Instituto Tecnológico De Lázaro Cárdenas

TÓPICOS AVANZADOS DE PROGRAMACIÓN

Meteoro Crash

Kevin Esteban Garibo Bracamontes

Marco Antonio Villanueva Guzmán

Carlos Héctor Cruz López

DESCRIPCIÓN:

El juego trata de una nave espacial la cual se mueve solo en dos direcciones izquierda y derecha, la cual tiene que disparar hacia unos meteoros los cuales se acercan a él y este los tiene que destruir, en caso de que algún meteoro llegue a la posición Y en la que la nave se encuentra el juego terminara, para ganar el juego es necesario que se destruyan exactamente 25 meteoros para esto existirá un marcador que estará contando el número de meteoros que has destruido.

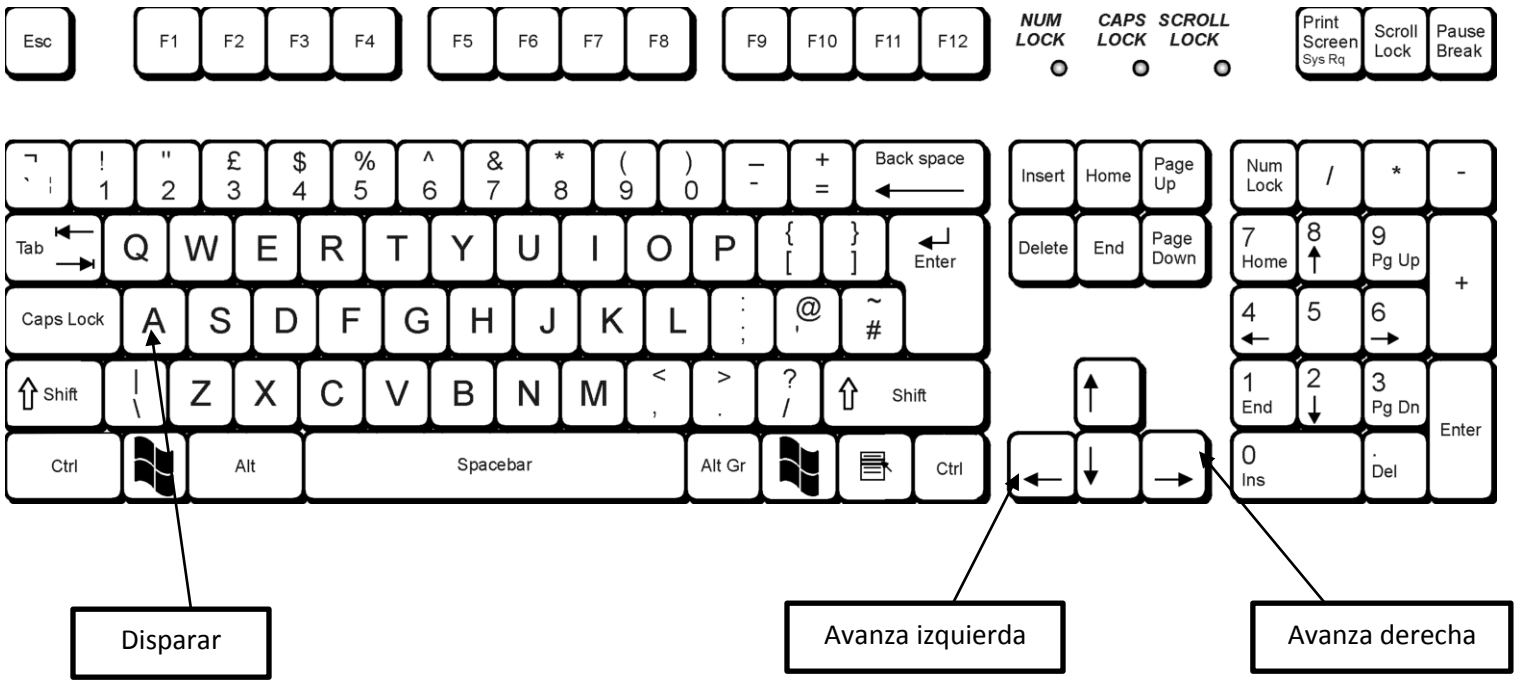
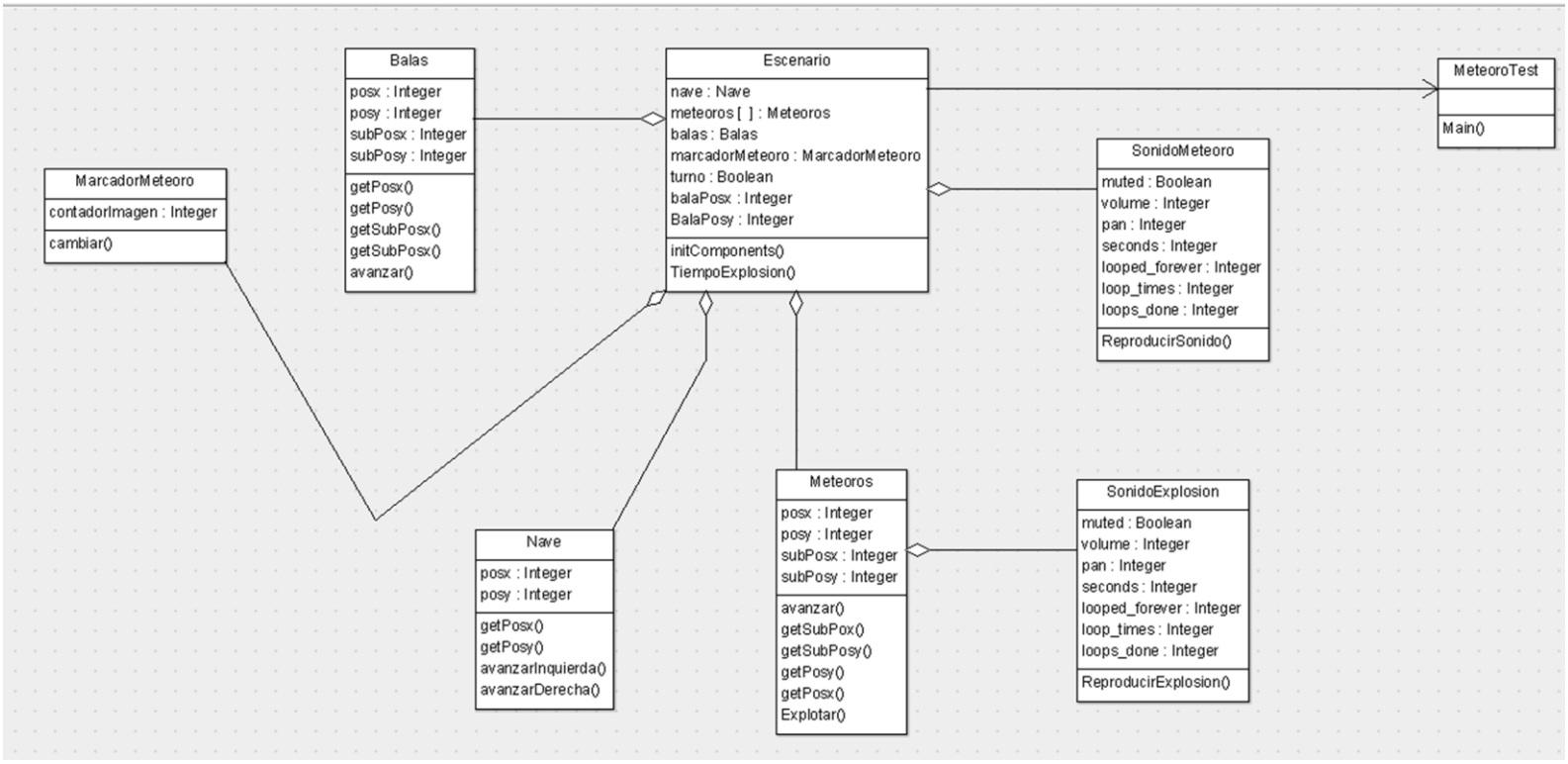


DIAGRAMA UML DE METEORO CRASH:



CLASE BALAS

```
import javax.swing.*;

public class Balas extends JLabel{

   int posx;

   int posy;

   public Balas(int posicionBalaX, int posicionBalaY)

   {

   this.posx=posicionBalaX;

   this.posy=posicionBalaY;

   reshape(this.posx,this.posy , 32, 100 );

   setIcon( new ImageIcon( "imagenes//Rocket.png" ));

   }

      public int getPosx()

      {

         return this.posx;

      }


      public int getPosy()

      {

         return this.posy;

      }


      public int getSubPosx()

      {

         return this.posx+23;
```

```
    }
    public int getSubPosy()
    {
       return this.posy+100;
    }

    public void avanzar(){
    this.posy--;
    this.reshape(this.posx,this.posy , 32, 100 );
  }

}
```

CLASE NAVE

```
import javax.swing.*;

public class Nave extends JLabel{

   int posx;

   int posy;

   public Nave(int posicionNaveX, int posicionNaveY)

   {

      this.posx=posicionNaveX;

      this.posy=posicionNaveY;

      this.setVisible(true);


      setIcon(new ImageIcon( "imagenes//NaveEspacial.png" ));

      reshape(this.posx,this.posy,118,123);

   }


   public int getPosx()

   {

      return this.posx;

   }

   public int getPosy()

   {

      return this.posy;

   }
```

```
    public void avanzarInquierda()
    {
      this.posx=this.posx-40;
      reshape(this.posx,this.posy,118,123);
    }
    public void avanzarDerecha()
    {
      this.posx=this.posx+40;
      reshape(this.posx,this.posy,118,123);
    }
}
```

CLASE MARCADOR METEORO

```
import javax.swing.ImageIcon;
import javax.swing.JLabel;
public class MarcadorMeteoro extends JLabel{
   int contadorImagen=0;
   public MarcadorMeteoro(int X, int Y)
   {
      setIcon( new ImageIcon( "imagenes//marcador//"+contadorImagen+".png" ));
      setVisible(true);
      reshape(600,80,150,650);
   }


   public void Cambiar()
   {
      this.contadorImagen++;
      setIcon( new ImageIcon( "imagenes//marcador//"+contadorImagen+".png" ));
   }


   public int getPuntaje()
   {
      return this.contadorImagen;
   }

}
```

CLASE METEOROS

```
import java.util.Random;
import javax.swing.*;
 public class Meteoros extends  javax.swing.JLabel{

    protected int posx;
    protected int posy;
    public Meteoros() {
        int lugar;
        Random Posicionar = new Random();
         lugar=Posicionar.nextInt(5);
        this.posx=300;
        this.posy=-100;

        if(lugar==0) {
            this.posx=0;
        }
        if(lugar==1)
        {
            this.posx=100;
        }
         if(lugar==2)
        {
            this.posx=200;
        }
```

```
        if(lugar==3)
        {
            this.posx=300;
        }

        if(lugar==4)
        {
            this.posx=400;
        }

            if(lugar==5)
        {
            this.posx=500;
        }
        setIcon( new ImageIcon( "imagenes//roka.png" ));

        this.reshape(this.posx,this.posy,120,120);

}


    public int getPosx(){
        return this.posx;
    }
```

```
  public int getPosy(){
     return this.posy;
  }

  public int getSubPosx()
  {
     return posx+120;
  }
  public int getSubPosy()
  {
     return posy+120;
  }


   public void avanzar(){
       this.posy++;
       this.reshape(posx,posy,120,120);
   }

}
```

CLASE ESCENARIO

```
import java.io.*;
import javax.sound.sampled.*;
import java.awt.event.*;
import javax.swing.*;



public class Escenario extends JFrame implements ActionListener, KeyListener{

   JLabel fondo = new JLabel( new ImageIcon( "imagenes//meteoro-crash.png" ) );
   JLabel gane = new JLabel( new ImageIcon( "imagenes//final//ganar.png" ) );
   JLabel perdi = new JLabel( new ImageIcon( "imagenes//final//perder.png" ) );
   JLabel universo = new JLabel( new ImageIcon( "imagenes//wallspace.png" ) );
   JLabel explosion = new JLabel( new ImageIcon( "imagenes//explosion.gif" ) );
   JButton jugar,salir,ok;

   protected Meteoros meteoros[];
   protected MarcadorMeteoro marcadorMeteoro;
   protected Nave nave;
   protected Balas balas;

   public boolean turno=true;

   int BalaPosx=250;
   int BalaPosy=580;
```

```
int x=0;
int exploit=0;
Timer tiempoBala;
Timer tiempoMeteoro;
Timer tiempoExplosion;

SonidoMeteoro play = new SonidoMeteoro();
SonidoExplosion playExplosion = new SonidoExplosion();

public Escenario()
{
    initComponents();

 for(int i=1;i<26;i++)
 {meteoros[i].setVisible(false);}
 tiempoMeteoro = new Timer (1, new ActionListener(){
 public void actionPerformed(ActionEvent e)
 {

   meteoros[x].avanzar();
    if(balas.getPosy()==meteoros[x].getPosy())
     {
```

```
if((balas.getPosx()<=meteoros[x].getSubPosx())&&(balas.getSubPosx()>=meteoros[x].getPosx())&&
(balas.isVisible()==true))
            {
            meteoros[x].setVisible(false);
            balas.setVisible(false);




            TiempoExplosion();
            tiempoExplosion.start();
            marcadorMeteoro.Cambiar();


            }
         }



    if(marcadorMeteoro.getPuntaje()==25)
    {
       tiempoMeteoro.stop();
       tiempoBala.stop();
       x=0;
       turno=false;
```

```
    marcadorMeteoro.setVisible(false);
    nave.setVisible(false);
    balas.setVisible(false);
    universo.setVisible(false);
    gane.setVisible(true);
    ok.setVisible(true);
    setSize(710,460);
    setLocationRelativeTo(null);

}

 if((meteoros[x].isVisible()==true)&&(meteoros[x].getPosy()==500))
{

    turno=false;
    marcadorMeteoro.setVisible(false);
    nave.setVisible(false);
    balas.setVisible(false);
    universo.setVisible(false);
    perdi.setVisible(true);
    ok.setVisible(true);
    setSize(710,460);
    setLocationRelativeTo(null);
    tiempoMeteoro.stop();
    tiempoBala.stop();
    x=0;
```

```
  }



  if(meteoros[x].getPosy()>720)
  {
  x++;
  if(x<26)
  {
   meteoros[x].setVisible(true);
  }
  }



  }
  });




}

public void initComponents()
{

        this.setLayout(null);
        this.setTitle("Meteoro Crash");
```

```
        this.setVisible(true);
this.addKeyListener(this);
        this.setSize(870,547);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);

this.jugar= new JButton("");
        this.jugar.setIcon(new ImageIcon( "imagenes//jugar.png" ) );
        this.jugar.setVisible(true);
        this.jugar.reshape(580,150,250,40);
this.jugar.addKeyListener(this);
this.jugar.addActionListener(this);
        this.add(jugar);

    this.salir= new JButton("");
        this.salir.setIcon(new ImageIcon( "imagenes//salir.png" ) );
        this.salir.setVisible(true);
        this.salir.reshape(580,300,250,40);
this.salir.addKeyListener(this);
this.salir.addActionListener(this);
        this.add(salir);

this.ok= new JButton("");
        this.ok.setIcon(new ImageIcon( "imagenes//final//ok.png" ) );
        this.ok.setVisible(false);
        this.ok.reshape(620,330,50,40);
```

```
this.ok.addKeyListener(this);
this.ok.addActionListener(this);
        this.add(ok);

this.fondo.reshape(0,0,860,510);
this.fondo.setVisible(true);
this.fondo.addKeyListener(this);
        this.add(this.fondo);

this.gane.reshape(0,0,700,425);
this.gane.setVisible(false);
this.gane.addKeyListener(this);
        this.add(this.gane);

this.perdi.reshape(0,0,700,425);
this.perdi.setVisible(false);
this.perdi.addKeyListener(this);
        this.add(this.perdi);


this.nave= new Nave(250,580);
this.setVisible(true);
add(this.nave);

this.marcadorMeteoro= new MarcadorMeteoro(1300,800);
this.marcadorMeteoro.setVisible(true);
```

```
add(this.marcadorMeteoro);

this.balas=new Balas(0, -100);
this.add(this.balas);
balas.addKeyListener(this);

this.meteoros= new Meteoros[26];
for(int a=0;a<26;a++){
this.meteoros[a]=new Meteoros();
this.add(this.meteoros[a]);
meteoros[a].addKeyListener(this);
}

this.explosion.reshape(0,0,860,510);
this.explosion.setVisible(false);
this.explosion.addKeyListener(this);
        this.add(this.explosion);

this.universo.reshape(0,0,750,800);
this.universo.setVisible(true);
this.universo.addKeyListener(this);
        this.add(this.universo);


  }
```

```
    public void keyTyped (KeyEvent e)
    {
    }
    public void keyReleased(KeyEvent e)
    {
    }
public void keyPressed (KeyEvent e)
{
if(e.getKeyCode() == KeyEvent.VK_LEFT)
{
if(nave.getPosx()>10)
{
this.nave.avanzarInquierda();
}

}
if(e.getKeyCode() == KeyEvent.VK_RIGHT)
{
if(nave.getPosx()<450)
{
this.nave.avanzarDerecha();
}
}

 if(e.getKeyCode() == KeyEvent.VK_A){
```

```
if(turno==true)
{
playExplosion.ReproducirExplosion();

this.BalaPosx=( nave.getPosx()+45);
this.BalaPosy=500;
this.balas=new Balas(BalaPosx, BalaPosy);
this.add(this.balas);
this.balas.addKeyListener(this);
this.balas.setVisible(true);
balas.reshape(BalaPosx,BalaPosy , 32, 70 );

 this.universo.reshape(0,0,750,800);
 this.universo.setVisible(true);
 this.add(this.universo);

 tiempoBala = new Timer (1, new ActionListener(){
 public void actionPerformed(ActionEvent e){

balas.avanzar();
```

```
        if(balas.getPosy()==meteoros[x].getPosy())
        {



if((balas.getPosx()<=meteoros[x].getSubPosx())&&(balas.getSubPosx()>=meteoros[x].getPosx())&&
(balas.isVisible()==true))
           {
           meteoros[x].setVisible(false);
           balas.setVisible(false);



           TiempoExplosion();
           tiempoExplosion.start();
           marcadorMeteoro.Cambiar();


           }
        }

       if(balas.getPosy()<-115)
        {
         turno=true;
         tiempoBala.stop();
        }
```

```
   else
   {
    turno=false;
   }


        }});

   tiempoBala.start();
   }









}
}
public void actionPerformed(ActionEvent e)
     {
   if(e.getSource()==jugar)
   {

   this.jugar.reshape(1000,150,250,40);
   this.salir.setVisible(false);
   this.fondo.setVisible(false);
   this.setSize(750,750);
   this.nave.addKeyListener(this);
```

```
        this.setLocationRelativeTo(null);
        tiempoMeteoro.start();
        play.ReproducirSonido();
        }
        if(e.getSource()==salir)
        {
           System.exit(0);
        }

        if(e.getSource()==ok)
        {
           System.exit(0);
        }
    }



public void TiempoExplosion()
{
   tiempoExplosion = new Timer (10, new ActionListener(){
  public void actionPerformed(ActionEvent e)
  {


    if(exploit<50)
    {
```

```
        explosion.reshape(meteoros[x].getPosx(),meteoros[x].getPosy(),150,230);
        explosion.setVisible(true);
        }
        else
        {explosion.setVisible(false);
        tiempoExplosion.stop();

           exploit=0;
        }
        exploit++;




      }
      });
    }

}
```

CLASE PARA EL SONIDO DE LAS EXPLOSIONES

```
import java.io.*;

import javax.sound.sampled.*;


public class SonidoExplosion {




    public void ReproducirExplosion()

    {


        sound = new File("sonidos//boom.wav"); // Write you own file location here and be aware that it need to be an .wav file


     new Thread(play).start();



  }


   static File sound;

        static boolean muted = false; // This should explain itself

        static float volume = 100.0f; // This is the volume that goes from 0 to 100

        static float pan = 0.0f; // The balance between the speakers 0 is both sides and it goes from -1 to 1
```

```
	static double seconds = 0.0d; // The amount of seconds to wait before the sound starts playing

	static boolean looped_forever = false; // It will keep looping forever if this is true

	static int loop_times = 0; // Set the amount of extra times you want the sound to loop (you don't need to have looped_forever set to true)
	static int loops_done = 0; // When the program is running this is counting the times the sound has looped so it knows when to stop

	final static Runnable play = new Runnable() // This Thread/Runnabe is for playing the sound
	{
		public void run()
		{
			try
			{
				// Check if the audio file is a .wav file
				if (sound.getName().toLowerCase().contains(".wav"))
				{
					AudioInputStream stream = AudioSystem.getAudioInputStream(sound);

					AudioFormat format = stream.getFormat();

					if (format.getEncoding() != AudioFormat.Encoding.PCM_SIGNED)
					{
```

```
                                        format = new
AudioFormat(AudioFormat.Encoding.PCM_SIGNED,
                                                format.getSampleRate(),
                                                format.getSampleSizeInBits() * 2,
                                                format.getChannels(),
                                                format.getFrameSize() * 2,
                                                format.getFrameRate(),
                                                true);

                                        stream =
AudioSystem.getAudioInputStream(format, stream);
                                    }

                                    SourceDataLine.Info info = new DataLine.Info(
                                            SourceDataLine.class,
                                            stream.getFormat(),
                                            (int) (stream.getFrameLength() *
format.getFrameSize()));

                                    SourceDataLine line = (SourceDataLine)
AudioSystem.getLine(info);
                                    line.open(stream.getFormat());
                                    line.start();

                                    // Set Volume
                                    FloatControl volume_control = (FloatControl)
line.getControl(FloatControl.Type.MASTER_GAIN);
```

```
                                        volume_control.setValue((float) (Math.log(volume /
100.0f) / Math.log(10.0f) * 20.0f));


                                        // Mute

                                        BooleanControl mute_control = (BooleanControl)
line.getControl(BooleanControl.Type.MUTE);

                                        mute_control.setValue(muted);


                                        FloatControl pan_control = (FloatControl)
line.getControl(FloatControl.Type.PAN);

                                        pan_control.setValue(pan);


                                        long last_update = System.currentTimeMillis();

                                        double since_last_update = (System.currentTimeMillis() -
last_update) / 1000.0d;


                                        // Wait the amount of seconds set before continuing

                                        while (since_last_update < seconds)

                                        {

                                                since_last_update = (System.currentTimeMillis() -
last_update) / 1000.0d;

                                        }




                                        int num_read = 0;

                                        byte[] buf = new byte[line.getBufferSize()];
```

```
                    while ((num_read = stream.read(buf, 0, buf.length)) >= 0)
                    {
                        int offset = 0;

                        while (offset < num_read)
                        {
                            offset += line.write(buf, offset, num_read -
offset);
                        }
                    }

                    line.drain();
                    line.stop();

                    if (looped_forever)
                    {
                        new Thread(play).start();
                    }
                    else if (loops_done < loop_times)
                    {
                        loops_done++;
                        new Thread(play).start();
                    }
                }
            }
```

```
                    catch (Exception ex) { ex.printStackTrace(); }
                }
            };

}
```

CLASE SONIDO METEORO USADA PARA REPRODUCIR EL SONIDO DE FONDO

```
import java.io.*;

import javax.sound.sampled.*;


public class SonidoMeteoro {




    public void ReproducirSonido()

    {


        sound = new File("sonidos//marcha.wav"); // Write you own file location here and be aware that it need to be an .wav file


     new Thread(play).start();



  }


   static File sound;

        static boolean muted = false; // This should explain itself

        static float volume = 100.0f; // This is the volume that goes from 0 to 100

        static float pan = 0.0f; // The balance between the speakers 0 is both sides and it goes from -1 to 1
```

```
	static double seconds = 0.0d; // The amount of seconds to wait before the sound starts playing

	static boolean looped_forever = false; // It will keep looping forever if this is true

	static int loop_times = 0; // Set the amount of extra times you want the sound to loop (you don't need to have looped_forever set to true)

	static int loops_done = 0; // When the program is running this is counting the times the sound has looped so it knows when to stop

	final static Runnable play = new Runnable() // This Thread/Runnabe is for playing the sound
	{
		public void run()
		{
			try
			{
				// Check if the audio file is a .wav file
				if (sound.getName().toLowerCase().contains(".wav"))
				{
					AudioInputStream stream = AudioSystem.getAudioInputStream(sound);

					AudioFormat format = stream.getFormat();

					if (format.getEncoding() != AudioFormat.Encoding.PCM_SIGNED)
```

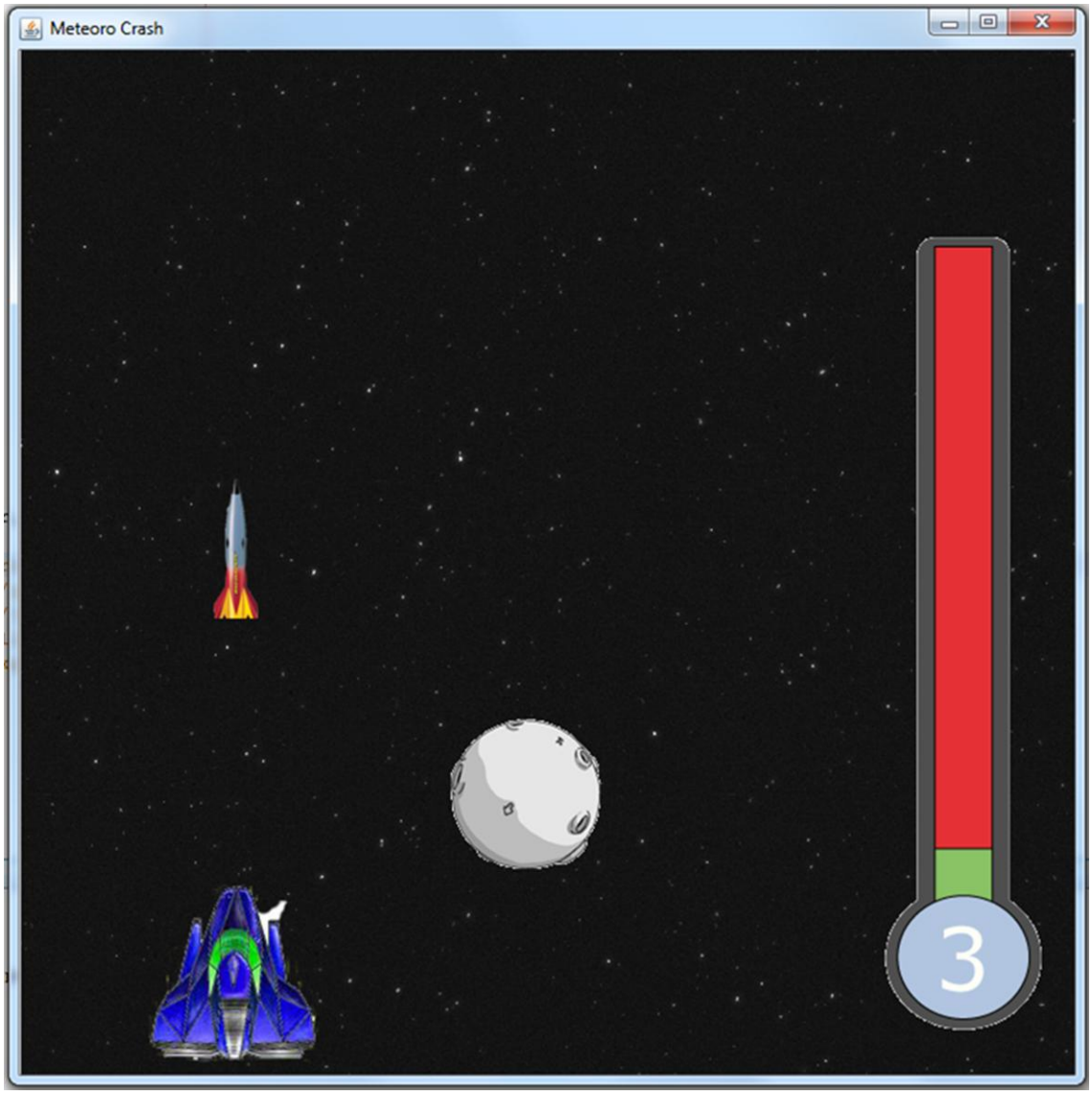
```
                {
                        format = new
AudioFormat(AudioFormat.Encoding.PCM_SIGNED,
                                        format.getSampleRate(),
                                        format.getSampleSizeInBits() * 2,
                                        format.getChannels(),
                                        format.getFrameSize() * 2,
                                        format.getFrameRate(),
                                        true);

                        stream =
AudioSystem.getAudioInputStream(format, stream);
                }

                SourceDataLine.Info info = new DataLine.Info(
                                SourceDataLine.class,
                                stream.getFormat(),
                                (int) (stream.getFrameLength() *
format.getFrameSize()));

                SourceDataLine line = (SourceDataLine)
AudioSystem.getLine(info);
                line.open(stream.getFormat());
                line.start();

                // Set Volume
```

```
                                        FloatControl volume_control = (FloatControl)
line.getControl(FloatControl.Type.MASTER_GAIN);

                                        volume_control.setValue((float) (Math.log(volume /
100.0f) / Math.log(10.0f) * 20.0f));



                                        // Mute

                                        BooleanControl mute_control = (BooleanControl)
line.getControl(BooleanControl.Type.MUTE);

                                        mute_control.setValue(muted);



                                        FloatControl pan_control = (FloatControl)
line.getControl(FloatControl.Type.PAN);

                                        pan_control.setValue(pan);



                                        long last_update = System.currentTimeMillis();

                                        double since_last_update = (System.currentTimeMillis() -
last_update) / 1000.0d;



                                        // Wait the amount of seconds set before continuing

                                        while (since_last_update < seconds)

                                        {

                                                since_last_update = (System.currentTimeMillis() -
last_update) / 1000.0d;

                                        }







                                        int num_read = 0;
```

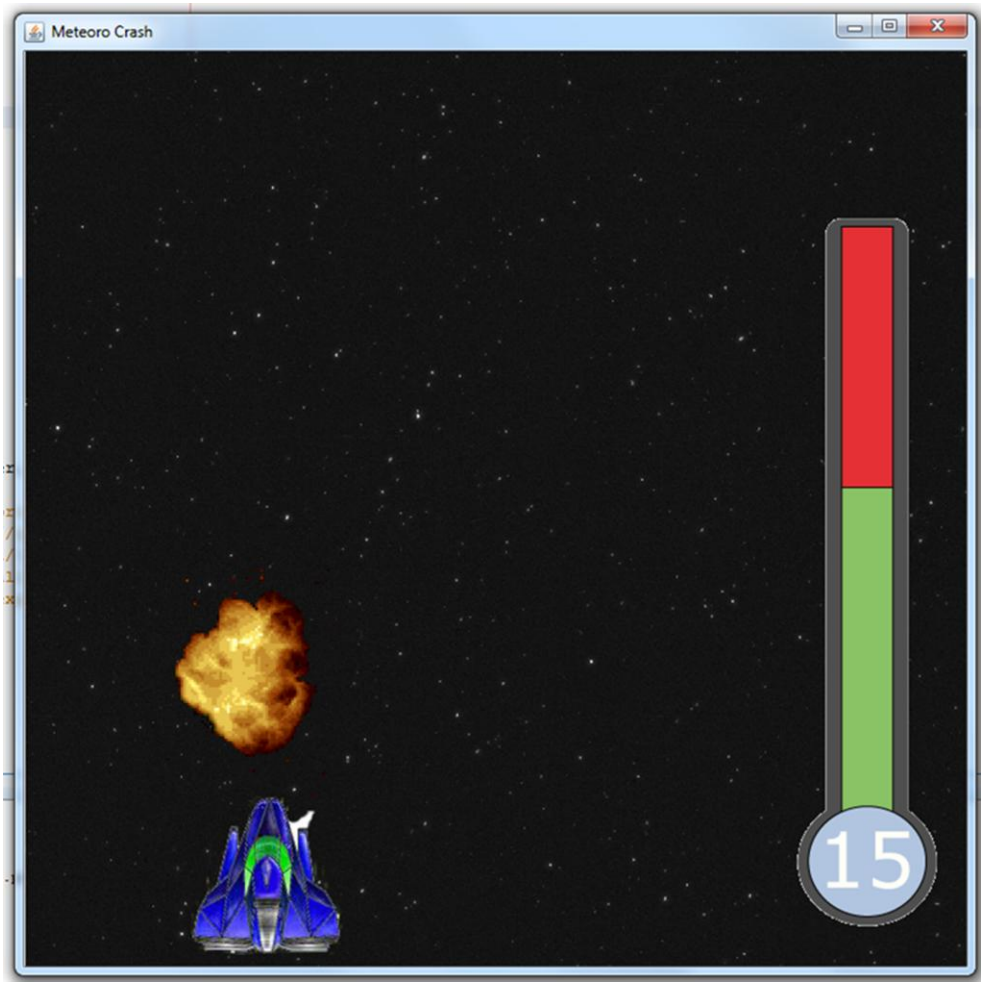
```
                byte[] buf = new byte[line.getBufferSize()];

                while ((num_read = stream.read(buf, 0, buf.length)) >= 0)
                {
                        int offset = 0;

                        while (offset < num_read)
                        {
                                offset += line.write(buf, offset, num_read -
offset);
                        }
                }

                line.drain();
                line.stop();

                if (looped_forever)
                {
                        new Thread(play).start();
                }
                else if (loops_done < loop_times)
                {
                        loops_done++;
                        new Thread(play).start();
                }
        }
```

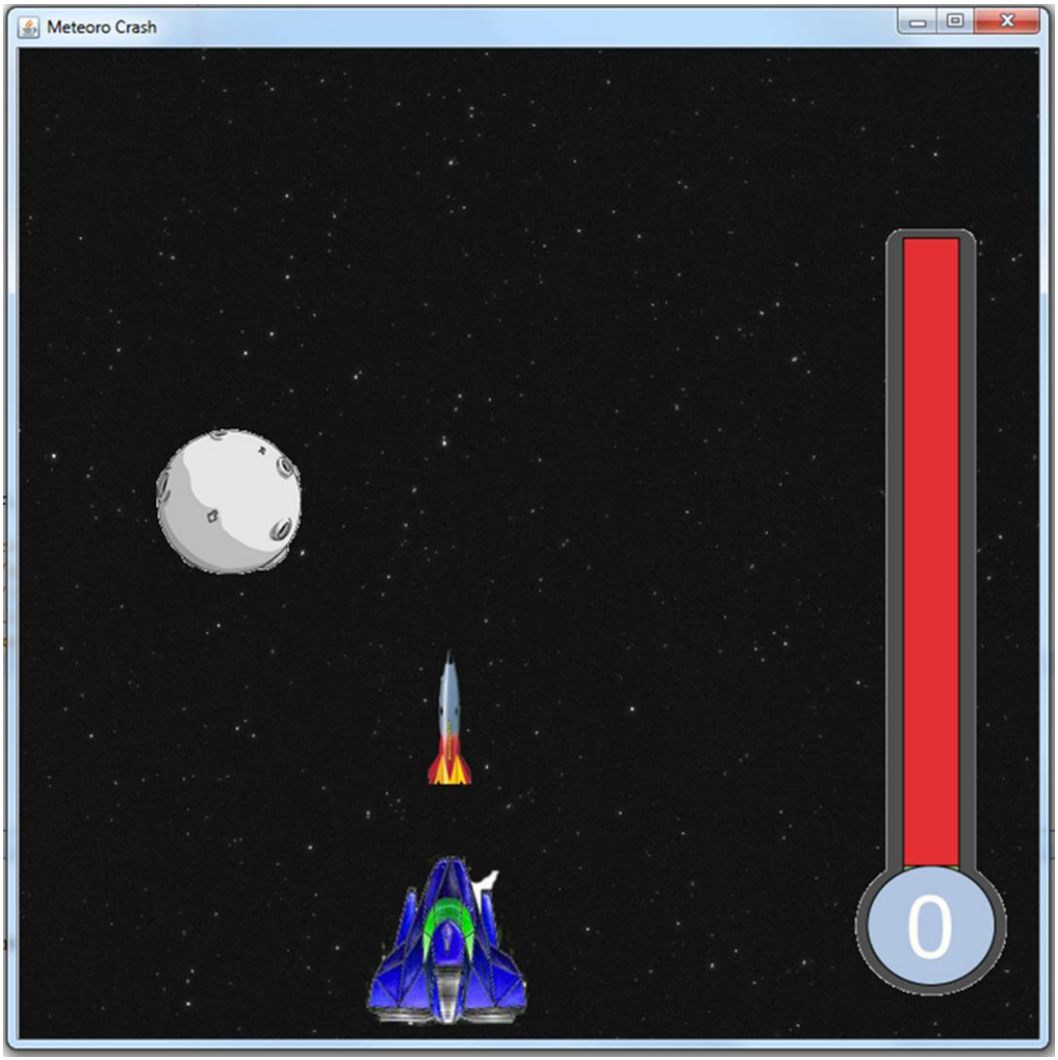
```
                    }
                    catch (Exception ex) { ex.printStackTrace(); }
                }
            };

}
```

Al iniciar el juego nos saldrá este intro aquí podemos escoger si queremos jugar o salir



Así se ve el juego cuando se está corriendo


Meteoro Crash
15


Meteoro Crash
0

En caso de ganar aparecerá esta ventana felicitándote

En caso de perder aparecerá esta imagen