.22 /04 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Stig Killendahl

**Udeblevne medlemmer:**

**Lovlig fravær:** Emma Arfelt (lejligheds tam tam)

### Dagsorden

- Status af hvad vi har arbejdet på
- Merge færdig
- Bug fix og kodning

### Referat

- Andreas og Dennis kiggede på LayeredPaneView refact
  - Fixed AdresseSøgning bug
  - Steffen kiggede på scalebaren
- Stig kiggede på at parse Danmark sammen med andreas
- Dennis kigger på regexbug
- Emma skriver i rapport

### Hjemmeopgave

- Det står foroven

### Punkter til næste møde

- Diskutere om aktivitets niveau / kommunikation
- Hvad vi har fået i løbet af sidste uge.
- Hvad vi vil nå i løbet af ugen
- Uddelegere arbejde

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 14 d.20 /04 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:**

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Møde med TA
- Gennemgå hvad vi har nået i løbet af ugen
- Rapport skrivning og kodning

### Referat

- Vi havde et møde med TA
  - Vi snakkede om hvordan vi loader en fil ind i JAR således at det virker
  - QuadTree status (Det så godt nok ud)
- Dennis arbejder på Refact af layeredPane view
- Andreas arbejder på coastline og relation
- Steffen laver flowchart til osmhandler og uml diagram
- Stig og Emma kiggede på icon load og de bugs som fulgte med.

### Hjemmeopgave

- Emma kigger på rapport
- Stig tegner 5 typer veje og optimerer på adresse parser
- Andreas kigger på coastline
- Stefffen, kigger scalebar og rapport

### Punkter til næste møde

- Status over hvad der er implementeret

## Førsteårsprojekt logbog - Gruppe C

Møde nr.13 d.13 /04 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Status for projekt
- Hvad vi skal nå og blive færdige i løbet af ugen

### Referat

- Vi har snakket om hvordan vi tegner coast line - Andreas
- Vi har snakket om hvordan vi gemmer vores nodes - Dennis stig Andreas
  - Vi gemmer ikke tags længere permanent, men derimod gemme dem midlertidigt i vores osmparser, mens den parser osm filen. Hver gang man støder på et endtag, så uddelegere vi dataen i forhold til hvilke tags der blev gemt midlertidigt.
    - Gemme cartografik i en liste for sig selv
    - Lade adresse objekten gemme en reference til en node objekt.
- Vi har snakket om quadtrees og hvordan vi vil dele dem op i flere - Stig
- Emma kigger på hvordan vi får tegnet ikoner
  - Samt refact af kode duplikation
- Vi skal ordne OSMhandleren - Stig Dennis Andreas
  - Adresse objekter bliver oprettet direkte i OSMhandleren
  - Ways skal deles op pr. type
    - MOTORWAY
    - PRIMARY
    - SECONDARY
    - TERTIARY
    - RESIDENTIAL/ROAD/SERVICE/TRACK ...
- Binær save og load skal fixes således at det virker igen
  - Implementering af serializable - IF WE HAVE TIME
- Vi skal kigge lidt på option-box i forhold til hvad vi vil tegne
- Tegne vej en gang med sort fed streg og en gang med hvid tynd streg.
- Vi blev enige om at vi gerne vil afsætte en time på rapporten i dag

### Hjemmeopgave

- Dennis, Andreas og Stig kigger på (Mødet)
  - Behandling af nodes, Oprættelse af addresser i OSMhandler og indeling af ways
- Dennis kigger videre på søge funktion
- Stig kigger på Quadtrees
- Andreas Kigger på relation og coastline
- Emma kigger på refact af cartografiske elementer
- Steffen kigger på hide/show af ikoner, optimering af zoom indicator og eventuel skaleringsbar.

### Punkter til næste møde

- Arbejd videre :D

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 12 d.08 /04 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Stig Killendahl

**Udeblevne medlemmer:**

**Lovlig fravær:** Andreas Hassing - (Reeksamen) (Han var til møde med Troels)

### Dagsorden

- Møde med Troels

### Referat

- Vi havde et møde med Troels og kom frem til følgende
  - Ændre OSMHandler
  - - Double til float.
  - - Hashmap og lister af nodes er ikke nødvendigt.
  - - Vi bør ikke gemme hashmap i hvert way (Memory overflow)
- Datastruktur
- - Way burde gemmes i hver deres datastruktur, frem for én.
  - eg adskil residential og highways fra hinanden
- Canvas
- - Canvas buffered strategy - Giver samme effekt som buffered image
  - Tjek om dette er muligt med JKomponent
- Adresser
- - Start med sorteret arraylist

### Hjemmeopgave

- Videre arbejde

### Punkter til næste møde

- Status for projekt
- Hvad vi skal nå og blive færdige i løbet af ugen

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 11 d. 07 / 04 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:** Andreas Hassing - (Reeksamen)

### Dagsorden

- Opsamling af hvad vi har lavet og hvad vi skal nå
- Status på projektet rapport

### Referat

- Vi havde opsamling og gennemgik kravene for vores projekt og hvad der skal nå.
- Steffen og Dennis kiggede på en bug som gjorde at vi andre fik en exception (IllegalArgument) når vi ville køre programmet. Dette skyldes pga. at stigen til de forskellige iconer var defineret forkert.
- Emma kigge på address parser og controllertest
- Stig kiggede på QuadTrees
- Steffen kiggede på zoomlevels

### Hjemmeopgave

- Der forberedes til Troels
- Arbejde videre

### Punkter til næste møde

- Møde med Troels
- Skrive rapport

## Førsteårsprojekt logbog - Gruppe C

Møde nr.10 d.30 /3- 2015

**Referent:** Emma Arfelt

**Ordstyrer:** Magnus

**Udeblevne medlemmer:**

**Lovlig fravær:** Steffen, (Dennis er på skype)

### Dagsorden

- Møde med magnus

### Referat

- Vi snakkede om hvordan vi kan skrive vores model således at det var mulig for os at opdele vores input af sax parseren
- Emma har skrevet ned om hvad vi kunne gøre
- Magnus kiggede på vores kode og kom med nogle forslag på hvad vi kan gøre
  - Scrap map fra viewet
  - Behold vores datastruktur og byg på det
  - Benyt observerpatteren (View ændres, controller modtager og kalder på model)
  - Kig på comparable
- Vi snakkede om hvorfor vores program laggede
  - Men der skete ikke rigtig noget?
- Snakkede om relationer og coastlines
  - Coast line er ikke nødvendigvis lukket og derfor vil det skabe problemer om at lukke den
  - Derfor skal vi lave nogle beregninger med skæring af boundingboks for at lukke den
    - Vi bør vente med dette medmindre der er nogen som vil nørde det.
    - Datastrukturen bør prioriteres først (Quad trees osv)

### Hjemmeopgave

- Dennis arbejder på addresseparser og forslagliste
  - Dennis skriver om brug af layeredpane og refact af view
- Stig arbejder på zoom
  - Stig og Steffen kigger på tegning af kortet
  - Stig og Steffen skriver lidt om refact af map view
- Steffen arbejder på paint og load af ikoner (kig på zoom level bar)
- Emma arbejder controller test
  - Emma skriver om control test
- Andreas tester model og fix af coast line
  - Skriver om den nye model struktur
- Der arbejdes videre på vores model og dets datastruktur

### Punkter til næste møde

- Opsamling af hvad vi har lavet og hvad vi skal nå
- Status på projektet rapport
- Møde med Troels

## Førsteårsprojekt logbog - Gruppe C

Møde nr.9 d.25/3 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Andreas Hassing

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Fixed model bug, så nu virker vores
- Andreas præsentere den nye refaktoriserede model
- 

### Referat

- Vi snakkede om hvad vi skal lave i løbet af ugen
- Vi arbejdede videre på projektet

### Hjemmeopgave

- Dennis arbejder på addresseparser og forslagliste
  - Dennis skriver om brug af layeredpane og refactor af view
- Stig arbejder på zoom
  - Stig og Steffen kigger på tegning af kortet
  - Stig og Steffen skriver lidt om refactor af map view
- Steffen arbejder på paint og load af ikoner (kig på zoom level bar)
- Emma arbejder controller test
  - Emma skriver om control test
- Andreas tester model og fix af coast line
  - Skriver om den nye model struktur

### Punkter til næste møde

- Møde med magnus

## Førsteårsprojekt logbog - Gruppe C

Møde nr.8 d. 23 /03 - 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Fremlæggelse af projekt
- Status af projektet
  - Refact af model
- Rapport indhold -Memory usage?
- Implementering af quad trees

### Referat

- Vi har fremlagt vores status af projektet og diskuteret lidt hvad vi ville nå

### Hjemmeopgave

- Stig kigger på scale line
- Dennis kigger adresse liste
- Emma kigger på test af controller

### Punkter til næste møde

- Andreas gennemgår model
- Rapport skrivning af rapport
- Fix af test i forhold til ny model
- Fix af lag
- Rapport view droppe
- Quad trees

## Førsteårsprojekt logbog - Gruppe C

Møde nr.7 d.18 /03-2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Stig KillenDahl

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Gennemgå refact
- Hvad der skal blive færdig
- Tildeling af opgaver

### Referat

- Andreas refactor Model - Steffen hjælper også Andreas
- Emma kigger på tests af controllers
- Stig og Dennis kigger på center map
  - Dennis kigger på search fokus
- Steffen kigger på iconer og tegning af zoom levels
- Dennis fik fixed anti-aliasing bug

### Hjemmeopgave

- Emma kigger på udvidning elementer
- Steffen og andreas arbejder videre på refact af model
- Dennis kigger på optionbox
- Stig kigger zoom

### Punkter til næste møde

- Fremlæggelse af projekt
- Status af projektet
  - Refact af model
- Rapport indhold -Memory usage?
- Implementering af quad trees

## Førsteårsprojekt logbog - Gruppe C

### Møde nr.6 d.16 /03- 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Dennis Thinh Tan Nguyen

**Udeblevne medlemmer:**

**Lovlig fravær:** Emma (Analog)

### Dagsorden

-

### Referat

- Gruppen er blevet enige om at de skal refaktorere klasse strukturen
  - Andreas arbejder med Model
  - Steffen kigger på Controller
  - Dennis og Stig kigger på View
- Gruppen er blevet enige om at ændre mødetider til Mandag og Onsdag i stedet for Mandag og Tirsdag
  - Om mandagen mødes vi kl 10 (Emma ikke analog) og 12(Emma Analog)
- Gruppekontrakt ændret
- Der skrives skrives i rapport - Emma

### Hjemmeopgave

- Emma skriver i rapport
- Andreas arbejder med Model
- Steffen kigger på Controller
- Dennis og Stig kigger på View

### Punkter til næste møde

- Gennemgå refact
- Hvad der skal blive færdig
- Tildeling af opgaver

# Førsteårsprojekt logbog - Gruppe C

Møde nr. 5 d. 10/03- 2015

**Referent:** Dennis Thinh Tan Nguyen

**Ordstyrer:** Andreas

**Udeblevne medlemmer:**

**Lovlig fravær:**

## Dagsorden

- Præsentation af GUI
- Diskussion over hvad vi skal nå til næste mandag
- Kig videre på programmet
- Kig på rapport

## Referat

- Stig og Dennis præsnterede gui mock-up.
  - Vi blev enige om at guien var fin, men at vi gerne vil have søg og rutevejledning inkorporeret i en kasse under søgerfeltet. Dette er for at undgå pop-up vinduer, som den nuværende løsning bruger
  - Stig reviderede mock-ups
- Gruppen blev enige om at vi vil nå følgende punkter i løbet af ugen:
  - Zoom level skal blive færdig
  - Tegning af cartografiske elementer på kortet ved hjælp af bufferet image
  - Tilføje adresse felt
  - Optimering af zoom/pan hvor vi slår AA fra
  - Rapport strukturen skrives
  - Kast af exceptions ved loading af OSM/Bin filer som ikke indholder kort data.
- Gruppen diskuterede lidt om hvad rapporten skal indeholde og kom frem til følgende som var vigtige :
  - Testing kode
  - Optimering af kort visning/kode
  - Fremgangsmåde af hvordan man er kommet frem til løsninger af problemer
  - Flowcharts
  - Mockups
  - Uml diagram
  - Struktur af kode - Passer med swing
  - Klasse struktur
  - Brug af algoritmer

## Hjemmeopgave

- Dennis arbejder på exception handling og adresse felt
- Stig arbejder på Zoom, Boundingbox brug og Haversine-formel til beregning af zoom mm
- Steffen arbejder på Cartographics
- Andreas kiggede relation af OSM

## Punkter til næste møde

- Ændring af mødetider til optimal brug af ugen
  - Hvor meget tid bliver der reelt brugt på projektet om ugen?
- Diskussion af præsentation d. 23/03
- Gennemgang af rapport struktur
- Gennemgang af hvad vi har nået og hvad vi skal nå
- Igangsætning af rapport skrivning (heraf 2 personer)
- Refaktorisering af klasse struktur

## Førsteårsprojekt logbog - Gruppe C

Møde nr.4 d.9 /3-2015

**Referent:** Dennis

**Ordstyrer:** Andreas

**Udeblevne medlemmer:**

**Lovlig fravær:** Emma

**Dagsorden**

- Gennemgang af UML af version 0.2
- Gennemgang af hvad der er blevet lavet og tilføjet
- Programmets nuværende status - Hvad det kan og hvad det ikke kan.
  - evt bugs
- Hvilke funktioner vi skal have fokus på og hvad vi skal nå i løbet af ugen
- Diskussion og eventuel ideer til hvordan vores gui skal se ud
  - Opret af mock ups - 2 personer

**Referat**

- Gennemgang UMI til gruppen
- Clean code workshop
- Mindre bug fixes af programmet og merging af det sidste
- Gruppen har diskuteret over hvilke funktioner skal prioriteres
  - Vi er kommet frem til at vi skal refactor tegnemetoderne, fixe coast line og optimering af tegne funktionen
- Dennis og Stig har arbejdet på Mock-ups af GUI
- Andreas og Steffen arbejder på visning af Cartographics element

**Hjemmeopgave**

- Andreas arbejder med refactor af tegne funktionen til i morgen
  - Kigger også på skalalinje
- Undersøg hvordan vi kan fixe coast line og eventuel optimering af kort
- Steffen kigger videre på cartographics elements
- Stig læser op på Coast line og zoom levels
- Emma kigger på bug fixing af fill
  - Emma kigger på rapport
- Dennis kigger på adresse felt og søgning

**Punkter til næste møde**

- Præsentation af GUI
- Diskussion over hvad vi skal nå til næste mandag
- Kig videre på programmet
- Kig på rapport

## Førsteårsprojekt logbog - Gruppe C

Møde nr.4 d.3 /3-2015

**Referent:** Dennis

**Ordstyrer:** Emma

**Udeblevne medlemmer:**

**Lovlig fravær:** Andreas (ferie)

### Dagsorden

- UML-diagrammer af nuværende klasser samt nye
- Diskussion af kritiske funktioner samt uddelegering af opgave
- Fix af få bugs

### Referat

- Stig og emma fixer git og små bugs af projektet
- Dennis og steffen laver UML diagram
- Dennis arbejder på filechooser funktion
- Stig arbejder på pan og zoom optimering
- Steffen arbejder på kort elementer (parkerings skildte osv)
- Emma tilføjer flere draw elementer

### Hjemmeopgave

- Dennis arbejder på filechooser funktion
- Stig arbejder på pan og zoom optimering
- Steffen arbejder på kort elementer (parkerings skildte osv)
- Emma tilføjer flere draw elementer

### Punkter til næste møde

- Gennemgang af UML af version 0.2
- Gennemgang af hvad der er blevet lavet og tilføjet
- Programmets nuværende status - Hvad det kan og hvad det ikke kan.
  - evt bugs
- Hvilke funktioner vi skal have fokus på og hvad vi skal nå i løbet af ugen
- Diskussion og eventuel ideer til hvordan vores gui skal se ud
  - Opret af mock ups - 2 personer

# Førsteårsprojekt logbog - Gruppe C

Møde nr. 3 d.02/03-2015

**Referent:** Emma

**Ordstyrer:** Andreas

**Udeblevne medlemmer:**

**Lovlig fravær:** Andreas (ferie)

## Dagsorden

- Kode skelet bygges og gennemgås
- UML-skabeloner

## Referat

- Vi har besluttet at benytte os af Swing i stedet for JavaFX til brugergrænsefladen, da vi efter to timers arbejde måtte opgive at implementere zoom og pan.
- Efterfølgende tid blev brugt på at merge Steffen og Stigs kode sammen, således at vi fik features fra begges kode implementeret: Zoom og pan med musen samt indsætning af ikoner på bestemte koordinator.
- Koden blev struktureret pænere

## Hjemmeopgave

- Overvejelser omkring UML diagrammer

## Punkter til næste møde

- UML-diagrammer af nuværende klasser samt nye
- Diskussion af kritiske funktioner samt uddelegering af opgaver

## Førsteårsprojekt logbog - Gruppe C

Møde nr. 2 d. 24/02-15

**Referent:** Steffen Immerkær OG Dennis

**Ordstyrer:** Emma Arfelt

**Udeblevne medlemmer:**

**Lovlig fravær:**

### Dagsorden

- Praktiske detaljer med hensyn til gruppearbejdet
  - Kort gennemgang af ændringer i gruppekontrakt
  - Hvordan vi får arbejdet til at passe med hensyn til de andre kurser (andre grupper som vi også er i)
  - Hvordan vi deler arbejdet op samt planlægning uge for uge
- Branching vha. git - Andreas holder en kort intro til dette
- Gennemgang af projektet
  - evt. kravspecifikation
  - Trello
- Tidsplan for features og projektforløbet
- Opsætning af kodestruktur og klasser (UML diagram)

### Referat

- Andreas har holdt et kort oplæg om git branching, trello og gradle
- Gruppen har gennemgået opgave beskrivelsen
- Gruppen har oprettet en tidsplan vha. google sheets
- Emma har plottet grundlæggende opgaver i trello
- Gruppen er blevet enige om at vi tager udgangspunkt i Dennis's projekt

### Hjemmeopgave

- Dennis har fået som opgave i at smide hans projekt op i git
- Læs på JavaFX og UML
  - Eventuel kom med nogle forslag på klasser/metoder
- Gennemse og forstå gammel kode

### Punkter til næste møde

- Der laves en UML-graf
- Kodeskelet gennemgåes og bygges
- Diskutere hvilke features er kritiske og skal ordnes først

## SKABELON

### Førsteårsprojekt logbog - Gruppe C

Møde nr. 1 d. 23/02-15

**Referent:** Emma Arfelt / Dennis Thinh Tan Nguyen

**Ordstyrer:** Emma

**Udeblevne medlemmer:**

**Lovlig fravær:**

#### Dagsorden

- Opstart af gruppe
- Valg af kontaktperson
- Diskuter gruppekontrakt
- Praktiske ting oprettes (eg. Facebook gruppe)
- Skabe god socialramme/dynamik

#### Referat

- Gruppen er dannet på baggrund af fælles interesse, ambitionsniveau og faglig niveau.
  - Gruppens medlemmer er:
    - Steffen Immerkær
    - Emma Arfelt
    - Stig Killendahl
    - Dennis Thinh Tan Nguyen
    - Andreas Bjørn Hassing Nielsen
  - Emma er blevet valgt som kontaktperson
  - Udkast af gruppekontrakt skrevet
  - Der blev oprettet en facebook gruppe og google drev.
  - Hvert medlem skal uploadet et billede af dem selv og deres underskrift i gruppekontrakten.
  - Gruppen har prøvet på at skabe en god social fundament vha et spil; Bezzewizzer
    - Winner : Stig og Steffen :)

#### Punkter til næste møde

- Praktiske detaljer med hensyn til gruppearbejdet
  - Kort gennemgang af ændringer i gruppekontrakt
  - Hvordan vi får arbejdet til at passe med hensyn til de andre kurser (andre grupper som vi også er i)
  - Hvordan vi deler arbejdet op samt planlægning uge for uge
- Branching vha. git - Andreas holder en kort intro til dette
- Gennemgang af projektet
  - evt. kravspecifikation
- Tidsplan for features og projektforløbet
  - evt. intro til Trello
- Opsætning af kodestruktur og klasser (UML diagram)

## **11.6 Gruppekontrakt**

# Gruppekontrakt

**Gruppenavn:**

Gruppe C

**Vejledere:**

Troels Bjerre Sørensen

**Kursus:**

Førsteårsprojekt: Danmarkskort: Visualisering, Navigation, Søgning og Ruteplanlægning  
(Spring 2015) - (BFST-Spring 2015)

**Dato:**

Afleveret: 24. februar 2015

## **Møde**

- 1.1. Vi har fast mødetid mandag kl.12.00 (med mindre Emma har fri, hvor vi møder tidligere - jf. Facebook gruppen) og onsdag kl. 10.00
  - a. Emma står i Analog hver anden mandag morgen (09-11.30)
- 1.2. Møderne kan ændre sig efter behov og kan arrangeres ad hoc.
- 1.3 Hvert møde starter med gennemgang af logbog/dagsorden efter behov
  - a. Der udvælges en ordstyrer og referent. (rollerne fordeles fra møde til møde)
- 1.4 Hvert møde skal slutte med en opsamling samt uddelegering af evt. arbejdsopgave.
- 1.5 Afbud skal helst ske med 24 timers varsel
- 1.6 Gruppemøder foregår på ITU med mindre andet er aftalt.
  - a. Ved møder på ITU: Møder foretrækkes at finde sted i bokse eller lokaler.
- 1.7 Kommer man for sent afleveres der 1 kroner pr. minut (efter 10 min. efter aftalt mødetidspunkt) Loft: 20 kr.
  - a. Undtagelse: Er der god begrundelse for forsinkelsen og at der gives besked forinden, slipper man for "bøden".

## **Fravær**

I tilfælde af, at en person er lovligt fraværende, må resten af gruppen så vidt muligt forsøge, at inddrage personen i beslutninger og arbejde der er lavet i dennes fravær.

## **Mødetidspunkter.**

- Mandag 10:00 - xx.xx
- Tirsdag 10:00 - xx.xx

## **Pauser**

Pauser bør begrænses til 5-10 minutter, da gruppemøderne generelt holdes efter frokosttid. Gruppen bør holde én generel pause i timen. Hvis et gruppemedlem bliver dårlig eller har behov for toiletbesøg, er vedkommende (selvfølgelig) undtaget.

## **Roller**

- 2.1 Emma Arfelt, ekoc@itu.dk, er projektleder. Dennes rolle er, at fastholde fokus, sikre overholdelse af deadlines og koordinering af opgaver.

## **Arbejdsproces/Værktøjer**

3.1 Det er vigtigt, at den kode som implementeres forstås af alle gruppemedlemmer.

3.2 Vi benytter os af branches under udvikling af vores projekt med Git. Hertil "Pull requests", således vi gennemgår hinandens kode før der merges ind.

3.3 Trello bruges overblik over tasks samt to-do's.

3.4 I koden prioriterer vi at lave en grundig dokumentation.

a. Heraf klasse og metode dokumentation(JDoc) og linjekommentering

3.5 JavaFX bliver brugt til at kode Interfacet af programmet

## **Arbejdsfordeling**

Der er ikke nogen i gruppen, der skal overbebyrdes med arbejde. Arbejdet skal uddeles retfærdigt mellem gruppemedlemmerne.

## **Mål og ambition**

4.1 Det forventes at alle yder deres bedste og at alle går den ekstra mil for et succesfuldt grupperarbejde.

4.2 Gruppen har fastlagt et højt ambitionsniveau. Dette indebærer, at vi alle stræber efter et 12-tal og arbejder målrettet og fokuseret på at opfylde dette, men vi ønsker alle som minimum en karakter over gennemsnittet.

4.3 Hvis ét eller flere medlemmer af grupper misligholder ovenstående punkter er det op til de øvrige gruppemedlemmer at tage konflikter op til møder.

## **Uoverensstemmelser**

- 4.1 Diskussioner og uoverensstemmelser afgøres ved fælles enighed.
- 4.2 Opnås fælles enighed ikke kan det i sidste instans afgøres ved afstemning. Flertal bestemmer.

## **Kommunikation**

- 5.1 Tekster og referater skrives i Google Docs på dansk
- 5.2 Kode og dokumentation skrives på engelsk.
- 5.3 Uddelegering af opgaver sker vha. trello.

### **Kommunikationskanal**

Vores primære kommunikationsmiddel er Facebook. Alle medlemmer er på facebook og bruger det dagligt. Alle vigtige beskeder (bl.a. vedrørende fravær) skal kommunikeres i en separat besked på vores fælles Facebook-gruppe (alternativt per SMS) og ikke som svar på andre tråde.

Facebook-gruppen faciliterer planlægning og gruppe-events, hvilket gør kommunikationen overskuelig og nem. Intern kommunikation skal være klar og entydig, således at antallet af misforståelser minimeres. Hvert gruppemedlem har ansvar for at holde sig ajour med informationer i gruppens Facebook-gruppe.

### **Gensidig respekt**

Selvom man er uenig, er stresset osv. skal man respektere hinanden og imødekomme alle input. Alle gruppemedlemmers taleret skal respekteres, og vi afbryder ikke hinanden.

### **Åben konstruktiv kritik**

Man kritiserer ikke for at kritisere, men kommer med forslag til hvordan tingene kan gøres bedre. Ligeledes skal man også kunne klare konstruktiv kritik uden at tage det personligt.

### **Ærlighed/åbenhed**

Hvis man er utilfreds med noget i gruppearbejdet, skal man bringe det op til diskussion i gruppen og ikke fortie det.

### **Frie spørgsmål**

Der findes ingen dumme spørgsmål. Man skal ikke være bleg for at spørge sine gruppemedlemmer, hvis der er noget, man ikke forstår.

## **Diverse**

6.1 Snack kan tages med ad hoc (og skal deles med gruppen)

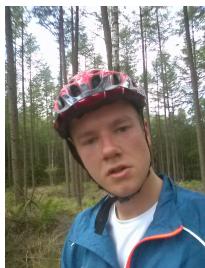
6.2 Der er plads til hygge ved siden af gruppearbejdet

## Medlemmer & underskrifter:



A handwritten signature in blue ink that appears to read "Dennis Thinh Tan Nguyen".

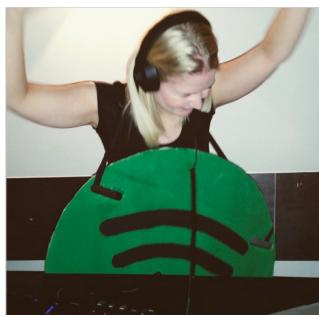
Dennis Thinh Tan Nguyen - (dttn)



A handwritten signature in blue ink that appears to read "Andreas".

A handwritten signature in blue ink that appears to read "Nielsen".

Andreas Bjørn Hassing Nielsen - (abhn)



A handwritten signature in blue ink on lined paper that appears to read "Emma Arfelt Kock".

Emma Arfelt Kock - (ekoc)



*Steffen Immerkær*

Steffen Immerkær - (shau)



*Stig K*

Stig Killendahl - (skil)