**package** com.example.raghav\_dell.my\_first\_game;  
  
**import** android.content.Context;  
*//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.graphics.Typeface;  
  
*//import android.net.Uri;***import** android.media.AudioManager;  
**import** android.media.MediaPlayer;  
**import** android.media.SoundPool;  
**import** android.view.MotionEvent;  
**import** android.view.SurfaceHolder;  
**import** android.view.SurfaceView;  
*//import android.widget.ImageView;  
  
//import java.io.InputStream;  
//import java.net.URL;***import** java.util.ArrayList;  
**import** java.util.Random;  
  
  
  
**public class** GamePanel **extends** SurfaceView **implements** SurfaceHolder.Callback  
{  
 **private** MediaPlayer **explo**;  
 **public static final int *WIDTH*** =856 ;  
 **public static final int *HEIGHT*** = 480 ;  
 **public static final int *MOVESPEED*** = -6;  
 **private long smokeStartTime**;  
 **private long missileStartTime**;  
 **private** MainThread **thread**;  
 **private** Background **bg**;  
 **private** Player **player**;  
 **private** ArrayList<Smokepuff> **smoke**;  
 **private** ArrayList<Missile> **missiles**;  
 **private** ArrayList<TopBorder> **topborder**;  
 **private** ArrayList<BotBorder> **botborder**;  
 **private** Random **rand** = **new** Random();  
 **private int maxBorderHeight**;  
 **private int minBorderHeight**;  
 **private boolean topDown** = **true**;  
 **private boolean botDown** = **true**;  
 **private boolean newGameCreated**;  
  
 *//increase to slow down difficulty progression, decrease to speed up difficulty progression* **private int progressDenom** = 15;  
 **private** Explosion **explosion**;  
 **private long startReset**;  
 **private boolean reset**;  
 **private boolean dissapear**;  
 **private boolean started**;  
 **private int best**;  
  
 **public** GamePanel(Context context)  
 {  
 **super**(context);  
 *//add the callback to the surfaceholder to intercept events* getHolder().addCallback(**this**);  
  
 *//make gamePanel focusable so it can handle events* setFocusable(**true**);  
 }  
  
 @Override  
 **public void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width, **int** height){}  
  
 @Override  
 **public void** surfaceDestroyed(SurfaceHolder holder){  
 **boolean** retry = **true**;  
 **int** counter = 0;  
 **while**(retry && counter<1000)  
 {  
 counter++;  
 **try**{**thread**.setRunning(**false**);  
 **thread**.join();  
 retry = **false**;  
 **thread** = **null**;  
  
 }**catch**(InterruptedException e){e.printStackTrace();}  
  
 }  
  
 }  
  
 @Override  
 **public void** surfaceCreated(SurfaceHolder holder){  
  
 **bg** = **new** Background(BitmapFactory.*decodeResource*(getResources(),R.drawable.***bg***));  
 **player** = **new** Player(BitmapFactory.*decodeResource*(getResources(), R.drawable.***helicopter***), 65, 25,3 );  
 **smoke** = **new** ArrayList<Smokepuff>();  
 **missiles** = **new** ArrayList<Missile>();  
 **topborder** = **new** ArrayList<TopBorder>();  
 **botborder** = **new** ArrayList<BotBorder>();  
 **smokeStartTime**= System.*nanoTime*();  
 **missileStartTime** = System.*nanoTime*();  
  
 **thread** = **new** MainThread(getHolder(), **this**);  
 *//we can safely start the game loop* **thread**.setRunning(**true**);  
 **thread**.start();  
  
 }  
 @Override  
 **public boolean** onTouchEvent(MotionEvent event)  
 {  
 **if**(event.getAction()==MotionEvent.***ACTION\_DOWN***){  
 **if**(!**player**.getPlaying() && **newGameCreated** && **reset**)  
 {  
 **player**.setPlaying(**true**);  
 **player**.SetUp(**true**);  
 }  
 **if**(**player**.getPlaying())  
 {  
  
 **if**(!**started**)**started** = **true**;  
 **reset** = **false**;  
 **player**.SetUp(**true**);  
 }  
 **return true**;  
 }  
 **if**(event.getAction()==MotionEvent.***ACTION\_UP***)  
 {  
 **player**.SetUp(**false**);  
 **return true**;  
 }  
  
 **return super**.onTouchEvent(event);  
 }  
  
 **public void** update()  
  
 {  
 **if**(**player**.getPlaying()) {  
  
 **if**(**botborder**.isEmpty())  
 {  
 **player**.setPlaying(**false**);  
 **return**;  
 }  
 **if**(**topborder**.isEmpty())  
 {  
 **player**.setPlaying(**false**);  
 **return**;  
 }  
  
 **bg**.update();  
 **player**.update();  
  
 *//calculate the threshold of height the border can have based on the score  
 //max and min border heart are updated, and the border switched direction when either max or  
 //min is met* **maxBorderHeight** = 30+**player**.getScore()/**progressDenom**;  
 *//cap max border height so that borders can only take up a total of 1/2 the screen* **if**(**maxBorderHeight** > ***HEIGHT***/4)**maxBorderHeight** = ***HEIGHT***/4;  
 **minBorderHeight** = 5+**player**.getScore()/**progressDenom**;  
  
 *//check bottom border collision* **for**(**int** i = 0; i<**botborder**.size(); i++)  
 {  
 **if**(collision(**botborder**.get(i), **player**))  
 { **player**.setPlaying(**false**);  
 **explo** =MediaPlayer.*create*(getContext(),R.raw.***explosion***);  
 **explo**.start();  
 **best** = **player**.getScore()\*3;}  
 }  
  
 *//check top border collision* **for**(**int** i = 0; i <**topborder**.size(); i++)  
 {  
 **if**(collision(**topborder**.get(i),**player**))  
 { **player**.setPlaying(**false**);  
 **explo** =MediaPlayer.*create*(getContext(),R.raw.***explosion***);  
 **explo**.start();  
 **best** = **player**.getScore()\*3;}  
 }  
  
 *//update top border* **this**.updateTopBorder();  
  
 *//update bottom border* **this**.updateBottomBorder();  
  
 *//add missiles on timer* **long** missileElapsed = (System.*nanoTime*()-**missileStartTime**)/(100000-4000);  
 **if**(missileElapsed >(2000 - **player**.getScore()/4)){  
  
  
 *//first missile always goes down the middle* **missiles**.add(**new** Missile(BitmapFactory.*decodeResource*(getResources(),R.drawable.***missile***),  
 ***WIDTH***+10, (**int**)(**rand**.nextDouble()\*(***HEIGHT*** )),45,15, **player**.getScore(),13));  
  
  
 *//reset timer* **missileStartTime** = System.*nanoTime*();  
 }  
 *//loop through every missile and check collision and remove* **for**(**int** i = 0; i<**missiles**.size();i++)  
 {  
 *//update missile* **missiles**.get(i).update();  
  
 **if**(collision(**missiles**.get(i),**player**))  
 {  
 **missiles**.remove(i);  
 **player**.setPlaying(**false**);  
 **explo** =MediaPlayer.*create*(getContext(),R.raw.***explosion***);  
 **explo**.start();  
 **best** = **player**.getScore()\*3;  
 **break**;  
 }  
 *//remove missile if it is way off the screen* **if**(**missiles**.get(i).getX()<-100)  
 {  
 **missiles**.remove(i);  
 **break**;  
 }  
  
  
 }  
  
 *//add smoke puffs on timer* **long** elapsed = (System.*nanoTime*() - **smokeStartTime**)/1000000;  
 **if**(elapsed > 120){  
 **smoke**.add(**new** Smokepuff(**player**.getX(), **player**.getY()+10));  
 **smokeStartTime** = System.*nanoTime*();  
 }  
  
 **for**(**int** i = 0; i<**smoke**.size();i++)  
 {  
 **smoke**.get(i).update();  
 **if**(**smoke**.get(i).getX()<-10)  
 {  
 **smoke**.remove(i);  
 }  
 }  
 }  
 **else**{  
 **player**.ResetDY();  
 **if**(!**reset**)  
 {  
 **newGameCreated** = **false**;  
 **startReset** = System.*nanoTime*();  
 **reset** = **true**;  
 **dissapear** = **true**;  
 **explosion** = **new** Explosion(BitmapFactory.*decodeResource*(getResources(),R.drawable.***explosion***),**player**.getX(),  
 **player**.getY()-30, 100, 100, 25);  
 }  
  
 **explosion**.update();  
 **long** resetElapsed = (System.*nanoTime*()-**startReset**)/1000000;  
  
 **if**(resetElapsed > 2500 && !**newGameCreated**)  
 {  
 newGame();  
 }  
  
  
 }  
  
 }  
 **public boolean** collision(GameObject a, GameObject b)  
 {  
 **if**(Rect.*intersects*(a.getRectangle(), b.getRectangle()))  
 {  
 **return true**;  
 }  
 **return false**;  
 }  
 @Override  
 **public void** draw(Canvas canvas)  
 {  
 **final float** scaleFactorX = getWidth()/(***WIDTH***\*1.f);  
 **final float** scaleFactorY = getHeight()/(***HEIGHT***\*1.f);  
  
 **if**(canvas!=**null**) {  
 **final int** savedState = canvas.save();  
 canvas.scale(scaleFactorX, scaleFactorY);  
 **bg**.draw(canvas);  
 **if**(!**dissapear**) {  
 **player**.draw(canvas);  
 }  
 *//draw smokepuffs* **for**(Smokepuff sp: **smoke**)  
 {  
 sp.draw(canvas);  
 }  
 *//draw missiles* **for**(Missile m: **missiles**)  
 {  
 m.draw(canvas);  
 }  
  
  
 *//draw topborder* **for**(TopBorder tb: **topborder**)  
 {  
 tb.draw(canvas);  
 }  
  
 *//draw botborder* **for**(BotBorder bb: **botborder**)  
 {  
 bb.draw(canvas);  
 }  
 *//draw explosion* **if**(**started**)  
 {  
 **explosion**.draw(canvas);  
 }  
 drawText(canvas);  
 canvas.restoreToCount(savedState);  
  
 }  
 }  
  
 **public void** updateTopBorder()  
 {  
 *//every 50 points, insert randomly placed top blocks that break the pattern* **if**(**player**.getScore()%50 ==0)  
 {  
 **topborder**.add(**new** TopBorder(BitmapFactory.*decodeResource*(getResources(),R.drawable.***brick*** ),**topborder**.get(**topborder**.size()-1).getX()+20,0,(**int**)((**rand**.nextDouble()\*(**maxBorderHeight** ))+1)));  
 }  
 **for**(**int** i = 0; i<**topborder**.size(); i++)  
 {  
 **topborder**.get(i).update();  
 **if**(**topborder**.get(i).getX()<-20)  
 {  
 **topborder**.remove(i);  
 *//remove element of arraylist, replace it by adding a new one  
  
 //calculate topdown which determines the direction the border is moving (up or down)* **if**(**topborder**.get(**topborder**.size()-1).getHeight()>=**maxBorderHeight**)  
 {  
 **topDown** = **false**;  
 }  
 **if**(**topborder**.get(**topborder**.size()-1).getHeight()<=**minBorderHeight**)  
 {  
 **topDown** = **true**;  
 }  
 *//new border added will have larger height* **if**(**topDown**)  
 {  
 **topborder**.add(**new** TopBorder(BitmapFactory.*decodeResource*(getResources(),  
 R.drawable.***brick***),**topborder**.get(**topborder**.size()-1).getX()+20,  
 0, **topborder**.get(**topborder**.size()-1).getHeight()+1));  
 }  
 *//new border added wil have smaller height* **else** {  
 **topborder**.add(**new** TopBorder(BitmapFactory.*decodeResource*(getResources(),  
 R.drawable.***brick***),**topborder**.get(**topborder**.size()-1).getX()+20,  
 0, **topborder**.get(**topborder**.size()-1).getHeight()-1));  
 }  
  
 }  
 }  
  
 }  
 **public void** updateBottomBorder()  
 {  
 *//every 40 points, insert randomly placed bottom blocks that break pattern* **if**(**player**.getScore()%40 == 0)  
 {  
 **botborder**.add(**new** BotBorder(BitmapFactory.*decodeResource*(getResources(), R.drawable.***brick***),  
 **botborder**.get(**botborder**.size()-1).getX()+20,(**int**)((**rand**.nextDouble()  
 \***maxBorderHeight**)+(***HEIGHT***-**maxBorderHeight**))));  
 }  
  
 *//update bottom border* **for**(**int** i = 0; i<**botborder**.size(); i++)  
 {  
 **botborder**.get(i).update();  
  
 *//if border is moving off screen, remove it and add a corresponding new one* **if**(**botborder**.get(i).getX()<-20) {  
 **botborder**.remove(i);  
  
  
 *//determine if border will be moving up or down* **if** (**botborder**.get(**botborder**.size() - 1).getY() <= ***HEIGHT***-**maxBorderHeight**) {  
 **botDown** = **true**;  
 }  
 **if** (**botborder**.get(**botborder**.size() - 1).getY() >= ***HEIGHT*** - **minBorderHeight**) {  
 **botDown** = **false**;  
 }  
  
 **if** (**botDown**) {  
 **botborder**.add(**new** BotBorder(BitmapFactory.*decodeResource*(getResources(), R.drawable.***brick*** ), **botborder**.get(**botborder**.size() - 1).getX() + 20, **botborder**.get(**botborder**.size() - 1  
 ).getY() + 1));  
 } **else** {  
 **botborder**.add(**new** BotBorder(BitmapFactory.*decodeResource*(getResources(), R.drawable.***brick*** ), **botborder**.get(**botborder**.size() - 1).getX() + 20, **botborder**.get(**botborder**.size() - 1  
 ).getY() - 1));  
 }  
 }  
 }  
 }  
 **public void** newGame()  
 {  
 **dissapear** = **false**;  
  
 **botborder**.clear();  
 **topborder**.clear();  
  
 **missiles**.clear();  
 **smoke**.clear();  
  
 **minBorderHeight** = 5;  
 **maxBorderHeight** = 30;  
  
 **player**.ResetDY();  
 **player**.ResetScore();  
 **player**.setY(***HEIGHT***/2);  
  
 **if**(**player**.getScore()>**best**)  
 {  
 **best** = **player**.getScore()\*3;  
  
 }  
  
 *//create initial borders  
  
 //initial top border* **for**(**int** i = 0; i\*20<***WIDTH***+40;i++)  
 {  
 *//first top border create* **if**(i==0)  
 {  
 **topborder**.add(**new** TopBorder(BitmapFactory.*decodeResource*(getResources(),R.drawable.***brick*** ),i\*20,0, 10));  
 }  
 **else** {  
 **topborder**.add(**new** TopBorder(BitmapFactory.*decodeResource*(getResources(),R.drawable.***brick*** ),i\*20,0, **topborder**.get(i-1).getHeight()+1));  
 }  
 }  
 *//initial bottom border* **for**(**int** i = 0; i\*20<***WIDTH***+40; i++)  
 {  
 *//first border ever created* **if**(i==0)  
 {  
 **botborder**.add(**new** BotBorder(BitmapFactory.*decodeResource*(getResources(),R.drawable.***brick***)  
 ,i\*20,***HEIGHT*** - **minBorderHeight**));  
 }  
 *//adding borders until the initial screen is filed* **else** {  
 **botborder**.add(**new** BotBorder(BitmapFactory.*decodeResource*(getResources(), R.drawable.***brick***),  
 i \* 20, **botborder**.get(i - 1).getY() - 1));  
 }  
 }  
  
 **newGameCreated** = **true**;  
  
  
 }  
 **public void** drawText(Canvas canvas)  
 {  
 Paint paint = **new** Paint();  
 paint.setColor(Color.***BLACK***);  
 paint.setTextSize(30);  
 paint.setTypeface(Typeface.*create*(Typeface.***DEFAULT***, Typeface.***BOLD***));  
 canvas.drawText(**"DISTANCE: "** + (**player**.getScore()\*3), 10, ***HEIGHT*** - 10, paint);  
 canvas.drawText(**"BEST:"** +**best**, ***WIDTH*** - 215, ***HEIGHT*** - 10, paint);  
  
 **if**(!**player**.getPlaying()&&**newGameCreated**&&**reset**)  
 {  
 Paint paint1 = **new** Paint();  
 paint1.setTextSize(40);  
 paint1.setTypeface(Typeface.*create*(Typeface.***DEFAULT***, Typeface.***BOLD***));  
 canvas.drawText(**"PRESS TO START"**, ***WIDTH***/2-50, ***HEIGHT***/2, paint1);  
  
 paint1.setTextSize(20);  
 canvas.drawText(**"PRESS AND HOLD TO GO UP"**, ***WIDTH***/2-50, ***HEIGHT***/2 + 20, paint1);  
 canvas.drawText(**"RELEASE TO GO DOWN"**, ***WIDTH***/2-50, ***HEIGHT***/2 + 40, paint1);  
 }  
 }  
  
  
  
}