Universitetet i Agder
Fakultet for Teknologi og realfag

EKSAMEN

Emnekode:	**DAT200**
Emnenavn:	**Grafisk Databehandling**
Dato:	10. desember 2007
Varighet:	0900 - 1300
Antall sider inkl. forside	8
Tillatte hjelpemidler:	Skrivesaker
Merknader:	Oppgavenes vektning 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 skjerm basert på "Random Scan" teknologi vil kunne gjengi linjer og sirkler bedre enn en "Raster Scan" skjerm	☐	☐
b)	En uniform skalering og en translasjon i planet (2D) er alltid kommutative (Det vil si at rekkefølgen er uvesentlig)	☐	☐
c)	Sutherland Hodgman sin klippingsalgoritme kan kun anvendes for kvadratiske klippeområder.	☐	☐
d)	LCD-skjermer egner seg bedre som dataskjermer enn plasmaskjermer.	☐	☐
e)	Ved bruk av en fargeoppslagstabell ("Look Up Table") kan vi få vist langt flere farger samtidig på en datamaskin.	☐	☐
f)	Perspektiviske transformasjoner etterligner virkemåten til et menneskelig øye.	☐	☐
g)	Menneskets oppfatning av intensitetssprang følger en lineær skala.	☐	☐
h)	En isometrisk projeksjon tilhører klassen perspektiviske projeksjoner.	☐	☐
i)	Ved "bump-mapping" så vil ikke et objekts silhuett påvirkes.	☐	☐
j)	En Bezier-spline oppfyller det vi på engelsk kaller "convex hull property"	☐	☐
k)	En B-spline bestående av mange kurvesegmenter må tegnes på nytt, i sin helhet, dersom et kontrollpunkt endres.	☐	☐
l)	Z-buffer er den mest brukte algoritmen for fjerning av skjulte flater.	☐	☐

OPPGAVE 2. TRANSFORMASJONER (16%)

a) De fire matrisene nedenfor representerer en transformasjon av punkter når vi bruker homogene koordinater i 2D:

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -2 & 0 & 4 \\ 0 & 3 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

(1) (2) (3) (4)

Beskriv de transformasjonene som hver matrise(1-4) ovenfor representerer. (En transformasjon kan være sammensatt av flere enkle transformasjoner)

b) Hvilke tre basistransformasjoner har vi?

c) Vi har et rektangel i planet med hjørnepunktene plassert slik: (0,0), (2,1), 0,5) og (-2,4). Vi ønsker å transformere dette rektangelet slik at det blir gjort om til et kvadrat med hjørnepunktente plassert slik: (0,0), (1,0), (1,1) og (0,1). Sett opp i riktig rekkefølge de transformasjonene som transformerer rektangelet til det angitte kvadratet. Du trenger ikke å multiplisere sammen matrisene. Dersom du ønsker kan du benytte vinkelen $\theta=\arctan(1/2)$ i løsningen.

OPPGAVE 3. INTENSITET, FARGER OG KURVER (16%)

a) Forklar forskjellen på diffus og spekulær refleksjon.

b) Forklar forskjellen på de to metodene for interpolert skyggelegging Gouraud og Phong.

c) Hvorfor bruker vi en annen fargemodell (CMY) når bilder skal ut på papir enn når de skal ut på skjerm (RGB)? Sett opp ligningen for overføring fra den ene modellen til den andre.

d) Hermitiske kurvesegment er definert av den hermitiske geometri-vektoren $[p_k\ p_{k+1}\ Dp_k\ Dp_{k+1}]^{-1}$

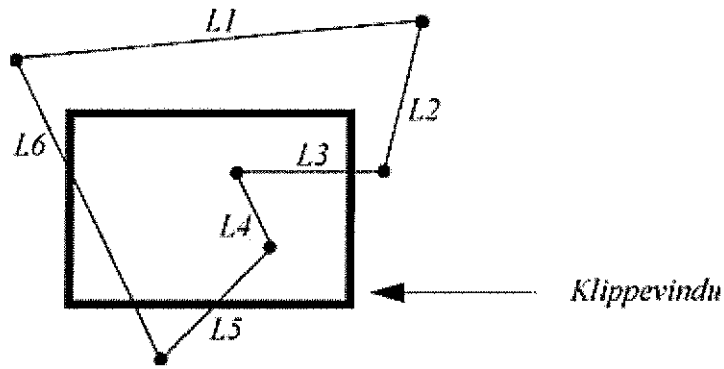
Forklar kort hvordan de fire komponentene som inngår i denne vektoren påvirker kurven. Hvilken betingelse må tangentvektorene i et knutepunkt for to nærliggende kurvesegment oppfylle for at kurven skal ha C^1 kontinuitet i dette punktet?

OPPGAVE 4. KLIPPING OG PROJEKSJONER (22%)

For effektiv klipping av linjer benytter *Cohen-Sutherland* algoritmen bitkoder for hvert punkt i forhold til et klippevindu i 2D og et klippevolum i 3D

a) Forklar hvordan *Cohen-Sutherland* algoritmen fungerer i 2D.

b) Det ikke-fylte polygonet nedenfor kan klippes mot angitt klippvindu ved hjelp av *Cohen-Sutherland* algoritmen ved å klippe hvert enkelt linjesegment for seg



Grupper de seks linjesegmentene i en av følgende tre grupper:

- Trivielt akseptert
- Trivielt forkastet
- Andre

c) Forklar hvordan denne algoritmen kan utvides til å fungere i 3D.

d) Hva forstår vi med en perspektivisk projeksjon som har tre forsvinningspunkt?

e) Regn ut den perspektiviske projeksjonen av punktet (30,30,50) ned i x-y-planet når projeksjonspunktet er i (0,0,-50).

OPPGAVE 5. 3D-Studio (14%)

Vi skal modellere deler av en lyskaster designet for teater og TV studio (Se figur til høyre). Selve kasteren kan rotere om to akser. En horisontal rotasjon (aksen gjennom lysinnfatningen) og en vertikal rotasjon (besørget av innfestningen).

a) Forklar kort og prinsipielt hvordan du vil gå frem i 3D-Studio for å modellere Innfestningen til lyskasteren (Delen i midten som lyset er montert inne i). Ta dine egne antagelser dersom du er usikker på formen på objektene.

b) Ved opprettelsen av objekter som sylindere, kuler, rør etc. i 3D Studio kan vi angi antall plane polygoner som objektene skal tilnærmes med. Kan du tenke deg fordeler og ulemper ved å angi henholdsvis mange/få flater i tilnærmingen.

OPPGAVE 6. Java (20%)

Vi skal animere lyskasteren fra oppgave 5 i Java 3D. Den består av tre deler en lyskaster en innfestning og et fundament. Lyskasteren skal kunne rotere om to akser som forklart i oppgave 5. Ta utgangspunkt i at objektene Fundament, Innfestning og Lyskaster er modellert i 3Dstudio og lagret som filer med samme navn.

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

b) Vi vil nå ha muligheten for å rotere lyskasteren om to akser. 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
```