**Android Manifest**

*<?*xml version="1.0" encoding="utf-8"*?>*

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.ryantran.rishirush" >

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name=".MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

**Java**

**Buttons:**

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Rect;

public class Buttons {

private int x;

private int y;

private int y2;

private Bitmap bmp;

private GameView gameview;

private int mColumnWidth = 4;

private int mColumnHeight = 1;

private int width;

private int height;

private int mcurrentFrame = 0;

private Rect playerr;

private Rect coinr;

public Buttons(GameView gameview, Bitmap bmp, int x, int y, int state)

{

this.x = x;;

this.y=y;

this.gameview = gameview;

this.bmp = bmp;

this.width = bmp.getWidth()/mColumnWidth;

this.height = bmp.getHeight()/mColumnHeight;

this.mcurrentFrame = state;

}

public void update()

{

}

public int getState(){

return mcurrentFrame;

}

public boolean checkCollision(Rect playerr, Rect coinr){

return Rect.*intersects*(playerr, coinr);

}

public Rect GetBounds()

{

return new Rect(this.getX(),this.y,this.getX()+width,this.y+height);

}

public void onDraw(Canvas canvas){

update();

int srcX = mcurrentFrame\*width;

Rect src = new Rect(mcurrentFrame\*width,0,srcX + width,72);

Rect dst = new Rect(x,y,x+width,y+64);

canvas.drawBitmap(bmp,src,dst,null);

}

public int getX() {

return x;

}

public int getY() {

return y;

}

}

**GameLoopThread:**

import android.graphics.Canvas;

public class GameLoopThread extends Thread{

private GameView view;

private final static int *MAX\_FPS* = 20;

private final static int *MAX\_FRAME\_SKIPS* = 5;

private final static int *FRAME\_PERIOD* = 1000 / *MAX\_FPS*;

boolean running = false;

public GameLoopThread(GameView view){

this.view = view;

}

public void setRunning(){

running = true;

}

@Override

public void run() {

Canvas canvas;

long beginTime;

long timeDiff;

int sleepTime;

int framesSkipped;

sleepTime = 0;

while (running) {

canvas = null;

try {

canvas = view.holder.lockCanvas();

synchronized (view.holder) {

beginTime = System.*currentTimeMillis*();

framesSkipped = 0;

this.view.onDraw(canvas);

timeDiff = System.*currentTimeMillis*() - beginTime;

sleepTime = (int)(*FRAME\_PERIOD* - timeDiff);

if (sleepTime > 0) {

try {

Thread.*sleep*(sleepTime);

} catch (InterruptedException e) {}

}

while (sleepTime < 0 && framesSkipped < *MAX\_FRAME\_SKIPS*) {

this.view.update();

sleepTime += *FRAME\_PERIOD*;

framesSkipped++;

}

}

} finally {

if (canvas != null) {

view.holder.unlockCanvasAndPost(canvas);

}

}

}

}

}

**GameView:**

import java.lang.reflect.Field;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import android.content.Context;

import android.content.SharedPreferences;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Rect;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.SurfaceHolder.Callback;

import android.view.SurfaceView;

public class GameView extends SurfaceView {

GameLoopThread gameLoopThread;

SurfaceHolder holder;

public static int *globalxSpeed* = 8;

Bitmap playerbmp;

Bitmap coinbmp;

Bitmap groundbmp;

Bitmap spikesbmp;

Bitmap powerupshieldbmp;

Bitmap buttonsbmp;

int xx = 0;

private List<Money> coins = new ArrayList<Money>();

private List<Player> player = new ArrayList<Player>();

private List<Ground> ground = new ArrayList<Ground>();

private List<Spikes> spikes = new ArrayList<Spikes>();

private List<PowerupShield> powerupshield = new ArrayList<PowerupShield>();

private List<Buttons> buttons = new ArrayList<Buttons>();

public static int *Coinscollected* = 0;

public static int *Score* = 0;

public static int *HighScore* = 0;

public static int *AchievmentScore10000* = 0;

private boolean PlayerGotPowerupShield = false;

private int PlayerPowerupShieldTimer = 120;

private static SharedPreferences *prefs*;

private int timerCoins = 0;

private int timerSpikes = 0;

private int timerPowerupShield = 0;

private int timerRandomPowerupShield = 0;

private int timerRandomSpikes = 1;

private int lastScore = 0;

private String saveAchievmentScore10000 = "Achievmentscore10000";

private String saveScore = "Highscore";

private String Menu = "Running";

public GameView(Context context) {

super(context);

*prefs* = context.getSharedPreferences("your.ligr.endlessrunninggame",context.*MODE\_PRIVATE*);

String spackage ="your.ligr.endlessrunninggame";

*HighScore* = *prefs*.getInt(saveScore , 0);

*AchievmentScore10000* = *prefs*.getInt(saveAchievmentScore10000, 0);

gameLoopThread = new GameLoopThread(this);

holder = getHolder();

holder.addCallback(new Callback() {

public void surfaceDestroyed(SurfaceHolder arg0) {

*// TODO Auto-generated method stub*

*Score* =0;

*Coinscollected* = 0;

*prefs*.edit().putInt(saveScore,*HighScore*).commit();

*prefs*.edit().putInt(saveAchievmentScore10000,*AchievmentScore10000*).commit();

gameLoopThread.running = false;

}

public void surfaceCreated(SurfaceHolder arg0) {

*// TODO Auto-generated method stub*

gameLoopThread.setRunning();

gameLoopThread.start();

}

public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {

*// TODO Auto-generated method stub*

}

});

playerbmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*rishiboard*);

coinbmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*rishicoin*);

groundbmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*snow*);

spikesbmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*bomb*);

powerupshieldbmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*powerup*);

buttonsbmp = BitmapFactory.*decodeResource*(getResources(), R.drawable.*retry*);

powerupshield.add(new PowerupShield(this,powerupshieldbmp,600,32));

spikes.add(new Spikes(this,spikesbmp,900,0));

spikes.add(new Spikes(this,spikesbmp,1800,0));

player.add(new Player(this,playerbmp,50,50));

coins.add(new Money(this,coinbmp,120,32));

coins.add(new Money(this,coinbmp,50,0));

*// TODO Auto-generated constructor stub*

}

@Override

public boolean onTouchEvent(MotionEvent e){

for(Player pplayer: player)

{

pplayer.ontouch();

}

if (Menu =="Mainmenu")

{

for(int i = 0; i < buttons.size(); i++){

if (buttons.get(i).getState() == 1){ *// Restart*

if ((buttons.get(i).getX()<e.getX() && buttons.get(i).getX()+84>e.getX())){

if (buttons.get(i).getY()<e.getY() && buttons.get(i).getY()+32>e.getY()){

Menu = "Running";

startGame();}

}

}

}

}

return false;

}

public void update(){

if(Menu=="Running"){

*Score* += 5;

lastScore = *Score*;

updatetimers();

deleteground();

if (*Score* >= 10000 && *AchievmentScore10000* == 0)

{

*AchievmentScore10000* = 1;

}

if (*Score* > *HighScore*)

{

*HighScore* = *Score*;

}}

}

public void updatetimers(){

timerCoins ++;

timerSpikes ++;

timerPowerupShield ++;

if (Menu =="Running"){

if (PlayerGotPowerupShield){

PlayerPowerupShieldTimer --;

if (PlayerPowerupShieldTimer <= 0)

{

PlayerGotPowerupShield = false;

}

}

switch(timerRandomPowerupShield){

case 0:

if(timerPowerupShield >= 150){

powerupshield.add(new PowerupShield(this,powerupshieldbmp,this.getWidth()+32,0));

Random randomPowerupShield = new Random();

timerRandomPowerupShield = randomPowerupShield.nextInt(3);

timerPowerupShield = 0;

}break;

case 1:

if(timerPowerupShield >= 250){

powerupshield.add(new PowerupShield(this,powerupshieldbmp,this.getWidth()+32,0));

Random randomPowerupShield = new Random();

timerRandomPowerupShield = randomPowerupShield.nextInt(3);

timerPowerupShield = 0;

}break;

case 2:

if(timerPowerupShield >= 350){

powerupshield.add(new PowerupShield(this,powerupshieldbmp,this.getWidth()+32,0));

Random randomPowerupShield = new Random();

timerRandomPowerupShield = randomPowerupShield.nextInt(3);

timerPowerupShield = 0;

}break;

}

switch(timerRandomSpikes){

case 0:

if(timerSpikes >= 125)

{

spikes.add(new Spikes(this,spikesbmp,this.getWidth()+24,0));

Random randomSpikes = new Random();

timerRandomSpikes = randomSpikes.nextInt(3);

timerSpikes = 0;

}break;

case 1:

if(timerSpikes >= 175)

{

spikes.add(new Spikes(this,spikesbmp,this.getWidth()+24,0));

Random randomSpikes = new Random();

timerRandomSpikes = randomSpikes.nextInt(3);

timerSpikes = 0;

}break;

case 2:

if(timerSpikes >= 100)

{

spikes.add(new Spikes(this,spikesbmp,this.getWidth()+24,0));

Random randomSpikes = new Random();

timerRandomSpikes = randomSpikes.nextInt(3);

timerSpikes = 0;

}

break;

}

if (timerCoins >= 100){

Random randomCoin = new Random();

int random;

random = randomCoin.nextInt(3);

switch(random){

case 1:

int currentcoin = 1;

int xx = 1;

while(currentcoin <= 5){

coins.add(new Money(this,coinbmp,this.getWidth()+(32\*xx),32));

currentcoin++;

xx++;

}

break;

case 2:

currentcoin = 1;

coins.add(new Money(this,coinbmp,this.getWidth()+32,32));

coins.add(new Money(this,coinbmp,this.getWidth()+64,48));

coins.add(new Money(this,coinbmp,this.getWidth()+96,32));

coins.add(new Money(this,coinbmp,this.getWidth()+128,48));

coins.add(new Money(this,coinbmp,this.getWidth()+160,32));

}

timerCoins = 0;

}

}

}

public void addground(){

while(xx < this.getWidth()+Ground.*width*)

{

ground.add(new Ground(this,groundbmp,xx,0));

xx += groundbmp.getWidth();

}

}

public void deleteground(){

for (int i = ground.size()-1;i >= 0; i--)

{

int groundx = ground.get(i).returnX();

if (groundx<=-Ground.*width*){

ground.remove(i);

ground.add(new Ground(this,groundbmp,groundx+this.getWidth()+Ground.*width*,0));

}

}

}

public void startGame(){

for(int i = 0; i < buttons.size(); i++){

buttons.remove(i);

}

player.add(new Player(this,playerbmp,50,50));

}

public void endGame(){

Menu = "Mainmenu";

timerCoins =0;

timerSpikes =0;

timerPowerupShield=0;

buttons.add(new Buttons(this,buttonsbmp,this.getWidth()/2-64,this.getHeight()/2+48,2));

for(int i = 0; i < coins.size(); i++)

{

coins.remove(i);

}

for(int i = 0; i < spikes.size(); i++)

{

spikes.remove(i);

}

for(int i = 0; i < powerupshield.size(); i++)

{

powerupshield.remove(i);

}

player.remove(0);

}

@Override

protected void onDraw(Canvas canvas) {

update();

canvas.drawColor(Color.*BLUE*);

if (Menu=="Mainmenu")

{

for(Buttons bbuttons: buttons)

{

bbuttons.onDraw(canvas);

}

}

if (Menu == "Running"){

addground();

Paint textpaint = new Paint();

textpaint.setTextSize(72);

canvas.drawText("SCORE: "+String.*valueOf*(*Score*), 0, 72, textpaint);

canvas.drawText("HIGH SCORE: "+String.*valueOf*(*HighScore*), 0, 138, textpaint);

canvas.drawText("RISHI COINS: "+String.*valueOf*(*Coinscollected*), 0, 200, textpaint);

for(Ground gground: ground){

gground.onDraw(canvas);

}

for(Player pplayer: player)

{

pplayer.onDraw(canvas);

}

for(int i = 0; i < spikes.size(); i++)

{

spikes.get(i).onDraw(canvas);

Rect playerr = player.get(0).GetBounds();

Rect spikesr = spikes.get(i).GetBounds();

if (spikes.get(i).checkCollision(playerr, spikesr)){

if(!PlayerGotPowerupShield){

*Score* = 0;

*Coinscollected*=0;

endGame();}

else{

spikes.remove(i);

PlayerGotPowerupShield = false;

}

break;

}

}

for(int i = 0; i < coins.size(); i++)

{

coins.get(i).onDraw(canvas);

Rect playerr = player.get(0).GetBounds();

Rect coinr = coins.get(i).GetBounds();

if (coins.get(i).returnX() < 0-32){

coins.remove(i);

}

else if (coins.get(i).checkCollision(playerr, coinr)){

coins.remove(i);

*Score* += 500;

*Coinscollected*+=1;

}

}

for(int i = 0; i < powerupshield.size(); i++)

{

powerupshield.get(i).onDraw(canvas);

Rect playerr = player.get(0).GetBounds();

Rect powerupshieldr = powerupshield.get(i).GetBounds();

if (powerupshield.get(i).returnX() < 0-32){

powerupshield.remove(i);

}

else if(powerupshield.get(i).checkCollision(playerr, powerupshieldr)){

powerupshield.remove(i);

PlayerGotPowerupShield = true;

PlayerPowerupShieldTimer = 120;

}

}

}

if (Menu=="Mainmenu")

{Paint textpaint = new Paint();

textpaint.setTextSize(48);

canvas.drawText("SCORE: "+String.*valueOf*(lastScore), canvas.getWidth()/2, canvas.getHeight()/2, textpaint);

}

}

}

**Ground:**

import android.graphics.Bitmap;

import android.graphics.Canvas;

public class Ground {

public static int *width*;

public static int *height*;

private GameView gameview;

private Bitmap bmp;

private int x;

private int y;

public Ground(GameView gameview, Bitmap bmp, int x, int y){

this.gameview = gameview;

this.bmp = bmp;

this.x = x;

this.y = y;

this.*width* = bmp.getWidth();

this.*height* = bmp.getHeight();

}

public void update(){

x-=gameview.*globalxSpeed*;

}

public int returnX(){

return x;

}

public void onDraw(Canvas canvas)

{

update();

canvas.drawBitmap(bmp, x, gameview.getHeight()-bmp.getHeight(), null);

}

}

**MainActivity:**

import android.app.Activity;

import android.os.Bundle;

public class MainActivity extends Activity {

private GameView gameView;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

gameView = new GameView(this);

setContentView(gameView);

}

@Override

public void onPause(){

super.onPause();

gameView.gameLoopThread.running = false;

finish();

}

}

**Money:**

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Rect;

public class Money {

private int x;

private int y;

private int y2;

private Bitmap bmp;

private GameView gameview;

private int xSpeed =-GameView.*globalxSpeed*;

private int mColumnWidth = 1;

private int mColumnHeight = 4;

private int width;

private int height;

private int mcurrentFrame = 0;

private Rect playerr;

private Rect coinr;

public Money(GameView gameview, Bitmap bmp, int x, int y)

{

this.x = x;

this.y2=y;

this.gameview = gameview;

this.bmp = bmp;

this.width = bmp.getWidth()/mColumnWidth;

this.height = bmp.getHeight()/mColumnHeight;

}

public void update()

{

x += xSpeed;

y = gameview.getHeight()-Ground.*height*-y2-bmp.getHeight();

if (mcurrentFrame >= (mColumnWidth-1))

{

mcurrentFrame = 0;

}

else

mcurrentFrame += 0;

}

public int returnX(){

return x;

}

public boolean checkCollision(Rect playerr, Rect coinr){

this.playerr = playerr;

this.coinr = coinr;

return Rect.*intersects*(playerr, coinr);

}

public Rect GetBounds()

{

return new Rect(this.x,this.y,this.x+width,this.y+height);

}

public void onDraw(Canvas canvas){

update();

int srcX = mcurrentFrame\*width;

Rect src = new Rect(mcurrentFrame\*width,0,srcX + width,32);

Rect dst = new Rect(x,y,x+width,y+32);

canvas.drawBitmap(bmp,src,dst,null);

}

}

**Player:**

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Rect;

public class Player {

static int *x*;

static int *y*;

static int *gravity* = (int) 1.2;

static int *vspeed* = 1;

static int *playerheight* = 24;

static int *playerwidth* = 23;

static int *jumppower* = -20;

private int width,height;

private int mColumnWidth = 1;

private int animationcolumn = 2;

private int mColumnHeight = 1;

private int mcurrentFrame = 4;

private int animationposy = 1;

private int animationstate = 1;

Rect playerr;

Bitmap bmp;

GameView gameview;

private int i;

private Money coin;

public Player(GameView gameview, Bitmap bmp, int x, int y)

{

this.*x* = x;

this.*y* = y;

this.gameview = gameview;

this.bmp = bmp;

this.width = bmp.getWidth()/mColumnWidth;

this.height = bmp.getHeight()/mColumnHeight;

*playerheight*=bmp.getHeight()/1;

}

public void update(){

checkground();

checkanimationstate();

switchanimations();

}

public void checkanimationstate(){

if (*vspeed* < 0){

animationstate =0;

}

else if(*vspeed* > 0){

animationstate =0;

}

else{

animationstate = 0;

}

}

public void switchanimations(){

if(animationstate ==0){

animationcolumn = 0;

animationposy = 0;

if (mcurrentFrame >= (animationcolumn-1))

{

mcurrentFrame = 0;

}

else

mcurrentFrame += 0;}

else if(animationstate ==0){

mcurrentFrame = 0;

animationcolumn = 0;

animationposy=0;

}

else if(animationstate == 0){

mcurrentFrame = 0;

animationposy=0;

animationcolumn = 0;

}

}

public void checkground(){

if (*y* < gameview.getHeight()-Ground.*height*-*playerheight*){

*vspeed*+=*gravity*;

if (*y* > gameview.getHeight()-Ground.*height*-*playerheight*-*vspeed*)

{

*vspeed* = gameview.getHeight()-Ground.*height*-*playerheight*;

}

}

else if (*vspeed*>0)

{

*vspeed* = 0;

*y* = gameview.getHeight()-Ground.*height*-*playerheight*;

}

*y* += *vspeed*;

}

public void ontouch(){

if (*y*>= gameview.getHeight()-Ground.*height*-*playerheight*)

{

*vspeed* = *jumppower*;

}

}

public Rect GetBounds()

{

return new Rect(this.*x*,this.*y*,this.*x*+width,this.*y*+height);

}

public void onDraw(Canvas canvas){

update();

int srcX = mcurrentFrame\*width;

int srcY = animationposy\*48;

Rect src = new Rect(srcX,srcY,srcX + width,srcY+height);

Rect dst = new Rect(*x*,*y*,*x*+(width),*y*+(height));

canvas.drawBitmap(bmp,src,dst,null);

}

*// TODO Auto-generated method stub*

}

**PowerUp:**

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Rect;

public class PowerupShield {

GameView gameview;

Bitmap bmp;

int x,y,y2;

private Rect powerupshieldr;

private Rect playerr;

public PowerupShield(GameView gameview, Bitmap bmp, int x, int y)

{

this.gameview = gameview;

this.bmp = bmp;

this.x = x;

this.y2 = y;

}

public boolean checkCollision(Rect playerr, Rect powerupshieldr){

this.playerr = playerr;

this.powerupshieldr = powerupshieldr;

return Rect.*intersects*(playerr, powerupshieldr);

}

public Rect GetBounds()

{

return new Rect(this.x,this.y,this.x+bmp.getWidth(),this.y+bmp.getHeight());

}

public void Update(){

x-=gameview.*globalxSpeed*;

y = gameview.getHeight()-Ground.*height*-bmp.getHeight()-y2;

}

public int returnX(){

return x;

}

public void onDraw(Canvas canvas){

Update();

int srcX = bmp.getWidth();

Rect src = new Rect(0,0,srcX,bmp.getHeight());

Rect dst = new Rect(x,y,x+bmp.getWidth(),y+bmp.getHeight());

canvas.drawBitmap(bmp,src,dst,null);

}

}

**Spikes:**

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Rect;

public class Spikes {

private int x,y;

private Bitmap bmp;

private GameView gameview;

private Rect playerr;

private Rect spikesr;

public Spikes(GameView gameview, Bitmap spikesbmp,int x,int y){

this.gameview = gameview;

this.bmp = spikesbmp;

this.x = x;

}

public boolean checkCollision(Rect playerr, Rect spikesr){

this.playerr = playerr;

this.spikesr = spikesr;

return Rect.*intersects*(playerr, spikesr);

}

public Rect GetBounds()

{

return new Rect(this.x,this.y,this.x+bmp.getWidth(),this.y+bmp.getHeight());

}

public void Update(){

x-=gameview.*globalxSpeed*;

y = gameview.getHeight()-Ground.*height*-bmp.getHeight();

}

public int returnX(){

return x;

}

public void onDraw(Canvas canvas){

Update();

int srcX = bmp.getWidth();

Rect src = new Rect(0,0,srcX,bmp.getHeight());

Rect dst = new Rect(x,y,x+bmp.getWidth(),y+bmp.getHeight());

canvas.drawBitmap(bmp,src,dst,null);

}

}

**Drawable:**

Here are the following images that must be placed in this directory, these images can be found in the App Proposal folder and or the Progress folder on the ESP 8 GitHub site. (NOTE: You can implement your own images/animations, but they might require recoding.)

back.png

bomb.png

powerup.png

retry.png

rishiboard.png

rishicoin.png

snow.png