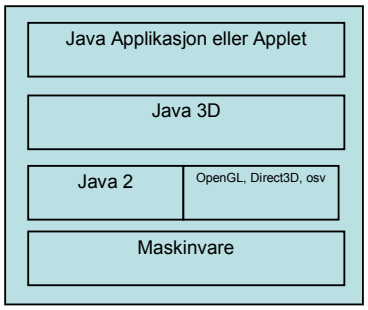
Vedlegg 2

1.1 Hva er Java 3d?

Java 3d er en API (Application Program Interface) laget for å gi Java utviklere mulighet til å utvikle omfattende, plattform uavhengige 3d appletter og applikasjoner som inneholder interaktiv grafikk og lyd.
APIen inneholder de mest essensielle verktøyene som man finner i de ”lav-nivå APIene” OpenGL og Direct3d.
Forskjellen på disse lavnivå APIene og Java 3d er at Java 3d inneholder høynivåklasser som gir utvikleren mulighet til å fokusere på hva som skal tegnes isteden for hvordan det skal tegnes opp.



Figur 1 - Java 3D ligger som et lag over lavnivå APiene.

Selv om Java 3d er en høynivå API som ligger over lavnivå APIene så har man også mulighet til å utføre lavnivå rendering i Java 3d hvis det er nødvendig. Selv om slik lavnivårenderingskontroll kan være nyttig i noen tilfeller så er det oftest en unødvendig byrde for utvikleren.

For å bygge opp et Java 3d program så benyttes det en scenegraf programmeringsmodell der applikasjonsprogrammet beskriver scenen med et høynivå interface. Denne scenegrafen gjør det mulig for den som utvikler 3d programmer å fokusere på hva som skjer med objektene i scenen, mens Java 3d kjøremiljøet og renderingsmotorer gjør arbeide med å finne ut hvordan scenen skal tegnes opp og hvordan den kan framstilles fortest mulig. Dette gjør at utvikleren kan fokusere på de mer generelle aspektene ved utviklingen. Slik som brukerinnputt (mus tastatur), og interaktiv oppførsel. Utvikling av 3d grafikk med hjelp av høynivåklasser gjør at utvikleren kan fokusere på den virtuelle verden isteden for å konstruere triangler og polygoner.

1.2 Hvordan konstruerer man et Java 3d program?

Java 3d programmer bygges opp av en scenegraf programmeringsmodell.
Applikasjonsprogrammet inneholder en beskrivelse av en scene, mens Java 3d utfører det som trengs for å vise scenen.
Scenegrafen brukes for å strukturere og organisere grafikken som skal bli framvist til brukeren og blir i Java 3d kalt det virtuelle universet.

Det virtuelle universet inneholder alle de grafiske elementene til scenen, slik som objektene som skal framstilles, oppførselen til objektene, lys og skyggeeffekter.

1.2.1 Scenegraf

En scenegraf er et ”familie tre” som inneholder scenedata (noder). Familietreet er bygd opp av barn og foreldre. Barna representerer objekter som lys, lyder og animasjoner, mens foreldrene blir brukt til å gruppere barn og andre grupper.
Applikasjonen bygger opp en scenegraf av barn og foreldre ved å bruke Java 3D klasser og metoder. Scenegrafen bygges opp av komponenter som:

- Grupper og transformer
- Shaper (geometri og utseende)
- Lys
- Skygge og bakgrunn
- Lyd og lydmiljøer
- Oppførsel klasser

For å strukturere dataene som skal framvises så finnes det en rekke grupperingsklasser. Disse klassene blir kalt foreldre i ” familietreet” og formerer strukturen til scenegrafen.
De organiserer data, kontrollerer hvilke deler av scenen som er rendert, holder data som hjelper til å forandre posisjon, orientering og størrelsen på objektene i scenen.

For å kunne bygge opp en scenegraf er det en fordel å vite forskjellen på klassene som representerer grupper, løvnoder og nodekomponenter. Disse klassene forklares i de neste avsnittene.

1.2.2 Grupper (foreldreklasser)

Group

Group er den mest generelle gruppen og inneholder enkle metoder for å legge til, innføre, fjerne og hente ut barn fra en gruppe:

```
void addChild(Node child);
void insertChild(Node child, int index);
void removeChild(int index);
void setChild(Node child, int index);
Node getChild(int index);
int numChildren();
```

Ser man på de metodene som er tilgjengelig i klassen Group så har den de samme funksjonalitetene som klassen java.util.vector og er dermed enkel å bruke for de som er kjent med denne datastrukturen.

Skal man legge til et barn til en gruppe kan man bruke følgende kode:

Oppretter et barn:

```
Shape3D myshape = new Shape3D(mygeo, myappear);
```

Oppretter gruppen og legger barnet til gruppen:

```
Group mygroup = new Group();
mygroup.addChild(myshape);
```

BranchGroup

En BranchGroup arver fra Group og har dermed alle egenskapene til Group klassen i tillegg til følgende egenskaper:

- Kan bli lagt til SimpleUniverse (klasse som hjelper til med oppsett av scenen)
- Kan bli optimalisert (kompilert)
- Kan være barn til hvilken som helst annen gruppe
- Kan frakobles fra foreldregruppen hvis forelderen har satt de riktige kapasitetsbitene
- Er den eneste gruppen som kan legges til en live node i scenen

TransformGroup

Alle shaper er bygd innen et delt world coordinate system i Java 3d. En TransformGroup lager et nytt koordinatsystem for barna som legges til gruppen. Dette koordinatsystemet blir likt for alle barna og er relativt til sin foreldre.
Siden alle barna som legges til TransformGroupen får det samme koordinatsystemet, så vil en transformasjon på gruppen føre til at alle barna får samme translasjon.

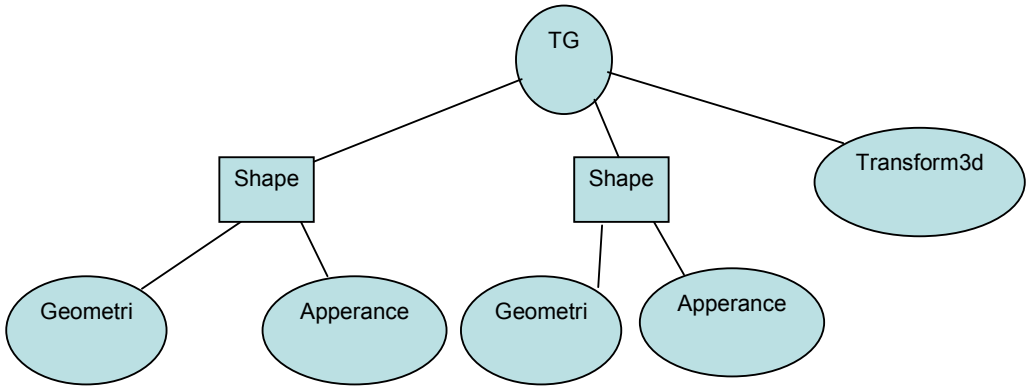
Skal man for eksempel rotere to objekter i scenen samtidig så kan man benytte følgende kode:

```
Shape3D myshape1 = new Shape3D(mygeo, myappear);
Shape3D myshape2 = new Shape3D(mygeo, myappear);

Transform3d mytrans = new Transform3d();
mytrans.rotZ(0.52);// 30 grader
TransformGroup mygroup = new TransformGroup();

mygroup.setTransform(mytrans);
mygroup.addchild(myshape1);
mygroup.addchild(myshape2);
```

Her opprettes det to objekter som skal visualiseres i Java. Det opprettes en transform3d som gjør det mulig å rotere, translere eller forandre størrelse på objekter. I dette eksemplet så roteres nodene 30 grader rundt z aksen med metoden rotZ(0.52).
Transform3d objektet legges til Gruppen.
Barna blir lagt til den samme gruppen og får da det samme koordinatsystemet. Dette gjør at begge objektene roteres 30 grader om z aksen.
For å få oversikt over et Java 3d program så lønner det seg ofte å tegne opp scenegraf modellen. Nedenfor er det en modell som beskriver hvordan eksempelkoden er strukturert.



Figur 2 scenegrafen til TransformGroup

I tillegg til at figurene (Shape) og Transformen blir lagt til TransFormGroup så er det også lagt til geometri og apperance til shape3d objektene.
Apperance er en nodekomponent som inneholder all informasjon om utseende til en shape node. Dette er informasjon som beskriver objektets farge, lysrefleksjoner, teksturer eller annen utseende informasjon.
Geometrien beskriver hvordan shape 3d er bygd opp. Dette kan være alt fra en enkel linje til avanserte 3d objekter med mange 1000 punkter og polygoner.

1.2.3 Løvnoder (barn)

I tillegg til grupperingsnodene så finnes det en rekke noder som kalles løvnoder. Disse nodene blir brukt til å spesifisere figurer (shapes), lyd, lys og oppførsel. Disse nodene kan ikke ha noen barn men kan referere til nodekomponenter.

1.2.4 Nodekomponenter

Nodekomponentklassene brukes til å spesifisere geometrien, utseende, teksturer og material komponenter til Shape3d noder. Disse nodene er ikke del av scenegrafen, men blir referert til via shape3d noder. Det er også mulig for flere shape3d noder å referere til samme node komponent.

1.3 Hvordan skrive et enkelt Java 3d program?

For å kunne vise objekter i Java 3d så trenger man å:

1. Lage et virtuelt univers som skal inneholde scenen
2. Lage en datastruktur som kan inneholde en gruppe av objekter
3. Legge objekter til gruppen
4. Posisjonere viewer slik at man kan se objektene
5. Legge til gruppen av objekter til universet.

På neste side er det et eksempel som viser hvordan et Java 3d program kan programeres:

```
import com.sun.j3d.utils.universe.SimpleUniverse;
import com.sun.j3d.utils.geometry.ColorCube;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;

public class Hello3d
{

    public Hello3d()
    {

       SimpleUniverse universe = new SimpleUniverse();

       BranchGroup group = new BranchGroup();
       Transform3D mytrans = new Transform3D();
       mytrans.rotY(0.52);// 30 grader
       TransformGroup tgroup = new TransformGroup();

       tgroup.setTransform(mytrans);
       tgroup.addChild(new ColorCube(0.3));
       group.addChild(tgroup);

       universe.getViewingPlatform().setNominalViewingTransform();

       universe.addBranchGraph(group);

    }

    public static void main( String[] args )
    {

       new Hello3d();

    }

} // end of class Hello3d
```

1.3.1 Forklaring kode

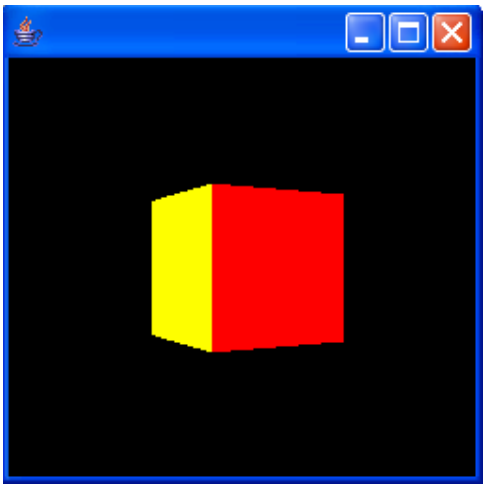
Programmet går ut på å legge til en 6 kantet kube til universet og rotere den 30 grader om y aksen.
For å få til dette så opprettes det først et objekt av klassen SimpleUniverse. SimpleUniverse er en klasse som oppretter det minimale brukermiljøet for raskt og enkelt kunne få et Java 3d program til å kjøre. Klassen setter opp alle objektene som trengs for å kjøre på visnings siden av scenegrafen.
Når ”universet” er opprettet så må selve scenemodellen lages. Dette gjøres ved først å opprette et objekt av klassen BranchGroup. Grunnen til at man starter med en BranchGroup er at det kun er instanser av BranchGroup som kan legges til universet, og dette gjør at alle Java 3d programmer minst har en instans av denne klassen.

Det opprettes en Transform3D som benytter metoden rotY(0.52) for å rotere koordinatsystemet 30 grader.

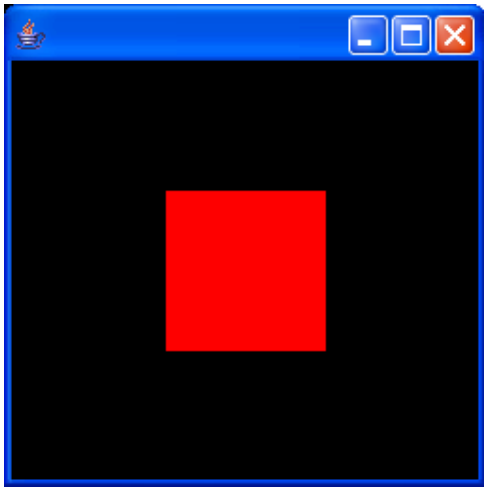
For å kunne translere objekter i scenen så må man opprette en TransformGroup og legge transformen til instansen av denne. Alle barn som legges til denne gruppen vil få denne translasjonen (siden de da deler det samme koordinatsystmet).
Det opprettes en instans av klassen ColorCube som legges til TransformGroup. ColorCube er en ”terning” som er ferdig definert objekt med forskjellig farge på hver av sidene. Før vi legger BranchGroupen til scenen så settes ViewPlatformen litt tilbake slik at objektene på scenen kan vises.

1.3.2 Skjermbilde og scenegraf for eksempel kode

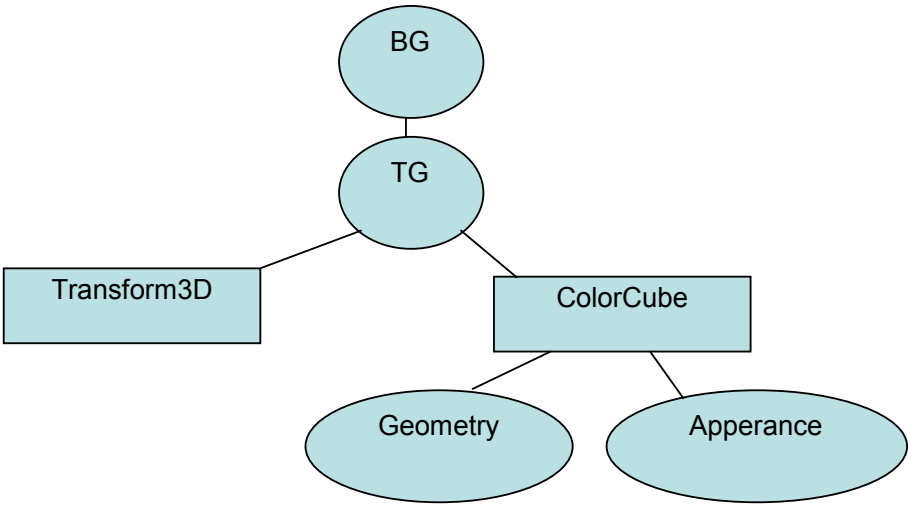
Nedenfor er det et skjermbilde fra eksempelkoden og et skjermbilde der kuben ikke er translert.



Skjermbilde 1 - Kuben med translasjon



Skjermbilde 2 - Kuben uten translasjon



Figur 3 - Scenegrafen til programmet

1.4 Viktige klasser og definisjoner i Java 3d

1.4.1 Behaviour

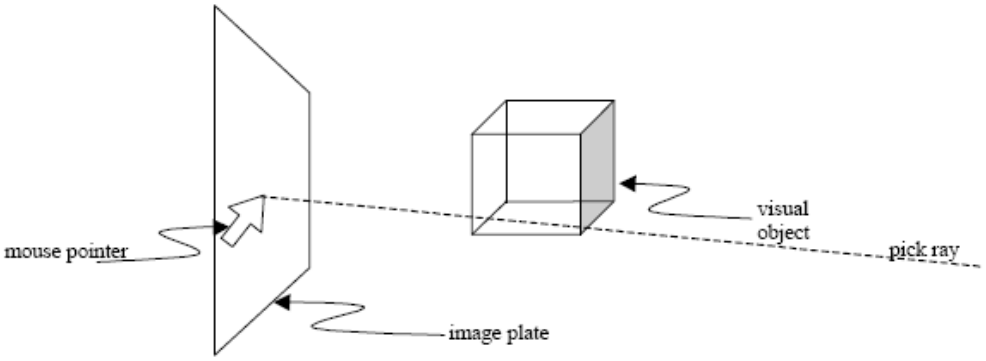
Behavior objekter i scenegrafen har som oppgave å forandre scenegrafen eller objekter i scenen i forhold til brukerinnputt, kollisjon av objekter eller etter et gitt tidsintervall.
Disse forandringene er for eksempel oppførsler som legger til og fjerner objekter fra scenegrafen, forandrer attributter til objekter i scenegrafene eller reorganiserer objekter. Det finnes en rekke ferdiglagde behaviour klasser i Java 3d men det er også mulig å lage egne behaviourklasser.
Det flere forhåndsdefinerte oppførselklasser, men de mest brukte er nok de som gjør det enkelt å legge til navigasjon i 3d universet.
Man har for eksempel en rekke musoppførselklasser der man har mulighet til å rotere og translere objektene i scenen:

Klasse		***Mus handling***
MouseRotate	Roterer objektet rundt koordinatsystemet.	Hold nede venstremusknapp og beveg musen
MouseTranslate	Flytter objektet nord, sør, øst, vest, i forhold til musbevegelsen	Hold nede høyre musknapp og beveg musen
MouseZoom	Zoomer inn eller ut i 3d universet	Hold nede midtre musknapp og beveg musen opp eller ned

Tabell1 - musoppførselklasser

1.4.2 Picking

Pickingklassene gir utvikleren en mulighet til å lage 3d applikasjoner hvor man kan velge 3d objekter fra scenen.
Picking er en oppførsel som blir kjørt når det skjer en mushendelse. Når en bruker plasserer muspekeren over et visuelt objekt og trykker på en musknapp, så blir oppførselen startet opp og begynner en plukke operasjon. Ut fra den posisjonen muspekeren har så blir det opprettet en stråle inn i 3d universet. De objektene som blir truffet av denne strålen er kandidater for utvelgelse. Resultatene fra plukkingen returnerer en liste av objekter.



Figur 4 - plukking av objekter

For å velge ut objekter fra scenen så er det 2 forskjellige moduser, med svært forskjellig nøyaktighet:
Den enkleste modusen for plukking er Bounds.
I modusen Bounds så blir utvelgelsen gjort via en rask skjæringspunkttest. Den er rask og enkel å sette opp men resulterer i svært unøyaktige resultater.
Den er unøyaktig siden den baserer seg på grenser som er knyttet opp mot geometrien til figurene, disse grensene passer ikke alltid like godt med geometrien og gjør at man noen ganger kan velge objekter uten å være i nærheten av dem, mens man andre ganger ikke klarer å treffe objektet man forsøker å velge. En mye mer nøyaktig plukkemodus er geometry. Metoden bruker selve geometrien for å bestemme om man har valgt et objekt eller ikke, denne metoden er derimot vanskeligere å bruke siden man må sette opp en rekke kapasitetsbit.

1.4.3 Compiled/live scene

Ved å utføre compile på scenegrafen så blir scenegrafen gjort om til en mer effektiv intern representasjon som brukes av Java 3d til å forbedre rendering ytelsen.

Compile kan kun brukes på instanser av BranchGroup (scenen) og blir som oftest utført rett før den legges til SimpleUniverse. Når en BranchGroup blir lagt til universet så blir den betegnet som live.

1.4.4 Capability bits

Java 3d er laget for å kunne optimaliseres internt. For å videre optimalisere hastighet så er det en rekke funksjoner som er slått av i utgangspunktet. For å skru på disse funksjonene så er det en rekke kapasitetsbit som må settes.
Skal man for eksempel forandre utseende på objekter i scenen under kjøring så må man sette på kapasitetsbitet : ENABLE_APPEARANCE_MODIFY på hvert objekt.

Hvis programmet prøver å forandre utseendet på et objekt som ikke har dette bitet satt så vil Java 3d produsere en feilmelding.

1.4.5 Loadere

Det er en rekke loaderklasser i Java 3d som gjør det mulig å lese inn 3d grafikk laget av avanserte 3d verktøy. Dataene leses inn fra filformatet og det lages en Java 3d representasjon av filens innhold som kan legges til Java 3d verden. Noen av filformatene som er støttet er 3DS (3D-studio), DXF (AutoCAD Drawing Interchange File), WRL (Virtual Reality Modeling Language)

1.4.6 Høynivåklasser for geometri

I Java 3d er det mulig å lage all mulig slags geometri, men opptegning av streker og polygoner tar som oftest lang tid og krever ofte en god del matematikk og kodelinjer for å få det resultatet man ønsker. I stedet for å tegne opp geometrien selv så finnes det en rekke høynivåklasser som representerer kjent geometri. Utvikleren kan fylle ut en rekke definerte parametere til geometrien mens Java 3d fyller inn detaljene. Det finnes for eksempel høynivåklasser i Java 3d for å konstruere firkanter, kjegler, sylindrer og kuler.