Universitetet i Agder
Fakultet for Teknologi og realfag

E K S A M E N

Emnekode:	**DAT200**
Emnenavn:	**Grafisk Databehandling**
Dato:	03. desember 2009
Varighet:	0900 - 1300
Antall sider inkl. forside	8
Tillatte hjelpemidler:	Skrivesaker
Merknader:	Oppgavenes vekting er angitt i overskrift på hver oppgave.

OPPGAVE 1. (12%)

NB: Du får + 1 poeng for riktig svar, - ½ poeng for feil svar.
(Skriv besvarelsen inn på eget ark sammen med resten av besvarelsen.)
Angi om du er enig(Ja) eller uenig(Nei) i følgende utsagn:

		Ja	Nei
a)	En LCD skjerm er basert på at lyset kan oppfattes som bølger, og at flytende krystaller kan endre lysets polarisasjonsretning.		
b)	Halvtoning er en effektiv bildepresisjonsteknikk for fjerning av skulte linjer og flater.		
c)	Cohen Sutherland sin klippingsalgortime for linjer kan anvendes både i 2 og 3 dimensjoner.		
d)	To rotasjoner om samme akse i rommet (3D) er alltid kommutative.		
e)	En Hermitekurve vil vanligvis tilfredsstille C^1 kontinuitet i skjøtet mellom kurvesegmentene.		
f)	Vi har totalt fire basistransformasjoner som benevnes translasjon, speiling skalering og rotasjon.		
g)	Et matt materiale har speilende ("Specular") refleksjon over et mindre område enn et blankt materiale.		
h)	En perspektivisk projeksjon fører til at fjerne ting blir forholdsvis større enn nære ting.		
i)	Ved en Octree-representasjon av et 3D objekt er rotasjoner om hovedaksene med 90 grader lett å gjennomføre.		
j)	Ved bruk av "antialiasing" oppnås en raskere gjengivelse av linjer på en dataskjerm.		
k)	I Java kan en klassedefinisjon inneholde flere konstruktører.		
l)	Ved bruk av et MouseAdapter objekt i Java kan vi begrense oss til å skrive inn koden for de metodene som skal gjøre noe i MouseListener-"interfacet".		

OPPGAVE 2. TRANSFORMASJONER (14%)

a) De tre matrisene nedenfor representerer en transformasjon av punkter når vi bruker homogene koordinater:

$$\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(1) (2) (3)

Beskriv de transformasjonene som hver matrise(1-3) ovenfor representerer.

b) Hvorfor benyttes homogene transformasjonsmatriser?

c) Vis transformasjonssekvensen av 4x4 homogene transformasjonsmatriser (3D transformasjoner) som transformerer et linjestykke med endepunkter (1,1,1) og (5,5,5) til et linjestykke med endepunkter (0,0,0) og (1,0,0). Det er ikke nødvendig å multiplisere ut matrisene, men sekvensen skal settes opp i riktig rekkefølge.

OPPGAVE 3. SKULTE FLATER OG INTENSITETER (12%)

a) Angi kort prinsippet for z-buffer algoritmen for fjerning av flater. Angi i beskrivelsen av algoritmen også hvilken form for koherens som algoritmen benytter

b) Er z-buffer algoritmen en objekt-presisjons algoritme og/eller en bilde-presisjons algoritme? Angi også fordeler og ulemper med algoritmen.

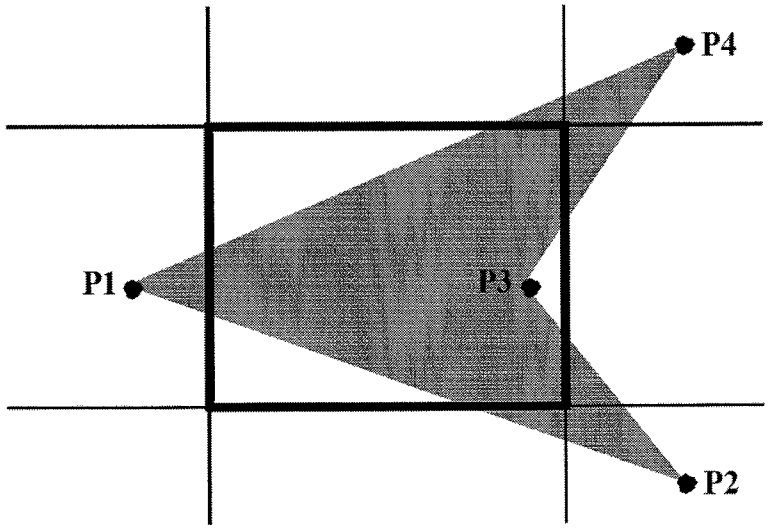
c) Angi hovedforskjellen på de to "shading"-algoritmene Gouraud og Phong. Hvorfor gjengir PHONG-"shading" speilende refleksjon("specular reflection") på en bedre måte enn Gouraud-"shading"?

OPPGAVE 4. KLIPPING, PROJEKSJONER, KURVER OG SOLIDER (28%)

Sutherland og Hodgman's polygonklippingsalgoritme benytter seg av en "splitt og hersk" strategi.

a) Forklar kort grunnprinsippet bak denne algoritmen.

Gitt følgende polygon:



b) Vis hvordan Sutherland og Hodgman's polygonklippingsalgoritme benyttes for å klippe polygonet mot angitt klippevindu. Gjør spesielt rede for hvor og hvordan skjæringspunkt mellom polygonet og klippekantene oppstår.

c) Hva forstår vi med en tre-punkts perspektivisk projeksjon?

d) Til hvilken klasse projeksjoner hører isometrisk projeksjon og hva kjennetegner denne projeksjonstypen?

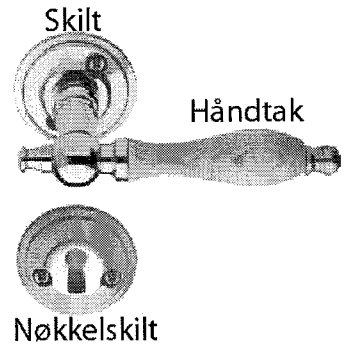
e) Regn ut projeksjonen av punktet (30,30,40) ned i x-y-planet når projeksjonspunktet er i (0,0,-40).

f) Bezier-kurver har den egenskap at en hver del av et kurvesegment ligger innenfor det konvekse omhyllingslegemet definert av de fire styrepunktene til kurvesegmentet. Hvorfor er dette tilfelle? Finnes det andre typer kubiske parametriske kurver som også har denne egenskapen? Hvordan kan denne egenskapen utnyttes i en linjeklippingsalgoritme for å gjøre den mer effektiv?

g) Naturlige kubiske splines har C^0, C^1 og C^2 kontinuitet, noe som gjør dem glattere enn hermite-kurver og kubiske bezier-kurver. Dessuten interpolerer de kontrollpunktene sine, noe som gjør dem enklere å styre enn f.eks. B-Splines. Likevel brukes disse i liten grad i f.eks. profesjonelle DAK-systemer. Hvorfor?

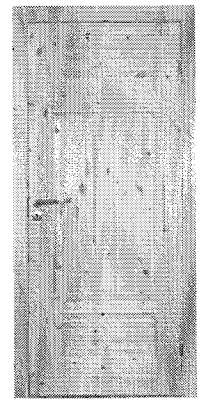
OPPGAVE 5. 3D-Studio (16%)

Vi skal modellere et dørhåndtak som er produsert i blankpolert messing med selve gripedelen i furu (Delene som inngår i et dørhåndtak er angitt i figuren til høyre) Dørhåndtaket skal senere i oppgave 6 brukes på en dør som består av selve døren (vanligvis kalles dette for et dørblad), dørhåndtaket m/skilt og nøkkelskilt, samt hengsler og selve rammen som døren står i. Døren kan rotere slik at den kan åpnes og dørhåndtaket kan også rotere.



a) Forklar kort og prinsipielt hvordan du vil gå frem i 3D-Studio for å modellere selve dørhåndtaket (Se bort fra skilt og nøkkelskilt). Ta dine egne antagelser dersom du er usikker på formen på objektet.

b) Angi hva slags materiale du ville definert i 3D Studio for den blankpolerte messingen og for selve gripedelen (furu).



OPPGAVE 6. Java (20%)

Vi skal nå laste inn (deloppgave a nedenfor) og deretter animere (deloppgave b nedenfor) døren fra oppgave 5 i Java 3D.

Døren består av tre deler:
1) Håndtak (hvor kun den delen som skal roteres inngår)
2) Dørblad, inkludert statisk del av dørhåndtak (skilt+nøkkelskilt)
3) Ramme (vi ser i denne oppgaven bort fra hengslene).

Døren skal kunne åpnes og dørhandtaket skal kunne roteres. Ta utgangspunkt i at objektene Håndtak, Blad og Ramme er modellert i 3Dstudio og lagret som filer med samme navn.

a) Skriv den rutinen i en ny klasse Dor som setter sammen selve objektet (Tilsvarende public BranchGroup createSceneGraph() i Vindmolle.java). Bruk egne antagelser vedrørende dimensjoner, akseretninger, avstander etc. Du skal **ikke** ta hensyn til at døren eller dørhåndtaket skal bevege(rotere) seg.

b) Vi vil nå ha muligheten for at dørhåndtaket og døren(dørbladet) skal kunne bevege seg. Tegn i diagramsform den BranchGroup, med nødvendige forklaringer, som rutinen (public BranchGroup createSceneGraph()) nå skal returnere.

VEDLEGG 1

```
package Vindmolle;

import java.awt.*;
import java.awt.event.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import javax.swing.*;
import com.mnstarfire.loaders3d.Inspector3DS;


class VindmollePanel extends JPanel implements ActionListener
{
  Button minus = new Button("+");
  Button pluss = new Button("-");
  Tastaturtrykk t;
  Alpha rotationAlpha;

  public VindmollePanel()
  {
    setLayout(new BorderLayout());

    GraphicsConfigTemplate3D template = new GraphicsConfigTemplate3D();
    template.setSceneAntialiasing(GraphicsConfigTemplate3D.REQUIRED);

    // Get the GraphicsConfiguration that best fits our needs.
    GraphicsConfiguration gcfg =
    GraphicsEnvironment.getLocalGraphicsEnvironment().
    getDefaultScreenDevice().getBestConfiguration(template);

    Canvas3D c = new Canvas3D(gcfg);
    add("Center", c);
    Panel p =new Panel();

    p.add(minus);
    p.add(pluss);
    add("North",p);

    pluss.addActionListener(this);
    minus.addActionListener(this);

    // Create a simple scene and attach it to the virtual
    // universe

    BranchGroup scene = createSceneGraph();
    UniverseBuilder u = new UniverseBuilder(c);
    u.addBranchGraph(scene);
  }
```

```
public BranchGroup createSceneGraph() {

    // Create the root of the branch graph
    BranchGroup objRoot = new BranchGroup();


    // Create the TransformGroup node and initialize it to the
    // identity. Enable the TRANSFORM_WRITE capability so that
    // our behavior code can modify it at run time. Add it to
    // the root of the subgraph.

    TransformGroup TGBlad1   = new TransformGroup();
    TransformGroup TGBlad2   = new TransformGroup();
    TransformGroup TGRotator = new TransformGroup();


    TGRotator.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

    objRoot.addChild(TGRotator);

    // Add the fundament and the base in the scene graph

    Inspector3DS loader = new Inspector3DS("c:/temp/Vindmolle/fundament.3ds"); // constructor
    loader.parseIt(); // process the file
    TransformGroup fundament = loader.getModel();
    objRoot.addChild(fundament);
    // get the resulting 3D model as a Transform Group with Shape3Ds as children


    // Create a new Behavior object that will perform the
    // desired operation on the specified transform and add
    // it into the scene graph.

    Transform3D zAxis = new Transform3D();
    zAxis.rotX(Math.PI/2);
    rotationAlpha = new Alpha( -1, Alpha.INCREASING_ENABLE,0, 0,
                    4000, 0, 0, 0, 0, 0);
    RotationInterpolator rotator = new RotationInterpolator(
      rotationAlpha, TGRotator, zAxis, 0.0f, (float) Math.PI*2.0f);
    BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0),200.0);
    rotator.setSchedulingBounds(bounds);
    TGRotator.addChild(rotator);

    // Add a Behavior that accepts keyboard input
    Tastaturtrykk t = new Tastaturtrykk(rotationAlpha);
    TGRotator.addChild(t);

    // Hent inn bladene
    Inspector3DS loader2 = new Inspector3DS("c:/temp/Vindmolle/blad.3ds"); // constructor
    loader2.parseIt(); // process the file
    TransformGroup blad1 = loader2.getModel();

    Inspector3DS loader3 = new Inspector3DS("c:/temp/Vindmolle/blad.3ds"); // constructor
    loader3.parseIt(); // process the file
```

```
    TransformGroup blad2 = loader3.getModel();

    Inspector3DS loader4 = new Inspector3DS("c:/temp/Vindmolle/blad.3ds"); // constructor
    loader4.parseIt(); // process the file
    TransformGroup blad3 = loader4.getModel();

    TGRotator.addChild(blad1);

    // Add the blades and rotate them
    Transform3D zAxis2 = new Transform3D();

    zAxis2.rotX(Math.PI/2);
    zAxis2.rotZ(2.0*Math.PI/3);

    TGBlad1.setTransform(zAxis2);
    TGBlad1.addChild(blad2);

    TGBlad2.setTransform(zAxis2);
    TGBlad2.addChild(blad3);

    TGBlad2.addChild(TGBlad1);
    TGRotator.addChild(TGBlad2);

    return objRoot;
  }

   public void actionPerformed(ActionEvent e )
   {
     long oldvalue= rotationAlpha.getIncreasingAlphaDuration();
     String kommando=e.getActionCommand();
     if (kommando=="+")
     {
       rotationAlpha.setIncreasingAlphaDuration(oldvalue/2);
     }
     else if (kommando=="-")
     {
       rotationAlpha.setIncreasingAlphaDuration(oldvalue*2);
     }
   } // End actionPerformed
}

class VindmolleFrame extends JFrame
{
  public VindmolleFrame()
  {
    addWindowListener(new WindowAdapter()
      {  public void windowClosing(WindowEvent e)
      { System.exit(0); }
    } );

    setSize(400, 400);
    setTitle(getClass().getName());

    Container contentPane = getContentPane();
    contentPane.add(new VindmollePanel());
  }
}

public class Vindmolle
{

  public static void main(String args[])
  {
    JFrame f = new VindmolleFrame();
    f.setSize(500,500);
    f.show();
  }
}

package Vindmolle;

import java.awt.*;
import java.awt.event.*;
import javax.media.j3d.*;
import javax.vecmath.*;

public class UniverseBuilder extends Object {

    // User-specified canvas

    Canvas3D canvas;

    // Scene graph elements to which the user may want access

    VirtualUniverse         universe;
    Locale                  locale;
    TransformGroup          vpTrans;
    View                    view;

    public UniverseBuilder(Canvas3D c) {
      this.canvas = c;


      // Establish a virtual universe that has a single
      // hi-res Locale

      universe = new VirtualUniverse();
      locale = new Locale(universe);


      // Create a PhysicalBody and PhysicalEnvironment object

      PhysicalBody body = new PhysicalBody();
      PhysicalEnvironment environment =
                              new PhysicalEnvironment();


      // Create a View and attach the Canvas3D and the physical
      // body and environment to the view.
```

```
        view = new View();
        view.addCanvas3D(c);
        view.setPhysicalBody(body);
        view.setPhysicalEnvironment(environment);
        view.setBackClipDistance(500);

        // Create a BranchGroup node for the view platform

        BranchGroup vpRoot = new BranchGroup();

        // Create a ViewPlatform object, and its associated
        // TransformGroup object, and attach it to the root of the
        // subgraph. Attach the view to the view platform.

        Transform3D t = new Transform3D();
        Transform3D s = new Transform3D();

        t.rotY(Math.PI/4);
        s.set(new Vector3f(0.0f, 0.0f, 200.0f));
        t.mul(s);
        s.rotX(-Math.PI/32);
        t.mul(s);

        ViewPlatform vp = new ViewPlatform();
        vpTrans = new TransformGroup(t);
        vpTrans.addChild(vp);
        vpRoot.addChild(vpTrans);
        view.attachViewPlatform(vp);

        // Attach the branch graph to the universe, via the
        // Locale. The scene graph is now live!

       locale.addBranchGraph(vpRoot);
    }

    public void addBranchGraph(BranchGroup bg) {
       locale.addBranchGraph(bg);
    }
  }

package Vindmolle;

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.media.j3d.*;
import javax.vecmath.*;


public class Tastaturtrykk extends Behavior
{
 Alpha alpha;
 WakeupCriterion[] keyEvents;
 WakeupOr keyCriterion;

 public Tastaturtrykk(Alpha alpha)
 {
  this.alpha=alpha;
  BoundingSphere bounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 200.0);
  this.setSchedulingBounds(bounds);
 }

 public void initialize()
 {
  keyEvents = new WakeupCriterion[1];
  keyEvents[0]=new WakeupOnAWTEvent(KeyEvent.KEY_PRESSED);
  keyCriterion = new WakeupOr(keyEvents);
  wakeupOn (keyCriterion);
 }

 public void processStimulus (Enumeration criteria)
 {
  WakeupCriterion wakeup;
  AWTEvent[] event;
  int id;
  char k;

  while (criteria.hasMoreElements()) {
   wakeup = (WakeupCriterion) criteria.nextElement();
   if (wakeup instanceof WakeupOnAWTEvent) {
    event = ((WakeupOnAWTEvent)wakeup).getAWTEvent();
    for (int i=0; i<event.length; i++) {
     id = event[i].getID();
     if (id == KeyEvent.KEY_PRESSED) {
      k = ((KeyEvent)event[i]).getKeyChar();
      long oldvalue= alpha.getIncreasingAlphaDuration();
      if (k=='+')
       {
        alpha.setIncreasingAlphaDuration(oldvalue/2);
        System.out.println("+");
       }
      else if (k=='-')
       {
        alpha.setIncreasingAlphaDuration(oldvalue*2);
        System.out.println("-");
       }
     } // End if
    } // End for
   } // End if
  } // End while
  wakeupOn (keyCriterion);
 } // End processStimulus
} // End class Tastaturtrykk
```