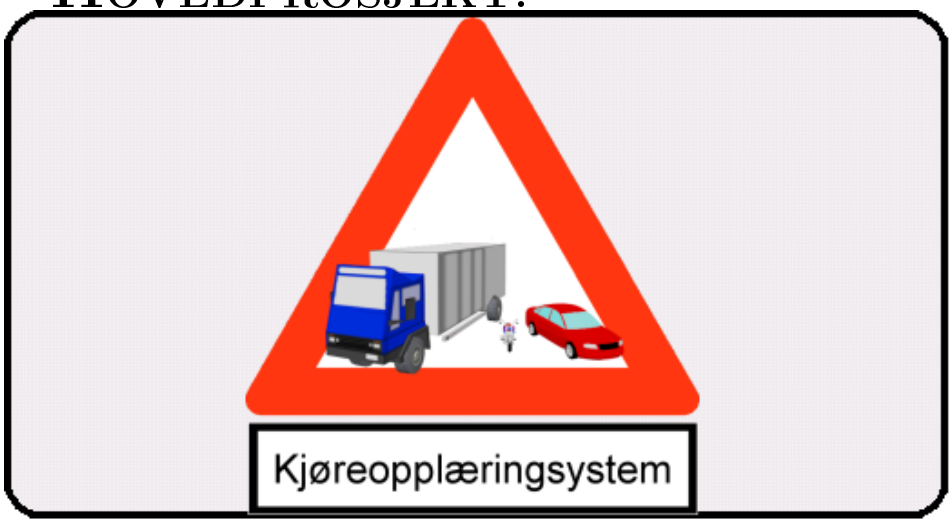
Hovedprosjekt:



Forfattere:

- Sverre Bakke
- Kenneth Rinnan
- Jonas Strømstad

Dato: 22. mai 2006

Sammendrag

Tittel:	Kjøreopplæringssystem (Kos)
Dato:	22. mai 2006
Forfattere:	Sverre Bakke Kenneth Rinnan Jonas Strømstad
Veileder:	Frode Haug
Oppdragsgiver:	Høgskolen i Gjøvik v/Frode Haug
Kontaktperson:	Jonas Strømstad
Nøkkelord:	Trafikkskole, Kos, Teoriprøve, 3D-generator
Antall sider:	125
Antall vedlegg:	10
Tilgjengelighet:	Åpen
Sammendrag:	Prosjektet er gjort på oppdrag fra Høgskolen i Gjøvik og i samarbeid med Thomtes trafikkskole. Målet med prosjektet var å få laget en applikasjon som kan forbedre og forenkle undervisningen ved trafikkskolene. Kos er en applikasjon hvor brukeren får muligheten til å generere trafikale situasjoner i full tredimensjonal grafikk. Disse situasjonene kan lagres og brukes i undervisningen, eller man kan faktisk også bruke Kos til å lage situasjoner der og da. I tillegg følger det med en spørsmålsgenerator hvor man kan lage teorispørsmål på basis av trafikale situasjoner man har opprettet fra før, eller på bakgrunn av digitalbilder importert inn i programmet.

Hovedprosjektrapport
Kos
Kjøreopplæringssystem

Jonas Strømstad, Sverre Bakke, Kenneth Rinnan

22. mai 2006

Forord

Kos er en applikasjon utviklet for trafikkskoler i Norge. Konseptet er basert på en idé unnfanget av Frode Haug. Frode Haug er høgskolelektor ved Høgskolen i Gjøvik hvor han underviser i objektorientert programmering og algoritmiske metoder. Han har på vegne av Høgskolen i Gjøvik fylt rollen som oppdragsgiver på dette prosjektet. Den orginale idéen var egentlig kun en spørsmålsgenerator med en tilhørende 3D-generator. Hele konseptet ble endret på mange måter etter at Thomtes Trafikkskole ble trukket inn som samarbeidspartner.

Kos fremstår i dag som et verktøy utviklet for å støtte opp under undervisningen for aspirerende sjåfører. I tillegg kan Kos brukes til å lage spørsmål basert på det undervisningsopplegget som er laget.

Produktet er utarbeidet av tre studenter ved Høgskolen i Gjøvik som en avsluttende oppgave på Bachelorgraden i informatikk. Prosjektet har vært veldig lærerikt og morsomt. Det er mye vi kan nå som vi ikke kunne før vi begynte, men vi kunne ikke ha gjennomført dette prosjektet uten hjelp fra utsiden. Derfor vil vi gjerne få takke noen få utvalgte. Føler du deg glemt, så si ifra.

Takk rettes til:

- Frode Haug: For idéen og veiledning.
- Ann Kristin Thomte: For faglige innspill, nye idéer og hjelp med testing.
- Øivind Kolloen: For lån av kode fra tidligere kurs og Java hjelp.
- Anders Oulie: For matematisk assistanse.
- Kenneth Russell (Uvikler av Jogl): For hjelp med Jogl programmering.
- Mads Byfuglien: For testing.
- Jon Langseth: For å ha reddet dagen, en dag det var skikkelig mørkt.

Jonas Strømstad
Prosjektleder

Sverre Bakke

Jonas Strømstad

Kenneth Rinnan

Innhold

Kapittel 1

Innledning

1.1 Problemområde

På de fleste trafikkskoler i dag benytter man seg av tegninger på tavler, magnetbaserte biler på brett eller ferdig foiler med tegninger på når man ønsker å presentere en trafikksituasjon. Disse metodene er i dag litt gammeldagse, men siden alternativer innen høyteknologien ikke finnes er det dette man er nødt til å benytte seg av. De fleste som deltar i et undervisningsopplegg ved en trafikkskole tar i løpet av kurset en form for tentamenstest med spørsmål som ligner de man får på den offisielle teoriprøven. Disse prøvene utarbeides enten av trafikkskolen selv eller av egne selskaper som har spesialisert seg på å lage slike prøver. Idag må trafikkskolene benytte seg av standariserte tekst og bildebehandlere hvis de ønsker å lage egne prøver. Kos gjør denne jobben mye lettere og med mye større fleksibilitet.

1.1.1 Oppgavebeskrivelse

Kos er en applikasjon for å:

1. Generere trafikksituasjoner for lokal presentasjon.
2. Generere teorispørsmål/prøver for visning på lokal PC.
3. Teste sine trafikkunskaper på de spørsmål som er generert.

Applikasjonen skal på den måten kunne erstatte dagens ordning som går ut på å tegne for hånd på tavle eller papir og vil være med på å både effektivisere undervisningen og å gjøre tegningene/bildene mer oversiktlige.

Generere trafikksituasjoner for lokal presentasjon.

Det skal benyttes 3D-grafikk for generering og presentasjon av situasjonene slik at brukeren selv kan velge plasseringen av og innsynsvinkel til trafikale objekter (kjøretøy, bygninger, o.l.) samt innsynsvinkel til situasjonen i sin helhet. Brukergrensesnittet til generatoren må være så enkelt som mulig slik at sluttbrukere ikke trenger lang opplæring for å bruke applikasjonen og derfor skal det brukes et dra og slipp ("drag and drop") grensesnitt hvor man kan dra objekter fra en liste og inn på situasjonen. Alle trafikale objekter er organisert i en liste ut i fra en logisk kategori-struktur faglig sett. Det skal være mulig å opprette/redigerer flere scenarier basert på forskjellige trafikk-lokasjoner samtidig, og for hvert scenario ha undersituasjoner som bygger på en og samme lokasjon, men med varierte trafikale objekter. Når man er fornøyd med situasjonen skal man kunne fryse den slik at ytterligere endringer ikke påvirker den fryste situasjonen, og situasjonen havner i listen over situasjoner for gjeldende scenario. Man skal lett komme inn i fremvisningsmodus hvor man enkelt kan bla gjennom alle scenarier og undersituasjoner som er åpnet, men med kun det mest nødvendige av brukergrensesnitt synlig. Mens man driver med fremvisning skal det være mulig å kunne endre på situasjonen hvis man der og da ønsker å vise et annet eksempel. Til hver situasjon skal det være mulig å legge ved et notat eller spørsmål.

Generere teorispørsmål/prøver for visning på lokal PC.

Det skal også være mulig å skifte enkelt til spørsmålsmodus for aktiv/gjeldende undersituasjon. Da vil brukergrensesnitt for å legget til spørsmålstekst, kategori og svaralternativer bli synlig sammen med den genererte undersituasjonen. Når man lagrer vil spørsmålet, sammen med undersituasjonen, lagres i den lokale spørsmålsdatabasen. Prøvene blir generert automatisk ved prøvestart utifra de kategoriene som prøven skal dekke og det antall spørsmål som er valgt. Det er viktig at programmet gir tilbakemelding ved å fortelle eleven hvor han/hun har svart feil eller riktig og som oppsummerer hele prøven.

1.1.2 Avgrensninger

Applikasjonen utvikles i utgangspunktet for å kjøre på en enkelt PC. Støtte for distribuering av spørsmål/prøver over internett eller lokalt nettverk o.l. er blitt vurdert, men har blitt bortvalgt ettersom prosjektlengden er begrenset og det ikke er vanlig at trafikkskoler har flere maskiner. Dette er selvfølgelig noe som kan implementeres i produktet etter behov. Selv om situasjonene genereres i 3D er det ikke en trafikk-simulator. Verken animasjoner av objekter eller forhold som f.eks. vær, tid på døgnet (dvs. lysforhold o.l.), årstider, eller tyngdekraft vil bli implementert. Det vil i utgangspunktet heller ikke lages støtte for spesielle elementer i terrenget som f.eks. fjell og tunneler, men vi er åpne for å implementere dette ved en senere anledning eller hvis det blir tid til overs. De trafikale objektene som er mulig å bruke vil også begrense seg til generelle objekter og ikke spesielle merker. Eksempel på dette er at vi har enkle biler og ikke BMW, Audi, Volvo, o.l. Det vil heller ikke bli fokusert på å legge inn masse reelle spørsmål og prøver, men heller legge inn funksjonalitet og objekter som kan brukes til å generere dette med det ferdige produktet. Andre mulige implementeringer senere, men som ikke er en del av dette prosjektet er støtte for audio/høytlesning av spørsmål og svaralternativer samt animasjoner/videofiler istedenfor bilder. Det er alikevel mulighet for at slike multimedia-utvidelser etter prosjektets slutt hvis behovet er til stede.

1.2 Målgruppe

1.2.1 Målgruppe for Kos

Målgruppen er trafikkskolelærere som ønsker å nå frem til sine elever på en ny måte. Et suksessfull programvare av denne typen kan fort bli en del av hverdagen på norske trafikkskoler. Elevene ved trafikkskolene er også en målgruppe for den delen av programmet hvor man besvarer tester.

1.2.2 Målgruppe for rapporten

Målgruppen for denne rapporten er firedelt. Først og mest åpenlyst er de personene som skal vurdere oppgaven. Dette er veilederen og sensor. Det må være lov å forutsette at disse innehar en hvis kompetanse innen fagfeltet og at terminologibruken i rapporten kan legges deretter. Den neste målgruppen som kan ha interesse av rapporten er samarbeidpartneren vår. Samarbeidspartneren innehar ikke den samme faglige innsikten, og de mest faglige delene av rapporten

er heller ikke beregnet på henne. Derimot kan hun ha interesse av å lese våre vurderinger av oppgaven som en helthet og konklusjonene våre. Den tredje gruppen interessenter er andre studenter. Dette gjelder både de som leverer hovedprosjekt samtidig med oss, hvis disse ønsker å se hva vi har gjort, og fremtidige studenter som jobber med hovedprosjekt. I tillegg vil det kanskje være tenkelig å benytte seg av rapporten i en jobbsøkningsprosess. Da vil rapporten bli presentert ovenfor arbeidsgiver som den fjerde delen av målgruppen for dette dokumentet.

1.3 Formål

1.3.1 Hovedmål

Det skal utvikles en programvare som kan benyttes i store deler av den trafikkmessige undervisningen ved norske trafikkskoler. Denne skal kunne brukes til å generere trafikksituasjoner i full tredimensjonal (3D) grafikk som senere kan presenteres via programvaren i fullskjerm i klasserom (via projektor e.l.) eller som grunnlag for å lage teorispørsmål via den innebygde spørsmålsgeneratoren. Disse kan senere settes sammen til tester med spørsmål fra ulike kategorier. Det skal også være mulig å endre på trafikksituasjonene underveis i en presentasjon ved å legge til, fjerne eller flytte objekter i bildet.

1.3.2 Effektmål

Denne programvaren skal endre måten trafikkskolene bruker datamaskiner i sin undervisning, samtidig som det skal gi dem et kraftig verktøy som øker kvaliteten på undervisningen og gjør den mer effektiv. Prosjektet skal også fungere som et utstillingsvindu for deltagerne og være noe de skal være stolte av å ha vært med på, samtidig som det gir fremtidige arbeidsgivere informasjon om prosjektdeltagernes kunnskaper og ferdigheter.

1.3.3 Prosessmål

Prosjektet skal gi de som deltar erfaring med å jobbe på et større prosjekt. De skal få erfaringer og kunnskaper om hvordan man jobber et en spesifikt gitt systemutviklingsmodell. På denne måten skal prosjektdeltagerne bli bedre skikket til å møte utfordringene som finnes i arbeidslivet etter endt utdanning.

Den viktigste utfordringen, med rot i det virkelige arbeidslivet, er forholdet mellom utviklerne og en reell oppdragsgiver. Vi får erfaring med hvordan det er å kommunisere og samarbeide med en annen part. Deltagerne skal få erfaring med å måtte tilegne seg fagkunnskap underveis i prosjektet. Gjennom prosjektet skal deltagerne få gode kunnskaper og evner innen Java-programmering, både innen enkle input/output operasjoner, men også mer avansert ved den tredimensjonale modulen.

1.4 Egen bakgrunn og kompetanse

Studentene som har jobbet med dette prosjektet har alle gjennomført alle nødvendige fag for studieretningen programutvikling. I tillegg har alle tre tatt ekstra fag med litt forskjellige fordypninger. To av deltagerne har også tatt fordypningsfag innen drift, mens den tredje har tatt ytterligere fag innen systemutvikling. Innenfor programmering har alle prosjektdeltagerne blandet seg med de dyktigste på skolen. Dette gjorde at alle var trygge på at vi ville få til det vi forsøkte. Innen OpenGL programmering er det kun en av prosjektdeltagerne som hadde tidligere erfaring og denne var også ganske overfladisk. For alle tre er dette tredje året vi driver med objektorientert programmering og for to av oss tredje året vi driver med programmering i det hele tatt. Applikasjonsutvikling i Java har vi alle under et års erfaring med, men med opplæring i objektorientert programmering i C++ er ikke overgangen så stor.

1.5 Rammer

1.5.1 Praktiske rammer

Prosjektet skal utvikles i samarbeid med Thomtes Trafikkskole representert ved Ann Kristin Thomte. Oppdragsgiveren er offisielt Høgskolen i Gjøvik representert ved Frode Haug, som også fungerer som veileder. Gruppedeltagerene stiller selv med teknisk utstyr til prosjektet. Tidsrammene på prosjektet er lagt i henhold til de frister som er satt fra skolens side. Absolutt sluttdato for prosjektet er 24/5-06.

1.5.2 Teknologiske rammer

Gjennom kontakt med Thomtes trafikkskole har vi vurdert statusen på det utstyret som finnes hos trafikkskolene til å være av typen enkeltstående maskiner som ofte vil ha en tilgang til Internet, uten at det er en forutsetning. Basert på deltagernes forkunnskaper og vurderinger har det allerede blitt avgjort at prosjektet i all hovedsak skal utvikles i Java 1.5. 3D-modulen skal basere seg på en OpenGL-implementasjon for Java kalt JOGL. Plattformen det skal utvikles for er i utgangspunktet Windows, men hvis mulig så skal plattformuavhengighet tilstrebes. Dette gjøres mulig av Javas portabilitet til alle systemer som har en JavaVM tilgjengelig. Produktet må kunne kjøre sømløst på en vanlig hjemme-PC uten ekstraordinære krav til maskinkraft eller grafikkprosesseringsenheten (skjermkort).

1.5.3 Prosjektrammer

Prosjektet er inndelt i inkrementer og er utviklet etter en avart av inkrementell utvikling. Oppdelingen av inkrementene står slik den er, men utviklingen innad hvert inkrement har utviklingen foregått evolusjonært. Endringer har kommet til ettersom tiden har gått og den overordnede tidsplanen ble etterhvert avslørt som veldig optimistisk. Prosjektet er gjennomført våren 2006 ved Høgskolen i Gjøvik. Prosjektstart var tidlig januar, og siste frist for innlevering av rapport og produkt var 24/5-2006. Å ha en utviklingsmodell av den typen vi valgte egner seg lite hvis man må i mål med hele produktet, men vi har via dette valget sikret oss at vi blir skikkelig ferdige med det vi har gjort.

1.6 Øvrige roller

Oppdragsgiver og veileder:	Frode Haug Epost: frode@haugianerne.no Telefon: 61135191

Vi har innledet et samarbeid med Thomtes Trafikkskole for å få faglig innspill underveis på prosjektet. Thomtes skal under prosessen delta på egne statusmøter som representant for målgruppen til produktet, men er fritatt for alt ansvar i forbindelse med prosjektet og det endelige produktet.

Thomtes Trafikkskole: Ann Kristin Thomte
Epost: firmapost@thomte.no
Nettside: http://www.thomte.no
Telefon: 61176079 / 91833454

1.7 Organisering av rapporten

Rapporten er oppdelt i seks deler:

1. Innledning
2. Kravspesifikasjon
3. Design
4. Implementasjon
5. Testing/kvalitetssikring
6. Oppsummering

Del 2-4 er igjen oppdelt i underseksjoner for hvert inkrement. Disse delene beskriver hva løsningen skal gjøre og hvordan den skal gjøre det.
Rapporten er i sin helhet skrevet på norsk men har innslag av mye engelske fagspesifikke terminologi. Terminologien er forsøkt holdt på et slik nivå at en person som kan objektorientert programmering og design, med overfladisk kjennskap til Java skal kunne forstå alt som presenteres. Alle ord, utrykk og forkortelser som ikke er en naturlig del av terminologien, eller som trenger utdypning er forklart under definisjoner (A).
Rapporten er skrevet i LaTeX og formatert etter rapportstandarden der. LaTeX støtter en mengde standarder for vitenskaplige dokumenter og inneholder kraftige løsninger for å holde orden på innholdfortegnelser, referanser og punktmerking. Når det gjelder skrifttyper har vi ikke valgt noe annet enn å følge standard innstillingene i LaTeX som egner seg bra for trykk.

Kapittel 2

Kravspesifikasjon

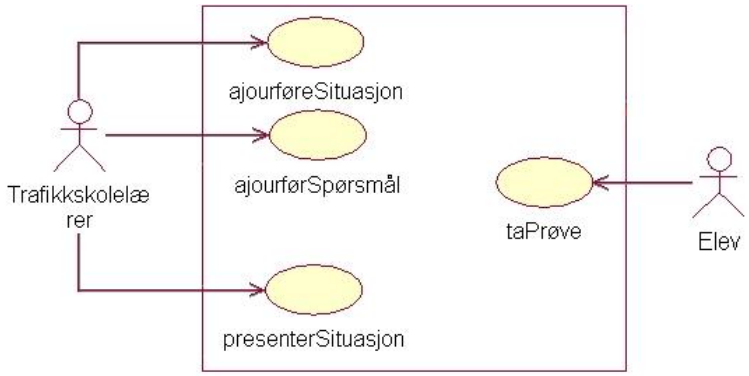
2.1 Beskrivelse

Kos skal være et digitalt verktøy for å erstatte mange av dagens manuelle læringsmetoder ved forbredelse til førerprøven. Det skal kunne benyttes i forbindelse med tavleundervisning fra trafikkskolelærere og ved egenstudier. Applikasjonen vil på den måten forhåpentligvis være med på å både effektivisere undervisningen og å gjøre tegningene/bildene mer oversiktlige. Systemet skal utvikles for å kunne:

- Generere trafikksituasjoner for lokal presentasjon
- Generere teorispørsmål for lokale teoriprøver

Se pkt. 1.1.1 for mer utfyllende oppgavebeskrivelse.

2.1.1 Use-case diagram



Beskrivelse av use-case diagram

Use-case diagrammet (over) beskriver de overordnede funksjonene aktørene som skal benytte KOS ønsker å få utført. Modellen er utformet med den overordnede oppdelingen av prosjektet i tankene. Hvert av de use-case som er tegnet inn i diagrammet vil etterhvert bli oppdelt i flere mer detaljerte use-case i kravspesifikasjonsdokumentetene til de enkelte inkrementene. Senere utgaver av diagrammet vil bli lagt ved dokumentene tilknyttet de enkelte inkrementene.

Inkrement 1 (- 3D-generator) vil sammen med inkrement 2 (-Menysystem for 3D-generator) utgjøre det brukeren av systemet vil få for å generere 3D-situasjoner, sammen med inkrement 3 (- Presentasjon) utgjøre det brukeren vil få for å presentere situasjonene, og sammen med inkrement 4 (- Spørsmålsgenerator) utgjøre det brukeren vil få for å generere spørsmål. Det er derfor svært viktig at inkrement 1 (-3D-generator) blir utviklet tilfredsstillende etter kravene oppgitt fra oppdragsgiver da senere inkrementer ikke kan utføres uten dette.

Use-casene er tilknyttet følgende inkrementer:

- ajourførSituasjon er tilknyttet inkrement 1 og inkrement 2
- presenterSituasjon er tilknyttet inkrement 3
- ajourførSpørsmål er tilnyttet inkrement 4
- taPrøve er tilknyttet inkrement 5

Use-case	ajourførSituasjon
Formål	Å kunne generere en trafikal hendelse i 3D for bruk i presentasjoner og teorispørsmål.
Aktører	Trafikkskolelærer
Beskrivelse	Det skal benyttes 3D-grafikk for generering av situasjonene slik at brukeren selv kan velge plasseringen av og innsynsvinkel til trafikale objekter (kjøretøy, bygninger, o.l.) samt innsynsvinkel til situasjonen i sin helhet. En situasjon skal kunne basere seg på en ferdigstilt lokasjon fra applikasjonens eget bibliotek eller fra en brukervalg bildefil. Trafikkskolelærer skal så kunne fra et bibliotek over trafikale objekter kunne dra ønsket objekt dit han/hun måtte ønske i situasjonen. Det skal også være mulig å markere et tidligere opprettet objekt og utføre diverse operasjoner på det (eksempelvis fjerne, rotere, legge til tekst, o.l.).

Use-case	presenterSituasjon
Formål	Å kunne presentere en trafikal hendelse for trafikkskoleelevene.
Aktører	Trafikkskolelærer, Trafikkskoleelev
Beskrivelse	En trafikkskolelærer skal kunne presentere en generert situasjon for eleven. Ved presentasjon skal det ikke være mulig å utføre endringer på situasjonen. Det er her viktig å minimalisere distraksjoner i brukergrensesnittet for å trekke fokus mot selve situasjonen.

Use-case	ajourførSpørsmål
Formål	Å kunne opprette et spørsmålsbibliotek for bruk ved teoriprøver.
Aktører	Trafikkskolelærer
Beskrivelse	Det skal være mulig å ajourføre spørsmål mot et lokalt bibliotek. Man skal kunne redigere eksisterende spørsmål, eller opprette et nytt spørsmål basert enten på en generert situasjon eller et stillbilde fra fil. Et spørsmål består av et bilde som beskriver spørsmålshendelsen, en spørsmåltekst, og alternativer med flervalg. Trafikkskolelæreren må også angi hvilke alternativer som er riktige og gale. I tillegg tilhører et spørsmål alltid en spørsmålskategori.

Use-case	taPrøve
Formål	Å kunne teste sine kunnskaper ved egenstudier.
Aktører	Trafikkskoleelev
Beskrivelse	Trafikkskoleelever skal kunne ta prøver lokalt på trafikkskolens datamaskin og få en oversiktlig oppsummering og gjennomgang ved prøvens slutt. Prøvene blir automatisk generert med tilfeldige spørsmål fra spørsmålsbiblioteket trafikkskolelæreren har opprettet.

Aktører

Trafikkskolelærer
Ønsker å få et system som kan hjelpe med å effektivisere undervisningen. Ønsker å få økt kvaliteten på undervisningen gjennom å kunne presentere problemstillinger i trafikken på en bedre og mer presis måte.

Trafikkskoleelev
Vil kunne ta prøver for å teste og styrke sine egne kunnskaper om mulige trafikale hendelser som kan oppstå.

2.1.2 Forutsetninger og Avhengighetsforhold

Prosjektet er i aller høyeste grad avhengig av tilbakemeldinger fra fagpersonell fra trafikkskoler for å kunne få til et intuitivt system som passer trafikkskolenes behov og ønsker. Vi forutsetter at tidligere nevnte avtale med Thomtes trafikkskole vil gi oss den tilbakemelding vi trenger fra fagmiljøet.

2.2 Supplementær spesifikasjon

2.2.1 Funksjonalitet

Objekt-bibliotek

For at applikasjonen skal kunne være nyttig trengs det et sett trafikale objekter man man bruke for å bygge opp en situasjon. Objektene skal deles inn i logiske kategorier basert på forslag fra vår kontakt i fagmiljøet (Thomtes Trafikkskole).

Applikasjonen skal ved prosjektets slutt inneholde et grunnleggende bibliotek med slike objekter. Det skal blant annet inneholde:

- Kjøretøy
  - Bil
  - Lastebil
  - Buss
  - Motorsykkel
- Bygninger
  - Hus
  - Bensinstasjon
  - Høyhus/blokk
  - Villa
- Individer
  - Person
  - Lite dyr
  - Stort dyr
- Trafikkskilter
  - Fareskilt (kategori med alle fareskilt)
  - Forbudskilt (kategori med alle forbudskilt)
  - Vikepliktskilt (kategori med alle vikepliktskilt)
  - Opplysningsskilt (kategori med alle vikeplikt)
  - Påbudskilt (kategori med alle alle påbudskilt)
- Lys
  - Lyskryss-lys m/rødt, gult og grønt
- Terreng
  - Forskjellige trær

Lokasjons-bibliotek

Applikasjonen skal inneholde et lite sett ferdigutformede lokasjoner for bruk som bakgrunn på situasjoner. Disse skal bestå av de mest vanlige lokasjoner man støter på i trafikken. Eksempler på dette er:

- T-Kryss
- X-Kryss
- Rundkjøring
- Rett strekning

2.2.2 Brukervennlighet

Brukervennlighet er svært viktig i dette prosjektet da brukerne ikke nødvendigvis trenger å ha brukt datamaskiner tidligere. Selv om vi i de detaljerte kravspesifikasjonene for hvert enkelt inkrement vil utdype de brukevennelighetskrav som er aktuelle for det enkelte inkrementet så er det noen generelle krav vi kan følge for hele systemet for å lette brukerns anvendelse av systemet:

- Det skal benyttes mus og grafisk brukergrensesnitt i størst mulig grad fremfor tastatursnarveier.
- Funksjonalitet i brukergrensesnitt skal kun være tilgjengelig der funksjonaliteten har en hensikt. Dette for å eliminere forvirring over funksjonalitet uten virkning og derfor både øke ryddigheten i grensesnittet og brukerens forståelse av funksjonalitetens omfang.
- Hele brukergrensesnittet skal være internasjonalisert og det skal følge med språkfiler for Norsk og Engelsk.
- Ikoner skal være forklarende i forhold til hvilken funksjonalitet de representerer.
- Knapper som har funksjonalitet som jobber mot samme mål , skal være gruppert slik at brukeren opplever plasseringen som logisk og dermed også finner den igjen uten å måtte lete.
- Alle knapper skal ha hjelpetekst tilknyttet som vises hvis man plasserer muspekeren over knappen.
- Feilmeldinger som gis skal være så logiske og forklarende at brukeren lett skal kunne se hva han/hun har gjort galt.

2.2.3 Stabilitet

Ettersom målgruppen av applikasjonen har få eller ingen forutsetninger for å forstå vanlige feilmeldinger som kan oppstå ved Java-applikasjoner er det viktig for oss å eliminere disse eller forbigå de i stillhet fremfor å krasje hele applikasjonen.

2.2.4 Robusthet

Vi antar at målgruppen til systemet ikke har spesielt god kunnskap om datamaskiner og applikasjoner av denne typen og derfor vil kunne utføre uforutsette handlinger. Vi vil derfor prøve å eliminere enhver mulighet for å kunne gjøre feil ved å begrense brukerne sin tilgang til funksjonaliteten. Hvordan dette utføres vil variere fra inkrement til inkrement. Alikevel er det noen punkter som må tas hensyn til i hele applikasjonen:

- Hvis det mangler ikoner eller andre filer tilknyttet brukergrensenittet, så skal programmet fortsatt kunne fungere, og inneholde de samme funksjonalitetene/knappene. Dvs at hvis brukeren sletter alle ikonene så vil programmet kunne fungere som normalt, men vil mangle grafikken som er tilknyttet knappene.
- Hvis programmet ikke klarer å finne den gitte språkfilen, så skal programmet fungere som normalt, men brukergrensesnitt-teksten vil da erstattes med navnet på tekststrengen. Tekstnavn er i denne sammenheng definert som et nøkkel-ord som representerer tekststrengen internt uavhengig av valgt språk.

2.2.5 Ytelse

Vi vil utvikle for at applikasjonen skal kunne brukes på eldre datamaskiner ettersom vi antar at systemet skal brukes av privatpersoner og mindre bedrifter. Dette er en målgruppe som i motsetning til større bedrifter ikke bytter utstyr ofte og derfor antagligvis sitter på eldre datamaskiner. Vi tar utgangspunkt i at systemet ikke vil kjøre på datamaskiner eldre enn utstyr som var nytt for 4 år siden og vil derfor utvikle for maskiner av den typen. Maskinkravet settes derfor til 1GHz prosessorkraft, 256 MB internt minne, 250 MB lagringsplass og skjermkort som støtter OpenGL 1.5 spesifikasjonene. Krav til lagringsplass vil øke med ytterligere 200 MB hvis brukeren ikke har Java installert fra før. Vi vil også utvikle mot en skjermstørrelse på 1024x768 punkter. Vi vil primært

utvikle for plattformen Microsoft Windows, ettersom dette utgjør mesteparten av markedet for skrivebordsmaskiner.

Det er spesielt i inkrement 1 (-3D-generatoren) at vi må legge vekt på ytelse da dette danner basis for mesteparten av videre utvikling. I brukergrensesnittet vil vi i stor grad være bundet til de ytelsesforhold som er lagt til rette i bibliotekene vi benytter og trenger ikke bruke mye tid på ytelsemessige hensyn. I senere inkrementer som tar for seg grafiske brukergrensesnitt i større grad er ytelsen stort sett ikke være et tema da vi vil jobbe utelukkende med ferdige biblioteker for grafiske brukergrensesnitt og innlesing fra fil og vil derfor være bundet til den ytelse dette måtte ha. Mesteparten av funksjonaliteten vil allerede eksistere i inkrement 1 (-3D-generatoren) og man vil utnytte og tilpasse dette. Det er derfor rimelig å anta at enhver datamaskin som uproblematisk kan bruke inkremenet 1 (-3D-generatoren) også kan kan bruke brukergrensesnittet i senere inkrementer helt uten problemer.

2.2.6 Støtte for endringer/utvidelser

Applikasjonen kommer høyst sannsynlig til å bli utvidet senere hvis prosjektet blir en suksess. Utvidelsene gjelder både utvikling av selve funksjonaliteten og på biblioteket med trafikale objekter. Planlagte utvidelser av funksjonaliteten går på distribuering av spørsmål over internett og utvikling av web-basert grensesnitt for å tillate brukeren å ta prøver fra egen datamaskin. Støtte for denne utvidelsen må finne sted i inkrement 1 (-3D-generatoren) og inkrement 4 (-spørsmålsgeneratoren), og vil først og fremst skje i form av måten situasjoner blir lagret på når de blir tatt i bruk i et teorispørsmål.

2.3 Inkrement 1: 3D-generator

2.3.1 Beskrivelse

Dette inkrementet (ink 1 - 3D-generator) skal være selve grunnstenen for systemet og vil være den delen av systemet resten av applikasjonen bygger videre på. Det vil ved inkrementets slutt inneholde mesteparten av funksjonaliteten forbundet med selve genereringen og visningen av 3D-bilder. Det vil i denne delen av utviklingsprosessen bli jobbet svært lite med brukergrensesnitt forutenom det som er tilknyttet tastatur og mus. Selve grafiske brukegrensesnittet vil først bli påbegynt i påfølgende inkrementer.

3D-generatoren som blir utviklet i inkrementet vil i all hovedsak ta for seg innlesing av trafikale objekter fra fil, redigering av egenskapene til disse, og hvordan det presenteres grafisk for brukeren av programmet avhengig av objektets egenskaper og gitte variable. Det vil bli fokusert på å sette opp en robust datastruktur for å representere objektenes relasjoner til gitte trafikksituasjoner og et internt grensesnitt for å oppdatere denne strukturen. Et optimalt internt grensesnitt vil i stor grad være det som er målet å få på plass før påfølgende inkrementer skal påbegynnes.

2.3.2 Supplementær spesifikasjon

Funksjonalitet

Scenarier og situasjoner Trafikale hendelser i applikasjonen deles opp i scenarier og situasjoner. Et scenario er en hendelse som kan oppstå på en gitt lokasjon (eksempelvis kryss eller rundkjøring). Denne hendelsen består i tillegg til lokasjonen av ulike trafikale objekter og en gitt innsynsvinkel. En situasjon er en fin-inndeling av et scenario som viser en bestemt variant av hendelsen som kan oppstå. Hensikten er å kunne ha forskjellige, men relaterte, varianter av samme hendelse tilknyttet ett og samme scenario. Et eksempel på bruk av situasjoner er hvis man har en hendelse og ønsker å kunne vise samme hendelse hvis en ny bil ankommer. Når en ny situasjon opprettes vil den alltid basere seg på en allerede eksisterende situasjon for samme scenario med unntak av når man oppretter nytt scenario og dermed også første situasjon. Hvert scenario kan inneholde mellom 1 og 20 situasjoner. Applikasjonen vil også utvikles for å kunne holde flere uavhengige scenarier åpne samtidig.

Plassering av objekter Man skal ha mulighet til å kunne laste inn trafikale objekter fra fil til gitte koordinater på skjermen. Objektene skal senere kunne velges ved å trykke på de. Et objekt skal når som helst kunne flyttes eller roteres ved hjelp av å klikke og dra med mus og minimalt med tastaturkombinasjoner. Et valgt objekt skal også enkelt kunne fjernes fra situasjonen. Kun 1 objekt skal kunne være valgt om gangen, og ved opprettelse av nytt objekt skal alltid det nye objektet bli valgt. Ved endring av objekter (flytting, rotering, o.l.) vil dette også skje i alle andre situasjoner som inneholder samme objekt instans (innad i samme scenario) forutsatt at bruker ikke var valgt på forhånd at situasjoner skal være uavhengig av hverandre.

Objekt-sti Et plassert objekt (primært kjøretøy, personer og dyr) skal kunne ha mulighet for å opprette en, og bare en, synlig sti som markerer hvor objektet skal bevege seg. Denne stien skal dele farge med objektets tekst og skal kunne endres for hvert unike objekt i situasjonen. En oppmerket sti skal enkelt kunne fjernes igjen eller overskrives av en ny sti.

Objekt-tekst Et plassert objekt (primært kjøretøy, personer og dyr) skal kunne ha mulighet for å opprette en, og bare en, synlig tilknyttet tekst for å representere objektet. Denne teksten skal dele farge med objektets sti og skal kunne endres for hvert unike objekt i situasjonen. En angitt tekst skal enkelt kunne fjernes igjen eller overskrives av en ny tekst.

Gå inn i objekt/Førstepersons-perspektiv Det skal på bestemte typer objekter (kjøretøy og personer) være mulig å gå inn i et førstepersons-perspektiv. I de tilfeller det skal være mulig er det nødvendig å erstatte den synlige modellen på objektet til en ramme rundt kameraet som representerer objektet innvendig. Hvis man går inn i objekter med speil (f.eks. biler) skal også disse være fungerende.

Trafikklys Applikasjonen skal også ha støtte for trafikk-lys. Disse skal opptre som vanlige objekter og vil derfor kunne opprettes, flyttes, roteres og fjernes dag på lik linje med andre objekter, men vil i tillegg ha ekstra funksjonalitet for å bytte farge på lyset. De mulige fargene skal være rødt, rødt + gult, gult, og grønt og vil ha ett enkelt grensesnitt for å gå til neste farge.

Fotgjenger-overganger Det skal være mulig å trekke opp fotgjengeroverganger på valgfri lengde, lokalisering og retning i situasjonen man jobber med.

Disse er ikke nødvendig å kunne redigere etterpå, men må enkelt kunne fjernes på lik linje med andre objekter.

Kamerastyring/kameramodus Innsynsvinkelen til scenariet skal i utgangspunktet være fritt i den forstand at man kan bevege seg i hvilken retning og avstand man selv måtte ønske ved hjelp av mus og minimale tastaturkombinasjoner. I tillegg til dette skal man kunne gå i et eget kameramodus for redigering som flytter innsynsvinkelen til rett over scenariet. Dette kameramoduset skal være globalt for hele scenariet så endringen i innsynsvinkelen vil være felles for alle åpne situasjoner. Ved å gå ut av denne kameramodusen skal innsynsvinkelen stilles tilbake til der den var på forhånd. I tillegg kommer kameramodusen, men også denne lar seg overstyre av kameramodus for redigering.

Situasjonstekst Hver enkelt situasjon skal kunne ha en tilknyttet tekst som kan brukes av trafikkskolelæreren til å beskrive situasjonen eller spørre et spørsmål tilknyttet situasjonen for bruk ved presentasjon. Denne teksten må kunne opprettes, endres og fjernes. Teksten skal ikke være synlig når brukeren befinner seg i kameramodus for redigering.

Skalering av objekter I de tilfeller man baserer et scenario på egendefinert lokasjon er det viktig å kunne skalere objektene i det bestemte scenariet. Hensikten med dette er at egendefinerte bilder kan være i forskjellig skala fra de medfølgende lokasjonene. Det må derfor eksistere en metode å kunne skalere alle objekter i et egendefinert scenario, men uten at det påvirker andre scenarier.

Brukervennlighet

Brukervennligheten er kritisk for at 3D-generatoren skal kunne brukes effektivt i de omgivelsene det er tiltenkt. For å øke brukervennligheten skal følgende krav oppfylles:

- Flytting og rotasjon av objekter skal være mulig ved hjelp av mus.
- Flytting og rotasjon av kamera skal være lagt opp til mus, og det skal være flere forskjellige kamera muligheter slik at brukeren lett skal kunne gå dit han/hun vil.
- Hurtigknapper som skal benyttes skal være logisk plassert på tastaturet slik at brukeren lett skal kunne sette seg inn i de.

- Oppmerking av objekt-stier og fotgjenger-overganger må spesielt tas hensyn til da det gjelder brukervennlighet. Vi vil her prøve å sørge for at man kun trenger å si i fra når man vil starte oppmerking også vil resten automatiseres gjennom mus-operasjoner.

Brukervennlighet vil i større grad bli fokusert på i senere inkrementer som tar for seg brukergrensesnitt til ulike deler av applikasjonen. Dette inkremenetet vil først og fremst ta seg av kjernen av funksjonalitet, og vil i svært liten grad involvere grensesnitt mot brukeren.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.2 - Brukervennlighet

Robusthet

Vi antar at målgruppen til systemet ikke har spesielt god kunnskap om datamaskiner og applikasjoner av denne typen og derfor vil kunne utføre uforutsette handlinger. Vi vil derfor prøve å eliminere enhver mulighet for å kunne gjøre feil ved å begrense brukerne sin tilgang til funksjonaliteten. I dette inkrementet betyr dette i all hovedsak å legge begrensninger på kameraet slik at brukeren ikke kan rote seg bort i 3D-bildet han/hun genererer. Begrensningene inkluderer, men begrenser seg ikke til maksimal rekkevide man kan flytte og rotere kameraet fra situasjonen.

Ved innlesing av datafiler som modeller og bilder bruker vi spesifikasjonene til kjente filformater (eksempel modellformatet ms3d og bildeformatene jpg og png) og brukeren av systemet vil ikke endre disse. Det er derfor rimlig å anta at filformater ved innlesing ikke avviker fra spesifikasjonene.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.4 - Robusthet

Ytelse

3D-generatoren er kjernen i systemet vårt hvor mesteparten av funksjonaliteten er implementert og flaskehalser på denne delen av systemet vil derfor kunne få fatale følger for senere funksjonalitet og utvidelser. Dette betyr at vi i mye større grad enn senere inkrementer må jobbe for god ytelse i denne delen av utviklingsprosessen. For å oppnå dette kravet vil vi sørge for at datafiler som f.eks. modeller og bilder kun eksisterer i minnet ved behov og kun en instans

av filen samtidig.Vi vil også sørge for at bildet på skjermen kun oppdateres ved behov fremfor å tegne det opp kontinuerlig slik som ofte er vanlig i 3D-applikasjoner.

Støtte for endringer/utvidelser

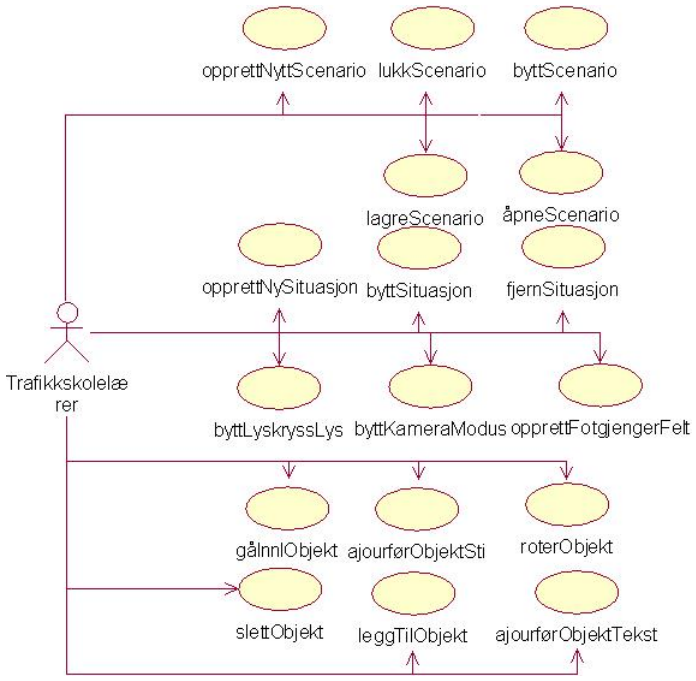
All funksjonalitet i 3D-generatoren skal tilegnes et internt grensesnitt som kan brukes av det grafiske brukergrensesnittet i påfølgende inkrementer av utviklingsprosessen. Dette vil i utgangspunktet bety at enhver funksjon man kan bruke 3D-generatoren til skal ha tilsvarende funksjon(er) med logiske navn i kildekoden for å starte/utføre/stoppe funksjonaliteten. På den måten vil implementasjonen av brukergrensesnitt i inkrement 2 (-menysystem for 3D-generator), samt senere utvidelser, gå bedre.

2.4 Inkrement 2: Brukergrensesnittet for 3D-generatoren

2.4.1 Beskrivelse

Dette inkrementet (ink 2 - menysystem for 3D-generator) skal bygge det grafiske grensesnittet som brukeren av systemet vil benytte seg av når han/hun skal generere en trafikal hendelse for presentasjon eller teori-spørsmål. Ved inkrementets slutt vil applikasjonen ha et komplett brukergrensesnitt for generering av trafikale hendelser og innlesing/lagring av disse fra/til fil samt mulighet til å ta med seg aktiv situasjon videre til presentasjon og spørsmåls-modus. Mesteparten av funksjonaliteten som vil bli gitt et brukergrensesnitt vil allerede eksistere gjennom et internt grensesnitt til forrige inkrement (-3D-generatoren), men det er forventet at det må utføres tilpassinger når design er på plass.

Use-case diagram



Figur 2.1: Usecase diagram for inkrement 2.

Beskrivelse av use-case diagram

Use-case	opprettNyttScenario
Formål	Å få en ny uavhengig trafikal hendelse i systemet som senere kan vises i presentasjonsmodus eller legges til et teori-spørsmål
Forutsetninger	Applikasjonen må være i generator-modus
Beskrivelse	Brukeren skal fra brukergrensesnittet kunne oppretter et nytt scenario. Et scenario er en hendelse som kan oppstå på en gitt lokasjon (eksempelvis kryss eller rundkjøring). Denne hendelsen består i tillegg til lokasjonen av ulike trafikale objekter og en gitt innsynsvinkel (kravspesifikasjon inkrement 1, 3D-generator, pkt 3.1.8). Når brukeren ønsker å opprette et nytt scenario vil han/hun bli presentert en liste over mulige predefinerte lokasjoner eller muligheten til å hente inn egendefinert lokasjon fra en bildefil. Ved opprettelse av nytt scenario vil det nye scenariet alltid settes til aktivt/fokusert (se byttScenario).
Resultat	Trafikkskolelæreren vil få et grunnlag han/hun kan plassere trafikale objekter på.

Use-case	lukkScenario
Formål	Å kunne fjerne et scenario som ikke lenger er i bruk for å rydde i brukergrensesnittet og frigi opptatte systemressurser.
Forutsetninger	Applikasjonen må være i generator-modus og det må være et aktivt scenario.
Beskrivelse	Det skal til enhver tid være mulig fra brukergrensesnittet å lukke et aktivt scenario og dermed alle medfølgende situasjoner. Ved lukking av et scenario vil brukeren bli spurt om han/hun ønsker å lagre scenariet til fil (se lagreScenario). Når scenariet blir lukket vil neste scenario bli satt til aktivt/fokusert. Hvis det lukkede scenariet var siste scenario blir forrige scenario satt til aktivt istedenfor.
Resultat	Applikasjonen vil ha færre scenarier åpne.

Use-case	lagreScenario
Formål	Å kunne ta vare på et scenario til senere bruk selv om man lukker applikasjonen eller skrur av datamaskinen.
Forutsetninger	Applikasjonen må være i generator-modus og det må være et aktivt scenario.
Beskrivelse	Det skal være mulig fra brukergrensesnittet å lagre all relevant informasjon tilknyttet et scenario til fil ved å kun angi filnavn man ønsker å lagre til. Med all relevant informasjon menes all informasjon som skal til for å gjennopprette scenariet i tilstanden det er før lagring. Filformatet kan være egendefinert, men skal være portabelt. Bruker blir bedt om å angi lokasjon på maskinen og filnavn hvor scenariet skal lagres.
Resultat	Scenariet blir lagret til valgfri fil.

Use-case	åpneScenario
Formål	Å kunne hente inn et tidligere lagret scenario til applikasjonen for å kunne slippe å starte på nytt.
Forutsetninger	Applikasjonen må være i generator-modus.
Beskrivelse	Det skal være mulig fra brukergrensesnittet å hente inn tidligere lagrede scenarier fra fil ved å kun angi filnavn man ønsker å gjennopprette fra. Det er viktig at scenariet er nøyaktig likt etter innhenting som det var før lagring. Filformatet kan være egendefinert, men skal være portabelt. Bruker vil bli bedt om å angi lokasjon på maskinen og filnavn hvor scenariet skal lagres. Ved gjennoppretting av scenario vil ønsket scenario bli satt til aktivt/fokusert i applikasjonen (se byttScenario).
Resultat	Et scenario blir gjenopprettet fra fil.

Use-case	byttScenario
Formål	Å kunne bruke flere scenarier i applikasjonen samtidig.
Forutsetninger	Applikasjonen må være i generator-modus og ha minst 2 scenarier.
Beskrivelse	For å kunne bytte mellom scenarier må applikasjonen implementere en oppgavelinje som holder en knapp for hvert åpent scenario i systemet. For å bytte mellom scenarier vil man da kunne bare trykke på knappen som representerer ønsket scenario i oppgavelinjen. Når en knapp i oppgavelinjen har blitt nedtrykt vil scenariet som knappen representerer valg og vist på skjerm og knappen vil bli forbli nedtrykt for å indikere hvilket scenario som er valgt.
Resultat	Ønsket scenario blir aktivt/fokusert i applikasjonen.

Use-case	opprettNySituasjon
Formål	Å få en ny variant av en trafikal hendelse.
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario.
Beskrivelse	Brukeren skal enkelt fra brukergrensesnittet kunne opprette en ny situasjon. En situasjon er en fininndeling av et scenario som viser en bestemt variant av hendelsen som kan oppstå. Hensikten er å kunne ha forskjellige, men relaterte, varianter av samme hendelse tilknyttet et og samme scenario. Når en ny situasjon opprettes vil den alltid basere seg på en allerede eksisterende situasjon for samme scenario med unntak av når man oppretter nytt scenario og dermed også første situasjon. Hvert scenario kan inneholde mellom 1 og 20 situasjoner. En ny situasjon vil alltid bli satt til aktiv/fokusert og vil alltid plassert etter situasjonen den baserer seg på i listen over situasjoner.
Resultat	Man får en ny situasjon tilknyttet aktivt scenario.

Use-case	fjernSituasjon
Formål	Å kunne fjerne situasjoner man ikke trenger.
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario med 2 eller flere situasjoner.
Beskrivelse	Hvis aktive scenario har flere enn 1 situasjon skal man fra brukergrensesnittet kunne lukke alle bortsett fra en. Man må alltid ha minst en situasjon i et scenario og man skal derfor ikke kunne slette aktiv situasjon hvis det bare eksisterer en. Ved lukking av situasjon settes neste situasjon til aktiv/fokusert. Hvis det var siste situasjon som ble lukket så settes forrige situasjon til aktiv/fokusert istedenfor.
Resultat	Ønsket situasjon blir fjernet.

Use-case	byttSituasjon
Formål	Å kunne bruke flere situasjoner i applikasjonen samtidig.
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario med 2 eller flere situasjoner.
Beskrivelse	Det må være mulig å gå navigere mellom situasjoner innad i et scenario. Man skal kunne gå til forrige og neste situasjon samt å gå rett til en angitt nummerert situasjon. Brukeren skal hele tiden få oppgitt hvilken situasjon han/hun befinner seg i og hvor mange situasjoner det eksisterer totalt.
Resultat	Man får opp ønsket situasjon på skjermen.

Use-case	byttLysKryssLys
Formål	Å kunne endre lys i et valgt objekt som skal representere et lys i et lyskryss.
Forutsetninger	Applikasjonen må være i generator-modus, ha et aktivt scenario og ha et lys-objekt valgt.
Beskrivelse	Hvis et lys-objekt er valgt skal det være mulig å endre fargen på dette fra brukergrensesnittet. De mulige fargene skal være rødt, rødt + gult, gult, og grønt og vil ha ett enkelt grensesnitt for å gå til neste farge.
Resultat	Lys-objektet endrer farge.

Use-case	roterObjekt
Formål	Å kunne snu et objekt i den retning som er ønsket.
Forutsetninger	Applikasjonen må være i generator-modus, ha et aktivt scenario og ha et objekt valgt.
Beskrivelse	Det skal fra brukergrensesnittet være mulig å rotere et objekt. Man skal kunne rotere både mot høyre og mot venstre, og hvor langt objektet roteres skal være avhengig av hvor lenge man trykker på knappen.
Resultat	Objektet peker i ønsket retning.

Use-case	leggTilObjekt
Formål	Legge til et nytt objekt til situasjonen.
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario.
Beskrivelse	Brukeren vil ha en liste over objekter, representert som en forhåndsvisning av objektet, sortert etter kategori. Brukeren skal så kunne velge ønsket objekt med muspekeren og dra det dit han/hun ønsker i situasjonen. Situasjonen vil da opprette et nytt objekt av ønsket type på det stedet brukeren dro muspekeren. Ved plassering av nye objekter skal alltid det nye objektet automatisk bli valgt for videre operasjoner.
Resultat	Et trafikalt objekt vil være plassert der brukeren ønsket i situasjonen.

Use-case	AjourførObjektTekst
Formål	Å tilknytte objektet en tekst for identifisering ved presentasjon og i spørsmålstekst.
Forutsetninger	Applikasjonen må være i generator-modus, ha et aktivt scenario og ha et gyldig objekt (primært kjøretøy og personer/dyr) valgt.
Beskrivelse	Et valgt objekt (primært kjøretøy, personer og dyr) skal kunne ha mulighet for å opprette en, og bare en, synlig tilknyttet tekst for å representere objektet. En angitt tekst skal enkelt kunne fjernes igjen eller overskrives av en ny tekst.
Resultat	Objektet får tilknyttet en identifikasjons-tekst.

Use-case	slettObjekt
Formål	Å fjerne uønskede objekter som allerede eksisterer.
Forutsetninger	Applikasjonen må være i generator-modus, ha et aktivt scenario og ha et valgt objekt (primært kjøretøy).
Beskrivelse	Brukeren trenger et enkelt grensesnitt for å kunne fjerne objekter fra aktiv situasjon uavhengig av type. Dette gjelder både vanlige objekter, trafikklys og fotgjengeroverganger. Når bruker velger å fjerne et objekt vil han/hun først bli spurt om dette virkelig er hva personen ønsker å gjøre.
Resultat	Valgt objekt forsvinner fra situasjonen.

Use-case	opprettFotgjengerFelt
Formål	Å kunne markere fotgjengerovergang i situasjonen
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario.
Beskrivelse	Det skal fra brukergrensesnittet være enkelt å starte oppmerking av fotgjengeroverganger. Fra brukergrensesnittet er det kun nødvendig å si i fra at man ønsker å starte markeringen, og så vil inkrement 1 (-3D-generatoren) ta seg av resten.
Resultat	Situasjonen vil få en fotgjengerovergang markert på ønsket plassering i situasjonen.

Use-case	ajourførObjektSti
Formål	Å kunne tydelig markere forventet retning et objekt skal bevege seg.
Forutsetninger	Applikasjonen må være i generator-modus, ha et aktivt scenario og må ha et bestemt objekt valgt (primært kjøretøy, personer og dyr).
Beskrivelse	Et plassert objekt (primært kjøretøy, personer og dyr) skal kunne ha mulighet for å opprette en, og bare en, synlig sti som markerer hvor objektet skal bevege seg. En oppmerket sti skal enkelt kunne fjernes igjen eller overskrives av en ny sti. Brukergrensesnittet trenger bare å si i fra til applikasjonen at man ønsker å starte oppmerkingen og så vil inkrement 1 (-3D-generatoren) ta seg av selve oppmerkingen.
Resultat	Objektet får en synlig sti.

Use-case	gåInnIObject
Formål	Å kunne se hendelsen fra et objekts perspektiv.
Forutsetninger	Applikasjonen må være i generator-modus, ha et aktivt scenario og ha et valgt objekt (primært kjøretøy).
Beskrivelse	I tillegg til fri innsynsvinkel i 3D og oversiktsperspektiv i 2D skal brukeren enkelt kunne velge å se situasjonen fra et objekts perspektiv på utvalgte objekt-typer (primært kjøretøy).
Resultat	Kameramodus vil være stilt fra objektets perspektiv hvis den var i fri 3D eller 2D, og i fri 3D hvis kameraet allerede var inni et objekt.

Use-case	byttKameraModus
Formål	Å bytte mellom 2D og 3D innsynsvinkel.
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario.
Beskrivelse	Innsynsvinkelen til scenariet skal i utgangspunktet være fritt 3D i den forstand at man kan bevege seg i hvilken retning og avstand man selv måtte ønske ved hjelp av mus og minimale tastaturkombinasjoner. I tillegg til dette skal man kunne gå i et eget 2D kameramodus for redigering som flytter innsynsvinkelen til rett over scenariet. Dette kameramoduset skal være globalt for hele scenariet så endringen i innsynsvinkelen vil være felles for alle åpne situasjoner. Ved å gå ut av denne kameramodusen skal innsynsvinkelen stilles tilbake til der den var på forhånd.
Resultat	Kameramodusen vil endre seg til 2D hvis den var i fri 3D og til fri 3D hvis den var i 2D.

Aktører Trafikkskolelærer
Trenger et grafisk og brukervennlig grensesnitt til alle funksjonene som legges inn i applikasjonen.

Forutsetninger og avhengighetsforhold

Utviklingen av brukergrensesnittet forutsetter og er avhengig av at all funksjonalitet som skal benyttes er ferdig utviklet. Det betyr at inkrement 1 (-3D-generatoren) må være ferdigstilt til en slik grad at vi under utviklingen av brukegrensesnittet alltid har den funksjonalitet som skal benyttes tilgjengelig.

2.4.2 Supplementær spesifikasjon

Brukervennlighet

Applikasjonens fremtidige bruk avhenger av hvor lett det er å bruke, og hvor raskt en trafikkskolelærer kan greie å få laget det han/hun ønsker. For å oppnå dette må det grafiske brukergrensesnittet utvikles med omhu. Det må tilstrebes en så flat menystruktur som mulig uten at dette fører til at grensenittet blir overfylt av knapper. Noen enkle punkter kan settes ned som en mal på hvordan grensesnittet skal se ut:

- Aldri mer enn tre nivåer i menyer.
- Alltid ha minst 50% av skjermflaten tilgjengelig for hovedformålet i programmet. Når man bygger situasjoner så vil det si at minst 50% av overflaten er selve situasjonen.
- Bruke vanlig strukturering av menyer, som forenkler forståelsen. Eksempel på dette er meny og verktøylinje øverst til venstre.
- Fjerne funksjonalitet fra brukergrensesnittet når man befinner seg utenfor den funksjonalitetens bruksområde.
- Knapper skal ha en fast posisjon i brukergrensesnittet slik at brukeren til enhver tid skal vite hvor den funksjonaliteten han/hun ønsker ligger.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.2 - Brukervennlighet

Robusthet

Vi ønsker at ethvert element i brukergrensesnittet skal være såpass robust at det ikke oppstår interne feil i applikasjonen ved uventede kombinasjoner i bruken av disse elementene. En gitt funksjonalitet har et bestemt bruksområde og må avgrenses mot det interne grensesnittet slik at brukeren av applikasjonen ikke er i stand til å bruke funksjonaliteten utenfor dette området. Eksempel på dette er at funksjonalitet i brukergrensesnittet for å slette et objekt kun virker hvis et objekt er markert.

Det må tas spesielt hensyn til at filer kan bli korrupte. Hvis en bruker ved uforsiktig bruk har greid å endre på de filene som inneholder den lagrede sitasjonen så vil denne kunne føre til problemer ved innlesing. Applikasjonen må derfor

kunne behandle feil som oppstår hvor som helst under filinnlesing uten at det skal krasje av den grunn.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.4 - Robusthet

Støtte for endringer/utvidelser

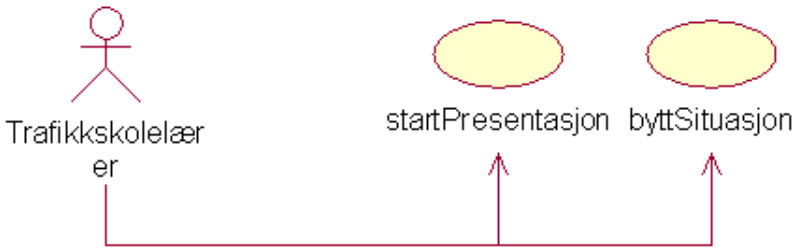
Ettersom all funksjonalitet er utviklet i inkrement 1 (-3D-generatoren) og gjeldende inkrement i stor grad bare tar for seg brukergrensesnitt vil det ikke legges noen stor vekt på støtte for utvidelser og endringer. Alle endringer/utvidelser for denne delen av applikasjonen må først og fremst implementeres i inkrement 1 (-3D-generatoren) og vi ser det derfor ikke hensiktsmessig å bruke ekstra tid på dette i brukergrensesnittet. Det eneste vi vil tilrettelegge i brukergrensesnittet for senere utvikling vil være muligheten til å endre til presentasjons og spørsmåls-modus med aktiv situasjon.

2.5 Inkrement 3: Presentasjonsmodus

2.5.1 Beskrivelse

Dette inkrementet (ink 3 - visning av 3D-bilde) skal bygge det grafiske grensesnittet som brukeren av systemet vil benytte seg av når han/hun skal presentere en trafikal hendelse. Ved inkrementets slutt vil applikasjonen ha et komplett brukergrensesnitt for presentasjon av trafikale hendelser. Mesteparten av funksjonaliteten som vil bli gitt et brukergrensesnitt vil allerede eksistere gjennom et internt grensesnitt til inkrement 1 (-3D-generatoren), men det er forventet at det må utføres tilpassinger når design er på plass.

Use-case diagram



Figur 2.2: Usecase diagram for inkrement 3.

Beskrivelse av use-case diagram

Use-case	startPresentation
Formål	Å kunne få ønsket hendelse på skjerm tilpasset presentasjon.
Forutsetninger	Applikasjonen må være i generator-modus eller spørsmåls-modus.
Beskrivelse	For presentasjon av en trafikal hendelse er det viktig at selve bildet får fokus. Derfor er det nødvendig å: • Fjerne all unødvendig brukergrensesnitt som kan oppleves som en distraksjon. • Kunne navigere mellom situasjoner innad i scenariet man presenterer. Se "byttSituasjon". • Kunne forlate presentasjonsmodus ved behov, f.eks. hvis man ønsker å gjøre endringer etter innspill fra trafikkskole-elever. • Ikke være mulig å endre scenariet eller situasjonene.
Resultat	Ønsket hendelse tar størsteparten av skjermflaten og kun det mest nødvendige av brukergrensesnitt er synlig.

Use-case	byttSituasjon
Formål	Å kunne bruke flere situasjoner i applikasjonen samtidig.
Forutsetninger	Applikasjonen må være i presentasjons-modus og ha et aktivt scenario med 2 eller flere situasjoner.
Beskrivelse	Det må være mulig å gå navigere mellom situasjoner innad i et scenario. Man skal kunne gå til forrige og neste situasjon samt å gå rett til en angitt nummerert situasjon. Brukeren skal hele tiden få oppgitt hvilken situasjon han/hun befinner seg i og hvor mange situasjoner det eksisterer totalt.
Resultat	Man får opp ønsket situasjon på skjermen.

Aktører Trafikkskolelærer
Presenterer en trafikal hendelse for elever.

Forutsetninger og avhengighetsforhold

Utviklingen av brukergrensesnittet forutsetter og er avhengig av at all funksjonalitet som skal benyttes er ferdig utviklet. Det betyr at inkrement 1 (-3D-generatoren) må være ferdigstilt til en slik grad at vi under utviklingen av brukegrensesnittet alltid har den funksjonalitet som skal benyttes tilgjengelig. Det er også en fordel om inkrement 2 (-Menysystem for 3D-generator) er ferdigstilt da vi i dette inkrementet vil bruke en del lignende funksjonalitet.

2.5.2 Supplementær spesifikasjon

Brukervennlighet

For presentasjon må både trafikkskolelærer som underviser kunne raskt kunne navigere rundt i applikasjonen med kun de forutsetninger han/hun har fra bruk av resten av applikasjonen. For elev som blir undervist må han/hun kunne bruke applikasjonen som tiltenkt uten noen form for forkunnskaper om applikasjonen. Noen enkle punkter kan settes ned som en mal på hvordan grensesnittet skal se ut:

- Alltid ha mest mulig av skjermflaten tilegnet hovedformålet. Dette for å tiltrekke fokus på den delen av skjermbildet som er relevant. Dette betyr å eliminere de deler av brukergrensesnittet som ikke er ytterst nødvendig.
- Funksjonalitet som også eksisterer i andre, tidligere utviklede, deler av applikasjonen bør forbli på samme sted i brukegrensesnittet og med samme utseende så brukeren til enhver tid vet hvor den er å finne.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.2 - Brukervennlighet

Robusthet

En gitt funksjonalitet har et bestemt bruksområde og må avgrenses mot det interne grensesnittet slik at brukeren av applikasjonen ikke er i stand til å bruke funksjonaliteten utenfor dette området Eksempel på dette er at funksjonalitet i brukergrensesnittet for å gå til neste situasjon kun lar seg utføre hvis det eksisterer en neste situasjon.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.4 - Robusthet

Støtte for endringer/utvidelser

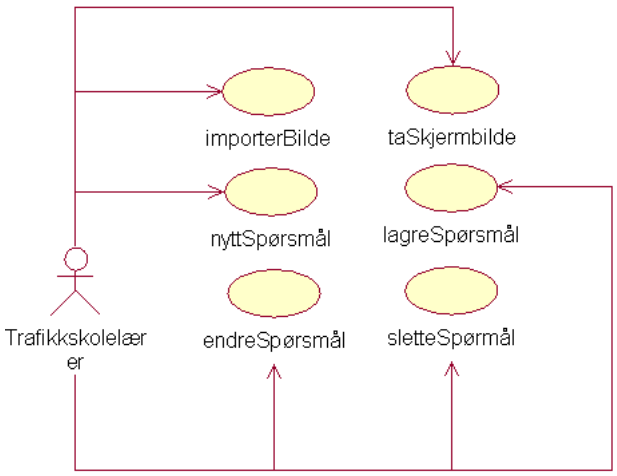
Ettersom all funksjonalitet er utviklet i inkrement 1 (-3D-generatoren) og gjeldende inkrement i stor grad bare tar for seg brukergrensesnitt vil det ikke legges noen stor vekt på støtte for utvidelser og endringer. Alle endringer/utvidelser for denne delen av applikasjonen må først og fremst implementeres i inkrement 1 (-3D-generatoren) og vi ser det derfor ikke hensiktsmessig å bruke ekstra tid på dette i brukergrensesnittet.

2.6 Inkrement 4: Spørsmålsgenerator

2.6.1 Beskrivelse

Dette inkrementet (ink 4 - spørsmål til bilde) tar for seg utviklingen av spørsmålsgeneratoren brukeren vil benytte seg av for å lage teorispørsmål. Spørsmålene skal så kunne brukes i egen applikasjon for å ta teoriprøver og ved senere utvidelser av applikasjonen. Ved inkrementets slutt vil applikasjonen ha et komplett verktøy for opprettelse av teorispørsmål basert på tidligere genererte trafikksituasjoner eller egendefinert stillbilde. Inkrementet baserer seg litt på funksjonalitet fra inkrement 1 (-3D-generatoren), men mesteparten utvikles uavhengig fra tidligere inkrementer.

Use-case diagram



Figur 2.3: Usecase diagram for inkrement 4.

Beskrivelse av use-case diagram

Use-case	nyttSpørsmål
Formål	Å lage et nytt spørsmål
Forutsetninger	Applikasjonen må være i spørsmåls-modus.
Beskrivelse	Bruker skal selv kunne opprette nye spørsmål. Et spørsmål består av følgende data: • Kategori • Bilde • Spørsmålstekst • Alternativer • Hvorvidt et alternativ er riktig eller ikke Bildet som brukes vil enten hentes inn fra generator-modus ved hjelp av use-case "taSkjermbilde" eller fra bildefil med use-case "importerBilde". Etter opprettelse av nytt spørsmål vil det forbli i den interne datastrukturen selv om bruker velger å redigere et annet spørsmål eller opprette et nytt. Et spørsmål skal ha mulighet for flervalg.
Resultat	Et nytt spørsmål blir opprettet.

Use-case	endreSpørsmål
Formål	Å kunne endre et eksisterende spørsmål.
Forutsetninger	Applikasjonen må være i spørsmåls-modus.
Beskrivelse	Det skal til enhver tid være mulig å endre et spørsmål som eksisterer i applikasjonens interne datastruktur. All data om spørsmålet (kategori, bilde, spørsmålstekst, alternativer, riktige svar) er da åpent for endringer, men selve bildet må enten genereres på nytt eller hentes fra ny bildefil hvis det er ønske om endringer. Endringer skal påvirke både den interne datastrukturen og spørsmålsbibliotek-fila.
Resultat	Et spørsmål blir endret.

Use-case	lagreSpørsmål
Formål	Å kunne lagre spørsmålsbiblioteket og hente det inn igjen.
Forutsetninger	Applikasjonen må ha skrive og lese-rettigheter til filen den ønsker å lagre datastrukturen til.
Beskrivelse	Applikasjonen skal selv ta seg av innlesing av spørsmålsbiblioteket fra fil under oppstart. Senere må applikasjonen selv skrive datastrukturen til fil hver gang det har blitt gjort endringer slik at den interne datastrukturen og filen alltid er synkronisert.
Resultat	Applikasjonen får skrevet datastrukturen med spørsmål til fil og får lest den inn fra fil igjen.

Use-case	sletteSpørsmål
Formål	Å kunne fjerne et spørsmål fra spørsmålsbiblioteket.
Forutsetninger	Applikasjonen må være i spørsmåls-modus og man må endre et spørsmål.
Beskrivelse	Det skal i tillegg til å kunne endre et spørsmål være mulig å fjerne det helt. Det skal være så enkelt så man bare trenger å velge spørsmålet for endring og så trykke på en knapp og så forsvinner det både fra den interne datastrukturen og fra spørsmålsbibliotek-fila.
Resultat	Spørsmålet forsvinner fra biblioteket.

Use-case	taSkjermbilde
Formål	Å kunne bruke en generert situasjon til spørsmål.
Forutsetninger	Applikasjonen må være i generator-modus og ha et aktivt scenario.
Beskrivelse	Det skal være mulig for brukeren å overføre en generert situasjon fra generator-modus til spørsmåls-modus. Ved denne overføringen vil aktiv situasjon bli tatt bilde av og lagret som stillbilde i spørsmålsbiblioteket. Et nytt spørsmål opprettes som i use-case "nyttSpørsmål" og bildet legges til.
Resultat	Aktiv situasjon vil bli overført fra generator-modus til et nytt spørsmål i spørsmåls-modus.

Use-case	importerBilde
Formål	Å kunne bruke et stillbilde til spørsmål.
Forutsetninger	Applikasjonen må være i spørsmåls-modus.
Beskrivelse	Det skal være mulig å gi et spørsmål et nytt bilde basert på stillbilde fra fil. Det kan enten være et nytt spørsmål fra use-case "nyttSpørsmål" eller et eksisterende spørsmål fra use-case "endreSpørsmål". Bruker skal selv kunne angi hvor på datamaskinen bildet er lokalisert.
Resultat	Et spørsmål får nytt bilde fra fil.

Aktører Trafikkskolelærer
Ajourfører et spørsmålsbibliotek som eleven senere skal kunne bruke i egenstudier.

Forutsetninger og avhengighetsforhold

Utviklingen av inkrementet er til dels avhengig av å ha ferdigstilt inkrement 1 (-3D-generatoren). Dette fordi man skal kunne bruke genererte trafikale situasjoner som bilde til spørsmål. Forutenom dette vil mye av utviklingen kunne gjennomføres i stor grad uavhengig av tidligere inkrementer.

2.6.2 Supplementær spesifikasjon

Brukervennlighet

Det er svært viktig at brukegrensesnittet er intuitivt og lettfattelig for trafikkskolelæreren og noen enkle punkter kan derfor settes ned som en mal på hvordan grensesnittet skal se ut:

- Spørsmålsgeneratoren vil måtte ha mange komponenter for å organisere spørsmålsdata og det er derfor viktig å ha en god beskrivelse av komponentene så brukeren til enhver tid vet hvilket felt som gjør hva.
- Av samme årsak er det også viktig å samle komponenter som har samme tilhørighet i applikasjonen, f.eks. at datafelter til selve spørsmålet ligger inntil hverandre i en logisk rekkefølge, og funksjonsknapper ligger samlet.

- Datafelter i brukergrensesnittet for å beskrive hvorvidt et spørsmålsalternativ er riktig eller ikke må ha en synlig, entydig, tilhørighet til det alternativet det er tiltenkt.
- Funksjonalitet som også eksisterer i andre, tidligere utviklede, deler av applikasjonen bør forbli på samme sted i brukegrensesnittet og med samme utseende så brukeren til enhver tid vet hvor den er å finne.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.2 - Brukervennlighet

Robusthet

For at applikasjonen skal kunne operere optimalt er det nødvendig at all inndata fra eksterne kilder oppfattes riktig av applikasjonen. For å sørge for at ikke applikasjonen får problemer ved feil inndata må vi ta hensyn til følgende:

- Datafilen til spørsmålsbiblioteket kan være korrupt eller ugyldig i forhold til det definerte filformatet. Korrupt og ugyldig data må ignoreres av applikasjonen.
- Bildefiler som skal importeres vil være begrenset til enkelte formater. Det er derfor viktig at applikasjonen ignorerer ugyldige bildefiler.
- Enkelte datafelter som bruker av applikasjonen kan fylle ut kan forbli tomme eller inneholde uventede verdier. Det er da viktig at applikasjonen tåler enhver mulig verdi fra disse feltene.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.4 - Robusthet

Støtte for endringer/utvidelser

Det er planlagt mulige funksjonelle endringer/utvidelser av prosjektet etter prosjektets slutt. Disse inkluderer blant annet i dette inkrementet muligheten for å velge hvorvidt spørsmål skal være flervalg eller ikke og støtte for distribuering av spørsmål over internett slik at eleven kan sitte på egen maskin å ta prøver og/eller at trafikkskoler kan utveksle spørsmål. Flervalgspørsmål skal bli implementert i den interne datastrukturen på en slik måte at datastrukturen i svært liten grad trenger å endres for å tillate enkeltvalg-spørsmål hvis ønsket. For

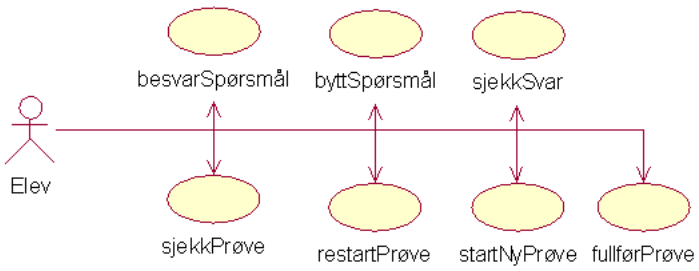
mulig internett-distribuering vil alle situasjoner som blir brukt i spørsmål lagret i et portabelt bildeformat (f.eks. JPG eller PNG) slik at visning av disse skal kunne gjøres uten støtte for vår 3D-generator.

2.7 Inkrement 5: Prøve

2.7.1 Beskrivelse

Dette inkrementet (ink 5 - visning av prøve) skal utvikle et verktøy tiltenkt for å la en trafikkskole-elev kunne ta en teoriprøve i forbindelse med egenstudier før førerprøven. I motsetning til tidligere inkrementer vil dette inkrementet ikke basere seg på tidligere kode og vil eksistere som en egen kjørbar modul i systemet. Ved inkrementets slutt vil systemet inneholde en fullverdig applikasjon for å ta en teoriprøve lokalt på trafikkskolen.

Use-case diagram



Figur 2.4: Usecase diagram for inkrement 5.

Beskrivelse av use-case diagram

Use-case	startNyPrøve
Formål	Å kunne ta en teoriprøve.
Forutsetninger	...
Beskrivelse	For å få ta en prøve må brukeren angi de valg som passer best og starte. Bruker skal i det minste kunne bestemme følgende: • Hvilke kategorier spørsmålene skal inkluderes fra. • Hvor mange spørsmål prøven skal bestå av. Når bruker har gjort sitt valg og ønsker å starte prøven basert på disse skal applikasjonen selv avgjøre tilfeldig hvilke spørsmål som som skal hentes ut fra de valgte kategorier og legge disse i tilfeldig rekkefølge i brukerens prøve. Antall spørsmål skal også stemme overens med brukers ønske. Hvis oppgitt antall spørsmål overstiger det totale antall spørsmål i oppgitte kategorier så vil bruker få en prøve bestående av alle spørsmål fra disse kategoriene.
Resultat	Brukere får en ny prøve som tilfredsstiller de krav han/hun har satt.

Use-case	besvarSpørsmål
Formål	Å avgi sitt svar på et spesifikk spørsmål i prøven
Forutsetninger	Må ha en aktiv prøve.
Beskrivelse	Et spørsmål skal for brukeren bli presentert med følgende: • Bilde • Spørsmålstekst • Svaralternativer Bruker besvarer spørsmål ved å trykke med muspekeren på ønsket alternativ og applikasjonen markerer hvilke alternativer som er valgt. Etter avgitt svar på et spørsmål skal bruker når som helst kunne endre svaret bare ved å trykke på alternativene. Spørsmålene skal være flervalgs-spørsmål.
Resultat	Bruker får avgitt sitt svar.

Use-case	sjekkSvar
Formål	Å kunne se fasit for et enkelt spørsmål.
Forutsetninger	Må ha en aktiv prøve.
Beskrivelse	Det skal være mulig å se fasit på et enkelt spørsmål. Når bruker velger dette vil alle riktige svar bli markert synlig som riktig og alle gale svar vil bli markert synlig som galt.
Resultat	Bruker får oppgitt hvorvidt hans/hennes svar på et gitt spørsmål er riktig eller galt.

Use-case	byttSpørsmål
Formål	Å kunne navigere seg frem og tilbake mellom besvarte og ubesvarte spørsmål.
Forutsetninger	Må ha en aktiv prøve.
Beskrivelse	Det skal være fullt mulig å hoppe frem og tilbake mellom hele prøvens spørsmål til ethvert tidspunkt under et prøve. De mest aktuelle navigeringsmetodene er: • Første spørsmål • Forrige spørsmål • Neste spørsmål • Siste spørsmål På den måten skal brukeren kunne f.eks. både hoppe over spørsmål eller gå tilbake for å rette tidligere besvarte spørsmål. I tillegg skal bruker hele tiden få oppgitt hvor i rekken av spørsmål han/hun befinner seg og hvor mange spørsmål det er totalt.
Resultat	Bruker får opp et ønsket spørsmål.

Use-case	fullførPrøve
Formål	Å kunne avslutte en prøve og se oppsummering.
Forutsetninger	Må ha en aktiv prøve.
Beskrivelse	Det skal når som helst være mulig å fullføre en prøve uavhengig om alle spørsmål er besvart. Når en prøve fullføres mister bruker muligheten til å besvare flere spørsmål og vil bli presentert en oppsummering av prøven. Denne oppsummeringen kan blant annet inneholde følgende: • Antall mulige • Antall riktig • Antall feil • Tidsforbruk
Resultat	Bruker får opp oppsummering av prøven han/hun har besvart.

Use-case	restartPrøve
Formål	Å kunne ta nøyaktig samme prøve en gang til.
Forutsetninger	Må ha fullført en prøve.
Beskrivelse	Når bruker har fullført en prøve kan han/hun velge å starte prøven på nytt fremfor å starte en ny prøve med nye spørsmål. Hvis bruker velger å ta prøven på nytt vil han/hun få opp nøyaktig samme spørsmål i nøyaktig samme rekkefølge, og alle tidligere svar vil være fjernet.
Resultat	Bruker starter en ny prøve med samme spørsmål som tidligere prøve.

Use-case	sjekkPrøve
Formål	Å kunne gå gjennom en hel prøve og se riktige/gale svar.
Forutsetninger	Må ha fullført en prøve.
Beskrivelse	Når bruker har fullført en prøve kan han/hun velge å gå gjennom hele prøven. Dette vil foregå ved at bruker får navigere mellom spørsmål som om han/hun faktisk besvarte prøven, men muligheten for å besvare er fjernet. Riktige og gale svar vil så bli automatisk markert for hvert enkelt spørsmål som i use-case "sjekkSvar".
Resultat	Bruker får se hva han/hun har svar i forhold til hva som er riktig.

Aktører Trafikkskole-elev
Trenger et grafisk brukergrensesnitt for å bli presentert de spørsmål som er generert av trafikkskolelæreren.

Forutsetninger og avhengighetsforhold

Utviklingen av dette inkrementet er ikke avhengig av at tidligere inkrementer er ferdigutviklet. Det eneste som er nødvendig å ha klart er en definisjon av filformatet til spørsmålene som blir generert i inkrement 4 (-spørsmålsgenerator). Dette fordi de to inkrementene vil dele en og samme fil for lesing/skriving av spørsmålsbiblioteket. For å kunne benytte inkrementets produkt som tiltenkt, f.eks. i forbindelse med testing, er det alikevel svært hensiktsmessig å være ferdig med foregående inkrement (-spørsmålsgenerator).

2.7.2 Supplementær spesifikasjon

Brukervennlighet

Applikasjonens fremtidige bruk avhenger av hvor lett det er å bruke, og hvor raskt en trafikkskole-elev klarer å ta i bruk dets fulle potensiale. Brukergrensesnittet må utvikles på en sånn måte at en person med liten eller ingen erfaring med applikasjonen og/eller programvare generelt kan sette seg rett ned og bruke applikasjonen uten problemer. Noen enkle punkter kan settes ned som en mal på hvordan grensesnittet skal se ut:

- Alltid ha størst mulig skjermflate tilgjengelig for hovedformålet til applikasjonen. Det vil si spørsmålsbildet, spørsmålstekst og svaralternativer.
- Det skal være minimalt med knapper og funksjonalitet i brukergrensesnittet så brukeren ikke har for mye å forholde seg til. Dette kan blant annet oppnås ved å fjerne funksjonalitet fra brukergrensesnittet når man befinner seg utenfor dets bruksområde.
- Ved opprettelse av ny prøve er det viktig at de parametere som avgjør prøvens utforming kun er de mest nødvendige. Man skal kunne starte en prøve raskt uten å måtte tenke over valg. Unødvendige valg vil bare forvirre brukeren og øke tiden trafikkskole-eleven bruker på å tilvende seg applikasjonen.
- Knapper skal ha en fast posisjon i brukergrensesnittet slik at brukeren til enhver tid skal vite hvor den funksjonaliteten han/hun ønsker ligger.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.2 - Brukervennlighet

Robusthet

Vi ønsker at ethvert element i brukergrensesnittet skal være såpass robust at det ikke oppstår interne feil i applikasjonen ved uventede kombinasjoner i bruken av disse elementene. En gitt funksjonalitet har et bestemt bruksområde og må avgrenses slik at brukeren av applikasjonen ikke er i stand til å bruke funksjonaliteten utenfor dette området. Eksempel på dette vil i dette inkrementet være at en knapp for å gå til neste spørsmål i prøven ikke er tilgjengelig hvis brukeren ikke holder på med en prøve.

Det må tas spesielt hensyn til at filer kan bli korrupte. Hvis en bruker ved uforsiktig bruk har greid å endre på de filene som inneholder den lagrede sitasjonen så vil denne kunne føre til problemer ved innlesing. Applikasjonen må derfor kunne behandle feil som oppstår under filinnlesing.

Ellers gjelder krav beskrevet i Overordnet kravspesifikasjon pkt 2.2.4 - Robusthet

Støtte for endringer/utvidelser

Applikasjonen vil eksistere som et fullverdig verktøy for å ta teoriprøven lokalt ved inkrementets slutt. Det er alikevel rimlig å tenke over muligheten for å

senere utvide til nettverksbasert innlesing av spørsmålsbiblioteket når man oppretter innlesingsrutinene fra fil og setter opp datastrukturen. Inkrementet omfang omhandler dog ikke nettverks og internett-distribuering av spørsmål, og det er heller ikke sikkert at å utvide produktet som utvikles i dette inkrementet er beste måten å implementere distribueringsrutiner ved en senere anledning. Vi vil derfor utforme den interne datastrukturen og innlesingsrutinene av spørsmål mest mulig robust, men utover dette vil det ikke tas spesielle hensyn til senere utvidelser.

Kapittel 3

Design

3.1 Introduksjon

Følgende dokumenter er en oversikt over den planlagte strukturen av Kos. Dokumentene skal overordnet beskrive hvordan de enkelte inkrementerne henger sammen. Inkrementene skal beskrives ned i den enkelte klasse som utgjør de enkelte inkrementene. Designet bygger på de definerte kravene som er beskrevet i kravspesifikasjonen.
Utviklingsmodellen legger egentlig opp til at designet for hvert inkrement gjøres ferdig før utviklingen påbegynnes. Dette vil være vanskelig å gjennomføre i praksis pga. prosjektmedarbeiderenes begrensede erfaring med jobbing på større prosjekter. Derfor vil designet gjennomgå de forandringene som fremgangen i prosjektet vil kreve. Designdokumentene i dette kapitlet i rapporten vil derfor inneholde siste versjon av designet før prosjektavslutningen.

3.1.1 Formål

Designdokumentene skal gi utviklerene en oversikt over hvordan løsningen skal settes sammen og hvordan den skal virke. Dette gjør utviklingsprosessen lettere siden man kan legge til rette for fremtidige funksjoner på et tidlig stadium og unngår dermed refactoring. Utviklingen skal følge en inkrementell modell. Derfor blir noe refactoring uungåelig siden man ikke besitter alle krav og et fullstendig design før utviklingen starter. Derfor dekker designdokumentet også en overordnet design som viser forholdet mellom inkrementene. I det overordnete designet vil også støtteklasser bli beskrevet.

3.1.2 Teori

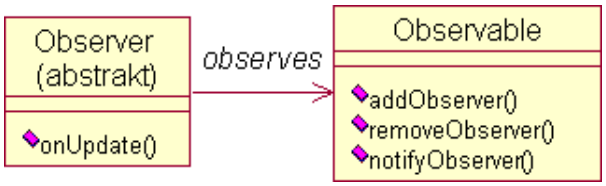
Java er et objektorientert og pakkebasert system. Dette legger føringer på hvordan man deler opp løsningen. Med god navngivning og riktig oppdeling kan et program skrevet i Java være veldig lettfattelig å lese og lett å vedlikeholde. Et prinsipp som kan og bør brukes (så langt det er praktisk mulig) er lagdeling av programmet. Lagdeling vil si at man deler opp systemet etter funksjon.

Designpattern

Pattern er velprøvde metoder for å løse ulike problemstillinger man kan komme ovenfor under utviklingen. Noen av disse er blitt så innarbeidet at de har blitt grunnleggende prinsipper for all utvikling. I denne seksjonen trekkes noen av de pattern som er benyttet i Kos frem og beskrives i korthet.

Lagdeling Mest normalt er en tredeling hvor man har et lag for lavnivå I/O, et lag for funksjonalitet innad i systemet og et lag for grafisk brukergrensesnitt. I denne konteksten er lavnivå I/O begrenset til disklesing/skriving. Initierende kall gjøres bare nedover i lagene. Prinsipielt sett skal aldri en klasse på et lavere nivå gjøre kall til en på et høyere nivå. Desto nærmere sluttbrukeren laget er, desto høyere er nivået. Så lenge man begrenser seg til denne måten å kommunisere på blir det lettere å bytte ut deler av systemet senere.

Observer-observable Selvom lagdeling (se forrige punkt) er et yndet prinsipp er det ofte ønskelig at hendelser på et lavere nivå skal påvirke de på et høyere nivå direkte. For at man skal unngå å ha koblinger direkte oppover i strukturen så implementerer man patternet "observer-observable". Man lager da et generelt grensesnitt mellom klasser som kan benyttes til å sende meldinger mellom de. Man implementerer da et "observablegrensesnitt på de klassene som ligger på et lavere nivå. Dette gjør disse klassene i stand til å motta klasser som implementerer "observergrensesnittet og lagre disse i en egnet datastruktur. Det eneste en "observablebehøver å gjøre da er å kalle en funksjon som er definert på observer grensesnittet som forteller "observeren"at noe har endret seg. Det blir da opp til "observeren"å hente ut de data den trenger fra respektive klasse. I praksis vil dette si at et objekt sier til et annet at det vil bli informert når endringer skjer. Når endringene skjer så er det opp til dette objektet om det ønsker å gjøre noe mer ut av det. Dermed er laginndelingen fortsatt intakt.



Figur 3.1: Prinsippet for observer-observable

Model-view-controller For å best holde orden på data i et program kan man implementere et pattern som baserer seg på en spredning av ansvarsområder i systemet. Man deler dermed koden opp i datastrukturen, presentasjonen av dataene og metoder som påvirker dataene (respektive model, view og controller). I denne strukturen er det kontrolleren som styrer hele showet. Den bestemmer hva som skjer med datastrukturen og den bestemmer hva som skal vises på skjerm.

Lave koblinger/høy styrke

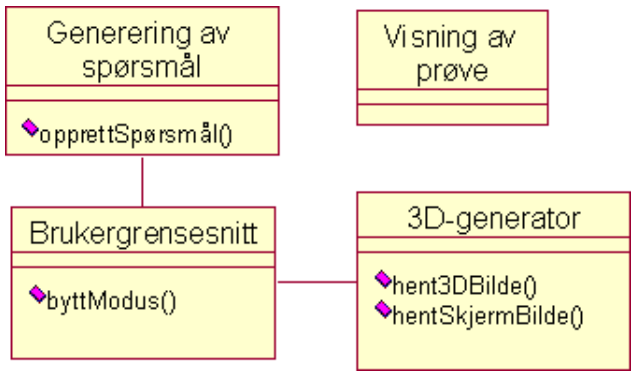
For at et program skal være så fritt for feil som mulig, samt vedlikeholdbart må det være oversiktlig. Et program der all koden er samlet i en klasse er nesten umulig å sette seg inn i fordi det blir for mye informasjon på et sted. Dessuten blir fordelingen av ansvar, et av de viktigste prinsippene i objektorientering, forferdelig dårlig. Man sier da at klasser som har for mye ansvar, utover det som er nødvendig, har lav styrke.
På samme måten ser man programmer der koden er spredd over alt for mange klasser og man sliter med å finne ut hvor selve jobben bli gjort. I slike tilfeller har klassene høy styrke, siden de ikke inneholder mer en strengt tatt nødvendig, men de blir knyttet til for mange andre klasser. Da sier man at de har høy kobling. Når man utvikler et program ønsker man ideelt sett å ha så høy styrke som mulig samtidig som man holder koblingene på et minimum.

3.2 Overordnet design

3.2.1 Logisk utseende

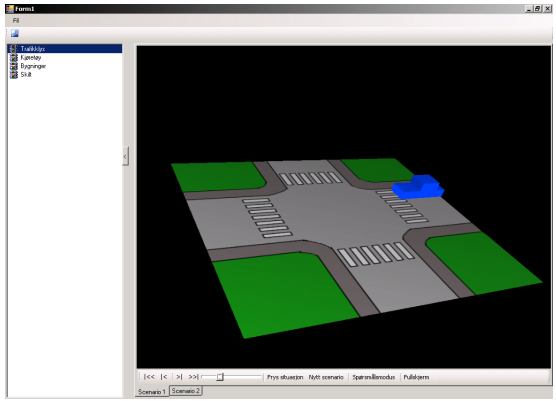
Inkrementene er ikke helt uavhengig av hverandre. Noen trenger litt informasjon fra de andre, mens andre er helt avhengig av noe et annet inkrement har produsert. Oppdelingen i pakker følger ikke oppdelingen i inkrementer. Dette er fordi en del av inkrementene vil kunne implementeres i de samme pakkene som

andre inkrementer. Strukturen blir lagt opp med et felles brukergrensesnitt for å

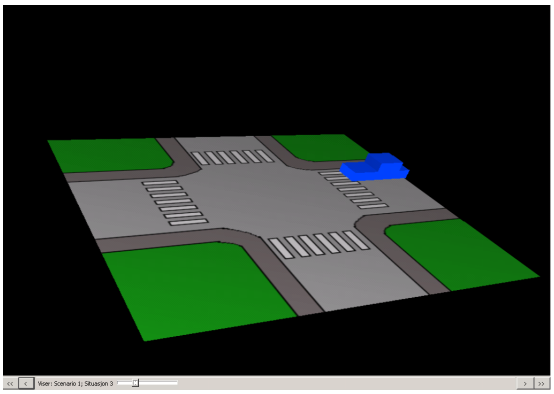


Figur 3.2: Overordnet diagram for pakkene i systemet

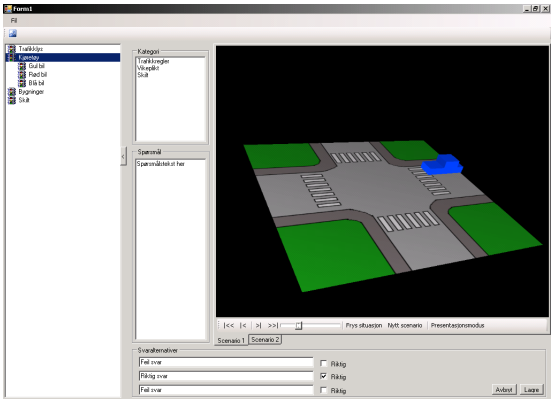
bytte mellom modusene 3D-generering, presentasjon og spørsmålsgeneratoren. Disse forskjellige modusene legges på hvert sitt panel som man kan bytte sømløst i mellom via knapper i programmet. Tidlige konseptskisser følger under her. Disse er grunnlaget for utformingen av det grafiske brukergrensesnittet, men er ikke et absolutt krav. Nærmere beskrivelse av de enkelte grensesnittene kommer under de respektive inkrementene.



Figur 3.3: Tidlig konseptskisse av 3D-generatoren



Figur 3.4: Tidlig konseptskisse av presentasjonsmodusen



Figur 3.5: Tidlig konseptskisse av spørsmålsgeneratoren

3.3 Inkrement 1: 3D-generator

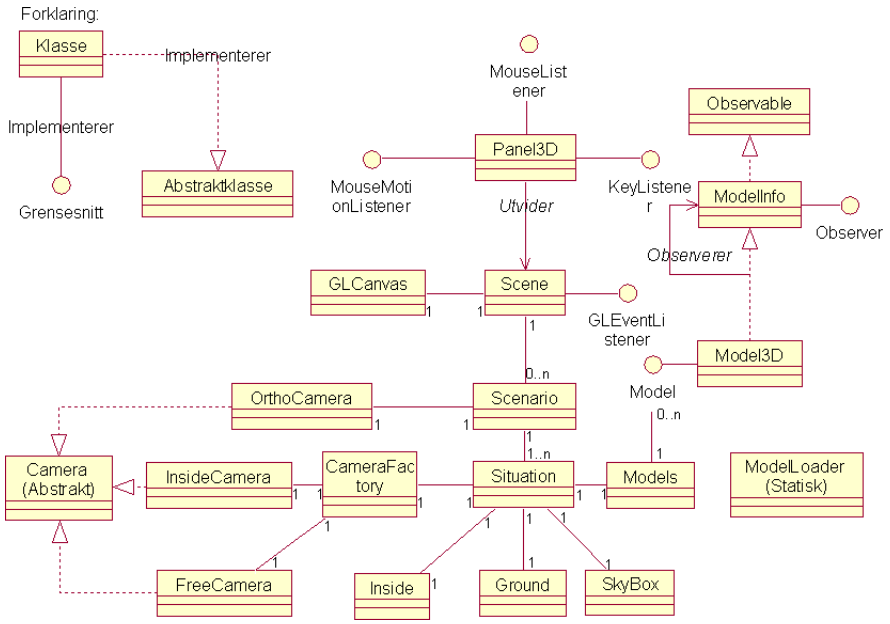
3.3.1 Biblioteker

JOGL

JOGL[1] importeres i dette inkrementet og brukes for all generering av 3D. En del klasser fra JOGL[1]-pakken brukes i programmet.

3.3.2 Logisk utseende

3D-generatoren baserer seg på et OpenGL[2] bibliotek som inneholder de klasser som trengs som grunnlag for å kunne utvikle en 3D-applikasjon uten at det blir for tidkrevende.
I grunnen for 3D-generatoren ligger et GLCanvas. Dette er en utvidelse av det Canvas som ligger i AWT pakken til Java. På dette canvaset tegnes de tredimensjonale bildene opp. Strukturen i programmet blir beskrevet ved hjelp av klassediagrammer. Diagrammet er begrenset til de klassene som har en sentral virkning i programmet. I tillegg kommer en del klasser som har en støttefunksjon, eller som bare tar seg av en liten finesse i programmet. (Komplett klassediagram finnes som vedlegg.) De viktigste av klassene i strukturen vil også bli beskrevet ned i detalj.



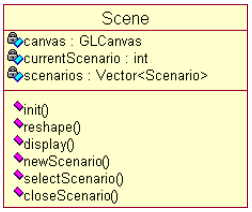
Figur 3.6: Klassediagram for inkrement 1. Revidert med siste endringer i design.

Utdypning av klassediagram

GLCanvas, Scene og GLEventListener Scene klassen er en utvidelse av JPanel som er en vanlig Swing konteiner. På Scene klassen ligger et GLCanvas hvor all grafikk tegnes opp. For at vi GLCanvas skal vite hva den skal tegne opp skal det legges en GLEventListener på GLCanvas. GLEventListener er et

grensesnitt i JOGL som implementeres av Scene klassen. Dette grensenittet mottar meldinger via Java sitt eventsystem hver gang noe hender som påvirker hva som skal tegnes på GLCanvas. Det er tre forskjellige hendelser som kan oppstå, som håndteres av respektive metoder i GLEventListener:

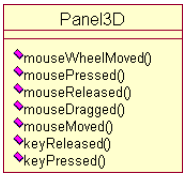
- init(..) : Bestemmer hva som skal skje når GLCanvas initieres.
- reshape(..) : Bestemmer hva som skal skje når GLCanvas endrer størrelse.
- display(..) : Tegner opp på GLCanvas de elementene som skal vises.



Figur 3.7: Scene klassen med sine viktigste metoder og varible

Scene er også den klassen som får ansvaret for å holde på alle de Scenario som opprettes i programmer. Metoder for å styre disse kommer av seg selv etterhvert og er avhengig av hvilke funksjonalitet som legges til i programmet.

Scene og Panel3D Panel3D implementerer alle de grensesnittene som tar imot input fra brukeren som påvirker 3D-scenen. Panel3D implementerer lyttere som tar imot hendelser fra tastatur og mus. Disse tolker Panel3D for så å gi Scene beskjed om hvordan 3D-scenen skal påvirkes. Panel3D tar imot alle tastatur og



Figur 3.8: Panel3D-klassen med sine viktigste metoder for håndtering av hendelser

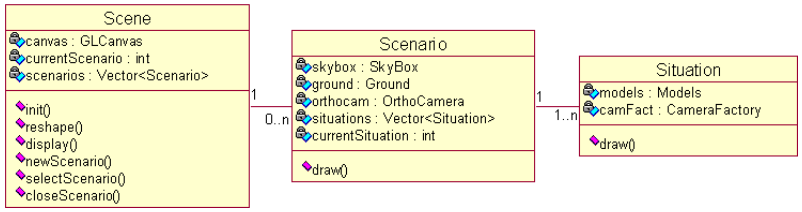
mus hendelser så lenge den har fokus. Avhengig av om en modell er valgt eller ulike kombinasjoner av tastetrykk påvirkes modeller eller hele scenen på ulike måter. Disse hendelsene skal styres mottas av Panel3D:

- Flytting/rotering av kameraet i scenen.

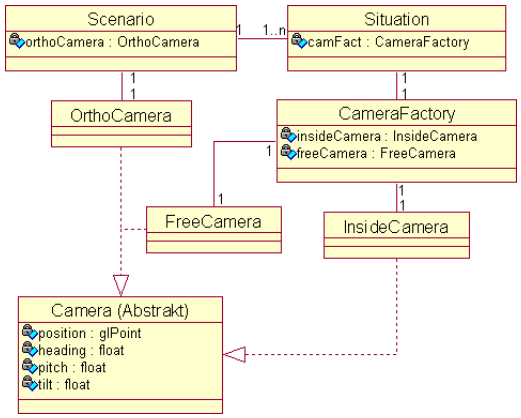
- Flytting/rotering/sletting av objekter i scenen.
- Tegning av objekter som ikke er forhåndsdefinert i størrelse og form. (Fotgjengerfelt og piler.)

Ved å kombinere MouseMotionListener, KeyListener og MouseListener er det mulig å kunne utføre temmelig komplekse kommandoer. Komplett liste over kommandoer finnes i brukerveiledningen.

Scene, Scenario og Situation I KOS kan man ha mange ulike scenarier som hver innneholder forskjellige situasjoner med utgangspunkt i det samme. Disse skal ligge i en datastruktur med utgangspunkt i en Vector[1] liggende i Scene klassen. Denne Vectoren inneholder Scenario objekter som igjen inneholder en Vector med Situation objekter.



Figur 3.9: Forholdet mellom Scene, Scenario og Situation klassene

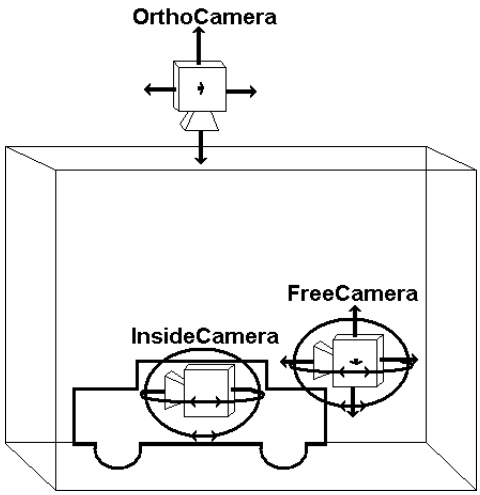


Figur 3.10: Forholdet mellom Scenario, Situation og Camera klassene

[1] Vector: En klasse i Java som fungerer som en dynamisk array.

Camera og CameraFactory Det finnes tre typer kameraer. Disse kameraene får den samme input fra brukeren, men behandler input på forskjellige måter. Derfor er det i designen lagt opp til en abstrakt kameraklasse som har abstrakte funksjoner som mottar denne input. Dermed blir det for de som bruker kameraene likegyldig hva slags kamera det er, man bruker de samme funksjonene uansett, men resultatet blir forskjellig utfra hvilken subklasse det er som ligger bak. CameraFactory klassen er den som holder styr på kameraene som ligger i hver Situation. En Situation kan presenteres med enten FreeCamera eller InsideCamera. Hvert Scenario har i tillegg ett OrthoCamera som virker inn på alle sine Situation objekter. Når det i Scenario er satt at OrthoCamera skal brukes så overstyrer dette det valget som er gjort i Situation.
Slik fungerer de enkelte kameraene:

- FreeCamera: Kan bevege seg langs alle tre aksene samt rotere opp/ned og til sidene.
- InsideCamera: Kan ikke bevege seg langs noen akser, men kan rotere opp/ned og til sidene.
- OrthoCamera: Kan bevege seg langs alle aksene, men kan ikke rotere.



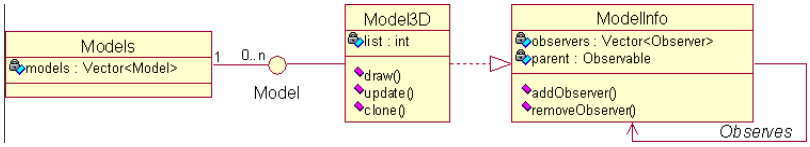
Figur 3.11: Slik er bevegelsesmønstrene til de enkelte kameratypene.

De ulike kamerane fyller ulike roller. OrtoCamera er i første omgang tiltenkt å brukes til redigering av hver enkelt Situation. FreeCamera kan brukes til å plassere kameraet hvor enn man vill ønske for å få presentert Situation fra den vinklingen man ønsker. InsideCamera er uløselig låst til det kjøretøyet som er valgt i det det blir valgt. Da vises Sitation slik den ville se ut fra valgt kjøretøy sitt ståsted. InsideCam simulerer bevegelsene man har tilgjengelig ved å vri på hodet når man sitter i forsetet av en bil eller lignende.

Situation og Models Et Situation objekt inneholder alltid et Models objekt. Models objektet tar seg av alt styret med modellene i den gitte Situation. De funksjonene som Models klassen oppfyller er:

1. Holde på datastrukturen for modellene til sitt Situation objekt.
2. Videreformidle kommandoer til riktig modell.

Models objektet inneholder derfor en Vector med objekter som implementerer Model grensesnittet. Metodene i Models påvirker modellene i datastrukturen på forskjellige måter. Eksempelvis flytte nåværende valgte modell. Alle modeller implementerer Model grensesnittet. I Model er metoder felles for alle modeller deklarert. Dette er funksjoner som draw(..) og metoder som påvirker posisjonen til modellene. Dette være seg posisjon, rotasjon og om modellen er valgt eller ikke. Update(..) metoden i Model3D er den metoden som tar seg av



Figur 3.12: Strukturen og de viktigste funksjonene og dataene i datastrukturen til modellene.

endringene som skal gjøres når den sin parent foretar seg noe. Hvis en modell sin parent flyttes skal modellen flyttes etter osv.. Derimot hvis en modell flyttes som har en parent så skal denne modellen slutte å følge etter sin parent. Den må da fjerne seg selv som Observer fra sin parent. Dette gjøres da ved å kalle parent.removeObserver(this). Det eneste bruksområde for ModelInfo::addObserver(..) er når en ny Situation opprettes. Alle nye Sitution objekter som opprettes skal om mulig være basert på en allerede eksisterende Situation. Alle modeller i den Situation som den nye Situation skal baseres på må derfor kopieres. Dette gjøres via Model3D::clone(). Denne metoden legger automatisk inn det nye Model3D objektet som en observer av det gamle via ModelInfo::addObserver(..) som også er tilgjengelig i Model3D. ModelInfo objektene fungerer både som Observer og Observable objekter.

draw() og GL-pipeline

OpenGL[2] oppretter sin egen tråd hvor den gjør alle sine operasjoner. Hvis brukeren ønsker å påvirke hva som tegnes opp og hvordan må han sørge for at det skjer innenfor denne tråden. Denne tråden kalles ofte for GL-pipeline

fordi dette er det eneste stedet der man i utgangspunktet får lov å gjøre såkalte GL-kall. Det finnes måter å få gjort disse kallene på utenfor pipeline, men dette anbefales ikke. Vår dør inn i denne pipeline går via den GLEventListener som implementeres i Scene klassen. GL-pipeline gjør kall til metodene displat(..), reshape(..) og init(..) i denne klassen. Alle kall som gjøres videre innenfor disse metodene vil derfor ligge innenfor GL-pipeline og metodene som påvirker GL-motoren kan benyttes. Det er tre ting som må gjøres innenfor pipline. Disse er:

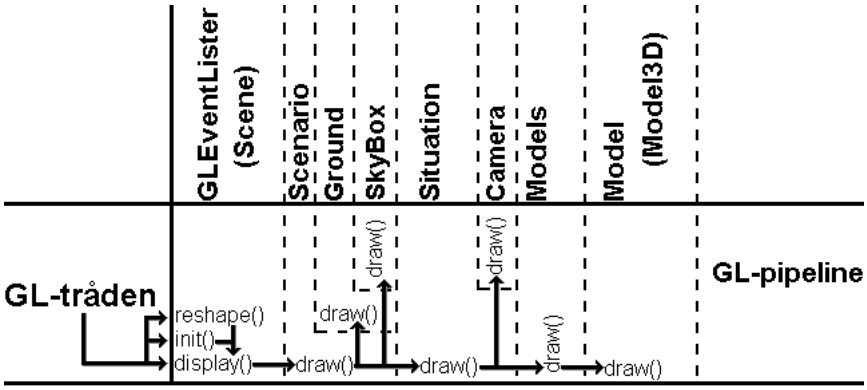
1. Tegne opp det som skal være med i scenen. Modeller, textures o.l.
2. Opprette instanser av modeller. Det vil si at lister over punktene i modellen legges inn i skjermminnet.
3. Binde teksturer til modeller.

Det er kun ved første instansiering av klasser at det er nødvendig å opprette lister og binde teksturer til disse. Den statiske klassen ModelLoader tar seg av innlesingen av modeller og sørger for at det ikke blir lagt identiske lister i skjermminnet. Den vanligste hendelsen som oppstår i GL-tråden er at metoden GLCanvas::repaint() blir kalt. Denne funksjonen oppretter da en GLEvent som blir tatt hånd om av en GLEventListener som i dette tilfellet er Scene klassen. Display(..) metoden blir da kjørt i denne klassen innenfor samme pipeline. Dette er måtene vi provoserer frem en GLEvent:

1. Kaller GLCanvas::repaint(). (display(..))
2. Instanierer et GLCanvas. (init(..) -> display(..))
3. Endrer størrelsen på et GLCanvas. (reshape(..) -> display(..))

For å holde GL-kallene innenfor pipeline skal i benyttes oss av en egen ModelEvents klasse. Denne klassen inneholder en gitt mengde konstanter som kan brukes til å angi statusen til Models klassen. Når ingen ModelEvents er satt så tegnes modellene opp på vanlig måte. Det er seks mulige hendelser som kan forekomme som påvirker opptegningen av modellene. Dette bringer totalen av ModelEvents opp i sju. Disse er:

1. NOTHING: Ingen endring. Tegn opp som normalt.
2. DELETE_MODEL: Valgt modell skal slettes og derfor ikke tegnes opp igjen.
3. NEW_MODEL: Det ligger en ny model i cache som skal lastes inn og tegnes opp.



Figur 3.13: Illustrasjon av GL-pipeline.

4. ROTATE_MODEL: Valgt modell skal tegnes opp med en annen rotasjon.
5. UPDATE_MODEL_COORDS: Valgt modell skal tegnes opp på en annen koordinat.
6. CHECK_HIT_MODEL: Det skal sjekkes om aktuelle muskoordinater samsvarer med plasseringen til en modell.
7. NEW_POINT: Et nytt punkt er lagt til på valgte modell sin rute.

Når draw(..) kjøres på gjeldende Models objekt må denne ta hensyn til hvilken modell hendelse som er satt.

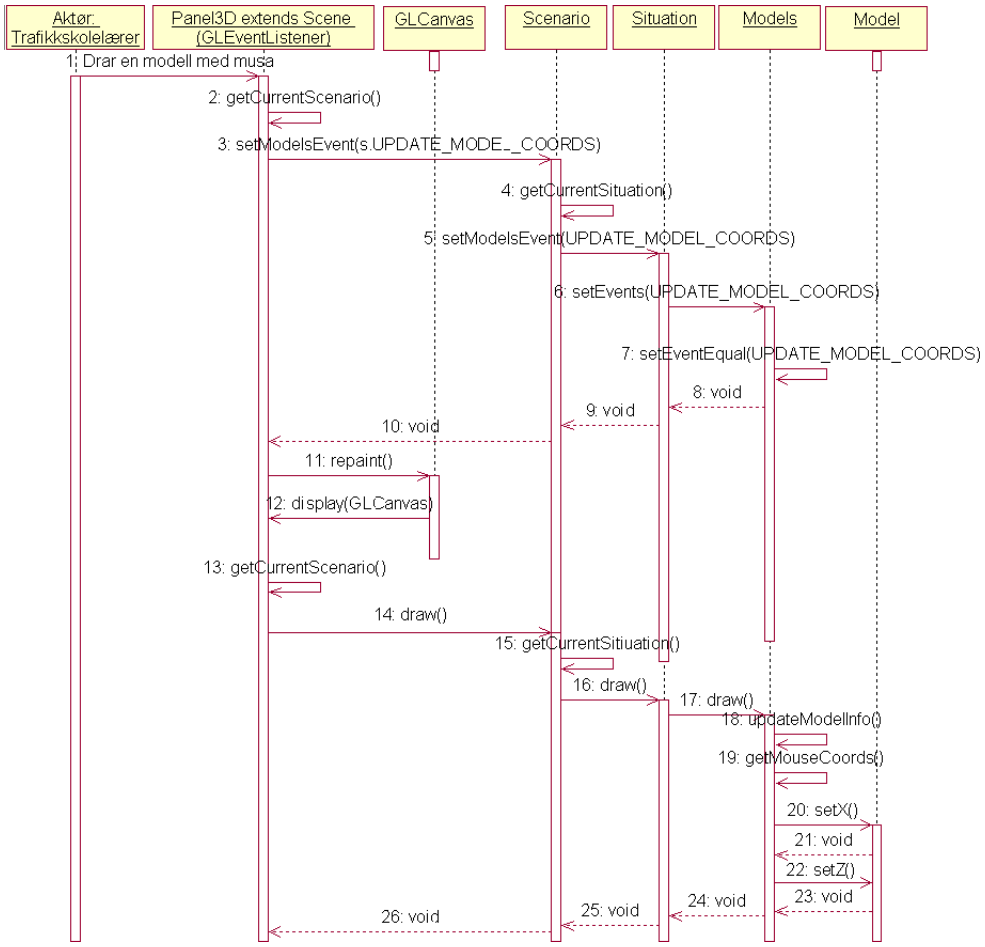
3.3.3 Prosess utseende

Her har vi valgt å ta for oss de mest innfløkte prosessene i inkrementet og beskrevet disse ved hjelp av sekvensdiagrammer.

Sekvensdiagram: Flytt modell

Forklaring til figur Flytting av en modell skjer gjennom to kallsekvenser.

1. Sette ModelEvents riktig.
2. Utføre flyttingen.



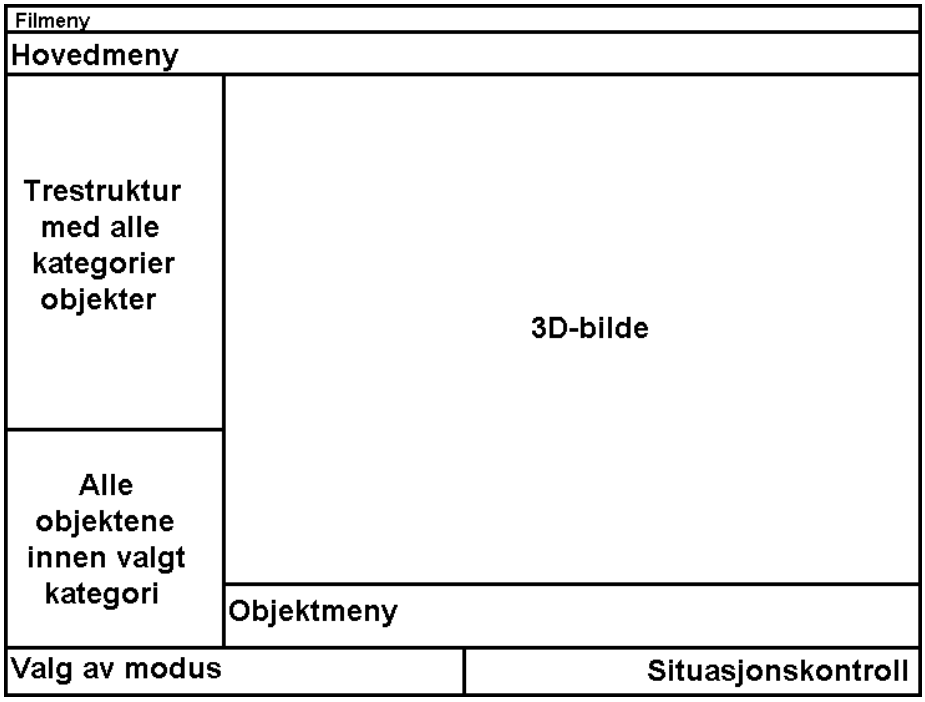
Figur 3.14: Sekvensdiagram som illustrerer prosessene som må gjennomgås for å flytte et objekt.

I den første kallsekvensen sørger for å sette riktig hendelse. Mens den andre sørger for at den blir utført og at bildet blir oppdatert. Grunnen til at det må gjøres på denne måten er beskrevet under punktet om GL-pipeline under segmentet Logisk utseende.

3.4 Inkrement 2: Brukergrensesnitt for 3D-generator

Brukergrensesnittet til 3D-generatoren må legges opp slik at det er lett å bruke. Etter samtaler med oppdragsgiver og samarbeidspartner har et ønsket utseende på grensesnittet blitt til.

3.4.1 Logisk utseende



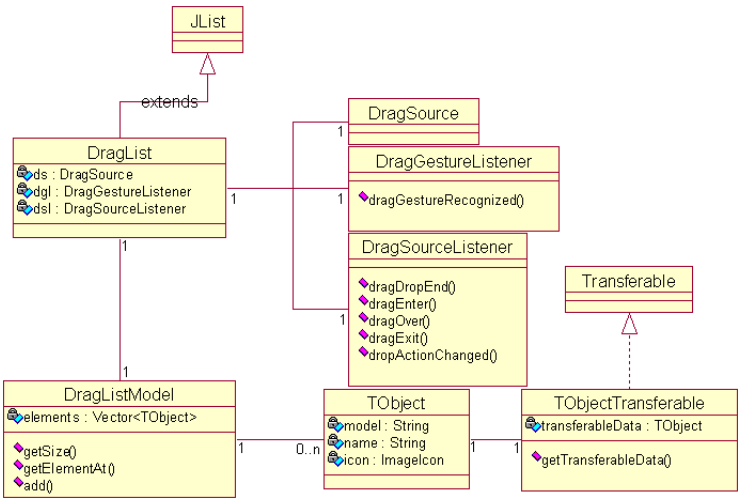
Figur 3.15: Skisse over hvordan brukergrensesnittet til 3D-generatoren skal se ut.

Trestruktur for kategorier

En trestruktur lignende den man finner over mapper i Windows Explorer skal representere alle kategoriene av objekter som kan lastes inn i 3D-generatoren. Den eneste kategorien objekter som krever en underkategori er skilter, derfor vil trestrukturen på det meste være i to nivåer. Trestrukturen skal baseres på JTree klassen som ligger i swing-biblioteket i Java. Det legges en TreeSelectionListener på nevnte JTree som bytter ut dataene i objektlista nedenfor.

Objektene innen en valgt kategori

Objektene i den kategorien som er valgt skal vises i en JList som har dra og slippsatt på. En egen struktur for listen styrer hvordan listen presenteres. TO-



Figur 3.16: Klassediagram over strukturen for lista med objekter.

bject holder på all informasjonen om et objekt. det vil si:

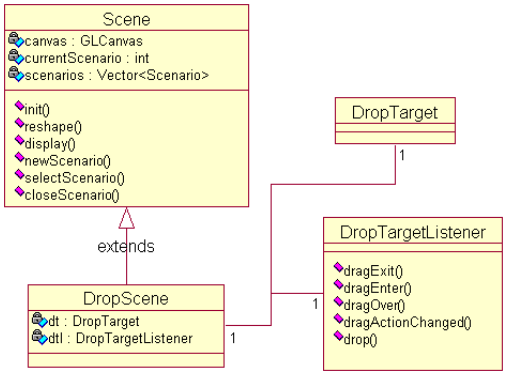
- Filsti til modellfila.
- Navn på modellen.
- Ikonrepresentasjon av modellen.
- Navnet på filene som inneholder alternative teksturer til.
- Hvorvidt piler kan tilknyttes objektet.

- Hvorvidt tekst kan tilknyttes objektet.
- Høyden på evt. tekst.
- Hvorvidt objektet er av en type som man kan se fra innsiden.

Når DragGestureListeneren merker at en dra og slipper startet oppretter den en ny TObjectTransferable som holder på det TObjectet som dra og slipp"manøveren startet på. TObjectTransferable implementerer Transferable grensesnittet som ligger i java.awt.transfer biblioteket. Objekter som implementerer dette grensesnittet kan mottas av DropTarget objekter.

3D-bildet

Utvidelsen av 3D-bildet for dette inkrementet er å implementere et grensesnitt for å motta dra og slipp"hendelser. Dette gjøres ved å opprette et DropTarget objekt som ligger på Scene objektet. Når et objekt slippes på 3D-bildet så



Figur 3.17: Klassediagram over strukturen for mottager klassen av dra og slipp hendelser.

vil det opprettes et DropEvent objekt som håndteres av DropTargetListener i DropScene. Når objektet slippes på DropScene hentes TObject ut fra TObjectTransferable og modellen som dette objektet henviser til lastes inn på de koordinatene som musepekeren i det øyeblikket henviser til. Opprettelsen av TOBjectTransferable gjøres i DragList.

Objektmenyen

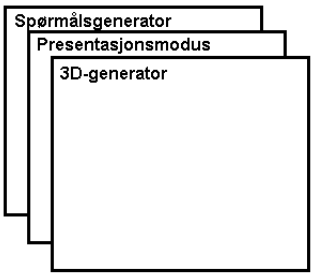
Objektmenyen skal innholde knapper for å:

- gå inn i et kjøretøy. Få kameravinkelen fra innsiden.
- slette en modell.
- rotere en modell. Både en knapp for rotering mot høyre og en for rotering mot venstre.
- opprette piler til de modellene som kan ha det. Det skal være knapper for begge typene piler.
- slette pilen fra en modell.
- bytte fargen som brukes på pil og tekst til en modell.
- sette teksten som skal knyttes til en modell.
- slette teksten som er knyttet til en modell.
- endre hvilke lys som lyser på et lys.
- sette teksten som skal vises som overskrift for scenen.
- slette nevnte overskrifttekst.
- starte opprettingen av fotgjengerfelt.
- bytte mellom OrthoCamera og det kameraet som er valgt for situasjonen.

Knappene skal ligge på en JToolbar. Plasseringen av knappene er ikke fastlåst og det er åpent for endringer hvis dette er hensiktsmessig. De knappene som ikke er relevante for valgte modell skal gråes ut. Når ingen modell er valgt skal alle knapper untatt kamera og fotgjengerknappen gråes ut. Utgråing av knapper gjøres enkelt ved å benytte seg av JButton.setEnabled(false).

Valg av modus

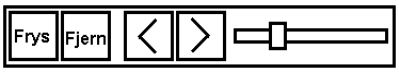
Valg av modus skal implementeres i dette inkrementet. Dette løses best ved å legge et panel for hver modus på en CardLayout. En CardLayout er en LayoutManager som kan ha flere paneler liggende oppå hverandre. Man knytter en String til hvert av panelene. Ved å kalle opp CardLayout og gi denne en bestemt String kan man lett bytte mellom kortene"i layouten. Andre forutsettninger for hvordan disse knappene opptrer vil komme i de inkrementene hvor spørmålsgeneratoren og presentasjonsmodusen blir implementert.



Figur 3.18: CardLayout hvor de forskjellige modusene ligger oppå hverandre og man kan lett bytte mellom de.

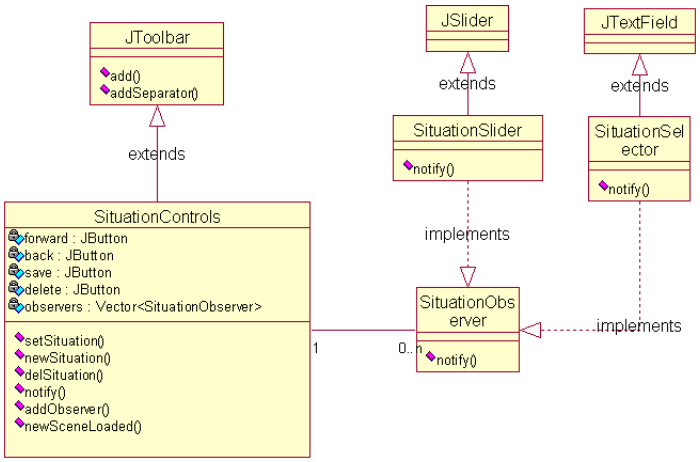
Situasjonskontroll

Dette skal være knappene som kontrollerer når man oppretter en situasjon, når man sletter dem og hvilken siutasjon i det valgte scenario som skal vises. Dette skal gjøres via knapper og en slidebar. Knappene lages enkelt av JButton



Figur 3.19: Illustrasjon av hvordan situasjonskontrollknappene skal se ut.

objekter og slidebaren lages av et JSlideBar objekt. Disse objektene skal det bygges en wrapper klasse kalt SituationControls rundt. For å holde orden på



Figur 3.20: Klassediagram som viser strukturen i situasjonskontrollen.

hendelser som påvirker hvordan situasjonkontrollene skal se ut oppretter vi et grensesnitt for å observere SituationControls objekter. Dette grensesnittet kaller vi SituationObserver. De klasser som implementerer dette grensesnittet må implementere en metode som heter notify(..) som tar imot parametere fra SituationControls når en hendelse opprinner. Vi lager en egen klasse for slideren som utvider den ferdige JSlider som ligger i java.swing biblioteket som implementerer dette grensesnittet. Funksjoner for å holde datastrukturen i orden legges til i SituationControls og hver gang den endres blir alle SituationObserver objektene oppdatert via notify(..) funksjonen. Knappene for å fryse og fjerne situasjoner kaller bare opp funksjoner som er skrevet ferdig i inkrement 1. Det samme gjelder hva som skjer når man bytter situasjon. Da kalles bare funksjonene som setter riktig situasjon opp med situasjonsnummeret som parameter.

Hovedmeny

Hovedmenyen inneholder knappene for å:

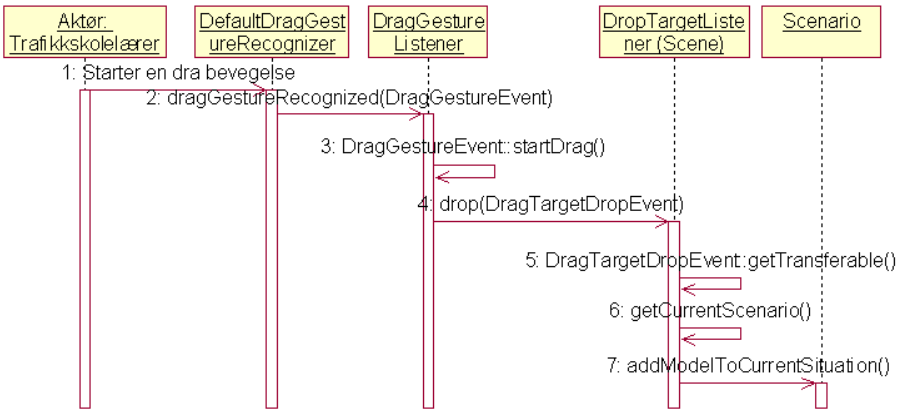
- opprette nye scenarier.
- lagre scenarier.
- åpne scenarier.
- velge scenario.
- lukke scenario.

Det finnes to filformater som kan leses inn i Kos. Et er et filformat for lagring av enkeltstående scenarier. Det andre er pakker med flere scenarier samlet. Knappen for å åpne disse er den samme. Det eneste man behøver å gjøre er å sjekke om første node i xml-fila er starten på en pakkefil eller en vanlig scenariofil.
Man kan lagre på tre måter. Er pakkemodus valget satt så lagres alle scenarier i en pakke. Er pakkemodus deaktivert lagrer samme knapp et og et scenario. Da dukker også en knapp for å lagre alle scenarier fortløpende opp. De forskjellige filformatene finnes som vedlegg.

3.4.2 Prosess utseende

Dra og slipp

DefaultDragGestureRecognizer er en standard klasse for å kjenne igjen bevegelsene som kjennetegner en dra og slipp"manøver. Denne klassen lar så bruk-

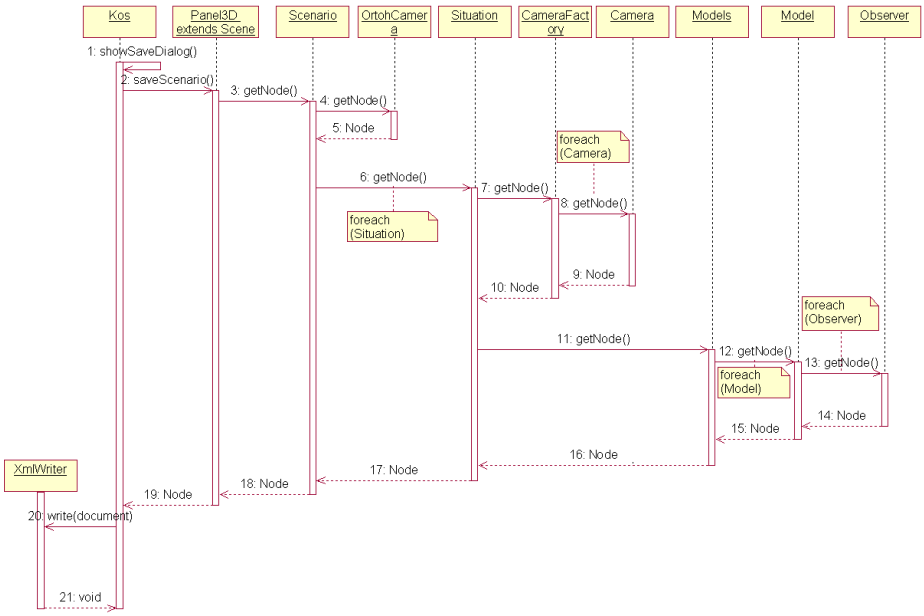


Figur 3.21: Fremdriften i en dra og slipp kommando frem til Scenario.

erens egen DragGestureListener vite at dra og slipper startet. Så skal DragGestureListener opprette et TObjectTransferable objekt og tilegne det et TObject. Når manøveren fullføres mottas TObjectTransferable av DropTargetListener i DropScene. Nærmere beskrivelse av klassene finnes under 3.4.1.

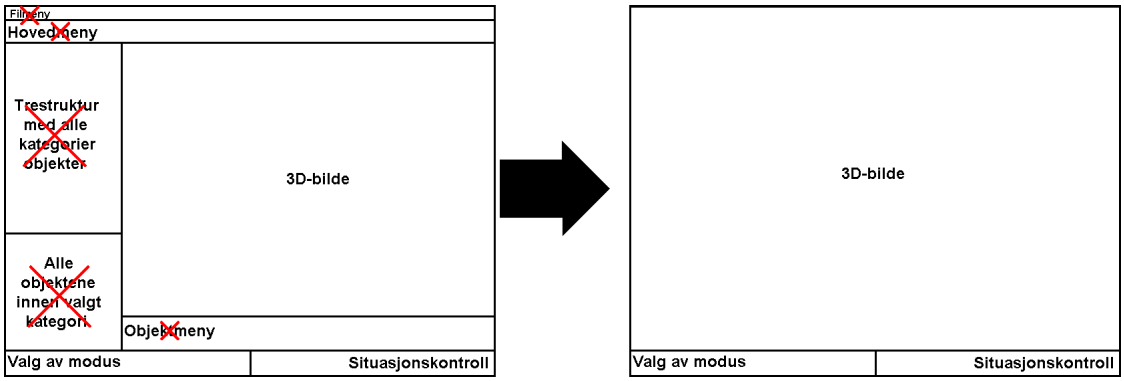
Lagre scenario

Scenarier lagres i XML-filer etter filformatet definert i punkt C.1. I Kos objektet opprettes et DOM-tre som Nodene som representerer alle objektene i Scenariet skal legges til på. Alle objekter som skal legges til i treet har selv ansvar for å opprette en Node med sin informasjon i. Det kallende objektet legger til den returnerte noden som sitt barn i DOM-treet. Når DOM-treet er bygd opp kalles en statisk funksjon hos XmlWriter klassen opp for å få skrevet treet til fil. javax.xml.transform biblioteket har klasser for å omforme et DOM-tre til en tekststreng som deretter kan skrives til fil.



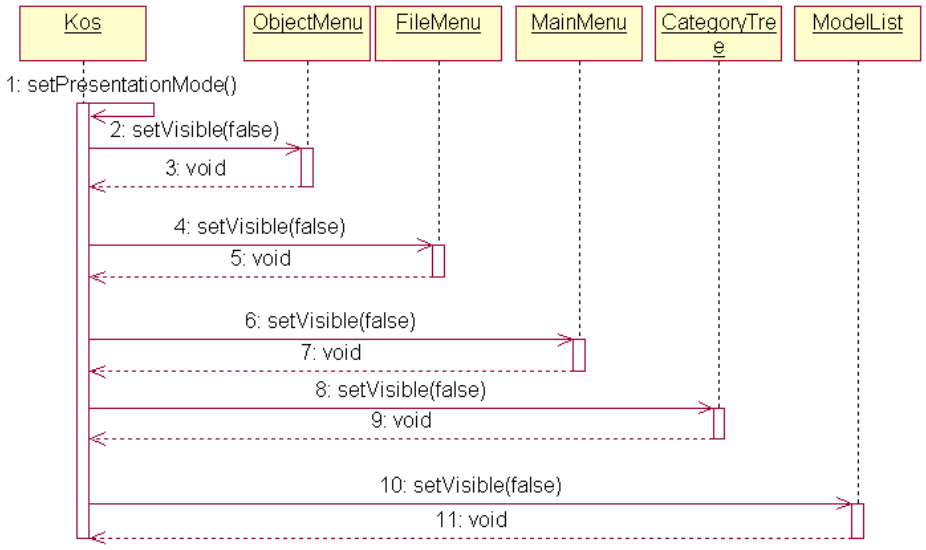
Figur 3.22: Forenkler kallsekvens når et scenario skal lagres.

3.5 Inkrement 3: Presentasjonsmodus



Figur 3.23: Slik skal man gå over i presentasjonsmodus.

Realisering av inkrement 3 kan gjøres veldig lett. Alle brukergrensesnittkomponenter basert på Component-klassen kan skjules ved å kalle funksjonen Component::setVisible(..) med "false" som parameter.

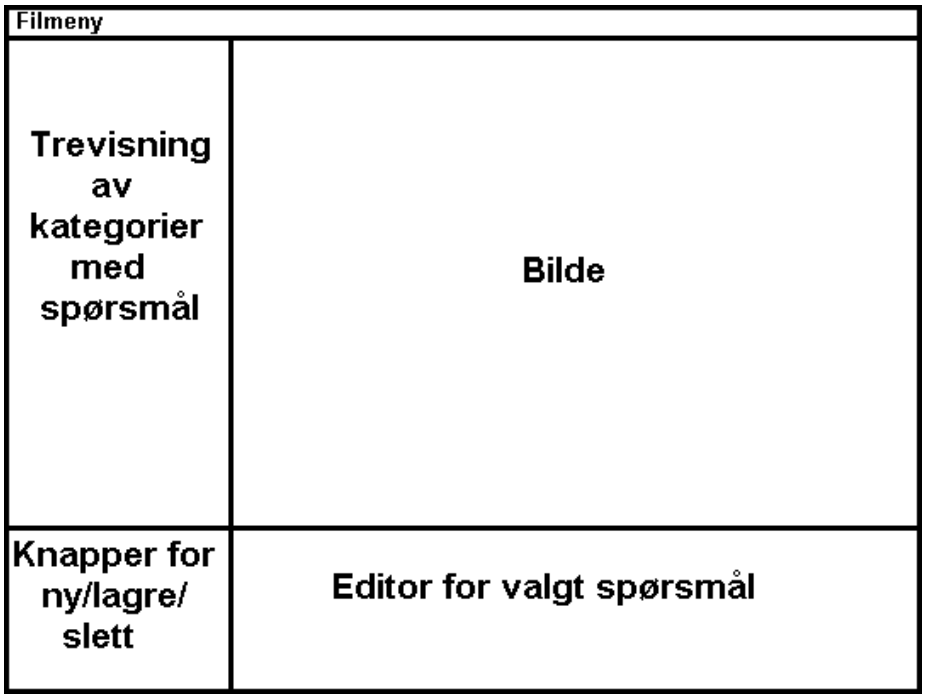


Figur 3.24: Illustrasjon av kall sekvensen for å starte presentasjonsmodusen.

3.6 Inkrement 4: Spørsmålsgenerator

Spørsmålgeneratoren skal bestå av fem hovedkomponenter:

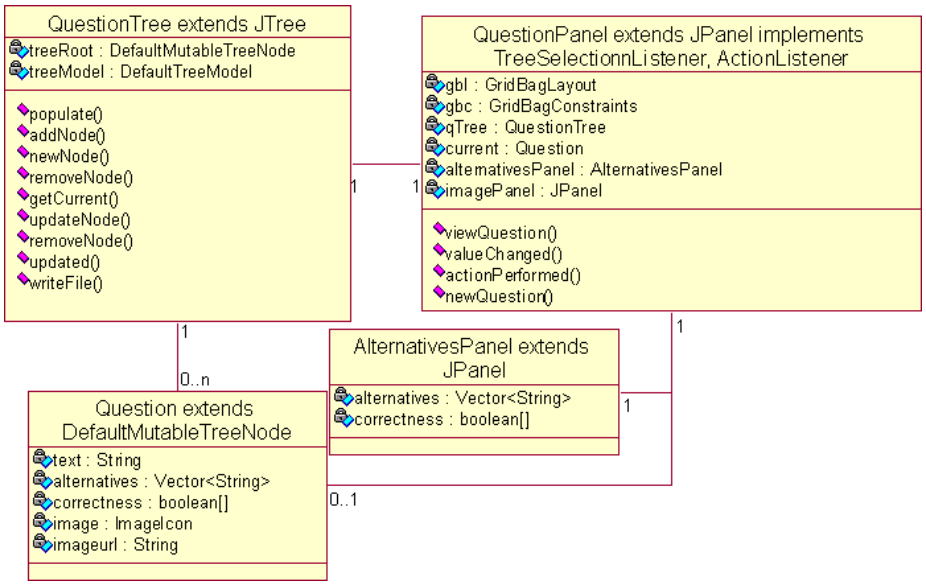
1. Treet som inneholder alle spørsmålene.
2. Tekstfeltene hvor man kan redigere spørsmålstekst og alternativer.
3. Panelet som kan inneholde bildet knyttet til spørsmålet.
4. Panelet som inneholder knappene som påvirker spørmålene.
5. Datastrukturen som inneholder spørsmålene.



Figur 3.25: Skisse over oppdelingen av det grafiske brukergrensesnittet for spørsmålsgeneratoren.

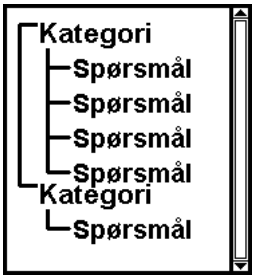
3.6.1 Logisk utseende

Spørsmålsgeneratoren blir oppdelt i fire klasser. Hovedklassen QuestionPanel inneholder alle de grafiske komponentene som ikke krever særlig utvidelse. Mens vi lager egne komponenter for visning av treet og alternativene.



Figur 3.26: Overordnet klassediagram over spørsmålsgeneratoren.

QuestionTree



Figur 3.27: Skisse over hvordan treet som inneholder spørsmålene skal se ut.

QuestionTree klassen utvider bibliotekklassen javax.swing.JTree. Ny funksjonalitet som legges til skal sørge for at treet skal kunne holde på datastrukturen. Funksjoner for å redigere datastrukturen må legges til. Kort beskrivelse av de viktigste funksjonene i QuestionTree:

- populate(String filename): Fyller datastrukturen med de data som ligger i filen angitt ved paramenteren filename.
- addNode(Question node, DefaultMutableTreeNode parent): Legger node inn under parent i datastrukturen og treet.

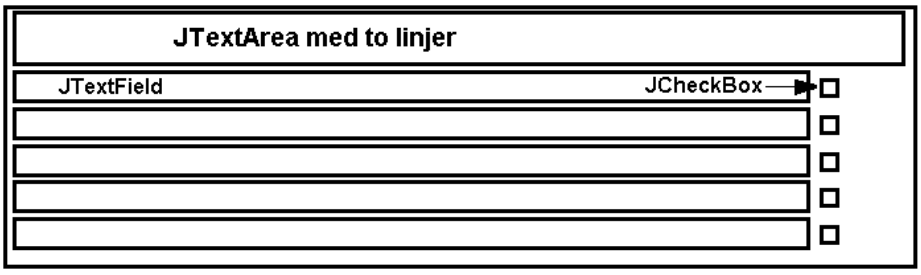
- newNode(): Oppretter et nytt Question objekt som legges inn i treet og returneres til kalleren.
- removeNode(): Fjerner valgte Question objekt.
- removeNode(Question q): Fjerner det Question objektet angitt ved q.
- getCurrent(): Returnerer det Question objektet som er valgt.
- updateNode(String category, String text, String alternatives[], boolean correctness[]): Legger dataene som er sendt med som parametere inn i valgte Question objekt.
- updated(): Oppdaterer visningen av treet.
- writeFile(String path): Skriver spørsmål og bilder til fil.

QuestionPanel

QuestionPanel er hovedpanelet for spørsmålsgeneratoren. Det er denne som styrer det hele. QuestionPanel implementerer to typer listenere. TreeSelectionListener for å oppdatere innholdet i de andre panelene når noe bli valgt i treet og ActionListener for å håndtere hendelser fra knappene på panelet.

AlternativesPanel

AlternativesPanel skal inneholde et JTextArea til redigering av spørsmålsgtekst og fem rader med plass til alternativer. Hvert alternativ er sammensatt av et JTextField og en JCheckBox. Tekstfeltet skal inneholde alternativet, mens boksen skal ta vare på om dette alternativet er riktig eller ei.



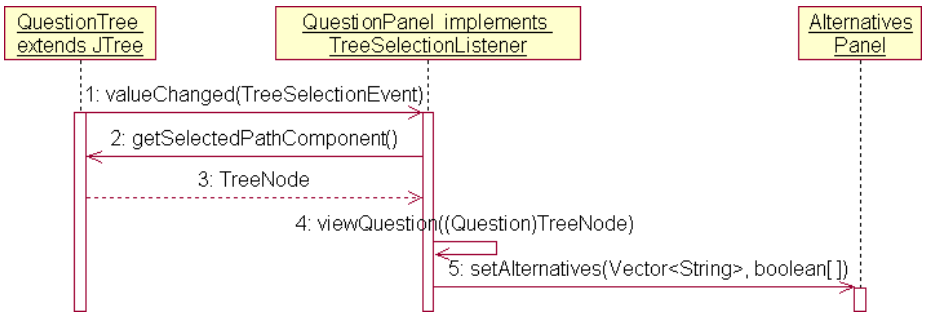
Figur 3.28: Skisse over hvordan alternativ panelet som inneholder spørsmålene skal se ut.

Question

Selve spørsmålet lagres i Question objekter. Question utvider klassen Default-MutableTreeNode som er Java sin standard trenode. Dette gjør at alle Question objektene kan bli lagret direkte i treet isteden for å ha en egen datastruktur i tillegg. På denne måten unngår vi redundans av informasjon i minnet.

3.6.2 Prosess utseende

Valgt spørsmål

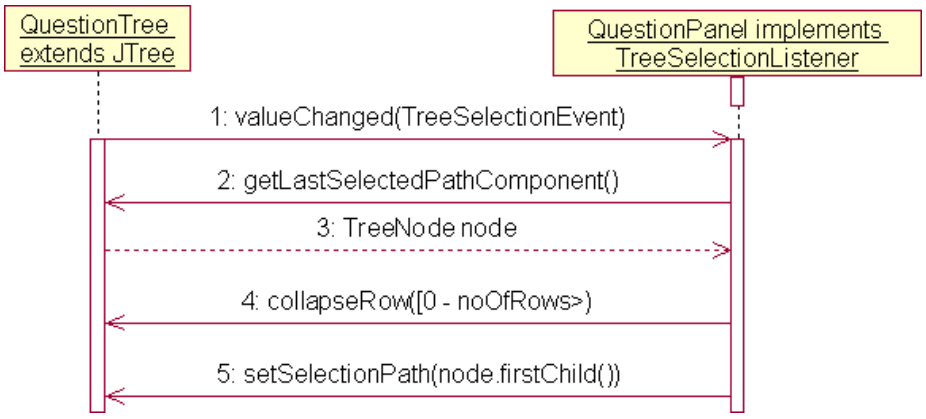


Figur 3.29: Kallsekvensen når et nytt spørsmål blir valgt.

Når et spørsmål blir valgt i QuestionTree så kalles metoden valueChanged(..) på alle TreeListener objekter som er tilknyttet dette treet. I vårt tilfelle er dette QuestionPanel. QuestionPanel ber deretter QuestionTree om å returenere det siste valgte objektet, som i dette tilfellet er det Question objektet som skal vises frem. Deretter hentes den aktuelle informasjonen ut av Question objektet og vises frem på de andre panelene i QuestionPanel.

Valgt kategori

Når en kategori i QuestionTree blir valgt så kalles QuestionPanel::valueChanged() på samme måte som i 3.6.2, men når det objektet som velges er en kategori så kollapses først hele treet før nodens første barn finnes frem og valgt objekt flyttes til denne.



Figur 3.30: Kallsekvensen når en kategori blir valgt.

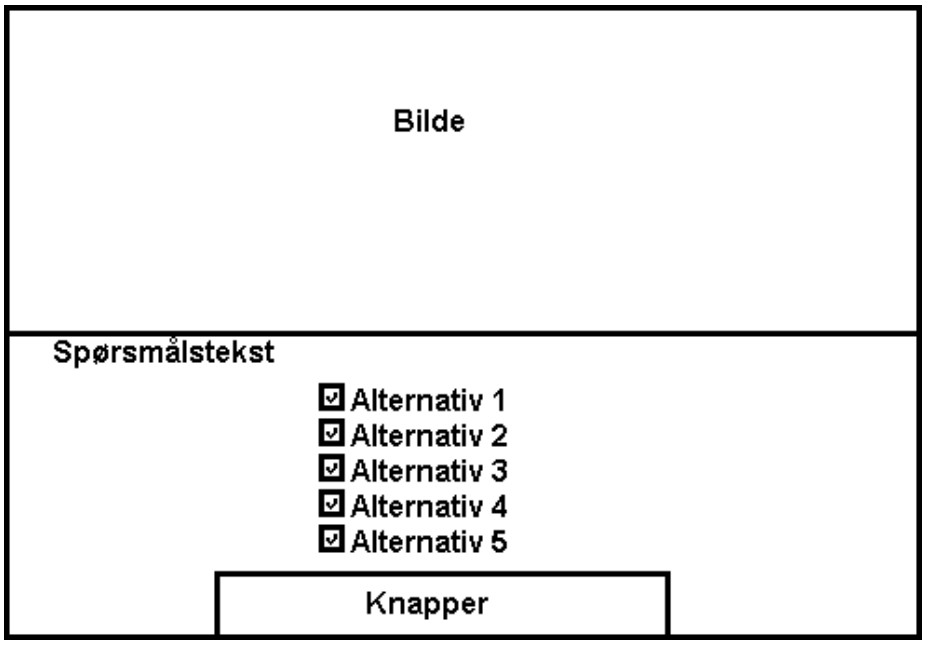
3.7 Inkrement 5: Prøve

Prøvemodulen skal være helt uavhengig av kode i reste av KOS. Den skal derfor pakkes i en separat JAR fil. Installasjonen av testmodulen må likevel ligge i samme mappe som KOS fordi den er avhengig av å ha tilgang til filene generert av KOS sin spørsmålsgenerator. Spørsmålsgeneratoren har tre seksjoner:

1. Oppsett av prøven med valg av hvilke kategorier som skal være med og antallet spørsmål.
2. Visning av prøven hvor man skal svare på spørsmålene.
3. Oppsummering av prøven med visning av antallet riktige og gale svar, samt en statistisk fremstilling av svarene.

Når kategori og antall spørsmål er valgt hentes alle spørsmålene ut fra de aktuelle kategoriene. Blant disse spørsmålene velges det antall spørsmål som skal inn i prøven ut tilfeldig. Store kategorier vil på denne måten bli overrepresentert i prøvene. Med den forutsetningen av at viktige spørsmål stilles oftere så er ikke dette noen ulempe.
Seksjon 2 skal også brukes etter at prøven er ferdig besvart for å vise hvilke svar som var gale, og hvilke som var riktige. Når man går gjennom prøven etterpå er skal alternativer som er riktig avmerket merkes med grønn bakgrunn og feil avmerking belønnes med rød bakgrunn.
I seksjon 3 skal det i hvertfall vises frem denne statistikken:

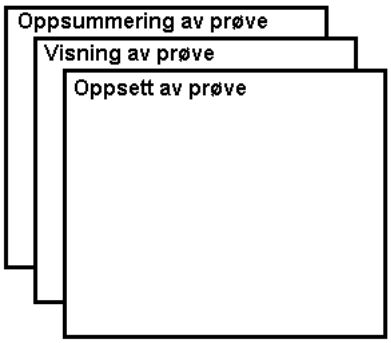
- Antall spørmål.



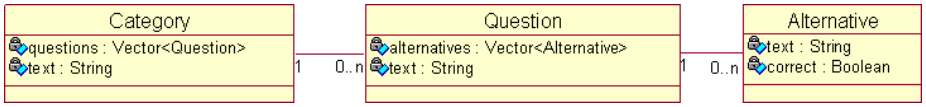
Figur 3.31: Skisse av brukergrensesnittet til testmodulen når den viser et spørsmål

- Antall mulige svar.
- Antall riktige svar.
- Prosentfordeling av riktige svar.

De tre forskjellige seksjonene i dette programmet skal legges på hvert sitt JPanel som igjen legges i en CardLayout. Innlesing av spørsmålene gjøres fra samme xml.fil som den spørsmålsgeneratoren til Kos skriver til. Formatet for denne fila ligger under C.2. Datastrukturen for spørsmålene i minnet bygges opp på denne måten:



Figur 3.32: Skisse av layout oppsettet i testmodulen.



Figur 3.33: Datastrukturen som holder på alle spørsmålene i testmodulen.

Hver av disse klassene skal i tillegg ha en konstruktor som kan lese inn alle sine data fra et DOM-tre som baserer seg på den xml-fila som er lest inn fra disk.

Kapittel 4

Implementasjon

4.1 Overordnet

4.1.1 Valg av teknologi

Vi har benyttet programmeringspråket Java som plattform for applikasjonen. Det eksisterer mange gode alternativer til Java som tilbyr mye av det samme, men gruppen i sin helhet har bakgrunn i programutvikling i Java så ved å velge dette så slapp vi å bruke tid på opplæring. Java sitt enorme klassebibliotek har også sørget for at vi har kunnet bruke mindre tid på å utvikle grafiske brukergrensesnitt og heller fokusere på de områder vi hadde mindre kompetanse. Alternative valg kunne vært C# .NET for samme solide klassebiblioteket, eller C/C++ for en sterk ytelse.

Rammeverket for selve 3D-utviklingen falt på OpenGL som er et portabelt lavnivå-API som fokuserer utelukkende på grafikk. Her hadde vi ett alternativ, nemlig DirectX, men når en av gruppens medlemmer hadde grunnleggende kunnskaper i OpenGL, samt at OpenGL er et modent API som er å betrakte som svært stabilt (lite endringer har blitt gjort) så falt valget på OpenGL. Ved ren 3D-utvikling spiller det i grunn svært liten rolle om man benytter OpenGL eller DirectX, forutenom at DirectX er svært bundet til Microsoft Windows.

For å kunne benytte OpenGL direkte i Java falt valget på JOGL (Java OpenGL) ettersom det kun bryr seg om ren OpenGL i Java og utelater alt unødvendig.

Når det gjelder filformater som er valgt så kan vi nevne følgende:

- Milkshape3D (ms3d) ble valgt som filformat for å representere de trafikale objektene. Dette på grunn av den informasjon som eksisterer i filformatet, og at dette formatet er spesielt egnet for enkle (få polygoner) og lite krevende objekter. Dette formatet med varianter er brukt i mange av de mest kjente moderne dataspillene.
- Portable Network Graphics (png) ble valgt som filformat for å representere spørsmålsbilder pga sin støtte i java sitt klassebibliotek og i andre applikasjoner, og for sine gode egenskaper. Filene blir små i størrelse, og bildekvaliteten blir topp.
- Extensible Markup Language (xml) ble valgt som filformat for å representere trafikale situasjoner og spørsmål. Dette fordi det er et portabelt og godt utbredt format som enkelt kan åpnes på tvers av applikasjoner. Java har også overlegne egenskaper for å arbeide mot slike filer.

4.1.2 Valg av utviklingsverktøy

Når vi skulle utforme trafikale objekter falt valget på Milkshape3D og Blender (med tilleggsmoduler). Milkshape3D var det naturlige valget da det er enkelt å lære, billig, og er det programmet som Milkshape3D filformatet er lagd for. Blender er et litt mer avansert verktøy og dermed er det også vanskeligere i bruk. Sistnevnte ble foretrukket av deler av gruppen basert på eksisterende kompetanse.

For vanlige grafiske operasjoner i forbindelse med 2D brukte vi stort sett The Gimp. The Gimp er et gratis, men svært kraftig verktøy som er et alternativ til litt mer kjente Adobe Photoshop. Også her ble valget basert på eksisterende kompetanse innad i gruppen.

Ved selve utviklingen av applikasjonen, dvs programmeringen, benyttet gruppen i sin helhet Vim. Vim er en gratis, men kraftig, teksteditor. Vim har sin styrke i mange tastatursnarveier, syntax highlighting, og ellers et stort sett nyttig funksjonalitet. Gruppen i sin helhet har tidligere kompetanse i bruk av Vim så det var aldri hensiktsmessig å bruke ekstra tid på å lære noe annet.

For rapportskriving benyttes Latex. Dette fordi det gjør det mye lettere å få oversikt i større dokumenter, og man trenger ikke bry seg om formatering da dette skjer automatisk. Fordelen ved Latex fremfor OpenOffice og Microsoft Office er at man ikke trenger å tenke på formatering av dokumentet og at

versjonskontrollsystemet selv tar seg av konflikter når flere jobber på samme dokument.

Vi har også benyttet Subversion for versjonskontroll av hele prosjektet. Dette inkluderer både kildekode og dokumentasjon. Skolen har under hele prosjektet stilt med Subversion-tilgang for prosjektgruppene, og gruppen i sin helhet har tidligere brukt nettopp Subversion på andre prosjekter. Det var derfor naturlig for oss å benytte oss av dette tilbudet fremfor å sette opp eget versjonskontrollsystem.

Ytterligere informasjon om utviklingsmiljø og verktøy finnes i vedlegg D.

4.1.3 Bruk av andres arbeid

I dette prosjektet har vi benyttet noe Java-kildekode og grafiske ressurser basert på andres arbeid.

I faget programutvikling høsten 2005 benyttet vi oss av parse-rutiner for XML og internasjonaliseringsbibliotek skrevet av faglærer Øyvind Kolloen. Vi har fått tillatelse til å benytte denne kildekoden også i hovedprosjektet. Kildekoden er skrevet om for å passe vårt behov, men det er fremdeles spor av den opprinnelige koden. Kildekoden ligger i filene XmlReader.java og Babelfish.java.

Vi har også benyttet en ferdig ikon-pakke som vi også brukte i faget programutvikling høsten 2005. Disse ikonene er lisensiert som gratis så lenge man krediterer den opprinnelige opphavsmannen. Noen av ikonene er modifisert av gruppen, men mange er som de eksisterer i den opprinnelige ikon-pakken. Nedenfor følger copyright-notisen til ikonene brukt i inkrementet:

Icons Copyright(C) 1998 by Dean S. Jones dean@gallant.com, www.gallant.com/icons.htm

4.2 Inkrement 1: 3D-generator.

4.2.1 Introduksjon

Inkrement 1 har vært en veldig stor del av prosjektet, det har inneholdt både innlesing av modeller samt alle translasjoner på objekter inne i selve scenen. Det har og tatt seg av slike ting som lyssetting og tåke. Det inneholder og kode som de andre inkrementene bruker for å aksessere forskjellige funksjoner i dette

inkrementet.

4.2.2 Porting og omskrivninger

Da vi startet på Kos hadde vi ikke noen kode for å kunne laste inn modeller fra fil. Vi var på leting etter noen som støttet våre krav til modeller slikt som teksturer. Vi undersøkte derfor videre og kom til slutt over kode for dette, men det var for et annet bibliotek som heter lwjgl. For å få denne til å kunne fungere måtte vi porte/skrive den om til jogl. Vi ble selv nødt til å skrive den om slik at den brukte display lister, slik at den skulle bli optimalisert for å kunne fylle de kravene vi hadde satt til ytelse.

4.2.3 Forklaring av implementasjon av avhengige situasjoner for modeller

For å få til avhengige situasjoner ble vi nødt til å skrive om store deler av koden, da dette ble bestemt at skulle være med i Kos . Det som ble den mest hensiktsfulle metoden for å få dette implementert var ved bruk av observers. For mer informasjon om observers se 3.1.2. Når en ny situasjon da blir opprettet, vil alle objektene kopieres inn til den nye situasjonen, og hvert objekt i den forrige situasjonen vil få ut referanser til objektene i den nye situasjonen. Når det så skjer en endring vil man bruke observer objektet til å si fra til de andre objektene at det har skjedd en endring ved Observable.hasChanged(). Neste gang objektene blir tegnet opp vil de da få beskjed om at en endring har skjedd, og de sjekker da opp hva som er endret og utfører den endringen som har skjedd.

4.2.4 Kodeeksempler som er relevante

Matematiske funksjoner

Grunnen til at vi valgte og legge ved noe av koden fra matematikk klassen som vi har laget, er nettopp fordi det er av høy relevanse til inkrement 1. De funksjonene som vi har valgt og liste opp er funksjoner som har blitt ofte brukt, både når det gjelder utregning av rotasjoner av objekter, samt utregninger for objektstier.

```
/**
```

```
 * Denne funksjonen henter ut gradene mellom
 * en vektor ut i fra x aksen
 * @param vVec som er en float[] som inneholder
 * float verdiene for vektoren
 * @return som er antall grader
 */
public static float getRotation2xf(float vVec[])
{
/* Setter her opp en vektor langs x-aksen for å
 * ha et utgangspunkt for å hente ut antall grader.*/
float uVec[] = {1.0f,0.0f};

//Normaliserer de to vektorene
float vVecL = (float)Math.sqrt(
      (double)((vVec[0]*vVec[0])+(double)(vVec[1]*vVec[1])));
float uVecL = (float)Math.sqrt(
      (double)((uVec[0]*uVec[0])+(double)(uVec[1]*uVec[1])));

//Henter ut prikkproduktet for de to vektorene
float dotProd = (vVec[0]*uVec[0])+(vVec[1]*uVec[1]);
//Beregner her antall grader mellom de
float deg = (float)Math.toDegrees(Math.acos(
            (double)(dotProd/(vVecL*uVecL))));
//Sjekker her om antall grader er negative eller ikke
return vVec[1] <= 0 ? deg : -deg;
}

/**
 * Denne funksjonen returnerer de nye punktene
 * for en vektor etter at den har blitt rotert.
 * @param rot som er en float som holder antall
 * grader som det skal roteres.
 * @param vVec[] som er en float som er vektoren
 * som skal roteres.
 * @return float[] som er den nye roterte vektoren.
 */
public static float[] getPoint(float rot, float vVec[])
{
//Bruker her en vanlig matrise rotasjons translasjon
return new float[]{(vVec[0]*(float)
       (Math.cos(Math.toRadians((double)rot))))-(vVec[1]*
(float)(Math.sin(Math.toRadians((double)rot)))),(vVec[0]*(float)
(Math.sin(Math.toRadians((double)rot))))+(vVec[1]*(float)
(Math.cos(Math.toRadians((double)rot))))};
}
```

Kode for å håndtere modell events

Denne koden brukes av forskjellige inkrementer for å bestemme hva som skal utføres av transformasjoner på et objekt. De inkrementene som vil bruke det setter en event, og setter de nye musekoordinatene hvis det er nødvendig. Deretter vil denne funksjonen ta seg av å utføre de riktige transformasjonene ut i fra event.

```
/**
   * Denne funksjonen tar seg av alt som har med
   * å tegne modellene og slette de og sjekke
   * om de har blitt trykt på.
   * @param float som er skaleringen på modellene.
   * @return int[] som brukes for å sette musen til
   * punktenes nullpunkt, den
   * returnerer x og y koordinatene i skjermkoordinater
   * til modellenes
   * nullpunkt, returnerer null hvis en modell ikke
   * har blitt truffet.
   */
  public int[] draw(float size)
  {
    int  robotCoords[] = null;
    Model model = null;
      switch(event)
      {
        case ModelEvents.NOTHING:
          break;
        case ModelEvents.DELETE_MODEL:
          if(changeColor > 0)
          {
             /*Fjerner her modellen */
              model = models.remove(changeColor-1);
             /* Må her sjekke om objektet skal
              * fjernes helt fra minnet. Grunnen til det
              * er at modeller legges inn i skjermkort
              * minnet ved hjelp av noe som heter
              * displaylister. Men hvis modellen ikke
              * skal brukes igjen, må de fjernes
              * igjen i tilfelle den som bruker
              * programmet har veldig lite skjermkortminne.
              * Det som skjer hvis det blir fylt,
              * er at de objekt listene blir
              * lagret i original minnet som er noe
              * tregere når det gjelder rendrering
```

```
              * av objekter. */
if(model instanceof Model3D)
deleteLists();
              changeColor = 0;
           }
           event = ModelEvents.NOTHING;
selectState = true;
           break;
         case ModelEvents.NEW_MODEL:
           while(cache.size() > 0)
           {
            /* Henter ut og fjerner den første
             * modellen fra et cache. */
             model = cache.remove(0);
            /* Henter her ut en modell hvis modellen
             * er av en type som er laget
             * med et modellerings verktøy */
             if(model != null)
             {
if(model instanceof Model3D)
              model = ModelLoader.getModel(gl, (Model3D)model);
              /* Initisierer modellen */
model.init(gl,canvas);
              Point p = model.getMouseLocation();
             /* Hvis modellen er av typen som har blitt lest
              * inn fra fil så skal x og y koordinatene
              * beregnes på en annen måte. */
              if(model instanceof Model3D)
              {
                DoubleBuffer b = MouseLocation.getMouseCoords(
                    gl, (int)p.getX(), (int)p.getY());
                float x = (float)b.get();
                b.get();
                float z = (float)b.get();
               /*Setter koordinatene til modellen. */
              model.setXZ(x, z);
              }
else
{
 DoubleBuffer b = MouseLocation.getMouseCoords(
                    gl, (int)mousePoint.getX(), (int)mousePoint.getY());
                float x = (float)b.get();
                b.get();
                float z = (float)b.get();
               /* Setter koordinatene til modellen. */
              model.setXZ(x, z);
```

```
}
              models.add(model);
            }
            changeColor = models.size();
selectState = true;
          }
          event = ModelEvents.NOTHING;
          break;
        case ModelEvents.ROTATE_MODEL:
          if(changeColor > 0)
          {
//Henter her ut modellen som skal roteres
            model = models.get(changeColor-1);
float modelCoords[] = new float[]{model.getX(),model.getZ()};
//Henter ut muse koordinatene på canvaset og gjør de om til
//koordinater i scenen.
            DoubleBuffer mouseCoords = MouseLocation.getMouseCoords(gl,
            (int)mousePoint.getX(), (int)mousePoint.getY());
            //Setter rotasjon på modellen
model.setRot(GlVecMath.getRotation2yf(new float[]{
(float)(modelCoords[0]-mouseCoords.get(0)),
(float)(modelCoords[1]-mouseCoords.get(2))}));
//Tegner opp linjen som viser rotasjonen
            gl.glPushMatrix();
            gl.glBegin(GL.GL_LINES);
              gl.glColor3f(0.0f,1.0f,0.0f);
              gl.glVertex3f(modelCoords[0],0.1f,modelCoords[1]);
              gl.glVertex3f((float)mouseCoords.get(0),0.1f,
              (float)mouseCoords.get(2));
gl.glColor3f(1.0f,1.0f,1.0f);
            gl.glEnd();
            gl.glPopMatrix();
          }
          event = ModelEvents.NOTHING;
          break;
        case ModelEvents.CHECK_HIT_MODEL:
/* Får her ut koordinatene som musen skal settes
 * til, er null hvis en modell ikke har blitt
 * trykket på, hvis ikke så returneres nullpunktet
 * til modellen.
 * Det den checkHit funksjonen gjør er å legge alle
 * modellene inn i et buffer, så bruker vi opengl
 * sin innebygde selection metode for å sjekke om
 * en modell har blitt truffet. Man vil da få tilbake
 * et resultat som holder informasjon om hvilken
 * av de modellene man har lagt inn i bufferet som
```

```
 * har blitt truffet, samt avstanden de har fra
 * viewPoint(dvs kamera). Så gjør vi diverse
 * utregninger for å finne nullpunktet til modellen
 * for så å returnere det. */
            robotCoords = checkHit(size);
          /* Må kjøre canvas.display, kjører det i en egen tråd. */
           Thread t = new Thread(this);
           t.start();
           event = ModelEvents.NOTHING;
break;
         case ModelEvents.UPDATE_MODEL_COORDS:
          /* Her kaller vi en funksjon som gjør om
           * skjermkoordinater som har blitt lagt
           * inn om til koordinater, og setter
           * de til gjeldende valgte modell. */
           updateModelInfo();
           event = ModelEvents.NOTHING;
           break;
      }
     /* Denne funksjonen tegner opp modellene,
      * false sier bare om modellene skal bli
      * lagt inn i bufferet som brukes av
      * selection metoden til opengl. /*
    drawModels(size,false);
   /*Returnerer de koordinatene som musen skal
    * settes til, null hvis musen ikke skal
    * flytte seg. */
    return robotCoords;
}
```

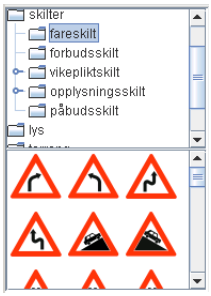
4.3 Inkrement 2: Brukergrensesnitt for 3D-generator.

4.3.1 Introduksjon

Inkrement 2 bød på nye utfordringer i forhold til inkrement 1. Implementasjonen av GUI sammen med 3D-generatoren var ikke uproblematisk. Samtidig måtte deler av 3D-generatoren skrives om for å tilpasses interaksjon med grensesnittet. I denne beskrivelsen av implementasjonen av inkrement 2 blir spesiellt viktige deler av inkrementet trukket frem. I tillegg vil kodesnutter som med sin egenart er interessante bli vist og beskrevet. Til slutt vil problemer og løsningene på disse problemene bli drøftet.

4.3.2 Viktige deler av inkrementet

Det er en del av dette inkrementet som peker seg ut som den desidert viktigste delen. Det er listen over de modellene som kan trekkes inn i 3D-generatoren. Implementasjonen av dra og slipp er den viktigste brukervennlighetsmessige funksjonaliteten i programmet. Hele menyen for valg av objekter er bygd opp som beskrevet i designdokumentet pkt. 3.4. Denne ble tilslutt seende slik ut:



Figur 4.1: Endelig utseende til menyen med objektene.

Når man drar et element ut av den listen over objektene som er nederst vil denne bevegelsen bli gjenkjent. Da kalles denne kodesnutten (Hentet fra DragList.java.):

```
/**
 * Kalles når en dra bevegelse blir gjenkjent
 */
  public void dragGestureRecognized(DragGestureEvent e)
  {
   /* Klippet ut uvesentlig kode */

   /* Henter ut det som ligger på den plassen i
    * listen som bevegelsen startet
    */
    Object val = DragList.this.getSelectedValue();
   /* Hvis ingenting var valgt: avslutt */
    if(val == null)
      return;

   /* Oppretter et nytt overføringsobjekt som
    * inneholder det objektet som ble valgt.
    */
    Transferable trans = new TObjectTransferable((TObject)val);
```

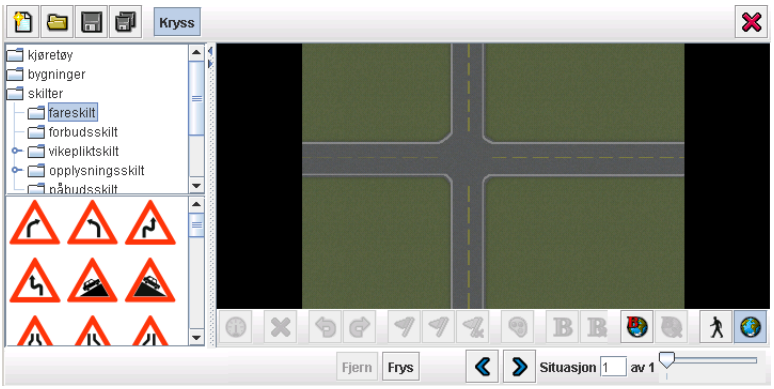
```
  /* Prøver å starte dra-bevegelsen */
   try
   {
     e.startDrag(DragSource.DefaultCopyNoDrop, trans,
         DragList.this.dsl);
   }
   /* Klippet vekk uvesentilg kode */
  }
```

Hvis starten av dra og slipp"manøveren gikk bra, så tar JVM seg av bevegelsen inntil slippet. For å håndtere slippet har vi skrevet en utvidelse til Scene klassen fra inkrement 1. Dette er bare en subklasse som har de metodene som er nødvendige for å kunne ta imot et slipp. Mottagelsen av et slipp gjøres slik (Hentet fra DropScene.java.):

```
 /* Metoden som kalles når et slipp skal gjøres.*/
  public void drop(DropTargetDropEvent e)
  {
   /* Klippet vekk uvesentilg kode /*
    Object data = null;

   /* Prøver å motta slippet. Returnerer hvis
    * ikke noe data blir overført.
    */
    try
    {
      e.acceptDrop(DnDConstants.ACTION_COPY);
      data = e.getTransferable().getTransferData(chosen);
      if(data == null)
        return;
    }
   /* Klippet vekk uvesentlig kode */

   /* Hvis dataene som kommer er av riktig type, gjøres
    * det kall til nåværende scenario om å legge til en
    * modell med de parameterene som ble overført.
    */
    if(data instanceof TObject)
    {
      canvas.requestFocus();
     /* Posisjonen til musa hentes ut. /*
      java.awt.Point mousePos = (java.awt.Point)e.getLocation();
      mousePos.y -= 0;
      scenarios.get(currentScenario).addModelToCurrentSituation(
          ((TObject)data).getModel(),
```

```
            mousePos,
            ((TObject)data).getWaypoint(),
            ((TObject)data).getTextEnabled(),
            ((TObject)data).getTextHeight(),
            ((TObject)data).getInsideEnabled(),
            ((TObject)data).getExtension());

        /* Tegn opp igjen scenen. */
         canvas.repaint();
      }
   }
```

De andre knappene og menyene er bare grensesnitt til ferdige metoder fra inkrement 1. Det endelige utseende til grensesnittet ble slik:



Figur 4.2: Det grafiske brukergrensesnittet til 3D-generatoren.

Dyptgående beskrivelse av hva de forskjellige knappene gjør og deres funksjon finnes i brukerveiledningen under vedlegg B.

4.3.3 Problemer

Det største problemet som kom til overflaten under jobbingen med dette inkrementet var at GLCanvas som vi bruker til å tegne opp Scenen på er basert på klasser fra det gamle grafiske biblioteket til Java. Når vi da kombinerte dette med komponenter fra det nye biblioteket oppsto det konflikter ved at GLCanvas uansett ble tegnet opp øverst. Dermed forsvant menyer som overlappet GLCanvaset borte. Etter mye leting i diverse Java forum fant vi endelig løsningen i det offisielle Jogl forumet[3]. Løsningen viste seg å ligge i en statisk variabel i en

klasse som vi ikke benyttet oss av, men som påvirket rekkefølgen på komponentene. Etter at denne var satt riktig fungerte alt fint.

4.4 Inkrement 3: Presentasjonsmodus

Presentasjonsmodusen ble løst ved på måten beskrevet i designdokumentet under pkt. 3.5. Koden for å sette Kosi presentasjonsmodus er fordelt på to klasser. Først oppdager hovedklassen at presentasjonsmodusknappen er trykket på. Slik ser det ut (Hentet fra Kos.java.):

```
/* Dette er metoden som kalles når en knapp blir trykket
 * på. Funksjonen avgjør da internt hvilken knapp det
 * var og kaller nye metoder der etter.
 */
 public void actionPerformed(ActionEvent ae)
 {
   if(ae.getSource().equals(mode_generation))
   {
  /* Klippet vekk uvesentlig kode /*
   }
  /* Presentasjonsmodusknappen er trykket på. */
   else if(ae.getSource().equals(mode_presentation))
   {
    /* Setter statusen på alle modusknappene der etter. /*
     mode_generation.setSelected(false);
     mode_presentation.setSelected(true);
     mode_question.setSelected(false);
     snapshot.setVisible(true);
    /* Ber det panelet som inneholder Scenen om å gå i
     * presentasjonsmodus. */
     editorpanel.setPresentationMode();
    /* Sørger for å vise generator kortet i layouten. */
     layout.show(cardpanel, "GENERATION");
   }
  /* Klippet vekk uvesentlig kode. */
 }
```

Etter dette tar EditorPanel.java seg av å skjule resten av de komponentene som ikke skal vises:

```
/* Metoden som setter panelet inn i presentasjonsmodus. */
```

```
public void setPresentationMode()
{
 /* Gjemmer skaleringsboksen som evt. er fremme .*/
  panel3d.hideScaleBox();
 /* Setter modusvarabelen riktig. /*
  modeChanged(1);
 /* Sørger for at panelet som inneholder scenen
  * ikke kan velges, men likevel er synlig. */
  panel3d.setSelectEnabled(false);
  panel3d.setVisible(true);

 /* Sørger for at situasjonskontrollen er synlig,
  * men at knappene for å opprette og slette
  * situasjoner er skjult. */
  sitCtrl.setVisible(true);
  sitCtrl.toggleButtonVisibility(false);

 /* Skrur av alle lytterne til scenen så input
  * ikke påvirker scenen. */
  panel3d.toggleListeners(true);
 /* Skjuler objektmenyen */
  toggleMenu(false);
 /* Tegner det hele opp igjen. */
  invalidate();
  validate();
  repaint();
  panel3d.repaint();
}
```

Når dette er gjennomført tar scenen nesten hele flaten til applikasjonen. Da ser det slik ut:

Figur 4.3: Kos i presentasjonsmodus.

4.5 Inkrement 4: Spørsmålsgenerator

4.5.1 Introduksjon

Spørsmålsgeneratoren var på mange måter en veldig spennende del av programmet implementasjonsmessig. Mye mer spennende enn det man ved først øyekast kanskje skulle tro. Først og fremst var det mange utfordringer knyttet til det å få hentet ut bildet fra 3D-generatoren og inn i spørsmålsgeneratoren. I tillegg var det en del utfordringer knyttet til det å få brukergrensesnittet til å tilpasse seg riktig når komponenter legge til og fjernes. Dette gjelder når man endrer antallet alternativer knyttet til et spørsmål. I tillegg har vi valgt å tilpasse oppførselen til treet hvor man velger spørsmål for at dette skal oppføre seg på en mer effektiv måte. På de neste punktene vil de viktigste delene av inkrementet bli beskrevet før problemer og løsningene på disse blir drøftet til slutt.

4.5.2 Viktige deler av inkrementet

Stillbildegenerator

For å få hentet ut et stillbilde av det som blir presentert på scenen var vi nødt til å hente ut det som lå i bufferet på skjermminnet.

```
/* Dette er metoden som tar det som ligger i skjermminnet og
```

```
  * skriver det til et ImageIcon som returneres /*
public ImageIcon capture(String filename)
{
 /* Setter variabelen som sier om bildet er ferdig til
  * ikke sant. */
    captured = false;
   /* Henter ut 3D konteksten til canvaset */
    GLContext context = canvas.getContext();
   /* Setter den nye konteksten til nåværende. */
    if(context.makeCurrent() == GLContext.CONTEXT_NOT_CURRENT)
    {
      return null;
    }

   /* Henter GL objektet til canvaset. Dette objektet sammen
    * med konteksten er det vi trenger for å kunne gjennomføre
    * GL-kallene utenfor pipeline. */
    GL gl = canvas.getGL();
    GLU glu = new GLU();

   /* Ber hele scenen vise seg på med den nye konteksten. */
    disp(gl);

   /* Henter størrelsen på vinduet som dermed også blir
    * størrelsen på bildet. */
    Dimension winSize = canvas.getSize();
    int width = winSize.width;
    int height = winSize.height;
   /* Oppretter et bytebuffer som kan holde på all informasjonen
    * om bildet. */
    ByteBuffer originalPixels =
      BufferUtil.newByteBuffer(width * height * 3);

   /* Velger hvilke bildebuffer vi skal lese fra. */
    gl.glReadBuffer(GL.GL_FRONT);
    gl.glPixelStorei(GL.GL_PACK_ALIGNMENT, 1);

   /* Leser alle pixler fra bildebufferet inn i bytebufferet */
    gl.glReadPixels(0, 0, winSize.width, winSize.height,
        GL.GL_RGB, GL.GL_UNSIGNED_BYTE, originalPixels);

   /* Oppretter en ny array som skal brukes til mellomlagring
    * mens vi gjør om bildeinformasjonen til et format som kan
    * lagres i et BufferedImage. */
    int[] pixelInts = new int[width * height];
```

```
 int p = width * height * 3;
 int q;
 int i = 0;
 int w3 = width*3;

/* Går igjennom alle pixlene og legger disse på riktig
 * plass i arrayen. */
 for (int row = 0; row < height; row++)
 {
   p -= w3;
   q = p;
   for (int col = 0; col < width; col++)
   {
     int iR = originalPixels.get(q++);
     int iG = originalPixels.get(q++);
     int iB = originalPixels.get(q++);

    /* Gjør logisk magi på R, G og B verdiene for at disse
     * skal bli representert riktig med alpha kanal. */
     pixelInts[i++] = 0xFF000000 | ((iR & 0x000000FF) << 16) |
        ((iG & 0x000000FF) << 8) | (iB & 0x000000FF);
   }
 }

/* Skriver alle pixlene til et BufferedImage. */
 BufferedImage bufferedImage = new BufferedImage(width, height,
     BufferedImage.TYPE_INT_ARGB);
 bufferedImage.setRGB(0, 0, width, height, pixelInts, 0, width);

/* Frigjør konteksten. */
 context.release();
/* Er ferdig med  bildet */
 captured = true;
/* Returnerer et nytt ImageIcon med ferdig skjermbilde */
 return new ImageIcon(bufferedImage);
}
```

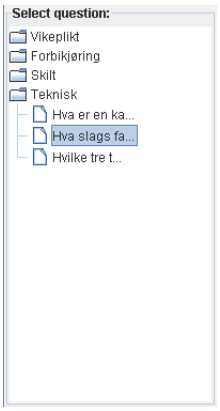
Det å få til å ta et skjermdump på en måte som fungerte i alle tilfeller viste seg å være vanskeligere enn antatt. Tilslutt fikk vi til en god løsning som kan leses ovenfor. Andre forsøk og fiaskoer er beskrevet under pkt. 4.5.3.

Spørsmålstreet

Det var ønskelig å lage et tre som inneholdt alle kategoriene og spørsmålene som lå i databasen. Dette var forsåvidt greit nok, men vi ønsket også å tilpasse måten treet oppførte seg når man klikket på det på en spesiell måte. Når man klikker på en kategori skal alle andre kategorier i treet trekke seg sammen (kollapse) og det første spørsmålet under den nyvalgte kategorien skulle velges og vises automatisk. Etter en del forsøk fikk vi til en ganske grei løsning. Dette er koden som håndterer alt som skal skje når en node i treet blir valgt (Koden ligger i QuestionPanel.java):

```
/* Metoden som kalles når en node blir valgt i treet */
 public void valueChanged(TreeSelectionEvent tse)
 {
  /* Noden som er valgt hentes ut */
   Object node = qTree.getLastSelectedPathComponent();
  /* Er noden et Question objekt? */
   if(node instanceof Question)
   {
    /* Kaller metoden som setter alle knappene brukbare. */
     setEnabled(true);
    /* Kaller metoden som viser et Question objekt */
     viewQuestion((Question)node);
    /* Henter ut kategorien til Question objektet via
     * den sin parent i treet. */
     String cat = (String)((DefaultMutableTreeNode)
         ((Question)node).getParent()).getUserObject();
    /* Sørger for å oppdatere "dropdown"-listen der man
     * setter kategorien til et spørsmål. */
     for(int i = 0; i < categorySelector.getItemCount(); i++)
     {
       if(((String)categorySelector.getItemAt(i)).equalsIgnoreCase(cat))
       {
         categorySelector.setSelectedIndex(i);
         break;
       }
     }
   }

  /* Er noden en default node. Da er en kategori valgt. */
   else if(node instanceof DefaultMutableTreeNode)
   {
    /* Minimerer alle radene i treet. Da blir alle
     * kategoriene slått inn. */
```

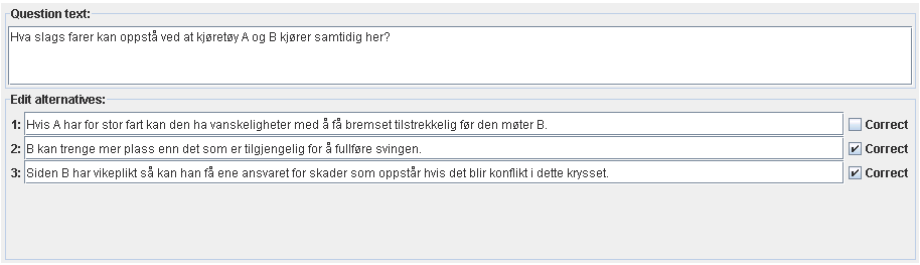
```
    for(int i = 0; i < qTree.getRowCount(); i++)
      qTree.collapseRow(i);

   /* Hvis denne noden ikke er en bladnode så har den
    * minst et Question objekt som barn, så da skal
    * det vises frem. */
    if(!((DefaultMutableTreeNode)node).isLeaf())
    {
     /* Knappene skal synes. */
      setEnabled(true);

     /* Setter ny valgt sti i treet til stien til første
      * barnet til den noden som er valgt. */
      qTree.setSelectionPath(new javax.swing.tree.TreePath(
            ((DefaultMutableTreeNode)
             ((DefaultMutableTreeNode)node).getFirstChild()).getPath()));
    }

   /* Hvis noden er en bladnode. */
    else
    {
     /* Alle bilder fjernes fra bildevisningspanelet. */
      imagePanel.removeAll();
     /* Legger inn Kos-logoen som nåværende bilde. */
      currentImage = logo;
     /* Legger til dette bildet i visningen. */
      imagePanel.add(new JLabel(currentImage), BorderLayout.CENTER);
     /* Skrur av alle knappene som ikke skal fungere når
      * ingen spørsmål er valgt. */
      setEnabled(false);
    }
  }
}
```



Figur 4.4: Treet der man velger spørsmål.

Panelet med alternativene

Panelet med alternativene måtte løses på en litt tungvint måte. På grunn av Java sitt ønske om å la alle komponenter få så stor plass som overhode mulig så vil panelet med alternativene bli mindre med en gang et alternativ blir tatt bort. Dette ble løst ved å bruke en layout-manager som setter av plass til et visst antall komponenter. Plassen disse komponentene før er definert utifra det største komponentet som er lagt inn på panelet. Hvert alternativ består grafisk sett da av et eget panel med et tekstfelt og en avkryssningsboks på. Dette fungerer fint, selv om nøstingen av paneler bli ganske dyp.



Figur 4.5: Panelet der man redigerer spørsmålet med alternativer. Legg merke til oppdelingen med alternativene på et eget panel nederst.

4.5.3 Problemer

Koden for å ta stillbilde av en situasjon ble laget i mange inkarnasjoner. Mange forskjellige vinklinger og teknologier ble prøvd ut, men enten så oppfylte de ikke kravene, eller så var de ikke stabile nok til at vi kunne benytte dem. En teknologi som tillater rendering av bilder utenfor skjermen er kalt Pbuffer. Dette er en hardware avhengig teknologi som egentlig skal være standard på alle skjermkort i dag, men det viste seg å ikke være riktig så denne teknikken måtte legges på hylla. Jogl kommer også med en innebygget screenshotklasse, men denne viste seg å ikke oppfylle våre krav og ble derfor skrotet. OpenGL Utilities (GLU) ble testet med en ferdiglaget funksjon, men denne førte til en "buffer underflow" som vi ikke var i stand til å finne ut av. I tillegg forsøket vi å bruke en teknikk kalt "tilerendering" som går ut på å dele bildet opp i flere små bilder, tegne disse hver for seg for å til slutt sette de sammen til et helt bilde. Denne teknikken ble litt for vanskelig å få implementer innenfor tidsfristene. Til slutt greide vi å finne en måte å lese direkte fra skjermminnet for å på den måten hente ned det vi trengte. Dette fungerer smertefritt på alle PC'er og skal være helt arkitektur uavhengig.

4.6 Inkrement 5: Prøve

4.6.1 Introduksjon

Implementasjon av prøve-modulen har vært mindre krevende enn tidligere inkrementer. Dette på grunn av at vi her ikke har trengt å integrere kildekoden med eksisterende kildekode. Utviklingen av inkrementet har i svært stor grad gått på brukergrensesnitt, men også en liten grad av fil-innlesing og en liten algoritme utplukking av hvilke spørsmål som skal være med på en prøve. Selve utviklingsprosessen for inkrementet har nok vært den minst problematiske i hele prosjektet. Implementasjonen følger design-dokumentet (pkt 3.7) for inkrementet svært nøye.

4.6.2 Viktige deler av inkrementet

Utvelgelse av ny prøve

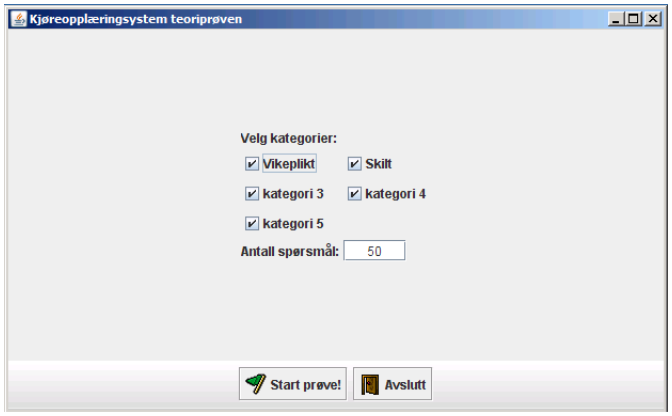
For utvelgelse av spørsmål som skal være med på ny prøve brukes følgende algoritme:

```
 int size = includecategories.length;
 questions = new Vector<Question>(); //oppretter ny tom prøve
 questionindex = 0; //sier at man starter på første spørsmål.

 Random rand = new Random();

 Vector<Question> allquestions = new Vector<Question>();

/* Henter ut alle spørsmål som skal være
 * med i uttrekningen. Dette basert på hvilke
 * kategorier som var valgt i brukergrensesnittet. */
 for(int i=0;i<size;i++)
 {
   if(includecategories[i].isSelected())
   {
   /* En midlertidig datastruktur med navn ŤallquestionsŤ
    * blir brukt for å holde samlingen av spørsmål. */
     Category cat = categories.get(i);
     for(int j=0;j<cat.getQuestionCount();j++)
       allquestions.add(cat.getQuestion(j));
   }
 }
/* Bestemmer at hvis ønsket antall spørsmål
 * er mindre enn mulig antall spørsmål så brukes
 * mulig antall spørsmål istedenfor. */
 int qcount = (allquestions.size() < QUESTION_COUNT ?
                allquestions.size() : QUESTION_COUNT);

/* Henter ut så mange spørsmål som skal være
 * med i prøven tilfeldig fra samlingen over
 * gyldige spørsmål. */
 for(int i=0;i<qcount;i++)
 {
   Question q = allquestions.remove(rand.nextInt(allquestions.size()));
   q.reset();
   questions.add(q);
 }
```

Brukergrensesnitt

Brukergrensesnitt har som sagt vært en svært avgjørende del av dette inkrementet. Vi har derfor lagt med noen skjermbilder av det endelige brukergrensesnittet som kan brukes som et direkte sammenligningsgrunnlag mot designskissene.

Brukergrensesnittet for ny prøve ser ut som følger:



Figur 4.6: Oppsettet av en prøve.

Brukergrensesnittet for spørsmålene ser ut på følgende måte:

Figur 4.7: Visning av en prøve.

Brukergrensesnittet for oppsummering av en prøve ser ut på følgende måte:

Figur 4.8: Oppsummeringen av en prøve.

Se brukermanual (B) for nærmere beskrivelse av skjermbildene. Se også designdokument (3.7) for sammenligning av skisse mot endelig resultat.

Kapittel 5

Testing og kvalitetssikring

5.1 Kvalitetssikring

Vi har under prosjektet benyttet versjonskontrollsystemet Subversion for å kvalitetssikre både kildekode og dokumenter. Subversion-tjeneren er satt opp av IT-tjenesten så vi har her ikke trengt å tenke over sikkerhetskopi selv da alt dette har blitt tatt hånd om. I tillegg har vi benyttet fem forskjellige datamaskiner i selve utviklingsprosessen og har derfor synkronisert alt matriale på disse med jevne mellomrom. Vi har synkronisert ved hjelp av versjonskontrollsystemet hver gang vi har fullført en funksjonalitet eller annen del av prosjektet, samt hver kveld før vi har gitt oss. På den måten har vi alle sittet med samme versjon av prosjektet hver morgen. Når flere gruppemedlemmer har jobbet med samme fil samtidig har vi heller ikke trengt å tenke over dataintegriteten da versjonskontrollsystemet har tatt seg av å sette sammen de forskjellige versjonene av samme fil. Hvis en prosjektmaskin hadde gått i stykker ville vi derfor hatt tilgang til nyeste versjon av prosjektmatrialet ved en hvilken som helst annen maskin tilknyttet Internett eller skolens lokale nettverk.

5.2 White-box testing

Under hele prosjektet har produktet blitt white-box testet etterhvert som funksjonalitet har blitt utviklet. Ettersom gruppens medlemmer gjerne har jobbet på forskjellige deler av applikasjonen har vi testet på egenhånd og for hverandre. Mot slutten har vi testet mer omfattende ved å sette opp hele trafikale situ-

asjoner og teorispørsmål. De fleste feil i applikasjonen har blitt oppdaget på nettopp denne typen testing.

5.3 Black-box testing

For å black-box teste applikasjonen har vi satt det i bruk i tre forskjellige kontekster. Å få satt applikasjonen i bruk i forskjellige kontekster har vært viktig for å kunne få tilbakemeldinger basert på de forskjellige bruksmønsterene testeren har hatt. Vi har så i møter med testerne fått tilbakemeldinger på alle områder i applikasjonen hvor brukeren føler det må gjøres endringer. Feil med applikasjonen som har blitt oppdaget i disse testene har blitt rettet umiddelbart uten diskusjon, mens endringsønsker har blitt diskutert i fellesskap med oppdragsgiver før gruppen har avgjort hvorvidt vi vil etterkomme ønsket. Som oftest har endringsønsker blitt gjennomført med mindre det har krevd for mye tid eller omorganisering av prosjektet og kildekoden.

5.3.1 Fagmiljø

De viktigste tilbakemeldingene for oss har vært fra målgruppen for selve applikasjonen. I vårt tilfelle er dette trafikkskoler og trafikkskoleelever. Thomtes Trafikkskole har under prosjektet fungert som vår samarbeidspartner og vi har under store deler av prosjektet hatt faste møter med Thomtes annenhver uke. På disse møtene har vi vist frem det vi har gjort og fortalt om planene våre for så å få tilbakemeldinger.

To ganger i løpet av prosjektet har vi installert den daværende versjonen av applikasjonen så trafikkskolen selv skal kunne teste det på fritiden. Thomtes har så hatt i underkant av to uker på seg til å teste dette i eget tempo og etter eget behov for så å komme med tilbakemeldinger på neste møte. Thomtes var først og fremst interessert i selve 3D-generatoren og presentasjons-modus og fikk derfor utlevert testversjon ved første brukbare versjon av 3D-generatoren og etter presentasjonsmodus var utviklet. Mot slutten av april satt vi stopper for endringsønsker pga tidsmessige aspekter, og derfor avsluttet de faste møtetidspunktene våre med Thomtes.

Tilbakemeldingene fra Thomtes har først og fremst gått på brukergrensesnittet og utvalget av trafikale objekter.

5.3.2 Oppdragsgiver

Vi har under prosjektet naturligvis fulgt de krav vi har fått oppgitt fra oppdragsgiver. Det har derfor vært viktig for oss å høre tilbakemeldinger fra oppdragsgiver underveis. Vi har under hele prosjektet hatt møter med oppdragsgiver Frode Haug hver fredag og har der vist frem fremgangen. Vi har så fått tilbakemeldinger på ønskede endringer, og disse har blitt fulgt såvidt det har latt seg gjøre.

Omtrent midtveis i prosjektet fikk også oppdragsgiver en egen versjon av applikasjonen for å kunne teste ut selv. Vi har så fått en konkret liste med endringer som måtte utføres og mulige feil. Denne listen ble fulgt og krysset av etterhvert som det ble løst. På møtene har også listen blitt vurdert og løste problemer har blitt presentert for oppdragsgiver.

Tilbakemeldingene fra oppdragsgiver har først og fremst gått på brukergrensesnitt og funksjonalitet , men også noen småfeil.

5.3.3 Nøytral tredjepart

Vi ønsket også en nøytral part til å bruke applikasjonen for å se hvordan det fungerte for en person uten relasjon til prosjektoppgaven. Mads Byfuglien, som i skrivende stund fullfører sin masteroppgave på Informasjonsikkerhet-studiet til Høgskolen i Gjøvik, sa seg villig til dette. Han fikk på slutten av prosjektperioden, det vil si begynnelsen av mai måned, utlevert en kopi av applikasjonen og fikk testet fritt. Mads fant noen feil i applikasjonen som har blitt rettet, men hadde ellers lite å utsette på produktet.

Kapittel 6

Oppsummering

6.1 Kritikk av oppgaven og drøfting av resultater

6.1.1 Produkt

Ved prosjektets slutt er Kos å anse som en komplett applikasjon rent funksjonelt ut i fra de krav vi hadde satt oss for denne perioden. Det skal være modent nok for bruk i den sammenheng den er tiltenkt, dvs for bruk lokalt av trafikkskoler.

Gruppen er godt fornøyd med funksjonaliteten applikasjonen inneholder, men synes alikevel det er litt trist at Internett-distribuering ikke kunne være en del av oppgaven grunnet begrenset tid. Den største svakheten produktet er nok alikevel størrelsen og kvaliteten på biblioteket med trafikale objekter. Selv om vi har fått dekket det grunnleggende behovet kan det nok oppstå situasjoner hvor denne pakken med trafikale objekter ikke tilfredsstiller produktets målgruppe sitt behov. Å utvikle slike objekter er svært tidkrevende og er en stor nok jobb til å være et eget hovedprosjekt.

De viktigste kravene til produktet har vært brukervennlighet og ytelse. Brukervennlighet er viktig ettersom produktets målgruppe, trafikkskolelæreren og trafikkskoleeleven, ikke nødvendigvis sitter med de forhåndskunnskapene om datamaskiner og programvare som vi gjør. Målet har vært at mannen i gaten, for å bruke et slikt uttrykk, skulle kunne sette seg ned foran applikasjonen og bruke den med minimal mengde opplæring. Vi mener at vi har lykkes med dette da vi har fått stort sett gode tilbakemeldinger fra oppdragsgiver Frode Haug og samarbeidspartner Thomtes Trafikkskole. Både oppdragsgiver og samarbei-

dspartner har dessuten selv vært med å bestemt hvordan applikasjonen skal se ut og fungere basert på eget behov så vi er absolutt fornøyd med hva vi har oppnådd med brukervennlighet. Det er alikevel ganske vanskelig å måle brukervennlighet så for sikkerhets skyld har vi alikevel skrevet en brukermanual (vedlegg B) som viser hvordan man bruker applikasjonen.

Ytelse har som nevnt ovenfor også vært en viktig faktor under prosjektet da det er rimlig å anta at målgruppen ikke har store budsjetter på datamaskiner og datautstyr, og dermed kan sitte på en del eldre maskinvare. Dette målet er absolutt oppnådd. Applikasjonen har vært under testing av trafikkskolen og av oppdragsgiver underveis i prosjektet, og vi har ikke fått noen negative tilbakemeldinger på ytelsesmessige aspekter til tross for at begge disse har benyttet det vi kan referere til som eldre datamaskiner. I tillegg har vi selv i tillegg til nyere datamaskiner også benyttet eldre datamaskiner som spesifikasjonsmessig er svært nærme minimumskravet, og heller ikke her har det vært noe særlig problemer med ytelse.

Prosjektrapporten som skal foreligge ved prosjektslutt er dessverre ikke den sterkeste og dette kan vi bare beklage. Det ble dessverre litt knapt med tid mot slutten. Selv om vi er ganske fornøyd med deler av den, bærer den også preg av den tidsmangelen som oppstod mot slutten.

6.1.2 Prosess

Ved prosjektets begynnelse hadde både prosjektgruppen og oppdragsgiver ganske store ambisjoner om prosjektets muligheter og omfang. Det ble tidlig klart at dette var for optimistisk med tanke på tiden vi hadde tilgjengelig og oppgavens omfang ble nedskalert allerede før kravspesifikasjonen var på plass. I tillegg var det en uenighet mellom oppdragsgiver og samarbeidspartner om applikasjonens formål som igjen førte til en liten utvidelse, nemlig presentasjonsmodus. Selv med de krav vi har forholdt oss til har oppgavens omfang vært godt over forventet størrelse, og vi har jobbet overtid daglig for å komme i mål.

Utviklingsmodellen som på forhånd så ut til å være best egnet for prosjektet var inkrementell modell, men prosjektet har båret preg av en blandingsmodell mellom inkrementell og evolusjonær med prototyping. Vi har underveis med jevne mellomrom vist frem prosjektet til oppdragsgiver Frode Haug og samarbeidspartner Thomtes Trafikkskole og har fått endringskrav tilbake med samme hyppighet som vi har presentert arbeidet. Prosjektet har vært oppdelt i inkrementer som har blitt utført mer eller mindre i den rekkefølgen det var planlagt (med unntak), men innad i hvert inkrement har utviklingen foregått evolusjonært.

Som nevnt har vi underveis i prosjektet måttet avvike fra fremdriftsplanen. Dette går først og fremst på tidsforbruket for hvert inkrement da dette har blitt feilberegnet, men for første inkrement (-3D-generatoren) har det vist seg hensiktsmessig å foregå utviklingen parallelt med de andre inkrementene underveis. Årsaken er at det viste seg å være mye mer arbeid med inkrementet når vi faktisk fikk oversikt over hva som måtte gjøres. Dette er noe vi ikke kunne noe for da vi på forhånd av prosjektet ikke hadde den kompetanse og innsikt i fagområdet som vi i dag sitter med.

Under hele prosessen har vi også ført en logg på arbeidsinnsats fra dag til dag. Denne er dessverre ikke komplett da vi ikke har vært så flinke til å logge som vi burde. Vi har alikevel svært god oversikt over vår egen innsats underveis da vi i gruppereglene har hatt et minimum antall arbeidstimer per dag og vi vet hvem som har jobbet med hva i hvilke perioder. Denne regelen om minimum antall arbeidstimer har blitt overhold uten betydelige avvik, og gjennomsnittlig ligger nok alle gruppens medlemmer godt over oppgitt antall.Møter med vår samarbeidspartner Thomtes Trafikkskole har vært annenhver uke fra januar til midtveis i april hvor vi har vist fremgangen og har fått tilbakemeldinger. Møter med oppdragsgiver/veileder med samme formål har funnet sted fredag hver uke og gruppemøte for å diskutere fremgang og organisering har funnet sted hver mandag.

Den delen av prosjektet som har måttet lide mest av prosjektets store omfang og mangel på tid er nok dessverre biblioteket med trafikale objekter. Vi mente det var viktigere å prioritere utviklingen av selve applikasjonen da en ustabil og mangelfull applikasjon ville ødelagt for prosjektets formål, mens objektbiblioteket senere kunne utvides etter behov. Utformingen av en komplett samling med trafikale objekter ville uansett prioritet vært for tidkrevende for oss, og kan derfor passe som et eget hovedprosjekt for en gruppe som utdanner seg i grafiske eller mediefag, eller som en utvidelse/forbedring fra vår egen gruppe etter prosjektets slutt. Objekt-biblioteket inneholder alikevel ved prosjektets slutt et grunnleggende sett objekter som kan brukes for å generere et veldig stort utvalg av de situasjoner og spørsmål man i dag finner i oppgavehefter og teoriprøver, og ikke minst demonstrere applikasjonens funksjonalitet.

Prosjektprosessen har selvfølgelig ikke vært helt uten problemer. Blant problemene vi har støtt på kan vi nevne:

- Driverproblemer. Enkelte skjermkortdrivere til maskinene vi har jobbet med har vært svært dårlig, noe som har stor innvirkning på mulighetene tilgjengelig. Vi har derfor måttet utvikle tungvinne løsninger for å omgå disse.
- Teknologi som ikke er tilpasset hverandre. Teknologi som ikke er ment for å arbeide sammen har til tider ført til problemer. Heldigvis er vi ikke de

første som har opplevd denslags problemer med teknologien vi har valgt, og derfor har vi funnet løsninger på det.

- Stadig forandringer i kravene har ført til et enormt tidsforbruk underveis. Dette er dessverre ikke noe vi kan gjøre noe med forutenom at vi har måttet satt bestemte tidspunkter hvor ingen flere store endringskrav vil bli tatt hensyn til. Stadige endringer har også til tider påvirket motivasjonen, men når vi kom fordi disse stadige endringskravene ordnet motivasjonen seg igjen.
- En bratt læringskurve når vi har tilegnet oss nødvendig kunnskap for å løse oppgaven har ført til at vi underveis enkelte steder har måttet ty til mindre elegante midlertidige løsninger. Disse har så måttet endres etterhvert som vi har lært bedre metoder. Dette har selvfølgelig vært litt tidkrevende.

Hvis vi i dag skulle startet på et tilsvarende prosjekt og hadde sittet på den kunnskap vi nå gjør så ville vi antagligvis lagd en mer korrekt fremdriftsplan på forhånd, men forutenom dette så tror vi at utviklingsprosessen ville foregått på en tilsvarende måte som dette prosjektet.

6.2 Hva har vi lært?

Prosjektet har for oss først og fremst vært en opplæringsprosess hvor vi har tilegnet oss relevante erfaringer for arbeidslivet og teknikker for å bedrive utvikling. Blant annet kan vi nevnte følgende punkter som kunnskap vi har tilegnet oss i løpet av prosjektarbeidet:

- OpenGL og 3D-utvikling
- Utforming av modeller og teksturering
- Utvikling og utforming av detaljert grafisk brukergrensesnitt
- Mattematiske utregninger relevant til grafisk programmering
- Systemering i bruk
- Praktisk bruk av designpatterns
- Generelt å jobbe med store prosjekter, ansvar, samarbeid innad i gruppen og interaksjon med oppdragsgiver og fagmiljø.

Alt dette er utvilsomt kunnskap som vil komme oss til gode ved senere studier og arbeid.

6.3 Videre arbeid

Selv om Kos er å anse som komplett ut i fra de krav vi hadde satt for perioden prosjektet foregikk (dvs applikasjon for lokal bruk på 1 datamaskin) har det vært snakk om senere utvidelser/forbedringer. Dette inkluderer både forbedringer/utvidelser i selve applikasjonen, utvikle nye eksterne moduler og forbedre/utvide grafikk-biblioteket. Utvidningsmulighetene er nært sagt uten grenser hvis man er litt kreativ. Blant mulige utvidelser kan vi foreslå:

6.3.1 Internett/nettverk distribuering

Det var i utgangspunktet et forslag fra oppdragsgiver om å distribuere spørsmål og prøver fra spørsmålsgeneratoren til andre maskiner over et nettverk og/eller Internett. Dette ble ikke en del av dette prosjektets spesifikasjoner, men er absolutt en aktuell utvidelse. Forslaget gikk ut på å inkludere støtte for følgende:

- Trafikkskolelærere skal kunne distribuere spørsmål og prøver fra spørsmålsgeneratoren til et sett lokale datamaskiner i trafikkskolens lokaler slik at elever kan sitte på skolen og jobbe med oppgaver.
- Trafikkskolelærere skal kunne publisere spørsmål og prøver til en sentralisert server på Internett og elever kan sitte hjemme og jobbe med oppgaver. Dette åpner igjen muligheten for å skape et "community" på Internett som lar trafikkskolen gi elevene sine tilgang til de oppgaver skolen selv har produsert eller dele oppgaver på tvers av skoler.
- Trafikkskolelærere kan publisere spørsmål og situasjoner til en sentralisert server på Internett og andre trafikkskolelærere kan hente ned for lokal bruk. Her er det også store muligheter for å skape et "community", men her for trafikkskolelærere fremfor elever.

Det ble også nevnt muligheter for å eksportere og importere prøver mellom lokale datamaskiner uten noen form for nettverk eller Internett, men da nettverk og Internett er såpass billig og utbredt som det er nå så er ikke dette like aktuelt å fokusere på.

6.3.2 Sammensetning av prøver

En annen mulig utvidelse er et eget verktøy for å sette sammen detaljerte teoriprøver, og utvide den delen av applikasjonen som tillater eleven å ta prøver

slik at den også kan ta en slik prøve fremfor bare en som er satt sammen tilfeldig.

6.3.3 Lokasjons-generator

En mulig forbedring er å kunne lage en lokasjon helt uten ferdige maler. Det vil si å kunne starte med en blank lokasjon og så måtte trekke opp veier, jernbane, o.l. selv slik som mange kjenner til fra strategispill som Sim City. Dette vil være en svært omfattende jobb som kanskje kan bli for avansert og tidkrevende for et hovedprosjekt av denne typen da det vil kreve en totalt omskriving av applikasjonen.

6.3.4 Objekt-bibliotek

Den kanskje mest aktuelle utvidelsen/forbedringen av applikasjonen er et større og bedre bibliotek av trafikale objekter. Ved prosjektets slutt vil applikasjonen inneholde en veldig grunnleggende samling objekter, og det vil uten tvil være hensiktsmessig å utvide denne samlingen slik at applikasjonen kan dekke ethvert behov en trafikkskolelærere kan finne på å måtte ha ved opprettelse av trafikale situasjoner. Å lage slike objekter er ikke spesielt vanskelig, men er svært tidkrevende, og kan kanskje derfor egne seg godt som senere hovedprosjekt for noen som driver med grafisk/medie-utdannelse.

6.3.5 Endring av utseende på individuelle objekter

En annen mulig forbedring er muligheten for å endre utseende på hvert individuelle objekt under kjøring. For eksempel kunne det da gått ann å endre farger på spesifikke objekter etter behov i motsetning til slik det er nå hvor hver objekt har en bestemt farge uansett. Dette ble ikke implementert i utgangspunktet da vi ikke fant en god måte å gjøre det på uten å redusere ytelsen, og dermed kanskje ville risikert å ikke oppnå ytelseskravet vi hadde satt oss.

6.3.6 Integrering i eksisterende løsninger

En annen mulighet er å tilpasse applikasjonen for allerede eksisterende løsninger for teoriprøve på datamaskin. Det eksisterer allerede mange løsninger for å ta teoriprøver både lokalt og over Internett, men ingen av disse har mulighet til å

opprette situasjoner i 3D. Derfor hadde det vært fullt mulig å integrere vår løsning sammen med eksisterende løsninger slik at Kos oppretter spørsmålene og de eksisterende løsningen tar seg av presentasjonen av disse. Hvis dette hadde blitt en realitet ville vi kanskje bare måttet tilpasse lagringsformatet vårt i spørsmålsgeneratoren så det passer med den eksisterende løsnings spørsmålsbibliotek.

6.4 Evaluering av gruppens arbeid

6.4.1 Innledning

Gruppens medlemmer har tidligere jobbet godt sammen på prosjekter ved høgskolens studieretning programutvikling med gode resultater. Dette har ført til at vi kjenner hverandre godt nok til å vite hvordan vi skal forholde oss til hverandre i skoleprosjekter. Vi var allerede før jul i full gang med planleggingen av dette prosjektet for å få mest mulig ut av det, med både god karakter og et godt produkt som mål.

6.4.2 Organisering

Gruppen har bestått av 3 personer som daglig under hele prosjektet har møttes på skolen eller hjemme hos en av gruppens medlemmer for å arbeide med oppgaven i fellesskap. I tillegg har gruppens medlemmer selv regulert sin egen arbeidsinnsats på fritiden for å oppnå de mål som er satt. Arbeidsmengden for hvert medlem av gruppen var i henhold til forprosjektets grupperegler satt til minimum 6 timer per dag hver ukedag, noe som vi uten tvil har oppfylt og vel så det. Ved avtalte møter mellom gruppen og oppdragsgiver/veileder eller samarbeidspartner har vi hatt minimalt med fravær. Selv om arbeidsinnsatsen har vært upåklagelig kan vi derimot kritisere vår egen innsats når det gjelder å logge aktivitet. I starten av prosjektet var vi veldig flinke til å logge, men mot slutten har vi dessverre blitt for slappe på akkurat dette området.

Vi har underveis hatt avtalte møter med vår samarbeidspartner på Thomtes Trafikkskole annenhver uke frem til midten av april og møter med oppdragsgiver hver fredag i hele prosjektperioden hvor vi har diskutert fremgangen i prosjektet. Gruppen selv har hatt faste møter hver mandag hvor vi har diskutert alt som etter behov rundt prosjektets fremgang og organisering.

Vi har hatt Jonas som gruppeleder under hele prosjektet og han har derfor holdt et overblikk over fremdrift og har fungert som kontaktperson mellom

gruppen og oppdragsgiver samt samarbeidspartner. Dette har for oss fungert, og vi har ikke hatt noen alvorlige brudd på de grupperegler som var satt på forhånd. Til tross for gruppeleder og grupperegler har organiseringen vært ganske uformell ettersom vi kjenner hverandre fra tidligere prosjekter og vet at vi kan stole på hverandre.

Som tidligere nevnt fulgte vi ikke valgte utviklingsmodell slavisk, men heller gikk for en blandingsmodell hvor vi delte opp i inkrementer og utførte hvert inkrement evolusjonært. Hvis vi ved prosjektets start hadde visst det vi vet i dag ville vi nok ha valgt en renere evolusjonær utviklingsmodell.

6.4.3 Fordeling av arbeidet

Gruppen har fordelt arbeidet underveis etter eksisterende kompetanse og ønsker. Selv om fordelingen av arbeidsmengden ikke har vært helt lik over hele gruppen så har alle gruppens medlemmer gjort en større innsats tidsmessig enn hva vi hadde forventet på forhånd. Vi kan trygt si at alle på gruppen i aller høyeste grad har gjort sin del av innsatsen på dette prosjektet. I grove trekk har arbeidet blitt fordelt på følgende måte:

- 3D-generatoren som har vært essensen i prosjektet har først og fremst blitt utviklet av Kenneth. Han hadde fra før kompetanse i OpenGL og ønsket å jobbe med dette. Derfor har dette vært hans primære oppgave gjennom hele prosjektet. Jonas har også jobbe litt med dette i første del av prosjektet.
- Grafisk brukergrensesnitt og integrering mellom brukergrensesnitt og 3D-generatoren har blitt utviklet av Sverre og Jonas.
- Utforming av grafiske objekter har blitt gjort av alle gruppens medlemmer når man har hatt tid. Gjerne mellom lange programmeringsøkter. Dette for å unngå alt for ensidig jobb og på den måten styrke motivasjonen over tid.
- Dokumentasjon og rapportskriving har vært en aktivitet alle gruppens medlemmer har deltatt på, men mot slutten har dette ansvaret falt spesielt på Sverre og Jonas.

6.4.4 Prosjekt som arbeidsform

Ingen av gruppens medlemmer har tidligere erfaring av prosjekter på denne størrelsen. Vår tidligere erfaringer er prosjekter som har strukket seg maksimalt over

noen få uker, og som har inkludert minimalt med systemering og planlegging. Vi har tidligere erfaring innen systemutvikling fra faget med samme navn, men å faktisk bruke systemutviklingen vi har lært på egenhånd i et reelt prosjekt med en oppdragsgiver var helt nytt for oss.

Å jobbe med prosjekt er både spennende og utfordrene, og gir oss større kontroll over egen læring og fremgang enn hva tradisjonell undervisning gjør. Det gir oss en unik innsikt i hvordan vi kanskje vil komme til å jobbe ved endt utdannelse. På en annen side så byr prosjektarbeid på muligheter for mye stress og konflikter. Sistnevnte har vi heldigvis ikke opplevd, men vi har til tider kjent motivasjonen har vært dalende, og det er en svært ubehagelig følelse når man vet at man har tidsfrister å forholde seg til. I tillegg er man i et prosjekt ansvarlig for å ikke henge etter ettersom prosjektgruppen er avhengig av hver enkelt medlem for å gjøre det best mulig.

Det er ihvertfall ingen tvil om at prosjekt som arbeidsform er en glimrende måte å skaffe seg erfaring mot jobbmarkedet, og å bygge opp egen kompetanse. Gruppen i sin helhet stiller etter prosjektets slutt mye sterkere og mer selvsikre på oss selv. Dette tatt i betraktning så har vi etter dette prosjektet ingen problemer med å forstå at undersøkelser viser at IT-jobber er de mest stressende jobbene man kan ha, men vi føler alikevel at dette ikke er den fulle sannhet da det også eksisterer mange gode sider.

6.4.5 Subjektiv opplevelse av hovedprosjektet

Hovedprosjektet har etterlatt et inntrykk om at det er et solid opplegg fra skolens side med svært bra formål, nemlig å la studentene benytte de kunnskapene de har tilegnet seg gjennom studieforløpet. Og å benytte oss mye av tidligere tilegnede kunnskaper har vi i aller høyeste grad gjort.

Vi synes informasjonen vi har fått på epost gjennom prosjektet har vært glimrende og har holdt oss oppdatert om hendelser fra skolens side og tidsfrister. Også de forelesninger som har vært for å informere oss om fremgangsmåter i systemeringen eller bare forfriske gammel kunnskap har vært glimrende og har kommet godt med underveis.

Veileder Frode Haug sitt engasjement i oppgaven har vært upåklagelig og han har underveis både deltatt på møter med samarbeidspartneren vår på Thomtes Trafikkskole og kommet med gode innspill til oppgaven og systemeringen. Vi har hatt faste møter med Frode hver eneste fredag gjennom prosjektet hvor vi har vist hva vi har fått til og fått verdifulle tilbakemeldinger. Forutenom veileder har vi fått svært gode råd av andre ansatte ved skolen. Blant annet kan vi nevne Øyvind Kolloen som har stilt opp og gitt oss svar og tips i forbindelse med Java

problemer og Anders Oulie som har holdt private matteforelesninger for gruppen ved behov. Å ha et såpass godt fagmiljø tilgjengelig under hele prosjektet var over all forventning og har hjulper oss mye.

Prosjektoppgaven vår har vært svært omfattende, men vi har hatt en svært solid gruppe og vi føler derfor at vi har kommet i mål med det vi skulle til tross for varierende motivasjon og knapp tid. Alikevel er det greit å ha kjent presset med tidsfrister i et reelt prosjekt før vi havner i tilsvarende situasjoner i arbeidsmarkedet.

For å sitere gruppeleder Jonas Strømstad: "Beste hovedprosjektet jeg har vært med på..."

6.5 Konklusjon

Vi har i dette prosjektet utviklet et system som skal bidra til å lære personer om trafikken, og hvordan forholde seg til denne. Dette er et svært viktig tema hvor den minste feiloppfatning kan føre til ulykker med tragiske resultater. Vi håper Kos vil bidra til å øke trafikanters forståelse for trafikken i sin helhet gjennom en klarere presentasjon av trafikale situasjoner og hvordan man skal forholde seg til disse.

Kos kan erstatte eller fungere som et tilskudd til den tradisjonelle tavleundervisningen med ferdige powerpointpresentasjoner og magnetbaserte biler på tavler, og det kan erstatte eller fungere som et tilskudd til dagens oppgavehefter og teoriprøve-tentamner. Produktet er klart for installering på trafikkskoler over hele landet og vi håper det også vil bli brukt landsdekkende.

Hvorvidt gruppen ønsker å fortsette arbeidet etter prosjektets slutt er i aller høyeste grad avhengig av hvorvidt prosjektet blir en suksess eller ikke, men etter å ha brukt et halvt år på produksjonen av dette produktet håper vi at det ikke blir nok et hovedprosjekt uten fremtid. Det er ingen tvil om at dette prosjektet har vært en nyttig erfaring for oss, og vi håper arbeidet vårt blir satt pris på av de som skal bruke det.

For gruppen har det vært artig å jobbe med et produkt det faktisk er behov for. Det eksisterer per dags dato ingen tilsvarende løsning, og vi har derfor fått ganske gode tilbakemeldinger fra fagmiljøet. Det har ikke eksistert noen ferdig mal vi kan følge for hvordan produktet skal se ut, og vi har derfor hatt relativt frie tøyler til å utvikle det sånn vi selv føler er best sammen med tilbakemeldinger fra oppdragsgiver og fagmiljøet. Vi sitter ihvertfall igjen med en følelse av at vi har utviklet et solid produkt som med enorme muligheter.

Selvfølgelig har ikke prosjektet vært fritt for problemer og stress, men det er greit å også ha opplevd dette før vi havner i samme situasjon i arbeidslivet. Vi har først og fremst lært om både samarbeid og organisering, men også en god mengde praktiske teknikker. Selv om dette har vært en interessant og spennende tid så skal det bli godt å bli ferdig. Motivasjonen har vært varierende og ville antagligvis dalt hvis prosjektperioden hadde foregått over lengre tid. Å jobbe med prosjekter av denne størrelsen er som nevnt nytt for oss og det er dermed naturlig at vi ikke har vært godt forberedt på det presset som har blitt lagt på oss under prosjektet. Som en avslutning av studiet har dette definitivt vært en glimrende mulighet til å vise hva vi er i stand til i en reell kontekst. Prosjektprosessen har ikke vært feilfri, men vi håper og tror at uttrykket "man lærer av sine feil" stemmer så vi neste gang stiller mye sterkere. Om ikke annet kan vi i det minste påstå at vi har fått en suveren innsikt i arbeidslivet samtidig som vi har opparbeidet oss en solid kompetanse gjennom den opplæring og erfaring vi har fått underveis.

Bibliografi

[1] http://jogl.dev.java.net. Hovedsiden til jogl hvor man kan få lastet ned og finne dokumentasjonen til jogl.

[2] http://www.opengl.org. Industristandarden for 3D-applikasjoner.

[3] http://www.javagaming.org/forums/index.php?board=25.0. Jogl forum.

[4] http://www.javagaming.org. Et forum hvor utviklerne av jogl holder til, har fått mye hjelp derfra.

[5] http://www.codesampler.com. En side som har masse opengl relaterte tutorials og eksempler.

[6] http://jerome.jouvie.free.fr/OpenGl/index.php. En side med mange jogl tutorials og eksempler.

[7] http://nehe.gamedev.net. En side som har masse opengl tutorials og eksempler.

[8] http://java.sun.com. Hovedsiden til java, her finner man dokumentasjonen til java.

[9] http://forum.java.sun.com. Hovedforumet til java.

[10] IRC: opengl kanalen på #Efnet. Programmeringsrelatert diskusjon.

Figurer

Tillegg A

Definisjoner

A.1 Akronymer

1. Kos: Kjøreopplæringssystem
2. DOM (Document Object Model): Er en representasjon av en XML-fil etter en definert standard. DOM er bygd opp som et tre og kan traverseres.
3. DTD (Document Type Definition): Et filformat laget for å definere gyldig innhold i en XML-fil.
4. XML (Extensible Markup Language): Et filformat basert på formateringsspråk. Kan enkelt leses inn i programmer og er enkle og oversiktlige å jobbe med.

A.2 Faguttrykk

1. Klasse: En samling data med tilhørende metoder i en enhet.
2. Refactoring: (en) Ombygging av programmet for å tilpasse det nye krav og nytt design.
3. Inkrement: En logisk adskilt del av helheten i programmet.
4. Scene: Det som vises frem i programmet akkurat nå. (Er mao. en situasjon i et gitt scenario.)

5. Ortho: Dette er en måte å presentere tredimensjonale bilder på som ikke tar hensyn til dybde eller perspektiv i bildet. Dette brukes for å fremstille en illusjon om at det man ser på er helt flatt. I Kos brukes denne teknikken for å vise scenen i en vinkel rett ovenfra.

6. Logisk utseende: Hvordan løsningen er satt sammen del for del.

7. Prosess utseende: Hvordan delene i løsningen jobber sammen.

8. Teksturering: Det vil si at man legger på bilder på modellene, kan sammenlignes med å legge på trekk på en pute.

9. Blueprints: Det vil si skjelett tegninger over objekter, f.eks plantegninger av hus.

10. White box testing: testing av programmet med kildekoden tilgjengelig. Testeren forventer en hvis respons siden han er kjent med hvordan programmet skal fungere.

11. Black box testing vil si at den personen som skal teste programmet ikke har kjennskap til koden, men kan ha en formening om hvordan programmet skal fungere. Testeren kjenner ikke til ønsket virkning, bare input og utput.

Tillegg B

Brukermanual

Kjøreopplæringsystem

KOS

Brukermanual Kos Jonas Strømstad, Sverre Bakke, Kenneth Rinnan

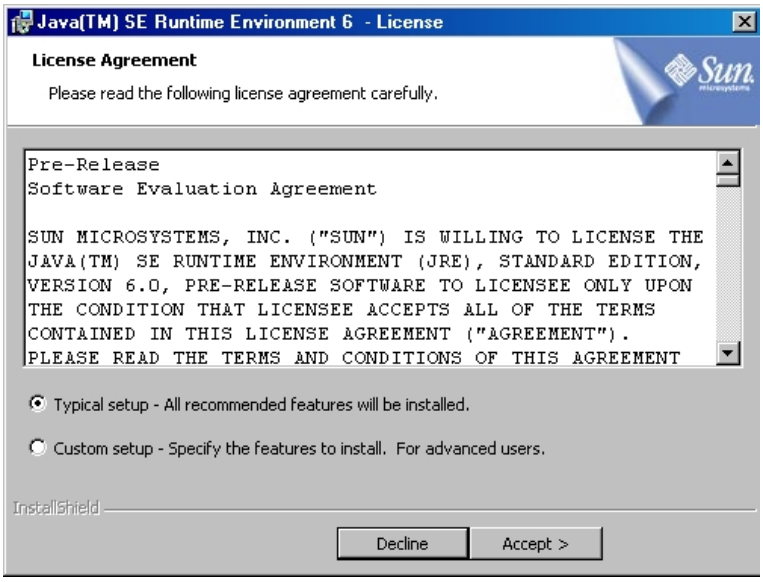
B.1 Introduksjon

Kos er ment for bruk av trafikkskoler. Det skal være til hjelp både i undervisning og for å opprette spørsmål som elevene kan bruke til å teste sine ferdigheter ved forskjellige trafikksituasjoner. Det inneholder store muligheter for å opprette trafikale situasjoner, og gir muligheter for å redigere disse.

B.2 Installasjon

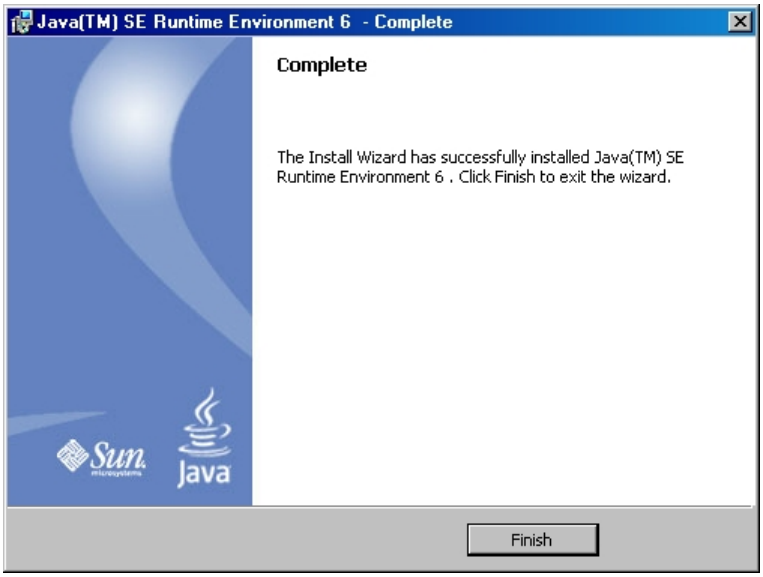
Installasjonen av Kos skal fungere så enkelt som mulig. Det første man gjør er å sette cden inn i cd-romen. Deretter vil installasjonen starte automatisk. Hvis man ikke har Java installert fra før av, så vil man først få opp en installasjonsveileder for å få installert det. Hvis man har Java installert fra før, vil installasjonsveilederen hoppe over det steget. Deretter vil man komme til installasjonen av Kos. For mer info om installasjon av Java se B.2.1. For mer info om installasjon av Kos se B.2.2.

B.2.1 Java installasjon



Figur B.1: Lisensavtale i Java installasjonen.

Dette er det første steget man vil komme til under installasjonen av Java. Hvis man velger «Typical setup», som er anbefalt, og trykker på «Accept», så vil installeringen gå av seg selv. Java vil installere seg til den disken som Windows er installert på. Hvis man ikke ønsker det så kan man velge «Custom setup», og for mer veiledning for hvordan man gjør dette, gå til http://java.sun.com/javase/6/webnotes/install/index.html.



Figur B.2: Installasjonen av Java er fullført.

Hvis man har valgt «Typical setup» så vil dette være det siste man vil få opp, og da trykker man på «Finish», og Java er ferdig installert.

B.2.2 Kos installasjon

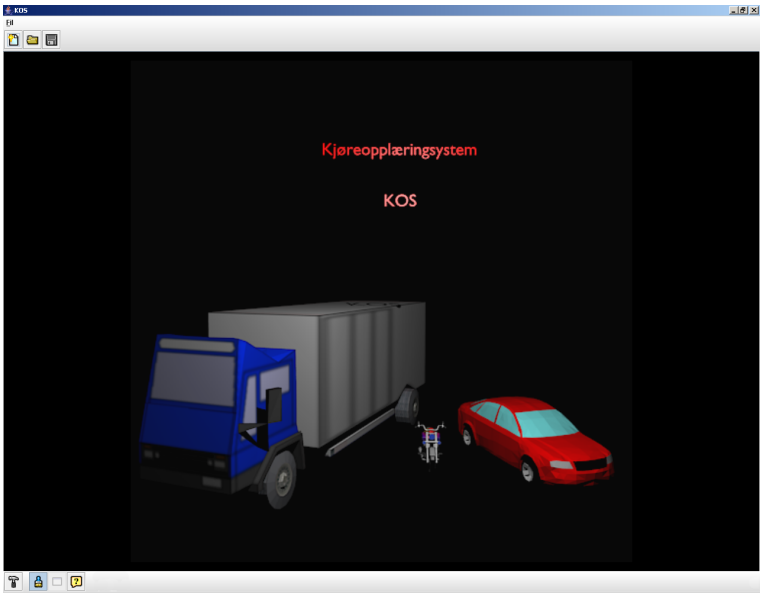
Installasjonen av Kos vil så være det neste man får opp. Det første man vil få opp er en lisensavtale, der trykker man på «Godta», hvis ikke vil installasjonen avbrytes. Det andre man gjør er enten å velge «Standard installasjon» for å la installasjonen gå automatisk, eller «Avansert», forskjellen på disse to valgene er:

- Hvis man velger standard så vil Kos installere seg under samme disken som windows ligger på, og legger seg da under programfiler. Det vil og bli opprettet en snarvei til Kos på skrivebordet automatisk.
- Hvis man velger avansert, kan man endre hvor Kos skal installeres, og man får da og muligheter til å velge om man ønsker snarvei til skrivebordet.

Det vil alltid være mulighet til å velge Kos fra startmenyen.

B.3 Brukerveiledning

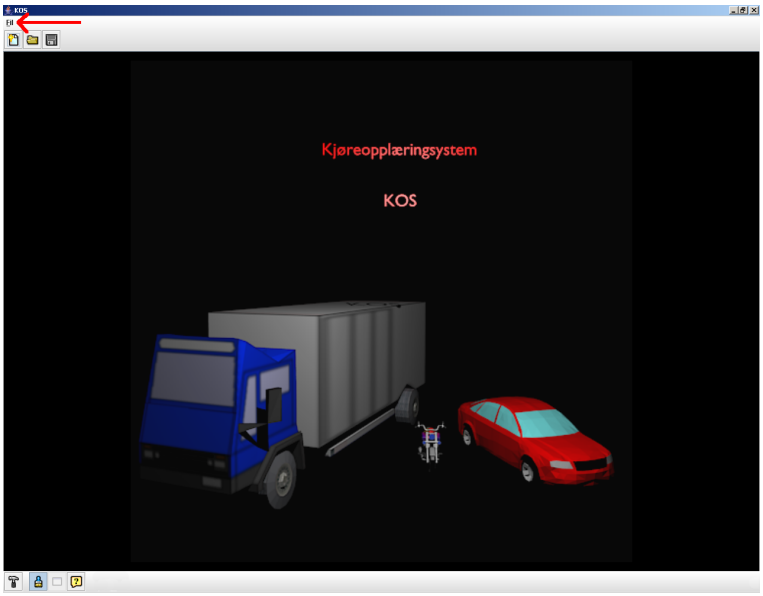
B.3.1 Introskjerm



Figur B.3: Dette er det første bildet man får opp når man starter Kos.

Introskjermen inneholder forskjellige knapper og menyer. Vi skal her se litt nærmere på de forskjellige funksjonalitetene.

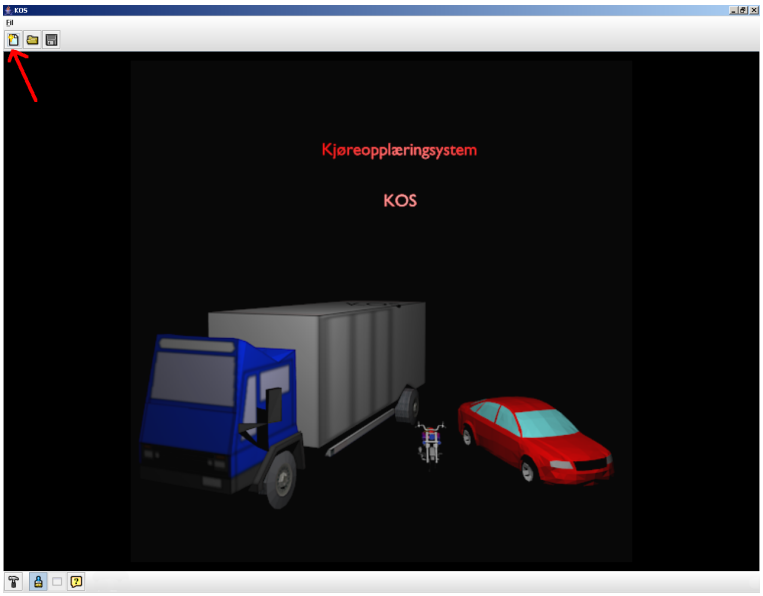
Forklaring til filmeny



Figur B.4: Introskjermen til Kos med filmenyen utpekt.

Der pilen peker er det en filmeny som inneholder de samme valgene som det menyen like under har, men det er ett ekstra valg, og det er å avslutte Kos.

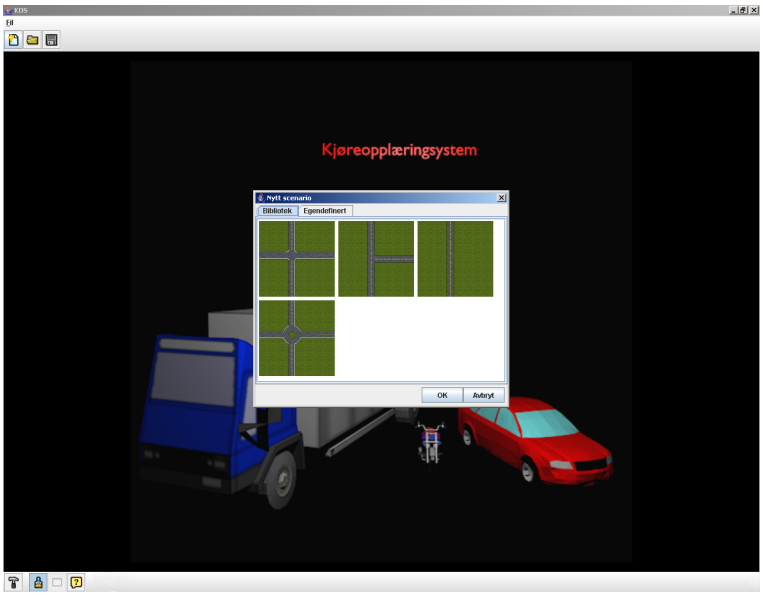
Forklaring til «nytt scenario»-knapp



Figur B.5: Her ligger knappen som oppretter et nytt scenario.

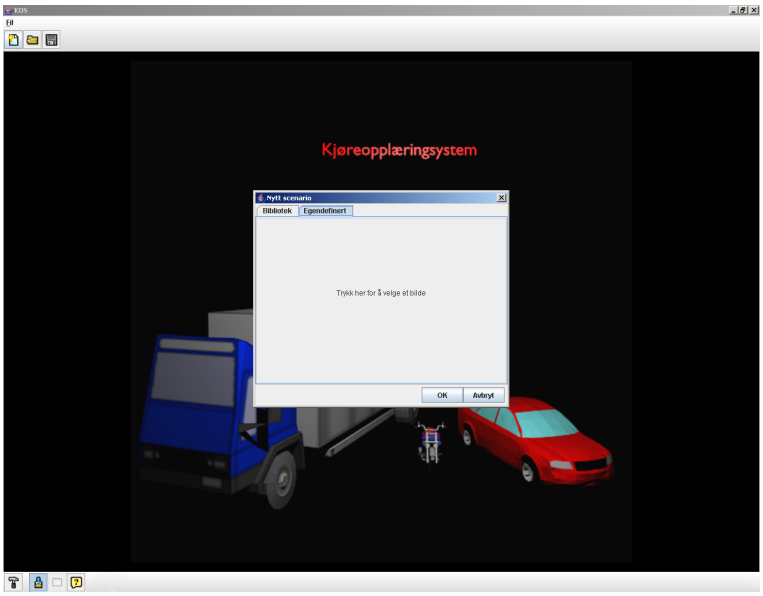
Den knappen som pilen peker på er for å starte et nytt scenario, og den vil da gi brukeren to nye valg, se B.3.1 og B.3.1 for mer informasjon.

Forklaring «nytt scenario»-dialog



Figur B.6: Valg mellom predefinerte bakgrunner.

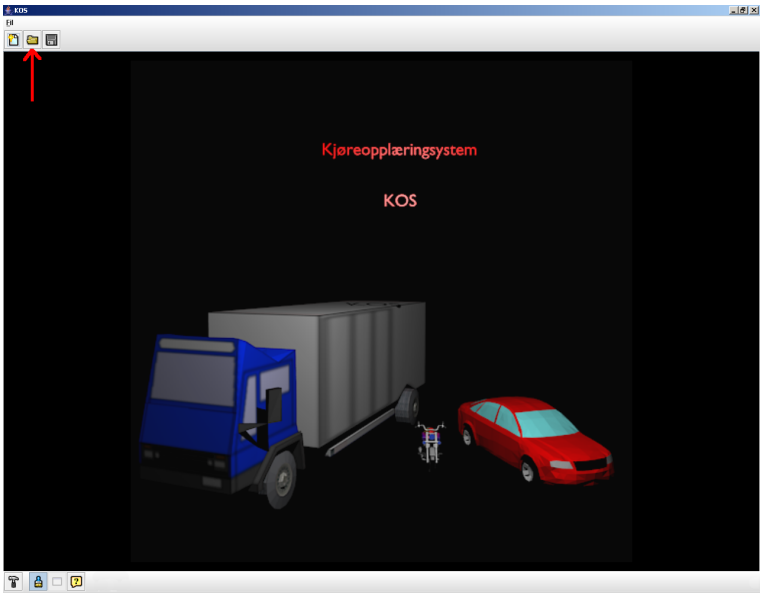
Dette valget vil gi brukeren mulighet til å velge en av de bakgrunnene som følger med Kos.



Figur B.7: Velger egendefinert bakgrunn.

Dette valget vil gi brukeren mulighet til å velge en egendefinert bakgrunn. Ved å trykke på med venstre musknapp, der det står trykk her for å velge bilde, åpnes en fildialog som lar brukeren velge filer av formater som bmp, png og jpg.

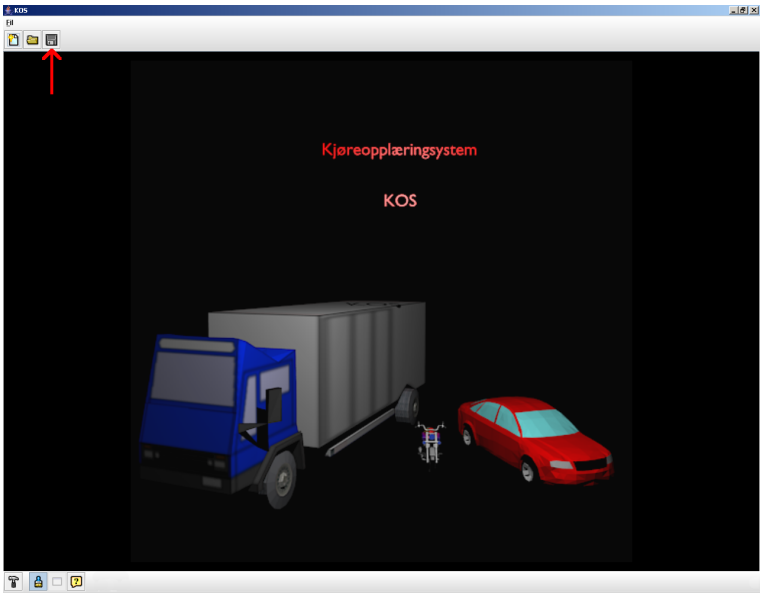
Forklaring «åpne scenario»-knapp



Figur B.8: Her kan man åpne lagrede scenarier eller pakker.

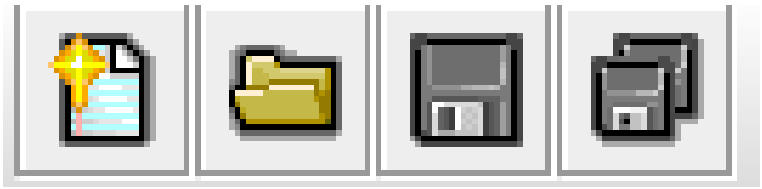
Dette valget vil gi brukeren en fil dialog som gir mulighet til å kunne åpne lagrede scenarier

Forklaring «lagre scenario»-knapp



Figur B.9: Knappen for å lagre alle åpne scenarier i en pakke.

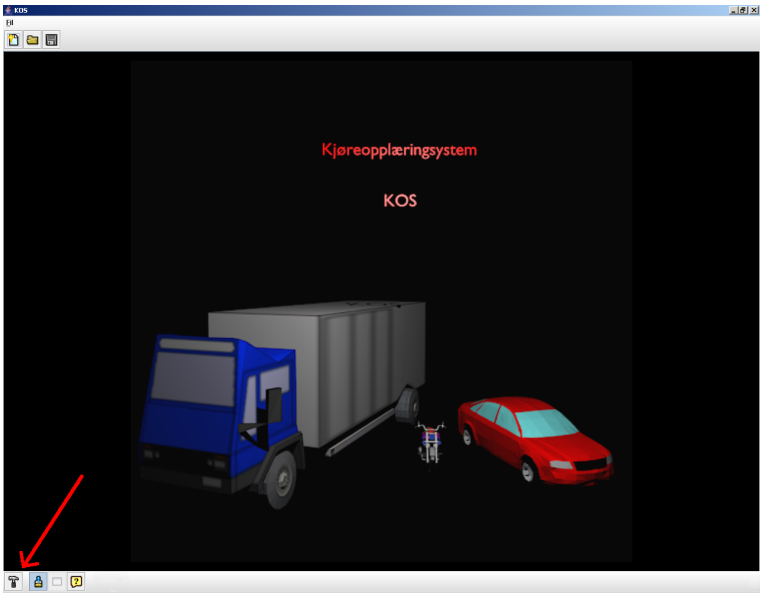
Dette valget vil gi brukeren mulighet til å lagre et scenario til fil, men vil ikke fungere hvis man ikke har opprettet et nytt scenario eller åpnet et som er lagret. Hvis man har valgt pakkemodus vil alle scenarier bli lagret til samme fil. Hvis man derimot ikke har valgt pakkemodus i valgdialogen, vil man få frem en ekstra «lagre» knapp slik som vist på bildet under.



Figur B.10: Knappen for å lagre alle scenarier fortløpende.

Som man ser her har man to lagre muligheter. Lagre knappen til venstre vil lagre valgt scenario til fil, mens lagre knappen til høyre vil føre til at man kan få lagret alle scenariene, men vil da bli spurt om hvor man vil lagre hvert eneste scenario til fil.

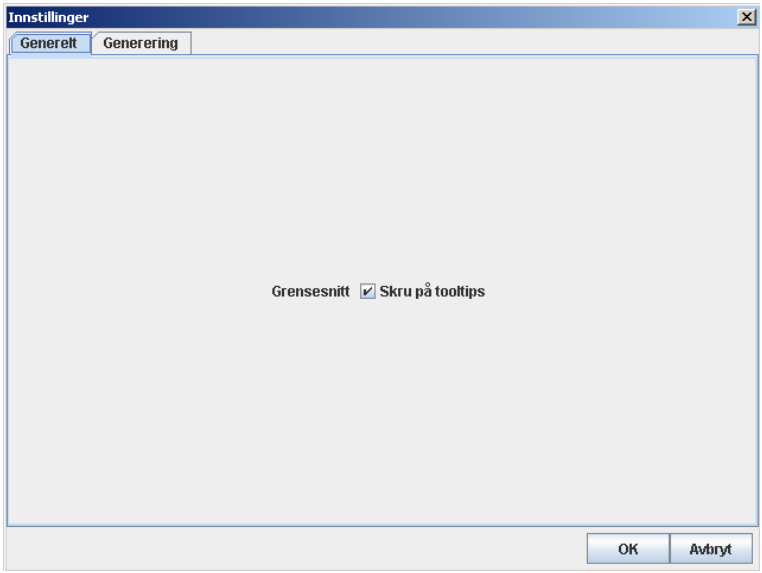
Forklaring «innstillinger»-knapp



Figur B.11: Innstillingsdialogen.

Dette valget vil gi brukeren en options dialog med valg som går på innstillinger av selve systemet. For mer forklaring av de forskjellige valgene under innstillinger se B.3.1 og B.3.1.

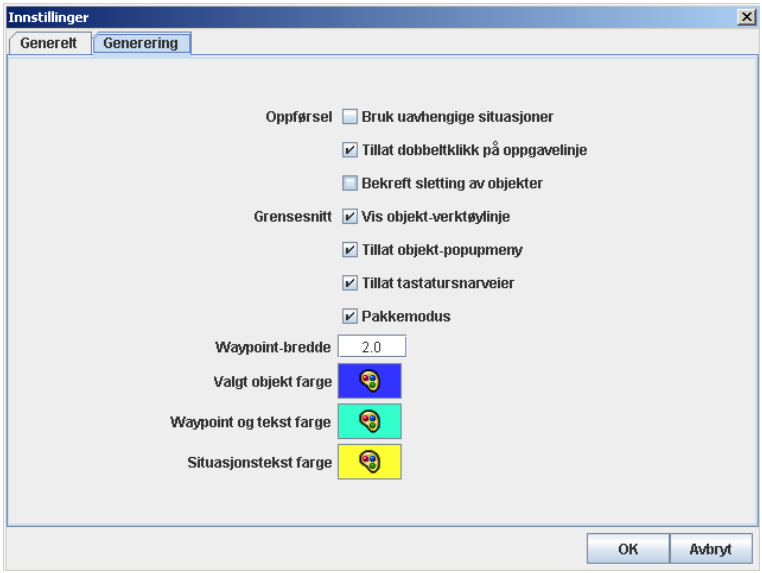
Generelle innstillinger



Figur B.12: Den generelle innstillingsdialogen.

Her har man et valg, og det er Grensesnitt > «Skru på tooltips». Det vil si at man kan slå av og på hjelpeteksten som kommer når man holder musen over en knapp eller et valg.

Genereringsinnstillinger



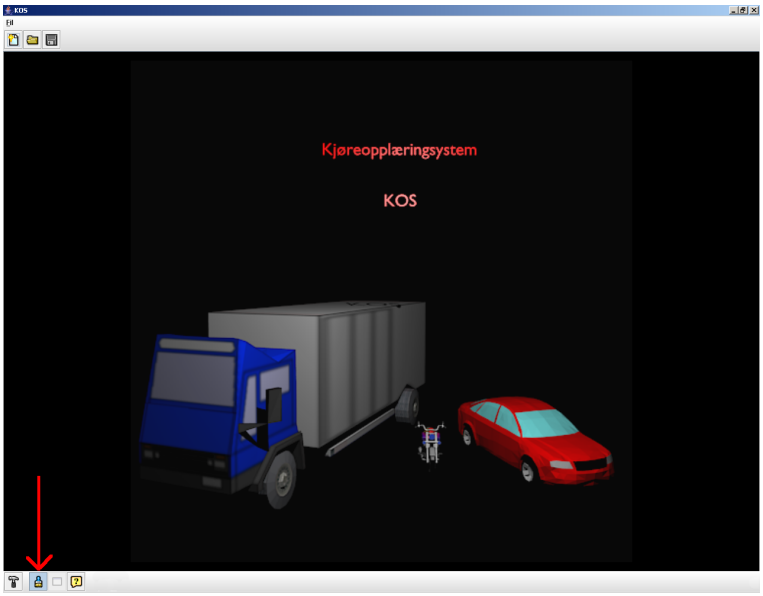
Figur B.13: Innstillingsdialogen for generelle innstillinger.

Her får man flere valg, disse er:

- Bruk uavhengige situasjoner, dette vil gi brukeren valg om å slå av eller på om situasjoner skal henge sammen eller ikke.
- Tillat dobbeltklikk på oppgavelinje, hvis brukeren velger å slå av dette, vil muligheten til å gi navn til scenariet ved å dobbelklikke på fanen som hører til scenariet forsvinne.
- Bekreft sletting av objekter. Med dette valget slått på må alle slettinger av objekter bekreftes.
- Vis objektverktøylinje, dette valget vil bestemme om verktøylinjen for objektene skal være synlig eller ikke.
- Tillat tastatursnarveier, dette valget vil bestemme om man skal kunne bruke knapper på tastaturet til å utføre redigering av objekter.
- Pakkemodus, dette valget velger om man ønsker å lagre et scenario med alle sine situasjoner som en pakke.
- Waypoint-bredde, dette valget setter hvor brede pilene fra objektene skal være
- Valgt objekt farge, dette valget setter hvilken farge som skal brukes på et objekt når det er valgt

- Waypoint og tekstfarge, dette valget setter fargen på tekst og piler for objekter
- Situasjonstekstfarge, dette valget setter hvilken farge som skal brukes på teksten for situasjoner

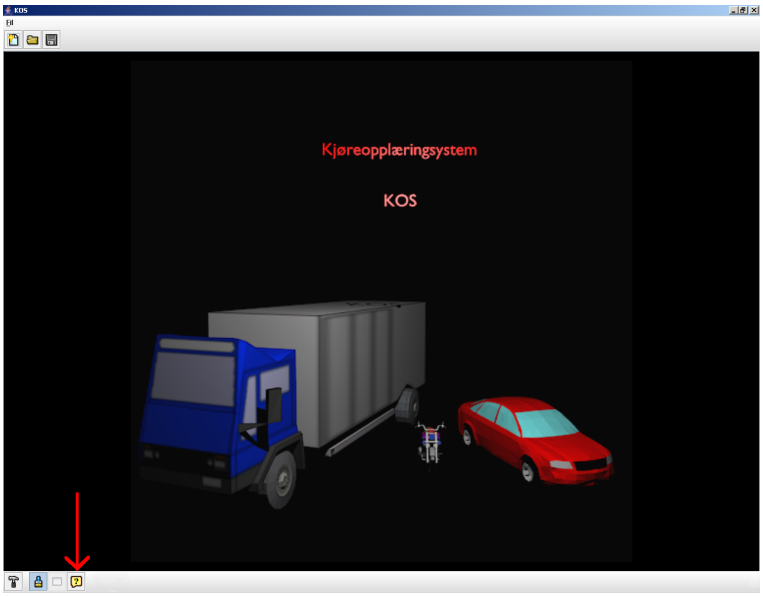
Generatormodusknapp



Figur B.14: Denne knappen setter programmet i generatormodus.

Det vil si at man kan redigere situasjoner. Når man ikke har opprettet et scenario, vil det kun vise introbildet.

Spørsmålsmodusknapp



Figur B.15: Denne knappen vil sette programmet over til spørsmålsmodus.

Hvis man ikke har opprettet et scenario, vil det ikke bli lagt ved noe bilde, men man vil fortsatt ha mulighet til å lage spørmål uten bilder, eller man kan laste opp bilde.

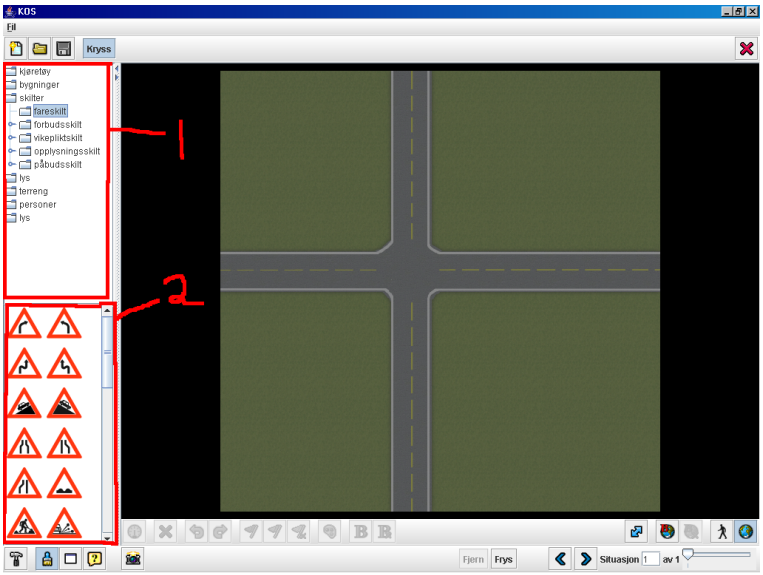
B.3.2 Knappene til scenarioredigering

Figur B.16: Dette er skjermen som vil møte deg når man oppretter et nytt scenario.

De forskjellige knappene og funksjonene er som følger:

1. Disse valgene fungerer som tidligere beskrevet, for mer informasjon se B.3.1
2. Denne knappen er for å fjerne en situasjon fra scenariet, men for at man skal kunne få trykket på den så kreves det at det er opprettet mer enn en situasjon
3. Denne funksjonen er for å opprette en ny situasjon
4. Disse 2 knappene brukes for å bla mellom situasjonene som har blitt opprettet
5. Denne slidebaren er som man kan bla mellom situasjonene ved at man drar den
6. Denne knappen er for å kunne endre størrelse på objektene, men den vil kun være synlig om man bruker et bakgrunnsbilde som ikke er standard i Kos
7. Disse 2 knappene er for å sette situasjonstekst. Den til venstre er for å opprette tekst, og man vil da få opp en dialog for å skrive inn tekst, og har man opprettet situasjonstekst, vil den andre knappen være mulig å trykke på for å fjerne teksten.

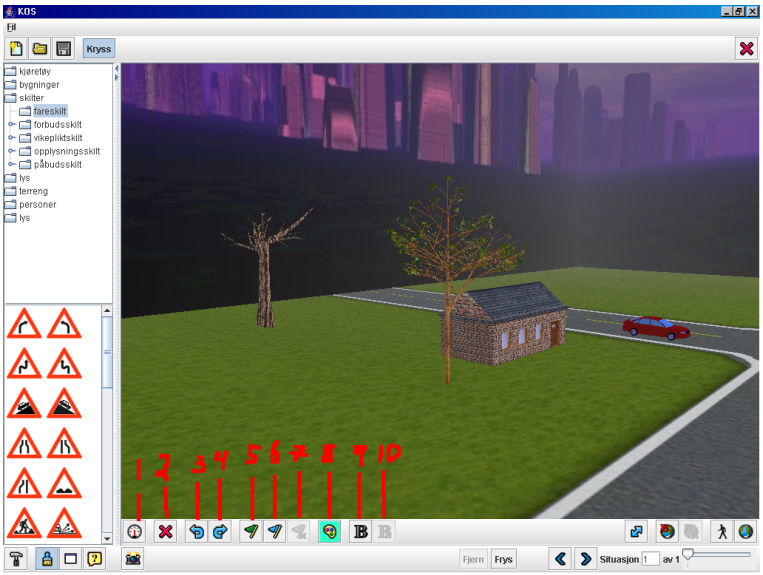
8. Denne knappen er for å kunne lage en fotgjengerovergang, dette skjer ved at man trykker på knappen og trykker på og holder museknappen nede i scenen der man vil den skal begynne og drar den dit man vil den skal gå til, og slipper museknappen
9. Denne knappen er for å skifte mellom 2d og 3d visning av scenariet.
10. Denne knappen holder navnet på selve scenariet, og har man satt tillat dobbelklikking på oppgavelinje i options, kan man dobbeltklikke på den for å endre navnet på scenariet.
11. Dette er en oversikt over alle kategoriene for objekter se B.3.2 for mer informasjon
12. Denne knappen vil ta en screenshot av det situasjonsbildet man har laget seg og vil deretter gå over til spørsmålsmodus og legge ved det situasjonsbildet



Figur B.17: Dette er slik menyen over kategorier ser ut.

Forklaring objekt valg meny Det man ser på nr 1 er oversikten over kategorier, og når man har trykket på noen av de, vil det enten åpne seg en ny underkategori om det finnes det, eller objektene vil bli listet opp slik man ser på 2. Når objekter har blitt listet opp kan man ta tak i de med musen, og dra de inn til scenen der man vil ha de.

Menyen for redigering av objektegenskaper



Figur B.18: Knappene som påvirker objektenes egenskaper.

Hvis man velger et objekt, vil man få frem forskjellige knapper for redigering. De forskjellige knappene fungerer på følgende måte:

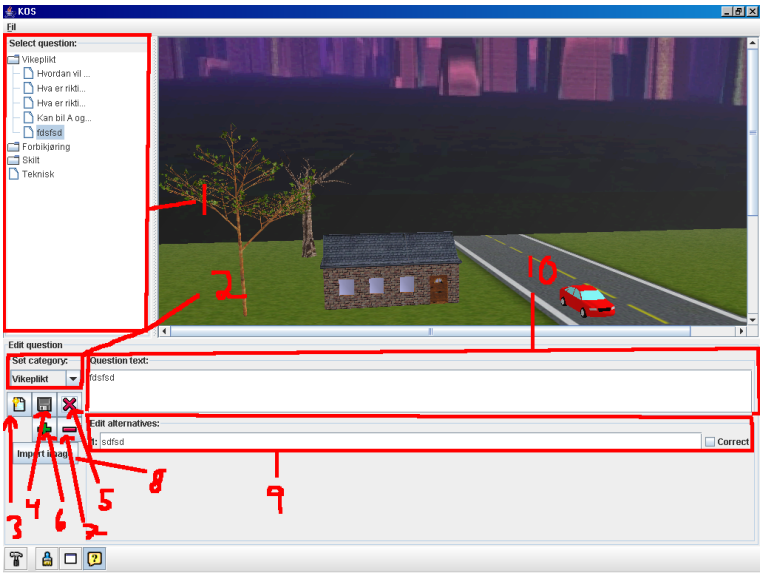
1. Denne knappen er for å gå inn i modellen, fungerer kun for kjøretøy.
2. Denne knappen er for å slette objektet.
3. Denne knappen er for å rotere objektet til venstre.
4. Denne knapper er for å rotere objektet til høyre.
5. Denne knappen er for å tegne en pil på frihånd. Den fungerer slik at man trykker på knappen, trykker og holder museknappen nede der man vil at pilen skal tegnes fra. Deretter drar man musen videre der man vil at pilen skal fortsette og slipper museknappen når man vil at stien skal slutte.
6. Denne knappen er for å få en buet pil. Den fungere slik at man trykker på knappen, og trykker deretter der man vil at stien skal slutte. Så flytter man musen rundt til man har fått den buen man vil ha, og trykker deretter museknappen ned for å fullføre pilen.
7. Denne knappen vil slette en evt. pil.
8. Denne knappen er for å sette farge på teksten og pilen til valgte objekt.
9. Denne knappen er for å sette tekst over objektet

10. Denne knappen er for å slette objektteksten, men krever at det har blitt opprettet en objekttekst

Figur B.19: Knappen som endre trafikklys sine egenskaper.

Trafikklysknappen Hvis man velger et lys objekt vil man på objekt redigerings menyen få et ekstra valg, som man ser helt til høyre. Trykker man på det skifter lyset farge.

B.3.3 Knappene i spørsmålsgeneratoren.



Figur B.20: Dette er skjermen som vil møte deg om du velger spørsmålsmodus.

Den inneholder forskjellige knapper og menyer, og funksjonaliteten til de er som følger:

1. En oversikt over spørsmål som allerede finnes og deres respektive kategorier. Trykker man på en kategori får man listet opp alle spørsmålene under den kategorien, og første spørsmål i den kategorien vises.

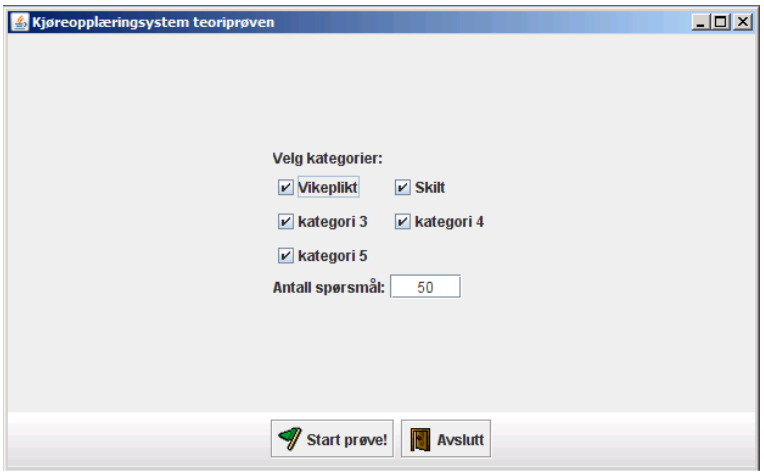
2. Denne nedtrekkboksen lar deg velge hvilken kategori spørsmålet skal være i.
3. Denne knappen er for å lage et nytt spørsmål.
4. Denne knappen er for å lagre spørsmålet.
5. Denne knappen er for å fjerne valgte spørsmål.
6. Denne knappen er for å legge til et nytt spørsmålsalternativ.
7. Denne knappen er for å fjerne et spørsmålsalternativ.
8. Denne knappen er for å hente inn et bilde fra fil.
9. Her vil oversikten over alle spørsmålsalternativene være. Her kan man redigere teksten på alternativene, og man kan ved å trykke på checkboksen til høyre velge hva som skal være riktig svar av alternativene
10. Her redigerer man spørsmålsteksten.

B.3.4 Prøvemodulen

Kos har en egen prøvemodul som gir elever mulighet til å teste seg med spørsmål som har blitt opprettet. Denne modulen er avhengig av at det er opprettet spørsmål i spørsmålsgeneratoren fra før.

Forklaring av oppsett av prøve

Det første som vil møte deg når du starter prøvemodulen er dette:



Figur B.21: Oppsettet av en prøve.

De forskjellige valgene man har er:

- Velg kategorier: her velger man hvilke kategorier man ønsker å teste seg i.
- Antall spørsmål: her velger man hvor mange spørsmål man ønsker å ha i prøven.
- Start prøve!: trykker man på den knappen starter prøven.
- Avslutt: trykker på man på den knappen avsluttes programmet.

Gjennomføringen av en prøve

Figur B.22: Dette bildet viser selve skjermen man får under prøven.

Det inneholder forskjellige muligheter. Det er først eventuelle spørsmålsbilder, nedenfor det er spørsmålsteksten etterfulgt av alternativene som kan velges/avvelges. Nedenfor der har man et sett med knapper, funksjonalitetene til de er som følger fra venstre:

- Avslutt prøve og gå til introskjerm.
- Gå til begynnelsen av prøve.
- Gå et spørsmål tilbake.
- Gå et spørsmål fremover.
- Gå til slutten av prøve.
- Vis riktige alternativer, der vil de alternativene som er riktige bli grønne, mens de som er feil vil bli røde.
- Avslutt prøve.

Fullføringen av en prøve

Resultat skjermen brukes for å gi tilbakemelding for hvordan prøven har gått. Den lister opp forskjellige data om resultatet. Det gis mulighet til å gå gjennom prøven for å se hva man har gjort feil, ta ny prøve, ta samme prøven på nytt eller avslutte. Et eksempel på hvordan det kan se ut er:

Figur B.23: Oppsummeringen av en fullført prøve.

Tillegg C

Filformater

Filformatet til xml-filene vil bli presentert som DTD[1]. I tillegg vil hver post også forklares med ord.

C.1 Filformat for .kos filer

```
<!ELEMENT SCENARIO (CAMERA,SITUATION*)>
  <!ATTLIST SCENARIO name CDATA #REQUIRED>
  <!ATTLIST SCENARIO groundname CDATA #REQUIRED>
  <!ATTLIST SCENARIO skybox CDATA #REQUIRED>
  <!ATTLIST SCENARIO scale CDATA #REQUIRED>
  <!ATTLIST SCENARIO inOrtho CDATA #REQUIRED>

<!ELEMENT CAMERA (POSITION,DIRECTION)>
  <!ATTLIST CAMERA type CDATA #REQUIRED>

<!ELEMENT POSITION EMPTY>
  <!ATTLIST POSITION x CDATA #REQUIRED>
  <!ATTLIST POSITION y CDATA #REQUIRED>
  <!ATTLIST POSITION z CDATA #REQUIRED>

<!ELEMENT DIRECTION EMPTY>
  <!ATTLIST DIRECTION heading CDATA #REQUIRED>
  <!ATTLIST DIRECTION pitch CDATA #REQUIRED>
```

[1] Document Type Definition

```
<!ELEMENT SITUATION (CAMERA*, MODEL*, TOPTEXT?)>
  <!ATTLIST SITUATION no CDATA #REQUIRED>
  <!ATTLIST SITUATION cameraIndex CDATA #REQUIRED>
  <!ATTLSIT SITUATION selectedModelIndex CDATA #REQUIRED>

<!ELEMENT TOPTEXT (#PCDATA, COLOR)>

<!ELEMENT MODEL (POSITION+, DIRECTION?, WAYPOINT?, MODEL*, COLOR?)>
  <!ATTLIST MODEL name CDATA #REQUIRED>
  <!ATTLIST MODEL situationNo CDATA #REQUIRED>
  <!ATTLIST MODEL type CDATA #REQUIRED>
  <!ATTLIST MODEL textEnabled CDATA>
  <!ATTLIST MODEL textHeight CDATA>
  <!ATTLIST MODEL insideEnabled CDATA>

<!ELEMENT WAYPOINT (POSITION+)>

<!ELEMENT COLOR>
  <!ATTLIST COLOR r CDATA #REQUIRED>
  <!ATTLIST COLOR g CDATA #REQUIRED>
  <!ATTLIST COLOR b CDATA #REQUIRED>
```

Filformatet for lagringen av scenarier er ganske rett fram å forstå unntatt når det gjelder måten modellene blir lagret på. Strukturen med at en modell er en kopi av en annen i løsningen gjør at det på en måte må representeres i fila hvilke modeller som er kopi av hvilke. Dette gjøres ved å lagre kopiene som barn av orginalen. På den måten kan man opprette kopiene med en gang når filen leses inn igjen, uten at man behøver å måtte implementere noen tyngre algoritmer. Når en modell leses inn så sjekkes det da bare på om denne har noen barn. Hvis den har det så kopieres bare modellen og legges til i den situasjonen som den hører til. Her er et eksempel på hvordan en scenariofil kan se ut:

```
<SCENARIO name="Situasjon nr. 1" groundname="ground/ground1.png"
  skybox="skybox/city1" scale="1.0" inOrtho="false">
  <CAMERA type="ortho">
    <POSITION x="20.5" y="-40.0" z="32.4"/>
    <DIRECTION heading="45.0" pitch="32.2"/>
  </CAMERA>
  <SITUATION no="0" cameraIndex="1" selectedModelIndex="2">
    <CAMERA type="free">
      <POSITION x="16.8" y="-2.3" z="76.1"/>
      <DIRECTION heading="0.0" pitch="12.2"/>
    </CAMERA>
```

```
    <CAMERA type="inside">
      <POSITION x="43.4" y="-1.5" z="12.1"/>
      <DIRECTION heading="322.5" pitch="0.0"/>
    </CAMERA>
    <MODEL name="/models/vehicles/truck.txt" situationNo="0"
      type="Model3D" textEnabled="true" textHeight="2.0" insideEnabled="true">
      <POSITION x="12.0" y="0.0" z="32.3"/>
      <DIRECTION heading="245.3" pitch="0.0"/>
      <COLOR r="255" g="0" b="0"/>
      <MODEL name="/models/vehicles/truck.txt" situationNo="0"
        type="Model3D" textEnabled="true" textHeight="2.0" insideEnabled="true">
        <POSITION x="12.0" y="0.0" z="32.3"/>
        <DIRECTION heading="245.3" pitch="0.0"/>
        <COLOR r="255" g="0" b="0"/>
        <WAYPOINT type="bezier">
          <POSITION x="12.0" y="0.0" z="36.3"/>
          <POSITION x="20.4" y="0.0" z="48.8"/>
          <POSITION x="23.6" y="0.0" z="65.2"/>
        </WAYPOINT>
      </MODEL>
    </MODEL>
  </SITUATION>
  <SITUATION no="1" cameraIndex="2" selectedModelIndex="0">
  *Redigert bort*
  </SITUATION>
</SCENARIO>
```

Når det skal lagres i en pakkefil puttes bare DOM-treene fra hvert scenario inn i et packageelement. DTDen for dette blir da:

```
<!ELEMENT PACKAGE (SCENARIO)*>
```

Filformatet er ellers uforandret.

C.2 Filformat for spørsmålsfila

```
<!ELEMENT qdb (CATEGORY)*>

<!ELEMENT CATEGORY (#PCDATA,QUESTION*)>

<!ELEMENT QUESTION (#PCDATA,ALTERNATIVE)
```

```
 <!ATTLIST QUESTION imgurl CDATA #REQUIRED>

<!ELEMENT ALTERNATIVE (#PCDATA)>
  <!ATTLIST ALTERNATIVE CORRECT CDATA #REQUIRED>
```

Fila blir dermed bestående av en rekke kategorier med sine spørsmål som igjen har sine alternativer. Et utdrag fra en fil kan dermed bli seende sånn ut:

```
<qdb>
  <CATEGORY>
    Vikeplikt
    <QUESTION imgurl="user/1147175676769.png">
      Kan bil A og B kjøre samtidig?
      <ALTERNATIVE CORRECT="FALSE">
        Nei
      </ALTERNATIVE>
      <ALTERNATIVE CORRECT="TRUE">
        Ja
      </ALTERNATIVE>
    </QUESTION>
    <QUESTION imgurl="user/1147172985479.png">
      Hva er riktig?
      <ALTERNATIVE CORRECT="'TRUE">
        Lastebil B har vikeplikt for bil A
      </ALTERNATIVE>
      <ALTERNATIVE CORRECT="FALSE">
        Bil A er på forkjørsveg
      </ALTERNATIVE>
    </QUESTION>
  </CATEGORY>
</qdb>
```

Tillegg D

Teori om utviklingsverktøy

D.1 Introduksjon

Dette dokumentet skal fortelle litt om hva vi har brukt for å utvikle dette produktet, samt hvilke teknologier vi kunne ha brukt istedet for de vi valgte. Vi skal og se litt nærmere på det miljøet vi valgte å utvikle det i. Vi skal og se på de verktøyene som vi har brukt for å lage modellene, samt de kildene vi har brukt under utviklingen.

D.2 IDE

Vi har utviklet alt i vim(vi improved) som er et veldig kraftig tekst editor som har muligheter for det meste. Vi brukte en grafisk versjon av vim som heter gvim.

D.2.1 Introduksjon til vim

Vim er en litt avansert tekst editor, som gir muligheter for å gjøre det meste, det har og muligheter for å lage scripts for å lage nye funksjoner. Vim har lenge hatt en lang tradisjon på unix platformen. Grunnen til at vi valgte å utvikle med vim var at det ikke krever noe særlig av maskinvare, samt at det er så kraftig(muligheter for folding og gode søke og erstatnings muligheter. Vi hadde

i tillegg alle kjennskap til vim, og har brukt det tidligere.

D.3 Klassebiblioteker

Et klassebibliotek er en samling med filer som man benytter seg av for å gjøre det enklere å opprette f.eks knapper og andre komponenter i et program, men kan og være filer som man bruker for å kunne bruke 3D i programmet, f.eks jogl. Det finnes utallig mange biblioteker for å gjøre mange forskjellige oppgaver.

D.3.1 Java

Java var det utviklingsmiljøet vi endte opp på, dette på grunn av at vi har utviklet en del i det, samt at det er veldig enkelt og opprette gui applikasjoner i det. Det har og muligheter for å enkelt kunne internasjonalisere programmer, noe som er veldig nyttig for vårt formål. Java er og platformuavhengig, noe som da vil føre til at vi skal kunne kjøre programmet på flere plattformer, men vi har hatt hovedfokus på Windows platformen.

D.3.2 openGL

Jogl er basert på openGL(open graphics library) som er et bibliotek man kan benytte seg av for å tegne opp enkle primitiver(firkanter, trekanter, linjer, punkter) i 3D. OpenGL har ikke muligheter til å kunne opprette et vindu i seg selv. Det er grunnen til at vi brukte java til å ta seg av den jobben. Det finnes i tillegg masse ferdig biblioteker for å gjøre det enkelt og dra inn modeller og slikt, men vi har ikke benyttet oss av det, mest på grunn av at vi ikke fant noe som var egnet til det vi skulle lage. OpenGL har lenge blitt brukt til å lage spill og andre 3D programmer. openGL har tapt ganske mye av markedet på grunn av direct3D, men openGL er fortsatt et fullverdig og veldig lavnivå bibliotek som fortsatt blir brukt mye innenfor forskning, og det blir fortsatt brukt til både å lage spill og andre 3D programmer. OpenGL er og veldig portabelt, noe som og var noe av grunnen til at vi valgte openGL fremfor direct3D.

Jogl

Jogl er opengl bindinger til java fra sun, det vil si at man kan programmere 3D i java. Jogl er ytelsesmessig ganske bra til sammenlikning med det samme programmet skrevet i c. Jogl er BSD lisensiert, noe som vil si at det er veldig lite restriksjoner på hvordan man bruker det. Jogl ble sluppet første gang i 2003 som et av første prosjektene på java.net. Etter hvert som vi begynte å programmere i jogl, støtet vi på flere problemer, spesielt problemer med kompatibilitet med forskjellig hardware. Problemene førte til at vi ble nødt til å finne andre løsninger slik at programmet skulle fungere på både gamle og nye pc'er. Vi så på andre løsninger som kunne være aktuelt i stedet for jogl, men flere av de hadde og mangler, eller det ville være for tidkrevende og lage gui, eller det var kun for å lage spill. Noen av disse var:

- xith3D - Som er et spill bibliotek basert på jogl, men det var for å lage spill, og ville derfor inneholde for mye som vi ikke trengte.
- java3D - Det kunne fungert ok, men da vi begynte å se på det, hadde vi kommet for godt i gang med jogl.
- Lwjgl - Lightweight java gaming library som er basert på opengl og kjører på java. Men det passet heller ikke for vårt formål
- openGL - Bruke openGL direkte på c/c++ og bruke en api som f.eks sdl eller win32API under for å ta seg av vindushåndteringen, men det ville vært for tidkrevende og sette seg inn i og programmere det grafiske slikt som knapper og tekstfelt.
- direct3D - Ville fungert til det vi skulle lage, men i og med at vi hadde sett bitte litt på opengl fra før av og for at det skulle være mulighet for platform uavhengig het, samt lage lage gui ble det valget sløyfet.

Etter å ha vurdert en del frem og tilbake kom vi frem til at jogl ville være det beste valget for oss.

D.4 Modellering

Modellering vil si at man bruker diverse verktøy for å grafisk kunne tegne opp og definere modeller, for så å lagre informasjon om hvordan modellene ser ut til filer. Dette for å gjøre det enkelt å lage objekter som kan brukes i programmet slik at man slipper å måtte definere hvordan modellen skal være manuelt. En modelleringsfase består av mer enn å bare si hvordan formen på modellene

skal være, man må og teksturere, og der har vi prøvd oss frem med forskjellige programmer, men vi valgte da av de som var gratis.

D.4.1 Modelleringsverktøy

Det finnes utallig mange forskjellige modelleringsverktøy. Forskjellen på de er at de varierer i pris, og hvor mye verktøy de har for å tegne modellene. Vi har i løpet av dette prosjeketet holdt oss til noen kjente modelleringsprogrammer. Vi hadde ikke behov for så veldig avanserte modeller, så vi hadde derfor mulighet til å velge mellom de fleste modelleringsverktøy som finnes, vi valgte derfor ut de vi mente ville passe best til oss med hensyn til pris og hvor enkelt det var å lære seg de.

Introduksjon til Milkshape3D

Milkeshape3D er et av modelleringsverktøyene som vi valgte å bruke, mest på grunn av at vi trengte kun enkle modeller, noe som milkshape3D er ment for. Milkshape3D var og et veldig billig valg, kostet 200 norske kroner pr lisens. Milkshape3D er et modelleringsprogram som er enkelt å sette seg inn i og bruke. Det fungerer ypperlig for enkle modeller, men har noe mindre funksjoner for å redigere modeller i forhold til 3D studio max, blender og maya, men har den funksjonaliteten vi trengte for å lage modellene. Milkshape3D har et veldig enkelt utseende, og gir muligheter for animasjoner hvis man ønsker det.

Introduksjon til Blender

Blender var et annet modelleringsvertøy som vi brukte. Blender har mye mer verktøy for å lage modeller enn det milkshape3D har, og det er i tillegg gratis. Men det krever mye mer tid å sette seg inn i, men har muligheter for å kunne legge til nye funksjoner med python script for å redigere modeller. Vi brukte blender til å teksturere noen modeller, men lithunwrap var bedre til teksturering. Blender støtter animasjoner. Det har et ganske vanskelig brukergrensesnitt.

Introduksjon til LParser

LParser var et program som vi brukte sammen med Blender for å enkelt kunne generere organiske modeller slik som trær og busker. Det er veldig lite, og fun-

gerer slik at man lager seg en tekst fil hvor man skriver inn forskjellige formler for hvordan modellen skal se ut. Deretter bruker man LParser til å genere en modell ut fra den filen. Modellen kan så dras inn i blender eller lithunwrap for teksturering.

Introduksjon til Lithunwrap

Lithunwrap er et teksturerings program som vi brukte. Det fungerte veldig enkelt, og fungerte veldig godt sammen med milkshape3D. Lithunwrap gjør det veldig enkelt å få lagt utover f.eks en sylinder slik at man enkelt kan få lagt på en tekstur.

Introduksjon til Gimp

For å lage teksturene som vi brukte på modeller og bakker og lignende, brukte vi gimp som er et tegne program. Gimp er gpl lisensiert, og er gratis. Det funker veldig greit for å lage teksturer for modeller, og har en god del funksjoner for å opprette gode teksturer.

D.4.2 Teknikker som vi tilegnet oss

Etter hver som vi fikk modellert og teksturert litt, kom vi frem til bedre og mer effektive metoder for å utføre oppgavene på.

- Bruke bokser og sylindre og andre ferdige objekter og bruke/redigere de for å sette de sammen til objekter.
- Når man bruker blueprints, passe på å få de riktig skalert med en gang.
- At man deler opp et objekt i flere grupper med en gang, slik at det blir enklere og få teksturert objektene, dette på grunn av at da kan man velge ut en gruppe og teksturere den, så slipper man å gruppere når man er ferdig.
- At når man har modeller f.eks en bil, deler man det i to, og speiler den ene siden slik at modellen blir identisk på begge sidene.

Tillegg E

Visjonsdokument

E.1 Introduksjon

Kos er et interaktivt opplæringsprogram som er ment for å benyttes i trafikkskole-undervisning, i tilegg til at det kan benyttes for å kunne generere spørsmål til prøver og sette sammen spørsmålspakker som så kan legges ut på web. Det skal i sin helhet fungere som et verktøy for å gjøre hverdagen lettere både for kjøreskole lærere og elever.

E.2 Posisjonering

Kos skal være et program som er ment for å kunne optimalisere opplæringen på kjøreskoler samt å kunne innføre flere muligheter for å kunne tilpasse opplæringen til trafikksituasjoner som vil være relevante for nærområdet, dvs at man vil kunne gi elevene opplæring i de trafikksituasjonene som de vil møte daglig, selv om de bor i en tettby eller i en utkant. Kos er ikke ment for å ta over all undervisning, men skal være et verktøy som kjøreskole lærere skal kunne benytte i hverdagen for å kunne optimalisere undervisningen, samt gi elevene mer reelle trafikksituasjoner enn ferdig definerte situasjoner.

E.2.1 Problemområde

Dagens løsninger ved opplæring til førerprøven lider av å være tidkrevende, begrenset og lite tilpasset for lokalmiljøet. Det stiller sterke krav til både elever lærere om oppmøte og tidsbruk.

Problemet med dagens løsninger består blant annet av: Tavleundervisning med tegning og magnetiske biler tar tid og setter begrensninger i hvilke situasjoner man kan presentere på en forståelig måte. Denne typen undervisning kan lett føre til seg kommunikasjonsproblemer og elever som har vanskelig for å se for seg situasjonen/problemstillingen vil derfor ikke kunne ta til seg ønsket kunnskap som er nødvendig for å fullføre førerprøven med tilfredsstillende resultat. Det er heller ikke ønskelig at elever skal bruke mye tid på å tolke situasjonen/problemet fremfor å løse det. Det er også svært tidkrevende for læreren å måtte tegne opp samme situasjon gang på gang uten noen mulighet til å ta vare på tidligere arbeid.

- Ferdiglagde spørsmål og situasjoner fra sentralt hold som for eksempel ATL vil være generelle og ikke tilpasset for lokale situasjoner. Elever vil derfor i stor grad miste muligheten til å få tilpasset undervisning etter behov.
- Lite fleksible løsninger for selvstudier på grunn av mangelfulle interaktive løsninger. Dette vil først og fremst gå ut over elever som ønsker å jobbe på egenhånd med oppgaver fremfor å lese teori, og gjerne til egendefinerte tider og lokasjoner. Det er viktig at det ikke oppleves som ensformig slik at det oppstår lav arbeidsmoral og selvstudiene avtar.

Vi mener at ved å utnytte datamaskiner og Internett kan man få en mer tilpasset og fleksibel løsning. En slik løsning bør blant annet inneholde muligheter for å:

- Digitalt kunne generere og redigere varierte situasjoner raskt, enkelt og brukervennelig fra et bibliotek med vanlige trafikale objekter.
- Kunne se genererte situasjoner fra forskjellige synsvinkler etter behov.
- Lagre og innhente situasjoner for senere bruk.
- Legge på spørsmålstekst og svaralternativer til situasjoner og distribuere det til elever over internett.
- Kunne besvare spørsmål med et interaktivt grensesnitt som under hele prosessen gir tilbakemeldinger fra sin egen hjemme-pc.

E.2.2 Produktposisjonering

Applikasjonen er først og fremst rettet mot trafikkskoler som ønsker å tilby elevene sine en forbedret undervisning som er tilrettelagt og tilpasset for lokalmiljøet.

Kos er et opplæringssystem rettet mot norske trafikkskoler som ønsker å tilby elevene sine en forbedret undervisning som er tilrettelagt og tilpasset lokalmiljøet. Produktet tilbyr blant annet:

- Muligheten til å enkelt kunne generere og redigere trafikale situasjoner med et enkelt dra-og-slipp grensesnitt for både presentasjon og prøver.
- Et stort bibliotek av trafikale objekter som kan brukes for å presentere ulike situasjoner.
- Muligheten til å se situasjoner fra forskjellige synsvinkler ved å benytte 3D teknologi.
- Lagrings og innhentings -funksjonalitet av situasjoner for å spare tid.
- Enkelt grensesnitt for å kunne digitalt distribuere spørsmål til elever for selvstudier.

I motsetning til andre digitale opplæringssystemer for førerkortet som benytter ferdige bilder satser vårt produkt på å faktisk kunne lage situasjonene og spørsmålene selv etter behov med et enkelt grensesnitt. Produktet kan derfor i større grad brukes til ulike deler av opplæringen fremfor å bare fokusere på ferdigdefinert teori og oppgaver.

E.3 Interessenter

Kjøreskole lærere som vil kunne benytte dette som et værktøy i undervisningen. Elever som vil være slutt brukerne, de vil og kunne ta tester som har blitt generert og vil dermed kunne bruke Kos utenom klasserommet i prøveform.

E.3.1 Interessent sammendrag

Navn	Beskrivelse	Ansvar
Kjøreskole lærer	Kjøreskolelæreren er den personen som underviser.	Kjøreskolelæreren vil kunne bruke Kos under undervisningen for å kunne vise elevene forskjellige situasjoner, de kan opprette situasjonene på forhånd eller kunne opprette situasjoner og ta de frem under undervisningen, men da og kunne flytte objekter og slikt etter hvert som det vil passe seg. De vil og ha mulighet til å kunne opprette spørsmålspakker som de kan introdusere på web, slik at andre har mulighet for å kunne bruke de i sin undervisning.
Eleven	Eleven er persjonen som deltar i undervisningen.	Eleven vil gjennom Kos få en mer tilpasset og interaktiv undervisning. Eleven vil og ha mulighet til å kunne ta prøver som kjøreskolelærere har introdusert på web, og de vil kunne teste seg selv på disse prøvene fra en hvilken som helst pc som møter systemkravene og er tilkoblet web.

E.3.2 Bruker område

For kjøreskolelærere vil Kos bli brukt i klasserommet, mens for elevene vil Kos kunne benyttes både i klasserommet, men og hvor som helst hvor det er en pc som er tilknyttet web og møter systemkravene.

E.4 Produkt Oversikt

Kos vil inneholde forskjellige moduser:

- Selve 3D generatoren som vil kunne benyttes interaktivt i undervisningen for fremvisning av enten ferdig lagde situasjoner som ligger på disk som man kan dra inn og redigere videre på under undervisningen.
- Spørsmålsgeneratoren som gir mulighet for å kunne legge på spørsmålstekst til situasjonene for å kunne gi mulighet til elevene å svare.
- SpørsmålsPakkeGeneratoren som gir mulighet til å sette sammen mange spørsmål som har blitt generert av spørsmålsgeneratoren til en pakke og publisere den på web.
- SpørsmålsTageren som er den delen elevene vil kunne ta i bruk selv, som gir muligheter for å kunne dra inn spørsmålspakker generert av spørsmålspakkegeneratoren. Den vil og gi tilbakemelding til elevene om hvordan testen har gått.

E.4.1 Produkt Perspektiv

Kos er et enkeltstående produkt som er uavhengig av andre systemer rundt seg. Alle data som skal behendles hentes enten fra filer knyttet til Kos eller fra brukeren.

E.4.2 Features

Kos skal gi trafikkskolene et verktøy til bruk i undervisningen. Dette verktøyet skal brukes til å vise ulike trafikksituasjoner på ulike måter i tredimensjonal grafikk. Verktøyet må fylle følgende krav:

- Være enkel og rask i bruk.
- Ha tydelige og gode modeller av de objektene som påvirker avgjørelser i trafikken.
- Kunne kjøre på en gjennomsnittlig datamaskin uten problemer.
- Ha enkelt grensenitt for deling av pensumrelaterte spørsmål med andre trafikkskoler som benytter seg av systemet.
- Eleven får et brukervennlig grensesnitt for å gjennomføre en prøve.

Kos vil ha en god del nyttige features. Blant disse er genereringen av en tredimensjonal situasjon en av de viktigste. Komplett liste over hovedfeatures i Kos:

- Egen generator for tredimensjonale trafikksituasjoner.
- Fullskjermsvisning av trafikksituasjoner, tilpasset visning på storskjerm.
- En generator for spørsmål. Disse kan være relatert eller uavhengige av trafikksituasjonene.
- Eksportering av trafikksituasjoner til et bildeformat tilpasset web. Disse kan knyttes til spørsmålene som lages i en annen del av Kos .
- Mulighet til å sette sammen egne spørsmål til en samlet teoriprøve som elevene kan bryne seg på.
- Automatisk generator av teoriprøver som setter sammen spørsmål basert på den kategorien spørsmålet er tilknyttet.
- Opp og nedlasting av spørsmål til web.
- Webgrensesnitt som lar elevene ta prøver hjemmefra.

E.4.3 Konkurrenter

Kos har i dag ingen reelle konkurrenter. All annen programvare myntet på trafikkskolene og deres aktivitet er laget med utgangspunkt i en rekke forhåndslagde spørsmål hvor selve programmet utføres ved å svare på disse. Mange av disse programmene finnes som enkle nettsider. Begrensningen ved et sånt system er at en vil ha en veldig begrenset mengde variasjon i hvilke spørsmål som stilles fordi oppdateringen av spørsmål avhenger av en liten gruppe eller enkeltpersoner. Den mest kjente konkurrenten er programmet "Veien til førekortet" som distribueres av ATL (Autoriserte Trafikkskolers Landsforbund) og er laget av TrafiData.

Tillegg F

Forprosjektrapport

F.1 Mål og rammer

F.1.1 Bakgrunn

Prosjektet er basert på en idé unnfanget av Frode Haug, lærer ved HiG, som har vært luftet for hovedprosjektkandidater med jevne mellomrom i ettertid, uten at det har blitt realisert. Prosjektgruppa som skal gjennomføre prosjektet våren 2006 ble kontaktet høsten 2005 og presentert for problemstillingen som de syntes var interessant. I løpet av tiden rett før jul ble Thomtes Trafikkskole på Gjøvik kontaktet med forespørsel om de kunne stille ressurspersoner tilgjengelig for prosjektgruppa, noe de sa seg villige til.

F.1.2 Prosjektmål

Hovedmål

Det skal utvikles en programvare som kan benyttes i store deler av den trafikkmessige undervisningen ved norske trafikkskoler. Denne skal kunne brukes til å generere trafikksituasjoner i full tredimensjonal (3D) grafikk som senere kan presenteres via programvaren i fullskjerm i klasserom (via projektor e.l.) eller som grunnlag for å lage teorispørsmål via den innebygde spørsmålsgeneratoren. Disse kan senere settes sammen til tester med spørsmål fra ulike kategorier. Det

skal også være mulig å endre på trafikksituasjonene underveis i en presentasjon ved å legge til, fjerne eller flytte objekter i bildet.

Effektmål

Denne programvaren skal endre måten trafikkskolene bruker datamaskiner i sin undervisning, samtidig som det skal gi dem et kraftig verktøy som øker kvaliteten på undervisningen og gjør den mer effektiv. Muligheten for å kunne laste opp og ned spørsmål via Internet kan endre måten spørsmål til teoriprøven og tentamensprøvene blir laget. Prosjektet skal også fungere som et utstillingsvindu for deltagerne og være noe de skal være stolte av å ha vært med på, samtidig som det gir fremtidige arbeidsgivere informasjon om prosjektdeltagernes kunnskaper og ferdigheter.

Prosessmål

Prosjektet skal gi de som deltar erfaring med å jobbe på et større prosjekt. De skal få erfaringer og kunnskaper om hvordan man jobber et en spesifikt gitt systemutviklingsmodell. På denne måten skal prosjektdeltagerne bli bedre skikket til å møte utfordringene som finnes i arbeidslivet etter endt utdanning. Den viktigste utfordringen, med rot i det virkelige arbeidslivet, er forholdet mellom utviklerne og en reell oppdragsgiver. Vi får erfaring med hvordan det er å kommunisere og samarbeide med en annen part. Deltagerne skal få erfaring med å måtte tilegne seg fagkunnskap underveis i prosjektet. Gjennom prosjektet skal deltagerne få gode kunnskaper og evner innen Java-programmering, både innen enkle input/output operasjoner, men også mer avansert ved den tredimensjonale modulen.

Målgruppe

Målgruppen er trafikkskolelærere som ønsker å nå frem til sine elever på en ny måte. Et suksessfull programvare av denne typen kan fort bli en del av hverdagen på norske trafikkskoler. Elevene ved trafikkskolene er også en målgruppe for den delen av programmet hvor man besvarer tester.

F.1.3 Rammer

Praktiske rammer

Prosjektet skal utvikles i samarbeid med Thomtes Trafikkskole representert ved Ann Kristin Thomte. Oppdragsgiveren er offisielt Høgskolen i Gjøvik representert ved Frode Haug, som også fungerer som veileder. Gruppedeltagerene stiller selv med teknisk utstyr til prosjektet. Tidsrammene på prosjektet er lagt i henhold til de frister som er satt fra skolens side. Absolutt sluttdato for prosjektet er 24/5-06.

Teknologiske rammer

Gjennom kontakt med Thomtes trafikkskole har vi vurdert statusen på det utstyret som finnes hos trafikkskolene til å være av typen enkeltstående maskiner som ofte vil ha en tilgang til Internet, uten at det er en forutsetning. Basert på deltagernes forkunnskaper og vurderinger har det allerede blitt avgjort at prosjektet i all hovedsak skal utvikles i Java 1.5. 3D-modulen skal basere seg på en OpenGL-implementasjon for Java kalt JOGL. Plattformen det skal utvikles for er i utgangspunktet Windows, men hvis mulig så skal plattformuavhengighet tilstrebes. Dette gjøres mulig av Javas portabilitet til alle systemer som har en JavaVM tilgjengelig. Produktet må kunne kjøre sømløst på en vanlig hjemme-PC uten ekstraordinære krav til maskinkraft eller grafikkprosesseringsenheten (skjermkort).

F.2 Omfang

F.2.1 Problemområde

På de fleste trafikkskoler i dag benytter man seg av tegninger på tavler, magnetbaserte biler på brett eller ferdig foiler med tegninger på når man ønsker å presentere en trafikksituasjon. Disse metodene er i dag litt gammeldagse, men siden alternativer innen høyteknologien ikke finnes er det dette man er nødt til å benytte seg av. De fleste som deltar i et undervisningsopplegg ved en trafikkskole tar i løpet av kurset en form for tentamenstest med spørsmål som ligner de man får på den offisielle teoriprøven. Disse prøvene utarbeides enten av trafikkskolen selv eller av egne selskaper som har spesialisert seg på å lage slike prøver. Idag kan trafikkskolene på ingen standardisert måte dele disse spørsmålene de har

lagd med hverandre.

F.2.2 Oppgavebeskrivelse

Koser en applikasjon for å:

1. Generere trafikksituasjoner for lokal presentasjon.
2. Generere teorispørsmål/prøver for visning på lokal PC og/eller over Internett.

Applikasjonen skal på den måten kunne erstatte dagens ordning som går ut på å tegne for hånd på tavle eller papir og vil forhåpentligvis være med på å både effektivisere undervisningen og å gjøre tegningene/bildene mer oversiktlige.

Generere trafikksituasjoner for lokal presentasjon.

Det skal benyttes 3D-grafikk for generering og presentasjon av situasjonene slik at brukeren selv kan velge plasseringen av og innsynsvinkel til trafikale objekter (kjøretøy, bygninger, o.l.) samt innsynsvinkel til situasjonen i sin helhet. Det må også kunne gå an å legge inn et digitalt stillbilde istedenfor 3D der det er nødvendig. Brukergrensesnittet til generatoren må være så enkelt som mulig slik at sluttbrukere ikke trenger lang opplæring for å bruke applikasjonen og derfor skal det brukes et dra og slipp (ńdrag and dropż) grensesnitt hvor man kan dra objekter fra en liste og inn på situasjonen. Alle trafikale objekter er organisert i en liste ut i fra en logisk kategori-struktur faglig sett. Det skal være mulig å opprette/redigerer flere scenarier basert på forskjellige trafikklokasjoner samtidig, og for hver scenario ha undersituasjoner som bygger på en og samme lokasjon, men med varierte trafikale-objekter. Når man er fornøyd med situasjonen skal man kunne fryse den slik at ytterligere endringer ikke påvirker den fryste situasjonen, og situasjonen havner i listen over situasjoner for gjeldende scenario. Man skal lett komme inn i fremvisnings-modus hvor man enkelt kan bla gjennom alle scenarier og undersituasjoner som er åpnet, men med kun det mest nødvendige av brukergrensesnitt synlig. Mens man driver med fremvisning skal det være mulig å kunne endre på situasjonen hvis man der og da ønsker å vise et annet eksempel. Til hver situasjon skal det være mulig å legge ved et notat eller spørsmål.

Generere teorispørsmål/prøver for visning på lokal PC og/eller over Internett.

Det skal også være mulig å skifte enkelt til spørsmålsmodus for aktiv/gjeldende undersituasjon. Da vil brukergrensesnitt for å legget til spørsmålstekst, kategori og svaralternativer bli synlig sammen med den genererte undersituasjonen. Når man lagrer vil spørsmålet, sammen med undersituasjonen, lagres i den lokale spørsmålsdatabasen. Man skal kunne publisere spørsmål ved å laste de opp til en database på internett. Dette skal kunne gjøres lettvint fra applikasjonens brukergrensesnitt. Fra denne databasen vil elever kunne få tilgang til spørsmål og prøver gjennom et webbasert grensesnitt. Ved opplasting/publisering av spørsmål skal 3D-grafikken gjøres om til et 2D-stillbilde som enkelt, portabelt og plassbesparende kan lagres sammen oppgavetekst og svaralternativer i en database. Det er også nødvendig å ha en applikasjon for å sette sammen spørsmål til prøver. Disse spørsmålene hentes enten fra den lokale katalogen med spørsmål man har generert selv eller ved å hente fra databasen på Internet. For å kunne bruke disse prøvene lokalt på trafikkskolens PC, uavhengig om den er tilknyttet Internet eller ikke, er det også nødvendig med et grensesnitt som elevene kan bruke for å hente prøver fra den lokale katalogen. Uavhengig om eleven bruker det webbaserte grensesnittet hjemme eller det lokale grensesnittet på trafikkskolen skal han/hun bli presentert med et interaktivt grensesnitt som gir tilbakemelding ved å fortelle eleven hvor han/hun har svart feil eller riktig og som oppsummerer hele prøven.

F.2.3 Avgrensninger

Applikasjonen utvikles i utgangspunktet for å kjøre på en enkelt PC som kan være tilkoblet internett for å laste opp/ned spørsmål. Støtte for distribuering av spørsmål/prøver over lokalt nettverk o.l. er blitt vurdert, men har blitt bortvalgt ettersom det ikke er vanlig at trafikkskoler har flere maskiner. Dette er selvfølgelig noe som kan implementeres i produktet etter behov. Selv om situasjonene genereres i 3D er det ikke en trafikk-simulator. Verken animasjoner av objekter eller forhold som f.eks. vær, tid på døgnet (dvs. lysforhold o.l.), årstider, eller tyngdekraft vil bli implementert. Det vil i utgangspunktet heller ikke lages støtte for spesielle elementer i terrenget som f.eks. fjell og tunneler, men vi er åpne for å implementere dette ved en senere anledning eller hvis det blir tid til overs. De trafikale objektene som er mulig å bruke vil også begrense seg til generelle objekter og ikke spesielle merker. Eksempel på dette er at vi har enkle biler og ikke BMW, Audi, Volvo, o.l. Det vil heller ikke bli fokusert på å legge inn masse reelle spørsmål og prøver, men heller legge inn funksjonalitet og objekter som kan brukes til å generere dette med det ferdige produktet. Andre mulige implementeringer senere, men som ikke er en del av dette prosjektet er støtte for audio/høytlesning av spørsmål og svaralternativer samt animasjoner/videofiler i

stedenfor bilder. Det er alikevel mulighet for at slike multimedia-utvidelser etter prosjektets slutt hvis behovet er til stede.

F.3 Prosjektorganisering

F.3.1 Ansvarsforhold

Prosjektleder sin oppgave er å ha en overordnet oversikt over prosessen og gripe inn hvis et medlem oppfører seg uansvarlig i henhold til de satte retningslinjer og regler for gruppen. Prosjektdeltagerne er selv ansvarlig for å følge disse retningslinjene. Disse reglene er nærmere presentert i pkt. 6.2. Avgjørelser rundt prosjektets prosess blir tatt i fellesskap hvor samtlige av gruppens medlemmer er representert.

Prosjektleder og utvikler:	Jonas Strømstad Epost: jonas@overhagen.no Telefon: 97688478
Utvikler:	Kenneth Rinnan Epost: kenneth.rinnan@gmail.com Telefon: 92808031
Utvikler:	Sverre Bakke Epost: sverre.bakke@hig.no Telefon 99578886

F.3.2 Øvrige roller og bemanning

Oppdragsgiver og veileder:	Frode Haug Epost: frode@haugianerne.no Telefon: 61135191

Vi har innledet et samarbeid med Thomtes Trafikkskole for å få faglig innspill underveis på prosjektet. Thomtes skal under prosessen delta på egne statusmøter som representant for målgruppen til produktet, men er fritatt for alt ansvar i forbindelse med prosjektet og det endelige produktet. Se punkt 8, Kontrakter og avtaler, for mer informasjon om denne avtalen.

Thomtes Trafikkskole: Ann Kristin Thomte
Epost: firmapost@thomte.no
Nettside: http://www.thomte.no
Telefon: 61176079 / 91833454

F.4 Planlegging, oppfølging og rapportering

F.4.1 Hovedinndeling av prosjektet

Prosjektets karakteristika med en grunnstruktur som er klart modulbasert gir føringer på hvilke utviklingsmodeller som egner seg. En metode der man kan jobbe med de enkelte modulene hver for seg og muligheter til å drive med småflikking på kravene underveis. Det kan bli nødvendig å endre på kravene til de enkelte modulene basert på resultatene som er blitt oppnådd i tidligere moduler. Derfor er det avgjort å følge en inkrementell utviklingsmetoddikk så vi har bedre kontroll på fremgangen i prosjektet og kan justere prioritetene våre underveis. Hadde vi fulgt en lineær metode hadde det sannsynligvis blitt vanskelig å kunne legge mer innsats i enkelte moduler fremfor andre. Utviklingsarbeidet blir delt opp etter malen lagt opp i punkt 7 og utdypet i vedlegget punkt 9.1.

F.4.2 Krav til statusmøter og beslutningspunkt

Statusmøter gjøres via innlevering av en statusrapport til veileder. Disse rapportene skal inneholde:

- Detaljert beskrivelse av fremgangen i prosjektet jamnført med fremdriftsplanen.
- Revidert risikoanalyse.
- Vurdering av motivasjon innad i gruppen.
- Vurdering av samarbeidet med oppdragsgiver og veileder.

Rapportene skal følge malen gitt av skolen[1].

[1] http://www.hig.no/dav/993F988C34624AA1BCEDA7DBE284A4D6.pdf

F.4.3 Prosjektmøter

Hver mandag har gruppen møte internt for å sjekke status og legge opp planene for den uken som de står ovenfor. Det som er viktig før prosjektmøtene hver mandag er at hver deltager stiller forberedt til møtene. Dvs. at han har tenkt gjennom hvordan man ligger an i forhold til tid og kunnskap til oppgaven. Hvis en av deltagerne ser at det blir vanskelig og gjennomføre den gitte oppgaven, skal det gis beskjed til gruppen evt. innkalle til et nytt møte og si fra om dette så raskt det blir oppdaget, slik at det blir mulig å løse situasjonen så raskt og enkelt som mulig. Når det gjelder beslutninger som skal tas så skal disse bli tatt opp på statusmøtene på mandager, men om det er krise skal det være mulig å få tatt det opp med en gang. For at noe som skal bli besluttet så skal alle i gruppen være enig, men om det er et godt forslag og en nekter, skal prosjektleder ha det siste ordet.

F.4.4 Beslutningspunkter

Beslutningspunktene i prosjektet følger tidsplanen fra gantt-skjemaet. Det vil si at vi må ha tatt alle avgjørelser knyttet til et inkrement før vi begynner utviklingen av dette inkrementet. Siste frist for å ta beslutninger er siste dag av kravspesifisering på inneværende inkrement. Siste frist for beslutninger knyttet til det overordnede systemet er første dag med jobbing på første inkrement.

F.5 Risikoanalyse

F.5.1 Kritiske suksessfaktorer

- Brukervennlige og intuitive menyer og redigeringsverktøy.

  For at programmet skal kunne brukes så interaktivt som målet er, må brukergrensesnittet være så godt at medlemmer av målgruppen enkelt kan benytte det uten særlig opplæring og uten at de skal måtte gå inn og ut av mange forskjellige menyer. Blir menysystemet for tungvint så vil bruk av programmet hindre fremgangen i opplæringen istedenfor å hjelpe den til å bli mer effektiv.

- Stabil og robust kode.

  Blir programmet stadig stanset fordi brukeren ved uhell har trykket litt feil på en meny e.l. så vil det hindre bruken av programmet. Koden må

være robust så man ungår at programmet krasjer ved ubetydelig feil i input.

- Lave ressurskrav.

  For at programmet skal kunne kjøre på vanlige arbeidsstasjoner så må renderingen av de tredimensjonale bildene kreve så lite ressurser som mulig, uten at grafikken skal lide alt for mye av dette.

- Enkelt grensesnitt til opp/nedlasting tjenesten.

  For at nedlastingstjenesten skal kunne være brukbar for brukerne så må den være mulig å bruke uten noe mer avansert oppsett. Man må med andre ord slippe unna med å f.eks. legge til en enkel URL eller lignende i oppsettet av tjenesten.

F.5.2 Risikoevaluering

Det som er den største suksess faktoren er den at vi får nok tid til å sette oss inn og lære OpenGL programmering, slik at vi får ferdig selve trafikksituasjongeneratoren. En annen kritisk suksessfaktor vil være at vi disponerer nok tid til hvert inkrement, men samtidig er fleksible nok slik at vi har mulighet til å legge på mer tid om vi trenger det på inkrementer som må bli ferdige før vi går videre. At vi har nok møter slik at vi ikke jobber på det samme, men planlegger godt nok slik at det ikke blir redundans på arbeidet, dvs at to stk jobber med nøyaktig det samme. Hvis noen blir syk, at vi har en plan for hvordan man skal kunne løse det hvis den som blir syk sitter med kunnskap de andre trenger for å komme videre.

Fare	Sannsynlighet	Skade	Backup
Feil tidsplanlegging	M	M	Nedprioritere senere inkrementer for å komme skikkelig i mål med de viktigste delene.
For dårlig oversikt over funksjonalitet i Java/JOGL, dvs at man ikke har sjekket ut at noe er mulig, men antar at det går, og finner senere ut at det ikke er mulig.	L	H	At man evt. må kutte ut noen inkrementer på prosjektet for å komme i mål
Sykdom	L	L	Ha mulighet til å jobbe hjemme, og jobber inn litt senere. Kompetansenivået er likt for gruppens medlemmer, så om noen blir syke så vil man fortsatt kunne jobbe videre.
Prosjektdeltagerne er ikke istand til å tilegne seg den kunnskapen som trenges for å gjennomføre prosjektet til rett tid.	M	M	Kutte ned på noen av inkrementene slik at man får mer tid til å sette seg inn i det som er de kritiske delene av prosjektet.

F.6 Kvalitetssikring

F.6.1 Organisering av kvalitetssikring

- Hver enkelt gruppemedlem har ansvar for å følge kodestandarder (pkt. 6.3) som er satt
- Alle i gruppa må forholde seg til den designen som er satt for hvert inkrement.
- Brukervennlighet skal sikres gjennom jevnlig møter med en representant fra målgruppen.
- Omveltende designendringer skal ikke gjennomføres uten at det har vært tatt opp i møte med veileder.
- Programmet Subversion skal benyttes til revisjonskontroll.

F.6.2 Grupperegler

1. Kommentering gjøres på norsk, siden prosjektet er basert på den norske kjøreopplæringen og tar ikke høyde for utenlandsk reglement og opplæringsform.
2. Kommentering av metoder skal gjøres før man skriver koden til metoden.
3. Arbeidstiden er fra 0900 - 1500 hver ukedag såfremt man ikke har gyldig grunn til fravær, eller er opptatt med forelesning eller lønnet arbeid.
4. Det forventes at det arbeides med prosjektet et minimum av 30 timer i uka, altså tilsvarende 20 studiepoeng. Det forventes også at gruppemedarbeiderene jobber utover disse timene hvis nødvendig eller avgjort av flertallet i gruppa.
5. Oppmøte på møter med veileder/oppdragsgiver og samarbeidspartner er obligatorisk hvis møteinnkalling er sendt ut.
6. Gruppemedarbeidere forplikter seg til å forberede seg til statusmøter ved å skaffe seg oversikt over sin egen fremgang i forhold til oppsatte planer.
7. Kommentarer til opplastinger til SVN-repository må inneholde alle de endringene som faktisk er gjort på de gitt filene med navn på den som har gjort oppdateringene.
8. Fastsatt kodestandard skal til enhver tid følges!
9. Prosjektets framgang har høyere prioritet enn den enkeltes anseelse innad i gruppa, så man må si ifra om det er noe man trenger hjelp til å få gjennomført.
10. Loggføring er obligatorisk!

F.6.3 Kodestandard

- Krøllparanteser skal stå på etterfølgende linje med samme margavstand som blokkidentifikatoren (metodenavn, kontrollstrukturer osv.).
- Innrykk er størrelsen til 2 mellomrom.
- En linje er 80 tegn lang.
- Hvis en funksjonsdeklarasjon eller et funksjonskall er så langt at parameterne overskrider lengden på linjen skal første parameter som overskrider lengden på linjen skrives under første parameter etter identifikatoren.
- Identifikatorer skal gis engelske navn.

- Exceptions skal håndteres via en logger klasse som rapporterer typen feil.
- Klassenavn starter med stor bokstav og har en ny stor bokstav for å skille ordene i navnet.
- Variabelnavn starter med liten bokstav og har en stor bokstav for å skille ordene i navnet.
- Metodenavn følger samme standard som variabelnavn.
- Konstanter skrives i bare store bokstaver med underscore som ordskiller.

F.6.4 Kvalitetssikring av kritiske suksessfaktorer

For å sikre brukervennlighet skal representant fra målgruppen brukes til hjelp under utvikling og under testing. Samtidig som vi bruker en rekke ulike pattern om god design til hjelp. For at koden skal bli stabil og robust skal vi bruke en egen logger klasse til å ta seg av feilmeldinger og rapportere disse videre i et egnet format. I tillegg skal vi bruke utstrakt testing ńas-we-goż for å hindre at en feil blir liggende igjen lenge i koden. For å holde maskinkravene nede skal vi bruke så få tunge grafiske elementer som mulig under renderingen av tredimensjonale bilder. Ved å samtidig holde kompleksiteten til de modellene som benyttes nede så vil ikke programmet kreve mer enn det en gjennomsnittlig arbeidsstasjon kan gi.

F.6.5 Testing

Det skal utføres to runder med brukertesting i løpet av prosjektet. Den første skal gjøres etter fullføring av inkrement 3 og mot slutten av prosjektet ved fullføring av inkrement 6. Testingen skal gjennomføres med et utvalg på fem testpersoner med ulik teknisk bakgrunn med spredning i alder. Testerne vil få tildelt hver sin testversjon av programmet sammen med et evalueringsskjema. De blir deretter gitt en uke på å teste programvaren når de har tid og etterhvert fylle ut evalueringsskjemaet. En uke etter starten skal vi ha en debriefing med de som deltok med en liten oppsummering.

F.7 Gjennomføring

F.7.1 Hovedaktiviteter

Prosjektet deles opp i fire hovedaktiviteter utifra den inkrementelle utviklingsmetodikken. Disse utføres i nevnt rekkefølge per inkrement i programvaren:

1. Planlegge (kravspesifisering, analyse og design)
2. Implementere
3. Teste
4. Oppsummere

Det skal først gjøres en overordnet planlegging av hele prosjektet på et lavere detaljeringsnivå før første inkrement settes i gang. Som støtteaktivitet skal det under hele prosjektet dokumenteres hva som gjøres og hvordan gjennom rapporter nevnt under ńKvalitetssikringż og loggføring av all aktivitet.

F.7.2 Milepæler

29/1-06 - Innlevering av forprosjektrapport

31/1-06 - Statusmøte med Thomtes Trafikkskole

15/2-06 - Statusmøte med Thomtes Trafikkskole

17/2-06 - Inkrement 1 ferdigstilles

20/2-06 - Statusrapport

1/3-06 - Statusmøte med Thomtes Trafikkskole

3/3-06 - Inkrement 2 ferdigstilles

15/3-06 - Statusmøte med Thomtes Trafikkskole

17/3-06 - Inkrement 3 ferdigstilles

31/3-06 - Inkrement 4 ferdigstilles

3/4-06 - Statusrapport

14/4-06 - Inkrement 5 ferdigstilles

28/4-06 - Inkrement 6 ferdigstilles

1/5-06 - Statusrapport

12/5-06 - Inkrement 7 ferdigstilles

24/5-06 - Innlevering av fullstendig rapport

29/5-06 - Innlevering av laminert plakat

7-8/6-06 - Presentasjon av prosjektet

F.7.3 Tids- og ressursplaner

Det er ikke lagt noen spesifikke tidsplaner fordelt per prosjektmedarbeider, men det forventes at alle jobber tilstrekkelig til at prosjektet blir fullført med utgangspunkt i et minimum av 30 uketimer. Tidsplanene for gruppa som helhet er lagt opp i vedlagte Gantt-skjema.

F.7.4 Kostnader

Alle utstyrsmessige kostnader dekkes av deltagerne i prosjektet gjennom bruk av eget privat datautstyr.

Innkjøp av lisens til 3D modelleringsprogrammet Milkshape 3D[2] (à 25 USD) dekkes av prosjektgruppas medlemmer.

[2] http://www.swissquake.ch/chumbalum-soft/index.html chUmbaLum Software

Tillegg G

Innholdet på CD-romen

- Prosjektrapporten på filen «rapport.pdf».
- Installeringsrutine for applikasjonen på filen «install-kos.exe».
- Installeringsrutien for Java 1.6 Mustang på filen «jdk-6-beta-windows-i586.exe».
- Mini-poster på filen poster.pdf
- Dokumentering av kildekode / JavaDoc i katalogen JavaDoc
- Kildekode i katalogen source

Tillegg H

Statusrapporter

H.1 Statusrapport 20/2-2006

H.1.1 Status for:

Planlegging (fremdriftsplan)

Overordnet plan for hele prosjektperioden er ferdig. Planlegging av de enkelte inkrementene underveis gjøres ved oppstart per inkrement. Planlegging for inneværende inkrement 2 har status påbegynt

Organisering av gruppens arbeid og ansvarsområder

Hovedansvarsområder er fordelt mellom prosjektdeltagerne. Ansvarsområder innen kodinen fordeles etter kompetansefelt og nivå.

Klargjøring av problemstilling (eks. Systemering)

Systemeringen på inkrement 1 henger fortsatt litt igjen og er enda bare gjort delvis ferdig. Systemeringen for inkrement 2 har så vidt kommet i gang, og skal være klar i første utkast snart.

Løsningsmetode (eks. Koding).

Fremdriften i løsningen har vært bedre enn ventet. Den noe tynne forhåndskompetansen vi hadde på området før vi begynte gjorde oss noe nervøse, men bekymringen har blitt gjort til skamme gjennom hardt arbeid. Inkrement 1 er som ventet i en versjon nå som inneholder det meste av funksjonaliteten som var planlagt. Det som enda mangler er tegning av linjer på veg, samt en ny og mer intuitiv måte å føre kameraet på. Inkrement 2 er kun i den lille oppstart enda. Ingen ferdig kode foreligger enda.

Rapportskriving.

Rapportskrivningen ligger noe etter ønsket fremdrift (jfr. pkt 1.4). Dokumentasjon av alle møter holdt mellom veileder og gruppen er på plass. Måloppnåelsesrapport for inkrement 1 vil ikke foreligge enda, fordi dette inkrementet fortsatt ikke er i helt ferdig versjon og vil som ventet strekke seg ut som en sideaktivitet videre ut i prosjektet. Dette er fordi andre inkrementer i prosjektet avhenger av dette inkrementet. Inkremente 1 kan derfor ikke være helt låst for endringer.

H.1.2 Totalstatus for punktene over (oppsummering)

Vi ligger godt an hva måloppnåelse angår. Koden fungerer, og gjør det den skal samtidig som vi har nådd så langt som vi hadde ønsket og håpet på dette punktet i prosjektet. Vi ligger litt dårligere an når det gjelder dokumentasjon, men dette er ikke verre enn at det vil ńgå seg tilż uten noen store endringer i planene.

H.1.3 Muligheter? Trusler/Problemer?

Sykdom har i perioder påvirket fremdriften i prosjektet noe, men har ikke hatt noen fatale følger enda. Nå er heldigvis den delen av prosjektet som krever kunnskaper som ikke besittes kolektivt i gruppe over. Dvs. at vi i de neste inkrementene har større muligheter til å falle tilbake på kunnskapene fra gjenværende prosjektdeltagere hvis en enkelt blir borte. Prosjektet har blitt plukket ut som et av de prosjektene skolen vil bruke i sin markedsføring. Representanter fra skolen har gjennomført et intervju med gruppa, som vil bli distribuert til lokal aviser i den enkelte gruppemedlems omegn, samt forsøkt trykket i ATLs bransjeblad.

H.1.4 Hva er avsluttet? Hvilke oppgaver er ferdige?

Vi anser oss ikke som ferdige med noen elementer av prosjektet enda. Inkrement 1 overskrider som ventet rammene satt i tidsplanen, men vi er godt fornøyd med mengden funksjonalitet som er lagt inn i denne delen av løsningen allerede ved dags dato.

H.1.5 Hva er under arbeid?

Følgende oppgaver er de som er under arbeid (i parantes: omtrentlig fullførings-grad så langt): Dokumentasjon (5%) Inkrement 1 (80%) Inkrement 2 (10%)

H.1.6 Tidsfrister

Levering av prøveversjon til Thomtes Trafikkskole ble dessverre ikke gjort innen avtalt tidsperiode pga. sykdom. Leveringen ble i samråd med Thomtes dyttet tilbake. Ellers er tidsfristene overholdt.

H.1.7 Hva med motivasjon:

Motivasjonen for fremdrift i prosjektet er fortsatt god. Varierende helse hos noen av prosjektdeltagerne har ført til at motivasjonen i perioder har svingt noe. Tilbakemeldinger fra samarbeidpartneren (Thomtes Trafikkskole) har vært veldig oppløftende. Gleden av å kunne presentere noe som noen ńfallerż såpass for gjør at vi får motivasjon for å gjøre det lille ekstra.

H.1.8 Hvordan oppleves veilederkontakt:

Kontakten med veileder er upåklagelig. Ukentlige møter med veileder gir gode muligheter til å korrigere kursen på prosjektet. I tillegg stiller veileder seg ofte tilgjengelig utenom den avtalte tiden. Kos som prosjekt stilles egentlig i en liten særstilling på dette punktet siden veilederen helt fra starten av har ønsket å delta mest mulig og har vært pådriver bak prosjektet. For ikke å glemme at det er han som kom med idéen i utgangspunktet.

H.2 Statusrapport 3/4-2006

H.2.1 Status for:

Planlegging (fremdriftsplan)

Overordnet plan for hele prosjektperioden er ferdig. Planlegging av de 3 første inkrementene (3D-generator, GUI, presentasjon) er ferdig, men er ikke formalisert ferdig i form av kravspesifikasjon og designdokument. Planlegging av inkrement 4 (Spørsmål) er påbegynt.

Organisering av gruppens arbeid og ansvarsområder

Hovedansvarsområder er fordelt mellom prosjektdeltagerne. Ansvarsområder innen kodinen fordeles etter kompetansefelt og nivå. Kenneth jobber fulltid med 3D-motoren, mens Jonas og Sverre jobber med det resterende. Alle gruppens medlemmer har bidratt med 3D-modellering.

Klargjøring av problemstilling (eks. Systemering)

Overordnet systemering er ferdig. Systemeringen på inkrement 1 er ferdig. Systemeringen for inkrement 2 er ikke ferdig, men er godt påbegynt. Systemering for inkrement 3 og 4 er påbegynt, men uferdig.

Løsningsmetode (eks. Koding).

Gruppen i sin helhet har nå tilegnet seg forventet kompetanse for å kunne utføre prosjektoppgaven. Kildekoden til de 3 første inkrementene er så godt som ferdigstilt og løst teknisk med unntak av noen få bugfikser. Det vi trenger å forbedre fra disse inkrementene er:

- Trafikklys
- Filinnlesing/skriving
- Innstillinger

Utseende og funksjonalitet for disse inkrementene er ellers ferdigbestemt og ferdigstilt. Inkrement 4 (Spørsmålsgenerator) er påbegynt som prototype som vil snart bli presentert for oppdragsgiver/veileder for tilbakemelding.

Rapportskriving.

Rapportskrivningen ligger noe etter ønsket fremdrift (jfr. pkt 1.4). Dokumentasjon av alle møter holdt mellom veileder og gruppen er på plass. Logging av oppmøte er også dokumentert tilstrekkelig til å få en helhetlig oversikt over aktivitetsnivået. Rapportskrivingen vil bli opptrappet betydelig etter påske for å kunne levere førsteutkast til veileder i rimlig tid før prosjektets sluttdato.

H.2.2 Totalstatus for punktene over (oppsummering)

Vi ligger godt an hva måloppnåelse angår. Koden fungerer, og gjør det den skal samtidig som vi har nådd så langt som vi hadde ønsket og håpet på dette punktet i prosjektet. Vi ligger litt dårligere an når det gjelder dokumentasjon, men dette er ikke verre enn at det vil ńgå seg tilż uten noen store endringer i planene. Flere av inkrementene har blitt utvidet tidsmessig grunnet endringer i krav til applikasjonen, og vi har valgt å jobbe med 3D-motoren gjennom hele prosjektet fremfor utelukkende de første ukene da dette er kjernen til applikasjonen.

H.2.3 Muligheter? Trusler/Problemer?

Sykdom har i perioder påvirket fremdriften i prosjektet noe, men har ikke hatt noen fatale følger enda. Nå er heldigvis den delen av prosjektet som krever kunnskaper som ikke besittes kolektivt i gruppe over. Dvs. at vi i de neste inkrementene har større muligheter til å falle tilbake på kunnskapene fra gjenværende prosjektdeltagere hvis en enkelt blir borte. Forutenom dette er tidsmangel definitivt den ene store trusselen for dette prosjektet. Prosjektet har blitt plukket ut som et av de prosjektene skolen vil bruke i sin markedsføring. Representanter fra skolen har gjennomført et intervju med gruppa, som vil bli distribuert til lokal aviser i den enkelte gruppemedlems omegn, samt forsøkt trykket i ATLs bransjeblad.

H.2.4 Hva er avsluttet? Hvilke oppgaver er ferdige?

Vi anser oss ikke som ferdige med noen elementer av prosjektet enda. Inkrement 1 overskrider som ventet rammene satt i tidsplanen, men vi er godt fornøyd med mengden funksjonalitet som er lagt inn i denne delen av løsningen. Inkrement 2 har også blitt utvidet tidsmessig grunnet endringer i krav for applikasjonen. Vi anser at arbeidet av de 3 første inkrementene er i sluttfasen. Flere av de trafikale objektene til applikasjonen er ferdigstilt, men biblioteket i sin helhet er uferdig.

H.2.5 Hva er under arbeid?

Følgende oppgaver er under arbeid: Dokumentasjon: Mye gjenstår Inkrement 1: Nesten fullført Inkrement 2: Nesten fullført Inkrement 3: Nesten fullført Inkrement 4: Påbegynt

H.2.6 Tidsfrister

Med unntak av tidligere nevnte utsettelser på levering av prøveversjon til Thomtes Trafikkskole er alle tidsfrister overholdt.

H.2.7 Hva med motivasjon:

Motivasjonen for fremdrift i prosjektet er fortsatt god. Varierende helse hos noen av prosjektdeltagerne har ført til at motivasjonen i perioder har svingt noe. Tilbakemeldinger fra samarbeidpartneren (Thomtes Trafikkskole) har vært veldig oppløftende. Gleden av å kunne presentere noe som noen ńfallerż såpass for gjør at vi får motivasjon for å gjøre det lille ekstra. Motivasjonen har i korte perioder vært noe under ønsket nivå grunnet software og maskinvare-problemer, men dette er nå på et stabilt nivå. Motivasjonen for 3D-modellering har vært jevnt synkende gjennom prosjektet grunnet tidkrevende ńskinningż. Det har også i en liten grad vært demotiverende med endringer i applikasjonens krav med jevne mellomrom som igjen har ført til forlengelse av inkrementenene.

H.2.8 Hvordan oppleves veilederkontakt:

Kontakten med veileder er upåklagelig. Ukentlige møter med veileder gir gode muligheter til å korigere kursen på prosjektet. I tillegg stiller veileder seg ofte tilgjengelig utenom den avtalte tiden. Kos som prosjekt stilles egentlig i en liten særstilling på dette punktet siden veilederen helt fra starten av har ønsket å delta mest mulig og har vært pådriver bak prosjektet. For ikke å glemme at det er han som kom med idéen i utgangspunktet.

H.3 Statusrapport 1/5-2006

H.3.1 Status for:

Planlegging (fremdriftsplan)

Overordnet plan for hele prosjektperioden er ferdig. Planlegging av de 4 første inkrementene (3D-generator, GUI, presentasjon, spørsmål) er ferdig, men er ikke formalisert ferdig i form av kravspesifikasjon og designdokument.

Organisering av gruppens arbeid og ansvarsområder

Hovedansvarsområder er fordelt mellom prosjektdeltagerne. Ansvarsområder innen kodinen fordeles etter kompetansefelt og nivå. Kenneth jobber fulltid med 3D-motoren, mens Jonas og Sverre jobber med det resterende. Alle gruppens medlemmer har bidratt med 3D-modellering.

Klargjøring av problemstilling (eks. Systemering)

Overordnet systemering er ferdig. Systemeringen på inkrement 1 er ferdig. Systemeringen for inkrement 2 er ikke ferdig, men er godt påbegynt. Systemering for inkrement 3, 4 og 5 er påbegynt, men uferdig.

Løsningsmetode (eks. Koding).

Gruppen i sin helhet har nå tilegnet seg forventet kompetanse for å kunne utføre prosjektoppgaven. Kildekoden til de 4 første inkrementene er så godt som ferdigstilt og løst teknisk med unntak av noen få bugfikser. Det vi trenger å forbedre fra disse inkrementene er:

- Filinnlesing/skriving

Utseende og funksjonalitet for disse inkrementene er ellers ferdigbestemt og ferdigstilt.

Rapportskriving.

Rapportskrivningen ligger noe etter ønsket fremdrift (jfr. pkt 1.4). Dokumentasjon av alle møter holdt mellom veileder og gruppen er på plass. Logging av oppmøte er også dokumentert tilstrekkelig til å få en helhetlig oversikt over aktivitetsnivået. Rapportskrivingen har blitt opptrappet betydelig etter påske, men det gjenstår fremdeles en betydelig del. Deler av gruppen jobber nå fulltid med kravspesifikasjon.

H.3.2 Totalstatus for punktene over (oppsummering)

Vi ser at vi har svært knapt med tid for prosjektet mot slutten. Det er i all hovedsak dokumentasjon, bugfikser i de første inkrementene, og utvikling av en prøve-modul til systemet som gjenstår. Gruppen jobber nå på overtid daglig for å komme i mål innen gitte prosjekt-tidsfrister.

H.3.3 Muligheter? Trusler/Problemer?

Sykdom har i perioder påvirket fremdriften i prosjektet noe, men har ikke hatt noen fatale følger enda. Nå er heldigvis den delen av prosjektet som krever kunnskaper som ikke besittes kolektivt i gruppe over og gjenværende inkrement kan utvikles i løpet av kort tid basert på den kunnskapen gruppen allerede besitter. Forutenom dette er tidsmangel definitivt den ene store trusselen for dette prosjektet. Prosjektet har blitt plukket ut som et av de prosjektene skolen vil bruke i sin markedsføring. Representanter fra skolen har gjennomført et intervju

med gruppa, som vil bli distribuert til lokal aviser i den enkelte gruppemedlems omegn, samt forsøkt trykket i ATLs bransjeblad.

H.3.4 Hva er avsluttet? Hvilke oppgaver er ferdige?

Vi anser nå flere av elementene i prosjektet som ńså godt som ferdigż. Inkrement 1 overskrider som ventet rammene satt i tidsplanen, men vi er godt fornøyd med mengden funksjonalitet som er lagt inn i denne delen av løsningen. Inkrement 2 har også blitt utvidet tidsmessig grunnet endringer i krav for applikasjonen. Vi anser at arbeidet av de 4 første inkrementene er i sluttfasen. Flere av de trafikale objektene til applikasjonen er ferdigstilt, men biblioteket i sin helhet er uferdig.

H.3.5 Hva er under arbeid?

Følgende oppgaver er under arbeid: Dokumentasjon: Mye gjenstår Inkrement 1 Ű 3D-generator: Nesten fullført Inkrement 2 Ű brukergrensesnitt for generator: Nesten fullført Inkrement 3 - presentasjonsmodus: Nesten fullført Inkrement 4 - spørsmålsgenerator: Nesten fullført Inkrement 5 Ű lokal prøvemodul: Påbegynt

H.3.6 Tidsfrister

Med unntak av tidligere nevnte utsettelser på levering av prøveversjon til Thomtes Trafikkskole er alle tidsfrister overholdt.

H.3.7 Hva med motivasjon:

Motivasjonen for fremdrift i prosjektet er fortsatt god. Varierende helse hos noen av prosjektdeltagerne har ført til at motivasjonen i perioder har svingt noe. Tilbakemeldinger fra samarbeidpartneren (Thomtes Trafikkskole) har vært veldig oppløftende. Gleden av å kunne presentere noe som noen ńfallerż såpass for gjør at vi får motivasjon for å gjøre det lille ekstra. Motivasjonen har i korte perioder vært noe under ønsket nivå grunnet software og maskinvare-problemer, men dette er nå på et stabilt nivå, men vi ser at vi gleder oss til å bli ferdig med det. Motivasjonen for 3D-modellering har vært jevnt synkende gjennom prosjektet grunnet tidkrevende ńskinningż. Det har også i en liten grad vært demotiverende med endringer i applikasjonens krav med jevne mellomrom som

igjen har ført til forlengelse av inkrementenene. Kort oppsummert gleder hele gruppen seg til prosjektets slutt da vi på gjeldende tidspunkt alle jobber overtid daglig for å komme i mål.

H.3.8 Hvordan oppleves veilederkontakt:

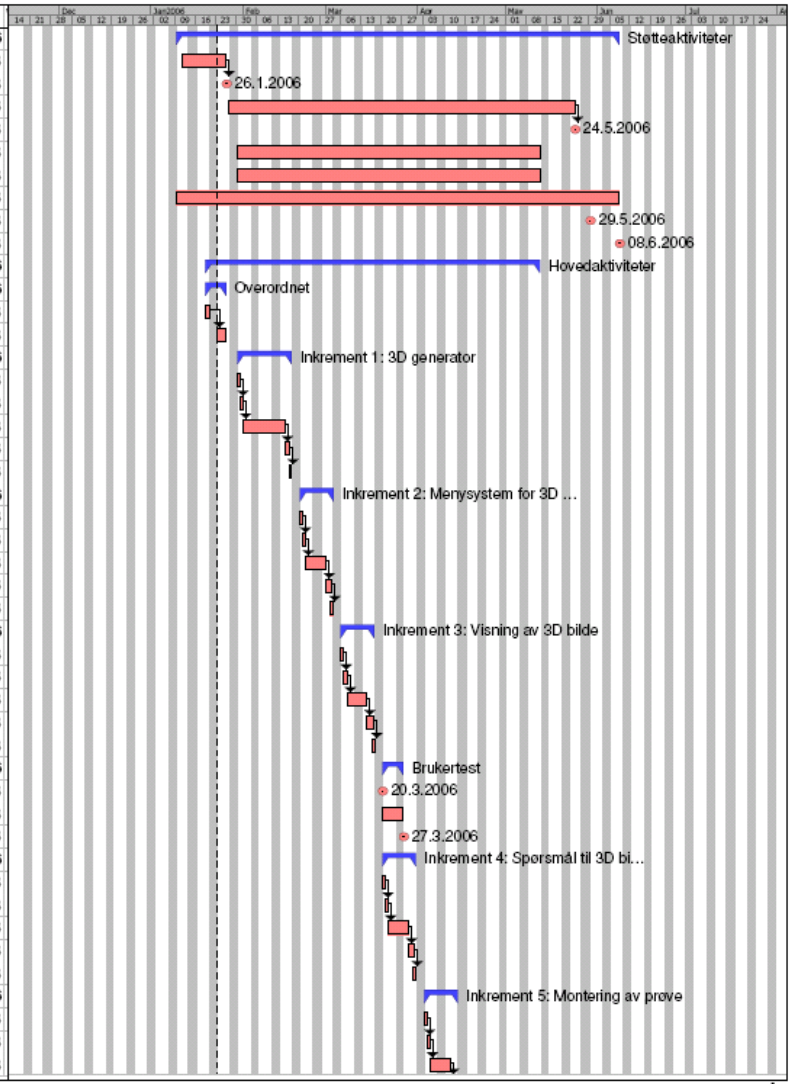
Kontakten med veileder er upåklagelig. Ukentlige møter med veileder gir gode muligheter til å korigere kursen på prosjektet. I tillegg stiller veileder seg ofte tilgjengelig utenom den avtalte tiden. Kos som prosjekt stilles egentlig i en liten særstilling på dette punktet siden veilederen helt fra starten av har ønsket å delta mest mulig og har vært pådriver bak prosjektet. For ikke å glemme at det er han som kom med idéen i utgangspunktet.

Tillegg I

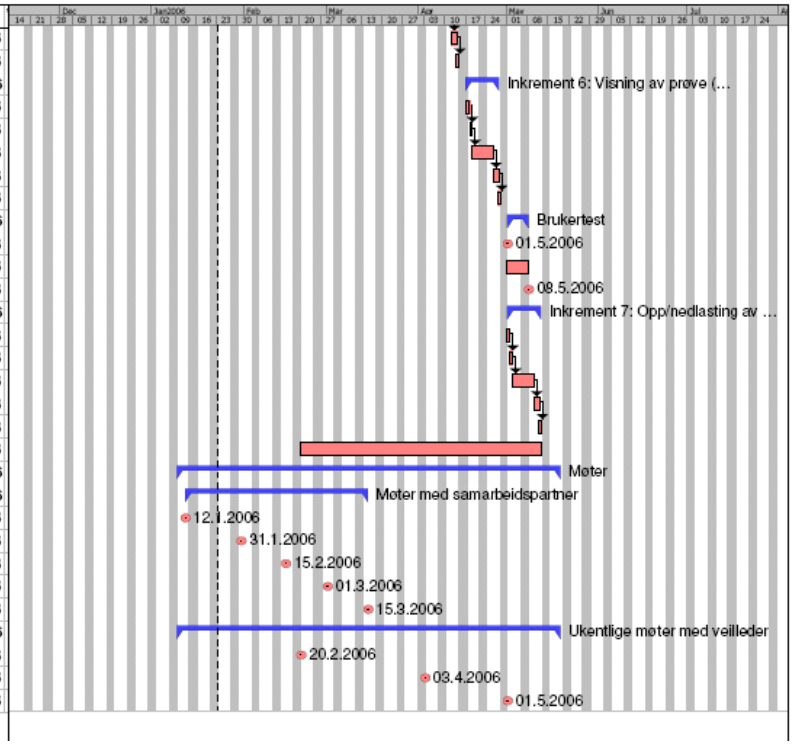
Gantt diagram

Dette er den fremdriftsplanen som ble utarbeidet rett etter prosjektstart. Denne samsvarer ikke med reell fremdrift, siden omfanget av oppgaven ble veldig endret etterhvert. Det kommer nå ganske tydelig frem at denne fremdriftsplanen var litt vel ambisiøs. Viktige møter og milepæler er også avmerket på skjemaet. Møtene og milepælene ble overholdt på tross av endret oppgavebeskrivelse.

ID	Description	Start	End
1	**Støtteaktiviteter**	**09.1.2006**	**08.6.2006**
2	Utarbeiding av forprosjektplan	11.1.2006	26.1.2006
3	Innlevering av forprosjektrapport	26.1.2006	26.1.2006
4	Utarbeiding av prosjektrapport	27.1.2006	24.5.2006
5	Innlevering av prosjektrapport	24.5.2006	24.5.2006
6	Dokumentering i JavaDOC	30.1.2006	12.5.2006
7	Dokumentering i kode	30.1.2006	12.5.2006
8	Loggføring av timer	09.1.2006	08.6.2006
9	Levering av laminert plakat	29.5.2006	29.5.2006
10	Presentasjon av prosjekt	08.6.2006	08.6.2006
11	**Hovedaktiviteter**	**19.1.2006**	**12.5.2006**
12	**Overordnet**	**19.1.2006**	**26.1.2006**
13	Kravspesifisering	19.1.2006	20.1.2006
14	Analyse og design	23.1.2006	26.1.2006
15	**Inkrement 1: 3D generator**	**30.1.2006**	**17.2.2006**
16	Kravspesifisering	30.1.2006	31.1.2006
17	Analyse og design	31.1.2006	01.2.2006
18	Implementering	01.2.2006	15.2.2006
19	Testing	15.2.2006	17.2.2006
20	Oppsummering	17.2.2006	17.2.2006
21	**Inkrement 2: Menysystem for 3D gen**	**20.2.2006**	**03.3.2006**
22	Kravspesifisering	20.2.2006	21.2.2006
23	Analyse og design	21.2.2006	22.2.2006
24	Implementering	22.2.2006	01.3.2006
25	Testing	01.3.2006	03.3.2006
26	Oppsummering	03.3.2006	03.3.2006
27	**Inkrement 3: Visning av 3D bilde**	**06.3.2006**	**17.3.2006**
28	Kravspesifisering	06.3.2006	07.3.2006
29	Analyse og design	07.3.2006	08.3.2006
30	Implementering	08.3.2006	15.3.2006
31	Testing	15.3.2006	17.3.2006
32	Oppsummering	17.3.2006	17.3.2006
33	**Brukertest**	**20.3.2006**	**27.3.2006**
34	Utvelging av testere	20.3.2006	20.3.2006
35	Brukertesting	20.3.2006	27.3.2006
36	Debriefing av testere	27.3.2006	27.3.2006
37	**Inkrement 4: Spørsmål til 3D bilde**	**20.3.2006**	**31.3.2006**
38	Kravspesifisering	20.3.2006	21.3.2006
39	Analyse og design	21.3.2006	22.3.2006
40	Implementering	22.3.2006	29.3.2006
41	Testing	29.3.2006	31.3.2006
42	Oppsummering	31.3.2006	31.3.2006
43	**Inkrement 5: Montering av prøve**	**03.4.2006**	**14.4.2006**
44	Kravspesifisering	03.4.2006	04.4.2006
45	Analyse og design	04.4.2006	05.4.2006
46	Implementering	05.4.2006	12.4.2006



ID	Description	Start	End
47	Testing	12.4.2006	14.4.2006
48	Oppsummering	14.4.2006	14.4.2006
49	**Inkrement 6: Visning av prøve (Egen applikasjon)**	**17.4.2006**	**28.4.2006**
50	Kravspesifisering	17.4.2006	18.4.2006
51	Analyse og design	18.4.2006	19.4.2006
52	Implementering	19.4.2006	26.4.2006
53	Testing	26.4.2006	28.4.2006
54	Oppsummering	28.4.2006	28.4.2006
55	**Brukertest**	**01.5.2006**	**08.5.2006**
56	Utvelging av testere	01.5.2006	01.5.2006
57	Brukertesting	01.5.2006	08.5.2006
58	Debriefing av testere	08.5.2006	08.5.2006
59	**Inkrement 7: Opp/nedlasting av spørsmål**	**01.5.2006**	**12.5.2006**
60	Kravspesifisering	01.5.2006	02.5.2006
61	Analyse og design	02.5.2006	03.5.2006
62	Implementering	03.5.2006	10.5.2006
63	Testing	10.5.2006	12.5.2006
64	Oppsummering	12.5.2006	12.5.2006
65	Utarbeide 3D modeller	20.2.2006	12.5.2006
66	**Møter**	**09.1.2006**	**19.5.2006**
67	**Møter med samarbeidspartner**	**12.1.2006**	**15.3.2006**
68	Møte med Thomtes Trafikkskole	12.1.2006	12.1.2006
69	Møte med Thomtes Trafikkskole	31.1.2006	31.1.2006
70	Møte med Thomtes Trafikkskole	15.2.2006	15.2.2006
71	Møte med Thomtes Trafikkskole	01.3.2006	01.3.2006
72	Møte med Thomtes Trafikkskole	15.3.2006	15.3.2006
73	**Ukentlige møter med veileder**	**09.1.2006**	**19.5.2006**
79	Statusrapport 1	20.2.2006	20.2.2006
80	Statusrapport 2	03.4.2006	03.4.2006
81	Statusrapport 3	01.5.2006	01.5.2006



Tillegg J

Logger

J.1 Møtelogg

Dette er en oversikt over hvilke møter som er holdt og hvem som var tilstede på møtet.

Dato:	Sted:	Deltagere	Temaer
20/1-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Prosjektavtaler Teknologivalg
28/1-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Første prototype Hva skal presenteres hos Thomtes
31/1-06	Thomtes Trafikkskole	Ann Kristin Thomte Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Presentere idéen Vise første prototypen

Dato:	Sted:	Deltagere	Temaer
3/2-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Kontrakter Oppmerking i vei Piler Terreng
10/2-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Presentasjon av tek. demo Hva skal vises til Thomtes Installasjonsveivisere
15/2-06	Thomtes Trafikkskole	Ann Kristin Thomte Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Presentasjon av ny prototype
21/2-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Statusrapport Nye krav til modeller Brukergrensesnitt
24/2-06	HiG	Frode Haug Jonas Strømstad	Evt. intervju til avis.
1/3-06	Thomtes Trafikkskole	Ann Kristin Thomte Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Tilbakemeldinger på første beta versjon
1/3-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Grensesnitt for opptegning av piler
3/3-06	HiG	Frode Haug Sverre Bakke Jonas Strømstad	Forespørsel om å presentere prosjektet for elever fra FiG.
10/3-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Situasjonskontroll

Dato:	Sted:	Deltagere	Temaer
15/3-06	Thomtes Trafikkskole	Ann Kristin Thomte Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Nyeste prototype Situasjonskontrollering Scenario begrepet
15/3-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Scenario vs. situasjon Pakke kontroll Bukergrensesnitt
17/3-06	HiG	Frode Haug Sverre Bakke Jonas Strømstad	Innstillinger Stillbilde i spørsmålsgenerator
31/3-06	HiG	Frode Haug Jonas Strømstad	Nyheter: Gangfelt Spørsmålsmodus Redusering av oppgave omfang
3/4-06	Thomtes Trafikkskole	Ann Kristin Thomte Sverre Bakke Kenneth Rinnan Jonas Strømstad	Tilbakemeldinger på siste betaversjonen. Alle forslag til endringer etter dette ses bortfra.
21/4-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Spørsmålsgenerator Gjennomgang av dokumentasjon
28/4-06	HiG	Frode Haug Sverre Bakke Kenneth Rinnan Jonas Strømstad	Gjennomgang av dokumentasjon

J.2 Aktivitetslogg

Uke 2: 9 januar - 15 januar

kenneth: 34
jonas: 30
sverre 30

Innledende arbeid. Forprosjekt rapport. Møter med thomtes og oppdragsgiver. Tilegning av kunnskap (OpenGL/Java).

Uke 3: 16 januar - 22 januar

kenneth: 41
jonas: 30
sverre: 34

Forprosjekt. Versjonskontroll. Tilegning av kunnskap om OpenGL. Vurdering av teknologi (ms3d). Lagd loggsiden. Gantt skjema. Opplæring i 3D-modellering. Introduksjon til JOGL. Portet milkshape loader. Nettsiden til prosjektet.

Uke 4: 23 januar - 29 januar

kenneth: 34
jonas: 29
sverre: 42

Begynt på den faktiske koden med musstyring. Begynt modellering. Lest videre på jogl. Screenshot. Gjort endringer på forprosjekt etter ønske fra veileder. Prosjektavtaler. Skybox. Benchmarking av ytelse. Møte med Thomtes. Kravspesifikasjon. Lasting, velging, rotering og flytting av objekter.

Uke 5: 30 januar - 5 februar

kenneth: 54
jonas: 31
sverre: 35

Visjonsdokument. Usecase. Modellering, Lasting, velging, rotering og flytting av

objekter. Lys. Kravspek. Fog/tåke. 2D view. Forbedret valg av objekter. Skalering. Fjerne objekter. Teksturer. Modellert Audi, Personbil, traktor, menneske.

Uke 6: 6 februar - 12 februar

kenneth: 28
jonas: 34
sverre: 26

Waypoints. Modellering. Kamera. Opprydding i testkode. De første skilt-modeller. Pbuffer (forkastet).

Uke 7: 13 februar - 19 februar

Kenneth: 30
Jonas: 32
Sverre: 30

Modellert trafikkskilt. Kravspesifikasjon, Installer til applikasjonen for distribuering til Thomtes. Generelt prototyping av mulig kode.

Uke 8: 20 februar - 26 februar

kenneth: 30
jonas: 33
sverre: 30

Kravspesifikasjon. Modellering. Drag'n'drop. GUI. Filformat/xml. Bugfikser. Generell prototyping av mulig kode.

Uke 9: 27 februar - 5 mars

kenneth: 33
jonas: 35
sverre: 27

Newdialog. Selectionlistener (oppdatere GUI ved events i 3D). Llagring. Fullskjerm (forkastet). Flere situasjoner samtidig.

Uke 10: 6 mars - 12 mars

kenneth: 30
jonas: 34
sverre: 54

Restrukturering av kode. Opprettelse av en solid GUI struktur. Oppdatering av GUI. Portet JOGL-koden til nyeste JOGL-versjonen. Flersituasjon (+GUI). Speiling. Mulighet for flere canvaser (forkastet). Egen tabbedpane for å løse flercanvas problematikken (forkastet). Workaround for jtabbedpane (forkastet).

Uke 11: 13 mars - 19 mars

kenneth: 30
jonas: 30
sverre: 31

Situasjonskontroller. Waypoints. Taskbar for scenarier. Objektverktøylinje. Bugfikser. Integrert GUI med 3D. Modellering av trafikklys. Generell ressurskode (innlesing av bilder og internasjonalisering). Mulighet for flere scenarier.Bezierkurver + synkronisering av farger i GUI.

Uke 12: 20 mars - 26 mars

kenneth: 30
jonas: 32
sverre: 29

Internasjonalisering. Tooltips. Lagring. Matteopplæring. Laget masse trafikkskilt. Flyttet på gui etter ønske fra oppdragsgiver.

Uke 13: 27 mars - 2 april

kenneth: 30
jonas: 31
sverre: 30

Piler på waypoints. Options. Fotgjengeroverganger. Internasjonalisert.

Uke 14: 3 april - 9 april

kenneth: 27
jonas: 28
sverre: 26

Spørsmålsgenerering. Kan endre tittel på scenario i GUI'et. Modellert elg og katt. Bugfikset.

Uke 15: 10 april - 16 april (påskeferie)

kenneth: 8
jonas: 14
sverre: 10

Kravspesifikasjon. Bugfikser. Jobbet med ńtodoż listen til frode. Spørsmålsgeneratoren. Designdokument.

Uke 16: 17 april - 23 april

kenneth: 30
jonas: 30
sverre: 31

Designdokument. Kravspesifikasjon. Forbedring av datastrukturen for modeller.

Uke 17: 24 april - 30 april

kenneth: 31
jonas: 31
sverre: 30

Designdokument. Kravspesifikasjon. Forbedring av kildekode. Prosjektrapport. Teoriprøve-modul.

Uke 18: 1 mai - 7 mai

kenneth: 35
jonas: 30
sverre: 31

Designdokument. Kravspesifikasjon. Forbedring av kildekode. Prosjektrapport. Teoriprøve-modul.

Uke 19: 8 mai - 14 mai

kenneth: 38
jonas: 36
sverre: 35

Prosjektrapport. Forbedring av kildekode.

Uke 20: 15 mai - 21 mai

kenneth: 50
jonas: 47
sverre: 45

Prosjektrapport.