University of Agder
Faculty of Technology and Science

E X A M

Code:	**DAT200**
Name:	**Computer graphics**
Date:	03. desember 2009
Duration:	0900 - 1300
Number of pages	8
Aids:	Stationary
Comments:	The weighting is specified in the header of each task

ENGLISH

TASK 1. (12%)

NB: You get + 1 point for correct answer, - ½ point for wrong answers.
(Write the answer on a separate sheet along with the rest of the paper.)
Please indicate whether you agree (Yes) or disagree (No) in the following statements:

		Yes	No
a)	An LCD screen is based on the property that light is waves, and that liquid crystals can change the light polarization.		
b)	Half toning is a powerful image-precision technique for the removal of hidden lines and surfaces		
c)	Cohen Sutherland's algorithm for clipping lines can be used in both 2 and 3 dimensions.		
d)	Two rotations about the same axis in space (3D) are always commutative.		
e)	A Hermite curve will usually satisfy C^1 continuity between the curve segments.		
f)	We have a total of four basic transformations that are called translation, mirroring, scaling and rotation.		
g)	A matte material has specular reflection on a smaller area than a shiny material.		
h)	With perspective projections objects farther away from the viewer looks bigger than objects close to the viewer.		
i)	In an Octree representation of a 3D object, rotations about the main axes with an angle of 90 degrees are easy to implement.		
j)	Using antialiasing we can achieve faster rendering of lines on a computer screen.		
k)	In Java, a class definition can contain multiple constructors.		
l)	Using a Mouse Adapter object in Java, we can limit ourselves to write the code for the methods that do something in the Mouse Listener interface.		

TASKS 2. TRANSFORMATIONS (14%)

a) The three matrixes below represent a transformation of points when we use homogeneous coordinates:

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

(1) (2) (3)

Describe the transformation that each matrix (1-3) above represents.

b) Why do we use homogenous transformation matrixes?

c) Show the transformation sequence of the 4x4 homogeneous transformation matrixes (3D transformations) that transforms a line with end points (1,1,1) and (5,5,5) to a line with end points (0,0,0) and (1,0,0). It is not necessary to multiply the matrixes, but the sequence should be set up in the correct order.

TASK 3. HIDDEN SURFACES AND INTENSITIES (12%)

a) Write a short principle description of the z-buffer algorithm for removing surfaces. Enter in the description of the algorithm also what kind of coherence that the algorithm uses.

b) Is the z-buffer algorithm an object-precision algorithm and/or an image-precision algorithm? Enter the advantages and disadvantages of the algorithm.

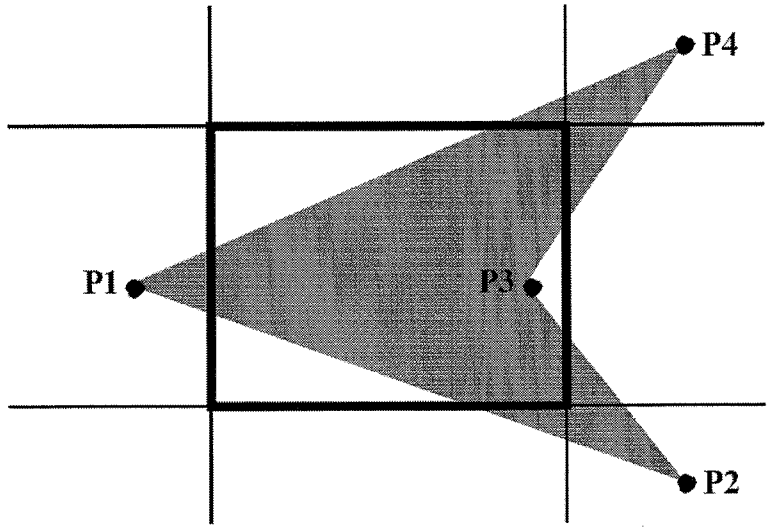
c) Enter the main difference between the two shading algorithms Gouraud and Phong. Why does Phong render specular reflection better than Gouraud?

TASK 4. CLIPPING, PROJECTIONS, CURVES AND SOLIDS (28%)

Sutherland and Hodgman's polygon clipping algorithm uses a "divide and conquer" strategy.

a) Explain briefly the basic principle behind this algorithm.

Given the following polygon:



b) Explain how the Sutherland and Hodgman's polygon-clipping algorithm is used to clip the polygon against the specified clipping window. Make particular account of how and where the intersection between the polygon and the cutting edges occurs.

c) What do we understand with a three-point perspective projection?

d) To which class of projections does isometric projection belong, and what characterizes this type of projection?

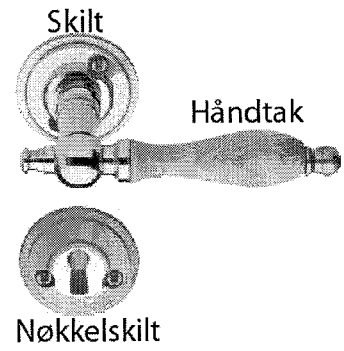
e) Calculate the projection of the point (30,30,40) into the xy-plane when the projection point is in (0,0, -40).

f) Bezier curves have the property that every part of a curve segment lies within the convex hull defined by the four control points of the curve segment. Why is this the case? Are there other types of cubic parametric curves that also have this property? How can this property be utilized in a line-clipping algorithm to make it more effective?

g) Natural cubic splines have C^0, C^1 and C^2 continuity, which make them smoother than Hermite curves and cubic Bezier curves. The spline also interpolates the control points, which makes them easier to control than B-Splines. Yet these are used to a limited extent in for example professional CAD systems. Why?

TASK 5. 3D-STUDIO (16%)

We want to model a door handle that is produced in polished brass with the grip section of pine (The parts of a door handle is specified in the picture to the right). The door handle is later to be used in Task 6 on a door, which consists of the door body (usually just called the door), the door handle ("Håndtak" in Norwegian) with plate and key plate ("Skilt" and "Nøkkelskilt" in Norwegian), as well as hinges and the frame that the door is in. The door can rotate so that it can be opened and the door handle can also rotate.



a) Explain briefly the principle and how to go forward in 3D Studio to model the door handle (Please disregard the plate and the key plate). Take your own assumptions if you are unsure of the shape of the object.

b) Specify the type of material you would choose in 3D Studio for the polished brass and the grip section (pine).

TASK 6. JAVA (20%)

We will now load (part a below) and then animate (part b below) the door from task 5 in Java3D.

The door consists of three parts:
1) Door handle (where only the part to rotate is part of the door handle)
2) Door body, including the static part of the door handle (the plate + key plate))
3) Frame (In this task we disregard the hinges).

The door can be opened and the door handle can be rotated. Assume that the objects Handle, Door and Frame are modelled in 3Dstudio and stored as files with the same name.

a) Write the routine in a new class Door who put together the actual object (corresponding to public Branch Group createSceneGraph () in Vindmolle.java). Use your own assumptions regarding the dimensions, axis directions, distances, etc. You shall **not** take into account that the door or door handle can rotate.

b) We will now have the possibility to open the door and rotate the door handle. Draw in a scene graph the Branch Group, with appropriate explanations, that the routine (Public Branch Group createSceneGraph ()) now returns.

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
```