

RV COLLEGE OF ENGINEERING
COMPUTER SCIENCE DEPT

R.V.COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING

PROJECT REPORT

GRAPHICS PROGRAMMING IN JAVA

GAME DEVELOPMENT.

DEVELOPED BY –

SRIVAS N CHENNU (1RV98CS86)

VISHWAS N (1RV98CS104)

CERTIFICATE

This is to certify that

Srivas N. Chennu (1RV98CS086)

of 7th semester CSE Department has successfully
and satisfactorily completed the project

Graphics programming using Java

in partial fulfillment of the Internet Programming lab
as prescribed by the VTU for the academic year 2001-
2002.

SIGNATURE OF THE
INTERNAL EXAMINER

SIGNATURE OF THE
EXTERNAL EXAMINER

SIGNATURE OF THE
HOD CSE DEPT.

TABLE OF CONTENTS

SYNOPSIS	4
INTRODUCTION	5
SOFTWARE REQUIREMENT SPECIFICATION .	7
<i>Requirement analysis</i>	7
<i>Problem</i>	7
<i>Goal</i>	8
<i>Scope, references, overview</i>	8
<i>Constraints</i>	9
DESIGN	10
<i>The frame class</i>	10
IMPLEMENTATION	14
<i>Functional description</i>	14
SOURCE CODE	19

SYNOPSIS

SPACEWOLF is a game designed and implemented in Java. This game is a standalone application and runs on any machine with a JVM. This game is single user game and is played with the help of keyboard. Mouse is also used for providing input in some cases.

The key features of this game are:

- This is a standalone game created using frames in java.
- This game is a single user game.
- This game uses image manipulation for providing optimal effect.
- This game uses sounds which is enhances the feel of the game.
- This is a uni-threaded game.
- This works on a standalone system and cannot be played over a network.
- This game has eleven difficulty levels and provides for starting and ending the game.
- The game provides for two kinds of interaction with the keyboard and mouse.
- Since java is an object oriented language the code is implemented in classes.

INTRODUCTION

Java provides a lot of features for graphics programming thus many games are written in java. Java has been used to create a lot of games in existence. Most of the graphics that we see in browsers are java applets which uses the java graphics package.

Java graphics package for the following operations, draw lines of any thickness, fill shapes with radients and textures, move, rotate, scale, and shear text and graphics and finally composite overlapping text and graphics.

The Java 2D API also enables you to store and to manipulate image data--for example, you can easily perform image-filter operations, such as blur and sharpen.

The Java 2D API introduced in JDK 1.2 provides enhanced two-dimensional graphics, text, and imaging capabilities for Java programs through extensions to the Abstract Windowing Toolkit (AWT). This comprehensive rendering package supports line art, text, and images in a flexible, full-featured framework for developing richer user interfaces, sophisticated drawing programs and image editors.

The Java 2D API provides

1. A uniform rendering model for display devices and printers
2. A wide range of geometric primitives, such as curves, rectangles, and ellipses and a mechanism for rendering virtually any geometric shape
3. Mechanisms for performing hit detection on shapes, text, and images
4. A compositing model that provides control over how overlapping objects are rendered
5. Enhanced color support that facilitates color management
6. Support for printing complex documents

SOFTWARE REQUIREMENT SPECIFICATION

This is mainly the specification of the problem and how the solution is brought about.

Since requirements of a typical game are known, only the basic requirements are stated below without elaborating each of the functionalities.

REQUIREMENT ANALYSIS

This phase deals with the statement and understanding of the problems, the goals, and the constraints, which are listed below for the development of Game (in Java).

PROBLEM

The problem or aim of this project was to create a standalone application using frames in java which explores the graphics capabilities of the java package. The game so designed thus should have the following minimum features, the game must have user interaction so the game is interactive the game must use the graphics object primitives to draw lines etc so that the primitives features are known, secondly the game must be capable of interactions with the mouse or the keyboard, thirdly there has image processing included in the game so its implementation is known and also the quality of the

game is enhanced and lastly the game must use sounds in the game to alert the user to the state of game.

The game must also be fast enough so that users experience the capabilities of the java language.

The game must be able to run on any machine with java installed on it.

GOAL

The goal of the graphics package is to implement all the features discussed in the problem statement efficiently with great performance and with minimum resources possible.

SCOPE, REFERENCES, OVERVIEW

The scope of the graphics project is to build a stand-alone application, which runs on the java platform. Specifically the following functionality should be implemented.

- The game must be standalone.
- The game must provide user interaction.
- The game must be of sufficient complexity.
- The game must use the frame classes and the graphics object.
- The game can include images and audio features.
- The game can be a single player game.

CONSTRAINTS

As the game is a standalone application it should not be designed using applets and should not be network playable. The game being highly interactive should have a fast response time and should be very efficient. The game should be made to consume minimal resources that way it can be run even on low-end systems. Needless to say the game must be robust and fast.

DESIGN

First we must understand the features that the java provides for creating a Java graphics application. So in the following section we first discuss the Java graphics package.

The classes and interfaces of the Abstract Windowing Toolkit (AWT) are used to develop stand-alone applications and to implement the GUI controls used by applets. These classes support all aspects of GUI development, including event handling.

The Component and Container classes are two of the most important classes in the java.awt package. The Component class provides a common superclass for all classes that implement GUI controls. The Container class is a subclass of the Component class and can contain other AWT components.

The Window class is a subclass of the Container class that provides a common set of methods for implementing windows. The Window class has two subclasses, Frame and Dialog, that are used to create Window objects. The Frame class is used to create a main application window, and the Dialog class is used to implement dialog boxes. Let's explore the Frame class first and then look at the Dialog class.

THE FRAME CLASS

Constructors:**public Frame()**

Constructs a new instance of Frame that is initially invisible.

public Frame(String title)

Constructs a new, initially invisible Frame object with the specified title.

Methods:**public void addNotify()**

Creates the Frame's peer. The peer allows us to change the look of the Frame without changing its functionality.

public String getTitle()

Gets the title of the frame.

Returns: the title of this frame, or null if this frame doesn't have a title.

public synchronized void setTitle(String title)

Sets the title for this frame to the specified title.

Parameters: title - the specified title of this frame.

public Image getIconImage()

Gets the icon image for this frame.

Returns: the icon image for this frame, or null if this frame doesn't have an icon image.

public synchronized void setIconImage(Image image)

Sets the image to display when this frame is iconized. Not all platforms support the concept of iconizing a window.

Parameters: image - the icon image to be displayed

public MenuBar getMenuBar()

Gets the menu bar for this frame.

Returns: the menu bar for this frame, or null if this frame doesn't have a menu bar.

public void setMenuBar(MenuBar mb)

Sets the menu bar for this frame to the specified menu bar.

Parameters: mb - the menu bar being set

public synchronized void setResizable(boolean resizable)

Sets the resizable flag, which determines whether this frame is resizable. By default, all frames are initially resizable.

Parameters: resizable - true if this frame is resizable; false otherwise.

public void remove(MenuComponent m)

Removes the specified menu bar from this frame.

public void dispose()

Disposes of the Frame. This method must be called to release the resources that are used for the frame. All components contained by the frame and all windows owned by the frame will also be destroyed.

protected String paramString()

Returns the parameter String of this Frame.

public synchronized void setCursor(int cursorType)

public int getCursorType()

This game is single user space fight game it has been designed as follows:

First, the background is created this is consists of a black screen and randomly generated stars in the background. The stars are colored.

Second, The ship is loaded and then the meteors are loaded this is also done randomly the meteors start falling down the screen and now the game starts

Third, the ship is controlled by user, hence keyboard events are processed and if the keys are arrow keys the ship moves in the appropriate direction.

Fourth, the ship's position is plotted and if the ship collides with the meteors the shield decreases and the audio-clip indicating the collision is played.

Fifth, the ship can fire at the meteors if the fire hit a meteor the laser decreases and the user gets additional points and the audio-clip is played.

Sixth, the ship regenerates its shield and laser this occurs at fixed intervals.

Seventh, the ship after a fixed time interval moves into the second level here the ship encounters a new object, another space ship it has to destroy this ship else loses its shield points.

This way the game has a sequential working logic, thus easy to understand and code, changes to the game can be easily incorporated and since java being a strictly Object oriented language has a higher level of abstraction.

IMPLEMENTATION

This section deals with the implementation details of the project and also explains the functions of the subroutines used in the game.

Java is an Object oriented language hence the functions are exclusively in a class so we explain the function declarations and their logic

FUNCTIONAL DESCRIPTION

Void init():

This function is the starting point of the program the function initializes all the required variables and loads all the images and the audios required for the game it also initializes the array used for the displaying of the meteors.

Void initStars():

This function is useful in initializing the background stars, this is done by using the `Math.random()` function which enables the creation of stars at random positions on the screen.

Boolean keyDown(Event e, int key):

This function processes the keyboard event when any key is pressed down. The following are the cases

- Down arrow key: $dy=1$, i.e the ship should move down (value =1 because of the coordinate axis).

- Up arrow key: $dy=-1$, i.e the ship should move up.
- Left arrow key: $dx=-1$ i.e the ship should move left.
- Right arrow key: $dx=1$ i.e the ship should move right.
- Space bar: fire the gun function is called.
- Escape key: The game is stopped.
- 'S' or 's' key: If the game is stopped it is restarted.

Returns true.

Void keyUp(Event e, int key):

This functions prints out to the std o/p the key if the key pressed is not any one of the arrow keys. Returns true.

Void paint(Graphics g):

This function paints the screen and calls the `PlayGame()` function if the game is being played or calls `ShowIntoScreen()` function.

Void NewMeteor():

This function checks the existing meteors in the screen and if the number is less than maximum then creates random meteors for the first line as the last line is being phased out.

Void MoveShip():

This function moves the ship in the corresponding direction according to the dx and dy values and while checking the screen height and width.

Void SunBird():

This function checks the position of the ship in correspondence with the meteors i.e their relative positions and then calls the HitShip() function is the ship has collided with the meteor.

Void HitShip():

This function checks to see if the shield is zero then the game is over else the shield count is decreased.

Void PlayGame():

This function calls NewMeteor(), MoveShip(), DrawPlayField() which are described above. It then calls the Showscore() function which then displays the score of the game. The game consists of 11 difficulty levels thus proper factors are increased. The function then renews the firepower i.e the ships energy.

Void ShowIntoScreen():

This function displays the text contents on the screen like the design patterns and the lines, the text displaying the authors names and the final score.

Void DrawPlayField():

As is obvious draws the play field of the game the engine fire of the ship etc.

Void ShowScore():

This displays the score while in game, the laser, shield values and the laser and shield bars.

Void FireGun():

This displays the gun fired shows the gun graphics and displays the sound of gun fire.

Void KillEmAll():

This shows the gun fire traveling across the screen.

Void MetHit():

This shows the bullet hit the meteor and sound for the kill being played. It then calls the function DelMeteor(n) for deleting meteor 'n' which was hit.

Void ShowMeteors():

This shows the meteors on the screen. The meteors are going down the screen hence have to be deleted.

Void MoveStars():

If a star in the background reaches the bottom then it will go back to the top.

Void Collisions():

This checks to see if the meteors collide with the ship if true the HitShip() and DelMeteor(n) are called.

Void DelMeteor():

The meteor is removed from the display.

Color NewColor():

This function selects a new color randomly and returns it.

Void start():

This starts the application by creating a new thread.

Void Stop():

This stops the game by setting the current thread to null.

Boolean mouseDown(Event e, int xx, int yy):

This takes care of the mouse movements and mouse clicks.

Void GameStart():

This sets up the ship and the other settings.

Void GameStop():

Sets ingame=false thus telling that the game is over.

SOURCE CODE

```
import java.awt.*;
import java.net.*;
import java.applet.Applet;
import java.applet.AudioClip;

public class Absolute extends Applet implements Runnable
{
    Dimension d;
    Font      largefont = new Font("Helvetica", Font.PLAIN, 24);
    Font      smallfont = new Font("Helvetica", Font.PLAIN, 14);

    FontMetrics  fmsmall, fmlarge;
    Graphics     goff;
    Image        ii;
    Thread       thethread;

    boolean     ingame=false;

    int         x, y, mousex, mousey, oldx, oldy, dx=0, dy=0, count, shield=0;
    boolean     showtitle=true;
    Image       ship;
    Image[]     fire;
    int         firecnt=0;

    // Bullet variables
    Image       bullet;
    int[]       bx;
    int[]       by;
    final int   bmy=16, bul_xs=54, bul_ys=8;

    // Meteor variables
    Image       meteor;
    int         maxmet, metcount, mtotal, mrenew, metmy;
    int[]       metx;
    int[]       mety;
    int[]       metf;
    boolean[]   metr;
    final int   sxmet=80, symet=84;

    // These are for the star field
    public int starsX[];
```

```

public int starsY[];
public Color starsC[];
public int numStars = 30;
public int speed = 6, xSize, ySize;

// Variables for big boom
Image[] boom;
int rndbx, rndby, rndcnt=777;
final int sxbom=71, sybom=100, bframes=4;

// Global Variables
int distance=0, maxdist=2000;
int slevel, blevel, difflev, bosslevel;
int smax, bmax;
int scur, bcur, renew, rcnt=0, sstretch, txtalign=100;
long score;

// Sounds
AudioClip blast, crash, kill;

// Bosses here
// Sunbird
boolean sunbird, sbefore, safer;
int sbx, sby, sbmove, maxtribe, tribe;
int[] sbfx, sbfy;

final int maxshield=9;
final int backcol=0x102040;
final int fireframe=2;
final int borderwidth=0;
final int sxsize=90, sysize=39, sxfire=11, syfire=6;
final int movex=10, movey=5;
final int scoreheight=45;
final int screendelay=300;

public String getAppletInfo()
{
    return("Absolute Space - by Vishwas N and Srivas N Chennu");
}

public void init()
{
    Graphics g;
    int n;
    d = size();
    setBackground(Color.black);
    g=getGraphics();
    g.setFont(smallfont);
    fmsmall = g.getFontMetrics();
    g.setFont(largefont);
    fmlarge = g.getFontMetrics();
    ship = getImage(getCodeBase(), "ship.gif");

```

```

bullet = getImage(getCodeBase(), "bullet.gif");
fire = new Image[fireframe];
for (n=0; n<fireframe; n++) {
    fire[n] = getImage(getCodeBase(), "fire"+n+".gif");
}
boom = new Image[bframes+1];
for (n=0; n<=bframes; n++) {
    boom[n] = getImage(getCodeBase(), "boom"+n+".gif");
}

xSize = d.width - borderwidth*2;
ySize = d.height - borderwidth*2 - scoreheight;

x = (xSize - sxsizex) / 2;
y = ySize - sysize - scoreheight - borderwidth;
mousex = -1;

// will be override by command line parameters
blevel = 3;
slevel = 3;

bx = new int[blevel*10];
by = new int[blevel*10];

for (n=0; n<blevel*10; n++) {
    bx[n] = -1;
}

// Meteor initialize
meteor = getImage(getCodeBase(), "meteor.gif");
maxmet = d.height / symet + 1;
maxmet = maxmet * 10;
metx = new int[maxmet];
mety = new int[maxmet];
metf = new int[maxmet];
metr = new boolean[maxmet];

// Audio
try {
    blast = getAudioClip(new URL(getDocumentBase(), "blast.au"));
    crash = getAudioClip(new URL(getDocumentBase(), "collisn.au"));
    kill = getAudioClip(new URL(getDocumentBase(), "mdestr.au"));
}
catch (MalformedURLException e) {}

blast.play(); blast.stop();
crash.play(); crash.stop();
kill.play(); kill.stop();

initStars();

```

```

rndcnt = 777;

// Bosses
// Sunbird
sbfx = new int[11];
sbfy = new int[11];
sbfx[0] = 10;
sbfy[0] = 0;
sbfx[1] = 15;
sbfy[1] = 10;
sbfx[2] = 0;
sbfy[2] = 10;
sbfx[3] = 3;
sbfy[3] = 15;
sbfx[4] = 17;
sbfy[4] = 15;
sbfx[5] = 20;
sbfy[5] = 20;
sbfx[6] = 23;
sbfy[6] = 15;
sbfx[7] = 37;
sbfy[7] = 15;
sbfx[8] = 40;
sbfy[8] = 10;
sbfx[9] = 25;
sbfy[9] = 10;
sbfx[10] = 30;
sbfy[10] = 0;

}

// This creates the starfield in the background
public void initStars () {
    starsX = new int[numStars];
    starsY = new int[numStars];
    starsC = new Color[numStars];
    for (int i = 0; i < numStars; i++) {
        starsX[i] = (int) ((Math.random() * xSize - 1) + 1);
        starsY[i] = (int) ((Math.random() * ySize - 1) + 1);
        starsC[i] = NewColor();
    }
}

public boolean keyDown(Event e, int key)
{
    if (ingame)
    {
        mousex = -1;
        if (key == Event.LEFT)
            dx=-1;
        if (key == Event.RIGHT)

```

```

    dx=1;
    if (key == Event.UP)
        dy=-1;
    if (key == Event.DOWN)
        dy=1;
    if (key == ' ')
        if (bcur>0) FireGun();
    if (key == Event.ESCAPE)
        ingame=false;
}
else
{
    if (key == 's' || key == 'S')
    {
        ingame=true;
        GameStart();
    }
}
return true;
}

public boolean keyUp(Event e, int key)
{
    System.out.println("Key: "+key);
    if (key == Event.LEFT || key == Event.RIGHT)
        dx=0;
    if (key == Event.UP || key == Event.DOWN)
        dy=0;
    return true;
}

public void paint(Graphics g)
{
    String s;
    Graphics gg;

    if (goff==null && d.width>0 && d.height>0)
    {
        ii = createImage(d.width, d.height);
        goff = ii.getGraphics();
    }
    if (goff==null || ii==null)
        return;

    goff.setColor(Color.black);
    goff.fillRect(0, 0, d.width, d.height);

    if (ingame)
        PlayGame();
    else

```

```

    ShowIntroScreen();
    g.drawImage(ii, 0, 0, this);
}

public void PlayGame()
{

    // Big bosses here
    if (sunbird) SunBird();

    ShowScore();
    distance++;
    score+=100;
    if (distance % maxdist == 0) {
        difflev++;
        if (difflev>2 & difflev<10) {
            renew-=20;
            bmax+=1;
            smax+=1;
            metmy++;
            mrenew--;
        }
        if (difflev>3 & difflev<11) {
            maxtribe++;
            sbmove++;
        }
        if (difflev>3) {
            sunbird = true;
            tribe = maxtribe;
        }
    }
}

// Renew Ship Energy
rcnt++;
if (rcnt % (renew / blevel) == 0) {
    bcur++;
    if (bcur>bmax) bcur=bmax;
}
if (distance % 500 == 0) {
    scur++;
    if (scur>smax) scur=smax;
}
if (rcnt>renew) rcnt=0;
}

public void ShowIntroScreen()
{
    String s;

    DrawPlayField();
    goff.setFont(largefont);

```

```

if (rndcnt > bframes) {
    rndbx = (int) (Math.random() * (xSize - sxbom) + 1);
    rndby = (int) (Math.random() * (ySize - sybom) + 1);
    rndcnt = 0;
}

goff.drawImage(boom[rndcnt], rndbx, rndby, this);
rndcnt++;
for (int i=0; i<xSize/bul_xs; i++) {
    goff.drawImage(bullet, i*bul_xs, 0, this);
    goff.drawImage(bullet, i*bul_xs, ySize-bul_ys, this);
}

if (showtitle)
{
    goff.setColor(new Color(0xff0000));
    s="Absolute Space";
    goff.drawString(s,(d.width-fmlarge.stringWidth(s)) / 2, (d.height-scoreheight-borderwidth)/2 -
20);
    goff.setColor(new Color(0xff00ff));
    s="Copyright - Vishwas N and Srivas N Chennu";
    goff.setFont(smallfont);
    goff.drawString(s,(d.width-fmsmall.stringWidth(s))/2,(d.height-scoreheight-borderwidth)/2 +
10);
    s="Project for IP lab in RVCE";
    goff.drawString(s,(d.width-fmsmall.stringWidth(s))/2,(d.height-scoreheight-borderwidth)/2 +
30);
}
else
{
    goff.setFont(smallfont);
    goff.setColor(new Color(0xffff00));
    s="Leftclick to start game";
    goff.drawString(s,(d.width-fmsmall.stringWidth(s))/2,(d.height-scoreheight-borderwidth)/2 - 10);
    goff.setColor(new Color(0x00ff00));
    s="Use cursor keys move, click or press SPACE to fire";
    goff.drawString(s,(d.width-fmsmall.stringWidth(s))/2,(d.height-scoreheight-borderwidth)/2 +
20);
    goff.setFont(largefont);
    goff.setColor(new Color(0xff00ff));
    s="LAST SCORE: "+score;
    goff.drawString(s,(d.width-fmlarge.stringWidth(s))/2,(d.height-scoreheight-borderwidth)/2 +
120);
}
count--;
if (count<=0)
{ count=screendelay; showtitle=!showtitle; }
}

```

```

public void DrawPlayField()
{
    // Show stars
    moveStars();
    for (int a = 0; a < numStars; a++) {
        goff.setColor(starsC[a]);
        goff.drawRect(starsX[a], starsY[a], 1, 1);
    }

    Show Meteors();
    KillEmAll();
    goff.drawImage(ship, x, y, this); // paint ship
    if (firecnt != 0) {
        goff.drawImage(fire[firecnt-1], x+( (sxsize-sxfire) / 2 ), y+sysize, this); // engine fire
    }
    firecnt++;
    if (firecnt > 2) firecnt=0;
    Collisions();

    if (shield>0) {
        goff.setColor(new Color(0x00ffff));
        goff.drawOval(x-shield, y-shield, sxsize+shield*2, sysize+shield*2);
        shield--;
    }
}

public void ShowScore()
{
    String s;
    int my;
    sstretch = (xSize-txtalign*2)/Math.max(bmax,smax);
    // Laser bar
    my = d.height-scoreheight+10;
    goff.setColor(new Color(0x00ff96));
    goff.drawRect(txtalign, my-10, bmax*sstretch, 10);
    goff.setFont(smallfont);
    s="laser: "+bcur+"/"+"bmax;
    goff.fillRect(txtalign, my-10, bcur*sstretch, 10);
    goff.drawString(s,10,my);
    // Shield bar
    my += 15;
    goff.setColor(new Color(0x00ffff));
    goff.drawRect(txtalign, my-10, smax*sstretch, 10);
    goff.setFont(smallfont);
    s="shield: "+scur+"/"+"smax;
    goff.fillRect(txtalign, my-10, scur*sstretch, 10);
    goff.drawString(s,10,my);
    // Score
    my += 20;
    goff.setColor(new Color(0xffffff));

```

```

goff.setFont(largefont);
s="score: "+score;
goff.drawString(s,10,my);
}

public void MoveShip()
{
int xx, yy;
oldx = x;
oldy = y;

xx = mousex;
if (xx>0) {
yy = mousey;
if (xx<x) dx=-1;
if (xx>x+sxsize) dx=1;
if (yy<y) dy=-1;
if (yy>y+sysize) dy=1;
if (xx>x & xx<x+sxsize & yy>y & yy<y+sysize) {
dx = 0;
dy = 0;
mousex = -1;
}
}

x+=dx*movex;
y+=dy*movey;

if (y<=borderwidth || y>=(d.height-sysize-scoreheight))
{
dy=0;
y=oldy;
}
if (x>=(d.width-borderwidth-sxsize) || x<=borderwidth)
{
dx=0;
x=oldx;
}
}

public void FireGun()
{
int n=0, f=-1;
while (n<blevel*10 && bx[n]>=0) n++;
if (n<blevel*10) f = n;
if (f>=0) {
bx[f] = x+( (sxsize-bul_xs) / 2);
by[f] = y;
bcur--;
blast.play();
}
}

```

```

    }
}

public void KillEmAll()
{
    int f;
    for (int n=0; n<blevel*10; n++) {
        if (bx[n]>0) {
            by[n] -= bmy;
            if ( by[n] < borderwidth | MetHit(n) | BirdHit(bx[n], by[n]) ) {
                bx[n] = -1;
            } else {
                goff.drawImage(bullet, bx[n], by[n], this); // paint bullet
            }
        }
    }
}

public boolean MetHit(int f)
{
    for (int n=0; n<maxmet; n++) {
        if (metx[n]>=0) {
            if (metr[n] & bx[f]+bul_xs>metx[n] & bx[f]<metx[n]+sxmet & by[f]+bul_ys>mety[n] &
            by[f]<mety[n]+symet) {
                DelMeteor(n);
                kill.play();
                return true;
            }
        }
    }
    return false;
}

public void ShowMeteors()
n {
    int n;
    mtotal = 0;
    for (n=0; n<maxmet; n++) {
        if (metx[n]>=0) {
            mtotal++;
            mety[n] += metmy;
            if (mety[n] > d.height-borderwidth-scoreheight) {
                DelMeteor(n);
            } else {
                if (metr[n]) {
                    goff.drawImage(meteor, metx[n], mety[n], this); // paint meteor
                } else {
                    goff.drawImage(boom[bframes-metf[n]], metx[n]+(sxmet-sxbom)/2, mety[n]+(symet-
                    sybom)/2, this); // paint boom
                    metf[n]--;
                    if (metf[n]<0) DelMeteor(n);
                }
            }
        }
    }
}

```

```

    }
    }
}

public void NewMeteor()
{
    int n=0, f=-1;
    metcount++;
    if (metcount > mrenew/metmy) {
        metcount = 0;
        while (n<maxmet & metx[n]>=0) n++;
        if (n<maxmet) f = n;
        if (f>=0) {
            metx[f] = (int) (Math.random() * (xSize - sxmet) + 1);
            mety[f] = borderwidth-symet;
            metr[f] = true;
            metf[f] = bframes;
        }
    }
}

// If a star in the background reaches the bottome then it will go back to the top
public void moveStars () {
    for (int i = 0; i < numStars; i++) {
        if (starsY[i] + 1 > ySize - (speed * 2)) {
            starsY[i] = 0;
            starsX[i] = (int) ((Math.random() * xSize - 1) + 1);
            starsC[i] = NewColor();
        }
        else {
            starsY[i] += speed;
        }
    }
}

public void Collisions()
{
    for (int n=0; n<maxmet; n++) {
        if (metx[n]>=0) {
            if (metr[n] & x+sxsize>metx[n] & x<metx[n]+sxmet & y+sysize>mety[n] & y<mety[n]+symet)
            {
                HitShip();
                DelMeteor(n);
            }
        }
    }
}
}

```

```

public void HitShip()
{
    crash.play();
    shield=maxshield;
    scur--;
    if (scur<0) GameOver();
}

public void DelMeteor(int n)
{
    if (metr[n]) {
        metr[n] = false;
        metf[n] = bframes;
    } else {
        metx[n] = -1;
        metr[n] = true;
        metf[n] = 0;
    }
}

public Color NewColor()
{
    int[] rgb;
    int t;
    rgb = new int[3];
    for (int i=0; i<3; i++) rgb[i] = 0;
    t = (int) (Math.random()*3);
    rgb[t] = (int) (Math.random()*128 + 1) + 127;
    return new Color(rgb[0], rgb[1], rgb[2]);
}

public void run()
{
    long starttime;
    Graphics g;

    Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
    g=getGraphics();

    while(true)
    {
        starttime=System.currentTimeMillis();
        try
        {
            paint(g);
            starttime += 30;
            Thread.sleep(Math.max(0, starttime-System.currentTimeMillis()));
        }
        catch (InterruptedException e)
        {
            break;
        }
    }
}

```

```

    }
    }
}

public void start()
{
    if (thethread == null) {
        thethread = new Thread(this);
        thethread.start();
    }
}

public void stop()
{
    if (thethread != null) {
        thethread.stop();
        thethread = null;
    }
}

// This class handles mouse clicking
public boolean mouseDown(Event e,int xx,int yy)
{
    if (ingame) {
        mousex = xx;
        mousey = yy;
        keyDown(e, 32);
    } else {
        keyDown(e, 'S');
    }
    return true;
}

// Game Start
public void GameStart()
{
    // Set Up Ship variables
    bmax = blevel*blevel;
    bcur = bmax;
    smax = slevel*slevel;
    scur = smax;
    difflev = 3;
    distance=0;
    score=0;
    renew=250;
    for (int n=0; n<maxmet; n++) {
        metx[n] = -1;
        metf[n] = 0;
        metr[n] = true;
    }
}

```

```

metcount=0;
metmy=2;
mrenew=60;

// Bosses init
// #1 - SunBird;
sbx = -1;
sbmove = 2;
maxtribe = 1;
sunbird=false;
sbefore = true;
safter = false;
}

// Game Over
public void GameOver()
{
    ingame=false;
}

// Boss #1 - Sunbird's pack
public void SunBird()
{
    int[] xcur, ycur;
    xcur = new int[11];
    ycur = new int[11];
    if (sbx<0) {
        sbx = (int) ((Math.random() * xSize - 40) + 1);
        sby = -5;
        sbefore = true;
        safter = false;
    }
    sby += sbmove;
    if (y+sysize/2<sby) safter = true;
    goff.setColor(new Color(0xffff00));
    if (sbefore & safter) {
        // hit ship
        goff.fillRect(0, sby+15, xSize, 2);
        HitShip();
    }
    for (int i=0; i<11; i++) {
        xcur[i] = sbfx[i] + sbx;
        ycur[i] = sbfy[i] + sby;
    }
    goff.fillPolygon(xcur, ycur, 11);
    if (sby>xSize+20) {
        sbx=-1;
        sbefore = true;
        safter = false;
    }
    sbefore=false;
    if (y+sysize/2>sby) sbefore = true;
}

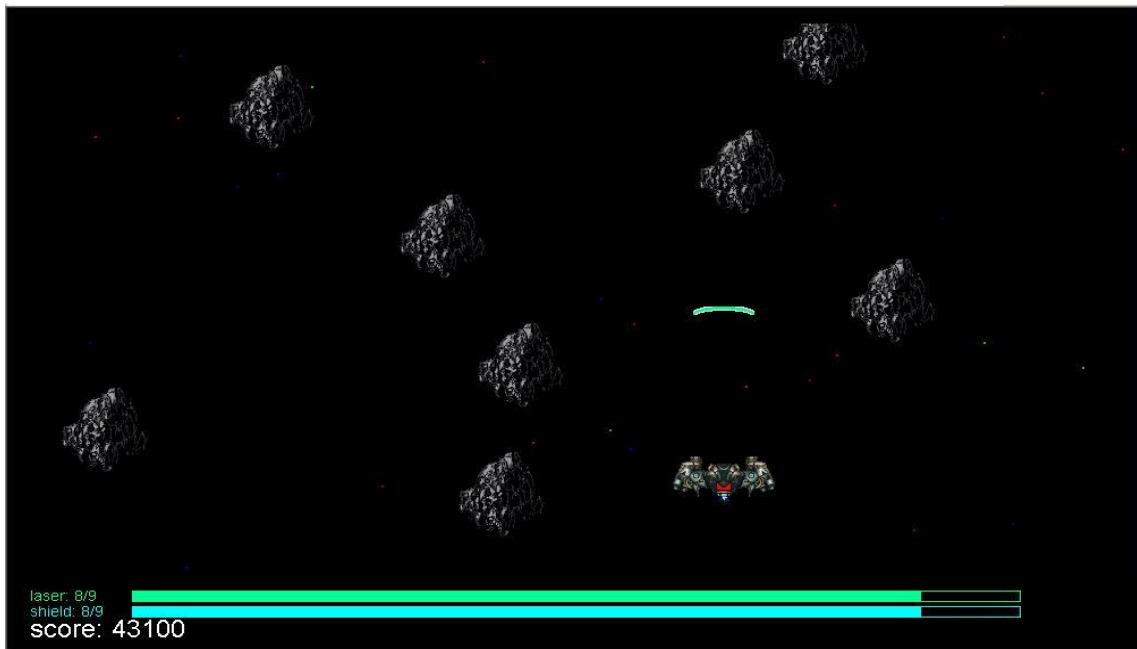
```

```
}  
  
public boolean BirdHit(int blx, int bly) {  
    if (sunbird) {  
        if (blx+bul_xs>sbx & blx<sbx+40 & bly+bul_ys>sby & bly<sby+20) {  
            tribe--;  
            if (tribe<0) sunbird=false;  
            sbx=-1;  
            sbefore = true;  
            safter = false;  
            return true;  
        }  
    }  
    return false;  
}  
  
}
```

TESTING

The game was tested on many java compliant machines and was found to work in them without a hitch.

During the process of testing we found that the game was really addictive and ran without any errors the game also had some very interesting features, which are appreciated by one and all. The screen shot of the game is shown below.



BIBLIOGRAPHY

The following references provided essential domain concepts and fundamentals required for implementing the game in java.

- ✓ Java 2 – The complete reference
-by Herbert Schildt
- ✓ Peter Norton's Guide to Java Programming (e-book)
-by Peter Norton & Wiliam Stanek
- ✓ 2D Graphics (e-book)
-by Deborah Adair

END
