Universitetet i Agder
Fakultet for teknologi og realfag

EKSAMEN

Emnekode:	**DAT200**
Emnenavn:	**Grafisk Databehandling**
Dato:	15. desember 2010
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)	"Raster Scan" skjermen er den mest utbredte skjermtypen i dag blant annet grunnet sin gode evne til å tegne fylte områder.		
b)	At to transformasjoner er kommutative vil si at sluttresultatet er uavhengig av hvilken transformasjon som utføres først.		
c)	Sutherland-Hodgman sin polygonklippingsalgoritme kan kun anvendes for konkave klippeområder.		
d)	To rotasjoner i rommet (3D) er alltid kommutative.		
e)	Fordelen med C^1 kontinuitet i forhold til G^1 kontinuitet er at kurven blir glattere og får bedre egenskaper i forhold til f. eks luft- og vannmotstand.		
f)	Et mindre vindu vil medføre at detaljer trer klarere frem i en "viewport" (skjermport) med fast størrelse.		
g)	Et matt materiale har speilende ("Specular") refleksjon over et mindre område enn et blankt materiale.		
h)	En isometrisk projeksjon har projeksjonsstråler som står normalt på projeksjonsplanet.		
i)	Ved bruk av "ray-casting" kan man gjennomføre fjerning av skjulte linjer og flater for en CSG-modell.		
j)	En kubisk Bezier-spline oppfyller det vi på engelsk kaller "convex hull property"		
k)	Ved bruk av fraktaler kan trær, skyer, kystlinjer og fjellkjeder gjengis på en mer naturtro måte enn med euklidsk geometri.		
l)	En applikasjonsmodell er et grafisk bilde av renderingsprosessen der man tar hensyn til "specular"(speilende) refleksjon.		

OPPGAVE 2. TRANSFORMASJONER (18%)

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

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(1) (2) (3)

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

b) Sett opp transformasjonen for å skalere med 2 i x-retning og med 3 i y-retning med fastholdingspunkt("pivot pont") i (4,2). Du trenger ikke å regne sammen enkeltmatrisene.

c) Sett opp transformasjonen for å rotere -30 grader om punktet (-2,3). Du trenger ikke å regne sammen enkeltmatrisene.

d) Forklar hvordan man kan gå frem for å overføre definisjonen til et objekt (punktenes koordinatverdier) fra et koordinatsystem til et annet. Kan du angi et eksempel der en slik overføring (transformasjon) blir benyttet?

OPPGAVE 3. FARGER, SKULTE FLATER OG INTENSITETER (12%)

a) Hva er komplementærfarger?

b) Angi kort prinsippet for "Painter's" (Malerens) algoritme for fjerning av skjulte linjer og flater. Er dette en objekt-presisjons algoritme og/eller en bilde-presisjons algoritme

c) Hvorfor gjengir PHONG-"shading" speilende refleksjon("specular reflection") på en bedre måte enn Gouraud-"shading"?

OPPGAVE 4. SPLINES, KLIPPING, PROJEKSJONER, OG SOLIDER (20%)

a) Forklar kort prinsippet for Octrees.

To endepunkt A(13,11) og B(4,15) begrenser et linjesegment AB.

b) Angi linjen AB mellom de to punktene på parametrisk form.

c) Anta at AB er en kant til et polygon. Finn skjæringen med scanlinjen y=13.

d) Forklar hvordan Cohen Sutherland sin linjeklippingsalgoritme tildeler koder til de to endepunktene til en linje.

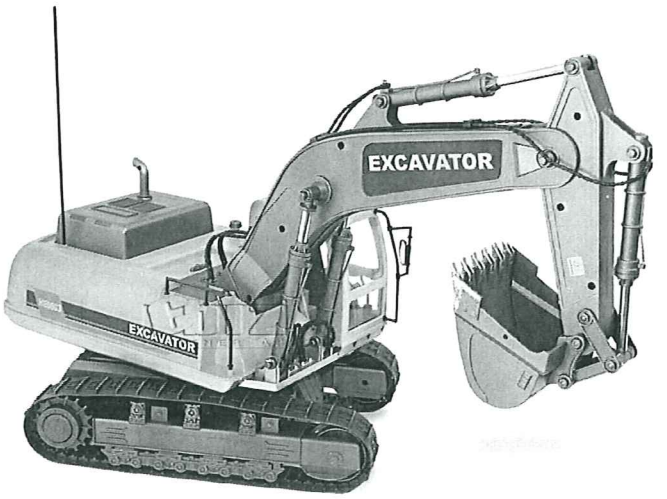
e) Angi og begrunn når du vil bruke henholdsvis perspektivisk og parallell projeksjon.

OPPGAVE 5. 3D-Studio (12%)

Vi skal modellere en av delene til en gravemaskin (Se figur til høyre).

a) Forklar kort og prinsipielt hvordan du vil gå frem i 3D-Studio for å modellere et av beltene til gravemaskinen (Kun belte ikke hjul/drivhjul). Ta dine egne antagelser dersom du er usikker på formen på objektet.

b) Angi hva slags materiale du ville definert i 3D Studio for beltet og grabben.



OPPGAVE 6. Java (26%)

Vi skal nå laste inn og animere gravemaskinen fra oppgave 5 i Java 3D. Den består av fem deler: Belter og fundament(som er å betrakte som ett objekt som skal stå i ro), gravemaskinhus (som skal kunne rotere), to armer (Arm 1 og Arm2 som skal kunne rotere (se bort fra hydraulikk og sylindre som skaper bevegelsene) og en grabb som skal kunne rotere (se også her bort fra sylindere og hydraulikk som skaper bevegelsen).

a) Vi vil nå ha muligheten for å bruke gravemaskinen og armer, grabb og gravemaskinhus skal kunne bevege seg (Belter og fundament skal være i ro). Tegn i diagramsform den BranchGroup som er nødvendig for å skape en funksjonell levende gravemaskin. Ta dine egne antagelser om akser og størrelser. Sørg for å angi hvilken akse det roteres om eller forflyttes langs i diagrammet.

b) Hva forstår vi med at Java er plattformuavhengig og hva er den store fordelen med dette? Kan du se noen ulemper med dette?

c) Hva betyr det når vi skriver *implements Mouselistener* i et program og hvilke konsekvenser har det? Hvorfor brukes slike konstruksjoner i språket Java?

d) Hva er XOR-modus og hvordan kan det brukes i programmering?

e) Hva betyr det når vi skriver *extends JFrame* i et program og hvilke konsekvenser har det? Hvorfor brukes slike konstruksjoner i språket 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
```