Final Report

TITLE: FLAPPY BIRD GAME

Name:AKSHAYA JONNALA

TITLE: FLAPPY BIRD GAME

ABSTRACT:

Flappy Bird is a mobile game that involves tapping the screen to navigate a bird through a gap between pairs of vertical pipes. When the bird passes through the gap, the score increments by one and the game ends when the bird hits the floor or a pipe. Surprisingly, Flappy Bird is a very difficult game and scores in single digits are not uncommon even after extensive practice. In this project we will implement this mobile game using java concepts.

In our game, the image moving is the most important function we need to implement. Besides the background horizontal moving, the vertical jumping combining keyboard input is our second challenge. In the process of the game, how to detect if the object satisfies the game over condition and the condition setting is our third problem.

INTRODUCTION:

Flappy Bird is a game where you tap the screen to make the bird fly. The objective is to direct or operate a flying bird, named Faby, who moves continuously to the right, between sets of Mario-like pipes. If the player touches the pipes, they lose. At each instant there are two actions that the player can take: to press the 'up' key, which makes the bird jump upward or not pressing any key, which makes it descend at a constant rate, Faby falls because of gravity; each pair of pipes that he navigates between earns the player a single point.

Flappy Bird is an arcade-style game in which the player controls the bird, which moves persistently to the right. The player is tasked with navigating Flappy bird through pairs of pipes that have equally sized gaps placed at random heights. Bird automatically descends and only ascends when the player taps the touchscreen. Each successful pass through a pair of pipes awards the player one point. Colliding with a pipe or the ground ends the gameplay. During the game over screen, the player is awarded with increase in points if they moves towards the increasing in the levels of the Game.

ABOUT THE PROJECT:

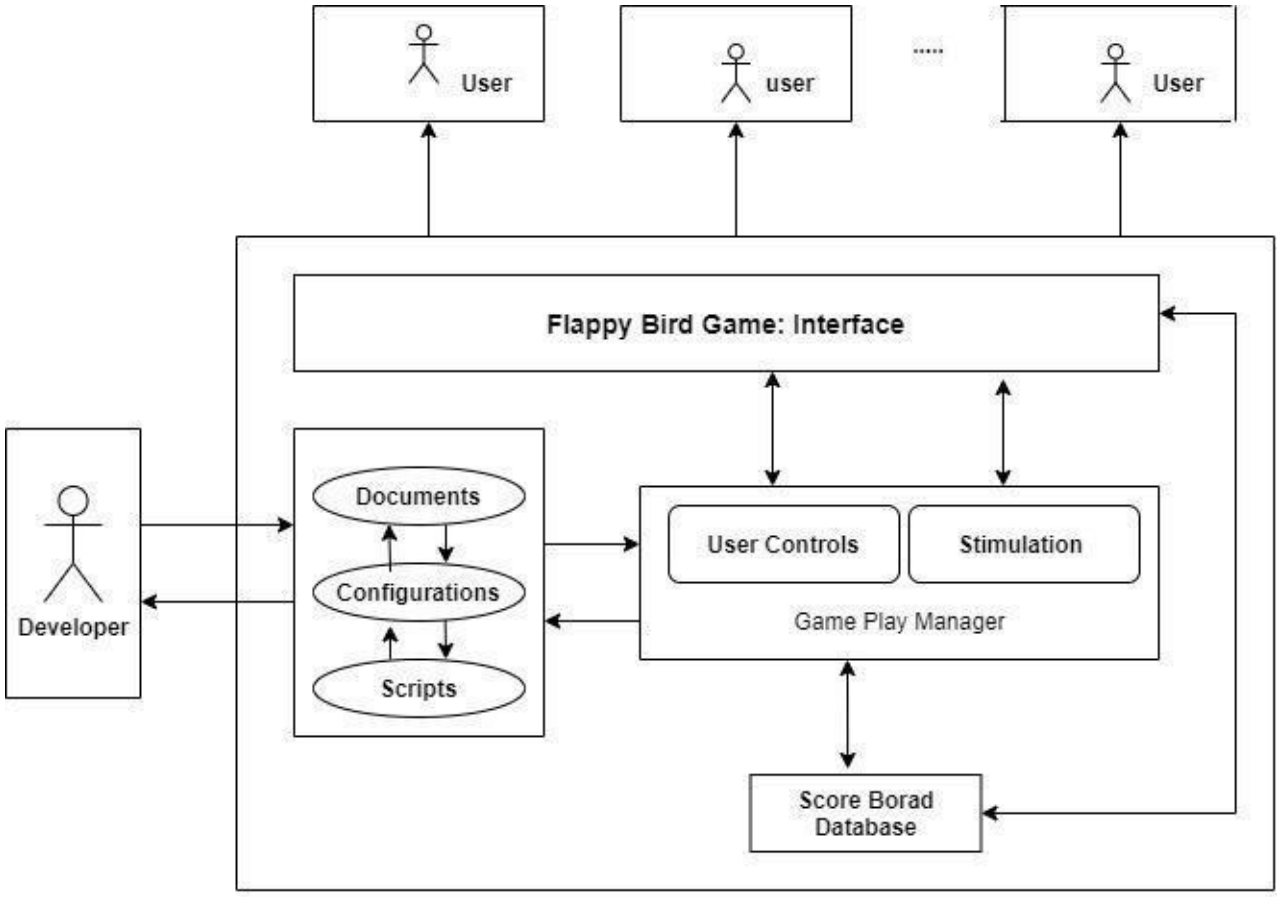
This whole project is accomplished in NetBeans IDE. In order to run this project, you will need an NetBeans IDE. First, download this project (The project ZIP file and the team members project explanation video is uploaded in google drive and link given below) and then extract it. Open your IDE and import this project. Also, before the project keep in mind that you must have your JDK up to date. Once you import your project, then run the project. Warning, do not run the jar files or the compiled files. Also, before you run the project, make sure that you have the server installed for the Netbeans IDE. Then after that, you have to enable the Netbeans connector for browser debugging. Then you can finally run the project. Once you open the game, you can hit the Enter to start the game. The controls for the games is the spacebar to jump. You have to careful with the vertical bar which is the obstacles.

As mentioned earlier, the game is made in Java. This game is truly fun to play. You know how to play this game. Also, this game uses NetBeans swing component for making the game board. This game is well validated.

Sources of the Project: Java and Net-beans IDE / Jdk

SYSTEM DESIGN:

ARCHITECTURE FOR FLAPPY BIRD GAME:

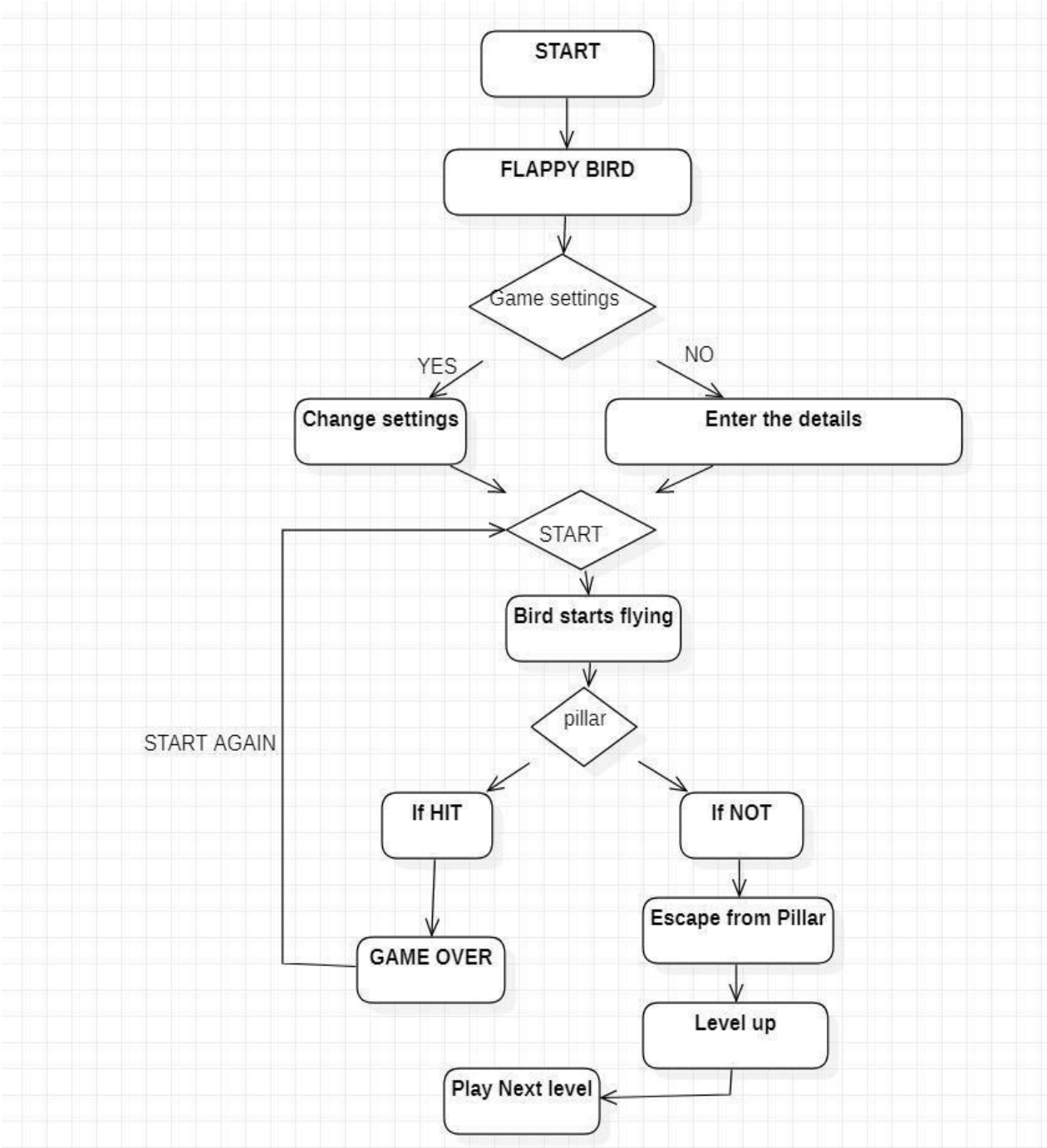


Architecture Design For Flappy Bird Game

- Flappy Bird game architectural diagram is a system that is used to abstract the overall outline system architecture of the flappy bird software system and the relationships between components that are related to the flappy bird game.

FLAPPY BIRD GAME: ACTIVITY FLOW CHART

The following Activity Flow the diagram shows about the overview of game plan of Flappy Bird.



GAME PLAN:

Step 1: First Run the Main file of the project FlappyBird.java, then u will get in To the game.

Step 2: Start game: Tap the key to start. Tap the screen again to allow your bird To fly and to start the game.

Step 3: Stay in the middle of the screen until the first set of pipes to be appear Measure your tap heights to go between the two pipes.

Step 4: We have to escape between the pipes and score increases by 1 point.

Step 5: Stay in the middle of the pipes. This is the main objective of the game. If you hit a pipe or the ground, the game ends.

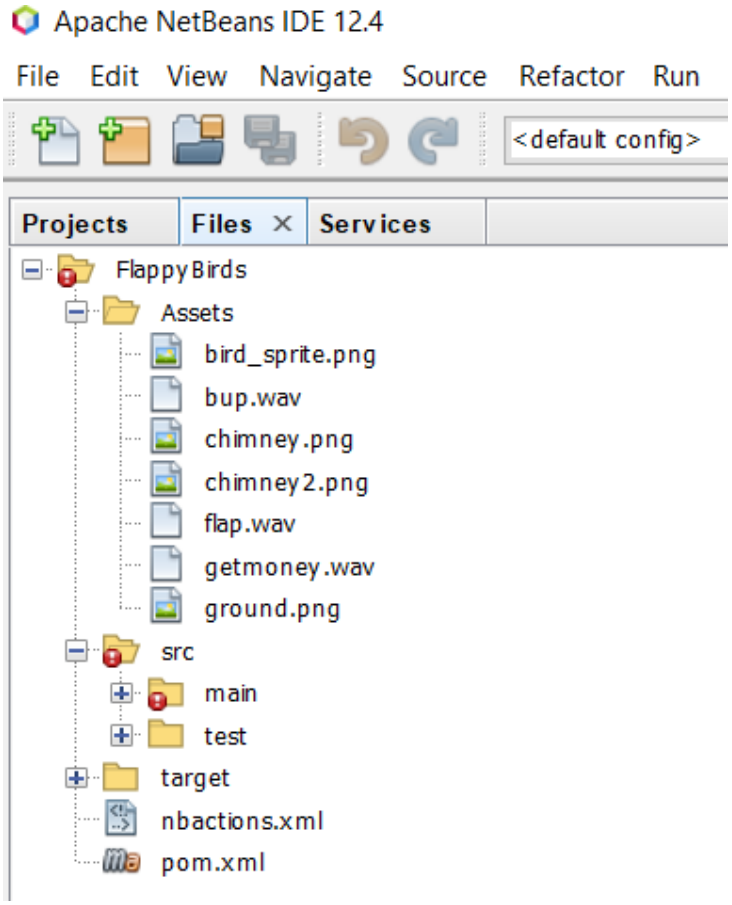
Step 6: Start the game again and follow the steps to score the points.

SOFTWARE REQUIREMENT SPECIFICATIONS:

4.2 Tools Used to Design the Flappy Bird Game

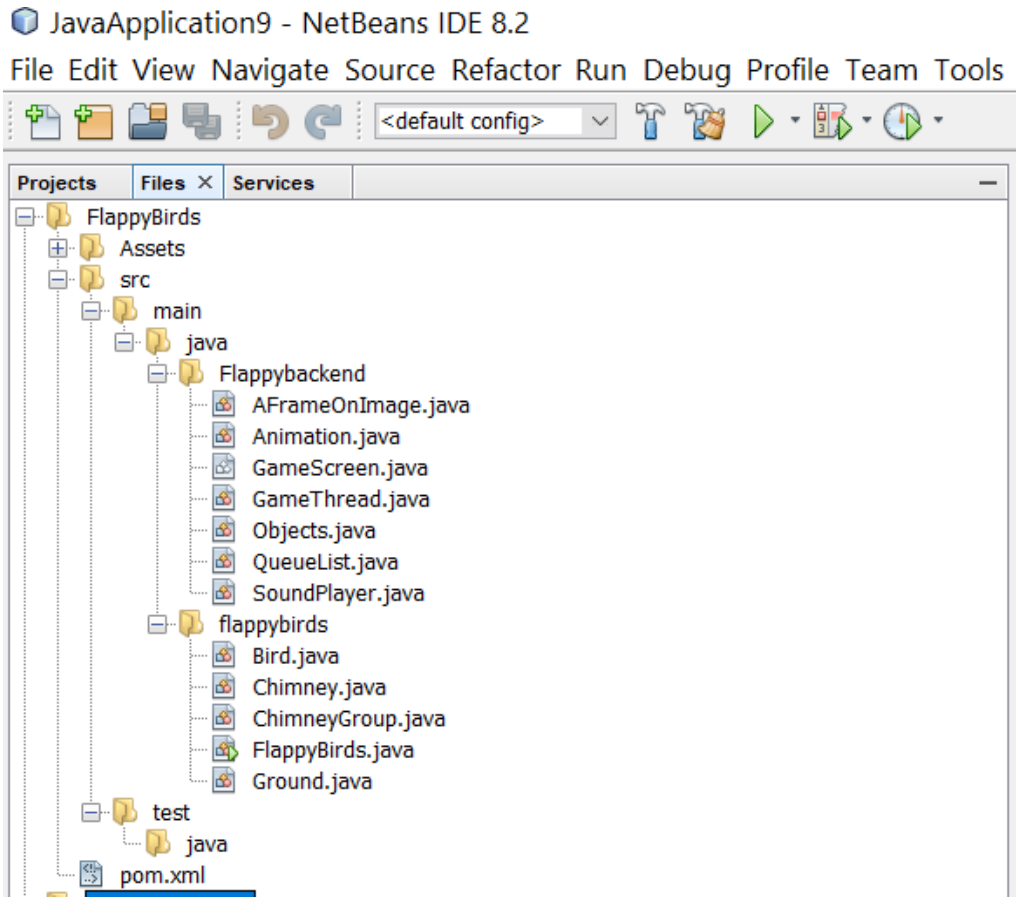
- Language Used Java
- Netbeans IDE with Jdk up to the latest Version
- RAM 4GB minimum and windows 10 operating System

MODULE & PACKAGES INVOLVED:



CLASSES INVOLVED IN THE PROJECT:

- FlappyBird.java
- Bird.java
- Chimney.java
- ChimneyBird.java
- Ground.java
- Animation.java
- Gamescreen.java
- Objects.java
- QueueList.java
- GameThread.java
- AframeOnImage.java
- Sound Player.java



SOURCE CODE:

FlappyBird.java

```
package flappybirds;
import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import Flappybackend.AFrameOnImage;
import  Flappybackend.Animation;
import Flappybackend.GameScreen;
import java.awt.Color;
import java.awt.Font;
public class FlappyBirds extends GameScreen{
    private BufferedImage birds;
    private Animation bird_anim;
    private  BufferedImage chimney;
    public static float g=0.15f;
    private Bird bird;
    private Ground ground;

    private ChimneyGroup chimneyGroup;
    private int Point=0;
    private int BEGIN_SCREEN = 0;
    private int GAMEPLAY_SCREEN = 1;
    private int GAMEOVER_SCREEN = 2;
    private int CurrentScreen = BEGIN_SCREEN;
```

```
public FlappyBirds(){
    super(800,600);
    try{
        birds =ImageIO.read(new File("Assets/bird_sprite.png"));
    }catch(IOException ex){}
    bird_anim = new Animation(70);
    AFrameOnImage f;
    f = new AFrameOnImage(0,0,60,60);
    bird_anim.AddFrame(f);
    f = new AFrameOnImage(60,0,60,60);
    bird_anim.AddFrame(f);
    f = new AFrameOnImage(120,0,60,60);
    bird_anim.AddFrame(f);
    f = new AFrameOnImage(60,0,60,60);
    bird_anim.AddFrame(f);
    bird=new Bird(350,250,50,50);
    ground = new Ground();
    chimneyGroup = new ChimneyGroup();
    BeginGame();
}
public static void main(String args[])
{
    new FlappyBirds();
}
private void resetGame(){
    bird.setPos(350,250);
    bird.setVt(0);
    bird.setLive(true);
    Point=0;
```

```
        chimneyGroup.resetChimneys();
    }
    @Override
    public void GAME_UPDATE(long deltaTime) {
      if(CurrentScreen==BEGIN_SCREEN) {
         resetGame();
      }else if(CurrentScreen ==GAMEPLAY_SCREEN){
          if(!bird.getLive()) bird_anim.Update_Me(deltaTime);
          bird.update(deltaTime);
          ground.Update();
          chimneyGroup.update();
          for(int i=0;i<ChimneyGroup.SIZE;i++){
             if(bird.getRect().intersects(chimneyGroup.getChimney(i).getRect())){
                 if(bird.getLive()) bird.bupSound.play();
                 bird.setLive(false);
                 System.out.println("Set live = false");
             }
          }
           for(int i=0;i<ChimneyGroup.SIZE;i++){
              if(bird.getPosX()>chimneyGroup.getChimney(i).getPosX()&&
!chimneyGroup.getChimney(i).getIsBehindBird()
                    && i%2==0){
                 Point++;
                 bird.getMoneySound.play();
                 chimneyGroup.getChimney(i).setIsBehindBird(true);
              }
           }
          if(bird.getPosY()+bird.getH() > ground.getYGround()) CurrentScreen =
GAMEOVER_SCREEN;
```

```
        }else {
        }
    }
    @Override
    public void GAME_PAINT(Graphics2D g2) {
        g2.setColor(Color.decode("#b8daef"));
        g2.fillRect(0,0,MASTER_WIDTH,MASTER_HEIGHT);
        chimneyGroup.paint(g2);
        ground.Paint(g2);
        if(bird.getIsFlying())
        bird_anim.PaintAnims((int)bird.getPosX(),(int)bird.getPosY(),birds, g2, 0,-
1);
        else
            bird_anim.PaintAnims((int)bird.getPosX(),(int)bird.getPosY(),birds, g2,
0,0);
      ground.Paint(g2);
      Font f1;
        f1 = new Font("Arial",Font.BOLD,20);
      if(CurrentScreen == BEGIN_SCREEN){
        g2.setColor(Color.black);
        g2.setFont(f1);
        g2.drawString("Press space to play game",300,200);
      }
      if(CurrentScreen == GAMEOVER_SCREEN){
          Font f2;
        f2 = new Font("Arial",Font.BOLD,40);
        g2.setColor(Color.black);
        g2.setFont(f2);
        g2.drawString("GAMEOVER",300,250);
```

```
        g2.setFont(f1);
        g2.drawString("YourScore:"+Point, 330, 285);
      }
        g2.setColor(Color.red);
      g2.setFont(f1);
      g2.drawString("Point:"+Point, 20, 50);
    }
    @Override
    public void KEY_ACTION(KeyEvent e, int Event) {
        if(Event == KEY_PRESSED){
            if(CurrentScreen == BEGIN_SCREEN){
                CurrentScreen = GAMEPLAY_SCREEN;
            }else if(CurrentScreen == GAMEPLAY_SCREEN){
              if(bird.getLive())bird.fly();
            }else if(CurrentScreen == GAMEOVER_SCREEN){
                CurrentScreen = BEGIN_SCREEN;
            }
        }
    }
}
```

Bird.java:

```
package flappybirds;
import Flappybackend.Objects;
import Flappybackend.SoundPlayer;
import java.awt.Rectangle;
import java.io.File;
public class Bird extends Objects{
```

```
private float vt = 0;
private boolean isFlying = false;
private Rectangle rect;
private boolean isLive=true;
public SoundPlayer f1apSound, bupSound, getMoneySound;

public Bird(int x, int y, int w, int h){
    super(x, y, w, h);
    rect = new Rectangle(x, y, w, h);
    f1apSound= new SoundPlayer(new File("Assets/flap.wav"));
    bupSound = new SoundPlayer(new File("Assets/bup.wav"));
    getMoneySound = new SoundPlayer(new File("Assets/getMoney.wav"));
}
public void setLive(boolean b){
    isLive=b;
}
public boolean getLive(){
    return isLive;
}
public Rectangle getRect(){
    return rect;
}
public void setVt(float vt){
    this.vt=vt;
}
public void update(long deltaTime){
    vt+=FlappyBirds.g;
    this.setPosY(this.getPosY()+vt);
    this.rect.setLocation((int) this.getPosX(),(int) this.getPosY());
```

```
        if(vt < 0) isFlying = true;
        else isFlying = false;
       }
    public void fly(){
     vt = -3;
     f1apSound.play();
    }
public boolean getIsFlying()
{
    return isFlying;
}
}
```

Ground.java:

```
package flappybirds;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
public class Ground {
    private BufferedImage groundImage;
    private int x1,y1,x2,y2;
    public Ground(){
    try{
        groundImage = ImageIO.read(new File("Assets/ground.png"));
    }catch(IOException ex){}
    x1=0;
    y1=500;
```

```
    x2=x1+830;
    y2=500;
    }
    public void Update(){
        x1-=2;
        x2-=2;
        if(x2<0)x1=x2+830;
        if(x1<0)x2=x1+830;
    }
    public void Paint(Graphics2D g2){
       g2.drawImage(groundImage,x1,y1,null);
       g2.drawImage(groundImage,x2,y2,null);
    }
    public int getYGround(){
        return y1;
    }
}
```

Chimney.java:

```
package flappybirds;
import Flappybackend.Objects;
import java.awt.Rectangle;
public class Chimney extends Objects{
    private Rectangle rect;
    private boolean isBehindBird = false;
    public Chimney(int x, int y,int w,int h){
        super(x,y,w,h);
       rect = new Rectangle(x, y, w, h);
    }
```

```
    public void update(){
        setPosX(getPosX()-2);
        this.rect.setLocation((int) this.getPosX(),(int) this.getPosY());
    }
    public  Rectangle getRect(){
        return rect;
    }
    public void setIsBehindBird(boolean b){
        isBehindBird = b;
    }
    public boolean getIsBehindBird(){
        return isBehindBird;
    }
}
```

ChimneyGround.java:

```
package flappybirds;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Random;
import javax.imageio.ImageIO;
import Flappybackend.QueueList;
public class ChimneyGroup {
    private QueueList<Chimney> chimneys;
    private BufferedImage chimneyImage, chimneyImage2;
    public static int SIZE = 6;
    private int topChimneyY = -350;
```

```
private int bottomChimneyY = 200;
public Chimney getChimney(int i){
    return chimneys.get(i);
}
public int getRandomY(){
    Random random = new Random();
    int a;
    a = random.nextInt(10);
    return a*35;
}
public ChimneyGroup(){
    try {
        chimneyImage = ImageIO.read(new File("Assets/chimney.png"));
        chimneyImage2 = ImageIO.read(new File("Assets/chimney2.png"));
    } catch (IOException ex) {}
    chimneys = new QueueList<Chimney>();
    Chimney cn;
    for(int i = 0; i< SIZE/2;i++){
        int deltaY = getRandomY();
        cn = new Chimney(830+i*300, bottomChimneyY + deltaY, 74, 400);
        chimneys.push(cn);
        cn = new Chimney(830+i*300, topChimneyY+ deltaY, 74, 400);
        chimneys.push(cn);
    }
}
public void resetChimneys(){
    chimneys = new QueueList<Chimney>();
    Chimney cn;
    for(int i = 0; i< SIZE/2;i++){
```

```
        int deltaY = getRandomY();
        cn = new Chimney(830+i*300, bottomChimneyY + deltaY, 74, 400);
        chimneys.push(cn);
        cn = new Chimney(830+i*300,  topChimneyY+ deltaY, 74, 400);
        chimneys.push(cn);
    }
}
public void update(){
    for(int i = 0;i < SIZE; i++){
        chimneys.get(i).update();
    }
    if(chimneys.get(0).getPosX()<-74) {
       int deltaY = getRandomY();
       Chimney cn;
        cn = chimneys.pop();
        cn.setPosX(chimneys.get(4).getPosX() + 300);
        cn.setPosY(bottomChimneyY+ deltaY);
        cn.setIsBehindBird(false);
        chimneys.push(cn);
        cn = chimneys.pop();
        cn.setPosX(chimneys.get(4).getPosX());
        cn.setPosY(topChimneyY + deltaY);
        cn.setIsBehindBird(false);
        chimneys.push(cn);
    }
}
public void paint(Graphics2D g2){
    for(int i = 0;i < 6; i++)
        if(i%2==0)
```

```
            g2.drawImage(chimneyImage, (int )chimneys.get(i).getPosX(),
(int)chimneys.get(i).getPosY(), null);
        else g2.drawImage(chimneyImage2, (int )chimneys.get(i).getPosX(),
(int)chimneys.get(i).getPosY(), null);
    }
}
```

AFrameonImage.java

```
package Flappybackend;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
public class AFrameOnImage {
    private boolean enablePaintRect = false;
    private int []DimsBounds = new int[4];
    public AFrameOnImage(int xOnImage, int yOnImage, int w, int h){
        DimsBounds[0] = xOnImage;
        DimsBounds[1] = yOnImage;
        DimsBounds[2] = w;
        DimsBounds[3] = h;
    }
    public void VisibleRectDebug(boolean enable){
        enablePaintRect = enable;
    }
```

```
    public int[] GetBounds(){
        return DimsBounds;
    }
    public void Paint(int x, int y, BufferedImage image, Graphics2D g2, int
anchor, float rotation){

        BufferedImage imageDraw = image.getSubimage(DimsBounds[0],
DimsBounds[1], DimsBounds[2], DimsBounds[3]);

        AffineTransform tx = new AffineTransform();
        tx.rotate(rotation, imageDraw.getWidth() / 2, imageDraw.getHeight() / 2);
        AffineTransformOp op = new AffineTransformOp(tx,
        AffineTransformOp.TYPE_BILINEAR);
        imageDraw = op.filter(imageDraw, null);
        g2.drawImage(imageDraw, x, y, null);
        if(enablePaintRect) PaintBound(g2);
    }
    private void PaintBound(Graphics2D g){

    }
}
```

Animation.java:

```
package Flappybackend;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
public class Animation {
```

```
private long beginTime = 0;
private long mesure = 20;
private AFrameOnImage[] frames;
private int NumOfFrame = 0;
private int CurrentFrame = 0;
public Animation(long mesure){
   this.mesure = mesure;
}
public void Update_Me(long deltaTime){
   if(NumOfFrame>0){
      if(deltaTime - beginTime > mesure){
         CurrentFrame++;
         if(CurrentFrame>=NumOfFrame)
            CurrentFrame = 0;
         beginTime = deltaTime;
      }
   }
}
public void AddFrame(AFrameOnImage sprite){
   AFrameOnImage[] bufferSprites = frames;
   frames = new AFrameOnImage[NumOfFrame+1];
   for(int i = 0;i<NumOfFrame;i++) frames[i] = bufferSprites[i];
   frames[NumOfFrame] = sprite;
   NumOfFrame++;
}
```

```
    public void PaintAnims(int x, int y, BufferedImage image, Graphics2D g2,
int anchor, float rotation){
        frames[CurrentFrame].Paint(x, y, image, g2, anchor, rotation);
    }
}
```

GameScreen.java:

```
package Flappybackend;
import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.image.BufferedImage;
import javax.swing.JFrame;
public abstract class GameScreen extends JFrame implements KeyListener{
    public static int KEY_PRESSED = 0;
    public static int KEY_RELEASED = 1;
    public int CUSTOM_WIDTH = 500;
    public int CUSTOM_HEIGHT = 500;
    private GameThread G_Thread;
    public static int MASTER_WIDTH = 500, MASTER_HEIGHT = 500;
    public GameScreen(){
        InitThread();
        InitScreen();
    }
    public void RegisterImage(int id, BufferedImage image){
    }
```

```
public BufferedImage getImageWithID(int id){
    return null;
}
public GameScreen(int w, int h){
    this.CUSTOM_WIDTH = w;
    this.CUSTOM_HEIGHT = h;
    MASTER_WIDTH = CUSTOM_WIDTH;
    MASTER_HEIGHT = CUSTOM_HEIGHT;
    InitThread();
    InitScreen();
}
private void InitScreen(){

    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.addKeyListener(this);
    setSize(CUSTOM_WIDTH, CUSTOM_HEIGHT);
    setVisible(true);
}
public void BeginGame(){
    G_Thread.StartThread();
}
private void InitThread(){
    G_Thread = new GameThread(this);
    add(G_Thread);
}
```

```
    @Override
    public void keyTyped(KeyEvent e) {}
    @Override
    public void keyPressed(KeyEvent e) {
        KEY_ACTION(e, GameScreen.KEY_PRESSED);
    }
    @Override
    public void keyReleased(KeyEvent e) {
        KEY_ACTION(e, GameScreen.KEY_RELEASED);
    }
    public abstract void GAME_UPDATE(long deltaTime);
    public abstract void GAME_PAINT(Graphics2D g2);
    public abstract void KEY_ACTION(KeyEvent e, int Event);
}
```

GameThread.java:

```
package Flappybackend;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import javax.swing.JPanel;
public class GameThread extends JPanel implements Runnable{
    private GameScreen context;
    private Thread thread;
    private Graphics ThisGraphics;
```

```
public static int FPS = 70;
private BufferedImage buffImage;
private int MasterWidth, MasterHeight;
public static float scaleX_ = 1, scaleY_ = 1;
public GameThread(GameScreen context){
    this.context = context;
    MasterWidth = context.CUSTOM_WIDTH;
    MasterHeight = context.CUSTOM_HEIGHT;
    this.thread = new Thread(this);
}
public void StartThread(){
    thread.start();
}
public void paint(Graphics g){
    g.setColor(Color.white);
    g.fillRect(0, 0, context.CUSTOM_WIDTH, context.CUSTOM_HEIGHT);
    Graphics2D g2 = (Graphics2D)g;
    if(buffImage!=null){
        g2.scale(scaleX_, scaleY_);
        g2.drawImage(buffImage, 0, 0, this);
    }
}

private void UpdateSize(){
    if(this.getWidth()<=0) return;
```

```
        context.CUSTOM_WIDTH = this.getWidth();
        context.CUSTOM_HEIGHT = this.getHeight();

        scaleX_ = (float)context.CUSTOM_WIDTH/(float)MasterWidth;
        scaleY_ = (float)context.CUSTOM_HEIGHT/(float)MasterHeight;
    }
    @Override
    public void run() {
        long T = 1000/FPS;
        long TimeBuffer = T/2;
        long BeginTime = System.currentTimeMillis();
        long EndTime;
        long sleepTime;
        while(true){
            UpdateSize();
            context.GAME_UPDATE(System.currentTimeMillis());
            try{
                buffImage = new BufferedImage(MasterWidth, MasterHeight,
BufferedImage.TYPE_INT_ARGB);
                if(buffImage == null) return;
                Graphics2D g2 = (Graphics2D) buffImage.getGraphics();

                if(g2!=null){
                    context.GAME_PAINT(g2);
                }
            }catch(Exception myException){
                myException.printStackTrace();
```

```
            }
            repaint();
            EndTime = System.currentTimeMillis();
            sleepTime = T - (EndTime - BeginTime);
            if(sleepTime < 0) sleepTime = TimeBuffer;
            try {
                Thread.sleep(sleepTime);
            } catch (InterruptedException ex) {}
            BeginTime = System.currentTimeMillis();
        }
    }
}
```

Objects.java:

```
package Flappybackend;
public class Objects {
    private float posX, posY;
    private float w, h;
    public Objects(){
        posX = posY = w = h = 0;
    }
    public Objects(float x, float y, float w, float h){
        this.posX = x;
        this.posY = y;
        this.w = w;
        this.h = h;
```

```
    }
    public boolean isCollisionHappenWith(float x, float y){
        if(x > posX && x < posX + w && y > posY && y < posY + h)
            return true;
        return false;
    }
    public boolean isCollisionHappenWith(float x, float y, float w, float h){
        if(x < posX + this.w && x + w > posX && y < posY + this.h && h + y >
posY)
            return true;
        return false;
    }
    public void setPos(float x, float y){
        posX = x;
        posY = y;
    }
    public void setPosX(float x){
        posX = x;
    }
    public void setPosY(float y){
        posY = y;
    }
    public float getPosX(){
        return posX;
    }
    public float getPosY(){
        return posY;
```

```
    }
    public float getW(){
        return w;
    }
    public float getH(){
        return h;
    }
    public void increasePosX(float m){
        posX+=m;
    }
    public void increasePosY(float m){
        posY+=m;
    }
}
```

QueueList.java:

```
package Flappybackend;
public class QueueList <T> {
    private Element head, foot;
    private int size = 0;
    public QueueList(){
        head = foot = null;
    }
    public int getSize(){
        return size;
    }
```

```
public void push(T  t){
    Element e = new Element(t);
    if(head == null){
        head = foot = e;
    }else{
        foot.next = e;
        foot = e;
    }
    size++;
}
public T pop(){
    T value = head.value;
    head = head.next;
    size--;
    return value;
}
public T get(int id){
    Element e = head;
    if(head == null) return null;
    for(int i = 0;i < id; i++){
        e = e.next;
        if(e == null) return null;
    }
    return e.value;
}
private class Element{
```

```
        Element(T value){
            this.value = value;
        }
        T value;
        Element next;
    }
}
```

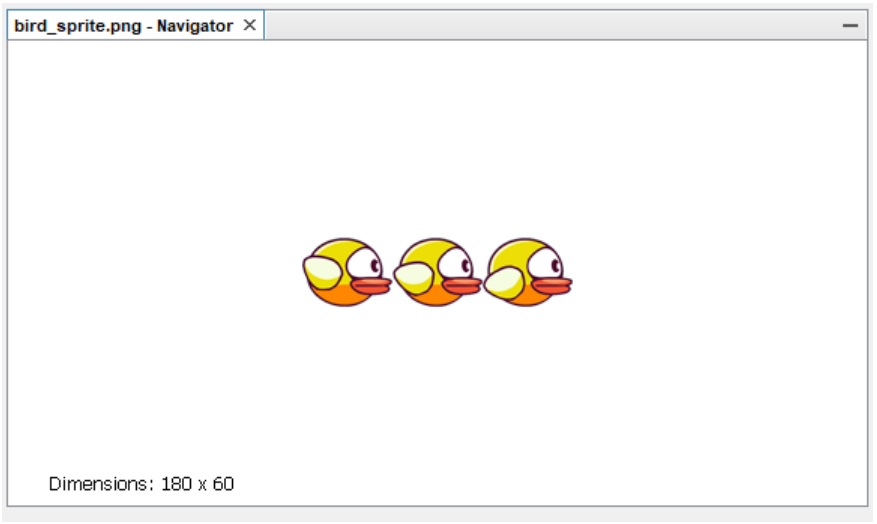
SoundPlayer.java:

```
package Flappybackend;
import java.io.File;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
public class SoundPlayer {
    private Clip clip;
    public SoundPlayer(File path){
        try{
            AudioInputStream ais;
            ais = AudioSystem.getAudioInputStream(path);
            AudioFormat baseFormat = ais.getFormat();
            AudioFormat decodeFormat = new AudioFormat(
                    AudioFormat.Encoding.PCM_SIGNED,
                    baseFormat.getSampleRate(),
                    16,
```

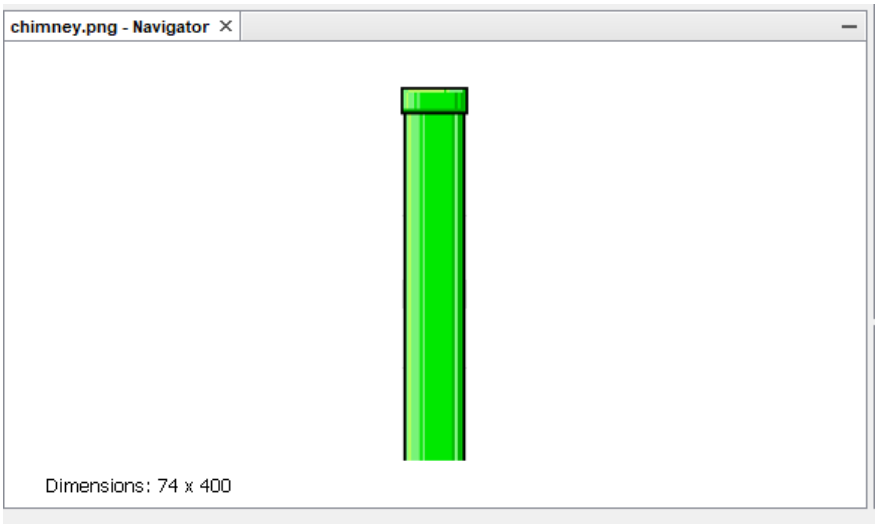
```
                baseFormat.getChannels(),
                baseFormat.getChannels()*2,
                baseFormat.getSampleRate(),
                false
            );
            AudioInputStream dais =
AudioSystem.getAudioInputStream(decodeFormat,  ais);
            clip = AudioSystem.getClip();
            clip.open(dais);
        }catch(Exception e){}
    }
    public void play(){
        if(clip !=null){
            stop();
            clip.setFramePosition(0);
            clip.start();
        }
    }
    public void stop(){
        if(clip.isRunning()) clip.stop();
    }

    public void close(){
        clip.close();
    }
}
```

IMAGES USED FOR THE PROJECT:

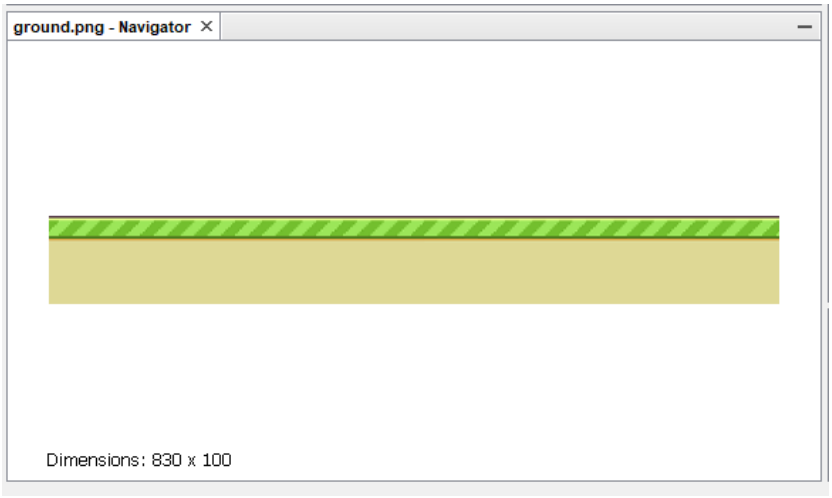
Flappy Bird.png:



Chimney.png:

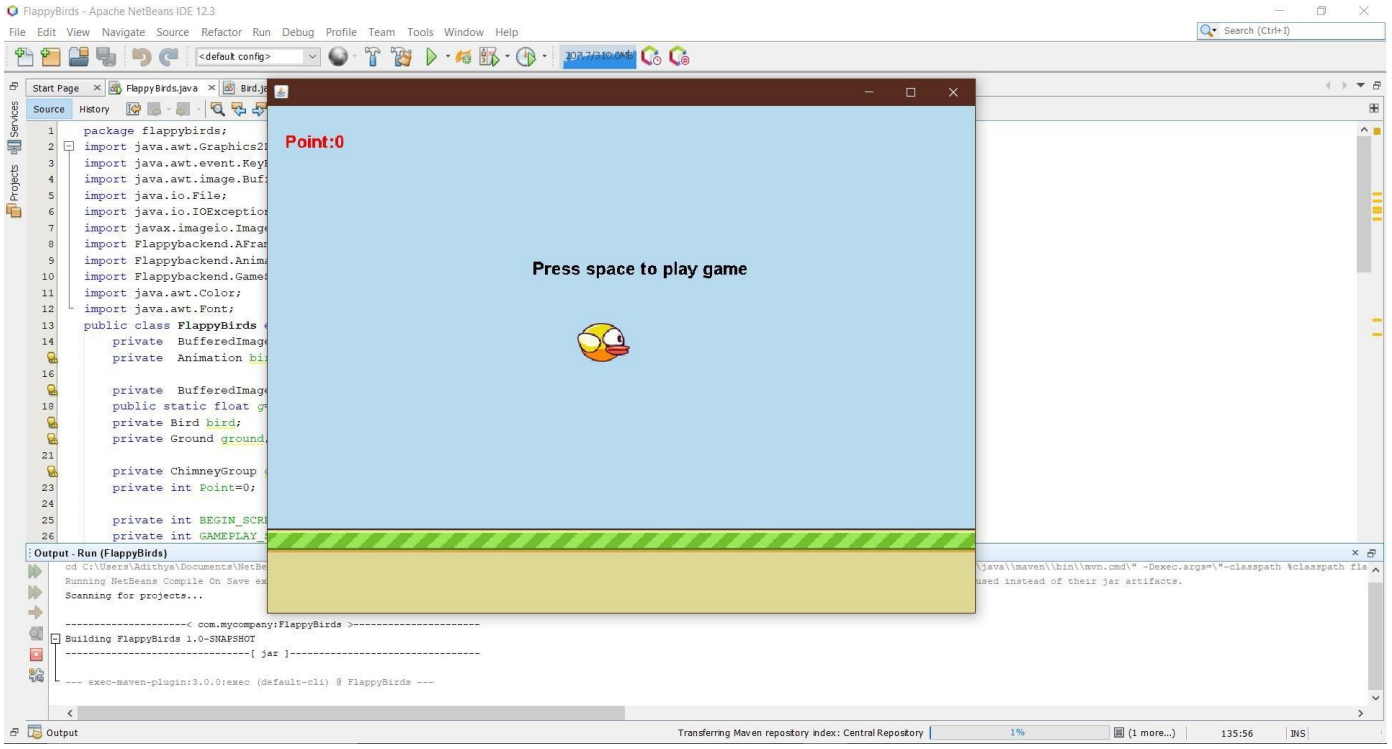


Ground.png:

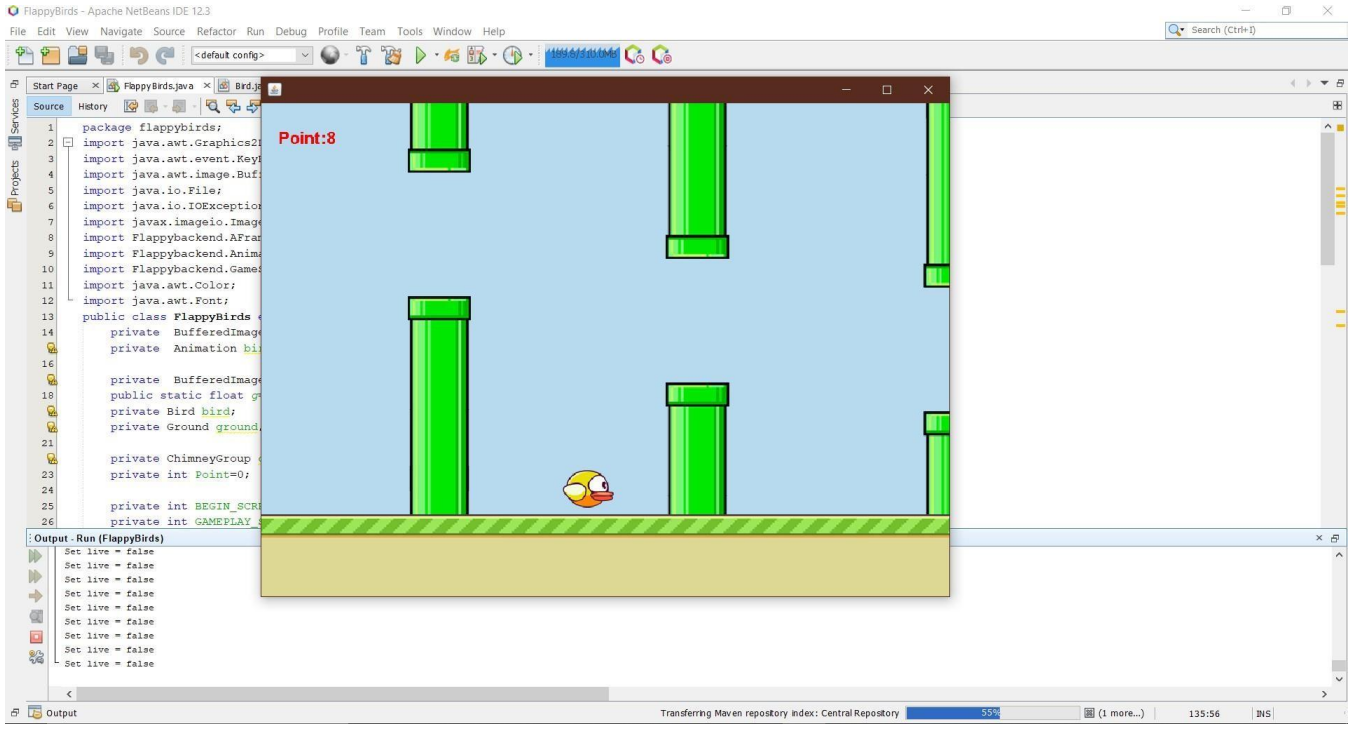


PROJECT OUTPUT SCREENSHOTS:

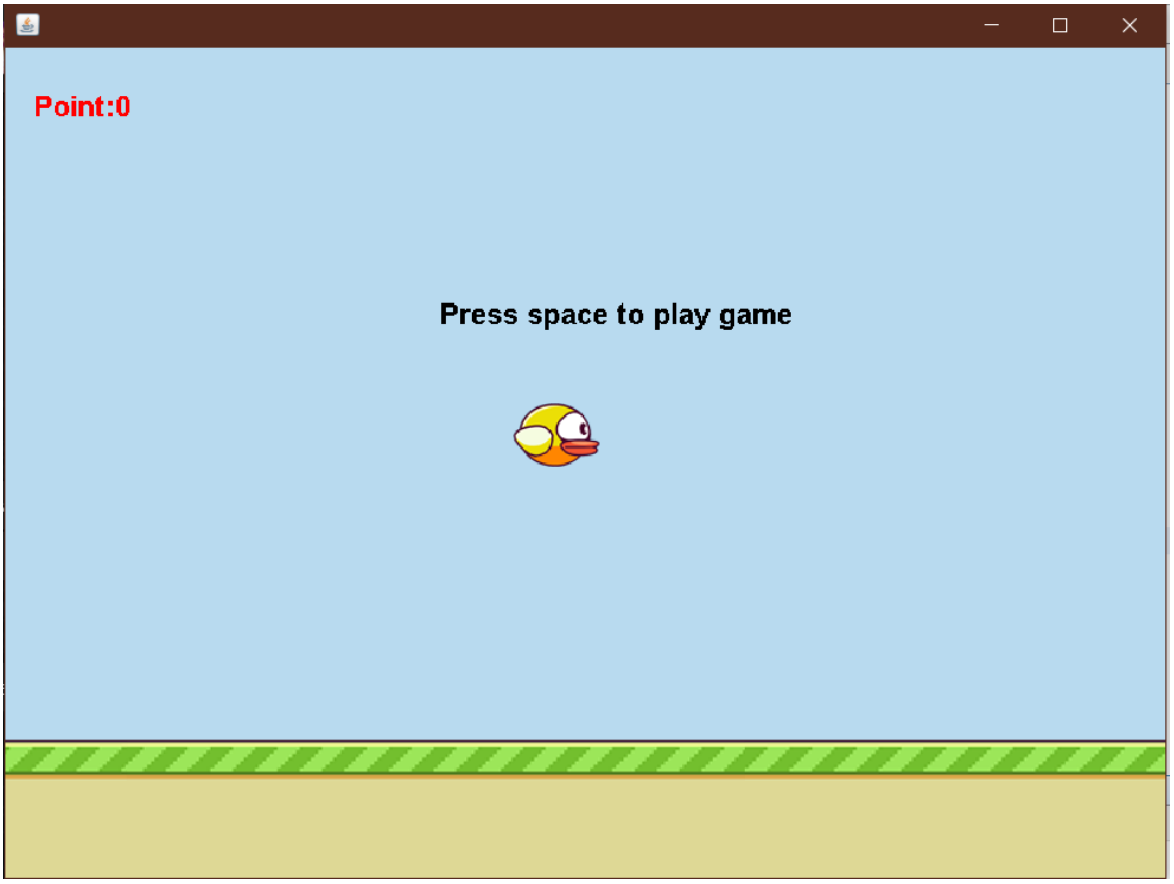
Game starting: Home page: Press Key to start the game.

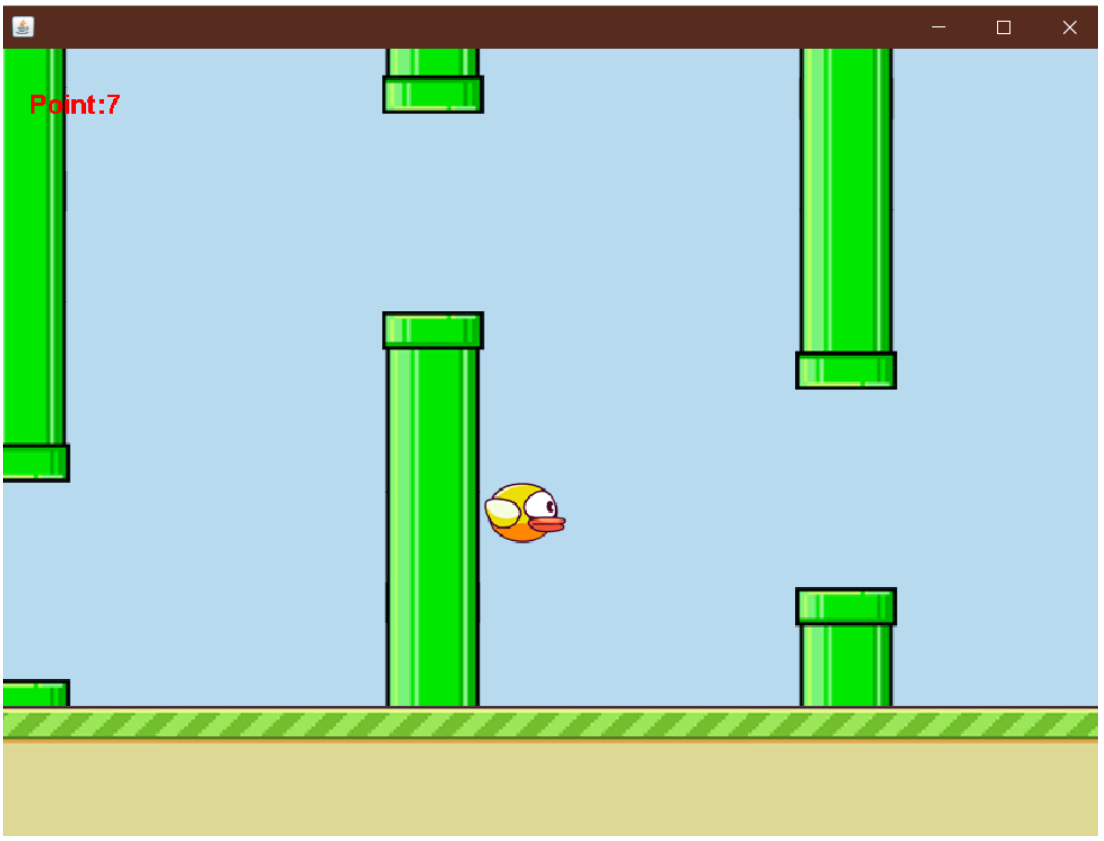


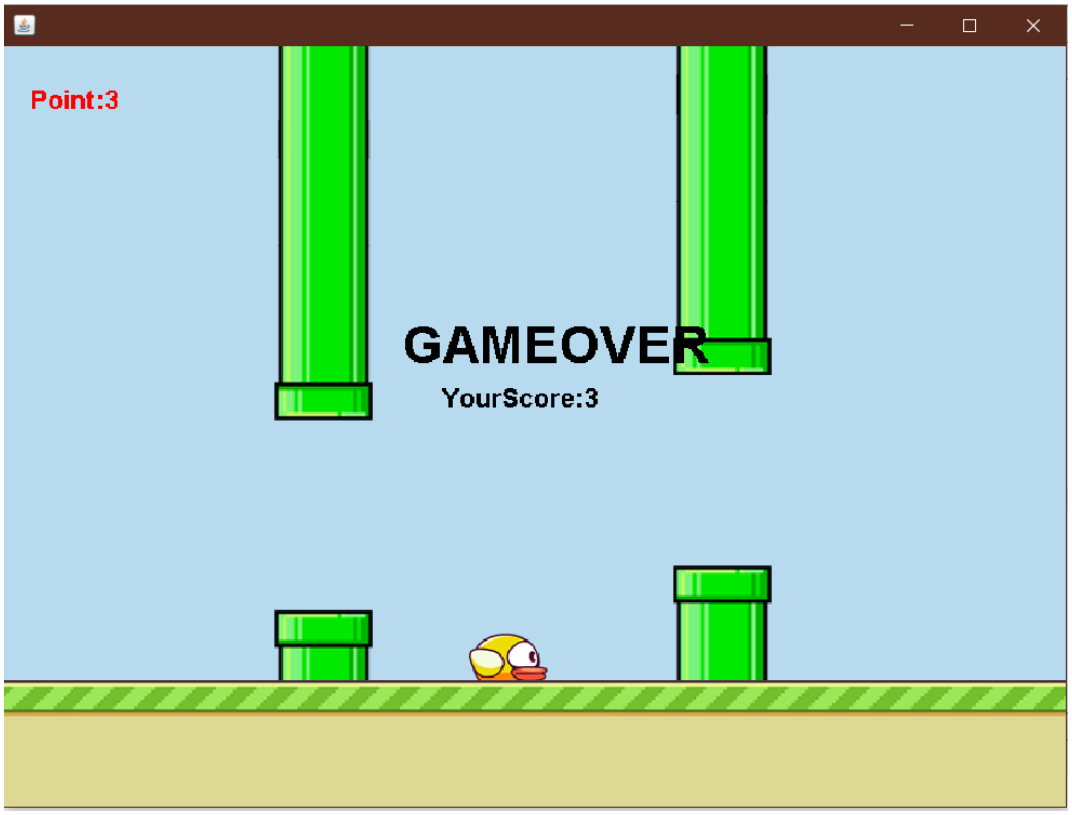
Playing Game:



Game over: Score Points = 2




Point:7


Point:3
GAMEOVER
YourScore:3

CONCLUSION:

This whole project is accomplished in NetBeans IDE. In order to run this project, we will need an NetBeans IDE. First, we have to download this project and then extract it. Open our IDE and import this project. Also, before the project we must have our JDK up to date. Once we import our project, then run the project, we should not run the jar files or the compiled files.

We have chosen the project: Flappy Bird to gain proper knowledge to make game application. It will allow to increase our knowledge for Object Oriented Language (JAVA). Getting experience in java GUI.