Førsteårsprojekt — endelig aflevering
Tirsdag 29. april 2008

2008-04-29

1 Formål med projektet

Efter førsteårsprojektet skal I kunne *forklare* projektet og den realiserede løsning; *evaluere* den realiserede løsning; *beskrive* og vurdere den gennemførte proces; samt *reflektere* over produkt og proces med henblik på at foreslå konstruktive forbedringer.

Konkret handler projektet om brugergrænseflader (design og konstruktion), om algoritmik (ruteplanlægning), og om håndtering af et stort datasæt som kræver opmærksomhed på optimeringer og på hjælpedatastrukturer.

Hensigten med de to delafleveringer (jævnfør *Opgaver uge 7* og *Opgaver uge 10*) var at give en solid fornemmelse for datasættets indhold og størrelse, for konstruktion af en brugergrænseflade til at udforske datasættet, og for at bruge algoritmik til at besvare spørgsmål om det.

Målet med den endelige aflevering er at I kan lave et afrundet og veldokumenteret stykke software (produktet), hvor gruppen *beslutter* hvad produktet skal kunne, og beskriver og begrunder dette; *designer* og *implementerer* den besluttede funktionalitet, beskriver og begrunder de trufne designvalg og implementationsteknikker, herunder valg af algoritmer og datastrukturer; *tester* produktets funktionalitet og *vurderer* i hvilken grad det opfylder de stillede krav; *beskriver* og *vurderer* udviklingsprocessen; og *reflekterer* over produkt og proces, og mulige forbedringer.

2 Krav til det endelige produkt

Det endelige system skal opfylde de minimumskrav som de to delafleveringer stillede, dvs. tegne vejdatasættet så det passer til vinduets aktuelle størrelse og zoome ind på det (uge 7), og finde og tegne korteste vej mellem to punkter der afmærkes med musen på kortet (uge 10).

I det endelige produkt skal I desuden:

- designe en sammenhængende brugergrænseflade som skal indeholde kortet, men formentlig også andre komponenter;
- gennemtænke hvordan musebevægelser, museknapper, tastatur og brugergrænsefladekomponenter benyttes på en konsekvent og brugervenlig måde;
- beslutte hvilken yderligere funktionalitet jeres produkt skal have og implementere denne — se idéliste nedenfor;
- gøre brugergrænsefladen hurtig nok til at den er behagelig at bruge, også når man arbejder på hele Danmarks-datasættet. Opstartstiden — omkring 5 sekunder for indlæsning af Sjællands-datasættet og 20 sekunder for hele Danmarks-datasættet — er det svært at gøre noget ved så længe dataene er på tekstform. Men efter opstarten skal brugergrænsefladen gerne reagere kvikt.

Noget af det I har programmeret i delafleveringerne kan I uden videre bruge i den endelige aflevering, men der er sikkert også dele der kræver at blive gennemtænkt igen og udskiftet. Tilsvarende kan meget af det I allerede har skrevet i arbejdsblade formentlig indgå i den endelige rapport, men i omarbejdet og gennemskrevet form.

3 Yderligere ideer til brugergrænseflade og funktionalitet

1. Gør det muligt at *zoome* både ind og ud og at *scrolle* mod øst, syd, vest eller nord.
2. Lav brugergrænsefladen den hele tiden *viser navnet* (og måske postnummer) på den vejstump der er nærmest på musen når den er inde på kortet. Man kan bruge en hjælpedatastruktur til at gøre dette effektivt, fx quadtree (se BADS-bogen afsit 12.3.1) eller en noget enklere inddeling af landet i lodrette (eller vandrette) striber af 100 meters bredde, og med et array af hashset af de kanter der krydser hver stribe.

3. Lav brugergrænsefladen så man kan *indtaste* start- og slut-punkt i en rute som vejnavn (i en tekstboks) og evt. husnummer, i stedet for at bruge musen til at angive dem.

4. Lav ovenstående så der dynamisk vises alle veje der begynder med den tegnfølge man har indtastet i tekstboksen. Hvis man har skrevet "hy" i tekstboksen får man foreslået "Hybenvænget", "Hyldegårds Tværvej", "Hyldegårdsvej" og så videre. Overvej datastrukturer som kan gøre dette effektivt.

5. Lav brugergrænsefladen så man kan få en *rutebeskrivelse*, fx i et separat vindue, med vejnavne og hvor langt man skal køre før man skal dreje, eller før den aktuelle vej skifter navn. Datasættet har åbenbart ikke oplysninger om man skal svinge til højre eller venstre, så det kan I udelade (eller forsøge at beregne det).

6. Lav brugergrænsefladen så man kan vælge mellem at få lagt en *bilrute* og en *cykel/gang-rute*. En bilrute skal være den *hurtigste* vej og må gerne bruge motorveje og motortrafikveje men ikke stier. En cykel/gang-rute skal være den *korteste* vej og må gerne bruge stier men ikke motorveje eller motortrafikveje.

7. Lav brugergrænsefladen så man med musen kan vælge en *sekvens* af punkter og få beregnet og tegnet den korteste vej som besøger punkterne *i den rækkefølge der blev angivet med musen*.

8. Lav brugergrænsefladen så man kan vælge at kun bestemte typer veje (fx motorveje, primærruter, stier) skal vises. Eller kun de veje som er relevante for biler, eller kun dem der er relevante for cyklister. Eller kun bestemte postnumre.

9. Lav brugergrænsefladen så man kan sætte et startpunkt (fx vist som en trekant) med musen, og få tegnet korteste/hurtigste vej til slutpunktet — der hvor musen er.

10. Lav foregående sådan at man hele tiden får tegnet den korteste/hurtigste vej fra startpunktet til punktet nærmest musen, mens man trækker musen rundt. På grund af den måde Dijkstras algoritme fungerer (ved gradvis at finde korteste afstand fra start til alle andre punkter) burde dette kunne gøres ret effektivt, forudsat man kan regne ruten uden at gentegne hele vejnettet. Det er mindre klart at A* algoritmen egner sig.

11. Få ruteplanlæggeren til at overholde svingrestriktioner. Dette kræver forståelse (og indlæsning) af filen `turn.txt` som findes i det fulde Krak-datasæt.

12. Lav brugergrænsefladen så kortet også viser vejnavne på vejene når der er zoomet tilstrækkelig langt ind.

13. Undersøg forskellige heuristikker ("tommelfingerregler") som måske kan gøre ruteplanlægning hurtigere. Fx at udnytte vejklassifikationen sådan at Dijkstra eller A* over længere afstande ignorerer alt andet end motorveje, motortrafikveje og primærruter. Det Det er tydeligt at Google Maps bruger denne type heuristikker og ind i mellem producerer nogle meget sære ruter.

14. Lav brugergrænsefladen så man med musen kan vælge en sekvens af punkter og få beregnet og tegnet den korteste vej som besøger alle disse punkter *i en eller anden rækkefølge*, ikke nødvendig i den rækkefølge der blev angivet med musen. NB: Dette er en version af Traveling Salesman Problem (TSP) og der kendes ingen garanteret effektive løsninger på dette problem. Men med branch and bound (BADS-bogen afsnit 13.5.2) og andre teknikker kan man ofte alligevel finde løsninger hurtigt. Dette er ret udfordrende.

15. Studér artiklen om *Highway hierarchies*-algoritmen af Sanders og Schultes 2005 (se kursushjemmesiden) og overvej om den er nyttig til at finde ruter inden for Danmark. Dette er ret udfordrende.

Ovenstående punkter er forslag. Det er ikke nødvendigt (eller muligt) at lave dem alle sammen, og numrene betyder ikke at forslagene skal laves i en bestemt rækkefølge. Endelig er listen ikke udtømmende: I opfordres til at finde på yderligere ideer, eller varianter af ovenstående.

Som sagt i afsnit 1 er det en væsentlig del af projektarbejdets afsluttende fase at gruppen diskuterer hvilken funktionalitet I vil implementere og i hvilken rækkefølge, og at rapporten beskriver og begrunder disse valg samt den grad af opfyldelse I har nået. Med andre ord, at I styrer mod et mål (også selv om I ikke når det) i stedet for tilfældigt at zigzagge mellem forskellige muligheder.

4 Regler mv

- Opgaven udleveres tirsdag 29. april og projektrapporten skal afleveres i **tre eksemplarer** senest **onsdag 21. maj kl 15:00 i studieadministrationen**.
- Projektet udføres i de dannede grupper.
- Hver gruppe skal registreres i projektbasen senest tirsdag 5. maj kl 1200. I projektbasen skal I registrere projektet med projekttitlen "FAAP projekt", hvem der indgår i gruppen, og Claus Brabrand og Peter Sestoft som vejledere. I behøver ikke udfylde problemformulering, forudsætninger og de andre felter.
- Systemet skal programmeres i Java.
- Projektrapporten (eksklusive bilag) bør ikke overstige 40 sider.
- Der vil være instruktor- eller vejlederhjælp til projektet tirsdag 0900-1200 og fredag 1000-1200 fra og med fredag 2/5 til og med fredag 16/5.
- Brug meget gerne kursets nyhedsgruppe `it-c.courses.BFAAP` til at stille spørgsmål af almen interesse, fx om forståelsen af denne opgavetekst, eller valg af algoritme eller lignende.

4.1 Vurdering af projektarbejdet

Hovedvægten i vurderingen lægges på projektrapportens beskrivelse af *produkt* og *proces*. Se også kravene på kursets hjemmeside hvor I finder:

- Kursets *Læringsmål og Eksamenskrav*.
- Dokumentet *FAAP proces*, udsendt 17. april, hvor vægtningen 60% produkt og 40% proces er angivet. Det er altså vigtigt at I dokumenterer gennem dagbog og arbejdsblade at I arbejder på en struktureret måde, tager beslutninger, lægger planer, forsøge at følge dem, og noterer resultaterne af eksperimenter.

4.2 Projektrapportens form

- Forside med projekttitel, gruppefarve, gruppemedlemmernes navn og ITU-emailadresse, afleveringsdato, vejledere, kursusnavn ("Førsteårprojekt, bachelor i softwareudvikling, IT-Universitetet").
- Forord (hvor, hvornår, hvorfor)
- Baggrund og problemstilling, datasættet, jeres krav til det resulterende programs funktionalitet.
- Begrundede designbeslutninger vedrørende brugergrænsefladens udseende og interaktion, herunder brugen af mus, museknapper, tastatur, brugergrænsefladekomponenter.
- Problemanalyse, tekniske overvejelser om hvordan brugergrænsefladen opbygges, hvilke algoritmer og datastrukturer benyttes. Også gerne beskrivelser af eksperimenter med hvilke datastrukturer og programstrukturer der i praksis er bedst til at løse et givet delproblem.
- Kort teknisk orienteret beskrivelse af programmets interne strukturer (klasser, objektrelationer), med UML-diagrammer.
- Systematisk afprøvning af det resulterende system, og vurdering af i hvilken grad systemet opfylder sit formål. Afprøvningen kan vedrøre brugergrænsefladens funktionalitet og unit test af klasser, fx jeres egen prioritetskø hvis I har sådan en.
- Brugerorienteret beskrivelse af det resulterende system (kort, fx skærmbillede med alle brugeroperation angivet).
- Produktkonklusion: i hvilken grad produktet opfylder de opstillede krav, herunder en mangelliste; og hvilke nye ideer til funktionalitet, design og implementering er dukket op men ikke forsøgt.
- Gruppenormer, altså gruppekonstitution.

- Kondenseret dagbog.
- Relevante arbejdsblade for del-aspekter af jeres projekt.
- Procesbeskrivelse med refleksion; hvad kunne være blevet gjort bedre.

5 Gode råd

- Husk at I har allerede arbejdet længe med dette projekt og derfor nemt kan glemme hvor lidt udenforstående ved om datasættene, den givne grafrepræsentation, problemstillingerne ved visualisering og rutefinding, osv. Husk derfor at forklare disse ting i rapporten i stedet for bare at tale om UTM32, KrakNoder osv, men gør det kort og præcist og brug tegninger, diagrammer og tabeller sammen med teksten, så kan I spare læseren for store mængder ord.
- Husk at det meget nemt bliver et større problem at rapporten er for lang end at den er for kort. Lad være med at "løse" dette problem ved at bruge en mikroskopisk font, skrive helt ud til margen af papiret, eller lægge vigtige afsnit som bilag.
- Husk helt generelt at bruge underafsnit, afsnitsnumre, tegninger, tabeller, figurnumre, krydshenvisninger, punktlister, grafer, litteraturliste, osv som det er beskrevet i forelæsningen om tekstbehandling.
- Husk at brugervenlighed er mange ting: skal det være nemt at lære for en nybegynder, eller hurtigt at bruge for en ekspert? Forklar hvad I finder vigtigst og hvordan jeres design understøtter jeres valg, og evaluer om det fungerer så godt som I havde tænkt jer.
- Husk at komponenterne i en Java-brugergrænseflade bør opbygges én gang for alle, så man fx ikke laver nye JButton-objekter i en lind strøm mens programmet kører. Det fører til unødigt pladsforbrug i computeren og til forvirring hos udviklerne.
- Datasættet indeholder kun postnumre, ikke bynavne. En fortegnelse over Danmarks postnumre og bynavne kan findes på www.postdanmark.dk under *Finde et postnummer* og *Kort over postnumre*. Denne kunne man gemme som tekstfil og indlæse til programmet, og derved få mulighed for at vise eller søge på bynavne.
- En gruppe gjorde opmærksom på at Javas klassebibliotek (pakke java.awt.geom) har nyttige rutiner til fx at finde afstanden fra et punkt til et liniestykke osv, så man ikke selv behøver opfinde og implementere dem.
- Husk at bruge hvad I har lært i BADS-kurset når I skal argumentere for at én datastruktur eller algoritme er bedre end en anden til netop jeres formål. Empiriske observationer om datasættet kan være nyttige — det faktum at Sjællands-datasættet har 171912 knuder og 209402 kanter viser fx at langt de fleste knuder kun har to tidstødende kanter. Med andre ord er der ikke brug for en fiks datastruktur til at holde styr på en knudes kanter, for der er sjældent ret mange af dem.
- Hvis muligt, så brug tidsmålinger (med en Timer-klasse eller lignende) når I skal argumentere for at én løsning er bedre end en anden. Eller argumentere for at den simple løsning I har valgt er god nok i praksis.
- Sørg for at alle i gruppen har ejerskab til programmet, dvs forstår hvad der foregår og tør ændre i programkoden. En måde at opnå dette er at forklare dele af programmet for hinanden. Hvis dette er meget svært, er det måske fordi programmet ikke er særlig godt skrevet . . . og generelt er det en af de bedste måder at finde fejl og lave forbedringer på.