RIMBALZO CUBI PROSPETTIVA

```
/**
 * Aggiungi qui una descrizione della classe Grafica2
 * Rimbalzo di figure (Cubi) dentro l'immagine del desktop
 * @author (Paolo P.)
 * @version (1.1.2016)
 */
import java.awt.*;
import java.awt.Robot;
import javax.swing.*;

/** Rimbalzo cubi(Animation) via custom thread */
@SuppressWarnings("serial")
public class Grafica2 extends JFrame {
   // Define named-constants
   private static final int CANVAS_WIDTH = 1440;
   private static final int CANVAS_HEIGHT = 900;
   private static final int UPDATE_INTERVAL = 50; // milliseconds
   private DrawCanvas canvas;  // the drawing canvas (extends JPanel)
   // Attributes of moving object
   private Color c1;
   private int maxz=500,maxx=1440,maxy=900,jmax=150,i,j,k,p,x0,x1,x2,x3,y0,y1,y2,y3;
   private int scambio,npoints = 5,latoa[]={1,5,5,8,7,6},latob[]={2,6,1,4,3,2},latoc[]={3,7,4,3,2,1},latod[]={4,8,8,7,6,5},ypoints[] =
{1,2,3,4,1},xpoints[] = {1,2,3,4,1};
   private double cx[]={0,-30,30,30,-30,-30,30,30,-30},cy[]={0,-30,-30,30,30,-30,-30,30,30},cz[]={0,-30,-30,-30,-
30,30,30,30,30},xip,yip,zip,xkp,ykp,zkp,distanzac1c2,m1,m2,vcx,vcy,vcz,v01modulo,v02modulo,cosalfa1,cosalfa2,scala=0,ox,oy,oz,scambiof
;
   private double[] raggiof=new double[jmax],tx=new double[jmax],ty=new double[jmax],tz=new double[jmax],costx=new
double[jmax],costy=new double[jmax],costz=new double[jmax],txr=new double[jmax],tyr=new double[jmax],tzr=new double[jmax],teta=new
double[jmax],kteta=new double[jmax];
   private double vcxr,vcyr,vczr,v01modulor,v02modulor,cosalfa1r,cosalfa2r;
   private double[][] x = new double[jmax][9],y = new double[jmax][9],z=new double[jmax][9],bkx=new double[jmax][9],bky=new
double[jmax][9],bkz=new double[jmax][9],distanze=new double[jmax][6];
   private double[][][] latix=new double[jmax][5][6],latiy=new double[jmax][5][6];
   private int[][] R=new int[jmax][6],G=new int[jmax][6],B=new int[jmax][6],Rk=new int[jmax][6],Gk=new int[jmax][6],Bk=new int[jmax]
[6];
   private static Image img;

   /** Constructor to setup the GUI components */
   public Grafica2() {
      canvas = new DrawCanvas();
      canvas.setPreferredSize(new Dimension(CANVAS_WIDTH, CANVAS_HEIGHT));
      this.setContentPane(canvas);
      this.setDefaultCloseOperation(EXIT_ON_CLOSE);
      this.pack();
      this.setTitle("Lastre che rimbalzano");
      this.setVisible(true);
      for(j=0;j<jmax;j++)
       {
            ox=-50;
            oy=-50;
            oz=-50;
            costx[j]=Math.random()*1200.01/1000.001-0.6;
            costy[j]=Math.random()*1200.01/1000.001-0.6;
            costz[j]=Math.sqrt(1-costx[j]*costx[j]-costy[j]*costy[j]);
            tx[j]=(Math.random()*100.01+30)/30.001;
            ty[j]=(Math.random()*100.01+30)/30.001;
            tz[j]=(Math.random()*100.01+30)/30.001;
            teta[j]=(Math.random()*300.01+20)/2000.001;
            scala=Math.random()*0.5+0.3;
            for(i=0;i<9;i++)
            {
                x[j][i]=cx[i]*scala;
                y[j][i]=cy[i]*scala;
                z[j][i]=cz[i]*scala;
            }
            raggiof[j]=Math.sqrt((x[j][1]-x[j][0])*(x[j][1]-x[j][0])+(y[j][1]-y[j][0])*(y[j][1]-y[j][0])+(z[j][1]-z[j][0])*(z[j][1]-
z[j][0]));
            for(i=0;i<6;i++)
            {
                R[j][i]=(int)(Math.random()*256);
                G[j][i]=(int)(Math.random()*256);
                B[j][i]=(int)(Math.random()*256);
            }
       }
      // Create a new thread to run update at regular interval
      Thread updateThread = new Thread() {
         @Override
         public void run() {
            while (true) {
               update();  // update the (x, y) position
               repaint();  // Refresh the JFrame. Called back paintComponent()
               try {
                  // Delay and give other thread a chance to run
                  Thread.sleep(UPDATE_INTERVAL);  // milliseconds
               } catch (InterruptedException ignore) {}
            }
         }
      };
      updateThread.start(); // called back run()
   }

   /** Update the (x, y) position of the moving object */
   public void update() {
        /*-----------------------------------------------Rimbalzo tra
cubi-------------------------------------------------------------------*/
        for(i=0;i<jmax;i++)
        {
            for (k=i+1;k<jmax;k++)
            {
                /* Calcolo nuova posizione punti centrali di due cubi presi in considerazione per il rimbalzo */
                xip=x[i][0]+tx[i];
                yip=y[i][0]+ty[i];
                zip=z[i][0]+tz[i];
                xkp=x[k][0]+tx[k];
                ykp=y[k][0]+ty[k];
                zkp=z[k][0]+tz[k];
                /* Calcolo distanza tra i centri */
                distanzac1c2=Math.sqrt((x[i][0]-x[k][0])*(x[i][0]-x[k][0])+(y[i][0]-y[k][0])*(y[i][0]-y[k][0])+(z[i][0]-z[k][0])*(z[i]
[0]-z[k][0])+0.001);
                /* Calcolo delle velocita' radiali */
                txr[i]=(((y[i][0]-y[k][0])/distanzac1c2)*costz[i]-((z[i][0]-z[k][0])/distanzac1c2)*costy[i])*teta[i];
                tyr[i]=(((z[i][0]-z[k][0])/distanzac1c2)*costx[i]-((x[i][0]-x[k][0])/distanzac1c2)*costz[i])*teta[i];
                tzr[i]=(((x[i][0]-x[k][0])/distanzac1c2)*costy[i]-((y[i][0]-y[k][0])/distanzac1c2)*costx[i])*teta[i];
                /* Verifica condizione di rimbalzo */
                if((distanzac1c2>(raggiof[i]+raggiof[k]))&&((xip-xkp)*(xip-xkp)+(yip-ykp)*(yip-ykp)+(zip-zkp)*(zip-zkp)<(raggiof[i]
+raggiof[k])*(raggiof[i]+raggiof[k])))
                {
                    /* Calcolo masse cubi */
                    m1=raggiof[i]*raggiof[i]*raggiof[i];
                    m2=raggiof[k]*raggiof[k]*raggiof[k];
                    /* Calcolo quantità di moto cubi */
                    vcx=(m1*tx[i]+m2*tx[k])/(m1+m2);
                    vcy=(m1*ty[i]+m2*ty[k])/(m1+m2);
                    vcz=(m1*tz[i]+m2*tz[k])/(m1+m2);
                    /* Trasformazione in coordinate di centro massa */
                    v01modulo=Math.sqrt((tx[i]-vcx)*(tx[i]-vcx)+(ty[i]-vcy)*(ty[i]-vcy)+(tz[i]-vcz)*(tz[i]-vcz));
                    v02modulo=Math.sqrt((tx[k]-vcx)*(tx[k]-vcx)+(ty[k]-vcy)*(ty[k]-vcy)+(tz[k]-vcz)*(tz[k]-vcz));
                    cosalfa1=((tx[i]-vcx)*(x[k][0]-x[i][0])+(ty[i]-vcy)*(y[k][0]-y[i][0])+(tz[i]-vcz)*(z[k][0]-z[i][0]))/
(distanzac1c2*v01modulo);
                    cosalfa2=((tx[k]-vcx)*(x[i][0]-x[k][0])+(ty[k]-vcy)*(y[i][0]-y[k][0])+(tz[k]-vcz)*(z[i][0]-z[k][0]))/
(distanzac1c2*v02modulo);
                    /* Calcolo nuove velocità dopo il rimbalzo */
                    tx[i]=tx[i]-2*cosalfa1*(x[k][0]-x[i][0])*v01modulo/distanzac1c2;
                    ty[i]=ty[i]-2*cosalfa1*(y[k][0]-y[i][0])*v01modulo/distanzac1c2;
                    tz[i]=tz[i]-2*cosalfa1*(z[k][0]-z[i][0])*v01modulo/distanzac1c2;
                    tx[k]=tx[k]-2*cosalfa2*(x[i][0]-x[k][0])*v02modulo/distanzac1c2;
                    ty[k]=ty[k]-2*cosalfa2*(y[i][0]-y[k][0])*v02modulo/distanzac1c2;
                    tz[k]=tz[k]-2*cosalfa2*(z[i][0]-z[k][0])*v02modulo/distanzac1c2;
                    /* Calcolo direzione di rotazione */
                    vcxr=(m1*txr[i]+m2*txr[k])/(m1+m2);
                    vcyr=(m1*tyr[i]+m2*tyr[k])/(m1+m2);
                    vczr=(m1*tzr[i]+m2*tzr[k])/(m1+m2);
                    /* Trasformazione in coordinate di centro massa */
                    v01modulor=Math.sqrt((txr[i]-vcxr)*(txr[i]-vcxr)+(tyr[i]-vcyr)*(tyr[i]-vcyr)+(tzr[i]-vczr)*(tzr[i]-vczr));
                    v02modulor=Math.sqrt((txr[k]-vcxr)*(txr[k]-vcxr)+(tyr[k]-vcyr)*(tyr[k]-vcyr)+(tzr[k]-vczr)*(tzr[k]-vczr));
                    cosalfa1r=((txr[i]-vcxr)*(x[k][0]-x[i][0])+(ty[i]-vcy)*(y[k][0]-y[i][0])+(tz[i]-vcz)*(z[k][0]-z[i][0]))/
(distanzac1c2*v01modulor);
                    cosalfa2r=((tx[k]-vcx)*(x[i][0]-x[k][0])+(ty[k]-vcy)*(y[i][0]-y[k][0])+(tz[k]-vcz)*(z[i][0]-z[k][0]))/
(distanzac1c2*v02modulor);
                    /* Calcolo nuove velocità di rotazione dopo il rimbalzo */
                    txr[i]=txr[i]-2*cosalfa1r*(x[k][0]-x[i][0])*v01modulor/distanzac1c2;
                    tyr[i]=tyr[i]-2*cosalfa1r*(y[k][0]-y[i][0])*v01modulor/distanzac1c2;
                    tzr[i]=tzr[i]-2*cosalfa1r*(z[k][0]-z[i][0])*v01modulor/distanzac1c2;
                    txr[k]=txr[k]-2*cosalfa2r*(x[i][0]-x[k][0])*v02modulor/distanzac1c2;
                    tyr[k]=tyr[k]-2*cosalfa2r*(y[i][0]-y[k][0])*v02modulor/distanzac1c2;
                    tzr[k]=tzr[k]-2*cosalfa2r*(z[i][0]-z[k][0])*v02modulor/distanzac1c2;
                    costx[i]=costx[i]+txr[i];
                    costy[i]=costy[i]+tyr[i];
                    costz[i]=costz[i]+tzr[i];
                    costx[k]=costx[k]+txr[k];
                    costy[k]=costy[k]+tyr[k];
                    costz[k]=costz[k]+tzr[k];
                    kteta[i]=Math.sqrt(costx[i]*costx[i]+costy[i]*costy[i]+costz[i]*costz[i]+0.0005);
                    kteta[k]=Math.sqrt(costx[k]*costx[k]+costy[k]*costy[k]+costz[k]*costz[k]+0.0005);
                    costx[i]=costx[i]/kteta[i];
                    costy[i]=costy[i]/kteta[i];
                    costz[i]=costz[i]/kteta[i];
                    costx[k]=costx[k]/kteta[k];
                    costy[k]=costy[k]/kteta[k];
                    costz[k]=costz[k]/kteta[k];
                }
            }
        }
        /*-----------------------------------------------------------------------------------------------------------------------
*/
        /* Calcolo nuova posizione (Translazione-rotazione) */
        for(j=0;j<jmax;j++)
        {
            for(i=0;i<9;i++)
            {
                x[j][i]=x[j][i]+tx[j]+((y[j][0]-y[j][i])*costz[j]-(z[j][0]-z[j][i])*costy[j])*teta[j];
                y[j][i]=y[j][i]+ty[j]+((z[j][0]-z[j][i])*costx[j]-(x[j][0]-x[j][i])*costz[j])*teta[j];
                z[j][i]=z[j][i]+tz[j]+((x[j][0]-x[j][i])*costy[j]-(y[j][0]-y[j][i])*costx[j])*teta[j];
            }
            /* Simulazione rallentamento per attrito */
            tx[j]=tx[j]*0.9998;
            ty[j]=ty[j]*0.9998;
            tz[j]=tz[j]*0.9998;
            if(tx[j]>3.01) tx[j]=tx[j]*0.94;
            if(ty[j]>3.01) ty[j]=ty[j]*0.94;
```

```
    if(tz[j]>3.01) tz[j]=tz[j]*0.94;
    teta[j]=teta[j]*0.998;
}
/* Salvo le coordinate per il disegno in prospettiva */
for(j=0;j<jmax;j++) for(i=0;i<9;i++)
{
    bkz[j][i]=Math.sqrt((x[j][i]-ox)*(x[j][i]-ox)+(y[j][i]-oy)*(y[j][i]-oy)+(z[j][i]-oz)*(z[j][i]-oz)+0.000001);
    bkx[j][i]=400.001*Math.atan((y[j][i]-oy)/(x[j][i]-ox));
    bky[j][i]=400.001*Math.acos((z[j][i]-oz)/bkz[j][i]);

}
for(j=0;j<jmax;j++)for(i=0;i<6;i++)
{
    Rk[j][i]=R[j][i];
    Gk[j][i]=G[j][i];
    Bk[j][i]=B[j][i];
}
/* Ordino rispetto alla distanza dei cubi dall'osservatore (Dal piu' lontano al piu' vicino) */
for(k=0;k<jmax-1;k++) for(i=k+1;i<jmax;i++) if(bkz[k][0]<bkz[i][0])
{
    for(p=0;p<6;p++)
    {
        scambio=Rk[k][p];
        Rk[k][p]=Rk[i][p];
        Rk[i][p]=scambio;
        scambio=Gk[k][p];
        Gk[k][p]=Gk[i][p];
        Gk[i][p]=scambio;
        scambio=Bk[k][p];
        Bk[k][p]=Bk[i][p];
        Bk[i][p]=scambio;
    }
    for(p=0;p<9;p++)
    {

        scambiof=bkx[i][p];
        bkx[i][p]=bkx[k][p];
        bkx[k][p]=scambiof;
        scambiof=bky[i][p];
        bky[i][p]=bky[k][p];
        bky[k][p]=scambiof;
        scambiof=bkz[i][p];
        bkz[i][p]=bkz[k][p];
        bkz[k][p]=scambiof;
    }
}
for(j=0;j<jmax;j++) for(i=0;i<6;i++)
{
    latix[j][0][i]=bkx[j][latoa[i]];
    latiy[j][0][i]=bky[j][latoa[i]];
    latix[j][1][i]=bkx[j][latob[i]];
    latiy[j][1][i]=bky[j][latob[i]];
    latix[j][2][i]=bkx[j][latoc[i]];
    latiy[j][2][i]=bky[j][latoc[i]];
    latix[j][3][i]=bkx[j][latod[i]];
    latiy[j][3][i]=bky[j][latod[i]];
    latix[j][4][i]=bkx[j][latoa[i]];
    latiy[j][4][i]=bky[j][latoa[i]];
    distanze[j][i]=bkz[j][latoa[i]]+bkz[j][latob[i]]+bkz[j][latoc[i]]+bkz[j][latod[i]];
}
/* Ordino il vettore e[][] rispetto alla distanza delle facce dei cubi dall'osservatore (Dalla piu' lontana alla piu' vicina) */
for(j=0;j<jmax;j++) for(i=0;i<5;i++) for(k=i+1;k<6;k++) if(distanze[j][i]<distanze[j][k])
{
    scambiof=distanze[j][i];
    distanze[j][i]=distanze[j][k];
    distanze[j][k]=scambiof;
    scambio=Rk[j][i];
    Rk[j][i]=Rk[j][k];
    Rk[j][k]=scambio;
    scambio=Gk[j][i];
    Gk[j][i]=Gk[j][k];
    Gk[j][k]=scambio;
    scambio=Bk[j][i];
    Bk[j][i]=Bk[j][k];
    Bk[j][k]=scambio;
    for(p=0;p<5;p++)
    {
        scambiof=latix[j][p][i];
        latix[j][p][i]=latix[j][p][k];
        latix[j][p][k]=scambiof;
        scambiof=latiy[j][p][i];
        latiy[j][p][i]=latiy[j][p][k];
        latiy[j][p][k]=scambiof;
    }
}
    /* Rimbalzo ai bordi (Cambio la velocita' nel senso del rimbalzo e il verso di rotazione del cubo) */
    for(j=0;j<jmax;j++)
    {
        if((x[j][0]>maxx)&&(tx[j]>0))
        {
            tx[j]=-tx[j];
            costx[j]=-costx[j];
            costy[j]=-costy[j];
            costz[j]=-costz[j];
        }
        if((x[j][0]<0)&&(tx[j]<0))
        {
            tx[j]=-tx[j];
            costx[j]=-costx[j];
            costy[j]=-costy[j];
            costz[j]=-costz[j];
        }
        if((y[j][0]>maxy)&&(ty[j]>0))
        {
            ty[j]=-ty[j];
            costx[j]=-costx[j];
            costy[j]=-costy[j];
            costz[j]=-costz[j];
        }
        if((y[j][0]<0)&&(ty[j]<0))
        {
            ty[j]=-ty[j];
            costx[j]=-costx[j];
            costy[j]=-costy[j];
            costz[j]=-costz[j];
        }
        if((z[j][0]>maxz)&&(tz[j]>0))
        {
            tz[j]=-tz[j];
            costx[j]=-costx[j];
            costy[j]=-costy[j];
            costz[j]=-costz[j];
        }
        if((z[j][0]<0)&&(tz[j]<0))
        {
            tz[j]=-tz[j];
            costx[j]=-costx[j];
            costy[j]=-costy[j];
            costz[j]=-costz[j];
        }
    }
}

/** DrawCanvas (inner class) is a JPanel used for custom drawing */
class DrawCanvas extends JPanel {
   //Image img = Toolkit.getDefaultToolkit().createImage("/home/paolo/Scrivania/img.jpg");
   @Override
   public void paintComponent(Graphics g) {
      super.paintComponent(g);  // paint parent's background
      g.drawImage(img,0,0,1440,900,null);
      /* Disegno i cubi partendo da quelli piu' lontani dall'osservatore */
            for(j=0;j<jmax;j++)
     {
        for(i=0;i<6;i++)
        {
            c1=new Color(Rk[j][i],Gk[j][i],Bk[j][i]);
            g.setColor(c1);
            for(k=0;k<5;k++)
            {
                xpoints[k]=(int)latix[j][k][i];
                ypoints[k]=(int)latiy[j][k][i];
            }
            g.fillPolygon(xpoints, ypoints, npoints);
            /* Disegno le linee di contorno dei cubi */
            x0=xpoints[0];
            y0=ypoints[0];
            x1=xpoints[1];
            y1=ypoints[1];
            x2=xpoints[2];
            y2=ypoints[2];
            x3=xpoints[3];
            y3=ypoints[3];
            g.setColor(Color.BLACK);
            g.drawLine(x0,y0,x1,y1);
            g.drawLine(x1,y1,x2,y2);
            g.drawLine(x2,y2,x3,y3);
            g.drawLine(x3,y3,x0,y0);
        }
     }
   }
}

/** The entry main method */
public static void main(String[] args) throws AWTException {
      // capture the whole screen
      final Image salvaschermo;
      salvaschermo = new Robot().createScreenCapture(
            new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()) );
   // Run GUI codes in Event-Dispatching thread for thread safety
   SwingUtilities.invokeLater(new Runnable() {
      @Override
      public void run() {
         new Grafica2(); // Let the constructor do the job
         Grafica2.img=salvaschermo;
      }
   });
}
}
```