MINI PROJECT

**INDEX**

1. Introduction 4
2. About the project workplace 5
3. Problem Definition 6
4. Analysis 7
5. Design 9
6. Implementation 13
7. I/O Screens 26
8. Conclusion 30
9. Reference 31

**INTRODUCTION**

We live in a computer generation and games are one of the major things for relaxation and entertainment. Among all the genres, action genre has got its own prominence. And games that involve shooting are on a run these days.

Among all the languages, Java is the most used language. It is also easy with so many features like packages, interfaces through which it supports a lot of an in-built facility of methods which we can import.

So we have come up with a small game using Java language and its description is below.

This is a simple shooting game written in JAVA language with the help of AWT. The aim of this game is to shoot persons who appears to fall from above.

Here, in this game, a person (blue in colour) stands left to the screen, holding a gun. Some other persons(black in colour) appear to fall from above. Blue person shoots them. If the bullet hits the person the black person dies.

As long as the blue person shoots and the bullet hits the black person, the blue person can paddle on with the game.

If the blue person misses out the bullets on the black person five times, then the game is over.

This is just a simple game where we need to concentrate on a line of sight i.e., a target line, so as to hit the target.

**ABOUT THE PROJECT WORKPLACE**

Most of the work is generally done in three places in the college.

1. Department of Computer Science and Engineering, UCEOU
2. Diamond Jubilee Library, UCEOU
3. Technology Development Center, UCEOU

Both the places really helped a lot by providing the serene and peaceful atmosphere required for the work to be done. Wi-Fi facility is available to go through the various websites and articles across the web to gather the right information regarding theoretical and coding concepts. All the basic facilities like charging/Plug-in facilities for the laptops and mobile phones are provided. Moreover availability of these places on all the days of the week except for Sunday made it even more flexible to the students. All the minor works, finishing touches (i.e., Secondary tasks) for the report and also the preparation required for the respective topic is done mostly at home.

**PROBLEM DEFINITION**

This is a simple game written in JAVA using AWT. The main aim of this game is to shoot the target as many times as possible.

In this game, the blue person is the shooter and he has to shoot the black people falling from top of the screen. When the bullet hits the black person, the person dies. Like this he has to shoot as many black people as possible. As long as the person shoots and the bullets hit the black person the blue person can paddle on with the game. If the blue person misses 5 times in shooting the target, then the game is said to be over. At the end of the game your score will be displayed on the scoreboard.

This game is a simple game where one needs a proper timing, concentration in mastering it.

**ANALYSIS**

* **ArrayLists** are used to create the number of objects falling down. In the code, four ArrayLists are used.
* **Random** class is used and an object is created for that class to invoke the method “**nextInt”,** which helps in making the objects fall in random trend/order.
* **Threads** are used to alter the time gap between each falling and object and also to alter the pace of falling object. Initially, when the score is low, the pace of objects is slow but the pace increases when the score increases.
* **currentThread( ) and getName()** methods are used to point a particular thread and to name it
* **MouseListener** interface is implemented and the methods **mouseEntered(), mouseExited(), mousePressed(), mouseReleased()** are used in order to send the information through mouse to the System to perform the actions.
* **mouseClicked()** is used to launch bullets and the bullets are added using **add()** method with size specification.
* **get(w)** is used to vary the positions along Y-axis.
* All the **Flags** are set to false initially, whenever a hit occurs, flag will be set to true.
* Multiple flags are being used and if anyone flag is set to **TRUE,** then the score will be incremented.
* If the misses are more than **Five**, then the game will be over.
* **JOptionPane** class and **showMessagedialog() method** are used to display a message in a dialog box.
* **ObjectOutputStream** from file streams is used to write the status of object into a text file.
* **dispose()** method is used to dispose a Frame in the game i.e., frame will be altered and we can switch between frames.
* **getImage() and drawImage()** are used to place any particular image into a certain frame with desired Co-ordinates.
* **setEditable** method’s parameter is given False because High Score cannot be changed by user, it changes only when a new High Score is achieved.
* **ImageIcon** class is used to load images into the frame.
* **setTitle(), setLayout(), setVisible(), setSize(), setBackground()** comes in handy insetting the title/caption, layout, visibility mode, size and background for the game frame.

**Game Operation Analysis**

* The User needs a Mouse and Keyboard for playing this game. In Mouse, Left Click Button(LCB) is used where as the Right Click Button is not used. Coming to the Keyboard, we make use of “Tab”, “Space bar” and “Enter” keys Effectively.

S**ignificance of LCB:**

1. LCB helps the user to shoot the targets in the game.
2. No other key/Button can be used instead of this one.

**Significance of Keys:**

1. Once the game is started, we make use of “Tab” key to toggle/Navigate between the options displayed.
2. “Space bar” is used to select an option among the displayed ones.
3. When the game is finished, user can make use of all the special characters, numbers and alphabets to enter a name.
4. Only “Enter key” is used to save the name.
5. In any other Intermediate Dialog boxes Space bar or Enter can be used to select an option.

**DESIGN**

Some of the methods and classes used in designing this program are:

1.java.util.Random- Here Random class is used. An instance of this class is used to generate a stream of pseudorandom numbers. The nextInt(int n) is used to return a pseudorandom value between 0 and the specified value

2. TreeMap<Integer,Integer>: A TreeMap contains values based on the key. It implements the NavigableMap interface and extends the AbstractMap class. It contains only unique elements and cannot have null key. It arranges the elements in ascending order.

lastkey()- used to return the highest key currently in this map

remove(Object key)- used to remove the mapping from this key from the treemap.

3. ArrayList<Integer>: The ArrayList class extends AbstractList and implements List interface. ArrayList supports dynamic arrays which can grow as needed.

int size()-returns the number of elements in the list.

get(int index)-returns the element at the specified index.

remove(int index)-removes the element at the specified position in the list.

set(intindex,E element)-replaces the element at the specified position in the list with the specified element.

4. Thread: A thread is basically a lightweight subprocess, a smallest unit of processing.

Runnable : The Runnable interface should be implemented by any class whose instances are to be executed by thread.

Thread.sleep()- used to sleep a thread for a specified amount of time.

Thread.start()-starts the execution of the thread.JVM calls the run() method.

Thread.getname()- returns the name of the thread.

Thread.run()- used to perform action for a thread.

Thread.setname(String name)- changes the name of the thread.

5. MouseListener: The Java MouseListener is notified whenever you change the state of the mouse. These are the 5 methods present:

1. **public** **abstract** **void** mouseClicked(MouseEvent e);
2. **public** **abstract** **void** mouseEntered(MouseEvent e);
3. **public** **abstract** **void** mouseExited(MouseEvent e);
4. **public** **abstract** **void** mousePressed(MouseEvent e);
5. **public** **abstract** **void** mouseReleased(MouseEvent e);
6. dispose(): It is used in closing the current frame window without closing the VM.
7. paint(Graphics g): When an applet starts executing, **paint( Graphics g )** is automatically called. It paints the screen.

setimage()-sets the image according to the respective object

1. **public abstract booleandrawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
2. getImage(URL u, String image){} : Used to return the object of the image.
3. ImageIcon(“xy.png”): An implementation of the Icon interface that paints icons from images. Images that are created from URL,Filename are preloaded using MediaTracker to monitor the loaded state of the image.
4. WindowListener: The listener interface for receiving window events. The class that is interested in processing a window event either implements this interface or extends the abstract *WindowAdapter* class.
5. ActionListener: You implement an action Listener to define what should be done when the user performs a certain action.
6. ‘this’- It can be used to refer to current class instance variable.

It can be used to invoke current class method implicitly.

this() can be used to invoke current class constructor.

1. Java.awt.Frame.\*: The Frame is the container that contains title bars and can have menu bars. It can have other components like textfield,buttons etc.

setbounds(intx,inty,intwidth,int height)- It is used to specify the postion and size of the GUI component.

TextField()- The object of a TextField class is a text component that allows the editing of a single line of text.

setsize(intwidth,int height)- sets the size of the component

setLayout(Layout m)- defines the Layout manager for the component

1. catch(Exception e): It involves declaring the type of exceptions you are trying to catch.
2. Serialization: It is a mechanism where an object can be represented as a sequence of bytes which include objects data as well as information about objects type.

Serializable()- It is a marker interface which is used to “mark” java classes so that objects of these classes may get certain capability.

ObjectOutputStream- This class is used to write primitive data types and Java objects to an output stream. Only objects that support java.io.Serializable interface can be written to streams.

ObjectInputStream- It deserializes objects and primitive data using ObjectOutputStream.

**IMPLEMENTATION**

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.util.\*;

public class Start extends Frame implements ActionListener ,Serializable {

String third,second,first;

Integer firs,secon,thir;

TextField tf1,tf2,tf3;

Start(){

setVisible(true);setSize(1000,1000);

setLayout(null);

Button b=new Button("StartGame");b.setBounds(300,300,60,30);add(b);

Button b1=new Button("HighScore");b1.setBounds(450, 300, 60, 30);add(b1);

Button b2=new Button("QuitGame");b2.setBounds(600, 300, 60, 30);add(b2);

Label l1=new Label("1st Highest");l1.setBounds(300, 500, 90 , 30);add(l1);

Label l2=new Label("2nd Highest");l2.setBounds(300, 550, 90 , 30);add(l2);

Label l3=new Label("3rd Highest");l3.setBounds(300, 600, 90 , 30);add(l3);

tf1=new TextField("-");tf1.setBounds(430,500,80,30);add(tf1);tf1.setEditable(false);

tf2=new TextField("-");tf2.setBounds(430, 550, 80, 30);add(tf2);tf2.setEditable(false);

tf3=new TextField("-");tf3.setBounds(430, 600, 80, 30);add(tf3);tf3.setEditable(false);

b.addActionListener(this);

b1.addActionListener(this);

b2.addActionListener(this);

addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{System.exit (0);

}

});

}

public void actionPerformed(ActionEvent ae) {

String str=ae.getActionCommand();

if(str.equals("QuitGame")){

System.exit(0);

}

else if (str.equals("StartGame")){

new Trial();

this.dispose();

}

else

{

try{

ObjectInputStream ois =new ObjectInputStream(new FileInputStream("temp.txt"));

@SuppressWarnings("unchecked")

TreeMap<Integer,String> ts=(TreeMap)ois.readObject();

tf1.setText(ts.lastKey().toString());

ts.remove(ts.lastKey());

if(ts.size()!=0)

{

tf2.setText(ts.lastKey().toString());

ts.remove(ts.lastKey());

}

if(ts.size()!=0)

{

tf3.setText(ts.lastKey().toString());

ts.remove(ts.lastKey());

}

}

catch(Exception e){}

}

}

public static void main(String args[]){

Start t=new Start();

}

}

import java.util.\*;

import java.util.Random;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.\*;

public class Trial extends Frame implements Runnable,MouseListener,Serializable

{

TreeMap<Integer,Integer> tmap=new TreeMap<Integer,Integer>();

ArrayList<Integer> a500 =new ArrayList<Integer>();

ArrayList<Integer> a550 =new ArrayList<Integer>();

ArrayList<Integer> a600 =new ArrayList<Integer>();

ArrayList<Integer> a650 =new ArrayList<Integer>();

ArrayList<Integer> bullet=new ArrayList<Integer>();

public TreeMap<Integer,String> ts=new TreeMap<Integer,String>();

static int score;

ImageIcon a,b,c,d1;

static Boolean flag=true;//set to false once game is over

public boolean flag1=false,flag2=false,flag3=false,flag4=false,flag11=false,flag22=false,flag33=false,flag44=false;

static int i,top;

int miss=0;

static TextField l;

static int random()

{

Random tem=new Random();

return(tem.nextInt(4));

}

public void run()

{String s;

if(Thread.currentThread().getName()=="generation"){

while(flag)

{try{

if(score<10)

Thread.sleep(5000);

else

{if(score<50)

Thread.sleep(2000);

else

Thread.sleep(1000);

}

}

catch(InterruptedException e)

{}

int k1=Trial.random();

switch(k1)

{

case 0:a500.add(0);

break;

case 1:a550.add(0);

break;

case 2:a600.add(0);

break;

case 3:a650.add(0);

break;

}

}

}

if(Thread.currentThread().getName()=="painting"){

while(flag){

repaint();

}

}

}

public void mouseClicked(MouseEvent me)

{

bullet.add(100);

for(int w=0;w<a500.size();w++){

if(400<=a500.get(w)&&a500.get(w)<=440)

{

flag1=true;flag11=true;

a500.set(w,a500.get(w)+1);

}

}

for(int w=0;w<a550.size();w++){

if(390<=a550.get(w)&&a550.get(w)<=430){

flag2=true;flag22=true;

a550.set(w,a550.get(w)+1);

}

}

for(int w=0;w<a600.size();w++){

if(380<=a600.get(w)&&a600.get(w)<=420){

flag3=true;flag33=true;

a600.set(w,a600.get(w)+1);

}

}

for(int w=0;w<a650.size();w++){

if(370<=a650.get(w)&&a650.get(w)<=410){

flag4=true;flag44=true;

a650.set(w,a650.get(w)+1);

}

}

if(flag1||flag2||flag3||flag4)

score++;

else miss++;

flag1=false;flag2=false;flag3=false;flag4=false;

if(miss==5){

flag=false;

JOptionPane.showMessageDialog(null,"Game Over!");

String q=JOptionPane.showInputDialog("Enter your Name");

int i=JOptionPane.showConfirmDialog(null,"Do you Want to continue the game?");

if(i==0)

{

try{

ts.put(score,q);

ObjectOutputStream oos= new ObjectOutputStream(new FileOutputStream("temp.txt"));

oos.writeObject(ts);

oos.close();

}

catch(Exception e){}

new Start();

this.dispose();

}

else if(i==1){

try{

ts.put(score,q);

ObjectOutputStream oos= new ObjectOutputStream(new FileOutputStream("temp.txt"));

oos.writeObject(ts);

oos.close();

}catch(Exception e){}

this.dispose();}

}

}

public void mouseEntered(MouseEvent me){}

public void mouseExited(MouseEvent me){}

public void mousePressed(MouseEvent me){}

public void mouseReleased(MouseEvent me){}

public void paint(Graphics g){

{

g.drawImage(b.getImage(),10,489, this);

for(int ij=0;ij<a500.size();ij++){

if(a500.get(ij)>700)

a500.remove(ij);

else{

if(((a500.get(ij)%10)!=0)&&(a500.get(ij)>500))

{

if(flag11)

{

Integer rea=score;

l.setText(rea.toString(score));}

g.drawImage(c.getImage(),500,a500.get(ij),this);

flag11=false;

}

else g.drawImage(a.getImage(),500,a500.get(ij),this);

a500.set(ij,a500.get(ij)+10);

}

}

for(int ij=0;ij<a550.size();ij++){

if(a550.get(ij)>700)

a550.remove(ij);

else{

if(((a550.get(ij)%10)!=0)&&(a550.get(ij)>500))

{

if(flag22){

Integer rea=score;

l.setText(rea.toString(score));

}

g.drawImage(c.getImage(),550,a550.get(ij),this);

flag22=false;

}

else g.drawImage(a.getImage(),550,a550.get(ij),this);

a550.set(ij,a550.get(ij)+10);

}

}

for(int ij=0;ij<a600.size();ij++){

if(a600.get(ij)>700)

a600.remove(ij);

else{

if(((a600.get(ij)%10)!=0)&&(a600.get(ij)>500))

{

if(flag33 ){

Integer rea=score;

l.setText(rea.toString(score));}

g.drawImage(c.getImage(),600,a600.get(ij),this);

flag33=false;

}

else

g.drawImage(a.getImage(),600,a600.get(ij),this);

a600.set(ij,a600.get(ij)+10);}

}

for(int ij=0;ij<a650.size();ij++){

if(a650.get(ij)>700)

a650.remove(ij);

else{

if(((a650.get(ij)%10)!=0)&&(a650.get(ij)>500))

{

if(flag44){

Integer rea=score;

l.setText(rea.toString(score));}

g.drawImage(c.getImage(),650,a650.get(ij),this);

flag44=false;

}

else

g.drawImage(a.getImage(),650,a650.get(ij),this);

a650.set(ij,a650.get(ij)+10);}

}

for(int d=0;d<bullet.size();d++){

g.drawImage(d1.getImage(),bullet.get(d),500,this);

bullet.set(d,bullet.get(d)+50);

}

try{Thread.sleep(100);}catch(Exception e){}

}

}

Trial()

{

flag=true;

b=new ImageIcon("shoot.png");

a=new ImageIcon("scroll.png");

c=new ImageIcon("die.png");

d1=new ImageIcon("bullet.png");

setTitle("Game");

setLayout(null);

setVisible(true);

setSize(1000,1000);

setBackground(Color.blue);

Thread mythread =new Thread(this,"generation");

Thread pain=new Thread(this,"painting");

mythread.start();

pain.start();

l=new TextField("0");l.setBounds(100,600,50,20);

l.setEditable(false);

add(l);

addMouseListener(this);

addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit (0);

}

});

}

public static void main(String a[])

{

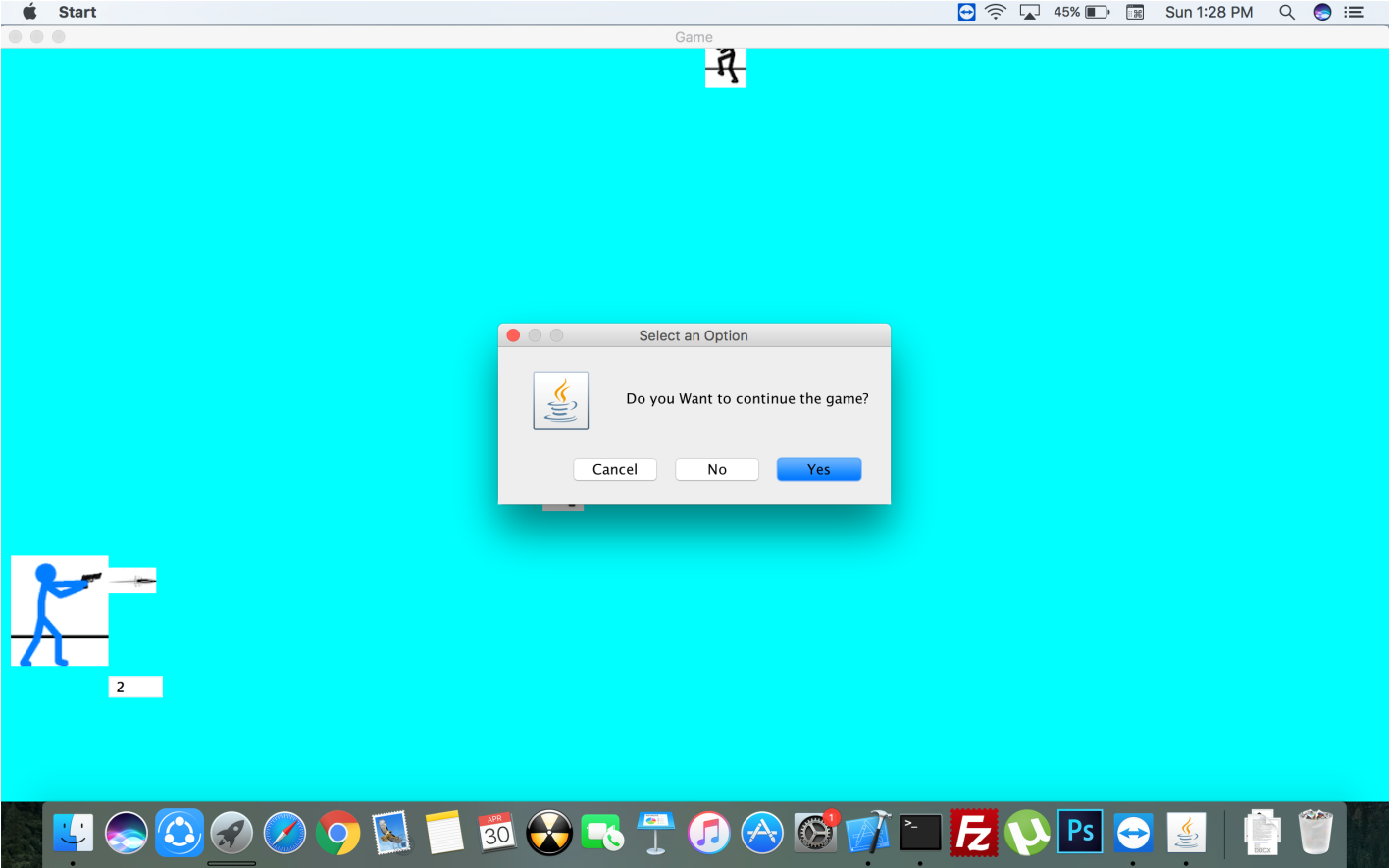
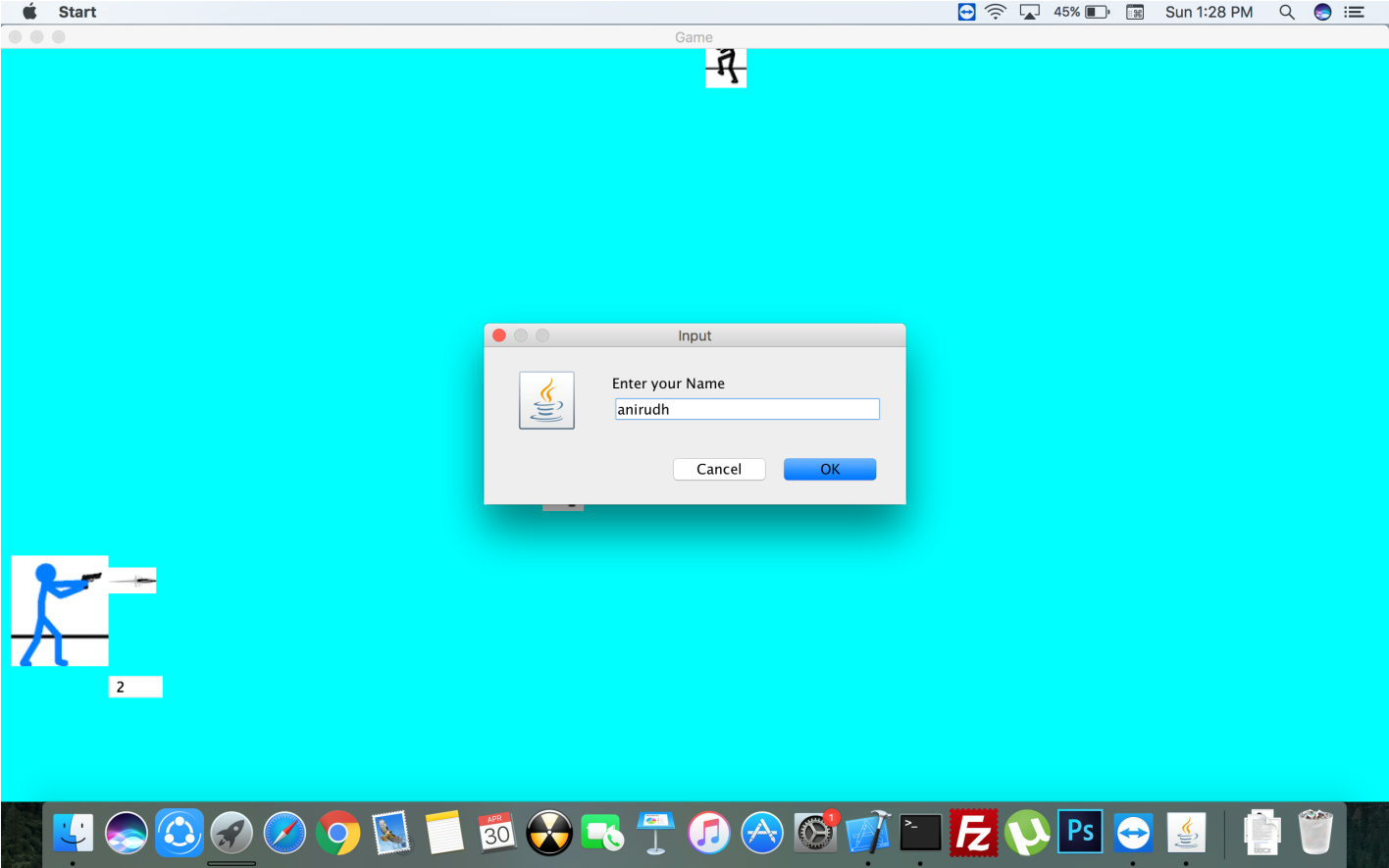
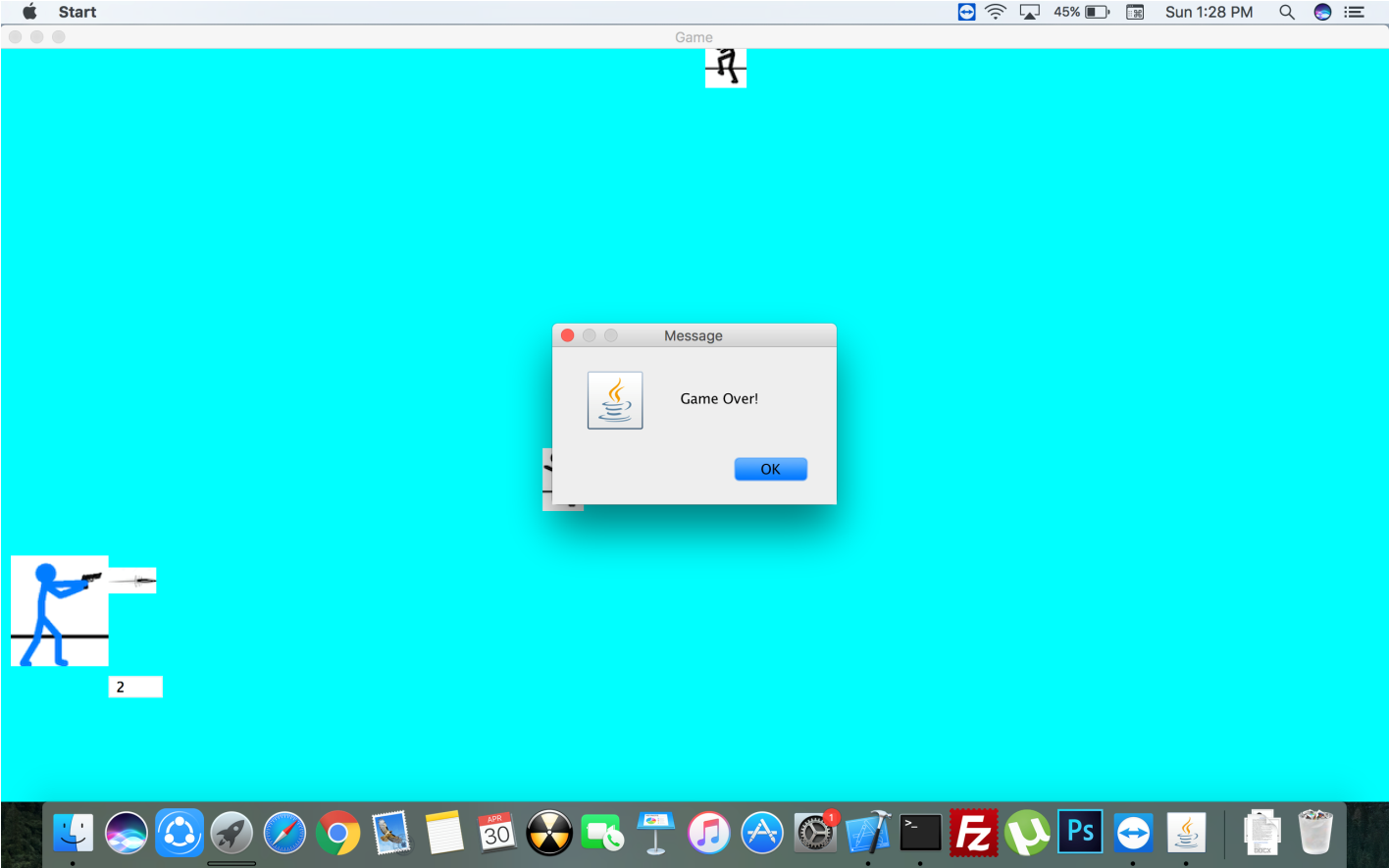
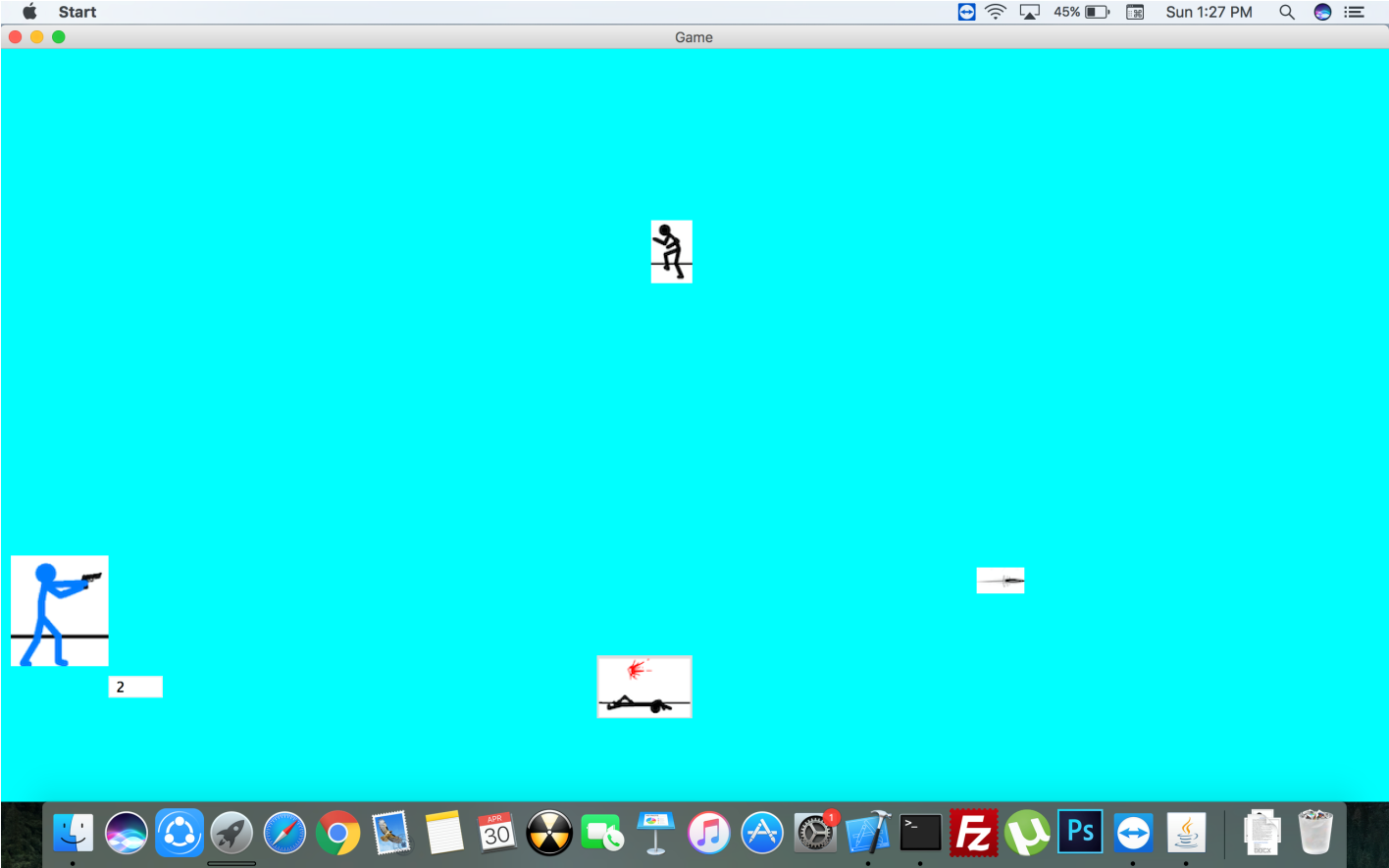
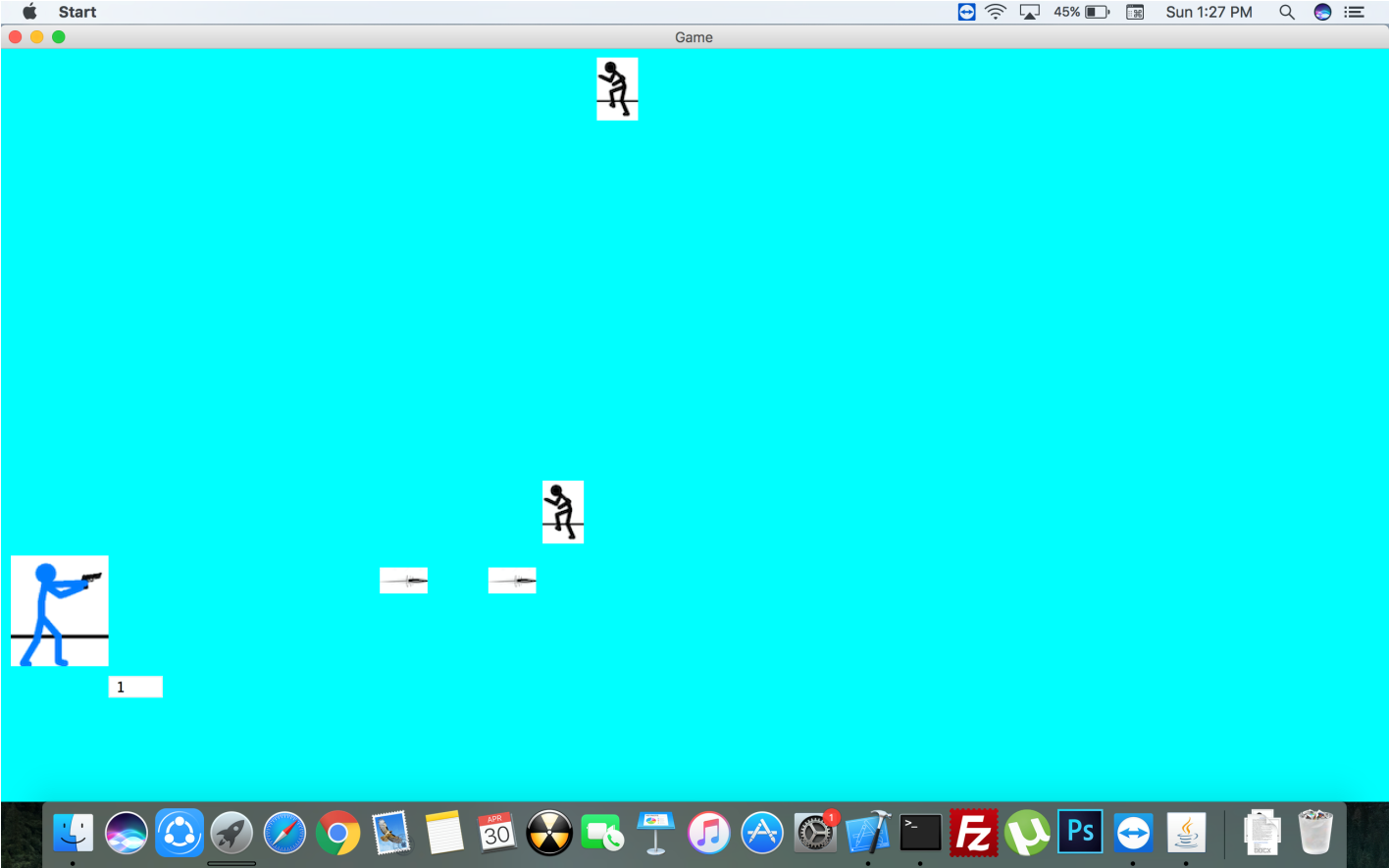
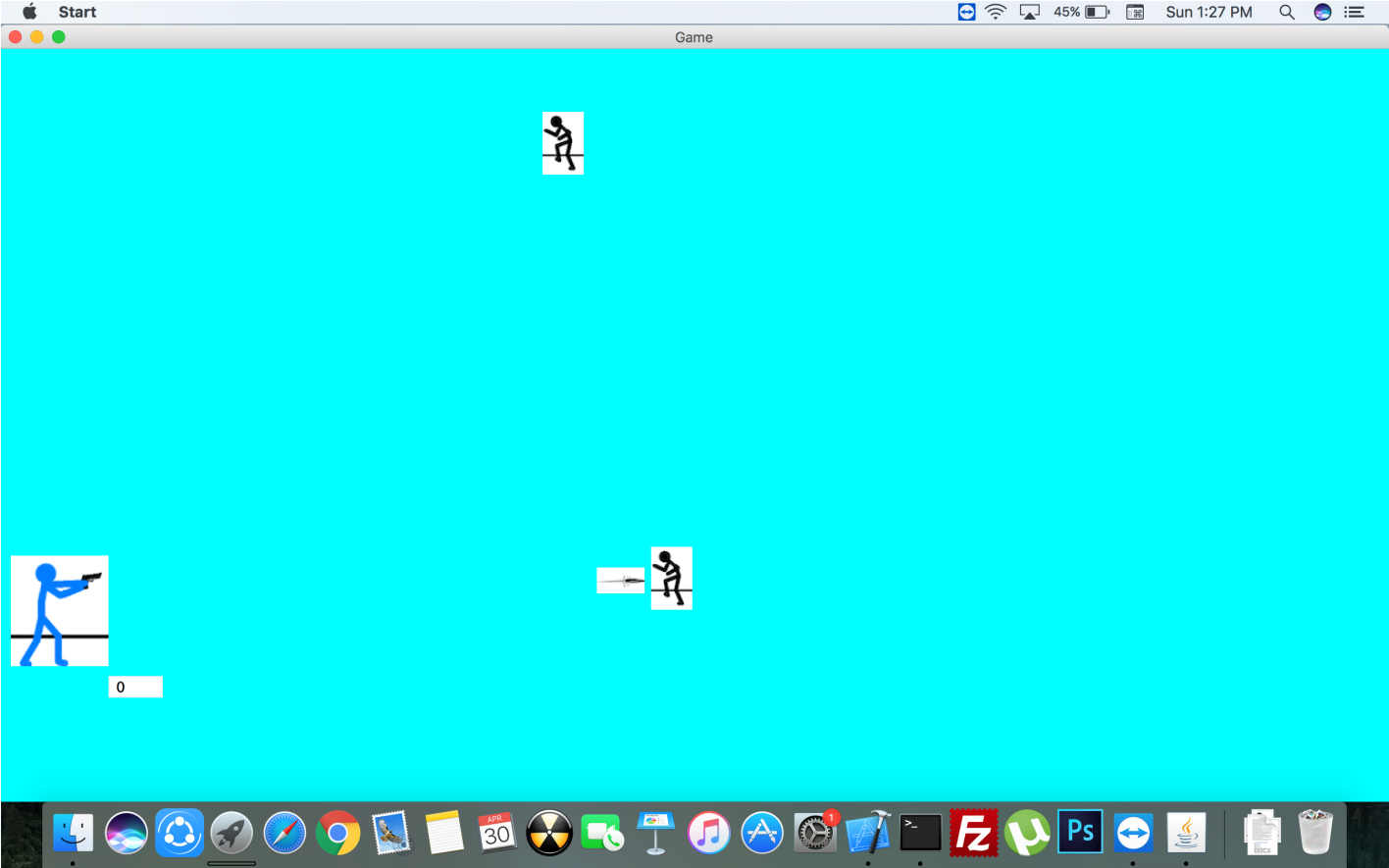
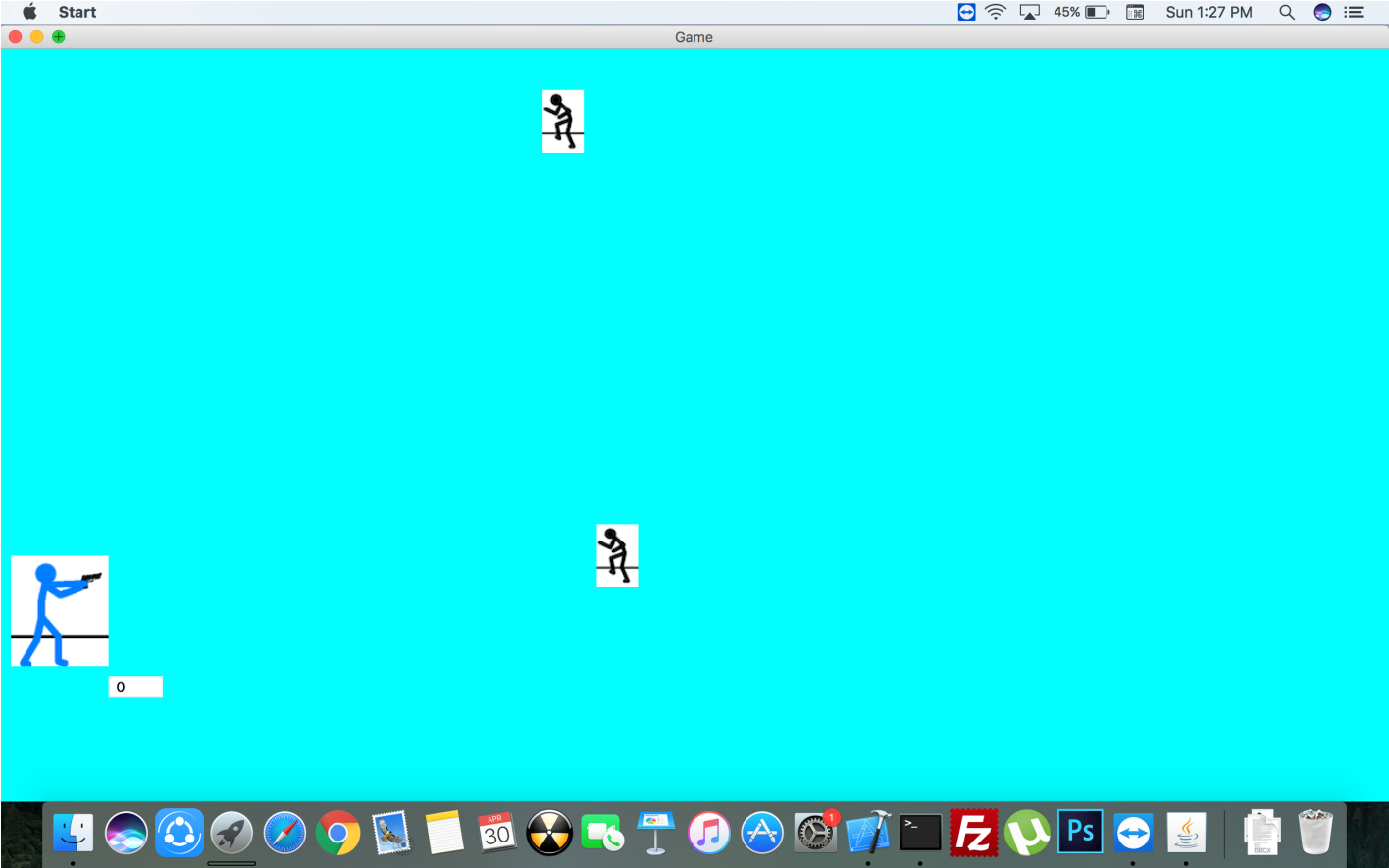
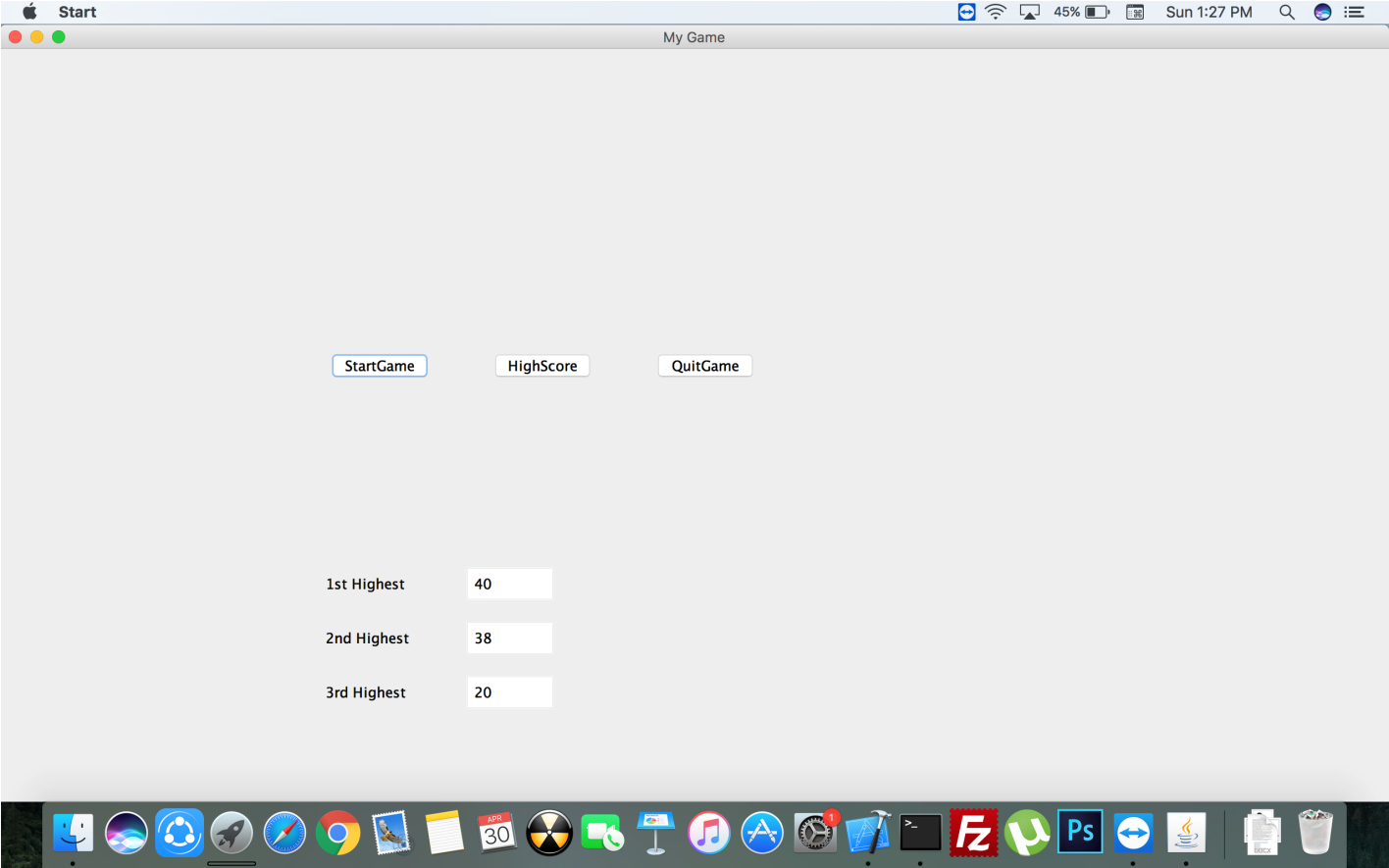
Trial t=new Trial();

while(flag)

i=random();

}

}

**I/O Screens**

**CONCLUSION**

In General, A Game is one of the best way to remove Stress in this present scenario. Shooter games are subgenre of action game, which often test the player’s speed and reaction time. It includes many subgeners that have the commonality of focusing on the actions of the avatar using some sort of weapon. Usually this weapon is a Gun.

Presently, The Game is with low graphics and needs some external features for Display paint to fill the empty space on the screen,movement of the Shooterover the screen and flexibilty to shoot in angle rather than in perpendicular direction. Change in view from 2D to 3D. saving Highscores in files so that it should be available for next games and Addition of Levels to the game like Easy, Medium, Hard and depending on the level increase in speed of the falling objects. Using methods to make the Game realistic and enhancing the features to Keyboard than limiting it to Mouse.

**REFERENCE**

* Serializable concepts from Java IO Packages extracted from *The Complete Reference JAVA,* Tata McGraw Hill.
* Thread concepts from*The Complete Reference JAVA,* Tata McGraw Hill.
* ActionListener concepts from *https://www.javatpoint.com/java-actionlistener*
* Button concepts from *https://www.javatpoint.com/java-awt-button*
* Label concepts from *https://www.javatpoint.com/java-awt-label*
* Textfield concepts from https://www.javatpoint.com/java-awt-textfield
* WindowListener concepts from https://www.javatpoint.com/java-windowlistener
* Treemap concepts from *The Complete Reference JAVA,* Tata McGraw Hill.
* ArrayList concepts from *The Complete Reference JAVA,* Tata McGraw Hill.
* Runnable Interface concepts from *The Complete Reference JAVA,* Tata McGraw Hill.
* ImageIcon concepts from *https://www.tutorialspoint.com/swing/swing\_imageicon.htm*
* Random number generator concepts from *https://www.tutorialspoint.com/java/util/java\_util\_random.htm*
* Painting in the AWT concepts from *http://www.oracle.com/technetwork/java/painting-140037.html*
* AWT MouseEvent class concepts from *https://www.tutorialspoint.com/awt/awt\_mouse\_event.htm*
* DialogBox concepts from *https://www.javatpoint.com/open-dialog-box*
* ObjectInputStream concepts from *https://www.tutorialspoint.com/java/io/java\_io\_objectinputstream.htm*
* ObjectOutputStream concepts from *https://www.tutorialspoint.com/java/io/java\_io\_objectoutputstream.htm*
* Exceptional Handling concepts from *The Complete Reference JAVA,* Tata McGraw Hill.