Universitetet i Agder
Fakultet for teknologi og realfag

EKSAMEN

Emnekode:	**DAT200**
Emnenavn:	**Grafisk Databehandling**
Dato:	04. Desember 2013
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 (Sant) eller uenig (Usant) i følgende utsagn:

		Sant	Usant
a)	Grafisk Databehandling har ingen anvendelser innen medisin.		
b)	Høyrehåndsregelen kan blant annet benyttes til å fastslå positiv rotasjonsretning i forhold til en hovedakse.		
c)	Midtpunktsalgoritmen for linjer utnytter første ordens differens for å effektivisere utregningene av nye piksler.		
d)	Vektet antialiasing arbeider hurtigere enn uvektet antialiasing.		
e)	Dersom høyde/bredde forholdet er ulikt i overføringen fra et vindu til en skjermport («ViewPort») risikerer vi at sirkler blir ellipser når vi ser modellen på en dataskjerm.		
f)	Et større vindu vil medføre at detaljer trer klarere frem i en skjermport med fast størrelse.		
g)	Sutherland-Hodgman sin polygonklippingsalgoritme kan ikke behandle konvekse klippeområder.		
h)	CIE-Chromaticity-diagrammet kan ikke gjengi samtlige intensitetsuavhengige farger.		
i)	En serie av rotasjoner i planet kan multipliseres sammen og vil alltid gi det samme sluttresultatet uavhengig av rekkefølgen (De er kommutative)		
j)	Navnet «spline» kommer av de tynne stripene av tre, metall eller plastikk som ble anvendt blant annet når man tegnet skipsskrog på papir.		
k)	Nær-og fjernplan har ved rendering kun den hensikt at renderingshastigheten øker.		
l)	NURBS er en forkortelse for Northern-Uniform Relational Bottom-Surface.		

OPPGAVE 2. TRANSFORMASJONER (14%)

a) De tre matrisene nedenfor representerer en transformasjon av punkter når vi bruker homogene koordinater. Beskriv de transformasjonene som hver matrise representerer.

$$\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 6 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

(1) (2) (3)

b) Hvilke tre basistransformasjoner har vi?

c) Gitt en rotasjonsmatrise i rommet. Hvordan vil du ut i fra matrisen kunne se hvilken akse det roteres rundt, gitt at det kun er rotasjon rundt en av de tre hovedaksene?

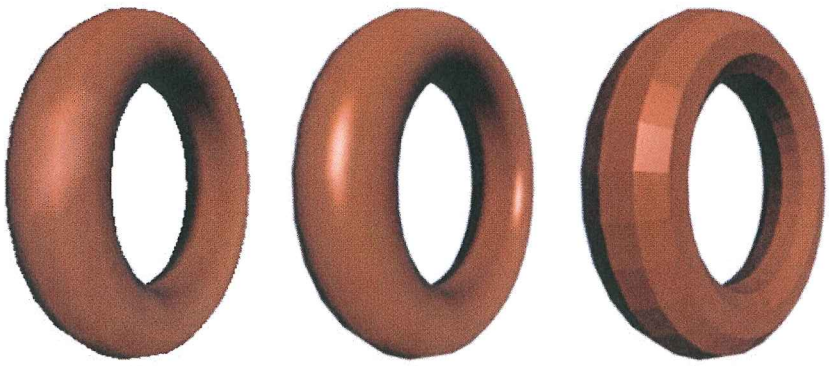
OPPGAVE 3. FARGER, SKULTE FLATER OG INTENSITETER (16%)

a) Tenk deg at du har en standard blekkskriver basert på fargene Cyan, Magenta, Yellow og BlacK (CMYK) der de to blekkpatronene cyan og magenta har byttet plass. Når man forsøker å skrive ut følgende farger hva vil du faktisk se på papiret (Tips: Husk formelen for overføring mellom CMY og RGB):

1. Rødt
2. Grønt
3. Blått
4. Cyan
5. Magenta
6. Yellow
7. Black
8. White

b) Forklar virkemåten til z-buffer algoritmen. Har metoden noen ulemper?

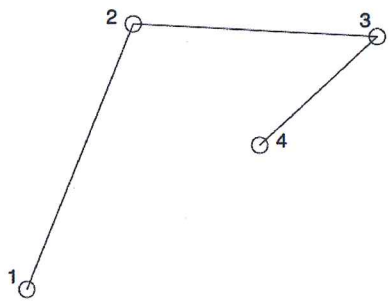
c) I figuren nedfor er gjengitt skyggelegging med de tre metodene konstant intensitet, «Gouraud Shading» og «Phong Shading». Indiker hvilken metode som hører sammen med hvilken smultring når smultringene er nummerert fra 1 til 3.



1. 2. 3.

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

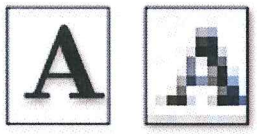
a) Tegningen til høyre viser kontroll-polygonet til et Bezier kurvesegment. Tegn kontrollpolygonet til et kurve-segment nummer to som starter i knutepunkt nummer 4 og som har G^1 kontinuitet i skjøten. Tegn også de to resulterende Bezier kurvesegmentene, med tilhørende kontrollpolygon, inn i den endelige figuren du tegner.



b) Forklar forskjellen på C^1 og G^1 kontinuitet.

c) Hermitiske kurvesegment er definert ved geometri-vektoren $[p_k\ p_{k+1}\ Dp_k\ Dp_{k+1}]^{-1}$
Forklar kort hvordan de fire komponentene som inngår i denne vektoren påvirker kurven.

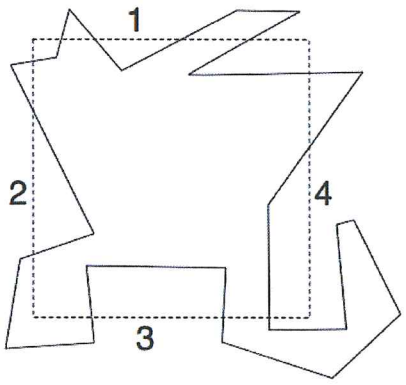
d) Figuren til høyre viser to 12 punkts versjoner av karakteren A som er kraftig forstørret. Forklar sannsynlig årsak til den store forskjellen på de to utgavene.



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

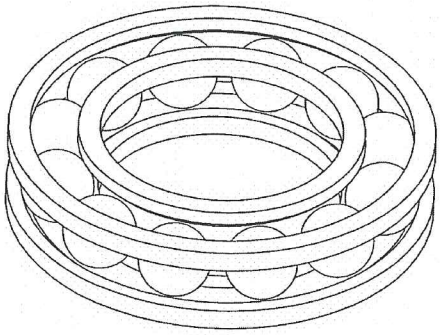
f) Regn ut den perspektiviske projeksjonen av punktet (20,40,50) ned i x-y-planet når projeksjonspunktet er i (0,0,-50).

g) I figuren til høyre er det gjengitt et polygon og et klipperektangel med 4 kanter nummerert fra 1 til 4. Nå skal vi anvende den klippingsalgoritmen for polygoner som vi har behandlet i pensum og klippe bort det overskytende i 4 operasjoner. Vis ved tegning de 4 trinnene i algoritmen og sluttresultatet etter hvert trinn.

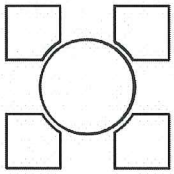


OPPGAVE 5. 3D-Studio (10%)

a) Forklar hvordan du vil gå frem i 3D-Studio for å generere kulelageret vist i figuren under. Mål er ikke vesentlig. Kun prinsippene kreves, men det vil bli lagt vekt på at en effektiv og oversiktlig fremgangsmåte er valgt.



Snitt av kulelager:



OPPGAVE 6. Java (20%)

a) Når vi skal addere en lytter til en klasse har vi tre alternative måter å beskrive hvem som skal ta seg av hendelsene. Eksempel:

1. addMouseListener(this);
2. addMouseListener(new MouseAdapter().....);
3. addMouseListener(musObjekt);
   //musObjekt er et objekt tilhørende en klasse som implementerer MouseListener interfacet

Forklar hva som skjer i de tre alternativene og angi fordeler/ulemper for hvert alternativ.

b) Vi skal modellere en forenklet bil bestående av et karosseri m/førerhus og 4 hjul. Samtlige 4 hjul kan rotere om sin akse. Forhjulene skal i tillegg kunne svinges om en vertikal akse. Ta utgangspunkt i at de to objektene karosseri (inkluderer førerhus) og hjul er modellert i 3Dstudio og lagret som filer med navnene karosseri og hjul.

1) Skriv den rutinen i en ny klasse Bil som setter sammen bilen ut fra de predefinerte objektene (Tilsvarende public BranchGroup createSceneGraph() i Vindmolle.java). Bruk egne antagelser vedrørende dimensjoner, akseretninger, avstander etc. Tenk spesielt nøye over hvor du vil plassere lokalt koordinatsystem for bilen **(Du behøver ikke her å tenke på at bilen skal kunne beveges, hjulene roteres ...)**. Øvrig innhold i klassen og øvrige klasser nødvendig for å vise bilen skrives ikke.

2) Vi vil nå ha muligheten for å bevege bilen slik at hjulene kan roteres og bilen kan 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
```