# Sammendrag

Dette dokumentet er sluttrapporten til Bård Terje Fallans hovedoppgave våren 2005 ved Institutt for Datateknikk og Informasjonsvitenskap, NTNU.

Målet med oppgaven har vært å forbedre den eksisterende modellen av Sverresborgen, ved å gjøre det mulig å innlemme stedfestet lyd fra de virtuelle kildene inne i modellen. Hensikten med dette har vært å øke realismen av opplevelsen til brukeren under demonstrasjoner av modellen i egnede omgivelser.

Ulike metoder for å gjengi stedfestet lyd er gjennomgått og vurdert. Dessuten er det sett på ulike måter å benytte disse teknikkene i den eksisterende applikasjonen. En metode er valgt og implementert som viser at dette er mulig uten å gjøre dyptgående endringer i strukturen til systemet som finnes i dag.

Det har vært ønskelig å gjøre applikasjonen så generell som mulig, noe som i stor grad har lykkes. Dette er i tråd med det eksisterende systemet laget av Kundestyrt Prosjekt Gruppe 13 høsten 2003 [KPO01].

Dessuten har det parallelt med dette vært et nært samarbeid med Sverresborg Folkemuseum om en utstilling sommeren 2005. Dette samarbeidet har resultert i en guidet animasjon på borgen, med tilhørende stedfestede lydeffekter.

# Forord

Denne rapporten er resultatet av Bård Terje Fallans hovedoppgave ved Institutt for Datateknikk og Informasjonsvitenskap, NTNU.

Tidligere prosjekter ved instituttet har resultert i en tredimensjonal virtuell modell av middelalderborgen Sverres borg i Trondheim. Målet har vært å se på hvordan denne eksisterende applikasjonen kan utvides til å også være i stand til å inneholde stedfestet lyd. Hensikten med dette har vært å øke graden av realisme og tilstedeværelse i simuleringen.

Prosjektet er gjennomført våren 2005. Det har vært et spennende, men tidkrevende arbeid. Det at prosjektet har interesser også utenfor universitetsmiljøet, har vært en positiv motivasjonsfaktor og har gitt innblikk også i andre interessante prosesser enn kun de rent tekniske i forbindelse med et slikt samarbeid.

Jeg vil gjerne takke veileder Torbjørn Hallgren for et godt samarbeid gjennom semesteret. Han har bidratt med gode og konstruktive tilbakemeldinger, og en hjelpende hånd i samarbeidet med Sverresborg Folkemuseum.

Det rettes også en takk til Jo Skjermo og Karstein Kristiansen ved IDI for assistanse med de mer praktiske utfordringene som oppsett av visningsutstyr og tilgang til ulike ressurser ved NTNU.

Det har under arbeidet med oppgaven vært et nært samarbeid med medstudent Espen Almdahl, blant annet i forbindelse med arbeidet med museumsutstillingen, og jeg vil derfor takke ham for et godt samarbeid. Vi har dessuten vært i kontakt med flere ansatte ved Sverresborg Folkemuseum, og vil takke disse for deres imøtekommenhet og gode forslag til utvidelser av funksjonalitet til den eksisterende Sverresborgmodellen. Spesielt vil jeg takke arkeologistudent Regin Meyer for det gode samarbeidet gjennom semesteret.

NTNU, Trondheim 15.06.2005.

Bård Terje Fallan

# Innholdsfortegnelse

# Figurliste



# Tabeller

# Liste over forkortelser

| | |
|---|---|
| API | *Application Programming Interface* |
| BEM | *Boundary Element Method* |
| CAVE | *Cave Automated Virtual Environment* |
| DSP | *Digital Signal Processing* |
| EAX | *Environmental Audio Extension* |
| FEM | *Finite Element Method* |
| HRIR | *Head-Related Impulse Response* |
| HRTF | *Head-Related Transfer Function* |
| IDT | *Interaural Delay Time* |
| IED | *Interaural Envelope Difference* |
| ILD | *Interaural Level Difference* |
| IRIX | *Operativsystem for SGI-maskiner* |
| ITD | *Interaural Time Difference* |
| OpenAL | *Open source Audio Library* |
| OpenGL | *Open source Graphic Library* |
| PCM | *Pulse Code Modulation* |
| RAVE | *Reconfigurable Advanced Visualization Environment* |
| RIR | *Room Impulse Response* |
| SGI | *Silicon Graphics Incorporated* |
| VBAP | *Vector Base Amplitude Panning* |
| VR | *Virtual Reality* |
| VSS | *Virtual Sound Server* |
| WAV | *Waveform audio file* |

# 1 Innledning

Virtuelle omgivelser har blitt stadig mer populært de siste årene. Tilgangen til stadig kraftigere grafisk hardware har gjort det mulig å drive virtuell virkelighetsapplikasjoner på alt fra hjemmedatamaskiner til romlig omhyllende omgivelser som for eksempel CAVE. Forskning innen dette feltet har hovedsakelig vært fokusert på stadig mer realistisk grafikk og ulike interaksjonsformer mellom brukeren og omgivelsene [NSG01]. I takt med det voksende kravet om økt realisme, har lyd etter hvert blitt sett på som en viktig del av opplevelsen. Den viktige rollen lyd spiller for opplevelsen av en scene, kan eksempelvis illustreres med å se en film med og uten lyd. Spesielt spenningsfilmer mister en stor del av intensiteten hvis lyden blir borte.

For systemer der grafikken blir vist i omsluttende omgivelser med flere bakprojiserte vegger og opplevelsen av å være til stede inne i den virtuelle modellen er stor, er det neste naturlige steget å også inkludere lyd til denne opplevelsen. Spesielt kan stedfestet lyd ved bruk av flere høyttalere plassert på hensiktsmessige steder rundt observatøren, gi den ekstra følelsen av å være ”til stede”.

For temaområdet virtuell kulturell arv, blir spørsmålet om hvilke lyder som skal kunne høres i modellen viktig. Hensikten med denne typen modeller er som regel å gjenskape en mest mulig realistisk og historisk korrekt modell, og det vil derfor være viktig å ikke innføre ”nye” eller kunstige lyder som ikke var til stede på det aktuelle historiske tidspunktet. Dette må likevel balanseres mot ”opplevelsen” man kan skape ved å bruke lyd på nye og interessante måter. Lyd har evnen til å skape ulike sinnsstemninger hos lytteren, og bakgrunnsmusikk kan eksempelvis gi en ekstra dimensjon som kan sette lytteren i en avslappet stemning eller motsatt, i en spent og ivrig stemning.

Lyd er ikke bare et tilegg eller en overflødig detalj, men kan i seg selv gi viktig informasjon som er vanskelig å visualisere på andre måter. Synssansen har begrenset kapasitet til å ta inn og prosessere flere konkurrerende inntrykk samtidig, og bruk av hørselssansen vil derfor kunne øke graden av informasjon som brukeren kan ta inn på et gitt tidspunkt. Dette poenget har spesielt blitt utnyttet innen ulike felter for visualisering av store datamengder, for eksempel innen oljebransjen. En lydkilde trenger ikke nødvendigvis å eksistere i som et fysisk objekt, men kan være knyttet til en spesiell datatype og variere i styrke og tonehøyde med dennes ulike verdier. Dette gjør brukeren i stand til å ta inn mer informasjon enn det ville vært mulig hvis kun synssansen var involvert.

Ved Institutt for Datateknikk og Informasjonsvitenskap ved NTNU har det gjennom flere ulike studentprosjekter blitt utviklet en virtuell rekonstruksjon av Sverres borg i Trondheim slik den så ut rundt 1100-tallet. Denne modellen har utviklet seg både grafisk, med stadige forbedringer av selve modellen, men også på det rent tekniske planet. Dette arbeidet har resultert i en virtuell modell der det er mulig å bevege seg ved bruk av spesielle innenheter. Til tross for stadig forbedringer av modellen, har det frem til nå ikke vært mulig å legge til lyder og lydkilder i simuleringen. Dette behovet har etter hvert fått høyere prioritet i takt med at ønsket om å vise frem modellen for et større publikum har blitt mer aktuelt.

Det ble høsten 2004 inngått et samarbeid med Sverresborg Trøndelag Folkemuseum om en utstilling av Sverresborgmodellen sommeren 2005. Arbeidet med dette kom ikke skikkelig i gang før våren 2005. Både museet og arkeolog Regin Meyer hadde i denne forbindelse ønsker om endringer og tilegg til den eksisterende modellen slik den så ut på det daværende tidspunktet. Foruten endringer av den eksisterende modellen for å gjøre den mer historisk og arkeologisk korrekt, var det også ønske om å inkludere animerte menneskelige modeller som skulle utføre sitt daglige virke på borgen, samt en guide som skulle lede publikum gjennom borgen og forklare hva man til enhver tid så.

Arbeidet med denne utstillingen er derfor en del av oppgaven, men vil likevel være forholdsvis vesensforskjellig fra det å implementere stedfestet lyd i den eksisterende Sverresborgapplikasjonen. På grunn av dette vil dette arbeidet bli dokumentert separat i denne rapporten.

## 1.1 Oppgaveformulering

Oppgaven er å se på ulike metoder for å integrere stedfestet lyd i en virtuell geometrisk omgivelse, både ved bruk av to eller flere lydkilder (høyttalere) for å øke opplevelsesverdien for brukeren og for å fremheve detaljer og informasjon i modellen. Testing utføres ved å implementere en demonstrasjon i den eksisterende virtuelle modellen av Sverresborgen.

Det taes sikte på at demonstrasjonen skal kunne brukes under sommerutstillingen 2005 ved Sverresborg Trøndelag Folkemuseum. I dette inngår å se på ulike teknikker og metoder for å på en best mulig måte skape en publikumsvennlig attraksjon.

## 1.2 Avgrensning

Det vil kun bli implementert én mulig løsning i den eksisterende Sverresborgmodellen. Valget av hvilken metode som blir implementert, vil likevel være begrunnet i den teoretiske delen av oppgaven og skal i størst mulig grad representere den beste løsningen for det gitte formålet.

## 1.3 Om oppbygningen av rapporten

Kapittel 2 presenterer en del bakgrunnsinformasjon og teori om lyd generelt og stedfestet lyd i datasimuleringer spesielt. Kapittelet gir dessuten en oversikt over ulike metoder som tidligere har vært benyttet for å implementere lyd i virtuell virkelighetsapplikasjoner.

I kapittel 3 blir de ulike alternativene som ble skissert i det forrige kapittelet vurdert, og et valg blir gjort for hvordan denne løsningen kan implementeres i den eksisterende modellen av Sverresborgen.

Kapittel 4 beskriver så hvordan dette faktisk ble implementert på det eksisterende systemet, mens kapittel 5 oppsummerer resultatet av arbeidet som har blitt gjort, og presenterer de oppnådde resultatene.

Kapittel 6 konkluderer rapporten og skisserer hvilke fremtidige utvidelser og forbedringer som vil være mest nærliggende for eventuelle videre arbeider på modellen.

På grunn av at arbeidet knyttet til utstilling av modellen på Sverresborg Folkemuseum som nevnt er vesensforskjellig fra resten av oppgaven, er det valgt å presentere dette i sin helhet under kapittel 4 (implementasjonskapittelet). Valg av ulike teknikker, skissering av hva som er blitt gjort etc. vil derfor kunne finnes igjen der.

# 2 Metoder

Dette kapittelet inneholder en del bakgrunnsinformasjon og forklaring av konsepter som er relevante for generering av stedfestet lyd. Kapittel 2.1 beskriver hvordan mennesker kan oppfatte lyder i omgivelsene og stedfeste disse basert på flere ulike og kompletterende metoder. I forbindelse med at lyd etter hvert har fått en større interesse blant utviklere av grafikkapplikasjoner, har det kommet flere såkalte lyd-API'er med formål å lette integreringen av lyd i disse applikasjonene. Det vil derfor videre bli gitt en oppsummering og vurdering av en del alternative løsninger som vil være mest aktuelle i dette tilfellet. Disse API'ene letter arbeidet til utvikleren ved at de skaper et felles grensesnitt mot selve maskinvaren som genererer lydeffektene. De har også en mer eller mindre godt innebygd mulighet for å gjenskape romlig stedfestet lyd.

Kapittelet avsluttes med en mer spesifikk betraktning av hvordan lyd kan innlemmes i den eksisterende Sverresborgmodellen. Dette innebærer en vurdering av hvilke lyder som faktisk vil være relevante (bakgrunnslyder, menneskelige lyder etc.) og hva som kreves rent teknisk av en slik løsning.

## *2.1 Stedfestet lyd*

Dette kapittelet vil gjennomgå ulike metoder for å gjenskape naturlig stedfestet lyd i en virtuell omgivelse inne i en datamaskin. Under dette inngår teknikker for hvordan man kan skape inntrykk av lyder som kommer fra ulike retninger i rommet, men også modellering av romklang for å skape et bedre inntrykk av omgivelsene brukeren befinner seg i.

### 2.1.1 Motivasjon

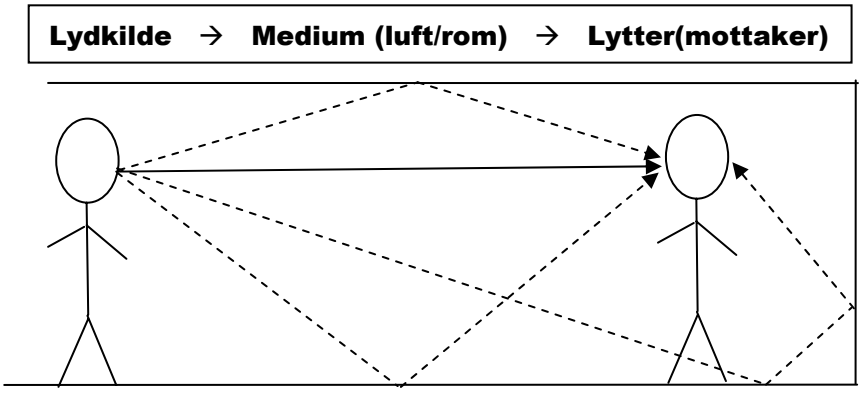
Stedfestet lyd er ikke bare et ønske fra underholdningsindustrien for å skape et mest mulig intenst inntrykk for brukeren i disse sammenhengene, men har også en mer generell og "nyttig" funksjon i mange av dagens industrielle datasimuleringer. Hånd i hånd med at stadig mer regnekraft har blitt tilbudt fra maskinvareprodusentene, har mer realistiske modeller for lydgjengivelse blitt utviklet. Denne selvforsterkende syklusen har gjort det mulig å realistisk simulere effekten av lyd i en geometrisk omgivelse, før denne faktisk har blitt realisert i den fysiske virkeligheten. Nye operabygninger og konsertsaler blir i dag bygget opp virtuelt inne i datamaskinen, og deres lydmessige egenskaper kan på denne måten testes og justeres før en starter på den kostbare byggeprosessen. Også eksisterende rom kan gjenskapes virtuelt for å bedre kunne se effekten av å endre ulike fysiske parametere ved rommets oppbygning, for eksempel plassering av lydisolerende plater.

Det virker naturlig å anta at metoder og verktøy som er utviklet for simulering av romakustikk, også kan brukes for å lage virtuelle lydomgivelser som ikke er basert på virkelige omgivelser, og også for å lage (kunstige) stedfestede lydeffekter [KDS01]. Dette er spesielt relevant for virtuell virkelighets- og multimediaapplikasjoner, der lyd etter hvert spiller en viktig rolle.

Det er et overlappende interesseområde mellom å gjenskape lyd i en virtuell modell (både i forsknings- og underholdningsøyemed) og forskningen på lyds fysiske egenskaper generelt. Det er nærliggende å tro at desto bedre en kan modellere de fysiske betingelsene for lyd, desto bedre vil også den virtuelle simuleringen bli. Lyd og hørselens fysiske egenskaper er etter hvert forholdsvis godt undersøkt og forstått, og det er derfor i teorien mulig å gjenskape et naturtro lydbilde ved hjelp av datamaskinen. Det er i denne sammenheng fristende å trekke en parallell til lys- og bølgeforskning og datagrafikk. Også på dette feltet er det fysiske grunnlaget godt dokumentert, blant annet gjennom Kajiyas *rendering equation* [KJ01]. Problemet er at det er umulig å regne seg frem til en løsning på dette problemet på grunn av begrensningene i regnekraften til datamaskinen. En må derfor basere seg på teknikker og metoder som tilnærmer den eksakte løsningen. Hvor gode (det vil si naturtro) disse tilnærmingene er, vil avhenge av hvor godt man er i stand til å finne modeller som tar flest mulig fysiske betraktninger med i bildet, og simulerer de man eventuelt utelater (eksempler på slike modeller innen grafikkfaget er *ray-tracing*- og *radiositetsmodellen*). Det finnes derfor innen begge fagfelt en rekke ulike teknikker og metoder som varierer både med tanke på realisme og regneintensitet.

### 2.1.2 Gjenskapning av lyd

Det er vanlig å dele simuleringer av lydutbredelse i tre steg: lydkilden, rommet eller mediet som lyden brer seg ut i og mottakeren (eller lytteren) [BD01]. Figur 2-1 illustrerer denne tredelingen.




**Figur 2-1 Kilde – medium – mottakermodellen i akustisk simulering**

Under følger en oversikt over de vanligste egenskapene som blir knyttet til hvert ledd i denne modellen:

1. *Lydkilde.* Simulering av lydkilden går ut på å modellere de romlige egenskapene som denne innehar. Dette vil vanligvis være plasseringen av lydkilden i det tredimensjonale rommet og retningen av denne. Retningen vil i denne sammenheng være bestemt av lydkildens fysiske utforming som gjør den i stand til å utstråle lydbølger i en gitt retning. Se [RMW01] for en undersøkelse av strålingsretningen fra musikalske instrument og det menneskelige hodet.

   Lyden som lydkilden utstråler kan enten være generert av datamaskinen eller den kan være tatt opp på forhånd og spilles av på det gitte tidspunktet. Uansett hvilken metode

som brukes, er denne lyden vanligvis uten endringer og tillagte effekter siden dette gjøres i de senere stegene.

2. *Medium.* Modellering av mediet går ut på å lage effekten av den simulerte omgivelsen, som så endrer lyden som kommer fra lydkilden. Det er i dette trinnet at lydens utbredelse i rommet (gjennom luften) modelleres, samt at eventuelle hindre for lydbølgene blir tatt hensyn til. Effekten av mediet avhenger av avstanden mellom kilden og mottakeren, noe som vanligvis fører til en demping av lyden og en mer spesifikk demping av de høyeste frekvensene (lavpass filtrering). Begge disse effektene er proporsjonal med avstanden mellom kilde og mottaker.

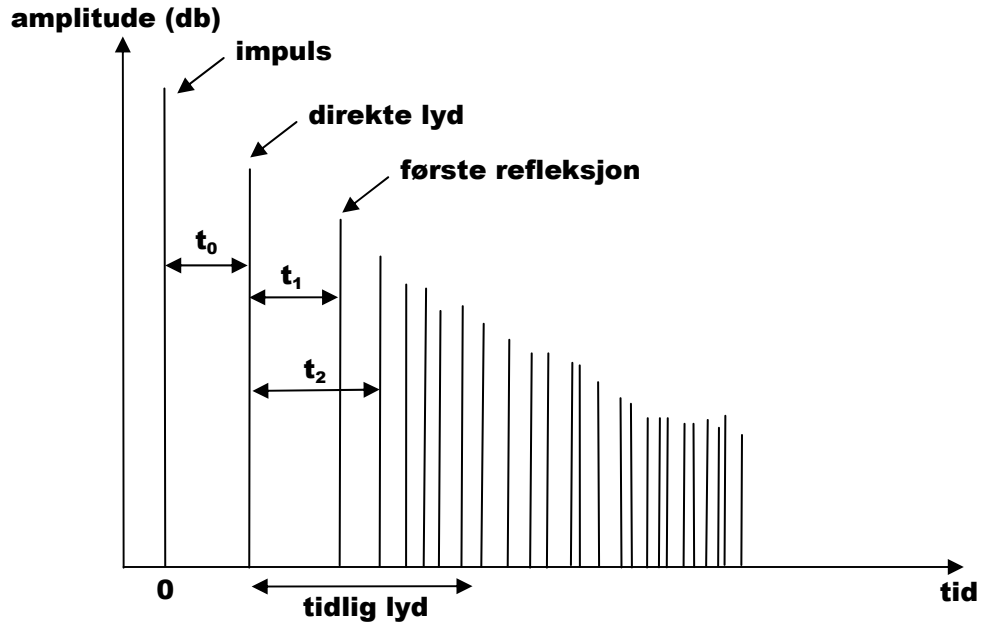
   Lyd vil også ha en bestemt hastighet for et gitt medium, og dette fører til en forsinkelse som er proporsjonal med avstanden (hvis denne avstanden forandres hurtig, gir denne effekten også opphav til den såkalte *Dopplereffekten*). Gjenstander i rommet (for eksempel vegger) skaper på den annen side refleksjoner, eller lydhindre hvis kilden og mottakeren befinner seg på motsatt side av gjenstanden. Slike refleksjoner gir opphav til *romklang*, som er en viktig faktor for hvordan mottakeren oppfatter dimensjonene av rommet.

3. *Mottaker.* Som for lydkilden, vil de romlige egenskapene til mottakeren måtte modelleres for å skape en realistisk representasjon av det virtuelle akustiske rommet. Romlig posisjon og retning er i denne sammenheng viktige faktorer. I virtuell virkelighetsapplikasjoner, der målet er å gi brukeren inntrykk av å være til stede inne i det virtuelle rommet, kan systemet benytte seg av en sofistikert modell for menneskelig hørsel. Høyttaleroppsett eller hodetelefoner vil også spille en rolle i dette steget for hvordan lyden må prosesseres.

Kapittel 2.1.2.1 vil presentere en del teknikker som har vært brukt for å simulere lydens utbredelse i rommet, mens kapittel 2.1.2.2 tar for seg første og siste punkt i denne tredelte modellen (det vil si stedfesting av lyden hos mottakeren, avhengig av lydkildens plassering og retning).

### 2.1.2.1 Lyd gjennom mediet

Romlig akustikk er et viktig tema når lyd skal gjenskapes i et virtuelt rom på en troverdig måte. Lydbølgene brer seg ut i rommet fra lydkilden og blir påvirket av utformingen av så vel rommets geometri som gjenstander som befinner seg i rommet. For å studere disse fenomenene deler en gjerne tidsgrafen av lydsignalet i rommet inn i tre ulike faser: direkte lyd, tidlig lyd og gjenklang [RMW01]. Disse tre parametrene avgjør et roms (eller omgivelses) *impuls respons* (RIR). Lydbølger har en hastighet på omtrent 344 m/s i luft ved normal temperatur og luftfuktighet, og den *direkte lyden* vil derfor nå lytteren etter 0,02-0,2s, alt ettersom hvor langt borte lydkilden er i forhold til lytteren. En liten stund senere vil den samme lyden nå lytteren fra ulike reflekterende overflater, hovedsakelig vegger og tak. Figur 2-2 viser disse ulike tidsintervallene ($t_1$, $t_2$ etc.).




**Figur 2-2 Tidsforsinkelser for en lydimpuls**

Den første gruppen av slike refleksjoner, som når lytteren etter omtrent 50-80ms etter den direkte lyden, kalles *tidlig lyd*. Etter den tidlige lyden, vil de reflekterte lydbølgene ankomme fra alle retninger og avta med tiden, før de til slutt faller sammen til *gjenklang*. Hvis lydkilden avgir en kontinuerlig lyd, vil gjenklangen bygge seg opp til den når et konstant nivå, for så å avta når lydkilden opphører.

Grunnen til at man vanligvis deler inn lyden i ulike intervaller på denne måte, er at de kan ses på som separate lyder. Den direkte og tidlige lyden kan karakteriseres med retning, forsinkelse, demping og annen modifikasjon av lydspektrumet (for eksempel absorpsjon av lyden i luften og ved reflekterende overflater). På den annen side representerer gjenklangen i de fleste rom en tilfeldig prosess som kan ses på som en eksponentielt avtagende støysekvens, uanhengig av plasseringen til lydkilden/mottakeren [RMW01]. Det er derfor mulig å simulere denne delen ved hjelp av standardiserte gjenklangsalgoritmer. Lydsignalet som når lytteren på et bestemt sted i rommet, er derfor den direkte lyden samt effektene av den tidlige lyden og gjenklangen (som kan prosesseres separat).

Under simulering av lyd i virtuelle rom, deler fremgangsmåtene seg i to ulike retninger; på den ene siden de *fysisk baserte* modellene og på den andre de *perseptuelle* modellene [DKP01]. Fysisk baserte modeller har som mål å gjenskape lydutbredelsen gitt de fysiske egenskapene til omgivelsene, mens perseptuelle modeller imiterer det oppfattede inntrykket av omgivelsene heller enn å simulere den romlige impulsresponsen (RIR) på en mest mulig presis måte. Den fysisk baserte fremgangsmåten har tradisjonelt vært brukt i applikasjoner innen arkitektur og evaluering av lydparametere innen forskning, mens den perseptuelle fremgangsmåten har vært benyttet for å reprodusere romlige lydeffekter innen musikkbransjen.

#### *2.1.2.1.1 Fysisk baserte modeller*

I fysisk baserte modeller defineres omgivelsenes geometri, og som regel kun én lytter og én lydkilde. Deres akustiske egenskaper, som for eksempel reflektivitet av overflater og retningen av lydkilden, bygges inn i den matematiske modellen. Ved hjelp av disse parametrene kan en impulsrespons lages ved å simulere lydbølgenes utbredelse i rommet. Vanligvis brukes kun stasjonære lyttere og lydkilder, da alle beregninger må gjøres på nytt hvis noen av disse flyttes eller skifter retning. En god del nyere forskning har studert hvordan disse metodene kan gjøres dynamiske, det vil si at lytteren, lydkilden eller begge kan bevege seg inne i det virtuelle rommet ved sanntids oppdateringshastigheter [TFNC01][DKP01][TG01]. Dette krever nødvendigvis forenklinger i de fysiske modellene av lydutbredelsen, samtidig som at forandringer i parameterverdiene ikke skaper hørbare overganger i den genererte lyden.

Modellering av lydutbredelsen i et rom kan løses eksplisitt ved bruk av *Helmholtz-Kirchoffs integral*, som uttrykker bølgefronten på hvert sted i rommet. Hovedproblemet med å løse denne likningen er at bølgene ikke er kontinuerlige på grunn av hindringer i rommet, diffraksjon og spekulære refleksjoner som skaper små variasjoner over deler av integrasjonsdomenet [FT01]. På grunn av dette, kan ikke en analytisk formel anvendes for det generelle tilfellet for å beregne bølgemønsteret. Det er derfor nødvendig å gjøre utvalg eller dele integrasjonsdomenet inn i ulike deler som hver for seg kan behandles effektivt. Nedenfor følger en kort oppsummering av ulike teknikker som har vært brukt for å tilnærme en slik løsning for modellering av akustikk i rom basert på geometriske beskrivelser av omgivelsene [KDS01].

- *Akustisk skalerte modeller*. I denne modellen bygges en mindre versjon av det fysiske rommet i et visst størrelsesforhold. Impulsresponsen fra dette rommet (RIR) måles så ved å bruke spesialbygde mikrofoner (for å kompensere for mindre bølgelengder). Også lydkilden må emittere lyd som er tilsvarende nedskalert med henhold til bølgelengde. Det må også tas hensyn til at refleksjoner fra enkelte typer materialer (for veggene, taket etc.) kan være bølgelengdeavhengig, og derfor må erstattes av materialer som har egenskaper som tilsvarer de opprinnelige i en nedskalert utgave. Det kan også være nødvendig å bytte ut luften med en annen gass (tørr luft, nitrogen) for å kompensere for mindre luftabsorpsjon [KDS01]. Fordelen med denne metoden er at enkelte akustiske fenomener som for eksempel diffusjon og kantdiffraksjon (som ikke er sanntidsvennlige operasjoner), blir en naturlig del av modellen uten ekstra omkostninger. På den annen side kan det være vanskelig å finne tilsvarende nedskalerte materialer med korrekte egenskaper, og korrekte lydkilder og mikrofoner er også en utfordring. Dessuten er det en forholdsvis kostbar og tidkrevende prosess i forhold til metoder som i sin helhet gjenskapes inne i datamaskinen.

- *Strålemodeller*. Denne typen modeller deles videre inn i to underkategorier; *strålesporing* (*Ray-tracing*)- og *image-sourcemetoder*. Image-sourcemetoder beregner lydrefleksjoner ved å danne nye lydkilder ved de reflekterende overflatene basert på spekulære refleksjoner. Disse blir i neste omgang behandlet som nye lydkilder med et lydsignal som avhenger av overflatens reflektivitet og avstand fra den forrige kilden [TN01]. Også synligheten av lytteren fra denne posisjonen avgjøres for å bestemme om denne nye lydkilden kan hoppes over. Metoden har vært benyttet i sanntidssystemer (for eksempel i MPEG-4 standarden [ISO01]), og brukes her for å beregne den tidlige lyden (det vil si de første refleksjonene). På grunn av at denne

teknikken er en forholdsvis ”lett” metode, har den sine begrensninger. Siden lydens utbredelse representeres ved stråler, vil lydens karakteristikker som relateres til dets bølgekarakter ikke gjengis. Fenomener som diffraksjon og diffuse refleksjoner blir derfor ikke korrekt gjengitt i slike sanntids simuleringer, og kan medføre hørbare ”feil” under kjøring.

Stålesporingsmetoder er også teknikker som baserer seg på den geometriske definisjonen av rommet. Lydstråler avgis fra en punktkilde i ulike retninger, og veien de tar gjennom rommet spores fra lydkilden og opp til et spesifisert antall steg (eller tid) langs denne veien. Mottakeren modelleres som et volum i en gitt lokasjon. De lydstrålene som treffer denne, registreres med retning, forsinkelse (avhengig av antall steg) og energi [KKH01]. I denne modellen tas diffusjon hensyn til ved å gi de reflekterende overflatene en frekvensavhengig parameter som angir hvordan lyden reflekteres i ulike retninger avhengig av frekvens. Hensikten med disse metodene har ofte vært å beregne frekvensavhengig impulsrespons for et rom, samt å visualisere lydens utbredelse i et rom ved hjelp av strålene som brukes under simulering av lydutbredelsen. [FT01] beskriver et slikt system som benytter flere ulike optimaliseringsteknikker for å oppnå strålesporing i sanntid.

- *Bølgebaserte modeller*. Ved å dele omgivelsene inn i diskrete deler kan det resulterende settet av likninger brukes for å finne en tilnærmet løsning på bølgelikningen [MG01]. *Finite Element Method* (FEM) og *Boundary Element Method* (BEM) er to eksempler på denne typen teknikker. I FEM er det rommets volum som blir delt opp, og i BEM er det overflaten av omgivelsene som blir delt opp i elementer. Kompleksiteten til disse metodene er proporsjonal med størrelsen på omgivelsene, og inverst proporsjonal med størrelsen av elementene som definerer den høyeste frekvensen som kan modelleres ved bruk av disse metodene. Hovedproblemet med disse teknikkene er å anslå størrelsen på elementene som må brukes under diskretiseringen av rommet for å fange faseforskjeller i lyden [FT01]. Det er dessuten kostbart å beregne formfaktorer mellom overflater og lydkilder. Begge teknikker har en del fellesnevnere med radiositetsmodellen innen grafikkfeltet, og de er i likhet med denne ikke egnet for sanntids modellering av rommets impulsrespons.

#### *2.1.2.1.2 Perseptuelle modeller*

Den perseptuelle fremgangsmåten bygger også på parametriske beskrivelser av omgivelsene. Forskjellen er imidlertid at man i disse sammenhengene ikke er ute etter en eksakt modellering av det gitte lydrommet, men heller en ”følelse” av at den lyden man hører kan ha kommet fra dette rommet. Denne retningen har vanligvis vært assosiert med musikkproduksjon, der en ikke forlanger en eksakt reproduksjon av gjenklangen i et spesifikt rom, men ønsker å gi lyden romfølelse. Det er derfor vanlig at også parametrene i slike simuleringer er av typen ”konsertsal”, ”baderom” etc. i forhold til detaljerte fysiske innstillinger. Disse innstillingene mapper ned til mer grunnleggende parametere i selve modellen, men også disse er mer *ad hoc* enn de man finner i de fysiske modellene (hvor skillet går mellom fysiske og perseptuelle modellene er selvsagt et definisjonsspørsmål, og enkelte modeller vil ligge i grenselandet mellom de to. Det kan likevel være nyttig å foreta en slik todeling, siden det gjør valget av metode for en spesifikk applikasjon enklere).

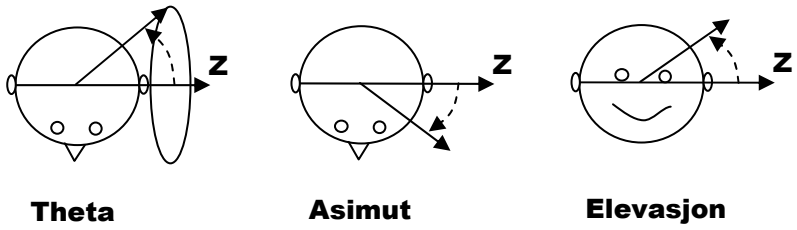
Et eksempel på denne typen modeller er [JW01], der ni ulike brukerkontrollerte parametere blir brukt til å generere den ønskede impulsresponsen. Disse parametrene ble utviklet ved psykoakustiske eksperiment, og innbefatter blant annet varme, gjenklanglengde etc.

Perseptuelle modeller har, som nevnt tidligere, i hovedsak vært forbundet med lydproduksjon innen musikk, men er også interessante for virtuell virkelighetsapplikasjoner på grunn av at de generelt krever mindre prosesseringsressurser enn de fysiske modellene. Avhengig av applikasjonen, er det som regel "godt nok" å legge en generell romfølelse til den direkte lyden. I slike applikasjoner vil stedfesting av lyd spille en større rolle for å gi brukeren inntrykk av å være til stede i det virtuelle rommet. Også kombinasjoner av de to metodene har vært brukt, blant annet i MPEG-4-standarden [ISO01]. Her er det mulig å rendre enkelte kilder med den fysiske metoden og andre med den perseptuelle metoden (i den samme scenen).

### 2.1.2.2 Mottakeren - Lokalisering av lyd

For å kunne lage overbevisende stedfestet lyd, er det nødvendig å kjenne til hvordan lokalisering av lyd fungerer, det vil si hvilke metoder vi bruker for å skape inntrykk av lyd fra ulike retninger. Det er etter hvert blitt opparbeidet en solid kunnskapsbase vedrørende hørsel og hvilke mekanismer vi bruker for å avgjøre hvor lyder kommer fra (se [RMW01] for en inngående presentasjon av hørsel generelt og [BJ01] for en detaljert beskrivelse av retningsbestemmende mekanismer).

For å forklare ulike hørselsfenomen, er det vanlig å etablere et koordinatsystem med origo i sentrum av hodet. Figur 2-3 viser grafisk hvordan dette koordinatsystemet defineres ut fra lytterens hode (figuren er inspirert av [BD01]).




**Figur 2-3 Definisjon av koordinatsystem**

Z-aksen har retning fra sentrum av hodet mot det venstre øret. Vinkelen fra z-aksen benevnes *theta*, og denne dekomponeres i det horisontale planet i vinkelen *asimut*, og i det vertikale planet vinkelen *elevasjon*.

Under følger et sammendrag av ulike metoder som det er antatt at mennesker bruker for å avgjøre *retningen* som en lyd stammer fra, det vil si hvordan vi bestemmer asimut- og elevasjonsvinkler basert på den innkommende lyden. Listen kan gjøres lengre og det har vært diskusjoner innen fagmiljøet hvilke metoder som er de mest betydningsfulle [RMW01].

- ITD (*interaural time difference*), også kalt IDT (*interaural delay time*), er tidsforsinkelsen som oppstår fra lyden når det nærmeste øre til det når det andre. ITD

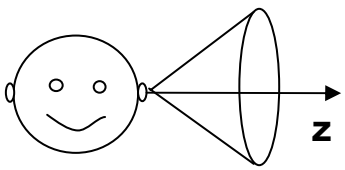
og IID (se neste punkt) ble undersøkt av Lord Rayleigh for over 100 år siden, og inngikk i hans *Duplex-teori*. Ifølge denne teorien, er det ITD og IID som er de viktigste faktorene lytteren benytter seg av under bestemmelsen av asimutvinkelen for en lydkilde. ITD er i de senere år ansett for å være den sterkeste kilden for å avgjøre hvor lyden kommer fra [BD01]. Denne tidsforskjellen er null for lyder med opprinnelse rett foran, bak eller over lytteren, og omtrent 0,63 ms for en lyd som kommer fra lytterens høyre eller venstre side. Denne tidsforskjellen er imidlertid avhengig av lydens frekvens, samt avstanden mellom kilden og lytteren [BJ01]. For signaler med lavere frekvens enn 1,6 kHz, skapes en faseforskjell mellom lyden som ankommer hvert øre. For lyder med høyere frekvens, manifesterer forsinkelsen seg som en amplitudeforskjell mellom de to lydene. Hvis en antar at hodet har en kuleform, kan IDT regnes ut ved hjelp av denne formelen [WEB04]:

$$ITD = \frac{r}{c}(\theta + \sin\theta) \qquad , -90° \le \theta \le 90°$$

Her er *c* lydhastigheten, det vil si 344 m/s for de fleste normale omgivelser, og *r* er hodets radius (modellert som en kule). ITD er null når lydkilden er foran eller bak lytteren, og har sin største verdi (r/c)( $\pi$/2 + 1) når lydkilden befinner seg på en av sidene (tilsvarer omtrent 0,63 ms for et normalt hode som nevnt over).

Hvor nøyaktig vi kan oppfatte denne forskjellen i ankomsttid avhenger av omstendighetene [WEB04]. For tale i et rom med normal akustikk, er det vanligvis mulig å avgjøre posisjonen til lydkilden innen 10˚ - 20˚. Under optimale forhold har det vært mulig å måle en nøyaktighet på omkring 1˚ hvis det er snakk om å avgjøre om en lydkilde beveger seg. Dette tilsvarer en forskjell på 10 μs og kan sammenliknes med samplingsraten til CD-kvalitets lyd (44.1 kHz) som tilsvarer 22,7 μs.

En gitt ITD-verdi gir ikke entydig plasseringen av en lydkilde, men begrenser kilden til å ligge innen en hyperbel med en akse som sammenfaller med z-aksen i koordinatsystemet beskrevet over. Denne hyperbelen kan tilnærmes med en kjegle med konstant theta, og dette fenomenet kalles gjerne *the cone of confusion* [WEB04].




**Figur 2-4 The cone of confusion**

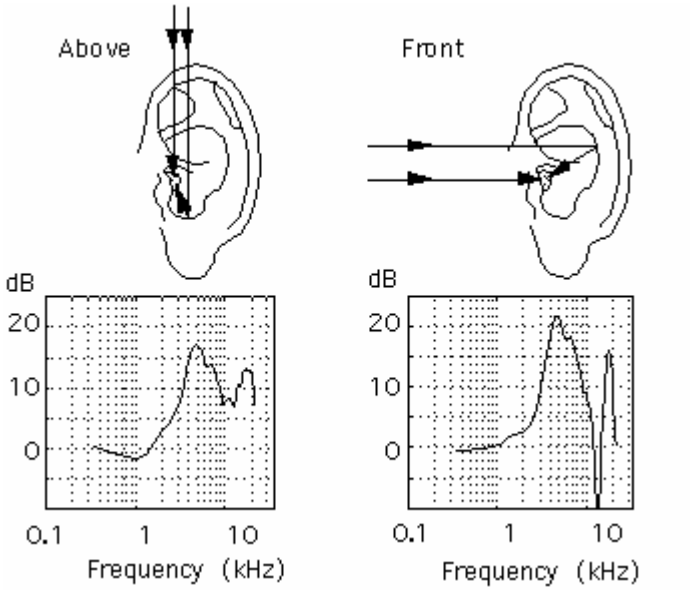
All lydkilder som befinner seg inne i denne kjeglen vil altså oppfattes å komme fra det samme fysiske stedet.

- IID (*interaural intensity difference*), også kalt *hodeskygge-effekten*. Lord Rayleigh observerte også at de innkommende lydbølgene gjennomgår diffraksjon når de treffer hodet [WEB04]. Han fant at det i tillegg til tidsforskjellen også var en forskjell i lydintensiteten mellom de to ørene. På grunn av hodets form er denne forskjellen spesielt avhengig av frekvensen på lyden. Ved lave frekvenser, der bølgelengden er forholdsvis stor i forhold til hodets størrelse, er det omtrent ikke noen forskjell i

intensitet mellom de to ørene. For høyere frekvenser (kortere bølgelengde) kan intensitetsforskjellen komme opp i 20 dB eller mer.

Det er derfor spesielt for de lavere frekvensene ITD spiller en viktig rolle i tolkningen av hvor lyden kommer fra. For høyere frekvenser (over 1500 Hz) er det IID som dominerer. Duplex-teorien hevder derfor at de to teknikkene er komplementære, og at de til sammen løser lokaliseringsproblemet over hele det hørbare spekteret (i asimutplanet).

- Det ytre øre (*pinna*) fungerer som en akustisk antenne, der den fysiske geometrien fører til at enkelte frekvenser blir forsterket, mens andre blir dempet av interferenseffekter skapt av den spesielle utformingen. Denne frekvensresponsen er dessuten retningsavhengig [WEB04]. Figur 2-5 illustrerer denne retningsavhengigheten i frekvensrespons for to ulike retninger. I begge tilfeller gjelder at lyden som når ørekanalen består av en komponent med direkte lyd (kortest vei), samt en komponent med reflektert lyd fra det ytre øre (lenger vei).




**Figur 2-5 Frekvensrespons for ulike lydkilderetninger [WEB04]**

For lave frekvenser fungerer det ytre øre som en oppsamler av lydenergi og de to komponentene når ørekanalen i fase. For høyere frekvenser (fra omtrent 4 kHz [BD01]) vil den andre komponenten ankomme ørekanalen ute av fase med den direkte lyden, og det oppstår destruktiv interferens [WEB04]. Av figuren under kan denne effekten ses som et drastisk fall i responsen rundt 10 kHz. Dette fallet (også kalt *pinna notch*) er på sitt maksimale når forskjellen i reiselengde mellom den direkte og den reflekterte lyden er en halv bølgelengde.

Som det også fremgår av figuren, er det ytre øre en mer effektiv reflektor for lyder som kommer forfra enn ovenfra. Dette skaper igjen et mer markant fall i frekvensresponsen for lydkilder som kommer forfra enn de som kommer ovenfra. Lengden av reisebanen for den reflekterte lyden varierer også med høyden, og skaper forskyvninger av frekvensfallet med varierende høyde av lydkilden. Disse to effektene til sammen gjør at det ytre øret har gått for å være vår viktigste kilde for å avgjøre lydkildens elevasjon (eller høyde i forhold til lytteren) [WEB04].

- Skygge fra skuldrene og overkroppen. Enkelte frekvenser (omtrent i spekteret 1-3 kHz, avhengig av lytterens fysiske attributter) når øret via refleksjoner fra overkroppen [WEB04]. Ekkoene fra disse refleksjonene når øret med en forsinkelse som er avhengig av blant annet lydkildens elevasjon. For kjente lyder, kan ekko fra skuldrene gi informasjon om både asimut- og elevasjonsvinkler. Dette er likevel et mye svakere indisium på retning enn de foregående metodene, men kan være utslagsgivende ved tvilstilfeller.

De overfor nevnte metodene bestemmer til sammen lydkildens retning forholdsvis godt. *Avstanden* til lydkilden er den siste parameteren som må vurderes for å endelig posisjonere lydkilden i det tredimensjonale rommet. Også her finnes det en rekke teorier om ulike teknikker som blir brukt, men i forhold til retningsbestemmelsen er disse ikke like godt kartlagt eller forstått. De viktigste er likevel [WEB04]:

- Lydstyrke. Generelt vil lyden som ankommer direkte fra kilden avta i styrke omvendt proporsjonalt med kvadratet av avstanden fra lydkilden.

$$L = \frac{V}{d^2}$$

  Det vil si at hvis $V$ er den opprinnelige lydstyrken ved lydkilden, vil denne ha sunket til $L$ ved avstanden $d$ til kilden [NSG01]. En effekt av dette forholdet er at for en lydkilde som avgir et konstant signal, vil lydstyrken hos lytteren øke når lydkilden nærmer seg denne. På grunn av desibelskalaens oppbygning, vil lyden avta med 6 dB for hver dobling av avstanden til lydkilden (hvis en antar en punktlydkilde og ser bort fra eventuelle fysiske hindre som kan blokkere lyden) [RMW01].

  Det er likevel ikke slik at det er en en-til-en sammenheng mellom lydstyrke og avstand, det er ikke nok å bare spille av lyden ved en lavere styrke for å gi inntrykk av avstand. Dette kommer av at lydkildens karakteristikker også spiller en rolle i denne avstandsbedømmingen. Det er for eksempel forskjell mellom å rope og å snakke med normal stemme, og selv om lydstyrken som når lytteren kan være den samme for begge disse tilfellene, oppfatter vi personen som roper som lenger unna på grunn av denne *a priori* kjennskapen til lydkilden.

- Bevegelsesparallakse. Dette fenomenet oppstår når lytteren beveger hodet for å oppfatte lyden bedre. Når hodet beveges vil også retningen av lydkilden relativt til hode forandres (det vil si asimutvinkelen), og hvor stor denne forandringen blir vil være avhengig av lydkildens avstand. For eksempel vil asimutvinkelen for kilder som er svært nære lytteren få store forandringer ved kun små hodebevegelser, mens det motsatte er tilfellet for lydkilder som er langt borte.

- Store IID-verdier. Denne effekten gjør seg gjeldende når lydkilden kommer veldig nær det ene øret (omtrent en meter), og lyden til en stor grad begrenser seg til denne siden. Hodeskyggen spiller også en rolle her, og hindrer lyden i å nå det andre øret. Et eksempel er et innsekt som summer nær det ene øret [WEB04].

- Forholdet mellom direkte og indirekte lyd. Som nevnt i kapittel 2.1.2.1 består lyden som når lytteren i et vanlig rom av den direkte lyden, samt lyd som er reflektert via

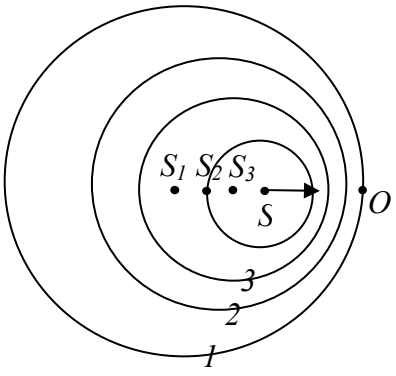
vegger og hindringer i selve rommet. I et normalt rom vil den indirekte lyden være forholdsvis konstant, uavhengig av lydkildens posisjon. Dette er ikke tilfellet for den direkte lyden. Når lydkilden er forholdsvis nære lytteren, vil denne dominere over den indirekte lyden som ankommer lytterens posisjon noe senere. Ved større avstander vil forholdet mellom direkte og indirekte lyd minke på grunn av at den direkte lyden avtar inverst med kvadratet av avstanden (som nevnt over), mens den indirekte lyden vil holde seg forholdsvis konstant.

Siden lydrefleksjoner og omgivelsenes bakgrunnsstøy kan være forholdsvis fremtredende i lydbildet, blir spørsmålet hvordan det er mulig å avgjøre retning når den direkte lyden "drukner" i denne bakgrunnen. Undersøkelser har vist at det er lyden som ankommer lytteren først som brukes under denne posisjoneringen [BJ01]. Dette fenomenet kalles *the precedence effect* eller *Haas-effekten*. Hvis liknende lyder ankommer innen 35 ms, bestemmes lydkilden til å stamme fra der den tidligste lyden kom fra. Øret antar at lyd som følger etter dette kommer fra den samme lydkilden, forutsatt at den påfølgende lyden har tilnærmet like spektra og ikke er mye sterkere enn den første [LC01]. Bakgrunnslyden utelukkes imidlertid ikke helt, og er blant annet en faktor for avstandsbedømmelse som nevnt over.

Fenomenet nevnt over er relatert til det som gjerne omtales som *cocktaileffekten*, som forklarer hvordan det er mulig å konsentrere seg om en spesiell lydkilde i et rom med mye bakgrunnsstøy. Bakgrunnsstøyen filtreres bort og det blir mulig å fokusere hørselen på det man er interessert i [RMW01].

I forbindelse med posisjonering av lydkilden, kan også *Franssenеffekten*, eller *interaural envelope difference* (IED) nevnes. Det er funnet at vi avgjør retningen av en lydkilde ved å analysere de høyfrekvente delene ved starten av en lyd [HR01]. Dette gjelder også hvis mesteparten av lydenergien stammer fra lave frekvenser fra en annen retning. Dette er grunnen til at plasseringen av subwoofer-høyttaleren er forholdsvis irrelevant i et flerkanals høyttalersystem.

De ulike mekanismene for bestemmelse av lydkildens posisjon som er nevnt over, vil fungere på både stasjonære og bevegelige lydkilder. For bevegelige lydkilder (eller tilsvarende en bevegelig lytter) er det likevel enkelte spesielle fenomener som gjør seg gjeldende. Den mest fremtredende av disse er *Dopplereffekten* [RMW01]. Vanligvis er lydens frekvens som når lytteren den samme som den som genereres hos lydkilden. Dette utsagnet holder likevel ikke hvis enten lydkilden, lytteren eller begge er i bevegelse mot eller fra hverandre. Hvis de beveger seg mot hverandre, vil frekvensen som lytteren måler være høyere enn den som avgis av lydkilden. Hvis de beveger seg fra hverandre vil det motsatt observeres en lavere frekvens hos lytteren enn det som avgis fra lydkilden. Figur 2-6 illustrerer denne effekten for en lydkilde i bevegelse.




**Figur 2-6 Dopplereffekten for en lydkilde i bevegelse**

Her har lydkilden *S* emittert lydbølge 1 i punktet $S_1$, 2 i punktet $S_2$ etc. og beveger seg mot lytteren (*O*). Bølgefrontene representeres av sirkler, med sentrum som beveger seg mot lytteren for hver ny bølge. Lytteren vil derfor oppfatte en høyere frekvens enn tilfellet ville ha vært hvis lydkilden var i ro. Hvis lyden har en hastighet $v$ i dette mediet, og lydkilden har en hastighet på $v_s$ og avgir lyd med frekvens $f_s$, vil lytteren oppfatte en frekvens $f$ gitt av (samme formel gjelder også for lydkilder som beveger seg i motsatt retning, men fortegnet på $v_s$ må da endres tilsvarende) [RMW01]:

$$f = f_s \frac{v}{v - v_s}$$

For en stasjonær lydkilde med en lytter som beveger seg mot lydkilden med en hastighet $v_O$, vil lytteren oppfatte en frekvens $f$ gitt av [RMW01]:

$$f = f_s \frac{v + v_O}{v}$$

Dette fenomenet brukes blant annet til å vurdere hastighet, både for lydkilden eller lytteren, avhengig av hvem som er i bevegelse.

### *2.1.2.2.1 Modeller for simulering av stedfestet lyd*

Simulering av akustiske fenomen er et godt dokumentert forskningsområde. Det er utviklet en rekke ulike systemer for ulike bruksområder, alt fra enkle tredimensjonale posisjoneringer av lydeffekter til komplekse, fysisk baserte simuleringer av konserthaller. Løsninger som lager lydspor *offline* i parallell til pre-rendret grafikk har eksistert i mange år. For interaktive virtuelle virkelighetsapplikasjoner, må derimot lyden rendres i sanntid sammen med grafikken. Dette skaper en del flere begrensninger som må tas hensyn til under design av systemet. Den største utfordringen i så henseende, er den begrensede regnekraften som er tilgjengelig for lydprosesseringen (spesielt hvis lyd- og grafikkprosesseringen må dele på ressursene).

Flere forsøk har blitt gjort for å utvide lydfeltet fra lydopptak [PV01]. De første opptakene var monofoniske og skapte punktliknende lydfelt. Et stort skritt ble tatt da to-kanals stereofoniske

opptak ble gjort mulig, der lydfeltet ble utvidet til en linje mellom de to høyttalerne. Denne metoden er fortsatt den mest brukte teknikken i både privat- og profesjonelt lydutstyr.

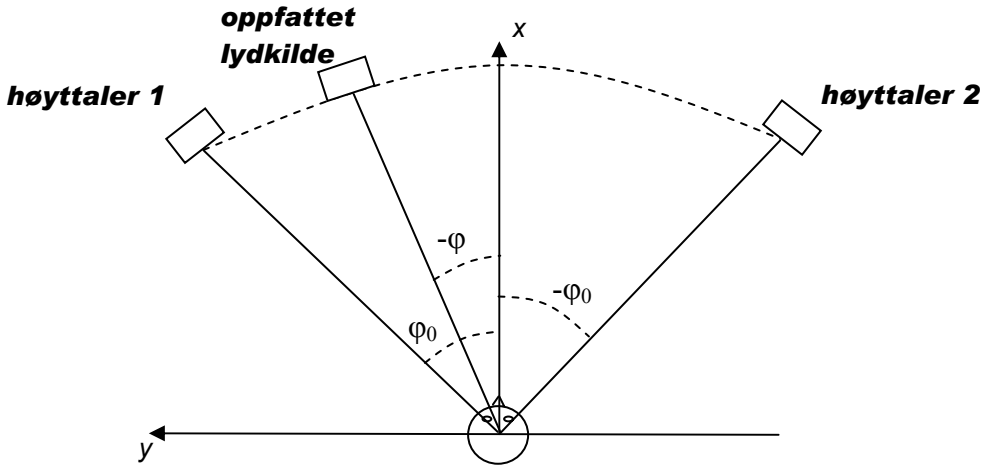
Det har også vært gjort en rekke forsøk på å utvide lydfeltet utover de metodene som er nevnt over. Plassering av høyttalere i et horisontalt plan (kalt *pantofonisk*) er den vanligste konfigurasjonen, mens også systemer med høyttalere med en tredimensjonal plassering på en imaginær kule (kalt *perifonisk*) har vært prøvd [SG01].

En skiller vanligvis mellom to ulike teknikker for å modellere stedfestet lyd på en datamaskin [NGS01]. Den ene metoden filtrerer lyden ved hjelp av en såkalt *head-related transfer function* (HRTF), og kan i teorien gjenskape lyd perifonisk. Den andre metoden, *volume panning*, modellerer et lydbilde med ulike intensiteter avhengig av hvor lydkilden befinner seg, og sender denne lyden til strategisk plasserte høyttalere rundt omkring i rommet, enten plassert pantofonisk eller perifonisk.

#### 2.1.2.2.1.1 Volume panning

Tredimensjonal lokalisering ved bruk av volume panning, er implementert i de fleste verktøy med multimedia og spillmarkedet som mål. Eksempler på dette er Microsofts Direct Sound [WEB01], Creative Labs EAX [WEB02] og OpenAL [WEB03]. Hardware-akselererte løsninger for disse grensesnittene finnes i flere av dagens lavbudsjetts lydkort. Disse benytter som regel et fire eller 5.1 høyttaleroppsett, og tilbyr minimal lokalisering langs z-aksen ved å dempe eller øke høyfrekvenskomponentene av signalet. Slike API med fastsatt høyttaleroppsett, vil i de fleste tilfeller være lite fordelaktige for omgivelser som for eksempel RAVE, som krever en mer dynamisk plassering av høyttalere i forhold til de store projeksjonsveggene som omgir brukeren. Et annet poeng som taler imot bruken av disse verktøyene, er at mange spesialiserte lydsystemer (som gjerne brukes i tyngre visualiseringsomgivelser) krever å kunne endre på signalrekkefølgen (for eksempel å sette inn spesialiserte romklangalgoritmer etc.) [NGS01].

Ved den enkleste formen for panning avgir to høyttalere det samme lydsignalet, men amplituden kan varieres mellom dem. Lytteren får på denne måten inntrykk av at det kun er én lydkilde, plassert på en todimensjonal sektor mellom de to høyttalerne, avhengig av amplitudeforskjellen mellom dem. Figur 2-7 viser hvordan lydkilden vil oppfattes når høyttaler én avgir et sterkere signal enn høyttaler to (figuren er inspirert av [PV01]).




**Figur 2-7 Tokanals panning**

To høyttalere er plassert symmetrisk, og danner $\varphi_0 = 30^\circ$ med medianplanet. Amplituden til hver høyttaler blir kontrollert med en faktor henholdsvis $g_1$ og $g_2$. Hvis lydsignalet fra den virtuelle kilden må holdes konstant (eksempelvis under bevegelse av denne) må disse faktorene normaliseres. Hvis lydstyrken settes til $C > 0$, kan dette uttrykkes tilnærmet som:

$$g_1^2 + g_2^2 = C$$

Parameteren $C$ blir dermed også en volumkontroll for den virtuelle kilden, og kan brukes til å gi inntrykk av avstand (jo større $C$ desto nærmere oppfattes lydkilden og omvendt) [PV01]. Som det fremgår av figuren, vil lydkilden bestemmes til et punkt på buen som dannes mellom de to høyttalerne og lytteren, også kalt den *aktive buen*. For å finne de to faktorene $g_1$ og $g_2$ som bestemmer fordelingen av lyden mellom de to høyttalerne, kan en benytte likningen over samt sammenhengen [BB01],

$$\frac{\sin\varphi}{\sin\varphi_0} = \frac{g_1 - g_2}{g_1 + g_2}$$

der $0^\circ < \varphi_0 < 90^\circ$, $-\varphi_0 \le \varphi \le \varphi_0$, og $g_1, g_2 \in [0,1]$.

De to faktorene som angir relativ amplitude for hver høyttaler kan også beregnes ved bruk av en vektorrepresentasjon, og omtales da gjerne som *vector base amplitude panning* (VBAP) [PV01]. Hvis $\boldsymbol{l}_1 = [l_{11}\ \ l_{12}]^T$ og $\boldsymbol{l}_2 = [l_{21}\ \ l_{22}]^T$ er to enhetsvektorer som peker i retning av hver sin høyttaler, og enhetsvektoren $\boldsymbol{p} = [p_1\ \ p_2]^T$ peker mot den oppfattede lydkilden, kan retningsvektoren $\boldsymbol{p}$ uttrykkes ved hjelp av $\boldsymbol{l}_1$ og $\boldsymbol{l}_2$:

$$\boldsymbol{p} = g_1\boldsymbol{l}_1 + g_2\boldsymbol{l}_2 \ ,$$

som videre kan uttrykkes på matriseformen:

$$\boldsymbol{p}^T = \boldsymbol{g}\boldsymbol{L}_{12} \ ,$$

der $\boldsymbol{g} = [g_1\ \ g_2]$ og $\boldsymbol{L}_{12} = [\boldsymbol{l}_1\ \ \boldsymbol{l}_2]^T$. Dette kan dermed løses ved:

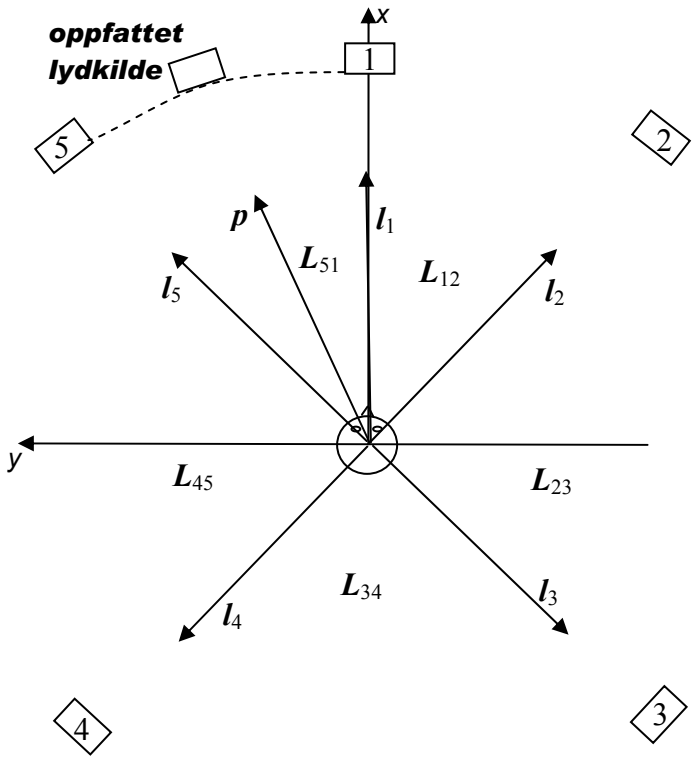
$$\boldsymbol{g} = \boldsymbol{p}^{\mathrm{T}} \boldsymbol{L}_{12}^{-1} = \begin{bmatrix} p_1 & p_2 \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix}^{-1}$$

Dette systemet kan løses trivielt for alle $\varphi_0 \neq 0^\circ$ og $\varphi_0 \neq 90^\circ$, der den inverse av $\boldsymbol{L}_{12}$ ikke eksisterer (disse tilfellene tilsvarer uansett som regel uinteressante høyttaleroppsett). Det siste steget i denne utregningen er å normalisere vektoren $\boldsymbol{g}$:

$$\boldsymbol{g}^{\text{normalisert}} = \frac{\sqrt{C}\boldsymbol{g}}{\sqrt{g_1^2 + g_2^2}}$$

For systemer som består av flere enn to høyttalere plassert rundt lytteren, kan beregningene over utvides til å også regne ut volumfaktorene for disse ekstra høyttalerne. En måte å løse et slikt system på, er å la to og to nabohøyttalere danne et par slik som det over [PV01]. En gitt høyttaler vil dermed delta i to par (med hver nabohøyttaler). Hvert slikt par vil da danne en aktiv bue som den oppfattede lydkilden befinner seg på, og disse vil være ikke-overlappende. På denne måten vil kun to høyttalere til enhver tid brukes for å generere en bestemt lydkilde, nemlig de to høyttalerne som er nærmest lydkilden. Når disse to høyttalerne er bestemt, kan likningene over brukes til å regne ut de to volumfaktorene som da trengs. For en bevegelig lydkilde vil derfor volumfaktoren for en gitt høyttaler avta når lydkilden beveger seg bort fra denne (og altså bli null når denne høyttaleren ikke lenger er en av de to nærmeste).

Figur 2-8 (inspirert av [PV01]) viser et eksempel med bruk av fem høyttalere plassert rundt lytteren. I dette tilfellet, der lydkilden er plassert mellom første og femte høyttaler, vil $\boldsymbol{l}_1$ og $\boldsymbol{l}_5$ brukes og derfor altså $\mathbf{L}_{51}$ i likningene over. Alle andre høyttalere vil i dette tilfellet ikke avgi lyd for denne lydkilden.




**Figur 2-8 VBAP med fem høyttalere**

De overfor nevnte utregingene av de relative volumfaktorene gjelder for det todimensjonale tilfellet (med høyttalerne og lytteren plassert i det samme horisontale planet), men det samme grunnprinsippet gjelder også for høyttalerplasseringer i tre dimensjoner. I stedet for en aktiv bue, vil det da være snakk om en *aktiv trekant* som må velges for en gitt plassering av lydkilden, og de tre relative volumfaktorene må videre regnes ut for de tre høyttalerne som befinner seg nærmest lydkilden.

Ved bruk av denne metoden må lydkilden befinne seg på den aktive buen, eller tilsvarende på den aktive trekanten. Dette gjør at for å reprodusere et komplett lydbilde, må høyttalere plasseres rundt brukeren slik at hele lydfeltet dekkes. For å oppnå best mulig lokaliseringspresisjon, er det nødvendig å gjøre den aktive buen eller trekanten så liten som mulig, noe som i praksis innebærer å øke antall høyttalere. For å unngå at kostnadene ved et slikt system blir for høye, er det mulig å konsentrere plasseringen av høyttalerne i den retningen eller område som krever størst lokaliseringspresisjon. Dette vil som regel si rett frem og på sidene (dette prinsippet kan også finnes igjen i en del forbrukerutstyr som finnes på markedet i dag, der man vanligvis har tre eller flere høyttalere i front, men bare to bak lytteren).

Som det fremgår av presentasjonen over, kan volume panning enkelt konfigureres til ulike høyttaleroppsett (både antall og plassering). Ved å angi retningsvektorene under oppstart kan det bygges et system som muliggjør lokalisering av en lydkilde, og man er på denne måten ikke avhengig av å binde seg til en spesifikk hardwareløsning. Teknikken er dessuten lite ressurskrevende i forhold til HRTF-metoden som blir presentert under ([PV01] angir kun tre multiplikasjoner per lydkilde, mot 12 multiplikasjoner i et forenklet og optimalisert HRTF-system).

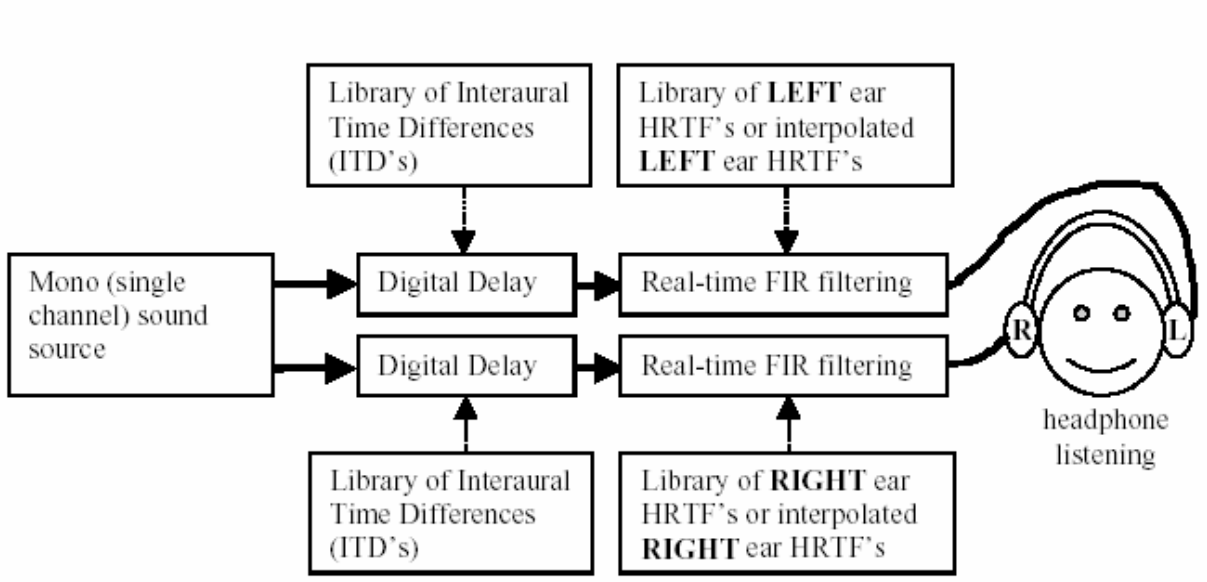
#### 2.1.2.2.1.2 Head-Related Transfer Function (HRTF)

Som nevnt i kapittel 2.1.2.2 er det flere ulike mekanismer som spiller inn under lokaliseringen av lyder i rommet. Alle disse mekanismene fanges opp av den såkalte HRTF-modellen. HRTF-funksjonen ”oversetter” den genererte lyden fra en lydkilde til den lyden som ankommer trommehinnen. HRTF-baserte metoder modellerer frekvensresponsen av det menneskelige hode og øre som en funksjon av retningen til lydkilden [RMW01]. Teknikken er spesielt egnelig ved bruk av hodetelefoner. Det er også mulig å benytte denne teknikken ved bruk av to høyttalere plassert rundt brukeren, men det må da benyttes spesielle teknikker siden HRTF-metoden antar at venstre øre ikke kan høre lyden fra den høyre høyttaleren og omvendt (noe som kalles *cross-talk cancellation*). Det kreves derfor en del finjustering for å oppnå en tilnærmet like god lokaliseringspresisjon som ved bruk av hodetelefoner.

HRTF-funksjonen varierer mellom forskjellige individer på grunn av ulik kroppsbygning (ørets utforming, skuldrenes størrelse og posisjon etc.) [RDD01]. Spesielt gir dette seg utslag i lokalisering i vertikalplanet. På grunn av kostnadene ved å måle hver bruker sin individuelle HRTF-funksjon, brukes i de fleste tilfeller en standardfunksjon for dette formålet. Det har blant annet blitt utviklet en spesiell dukke, KEMAR, utformet som overkroppen til et reelt menneske med mikrofoner i hver øregang med hensikt å måle HRTF-funksjonen. Disse dataene er fritt tilgjengelig gjennom blant andre [WEB05].

Målingene foregår ved å variere lydkildens asimut (θ) og elevasjonsvinkler (ϕ), som definert i kapittel 2.1.2.2, med konstant avstand fra lytteren. Ved å samtidig variere frekvensen på kildens lydsignal, *f*, får en for hvert øre en tredimensjonal funksjon $HRTF_{L/R}(\theta, \phi, f)$ [CC01]. Denne funksjonen er fouriertransformasjonen av Head-Related Impulse Response (HRIR), som er impulsresponsen av lydsignalet ved trommehinnen. Denne funksjonen gir altså for en gitt frekvens og plassering av lydkilden, den tilsvarende frekvensresponsen i dB i det indre øret. Disse målingene gjøres som regel på uniforme intervaller rundt brukeren, med de relevante frekvensene. Dette gjør at man som regel interpolerer mellom to eller flere slike funksjoner når man ønsker HRTF-verdien fra en spesifikk lokasjon og frekvens.

For å generere de to signalene (et til hver hodetelefon) som trengs for å oppnå retningsbestemt lyd fra et monosignal, konvolveres det originale lydsignalet med dette HRTF-filteret. Figur 2-9, hentet fra [CC01], viser skjematisk hvordan et HRTF-system kan realiseres.




**Figur 2-9 Skjematisk oppsett av et HRTF-system [CC01]**

Som det fremgår av figuren er det her også lagt inn en tidsforsinkelse på grunn av IDT-effekten, som er beskrevet i kapittel 2.1.2.2. For mer informasjon om denne teknikken refereres det til [CC01].

HRTF-baserte systemer bør, til forskjell fra volume panning-metoden nevnt over, tilpasses hver enkelt brukers fysiske mål (hode og ører) for å fungere optimalt. Dette er en tid- og ressurskrevende prosess der det lett kan oppstå små unøyaktigheter [CC01]. Dessuten er konvolusjonssteget i algoritmen regneintensiv, og det kreves derfor betydelige maskinvareressurser for å kunne kjøres i sanntid. Da lyden i de fleste tilfeller reproduseres ved hjelp av høretelefoner, er det videre nødvendig å utstyre alle samtidige brukere av systemet med hvert sitt par av disse, noe som kan være uønsket i enkelte situasjoner ved for eksempel samarbeid i virtuelle omgivelser på samme sted.

Videre kreves det tracking av hodet ved bruk av høretelefoner i virtuell virkelighetsapplikasjoner der brukeren er fullstendig omgitt av omgivelsene, for å oppnå presis lokalisering av lydkilden. Da slikt utstyr som regel finnes i denne type systemer, er ikke nødvendigvis dette noe problem (RAVE-loboratoriet på Lerkendal har blant annet relevant utstyr for dette formålet, men igjen er tilgangen til dette begrenset av én samtidig bruker). På

tross av disse utfordringene, er det fortsatt denne teknikken som er ansett å gi best resultater for presis stedfesting av lyd.

## 2.2 Lydbibliotek og programvare

Dette kapittelet vil presentere en del programmeringsbibliotek (API) som er aktuelle under en implementasjon av et system for stedfestet lyd. Det er i denne sammenheng valgt å se bort fra HRTF-baserte systemer på grunn av deres ressurskrav og nødvendigheten av individuelle tilpasninger av systemet. Denne teknikken er likevel interessant på grunn av den høye graden av lokaliseringspresisjon den tilbyr. Utviklingen innen maskinvare fører også til at problemet med beregningsressurser sannsynligvis vil kunne ses bort fra i fremtidige systemer.

Et programmeringsbibliotek for stedfestet lyd gjør det enklere å inkludere lyd i en virtuell virkelighetsapplikasjon, da mye av lavnivåprogrammeringen som må til for å generere lyd kan overlates til dette. De fleste bibliotekene av denne typen krever lydkildens og lytterens posisjon, samt retningen av disse. Denne informasjonen inkludert den faktiske lyden som skal spilles av, er de grunnleggende data som API'et krever for å kunne utføre denne operasjonen (bibliotekene varierer med tanke på annen informasjon som er nødvendig, eksempelvis romklangparametere, materiale på medium etc.). At man på denne måten får abstrahert bort en del detaljer forbundet med maskinvarekonfigurasjon etc., gjør at fokuset kan rettes mot hvordan lyden sømløst skal integreres i den ferdige applikasjonen, samt på lydene som skal brukes.

Det har etter hvert blitt utviklet generelle standarder også på dette området, og dagens lydkort har gjerne innebygd egne algoritmer i maskinvaren for å håndtere stedfestet lyd. Programmeringsbiblioteket sender da denne informasjonen (steds- og retningsinformasjon) direkte til lydkortet som tar seg av beregningene internt på kortet. På denne måten spares hovedprosessoren for regnekraft tilknyttet lyden, og kan da fokusere på grafikken og andre deler av applikasjonen. Det er likevel verdt å merke seg at ulike lydkortprodusenter ikke nødvendigvis bruker de samme algoritmene, og at det derfor kan være forskjeller i hvor godt man er i stand til å gjenskape den stedfestede lyden.

Mesteparten av den innformasjonen som finnes i kapitlene nedenfor er hentet fra [SK01] og [WEB06]. Denne presentasjonen av ulike lyd-API'er er ikke komplett, og er kun ment som en rask oversikt over hvert formats sterke og svake sider. For mer inngående informasjon om hvert bibliotek, henvises det til de enkelte produsenters hjemmesider på internett.

### 2.2.1 OpenAL

OpenAL (Open Audio Library) er et standard API for lydprogrammering laget av Creative Labs og Loki Entertainment. Det var opprinnelig ment som et portabelt alternativ til lyddelen av Microsofts DirectX API (Direct Sound). Prinsippene bak OpenAL ble i stor grad hentet fra eksisterende API'er og syntaksen er inspirert av grafikk-API'et OpenGL. OpenAL har blitt implementert på de fleste større operativsystemer, blant annet IRIX 6.5 (som brukes på grafikkserveren som driver RAVE-laboratoriet på Lerkendal) og Microsoft Windows. Det er derfor et interessant API hvis applikasjonen må kunne brukes på ulike plattformer. OpenAL fungerer på de ulike plattformene som en *wrapper*, det vil si at den fungerer som et

mellomledd mellom brukeren og funksjonaliteten som finnes i operativsystemet. Eksempelvis bruker OpenAL DirectSound3D-driveren under Windows-plattformen.

OpenAL lar brukeren laste inn en lyd og videre definere lydparametrene i den virtuelle verdenen, som for eksempel romakustikk, lytter- og lydkildeposisjon etc., samt å endre disse parametrene dynamisk underveis i simuleringen. Alle lyder er posisjonert i forhold til brukeren, som er definert som senteret i den virtuelle verdenen. OpenAL kan også brukes til å spille av generell bakgrunnsmusikk, og det er også mulig å bruke *streaming* fra en annen kilde for dette formålet. Dessuten støttes både Doppler-effekten og en del andre lydeffekter som for eksempel forandring av lydens tone og frekvensskifting under kjøring.

Den største fordelen med dette API'et er at det støtter ulike plattformer, i motsetning til DirectX og EAX som er knyttet til et spesifikt operativsystem. For mer inngående informasjon om dette biblioteket, se [WEB03].

### 2.2.2 DirectSound3D (DS3D)

DirectSound3D, som er en del av Microsofts DirectX, gir brukeren direkte tilgang til den underliggende maskinvaren. På grunn av den nære tilknytningen med maskinvaren, er dette biblioteket noe vanskeligere å komme i gang med i forhold til OpenAL (som bruker kjent syntaks fra OpenGL). Det har mye av den samme funksjonaliteten som det nevnte OpenAL, men er også forskjellige på ulike områder. Som i OpenAL blir lydfilene lastet inn i et *buffer* og kan etter dette flyttes rundt i omgivelsene. Men i motsetning til OpenAL kan ikke et buffer assosieres med en ny lyd når den første er opprettet, noe som gjør bruken mindre fleksibel.

DirectSound3D har syv ulike lydeffekter innebygd som en del av API'et (inkludert romklang og ekko). Dette gjør at man ikke er avhengig av et tilleggsbibliotek som for eksempel EAX (se under) for å skape romfølelse. En fordel med DirectSound3D over andre lydbiblioteker som for eksempel OpenAL, er at det automatisk støtter såkalt *LiveVoice*. Det vil si at ulike brukere utstyrt med hver sin mikrofon har mulighet til å kommunisere direkte ved hjelp av disse under kjøring av applikasjonen. For mer informasjon om DirectSound3D, se [WEB01].

### 2.2.3 EAX

EAX (*Environmental Audio Extensions*) er ikke et eget lyd-API på lik linje med OpenAL og DirectSound3D, men er ment som et supplement til disse. Det er utviklet av Creative Labs og gir brukeren tilgang til en stor mengde lydeffekter. Hensikten er å generere effekter som er sammenlignbare med det man ville oppnådd ved å bruke en fysisk basert modell (se kapittel 2.1.2.1.1), der en modellerer omgivelsene i datamaskinen og tar hensyn til lydens vei gjennom mediet (hindringer etc.). Sammenlignet med disse metodene, bruker den forholdsvis lite regnekraft og er kompatibel med både OpenAL og DirectSound3D.

Lydeffekter kan enten assosieres med hver enkelt lydkilde, eller kan virke på hele omgivelsen. For å kunne simulere virkningen av et spesifikt roms geometri, er det mulig å angi en lang rekke parametere for hver lydeffekt. Det finnes også 26 predefinerte lydomgivelser der disse parametrene er satt på forhånd (noe som i de fleste tilfeller gir et tilfredsstillende resultat). For mer informasjon om EAX, se [WEB02].

### 2.2.4 A3D

A3D ble opprinnelig utviklet gjennom et samarbeid mellom NASA og Aureal i 1997, og var ment for å brukes i flysimulatorer. Siden den gang, har teknologien utviklet seg gjennom flere versjoner frem til dagens versjon som er 3.0. Den første versjonen av biblioteket var et fremskritt i forhold til Microsofts DirectSound3D-teknologi ved at det var støtte for maskinvareakselerasjon og en mer avansert modell for gjengivelse av romklang og andre atmosfæriske lydeffekter.

Neste versjon av biblioteket, versjon 2.0, ga videre støtte for troverdig gjengivelse av lydens reise gjennom mediet fra lydkilden til lytteren. Ved å analysere omgivelsenes geometriinformasjon, som vanligvis bare sendes til grafikkortet, ble det ved hjelp av teknologien *WaveTracing* mulig å ta hensyn til opptil 60 refleksjoner mellom lydkilde og lytter på grunn av hindringer i omgivelsene.

Nåværende versjon administreres av Creative Technologies, som kjøpte opp Aureal etter at disse gikk konkurs. Denne versjonen forbedrer tidligere utgivelser ved at lydkilder med utstrekning nå er tillatt (for eksempel en foss med rennende vann), flere lydfilformater er støttet (inkludert MP3), samt at streaming av lyd også er gjort tilgjengelig for dette biblioteket (i likhet med OpenAL).

A3D-biblioteket har vært regnet som det teknisk sett beste API'et på grunn av resultatene som kan oppnås, samt den detaljerte fysiske modellen som ligger til grunn for gjengivelsen av romklang-effekter. Etter oppkjøpet, har utviklingen av API'et gått saktere, og DirectSound3D har på de fleste områder innlemmet de samme egenskapene i sitt API. Dette har ført til at det er DirectSound3D som har ligget til grunn for de fleste nye spilltitler [WEB06]. For mer informasjon om A3D, se [WEB07].

### 2.2.5 Andre

Det finnes et stort antall programmeringsbibliotek for stedfestet lyd. Foruten de ovenfor nevnte API'ene, kan VSS og Csound også nevnes i denne sammenhengen. Disse har vært mye brukt tidligere, men har i den senere tiden blitt mindre interessante på grunn av den raske utviklingen på dette området.

Virtual Sound Server (VSS) er et plattformuavhengig verktøy, spesielt egnet for interaktive applikasjoner. Foruten å spille av på forhånd innspilte lyder, har det også mulighet for å generere lyd i sanntid. VSS fungerer etter klient-tjener-prinsippet, der beskjeder går over nettet til en egen lydtjener om når og hvilke lyder som skal spilles av (lydtjeneren kan selvsagt også være den samme maskinen som klienten kjører på, hvis dette er ønskelig). For mer informasjon om dette verktøyet, se [WEB08].

Csound er et omfattende programmeringsverktøy som har støtte for både gjenskapning av lyd og generell signalprosessering. Hovedfokuset er på syntetisering av lyd, men Csound har også vært brukt i applikasjoner for stedfestet lyd. Imidlertid vil det kreves en del mer arbeid hvis dette verktøyet skal brukes for dette formålet, da det ikke finnes innebygd støtte for dette i utgangspunktet. For mer informasjon, se [WEB09].

### 2.2.6 Oppsummering

De foregående kapitlene har kort gjort rede for noen av API'ene som finnes for stedfestet lyd. Tabell 1 gir en oversikt over hvilke plattformer hvert bibliotek har støtte for.

| API | Operativsystem |
|---|---|
| **OpenAL** | MS Windows, IRIX, BSD, Solaris, Linux, Macintosh OS X/9/8, Sony Playstation, Microsoft Xbox, Nintendo GameCube |
| **Microsoft DirectSound3D** | MS Windows, Microsoft Xbox (generelt plattformer som støtter DirectX) |
| **EAX** | MS Windows |
| **A3D** | MS Windows |
| **Virtual Sound Server (VSS)** | MS Windows, IRIX, Linux |
| **CSound** | MS Windows, Linux, Macintosh OS X |

**Tabell 1 – Lyd-API og operativsystem**

Som det fremgår av tabellen, er alle bibliotekene kompatible med Microsoft Windows-plattformen, noe som kan forklares med at valget av hvilke API'er som er presentert har tatt utgangspunkt i at applikasjonen som skal utvikles i denne oppgaven vil kjøre på dette systemet.

## *2.3 Lyd i OpenGL Performer*

Dette kapittelet beskriver hvordan lyd kan implementeres som en del av den eksisterende applikasjonen som er laget i OpenGL Performer. Datastrukturen i dette verktøyet bygger på scenegraf-prinsippet, som i utgangspunktet er ment for å strukturere grafikken i scenen. Objekter som er plassert nær hverandre i rommet, eller på en annen måte har en romlig sammenheng, blir vanligvis også plassert "nær" hverandre i scenegrafen. Dette gjør at scenen kan optimaliseres under kjøring, ved at store deler av treet automatisk kan ses bort fra på grunn av at det ikke vil være synlige (eller ligger bak andre objekter).

Når det gjelder lyd i applikasjoner som bygger på scenegraf-prinsippet, er det delte meninger om det er fordelaktig å innlemme lyden som en del av scenegrafen, eller holde denne separat som en egen prosess ved siden av geometrien (se [WEB10] for en interessant diskusjon rundt dette). I teorien er det mulig å lage lydnoder, som på lik linje med andre geometrinoder i scenegrafen, kan plasseres under transformasjonsnoder. En lydnode kan på denne måten assosieres med en geometrinode, og gjennomgå de samme transformasjoner (translasjon, rotasjon etc.) som den tilhørende geometrinoden. Det er derfor mulig å assosiere en lydnode med for eksempel geometrinoden for et bilobjekt, og la denne noden gjennomgå de samme transformasjoner som bilnoden. Lydnoden vil da automatisk få oppdatert korrekt posisjon og retning i tråd med det tilhørende bilobjektet (hvis denne for eksempel er i bevegelse). Dette letter i stor grad arbeidet med å opprettholde bindingen mellom lyd og grafikk i scener som tillater intern forflytning av objektene (og ikke kun synsvinkelen). Selve rendering-løkken kan da deles inn i to steg; i det første steget oppdateres og rendres grafikken som i en vanlig rendering-løkke, mens i det andre steget blir scenegrafen gjennomgått for alle lydnoder. Dette skillet er logisk siden grafikk og lyd uansett "rendres" av to ulike prosesser.

Et ankepunkt mot løsningen som er skissert over har vært at lyd og grafikk er vesensforskjellige, og at de prinsippene som gjelder for grafikk ikke nødvendigvis kan tilpasses lyd på en effektiv måte. Et eksempel på dette er *culling*, der visse deler av scenegrafen blir hoppet over under rendering på grunn av romlig plassering i forhold til synsvinkel. For lyd vil dette ofte være uhensiktsmessig, da det er mulig å høre lyder selv om man ikke nødvendigvis ser lydkilden. Eksempelvis vil lyder med stor amplitude eller lav frekvens, ha et mye større "virkeområde" enn tilsvarende lave lyder med høy frekvens. En lyd med spesiell høy amplitude kan i det ekstreme tilfellet være hørbar på alle steder i scenen, og dette er ikke rett frem å implementere i en scenegrafstruktur. En løsning er å plassere lyden høyere opp i hierarkiet, men dette kan lett motvirke den opprinnelige hensikten med å bruke en scenegraf (romlig tilhørighet eller funksjonell samhørighet mellom objekter i scenen).

## 2.4 Lyd i Sverresborgmodellen

Foruten den rent tekniske delen ved innføring av lyd i Sverresborgmodellen, vil det være like viktig å avgjøre hvilke lyder som faktisk skal høres under simuleringen. Eksempler kan være gjøende hunder, fotsteg når en går, samtaler i bakgrunnen som ikke nødvendigvis er interessante for brukeren, lyden av arbeidende håndverkere, bakgrunnsmusikk og så videre. Bruk av lyd gjør det mulig å gi inntrykk av at omgivelsene er mer detaljrike enn de i virkeligheten er. Selv om man kanskje ikke har tid eller ressurser til å modellere og animere en arbeidende smed, er det mulig å spille av den tilhørende lyden uten den grafiske modellen. De fleste lyttere vil på denne måten få en sterkere følelse av innlevelse enn om man kun spilte av lyder for objekter som faktisk var modellert. Denne effekten har vært studert i blant annet dataspill, og det er påvist at for to identiske geometriske modeller, vil brukeren oppfatte den modellen som har stedfestet lyd som den mest detaljrike (også grafisk sett). Stedfestet lyd kan derfor spille en viktig rolle for systemer som ikke har nok regnekraft til å vise detaljerte og/eller store modeller. Det er her likevel viktig at brukeren er klar over at ikke alle lyder nødvendigvis har en synlig kilde, slik at de for eksempel ikke begynner å gå rundt å lete etter en bestemt lydkilde som de hører lyden av.

På grunn av at Sverresborgmodellen har to mulige innfallsvinkler, både en arkeologisk og i et underholdningsperspektiv, vil det også være visse motstridende hensyn som må tas under valget av lyder. Fra et arkeologisk ståsted, er det kun de lydene som faktisk var til stede på borgen på den tiden som er av interesse. Det vil si at det legges størst vekt på bakgrunnslyder fra hverdagslivet på borgen. Hunder som gjør, barn som leker, håndverkere som arbeider, bygningsarbeidere på murene, diverse samtaler og diskusjoner i bakgrunnen, samt vind, fuglekvitring, fotsteg og andre naturlige lyder. Hensikten med denne typen modeller, er å på en mest mulig objektiv måte beskrive miljøet og omgivelsene, så historisk korrekt som mulig. Siden krigshandlinger også på denne tiden var forholdsvis sjeldne, er det gjerne hverdagslivet på borgen som er av interesse fra dette ståstedet. Det var i fredstid et yrende liv av både soldater og vanlige folk innenfor borgmurene, og kvinner og barn var også representert. For brukere uten spesielle interesser innen arkeologi, kan nok dette i lengden bli oppfattet som noe "kjedelig".

Fra et underholdningsmessig ståsted, vil bruk av mer "eksotiske" lyder gjøre opplevelsen av simuleringen mer interessant for den vanlige bruker. I dette tilfellet er bakgrunnsmusikk den kanskje mest brukte stemningsskaperen i filmer og dataspill. Valg av bakgrunnsmusikk kan være avgjørende for hvordan man oppfatter stemningen i en scene (et eksempel fra filmindustrien kan være forskjellen mellom stemningen i en thriller og et drama, som

underbygges av den tilhørende bakgrunnsmusikken). For å gjøre opplevelsen mer realistisk, kan musikk fra denne spesifikke tidsperioden brukes. Selv om det er tvilsomt om det ble spilt slik musikk i bakgrunnen i en hverdagssituasjon, vil de fleste brukere assosiere denne typen musikk med denne tidsperioden. For å skape stemning og innlevelse, kan handlingen legges til krigsfylte perioder i borgens historie, noe som skaper behov for enda flere lydeffekter (flyvende piler, klingende sverd etc.). Ellers er det også fra dette perspektivet viktig med bakgrunnslyder (både naturlige og menneskeskapte) for å gi et rikere inntrykk av modellens detaljer.

# 3 Vurdering og valg

Dette kapittelet vil skissere den valgte fremgangsmåten for å implementere lyd i den eksisterende Sverresborgmodellen. Denne modellen har gjennomgått flere iterasjoner med forbedringer og utvidelser gjennom ulike studentprosjekt og oppgaver frem til dagens versjon. For en inngående beskrivelse av systemet som er blitt laget frem til dags dato, refereres det til [KP01], [SJO01] og [FA01] som er den nyeste versjonen. Den eksisterende applikasjonen ble opprinnelig utviklet for kjøring i RAVE-laboratoriet ved NTNU. Grafikkserveren som driver dette anlegget, kjører operativsystemet IRIX 6.5 på en Silicon Graphics Onyx2-maskin. Til denne maskinen er det tilkoblet en del proprietært tilleggsutstyr, blant annet et sporingssystem til bruk under simuleringen. En egen styringsmekanisme, kalt en *wand*, brukes for forflytning og som pekestokk i applikasjonen. Det har tidligere ikke vært implementert lyd i Sverresborgapplikasjonen, men relevant lydutstyr finnes tilgjengelig i laboratoriet.

Det har vært et ønske om at applikasjonen også skal kunne benyttes på Microsoft Windows-plattformen. Det er derfor et pågående prosjekt parallelt med denne oppgaven med hensikt å gjøre dette mulig. OpenGL Performer, som den eksisterende applikasjonen er bygget på, eksisterer også i en Windows-utgave, men det er likevel ikke en rett frem oppgave å overføre systemet til denne plattformen. Noe av grunnen til dette er at grafikkbiblioteket CAVELib er nært knyttet opp mot systemet. Dette biblioteket tar hånd om en del utfordringer når det gjelder visning av grafikkapplikasjoner i RAVE-miljøet (blant annet for stereovisning på flere flater), samt at styrings- og sporingssystemet også er knyttet til dette. Da mye av grunnen for å benytte dette verktøyet faller bort på Windows-plattformen, samt at programmet krever lisens for å kjøre, er det ønskelig at Windows-versjonen ikke benytter seg av CAVELib (selv om CAVELib også finnes i en Windows-versjon).

Grunnlaget for denne oppgaven blir derfor Windows-versjonen av applikasjonen, noe som vil kreve parallell jobbing mellom arbeidet som blir gjort under konverteringen av programmet til ny plattform og utviklingen av lydsystemet.

Som nevnt i kapittel 2.2, finnes det i dag en lang rekke verktøy for å lette arbeidet med utvikling av lyd til bruk i grafiske applikasjoner. Disse tilbyr programmereren et grensesnitt mot den tilgjengelige maskinvaren som ligger på et forholdsvis høyt nivå. Dette betyr at en del detaljer vedrørende gjenskapningen av lyd kan abstraheres bort, og at fokuset kan flyttes over mot hvordan man på en best mulig måte kan integrere lyden som en fornuftig del av den eksisterende grafiske applikasjonen. Det vil i denne sammenheng kunne settes spørsmålstegn ved den forholdsvis detaljerte gjennomgangen av lydens (fysiske og psykoakustiske) egenskaper i tidligere kapittel, men det har likevel vært viktig å danne et solid teoretisk grunnlag for den faktiske implementasjonen, samt en forklaring på en del begreper som vil brukes i det videre arbeidet.

## *3.1 Valg av API for stedfestet lyd*

Fokuset for denne oppgaven vil være Windows-versjonen av programmet. Det har likevel vært interesse for at også den versjonen som kjøres i RAVE-laboratoriet ved NTNU, skal ha mulighet for å spille av lyd. For å gjøre det arbeidet som utføres i denne oppgaven mest mulig

relevant også med tanke på dette formålet, er det derfor nærliggende å se på lyd-API'er som er tilgjenglige også på denne plattformen (IRIX). Fra Tabell 1 er det mulig å konkludere med at det vil være OpenAL eller VSS som er de to bibliotekene som oppfyller dette kravet. Valget mellom disse er forholdsvis åpent, men som det fremgår fra diskusjonen av VSS i kapittel 2.2.5, er dette et eldre og et ikke like godt dokumentert verktøy som OpenAL. Dessuten tilbyr OpenAL en bedre støtte for maskinvareakselerasjon på Windows-plattformen enn det som tilbys av VSS (for IRIX-plattformen er verktøyene like gode med tanke på dette).

Det mest nærliggende er derfor å velge OpenAL for denne implementasjonen, noe som vil kreve mindre arbeid hvis systemet skal overføres til IRIX. OpenAL bruker noe av DirectSound3Ds funksjonalitet mot maskinvaren når det skal generere lyder, noe som angis under initialisering av programmet. Det er i all hovedsak denne delen som må byttes ut for en eventuell IRIX-versjon.

OpenAL åpner også for muligheten av å bruke eksterne lydbibliotek for håndtering av andre lydparametere. Det er blant annet mulig å bruke EAX som verktøy for å gjenskape romklang og andre effekter som tilskrives mediet som lyden reiser gjennom (noe OpenAL har begrensede og i enkelte tilfeller ikke-eksisterende muligheter for). EAX versjon 2.0 er gratis tilgjengelig fra blant annet nettstedet [WEB11]. Dette biblioteket vil derfor bli brukt for å simulere de romavhengige egenskapene ved lyden i applikasjonen. Som nevnt tidligere, har både OpenAL og EAX mulighet for maskinvareakselerasjon der dette er tilgjengelig (de fleste nyere lydkort). EAX er i motsetning til OpenAL ikke tilgjengelig for IRIX-plattformen, men dette vil ikke skape større problemer for en eventuell portering til dette systemet, da det er mulig å hoppe over denne funksjonaliteten under oppstart av applikasjonen hvis systemet ikke finner de nødvendige bibliotekene (gjelder også for Windows-systemer som ikke har dette biblioteket installert).

## *3.2 Lyder i modellen*

Tidligere versjoner av Sverresborgmodellen har i stor grad fokusert på modellering av selve borgbygningen og omkringliggende bakkelandskap. Det er få eller ingen objekter i den nåværende modellen som genererer lyder naturlig av seg selv (mennesker, dyr, redskap etc.). På grunn av en utstilling av prosjektet på Sverresborg Folkemuseum sommeren 2005, vil det i den sammenheng bli modellert noen ekstra objekter som skal plasseres rundt omkring inne i borgen. Disse vil være potensielle kilder til lyd, og vil også inngå i modellen som dette arbeidet resulterer i. Da samarbeidet med museet om hvilke objekter som skal modelleres og hvordan disse skal se ut ikke har en fastsatt plan, er det vanskelig å på forhånd skulle skissere hvilke lyder som til slutt vil være nødvendige å lage eller eventuelt innhente.

Uansett hvilke fysiske objekter som faktisk blir modellert, er det mulig å legge inn lyder som ikke nødvendigvis har noen spesifikk lydkilde. Sverresborgapplikasjonen er ikke bare ment å være en hundre prosent korrekt gjengivelse av en middelalderborg (selv om dette etterstrebes), men også en arena å prøve ut nye teknikker og metoder på. I denne forbindelse er det derfor også interessant å skape et lydbilde som er så rikt som mulig, noe som kan øke innlevelsen og opplevelsesverdien av modellen. Som nevnt tidligere, vil ikke brukeren nødvendigvis sette et spørsmålstegn ved om en lyd faktisk har en observerbar lydkilde i den virtuelle verdenen. Tvert imot kan den ekstra dimensjonen som lyden skaper, faktisk øke brukerens inntrykk av de grafiske detaljene i simuleringen.

I samarbeid med arkeolog Regin Meyer ved Sverresborg Folkemuseum, er det blitt identifisert flere lyder som ville ha vært vanlige i den daglige driften av borgen. Det er valgt å dele disse inn i *bakgrunnslyder* og *spesifikke lyder*. Med bakgrunnslyder menes de naturlige lydene som har sitt opphav i naturen (vær og vind), og som derfor ikke har sitt opphav i noen spesifikk lydkilde som sådan. De spesifikke lydene inkluderer de lydene som er menneskeskapte, og derfor de man gjerne kan knytte til en spesifikk kilde. Listen under er ikke ment å være uttømmende, men representerer et utvalg av de mer interessante lydene man antar var vanlige under fredstider:

- Bakgrunnslyder:
    - *Vind*. Da borgen ligger forholdsvis utsatt til på en høyde i terrenget, kan det i enkelte perioder være sterk vind rundt borgen.
    - *Vann*. Borgen har en brønn som ville ha vært en potensiell lydkilde. Dessuten forekom regn selvsagt av og til.
    - *Fuglekvitring*. Fuglekvitring ville ha vært en naturlig bakgrunnslyd under rolige omstendigheter på borgen.
    - *Bakgrunnsmusikk*. Som nevnt over er bakgrunnsmusikk et stridsspørsmål når det gjelder slike gjengivelser. Fra et arkeologisk ståsted, ønsker en ingen spesiell bakgrunnsmusikk når en skal gjengi det daglige virket i borgen. Det er likevel klart at et relevant musikkvalg kan skape en ekstra atmosfære for brukeren. Slik sett bør derfor bakgrunnsmusikken være mulig å slå av og på, ettersom hvilket bruksområde en ønsker.

- Spesifikke lyder:
    - *Fotsteg*. Lyden av fotsteg kan assosieres både med brukeren selv (under forflytning) eller med andre personer/dyr i modellen.
    - *Hundegjøing*. Det er kjent at det har vært hunder på borgen. Enkelte hundebjeff kan derfor være en effektiv indikator på dette i simuleringen.
    - *Lyder fra en smed*. Det har (spesielt i fredstid) vært håndverkere på borgen. Spesielt har det vært en eller flere smeder der. Lyder som vil være aktuelle i denne sammenheng er derfor hammerslag mot ambolt, lyden av et bål eller en grue, gnidning av jern mot jern (sliping) etc.
    - *Snakkende mennesker*. Samtaler mellom mennesker på borgen kan være et virkemiddel for å gi informasjon til brukeren på. Samtaler, latter etc. vil dessuten være en naturlig del av bakgrunnslyden når en antar at det minst har vært 80 personer på borgen bare i fredstid. Det er ikke nødvendig at hver av disse samtalene skal kunne overhøres ord for ord, men de kan gi inntrykk av at det er flere personer tilstede enn det antallet som faktisk er modellert og til enhver tid vises.
    - *Kvinner og barn*. Barnerop og latter kan brukes til å effektivt fortelle brukeren at det (i fredstid) både var barn og kvinner på borgen.
    - *Lyder fra vindebro*. Vindebroen som markerer inngangen til borgen har vært mulig å lukke ved hjelp av kjettinger på hver side. Lyden av kjettinger er derfor en lydeffekt som kan brukes når denne senkes eller heves.
    - *Lyder fra rustninger og uniformer*. Hverdagsklær var i denne perioden vanligvis ”myke” klær av lær og skinn etc. Rustningene var derimot ofte laget av metall, blant annet ringbrynjer som vaktsoldater brukte også i fredstid.

Disse lager karakteristiske metalliske lyder som kan være aktuelle når en kommer nær en slik soldat.

- *Våpenlyder*. Lyder fra spesifikke våpen er av mindre interesse når en skal gjengi livet på borgen i fredstid, men kan være relevante hvis trening og annen aktivitet modelleres. Lyden av klingende sverd og skjold vil da være aktuelle.

Under implementasjonen av disse lydene, vil det sannsynligvis være de spesifikke lydene som vil være av interesse å stedfeste. Disse må derfor knyttes til en eventuell fysisk lydkilde (en geometrisk modell), eller i de tilfeller der denne modellen mangler, gis en posisjon og retning i rommet. Da en del av de nevnte lydene er spesielle lyder som kun vil forekomme på gitte tidspunkter (for eksempel heving eller senking av vindebroen), er det videre nødvendig å avgjøre når en gitt lyd skal spilles av. Et nyttig hjelpemiddel for å bestemme tilstanden til en gitt lydkilde, kan være en enkel tilstandsmaskin med for eksempel tilstandene spiller, stoppet, pauset etc. For de fleste lydkilder vil det derimot være tilstrekkelig å spille av lyden én gang, eller eventuelt å gjenta lyden kontinuerlig (såkalt *loop*).

## *3.3 Integrasjon med eksisterende applikasjon*

Dette kapittelet vil, på et overordnet nivå, gå gjennom hva som må endres i den eksisterende Sverresborgapplikasjonen for å kunne innlemme stedfestet lyd. Som nevnt innledningsvis i dette kapittelet, kan inngående informasjon om det eksisterende programmet finnes fra tidligere prosjekters dokumentasjon, der [KP01], [SJO01] og [FA01] beskriver de siste versjonene. For dette prosjektet er det besluttet å bygge videre på arbeidet som ble utført av en Kundestyrt Prosjektgruppe høsten 2003 (se [KP01] for mer detaljer om dette). Grunnen til at det ikke er siste versjon av programmet som ønskes brukt (inkludert den virtuelle guiden og muligheten for animerte objekter), er den sterke koblingen mot CAVELib-biblioteket som nevnt tidligere. Det har vært et mål at Windows-versjonen av programmet skulle løsrive seg fra dette verktøyet, da det krever en forholdsvis dyr lisens og at nytteverdien av å bruke det var knyttet til kjøring av programmet i RAVE-miljøet.

### 3.3.1 OpenAL

OpenAL har, som nevnt tidligere, en forholdsvis lik logisk oppbygning som OpenGL. Man definerer under oppstart av programmet en *kontekst*, der bindingen mot den underliggende maskinvaren blir definert. Under denne initieringen, avgjøres også hvilket lydutstyr som er tilgjengelig på vertsmaskinen. Dette gjør at programmet med letthet kan flyttes til ulike maskiner (med ulike oppsett), uten å måtte ta spesiell høyde for dette under utviklingen; OpenAL vil selv bruke det oppsettet som er tilgjengelig.

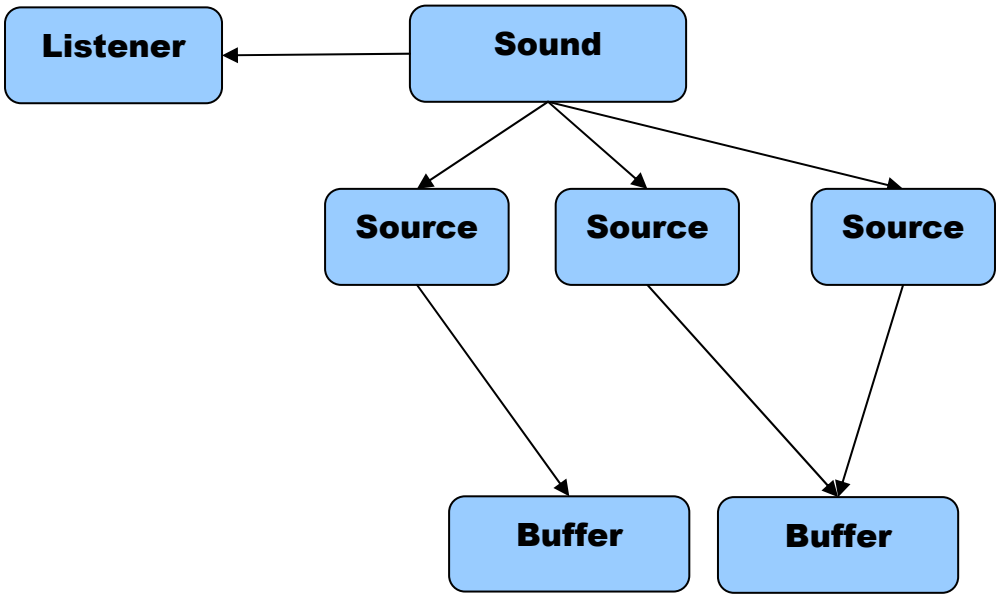
Foruten konteksten, er det tre andre konsepter som er sentrale i OpenAL: *listener*, *source*, og *buffer*. Listener er, som navnet tilsier, parametere knyttet til lytteren eller brukeren av applikasjonen. Posisjon, retning og hastighet er data som må defineres for dette objektet.

Source er knyttet til de ulike lydkildene i modellen. I motsetning til listener, vil det som regel finnes flere source-objekter i scenen. Posisjon, retning og hastighet er basisparametere som også er knyttet til source, men i tillegg er det flere spesifikke verdier som kan settes for å oppnå den ønskede funksjonaliteten til en lydkilde. Den overordnede lydstyrken kan settes uavhengig av avstanden fra lytteren. Dessuten kan lydens tonehøyde endres manuelt ved å

sette denne spesifikt (er implementert ved å endre avspillingshastigheten). Lydkilden befinner seg til enhver tid i en av tilstandene *playing*, *paused*, *stopped* eller *idle*, og det er også mulig å angi parameteren *loop* for å spille av den samme lyden suksessivt flere ganger.

Buffere er lagringsplass for alle data som er knyttet til en spesifikk lyd. Med dette menes selve lydfilen (innlasting og lagring av lydfilen i temporært minne) og informasjon om lydens format. Frekvens, bit-rate, lengde og oppløsning er parametere som blir lagret sammen med lyden, og kan hentes ut av applikasjonen hvis dette er ønskelig. Buffere assosieres så med en source (som altså ikke selv lagrer lyddata). Et buffer kan på denne måten knyttes til flere ulike source-objekter, noe som sparer minne under kjøring av programmet. Bindingen mellom et buffer og et source-objekt er ikke statisk, slik at det er mulig å endre lyden som hver lydkilde spiller av under kjøring.

Da OpenAL ikke er laget spesifikt med tanke på det objektorienterte paradigme, kan det være greit å gjøre dette på egen hånd for å skape en bedre oversikt over bindingene mellom de ulike delene. Figur 3-1 viser et eksempel på klassehierarkiet for lyddelen av applikasjonen.




**Figur 3-1 Klassediagram for objekter knyttet til lyddelen av applikasjonen**

Som det fremgår av figuren, er Sound hovedklassen som inneholder referanser til både lytterdataene (gjennom Listener-bindingen), lydkildene (gjennom Source-bindingene) og indirekte selve lyddataene som er koblet til Source-objektene. Hver av disse klassene har ansvar for de tilsvarende konseptene Listener, Source og Buffer i OpenAL. De fungerer på denne måten som en *wrapper* for disse, og har derfor også metoder som henter ut og setter tilsvarende parameterverdier som er assosiert med disse konseptene. Denne delingen sørger for at Sound, som er hovedklassen og vil være bindeleddet mellom den eksisterende applikasjonen og lyddelen, blir mindre kompleks og mer oversiktelig.

Det har vært mulig å la Buffer-klassen være direkte assosiert med Sound-klassen, men da denne vanligvis benyttes kun under initialisering av programmet, er det valgt å flytte denne nedover i hierarkiet. Det er dessuten bare Source-objekter som har direkte bruk for dataene

som finnes i Buffer-objektene, og det er derfor naturlig å la disse befinne seg under Source-objektene.

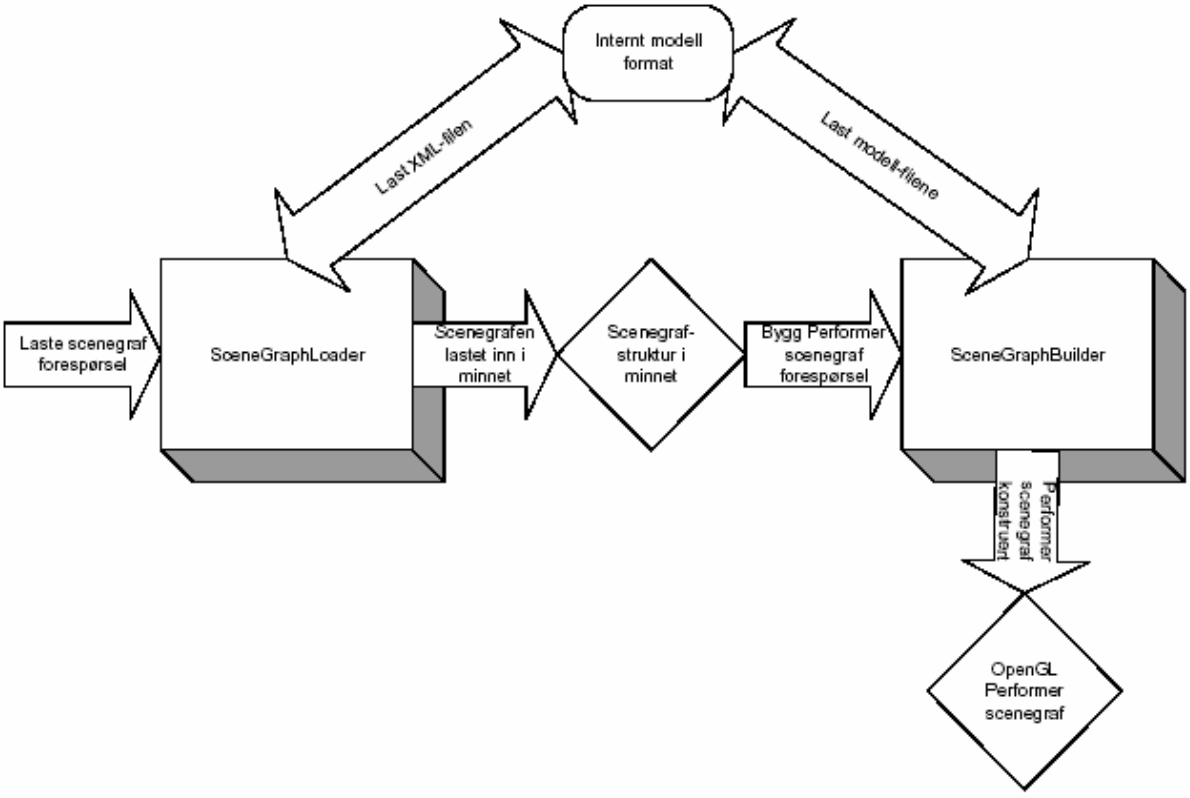
### 3.3.2 OpenGL Performer og scenegraf

Den eksisterende Sverresborgapplikasjonen er, som nevnt over, implementert ved bruk av OpenGL Performer. Denne bygger opp en scenegraf, der de geometriske objektene som inngår i modellen blir lastet inn i en trestruktur. Under oppdatering av scenen for hvert nytt bilde som skal vises, traverserer applikasjonen så dette treet og klipper bort greiner som ikke vil være synlige på grunn av plasseringen av objektene på denne greinen (scenegrafprinsippet har for øvrig også en del andre fordeler, men disse vil ikke bli videre gjennomgått her).

Sverresborgmodellen har, etter flere modifikasjoner fra ulike studentprosjekter og hovedoppgaver, blitt relativt kompleks. Den består av ulike klasser som hver har en spesifikk oppgave under simuleringen. For dette prosjektet vil det derfor kun være aktuelt å beskrive og nevne de mest relevante delene av modellen som har betydning for implementeringen av lyd. For en fullstendig beskrivelse av alle klasser og avhengigheter, refereres det til [KPO01].

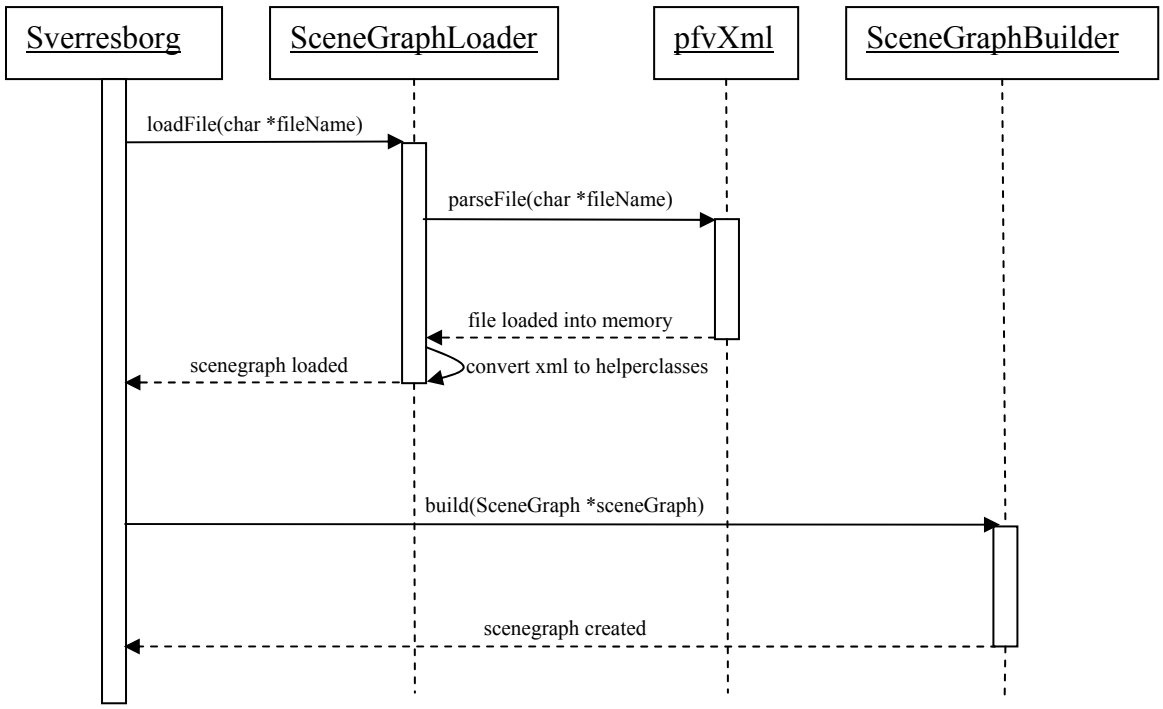
#### 3.3.2.1 Lasting og initialisering av lyd

Før noen lyder kan spilles av, er det nødvendig å initialisere lydsystemet slik at det kobler seg opp mot maskinvaren på maskinen. Denne funksjonaliteten blir tatt hånd om av klassen Sound, som nevnt over. Videre må modellene og de faktiske lydene som skal spilles av lastes inn i applikasjonen og tilegnes en bestemt lydkilde. Alt dette gjøres kun en gang under oppstart, og den overordnede flyten i denne prosessen er vist i figuren under (hentet fra [KPO01]).




**Figur 3-2 Overordnet skjema for innlasting av scenegrafen**

Figur 3-3 (modifisert fra [KPO01]) viser den mer detaljerte flyten av metodekall for innlasting av modeller og lyd under oppstart av applikasjonen.




**Figur 3-3 Sekvensdiagram for lasting av modeller og lyd inn i scenegrafen**

Første skritt i prosessen er å lese en definisjonsfil, nærmere bestemt *scenegraph1.xml*, der alle modeller som skal inngå i simuleringen er definert med navn og hvor filen ligger lagret. Denne definisjonsfilen kan spesifiseres av brukeren i en hvilken som helst teksteditor, og gjør at man kan sette inn og fjerne objekter i modellen uten å måtte rekompilere programmet mellom hver gang. Filen har også en bestemt struktur, slik at brukeren kan tilpasse scenegrafen til sitt behov (hvilke objekter som skal være under, eventuelt likestilte med et gitt objekt i scenegrafen og så videre). Figur 3-4 viser et eksempel på en definisjonsfil der objektet *smed* blir tilegnet en lyd.

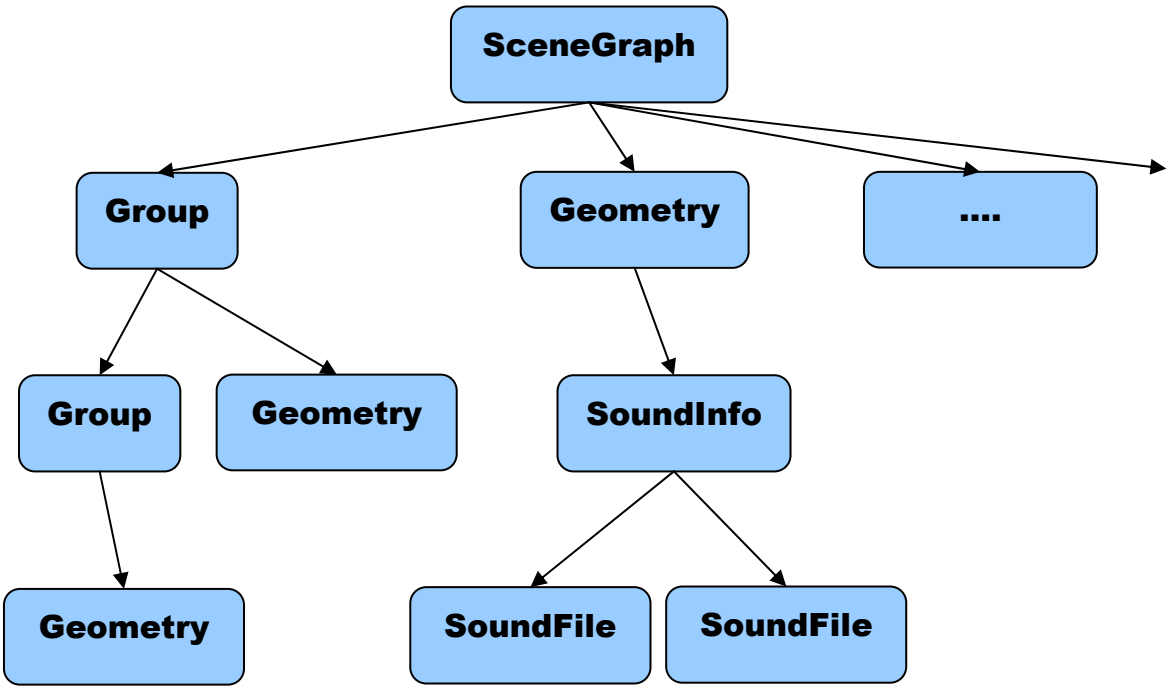
```
<geometry name="smed" selectable="true" sound = "true">

    <lod>
       <lodfile src="models/smed.pfb" range="500" />
    </lod>

    <sound>
       <soundfile src="sounds/smed.wav" xpos="1.0" ypos="21.0" zpos="-2.0"
       loop="true" />
    </sound>

</geometry>
```

**Figur 3-4 Eksempel på ny struktur for XML-definisjonsfilen for å kunne laste inn lyd**

Den initielle posisjonen til lydkilden defineres også i denne filen. Grunnen til at denne må settes spesifikt under oppstart, er at OpenGL Performer under innlasting posisjonerer geometrimodellene ut fra hvilken posisjon de er lagret i sitt lokale koordinatsystem. Dette er praktisk for geometrien, da en kan posisjonere de ulike modellene i forhold til hverandre i 3D Studio MAX, lagre hver enkelt modell i hver sin fil, og så laste dem rett inn i scenegrafen. Modellene vil da havne på samme lokasjon som de hadde da de ble lagret (i forhold til de andre objektene i modellen). Dette gjør derimot at alle modellene får samme posisjon i verdenskoordinatsystemet i OpenGL Performer (nærmere bestemt i origo). Dette fører dermed til at dersom en ikke spesifiserer hver lydkilde med posisjon, vil alle lydene høres ut som om de kommer fra sentrum i modellen. Dette er imidlertid ikke et problem i den videre simuleringen, siden transformasjonsmatrisene for hvert objekt som er flyttet kan hentes ut. Hvis denne matrisen også assosieres med lydkilden, vil denne derfor transformeres (relativt verdenskoordinatsystemet), like mye som objektet og dermed havne på samme posisjon.

Som det fremgår av Figur 3-3, er det klassen *SceneGraphLoader* som har ansvaret for å parse definisjonsfilen. Basert på denne beskrivelsen, bygger klassen et tre av hjelpeobjekter som ligner på den ferdige scenegrafen, men som ikke er leselig av OpenGL Performer. Hjelpeobjektene er definert i filen *HelperClasses*, og inneholder blant annet klasser for å lagre den innleste informasjonen fra definisjonsfilen. For å gjøre programmet mest mulig fleksibelt, bør lyden derfor også kunne defineres sammen med grafikken i denne filen. Relevante metodekall for å lagre denne informasjonen innlemmes derfor som en del av SceneGraphLoader.

Figur 3-5 illustrerer hierarkiet av relevante hjelpeklasser i den temporære trestrukturen som blir bygget av SceneGraphLoader.




**Figur 3-5 Eksempel på hierarkiet av noder i den temporære trestrukturen av hjelpeklasser**

Et *Geometry*-objekt (som altså inneholder informasjon om et gitt objekt i modellen), har bindinger til et *SoundInfo*-objekt hvis det er en lydkilde (Geometry-noder har for øvrig også

barnenoder som lagrer informasjon om modellen som skal lastes inn, men dette er ikke vist i figuren). Dette objektet har i sin tur bindinger til ett eller flere *SoundFile*-objekter, som lagrer informasjonen om en spesifikk lydfil som ble lest inn under parsingen av XML-filen.

Neste skritt i initialiseringen er derfor å bygge selve scenegrafen som programmet kan bruke under simuleringen. Dette gjøres av *SceneGraphBuilder* ut fra den temporære datastrukturen som ble bygget av SceneGraphLoader. Denne klassen traverserer den temporære trestrukturen av hjelpeklassenoder, og kaller relevante metoder for de ulike nodetypene (for eksempel Group-, Geometry- og SoundInfo-noder). Disse metodene oppretter så scenegrafen ved å opprette de tilsvarende OpenGL Performer nodene på riktig plass i trestrukturen. Dette resulterer i en OpenGL Performer scenegraf og brukes under kjøring av applikasjonen.

Lastingen av modeller og lyder initieres av klassen *uiWin*. Dette skjer gjennom et kall til klassen *Sverresborg*, som igjen kaller de relevante metodene i SceneGraphLoader og videre SceneGraphBuilder. Det er også denne klassen som inneholder selve renderingløkken i programmet og tar seg av input fra brukeren. Lyden initialiseres derfor i denne klassen under oppstart, og dette blir det sentrale bindeleddet mellom den eksisterende applikasjonen og lyddelen av systemet.

Input fra brukeren under kjøring, det vil si i forbindelse med forflytning av brukeren inne i modellen, skjer som nevnt i klassen uiWin. Det er derfor nærliggende å også plassere oppdateringen av lytteren i lydsystemet i denne klassen. Lytteren er, som nevnt over, definert i klassen SoundListener, og oppdateringer av posisjon, retning, hastighet etc. av denne skjer derfor gjennom denne. Ved å oppdatere feltene i SoundListener-objektet i tråd med oppdateringer av posisjonen og retningen til brukeren i modellen, vil lyden som lytteren hører være konsistent med det han/hun samtidig ser i modellen.

OpenAL baserer seg på et høyrehånds koordinatsystem, der x-aksen peker mot høyre, y-aksen peker rett oppover og z-aksen peker ut av skjermen. OpenGL Performer baserer seg derimot på at x-aksen peker mot høyre, y-aksen peker inn i skjermen og z-aksen peker rett opp. Dette gjør at det er nødvendig å konvertere brukerens posisjon og retning (som definert i OpenGL Performers koordinatsystem), til koordinatsystemet definert av OpenAL.

Under kjøring foregår avspilling av lyder ved at applikasjonen foretar kall til Sound-objektet (som igjen fungerer som bindeleddet mellom det grafiske systemet og lyddelen). For å gjøre det mulig å spille av lyder til bestemte hendelser, er det nødvendig at hvert objekt som er tilknyttet en lyd har en egen tilstandsmaskin som avgjør når og hvilken lyd som skal spilles av. I det ideelle tilfellet hadde den beste løsningen vært en generell tilstandsmaskin som kunne bli brukt av flere ulike typer objekter (for eksempel en menneskelig modell og et sverd), men på grunn av at disse objektene kan være forholdsvis vesensforskjellige, er det muligens enklere å lage ulike tilstandsmaskiner for ulike objekter (eksempelvis en for mennesker, en for verktøy etc.).

# 4 Implementasjon

Dette kapittelet gjennomgår hva som er blitt gjort av implementasjon basert på de tidligere kapitlene. Hva som er blitt programmert og hva som er endret i den grafiske modellen blir gjennomgått. Kapittel 3 gikk forholdsvis grundig gjennom hvilke endringer som var nødvendige å utføre på den eksisterende applikasjonen for å kunne innlemme stedfestet lyd. Både verktøy og den overordnede arkitekturen for det ferdige resultatet ble skissert, og dette kapittelet vil derfor beskrive de praktiske konsekvensene av disse endringene. Siden kapittelet også beskrev forholdsvis detaljert hvordan applikasjonen skulle utvides, vil det følgende kapittelet kun beskrive de viktigste endringene som er gjort under implementasjonen. Alle endringer som er gjort i den eksisterende koden er kommentert og innledet med teksten:

```
// added by Bård Terje Fallan
```

Ellers er nye metoder og klasser innledet med teksten:

```
/**
 * En beskrivelse av hva klassen/metoden gjør
 *
 * @author Bård Terje Fallan
 * @date 21.04.05
 * @param Eventuelle paramtere for klassen/metoden
 */
```

Det refereres derfor til kildekoden vedlagt på CD for alle detaljer vedrørende programmeringsdelen av oppgaven. Alle 3D Studio MAX modeller/animasjoner, lyder som er brukt, kildekode, samt den kompilerte applikasjon er dessuten vedlagt på medfølgende CD.

Som nevnt i innledningen av rapporten, er alt arbeid knyttet til utstillingen på Sverresborg Folkemuseum presentert på slutten av dette kapittelet. Det vil der bli gjennomgått hva som er gjort på selve modellen med tanke på grafiske endringer. Det vil der også bli gjort rede for hvorfor løsningen med en egen videofilm ble valgt. En vurdering av resultatet som ble oppnådd under dette arbeidet kommer i kapittel 5.

## *4.1 Endringer av eksisterende applikasjon*

Som nevnt i kapittel 3.3 er det flere endringer som må gjøres for at den eksisterende applikasjonen skal kunne spille av stedfestede lyder. Dette kapittelet tar for seg hver klasse som er endret og skisserer hva som er gjort. Enkelte endringer kan være forholdsvis små og utenfor den sentrale funksjonaliteten som blir lagt til, og disse vil ikke bli beskrevet i detalj (kildekoden er som nevnt vedlagt på CD, og det refereres til denne for en full dokumentasjon av systemet).

Rekkefølgen av kapitler reflekterer den logiske gangen som lyden gjennomgår, fra den defineres i XML-fila, til den blir lastet inn og spilt under kjøring av applikasjonen.

### 4.1.1 Klassen Sound

Klassen Sound fungerer som kontaktpunktet mellom den eksisterende applikasjonen og lyddelen. Under oppstart blir et Sound-objekt opprettet og initialisert, og det er i denne prosessen at de maskinvarespesifikke lydparametrene blir satt, basert på hva som er tilgjengelig på den spesifikke maskinen.

Som nevnt, fungerer denne klassen som kontaktpunktet mot applikasjonen. Den inneholder derfor referanser til de andre lydspesifikke klassene Listener og Source. Metoder for å hente og sette informasjon fra disse klassene er derfor tilgjenglig fra denne klassen. Under oppstart og innlasting av lyden sammen med geometrien i modellen, opprettes tilsvarende Listener- og Source-objekter basert på informasjonen som er definert i XML-fila.

### 4.1.2 Klassen Listener

Dette er en oppbevaringsklasse for all informasjon vedrørende lytteren i modellen. Det vil si posisjon, retning og alle andre parametere som er relevante i denne sammenhengen.

### 4.1.3 Klassen Source

Source-klassen inneholder, som navnet tilsier, informasjon om lydkildene i modellen. Posisjon, retning, hastighet, navn etc. er relevante parametere her. Dessuten har hver kilde en eller flere Buffer-objekt knyttet til seg som inneholder den faktiske lydfilen som skal spilles av.

### 4.1.4 Klassen Buffer

Denne klassen inneholder informasjon om de spesifikke lydene i modellen, altså det laveste abstraksjonsnivået for lydfilene i applikasjonen. Klassen brukes av Source-klassen som en lagringsplass, og tar seg av den faktiske innlastingen av den spesifikke lydfilen fra disk.

### 4.1.5 Endringer i HelperClasses

I tråd med tidligere prosjekter har hjelpeklassene SoundInfo og SoundFile, som beskrevet i kapittel 3.3.2.1, blitt lagt til. Disse brukes av SceneGraphLoader når den temporære scenegrafen blir lastet på grunnlag av definisjonen i XML-fila. Disse klassene har metoder for å sette og hente ut de dataene som er relevante for den videre innlastingen av lydene inn i den ferdige scenegrafen. Disse klassene fungerer altså som en lagringsplass for beskrivelsen av hvordan lydkildene skal behandles videre av SceneGraphBuilder.

### 4.1.6 Endringer i SceneGraphLoader

SceneGraphLoader benytter seg av de nye hjelpeklassene som er definert for å lagre informasjonen som hentes inn fra XML-fila. Klassen utvider altså den temporære scenegrafstrukturen med nodetyper for SoundInfo- og SoundFile-noder.

### 4.1.7 Endringer i SceneGraphBuilder

Klassen har blitt utvidet til å kjenne igjen de nye strukturene i den temporære scenegrafen som ble bygget av SceneGraphLoader i trinnet før. Når et SoundInfo-objekt gjenkjennes under traverseringen, kalles metoden *buildSound* som henter ut informasjonen som ligger lagret i denne noden. Denne informasjonen brukes så videre under et metodekall til Sound-objektet (som er initialisert under oppstart av applikasjonen). Dette objektet laster inn og initialiserer så lyden.

### 4.1.8 Endringer i Sverresborg

Sverresborg-klassen er den overordnede klassen for hele systemet. Denne kaller de nødvendige metoder for initialisering under oppstart og oppretter de andre klassene.

### 4.1.9 Endringer i uiWin

Som nevnt i kapittel 3.3.2, er det denne klassen som inneholder renderingløkken for applikasjonen, samt at den mottar input fra brukeren under kjøring. Siden ulike lydparametere må endres fra denne klassen, er derfor denne klassen bindeleddet mellom lydsystemet og resten av programmet. Lyden blir av denne klassen initialisert ved å opprette et Sound-objekt. Dette objektet blir sendt med til Sverresborg-objektet under innlasting av modellen og lyden under oppstart. Denne prosessen laster inn alle lyder og parametere knyttet til lydkildene, og når den returnerer er det mulig for uiWin-objektet å spille av lyder til bestemte tidspunkt.

Oppdateringen av lytterens posisjon og retning skjer parallelt med oppdateringen av synsfeltet under kjøring. Som nevnt i kapittel 3.3.2 er det i denne fasen også innebygd en konvertering mellom de ulike koordinatsystemene som brukes av lyddelen og OpenGL Performer.

## *4.2 Grafiske endringer av Sverresborgmodellen*

I dette kapittelet vil arbeidet som er gjort i forbindelse med utstillingen på Sverresborg Folkemuseum bli presentert. Utstillingsmodellen er et samarbeid med medstudent Espen Almdahl, og arbeidet med denne delen av oppgaven er gjort parallelt med implementering av lyd i Sverresborgapplikasjonen.

Den eksisterende modellen er modellert i 3D Studio MAX, og blir konvertert til et format som kan leses av OpenGL Performer før den lastes inn av Sverresborgapplikasjonen. Endringer av modellen, samt modellering og animering av mennesker gjøres derfor enklest i 3D Studio MAX. Museet ønsket en guidet tur gjennom modellen, der en animert guide skulle forklare underveis. Dette gjøres best med en ferdiggenerert kamerabane, slik at ingen styring og manipulering er nødvendig under selve demonstrasjonen. Denne kamerabanen modelleres

også enklest i 3D Studio MAX, og et alternativ var derfor å lage en slik bane der, lagre den, og videre utvikle en mekanisme som kunne lese ut kamerakoordinatene fra denne og sette det tilsvarende kameraet i OpenGL Performer applikasjonen.

OpenGL Performer har liten eller ingen automatisk støtte for visuelle effekter som for eksempel ild og røyk, noe som fører til at denne løsningen også er begrenset med tanke på den visuelle kvaliteten som kan oppnås.

Animering og innlemming av menneskelige modeller i borgen var tema for en prosjektoppgave av Espen Almdahl og undertegnede høsten 2004 [FA01], og ville følgelig ikke by på noe større problem i seg selv. Modellene måtte likevel ha blitt modellert og animert i 3D Studio MAX før man lastet dem inn i modellen. Disse momentene gjorde det derfor klart at den beste løsningen for museet ville være å bruke 3D Studio MAX til å direkte generere bildene som var nødvendige for animasjonen, slik at resultatet ble en videofil som kunne spilles av på en vanlig PC.

Under følger en forholdsvis kort gjennomgang av hva som er blitt gjort på denne delen av oppgaven (hva som er modellert, animert etc.). Da det er brukt en rekke ulike teknikker og innebygd funksjonalitet i 3D Studio MAX under dette arbeidet, vil ikke hver av disse bli gjennomgått spesifikt. Det refereres til den vedlagte modellen i 3D Studios MAX-format vedlagt på medfølgende CD for nærmere detaljer rundt dette.

### 4.2.1 Endringer og tillegg til den grafiske modellen

Arbeidet med å gjøre endringer og tillegg i modellen har skjedd i nært samarbeid med arkeologistudent Regin Meyer. På grunn av tidspress, har disse endringene ikke vært detaljert på forhånd i en rigid kravspesifikasjon, men heller utviklet seg gjennom iterasjoner med ønsker, endringer og godkjenninger fra ham. Denne arbeidsmåten har gjort det mulig å endre en god del detaljer på kort tid, og gjort resultatet mer i tråd med hvordan man faktisk mener borgen så ut på den tiden.

På grunn av arbeidsmetoden nevnt over, anses det som uhensiktsmessig å beskrive absolutt alle endringer som er gjort (for eksempel flytting av et enkelt punkt i modellen av borgen for å gjøre overflatestrukturen mer korrekt). Under følger imidlertid en oversikt over de største og viktigste endringene, samt hvilke nye objekter som er lagt til modellen som ble utviklet av Kundestyrt prosjekt gruppe 13, høsten 2003 [KPO01].

Endringer av eksisterende modell:

- *Vindebro.* Vindebroen er blant annet skalert slik at den dekker hele åpningen av borgen.
- *Utspring foran vindebro*. Endret for å gjøre modellen mer historisk korrekt, samt for å passe den nye vindebroen.
- *Tregang på forside av borgtårn.* Forlenget de utstikkende tregangene på forsiden av borgen slik at det er mulig å gå langs hele borgens forside inne i disse. Dessuten lagt til støttestrukturer i tre under disse for å holde de oppe.
- *Krenelering.* Kreneleringen på borgtårnet er gjort bredere.
- *Vindu på borgens forside.* Flyttet det nederste vinduet på forsiden av borgen lenger ned slik at det ligger i plan med ringmuren, samt gjort det smalere.

- *Bakketekstur.* Enkelte visuelle endringer er gjort på bakketeksturen for å blant annet gjøre fjellsidene mer realistiske.
- *Vann i brønn.* Teksturen på vannet i brønnen er endret for å gjøre det mer realistisk.
- *Brønn.* Brønnen er endret ved at sideveggene er gjort smalere. Dette har blitt gjort for å passe målestokken til de menneskene som skal hente vann i brønnen.

Nye objekter i modellen:

- *Kjettinger for vindebroen.* Modellering av kjettinger, samt endringer av selve borgen for å lage hull til de to kjettingene. Animering av åpning og lukking av vindebroen.
- *Gitter*. Foran inngangen av borgen, samt endringer av selve borgen for å plassere gitteret.
- *Laftet bygning*. Det er satt inn en lengre, laftet bygning i den nordvestlige delen av borgen.
- *Kjøkken.* I den nordøstlige enden, langs muren, er det plassert en kjøkkenbygning.
- *Smie.* En av de eksisterende bygningene langs den vestlige delen av borgen er omgjort til en smie. Et ildsted med tilhørende flammer, kull og ved er plassert inne i bygningen, samt et tilsvarende ildsted på utsiden med kun glødende kull. Det har vært nødvendig å åpne døren inn til smien for å kunne se hva som skjer der inne. I den forbindelse viste det seg at disse eksisterende husene ikke var hule, og det har derfor vært nødvendig å gjøre om innsiden slik at vegger og gulv har blitt synlige. Det har dessuten blitt laget en åpning i taket slik at røyken kan slippe ut.
- *Flammer fra ildsted.* Flammen som kan skimtes inne i smia, er laget ved hjelp av en såkalt atmosfæreeffekt i 3D Studio MAX. Da denne delen av animasjonen er forholdsvis lite synlig, var det greit å bruke den innebygde funksjonaliteten for dette (flammene ser forholdsvis bra ut, men kan i lengden bli noe ensformige). De genererte flammene er ikke et fysisk objekt i modellen, men legges på i etterkant av programmet under rendring (effekten gjennomgår likevel synlighetstester etc. som vanlige objekter før det rendres). Den bakenforliggende teknikken bak denne funksjonaliteten har ikke vært mulig å finne dokumentert, men er sannsynligvis basert på en klassisk teknikk med bruk av en flammetekstur med fargene i en flamme, samt en *alphatekstur* for å gjøre flammene gjennomskinnelige. I tillegg brukes en *noisegenerator* for å gjøre det hele mer kaotisk og tilfeldig (samt under animeringen av flammene).
- *Røyksøyler*. To røyksøyler fra smia i den vestlige delen av borgen og kjøkkenet i den nordøstlige delen. Disse er laget ved hjelp av partikkelgeneratorer. Hver partikkel får en tilfeldig størrelse (mellom to grenseverdier) og en tekstur. Denne teksturen er satt sammen av flere ulike teknikker for å gjøre den mest mulig realistisk. For det første er grunnfargen basert på livsløpet til partikkelen. Ved fødsel er den hvit og blir gradvis mørkere (som i vanlig røyk). Dette er en spesiell tekstur som er innebygd i 3D Studio MAX, og heter *Particle Age*. Siden det opereres med diskrete partikler, har hver partikkel også en *alphatekstur* som bestemmer gjennomskinneligheten. Alphateksturen som ble brukt i dette tilfellet er hvit innerst (det vil si ingen gjennomskinnelighet og fargen i teksturen vises i sin helhet) og svart ytterst. Denne overgangen er i sin tur basert på en såkalt *noisegenerator* som gjør overgangen mindre regulær og dermed mer realistisk. Ved å sette hastigheten og livslengden til partiklene, er det på denne måten mulig å skape realistisk røyk. I tillegg til selve partikkelgeneratoren, er det også laget to fysiske vindgeneratorer plassert på hver side av røyksøylene med ulik styrke. Disse påvirker partiklene i hver sin retning og skaper et realistisk inntrykk av turbulens inne i røyksøylene.
- *Toetasjes forlegningsbygning*. Laget og plassert langs muren i den sørøstlige delen av borgen.

- *Bord og stoler*. Plassert i vaktrommet for soldatene som sitter der, samt en stol for smeden.
- *Hnefatafl.* Det gamle vikingespillet *hnefatafl* er modellert og plassert utenfor nabobygningen til smia, langs den vestlige delen av borgen. Her inngår også to stoler for spillerne, samt et bord for spillet.
- *Våpen.* Sverd, øks, spyd, skjold og hjelmer er modellert og plassert rundt omkring i borgen, samt at alle soldater har sverd, skjold og hjelm.
- *Ildtang og hammer for smed.*
- *Flamme fra stikkerter.* Lyskilder på innsiden av borgen (rett etter inngangen fra vindebroen) og inne i vaktrommet. Disse er modellert som et animert flammeobjekt med tre tilhørende punktlyskilder for å gi inntrykk av lyset fra flammen. Da kameraet fokuserer forholdsvis lenge på disse objektene, var det viktig at de ble modellert så realistisk som mulig. Selve flammen er som nevnt et eget objekt, og animeringen av disse er laget ved hjelp av en såkalt *noisegenerator* som endrer parametrene for objektets høyde og deformasjon i planet. Teksturen er satt sammen av fargene til en vanlig flamme, samt at en alphatekstur er brukt for å gjøre flammen gjennomskinnelig.
- *Trapp opp til borgmur.* En tretrapp fra borgplatået opp til toppen av muren er lagt til i den sørvestlige delen av borgen.
- *Gress.* Langs alle bygninger og på innsiden av borgmuren, er det på den nedre delen plassert bokser med en gresstekstur. Denne teksturen benytter seg av en *alpha map* for å gjøre boksen gjennomsiktig der det ikke er gress. Dette skaper et bedre visuelt bilde, da den brå overgangen mellom bygninger og den forholdsvis flate og udetaljerte bakken på borgplatået er forholdsvis kunstig.
- *Utsikt mot Munkholmen.* Fra kongeboligen i nord er utsikten fra det østlige vinduet modellert som en plate like utenfor bygningen med bilde av Munkholmen som tekstur. Dette skaper inntrykk av utsikten fra dette vinduet uten å modellere selv holmen noen kilometer unna. Bildet er tatt av dagens utsikt, og redigert i ettertid for å fjerne bygninger, båter etc. som har kommet til i ettertid.
- *Lyssetting.* Fire punktlyskilder uten skygge er lagt til for å skape et bedre inntrykk av dagslys. Disse kommer i tillegg til sola, som er modellert som en retningslyskilde med skygger (ved bruk av *ray-tracing*). Punktlyskildene i modellen genererer ikke skygge for å spare på renderingstiden.
- *Synlig sol.* Ved å bruke en glødeeffekt, skapes et bilde av en diffus sol bak skylaget der den opprinnelige sollyskilden er plassert.
- *Himmel.* En halvkule med en innvendig himmeltekstur omgir borgen (et himmelfotografi tatt med fiskeøyelinse, for å skape korrekt inntrykk av himmelen ved mapping på en kule).

Personer i modellen:

- *Guide*. Modellert som en soldat og animert med gange og arm- og hodebevegelser. Denne personen følger kameraet rundt omkring i borgen, og stopper på enkelte steder for å forklare nærmere spesielle hendelser.
- *Torstein Kugad med kone og to barn.* Bestyreren av borgen i kongens fravær, er plassert på den nordlige delen av borgplatået, utenfor kongeboligen. Han og barna er animert med arm- og hodebevegelser, samt bevegelse av overkroppen. Kona Borghild er dessuten animert med gange.
- *Soldater på muren*. To soldater er modellert og animert med blant annet gange på den vestlige muren.

- *Soldat på borgtårn*. En soldat står oppe på borgtårnet å ser ned mot vindebroen (beordrer senking og heving av denne).
- *Soldater inne i borggården*. To soldater står inne på borgplatået å prater (gestikulerer med hånd- og hodebevegelser).
- *Soldater som spiller hnefatafl*. To soldater sitter på den vestlige siden av borgen og spiller det gamle spillet hnefatafl. Begge er animerte.
- *Soldater inne i vaktrom*. To soldater og en sivilist sitter inne i vaktrommet (til venstre for og på innsiden av vindebroen) og prater. Disser er animerte, med blant annet arm- og hodebevegelser.
- *Sivilister på svalganger*. To sivilister er plassert på svalgangene på trebygningen ved siden av kongeboligen, samt på innsiden av borgtårnet. Disse er ikke animerte, men står lent mot rekkverket og ser ned mot platået.
- *Sivilister på borgplatået.* To sivilister står midt på borgplatået og prater (som for de to soldatene nevnt over).
- *Smeder.* En smed sitter utenfor smia og hamrer på en spydspiss, samt at en annen smed sitter ved inngangen til smia og spisser spydspisser (begge er animerte).
- *Sivilist og dame utenfor forlegningshus.* En sivilkledd mann og dame står utenfor forlegningshuset på den sørøstlige delen av borgplatået og diskuterer (armbevegelser etc.).
- *Dame som bærer vann.* En dame står ved brønnen og bruker en bøtte til å hente vann. Etter dette er gjort, går hun bort fra brønnen mot kjøkkenet.
- *Hund.* En hund er plassert like innenfor inngangen til borgplatået. Denne er animert slik at den bjeffer på den forbipasserende guiden.

Når det gjelder de menneskelige modellene, er disse laget med så lav oppløsning som mulig for å spare prosesseringskraft. Det er derfor forsøkt å lage teksturene på disse mest mulig detaljerte for å skjule dette. En teknikk som blir benyttet her er såkalt *texture unwrapping*. Det vil si at man ber programmet generere et enkelt (todimensjonalt) bilde som inneholder alle polygonene til modellen lagt ut på denne overflaten. Ved å bruke et tegneprogram kan man da tegne inn teksturen direkte i det ønskede polygonet, før man til slutt tilegner denne teksturen til modellen. Dette gjør at en får bedre kontroll over hvor på modellen eventuelle endringer i teksturen vil ha effekt.

### 4.2.2 Animering av modellen

Det er i hovedsak personene inne i borgen som er animerte. For å skape en mer livfull opplevelse for brukeren, er hver person animert i større eller mindre grad. Under dette arbeidet er *keyframe-teknikken* brukt. Man flytter for eksempel hånden til den posisjonen man vil ha den i det nåværende bildet, og angir at dette skal være et keyframe. Så hopper man til et bilde lenger frem, og flytter hånden i den posisjonen man ønsker at den skal ha i dét bildet og setter et nytt keyframe der. Programmet vil da automatisk regne ut håndens posisjon i de mellomliggende bildene basert på disse to endeposisjonene (for mer informasjon om dette, se [FA01]). Dette er en forholdsvis tidkrevende prosess, og siden menneskelige bevegelser er forholdsvis vanskelig å gjenskape på en korrekt måte, er antallet personer i borgen mindre enn det egentlig skulle ha vært på den tiden.

Personene i modellen er bygd rundt et skjellett (en såkalt *biped*), noe som gjør at for eksempel bevegelse av skulderen også fører til bevegelse av overarm, underarm og håndflaten (se mer

om denne teknikken i [FA01]). Dette letter animeringen av personene. Dessuten har 3D Studio MAX innebygd funksjonalitet for å gjenskape gange. Dette gjøres ved å plassere hvor fotbladene skal treffe bakken, og programmet vil så automatisk beregne hvilke bevegelser som er nødvendige for å oppnå dette. Dette er en meget tidsbesparende funksjon, men krever en del prosesseringsressurser.

Også kameraet er animert på denne måten (flytte det til ønsket posisjon, sette et keyframe etc.). Da Regin Meyer hadde laget et utkast til et manuskript for guidens bevegelser inne i borgen (se Vedlegg B), ble det laget et såkalt *storyboard* for kameraets (det vil si brukerens) posisjon og retning til enhver tid under animasjonen. Her inngår også hvilke lyder som på det gitte tidspunktet skal høres. Vedlegg A inneholder den tekstlige beskrivelsen av dette.

Enkelte effekter, slik som flammene og røyken inne i smia og røyken fra kjøkkentaket, er også animerte. For røyken sin del, er denne en partikkelgenerator der hver partikkel har en halvgjennomsiktig tekstur. Flammene er en annen innebygd effekt, der animeringen består av å lage keyframes med ulike parametere. Under animasjonen vil disse parametrene endre seg, og dermed skape inntrykk av at flammene er i bevegelse.

Til sammen ga dette en animasjon med 9500 frames i 3D Studio MAX (noe som dobles på grunn av man trenger to kamera for stereoeffekten), som tilsvarer omtrent 6 minutter og 36 sekunder med en hastighet på 24 bilder per sekund.

### 4.2.3 Visning av Sverresborgmodellen

For å skape inntrykk av dybde, brukes stereografisk projeksjon under fremvisningen av videosekvensen. I denne prosessen brukes to projektorer med hvert sitt videosignal. Disse to videosignalene er generert med hvert sitt kamera i 3D Studio MAX. Kameraene er plassert parallelt med hverandre, med en avstand som tilsvarer avstanden mellom øynene til en person i modellen. Dette gjør det mulig å simulere, på et todimensjonalt lerret, hva som fysisk skjer når vi oppfatter gjenstander i den fysiske verden med dybde (det finnes dessuten en rekke psykologiske faktorer som spiller en rolle når vi avgjør en gjenstands avstand fra oss, men disse vil ikke bli gjennomgått her). Disse bildene blir dermed forskjøvet noe i forhold til hverandre, og ved å bruke polariserte briller (hvert brilleglass har enten horisontal eller vertikal polarisasjon) oppfattes kun det ene bildet som projiseres på lerretet i hvert øye. Dette krever selvsagt at bildet som hver projektor viser på lerretet er polarisert (ved bruk av et enkelt filter foran hver projektor), med den tilsvarende horisontale eller vertikale retningen.

Praktisk lages de to kameraene som er nødvendig i 3D Studio MAX ved hjelp av en *plug-in* for dette formålet kalt *StereoGrapher MAX*. Denne er fritt tilgjengelig fra blant annet [WEB12]. Det er her mulig å manuelt spesifisere øyeavstanden til det gitte formålet. En større øyeavstand vil produsere en større dybdeeffekt, men kan over lengre tid virke slitsomt og urealistisk for brukeren. Dette programmet oppretter to kameraer i tillegg til det opprinnelige, plassert parallelt med dette på hver sin side med en avstand som tilsvarer den oppgitte øyeavstanden. Da den opprinnelige modellen som var utviklet av tidligere prosjekter ikke brukte fysisk korrekte avstander i modellen (måleenheten var tommer, og en person inne i borgen var for eksempel bare omtrent 10 tommer), var det ikke mulig å direkte bruke den vanlige standarden 6,5 cm som øyeavstand. En del prøving å feiling var derfor nødvendig for å finne den øyeavstanden som balanserer et godt dybdeinntrykk for objekter langt unna

brukeren, mot at objekter som er nærme ikke oppfattes som doble (på grunn av for stor øyeavstand).

Introduksjonstekster og rulletekster ble lagt til ved hjelp av programmet *Adobe Premiere Pro versjon 1.5*. Det var også i denne applikasjonen at lyden til slutt ble lagt til. Lydeffekter som for eksempel fotsteg, rasling av vindebrokjettinger, hundebjeff etc. ble hentet fra internett gjennom den gratis tilgjengelige tjenesten [WEB13]. Disse ble tilpasset animasjonen som var laget på forhånd, slik at timing av grafikk og lyd ble så god som mulig. Dialogen som snakkes av guiden og diverse andre personer inne på borgplatået (med utgangspunkt i manuskriptet som ble skrevet av Regin Meyer), ble innlest av personer fra Trøndelag Teater og resulterte i lydfiler på wav-formatet som så ble lastet inn sammen med grafikken.

Etter at timingen av lyden og grafikken var tilfredsstillende, ble lydene stedfestet ved hjelp av Premieres innebygde støtte for *5.1 surround sound-standarden*. Denne standarden benytter én fronthøyttaler, to sidehøyttalere, to høyttalere på hver side bak lytteren, samt en såkalt *subwoofer* for de dypeste tonene i lydsporet (denne kan i utgangspunktet plasseres vilkårlig på grunn av ørets manglende evne til å retningsbestemme lave frekvenser på en god måte, som nevnt tidligere). Hver av disse kanalene får ved bruk av dette formatet sitt eget lydspor som lagres sammen med selve grafikken, og kan spilles av på alle lydkort som har innebygd støtte for denne standarden (avspillerprogramvaren vil selv automatisk *downmixe* disse sporene til vanlig, to-kanals, stereolyd hvis slik maskinvare ikke er tilgjengelig). Denne stedfestingen er forholdsvis enkel å utføre ved hjelp av det grafiske brukergrensesnittet til Premiere, og de praktiske detaljene rundt dette vil ikke bli videre detaljert her.

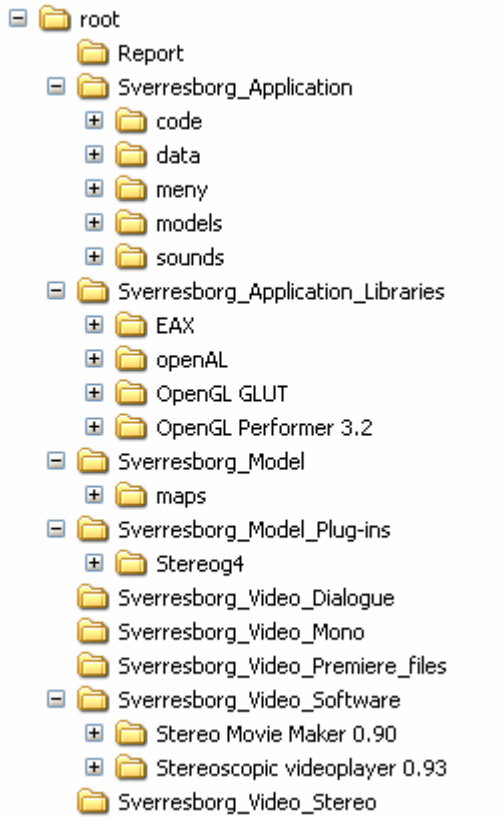
De ovenfor nevnte prosessene resulterte tilslutt i to videofilmer med lyd (en for hvert øye) i oppløsningen 800x600. Disse to signalene ble til slutt satt sammen ved hjelp av programmet *StereoMovieMaker* (som også er fritt tilgjengelig fra blant annet [WEB14]). Avhengig av maskinvareoppsettet, velges et format for denne samlede stereovideoen (eksempelvis side-ved-side bilde, over-under bilde etc.). For dette formålet ble side-ved-side-formatet valgt, der de to signalene blir lagret ved siden av hverandre i den ferdige videoen (noe som resulterer i en videofil med en oppløsning på 1600x600). Grunnen til dette var at grafikkortet på avspillermaskinen hadde to videoutganger, og det var derfor mulig å sende hvert bilde i denne videofilmen til hver sin utgang. Disse utgangene var i sin tur koplet til hver sin projektor. Hvis grafikkortet hadde hatt kun én videoutgang, samt en såkalt 3-pins DIN synkroniseringstilkopling, hadde det vært nødvendig å splitte opp de to videobildene i ekstern maskinvare før de når projektorene. Dette er en løsning som finnes i dag på visualiseringslaboratoriet ved IDI, men vil i de fleste tilfeller bli en dyrere løsning da maskinvaren for å splitte videosignalet tilbake til to separate kilder er forholdsvis kostbar.

## 4.3 Installasjon og oppsett av applikasjon og video

På vedlagt CD finnes applikasjonen som et kompilert program, samt at all kildekode (inkludert dokumentasjon) også kan finnes igjen der. Da programmet benytter seg av flere ulike tilleggsbibliotek, er det nødvendig å installere disse før en kjører programmet. Mer spesifikt gjelder dette OpenAL- og EAX-kjøretidsbibliotek, samt OpenGL Performer versjon 3.2. Alt dette finnes vedlagt på CD under katalogen *root\Sverresborg_Appliaction_Libraries*, se Figur 4-1. OpenGL Performer er lisensiert, men det er mulig å kjøre en demoversjonen av programmet (som er vedlagt).

Under utviklingen av applikasjonen, ble programmeringsmiljøet Microsoft Visual C++ versjon 6 brukt. Dette lager en del ekstra filer knyttet til kompileringen og hvilke programmeringsbibliotek som blir benyttet. Disse ligger derfor i samme katalog som selve kildekoden (*root\Sverresborg_Application\code*).

For å kjøre programmet startes *svb_kpro.exe*. I samme katalog ligger også XML-filen som definerer hvilke modeller og lyder som skal lastes inn. Det er derfor mulig å endre objektene og lydene i scenen uten å måtte rekompilere hele programmet på nytt.




**Figur 4-1 Filstruktur for vedlagt CD**

Videoen som ble laget i forbindelse med utstillingen på Sverresborg Folkemuseum, er også vedlagt, og befinner seg under katalogen *root\Sverresborg_Video_Stereo*. For å spille av denne, er det nødvendig med en spesiell videoavspiller som heter *Stereoscopic Videoplayer versjon 0.93*. Denne er gratis og kan finnes på vedlagt CD (innstillingene i dette programmet vil avhenge av hvilken maskinvare som blir brukt for å vise filmen i stereo). Dessuten er en monoversjon av filmen også vedlagt som er mulig å spille av på en vanlig skjerm.

Hvis det i fremtiden skal endres på denne filmen, er prosjektfilen som ble laget i Adobe Premiere Pro versjon 1.5 også lagt ved (i katalogen *root\Sverresborg_Video_Premiere_files*). Denne inneholder timingen av alle lydeffekter og dialog, samt teksten som vises i introduksjons- og rulleteksten. Dette filformatet lagrer ikke selve grafikken og lyden i den lagrede filen, men inneholder kun lenker til de angitte kildene. Det vil derfor være nødvendig å rendre videoene på nytt hvis disse skal brukes videre. Dialogen, som ble innlest av ansatte ved Trøndelag Folkemuseum, kan forøvrig finnes igjen under katalogen *root\Sverresborg_Video_Dialogue*.

Modellene som ble brukt som utgangspunkt for renderingen av filmen, kan finnes under katalogen *roo\Sverresborg_Model*. Denne er lagret i 3D Studio MAX versjon 5 format. Alle bildefiler som er utgangspunkt for teksturene i modellen ligger også under denne katalogen.

# 5 Resultat og diskusjon

I dette kapittelet vil det gjennomgås hva som ble oppnådd av resultater etter implementeringen av de foreslåtte endringene fra kapittel 3. Dessuten vil det bli gitt en kort oppsummering av demonstrasjonsversjonen av modellen som ble laget i forbindelse med utstillingen.

## *5.1 Resultater fra endringer av eksisterende applikasjon*

Resultatet av endringene av den eksisterende Sverresborgapplikasjonen, er at det nå er mulig å inkludere lyder og musikk sammen med grafikken. På grunn av tidsmangel, er det i hovedsak selve muligheten til å innlemme lyd som er blitt implementert. Det er lagt til enkelte demonstrasjonslyder for å vise hvordan systemet virker i selve applikasjonen som er vedlagt, men for å skape et fullgodt lydlandskap, som diskutert i kapittel 3.2, er det sannsynligvis også nødvendig å ta opp egne lyder ettersom det er vanskelig å finne de lydene man eksakt har bruk for fritt tilgjengelig på internett.

De lydene som nå kan høres i modellen er blant annet bakgrunnslyder i form av fuglekvitring, vind og en generell bakgrunnsatmosfære (slik at det aldri blir helt stille under simuleringen). Dessuten er det lagt til lyden av rennende vann, for å demonstrere stedfestingsmekanismen i applikasjonen. Denne lyden er tilfeldig valgt (siden det ikke finnes andre vannkilder enn brønnen i modellen), men er tatt med for å illustrere hvordan det er mulig å stedfeste lyder i modellen. Lyden kan høres under oppstart av programmet, og lydkilden er initielt plassert i samme punkt som brukeren befinner seg på under oppstart. Denne funksjonen krever at maskinvaren består av flere enn to høyttalere. Et 5.1-oppsett, som nevnt tidligere, gir gode resultater i dette tilfellet, men på grunn av at OpenAL selv vil konfigureres til å bruke den maskinvaren som er tilgjengelig under initialiseringen av programmet, er andre konfigurasjoner også mulig uten å selv måtte endre noen av disse parametrene.

For å gjøre applikasjonen så fleksibel som mulig, har bruken av en definisjonsfil (i form av en XML-fil) blitt videreført fra tidligere prosjekter. Dette gjør at det er mulig å definere hvilke lyder som skal spilles av, samt hvilke objekter disse lydene stammer fra, i denne tekstfilen. Dette gjør at endringer av den grafiske modellen eller lydene kan foregå uten at man må rekompilere hele applikasjonen på nytt.

Arbeidet som er blitt utført parallelt med denne oppgaven med å overføre systemet fra IRIX- til Windows-plattformen, har ført til at input fra brukeren under kjøring av systemet (for å manøvrere seg omkring i modellen) er blitt endret i forhold til tidligere versjoner. Denne konverteringen har ikke vært helt optimal med tanke på selve styringen rundt omkring i scenen, da selve modellen blir flyttet under oppdatering av synsfeltet. Dette har derfor også ført til at lyddelen av applikasjonen ikke vil fungere optimalt, da grafikken ikke vil stemme med lyden ved alle tilfeller. Dette er likevel en feil med selve inputmekanismen som er laget, og den stedfestede lyden vil derfor fungere korrekt hvis dette blir endret i fremtiden.

## 5.2 Resultater fra demonstasjonsversjonen av Sverresborgmodellen

Arbeidet med utstillingsversjonen resulterte, som nevnt tidligere, i en ti minutter lang video (inkludert introduksjon og rulletekst) der modellen blir presentert under en guidet tur rundt omkring i borgen. Det meste av arbeidet rundt dette ble dokumentert i kapittel 4.2, det refereres derfor til vedlagt video for å få et bedre inntrykk av hva som har blitt gjort.

Under følger noen bilder fra den resulterende videoen (uten bruk av stereo).

**Figur 5-1 Oversiktsbilde fra video**

**Figur 5-2 Bilde fra vaktrommet til soldatene**

**Figur 5-3 Bilde av menneskelige modeller i borgen**

**Figur 5-4 Foran hnefatafl-spillende soldater og smed i bakgrunnen (guide i forgrunnen)**

## 5.3 Oppsummering

Målet med å innlemme stedfestet lyd som en integrert del av Sverresborgmodellen ble oppfylt, og det viste seg å være flere aktuelle løsninger å velge mellom for å oppnå dette målet. Valget av OpenAL som lydbibliotek foran andre potensielle muligheter, ser i ettertid ut til å ha vært bra. Biblioteket er, som nevnt tidligere, godt egnet til å brukes på ulike plattformer, og dette er viktig i denne sammenheng da det nå finnes både en IRIX- og en Windows-versjon av programmet. Lyden er riktignok ikke implementert i IRIX-versjonen i skrivende stund, men å overføre lydsystemet til denne plattformen vil være forholdsvis rett frem (det vil her være snakk om enkelte endringer i selve initialiseringen av systemet).

Under testingen av applikasjonen, viste det seg at de enkle tilleggene av lyd skapte en langt større innlevelse enn den tidligere versjonen av programmet. Ved å bruke lyd som en aktiv del av opplevelsen, trekkes dessuten noe av fokuset bort fra enkelte begrensninger i grafikken og, som nevnt i kapittel 2, den subjektive oppfatningen av grafikken kan faktisk øke ved bruk av lyd.

Demonstrasjonsmodellen er i dag i bruk ved Sverresborg Folkemuseum, der den vises ved hjelp av to prosjektører og en forholdsvis standard datamaskin utstyrt med et grafikkort som har to skjermutganger.

# 6 Konklusjon

I denne hovedoppgaven har det blitt gjennomgått en del teori rundt lyd, og stedfestet lyd spesielt. Dette har ledet frem til en implementasjon i den eksisterende grafiske modellen av Sverreborgen, der det nå er mulig å innlemme stedfestet lyd som en del av den historiske gjenskapningen av borgen og miljøet rundt. Godt utviklede lydbiblioteker har gjort det mulig å innlemme flere aspekter av lyd (som for eksempel bruk av biblioteket EAX for gjengivelse av romklang) innenfor kortere tidsrammer enn det hadde vært mulig å oppnå hvis dette hadde blitt gjort manuelt. Dette fører igjen til at fokuset kan rettes mot selve innholdet av det som skal presenteres, og bort fra de rent tekniske detaljene knyttet til konstruksjon og implementasjon av selve lydsystemet.

Da det som nevnt har vært et parallelt arbeid med å konvertere det eksisterende systemet over til Windows-plattformen, har det vært enkelte problemer med synkronisere dette arbeidet med innlemmelsen av lydsystemet. Som et resultat av dette, er lydsystemet blitt gjort så uavhengig og enhetlig som mulig, for å gjøre forbindelsen til resten av applikasjonen mest mulig ”rent”.

Arbeidet med utstillingsversjonen av borgen, som resulterte i en videosekvens av en guidet tur omkring på borgplatået, har i stor grad vært et tverrfaglig samarbeid. Arkeologi, representert ved hovedfagsstudent Regin Meyer, har i denne sammenheng vært et fagfelt som har ført til at de forbedringer som er gjort i den grafiske modellen, samt i utviklingen av et lydlandskap for borgen, i større grad har blitt historisk korrekt. Dette arbeidet har likevel krevd en del tid, og anslagsvis seks uker har gått med til denne delen av oppgaven.

Utstillingsmodellen, som i all hovedsak ble laget ved hjelp av 3D Studio MAX, viser potensialet som kan etterstrebes under videreutviklingen av den grafiske applikasjonen. Mange teknikker og effekter som er brukt under dette arbeidet kan med enkle grep overføres til OpenGL Performers miljø, mens andre vil kreve noe mer utvikling med tanke på dagens system (for eksempel innlemmelse av et partikkelsystem for å blant annet kunne simulere vann og røyk etc.).

Utstillingsmodellen som ble laget viser i større grad hva som er mulig å oppnå med stedfestet lyd i en grafikkapplikasjon. Både bakgrunnslyder (blant annet fuglekvitring og vind som nevnt over) og spesifikke lydkilder, gjør opplevelsen av å være tilstede i borgen på det aktuelle tidspunktet mer realistisk. Stedfestet lyd fra blant annet arbeidende smeder, lyden av vindebroen som åpnes og lukkes, latter og stemmer fra andre personer til stede inne i borgen gjør at lydlandskapet blir rikere, og det skapes en ekstra dimensjon i tillegg til de tre dimensjonene grafikken opptar.

Resultatet som ble oppnådd ved å bruke en ferdiggenerert videofilm, har sannsynligvis blitt bedre enn ved bruk av et sanntidssystem. Dette gjelder spesielt med de nevnte begrensningene som ligger i dagens sanntidssystem. Fremgangsmåten som er skissert i kapittel 4.2 for å lage denne filmen, kan benyttes også for eventuelle fremtidige prosjekter da den ga et godt sluttresultat. Mer utvikling på sanntidssystemet kan i fremtiden gjøre det mulig å bruke dette også i demonstrasjoner av denne typen.

## 6.1 Gjenstående arbeid og videre utvidelser

Sverresborgmodellen slik den ser ut i dag er, som nevnt tidligere, resultatet av en rekke større eller mindre forbedringer gjennom ulike studentprosjekter og oppgaver. Ny funksjonalitet har blitt lagt til og eksisterende modell har blitt forfinet. Ny funksjonalitet skaper i sin tur ofte potensiale for nye forbedringer, ny funksjonalitet og så videre. Denne oppgaven er intet unntak i så måte. Under følger derfor en oversikt over de mest nærliggende utvidelsene som har dukket opp under arbeidet med denne oppgaven.

- På lydsiden er det i nåværende system laget mulighet for stedfestet lyd. Det vil likevel være behov for en mer omfattende tilstandsmaskin for å styre avspillingen til hvert grafiske objekt. Som nevnt tidligere, vil det sannsynligvis være mest hensiktsmessig å lage en egen type tilstandsmaskin for hver kategori av objekter (for eksempel en for menneskelige modeller, en for verktøy etc.), siden disse vil ha noenlunde lik oppførsel. I denne sammenheng kan det være aktuelt å også inkludere muligheten for animerte modeller, som ble tatt ut under konverteringen til Windows-plattformen. Animerte modeller var hovedtema for prosjektoppgaven til undertegnede og Espen Almdahl høsten 2004 [FA01], og kan være et godt utgangspunkt i så måte.

- For den grafiske applikasjonen er det flere utvidelser som kan gjøre opplevelsen av modellen mer spennende. Visuelle effekter (eksempelvis røyk, ild, vann etc.) er som nevnt ikke direkte støttet av OpenGL Performer. Det har derfor kunnet vært interessant å se på hvordan slike forbedringer har kunnet brukes i programmet. På dette punktet kan videoen som ble laget i 3D Studio MAX muligens kunne være et utgangspunkt for hva som eventuelt må gjøres. Dette programmet har innebygd støtte for en rekke slike effekter, men det er vanskelig å overføre disse til scenegrafen i OpenGL Performer, siden det ikke finnes nodetyper for disse direkte. I utgangspunktet er det kun geometri som kan overføres ved hjelp av den innebygde innlastingsmekanismen. Et alternativ er derfor å enten lage et program som kan konvertere effektene fra 3D Studio MAX sitt format, eller å programmere disse effektene direkte i OpenGL Performer.

- Applikasjonen har tidligere vært basert på IRIX-plattformen, og det er fortsatt aktuelt å videreutvikle programmet også på denne plattformen. For lydsystemets sin del vil en slik konvertering ikke by på større problemer, da OpenAL er bygd for å være kompatibelt med begge plattformer. De grafiske modellene som er laget i 3D Studio MAX er også kompatible mellom de to plattformene, da innlastingsrutinene er like for de to versjonene.

- Når det gjelder den geometriske modellen av Sverresborg, vil det selvfølgelig alltid være rom for nye forbedringer av denne. Flere mennesker i borgen, flere bygninger, innredning av rommene i borgen med ulike gjenstander etc. er bare noen av de mest nærliggende tilleggene. Landskapet rundt borgen har også et stort forbedringspotensiale. Dagens modell begrenser seg til å modellere de aller nærmeste omgivelsene, og det er derfor ikke mulig å skape noe inntrykk av det omkringliggende landskapet. Dagens borgplatå er modellert ved bruk av digitale terrengkart, og det burde derfor ikke være vanskelig å utvide disse videre.

  Til nå har modelleringen av borgen foregått i samarbeid med personer tilknyttet

museet på Sverresborg, og dette vil anbefales også for videre arbeid på modellen for å oppnå en mest mulig historisk korrekthet.

- Videoen som ble laget i 3D Studio MAX kan også utvides og forbedres. Spesielt gjelder kanskje dette animeringen av de menneskelige modellene. I dagens modell, er menneskene animert kun når kameraet fokuserer på dem. For å gjøre modellen mer generell, er det derfor nødvendig å enten kopiere denne animeringen gjennom resten av filmen, eller skape nye animasjoner som dekker hele tidsrommet.

  På grunn av arbeidsmengden med resten av modellen, ble ansiktsbevegelser ikke modellert i denne omgang. Å synkronisere dialog med munnbevegelser, vil derfor kunne være et interessant område å utdype videre. Dette vil skape et enda bedre inntrykk av tilstedeværelse i modellen, og fjerne mulig forvirring om hvem som til enhver tid snakker.

- Et siste punkt, som sannsynligvis vil være det mest arbeidskrevende, er å kombinere de overfor nevnte momentene i et felles sanntidssystem. Dette var et alternativ også for denne oppgaven, før det ble besluttet å lage hele videoen i 3D Studio MAX, og å utvide selve applikasjonen med stedfestet lyd. Dette ble gjort for at resultatet som skulle vises frem skulle bli best mulig innenfor den gitte tidsrammen, men en slik løsning vil kunne være interessant for fremtidige oppgaver. I dette inngår å implementere alt som ble modellert og animert i 3D Studio MAX i et sanntidssystem (muligens basert på OpenGL Performer, men andre grafikkmotorer vil også kunne være interessante).

  En slik løsning vil også åpne for muligheten av å la brukeren selv manøvrere seg rundt i modellen, muligens akkompagnert av en guide. På grunn av at maskinvare og regnekraft stadig er i rivende utvikling, kan det i nær fremtid bli mulig å ha et slikt sanntidssystem kjørende på en vanlige PC (inkludert en utvidet borgmodell, nye objekter, samt andre visuelle effekter).

# 7 Referanser

## LITTERATUR

**[BB01] Bauer, B.B.**
*(1961), Phasor Analysis of Some Stereophonic Phenomena*
Journal of the Acoustical Society of America, 33, pp. 1536-1539, 1961

**[BD01] Begault, D.**
*(1994), 3-D Sound for Virtual Reality and Multimedia*
Academic Press, Cambridge, Massachusetts, USA
ISBN 0-12-084735-3

**[BJ01] Blauert, J.**
*(1996), Spatial Hearing: The Psychophysics of Human Sound Localization, Revised Edition*
MIT Press: Cambridge, Massachusetts, USA
ISBN 0-262-02413-6

**[BD01] Burgess, D.A.**
*(1992), Techniques for Low Cost Spatial Audio*
Proceedings of the 5th annual ACM symposium on User interface software and technology, pp. 53-59, Monteray, California, USA, 1992

**[CC01] Cheng, C.I.**
*(2001), Visualization, Measurement, and Interpolation of Head-Related Transfer Functions (HRTF's) With Applications in Electro-Acoustic Music*
Dissertion in fulfillment for the degree of Doctor of Philosophy at the University of Michigan, 2001

**[DKP01] Doel, K. v., Kry, P. G., Dinesh, K. P.**
*(2001), FoleyAutomatic: Physically-based Sound Effects for Interactive Simulation and Animation*
Computer Graphics (ACM SIGGRAPH 2001 Conference Proceedings), august 2001

**[FA01] Fallan, B.T., Almdahl, E.**
*(2004), Integrasjon av animerte, menneskelige figurer i Sverresborgmodellen*
Prosjektoppgave ved Institutt for datateknikk og informasjonsvitenskap, NTNU, høst 2004

**[HR01] Hartmann, W.M., Rakerd, B.**
*(1989), Localization of Sound in Rooms IV: The Franssen Effect*
Journal of the Acoustical Society of America, 86(4), pp. 1366-1373, 1989

**[FT01] Funkhouser, T., Tsingos, N., Carlbom, I., Elko, G., Sondhi, M., West, J.**
*(2002), Modeling Sound Reflection and Diffraction in Architectural Environments with Beam Tracing*
Forum Acusticum, Sevilla, Spania, september 2002

**[ISO01] ISO/IEC 14496**
*(1999), International Standard (IS) 14496:1999. Information Technology – Coding of Audiovisual Objects (MPEG-4). 1999*

**[JW01] Jot, J. M., Warusfel, O.**
*(1995), A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications*
Proceedings of the International Music Conference, pp. 294-295, Canada, september 1995

**[KDS01] Kleiner, M., Dalenbäck, B. I., Svensson, P.**
*(1993), Auralization – an overview*
Journal of the Audio Engineering Society, 41(11), pp. 861-875, november 1993

**[KJ01] Kajiya, J. T.**
*(1986), The Rendering Equation*
Computer Graphics (Proc. ACM SIGGRAPH 1986), volume 20, pp. 143-150, april 1986

**[KKH01] Kuttruff, K. H.**
*(1993), Auralization of Impulse Responses Modeled on the Basis of Ray-tracing Results*
Journal of the Audio Engineering Society, 41(11), pp. 876-880, 1993

**[KP01] Kundestyrt Prosjekt Gruppe 13: Skaarud Karlsen, Jørn m.fl.**
*(2003), Virtuell Rekonstruksjon av Sverresborg*
Kundestyrt prosjekt ved Institutt for datateknikk og informasjonsvitenskap, NTNU, høst 2003

**[LC01] Litovsky, R.Y., Colburn, H.S., Yost, W.A., Guzman, S.J.**
*(1999), The Precedence Effect*
Journal of the Acoustical Society of America, 106, pp. 1633-1654, oktober 1999

**[MG01] Moore, G. R.**
*(1984), An Approach to the Analysis of Sound in Auditoria*
PhD thesis, Cambridge, UK, 1984

**[NSG01] Naef, M., Staadt, O., Gross, M.**
*(2002), Spatialized Audio Rendering for Immersive Virtual Environments*
Proc. Virtual Reality Software and Technology 2002, pp. 65-72, november 11-13, Hong Kong

**[PV01] Pulkki, V.**
*(1997), Virtual Sound Source Positioning Using Vector Base Amplitude Panning*
Journal of the Audio Engineering Society, 45(6), pp. 456-466, 1997

**[RDD01] Raykar, V.C., Duraiswami, R., Davis, L.**
*(2003), Extracting Significant Features From The HRTF*
Proc. International Conference on Auditory Display, pp. 115-118, juli 2003, Boston, USA

**[RMW01] Rossing, T. D., Moore, F. R., Wheeler, P. A.**
*(2002), The Science of Sound, 3. utgave*
Addison-Wesley, San Francisco, USA
ISBN 0-8053-8565-7

**[SG01] Steinke, G.**
*(1996), Surround Sound – The New Phase: An Overview*
Journal of the Audio Engineering Society, 44, pp. 651, juli 1996

**[SJO01] Serrano, Jorge Ordóñez**
*(2004), Virtual Guide for a Virtual Heritage Environment*
Hovedoppgave ved Institutt for datateknikk og informasjonsvitenskap, NTNU, vår 2004

**[SK01] Shilling, R., Krebs, E.**
*(2002), Videogame and Entertainment Industry Standard Sound Design Techniques and Architectures for Use in Videogames, Virtual Environments and Training Systems*
ACM symposium on Virtual Reality Software and Technology, Hong Kong, Kina, 11-13 november 2002

**[TG01] Tsingos, N., Gascuel, J. D.**
*(1998), Fast Rendering of Sound Occlusion and Diffraction Effects for Virtual Acoustic Environments*
104th AES convention, Amsterdam, Nederland, mai 1998

**[TFNC01] Tsingos, N., Funkhouser, T., Ngan, A., Carlbom, I.**
*(2001), Modeling Acoustics in Virtual Environments Using the Uniform Theory of Diffraction*
Proceedings of ACM SIGGRAPH 2001, pp. 545-552, august 2001

**[TN01] Tsingos, Nicolas**
*(2001), A Versatile Software Architecture for Virtual Audio Simulations*
Proceedings of the International Conference on Auditory Display, Espoo, Finland, juli 2001

# Internett

**[WEB01] Microsoft DirectX Audio Overview**
Web (*http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndxgen/html/directx9devfaq.asp*)
Besøkt: 15.02.05

**[WEB02] EAX.Creative.com Redefining the Audio Experience**
Web (*http://eax.creative.com*)
Besøkt: 15.02.05

**[WEB03] Creative OpenAL Programmer's Reference Version 1.0**
Web (*ftp://216.61.164.51/OpenALProgGuide.pdf*)
Besøkt: 15.02.05

**[WEB04] 3-D Audio for HCI**
Duda, R.O., Department of Electrical Engineering, San Jose State University
Web (*http://interface.cipic.ucdavis.edu/CIL_tutorial/3D_home.htm*)
Besøkt: 03.03.05

**[WEB05] HRTF Measurements of a KEMAR Dummy-Head Microphone**
Gardner, B., Martin, K., MIT Media Lab
Web (*http://sound.media.mit.edu/KEMAR.html*)
Besøkt: 31.03.05

**[WEB06] 3DsoundSurge: Gamer's Guide to 3D Sound and Reverb API's**
Web (*http://www.3dsoundsurge.com/features/articles/APIs/APIs.html*)
Besøkt: 15.03.05

**[WEB07] Aureal A3D Central**
Web (*http://members.optushome.com.au/kirben/*)
Besøkt: 12.03.05

**[WEB08] NCSA Sound Server Reference Manual**
Web (*http://www.isl.uiuc.edu/software/vss_reference/vss3.0ref.html*)
Besøkt: 12.03.05

**[WEB09] cSounds.com – Official Csound Website**
Web (*http://www.csounds.com*)
Besøkt: 12.03.05

**[WEB10] Would You Have Sound Managed By a Scenegraph?**
Web (*http://www.gamedev.net/community/forums/topic.asp?topic_id=308566*)
Besøkt: 30.03.05

**[WEB11] Creative Labs Developer Relations**
Web (*http://developer.creative.com/landing.asp?cat=1&sbcat=31*)
Besøkt: 10.02.05

**[WEB12] BurningPixel Productions: StereoGrapher MAX**
Web (*http://burningpixel.com/Max/StereoG.htm*)
Besøkt: 19.03.05

**[WEB13] Partners In Rhyme**
Web (*http://www.partnersinrhyme.com/pir/PIRsfx.html*)
Besøkt: 14.05.05

**[WEB14] Stereo Movie Maker**
Web (*http://stereo.jpn.org/eng/stvmkr/index.html*)
Besøkt: 19.03.05

# Vedlegg A Storyboard for animering av Sverresborgmodellen

Denne delen er ment som en oppsummering av animeringen av Sverresborgen i 3D Studio MAX. Til grunn ligger det tekstlige manuskriptet fra Regin Meyer, og hovedhensikten har vært å skape en oversikt slik at eventuelle fremtidige endringer av animasjonen/modellen blir enklere.

Tidskolonnen angir i *minutter.sekunder* hvor på tidslinjen aktiviteten foregår. Der flere tidspunkter er oppført, betyr dette at kameraet skifter retning eller plassering underveis. Frames viser henholdsvis start- og slutt-nummer for bildene som er nødvendig å bruke gitt tiden som aktiviteten tar (med 24 frames i sekundet). Lyd beskriver hvilke lyder som kan høres under aktiviteten, og beskrivelse gir en kortfattet forklaring på hva som skjer på det gitte tidspunktet (plassering, animasjoner etc.).

| Tid (s) | Frames | Lyd | Beskrivelse |
|---|---|---|---|
| **-30-0** | **X** | Bakgrunnsmusikk. | Introduksjonstekst eller stemme. |
| **0.0-0.30** | **0-720** | Bakgrunnsmusikk? | Fugleperspektiv, sirkuler rundt borgen. |
| **0.30-0.34** | **720-816** | Fade bakgrunnsmusikken bort. | Ned fra fugleperspektiv til guiden (hjelm, våpen, ringbrynje) på forsiden av borgen. |
| **0.34-0.36** | **816-864** | | 2 sek pause. |
| **0.36-0.56** | **864-1344** | Replikk. | Velkomstreplikk av guiden. |
| **0.56-1.21** | **1344-1944** | Fotsteg, pust, pes, replikk. | På vei opp til borgen og porttårnet. Guiden snakker. |
| **1.21-1.36** | **1944-2304** | Replikk. | Foran hevet vindebro og senket gitter. Guiden snakker. |
| **1.36-1.38** | **2304-2352** | Replikk. | Guiden roper om senking av vindebroen. |
| **1.38-1.43** | **2352-2472** | Vindebrokjetting. | Senking av vindebro. |
| **1.43-1.44** | **2472-2496** | | 1 sek pause. |
| **1.44-1.48** | **2496-2592** | Jernlyder av gitter. | Gitteret heves. |
| **1.48-1.54-1.58** | **2592-2736-2832** | Lyd av kjetting pga. vindebro + smell av vindebro som går opp. | Gjennom passasjen i porttårnet. |
| **1.58-2.01-2.04** | **2832-2904-2976** | Bakgrunnssnakk, latter fra soldater, fotsteg. | Snur venstre mot vaktrommet, der soldater sitter og snakker. |
| **2.04-2.06-** | **2976-3024-** | Bakgrunnslyder: smed, snakking, bjeff ++. | Inne i borggården. ”Yrende liv”: se * under. Observerer en liten stund før |

| | | | |
|---|---|---|---|
| **2.25** | **3480** | | guiden begynner å snakke. |
| **2.25-<br>2.34** | **3480-<br>3696** | Replikk, bakgrunnslyder som over. | Ser mot kongeboligen på andre siden (følg replikk fra guide på de ulike bygningene). Går mot midten av borggården og blir stående der. |
| **2.34-<br>2.41** | **3696-<br>3864** | Replikk, bakgrunnslyder som over. | Ser mot venstre side langs ringmuren på de ulike bygningene / verkstedene. |
| **2.41-<br>2.49** | **3864-<br>4056** | Replikk, bakgrunnslyder som over. | Ser mot kjøkkenet. |
| **2.49-<br>2.52-<br>3.05** | **4056-<br>4128-<br>4440** | Replikk, bakgrunnslyder som over. | Ser mot brønnen. |
| **3.05-<br>3.09** | **4440-<br>4536** | Replikk, bakgrunnslyder som over. | (med retning mot hovedbygningen). Fokuserer på øvre, venstre rom (kapell). |
| **3.09-<br>3.13** | **4536-<br>4632** | Replikk, bakgrunnslyder som over. | Fokuser på nederste etasje. |
| **3.13-<br>3.18** | **4632-<br>4752** | Replikk, bakgrunnslyder som over. | Fokuser mot den andre øvre delen. |
| **3.18-<br>3.28** | **4752-<br>4992** | Replikk, bakgrunnslyder som over. | Fokuser mot den andre nedre delen. |
| **3.28-<br>3.38** | **4992-<br>5232** | Replikk, bakgrunnslyder som over. | Snu mot kongeboligen. |
| **3.38-<br>3.43-<br>3.53-<br>4.03-<br>4.14** | **5232-<br>5352-<br>5592-<br>5832-<br>6096** | Replikk, bakgrunnslyder som over + engelsktalende håndverkere? | Går mot kongeboligen, langs rekken av verksteder langs ringmuren. |
| | | Replikk, bakgrunnslyder som over + bjeffende hund. | På veien kommer det bort en hund som bjeffer. |
| | | Replikk, bakgrunnslyder som over. | Se opp mot ringmuren der soldater patruljerer. |
| **4.14-<br>4.26** | **6096-<br>6384** | Replikk, bakgrunnslyder som over. | Passerer to soldater som spiller brettspill. |
| **4.26-<br>4.34** | **6384-<br>6576** | Replikk, bakgrunnslyder som over + slag og hamrelyder. | Hos smeden. |
| **4.34-<br>4.43** | **6576-<br>6792** | Replikk, bakgrunnslyder som over + slag og hamrelyder. | Hos smeden. Ser en person som sitter og spisser et spyd utenfor smia. |
| **4.43-<br>4.47-<br>4.58** | **6792-<br>6888-<br>7152** | Bakgrunnslyder som over + slag og hammerlyder. | På vei bort fra smeden. To personer som snekrer på et skur. |
| **4.58-<br>5.10** | **7152-<br>7440** | Replikk, bakgrunnslyder som over. | Ved kongeboligen. Hilser på kongen. |
| **5.10-<br>5.14** | **7440-<br>7536** | Replikk, bakgrunnslyder som over. | Hilser/ser på ”eldstekaren”. |
| **5.14-** | **7536-** | Replikk, bakgrunnslyder | Hilser/ser på to jenter. |

| | | | |
|---|---|---|---|
| **5.17** | **7608** | som over. | |
| **5.17-5.22** | **7608-7728** | Replikk, bakgrunnslyder som over. | Ser mot kongen (forteller om kona, vet ikke hvor hun er). |
| | | Bakgrunnslyder som over + lyder av nøkkelknippe. | Ser mot kona som kommer gående med nøkkelknippe. |
| **5.22-5.43** | **7728-8232** | Replikk, bakgrunnslyder som over. | Kona står foran kongen og kjefter på han. |
| **5.43-5.59** | **8232-8616** | Replikk, bakgrunnslyder som over. | Kona snur seg mot gjestene. |
| **5.59-6.04-6.08-6.10-6.12-6.15** | **8616-8736-8832-8880-8928-9000** | Bakgrunnslyder som over. | Kona går oppover trappa utenfor kongeboligen, og gjestene følger etter. |
| **6.15-6.20-6.23** | **9000-9120-9192** | Fotsteg, diffuse lyder? | I 2.etg går veien gjennom huset og mot vinduet på andre siden av huset. |
| **6.23-6.27.5** | **9192-9300** | Diffuse lyder? | Ser ut av vinduet en stund mot Munkholmen. Kameraet zoomer inn mot havet. |
| | | Replikk (rop om at fienden angriper) + lur (varselhorn). | Panorer kameraet bort fra vinduet og fade til svart eller hvitt. |

**Tabell 2 – Tekstlig storyboard for Sverresborgmodellen**

* Animasjoner inne i borggården:
- løpende og bjeffende hund
- kvinner som henter vann fra brønnen
- soldater som prater og diskuterer
- håndverkere som reparerer murverk
- tjenestefolk som bærer proviant og varer
- et par guttunger som leker
- soldater som patruljerer på ringmuren
- enkelte personer inne i borggården snur seg mot gjesten, de fleste ikke

# Vedlegg B Manuskript for guidet tur

Under følger manuskriptet som ble utarbeidet av Regin Meyer i forbindelse med den guidede turen omkring på borgen. Dette var utgangspunktet for filmen som ble laget i forbindelse med utstillingen, og inneholder dialogen samt beskrivelse av hva man til enhver tid ser av aktivitet inne på borgplatået. Manuskriptet er tatt med i denne sammenheng for å lette eventuelt fremtidig arbeid med modellen og videoen.

## Introduksjon

**Beskrivelse:**
Middelaldermusikk, stemningsmusikk eller lydeffekter spilles i bakgrunnen mens det kommer en kort innføring i den historiske konteksten og Sverresborgs opprinnelse. Dette blir enten lest opp av en fortellerstemme eller en rulletekst vises på skjermen slik som "på film".
Forslaget under er inspirert av historiske filmer:

"*...Norge, Herrens år 1197… utbrudd av borgerkrig har herjet landet i mer enn 60 år. Mange blodige lidelser har tynget uskyldige kvinner, menn og barn, så vel byfolk som bønder. Gårder har blitt brent, hus har blitt plyndret, Nidaros, Oslos og Bjørgvins gater stormet i blod… Rotløse opprørere kjemper mot hirdmenn, lendmenn og stormennenes hærskarer. Birkebeinere og Baglere måler kreftene på slagmarken… Det er mange om beinet i denne striden om hvem som skal bli landets enerådende konge. En av dem, Sverre Sigurdsson fra Færøyene har bygget to borger som folk knapt har sett maken til, en ved Nidaros og en ved Bjørgvin. Sverre legger solide fundamenter når han først skal grunnlegge landets nye kongedynasti. Borgen på Steinberget ved Nidaros blir trolig Norges første borg bygget i stein. Etter at Kong Magnus falt i slaget ved Fimreite i 1184 hylles Sverre som enerådende konge. Men misnøyen er fremdeles stor blant restene til de gamle makthaverne…*

*Kong Sverres borg på Steinberget ved Nidaros…"*

**Beskrivelse:**
Tilskuerne får nå se borgen i fugleperspektiv. Perspektivet går en runde rundt borgen og ender til slutt opp et stykke nedenfor borgen i skråningen i sydvest. Her står guiden som tar i mot gjestene.

## Velkomst

**Beskrivelse:**
Guiden vår er en birkebeinersoldat som tilhører mannskapet på borgen. Han er ikledd ringbrynje, hjelm, våpen etc.. Velkomsten foregår nede ved området hvor Lo kirka står i dag noe som vil gi tilskuerne/gjestene anledning til å se borganlegget på avstand.
Han tar vel imot gjestene på ekte sagamaner. Velkomstfrase sies.
**Replikk:**
*"Det er godt å se at fremmedfolk fremdeles våger seg ut på landeveien i slike urolige tider. Hvem er så dere? Dere ser vennlige ut og ligner slettes ikke på noen farlige baglere. Dere har*

*vel ikke sett noe til det fandenske pakket på veien hit? Hmmm…? Vel, dere får følge etter så skal jeg vise dere hvordan vi har det her på kong Sverres borg…*

## På vei opp til borgen

**Beskrivelse**:
Guiden med gjestene går så opp bakken i retning mot porttårnet.
På vei opp forteller guiden raskt om situasjonen på borgen: Beredskap, antall menn og kvinner.., kong Sverres fravær ettersom han er nede i Viken, Torstein Kugad er derfor fungerende borgherre i kongens fravær… mye bygningsarbeid pågår inne og utenpå borgen, birkebeinerne har fått god hjelp av håndverkere fra Kristkirken…
Bakgrunnslyder: Fotsteg mens guiden og gjestene går opp bakken.

**Replikk:**
*"…pust …pes… nå er kongen selv dessverre ikke til stedet i borga, han er nede på Austlandet for å få skikk på de opprørske vikværingene… Det er derfor Torstein som er sjef her nå, Torstein Kugad fra Gaustad på Byneset, ellers har vi ganske høy beredskap på borgen for tiden med vel 80 mann, alle godt bevæpnet, de fleste trøndere, og noen av dem har til og med tatt med seg familien… Det kommer godt med i det daglige virket…*

## Foran porttårnet

**Beskrivelse:**
Til slutt kommer følget opp til porttårnet og står like foran vindebroen som er hevet.
Guiden forteller om ryktene som sier at Baglerne er på vei opp Gudbrandsdalen mot Nidaros,
Guiden roper til vaktmann opp i tårnet om at vindebroen skal senkes.

**Replikk:**
*"Som dere ser så holder vi for tiden vindebrua hevet. Ryktene sier nemlig at krigslystne Balgerene er på vei opp Gudbrandsdalen med en stor hærflokk og kan komme når som helst. Slik som de brant og herjet sist de var i Nidaros så er det lurt å ikke ta noen sjanser… "*

Guiden roper til vaken oppe i porttornet:
*"Senk vindebroa!"*

## Gjennom passasjen i porttårnet

**Beskrivelse:**
Gjestene går rolig gjennom passasjen og hører et lite smell bak seg når vindebrua går opp.
Passasjen er noe mørkere enn lysforholdene utendørs. Til venstre er døren åpen inn til vaktrommet hvor gjestene ser vaktmenn som prater. (Er det her også mulig å få laget et fallgitter i passasjen som heves?) Foran seg ser gjestene lyset som kommer inn fra borggården.
Bakgrunnslyder (med ekstra romklang ettersom scenen foregår inne i en lukket passasje):
-Vindebrua som heves og svakt smeller når den treffer veggen.
-Fotstegene til gjestene når de går.
-Bakgrunnslyder fra vaktrommet i: *latter, prating i bakgrunnen…*

## Inne i borggården

**Beskrivelse:**
Gjestene møter synet at et yrende liv inne på borggården. Kvinner som henter vann i brønnen, soldater som prater og diskuterer, håndverkere som reparerer murverk, tjenestefolk som bærer proviant og varer, et par guttunger som leker, hund som bjeffer, soldater som patruljerer på ringmuren. Noen av personene inne på borggården snur seg og ser på gjestene som ankommer, andre fortsetter med sitt daglige virke.
Guiden gir en rask beskrivelse av borgplatået. Perspektivet/synsfeltet fokuserer på de forskjellige bygningene og stedene etter hvert som han prater og beskriver.
Guiden og gjestene snur seg for å se på bygningsfløyene på hver side av porttårnet. Guiden forteller og viser hvor kapellet ligger, boligkvarter ligger, lagerrom, fengsel…etc.

Bakgrunnslyder: Lydeffekter som gir inntrykk av et mangfold av aktiviteter.

**Replikk:**
*"Sånn, endelig fremme i borggården. Her bor og lever vi altså, inne på borggården godt beskyttet bak murene. Et rett så livlig og trivelig sted nå når det er ro og fred, men en heller dyster og alvorlig plass når fienden truer, det kan jeg love dere…*
*Vi har forskjellige hus og bygninger til mangt slags bruk.*
*I den andre enden ser dere kong Sverres praktfulle herberge, men det er dessverre ikke så ofte han bor der nå for tiden, langs ringmuren ligger verksteder, soldatenes forlegningshus, stall, lagerhus, og på den andre siden står kjøkkenet. Og like foran oss, kanskje det viktigste av alt, nemlig brønnen vår. Hvis baglere eller andre fiender skulle være frekke nok til å beleire oss må vi passe på at de ikke sulter oss ut, ja som dere kanskje vet: uten brønn ingen borg…*
*og bak oss…*(gjestene snur seg)
*…har vi øverst til venstre kappellet vårt, i etasjen under ligger boligkvarter for gjester, fintfolk og hirdmenn, på den andre siden gildehallen til møter, fester og høytidelige begivenheter, og under det et godt bevoktet fengsel. Der har vi fremdeles et par kulvunger lagt i fotlenker til skrekk og advarsel, etter forræderi og mannedrap…*

## På vandring mot den kongelige residensbygningen (langs ringmuren)

På vei mot den kongelige residensbygningen som ligger lengst nord på platået passerer gjestene flere verksteder og lagerbygninger i lafta tømmer. Ved et av verkstedene får gjestene hilse på en av håndverkerne, for eksempel en smed eller en snekker? Under vandringen ser mann soldater som går oppe på ringmuren. Kanskje kommer det en hund som bjeffer etter gjestene?

**Replikk:**
*Følg på, så skal jeg vise dere litt omkring her borte. Pass dere for den fillebikkja, den er litt gretten i dag og kan bite!*
*Det er nemlig slik at så lenge ikke fienden lusker omkring må vi bruke all den tiden vi har til forberedelser. I fredstid er derfor kanskje håndverkere de viktigste på borgen for å gjøre den sterk og stødig mot beleiringer. Mange dyktige håndtverkerne har vi hentet fra byen og noen utenlandske speisalister kommer til og med fra Kristkirka* (Bakgrunnslyd: noen som snakker engelsk el. et fremmed språk?)*… Deler av den gamle ringmuren må repareres, veggene skal kalkes, nye laftahus skal settes opp, tømmer som skal skiftes ut, proviantlagrene må fylles opp, reservedeler, pilspisser og bolter må lages, nye våpen skal smies… ja, sukk, mye slit når mann er i kongens tjeneste, også for de som ikke er stridførne menn…*

## Underveis…

Underveis passerer guiden og gjestene to soldater som sitter ved et bord og spiller brettspill ”hnefatafl” (uttales: nævatavl). Guiden vår kjefter sint på dem:
**Replikk:**
*Sa jeg ikke at dere to skulle ut og kalke murene? I stedet sitter dere her og spiller hnefatafl….. Av gårde før jeg setter Borghild på dere !!…*
*..spillegale vestlendinger…*

## Underveis…

Like ved passerer følget en bygning som er smia på borgen. Det kommer røyk ut en åpning i taket. Innenfra hører vi ”slag og hamrelyder”. Guiden:

**Replikk:**
*…det er bra å høre at ihvertfall noen tar arbeidet på alvor, her inne holder smedene våre til, Sjur og Eirik Stål,*

Og litt ved siden av smia sitter en kar på en stubbe og sliper spydspisser,
Guiden:
*…og her sitter han Brynjolv og sliper våpen…..bra Brynjolv!... sørg for at de spydene biter godt fra seg hvis vi kommer i kamp….*

Og bortenfor han igjen står et par snekkere som hamrer på et skur…

## Ved den kongelige residensbygningen

Her får gjestene hilse på Torstein Kugad, Sverresborg borgherre og kongens stedfortreder. Torstein forteller at han har tatt med seg kona, barna og tjenestefolket fra storgården hans på Byneset. Gjestene blir introdusert for familien. Torstein:

**Replikk:**
*…velkommen hit til Sion, kongens borg, jeg heter Torstein Kugad og bestemmer her på borgen når kongen selv ikke er til stede, som dere ser har jeg med meg tjenestefolket og familien fra garden min på Byneset…*
*…dette er eldstekaren… han blir nok en stor birkebeiner en dag…*
*…her har vi jentene...*
*…og kona mi, Borghild hun er et eller annet sted…*

Kona, Borghild kommer plutselig, hun har et stort nøkkelknippe festet i beltet (lyd av nøkler som rasler). Hun er lettere irritert, noen har forsynt seg av forrådslagrene. Hun begynner å kjefte på Torstein:

**Replikk:**
*…så der er du!! Det er fælt så mye denne borgrampen driver og forsyner seg av forrådslagrene våre når mann ikke passer på… Nå må du få skikk på dem Torstein! Vi har sikkert gått tom for mat lenge før vikværingene kommer og beleirer oss… Og Torstein, ditt surrehode, hvor mange ganger har jeg ikke sagt at du må huske å lukke latrinedøra etter deg…den glemskheten din kan komme til å koste oss dyrt en dag!*
Borghild henvender seg til gjestene:

*...så? Har vi fått fremmedfolk på besøk? Jeg pleier vanligvis ikke å vise fremmede rundt i kongens herberge… det ville Sverre nok ikke like… men jeg kan ta dere med inn hit litt så kan dere få nyte utsikten…*

## Inne i et rom i den kongelige residensbygningen

Kona til Torstein tar gjestene med inn i et lite rom i residensbygningen. De stopper ved et vindu mot fjorden og nyter utsikten i noen sekunder. Plutselig hører de noen som blåser i lur og folk som roper:

**Replikk:**
*"BAGLERNE KOKMMER!..."*