UNIVERSITETET I AGDER

E K S A M E N

Emnekode:	**IS-207**
Emnenavn:	**Algoritmer og datastrukturer**
Dato:	**31. mai 2007**
Varighet:	**0900-1300**
Antall sider inkl. forside:	4
Målform:	**Norsk**
Tillatte hjelpemidler:	**Alle trykte og skrevne**
Merknader:	

UNIVERSITETET I AGDER
SERVICEBOKS 422 4604 KRISTIANSAND
TELEFON 38 14 10 00 FAKS 38 14 10 01
ORG.NR. 970 546 200 MVA postmottak@uia.no www.uia.no

Innledning

I denne oppgaven skal vi ta for oss sanntids 3d grafikk som brukes bl.a. i simulatorer og spill. For slike applikasjoner er det viktig at oppdateringen av skjermbildet går raskt nok, ellers vil animasjonen på skjermen gå i rykk og napp. For at brukeren skal oppleve animasjonen som realistisk må skjermen oppdateres med helt jevne mellomrom, som ikke bør være større enn 1/30 sekund.

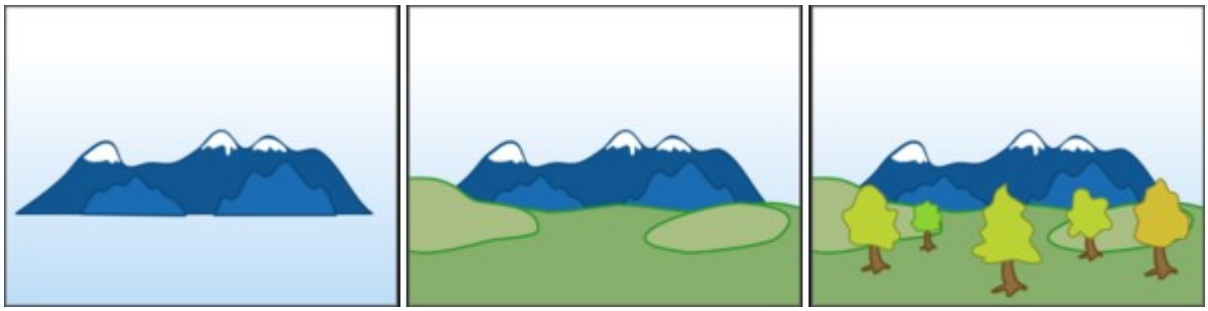
"Verden" i en simulator modelleres gjerne med en trestruktur. En node kan f.eks. representere en person, og inneholde informasjon om hvor personen er i forhold til sin foreldrenode, og hvilken vei han er vendt. Barnenodene til personen kan f.eks. representere armer og bein, og inneholder informasjon om hvordan de er plassert i forhold til kroppen. Disse nodene kan igjen ha barn og barnebarn, f.eks. hender og fingrer, avhengig av hvor detaljert simuleringen er. Foreldrenoden til personen kan f.eks. være en bil, som er plassert i forhold til sin foreldrenode osv. Rotnoden i treet representerer hele verden. Du finner et interface til en slik node i Vedlegg 1. (Du skal ikke skrive noen del av nodeklassen. Anta at den finnes.)

Hovedprogrammet er gjerne en evig løkke som vekselvis kaller en metode som gjør selve simuleringen (oppdaterer treet), og en metode som oppdaterer skjermbildet.

For å lage et skjermbilde må det utføres en del geometriberegner for hver node, for å finne ut om den er i synsfeltet eller ikke, og hvor i synsfeltet den er. Vi skal ikke gå inn på dette, men i stedet se på hvordan vi kan håndtere at en node er nærmere enn, og (delvis) skjuler, en annen node.

Oppgave 1

Den enkleste løsningen er kjent som "Painter's algorithm", sannsynligvis fordi det er slik malere arbeider. Den går i korthet ut på at de fjerneste delene av bildet males først, og det som er nærmere males rett og slett over.



Oversatt til vår Node3D-verden betyr det at vi må gå gjennom treet, finne de nodene som er i synsfeltet, og tegne dem i rekkefølge fra fjerneste til nærmeste. Vi skal lage en metode redrawScreen() som gjør dette.

a) Du trenger sannsynligvis en midlertidig datastruktur i metoden. Hva slags datastruktur vil du bruke? Begrunn svaret.

b) Skriv metoden void redrawScreen(Node3D root). Parameteren root peker på rotnoden i treet. Kan du si noe om hvor lang tid den bruker?

Oppgave 2

En ulempe med Painter's algorithm er at den tegner deler av skjermen flere ganger. Det kan også være vanskelig å avgjøre hvilken node som er nærmest i noen tilfeller. (Hvem er nærmest, bilen eller personen som sitter inni?)

En alternativ tilnærming er det som kalles z-buffering (eller dybde-bufring). Den går ut på at man finner ut hvilken node som skal vises i hvert punkt (piksel) på skjermen. Vanligvis gjøres dette i hardware (i skjermkortet), men vi skal se på hvordan det kan programmeres.

a) Skriv en ny versjon av void redrawScreen(Node3D root), som bruker z-buffering.

   For hver piksel må vi huske avstanden til nærmeste node, og hvilken farge pikselen skal ha. Vi kan bruke to arrayer, pixel til fargen, og zbuffer til avstanden:

   ```
   int  pixel[SCREENWIDTH][SCREENHEIGHT];
   double zbuffer[SCREENWIDTH][SCREENHEIGHT];
   ```

   Metoden må finne alle nodene i synsfeltet, og for hver piksel noden dekker, må den sjekke om noden er nærmere en nærmeste node så langt. Hvis den er nærmere må arrayene oppdateres.

   Når alle nodene er behandlet må arrayet pixel kopieres til skjermen. Dette skal du ikke skrive kode for.

b) Er denne versjonen av metoden raskere enn den første? Har tiden det tar å oppdatere skjermen betydning? Begrunn svaret.

c) Vi kan få metoden til å gå raskere hvis vi behandler de nærmeste nodene først (dvs. omvendt av Painter's algorithm). Hvorfor?

UNIVERSITETET I AGDER
SERVICEBOKS 422 4604 KRISTIANSAND
TELEFON 38 14 10 00 FAKS 38 14 10 01
ORG.NR. 970 546 200 MVA postmottak@uia.no www.uia.no

Vedlegg 1 – interface Node3D

```
import java.util.List;



/**
 * Representasjon av en 3D modell av verden.
 */
public interface Node3D
{
    /** Returnerer barna til denne noden */
    List<Node3D> getChildren();


    /** Returnerer true hvis noden eller deler av den
     * er i synsfeltet. */
    boolean isVisible();


    // Disse metodene kan brukes i Painter's algorithm

    /** Returnerer avstanden fra synspunktet ("øyet") i meter */
    double getDistance();


    /** Viser noden på skjermen. Denne metoden kopierer alle
     * pikslene direkte til skjermen. */
    void paint();


    // Disse methodene kan brukes med z-buffering

    /** Disse metodene antar at alle nodene dekker et
     * firkantet område av skjermen, og returner
     * pikselkoordinatene til sidene av firkanten. */
    int getMinX();
    int getMaxX();
    int getMinY();
    int getMaxY();

    /** Returnerer avstanden fra synspunktet til den delen av
     * denne noden som vises i et bestemt piksel.
     * Returnerer Double.POSITIVE_INFINITY hvis x,y er utenfor
     * området noden dekker. */
    double getDistance(int x, int y);


    /** Returnerer fargen som skal vises i et bestemt piksel.
     * Returnerer en tilfeldig farge hvis x,y er utenfor
     * området noden dekker. */
    int getColour(int x, int y);
}
```