University of Agder
Faculty of Engineering and Science

EXAMINATION

Course Code:	**DAT200**
Course Name:	**Computer Graphics**
Date:	2 December 2011
Duration:	0900 – 1300
Number of pages including cover page:	9
Remedies:	Only stationary
Notes:	The weight of each assignment is specified in the header

ASSIGNMENT 1. (12%)

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

		Yes	No
a)	LCD screens use the property that the phosphor can emit light in different colours.		
b)	When extending the Cohen Sutherland line clipping algorithm's from 2 to 3 dimensions, we must add 2 bits for each endpoint code.		
c)	Sutherland Hodgman's polygon clipping algorithm can only be used with convex cutting areas.		
d)	Two rotations in space (3D) are always commutative.		
e)	The benefit of C^1 continuity compared to G^1 continuity is that the curve becomes smoother and has better features in regard to air- and water- resistance.		
f)	A small window, lead to a better view of the small details, in a "viewport" (video port) with a fixed size.		
g)	A glossy material has specular ("Specular") reflection on a larger area than a matte material.		
h)	In an isometric projection the projection rays are normal to the projection plane.		
i)	When using a BSP tree (Binary Space Partitioning), one can carry out the removal of hidden surfaces with an intelligent traversal of the tree.		
j)	A Bezier spline does not meet what we call the convex hole property		
k)	Fractals can be used to generate trees, clouds, coastlines and mountain ranges in a better way than normal Euclidean geometry.		
l)	The Z-buffer algorithm (Depth Buffer Algorithm) is an object-precision algorithm for removing hidden surfaces.		

ASSIGNMENT 2. TRANSFORMATIONS (18%)

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

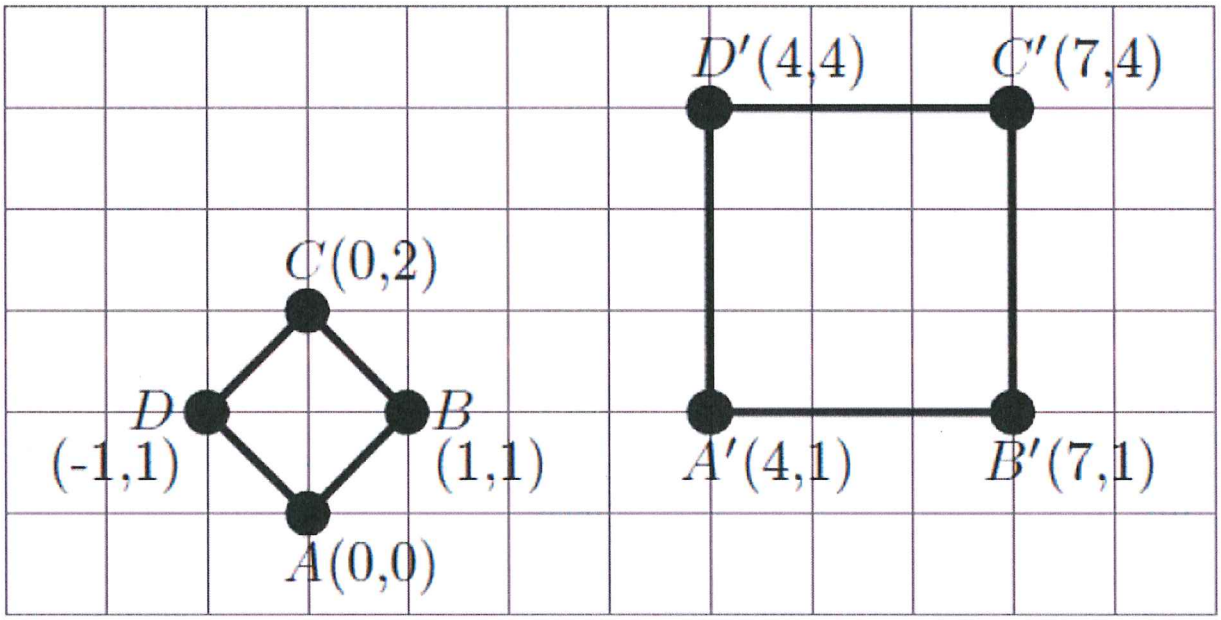
$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(1) (2) (3)

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

b) Homogeneous coordinates are often used to represent transformations in 3D. Specify the procedure to convert a position from the coordinate system, which includes the homogeneous coordinate (4 coordinate values (x_h, y_h, z_h, h), to the ordinary Cartesian coordinate system (3 coordinate values (x, y, z). How would you go the opposite way from the Cartesian coordinate system to the homogeneous coordinate system?

c) Set up the transformations, using homogeneous coordinates, which transforms the square ABCD to rectangle A'B'C'D '. You do not have multiplying together the matrices.



d) Show what happens if this transformation applied to the square A'B'C'D '

ASSIGNMENT 3. COLOURS, HIDDEN SURFACES AND INTENSITY (12%)

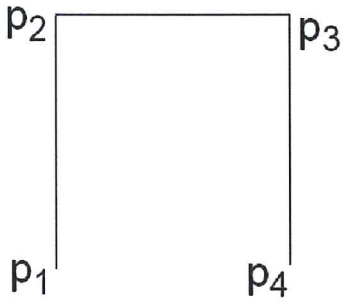
a) What is a "CIE chromaticity" diagram and what can it be used for?

b) How can the first step in a "ray tracer" be used for the removal of hidden lines and surfaces? Is this algorithm an image- or object-precision algorithm?

c) Why is Phong shading rendering Specular reflection in a better way than Gouraud shading?

ASSIGNMENT 4. SPLINES AND SOLIDS (20%)

Given the following open control polygon (4 points p1, p2, p3 and p4):



a) Draw the polygon and the points and sketch a cubic Bezier curve in the polygon.

b) Draw the polygon and the points and sketch a cubic uniform B-spline curve in the polygon.

c) What are the main differences between the cubic Bezier curves and uniform cubic B-spline curves?

d) How can we ensure that we get a continuous (G^1 continuity) Bezier curve when we connect two Bezier curve segments?

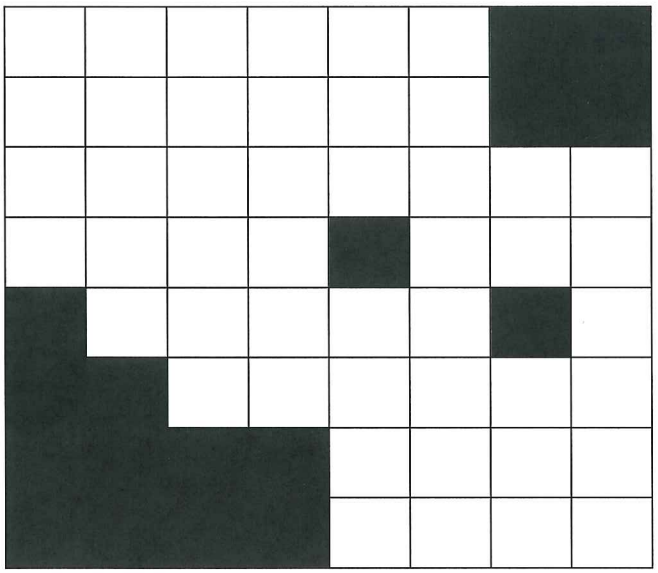
e) Draw a "Quad Tree" representation of the figure below, which is part of a picture (filled areas are indicated with black, empty spaces with white).

Use the following symbols to create the tree:

Δ Node with children

Node representing one or more pixels that are of

Node representing one or more pixels that are on



ASSIGNMENT 5. 3D-Studio (12%)

We will model the parts of a combined oven and dishwasher.

a) Explain briefly the principle and how to go forward in 3D Studio to model the handle (of the two doors) and the 7 switches in steel. Use your own assumptions if you are unsure of the shape of the object.

b) Specify the type of material you would define in 3D Studio for the handles, the ceramic cooking plate and the steel plate in front of the dishwasher.

ASSIGNMENT 6. Java (26%)

We will now load and animate the stove / dishwasher from assignment 5 in Java 3D. It consists of 4 parts: **Cabinet** (The machine), **Door1** (The door of the oven), **Door2** (The door to the dishwasher), **Switch** (A total number of 7 switches that control the oven and hotplates.

a) We will now be able to use the combined stove / dishwasher. Doors and switches should be able to rotate. Draw the scene graph for the branchgroup that is necessary to create a functional "live" stove / dishwasher. Bring your own assumptions about the axes and sizes. Be sure to specify which axis you are rotating around or move along in the graph.

b) What does it mean that Java is platform independent, and what is the big advantage of this? Can you see any disadvantages with this?

c) What does it mean when we write implements KeyListener in a program, and what impact does it have? Why are such structures used in Java?

d) What is the task of objects made of the class Alpha in a Java3D program?

e) What does it mean when we write extends JPanel in a program, and what impact does it have? Why are such structures used in Java?

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
```