FØRSTEÅRSPROJEKT DEL 1

Danmarkskort

Udarbejdet i forbindelse med Førsteårsprojektet på bacheloruddannelsen
Softwareudvikling på IT-Universitetet i København forår 2014.

Gruppe A: Peter Clausen, Kristian Nielsen, Adam Engsig og Johan Arleth
Email: pvcl@itu.dk, kvin@itu.dk, adae@itu.dk, johaa@itu.dk

Indholdsfortegnelse

Forord

Denne rapport omhandler java programmet TimTim, som vi i en gruppe af fire bachelorstuderende, over et fire ugers langt forløb, har udviklet i forbindelse med første del af Førsteårs-projekt, forår 2014 på IT-Universitetet i København. Rapporten henvender sig mod læsere, der har erfaring med det objektorienteret programmeringssprog Java SE og kendskab til Swing i denne, da koden og rapporten ligger med fundament på disse.

TimTim er et program der giver en grafisk præsentation af Danmarks vejnet, stier og søruter. Programmet stiller en række intuitive funktionaliteter til rådighed for at hjælpe brugeren rundt i programmet. Det minimalistiske design gør TimTim meget håndgribeligt at bruge, selv for brugere uden meget computer erfaring.

I rapporten vil der gås i dybden med hvilke tanker vi har gjort omkring vores valg af implementation, samt en teknisk beskrivelse af den faktiske implementation, med de vigtigste metoder og klasser nævnt. Til sidst vil der være en diskussion af vores løsning samt eventuelle mangler, og om man i bagklogskabens lys, kunne have valgt en bedre løsning på dele af problemet.

Problemstilling og Baggrund

Problemstillingen er at lave et interaktivt Danmarkskort ud fra et udleveret datasæt lavet af Krak. Kraks datasæt er 2 tekstfiler bestående af al den information der er værd at vide om veje i Danmark. Programmet skal kunne bruges af en almindelig bruger, der ved hvordan man betjener en computermus. Kortet *skal* opfylde de 5 krav angivet herunder, men må gerne indeholde mere funktionalitet:

Til programmet blev vi stillet følgende krav:

1. Danmarkskortet skal tegnes visuelt ud fra datasættet i et programvindue.
2. Forskellige slags vejsegmenter skal tegnes med forskellige farver, fx motorveje røde, hovedveje blå, stier grønne og resten sorte.
3. Danmarkskortet skal skalerer efter størrelse når der trækkes i vinduets ramme, således at hvis vinduet bliver større eller mindre, skal vinduet stadig vise samme vejsegmenter, men skaleret op i en henholdsvis større eller mindre størrelse.
4. Det skal være muligt at zoome ved at trække en firkant med musen, således at firkanten bliver det nye billede, og alle vejsegmenter inden for firkanten skaleres op til det nye billede.
5. Vis navnet på vejen der er tættest på musen.

Vores egne krav til programmet lød som følgende:

1. Programmet skal lave en inddeling af Kraks data, således at programmet ikke skal tegne hele kortet, når man har zoomet helt ud eller helt ind.
2. Koden til programmet skal være logisk struktureret og vedligeholdelsesvenligt. Det vil sige have lav kobling, og et højt abstraktions- og kohæsions niveau. På denne måde vil programmet også være videre udviklingsvenligt.
3. Mulighed for at bevæge kortet, med musen (Paning).

Problemanalyse

Dette afsnit rummer en argumentation for måden vi har valgt at opbygge systemet og løse de krav vi har fået tildelt. Vi vil diskutere nogle af de centrale overvejelser, vi har gjort os før det endelige design blev fastlagt.

KRAV 1 & 2

Det udleveret datasæt kommer med en parser som læser de 2 filer og opdeler dem i edges og nodes. En vej består af en eller flere edges, og hver edge består af 2 nodes. Kraks datasæt er således lavet, at det kun består af rette linjer (læs edges), så når en vej svinger, har de opdelt vejen i mange små brudstykker, som hver især er rette. Hver edge indeholder information omkring hvilken node den starter og slutter i. Disse nodes indeholder de præcise koordinatsæt til hvor vejen faktisk starter og slutter i Danmark. Al information vedrørende vejtype, vejlængde, postnummer m.m. findes i den pågældende edge. For at tegne Danmarks veje må altså for hver edge finde vej typen samt de 2 tilhørende nodes og tegne et linjestykke mellem disse i den korresponderende farve. Vi oplevede i vores forsøg på at tegne Danmarks veje at datasættet indeholdte havde værdier svarende til Danmarks rigtige størrelse, hvilket skulle skaleres ned. Til at starte med skalerede vi alt ned med en fastslået konstant. Dette viste sig dog at være upraktisk i forhold til nogle af de kommende krav, specielt at skulle have muligheden for at resize vores vinduet. Vi valgte at skifte til at skalere i forhold til størrelsen på vores JComponent.

De nedskaleret tal blev nogle lange decimaltal. Dette blev til et overraskende problem da Graphics som var vores umiddelbare bud på et tegne bibliotek kun tager heltal. Det ville være spildt arbejde for mindre præcision, at runde alle tal af til heltal. Det viste sig at Graphics2D, en klasse som arver fra Graphics, kan tegne former med decimaltal som værdier.
Dette er dog gemt væk bag et interface, så vi ikke er afhængige af at bruge Graphics2D, hvilket vil blive forklaret senere i denne rapport.

Det andet krav kunne ordnes med en switch-case med TYP feltet fra EdgeData som case, dette har vi holdt fast i og ikke overvejet nogle andre løsningsmodeller for, da denne løsning er kodemæssig overskuelig, kort og videre udviklingsvenlig.

KRAV 3

Vores første løsning omkring skaleringen af vinduet var at bruge mouseListeners metoder, mouseExited og mouseEntered således at man tog størrelsen på vinduet ved mouseExited og igen ved mouseEntered og ud fra det, beregne hvor meget større eller mindre størrelsen på vinduet havde ændret sig. Denne løsning fungerede ikke videre godt, da metoderne betragtede musen stadig var indenfor Canvas når man ændrede vinduesstørrelsen, man var derfor nødt til at fører musen ind og ud af vinduesrammen for at få den til at opdatere efter ændring af vinduesstørrelse, hvilket vi så som en fejl i implementationen.

Vi overvejede så at bruge en componentListener. ComponentListeneren har metoden componentResized, som kaldes når componentet (her vinduesstørrelsen) ændres. Man kunne derfor kalde repaint fra denne metode så kortet blev tegnet på nyt, med den nye størrelse.

Til sidst fandt vi ud af, at metoden repaint, herunder metoden paintComponent, bliver kaldt hver gang størrelsen af vinduet ændres. Vi endte derfor med at beregne skaleringen ud fra components størrelse i metoden drawMap, som også tegner Danmarkskortet på ny, og som kaldes hver gang der kaldes paintComponent.

KRAV 4

Vores første forsøg på at implementerer en zoom funktion, var at bruge AffineTransform biblioteket. Dette havde dog det problem, at alt blev forstørret, svarende til at holde en lup over det valgte område, som man ville gøre hvis man ville forstørre et billede. Det vil sige at vores vejes tykkelse også blev skaleret op.

I stedet brugte vi det rektangel, der bliver dannet når man trækker musen, til at hente veje i vores QuadTree. Bagefter kunne rektanglet bruges til at skalere længden på vejene ned, så de passer med den nye vinduesstørrelse.

Vi valgte også at tegne det rektangel man har markeret, mens man trækker musen for at vise området man ønsker at zoome på. Vi overvejede at bruge et glass-pane, som fungere som en glasplade over kortet, som man tegner på, i stedet for at tegne alle veje hver gang størrelsen på firkanten ændres. Pga. tidspress nøjes vi med at tegne rektanglet og kalde repaint hver gang størrelsen på rektanglet ændres, dvs. hele alle veje også tegnes på ny i billedet. Vi ved dette ikke er en optimal løsning, men den var nem og hurtig at implementere. Dette ændre vi nok i anden del af projektet.

KRAV 5

For at finde den tætteste vej på musen, må vi først lave en antagelse omkring hvor på vejen vi laver sammenligning med musen. Er det vejens startpunkt, midtpunkt eller måske et tredje sted?. Vi synes den letteste måde at implementere dette, er at sammenligne med vejens midtpunkt, men dette kan stadig godt skabe upræcision på lange lige veje, da disse kun har ét midtpunkt, mens at buede vejsegmenter har flere. Vi har dog valgt at abstrahere en smule, og sammenligne musen med vejenes midtpunkt, selvom vi godt ved det ikke er helt præcist.

For at finde den tættest liggende vej, ville den umiddelbare løsning ville være at kigge alle veje igennem og sammenligne med musens koordinater for at finde den tætteste. Denne løsning vil dog tage for lang tid. Vi har derfor valgt at opdele alle veje i et QuadTree, som vi søger i. Implementationen af QuadTree vil vi uddybe længere nede i rapporten. Simplificeret har vi delt alle veje op i en masse små firkanter, hvilket gør det muligt for os kun at søge efter den tætteste vej i netop den firkant musen befinder sig i på et givent tidspunkt, hvilket gør søgningen af den tætteste vej på musen væsentligt hurtigere.

For at vise vejen har vi et JLabel, som vi har placeret i bunden af skærmen under kortet. Vi opdagede at kortet godt kunne “hoppe” på bestemte tidspunkter, når navnet på den tætteste vej ikke fandtes, hvilket resulterede i at JLabel’et forsvandt, og ændrede skaleringen på kortet, hvilket fik kortet til at “hoppe” på skærmen. For at løse dette problem ændrede vi søgningen, så vi finder den tætteste vej med et navn, i stedet for bare tætteste vej. Også indsatte vi teksten: “Vejen kunne ikke findes!” når der ikke var noget vejnavn at vise. Sidstnævnte sker dog kun når firkanten i QuadTræet ikke indeholder nogen veje, hvilket sjældent forekommer.

VORES KRAV 1 & 2

Programmet er designet med udgangspunkt i MVC (Model - View - Controller) design pattern. Da man har Model til at opbevare dataen fra Krak, View til at vise dataen i form af et kort og Controller til at håndtere input fra brugeren. Derfor er vores program lavet med udgangspunkt i MVC, da det giver en god struktur, der er nem at arbejde med.

En af de store udfordringer ved projektet var at lave Model ordentligt, selvom vores kort ‘kun’ skal tegne Danmark, er der stadig nok linjer og punkter der skal bearbejdes til at programmet vil køre meget langsomt,

hvis ikke det var optimeret. For at finde vejnavnet der er tættest på musen ville det være spild af tid at gå alle vejene igennem og sammenligne med musens position, hver gang musen bare bevæges 1 pixel. Ligeledes vil det være spild af computerkraft, at tegne hele Danmark, hvis man eksempelvis er zoomet ind på Amager.

Til at starte med havde vi to ArrayLists, med henholdsvis edges og nodes. Det var nemt at tegne ud fra, men det var svært at filtrerer unødvendig data fra, hvilket medførte programmet kørte meget langsomt.

To muligheder for at sortere i dataen, som vi overvejede at benytte os af, var: KD-tree og Quadtree, der inddeler dataen i mindre rektangler, på hver deres forskellige måder.

Vi valgte at implementerer Quadtree, da det opdeler dataen i fire lige store rektangler, hvilket virker mest relevant, i forhold til at det er et kort, i modsætning til KD-tree, der opdeler kortet i en masse forskellige rektangler alt efter punkternes fordeling. Begge metoder ville give os mulighed for at sortere i dataen. Vi så en åbenlys måde at implementerer Quadtree og valgte derfor denne.

For at gemme dataen i vores Quadtree, oprettede vi klassen Road, som indeholder dataen fra både edges og nodes. For at gøre det nemt at lægge Road objekter i vores Quadtree, valgte vi at lave et midtpunkt til vær Road (dvs. hvert vejstykke), som man kunne opdele efter. Dette medførte dog problemet, at nogle steder ville man kunne se at der manglede veje rundt i kanterne af kortet, hvis man er zoomet ind. Dette har vi delvist løst ved, at forstørre det område man faktisk tegnede, til 125% af det man ser. Det betyder at vi tegner lidt mere data end nødvendigt, men også at hvis der er en meget lang og lige vej, at denne har midtpunkt uden for de ekstra 25%. Dette er også dele vi vil overveje i 2. del af projektet, dog er det ikke bemærkelsesværdigt i nuværende implementation af programmet.

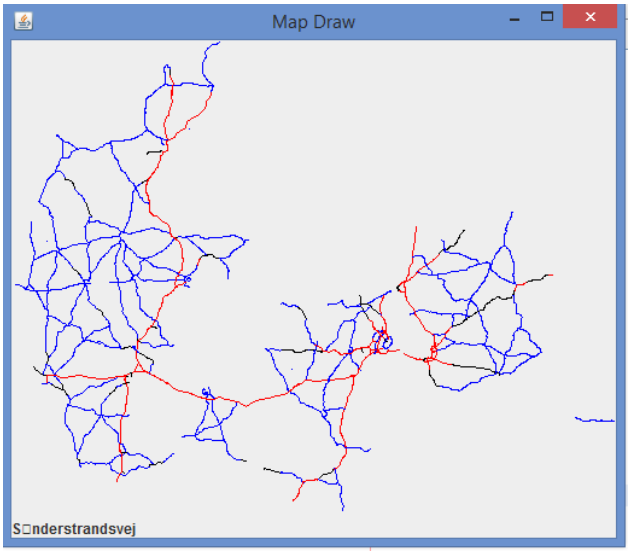
I forhold til de krav vi stillede os selv, løste vi det første via et quadtree. Det andet har vi, for at opnå et højt abraktionsniveau og løs kobling, lavet to interfaces til de to klasser vi ville være afhængige af. Netop vores quadtree og dets data, samt et interface til vores Graphics2DDraw klasse, hvis metoder vi bruger når vi tegner kortet i klassen Canvas. Det gør at vi til hver en tid meget nemt kan skifte vores datasortering til fx. KDTree og tegneklasser til Graphics eller andet.

VORES KRAV 3

I forhold til det tredje krav, med paning havde vi næsten allerede en løsning fra zoom, da programmet arbejder med rektangler, har vi simpelt flyttet det rektangel vi skalerede med, ved at lægge den afstand musen havde flyttet sig, til rektanglets øverste venstre hjørne, hvilket resulterede i en præcis og god følelsesmæssig paning-effekt.

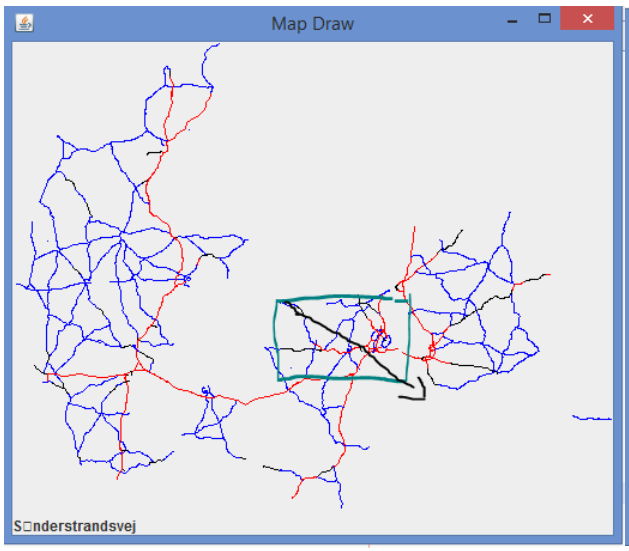
Brugervejledning og eksempler

Når programmet startes vil man se kortet, zoomet helt ud (se figur 1), når der er zoomet helt ud, vil der kun blive vist motor- og hovedveje, efterhånden som man zoomer ind, vil flere typer veje blive vist. I bunden af kortet vil navnet på vejen nærmest musen, blive vist.

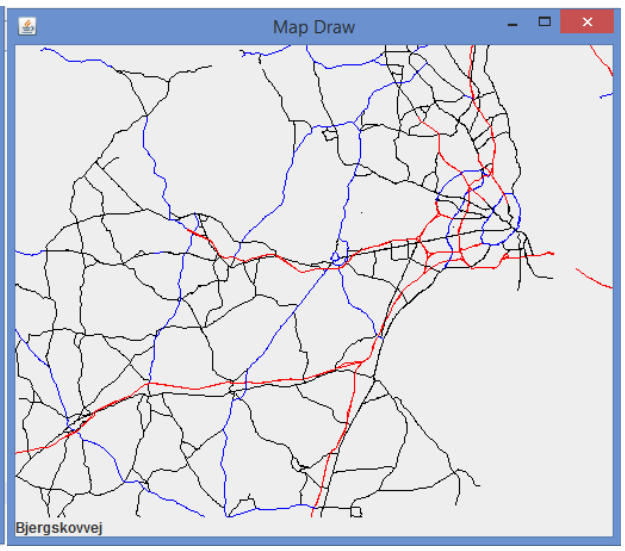


Figur 1

Ved at trække en firkant, med venstreklik, kan man zoome ind på det område rektanglet indkredser (se figur 2 og 3).

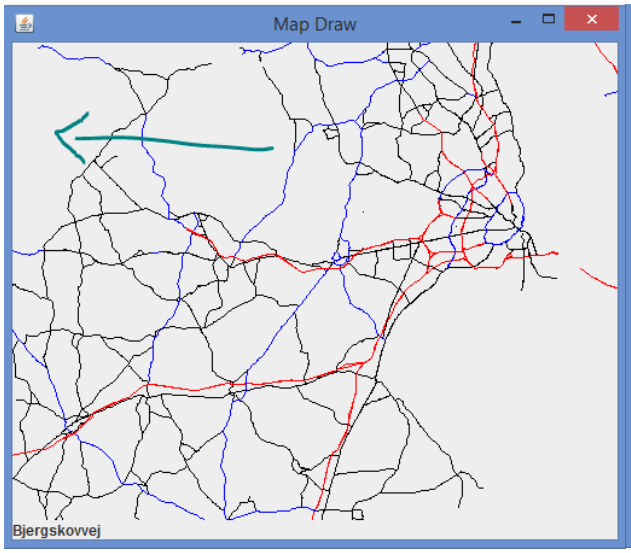


Figur 2

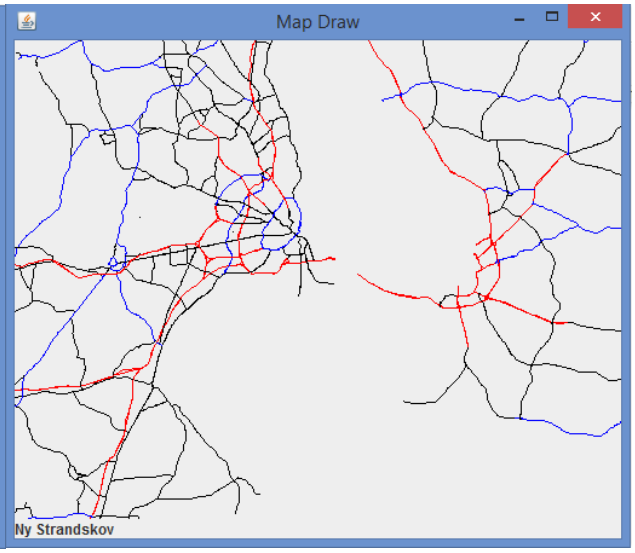


Figur 3

Man kan også bevæge kortet med musen ved at trække med højre klik.



Figur 4



Figur 5

Ved et enkelt venstre klik på kortet, kan man genskabe det originale zoom og centrering. Når man ændre vinduets størrelse vil kortet skalere til at passe, i forhold til den mindste akse.

Teknisk beskrivelse af programmet

Struktur

Programmets struktur tager udgangspunkt i MVC design pattern. Helt grundlæggende svarer ML klassen til Control, Canvas klassen til View og CurrentData klassen til Model.

I vores ctrl pakke har vi StartMap-klassen, der opsætter View, og loader-dataen ind i Model, samt ML der er vores mouseListener/mouseMotionListener, der ændrer Model når brugeren gør noget.

I model pakken har vi CurrentData der indeholder den data, View skal vise. Road klassen svarer til et stykke vej bestående af en edge og to nodes, samt et midtpunkt. Udover disse klasser bruger Model også QuadTreePack til at gemme data og hente det der skal bruges.

I view pakken har vi Canvas klassen, der tegner det data CurrentData indeholder. Canvas og CurrentData bruges i et observer pattern, hvor Canvas er observer og CurrentData er observable. View indeholder også et interface med metoder til at tegne streger og ændrer farve, samt interfacets implementation.

Tilsidst har vi krakloader pakken, der indeholder det udleverede kode fra krak, med få modificeringer, der kan hente data fra filerne.

Klassediagrammet kan ses i figur 6 i bilaget.

StartMap

Klassen ligger i ctrl-pakken og indeholder public static void main metoden. StartMap opsætter vinduet, samt læser og indsætter dataen, vha. den KrakLoader, der fulgte med dataen.

Road

Et Road-objekt indeholder en EdgeData og de to tilhørende NodeData tildelt fra KrakLoader. Ved instantieringen af Road-objektet, udregnes også et midtpunkt for den graf EdgeData og de to NodeData udgør, som bruges til at indsætte i QuadTree og søge efter nærmeste vej.

CurrentData

CurrentData bruger singleton design pattern da der kun skal bruges én instans af klassen. Klassen indeholder den relevante information, som Canvas skal tegne.

CurrentData extender også klassen Oberservable, som er Javas implementation af den ene type i observer design pattern. Som Observer har den Canvas, hvilket betyder at Canvas vil blive gjort opmærksom på ændringer i CurrentData.

CurrentData indeholder: En ArrayList med alle de relevante Roads at tegne, et Rectangle2D der svarer til den områder af kortet man ser hvilke koordinater det øverste venstre hjørne af kortet svarer til, og størrelsen på området der tegnes når man zoomer helt ud.

Canvas

Klassen Canvas, som udgør hoveddelen af View, kan skalere og tegne et kort ud fra en ArrayList af Road objekter. Canvas tegner ved hjælp af DrawInterface, der har metoder til at tegne en linje og et rektangel ud fra decimaltal, samt metoder til at skifte farve på det der tegnes. DrawInterface implementeres af Java2DDraw, der bruger Javas Graphics2D, til at tegne og skifte farve.

Canvas tegner den data der ligger i CurrentData og implementere derfor interfacet observer, der bruges sammen med CurrentData, der er observable.

Canvas extender også JComponent, hvilket gør det muligt at tildele det til en frame. Da klassen extender JComponent har det også en højde og bredde, hvilket bruges til at beregne skaleringen. Området man kan se og som skal skaleres ned, gemmes som et Rectangle2D i skaleringen 1, så den svarer til de værdier punkterne normalt har og de værdier Quadtree'et bruger.

ML

ML implementere mouseListener og mouseMotionListener, og er den klasse der håndterer input fra brugeren. Den implementerer begge, da de forskellige funktioner skal bruge både klik, træk og bevægelse med musen.

Mange af de beregninger, der laves når man zoomer og trækker kortet, ligger i ML, da værdierne skal skaleres op og rykkes, for at passe med punkternes koordinater. ML bruger information fra både CurrentData og Canvas til at beregne det nye område der skal tegnes og/eller beregnes hvor musen er i forhold til punkterne.

DrawInterface & Graphics2DDraw

DrawInterface er et af vores mange skridt for at opnå løs kobling, samt et højere abstraktionsniveau. Interfacet indeholder kun få metoder vi har brug for, for at kunne tegne og farve vores grafiske præsentation af Danmarks veje. Det er dog yderst nemt at tilføje nye metoder. Da vi i vores program har brug for at tegne linjer og firkanter af double værdier, har vi valgt at implementere vores DrawInterface på en klasse vi har kaldt Graphics2DDraw. Denne klasse gør brug af Graphics2D til at tegne streger og firkanter.

QuadTree

Pakken QuadTree i programmet indeholder klasser der tilsammen danner et quadtree som kan sortere objekter af typen road der er lavet i en af de andre pakker.

Pakken indeholder klasserne: QuadTree, Boundary, Center, NSEW og QuadTreeInterface. De er nævnt i rækkefølge efter hvor meget der foregår i hver enkelt klasse. QuadTree indeholder to public metoder, "public void insert(Road rd)" som tager et "Road" objekt og indsætter det i quadtræet, her bliver holdt styr på mængden af indsatte objekter i hver quad og der bliver delt op i fire mindre quads når grænsen af en quad er nået.

Den anden er "public ArrayList<Road> search(double x1, double x2, double y1, double y2)". Metoden tager to punkter der repræsenterer det område på kortet du ønsker at få veje fra, og de fundne veje bliver returneret i en arrayliste. Selve search metoden er et mellemled der kalder videre til den private metode "getRoads" som laver rekursive kald ned igennem quadtræet for at finde de relevante veje. Search laver en arrayList og giver den som pointer til getRoads, der så bruger den til at gemme vejene i efterhånden som de bliver fundet.

Klassen Boundary bliver lavet i hver Instans af klassen QuadTree, og indeholder informationer om den pågældende quads grænser. dvs. hvilket koordinatsæt den dækker, samt metoder til at checke om et givent punkt eller en given firkant er indenfor/overlapper grænsen. Boundary opretter også en instans af Center klassen til at holde styr på midtpunktet i den pågældende quad.

NSEW klassen er en enum der bruges til at styre hvad der skal ske når en ny quad laver i henholdsvis nordøst, nordvest, sydøst og sydvest retningerne. Klassen boundary bruger denne information til at udregne de korrekte grænser og center punktet.

QuadTreeInterface er lavet for at gøre det nemt at se hvad man skal give og hvad man får tilbage når man kalder quadtræet, klassen indeholder abstrakte versioner af insert og search som implementers i QuadTree.

GUI:

Vores GUI er opbygget af en frame med layouttypen: BorderLayout, som opdeler frame i fem dele: nord, center, syd, øst og vest. I center har vi placeret instansen af Canvas og i syd har vi placeret det label hvor navnet på den nærmeste by står.

Afprøvning / tests

Manuel test:

Da programmet afhænger meget af brugerinput via musen har vi valgt at holde os til manuelle test af vores program.

Test af zoom:

Zoom med rektangel, der passer med vinduet	Fungerer helt perfekt, ingen fejl fundet
Zoom med et smalt og langt rektangel	Tegner kun kort, på et smalt område af vinduet.
Test for grænsen af zoom	Hvis man zoomer langt nok ind vil den stoppe med at tegne nogle streger (sandsynligvis pga. anti-aliasing), men det er ikke noget der skaber problemer ved normalt brug.
Test af zoom, hvor musen slutter udenfor vinduet	Den zoomer, men det områder der tegnes, bliver ikke skåret af ved kanten af vinduet, hvilket resulterer i at den egentlig zoomer ud hvis firkanten trukket er større end det visuelle vindue. Vi ser det som et problem, da den helst ikke bør zoome ud, på den måde.

Det omtalte rektanglet er rektanglet der tegnes når man trækker musen. Det er også et problem at denne forsvinder når musen står stille.

Test af reset:

Dette fungerer fint, da den ikke har nogle variabler og reagerer på et klik. Ingen fejl fundet.

Test af paning:

Normal paning, inden for vinduet	Fungerer som det skal. Ingen fejl fundet
Paning, med mus uden for vinduet	Paningen fortsætter når musen kører ud over vinduet, hvilket vi ikke ser som et problem, men giver egentlig brugermæssigt god mening.
Test af grænsen for paning	Umiddelbart, kan man trække så langt til siden man vil, men der tegnes så heller ikke noget. Muligvis vil programmet loope til sidst pga. doubles max størrelse, eller programmet vil kaste en exception. Disse fejl har vi ikke stødt på, men kunne i teorien godt ske.

Test af nærmeste vej:

Nærmeste vej, ved normal bevægelse af musen	Fungerer som det skal. Ingen fejl fundet.
Nærmeste vej, mens man zoomer/panner	Stopper med at opdatere indtil man stopper med at trække.
Nærmeste vej, mus ude for vinduet	Opdatere ikke.
Nærmeste vej, mus ude for det område der tegnes, men inden for vinduet.	Opdatere ikke.

Test af flere input samtidig.

Zoom og paning samtidig, zoom først	Den panner normalt og afbryder zoom.
Zoom og paning samtidig, zoom sidst	Den zoomer normalt, når man slipper første museknap, men zoomer igen når man slipper den anden.

Programmet understøtter de funktioner det skal, men der er nogle ekstreme situationer, samt nogle situationer, der ikke normalt sker, der ikke fungerer helt efter hensigten.

Refleksion over gruppearbejdet

Brugen af GitHub, Google Docs og Facebook.

Til versionsstyring af koden, har vi gjort brug af Git. Vi havde alle hørt om Git og GitHub, men brugte før en gruppe på Facebook til at dele den nyeste implementation. Brugen af Git har gjort det nemmere for os at arbejde flere personer samtidigt og mere effektivt på samme kode, uden manuelt at skulle lave rettelser. Dog har konverteringen til brugen af Git, herunder slåskampe med git-kommandoer og løsning af konflikter, også taget en del af vores tid i starten af læringsforløbet. Vi er dog alle blevet gode til brugen af Git nu, og regner derfor med at få tiden tilbagebetalt gennem øget effektivitet. Summa summarum er vi glade for at have blevet introduceret til Git.

Ved brug af Google Docs har vi kunnet dele dokumenter med hinanden meget nemt over internettet. På vores Google Docs ligger både vores samarbejdsaftale, logbog og rapport samt andre dokumenter vi ønsker at dele og redigere i. På den måde har vi alle haft adgang til de nyeste delte dokumenter, aftaler og logs, som vi også kunne skrive på i fællesskab samtidigt.

Som primær kommunikationsplatform har vi gjort brug af Facebook, hvor vi har oprettet en gruppe hvor alle beskeder angående projektet bliver delt. Alle har mulighed for at tjekke op på de nyeste beskeder, og da vi alle er brugere af Facebook i dagligdagen, burde disse også blive læst hver dag, men som der vil blive forklaret under "Møder og kommunikation", kunne gruppen godt have været bedre til at kommunikere. I starten brugte vi også Facebook-gruppen til deling af nyeste version af koden, men denne er som sagt blevet erstattet med versionsstyring programmet GitHub, da vi med sidstnævnte har opnået større effektivitet.

Samarbejdsaftalen, Gantt diagram, logbog og worksheets.

Samarbejdsaftalen blev udviklet i starten af projektet som en slags "kontrakt" vi havde imellem os. Vi var alle enige om hvordan arbejdsforholdene skulle være i gruppen. Samarbejdskontrakten er blevet overholdt som den skulle angående mødetider, kommunikationsform og arbejdsmorale. Den fik os til at diskutere hvilke normer og arbejdsforhold vi ønskede under projektet, så der var enighed fra starten af projektet.

Gantt diagram var noget vi alle gerne ville have med i planlægning af projektet, dette følte vi dog ikke nødvendigt i første del af projektet. Vi regner dog med at gøre brug af dette værktøj til anden del af projektet, således at der ikke sker noget tidsspilde, tidspres og så alle ved hvad de skal lave hvornår. Efter erfaring med aftalen skal den nok revideres i forhold til møder, hvilket vi vil blive beskrevet senere, under "Møder og kommunikation".

Vi har været gode til at skrive logbog til hver arbejdsgang, således at alle gruppens medlemmer kunne følge med i projektets kronologiske udvikling. Ligeledes kan vi nu også læse projektudviklingen bagudrettet. Indtil videre har vi ikke haft særlig brug for logbogen rent produktivt, men dette ændres nok når vi skal skrive anden del af projektet.

Worksheets valgte vi at skrive bagudrettet, så vi kunne se hvem der havde lavet hvad og hvornår. Vi har forsøgt både at skrive worksheets gruppevis og individuelt. Worksheets er desværre ikke noget vi har brugt så meget endnu, nok fordi det er et nyt arbejdsredskab, som vi endnu ikke helt har lært at arbejde med endnu. Vi har dog tænkt os at undersøge mulighederne med dette værktøj nærmere i anden del af projektet.

Arbejdsfordeling og teamroller.

Arbejdsfordelingen har været ligeligt fordelt. Vi har hver vores nøglekompetencer, som vi har forsøgt at udnytte. Alle har dog både været med til at kode, skrive rapport og deltage i arbejdsplanlægningen. Personerne i gruppen der har været bedre til at kode fra start, har stået for en større del af koden, men undervejs er der blevet udvekslet idéer, valg og forklaringer af implementationer gennem hele forløbet, således at alle i gruppen har forstået den, lært fra den, og har haft mulighed for at arbejde videre på den. Ligeledes har andre været mere med til planlægning af gruppearbejdet således at gruppen var på rette vej under hele forløbet, at deadlines blev overholdt og arbejdet diskuteret og dokumenteret. Alle i gruppen har været med til hele forløbet, men samtidig har vi også forsøgt at udnytte vores nøglekompetencer så godt vi kunne, hvilket har resulteret i et stærkt team, med tilfredse gruppemedlemmer. Vi har ikke diskuteret teamroller gennem projektet, da vi ikke har lyst til "at sætte folk i bås", i stedet har vi ladet folk selv vælge hvilken rolle de vil spille i gruppearbejdet, hvilket også har fungeret meget godt.

Andre arbejdsværktøjer

Gruppen har også gjort brug af "Parprogrammering" til mere indviklede dele af programmet. Dette har været en effektiv arbejdsmetode for nogle af os, da koden tænkes dybere igennem og fremadrettet mens den skrives. Vi har opnået færre fejl, og har brugt kortere tid på at udvikle særligt komplicerede implementationer til vores program.

Møder og kommunikation

Vi aftalte et fast møde om ugen, hver mandag. Her mødtes vi samlede op på, hvad folk havde lavet og aftalte hvad der videre skulle ske. Dette fungerede godt, mens der ikke var nogle problemer, da folk fik frihed til arbejde som de ville.

Med facebook og Skype havde vi gode muligheder for at kommunikere, men vi har ikke været så gode til at udnytte det og kombineret med de få møder gjorde det, det svært at løse problemer hurtigt.

Det problem løb vi ind, da nogle problemer med strukturen, gjorde det svært at arbejde videre med koden. Dette gjorde at vi i to uger næsten ikke gjorde nogen fremskridt, da vi måtte droppe et møde og først næste møde begyndte at kigge på problemet, samtidig kæmpede vi også med Git, som også havde delt skyld i tidsspildet.

Til næste del af projektet, ville et ekstra, fast møde om ugen, nok være en god idé, samt at folk prøver at blive bedre til at kommunikere.

Konklusion

Vi har lavet et program der grafisk præsentere Danmarks veje med forskellige farver efter vejtypen. Vi har løst de opstillede krav, om reskalering af vinduet, zooming og tætteste vej på musen samt tilføjet ekstra funktionalitet. Vi har så vidt muligt bestræbt os på at skrive en kode der har en logisk struktur og er vedligeholdelsesvenlig. Vores program har ingen store mangler, og de mindre mangler vil blive diskuteret i diskussionsafsnittet.

I forhold til gruppearbejdet, har vi fundet ud af hvad der fungerer og ikke fungerer, så vi kan gøre det mere effektivt, i næste halvdel af forløbet.

Diskussion og mangler

Angående at vise den tætteste vej på musen, så er der en nuværende fejl ved visning af bogstaverne "Ææ", "Øø" og "Åå", som i stedet vises som "□". Vi mener dette kan skyldes tekst formatet på det JLabel der viser vejens navn, eller parseren fra tekstfilerne til arrayLists. Formatet skal nok ændres, således at Æ, Ø og Å også vises (UTF-8), dette har der dog ikke været helt tid til endnu, men vi vil se på det i anden del af projektet.

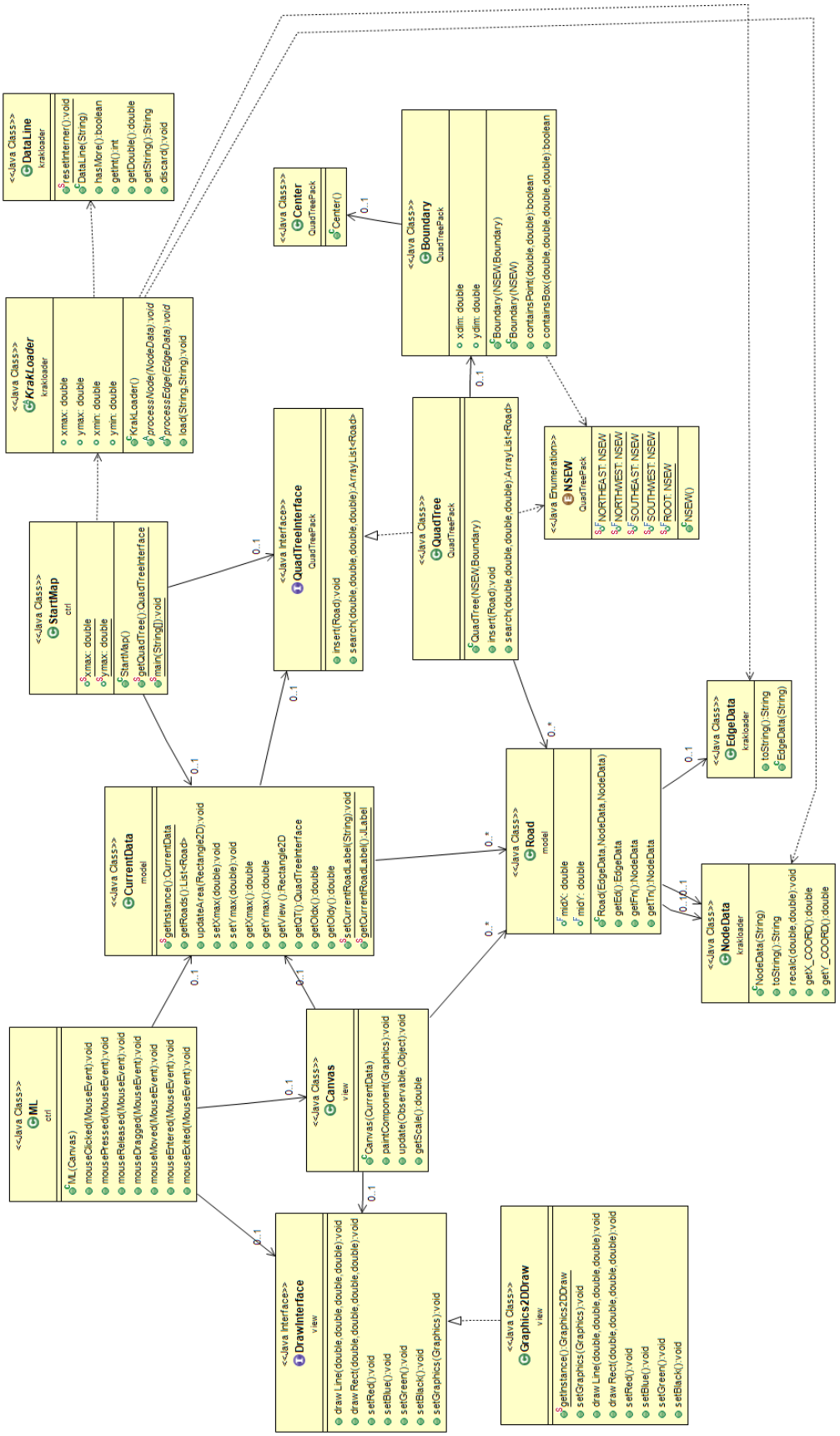
Vi har optimeret vejvisningen idet at vi kun vise de største veje når man er zoomet helt ud. Det har dog den konsekvens at mindre og nogle større øer forsvinder helt fra kortet, indtil man zoomer ind. Dette kan afhjælpes ved at tegne kystlinjen for alle øer i Danmark, som vi vil kigge på i anden del af projektet.

Firkanten som markerer hvor man zoomer ind, forsvinder når man holder musen stille. Dette sker fordi metoden til at tegne firkanten kun bliver kaldt når musen er i bevægelse, hvilket er et mindre problem, som forhåbentlig kan løses ved nærmere undersøgelse af events, herunder consume() metoden, og repaint.

Derudover opfører zoom og pan sig lidt forkert, ved specielle input, der ikke sker så ofte, hvilket burde kunne løses, ved nogle flere check af inputtet.

Bilag

Obs: Bilaget findes på efterfølgende sider…



Figur 6 Klasse diagram over vores program.

Samarbejdstalen:

Gruppe møder:

-Hvornår
Permanent dag: Mandag fra klokken 10:00
Mulig dag: Onsdag (Eftermiddag efter 12)
-Hvor
ITU
-Hvordan
Opsamling fra sidste møde - Worksheet færdiggøres fra sidste uge
Nuværende status
Planer for dagen
Planer til næste gang - Worksheet skrives
Gantt diagram vurderes i forhold til status - OBS: Glemte vi til første del, men har tænkt os at gå i krig med til anden del.
Logbogsskrivning
-Andet
Kan man ikke komme informeres det via en eller flere kommunikationsmetoder så hurtigt så muligt
Hvis 1 ikke kan komme mødes vi alligevel
Hvis 2 ikke kan vurderes situationen, evt skype møde.
Logbogsskrivning går på tur
Pauser - 10 min per time.
Frokost pause
Lang dag - en til længere pause udover frokost pause
-Ambitioner:

God karakter, men ikke på bekostning af andet.

Tidsplanlægning:

Gantt diagram

Gruppe logbog:

Agenda
Færdiggjort arbejde
Store beslutninger
De uddeligeret opgaver
Andet
Kommunikations metoder:

Facebook
Skype
Google drive
Telefon
Arbejdsformer
Der er mulighed også for at arbejde på egen hånd, hvorefter der deles med gruppen.
Vi skal fortælle hinanden hvis vi har fundet ud af noget nyt, som vi bruger til implementationen.

Worksheets

Worksheet 1 Group A

Work subset	Drawing the map - Requirements 1-3
Step	Basic features
Date	03/03/2014
Assigned to	Whole group
Description of work	• Drawing the map • Scaling map to frame • Road types assigned to a color
Implementation	Canvas

Next step	• Zoom
	  o Quad tree
	• Class structure
	• Detect nearest road
	  o Priority queue

Issues	Current implementation of zooming does not work properly. Consider hardcoded aspects of current implementation.

Work subset	Implement QuadTree - Expansion
Step	Create QuadTree
Date	17/03/2014
Assigned to	Johan, Peter helped by Kristian
Description of work	• Implemented a quadtree
Implementation	QuadTree Package
Next step	• Integrate quadtree into program
Issues	Can’t query for nodes colliding with selected area..

Work subset	Implement zoom - Requirement 4
Step	Detect area of zoom - Draw and zoom
Date	17/03/2014
Assigned to	Adam & Peter
Description of work	• Draw zoom area (Done) • Zoom, so that box = new window (Done) • Zoom accurate (Under progress) • Zoom using QuadTree (Under progress)
Implementation	Canvas, QuadTree Package
Next step	• Make zoom more accurate • Zoom using Quad tree
Issues	Current implementation of zooming does not work properly. Zoom area and zoom is not accurate. We need the implementation of QuadTree to be done before starting implementing QuadTree to zoom method.

Worksheet 2 Group A

Work subset	Class structure, zoom
Step	Rework class structure after MVC, implement better zoom
Date	26/03/2014
Assigned to	Kristian
Description of work	• Restructure classes in package after the Model View Controller pattern. • Improve the zoom function to zoom correctly
Implementation	All packages/classes, CurrentData
Next step	• Implement optional features
Issues	None known

Work subset	Zoom, draw zoom area, graphics interface, report writing
Step	Finishing some stuff regarding the graphics
Date	26/03/2014
Assigned to	Adam
Description of work	• Drawing the map • Scaling map to frame • Road types assigned to a color
Implementation	Canvas
Next step	• Implement optional features • Fixing small bugs
Issues	The drawed square disapears when the mouse is not moving

Work subset	Implement QuadTree into program, Closest road, Plan and start report writing
Step	Finish QuadTree, debug and make adjustments, and make it work with other functions, implement closest road, research what to write in the report, and start writing.
Date	26/03/2014
Assigned to	Peter
Description of work	• Finish implementation of QuadTree • Debug and make it work properly • Adjust other methods to work with QuadTree • Implement Closest Road • Make it show in canvas • Plan and start report writing
Implementation	QuadTree Package, ctrl Package, model Package.
Next step	• Implement optional features • Write report
Issues	Report not done yet.

Work subset	Integrate QuadTree into program, Implement closest road feature
Step	Finish QuadTree and make it work with our map functions. Make a feature that shows the road closest to the cursor when browsing the map.
Date	26/03/2014
Assigned to	Johan
Description of work	• Finish implementation of quadtree • Fixed final bugs in quadtree and implemented it into the project • Implemented feature to show name of road closest to the cursor, using the quadtree.
Implementation	QuadTree Package, ctrl Package, model Package.
Next step	• Implement optional features • Write report
Issues	None known

Worksheet 3 Group A

Work subset	Write first part of report and make adjustments to code
Step	Write, edit, format and send group project
Date	01/04/2014
Assigned to	Everyone
Description of work	• Write the first part of the report • Edit and correct until satisfied • Format it so it looks nice • Add it to the project and send group project • Make small adjustments to code if necessary
Implementation	Førsteårsprojekt del 1.docx
Next step	• Implement optional features • Plan next part of project
Issues	None known

Logbog

Logbog d. 24/02-2014 - Forfatter: Peter.

24. februar 2014
15:04

Mangler:
Samarbejdsaftale.
Ét worksheet pr. person.
Fordeling af opgaver.
Overblik over arbejdsopgaver.
Overblik over tidsfrist og tidsplan

Diskussion:
Krak vs. Open
Kristian og Adam: Helst krak
Peter: Vi kigger på de forskellige datatyper, det er for tidligt at beslutte noget endnu. Open giver måske nye muligheder for funktionalitet end krak, men krak er nok nemmere at behandle.

Plan:

Kig på data og vælg datagruppe. - **(FDS: Dette diskuteres og undersøges næste gang)**

Vi aftaler en dag i denne uge på facebook (nok i aften) - Tirsdag (i morgen) er nu forslået.
Undersøge: Hvordan man tegner kortet i det hele taget. - **(FDS: Dette diskuteres og undersøges næste gang)**
Interesseret i github, men dropbox kunne også bruges. - Oplæg om Github af TA's TBA
KD-tree eller Quad-tree. **(FDS: Dette diskuteres og undersøges næste gang) - dog skal fokus være på at tegne kortet.**

Information:

d. 7-11 Adam Engsig er på skiferie.
Johan er syg i dag, også vil vi gerne læse lidt op på opgaven, før vi laver samarbejdsaftale og worksheets.

Mangler til næste gang:
Vi aftaler i aften hvornår vi mødes næste gang.

Næste gang:
Analyser opgaven. Lav evt. interfaces. Samarbejdsaftale. Diskussion og undersøgelse af data, tegn af kort, KD-tree og Quad-tree.

Logbog d.25-02-2014 - Forfatter: Adam

Samarbejdsaftale er blevet lavet.
Vi kigger på kraks data, dokumenter givet hertil og deres loader til på torsdag hvor vi holder skype møde kl 17.

- Evt krak vs openstreetmaps. Hvad skal vi bruge?
- Basic graph library? Edge og nodes.

- Læs dokument og kode.

Vi vil bestræbe os på at få tegnet kortet til på mandag, men den endelig dato for at tegne kortet bliver bestemt under skype mødet

Logbog d.27-02-2014 - Forfatter: Peter

45 minutters møde på Skype.
Data er bestemt: Krak.
Der blev besluttet at se på det over weekenden, vi mødes mandag d. 3/3 kl. 11 og koder kortet færdig.

Logbog d.03-03-2014 - Forfatter: Kristian / Peter

Mødtes kl 11:00 og arbejdede til 15:40 hvor der var møde med Søren. Herefter lavede vi logbog/worksheet og aftalte næste møde / hjemmearbejde.
Da vi mødtes begyndte vi med at se på hvad folk havde fundet frem til og valgte at arbejde videre med Johans kode, da han havde fået den til at tegne Danmark. Derefter blev der arbejdet med at få kortet til at passe i frame og skalere i forhold til det, når det ændres. Til sidst begyndte vi at kigge lidt på at zoome.
Til næste gang vil Johan og Adam kigge på zoom/quadtree, Peter vil kigge på nærmeste veje og Kristian vil kigge på klasse struktur, da koden pt. er ret rodet.
Vi har aftalt at mødes næste mandag kl. 11.
Note: Adam er på skiferie.

Indtressante begreber - af Peter:

Open street maps - data. - Større datasæt, mere detaljeret, hver hustand har en knude.
Quadtræ.
OpenGL - hold os til swing siger Søren, indtil det virker begrænsende.
AffineTransform: http://docs.oracle.com/javase/7/docs/api/java/awt/geom/AffineTransform.html
Model view controller pattern.

Fra mødet med Søren - af Peter:
Dårlige valg skrives i rapporten, der kan spørges om ekstra planer.
Ingen rapport til første del, men flere requirements og rapportskrivning venter.
Vi kan gå amok med ekstra implementationer - udnyt dataen vi har fået tildelt, overvej navigation og multithreading.
Eksamen: Andel del af projektet som fokus. Giv præsentation af hvad vi har lavet, salgsagtigt/forsvar, hvor vi står, gode valg/dårlige valg. Forstå og refflektér.

Logbog d.10-03-2014 - Forfatter: Kristian/Peter

Dagen startede med undervisning af TA's hvor vi lærte om Git og GitHub. Vi tænkte vi af nysgerrighed ville prøve dette.
Pga. det kun var Peter og Kristian der kunne møde, valgte vi at udskyde videre arbejde med vores projekt. I stedet har vi sat os ind i Git og GitHub.
Vi fik opsat et repository som vi alle har tænkt os at arbejde fra. Vi prøvede push og pull funktioner.
Vi mødes i morgen tirsdag efter undervisning i Diskret Matematik, da mandagsmødet ikke blev til noget. Vi sætter alle op på GitHub når vi mødes tirsdag.

Logbog d.11-03-2014 - Forfatter: Peter

Johan, Kristian og Peter mødtes efter Diskret matematik. Vi snakkede om Github og fik Johan connected til vores repository også. Også snakkede vi om alternative implementationer til tegnemetoden for ændring af vinduesstørrelse.
Den nye implementation er pænere kode, og muligvis også mere effektiv, men det kunne godt se ud som om den kommer i komplikationer med vores Zoom metode. Dette har vi dog ikke fået bekræftet endnu da koden opførte sig meget underligt på den computer vi brugte. Til næste gang tjekker vi op på om vores zoommetoden er forandret efter den nye implementation. Også ville Kristian lægge projektet op på Github således at vi kunne "pushe" og "pulle" fra denne i fremtiden.
Næste gang vi mødes bliver formentlig på mandag d. 17 kl. 11 som sædvanligt. Vi mangler at få sat alle op på Github, også mangler vi at løse det sidste krav: Requirement 5 - Display closest toad name. Der er frit slag på at arbejde med hvad man vil.

Logbog d.17-03-2014 - Forfatter: Kristian / Peter

Vi startede med at forbinde alle, samt lægge projektet på github.itu.dk. Efter lidt tid flyttede vi det dog over på github.com, da nogle havde oplevet problemer med at forbinde til det hjemmefra. Bagefter begyndte vi at arbejde på projektet: Adam og Peter arbejdede videre med zoom og ryddede lidt op i koden, Johan og Kristian arbejdede på at integrere Johans implementation af et quadtree i programmet. Peter hjalp Johan til sidst, men quadtree er stadig ikke færdig. Den bliver loaded i starten af programmet, men ikke brugt endnu. Vi skal nok lave vores implentation om så vores requirements understøtter quadtree.

Logbog d.24-03-2014 - Forfatter: Peter / Johan / Kristian / Adam

Vi mødtes kl. 11, planlagde hvad vi ville lave for dagen. Kristian havde lavet strukturen i projektet om, således at den understøttede Model View Controller pattern. Dette synes gruppen var godt, og vi skiftede så til denne version på GitHub.
Johan og Peter brugte dagen på at debugge quadtree for at få det til at virke, efter mange timers arbejde lykkedes det, dog blev kortet stadig ikke tegnet, sandsynligvis på grund af paint metoden ikke tegner koordinaterne korrekt. Denne skal ændres, så den ikke bare tegner kortet, men gør det "smart", således at vi kan få musekoordinater præcist oversat til kort koordinater. Også skal andre requirements metoder ændres, således at de også udnytter QuadTree.
Kristian har arbejdet på at scale det område man zoomer på rigtigt: Dette fungerer pt. i et test projekt, men er endnu ikke blevet sat ind i det rigtige projekt, da quadtree ikke virkede mens Kristian arbejdede med det.
Adam har påbegyndt rapporten, samt arbejdet på den grafiske del. Der er blandt andet blevet lavet et graphicsInterface for mere løs kobling
På onsdag snakker vi om rapport, dertil skal dokumentet Peter Sestoft - "Udformning af rapporter" læses.
Næste mødedag: Onsdag d. 26/3-2014: Peter møder ca. kl 12:30. Adam ville møde lidt før, Kristian skulle på arbejde men ville møde senere, og Johan møder også omkring 12-12:30.

Logbog d.26-03-2014 - Forfatter: Peter / Johan

Vi mødtes fra kl. 12, og hvornår folk kunne mødes efter. Peter og Adam startede med at skrive videre på rapporten. Da Kristian og Johan ankom, og gruppen samledes, snakkede vi om hvordan vi kunne samle projektet. Vi støtte så på fejl med blandt andet QuadTree'et som vi hurtigt fik rettet. Efter at QuadTree'et kom op at køre, arbejdede Johan og Peter videre med at implementere nærmeste vej funktionen. Adam og Kristian fortsatte med at få zooming til at virke korrekt, samt at implementere paning og filtrering af kortets vejsegmenter.
Vi har nu fået programmet til at virke efter omstruktureringen af klasserne samt QuadTree implementationen.
Til sidst aftalte vi, at alle skriver rapporten på Google Docs med deres farver som vi så fletter sammen på mandag.
Peter vil også gerne kigge på zoom ud, og i det hele taget zoom ind og ud via. scroll.
Næste mødedag: Mandag d. 31.03.2014 kl. 11.

Logbog d.31-03-2014 - Forfatter: Adam

Vi mødtes Adam, Peter og Kristian. Johan var syg. Vi gik i fuld gang med rapporten, samt bug fixes. Et par bugs dukkede op efter vi fik ord på vores kode. De blev udrettet løbende. Alle arbejde på rapporten. Der er stadig nogle få bugs, men overordnet går det godt frem. Paning er blevet implementeret og fungerer. Vi færdiggøre rapporten i morgen.

Logbog d.01-04-2014 - Forfatter: Peter

Peter, Kristian og Adam mødtes efter Diskret Matematik kl. 14 og skrev første del af Rapporten færdig. Johan var syg, men havde skrevet lidt til rapporten samt lidt javadoc. Små justeringer på koden blev lavet før aflevering. Rapporten blev rettet, formateret og færdiggjort. Planen er nu, at Adam afleverer i morgen tidligt.