HOVEDPROSJEKT:

iSlideS

SLIDESHOWGENERATOR

FORFATTERE:

TOMAS HENSRUD GULLA
HAAKON SPORSHEIM

DATO:

14.05.2004

SAMMENDRAG AV HOVEDPROSJEKT

Tittel:	iSlideS – Slideshowgenerator (norsk) iSlideS – Slide Show Generator (English)	Nr. : 09 Dato : 14/5-2004
Deltakere:	Tomas Hensrud Gulla Haakon Sporsheim	
Veileder:	Førsteamanuensis dr. scient. Ivar Farup	
Oppdragsgiver:	Maxus Media & Software	
Kontaktperson:	Sven Berg Ryen	
Stikkord (4 stk)	Slideshow, bildebehandling, .NET, DirectX	

Antall sider: 98	Antall bilag: 14	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Vi har utviklet en slideshowgenerator for Windows-plattformen, en applikasjon som setter sammen bilder, musikk og lydeffekter til en lysbildefremvisning. Applikasjonen har også enkle bilderedigeringsfunksjoner, slik at bildene kan redigeres uten at brukeren trenger å benytte en annen applikasjon for dette. Det er lagt vekt på å utvikle et modulært system med dynamisk klasseloading, slik at ny funksjonalitet kan legges til som egne plugin, uten behov for å endre på resten av applikasjonen. Slideshowgeneratoren kan lagre et slideshow som en kjørbar fil, filen kan distribueres til andre, og kan spilles av uten å være i besittelse av selve slideshowgeneratoren.

Applikasjonen er utviklet ved hjelp av Microsoft Visual Studio .NET 2003 i programmeringsspråket C#. Dette er et nytt og fremtidsrettet utviklingsmiljø, som det har vært en fornøyelse å jobbe med.

Vi har valgt å jobbe etter en inkrementell systemutviklingsmodell. Dette kommer blant annet av at oppgaven enkelt kunne deles inn i inkrementer. I tillegg muliggjør denne modellen endring i krav underveis, og gir synlig resultat relativt tidlig i forhold til andre systemutviklingsmodeller.

Forord

iSlideS – Slideshowgenerator er utviklet av to studenter på datalinjen ved Høgskolen i Gjøvik våren 2004. Prosjektoppgaven ble utformet i samarbeid med oppdragsgiver Maxus Media & Software representert ved Sven Berg Ryen, og har bestått i å utvikle en moderne Windows-applikasjon for generering av digitale lysbildefremvisninger. Det er blitt lagt stor vekt på brukergrensesnitt og morgendagens utviklingsverktøy.

Vi ønsker takke følgende personer for deres bidrag til prosjektet

- Dr. Ivar Farup, vår veileder, som under hele prosjektperioden har vært interessert og engasjert i oppgaven vår, og kommet med mange gode råd og vink til selve utviklingsprosessen og systemutviklingsaktiviteter.

- Tom Røise som var veileder for oss i hele april måned, mens Ivar Farup hadde permisjon. Røise har en god kritisk sans, og ga oss også god veiledning i forhold til systemutviklingsaktiviteter.

- Josh McIntyre for design av bakgrunnsbildet til applikasjonens ”splash screen”.

- Erling Andersen, Anders Enger Jensen, Tarjei Mandt, Siw Monica Myhren, Kristian Sporsheim og Sverre Bakke, som har hjulpet oss med å forbedre resultatet i form av testing.

Gjøvik, 14. mai 2004.

Haakon Sporsheim	Tomas Hensrud Gulla

Innhold

Figurer

Tabeller

1 Innledning

Digitale fotokamera selges som aldri før og de gamle fotoapparatene blir mindre og mindre synlige. Dette betyr at svært mange oppbevarer bildene sine på en datamaskin og ikke i et fotoalbum i bokhylla. På markedet finnes et stort utvalg kjent programvare som tar vare på bilder, og der man kan organisere bildene sine akkurat slik man selv ønsker. Det finnes også utallige høykvalitets redigeringsprogram som kan gjøre alt fra å legge til/fjerne personer til å fjerne uønskede elementer og støy.

Digitale bilder er mye enklere å dele med andre, både på godt og vondt. Dette pga. "Internetts inntog i hjemmet" og andre kommunikasjonsmidler som behandler data. Et lite problem er hvordan man kan, og/eller skal formidle bilder til andre. På internett finnes det flere aktører som tilbyr sine kunder oppbevaringsplass der kunden selv kan lagre bilder, og på en eller annen måte publisere bildene sine til andre. En annen måte å dele bildene sine på er å sende bildene i for eksempel en e-post. Andre alternativer finnes i stor grad, men en svært attraktiv måte ville vært å kombinere flere viktige funksjoner fra forskjellig programvare og utvikle en ny måte å dele bilder med andre på. En mer underholdende fremvisning av bildene er mangelvare i eksisterende løsninger på markedet.

1.1 Oppgavedefinisjon / begrensinger

Oppdragsgiver, Maxus Media & Software representert ved daglig leder Sven Berg Ryen, ønsker å se på muligheter for å utvikle programvare for Windows-plattformen. Han ønsker å få et innblikk i hvilke muligheter som finnes, og se hva som er mulig å få til i løpet av et studentprosjekt. I den forbindelse skal prosjektgruppen utvikle en applikasjon som på en enkel måte skal kunne behandle, organisere og vise frem digitale bilder på en engasjerende og underholdende måte.

Applikasjonen som skal utvikles er en slideshowgenerator. Altså en applikasjon som skal kunne sette sammen bilder til et *slideshow*, en lysbildefremvisning med musikk, lydeffekter og flotte overganger mellom bildene. Denne lysbildefremvisningen skal siden kunne spilles av i selve slideshowgeneratoren, og i en egen fremviserapplikasjon uavhengig av slideshowgeneratoren. Det skal også være mulig å lagre slideshowet til en kjørbar fil som inneholder både slideshowet og fremviseren som viser slideshowet. Denne filen kan så distribueres til, og sees av, familie og venner som ikke har selve slideshowgeneratoren. Applikasjonen skal ha funksjoner for å organisere og vedlikeholde en samling av opprettede slideshow, slik at disse er lett tilgjengelig for visning, redigering og uthenting av enkeltbilder.

Siden digitale bilder i utgangspunktet sjelden er perfekte, skal det være mulig å foreta enkle redigeringsfunksjoner direkte i slideshowgeneratoren. Det kan for eksempel ofte være ønskelig å gjøre et utsnitt i bildet (beskjære det), justere farge/kontrast eller fjerne uønskede elementer.

Applikasjonen skal bygges opp på en modulær måte, slik at ny funksjonalitet kan legges til som en nye *moduler*, uten å endre på kjernen i applikasjonen.

I første omgang skal applikasjonen være engelskspråklig, men det skal tilrettelegges for eventuell oversetting til andre språk.

Det skal være mulig å generere et slideshow med standardinnstillinger med svært få museklikk, men det skal også lages muligheter for at mer avanserte og erfarne brukere detaljert skal kunne konfigurere hvordan slideshowet skal vises. Etter ønske fra oppdragsgiver skal det legges vekt på å skape en applikasjon med et tiltalende utseende og intuitivt grensesnitt.

Siden tiden er begrenset og prosjektgruppen kun består av to medlemmer har vi sett oss nødt til å sette visse begrensninger for applikasjonens omfang. Bilderedigerings-funksjonene skal holdes på et relativt enkelt nivå, og på ingen måte konkurrere med avanserte bilderedigeringsverktøy. Applikasjonen skal ikke håndtere direkte kommunikasjon med digitalkamera og skal først og fremst fungere på Windows XP og senere versjoner, det vil derfor ikke legges ned mye arbeid i å tilpasse applikasjonen til eldre versjoner av Windows. Det blir ansett som viktigere at kjernefunksjonaliteten er stabil, gjennomtenkt og funksjonell enn at det blir utviklet en stor mengde moduler. Derfor vil det bli lagt størst vekt på kjernefunksjonaliteten, og ikke laget flere moduler med ekstrafunksjonalitet enn nødvendig, dersom det ikke skulle bli ekstra tid til dette på slutten av prosjektet.

1.2 Prosjektorganisering

Oppdragsgiver:

Maxus Media & Software ved Sven Berg Ryen.

Veileder:

Førsteamanuensis dr. scient. Ivar Farup ved HiG[1]

Studenter:

Tomas Hensrud Gulla
Haakon Sporsheim

1.2.1 Oppdragsgiver

Prosjektets oppdragsgiver er firmaet Maxus Media & Software, representert ved daglig leder Sven Berg Ryen.

Firmaet leverer tjenester innen design og programmering. Hovedfokus er på web-løsninger som er oversiktlige og enkle i bruk. Publiseringssystemer for effektiv og enkel oppdatering av nettsteder har siden 2001 vært et kjerneområde for virksomheten.

Eksempler på andre tjenester:

- utforming av trykt materiell/brosjyrer
- utforming av logoer/grafisk profil
- søkeoptimalisering av internettsider
- support og opplæring

[1] HiG - Høgskolen i Gjøvik

1.2.2 Ansvarsforhold

Selv om gruppen består av kun to deltakere ble det besluttet å utnevne en prosjektleder, som vil ha siste ord i eventuelle uoverensstemmelser. Selv om vi har utnevnt en prosjektleder har ikke denne noe overordnet ansvar for prosjektet, utover å være kontaktperson mot oppdragsgiver og veileder. I tillegg har vi satt opp ansvarsområder som beskrevet i Tabell 1-1.

Tabell 1-1– Ansvarsforhold.

Prosjektleder	Haakon Sporsheim
Backup	Tomas H. Gulla
Dokument	Tomas H. Gulla
Prosjektwebside	Haakon Sporsheim
Versjonsstyring[2]	Haakon Sporsheim
Testing og kvalitetssikring	Begge
Møtereferat og loggføring	Begge

Begge er inneforstått med hvilket ansvar den enkelte har i forhold til hvert enkelt ansvarsområde og ansvaret i forhold til prosjektet som helhet. Hver enkelt er ansvarlig for å møte på avtalte møter samt forholde seg til alle avtalte tidsfrister. Ved avvik i forhold til interne avtaler av mindre art må alle involverte informeres i god tid. Andre avvik skal diskuteres i gruppen, og med veileder dersom det skulle være nødvendig.

Møtereferat og loggføring har begge ansvar for. Det vil si det skal veksles på å skrive møtereferat og føre logg.

1.3 Målgruppe

Vær oppmerksom på at målgruppen til rapporten er forskjellig fra målgruppen(e) til applikasjonene. Målgruppen(e) til applikasjonene vil sannsynligvis ikke lese rapporten, derimot bør rapportens målgruppe prøve applikasjonene, og være inneforstått med applikasjonenes målgruppe(r).

1.3.1 Rapporten

Rapporten vil i første rekke være rettet mot oppdragsgiver og sensor, men også mot andre som har interesse for oppgaven. Denne målgruppen fører til at rapporten vil bli av en mer teknisk art, og inneholde vurderinger av alternative løsningsmetoder i de tilfeller vi har stått ovenfor flere reelle alternativer. Noe forståelse for programmering/systemutvikling vil være en fordel, men ingen nødvendighet for å lese den. Rapporten er ikke ment som en brukerveiledning for applikasjonen, da applikasjonens hjelpesystem vil tjene dette formålet.

1.3.2 Oppgaven/Applikasjonene

Produktet som dette prosjektet resulterer i vil være rettet mot den vanlige datamaskinbruker, som benytter Microsoft Windows operativsystem. Den største målgruppen for slideshowgeneratoren ansees å være brukere med digitalt fotokamera.

[2] Versjonsstyring – CVS (Concurrent Versions System)

Disse brukerne vil generere fremviserapplikasjonen som da vil være rettet mot familie og bekjente. For at applikasjonen skal være så enkel som mulig å ta i bruk, vil vi forsøke å holde oss så nærme det som blir ansett som ”standard utseende” på det grafiske brukergrensesnittet. Dette vil bli gjort for at så mange som mulig skal føle seg ”hjemme” og at man skal finne funksjoner der det forventes at de er plassert.

1.4 Formål

Oppdragsgiver ønsket å se på mulighetene for å utvikle programvare for Windows. Hovedformålet med prosjektet er å utvikle en konkurransedyktig Windows-applikasjon med et tiltalende brukergrensesnitt og flotte visuelle effekter. Et formål med hovedprosjektperioden for prosjektgruppen vil være tilegning av ytterligere kunnskaper og ferdigheter innen prosjektstyring, systemutvikling og programmering.

1.5 Studentenes faglig bakgrunn

Begge studentene er studenter ved dataingeniørlinjen ved HiG og begge har tatt fordypning innen programutvikling, og er svært interessert i programmering.

Som et ledd i studiet har begge programmert en god del, og sammen har det blitt utviklet bl.a. en *IRC*-klient i programmeringsspråket Java som prosjektoppgave i faget Programvareutvikling[3]. Prosjektet resulterte til en enighet om å jobbe sammen med hovedprosjektet. I tillegg til erfaring med Java har Haakon en del erfaring med C++ Win32-programmering. Tidligere har begge også utviklet forskjellige enkle små applikasjoner basert på *DirectX* og *OpenGL*, noe som sikkert vil være nyttig under utviklingen av applikasjonen.

Mye av pensum ved studiet begge har vært gjennom vil være til stor nytte under gjennomføringen av hovedprosjektet, bl.a. systemutvikling, diverse programmeringsfag, matematikk og grafikk- og bildebehandling. Det tidligere nevnte IRC-klientprosjektet, i tillegg til andre mindre prosjekt, har vært gode gjennomkjøringer til hva som venter i hovedprosjektperioden.

1.6 Utviklingsmodell

En systemutviklingsmodell er et rammeverk for hele utviklingsprosessen. Den angir strukturen man arbeider etter i prosjektet, både med hensyn til hvordan man angriper problemstillingen og i hvilken rekkefølge de ulike aktivitetene utføres. Derfor er det viktig å velge en utviklingsmodell som passer dette spesielle prosjektet.

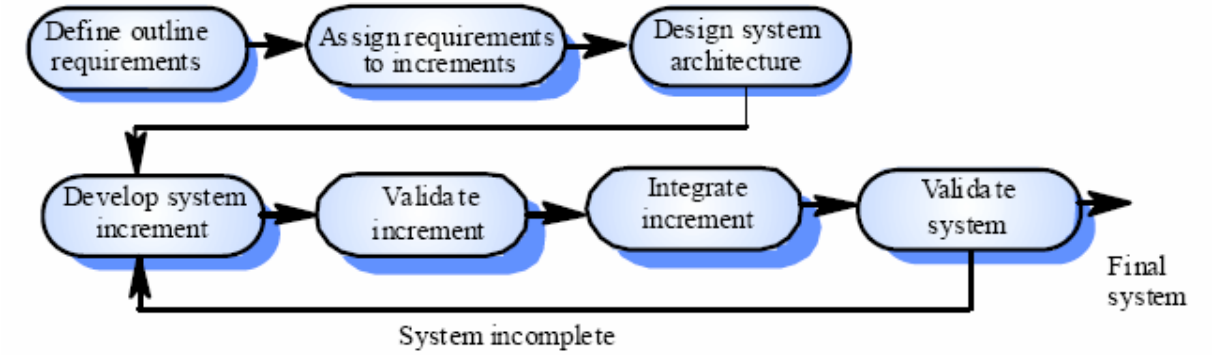
Etter å ha diskutert forskjellige alternative systemutviklingsmodeller, kom vi fram til at den inkrementelle systemutviklingsmodellen passer prosjektet vårt best. En av grunnene til at dette er en passende modell for prosjektet er at prosjektoppgaven lett kan deles opp i forskjellige moduler, som blir egne inkrement i utviklingen.

[3] IMT3281 Programvareutvikling ved HiG

Fossefallsmetoden ble vurdert, men vi kom frem at dens låsing av kravspesifiseringen tidlig i prosessen var uheldig i forhold til vår oppgave. I tillegg ville det med fossefallsmetoden ta lang tid før oppdragsgiver fikk se synlige resultater av prosjektet.

RUP[4] er en annen utviklingsmodell som er mye brukt og veldig sterk på dokumentasjon. Dette er en tungrodd utviklingsmodell, best egnet for større prosjekter, og med denne utviklingsmodellen ville vi antagelig måtte bruke mer tid på selve modellen slik at det ble mindre tid til å skape produktet. Vi vil allikevel kunne benytte enkelte metoder og teknikker fra denne utviklingsmodellen, for eksempel use case.

Tidlig i planleggingsfasen hadde vi stor tro på XP (eXtreme Programming), men siden vi ikke har oppdragsgiver tilgjengelig hele tiden ble denne modellen valgt bort. Pga. oppgavens nokså omfattende rammer, og gruppen kun består av to medlemmer, vil parprogrammering ikke strekke til tidsmessig. Vi har derimot gode erfaringer fra tidligere prosjekter, der vi har delt programmeringen mellom oss.



Figur 1-1 – Inkrementell systemutviklingsmodell

Med denne modellen (se Figur 1-1) får vi god struktur og styring. Systemet blir utviklet modul for modul, og hver modul kan testes hver for seg. Oppdragsgiver vil også tidlig i prosessen kunne se og prøve ut deler av systemet, slik at det blir lettere for ham å komme med tilbakemeldinger nye/endrede ønsker/krav. Ved valg av denne modellen kan man i forkant av prosjektet prioritere rekkefølgen man skal implementere modulene i, som resultat av dette vil man alltids ha noe å levere til slutt selv om tiden blir knapp og man ikke rekker alt.

1.7 Arbeidsform

Gruppen ble tildelt grupperom A115 sammen med seks andre hovedprosjektgrupper. Stort sett alt arbeidet med prosjektet vil bli utført på dette grupperommet, der begge gruppens medlemmer vil jobbe hver hverdag fra rundt kl 09.00-10.00 og utover, med noen unntak som ved høytid og lignende. Siden vi kun er to på gruppa har vi ikke satt opp noen faste interne gruppemøter, men vil diskutere ting etter hvert som de dukker opp. Det å jobbe på samme sted til samme tid vil nok være en stor fordel, siden problemer og uklarheter kan diskuteres og løses i fellesskap etter hvert som de dukker opp, istedenfor å bli satt til side og glemt.

[4] Rational Unified Process

Sammen med veileder har vi avtalt ukentlige møter, mandager kl 13.30. Med oppdragsgiver har vi ikke avtalt faste møtetidspunkt, men det vil bli avholdt møter når en av partene føler behov for det.

Gjennom hele prosjektets gang vil vi forsøke å jobbe litt med rapporten, i form av loggføring, møtereferater, statusrapporter og dokumentering av arbeidet etter hvert som det blir utført.

1.8 Organisering av rapporten

Høgskolen i Gjøvik har satt opp en mal for hvordan en hovedprosjektrapport kan/bør organiseres, denne malen er brukt som utgangspunkt for oppbygningen av denne rapporten.

1.8.1 Kapitteloppsummering

Etter konsultasjon med veileder ble det besluttet å dele rapporten inn i følgende ni kapitler.

Kapittel 1 – Innledning

Inneholder en overordnet beskrivelse av formålet med prosjektet, samt en beskrivelse av prosjektets ansvarsforhold og de involverte.

Kapittel 2 – Kravspesifikasjon

Inneholder alle krav til applikasjonen som forelå før selve utviklingen startet.

Kapittel 3 – Design

Inneholder designet av løsningen, det vil si hvordan vi vil løse oppgaven.

Kapittel 4 – Implementering

Inneholder en beskrivelse av hvordan vi har implementert løsningen i forhold til kravspesifikasjonen og designet, samt en beskrivelse av utviklingsmiljø, verktøy og valg av disse.

Kapittel 5 – Kvalitetssikring og testing

Inneholder en beskrivelse av de rutiner og aktiviteter som er fulgt og utført i forbindelse med testing og kvalitetssikring av løsningen.

Kapittel 6 – Diskusjon av resultater

Inneholder kritikk og evaluering av det utførte arbeidet, beskrivelse av mulig videre-utvikling.

Kapittel 7 – Konklusjon

Inneholder prosjektets konklusjon og oppsummering.

Kapittel 8 – Referanser

Inneholder en nummerert oversikt over rapportens referanser, trykte og elektroniske.

Kapittel 9 – Vedlegg

Inneholder en oversikt over alle vedleggene til denne rapporten.

1.8.2 Layout

Rapportens hoveddeler vil være organisert i kapitler, der hvert kapittel vil ha underkapitler for forskjellige emner innen hver hoveddel. Disse underkapitlene vil også inneholde en videre inndeling. De tre første nivåene er nummerert, disse finnes også i innholdsfortegnelsen.

Ord som finnes i terminologilisten (vedlegg A) er uthevet med *kursiv* første gang de forekommer i rapporten. Utover dette vil den samme standardfonten bli benyttet gjennom hele rapporten, for å gi et ryddig og helhetlig inntrykk.

I tråd med standarder for faglitteratur starter vanlig sidenummerering fra første kapittel, mens de foregående sidene er nummerert med romertall.

2 Kravspesifikasjon

2.1 Innledning

Løsningen vil bestå av to applikasjoner, selve slideshowgeneratoren og en egen fremviserapplikasjon.

2.1.1 Slideshowgenerator

Her skal man kunne sette sammen bilder til et slideshow og legge til musikk og/eller lydeffekter. Slideshowet kan siden lagres, vises og eksporteres til en kjørbar fremviser-applikasjon.

2.1.2 Fremviserapplikasjon (Player)

Fremviserapplikasjonen er en applikasjon generert av en slideshowgenerator. Denne applikasjonen inneholder et slideshow, og brukes kun til å vise dette slideshowet.

2.1.3 Kort om krav til applikasjonene

Siden begge applikasjonene, både slideshowgeneratoren og fremviserapplikasjonen, skal kunne vise et slideshow vil denne funksjonaliteten være lik i begge applikasjonene, og kravene til den vil også være like. Brukeren av applikasjonene skal kunne vise frem et slideshow, med dets bilder, overganger og musikk. Under fremvisningen av slideshowet skal brukeren ha tilgang på en navigasjonsmeny, som skal kunne brukes til å navigere frem og tilbake i slideshowet og sette slideshowet på pause. Det skal også være mulig å benytte hurtigtaster for funksjonene i navigasjonsmenyen. Visningen av slideshowet, og brukerens navigasjonsmuligheter, vil variere i forhold til de valg brukeren har foretatt under opprettelse av slideshowet.

Slideshowgenerator

Brukeren av applikasjonen skal kunne importere bilder tatt med for eksempel digitalkamera og sette disse sammen til et slideshow. Brukeren skal selv kunne konfigurere overgangene mellom hvert enkelt bilde. I tilegg skal det være mulig å sette inn musikk og lydeffekter. Når brukeren har laget slideshowet ferdig skal det være mulig å eksportere slideshowet til en kjørbar applikasjon som brukeren kan sende til familie og venner.

Fremviserapplikasjon (Player)

Slideshow skal kunne distribueres som én kjørbar fil bestående av både slideshowet og funksjonalitet for å vise det.

2.1.4 Applikasjonenes målgruppe

Slideshowgenerator

Brukere av Microsoft Windows som har en bildeserie de vil lage en fremvisning av. Bildeserien kan være bilder hentet fra et digitalt fotokamera, men kan også for eksempel være bilder laget av en artist, som ønsker å vise sin portofolie som et slideshow.

Fremviserapplikasjon (Player)

Brukere av Microsoft Windows som har mottatt eller lastet ned et slideshow generert i en slideshowgenerator.

2.1.5 Operasjon

Slideshowgenerator

Brukeren av slideshowgeneratoren skal enkelt kunne sette sammen et slideshow vha en *wizard* og/eller manuelt endre slideshowets egenskaper, legge til og fjerne bilder, musikk og lydeffekter. De fleste operasjoner vil kunne utføres med musepeker, men der det trengs å sette inn tekst, må selvsagt tastaturet benyttes.

Fremviserapplikasjon (Player)

Brukerens muligheter til å operere applikasjonen vil bli spesifisert av brukeren som har laget fremvisningsapplikasjonen vha. slideshowgeneratoren. Brukeren kan ha mulighet til

- å navigere fram og tilbake mellom de forskjellige bildene.
- å velge om applikasjonen skal vises i fullskjerm eller i et vindu.

For å navigere mellom bildene vil brukeren bruke musepekeren eller hurtigtaster. Brukeren skal kunne forandre fullskjerm/vindu modus før slideshowet vises.

2.1.6 Aspekter omkring livssyklus

Applikasjonene skal bygges opp på en modulær måte, slik at ny funksjonalitet kan legges til som nye moduler, uten å endre på applikasjonens kjernefunksjonalitet. Dette vil gjøre det enkelt å legge til ny funksjonalitet senere. For at dette skal kunne fungere i praksis må det opprettes egne grensesnitt for de forskjellige typer funksjonalitet, disse grensesnittene definerer hvilke funksjoner de forskjellige modultypene må ha.

Siden hvem som helst kan lage nye moduler, vil det ligge til rette for at applikasjonene har lang levetid, og ikke trenger å kompileres på nytt hver gang ny funksjonalitet skal legges til. Slideshowgeneratoren vil dynamisk benytte de moduler som er tilgjengelig. Dette betyr også at det må etableres en form for dokumentasjon til utviklere som har interesse av å utvikle moduler.

2.1.7 Ytelse

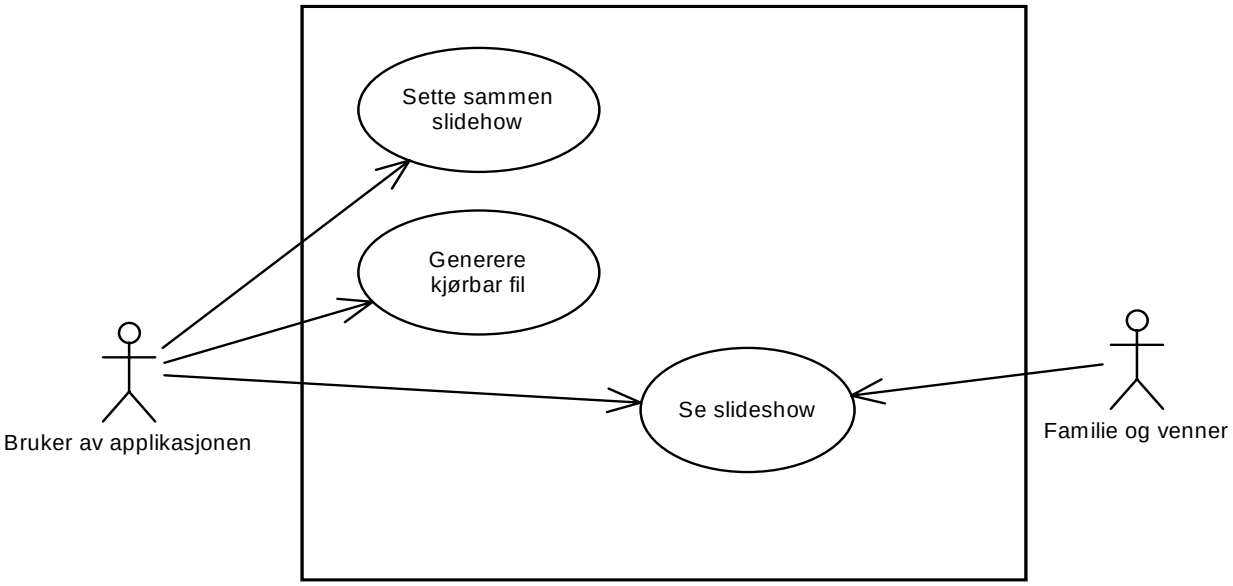
Fremviserdelen av applikasjonene kan kreve høy ytelse av maskinen om overgangene mellom bildene krever mye prosessering. Dette kommer mye an på hva slags grafikk-hardware brukeren har. Applikasjonene skal først og fremst utvikles og testes mot Windows XP.

2.1.8 Begrensninger

Applikasjonen skal ikke håndtere direkte kommunikasjon med digitalkamera. Bilderedigeringsfunksjonene skal holdes på et relativt enkelt nivå, og på ingen måte konkurrere med avanserte bilderedigeringsverktøy. Applikasjonen skal først og fremst fungere på Windows XP, og senere versjoner.

2.2 Use case

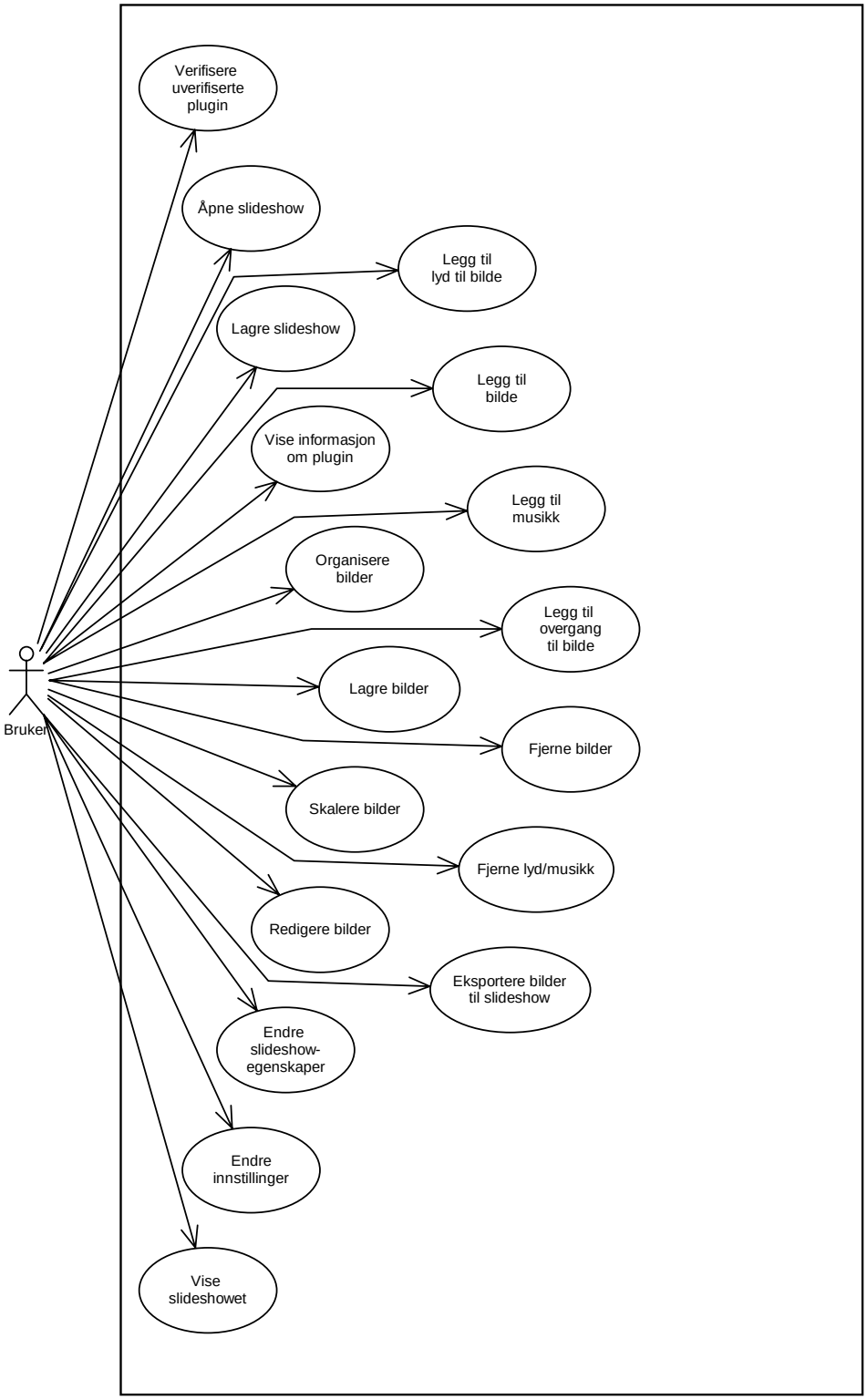
2.2.1 Overordnet funksjonalitet



Figur 2-1 – Use case diagram for applikasjonens overordnede funksjonalitet

Figur 2-1 illustrerer at brukeren av selve slideshowgeneratoren både skal kunne sette sammen og se på et slideshow, mens mottager av den kjørbare filen (i dette tilfellet familie og venner) kun har muligheten til å se på slideshowet. Se vedlegg B for tilhørende use case-beskriveler.

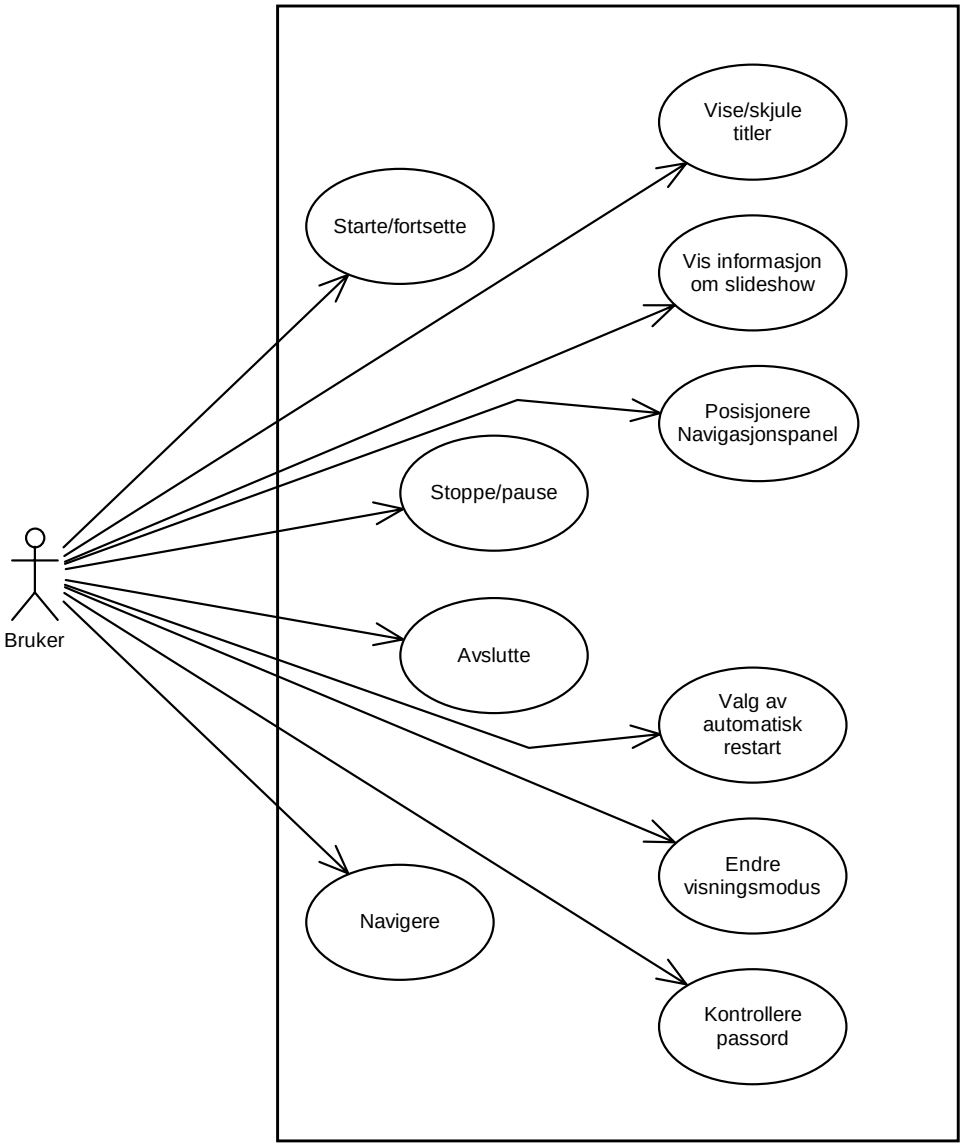
2.2.2 Slideshowgenerator



Figur 2-2 – Use case digram for slideshowgeneratorens funksjonalitet.

Figur 2-2 viser de forskjellige operasjonene brukeren av slideshowgeneratoren kan utføre. I korte trekk kan et slideshow genereres, lagres, åpnes, redigeres og lagres som en kjørbar fil. Se vedlegg B for tilhørende use case-beskriveler.

2.2.3 Fremviserapplikasjon (Player)



Figur 2-3 – Use case diagram for fremviserapplikasjonens funksjonalitet.

Figur 2-3 viser de forskjellige operasjoner mottager av den kjørbare filen kan utføre i forbindelse med avspilling av slideshowet. Se vedlegg B for tilhørende use case-beskriveler.

2.3 Detaljert kravspesifikasjon

2.3.1 Funksjonell struktur, spesifikasjon og tverrelasjoner

Slideshowgenerator

Brukeren skal kunne:

- Opprette nytt slideshow.
- Åpne eksisterende slideshow.
- Inkludere nye bilder i slideshowet.
- Velge hvilke overganger som skal benyttes mellom de enkelte bildene, og bestemme parametere for den enkelte overgang.
- Velge hvilke lydeffekter som eventuelt skal benyttes mellom bildene.
- Velge hvilken musikk som eventuelt skal spilles til slideshowet.
- Lagre slideshowet til fil.
- Lagre slideshowet til en kjørbar fil, og velge rettigheter for de som skal kjøre filen.
- Spille av slideshowet.

Brukerens valg under opprettelse av slideshow

Ved opprettelse av et nytt slideshow skal bruken kunne gjøre en hel rekke valg som har konsekvenser for hvordan fremvisningen av slideshowet vil se ut.

Bakgrunnsfarge

Hvilken farge som skal vises før første slide, og rundt eventuelle bilder som ikke dekker hele skjermens flate.

Titler

To forskjellige titler kan vises under visningen av slideshowet, slideshowets tittel og slidenes titler. Ved begge typer titler kan brukeren gjøre de samme innstillinger.

- Om de skal vises
- Posisjon
- Font
- Størrelse
- Skriftfarge
- Omrissfarge
- Justering (høyre, venstre eller midtjustert innenfor valgt område).

Slideshowets tittel vil være uforandret gjennom hele fremvisningen, mens slidetitlene vil unike for hver enkelt slide.

Navigasjonsmenyens plassering

Hvor navigasjonsmenyen er plassert når den dukker opp.

Loop

Om slideshowet skal spilles kun én gang, eller om det skal gå i en evig løkke (loopes).

Scale

Om bilder skal vises i sin opprinnelige størrelse, eller om de skal skaleres for å tilpasses skjermens oppløsning, og om de i så fall skal beholde opprinnelig lengde-/breddeforhold.

Cursor

Hvordan musepeker skal se ut under fremvisningen, her er det også mulig med egendefinert musepeker (CUR-fil).

Cursor Timeout

Hvor lang tid det skal ta før musepeker skjules etter siste musebevegelse.

NavigationMenu Timeout

Hvor lang tid det skal ta før navigasjonsmenyen skjules etter siste tastetrykk/musebevegelse.

Allow navigation

Om det skal være mulig å navigere (gå frem og tilbake) i slideshowet under fremvisning.

Allow pause

Om det skal være mulig å pause slideshowet.

Allow title toggling

Om det skal være mulig å skru titler (på selve slideshowet, og på enkeltslides) av og på under fremvisningen.

Bilderedigeringsverktøy

Brukeren av applikasjonen skal kunne bruke enkle bilderedigeringsfunksjoner, som bør være kjent fra andre bilderedigeringsapplikasjoner.

- Tegne med forskjellige ”brushes”.
- Beskjæring
- Rotering
- Skalering
- Legge på tekst
- Mulighet for å legge "clip-art"/små bilder over
- Auto-correct (forbedre kontrast, farger og hvitpunkt)
- Fjerning av røde øyne
- Kontrast/lysstyrke
- Fargebalanse
- Retusjering

Brukergrensesnitt

Brukergrensesnittet skal inneholde følgende hovedkomponenter:

- Menylinje.
- Verktøylinje med ofte brukte funksjoner lett tilgjengelig.
- Statuslinje.
- Slideshow Center, med oversikt over lagrede slideshow.
- Verktøyboks, med verktøy for bilderedigering.
- Slides, oversikt over bildene i gjeldende slideshow i liten størrelse.
- Valgt bilde, i stor størrelse slik at dette kan redigeres.
- Forskjellige dialoger.

Fremviserapplikasjon (Player)

Fremviserapplikasjonen kan kun brukes til å spille av slideshowet. Før slideshowet spilles av må det kontrolleres om slideshowet er passordbeskyttet.

Brukerens valg under visning av slideshow

Under visningen av slideshowet kan personen som viser slideshowet, dersom dette er godkjent av personen som genererte slideshowet, foreta følgende valg, enten fra navigasjonspanelet eller vha en kjent tastekombinasjon.

Vise/skjule titler

Brukeren skal kunne velge om slideshowets tittel, og de enkelte slidenes titler skal vises.

Pause

Brukeren skal, når som helst, under fremvisningen kunne fryse/pause fremvisningen.

Starte/fortsette fremvisningen

Dersom brukeren har satt slideshowet på pause, skal det også være mulig å starte det igjen.

Stoppe/avslutte fremvisningen

Brukeren skal når som helst under fremvisningen kunne avslutte fremvisningen. Under gitte betingelser skal fremvisningen avsluttes når slideshowet stoppes.

Navigere

Brukeren skal kunne navigere (gå frem og tilbake) i slideshowet under fremvisning.

Velge visningsmodus

Brukeren skal før visningen startes kunne velge mellom fullskjerm og vindusmodus, samt velge oppløsningen i fullskjerm eller størrelsen på vinduet i vindusmodus.

Posisjonere navigasjonsmeny

Brukeren skal ved hjelp av ”*drag and drop*” kunne endre navigasjonsmenyens plassering på skjermen.

Valg av automatisk restart

Brukeren skal kunne velge om fremvisningen skal starte på nytt etter visningen av siste slide (loopes), eller om den da skal stoppe.

Vise informasjon om slideshowet

Brukeren skal kunne vise en beskrivelse av slideshowet og informasjon om personen som har laget det.

2.3.2 Dataspesifikasjon

Datainput

Bilder

Brukeren av applikasjonen skal kunne benytte følgende bildeformat i sine slideshow:

- JPG (JPEG/JIFF Image)
- BMP (Windows OS/2 Bitmap Graphics)

Vi går ut ifra at de fleste digitalkameraer leverer fra seg bilder i JPG-format, og at dersom brukere av applikasjonen vår er i besittelse av bilder i andre formater, innehar de også applikasjoner som er i stand til å konvertere disse bildene til JPG-format. Det skal tilrettelegges for senere utvikling at støtte for nye bildeformater.

Lyd

Brukeren skal kunne benytte følgende lyd-/musikkformat i sine slideshow:

- MP3 (MPEG Audio Stream, Layer III)
- WAV (Waveform Audio)

Siden WAV-formatet er ukomprimert rådata er det hensiktsmessig å bruke dette formatet kun til de mindre lydeffektene og ikke lengre musikk. MP3 er et komprimert lydformat som i den senere tid har blitt svært vanlig.

Dataoutput

Kjørbar slideshowfil

Brukeren av applikasjonen skal kunne lagre sitt egenkomponerte slideshow som en kjørbar EXE-fil. Denne filen kan ikke senere redigeres, men kun spilles av, uavhengig av slideshowgeneratoren. Denne filen vil ha en datapakke lagt til på slutten, denne datapakken vil bestå av en *XML*-fil med beskrivelse av slideshowet samt en *ZIP*-komprimert katalogstruktur inneholdende de bilder, lydfiler og eventuelle moduler som er nødvendig for å vise slideshowet

Redigerbar slideshowfil

Brukeren av applikasjonen skal også kunne lagre sitt genererte slideshow til én fil, som senere kan redigeres, slik at nye bilder kan legges til, overganger endres osv. Denne filen vil være en ren XML-fil, som kun inneholder referanser til de data (bilder, musikk og moduler) som slideshowet benytter seg av. For å kunne spille denne filen trenger man selve sideshowgeneratoren.

2.3.3 Overordnede operasjonelle systemkrav

Modus og kontroll

Slideshowgenerator

Slideshowgeneratoren skal ha to hovedmodi.

Slideshowgenerering

Brukeren kan her sette sammen bilder, musikk og lydfiler. Mellom bildene kan brukeren spesifisere forskjellige overgangseffekter. Samtidig kan også bildene redigeres ved hjelp av applikasjonens redigeringsverktøy.

Fremvisning

Slideshowet skal kunne avspilles, enten i fullskjerm eller i et enkelt vindu. Det skal utvikles en navigasjonskontroll der man kan navigere fram eller tilbake, pause slideshowet og avslutte fremvisningen. Det skal også være mulig å benytte hurtigtaster for de samme valgene.

Fremviserapplikasjon (Player)

Fremviserapplikasjonen har kun én modus, fremvisning. Fremvisning vil være identisk med slideshowgeneratorens fremvisningsmodus.

Sikkerhet

Passordbeskyttelse

Når slideshow genereres skal det kunne passordbeskyttes, slik at slideshowet det ikke er tilgjengelig for uvedkommende. Passordet må ikke lagres i klartekst, men krypteres slik at det senere ikke kan hentes ut fra den passordbeskyttede filen.

Modul verifisering

Siden hvem som helst kan lage en modul, må hver modul returnere en signert *hash* til applikasjonen som bruker den, slik at applikasjonen kan *verifisere* den. Hvis en modul returnerer en ugyldig signert hash må applikasjonen få brukeren til å verifisere den

Hjelpesystem

Det skal utvikles et hjelpesystem til slideshowgeneratoren basert på Microsoft HTML Help. Dette er et verktøy som genererer en ”Hjelp-applikasjon” (CHM fil) ut ifra HTML-filer. Denne applikasjonen vil bli integrert i slideshowapplikasjonen slik at man kan få hjelp via hjelpalternativet på menyen, og ved for eksempel å trykke på en knapp i en dialog man er usikker på hvordan fungerer.

2.4 Begrensninger

2.4.1 Lydformat

Applikasjonen skal ikke kunne eksportere lyd til spesielle format, eller kunne konvertere mellom forskjellige lydformat.

2.5 Modul- og integrasjonstesting

Etter at hvert inkrement er ferdig skal den nye modulen testes for å sikre at den oppfyller kravene i kravspesifikasjonen og at den fungerer feilfritt. I tillegg skal inkrementet/modulen testes i sammenheng med den totale løsningen, for å sikre at samspillet med andre deler av systemet fungerer smertefritt.

2.6 Dokumentasjon

2.6.1 Prosess

Alle inkrement/moduler skal dokumenteres underveis i prosessen, enten mens de utvikles eller umiddelbart etter de er utviklet.

2.6.2 Resultat

Den ferdige applikasjonen skal inneholde et søkbart hjelpesystem, der brukeren enkelt skal kunne finne svar på sine spørsmål og kunne få nyttige tips om hvordan han kan skape fascinerende slideshow. Hjelpesystemet vil bli bygget opp vha HTML, slik at publisering på web vil være mulig uten mye ekstra arbeid.

2.6.3 Kildekode

Til dokumentering av kildekode vil vi benytte en standard støttet av et verktøy som kan generere dokumentasjon ut fra riktig kommentert kildekode.

Navngiving

For å kunne produsere kode som er vedlikeholdbar, oversiktlig og har et homogent utseende vil vi benytte en kodestandard. Vi vil ta utgangspunkt i *Hungarian Notation*, som er mye brukt, og tilpasse denne til vårt bruk.

Formatering

Alle krøllparenteser skrives på egen linje, og all kode mellom et par krøllparenteser rykkes inn med et tabulatorhopp. Koden deles opp i blokker med logisk sammenhørende kode, og blokkene skilles med linjeskift. Koden kommenteres, om nødvendig, før hver blokk.

2.7 Krav til installasjon og utgivelser

2.7.1 Utgivelser underveis

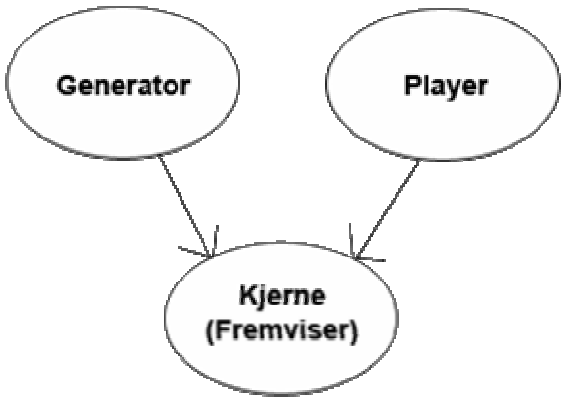
Det vil bli utgitt en ny testversjon etter hvert inkrement som er fullført. Testversjonene vil bli gjort tilgjengelig for oppdragsgiver og veileder, slik at disse også kan teste applikasjonene.

3 Design

3.1 Innledning

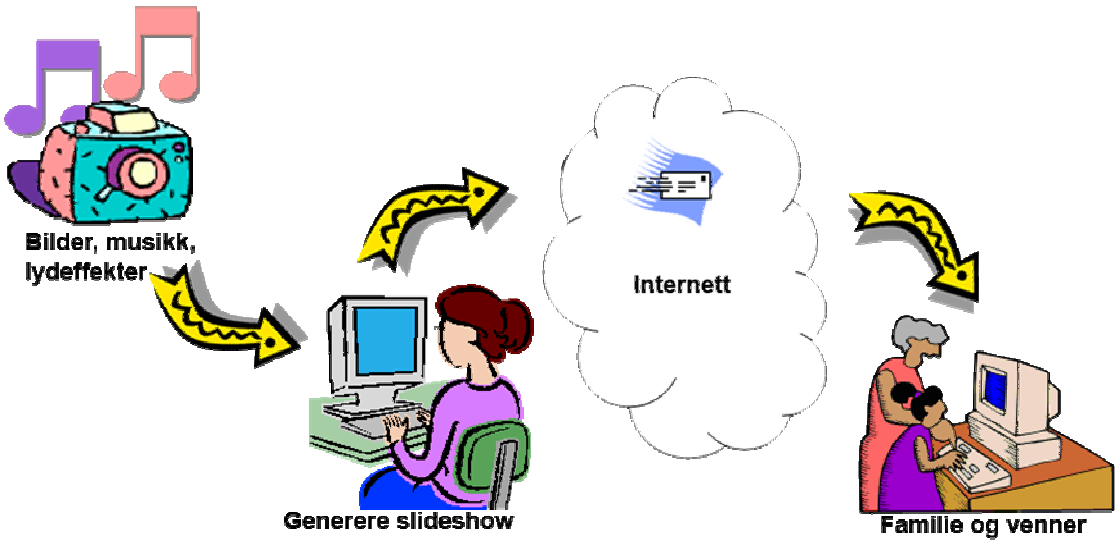
Løsningen vil bestå av to applikasjoner, en slideshowgenerator og en fremviserapplikasjon (Player). Disse to applikasjonene vil begge måtte innholde funksjonalitet for å åpne og spille av et slideshow.

Selve slideshowgeneratoren skal kunne sette sammen et slideshow og spille av dette. Slideshowgeneratoren skal også generere en kjørbar fil (fremviserapplikasjon) som inneholder både et slideshow og funksjonalitet for å spille av dette slideshowet. Dette betyr at fremviserdelen, som benyttes for å spille av et slideshow, trengs i begge applikasjonene, både i slideshowgeneratoren og i fremviserapplikasjonen. En mulighet her ville vært å legge til denne funksjonaliteten begge steder, altså utvikle først den ene applikasjonen for så å kopiere fremviserdelen til den andre applikasjonen. Dette kan raskt skape problemer dersom man senere vil endre på fremviserdelen. For å unngå dette problemet, og ved å bruke god programmeringsskikk, vil vi benytte samme funksjonalitet i begge applikasjonene vha et dynamisk link library (*DLL*). Begge applikasjonene vil benytte dette biblioteket, som vist i Figur 3-1, slik at vi minimerer plassbruk og andre problemer som kan oppstå.



Figur 3-1 – Fremviserdel som benyttes av både slideshowgeneratoren og fremviserapplikasjonen.

Et aktuelt scenario, som vist i Figur 3-2, vil da være at brukeren av slideshowgeneratoren setter sammen et slideshow med bilder tatt med eget digitalkamera, og sin favorittmusikk. Slideshowgeneratoren kan så generere en kjørbar fil som kan sendes til familie og venner via e-post, og disse kan se slideshowet selv om de ikke har slideshowgeneratoren.



Figur 3-2 – Distribuering av kjørbar slideshowfil.

Under kravspesifiseringen kom vi frem til at vi skulle utvikle et system der store deler av applikasjonens funksjonalitet skulle lastes dynamisk fra egne moduler under oppstart av applikasjonen. Det er spesielt fire typer funksjonalitet som over tid kan være aktuelt å legge til denne typen applikasjon; støtte for nye bildeformater, støtte for nye lydformater, nye overganger mellom bildene i slideshowet og nye bilderedigeringsverktøy. Vi har i den sammenheng valgt å kalle disse modulene for *plugin*.

3.2 Inndeling i inkrement

Vi har delt inn arbeidsmengden i seks deler, som blir egne inkrement i den inkrementelle systemutviklingen.

3.2.1 Kjerne-inkrementet:

Utvikle selve applikasjonsvinduet.

- Menyer
- Dialoger (f.eks. innstillingsdialog)
- Verktøylinje
- Statuslinje
- Språkuavhengighet

Utvikle ofte brukte objekter og grensesnitt, som brukes videre i utviklingen.

3.2.2 Generator-inkrementet:

Utvikle løsningen som brukes til oppbygningen av slideshow framvisningen.

- Beskrive filformat.
- Åpne fra, og lagre til, fil.
- Sette inn bilder, overganger, musikk mv.
- *GUI* for visning av bilder, overganger og musikk.

- Utvikle grensesnitt mot framviser og editor.

3.2.3 Framviser-inkrementet:

Utvikle lysbildefremviseren.

- Utvikle bildeoverganger, musikk, lydeffekter som plugin.
- Utvikle DirectX miljø.
- Sette opp grensesnittet mellom applikasjonen og plugin via DirectX.
- Utvikle tidsstyring for når plugin skal åpnes, kjøres og lukkes.

3.2.4 Editor-inkrementet:

Utvikle editoren der man kan redigere bilder. Utvikle funksjoner til editoren:

- Tegne med forskjellige ”brushes”.
- Beskjæring
- Rotering
- Skalering
- Legge på tekst
- Mulighet for å legge "clip-art"/små bilder over.
- Auto-correct (forbedre kontrast, farger og hvitpunkt)
- Fjerning av røde øyne
- Kontrast/lysstyrke
- Fargebalanse
- Retusjering

3.2.5 Player-inkrementet:

Utvikle fremviserapplikasjonen som kun viser lysbildene uavhengig av slideshowgeneratoren.

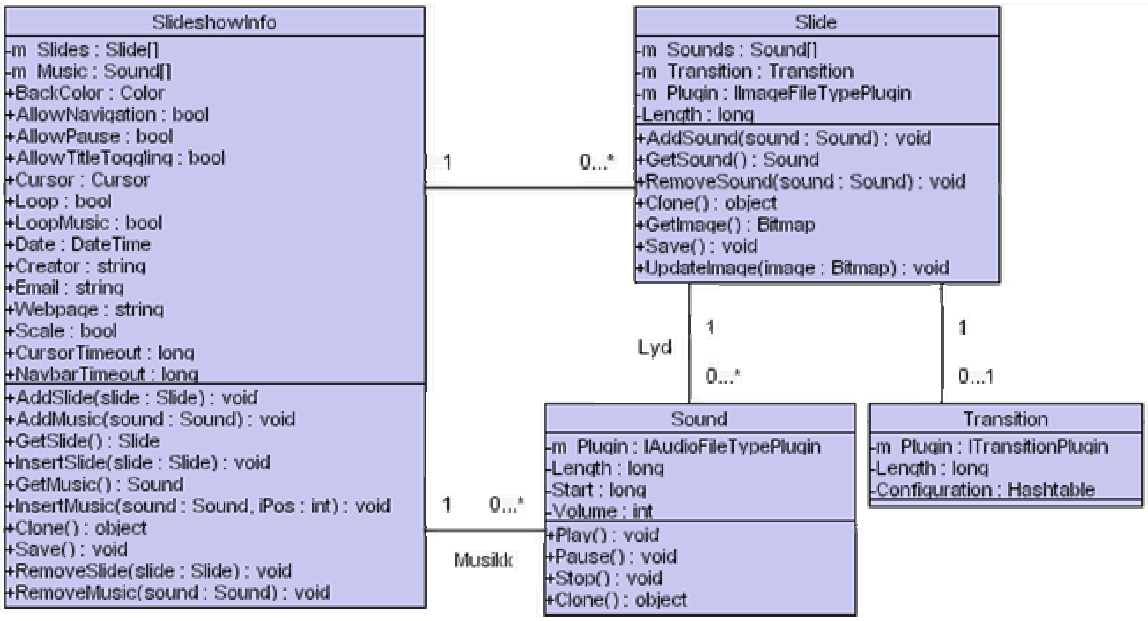
- Pakker alle nødvendige data ned i én fil.
- Bruker objekter laget i framviser-inkrementet.

3.2.6 Distribusjons-inkrementet:

- Utvikle en løsning for distribusjon av slideshowgeneratoren, og fremviser-applikasjonen

3.3 Softwarearkitektur

Figur 3-3 viser klassediagrammet over de fire klassene som danner kjernen av et slideshow. Dette er klasser som beskriver hvordan et slideshow skal vises og hvilke bilder, lyder og musikk som skal vises og spilles av.



Figur 3-3 – Kjernen av slideshowet.

3.3.1 Slideshowgenerator

Formålet med slideshowgeneratoren er at den skal være et miljø der brukeren kan utføre alle de operasjoner som er ønskelig ved generering, håndtering og visning av et slideshow. Slideshowgeneratoren bør være enkel å komme i gang med for uerfarne databrukere, samtidig som den skal tilby nødvendige funksjoner for å tilfredsstille mer avanserte brukeres krav.

3.3.2 Brukergrensesnitt i slideshowgenerator

Ved design av det grafiske brukergrensesnittet tegnet prosjektgruppen et forslag, og oppdragsgiver tegnet et forslag (vedlegg C). Disse skissene ble senere diskutert og endret. Vi vil forsøke å hente ut det beste fra disse to skissene i den endelige løsningen.

Det ble tidlig i prosessen klarlagt hvilke hovedkomponenter skjermbildet skulle bestå av, derimot var det forskjellige oppfatninger om hvor i skjermbildet de forskjellige komponentene burde plasseres, og hvor stor plass og viktighet disse skulle ta. Etter å ha diskutert forskjellige håndtegnede skisser ble vi enige om at det ville vært lettere å ta en avgjørelse hvis vi hadde hatt mulighet til å prøve de forskjellige mulighetene i praksis. På bakgrunn av dette ble det besluttet å legge til funksjonalitet for å endre plasseringen av applikasjonens hovedkomponenter. Vi vil siden bestemme oss for det oppsettet som virker mest hensiktsmessig, og bruke dette som standardoppsett i den ferdige applikasjonen. Avanserte brukere vil da også ha muligheten til å endre på dette oppsettet, hvis de har ønske om det.

Brukergrensesnittet skal gi mulighet for å sette sammen et slideshow på forskjellige måter. For avanserte brukere, eller de som ønsker full kontroll over hvordan det ferdige slideshowet skal se ut, skal det være mulig å gjøre å konfigurere parametere for hver enkelt bilde og hver enkelt overgang. Men for de som raskt vil sette sammen et slideshow

med standardverdier, skal det være mulig å velge en katalog med bilder og ved hjelp av få museklikk ha satt sammen et fullverdig slideshow. Selv om et slideshow er satt sammen på sistnevnte måte skal det være mulig å endre på de mer avanserte innstillingene i etterkant, dersom dette skulle være ønskelig.

Under utvikling av brukergrensesnittet vil vi forsøke å følge følgende prinsipper [1]:

- Gjenkjennelse – brukergrensesnittet skal bygges opp på en slik måte at applikasjonens målgruppe føler seg mest mulig hjemme, og finner funksjonalitet der den forventes å være. Siden applikasjonens målgruppe er en gjennomsnittlig Windows-bruker vil brukergrensesnittet bære preg av dette.

- Konsistens – brukergrensesnittet skal være mest mulig konsistent og helhetlig, og der det er mulig bør lignende operasjoner utføres på lignende måter.

- Ingen overraskelser – applikasjonen bør i oppføre seg slik brukeren forventer det, og ikke overraske/skremme brukeren unødig.

- Brukerveiledning – dersom feil skulle oppstå bør brukeren bli presentert med gode feilmeldinger som gjør at feilen kan rettes opp og bruken av applikasjonen fortsette.

- Bruker diversering – siden brukere er forskjellige, og tenker forskjellig, bør det være flere måter å utføre enkelte sentrale operasjoner på. Det bør f.eks. være mulig å opprette et slideshow raskt og enkelt, men det må også være mulig å gjøre mer avanserte valg for krevende/avanserte brukere.

GUI-komponenter

Menylinje

Menylinjen skal inneholde hovedmenyene file, edit, view, slideshow, image og help. Ved hjelp av disse menyene bør det være mulig å utføre alle applikasjonens operasjoner. Det vil si at en bruker skal ha muligheten til å generere et slideshow kun ved hjelp av tastaturet, selv om det nå til dags er svært vanlig å benytte seg av musepeker.

Verktøylinje

Verktøylinjen skal inneholde de mest brukte elementene fra menyene, slik at disse alltid er lett tilgjengelig, kun ett museklikk unna. Når man holder musepekeren over en knapp på verktøylinjen skal det sprette opp en forklarende tekst, slik at brukeren forstår hvilken funksjon som er knyttet opp mot ikonet, dersom ikonet ikke skulle være selvforklarende.

Statuslinje

På statuslinjen, som skal befinne seg nederst i skjermbildet, skal det kunne vises forskjellig statusinformasjon, som f.eks. navn på åpent slideshow.

Slideshow Center

Fra dette senteret skal det være enkelt for brukeren å åpne slideshow som man tidligere har jobbet med, se på, og eventuelt legge til nye bilder i disse. Lagrede slideshow skal her vises slik at slideshowets første bilde er synlig, sammen med annen relevant informasjon som antall bilder, dato og lignende. Slideshow Center er tenkt plassert langs skjermbildets venstre kant. Bredden på Slideshow Centeret skal kunne justeres, det skal også kunne skjules helt. Figur 3-4 viser et utkast fra oppdragsgiver for hvordan Slideshow Centeret kunne se ut.

Figur 3-4 – Utkast til Slideshow Center fra oppdragsgiver.

Verktøyboks

Verktøyboksen skal deles i to, en del for ”plugin tools”, og en del for ”standard tools”. Førstnevnte skal inneholde en oversikt over alle tilgjengelige *bilderedigeringsplugin*, når et plugin velges fra denne listen skal det også vises et konfigurasjonspanel der brukeren har mulighet til å konfigurere det valgte pluginet. Når brukeren klikker på det valgte bildet, mens et plugin er valgt, skal det skje en hendelse spesifisert av pluginet. Plugin vises her med en kort tekst, og et ikon, dersom pluginet har det. Nye plugin skal automatisk lastes inn i denne listen under oppstart av applikasjonen, dersom de på forhånd er plassert i riktig katalog.
Den andre delen, ”standard tools”, skal inneholde noen standardverktøy som alltid er tilgjengelig for brukeren. Her skal det finnes et verktøy for å velge zoom, forgrunns-farge/bakgrunnsfarge, mulighet for å velge et markeringsverktøy og velge et verktøy som kan flytte valgt bilde. Bredden på verktøyboksen skal kunne justeres, den skal også kunne skjules helt.

Slides-panel

I applikasjonen skal det vises en oversikt over de bilder som tilhører slideshowet det jobbes med. Brukeren skal kunne velge om overgangene mellom bildene også skal vises. Det skal utvikles funksjonalitet for ”drag and drop” slik at brukeren enkelt skal kunne bytte rekkefølge på bildene. Bredden/høyden på denne komponenten skal kunne justeres, den skal også kunne skjules helt.

Valgt bilde

Midt på skjermbildet skal det bildet som er valgt vises i stor størrelse, ved å benytte seg av funksjonaliteten fra verktøyboksen skal man kunne endre på zoom, samt utføre bilderedigeringsoperasjoner vha de tilgjengelige plugin.

Dialoger

Det skal utvikles følgende dialoger.

- Dialog for å opprette nytt slideshow.
- Dialog for å eksportere bilder til nytt slideshow.
- Dialog med oversikt over tilgjengelige plugin.
- Dialog med applikasjonsinnstillinger.
- Dialog for tilpasning er brukergrensesnittet, utseende og farger.
- Dialog for å legge til et nytt bilde.
- Dialog for å legge til en lydeffekt.
- Dialog for å legge til en overgang.
- Dialog med slideshowspesifikke innstillinger.
- Dialog med kortfattet informasjon om applikasjonen.

I tillegg benyttes ferdige standarddialoger for åpning av fil, lagring til fil og valg av farge og font

3.3.3 Fremviserapplikasjonen (Player)

En viktig del av programvaren er muligheten til å distribuere slideshow som en kjørbar fil, slik at mottaker kan spille av dette slideshowet uten å være i besittelse av slideshow-generatoren.

Den kjørbare filen vil bestå av en EXE-fil som bl.a. inneholder funksjonalitet for å spille av et slideshow, denne filen vil ha en komprimert fil inneholdende et slideshow lagt til på slutten, se Figur 3-5. Når fremviserapplikasjonen kjøres vil den søke seg frem til slutten av filen den selv ble startet fra og hente ut et 32-bit tall som forteller hvor langt fra starten av filen datapakken med slideshowet ligger lagret. Denne datapakken vil så kopieres til en temporær katalog må mottagers harddisk, pakkes ut, og slideshowet kjøres.

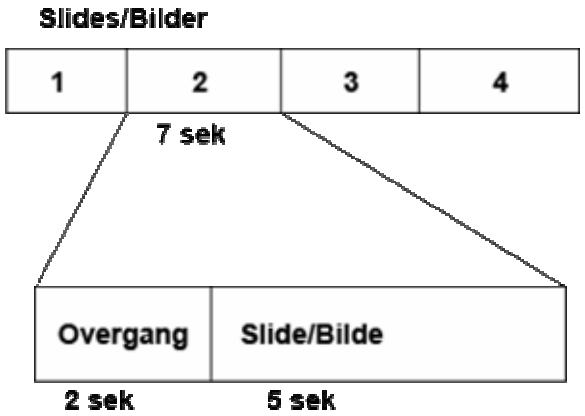
Figur 3-5 – Fremviserapplikasjonens oppbygning.

3.3.4 Komponentspesifikasjon

Fremviser

Både slideshowgenerator- og fremviserapplikasjonen skal ha mulighet til å vise slideshow, som vist i Figur 3-1. Fremviserdelen er delen som skal vise slideshowet. Med andre ord en svært viktig del i programvaren. Slideshowet skal vises slik brukeren har laget det. Flere innstillinger, som for eksempel om titler skal vises eller om slideshowet skal loopes, vil påvirke visningen. Fremviseren må inneholde en enhet som påser å vise rett slide til rett tid og spille av musikk og lyd til rett tid og på rett plass. En slide kan ha en overgang som betyr at overgangen vil spilles av fra forrige slide til gjeldene.

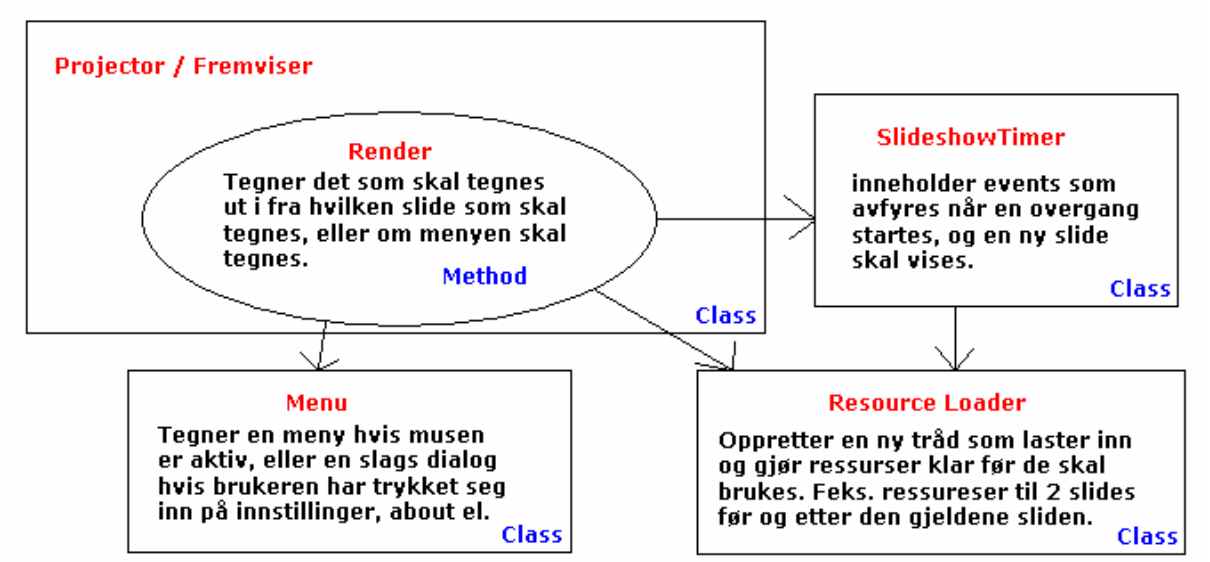
Figur 3-6 viser hvordan en slides lengde er bestemt. Her skal slide nr 2 vises i syv sekunder, der overgangen fra forrige bilde varer i to sekunder, mens selve bilde vises i fem sekunder.



Figur 3-6 – Lengden av en slide.

Fremviseren må benytte seg av tre tråder. Den første tråden tegner opp bildet som vises. Den andre tråden forteller den første tråden når et nytt bilde skal vises eller en lyd skal

spilles av. Den siste tråden håndterer bufring av data. Dvs. den gjør klar bilder en stund før de skal vises. Figur 3-7 viser hvordan de viktigste delene i fremviseren er koblet sammen.



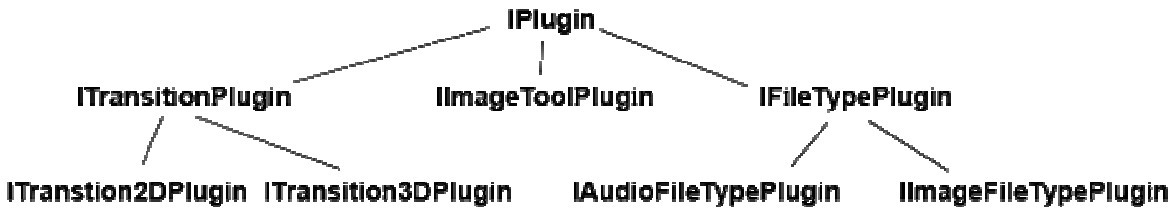
Figur 3-7 – Fremviserens viktigste komponenter.

PluginManager

Siden en stor del av applikasjonens funksjonalitet skal kunne legges til som plugin, vil vi opprette en egen klasse som håndterer plugin. Ved oppstart av applikasjonen vil plugin lastes inn fra en katalog. Under lasting av plugin må PluginManageren sjekke om pluginene implementerer et gyldig interface og om alle pluginets referanser er gyldige. Plugin skal kunne verifiseres, av programvaren dersom de er laget av godkjente utviklere, eller av brukeren. Plugin skal kunne deaktiveres og aktiveres igjen. Informasjon om hvert enkelt plugins tilstand må håndteres, og de nødvendige data lagres i registeret, slik at alle plugin har riktig tilstand neste gang applikasjonen kjøres. PluginManageren må på forespørsel returnere et, eller en samling av plugin. Dersom et plugin har en tilstand som tilsier at det ikke kan benyttes (at det f.eks. er deaktivert eller uverifisert) må ikke pluginet returneres, da slike plugin ikke skal kunne brukes i applikasjonen. Det må isteden være mulig å hente ut informasjon om alle plugin, uavhengig av deres tilstand.

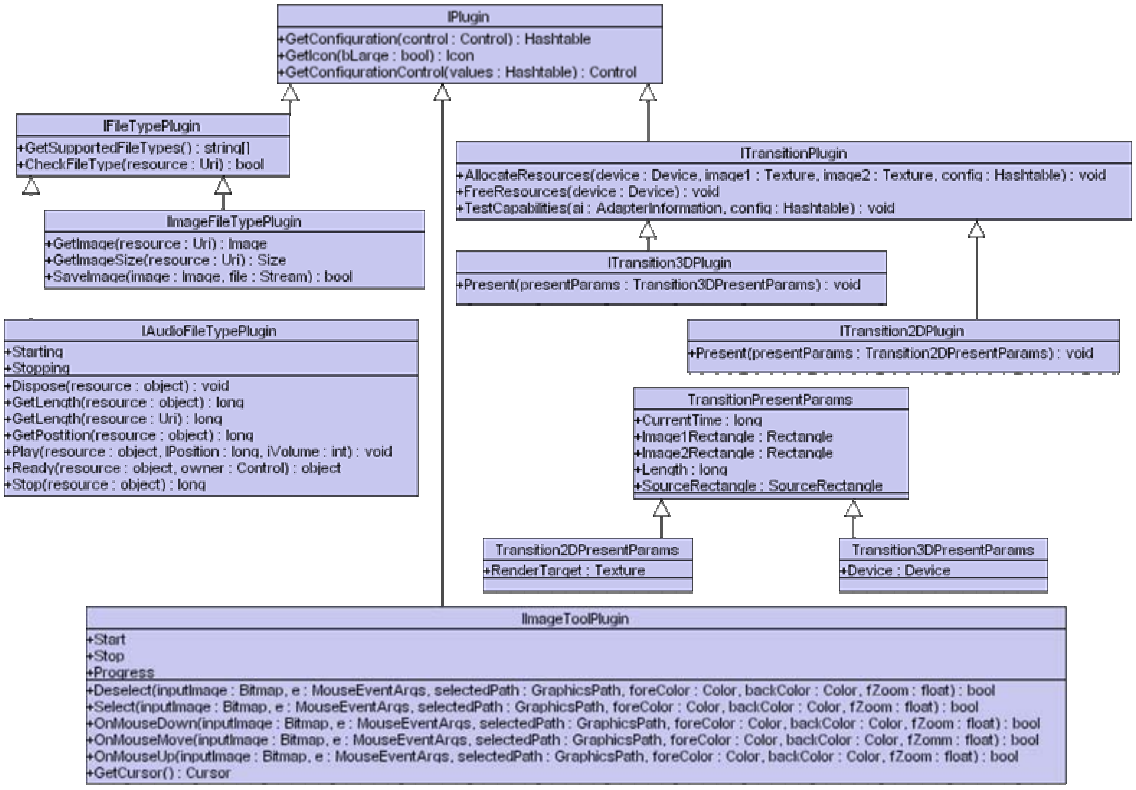
3.4 Pluginstruktur

Som nevnt tidligere vil applikasjonen benytte seg av plugin som dynamisk kan lastes og brukes i applikasjonen. For at applikasjonen skal kjenne igjen pluginene under lasting, og kunne benytte seg av deres funksjonalitet må alle plugin implementere et felles grensesnitt, disse grensesnittene definerer metoder som alle implementerende plugin må inneholde. Denne strukturen gjør det også mulig for andre å lage plugin til applikasjonene. Grensesnittene som alle plugin må implementere (se Figur 3-8) vil i senere tid gjøres fritt tilgjengelig slik at hvem som helst kan utvikle et nye plugin.



Figur 3-8 – Plugingrensesnittenes arvehiarkeri.

For at ikke hvem som helst skal kunne utvikle plugin som ved oppstart av en av applikasjonene vil kunne virke ødeleggende på brukerens pc, vil visse sikkerhetstiltak som for eksempel brukerverifisering av hvert enkelt plugin utvikles. Plugin som ikke kan benyttes på brukerens PC pga at nødvendig programvare ikke er installert, må også markeres 'virker ikke'.



Figur 3-9 – Klassediagram for plugingrensesnitt.

Figur 3-9 viser en mer detaljert oversikt over plugingrensesnittene enn det man ser i Figur 3-8 som kun viser arvehiarkeriet. Det er flere grunner til at grensesnittene er delt inn i et arvehiarkeri, istedenfor å benytte fem uanhengige grensesnitt. Det er en del metoder som alle plugintyper, skal implementere (de skal for eksempel returnere et ikon), disse metodedefinisjonene er plassert i IPlugin som implementeres av de andre grensesnittene. På denne måten unngår vi å skrive samme kode flere steder, med de problemer det kan medføre. I tillegg gjør dette at vi ved de anledninger vi håndterer plugin, som plugintypen

er irrelevant, kan håndtere pluginet som et IPlugin, istedenfor å måtte gjøre dette på fem forskjellige måter uavhengig av plugintype.

Felles for alle plugin er at de må inneholder metoder som:

- Returnerer et ikon for pluginet.
- Returnerer en konfigurasjonskontroll, dersom pluginet har dette.
- Returnerer en oversikt over konfigurasjonskontrollens verdier.

I tillegg blir det, før det eventuelt opprettes en instans av pluginet hentet ut følgende informasjon fra selve DLL-filen:

- Pluginets tittel.
- En beskrivelse av pluginet.
- Pluginets utvikler.
- Eventuelt en verifiseringshash.

3.4.1 Bildefiltypeplugin

Bildefiltypeplugin er plugin som applikasjonene bruker for å åpne og lagre en bildefil i et gitt format som for eksempel jpg eller bmp. Alle bildefiltypeplugin må benytte seg av grensesnittet IImageFileTypePlugin (se Figur 3-9) slik at applikasjonene kan laste de dynamisk.
Et bildefiltypeplugin må inneholde metoder som:

- Sjekker om en fil er i dens format.
- Returnerer en oversikt over støttede filtyper.
- Åpner en fil sitt eget format og returnerer en *bitmap*.
- Returnerer størrelsen på en fil av eget format ut fra filnavn.
- Lagrer data til fil i sitt eget format.

3.4.2 Lydfiltypeplugin

Lydfiltypeplugin er plugin som applikasjonene bruker til å spille av musikk eller lydfiler i et slideshow. Alle lydfiltypeplugin må benytte seg av grensesnittet IAudioFileTypePlugin (se Figur 3-9) slik at applikasjonene kan benytte de dynamisk avhengig av hvilket format en lydfil er lagret i.
Et lydfiltypeplugin må inneholde metoder som:

- Sjekker om en fil er i dens format.
- Returnerer en oversikt over støttede filtyper.
- Åpner en fil i sitt eget format.
- Spiller av lyden.
- Stopper avspillingen
- Returnerer lengden på en lydfil av eget format ut fra filnavn

I tillegg inneholder lydfiltypeplugin to *events* som avfyres når lydavspilling starter og stopper.

3.4.3 Bilderedigeringsplugin

Bilderedigeringsplugin er plugin som i applikasjonen kan brukes til å redigere bilder. Alle bilderedigeringsplugin må implementere grensesnittet IImageToolPlugin (se Figur 3-9) slik at applikasjonene kan laste de dynamisk.
Et bilderedigeringsplugin må inneholde metoder som:

- Spesifiserer hva som skal skje med bildet når verktøyet blir valgt.
- Spesifiserer hva som skal skje med bildet når en museknapp trykkes ned.
- Spesifiserer hva som skal skje med bildet når musepeker beveges.
- Spesifiserer hva som skal skje med bildet når en musknapp slippes opp.
- Returnerer en musepeker som skal brukes under redigeringen.

I tillegg inneholder bilderedigeringsverktøy tre events, som kan benyttes for å vise en progressdialog dersom pluginet bruker lang tid på å utføre sine operasjoner. Start-eventen avfyres før operasjonen starter. Ved passende intervaller under gjennomføringen kan progress-eventen avfyres med en parameter som forteller hvor stor andel av den totale operasjonen som er utført, slik at fremdriftindikatoren i progressdialogen kan oppdateres. Til slutt avfyres stop-eventen når operasjonen er ferdig utført.

3.4.4 Overgangsplugin

Overgangsplugin er plugin som brukes til å vise en overgang mellom to bilder under fermvisning av slideshowet. Alle ovegangsplugin må implementere grensesnittet ITransition2DPlugin eller ITransition3DPlugin (se Figur 3-9) slik at applikasjonene kan laste de dynamisk.
Et overgangsplugin må inneholde metoder som:

- Kjøres før pluginet skal vises og lar det gjøre seg klar til bruk.
- Kjøres etter pluginet er brukt, og frigjør nødvendige ressurser.
- Tester om det er mulig å bruke pluginet på maskinen, og kaster en exception med en feilmelding dersom det ikke er mulig.
- Tegner opp bildet ut fra parametere som hvor langt i overgangen man har kommet.

Overgangspluginene benytter klassen TransitionPresentParams (se Figur 3-9) for overføring av parametere til metoden som tegner opp bildet. Flere av parametrene er primitiver, som normalt ville blitt verdioverført, men siden de legges inn i en egen klasse blir de referanseoverført. Dette gjøres fordi referanseoverføring er raskere enn verdioverføring, og denne metoden vil bli kalt mange ganger under visningen av en overgang, så ytelse er kritisk.

3.5 Datastruktur/filformat

For å tilfredsstille brukernes behov skal det bli laget tre forskjellige måter å lagre et slideshow på. I alle filformatene blir slideshowets egenskaper skrevet til en XML-formatert fil, se vedlegg D for et eksempel på hvordan denne filen kan se ut. Denne filen inneholder som sagt alle innstillinger rundt visning av slideshowet samt data om alle slides, lydfiler og plugin.

3.5.1 Ukomprimert, data som referanser

Dette filformatet, dvs. denne måten å lagre på, er god hvis man kun skal lage et slideshow fort, uten å endre på noen av bildene. Slideshowet vil kun bli lagret til en XML-formatert fil som inneholder absoluttreferanser til alle data i slideshowet. Endrer man på for eksempel et bilde vil originalbildet overskrives.

3.5.2 Ukomprimert, data kopieres

Ved lagring til dette filformatet vil alle bilder, lydfiler og plugin kopieres til en underkatalog med samme navn som selve slideshow XML filen. På denne måten kan brukeren endre bildet uten at det originale bildet går tapt. Dette formatet passer godt hvis brukeren ønsker å endre på bildene.

3.5.3 Komprimert

Det komprimerte filformatet komprimerer alle data til en arkivfil. Dataene blir først strukturert slik som formatet ’ukomprimert, data kopieres’, men i tillegg blir alle data komprimert til en fil. Dette formatet passer godt hvis man vil ta med seg et slideshow til en annen maskin, kun i en fil. Dessuten brukes dette formatet i den kjørbare filen. Når en av applikasjonene åpner en komprimert slideshow fil, må data først pakkes ut til en temporær katalog slik at applikasjonen kan bruke samme funksjonalitet som filformatet ’ukomprimert, data kopiers’.

4 Implementering

4.1 Valg av verktøy

For å gjennomføre et systemutviklingsprosjekt er det behov for forskjellige typer verktøy. Først og fremst trenger man en kompilator, men også programvare for versjonsstyring, figurtegning, generering av hjelpefiler og dokumentasjon kan være nyttig.

4.1.1 Programmeringsmiljø og språk

Dersom vi skulle realisere alle de krav og forventninger oppdragsgiver, og ikke minst prosjektgruppen selv satt med til det ferdige produktet ville det være nødvendig å produsere en ganske omfattende mengde kode i løpet av prosjektet. På grunn av dette ville det være nærliggende å velge et av de programmeringsspråkene vi hadde mest erfaring med fra tidligere, for vårt tilfelle Java eller C++. Java har bl.a. den fordelen at det er plattformuavhengig, men siden vi skal utvikle en applikasjon rettet kun mot Windows-plattformen er dette uten betydning i dette prosjektet. Men på grunn av Javas plattformuavhengighet er mulighetene til å utvikle tiltalende brukergrensesnitt, tilsvarende det en Windows-bruker er kjent med, sterkt begrenset. Vi har satt opp en oversikt over alle aktuelle programmeringsspråk vi kjenner til (se vedlegg E), og vurdert disse språkenes positive og negative sider i forhold til dette spesielle prosjektet.

Vi kom da frem til at *.NET*-plattformen generelt, og språket *C#* spesielt, ville være det beste valget, dette var også noe oppdragsgiver så positivt på. Dette er ikke bare fordi dette fra Microsofts side er annonsert som "fremtiden", men også fordi det syntaktisk minner mye om Java og C# som vi er godt kjent med fra før, .NET-rammerverkets klassebibliotek kan benyttes tilsvarende Javas klassebibliotek, slik at ikke all funksjonalitet nødvendigvis må skrives helt fra grunnen av. I tillegg vil vi i C# ha større muligheter til å produsere en Windows-applikasjon med et profesjonelt Windows-utseende, enn i f.eks. Java. En annen fordel med C# og .NET er at rammeverket håndterer minnestyring, og i tillegg til det rikholdige klassebiblioteket er dette faktorer som gjør at man i dette miljøet kan utvikle en applikasjon langt raskere enn f.eks. med bruk av win32 *API* fra C++. Spesielt siden vi bare er to vil det være en fordel å kunne bruke mer tid på å utvikle selve applikasjonens funksjonalitet og mindre tid på å tenke på minne-allokering/deallokering. Siden vi skulle starte utviklingen i et programmeringsspråk ingen av oss hadde benyttet tidligere var vi forberedt på å måtte bruke noe tid på å gjøre oss kjent med utviklingsmiljøet, programmeringsspråket C# og .NET-rammeverket. Prosessen med å sette seg inn i det nye utviklingsmiljøet gikk overraskende fort, og allerede etter få dager var vi i stand til å starte utviklingen for fullt.

4.1.2 Grafikk API

Siden applikasjonen bl.a. skal være i stand til å vise fascinerende tredimensjonale overganger mellom bildene i slideshowet vil det være nødvendig å benytte en ekstern grafikk API.

Siden det er en tett kobling mellom programmeringsspråk og grafikk API gjorde vi vurderingen av forskjellige grafikk API parallelt med vurderingen av programmeringsspråk (vedlegg E).

Det har blitt besluttet å benytte DirectX 9.0, dette ble et naturlig valgt sammen med C# bl.a. pga *Managed* DirectX som lar oss benytte .NET-rammeverkets minnehåndtering.

4.1.3 Beskrivelse av programvare

Etter litt leting og testing av forskjellige alternativer kom vi frem til følgende verktøy som vi har brukt under gjennomføringen av prosjektet. All programvare på denne listen er enten fritt tilgjengelig eller installert på skolens maskiner.

Adobe Photoshop [2]

Brukt til tegning av enkle figurer og illustrasjoner.

CVSNT [3]

Dette er en *CVS*-server for Windowsplattformen, som vi har brukt til versjonsstyring.

HTML Help Workshop 1.3 [4]

Brukt til generering av applikasjonens hjelpesystem.

Microsoft Paint [5]

Brukt til tegning av figurer.

Microsoft Project [6]

Dette programmet ble brukt til å lage Ganttskjemaer.

Microsoft Visual Studio .NET 2003 [5]

Integrated Developer Environment (IDE), brukt til å skrive og kompilere kildekode, oppbygning av applikasjonens brukergrensesnitt, debugging og generering av XML-dokumentasjon. Det innebygde hjelpesystemet er også flittig benyttet.

Microsoft Windows XP [5]

Operativsystemet som applikasjonene er rettet mot, ble også brukt under utviklingen.

Microsoft Word [7]

Brukt til rapportskriving.

NDoc [8]

Brukt til generering av dokumentasjon (CHM og HTML) ut fra XML-dokumentene Visual Studio genererer ut fra vår kommenterte kildekode.

PuTTy Secure Copy Client [9]

Brukt til backup via SSH protokollen.

TortoiseCVS [10]

CVS-klient for Windows, integreres med Windows utforsker, enkel i bruk.

4.2 Utviklingsmiljø

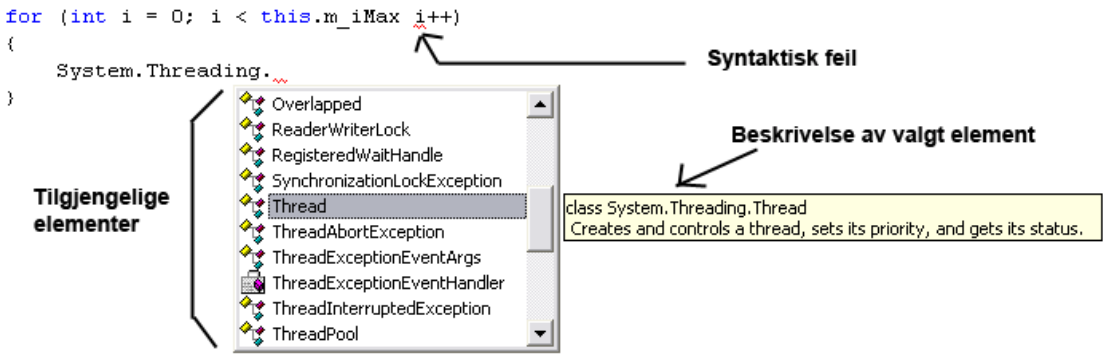
Selv om det er fult mulig å skrive C#-kode i en enkel teksteditor, som for eksempel notepad, og manuelt kompilere kildekoden med kommandolinjekompilatoren csc som er fritt tilgjengelig fra Microsoft [5], er ikke dette den mest effektive arbeidsmåten. I de fleste tilfeller vil man kunne jobbe mye mer effektivt i et IDE[5], der man kan skrive og kompilere fra det samme grafiske brukergrensesnittet, og også gjerne har mer avanserte funksjoner for debugging osv. Visual Studio .NET 2003 ble et naturlig valg for oss, siden det er skreddersydd for utvikling av .NET-applikasjoner og er det er det eneste utviklingsmiljøet med støtte for C# som skolen har lisens på.

4.2.1 Microsoft Visual Studio .NET 2003

Som alle andre IDE har man i Visual Studio mulighet til å skrive og kompilere kildekode, og vise kildekoden med syntax highlighting som gjør koden enklere å lese. I tillegg til alle standardfunksjoner har Visual Studio .NET 2003 en rekke smarte funksjoner som vi har lært å sette pris på i løpet av prosjektet.

Debugging er noe man stort ofte vil benytte en del tid på, og i Visual Studio finnes det en rekke tilgjengelige hjelpemidler som forenkler debuggingprosessen.

Mens man skriver kode får man kontinuerlig hjelp av IntelliSense-systemet. Når man skriver et punktum dukker det opp en nedtrekkliste med alternative metoder/klassenavn/*properties* osv, ved å velge fra denne listen unngår man skrivfeil og man får raskt oversikt over hvilke muligheter som finnes. Syntaktiske feil blir automatisk markert med rød understreking, på samme måte som med stavekontrollen i Microsoft Word (se Figur 4-1), slik at feil kan oppdages og rettes opp før man kompilerer. Dette har gjort at vi har kunnet skrive kode raskere, med mindre feil og brukt mindre tid på å slå opp i dokumentasjonen.



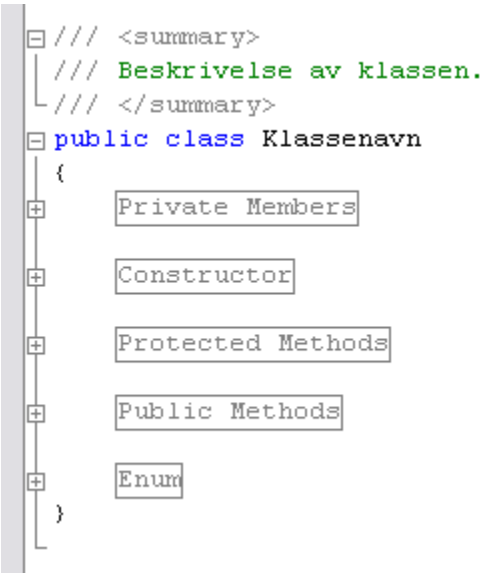
Figur 4-1 – Bruk av IntelliSense i Visual Studio.

[5] Integrated Developer Environment

Visual studio kan integreres med et omfattende hjelpesystem fra MSDN som inneholder dokumentasjon av .NET rammeverkets klassebibliotek og relevante artikler. I tillegg til at hjelpesystemet er søkbart på vanlig måte, finnes en funksjon kalt 'Dynamic Help' som mens man skriver kode kan vise tilgjengelige hjelpeemner for de klasser og metoder man benytter i øyeblikket.

Ved oppbygging av det grafiske brukergrensesnittet kan Visual Studios designer benyttes. Her kan vanlige GUI-komponenter som knapper og tekstbokser velges fra en verktøysamling og plasseres direkte i brukergrensesnittet uten behov for å skrive kode for dette manuelt.

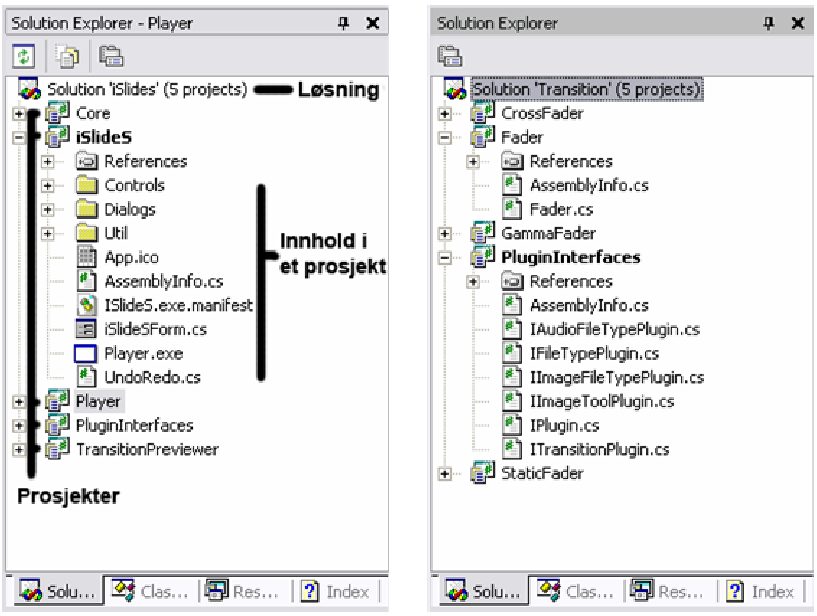
For å strukturere koden kan den deles inn i navngitte regioner. Innholdet i disse regionene kan skjules/vises ved å klikke på -/+ tegnet i margen til venstre for regionen. I tillegg kan innholdet mellom sammenhørende krøllparenteser skjules og vises på samme måte. Vi har gjennom hele prosjektet benyttet regioner til å skille variable, konstruktører, properties og forskjellige typer metoder som vist i Figur 4-2. Dette gjør koden mye mer oversiktlig og lettere og finne frem i, siden man kan finne frem til ønsket sted i koden uten å scrolle gjennom hundrevis av kodelinjer.



Figur 4-2 – Bruk av regioner i Visual Studio.

4.2.2 Filstruktur

Siden det skal utvikles to applikasjoner samt en felles del (kjerne) vil det være hensiktsmessig å dele inn disse tre. I CVS-sammenheng betyr dette at det vil bli tre moduler, som også betyr at hver del legges i en egen katalog og i et eget prosjekt. Deretter legges de forskjellige prosjektene i en *løsning* (*solution*) (Se Figur 4-3, til venstre). Den overordnede løsningen vil bli lagt utenfor CVS filstrukturen. I tillegg til de tre hovedmodulene, lager vi en modul der forskjellige plugin legges i et hierarki som tilsvarer plugin-typene. Hvert plugin legges i sitt eget prosjekt og det vil opprettes en løsning (solution) for hver enkelt plugintype (Se Figur 4-3, til høyre).



Figur 4-3 – Inndeling i ulike prosjekt i Visual Studio.

For hjelpesystemet lages det også en egen modul. Hjelpesystemet katalogiseres slik at det første nivået står for hvilket språk filene under er laget for.

4.2.3 Kodestandard

For å kunne produsere kode som er vedlikeholdbar, oversiktlig og har et homogent utseende ble det besluttet å benytte en kodestandard. Vi har tatt utgangspunkt i Microsoft sine retningslinjer for navngiving [11].

Navngiving av variable

Vi har i tillegg brukt noen variabelprefiks, hentet fra Hungarian Notation [12] og tilpasset vårt behov, som vist i Tabell 4-1.

Tabell 4-1 – Oversikt over benyttede variabelprefiks.

Prefiks	Betydning
b	Bool
c	Char
i	Int
f	Float
d	Decimal
s	String
m_	Medlemsvariabel

Formatering

Alle krøllparenteser er skrevet på egen linje, og all kode mellom et par krøllparenteser rykket inn med et tabulatorhopp. Koden er delt opp i blokker med logisk sammenhørende kode, og blokkene skilt med linjeskift. Koden er, om nødvendig, kommentert før hver blokk.

Kommentering av kildekode

Vi har sett på forskjellige standarder for kommentering av kildekode, felles for disse standardene er at det finnes verktøy som ut fra den dokumenterte kildekoden kan generere dokumentasjon i et eller flere format (CHM/PDF/HTML). De to reelle alternativene vi satt igjen med var Doxygen [13] og Microsofts standard [14] som det er integrert støtte for i Visual Studio. Den store fordelen med Doxygen er at vi her kunne skrevet kommentarer etter *JavaDoc*-standarden som vi er godt kjent med fra før, Doxygen kan også integreres med Visual Studio vha KingsTools [15] og det er her muligheter for å generere flere forskjellige typer dokumentasjon. Den store ulempen med Doxygen er at det har begrenset støtte for C#, det vil si at det ikke finnes støtte for kommentering av C#-spesifikke strukturer som f.eks. properties og events. Microsofts standard med XML-tagger har full støtte for C# og sammen med verktøyet NDoc [8] (se avsnitt 4.1.3 Beskrivelse av programvare) kan det generere dokumentasjon med profesjonelt utseende av samme type som dokumentasjonen av .NET-rammeverkets egne klasser. På grunn av full C#-støtte, integrering med Visual Studio og muligheten for å generere dokumentasjon av samme type som for .NETs eget rammeverk har vi kommet frem til at dette er det beste alternativet.

Her følger et eksempel på hvordan kommentering av en metode kan se ut med de mest brukte taggene.

```
/// <summary>
/// Et kort og lettfattelig sammendrag av hva
/// som er formålet med metoden.
/// </summary>
/// <exception cref="Exception">Beskrivelse av exception.</exception>
/// <param name="sFirstParam">
/// En beskrivelse av første parameter.
/// </param>
/// <param name="sSecondParam">
/// En beskrivelse av andre parameter.
/// </param>
/// <returns>En beskrivelse av verdien metoden returnerer.</returns>
public int MyMethod(string sFirstParam, string sSecondParam)
{
      /// Metodens innhold
}
```

Vi vil også benytte oss av samme standard for kommentering av klasser, variable, properties, events og enums. I tillegg vil vi innlede hver kildekodefil med en ”changelog” som innholder en liste over de endringer som er gjort på filen, denne vil ha følgende format.

```
// <filnavn>
//
// Opprettet:     <dato> - <utvikler>
// Endring:       <dato> - <utvikler> - <beskrivelse av endring>
// Endring:       <dato> - <utvikler> - <beskrivelse av endring>
...
```

4.3 Læring av utviklingsmiljø

Siden vi bestemte oss for å bruke et språk og et utviklingsmiljø vi ikke har noen kjennskap til fra før, brukte vi litt tid i starten for å bli kjent med både språket og utviklingsmiljøet. I den forbindelse har vi bl.a. benyttet tre bøker om C# [16][17][18] og to bøker om DirectX [19][20].

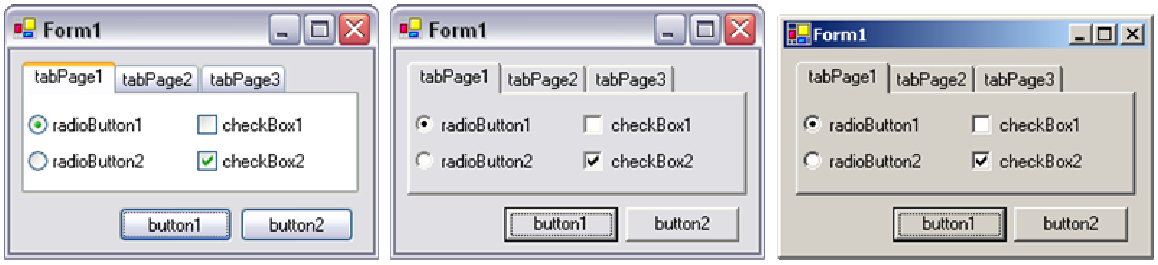
4.4 Slideshowgenerator

Slideshowgeneratoren kan ta imot kommandolinjeparametere, og dersom denne parameteren er filbanen til et gyldig slideshow, vil dette slideshowet bli åpnet i slideshowgeneratoren. Selv om det sjelden er ønskelig å starte applikasjonen manuelt på denne måten har dette andre bruksområder, som for eksempel ved registrering av slideshowfiltypene blant Windows sine kjente filtyper. Ved å dobbeltklikke på en slideshowfil i Windows utforsker vil man da åpne slideshowgeneratoren med det valgte slideshowet åpent og klart for redigering/avspilling.

4.4.1 Brukergrensesnitt og GUI-komponenter

Brukergrensesnittet er en viktig del av en applikasjon, både visuelt og funksjonelt. Siden standardkomponentene som er tilgjengelig gjennom .NET-rammeverket ofte ser kjedelige ut og/eller mangler nødvendig funksjonalitet har vi selv utviklet en del komponenter selv, for å få noe som passer til vårt behov.

Windows XP har en del nyheter når det gjelder visualisering av GUI-komponenter, i forhold til eldre Windows-versjoner. For å kunne vise komponentene med XP-utseende har vi inkludert en manifestfil i selve EXE-filen som gjør at disse komponentene vises med det nye flotte XP-utseende. Figur 4-4 viser forskjellen på GUI med manifestfilen og XP-utseende (til venstre), XP-utseende uten manifestfil (i midten) og klassisk Windows-utseende (Windows 98 og Windows 2000) til høyre.



Figur 4-4 – Forskjell på grafisk brukergrensesnitt med og uten manifestfil, og med klassisk utseende.

ColorComboBox

Flere steder i applikasjonen har brukeren mulighet til å velge en farge. For å velge farge tilbyr .NET-rammeverket en dialog som lar brukeren velge en farge. Denne dialogen har ikke mulighet for å la brukeren velge noen av systemfargene, altså farger som endres ved endring av Windows-utseende. Vi har derfor utviklet en egen ColorComboBox som er en utvidelse av standard ComboBox. Denne komponenten viser en farge sammen med fargens tittel. Vi laget en metode som legger inne alle kjente farger. Dvs. alle tilgjengelige *systemfarger* i tillegg til de 256 websikre fargene. I tillegg har brukeren

mulighet til å velge en egendefinert farge, og vil da bli presentert med standarddialogen fra .NET-rammeverket. Figur 4-5 viser kontrollen, til venstre med standardutseende og til høyre slik den ser ut etter den har blitt klikket på.

Figur 4-5 – ColorComboBox

NumericUpDown

Flere steder i applikasjonen er det behov for å la brukeren spesifisere lengden på et tidsintervall, siden det ikke finnes noen gode standardkomponenter for dette har vi utviklet en. Figur 4-6 viser komponenten som består av et tekstfelt med tilhørende *scrollbar*, og nødvendig funksjonalitet. Brukeren kan endre verdiene ved å skrive inn tall, bruke tastaturets piltaster, eller kontrollens piler, for å inkrementere/dekrementere markert verdi. Formatet er minutter:sekunder:hundredeler, og verdier hentes og settes som antall millisekunder.

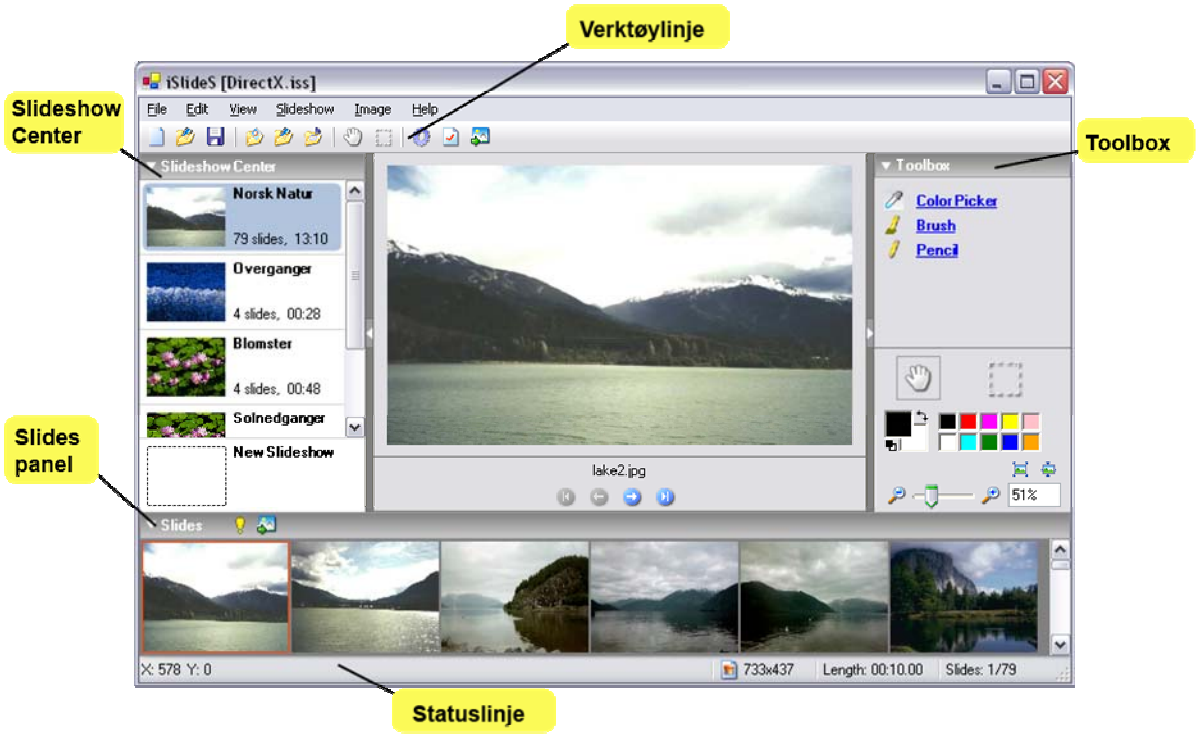
Figur 4-6 – NumericUpDown

Button

På verktøylinjen og andre steder i brukergrensesnittet har vi hatt behov for knapper som viser et ikon istedenfor kun en tekst, siden det ikke fantes noen slik knapp tilgjengelig har vi utviklet en. Knappen har mulighet for å vise et bilde og for å endre alle farger. Knappen kan også brukes som en ”toggle-button”. Denne knappen brukes blant annet på verktøylinjen, på alle pilknapper i dialoger, for å velge standardverktøy i verktøyboksen og for å bla frem og tilbake mellom slideshowets bilder.

Figur 4-7 – Button

Figur 4-7 viser knappens tilstander. Til venstre normal tilstand, i midten nedtrykket og til høyre med musepeker over. I det siste tilfellet vises også en ”tool-tip-tekst” dersom dette er spesifisert for den spesielle knappen.



Figur 4-8 – Applikasjonens hovedelementer.

Slideshow Center

Slideshow Center er en komponent i hovedapplikasjonen som viser en liste over alle slideshow lagret i en spesifisert katalog (standard ”$HOME$/My Documents/My Slideshows”), se Figur 4-8. Listen vil ikke inneholde slideshow som er komprimert. Dette fordi det vil ta for mye prosesseringskraft å gå inn i den komprimerte filen, for å finne fram til de data som trengs for å opprette kontrollen som representerer slideshowet. Nederst finnes det et element som oppretter et nytt slideshow. Dette elementet vises alltid, uansett hvor mange allerede eksisterende slideshow som finnes i listen. Hvert element har en tittel, oversikt over antall slides, total lengde og den første sliden nedskalert til 80x60px. Hvert element er laget som en egen (ikke standard) kontroll som tegnes med oppgitte farger og attributter. Selve slideshowcenterkomponenten er en container som håndterer opprettelse av elementene og deres grafiske framstilling.

Når brukeren trykker på et av elementene må brukeren bekrefte om gjeldene slideshow skal lagres, hvis det er gjort endringer. Deretter lastes det valgte slideshowet inn i minnet og inn i vinduets GUI, hvor den første sliden blir valgt. Hvis brukeren har krysset av for ”Hide Slideshow Center after opening new/existing slideshow” i Preferences dialogen vil Slideshow Centeret bli skjult. Brukeren har også muligheten til å dra bilder/slides fra panelet som inneholder disse, og slippe de på et eksisterende slideshow i Slideshow Centeret eller ’nytt slideshow’-komponenten slik at det dukker opp en eksporterdialog (se ExportSlideshowDialog under avsnitt 4.4.1). I tillegg kan brukeren trykke på et slideshow med høyre musknapp, og fra høyreklikkmenyen enten åpne, slette eller vise en dialog med slideshowets egenskaper. Når et slideshow slettes, vil slideshowet fysisk slettes fra

harddisken. Når slideshowets egenskapsdialog vises og endres, blir slideshowet lagret på nytt, og kontrollens GUI oppdatert.

Slidespanelet

I dette panelet finner man alle bildene i det gjeldende slideshowet, se Figur 4-8. Brukeren kan markere en eller flere slides for enten å flytte, slette eller kopiere. I stedet for å legge til bilder i slideshowet fra menyen er det også mulig å dra bildefiler fra et annet program som for eksempel Explorer og slippe de i slidespanelet for at de skal opprettes som slides i slideshowet. Det er også mulig å dra markerte slides til Slideshow Center for å eksportere slides med eller uten overganger til et nytt eller allerede eksisterende slideshow. Når man velger kun en slide, vil slidens bilde dukke opp midt i hovedvinduet slik at man kan se på bildet og om ønskelig redigere dette. Panelet slidene ligger i har *autoscroll* funksjonalitet, som betyr at når panelet inneholder flere bilder enn det er mulig å vise, dukker det opp en scrollbar. Hvis brukeren har aktivert 'show transitions' vil overgangene vises som en boks mellom hvert bilde. Brukeren kan også aktivere 'show info' som vil vise informasjon om hver enkelt slide, på selve bildet. Ved høyreklikk i panelet vil det dukke opp en høyreklikksmeny der brukeren kan velge å vise overganger, vise info, minimere eller maksimere panelet og legge til ny slide, lyd eller overgang.

Toolbox

Toolboxen er en todelt komponent i hovedapplikasjonen, se Figur 4-8. Den øverste delen, "plugin tools", inneholder en liste over tilgjengelige bilderedigeringsplugin som lastes dynamisk fra en bestemt katalog under oppstart av applikasjonen. I denne listen vises tittel og eventuelt ikon som pluginet selv returnerer. Når brukeren har valgt et plugin vises det et konfigurasjonspanel der brukeren har mulighet til å konfigurere det valgte pluginet. Når brukeren klikker på det aktive bildet, mens et plugin er valgt, vil det skje en hendelse spesifisert av pluginet, på denne måten kan verktøyene fra "plugin tools" brukes til å redigere bildet.

Den nederste delen, "standard tools" inneholder noen standardverktøy som alltid er tilgjengelig for brukeren. Her ligger en kontroll for valg av zoom (i prosent), der brukeren kan taste inn ønsket zoom i en tekstboks, velge zoom (1 % - 1600 %) ved å dra på en "slider" (der 100 % er midt på "slideren"), klikke på to knapper for å zoome inn/ut til nærmeste "vanlige tall", samt knapper for å velge full størrelse eller beste tilpasning til vinduet. Et annet (egenprodusert) standardverktøy er en kontroll for valg av farge (forgrunn- og bakgrunnsfarge), her lagres også de åtte sist brukte fargene, slik at brukeren lett kan finne tilbake til tidligere brukte farger. Her ligger også en liten knapp for å bytte om på forgrunns- og bakgrunnsfarge, samt en knapp for å velge standardfarger (svart forgrunns- og hvit bakgrunnsfarge).

Under standardverktøy finnes det et verktøy for å gjøre en, rektangulær, markering i bildet, slik at denne senere kan behandles uavhengig av resten av bildet. Det markerte området vil også være tilgjengelig for bilderedigeringsverktøy, slik at disse kan utføre operasjoner på det markerte området. Det er lagt til rette for at applikasjonen senere kan utvides med andre typer markeringsverktøy, som ikke nødvendigvis må være rektangulære. Ved siden av markeringsverktøyet er det en knapp for å velge en hånd,

denne hånden kan brukes til å flytte rundt på bildet dersom man har zoomet inn så mye at hele bildet ikke kan vises samtidig. Bredden på hele toolboxen kan justeres ved å dra i venstre kant av den, eller den kan skjules helt ved å dobbeltklikke på det trekantede symbolet midt på kontrollens vestre kant.

Toolboxen har også en høyreklikksmeny som kan brukes for å skjule/vises toolboxen, laste inn alle pluginverktøy på nytt, og vises egenskaper for et pluginverktøy.
De knappene som brukes for hånden og markeringsverktøyet er den samme komponenten som også brukes andre steder i applikasjonen, som for eksempel på verktøylinjen.

Verktøylinje

En verktøylinje er en svært vanlig komponent i en Windows applikasjon. .NET tilbyr en ferdig kontroll, med den mest nødvendige funksjonaliteten. Vi utviklet heller en ny kontroll som fungerer på samme måte, men ser mer moderne ut (se Figur 4-8). Bakgrunnen består av en fargeovergang fra fargen Windows-Menu til Windows-Control. Knappene på verktøylinjen er av samme type som knappene som styrer Zoom, og flere andre knapper rundt om i applikasjonen. De inneholder et bilde som tilsvarer ikonene man finner i menyene. Knappene har HotTracking funksjonalitet som gjør at de markeres når musen føres over. Det spretter også opp *tooltip* som forklarer knappens funksjon. Når brukeren trykker på en av knappene, skal den utføre akkurat det samme som tilsvarende menyelement eller annen funksjon i applikasjonen gjør.

Knapper på verktøylinjen

- New/Ny
- Open/Åpne
- Save/Lagre
- Add Image/Legg til bilde
- Add Sound/Legg til lyd
- Add Transition/Legg til overgang
- Move Tool/Flytte verktøy
- Selection Tool/Markerings verktøy
- Preferences/Innstillinger
- Slideshow Properties/Egenskaper
- Run/Kjør Slideshow

I samråd med oppdragsgiver ble det besluttet at vi ikke skulle bruke tid på å tegne ikoner selv, men isteden midlertidig låne passende ikoner fra andre applikasjoner. Alle ikonene på verktøylinjen må derfor byttes ut før applikasjonen kan benyttes i kommersiell sammenheng.

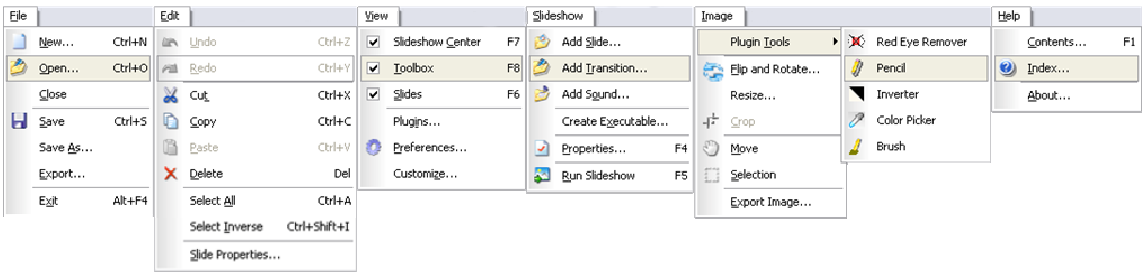
Statuslinje

Statuslinjen ligger nederst i hovedvinduet og benyttes til flere formål. Det viktigste formålet er å gi brukeren en liten beskrivelse av hvilken slide som er valgt (se Figur 4-8). Kontrollen baserer seg på .NET rammeverkets kontroll, men er bygget opp slik at det er enklere å bytte ut tekst, og sette andre verdier i sammenheng med statuslinjen.

Menylinje

Siden standardmenyene som Visual Studio tilbyr både er kjedelige å se på, og mangler mulighet for å vise ikon sammen med menyelement, har vi utviklet et eget menyelement som er en utvidelse av et standard menyelement. Med menyelementene våre vises menyene med en *gradient* pyntelinje til venstre, fargen på denne blir regnet ut fra en av systemfargene slik at den vil tilpasse seg forskjellige fargeoppsett. Denne fargen blir regnet ut på nytt dersom brukeren skulle endre fargeoppsett mens applikasjonen kjøres, fargene på alle menyelementene blir deretter oppdatert rekursivt, slik at også alle undermenyer oppdateres. Det er mulig å legge til et ikon eller bilde til hvert meny-element, dette tegnes opp med skygge når menyelementet er valgt, slik at vi får en tredimensjonal virkning. Menyelementene kan også markeres som avkryssbare, slik at de kan vises med en boks som kan krysses av, dette er heller ikke noe standard menyelement har full støtte for. Ved å utvikle kontrollen selv på denne måten, får vi i ikke bare mulighet til å legge til ekstra funksjonalitet, men vi får også full kontroll over utseende og farger. De menyvalgene som brukeren kan antas å ville bruke hyppig har vi utstyrt med tastatursnarveier, slik at man brukeren for eksempel kan trykk ctrl+c istedenfor å velge ”copy” på ”edit”-menyen, disse tastatursnarveiene vises i høyre kant av menyene.

I tillegg til selve menyelementene har vi også utviklet menyelementene som vises på menylinjen som fil, edit, view osv. Disse har på samme måte som de andre menyelementene HotTracking funksjonalitet som gjør at de markeres når musen føres over dem. Figur 4-9 viser menyelementenes utseende og alle undermenyene på applikasjonens hovedmeny. Det samme menyelementet benyttes også på høyre-klikksmenyer i hele applikasjonen.



Figur 4-9 – Applikasjonens hovedmeny.

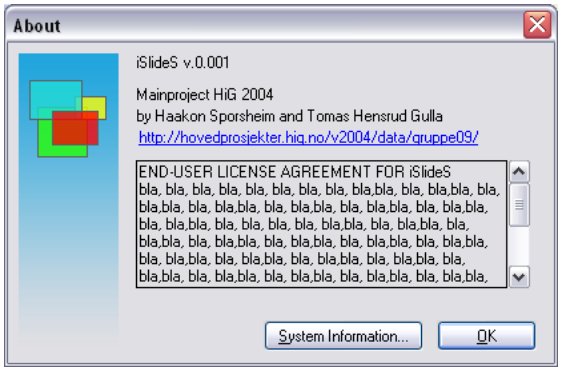
I likhet med verktøylinjen er alle ikonene på menyen lånt fra andre applikasjoner, og må derfor byttes ut før applikasjonen kan benyttes i kommersiell sammenheng.

Dialoger

Vi har utviklet følgende dialoger, for å ta imot informasjon fra brukeren. I tillegg benyttes standard dialoger for lagring/åpning av fil og valg av farger og fonter. Felles for alle dialoger er at de har en OK-knapp for å godta endringer og en Cancel-knapp for å lukke dialogen uten å lagre endringer, i tillegg har alle dialogene (med unntak av AboutDialogen) en Hjelp-knapp for å åpne applikasjonens hjelpesystem med informasjon om aktiv dialog.

AboutDialog

Dialog, med utseende som vist i Figur 4-10, som viser kortfattet informasjon om applikasjonen, kontaktinformasjon og lisensavtale. Inneholder en klikkbar link som åpner applikasjonens nettside i en nettleser. Knappen merket ”System Information” kjører filen MSINFO32.EXE som åpner en standard Windowsdialog med systeminformasjon, på samme måte som man er vant til i andre applikasjoner, f.eks Microsoft Word.

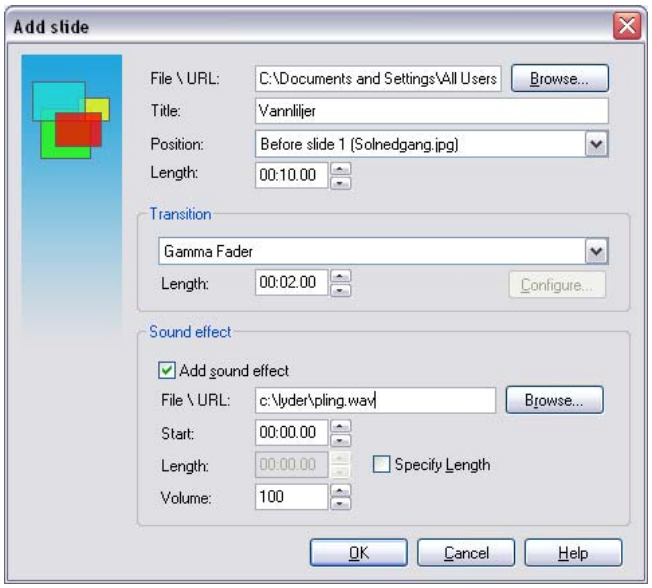


Figur 4-10 – AboutDialog.

Denne dialogen er tilgjengelig ved å velge About i Help-menyen.

AddSlideDialog

Dialog, med utseende som vist i Figur 4-11, for å legge til et nytt bilde i slideshowet. Den lar brukeren velge hvilket bilde som skal settes inn, hvilken tittel den nye sliden skal få, hvor i slideshowet sliden skal settes inn og hvor lenge den skal vises. Det er samtidig mulig å velge overgangen som skal vises før bildet, og å legge til en lydeffekt til sliden. Dersom brukeren forsøker å legge til en ugyldig bilde- eller lyfil vil brukeren få beskjed om det. Når brukeren trykker OK, blir det opprettet et nytt Slide-objekt, og dersom brukeren har valgt det, tilhørende Transition- og Sound-objekt. Dette Slide-objektet blir så lagt til applikasjonens SlideshowInfo-objekt på spesifisert sted. Sammenhengen mellom Slide, Transition, Sound og SlideshowInfo vises i Figur 3-3.

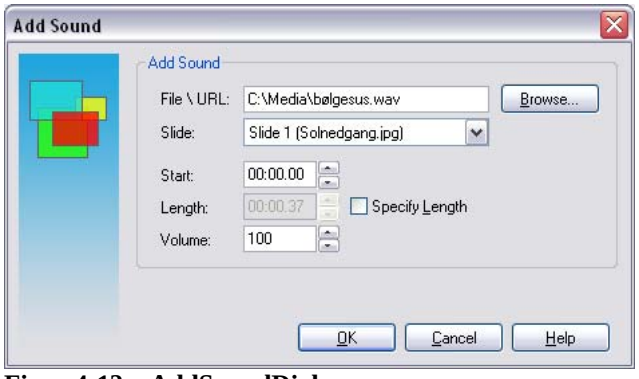


Figur 4-11 – AddSlideDialog.

Denne dialogen er tilgjengelig ved å velge Add Slide i Slideshow-menyen og fra verktøylinjen.

AddSoundDialog

Dialog, med utseende som vist i Figur 4-12, for å legge til en lydeffekt til en slide. Den lar brukeren velge hvilken lydfil som skal legges til, om lydfilen skal legges til valgt slide (standard) eller en annen slide. Det er også mulig å spesifisere hvor lenge etter bildet lyden skal starte, hvor lenge lyden skal spilles (maksimalt lengden på lydfilen) og med hvilket volum lyden skal spilles. Når brukeren trykker OK blir det opprettet et nytt Sound-objekt med de spesifiserte parametere, og dette blir lagt til spesifisert slide i slideshowet.

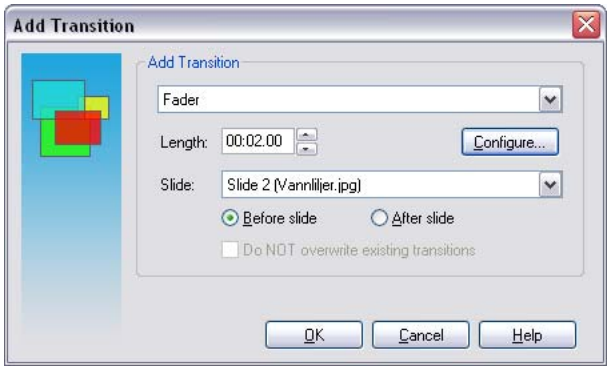


Figur 4-12 – AddSoundDialog.

Denne dialogen er tilgjengelig ved å velge Add Sound i Slideshow-menyen og fra verktøylinjen.

AddTransitionDialog

Dialog, med utseende som vist i Figur 4-13, for å legge til en overgang til en slide. Overgangen velges fra en liste over tilgjengelige overganger, og dersom overgangen har støtte for dette kan også overgangen konfigureres. Hvilken slide overgangen skal assosieres med velges fra en liste over slideshowets slider, og brukeren kan også velge om overgangen skal vises før (standard) eller etter sliden. Fra listen over slider er det også mulig å velge at overgangen skal legges til alle markerte slider, og om dette alternativet velges vil det også være mulig å velge at eksisterende overgaver ikke skal overskrives. Når brukeren trykker OK blir det opprettet et nytt Transition-objekt med de spesifiserte parametere og dette blir lagt til spesifisert slide i slideshowet. Alle Transition-objektet har referanse til et overgangsplugin, disse overgangspluginene må til forskjell fra de andre plugintypene klones ved bruk, slik at hvert Transition-objekt har sitt eget plugin-objekt. Dette er fordi under avspilling av slideshowet vil flere overganger av samme type, som kan ha forskjellige innstillinger, måtte ligge i minnet samtidig.

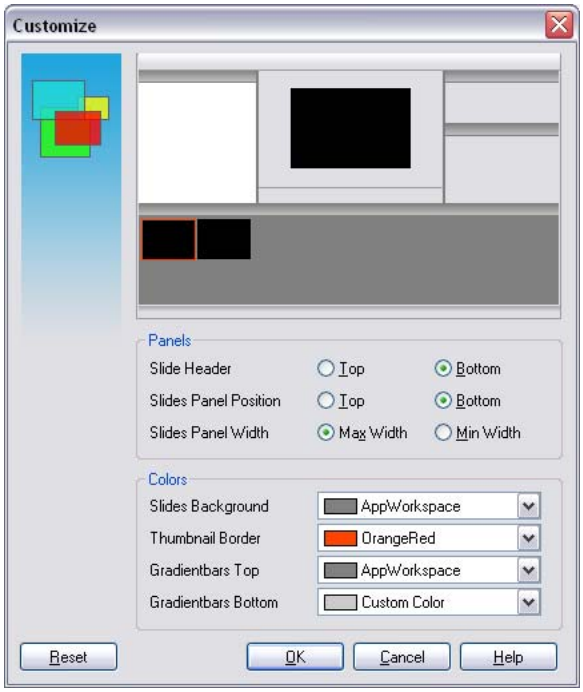


Figur 4-13 – AddTransitionDialog

Denne dialogen er tilgjengelig ved å velge Add Transition i Slideshow-menyen og fra verktøylinjen.

CustomizeDialog

Dialog, med utseende som vist i Figur 4-14, for tilpasning av brukergrensesnittet, utseende og farger. For avanserte brukere som bruker en del tid på å jobbe med applikasjonen kan det være nyttig å kunne bestemme hvor de forskjellige komponentene i brukergrensesnittet skal vises og hvilke farger de skal ha, dette kan enkelt gjøres vha denne dialogen. I øverste del av dialogen vises et forhåndsvisningspanel, dette er en UserControl vi har laget selv, og selv tegner opp ut fra de valg som gjøres i dialogen. Hvis man vil sette alle verdier tilbake til standard, finnes det en reset knapp. Vedlegg F viser noen av oppsettene som er mulig ved hjelp av denne dialogen.

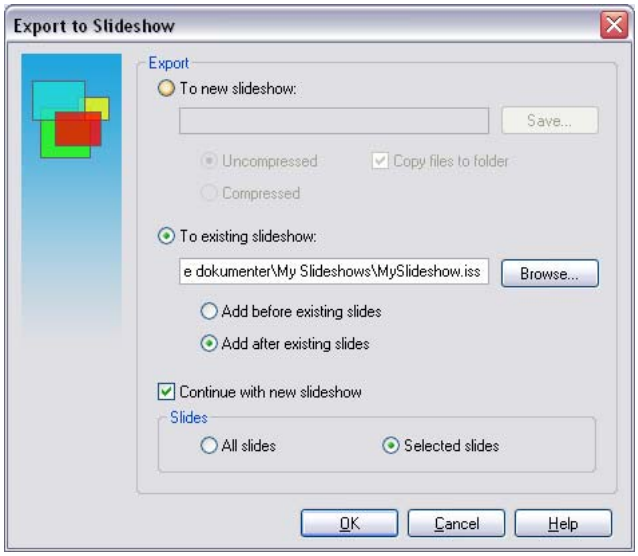


Figur 4-14 – CustomizeDialog

Denne dialogen er tilgjengelig ved å velge Customize i View-menyen.

ExportSlideshowDialog

Dialog, med utseende som vist i Figur 4-15, for å eksportere bilder til et nytt slideshow. Brukeren kan velge å eksportere bildene til et annet, tidligere lagret slideshow, eller til et helt nytt slideshow. Dersom bildene eksporteres til et tidligere lagret slideshow, kan brukeren bestemme om de nye bildene skal legges til på starten eller slutten av det gamle slideshowet. Dersom bildene eksporteres til et nytt slideshow kan brukeren velge hvilket filformat det nye slideshowet skal lagres i. Brukeren kan eksportere alle bildene i det åpne slideshowet, eller kun de bildene som er markert. Brukeren kan til slutt velge om han vil fortsette å jobbe med det slideshowet han jobbet med fra før, eller om han vil fortsette med det nye slideshowet (det som bildene ble eksportert til).



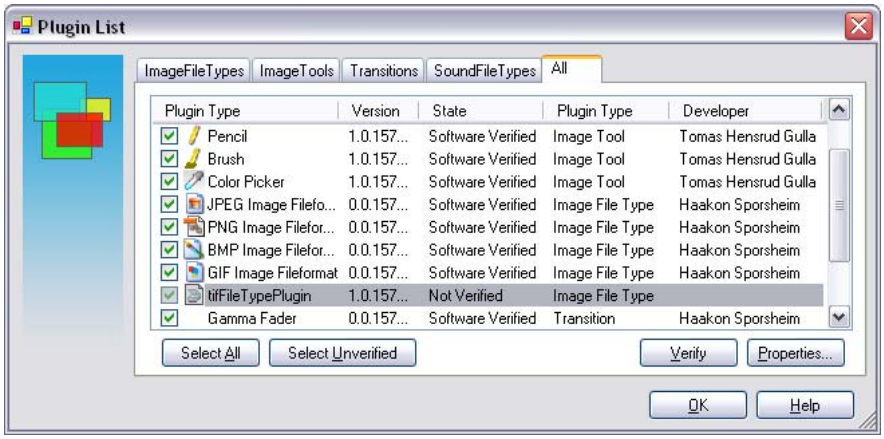
Figur 4-15 – ExportSlideshowDialog

Denne dialogen er tilgjengelig ved å velge Export i File-menyen eller ved å dra to, eller flere, bilder fra et åpent slideshow til et slideshow som ligger i SlideshowCenter.

PluginListDialog

Dialog, med utseende som vist i Figur 4-16, med oversikt over tilgjengelige plugin som hentes fra PluginManager (beskrevet under avsnitt 3.3.4). Dialogen består av fem faner (tabs). De første fire fanene viser oversikt over tilgjengelige plugin kategorisert etter de fire hovedtypene plugin; bildefiltypeplugin, lydfiltypeplugin, overgangsplugin og bilde-redigeringsplugin. Den femte og siste fanen viser alle tilgjengelige plugin samlet i én liste. Siden alle plugin da vises på to lister, betyr dette at hver gang et plugin endres i en liste må det også oppdateres i den andre. Alle listene kan sorteres på en hvilken som helst kolonne, ved å klikke på den kolonneoverskriften det ønskes sortert etter. Klikk én gang for å sortere alfabetisk, og en gang til for å sortere omvendt alfabetisk. Funksjonaliteten for sortering er ikke innebygd i ListView-kontrollen som brukes her, klassen som tar seg av denne sorteringen (ListViewColumnSorter) har vi ikke kodet selv, men hentet fra Microsoft Support [21]. Om man klikker på knappen merket ”properties” dukker det opp en PluginPropertiesDialog med all tilgjengelig informasjon om det pluginet som er markert. I tillegg til å vise en oversikt over tilgjengelige plugin kan denne dialogen også brukes til å endre pluginenes tilstand. For å deaktivere et plugin kan man fjerne avkrysningen til venstre for pluginets navn, for å aktivere pluginet igjen krysser man det av igjen. Det finnes også en knapp for å verifisere/avverifisere markerte plugin. Dersom alle markerte plugin er verifiserte vil knappen ha tittelen ”Unverify” og brukes til å avverifisere markerte plugin, og dersom ett eller flere av de markerte plugin er uverifiserte vil knappen ha tittelen ”Verify” og brukes til å verifisere markerte plugin. Når brukeren verifiserer et plugin vil han få spørsmål om pluginet skal verifiseres for all fremtid (opplysninger om dette vil da lagres i registeret og pluginet vil fremdeles være verifisert neste gang applikasjonen startes) eller om det kun skal verifiseres for denne ene gangen, slik at pluginet ikke er verifisert neste gang applikasjonen startes. Dersom et plugin er uverifisert, eller deaktivert når dialogen åpnes vil dette si at det ikke er opprettet en instans av pluginet, og dermed er det en del informasjon om pluginet som ikke er

tilgjengelig, for eksempel ikon, og støttede filtyper for filtypeplugin. Dersom et sånt plugin verifiseres/aktiveres vil det opprettes en instans av pluginet, og all tilgjengelig informasjon vil automatisk bli oppdatert i dialogen.

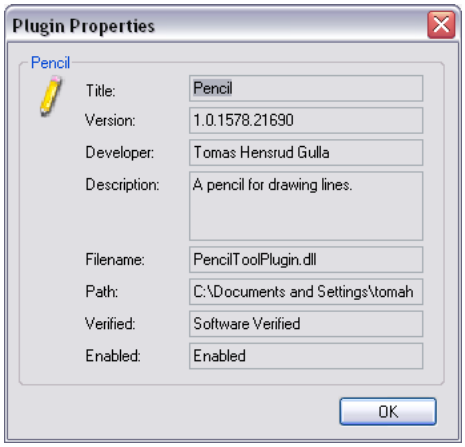


Figur 4-16 – PluginListDialog

Denne dialogen er tilgjengelig ved å velge Plugins i View-menyen.

PluginPropertiesDialog

Dialog, med utseende som vist i Figur 4-17, som viser informasjon om et bestemt plugin. Viser informasjon som ikon, tittel, versjon, utvikler(e), beskrivelse, filnavn (filen pluginet er lastet fra), verifiseringsgrad og om det aktivt /deaktivert for alle typer plugin, der dette er tilgjengelig. I tillegg vises en oversikt over støttede filtyper for *filtypeplugin*.



Figur 4-17 – PluginPropertiesDialog

Denne dialogen er tilgjengelig fra PluginListDialog der man kan bruke Properties-knappen eller velge Properties fra høyreklikkmenyen. I tillegg vises dialogen med informasjon om et bilderedigeringsverktøy ved å høyreklikke på dette og velge Properties fra Toolboxen.

PreferencesDialog

Dialog, med utseende som vist i Figur 4-18, som inneholder applikasjonsinnstillinger. Lar brukeren velge om det skal vises en enkel veiviser (wizard), istedenfor den mer avanserte dialogen, når brukeren velger å opprette et nytt slideshow, om slideshowcenter (oversikt over tilgjengelige slideshow) skal vises ved oppstart, om slideshowcenter skal skjules når et slideshow blir valgt, foldere for plugin og slideshow, samt hvordan ikke-verifiserte plugin skal behandles. Hvis man velger ”Add Slide” fra menyen til venstre vil man få mulighet til å velge om nye slider automatisk skal legges sist eller etter markert slide, standardlengde på slider og overganger, om det skal vises en dialog når brukeren drar ett bilde inn i applikasjonen og slipper det og om filnavn eller en annen tekst skal brukes som standardtittel på nye slider.

Alle panelene i denne dialogen (foreløpig kun to) er skilt ut i egne filer, og må implementere grensesnittet IPrefPanel som spesifiserer tre metoder. Disse metodene er LoadSettings, SaveSettings og ResetSettings. Når dialogen lastes vil den kalle metoden LoadSettings i alle sine panel, som inneholder funksjonalitet for å hente sine egne innstillinger fra det globale preferencesobjektet og oppdatere sitt eget brukergrensesnitt. På samme måte kalles SaveSettings ved trykk på OK, og ResetSettings (som henter tilbake standardverdier) når brukeren trykker reset.

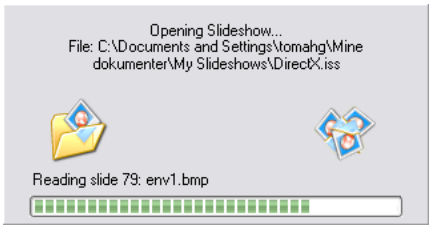
Det globale preferencesobjektet, en singletonklasse vi selv har utviklet, håndterer alle brukerinnstillinger og sørger for at disse blir lagret til, og hentet ut fra registeret. Dette objektet inneholder også standardverdier for alle brukerinnstillingene. Dette objektet håndterer opprettelse av registerverdiene, dersom de ikke eksisterer fra før, slik at dette ikke er nødvendig å gjøre i en installasjonsapplikasjon.

Figur 4-18 – PreferencesDialog

Denne dialogen er tilgjengelig ved å velge Preferences i View-menyen og fra verktøylinjen.

ProgressDialog

Dialog, med utseende som i Figur 4-19, som vises mens applikasjonen utfører tidkrevende operasjoner. Dialogen kjøres i en egen tråd, slik at den blir oppdatert mens maskinen utfører den krevende operasjonen. Dialogen inneholder en tekst som beskriver handlingen som utføres og en animasjon. Denne animasjonen vises slik at brukeren skal se at applikasjonen jobber, i tillegg vil brukeren som oftest oppfatte ventetiden som kortere dersom det vises en slik animasjon [22]. Det vises en tekst som kan brukes til å vise fortløpende informasjon om hva som skjer, for eksempel filnavn på alle bildefiler ved åpning av et slideshow. Nederst vises en fremdriftsindikator (*progressbar*) som angir hvor stor del av den totale operasjonen som til en hver tid er utført.



Figur 4-19 – ProgressDialog.

NewSlideshowDialog

Dialog for å opprette et nytt slideshow, enten fra grunnen av eller med utgangspunkt i et tidligere laget slideshow. Det er når som helst mulig å bytte til wizarden, ved å trykke på knappen ”Wizard”, og fortsette å spesifisere parametere for det nye slideshowet der. Når brukeren veksler mellom disse to dialogene vil alle innstillinger overføres mellom de to dialogene, slik at ingen innstillinger går tapt.

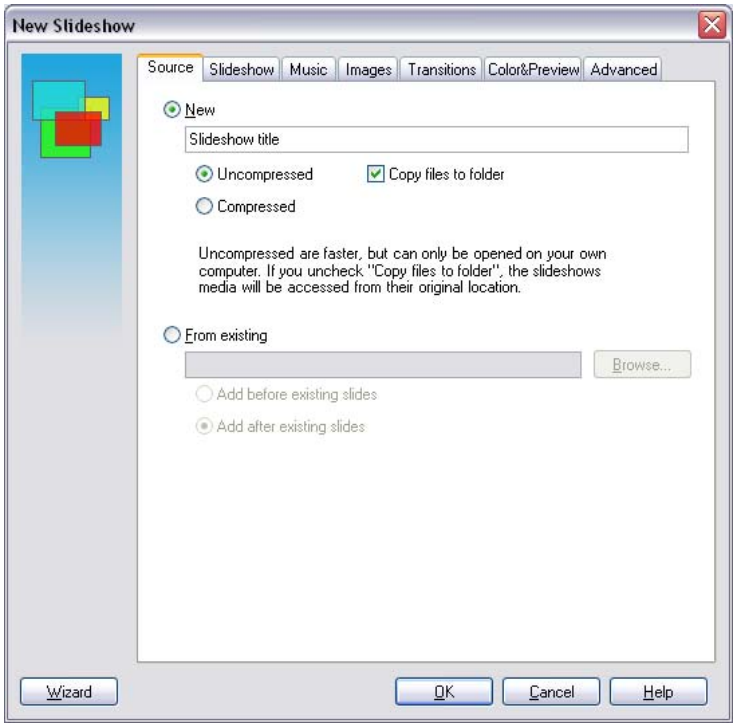
Dialogens innstillinger er gruppert under syv faner (tabs). Siden dialogen inneholder mange GUI-komponenter vil den ta lang tid å laste. Dersom alle komponentene skulle lastes før dialogen vises ville dette kunne ta flere sekunder, for at brukeren skal slippe å vente så lenge lastes kun fanen som skal vises først, før selve dialogen vises. Resten av fanene lastes kun inn dersom brukeren aktivt velger å vise disse fanene. Dette problemet kunne alternativt vært løst ved at første fane ble lastet før dialogen ble vist, og at de resterende fanene deretter ble lastet inn i en egen tråd i bakgrunnen. Dette ble forsøkt, men løsningen ble forkastet siden enkelte av våre egenproduserte GUI-komponenter da begynte å oppføre seg uregelmessig og ukorrekt. Løsningen med lasting i en egen tråd ville vært å foretrekke i en profesjonell applikasjon, men vi fant ut at vi ikke hadde tid til å bruke mer tid på dette problemet. Den nåværende løsningen viser dialogen like rask som trådløsningen (oppstartstid redusert fra 3,0 til 0,3 sekunder på vår testmaskin), men vår løsning vil medføre en liten ventetid før de andre fanene kan vises for første gang.

Innholdet i alle fanene er skilt ut i egne filer som implementerer grensesnittet IPropPanel, som spesifiserer to metoder. En metode for å lagre egne innstillinger i medsendt SlideshowInfo-objekt og en metode for å oppdatere eget brukergrensesnitt i forhold til medsendt SlideshowInfo-objekt. Når brukeren trykker OK lagrer alle fanene sine innstillinger til et nytt SlideshowInfo-objekt. Deretter blir alle slider lagt til med sine lydeffekter og overganger blir generert, siden dette må utføres i en operasjon som krever

informasjon fra flere faner, må dette utføres i selve dialogen. Til slutt blir dette SlideshowInfo-objektet satt som applikasjonens globale SlideshowInfo-objekt og brukergrensesnittet oppdatert.

Source

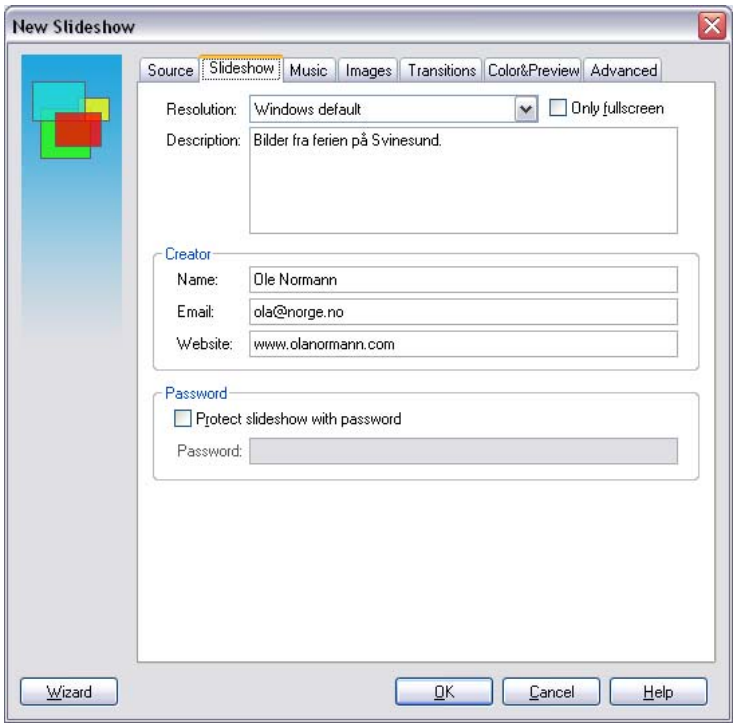
Denne fanen, se Figur 4-20, lar brukeren velge om det skal opprettes et nytt slideshow, eller om han vil fortsette på et slideshow som er opprettet tidligere. Dersom brukeren velger å opprette et nytt slideshow må brukeren også velge hvilket filformat som skal benyttes. Når musepeker holdes over de forskjellige filformatalternativene vil det vises en forklarende hjelpetekst. Dersom brukeren isteden velger å fortsette med et tidligere lagret slideshow kan han velge om de nye slidene skal legges til før, eller etter, de eksisterende.



Figur 4-20 – NewSlideshowDialog, source tab.

Slideshow

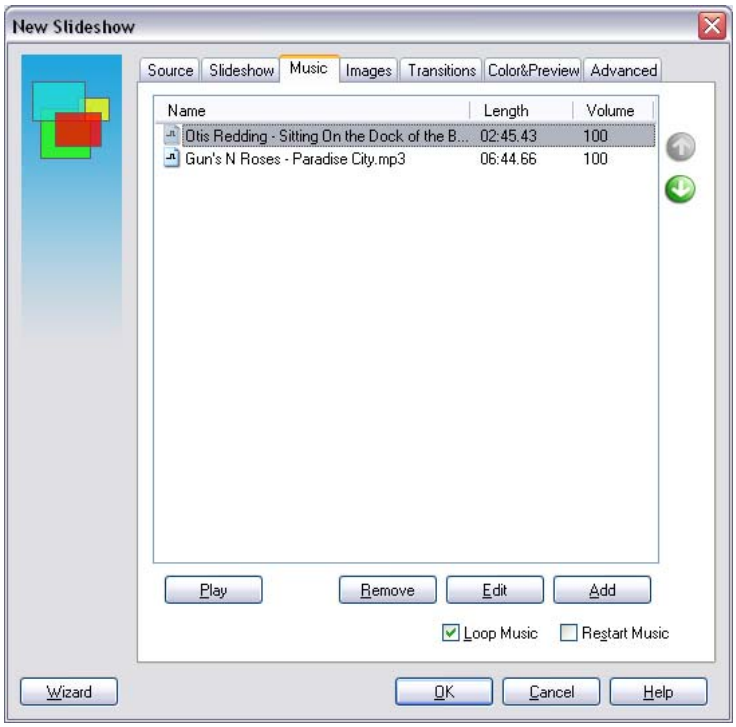
Denne fanen, se Figur 4-21, lar brukeren velge hvilken oppløsning slideshowet skal spilles av i, og om det skal være mulig å spille det av i et vindu eller kun i fullskjerm. Brukeren kan skrive inn en beskrivelse av slideshowet, samt informasjon om eget navn, e-postadresse og webadresse, dette er informasjon som også vil være tilgjengelig for mottagere av en kjørbar fil. Til slutt er det også mulig å passordbeskytte slideshowet.



Figur 4-21 – NewSlideshowDialog, slideshow tab.

Music

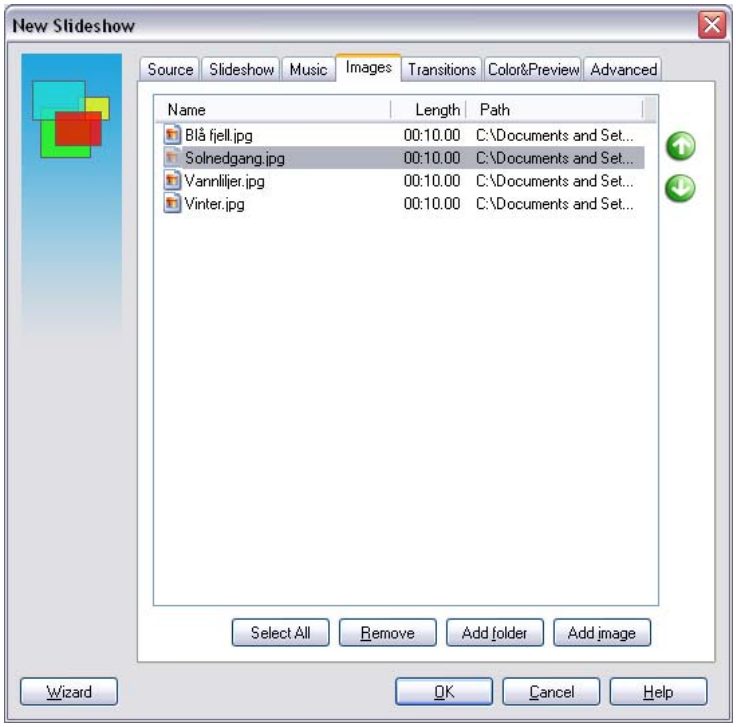
Denne fanen, se Figur 4-22, lar brukeren velge hvilke musikkfiler som skal være med i det nye slideshowet, brukeren kan legge til enkeltfiler eller en hel katalog med lydfiler. For å endre rekkefølgen på lydfilene kan brukeren benytte de to knappene i høyre kant av dialogen for å flytte markert(e) lydfil(er) opp eller ned. Lydfiler kan også spilles av. Til slutt kan brukeren velge om de valgte lydfilene skal loopes, dvs. at dersom lengden av lydfilene er kortere enn slideshowets lengde vil den første lydfilen starte på nytt når den siste er ferdig. I tillegg kan brukeren velge om lyden alltid skal starte på nytt fra starten hver gang slideshowet starter på nytt.



Figur 4-22 – NewSlideshowDialog, music tab.

Images

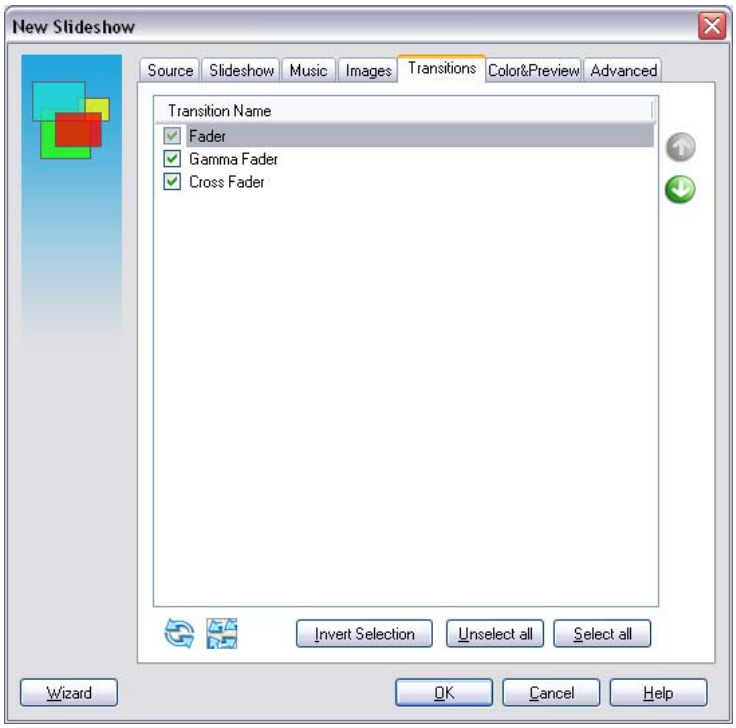
Denne fanen, se Figur 4-23, lar brukeren velge hvilke bilder som skal være med i det nye slideshowet, brukeren kan legge til enkeltbilder eller en hel katalog med bilder. For å endre rekkefølgen på bildene kan brukeren benytte de to knappene i høyre kant av dialogen for å flytte markert(e) bilde(r) opp eller ned.



Figur 4-23 – NewSlideshowDialog, images tab.

Transitions

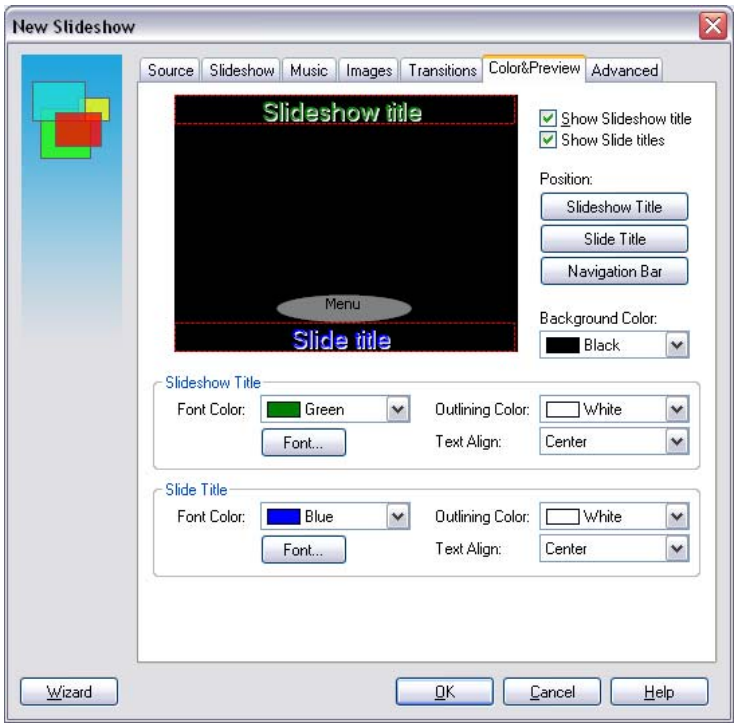
Denne fanen, se Figur 4-24, lar brukeren velge hvilke overganger som skal brukes i det nye slideshowet. Her krysser brukeren av de overganger han vil bruke, og velger om disse skal benyttes sekvensielt eller i tilfeldig rekkefølge. Knappene i høyre kant kan brukes til å endre rekkefølge på overgangene, men dette vil kun ha betydning for sluttresultatet dersom brukeren velger å benytte overgangene sekvensielt. Når brukeren til slutt trykker OK vil det ut fra valgene brukeren har gjort på denne siden (så lenge brukeren har valgt minst én overgang) genereres overganger for alle slideshowets bilder.



Figur 4-24 – NewSlideshowDialog, transitions tab.

Color & Preview

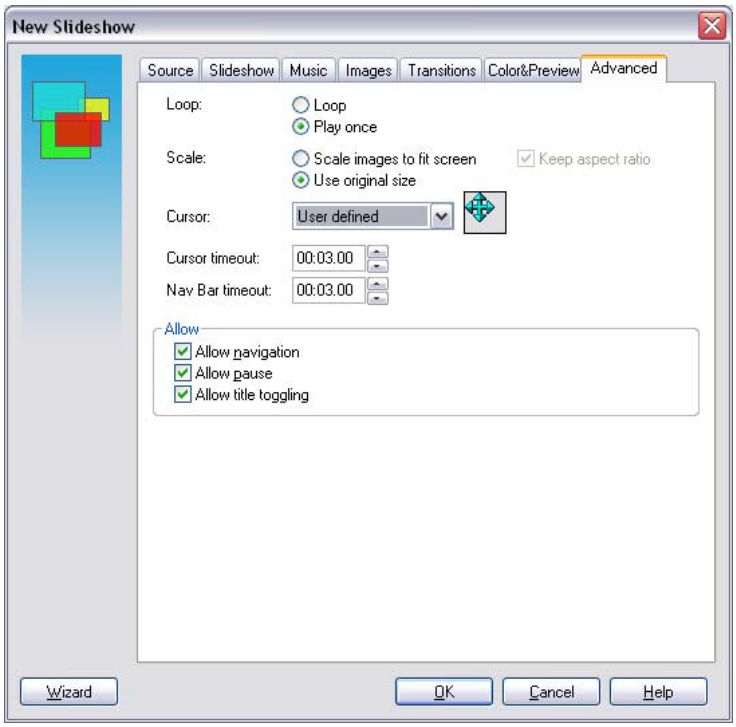
Denne fanen, se Figur 4-25, lar brukeren posisjonere slideshowtittel, slidetittel og navigasjonsmeny, ved å klikke på ønsket knapp til høyre for forhåndsvisningen og siden klikke og dra (for titlene) for å markere hvor disse skal plasseres i skjermbildet. Det er her også mulig å velge bakgrunnsfarge for slideshowet, dvs. hvilken farge som vises før første bilde, etter siste bilde og eventuelt rundt bilder som ikke fyller hele skjermen. Til slutt er det mulig å bestemme font, tekstfarge, farge for tekstomriss, og justering (venstre, midt, høyre) for slideshowtittel og slidetittel. Alle endringer blir automatisk visualisert i forhåndsvisningspanelet, som er en egenutviklet komponent. Denne komponenten registrerer også posisjonene når brukeren plasserer titler og navigasjonsmeny.



Figur 4-25 – NewSlideshowDialog, color & preview tab.

Advanced

Denne fanen, se Figur 4-26, lar brukeren mulighet til å foreta avanserte slideshowinnstillinger. Her kan brukeren velge om slideshowet skal spilles av kun én gang, eller om det skal loope til det blir eksplisitt stoppet. Det er mulig å velge om bilder skal vises i opprinnelig størrelse, eller om de skal strekkes, og om de i så fall skal beholde opprinnelig lengde-bredde-forhold. Videre kan brukeren velge hvor lang tid det skal ta før musepeker og navigasjonsmenyen skal forsvinne og hvordan musepeker skal se ut. Siden .NET-klassebiblioteket ikke har støtte for å opprette musepeker med farger, har vi måttet benytte en win32-funksjon fra user32.dll til dette. Til slutt går det an å begrense navigasjonsmulighetene for personen som skal spille av slideshowet; om navigasjon er tillatt, om det skal være mulig å pause fremvisningen og skjule/vise titler.

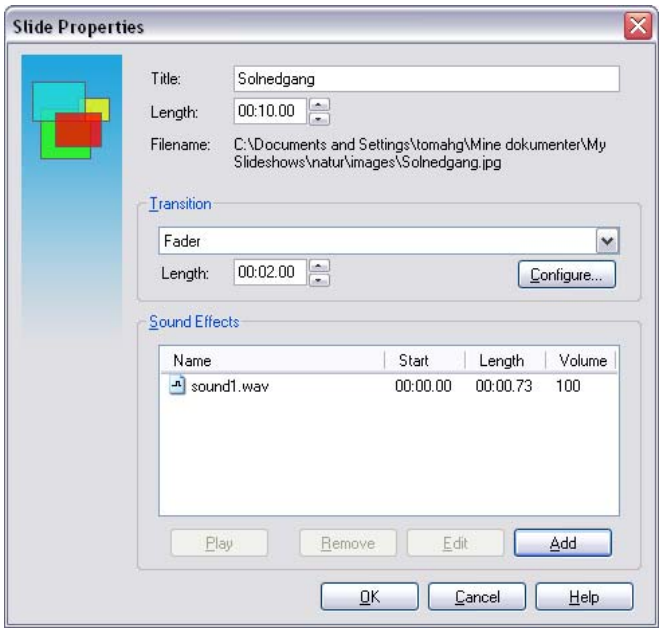


Figur 4-26 – NewSlideshowDialog, advanced tab.

Denne dialogen er tilgjengelig ved å velge New i File-menyen, fra verktøylinjen og fra SlideshowCenter dersom det i PreferencesDialog ikke er krysset av for at wizarden skal vises isteden. Det er også mulig å vise denne dialogen ved å velge Advanced fra wizarden.

SlidePropertiesDialog

Dialog, med utseende som i Figur 4-27, der brukeren kan endre innstillinger for en bestemt slide. Slidens tittel og lengde kan endres, slidens overgang kan velges og konfigureres, overgangens lengde kan endres. I tillegg kan lyder legges til og fjernes. Når brukeren trykker OK, blir den aktuelle sliden oppdatert med de nye innstillingene.



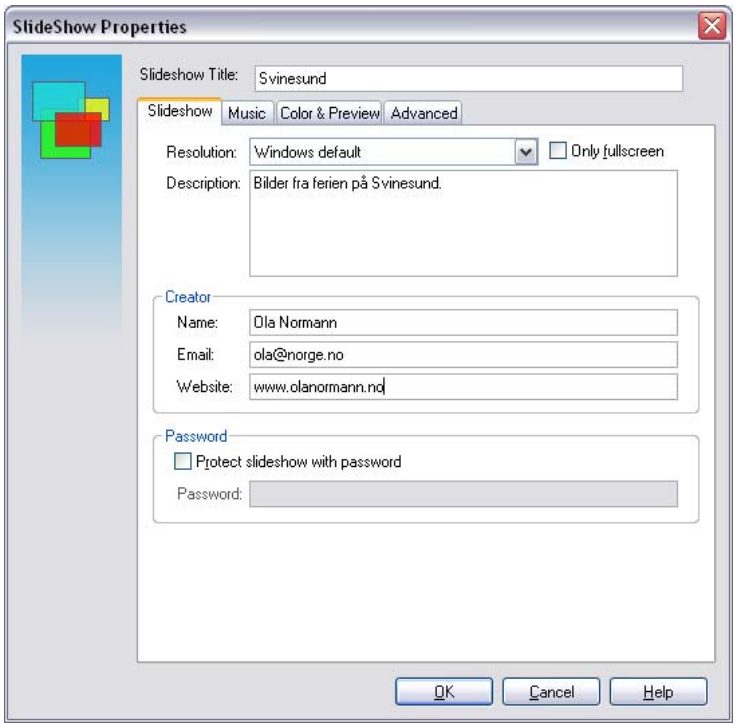
Figur 4-27 – SlidePropertiesDialog

Denne dialogen er tilgjengelig ved å velge Slide Properties i Edit-menyen, ved å dobbeltklikke på en slide i slidepanelet eller ved å høyreklikke på en slide i slidepanelet og velge properties fra høyreklikksmenyen.

SlideShowPropertiesDialog

Dersom man har opprettet et nytt slideshow vha wizarden, vil man ha gått glipp av muligheten til å spesifisere mer avanserte valg slik man kan gjøre i den mer avanserte NewSlideshowDialog. I denne properties dialogen, se Figur 4-28, kan man, etter et slideshow er opprettet, og uavhengig av hvordan det er opprettet, endre på alle innstillinger man kan gjøre under opprettelse av slideshowet. Det er også mulig å endre innstillinger for et slideshow uten å åpne det, dette kan gjøres ved å høyreklikke på et slideshow i SlideshowCenter og velge ”properties” fra høyreklikksmenyen.

Denne dialogen inneholder de samme tabbene som NewSlideshowDialog unntatt Source, Images og Transition, isteden vises slideshowets tittel over alle tabbene. Denne dialogen laster kun inn de tabbene som vises, i likhet med NewSlideshowDialog. Når brukeren trykker OK, blir det aktuelle slideshowets innstillinger oppdatert.



Figur 4-28 – SlideshowPropertiesDialog

Denne dialogen er tilgjengelig ved å velge Properties i Slideshow-menyen og fra verktøylinjen. I tillegg kan man høyreklikke på et slideshow i SlideshowCenter og velge Properties fra høyreklikkmenyen der, uavhengig av om slideshowet er åpent eller ikke, dersom det ikke er åpnet blir det lagret når brukeren trykker OK.

SplashScreenDialog

Siden applikasjonen bruker litt tid på å starte opp, vil vi vise en ”Splash Screen” mens applikasjonen starter, slik at brukeren blir informert om at applikasjonen er i ferd med å starte opp. Dialogen lukkes automatisk så snart applikasjonen har startet. Dialogen er en helt standard dialog uten den vanlige rammen rundt, den har i tillegg et bakgrunnsbilde som er det eneste av dialogen som synes. Dette bakgrunnsbildet (se Figur 4-29), er tegnet for oss av Josh McIntyre.

Figur 4-29 – SplashScreenDialog.

WizardDialog

Ideen med Wizarden er at det skal være så enkelt som mulig for en bruker å raskt generere et nytt slideshow. Dersom valgmulighetene skulle bli for begrenset kan brukeren, når som helst, skifte til NewSlideshowDialogen ved å trykke på knappen ”Advanced” og fortsette å spesifisere parametere for det nye slideshowet der. Dersom brukeren etter at slideshowet er opprettet ønsker å gjøre mer avanserte innstillinger kan dialogen SlideshowPropertes brukes til dette.

Dialogen er delt inn i tre sider, som det er mulig å gå frem og tilbake mellom, fra den siste siden kan man trykke på knappen ”Finish” for å generere selve slideshowet. Denne genereringen foregår på samme måte som for NewSlideshowDialog. For at denne dialogen skal kunne vises raskere lastes kun den første siden inn før dialogen vises, de andre sidene lastes kun inn ved behov.

Første side – Kilde

På den første siden, se Figur 4-30, må brukeren velge om det skal opprettes nytt slideshow fra grunnen av, i så fall kan tittel angis og brukeren kan velge om det nye slideshowet skal lagres i komprimert format. Den andre muligheten er at brukeren kan opprette et nytt slideshow med utgangspunkt i et tidligere lagret slideshow, da må plasseringen til dette slideshowet angis.

Figur 4-30 – WizardDialog, source.

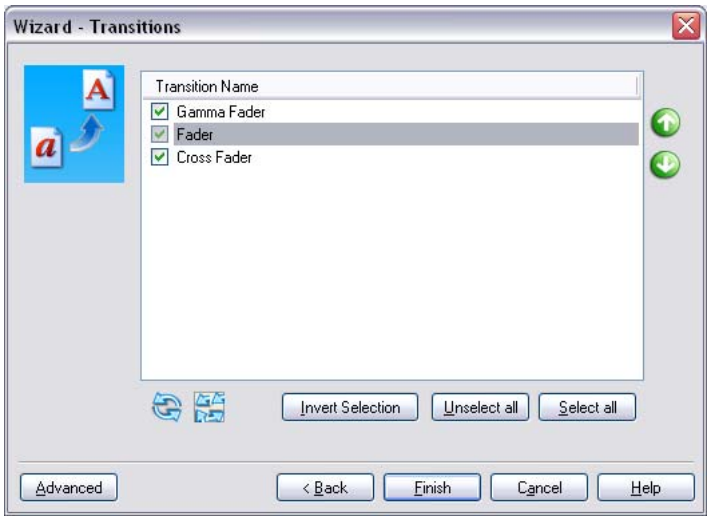
Andre side – Media

På den andre siden, se Figur 4-31, kan brukeren legge forskjellige media til slideshowet. Øverst kan brukeren legge til en folder med bilder som skal inkluderes i slideshowet, og angi hvor lenge hvert enkelt bilde skal vises.
Om ønskelig kan det også legges til en lydfil som blir benyttet som bakgrunnsmusikk, man kan også velge om denne skal spilles igjen og igjen (loopes) dersom slideshowet er lengre enn bakgrunnsmusikken.

Figur 4-31 – WizardDialog, media.

Tredje side – Overganger

Innholdet på denne siden, se Figur 4-32, er identisk med det man finner på tilsvarende fane (tab) i NewSlideshowDialogen, det samme panelet blir benyttet begge steder. Dette panelet lar brukeren velge hvilke overganger som skal brukes i det nye slideshowet. Her krysser brukeren av de overganger han vil bruke, og velger om disse skal brukes sekvensielt eller i tilfeldig rekkefølge.



Figur 4-32 – WizardDialog, transitions.

Denne dialogen er tilgjengelig ved å velge New i File-menyen, fra verktøylinjen og fra SlideshowCenter dersom det i PreferencedDialog er krysset av for at wizarden skal vises. Det er også mulig å vise denne dialogen ved å velge Wizard fra NewSlideshowDialog.

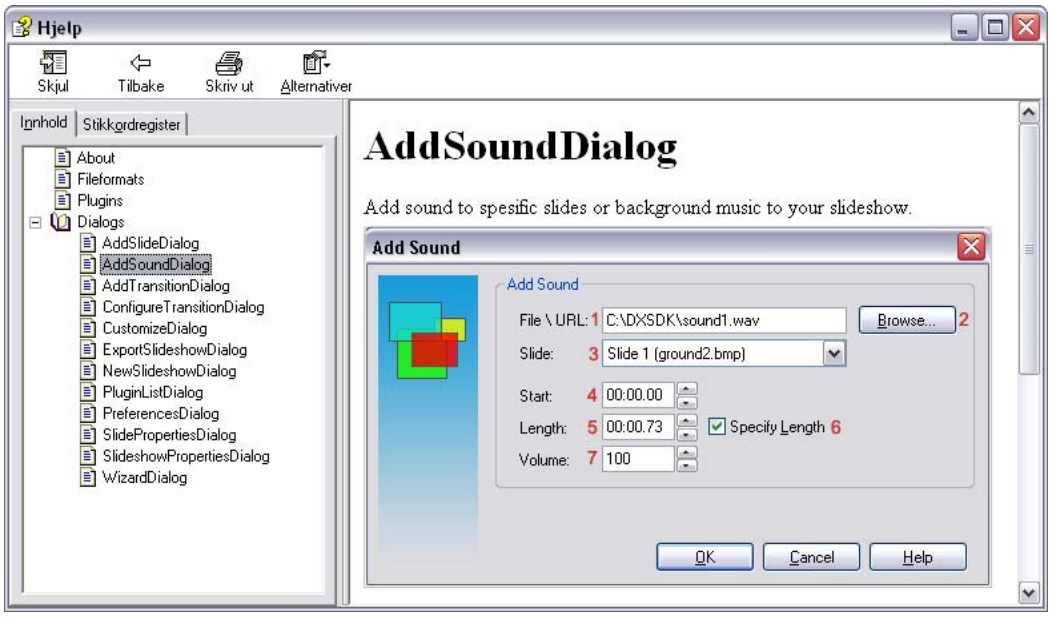
4.4.2 Andre komponenter

Hjelpesystem

For å opprette et hjelpesystem for applikasjonen, se Figur 4-33, har vi benyttet HTML Help Workshop 1.3 [4]. Dette systemet lar oss skrive et HTML-dokument for hvert tema det skal være tilgjengelig hjelp for, for så å kompilere disse HTML-filene sammen med eventuelle bilder og informasjon om innholdsfortegnelse og stikkordregister til en CHM-fil, som vi kan bruke fra vår applikasjon. Denne kompilerte CHM-filen har utseende og virkemåte som hjelpefunksjonen man finner igjen i de fleste moderne dataprogram, slik at brukeren bør føle at han er i kjente omgivelser. Her har man muligheten til å velge tema fra en innholdsfortegnelse med kategorier og underkategorier, og man kan også søke i stikkordregisteret.

I alle applikasjonens dialoger har vi lagt til en hjelp-knapp, som når brukeren klikker på den, åpner hjelpesystemet og viser tilgjengelig hjelp for den aktuelle dialogen. Hjelpesystemet er i tillegg tilgjengelig fra hovedmenyen, og det åpnes om man trykker kontrolltasten F1.

I samråd med oppdragsgiver har det blitt besluttet å ikke bruke tid på å utvikle innhold til hjelpesystemet. Men siden vi har laget grunnsystemet og integrert dette i applikasjonen, er det enkelt å senere fylle inn innhold ved å skrive HTML-kode, for siden å kompilere hjelpefilen med programmet HTML Help Workshop som er fritt tilgjengelig.



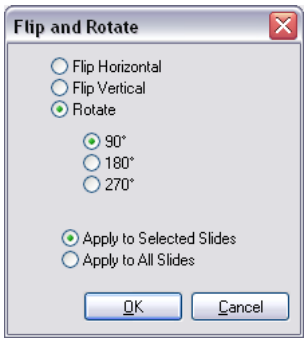
Figur 4-33 – Hjelpesystemet.

Bildebehandling

På Image-menyen er det tilgjengelig en del verktøy for å endre på ett, eller flere av slideshowets bilder.

Rotering /vending

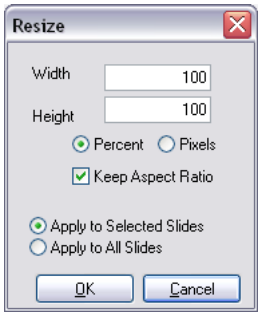
Digitale bilder kan ofte ha feil orientering, dersom fotografen har holdt kameraet på siden da bildene ble tatt, derfor er det aktuelt å kunne endre bilenes orientering i applikasjonen. Her kan brukeren rotere 90, 180 eller 270 grader, vende horisontalt eller vertikalt. Operasjonen kan utføres på markerte bilder, eller på alle bilder. Dialogen for denne funksjonen finnes i Figur 4-34. Ved utføring av disse operasjonene benyttes ferdige metoder i .NET-rammeverkets Bitmap-objekt.



Figur 4-34 – Dialog for rotering og vending.

Skalering

Digitale bilder har ofte så høy oppløsning at det er ønskelig å forminske dem, derfor det aktuelt å kunne skalere ett eller flere bilder fra applikasjonen. Denne funksjonen kan utføres på markerte bilder, eller alle bilder, og kan endre størrelsen på bildene prosentvis. Dersom kun et bilde er markert er det mulig å angi størrelsen på resultatbildet i antall pixler. Det er også mulig å velge om det skal tillates at resultatbildet har et høyde-bredde-forhold forskjellig fra originalbildet. Dialogen for denne funksjonen vises i Figur 4-35. Ved skalering av bildene opprettes et nytt Bitmap-objekt med riktig størrelse, med utgangspunkt i det gamle. Det gamle opprinnelige bildet blir så byttet ut med det skalerte.



Figur 4-35 – Dialog for skalering av bilder.

Beskjæring

Ofte er det bare en liten del av et bilde som er interessant, derfor har vi et verktøy for å gjøre det utsnitt i bildet. Ved å først bruke applikasjonens markeringsverktøy for å markere det området man ønsker å beholde, for så å velge ”crop” fra Image-menyen vil det opprinnelige bildet erstattes med det ønskede utsnittet. Ved beskjæring blir det først opprettet et nytt Bitmap-objekt med størrelse lik det markerte området, så blir

pixelverdiene fra originalbildets markerte område kopiert til det nye bildet. Til slutt blir originalbildet byttet ut med resultatbildet.

4.5 Fremviserapplikasjon

Fremviserapplikasjonen benytter seg hovedsakelig av de samme funksjoner som benyttes i slideshowgeneratoren. Det eneste som er spesielt for fremviserapplikasjonen er at den ved kjøring først kopierer ut to nødvendige DLL-filer, som er lagt til som ressurser i EXE-filen, disse filene inneholder bl.a. funksjoner for dekomprimering og avspilling av slideshowet og grensesnitt mot plugin. Deretter leter den seg frem til slutten av filen den ble startet fra, der den henter ut et tall, dette tallet benyttes videre som en adresse fra starten av filen. Ved denne adressen ligger selve slideshowet lagret, det blir pakket ut til en temporær katalog og lastet. Dersom slideshowet er passordbeskyttet, vil det bli vist en dialog der brukeren må angi riktig passord for å kunne fortsette lastingen av slideshowet. Passordet krypteres vha MD5-algoritmen, og er derfor ikke mulig å hente ut fra fremviserapplikasjonen av uvedkommende. Det blir videre undersøkt om slideshowet inneholder uverifiserte plugin, og i så fall får brukeren mulighet til å verifisere disse før kjøring av slideshowet.

4.6 Fremvisning

Fremvisning av slideshow kan vises enten i fullskjerm eller i et vindu. Forskjellen mellom disse modi er avgrenset til DirectX. Det betyr at man ikke trenger å bry seg spesielt om en DirectX applikasjon vises i fullskjerm eller i et vindu. Det eneste man må gjøre er å opprette DirectX Device'n i ønsket modus. Figur 4-36 viser et slideshow i et vindu.

Figur 4-36 – Fremvisning uten, og med titler.

For å projisere bildene i et slideshow trenger man ikke støtte for 3D. Det enkleste er å tegne bildene rett til skjermen. Allikevel ble det tidlig bestemt at fremvisningen skulle inneholde elementer som måtte være i det tredimensjonale rom. Dette fordi vi ønsket overganger som kunne operere i tre dimensjoner. Dermed ble det bestemt at vi skulle benytte oss av DirectX Direct3D API. For å tegne bilder i 2D på en 3D Device benyttes en hjelpeklasse som håndterer dette. Denne klassen oppretter et rektangel (to trekanter)

som plasseres slik at de dekker hele skjermen. Deretter tegnes de to trekantene opp med et bilde som teksturer. Dette betyr at alle bildene må lastes inn som teksturer for at de skal tegnes opp i det tredimensjonale miljøet som er satt opp.

Som beskrevet i tidligere avsnitt og kapitler kan man i slideshowgeneratoren sette sammen slideshow og velge hvordan slideshowet skal vises i fremviseren. Figur 4-36 er et eksempel der brukeren til venstre har valgt å ikke vise titler, mens brukeren til høyre har valgt å vise titler. I tillegg har brukeren til høyre selv valgt posisjonen til disse titlene.

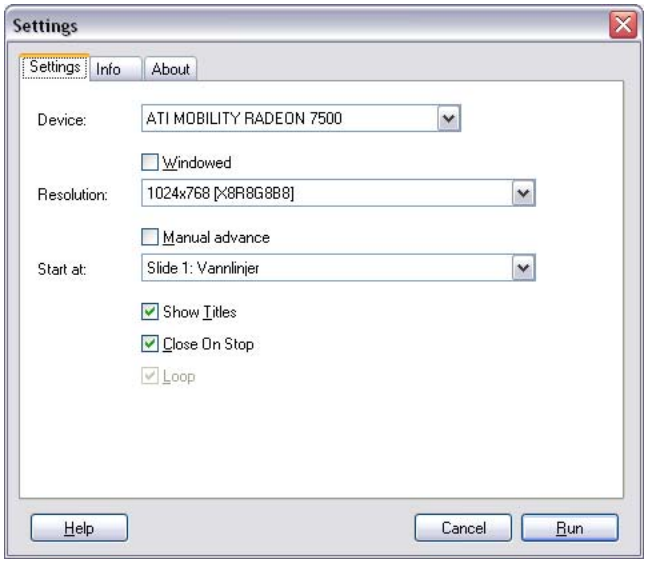
Tabell 4-2 viser hvilke valg som påvirker fremvisningen. Noen av disse valgene kan også justeres på forhånd. Mer om det i avsnitt 4.6.1.

Tabell 4-2 – Valg som påvirker visningen gjort i slideshowgeneratoren.

Bakgrunnsfarge	Brukeren kan velge bakgrunnsfargen som dekker alt av skjermen et bilde ikke dekker.
Titler	Brukeren kan velge om titler skal vises og font, farge og posisjon. Se Figur 4-36.
Skalering	Brukeren kan velge om bildene skal skaleres. Bildene kan enten skaleres ved å beholde bredde-høyde-forholdet eller ved å strekke bildene så de dekker hele skjermen
Musepeker	Brukeren kan velge hva slags musepeker som skal brukes.
Loop	Brukeren kan velge om slideshowet skal startes på nytt etter siste slide.
Musikk Loop	Brukeren kan velge om musikken skal starte på nytt etter at siste musikkelement er avspilt.
Timeout	Brukeren kan velge hvor lang tid det skal ta før menyen og/eller musepekeren skal forsvinne, etter at brukeren av fremviseren har vært aktiv.

4.6.1 Valg før visning

For å gjøre fremviseren mindre komplisert er det utviklet en dialog der brukeren kan justere forskjellige innstillinger før visning av slideshow. Figur 4-37 viser disse innstillingene og Tabell 4-3 beskriver hva og hvordan innstillingene påvirker visningen. Brukeren som har generert/laget slideshowet som skal vises har som tidligere nevnt mulighet til å begrense funksjonaliteten til slideshowet. For eksempel muligheten til å sette største mulige oppløsning eller velge om titler skal vises og brukeren ikke har tilgang til å skifte på dette valget. Derfor kan flere av innstillingene i spesielle tilfeller ikke være tilgjengelig for brukeren som skal se slideshowet.

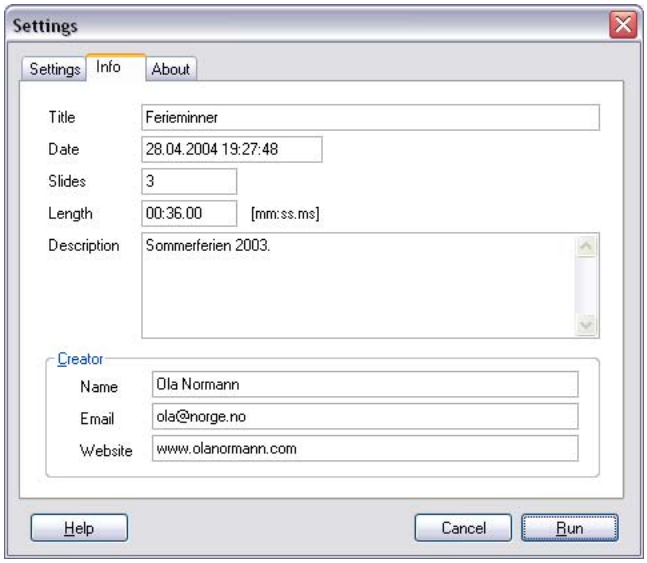


Figur 4-37 – Valg før start av fremvisning.

Tabell 4-3 – Innstillinger før start av fremvisning.

Device	Brukeren kan velge hvilket skjermkort slideshow skal projiseres til.
Windowed	Om slideshowet skal dekke hele skjermen eller komme opp i et vindu.
Resolution	Oppløsningen på skjermen eller størrelsen på vinduet.
Manual advance	Om brukeren selv må gå til neste slide.
Start at	Hvilken slide som skal vises først.
Show titles	Om titler skal vises.
Close on stop	Om visningen skal lukkes når slideshowet stoppes.
Loop	Om slideshowet skal starte på nytt etter siste slide.

I tillegg til å endre på innstillinger har brukeren også mulighet til å se generelle data om slideshowet som skal vises. Figur 4-38 viser informasjon om et slideshow.



Figur 4-38 – Informasjon om slideshow.

4.6.2 Navigasjonsmeny

Under fremvisningen vises også en navigasjonsmeny. Denne menyen er laget slik at det er mulig å forandre utseende svært lett. Figur 4-39 viser til venstre hvordan navigasjonsmenyen blir tegnet i fremviseren, og til høyre hvordan navigasjonsmenyen egentlig ser ut. Den øverste delen av bildet består av selve menyen, mens knappene er plassert under. Når menyen tegnes under visning av et slideshow, tegnes knappene inn i "hullene". Disse "hullene" er definert som rektangler som returneres av et grensesnitt (interface). "Hullene" behøves fordi det blir tegnet en annen bakgrunnsfarge når brukeren holder musen over en av knappene. Fargen returneres også som en egen metode i grensesnittet. Grensesnittet definerer også selve menyens og knappenes plassering i kildebildet.

Figur 4-39 – Navigasjonsmeny, både i fremviseren og selve bildet

Vi har utviklet en klasse som implementerer dette grensesnittet og enkelt henter ut verdiene om en meny fra en ressursfil. Ressursfilen er kompilert inn i kjernen (Core.dll) av applikasjonen. Slik unngår vi at andre enkelt kan endre på standardmenyen vi har laget. Allikevel er det enkelt for oss som utviklere å endre denne, da vi kun trenger å legge bildet til som en kildefil under kompilering av applikasjonen.

Grensesnittet er laget for at man senere kan utvikle andre måte å laste inn forskjellige menyer på. For eksempel kunne det vært interessant å lage et eget filformat for en meny, slik at brukeren kunne velge "skin" på menyen selv.

4.6.3 Timing

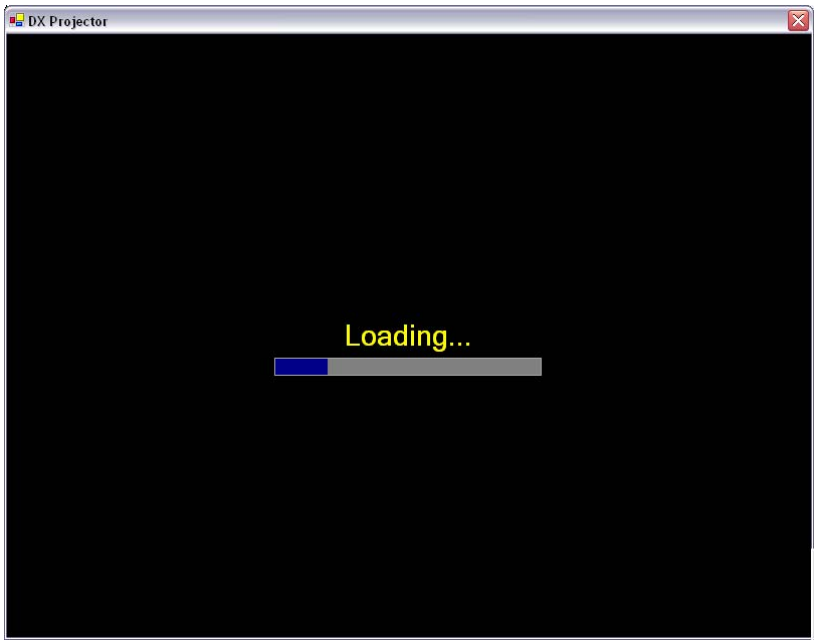
Timing er essensielt for å få vist rett bilde til rett tid og spilt av musikk og andre lydeffekter på rett tidspunkt. I fremviseren er dette løst ved at det går en tråd parallelt med hovedtråden. Hovedtråden har ansvaret for å tegne opp skjermen eller vinduet når det er nødvendig, mens timing-tråden informerer hovedtråden om hva som skal tegnes opp og sier ifra om når musikk og lyd skal avspilles. I tillegg til disse trådene er det en tredje tråd, som brukes til å bufre bilder. Denne er spesifisert i 4.6.4. For at alle trådene skal samarbeide må trådene synkroniseres. Dette gjøres ved å benytte seg av Monitor-klassen i .NET rammeverket. Denne klassen inneholder metoder som låser et objekt, slik

at en ny tråd som vil foreta samme lås på dette objektet må vente til den tråden som har låst objektet låser opp igjen.

4.6.4 Bufring

For at fremviseren skal vise rett bilde til rett tid, uten at brukeren må vente mellom hvert bilde, må det bufres mens slideshowet vises. I dette tilfelle betyr bufring at bildene gjøres klar som teksturer i skjermkortets minne, eller i systemminnet om nødvendig.

Lasting av bilder til teksturerer vil aldri ta like lang tid på forskjellig maskinvare. I noen tilfeller vil det raskt, mens andre tilfeller gå tregt. Derfor bufres det minst fire bilder ved oppstart. I tillegg bufres det når applikasjonen finner det nødvendig. Figur 4-40 viser skjermbildet som tegnes mens det bufres. Bufringen er som nevnt i forrige avsnitt skilt ut i en egen tråd, som betyr at tråden som bufrer og hovedtråden (den som tegner til skjermen) må synkroniseres. Dette fordi de trenger synkron adgang til samme variabler i samme minneområde. Siden bufringen bruker mye av maskinens ressurser både når det gjelder prosessorkraft og ”skjermkort-kraft”, blir bufringen satt på pause mens en overgang vises. Dette fordi når en overgang vises, kan et overgangsplugin benytte seg av svært tunge operasjoner som betyr at det trengs maks kraft fra maskinvaren. Dvs. at det kun bufres ved oppstart og når hvert enkelt bilde vises, og altså ikke mens en overgang vises.



Figur 4-40 – Oppstart av fremviseren. Bufrer de fire første bildene.

4.6.5 Minnebruk

Når det blir laget teksturer av bildene i et slideshow blir teksturene plassert både i skjermkortminne og systemminne. Rent teknisk betyr dette at teksturene legges i en ”Managed Memory Pool”. Dette betyr at hvis teksturen av en eller annen grunn skulle bli borte fra skjermkortminne vil systemet oppdage det og lage en ny kopi i fra system-minnet. Teksturer eller andre ressurser kan bli borte fra skjermkortminnet hvis

applikasjonen som viser et slideshow mister fokus under fullskjermmodus. Dette fordi Windows da legger til rette for at andre applikasjoner kan få tilgang til skjermkortet. Denne tilstanden kalles DeviceLost og forteller applikasjonen at den nå må vente til den kan få tilgang på skjermkortet (”Device’n”) igjen. Når applikasjonen på nytt får tilgang til skjermkortet, må den sørge for at alle ressurser er klar til bruk.

Managed DirectX er veldig greit å bruke når det kommer til minnehåndtering. Alle ressurser som blir opprettet har implementert IDisposable grensesnittet (interface), som betyr at alle klasser har metoden Dispose(). Denne metoden kalles når man vil frigjøre minnet objektet har tilegnet seg. Denne metoden blir kjørt, på alle teksturer som er bufret til minnet, når slideshowet er ferdig og vinduet lukkes. Hvis vi som utviklere ikke kaller metoden, blir dette objektet uansett plukket opp av Garbage Collectoren.

Mens slideshowet vises blir også alle teksturer til slides som er blitt vist frigjort, slik at det blir plass til nye bilder. Kun to bilder blir spart slik at brukeren også kan navigere seg bakover to slides før applikasjonen finner ut at den må bufre. Hvis brukeren raskt hopper tre bilder tilbake må det bufres på nytt og Figur 4-40 vil bli vist til applikasjonen har bufret ferdig.

4.7 Systemkrav

På grunn av bruken av .NET-rammeverket stiller dette en del krav til systemet som skal kjøre applikasjonen. En bok om .NET-programmering [16], som etter hvert kapittel har noen enkle programmeringsøvelser opererer med følgende minimum systemkrav:

- Microsoft Windows 2000, eller nyere.
- 450 Mhz prosessor (600Mhz anbefales).
- 192 MB RAM (256 MB anbefales).
- SVGA monitor, oppløsning på minst 800 x 600 og minst 256 farger.
- Mus, eller annen kompatibel pekeenhet

I tillegg må man ha installert .NET rammeverket versjon 1.1.
Dette er systemkrav for å kjøre utviklingsmiljøet Visual Studio .NET 2003, og en enkel .NET-applikasjon bør kunne kjøre på et system med vesentlig dårligere spesifikasjoner. Men siden vår applikasjon i tillegg benytter seg av DirectX bør ikke systemet være noe dårligere enn dette, for at applikasjonen skal kjøre glatt. 256 MB RAM er anbefalt for bruk av Windows XP, i tillegg bør man ha et ikke alt for gammelt skjermkort for å få full glede av fremvisningen av slideshowet.

4.8 Komprimering

Siden vi ønsket å kunne lagre et slideshow til en komprimert fil, en fil som også skulle benyttes i fremviserapplikasjonen hadde vi behov å kunne komprimere flere filer til én fil. Siden det ville vært en omfattende jobb å skrive dette på egenhånd, falt valget isteden #ziplib [23] (kort for SharpZipLib). SharpZipLib er et bibliotek som har funksjoner for komprimering (og dekomprimering) i formatene ZIP, GZIP, BZIP2 og TAR. Det er skrevet i C# for .NET-plattformen, har en lisens som tillater bruk i kommersielle closed-source-applikasjoner og er tilgjengelig med kildekode. Vi har valgt å bruke ZIP-komprimeringen For at biblioteket skulle kunne brukes til å komprimere filer der

filnavnet inneholder tegn som ikke finnes i 7 bits ASCII[6] (f.eks. Æ, Ø og Å) måtte vi gjøre en liten endring i koden.

4.9 Internasjonalisering

Selv om applikasjonen skal være engelskspråklig var det et krav at det skulle tilrettelegges for at en eventuell senere oversettelse skulle være så enkel som mulig. Vi har brukt Visual Studios designer til å sette opp alt brukergrensesnittet, og denne designeren har en mulighet for å skille ut alle språkavhengige data i egne ressursfiler, som vi har benyttet oss av. Disse ressursfilene inneholder ikke bare selve tekstene, men også data om størrelse på, og plassering av GUI-komponenter. Dette er også en stor fordel siden enkelte språk krever flere ord for å formidle en melding enn andre. Vi har benyttet oss av disse automatisk genererte ressursfilene alle steder tekster vises direkte i brukergrensesnittet. I tillegg har vi brukt en del tekster som ikke ligger direkte i brukergrensesnittet, f.eks. knapper som avhengig av status kan ha forskjellige tekster, feilmeldinger osv. Dersom vi hadde lagt disse tekstene inn i de samme ressursfilene som Visual Studio genererer automatisk ville de manuelt innlagte tekstene blitt slettet automatisk, siden Visual Studio ikke ville forstå hvorfor de var der. Vi har isteden opprettet en egen ressursfil til sånne tekster.

Dersom applikasjonene senere skal oversettes til et annet språk kan Visual Studio brukes til å opprette ressursfiler for det nye språket, man må da spesifisere et nytt språk for hovedvinduet og alle dialogene og endre på de tekstene som skal være forskjellige på det nye språket. Disse ressursfilene er i XML-format, så det er også mulig å oversette dem manuelt vha en enkel teksteditor. I tillegg må ressursfilen med tekstene vi har lagt til manuelt oversettes til det nye språket. Siden alle tekster er skilt ut i egne filer vil oversettelsesprosessen kunne gå mye raskere enn om alle tekster var flettet inn i selve kildekodefilene, siden man da måtte finlese alle kildekodefilene på jakt etter språkavhengige tekster.

4.10 Distribusjon av slideshowgeneratoren

For at den vanlige Windows-brukeren enkelt kan prøve eller bruke slideshowgeneratoren, har vi satt opp en installasjonsapplikasjon som installerer slideshowgeneratoren og annen nødvendig software. Installasjonsapplikasjonen er laget i Microsoft Visual Studio .NET 2003, og genererer en MSI[7]-pakke [24] som er et format Microsoft bruker for å installere sine produkter. Microsoft Visual Studio .NET 2003 gjør denne jobben svært enkel, ved at man kan legge til støtte for .NET. I tillegg satte vi DirectX som en betingelse for å få kjørt installasjonen. Dvs. at hvis brukeren ikke har .NET eller DirectX vil installasjonsapplikasjonen henvise brukeren til en online installasjon av .NET og/eller installere DirectX versjonen vi har lagt ved. MSI systemet lagrer selvsagt avinstallasjonsinformasjon slik at brukeren kan fjerne slideshowgeneratoren på lik linje med andre applikasjoner.

[6] American Standard Code for Information Interchange
[7] MSI – Microsoft (Windows) Installer

4.11 Implementering av plugin

Siden hvem som helst kan lage et plugin til applikasjonen, og det ikke finnes noen begrensninger for hvilke operasjoner et slikt plugin kan utføre, er dette i utgangspunktet et potensielt sikkerhetshull. Derfor må hvert plugin, på forespørsel fra applikasjonens PluginManager returnere en signatur slik at applikasjonen kan verifisere pluginet.

Et plugin kan ha fem forskjellige tilstander:

- Software Verified
- User Verified
- User Verified One Run
- Not Verified
- Not Working

Dersom pluginet returnerer en gyldig signatur blir pluginets tilstand satt til ”Software Verified”, og det kan da fritt brukes. Denne signaturen vil være forskjellig fra plugin til plugin, og algoritmen som brukes her vil kun bli delt ut til utviklere man stoler på, utover dette må algoritmen hemmeligholdes for å bevare sin funksjon.
Dersom pluginet ikke returnerer en gyldig signatur vil pluginets status bli satt til ”Not Verified”, dette vil da si at pluginet ikke kan brukes før brukeren av applikasjonen selv eventuelt verifiserer pluginet for bruk. Brukeren kan da velge å verifisere pluginet for all fremtid (User Verified), eller kun frem til neste gang applikasjonen startes (User Verified One Run).

Et plugin som er verifisert av brukeren kan også avverifiseres av brukeren, mens et plugin som er ”Software Verified”, ikke kan avverifiseres.
Før et plugin blir forsøkt lastet blir det også undersøkt om det inneholder noen ugyldige referanser (referanser til bibliotek som ikke er tilgjengelig), dersom det inneholder slike referanser vil pluginets tilstand bli satt til ”Not Working”, og det vil da ikke blir forsøkt brukt.

Uavhengig av disse tilstandene kan brukeren også deaktivere plugin, et plugin som er deaktivert vil ikke bli forsøkt lastet, eller brukt, selv om det skulle være verifisert. På denne måten har brukeren mulighet til å deaktivere plugin han måtte mislike, eller ha problemer med, uten å fysisk slette selve pluginet. Informasjon om hvilke plugin som er deaktivert og verifisert av bruker blir lagret i registeret.

For plugingrensesnittenes kildekode, se vedlegg G.

4.11.1 Bildefiltypeplugin

Bildefiltypepluginene vi har utviklet har støtte for de mest brukte bildeformatene som finnes på Microsoft Windows plattformen. .NET Framework har støtte for disse bildeformatene, som betyr at vi ikke har brukt mye tid på lesing fra, og skriving til, de spesifikke bildeformatstandardene, men benyttet rammeverkets klasser for det de er verdt. Senere kan det være aktuelt å utvikle plugin for flere format, som må bygges opp helt fra bunnen av, ved for eksempel å studere spesifikasjonen til filformatet.

JPEG, PNG, BMP, GIF og TIF Bildefiltype plugin

Disse pluginene er svært like. De gjør akkurat det samme, men aksepterer kun sitt eget filformat. Det vil ikke si at det sjekkes på filendelse, som for eksempel .bmp, .jpg og .png, men det sjekkes om selve filen som prøves ut er i riktig format. Pluginets ikon hentes ut ifra brukerens filtypeassosiasjon, altså det samme ikonet som vises i Explorer.

4.11.2 Lydfiltypeplugin

I likhet med bildefiltypepluginene har vi benyttet oss av ferdiglagede klasser som enkelt leser forskjellige filformat. Her er ikke .NET Framework benyttet, men Microsoft DirectX (DirectShow) og en utvidelse kun til Managed DirectX. I stedet for å lage flere plugin til forskjellige format har vi her laget et plugin som tar flere forskjellige format.

DirectX Audio plugin

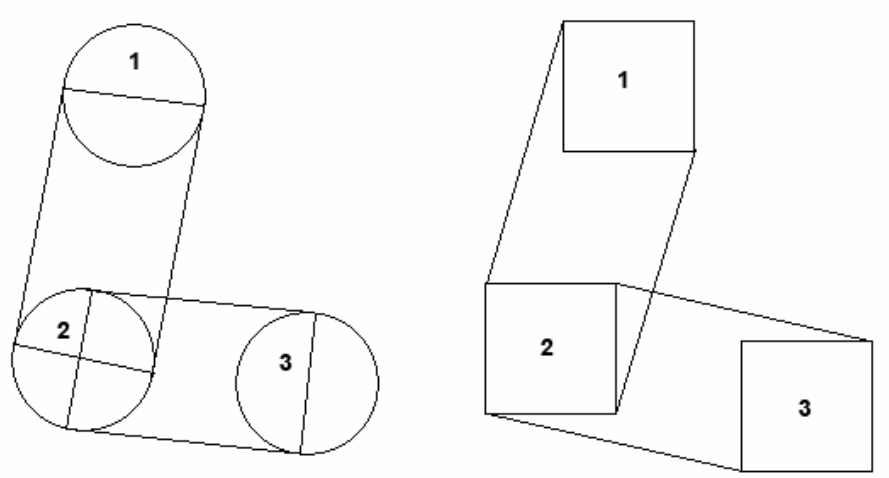
DirectX Audio Plugin er et plugin som brukes til avspilling av lyd. Pluginet støtter alle lydformat MediaPlayer kan spille fordi det baserer seg på DirectShow som er en del av DirectX. Managed DirectX inneholder AudioVideoPlayback namespacet som inneholder flere objekter som enkelt lar en benytte seg av DirectShow funksjonaliteten. Til dette audiopluginet benyttes kun klassen Audio, som inneholder simple metoder for å laste inn en lydfil samt Play() og Stop() som er svært selvforklarende. Pluginets ikon hentes ut ifra brukerens wave (WAV) filtypeassosiasjon, dvs. det samme ikonet som vises i Explorer for alle wave filer.

4.11.3 Bilderedigeringsplugin

Bilderedigeringsplugin har mulighet til å gjøre endringer på bildet. Data som valgt forgrunnsfarge, bakgrunnsfage, zoom og en eventuell markering er også tilgjengelig for pluginet. Fargene kan i tillegg også oppdateres av pluginet, slik at valgt farge i Toolbox blir endret.

Brush bilderedigerings plugin

Dette er et plugin som lar brukeren tegne på valgt bilde med en kvadratisk eller sirkulær pensel. Fra pluginets konfigurasjonspanel, som vises i toolboxen når pluginet er valgt, kan brukeren velge penselens form og størrelse. Ved å bevege musepeker over bildet mens en av museknappene holdes nedtrykket kan brukeren tegne på bildet med den valgte pensel. Dersom venstre musknapp holdes nede tegnes det med forgrunnsfargen og dersom høyre museknapp holdes nede tegnes det med bakgrunnsfargen.



Figur 4-41 – En linje tegnet fra punkt 1 til punkt 3, via punkt 2.

Når brukeren trykker ned en museknapp tegnes det en sirkel/firkant med aktuell størrelse, det samme gjøres når musepekeren slippes opp og hver gang applikasjonen får beskjed om at musepekeren har flyttet på seg. Hvis vi hadde fått beskjed hver gang musepekeren hadde flyttet seg, selv om det var kun én *pixel* hadde dette vært alt som var nødvendig for å tegne en jevn linje. Men dersom musepekeren beveges fort vil ikke disse beskjedene avfyres raskt nok og vi vil få en usammenhengende rekke av sirkler eller firkanter. For å sørge for at linjene blir sammenhengende blir det for sirkulære pensler tegnet en linje (se til venstre i Figur 4-41) med bredde lik sirklenes diameter fra nåværende sirkel til forrige sirkel hver gang en musebevegelse registreres. Siden en firkantet pensel vil gi linjer med forskjellig bredde avhengig av linjens retning måtte dette løses på en annen måte. Her blir det tegnet et polygon (se til høyre i Figur 4-41) mellom nåværende kvadrat og forrige kvadrat hver gang en musebevegelse blir registrert.

Pencil bilderedigerings plugin

Dette pluginet lar brukeren tegne en strek med én pixels bredde. Det fungerer med andre ord på nøyaktig samme måte som brush-pluginet dersom man har hadde satt penselens størrelse til én pixel. Siden det i andre enkle tegneprogram er vanlig å skille brush/pencil fra hverandre har vi valgt å gjøre det samme.

ColorPicker bilderedigerings plugin

Dette pluginet lar brukeren hente ut fargen fra et bestemt pixel i bildet og sette den fargen som aktiv farge i applikasjonen. Ved å klikke med venstre museknapp velges fargen som forgrunnsfarge og ved å klikke med høyre museknapp velges fargen som bakgrunnsfarge.

Inverter bilderedigeringsplugin

Dette er et plugin som inverterer fargene innen det markerte området, og dersom det ikke er markert noe område blir fargene i hele bildet invertert.

Fargen inverteres på følgende måte:
Ny farges rød-komponent = 255 – eksisterende farges rød-komponent.
Ny farges grønn-komponent = 255 – eksisterende farges grønn-komponent.

Ny farges blå-komponent = 255 – eksisterende farges blå-komponent.

RedEyeRemover bilderedigeringsplugin

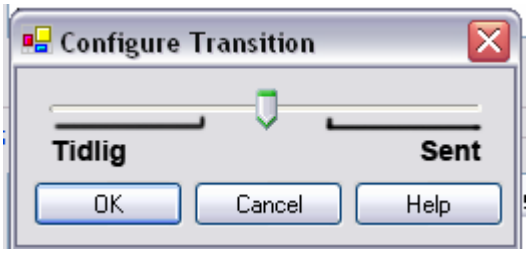
Som navnet tilsier er dette et plugin for fjerning av røde øyne. Før pluginet benyttes må et område inneholdende et eller flere øyne markeres. Når pluginet så velges vil det markerte området bli analysert og det rødeste pixelet bli lagret unna, hvor rød fargen er beregnes slik: *((andel rødt) / (andel rødt + andel grønt + andel blått)*. Så går pluginet gjennom alle pixlene i bildet og erstatter alle pixler som er så røde som det rødeste pixelet og inntil 3 % mindre rødt enn dette. Dersom det ikke ble fjernet tilstrekkelig rødfarge fra øynene ved første forsøk, kan pluginet kjøres igjen til ønsket resultat er oppnådd. Fargen rødfargen erstattes med er en brunfarge med noen tilfeldige variasjoner, slik at øynene ikke vil se helt ensfarget og kunstige ut etter operasjonen.

4.11.4 Overgangsplugin

Det er to typer overgangsplugin (se Figur 3-8); 2D- og 3D-plugin. 2D-plugin er plugin som endrer på pixelverdiene i bildene for å skape en overgang. 3D-plugin er plugin som ikke direkte endrer kun pixelverdier i bildene, men for eksempel tegner bildene i tre dimensjoner.

Fader

Dette pluginet er et 3D-plugin som gradvis minsker lysstyrken i det første bildet, og når dette bilde ikke lenger synes øker pluginet lysstyrken på det andre bildet. På denne måten ser det ut som om det første bildet blir borte i bakgrunnsfargen, mens det andre bildet dukker opp fra bakgrunnsfargen. Rent teknisk kan dette betegnes som alphablending. Dvs. at alphaverdiene i bildet blir multiplisert med en ny faktor, som her er nevnt som lysstyrke. Et enkelt eksempel er å si at når lysstyrken er 255 vises bildet slik det originalt er, mens når lysstyrken er 0 vises ikke bildet i det hele tatt. Dvs.; i starten er lysstyrken 255 og den går mot 0 mens det første bildet vises. Deretter går lysstyrken mot 255 igjen, mens det neste bildet vises.



Figur 4-42 – Konfigurasjonspanel for Fader-pluginet.

Dette pluginet har også et konfigurasjonspanel. I panelet kan brukeren stille på når lysstyrken skal være 0. Figur 4-42 viser hvordan brukeren kan stille på dette i slideshow-generatoren.

CrossFader

Dette pluginet er et 3D-plugin som i er svært likt Fader. I steden for å gå mot bakgrunnen, altså mot lysstyrke 0 blir heller neste bilde tegnet som bakgrunn. Det vil si at

lysstyrken/alphverdien til det første bildet går fra 255 til 0, og vil da gjøre neste bilde mer tydelig etter hvert som verdien nærmer seg 0.

StaticFader

Dette pluginet er et 2D-plugin som endrer pixelverdiene i det første bilde til gradvis å bli mer likt en ”tv-skjerm med snø” (static), for deretter å gå fra denne tilstanden tilbake til neste bilde. For å generere støy i bilde (static) brukes det tilfeldige tall mellom 0 og 255 som utgjør gråtoner. For å få det første bilde til å gå mot static og det andre bilde til å komme ut igjen, blir pixelverdiene fra bildene blandet med staticverdiene. Dette gjøres ved:

Fra første bilde til static:

*(1 - (tid / lengde)) * (pixelverdi) | (tid / lengde) * (staticverdi)*

Fra static til neste bilde:

*(tid / lengde) * (pixelverdi) | (1 - (tid / lengde)) * (staticverdi)*

SlideOut

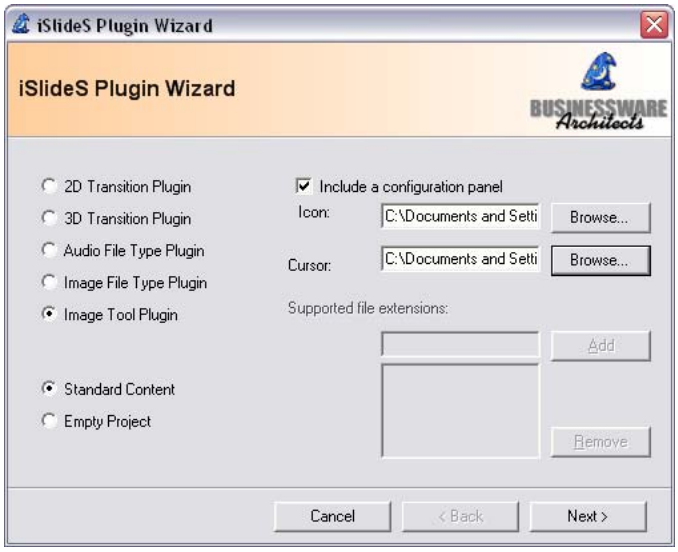
Dette 3D-pluginet skyver det gamle bildet ut av skjermen. En ganske generell effekt som ikke byr på så mange utfordringer for DirectX API’en. Overgangen har et konfigurasjonspanel der brukeren kan velge hvilken retning det gamle bildet skal gå ut. Valgene er høyre, venstre, opp, ned eller vilkårlig. Hvis vilkårlig er valgt vil det velges ny vei hver gang overgangen kjøres. Første gang pluginet blir valgt blir det vilkårlig satt til enten høyre, venstre, opp eller ned. Dette for å få variasjon ved automatisk generering ved bruk av ’Slideshow Wizard Dialog’ eller ’New Slideshow Dialog’.

SlideIn

I likhet med SlideOut overgangen er dette et 3D-plugin og en generell effekt. Overgangen skyver det nye bildet inn enten fra venstre, høyre, toppen, bunnen eller helt vilkårlig hver gang det brukes. For å bestemme hvilken side det nye bildet skal komme inn fra, brukes konfigurasjonspanelet som er svært likt SlideOut sitt konfigurasjonspanel.

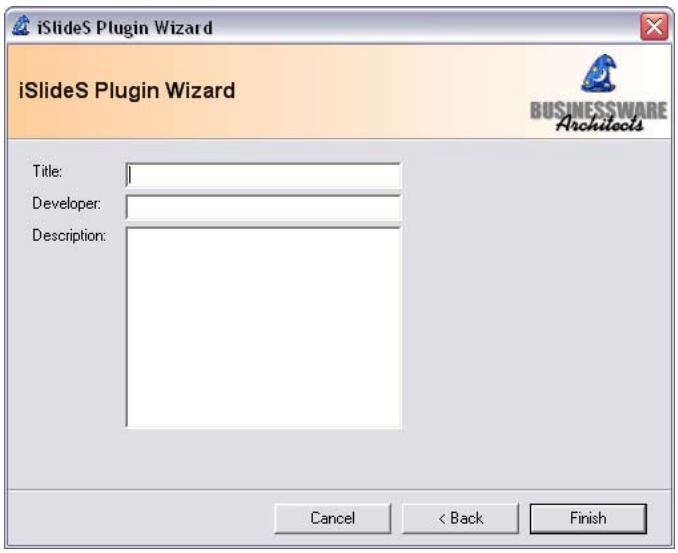
4.12 Plugin Wizard for Visual Studio

For å gjøre det enkelt å utvikle flere plugin, og å senke terskelen for andre utviklere som vil utvikle plugin for applikasjonen, har vi utviklet en Wizard for Visual Studio (se Figur 4-43). Denne Wizarden lar brukeren først velge hvilken type plugin som skal utvikles (2D Transisition, 3D Transition, Audio FileType, Image FileType, Image Tool), og om prosjektet kun skal inneholde klasse-/metodedefinisjoner eller også relevant eksempelkode. I tillegg kan man legge til ikon, musepeker og konfigurasjonspanel for de plugintyper der dette er aktuelt, og støttede filtyper for filtypeplugin.



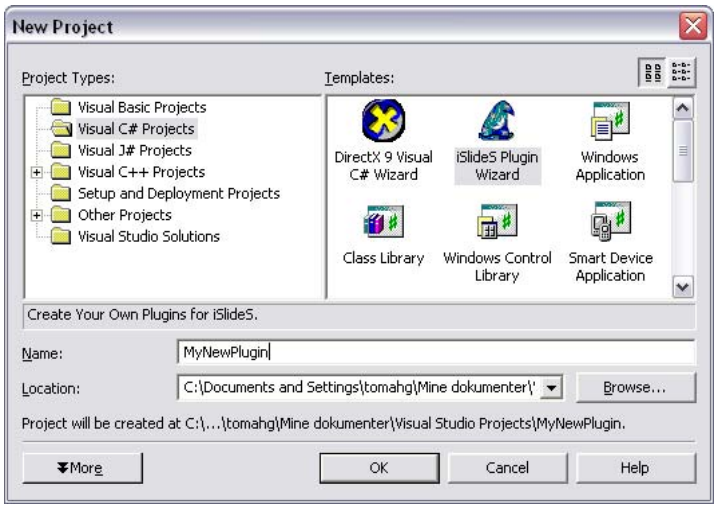
Figur 4-43 – PluginWizard for VisualStudio – første side.

På wizardens neste side, se Figur 4-44, er det mulighet for å skrive inn tittel på pluginet, navn på utvikler og en beskrivelse av selve pluginet. Denne informasjonen vil bli lagt til i prosjektets AssemblyInfo.cs-fil som brukes av slideshowgeneratoren for å hente ut denne informasjonen om pluginet.



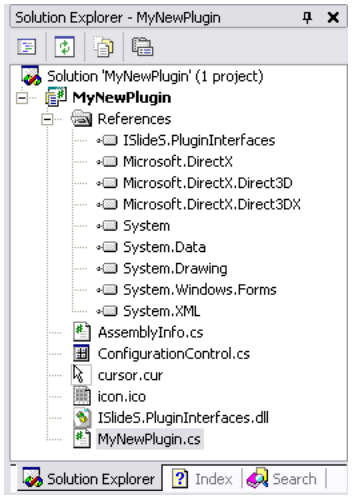
Figur 4-44 – PluginWizard for VisualStudio – andre side.

Wizardens grunnleggende oppbygning er opprettet ved å bruke The Wiz Wiz Wizard fra Businessware Architects [25], som enkelt lot oss opprette en tom Wizard. Vi har kun lagt til ønsket brukergrensesnitt og funksjonalitet for å inkludere nødvendige referanser og kildekodefiler i prosjektet. Denne løsningen inneholder også en installasjonsapplikasjon som integrerer Wizarden med Visual Studio, se Figur 4-45, slik at denne kan velges fra dialogen New Project. Dette gjør også at Wizarden enkelt kan fjernes ved å velge legg til/fjern programmer fra kontrollpanelet.



Figur 4-45 – Wizardens integrering med Visual Studio.

Når man trykker Finish i Wizarden vil det bli opprettet en ny solution med et nytt prosjekt (se Figur 4-46) som inneholder en fil som implementerer de metodene spesifisert i ønsket plugins interface, de nødvendige referanser, og en AssemblyInfo-fil som inneholder tittel, navn på utvikler og beskrivelse av pluginet. Det eneste som gjenstår for utvikleren er å fylle inn ønsket kode i metodedefinisjonene og kompilere.



Figur 4-46 – Prosjekt opprettet vha Wizarden.

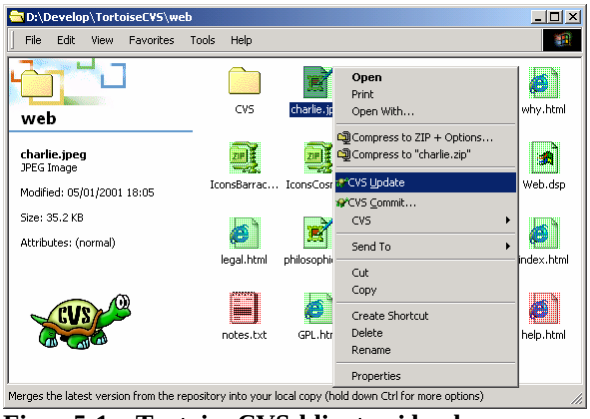
5 Kvalitetssikring og testing

For å sikre kvaliteten av arbeidet mens vi har jobbet, har versjonsstyring, backup, veileder- og oppdragsgivermøter og samtaler eller møter oss imellom vært svært viktig. På veileder- og oppdragsgivermøter har vi fått respons på det vi har utviklet eller skulle utvikle, som vi har tatt med oss videre. Det at vi alltid har diskutert oss imellom før vi har gått i gang med nye deler av applikasjonene, har også ført til liten tvil om hvilke arbeidsoppgaver den enkelte har hatt.

5.1 Versjonsstyring

Fra tidligere prosjekter har vi erfaring med manglende versjonsstyring. Det har ført til problemer underveis, og vi gikk derfor inn for å benytte oss av et versjonsstyringssystem til hovedprosjektets kildekode.

I hovedsak vurderte vi to alternative løsninger, Microsoft Visual SourceSafe og CVS. Microsoft Visual SourceSafe kan enkelt integreres i Visual Studio, men det finnes også forskjellige plugin som integrerer CVS-løsninger på samme måte. Valgt falt til slutt på CVS, først og fremst fordi dette er mest brukt og fordi det er lettere å jobbe på samme fil med CVS. Siden begge maskinene brukt i prosjektperioden har hatt Windows XP som operativsystem, falt valget av CVS-server på CVSNT [3], og vi satte opp CVS-serveren på den ene utviklingsmaskinen. Når det kom til valg av klient fant vi to alternativer, WinCVS og TortoiseCVS [10]. Valget falt på TortoiseCVS-klienten siden dette virket som et enkelt verktøy å bruke. TortoiseCVS-klienten er en CVS-klient som integreres i Windows Explorer. Dette gjør den veldig enkel å bruke vha. høyreklikksmenyer og ikon-overlay som vist i Figur 5-1.



Figur 5-1 – Tortoise CVS-klienten i bruk

5.2 Backup

Siden tap av data i et prosjekt av denne størrelsen er meget kritisk, valgte vi å ta jevnlig backup av CVS-filstrukturen. På CVS-serveren finnes all kildekode som er blitt produsert gjennom prosjektperioden, samt historikken med endringskommentarer. Innholdet på CVS serveren har blitt kopiert hver dag kl 00.00 og kl 12.00 til en maskin utenfor Høgskolens område. For å overføre filene brukte vi PuTTy Secure Copy Client [9], som er en console-applikasjon som overfører data over SSH(2) protokollen. Vha. Microsoft

Scheduled Task ble klienten kjørt til riktige tider og med innlogging som kommandolinjeparametere.

Når filene hadde blitt overført til serveren ble filene igjen kopiert videre inn i en katalogstruktur som innholder dato og klokkeslett for når backupen ble tatt vha. et perl-script. Denne måten for backup har sikret oss to sikkerhetskopier fra hver dag gjennom hele prosjektperioden. Noen ganger har CVS-serveren vært avslått eller på en annen måte utilgjengelig, som vil si at vi ikke har backup for akkurat disse dagene. Serveren har aldri vært avslått i lengre perioder da det har vært risiko for store tap.

I tillegg til automatisk backup som hittil nevnt, og at begge maskinene har hatt en kopi av kildekoden (uten historikk), har vi som regel tatt ukentlig manuell backup av CVS-serveren og all dokumentasjon til våre maskiner hjemme og hjemmeområder på HiG.

All form for dokumentasjon i forbindelse med prosjektet ble plassert på prosjektets hjemmeområde administrert av IT-tjenesten[8] v/HiG. Det har jevnlig blitt tatt backup av dette området i regi IT-tjenesten. Mot slutten av prosjektet har det blitt tatt daglig manuell backup av all dokumentasjon.

5.3 Testing

Under testning er både white-box- og black-box-testning gjennomført. Forskjellen på white og black er om testeren har innsikt i oppbygningen av applikasjonen. Når man tester et system eller en applikasjon etter black-box-prinsippet, betyr dette at man tester med tanke på input og output. Testeren har altså ingen anelse om hva systemet eller applikasjonen spesifikt gjør, men kan sjekke hva resultatet av for eksempel en funksjon i systemet eller applikasjonen resulterer i. White-box-testing derimot bygger på prinsippet om at brukeren har innsikt i kildekoden, og da kan utføre testingen på en slik måte at alle mulige kodeveier blir forsøkt. Hvis brukeren finner en feil kan brukeren vite nøyaktig hvor feilen oppstod og kanskje hvordan feilen eller problemet skal løses.

5.3.1 White-box

White-box testing har blitt utført av utviklerne selv. Altså av vi som har skrevet denne rapporten. Testing har blitt utført etter at hver enkelt del har blitt fullført. For eksempel etter endt utvikling av slideshowgeneratoren foretok vi en test av applikasjonen der vi testet alle funksjoner som hittil var ferdig utviklet. Hver funksjon ble grundig testet på våre system, og alle funksjoner som ikke fungerte som ønskelig ble notert og senere utbedret. Underveis har også mindre enkeltkomponenter blitt testet, på en lignende måte, etter hvert som disse har blitt implementert.

5.3.2 Black-box

Black-box testing har blitt utført av en liten gruppe personer som har sagt seg villige til å teste applikasjonen. Disse personene har hver for seg, og uten innblanding fra utviklerne testet applikasjonen. Alle testpersonene fikk utdelt et spørreskjema (vedlegg H) sammen med applikasjonen, der all applikasjonens funksjonalitet var listet opp sammen med et

[8] IT-Tjenesten administrerer alt som har med HiGs datamaskinpark.

felt med plass til å skrive om funksjonen fungerte tilfredsstillende, eller hva som eventuelt var feil. I tillegg ble testpersonene oppfordret til å komme med tilbakemelding på applikasjonens utseende, brukervennlighet og eventuelt andre feil, mangler eller andre kommentarer.

5.3.3 Resultat av brukertesting

Da disse spørreskjemaene ble returnert fra testerne, viste det seg at flere av dem hadde støtt på enkelte problemer. En stor del av disse problemene skyldtes at testerne ikke hadde installert nyeste versjon av DirectX, men også en del andre problemer. Et eksempel på dette er ProgressDialogen som vises under tidkrevende operasjoner i applikasjonen. Under white-box-testingen oppdaget vi at denne dialogen enkelte ganger ble stående igjen når operasjonen var ferdig, og altså ikke lukket seg automatisk, slik som den burde. Denne feilen ble så utbedret, men som nevnt, da vi fikk tilbake spørreskjemaene fra testerne viste det seg at flere av dem også hadde opplevd dette fenomenet. Dette førte til at vi måtte se på problemet en gang til, og denne gangen mener vi at det er løst riktig.

Ved å få andre til å utføre black-box-testing med andre systemoppsett enn våre, erfarte vi at det fantes en del feil og mangler vi ikke hadde funnet på våre system. Våre betatestere utførte handlinger som fremprovoserte feil vi ikke hadde forutsett, noe som var helt i tråd med planen. Det virket som om de fleste fikk de samme feilene, som gjorde at vi kunne konsentrere oss om nettopp disse feilene den siste tiden som var igjen. I tillegg fikk vi mange gode råd og tips om hva som var bra, og hva som kunne vært bedre. I vedlegg H finnes resultatet fra alle betatesterne som deltok i black-box-testingen.

6 Diskusjon av resultater

Prosjektet har resultert i en fungerende applikasjon som inneholder alle de hovedelementene som ble skissert under kravspesifiseringen. Muligheten for at det fortsatt finnes noen feil, eller noen deler av applikasjon ikke fungerer på forskjellig maskinvare er selvfølgelig tilstede, siden vi har hatt begrenset tid til retting av feil funnet ved testing.

6.1 Drøfting av resultater

I kravspesifikasjonen står det oppført flere punkt i Bilderedigeringsverktøy under avsnitt 2.3.1 som ikke er ferdig utviklet. Verktøyene det er snakk om er som kjent mulig å utvikle senere som plugin og er opplistet i Tabell 6-1. Disse og andre plugin vil i samarbeid med oppdragsgiver bli utviklet senere. Bakgrunnen for at de ikke er utviklet er tidsmangel. Vi prioriterte testing av applikasjonene over utvikling av nye plugin.

Tabell 6-1 – Bilderedigeringsverktøy som ikke ble utviklet.

Legge på tekst
Mulighet for å legge "clip-art"/små bilder over
Auto-correct (forbedre kontrast, farger og hvitpunkt)
Kontrast/lysstyrke
Fargebalanse
Retusjering

Om DirectX har vært nødvendig i visning av et slideshow kan helt klart diskuteres. Det hadde vært enklere å gjøre alt med .NET rammeverkets grafikklasser og dermed ikke brukt DirectX i det hele tatt, men i den forbindelse hadde vi mistet muligheten for tredimensjonale overganger. DirectX gjør også at fremviseren ikke er så rask. Dette fordi vi først oppretter et Bitmap-objekt, for deretter å generere en tekstur ut i fra dette objektet. Det betyr at maskinen må allokere minne og kopiere minnet til skjermkortet, noe som enkelt kunne vært unngått ved å kun bruke .NET rammeverket og altså kun Bitmap objektet. Enda en negativ ting rundt DirectX er at Managed DirectX versjonen vi bruker ikke er gitt ut som ”End-User Runtime”. Den offisielle versjonen Microsoft har utgitt, er i skrivende stund DirectX 9.0b build 5.03.0000.0900. Allikevel har Microsoft gitt ut ”Summer update 2003” av SDK pakken. Denne pakken benytter build 5.03.0001.1126, som veldig få ikke-utviklere har. Microsoft har utgitt en retail-versjon av DirectX 9.0b (Summer Update 2003). Dette er en versjon spillutviklere kan sende med sine spill på CD. Retail-versjonen er på hele 35MB komprimert. Fremviserapplikasjonen som også må bruke denne versjonen kan ikke ha med 35MB bare fordi brukeren som skal se slideshowet kanskje ikke har den nyeste versjonen av DirectX. Så dette er et stort problem som helt sikkert ikke vil løse seg før Microsoft gir ut neste major release av DirectX. Nemmelig DirectX 10.

Alle .NET-applikasjoner kjøres i *CLR*[9], et innkapslet og styrt miljø der applikasjonen kjører atskilt fra andre prosesser på maskinen. CLR er en virtuell maskin, og denne

[9] CLR – Common Language Runtime

virtuelle maskinen gir oss mange fordeler som automatisk minnehåndtering, og dersom programmet krasjer skal ikke dette påvirke resten av maskinen og de andre prosessene som kjøres. Tilstedeværelsen av denne virtuelle maskinen gjør også at applikasjonen vil kjøre noe tregere. Det også en ulempe at man må ha installert .NET rammeverket versjon 1.1 for å kjøre både slideshowgeneratoren og fremviserapplikasjonen.

Vi har utviklet en applikasjon for generering og visning av slideshow, med mye funksjonalitet, bl.a. innebygd bilderedigeringsprogram. Systemet er meget fleksibelt, både med hensyn til brukerinnstillinger og at ny funksjonalitet kan legges til som plugin, som dynamisk lastes under oppstart av applikasjonen. Vi har selv utviklet og skreddersydd en rekke av komponentene i det grafiske brukergrensesnittet. På grunn av bruken av DirectX er det mulig å lage tredimensjonale overganger mellom bildene. Applikasjonen kan lagre et slideshow til en kjørbar fil, også kalt fremviserapplikasjonen, som kan kjøres uavhengig av slideshowgeneratoren og altså spille av slideshowet som er generert. Det var et viktig poeng at dette skulle kunne lagres som én fil, slik at man ikke er avhengig av å komprimere fremviseren og slideshow-data for å få sendt dette til for eksempel familie og venner.

Det må også til forsvar for .NET nevnes at vi ved hjelp av Microsoft Visual Studio .NET 2003, C# og .NET har utviklet mye på kort tid. Antagelig mye mer enn om vi skulle ha valgt et av de andre programmeringsspråkene vi vurderte.

6.2 Utvidelse og forbedringsmuligheter

Noe veileder tidlig påpekte, og som vi etter hvert ble klar over, er at det er svært vanskelig å skape et 100 % ferdig, fullt funksjonelt og gjennomtestet produkt, klart til lansering i løpet av den tilmålte prosjektperioden. Allikevel har vi utviklet en prototyp der alle hovedelementene er på plass, og det er lagt til rette for videreutvikling av ytterligere funksjonalitet.

6.2.1 Plugin

Det er lagt til rette for at ny funksjonalitet skal kunne legges til som nye plugin. Dette gjør at det enkelt kan legges til ny funksjonalitet, slik at de som lager denne funksjonaliteten kun trenger å sette seg inn i plugintypens oppbygning og virkemåte. I tillegg kan disse pluginene distribueres til brukere som allerede har applikasjonen, ved å plassere pluginet i riktig katalog. Pluginet vil da bli lastet automatisk neste gang applikasjonen startes. Vi har også utviklet en wizard for Visual Studio som forenkler arbeidet med å utvikle nye plugin.

6.2.2 Ikoner

I samråd med oppdragsgiver ble det tidlig i utviklingsfasen besluttet av vi ikke skulle bruke mye tid på design av ikoner til applikasjons menyer, knapper osv. Isteden skulle vi lage raske midlertidige løsninger eller ”låne” ikoner fra annen programvare, slik at vi kunne konsentrere oss om å utvikle selve applikasjonen fremfor å tegne ikoner. Dette medfører at alle applikasjonens ikoner må byttes ut før en eventuell kommersiell lansering av produktet foreligger.

6.2.3 Hjelpesystem

Det søkbare hjelpesystem som er integrert i applikasjonen ble i samråd med oppdragsgiver gitt liten prioritet. Det ble bestemt at vi ikke skulle bruke tid på å legge til mer innhold enn det mest elementære. Hjelpesystemet kan enkelt utvides senere ved å benytte fritt tilgjengelig programvare [4] og enkle HTML-dokumenter.

6.3 Evaluering av eget arbeid

Vi har jobbet jevnt gjennom hele prosjektperioden, og selv om vi har møtt på flere problemer eller utfordringer underveis i prosjektet, har ingen av dem vært umulig å løse. Det største problemet har utvilsomt vært tidsbegrensningen.

Fremdriftplanen (vedlegg I) har ikke blitt fulgt slavisk gjennom hele prosjektperioden, men brukt som en oversikt over hvilke arbeidsoppgaver som til enhver tid skulle ha vært fullført eller i gang, og hvilke som gjenstår. Noen inkrement, spesielt i starten av prosjektet, tok lengre tid enn planlagt, mens noen av de siste inkrementene gikk mye raskere enn planlagt. I tillegg er noen inkrement kjørt parallelt, istedenfor sekvensielt, siden vi fant det hensiktsmessig at én person hadde hovedansvaret for fremviseren. Siden vi startet selve utviklingen noe tidligere enn først planlagt har vi også fått noe mer tid til testing og rapportskriving til slutt. Vedlegg J er et revidert ganttskjema som viser faktisk fremdrift.

6.3.1 Organisering

Som nevnt i avsnitt 1.2.2 (Ansvarsforhold) har ikke prosjektleder vært nødt til å bestemme noe ved at prosjektleder har hatt siste ord i eventuelle uoverensstemmelser. Vi har som regel diskutert og til slutt blitt enige, i de tilfeller det i det hele tatt har vært noe å diskutere. Statusrapporter (vedlegg K) har blitt skrevet i fellesskap, og prosjektleder har vært kontaktperson mot oppdragsgiver og veileder. I tillegg har vi hatt forskjellige ansvarsområder, som beskrevet i avsnitt 1.2.2. Dette har fungert utmerket gjennom hele prosjektperioden.

6.3.2 Fordeling av arbeidet

Vi har gjennom hele prosjektperioden fortløpende fordelt arbeidet mellom oss, slik at vi har jobbet med forskjellige deler hver for oss. Dette er en arbeidsform vi har benyttet også i tidligere prosjekt, og det fungerte meget bra denne gangen også. Enkelte krevende eller kritiske oppgaver er blitt løst i fellesskap. Ved slutten av hver arbeidsdag har det blitt ført logg over hva som er utført, og hvor mange timer som har gått med til dette. Denne loggen finnes i vedlegg L, og har gjennom hele prosjektet vært tilgjengelig for oppdragsgiver og veileder via prosjektets hjemmeside. Møtereferater (vedlegg M) har også på samme måte vært tilgjenglige her. Slik har oppdragsgiver og veileder(e) hatt mulighet til å følge prosjektets fremgang dag for dag.

6.3.3 Prosjekt som arbeidsform

Hovedprosjektet er noe vi har sett frem til, og håpet skulle bli en spennende og lærerik opplevelse. Siden prosjektarbeid er en vanlig arbeidsform ute i arbeidslivet, er dette en nyttig erfaring å dra med seg.

Gjennom våre tre år på HiG har det vanlige undervisningsopplegget vært forelesninger, øvingstimer og noen mindre prosjekt. Disse prosjektene har alle hatt et omfang og en arbeidsmengde som er langt mindre enn et hovedprosjekt. Vi ante lite om hva som ventet oss og hvor stor arbeidsmengde vi egentlig la opp til i forhold til disse tidligere erfaringene.

6.3.4 Subjektiv opplevelse av hovedprosjektet

En svært motiverende faktor for oss har vært at vi skulle utvikle et produkt. Dette har fått oss til å tenke svært mye på hvordan det er å utvikle noe som kan selges, og noe som altså må fungere for andre enn oss selv. Vi har gjort oss svært mange gode erfaringer med å la andre teste for oss, noe vi har satt stor pris på. Ved betatestingen kom det fram mange feil og mangler pga. at distribusjonen av applikasjonene til betatesterne ikke var vellykket. Vi forstod med en gang hvilken utfordring det egentlig er å distribuere programvare som benytter seg av eksterne bibliotek og andre komponenter.

Oppdragsgiver er svært opptatt av utseende, grafiske brukergrensesnitt og hvilken plassering forskjellige GUI-komponenter bør ha, for å gjøre det mest mulig intuitivt for brukeren. Dette dro vi nytte av tidlig i prosjektperioden, da vi i samarbeid med oppdragsgiver utformet applikasjonens GUI. Derimot har veileder vært mer interessert i applikasjonens underliggende struktur, oppbygning og virkemåte og har kunnet komme med innspill her. På denne måten har oppdragsgiver og veileder utfylt hverandre bra.

Som tidligere nevnt har vi benyttet oss av en Windows-basert CVS-server [3]. Dette har ikke vært helt problemfritt. Forøvrig har serveren også vært brukt som utviklingsmaskin uten at dette skulle ha noen betydning. Problemet har vært at CVS-brukerne blir hentet ut i fra ”Windows-brukerne”, noe som har ført til uregelmessigheter i brukerautentifikasjonssystemet (LSASS.EXE). Resultatet har vært en svært unormal feil som ”fryser” hele systemet. Eneste mulighet til å gjenopprette Windows til operasjonell status har vært å avslutte alle applikasjoner om mulig, ”dra ut strømmen” for så å starte på nytt. I perioder da vi utviklet svært mye og dermed benyttet CVS-klienten ofte for å hente ned siste versjon, legge til nye filer eller oppdatere filer, var det svært irriterende å måtte restarte maskinen flere ganger om dagen. Vi hadde selvsagt muligheten til å benytte en annen server, på et annet operativsystem, men fant aldri tid til dette. Uansett har det vært en erfaring vi vil ta med oss videre.

Halvveis inn i prosjektet lå vi et par uker etter fremdriftsplanen. Som beskrevet i andre statusrapport (vedlegg K) ble det besluttet å ikke gjennomføre utvikling av en installasjonsapplikasjon. Senere fant vi ut at en installasjonsapplikasjon ville være en stor fordel under betatestingen. Siden det også viste seg at dette ville ta relativt kort tid å implementere, ble det besluttet å utvikle allikevel. Under betatestingen viste det seg at installasjonsapplikasjonen ikke fungerte tilfredsstillende, da DirectX burde installeres uansett. Dette har siden blitt utbedret.

Selv om vi har hatt enkelte problemer underveis, og det har blitt mange lange arbeidsdager, har hovedprosjektet vært en positiv og lærerik opplevelse.

7 Konklusjon

Gjennom et halvt års prosjektarbeid har vi hatt muligheten til å følge applikasjonen fra idé til en ferdig fungerende, dokumentert og testet applikasjon. Dette har stilt store krav til planlegging og evne til å foreta veloverveide beslutninger.

Vi har lært mye av dette prosjektet, først og fremst rent programmeringsmessig, men også en del om systemutvikling og hvordan det er å jobbe på, og ha ansvaret for, et større prosjekt. Vi har blant annet blitt godt kjent med C# og .NET, som ingen av oss hadde noen erfaring med fra før.

Vi er selv godt fornøyd med resultatet vi endte opp med, hvis vi ser bort i fra problemene rundt DirectX og avhengigheten av .NET rammeverket. Om oppdragsgiver vil videreutvikle applikasjonen vil vi anbefale at fremviseren blir skrevet om, slik at DirectX byttes ut med standard Windows GDI+. Uansett håper vi resultatet av dette prosjektet vil være til nytte for oppdragsgiver, enten i form av selve applikasjonen eller i form av rapporten og de erfaringer han sitter igjen med.

8 Referanser

[1] Ian Sommerville. Software Engineering. Addison-Wesley, 2001.
Systemutviklingsbok, også brukt i faget systemutvikling ved HiG.

[2] Adobe Photoshop. http://www.adobe.com/products/photoshop/

[3] CVSNT, CVS-server for Windows. http://www.cvsnt.org/wiki/

[4] HTML Help Workshop 1.3. http://msdn.microsoft.com/workshop/author/htmlhelp/

[5] Microsoft http://www.microsoft.com/

[6] Microsoft Project. http://www.microsoft.com/office/project/

[7] Microsoft Word. http://www.microsoft.com/office/word/

[8] NDoc http://ndoc.sourceforge.net/wiki/HomePage/

[9] PuTTy Secure Copy Client http://www.chiark.greenend.org.uk/~sgtatham/putty/

[10] TortoiseCVS http://www.tortoisecvs.org/

[11] .NET Naming Guidelines
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconnamingguidelines.asp

[12] Hungarian Notation.
http://msdn.microsoft.com/library/en-us/dnvsgen/html/HungaNotat.asp

[13] Doxygen. http://www.doxygen.org

[14] Microsofts kommenteringsstandard.
http://msdn.microsoft.com/msdnmag/issues/02/06/xmlc/default.aspx

[15] Kings Tools, Doxygen-plugin for Visual Studio.
http://www.codeproject.com/macro/KingsTools.asp

[16] John Sharp, Jon Jagger. Microsoft Visual C# .NET, step by step. Microsoft Press, 2003.
Rask gjennomgang av grunnleggende programmering I
i C# og bruk av Visual Studio.

[17] Ben Albahari, Peter Drayton, Brad Merrill. C# Essentials. O´Reilly, 2002.
Oppslagsverk som inneholder kort informasjon om all C#-syntax.

[18] Tom Archer. Programmering i C#. Gyldendal, 2002.
Norsk innføring i C# og .NET utvikling.

[19] Tom Miller. Managed DirectX 9, Graphics and Game Programming. Sams, 2004.
Innføring i Managed DirectX 9 av Tom Miller, lead developer for Managed DirectX API.

[20] Mason McCuskey. Special Effects Game Programming with DirectX. Premier Press, 2002.
Innføring i spesialeffekter i DirectX, rettet mot C++.

[21] Sortering av ListView. http://support.microsoft.com/default.aspx?scid=kb;EN-US;319401

[22] Laura Arlov. GUI Design for Dummies. IDG Books, 1998.
En bok om design av grafiske brukergrensesnitt.

[23] SharpZipLib, komprimeringsbibliotek.
http://www.icsharpcode.net/OpenSource/SharpZipLib/

[24] Windows Installer (MSI).
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/windows_installer_start_page.asp

[25] Businessware Architects
http://home.telkomsa.net/businessware/