**Dharmsinh Desai University**



**Project Title : Candy Breakers (Game)**

**Group No. : 27**

**Submitted By :**

**Roll No Name**

**MCA 36 Mishra Megha**

**MCA 38 Moradiya Gautam G**

**Report Date : 14-Oct-2014**

**Submitted To : prof. Purvi N Jardosh**

**(MCA Department)**

**Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Code**

**MainActivity.Java**

**package** com.example.candybreakers;

**import** android.support.v7.app.ActionBarActivity;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.View;

**import** android.widget.EditText;

**import** android.widget.TextView;

**import** android.widget.Toast;

**public** **class** MainActivity **extends** ActionBarActivity

{

Intent i;

EditText euser;

TextView tv;

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

euser=(EditText)findViewById(R.id.*editText1*);

}

**public** **void** level(View v)

{

**if**(euser.getText().toString().equals(""))

{

Toast.*makeText*(getApplicationContext(), "Please Enter User Name",Toast.*LENGTH\_LONG*).show();

}

**else**

{

i=**new** Intent(getApplicationContext(), PlayStages.**class**);

i.putExtra("user",euser.getText().toString() );

euser.setText("");

startActivity(i);

finish();

}

}

}

**Stages.Java**

**package** com.example.candybreakers;

**import** java.util.Random;

**public** **class** Stages

{

**public** **int** Bricks[][] = **new** **int**[4][10]; // Bricksrix for BRICKS

**int** i, j; // For Row And column

**public** **int** row=4,col=10;

//randomly power

**public** **void** power()

{

Random random=**new** Random();

**int** pwr=random.nextInt(7)+4;

**int** c = random.nextInt(10);

**int** r = random.nextInt(3)+1;

**if**(r%2!=1)

r=r+1;

Bricks[r][c]=pwr;

}

//all bricks is set to 0

**public** **void** blank()

{

**for**(i=0;i<row;i++)

{

**for**(j=0;j<col;j++)

Bricks[i][j]=0;

}

}

// First STAGE of GAME

**public** **void** stage1()

{

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if** (i == 2 )

Bricks[i][j] = 1;

**else**

Bricks[i][j] = 0;

}

}

Random random=**new** Random();

**int** pwr=random.nextInt(7)+4;

**int** c = random.nextInt(10);

Bricks[2][c]=pwr;

Bricks[2][8]=9;

}

// second stage of game

**public** **void** stage2()

{

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if** (i%2==1)

Bricks[i][j] = 1;

**else**

Bricks[i][j] = 0;

}

}

power();

}

//Third stage

**public** **void** stage3()

{

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = 1;

}

}

power();

//Bricks[3][7]=7;

}

//Fourth stage

**public** **void** stage4()

{

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i==1)

Bricks[i][j] = 2;

**else** **if**(i==3)

Bricks[i][j]=1;

**else**

Bricks[i][j]=0;

}

}

power();

}

//Fifth stage

**public** **void** stage5()

{

**int** mat[][]=**new** **int**[][]{{2,2,2,2,1,1,2,2,2,2},{1,1,1,1,2,2,1,1,1,1},{2,2,2,2,1,1,2,2,2,2},{1,1,1,1,2,2,1,1,1,1}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j]=mat[i][j];

}

}

power();

}

//Sixth stage

**public** **void** stage6()

{

**int** mat[][]=**new** **int**[][]{{13,1,1,1,1,1,1,1,1,13},{2,13,2,13,2,2,13,2,13,2},{1,1,1,1,1,1,1,1,1,1},{2,2,1,1,2,2,1,1,2,2}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j]=mat[i][j];

}

}

power();

}

//Seventh stage

**public** **void** stage7()

{

**int** mat[][]=**new** **int**[][]{{2,1,1,1,1,1,2,1,1,1},{1,2,1,1,1,2,1,2,1,1},{1,1,2,1,2,1,1,1,2,1},{1,1,1,2,1,1,1,1,1,2}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

}

//Eigth stage

**public** **void** stage8()

{

**int** mat[][]=**new** **int**[][]{{2,2,2,2,2,2,2,2,2,2},{1,1,1,13,13,13,13,1,1,1},{1,1,1,13,13,13,13,1,1,1},{2,2,2,2,2,2,2,2,2,2}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

}

//Ninth Stage

**public** **void** stage9()

{

**int** mat[][]=**new** **int**[][]{{1,1,3,3,1,1,3,3,1,1},{3,3,1,1,3,3,1,1,3,3},{1,1,3,3,1,1,3,3,1,1},{3,3,1,1,3,3,1,1,3,3}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

}

//Tenth Stage

**public** **void** stage10()

{

**int** mat[][]=**new** **int**[][]{{3,2,2,13,3,3,13,2,2,3},{1,1,1,1,3,3,1,1,1,1},{1,1,1,1,13,13,1,1,1,1},{3,2,2,13,3,3,13,2,2,3}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

}

//Eleventh Stage

**public** **void** stage11()

{

**int** mat[][]=**new** **int**[][]{{3,3,3,3,3,3,3,3,3,3},{2,2,2,2,2,2,2,2,2,2},{3,3,3,3,3,3,3,3,3,3},{0,0,0,0,0,0,0,0,0,0}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

power();

}

//Twelve Stage

**public** **void** stage12()

{

**int** mat[][]=**new** **int**[][]{{2,2,3,2,2,3,2,2,3,13},{13,3,2,2,3,2,2,3,2,2},{2,2,3,2,2,3,2,2,3,13},{13,3,2,2,3,2,2,3,2,2}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];;

}

}

power();

}

//Thirteenth Stage

**public** **void** stage13()

{

**int** mat[][]=**new** **int**[][]{{0,0,13,13,0,0,13,13,0,0},{3,3,3,3,3,3,3,3,3,3},{13,13,0,0,0,0,0,0,13,13},{0,0,0,0,0,0,0,0,0,0}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

power();

}

//Fourteenth Stage

**public** **void** stage14()

{

**int** mat[][]=**new** **int**[][]{{3,3,3,3,3,3,3,3,3,3},{0,0,0,13,13,13,13,0,0,0},{13,2,2,3,3,3,3,2,2,13},{0,0,0,0,0,0,0,0,0,0}};

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

**if**(i!=0)

Bricks[i][j] = mat[i][j];

}

}

power();

power();

}

//Fifteenth Stage

**public** **void** stage15()

{

**int** [][]mat=**new** **int**[][]{ {2,2,2,0,13,2,0,0,0,2},{2,0,0,0,13,2,2,0,2,2,},{2,0,2,2,13,2,0,2,0,2},{2,2,2,0,13,2,0,0,0,2} };

**for** (i = 0; i < row; i++)

{

**for** (j = 0; j < col; j++)

{

Bricks[i][j]=mat[i][j];

}

}

power();

}

//Power For Break Horizontal Line

**public** **void** break\_horizontal(**int** i1)

{

**for** (**int** j = 0; j < col; j++)

break\_candy(i1, j);

}

//Power For Break Vertical Line

**public** **void** break\_vertical(**int** j1)

{

**for** (**int** i = 0; i < row; i++)

break\_candy(i, j1);

}

//Power For Break Nearest Candy

**public** **void** break\_nearest(**int** i1, **int** j1)

{

**if** (i1 != row-1)

{

break\_candy(i1 + 1, j1);

**if** (j1 != 0)

break\_candy(i1 + 1, j1 - 1);

**if** (j1 != col-1)

break\_candy(i1 + 1, j1 + 1);

}

**if** (i1 != 0)

{

break\_candy(i1 - 1, j1);

**if** (j1 != 0)

break\_candy(i1 - 1, j1 - 1);

**if** (j1 != col-1)

break\_candy(i1 - 1, j1 + 1);

}

**if** (j1 != 0)

break\_candy(i1 , j1-1);

**if** (j1 != col-1)

break\_candy(i1 , j1+1);

}

//For Break The Candy

**public** **void** break\_candy(**int** i1, **int** j1) {

**if** (Bricks[i1][j1] == 1 || Bricks[i1][j1] == 2 || Bricks[i1][j1] == 3)

Bricks[i1][j1]--;

**else** **if** (Bricks[i1][j1] == 4) // break horizontal line

{

Bricks[i1][j1] = 0;

break\_horizontal(i1);

}

**else** **if** (Bricks[i1][j1] == 5)// break vertical line

{

Bricks[i1][j1] = 0;

break\_vertical(j1);

}

**else** **if** (Bricks[i1][j1] == 6)// break nearest candy

{

Bricks[i1][j1] = 0;

break\_nearest(i1, j1);

}

**else** **if** (Bricks[i1][j1] == 7)// increase size of stick

{

Bricks[i1][j1] = 0;

PlayStages.*board\_width*=(**float**) (PlayStages.*board\_width*\*1.25);

}

**else** **if** (Bricks[i1][j1] == 8)// decrease size of stick

{

Bricks[i1][j1] = 0;

PlayStages.*board\_width*=(**float**) (PlayStages.*board\_width*\*0.80);

}

**else** **if**(Bricks[i1][j1]==9)//inc speed

{

Bricks[i1][j1] = 0;

PlayStages.*speed*-=9;

}

**else** **if** (Bricks[i1][j1] == 10)// skip level

{

blank();

}

**else** **if** (Bricks[i1][j1] == 11)// remove time

{

PlayStages.*timer*.cancel();

Bricks[i1][j1] = 0;

}

}

}

**PlayStages.Java**

package com.example.candybreakers;

import java.util.concurrent.TimeUnit;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.Typeface;

import android.media.MediaPlayer;

import android.os.Bundle;

import android.os.CountDownTimer;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

import android.view.View;

import android.view.View.OnTouchListener;

public class PlayStages extends Activity

{

OurView v;

static float board\_width; //width of board

static int speed; //speed of the ball

static CounterClass timer; //for timer

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

v = new OurView(this);

setContentView(v);

}

@Override

protected void onPause()

{

super.onPause();

v.pause();

}

@Override

protected void onResume()

{

super.onResume();

v.resume();

}

//inner class counter for time calculation

@SuppressLint("NewApi")

public class CounterClass extends CountDownTimer

{

long millis;

public CounterClass(long millisInFuture, long countDownInterval)

{

super(millisInFuture, countDownInterval);

}

@Override

public void onFinish() { }

String hms; //for print hour minit and time

@Override

public void onTick(long millisUntilFinished)

{

millis=millisUntilFinished;

hms=String.format("%02d : %02d : %02d", TimeUnit.MILLISECONDS.toHours(millis),

TimeUnit.MILLISECONDS.toMinutes(millis)-TimeUnit.HOURS.toMinutes(TimeUnit.MILLISECONDS.toHours(millis)),

TimeUnit.MILLISECONDS.toSeconds(millis)-TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(millis))-1);

}

}

//inner class counter end

//For Resizing the images

public Bitmap getResizedBitmap(Bitmap bm, float newHeight, float newWidth)

{

int width = bm.getWidth();

int height = bm.getHeight();

float scaleWidth = ((float) newWidth) / width;

float scaleHeight = ((float) newHeight) / height;

Matrix matrix = new Matrix();

matrix.postScale(scaleWidth, scaleHeight);

Bitmap resizedBitmap = Bitmap.createBitmap(bm, 0, 0, width, height,

matrix, false);

return resizedBitmap;

}

//inner Ourview class started for surface

@SuppressLint({ "WrongCall", "DrawAllocation" })

public class OurView extends SurfaceView implements Runnable, OnTouchListener

{

Thread t = null; //For thread

SurfaceHolder holder; //Holder object

Stages s = new Stages(); //Object of stages class

SQLiteDatabase userDB; //for Database

Cursor c; //Cursor for DB

String user; //For User

int stage; //Initial position of stage that user completed

int game\_screen; //Different screens of stage(Game over/Level l Level Completed/Diff Stages)

int board\_inc\_x; //Increment in X position of Board

int ball\_positive\_inc; //positive increment of Ball

int ball\_negative\_inc; //Negative increment of Ball

int cnt; //For counting remaining bricks

int score; //For score

int remaining\_time; //remaining time of perticular level

int l; //For Selecting level

//int scr[]=new int[]{0,10,20,30,30,40,50,50,30,40,40,40,50,30,60,70};

int level\_time[]=new int[] {0,151,201,201,201,301,401,501,401,401,501,401,401,401,401,501};

//For diff Stage Time

long milli;

float ballX,ballY; //Position of Ball

float ball\_inc\_x; //Increment in X position of Ball

float ball\_inc\_y; //Increment in Y position of Ball

float touchX,touchY; //X & Y position of touch

float boardX,boardY; //X & Y position of Board

float maxX,maxY; //Increment in Y position of Board

float board\_height; //Height & Width of Board

float candy\_width,candy\_height; //Height & Width of Candy or Ball

float left\_margin,top\_margin; //Left & Top Margin

Bitmap ball,candy; //For Ball & Candy Images

Bitmap board; //For Board Images

Bitmap right\_touch,left\_touch; //For Right Touch & Left Touch Image

Bitmap play,replay; //For Play & Replay Image

Bitmap background; //For Background Image

Bitmap level; //For Diff level number images

Bitmap gameover,close,resume; //For Resume ,Close ,gameover images

Bitmap level\_screen,next; //For Level\_screen & Next Image

boolean running = false; //Handling thread

boolean play\_flag=false; //For Play & Pause Game

boolean isFirstLoad=true; //Check for the Screen is Drawing first time or not

boolean touch\_right\_down,touch\_left\_down; //Flag For Checking Key Is Down or not

Paint fonts=new Paint();

MediaPlayer start\_sound, stage\_sound;

MediaPlayer game\_sound, complete\_sound;

MediaPlayer touch\_sound, stick\_sound, break\_sound;

public OurView(Context context)//constructor

{

super(context);

setOnTouchListener(this);

holder = getHolder();

Bundle usr=getIntent().getExtras();

user=usr.getString("user"); //get the user name

start\_sound = MediaPlayer.create(context, R.raw.start\_sound);

stage\_sound = MediaPlayer.create(context, R.raw.stage\_sound);

game\_sound = MediaPlayer.create(context, R.raw.game\_sound);

complete\_sound = MediaPlayer.create(context, R.raw.complete\_sound);

touch\_sound = MediaPlayer.create(context, R.raw.touch\_sound);

stick\_sound = MediaPlayer.create(context, R.raw.stick\_sound);

break\_sound = MediaPlayer.create(context, R.raw.break\_sound);

fonts.setColor(Color.DKGRAY);

fonts.setTextSize(fonts.getTextSize()+15);

fonts.setTypeface(Typeface.defaultFromStyle(Typeface.BOLD));

fonts.setTypeface(Typeface.createFromAsset(getAssets(), "candy\_breakers.ttf"));

}

//init() start For Firt time initialization

void init()

{

maxX=getWidth(); //maximum width of screen

maxY=getHeight(); //maximum height of screen

left\_margin=(float) (maxX\*0.21); //space from left side

top\_margin=(float) (maxY\*0.10); //space for top

candy\_height=(float) (maxY\*0.06); //height of candy and ball

candy\_width=(float) (maxX\*0.04); //width of candy and ball

board\_inc\_x=35;

game\_screen=0; //screen of game(diff level)

create\_db();

set\_user(user); //set anew user

reinit(); //initilize the value that change when level start

}//init() end

//reinit() start

public void reinit()

{

boardX=(float) (maxX\*0.42); //init X position of board

boardY=(float) (maxY\*0.80); //init Y position of board

board\_width=(float) (maxX\*0.16); //width of board

board\_height=(float)(maxY\*0.04); //height of board

ballX= (float) (boardX+(maxX\*0.06)); //init X position of ball

ballY= (float) (boardY-candy\_height); //init Y position of ball

play\_flag=false; // flag for play/pause

touch\_right\_down=touch\_left\_down=false; //flag when touch is down

ball\_positive\_inc=8; //ball positive increment

ball\_negative\_inc=-8; //ball negative increment

ball\_inc\_x=ball\_positive\_inc; //increment in X position of ball

ball\_inc\_y=ball\_negative\_inc; //increment in Y position of ball

speed=10; //reinitialize speed

create\_db();

stage=get\_stage(user);

score=0;

}//reinit end

public void run()

{

while (running == true)

{

if (!holder.getSurface().isValid())

{

continue;

}

Canvas c = holder.lockCanvas();

if (isFirstLoad)

{

create\_db();

init();

isFirstLoad=false;

}

onDraw(c);

try

{

Thread.sleep(speed);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

holder.unlockCanvasAndPost(c);

}

}

//onDraw() start For drawing Stages

// @SuppressLint("DrawAllocation")

protected void onDraw(Canvas c)

{

c.drawARGB(255, 0, 0, 0);

fonts.setColor(Color.DKGRAY);

fonts.setTextSize(maxY/15);

if(game\_screen==1)//for game

{

fonts.setColor(Color.WHITE);

fonts.setTextSize(maxY/25+8);

c.drawText(timer.hms, maxX/2-100, maxY-50, fonts);

int scr=get\_high\_score(l);

Cursor c1 = userDB.rawQuery("SELECT \* from high\_score where Level="+l+" ",null);

c1.moveToLast();

String usr="";

if(c1.getCount()!=0)

usr=c1.getString(1);

if(user.equals(usr))

fonts.setColor(Color.MAGENTA);

else

fonts.setColor(Color.DKGRAY);

fonts.setTextSize(maxY/25+8);

c.drawText(" H/S :" + scr +" By " + usr ,maxX/2-(maxX/7),maxY-20, fonts);

//c.drawText( ,(maxX/11)+(maxX/11)\*j\*2-20,((maxY/7)\*2)+(maxY/25)+25+(maxY/7)\*i\*2, fonts);

if (!game\_sound.isPlaying())

{

game\_sound.seekTo(0);

game\_sound.start();

}

if (stage\_sound.isPlaying())

stage\_sound.pause();

if (start\_sound.isPlaying())

start\_sound.pause();

if (complete\_sound.isPlaying())

complete\_sound.pause();

draw\_stage(c);

}//end of if(game\_screen=1)

else if(game\_screen==-3)//for Game over acreen

{

if (!complete\_sound.isPlaying())

{

complete\_sound.seekTo(0);

complete\_sound.start();

}

if (stage\_sound.isPlaying())

stage\_sound.pause();

if (start\_sound.isPlaying())

start\_sound.pause();

if (game\_sound.isPlaying())

game\_sound.pause();

gameover = BitmapFactory.decodeResource(getResources(), R.drawable.level\_complete);

gameover = getResizedBitmap(gameover, maxY, maxX);

c.drawBitmap(gameover, 0, 0, null);

resume=BitmapFactory.decodeResource(getResources(), R.drawable.resume);

resume=getResizedBitmap(resume, maxY/10, maxX/10);

c.drawBitmap(resume,(maxX\*3)/10,maxY/2, null);

level=BitmapFactory.decodeResource(getResources(), R.drawable.select\_level);

level=getResizedBitmap(level, maxY/10, maxX/10);

c.drawBitmap(level,(float)(maxX\*4.5)/10,maxY/2, null);

close=BitmapFactory.decodeResource(getResources(), R.drawable.close);

close=getResizedBitmap(close, maxY/10, maxX/10);

c.drawBitmap(close,(maxX\*6)/10,maxY/2, null);

c.drawText("Game Over",(maxX/2)-100, maxY/2-200, fonts);

}//end of if(game\_screen=-3)

else if(game\_screen==-2)//for disp score after level and next level button and all level button

{

if (!complete\_sound.isPlaying())

{

complete\_sound.seekTo(0);

complete\_sound.start();

}

if (stage\_sound.isPlaying())

stage\_sound.pause();

if (start\_sound.isPlaying())

start\_sound.pause();

if (game\_sound.isPlaying())

game\_sound.pause();

gameover = BitmapFactory.decodeResource(getResources(), R.drawable.level\_complete);

gameover = getResizedBitmap(gameover, maxY, maxX);

c.drawBitmap(gameover, 0, 0, null);

resume=BitmapFactory.decodeResource(getResources(), R.drawable.resume);

resume=getResizedBitmap(resume, maxY/10, maxX/10);

c.drawBitmap(resume,(maxX\*3)/10,maxY/2, null);

level=BitmapFactory.decodeResource(getResources(), R.drawable.select\_level);

level=getResizedBitmap(level, maxY/10, maxX/10);

c.drawBitmap(level,(float)(maxX\*4.5)/10,maxY/2, null);

next=BitmapFactory.decodeResource(getResources(), R.drawable.next);

next=getResizedBitmap(next, maxY/10, maxX/10);

c.drawBitmap(next,(maxX\*6)/10,maxY/2, null);

c.drawText("Level Completed",(maxX/2)-150, maxY/2-200, fonts);

c.drawText("Score is :"+score,(maxX/2)-150, maxY/2-150, fonts);

}//end of if(game\_screen=-2)

else if(game\_screen==-1)//all level completed screen

{

if (!complete\_sound.isPlaying())

{

complete\_sound.seekTo(0);

complete\_sound.start();

}

if (stage\_sound.isPlaying())

stage\_sound.pause();

if (start\_sound.isPlaying())

start\_sound.pause();

if (game\_sound.isPlaying())

game\_sound.pause();

gameover = BitmapFactory.decodeResource(getResources(), R.drawable.level\_complete);

gameover = getResizedBitmap(gameover, maxY, maxX);

c.drawBitmap(gameover, 0, 0, null);

resume=BitmapFactory.decodeResource(getResources(), R.drawable.resume);

resume=getResizedBitmap(resume, maxY/10, maxX/10);

c.drawBitmap(resume,(maxX\*3)/10,maxY/2, null);

level=BitmapFactory.decodeResource(getResources(), R.drawable.select\_level);

level=getResizedBitmap(level, maxY/10, maxX/10);

c.drawBitmap(level,(float)(maxX\*4.5)/10,maxY/2, null);

close=BitmapFactory.decodeResource(getResources(), R.drawable.close);

close=getResizedBitmap(close, maxY/10, maxX/10);

c.drawBitmap(close,(maxX\*6)/10,maxY/2, null);

c.drawText("All level are completed",(maxX/2)-150, maxY/2-200, fonts);

c.drawText("Score is :"+score,(maxX/2)-150, maxY/2-150, fonts);

}//end of if(game\_screen=-1)

else if(game\_screen==0)//draw level screen

{

if (!stage\_sound.isPlaying())

{

stage\_sound.seekTo(0);

stage\_sound.start();

}

if (game\_sound.isPlaying())

game\_sound.pause();

if (start\_sound.isPlaying())

start\_sound.pause();

if (complete\_sound.isPlaying())

complete\_sound.pause();

level\_screen = BitmapFactory.decodeResource(getResources(), R.drawable.level\_screen);

level\_screen = getResizedBitmap(level\_screen, maxY, maxX);

c.drawBitmap(level\_screen, 0, 0, null);

int i,j;

for(i=0;i<03;i++)

{

for(j=0;j<5;j++)

{

Bitmap level = null;

int l=(i\*5)+j+1;

if(l<=stage)

{

int DrawableID=getResources().getIdentifier("level"+l, "drawable",getPackageName());

level= BitmapFactory.decodeResource(getResources(), DrawableID);

level = getResizedBitmap(level, maxY/7, maxX/11);

c.drawBitmap(level, (maxX/11)+(maxX/11)\*j\*2,(maxY/7)+(maxY/7)\*i\*2, null);

}

else

{

level= BitmapFactory.decodeResource(getResources(), R.drawable.lock);

level = getResizedBitmap(level, maxY/7, maxX/11);

c.drawBitmap(level, (maxX/11)+(maxX/11)\*j\*2,(maxY/7)+(maxY/7)\*i\*2, null);

}

int scr=get\_high\_score(l);

Cursor c1 = userDB.rawQuery("SELECT \* from high\_score where Level="+l+" ",null);

c1.moveToLast();

String usr="";

if(c1.getCount()!=0)

usr=c1.getString(1);

if(user.equals(usr))

fonts.setColor(Color.MAGENTA);

else

fonts.setColor(Color.DKGRAY);

fonts.setTextSize(maxY/30);

c.drawText("High Score is :" + scr ,(maxX/11)+(maxX/11)\*j\*2-20,((maxY/7)\*2)+20+(maxY/7)\*i\*2, fonts);

c.drawText("user :" + usr ,(maxX/11)+(maxX/11)\*j\*2-20,((maxY/7)\*2)+(maxY/25)+25+(maxY/7)\*i\*2, fonts);

}//end of for loop(j)

}//end of for loop(i)

c.drawText(" l = "+l + " stg = "+stage, 50, 50, fonts);

}//end of if(game\_screen=0)

check();

}//onDraw() end

//check() start

void check()

{

if(game\_screen==1) // for Diff Stages

{

if(play\_flag==true) //move ball in playing mode

{

ballX+=ball\_inc\_x;

ballY+=ball\_inc\_y;

}

else

timer.cancel();

cnt=0;

int i,j;

for(i=0;i<s.row;i++)

{

for(j=0;j<s.col;j++)

{

if(s.Bricks[i][j]==0 || s.Bricks[i][j]==13)

cnt++;

}

}

if(cnt==s.row\*s.col) //checking for level completion

{

remaining\_time=(int) (timer.millis/1000);

score=5000+(remaining\_time\*100);

set\_high\_score(score,l,user); //for updating the score of user

if(l==15)

game\_screen=-1;

else

{

game\_screen=-2;

if(l>stage && l<=15)

set\_stage(l,user); //for updating the stages completed by the user

}

}

if(timer.hms.equals("00 : 00 : 00"))

{

game\_screen=-3;

}

if(ballY==boardY-candy\_height) //check for Game over

{

if(ballX<boardX-(candy\_width/2) || ballX>boardX+board\_width-(candy\_width/2))

game\_screen=-3;

}

if(ballX<0) //Check when ball Touch left boarder of screen

{

ball\_inc\_x=ball\_positive\_inc;

touch\_sound.start();

}

if(ballY<0) //check when ball touch upper boarder of screen

{

ball\_inc\_y=ball\_positive\_inc;

touch\_sound.start();

}

if(ballX>maxX-candy\_width) //check when ball touch right boarder of screen

{

ball\_inc\_x=ball\_negative\_inc;

touch\_sound.start();

}

if(ballY==boardY-candy\_height) //check when ball touch on to board

{

ball\_inc\_y=ball\_negative\_inc;

if(play\_flag)

stick\_sound.start();

if(ballX+(candy\_width/2) >= boardX+(board\_width/2))

ball\_inc\_x=ball\_positive\_inc;

else

ball\_inc\_x=ball\_negative\_inc;

}

if(ball\_inc\_y==ball\_negative\_inc)//check when ball move up

{

int val,diff,i1,j1;

val=(int) (ballY-top\_margin);

diff=(int) (val/(maxY\*0.08));

val=(int) (val-(diff\*(maxY\*0.02)));//0.02 % diff between 2 candies

i1=(int) (val/(candy\_height));

val=(int) (ballX-left\_margin);

diff=(int) (val/(maxX\*0.06));

val=(int) (val-(diff\*(maxX\*0.02)));//0.02 % diff between 2 candies

j1=(int) (val/(candy\_width));

if(i1>=0 && i1<s.row && j1>=0 && j1<s.col && s.Bricks[i1][j1]!=0)

{

s.break\_candy(i1, j1);

break\_sound.start();

ball\_inc\_y=ball\_positive\_inc;

}

}

else if(ball\_inc\_y==ball\_positive\_inc)//check when ball move down

{

int val,diff,i1,j1;

val=(int) (ballY+candy\_height-top\_margin-(maxY\*0.02));

diff=(int) (val/(maxY\*0.08));

val=(int) (val-(diff\*(maxY\*0.02)));//0.02 % diff between 2 candies

i1=(int) (val/(candy\_height));

val=(int) (ballX-left\_margin);

diff=(int) (val/(maxX\*0.06));

val=(int) (val-(diff\*(maxX\*0.02)));//0.02 % diff between 2 candies

j1=(int) (val/(candy\_width));

if(i1>=0 && i1<s.row && j1>=0 && j1<s.col && s.Bricks[i1][j1]!=0)

{

s.break\_candy(i1, j1);

break\_sound.start();

ball\_inc\_y=ball\_negative\_inc;

}

}

if(ball\_inc\_x==ball\_negative\_inc)//check when ball move left side

{

int val,diff,i1,j1;

val=(int) (ballY-top\_margin);

diff=(int) (val/(maxY\*0.08));

val=(int) (val-(diff\*(maxY\*0.02)));//0.02 % diff between 2 candies

i1=(int) (val/(candy\_height));

val=(int) (ballX-left\_margin);

diff=(int) (val/(maxX\*0.06));

val=(int) (val-(diff\*(maxX\*0.02)));//0.02 % diff between 2 candies

j1=(int) (val/(candy\_width));

if(i1>=0 && i1<s.row && j1>=0 && j1<s.col && s.Bricks[i1][j1]!=0)

{

s.break\_candy(i1, j1);

break\_sound.start();

ball\_inc\_x=ball\_positive\_inc;

}

}

else if(ball\_inc\_x==ball\_positive\_inc) //check when ball move right side

{

int val,diff,i1,j1;

val=(int) (ballY-top\_margin);

diff=(int) (val/(maxY\*0.08));

val=(int) (val-(diff\*(maxY\*0.02)));

i1=(int) (val/(candy\_height));

val=(int) ((ballX + candy\_width) -left\_margin-(maxX\*0.02));

diff=(int) (val/(maxX\*0.06));

val=(int) (val-((diff)\*(maxX\*0.02)));

j1=(int) (val/(candy\_width));

if(i1>=0 && i1<s.row && j1>=0 && j1<s.col && s.Bricks[i1][j1]!=0)

{

s.break\_candy(i1, j1);

break\_sound.start();

ball\_inc\_x=ball\_negative\_inc;

}

}

if(touch\_right\_down)//when long touch right button image

{

if(boardX+board\_width<maxX && play\_flag)

boardX+=board\_inc\_x;

}

if(touch\_left\_down)//when long touch left button image

{

if(boardX>0 && play\_flag)

boardX-=board\_inc\_x;

}

}

}//check() end

//for drawing stages

public void draw\_stage(Canvas c) //screen>0

{

draw\_candy(c);

ball=BitmapFactory.decodeResource(getResources(), R.drawable.candy);

ball = getResizedBitmap(ball, (int)(candy\_height), (int)(candy\_width));

c.drawBitmap(ball, ballX, ballY, null);

board = BitmapFactory.decodeResource(getResources(), R.drawable.board);

board = getResizedBitmap(board, (int)(board\_height), (int)(board\_width));

c.drawBitmap(board, boardX, boardY, null);

left\_touch = BitmapFactory.decodeResource(getResources(), R.drawable.left\_touch);

left\_touch= getResizedBitmap(left\_touch, (int)(candy\_height\*2), (int)(candy\_width\*2));

c.drawBitmap(left\_touch,10 , maxY-(float)(candy\_height\*2) , null);

right\_touch = BitmapFactory.decodeResource(getResources(), R.drawable.right\_touch);

right\_touch = getResizedBitmap(right\_touch, (int)(candy\_height\*2), (int)(candy\_width\*2));

c.drawBitmap(right\_touch,maxX-(float)(candy\_width\*2)-10, maxY-(float)(candy\_height\*2), null);

if(play\_flag==false)

{

play = BitmapFactory.decodeResource(getResources(), R.drawable.play);

play = getResizedBitmap(play, (int)(candy\_height\*2), (int)(candy\_width\*2));

}

else

{

play = BitmapFactory.decodeResource(getResources(), R.drawable.pause);

play = getResizedBitmap(play, (int)(candy\_height\*2), (int)(candy\_width\*2));

}

c.drawBitmap(play, 50+(float)(candy\_width\*2),maxY-(float)(candy\_height\*2), null);

replay = BitmapFactory.decodeResource(getResources(), R.drawable.replay);

replay = getResizedBitmap(replay, (int)(candy\_height\*2), (int)(candy\_width\*2));

c.drawBitmap(replay,(maxX-(float)(candy\_width\*4))-90 , maxY-(float)(candy\_height\*2), null);

}

//drawing all candies

public void draw\_candy(Canvas c)

{

for(int i = 0;i<s.row;i++)

{

for(int j = 0;j<s.col;j++)

{

if(s.Bricks[i][j]==0)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.blank);

else if(s.Bricks[i][j]==1)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.candy1);

else if (s.Bricks[i][j]==2)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.candy2);

else if(s.Bricks[i][j]==3)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.candy3);

else if(s.Bricks[i][j]==4)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.horizontal\_strips);

else if(s.Bricks[i][j]==5)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.vertical\_strips);

else if(s.Bricks[i][j]==6)

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

else if(s.Bricks[i][j]==7)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.inc\_stick\_size);

else if(s.Bricks[i][j]==8)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.dec\_stick\_size);

else if(s.Bricks[i][j]==9)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.inc\_speed);

else if(s.Bricks[i][j]==10)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.skip\_level);

else if(s.Bricks[i][j]==11)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.remove\_time);

else if(s.Bricks[i][j]==12)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.wall);

else if(s.Bricks[i][j]==13)

candy= BitmapFactory.decodeResource(getResources(), R.drawable.wall);

candy=getResizedBitmap(candy, candy\_height,candy\_width);

c.drawBitmap(candy, left\_margin+(float)(0.06\*maxX)\*j, top\_margin+(float)(0.08\*maxY)\*i, null);

}

}

}

//switch\_stage() start

public void switch\_stage(int stg)

{

reinit();

timer=new CounterClass(level\_time[stg]\*1000, 1000);

timer.millis=level\_time[stg]\*1000;

timer.start();

milli=timer.millis;

s.blank();

if(stg==15)

top\_margin=(float) (maxY\*0.15); //space for top

switch(stg)

{

case 1: s.stage1();

break;

case 2: s.stage2();

break;

case 3: s.stage3();

break;

case 4: s.stage4();

break;

case 5: s.stage5();

break;

case 6: s.stage6();

break;

case 7: s.stage7();

break;

case 8: s.stage8();

break;

case 9: s.stage9();

break;

case 10:s.stage10();

break;

case 11:s.stage11();

break;

case 12:s.stage12();

break;

case 13:s.stage13();

break;

case 14:s.stage14();

break;

case 15:s.stage15();

break;

}

}//switch\_stage() end

//onTouch() start

@Override

public boolean onTouch(View v, MotionEvent event)

{

touchX=touchY=0;

switch (event.getAction())

{

case MotionEvent.ACTION\_DOWN:

touchX = event.getX();

touchY = event.getY();

break;

case MotionEvent.ACTION\_UP:

touch\_right\_down=false;

touch\_left\_down=false;

break;

case MotionEvent.ACTION\_MOVE:

break;

}// end switch

if(game\_screen==0)

{

int i,j,row,col;

row=(int) (touchY/(maxY/7));

col=(int) (touchX/(maxX/11));

if(row%2==1 && col%2==1)

{

i=row/2;

j=col/2;

l=(i\*5)+j+1;

if(l<=stage)

{

game\_screen=1;

switch\_stage(l);

}

}

}

else if(game\_screen==1)

{

if(touchX>10 && touchX<(candy\_width\*2)+10 && touchY<maxY && touchY>maxY-(candy\_height\*2))

{

if(boardX>0 && play\_flag)

{

boardX-=board\_inc\_x;

touch\_left\_down=true;

}

}

else if(touchX>(maxX-(candy\_width\*2)-10) && touchX<maxX-10 && touchY<maxY && touchY>maxY-(candy\_height\*2) )

{

if(boardX+board\_width<maxX && play\_flag)

{

boardX+=board\_inc\_x;

touch\_right\_down=true;

}

}

else if( touchX>(50+candy\_width\*2) && touchX<(candy\_width\*4)+50 && touchY<maxY && touchY>maxY-(candy\_height\*2))

{

if(play\_flag==false)

{

play\_flag=true;

timer=new CounterClass(milli, 1000);

timer.start();

}

else

{

play\_flag=false;

milli=timer.millis;

timer.cancel();

}

}

else if(touchX>(maxX-(candy\_width\*4))-90 && touchX<(maxX-(candy\_width\*2))-90 && touchY<maxY && touchY>maxY-(candy\_height\*2))

{

switch\_stage(l);

}

}

else if(game\_screen==-1 || game\_screen==-3) //Game Over or All level Completed

{

//c.drawBitmap(resume,(maxX\*3)/10,maxY/2, null);

if(touchX>(maxX\*3)/10 && touchX<(maxX\*4)/10 && touchY>maxY/2 && touchY<(maxY/2)+(maxY/10) )

{

game\_screen=1;

switch\_stage(l);

}

else if(touchX>(maxX\*4.5)/10 && touchX<(maxX\*5.5)/10 && touchY>maxY/2 && touchY<(maxY/2)+(maxY/10) )//c.drawBitmap(level,(float)(maxX\*4.5)/10,maxY/2, null);

{

game\_screen=0;

}

else if(touchX>(maxX\*6)/10 && touchX<(maxX\*7)/10 && touchY>maxY/2 && touchY<(maxY/2)+(maxY/10)) //c.drawBitmap(close,(maxX\*6)/10,maxY/2, null);

{

finish();

}

}

else if(game\_screen==-2)//if Level completed

{

if(touchX>maxX\*0.3 && touchX<maxX\*0.4 && touchY>maxY/2 && touchY<(maxY/2)+(maxY/10) )

{

game\_screen=1;

switch\_stage(l);

}

if(touchX>(maxX\*4.5)/10 && touchX<(maxX\*5.5)/10 && touchY>maxY/2 && touchY<(maxY/2)+(maxY/10) )//c.drawBitmap(level,(float)(maxX\*4.5)/10,maxY/2, null);

{

game\_screen=0;

}

if(touchX>(maxX\*6)/10 && touchX<(maxX\*7)/10 && touchY>maxY/2 && touchY<(maxY/2)+(maxY/10)) //c.drawBitmap(next,(maxX\*6)/10,maxY/2, null);

{

l++;

game\_screen=1;

if(l>stage)

set\_stage(l, user);

switch\_stage(l);

}

}

return true;

}//onTouch() end

public void pause()

{

start\_sound.stop();

stage\_sound.stop();

complete\_sound.stop();

game\_sound.stop();

running = false;

while (true)

{

try

{

t.join();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

break;

}// end while

}

public void resume()

{

running = true;

t = new Thread(this);

t.start();

}

//database functions

public void create\_db()

{

userDB = openOrCreateDatabase("gamedb",MODE\_PRIVATE,null);

userDB.execSQL("CREATE TABLE IF NOT EXISTS user(UserName VARCHAR,Level INTEGER);");

userDB.execSQL("CREATE TABLE IF NOT EXISTS high\_score(Level INTEGER,UserName VARCHAR,HighScore INTEGER); ");

//userDB.execSQL("DELETE from user where UserName!='gg' ;");

//userDB.delete("user", null, null);

}

public int get\_stage(String user)

{

int stg=0;

//select query for getting the stage of user

c = userDB.rawQuery("SELECT \* from user where UserName='"+user+"' ",null);

c.moveToLast();

stg=c.getInt(c.getColumnIndex("Level"));

return stg;

}

public void set\_stage(int stg,String user)

{

//update query for set the stage after completion of level

userDB.execSQL("UPDATE user set Level = "+stg+" WHERE UserName='"+user+"' ");

}

public void set\_user(String user)

{

//set a user and stage if not exist

c = userDB.rawQuery("SELECT \* from user where UserName='"+user+"'",null);

c.moveToLast();

int cnt=c.getCount();

if(cnt==0)

{

userDB.execSQL("INSERT INTO user VALUES ('"+user+"',1);");

}

}

public void set\_high\_score(int score,int l,String user)

{

//update high score with user name at perticular level that is l

c = userDB.rawQuery("SELECT \* from high\_score where Level="+l+" ",null);

c.moveToLast();

int cnt=c.getCount();

if(cnt==0)

userDB.execSQL("INSERT INTO high\_score VALUES ("+l+",'"+user+"',"+score+" );");

else

{

int old\_score=c.getInt(2);

if(score>old\_score)

userDB.execSQL("UPDATE high\_score set HighScore = "+score+" ,UserName= '"+user+"' where Level="+l+"; ");

}

}

public int get\_high\_score(int l)

{

int score;

c = userDB.rawQuery("SELECT \* from high\_score where Level="+l+" ",null);

c.moveToLast();

if(c.getCount()==0)

score=0;

else

score=c.getInt(c.getColumnIndex("HighScore"));

return score;

}

}//inner class completed

}

**Activity\_main.xml**

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

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:background=*"@drawable/firstscreen"*>

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginLeft=*"35dp"*

android:layout\_marginTop=*"60dp"*

android:textColor=*"#000000"*

android:ems=*"10"* >

<requestFocus />

</EditText>

<ImageView

android:id=*"@+id/imageView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/editText1"*

android:layout\_below=*"@+id/editText1"*

android:layout\_marginTop=*"42dp"*

android:onClick=*"level"*

android:src=*"@drawable/play1"* />

</RelativeLayout>

**Activity\_play\_stage.xml**

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

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"* >

<SurfaceView

android:id=*"@+id/surfaceView1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginLeft=*"0dp"*

android:layout\_marginTop=*"0dp"* />

</RelativeLayout>

**AndroidManifest.xml**

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

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

package=*"com.example.candybreakers"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"8"*

android:targetSdkVersion=*"19"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/firstscreen"*

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

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.example.candybreakers.MainActivity"*

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

android:screenOrientation=*"landscape"*>

<intent-filter>

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

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

</intent-filter>

</activity>

<activity

android:name=*"com.example.candybreakers.PlayStages"*

android:label=*"@string/title\_activity\_play\_stages"*

android:screenOrientation=*"landscape"* >

</activity>

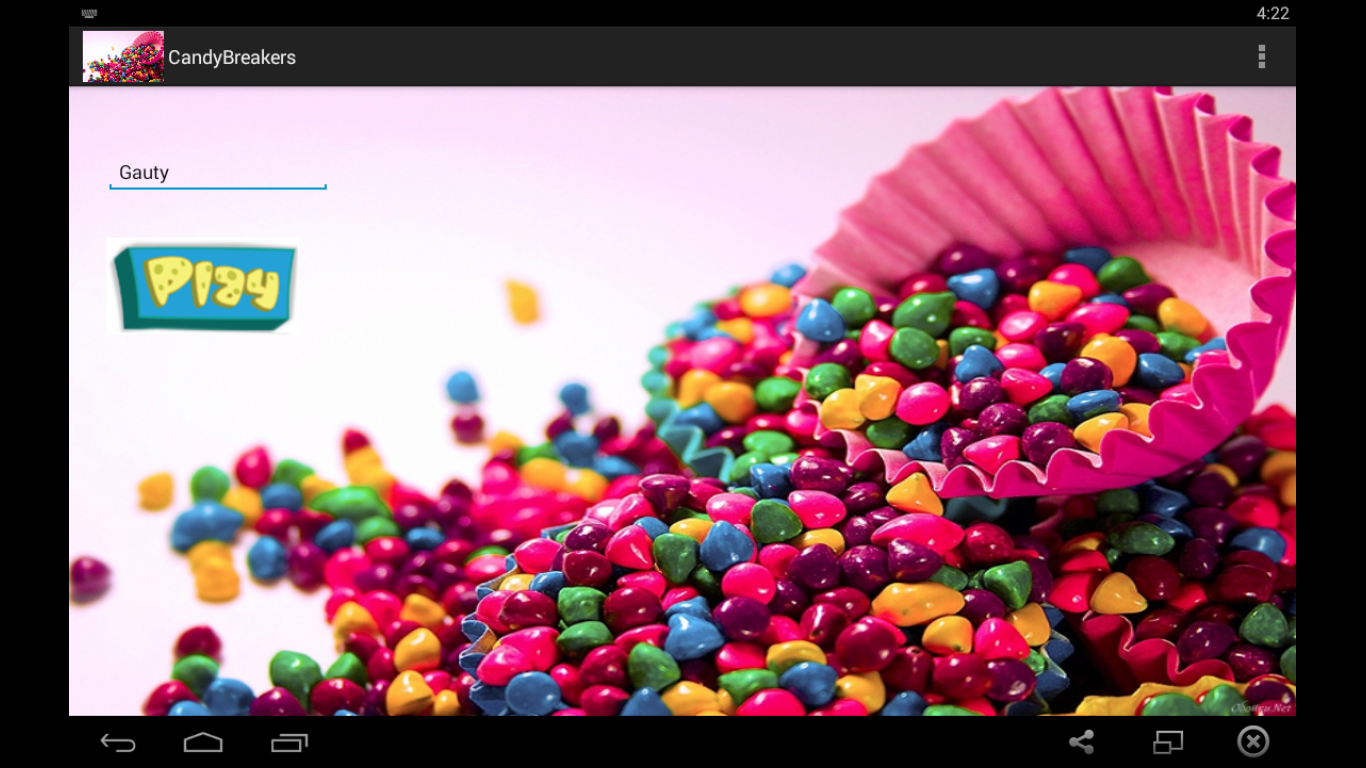
</application>

</manifest>

**Note : Save angrybird.ttf File into asset Folder**

**Screenshots**

**First Screen**



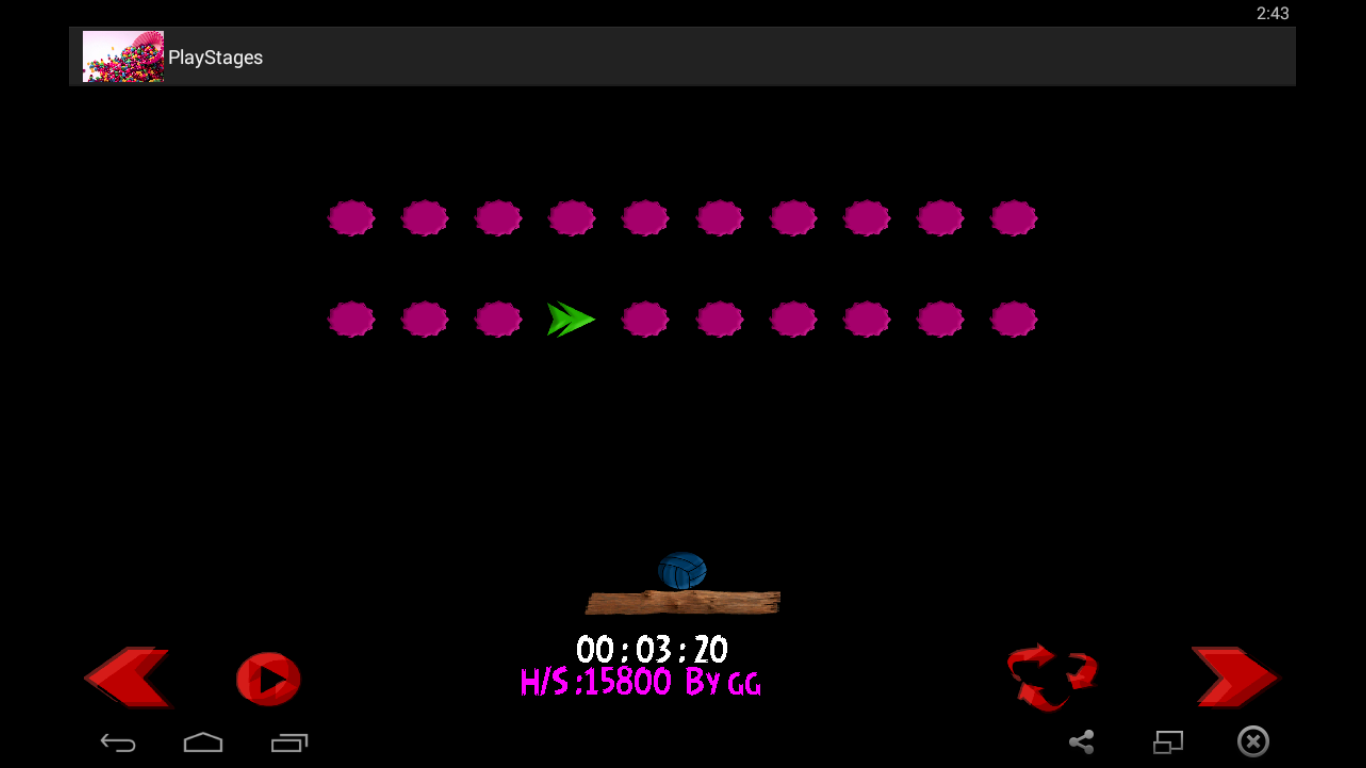
**Select Level Screen**



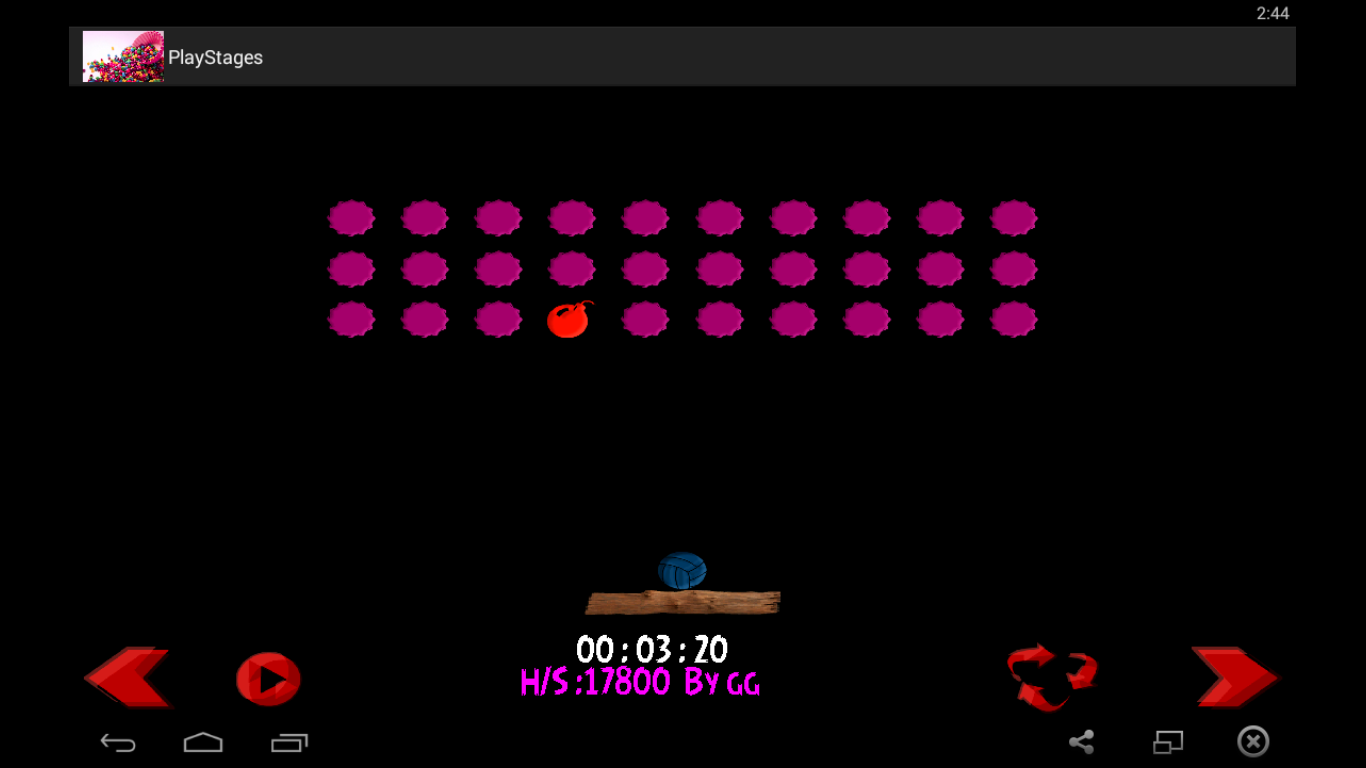
**Level 1**



**Level 2**



**Level 3**



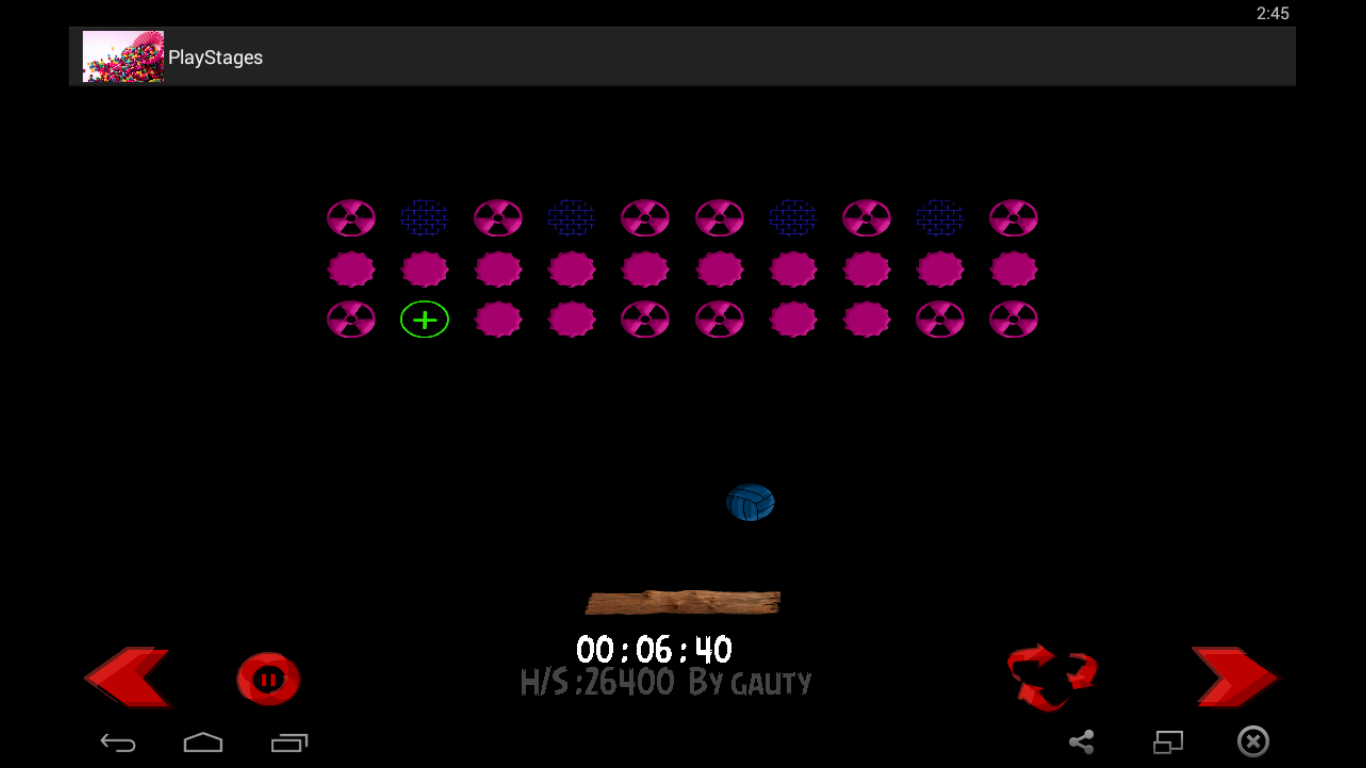
**Level 4**



**Level 5**



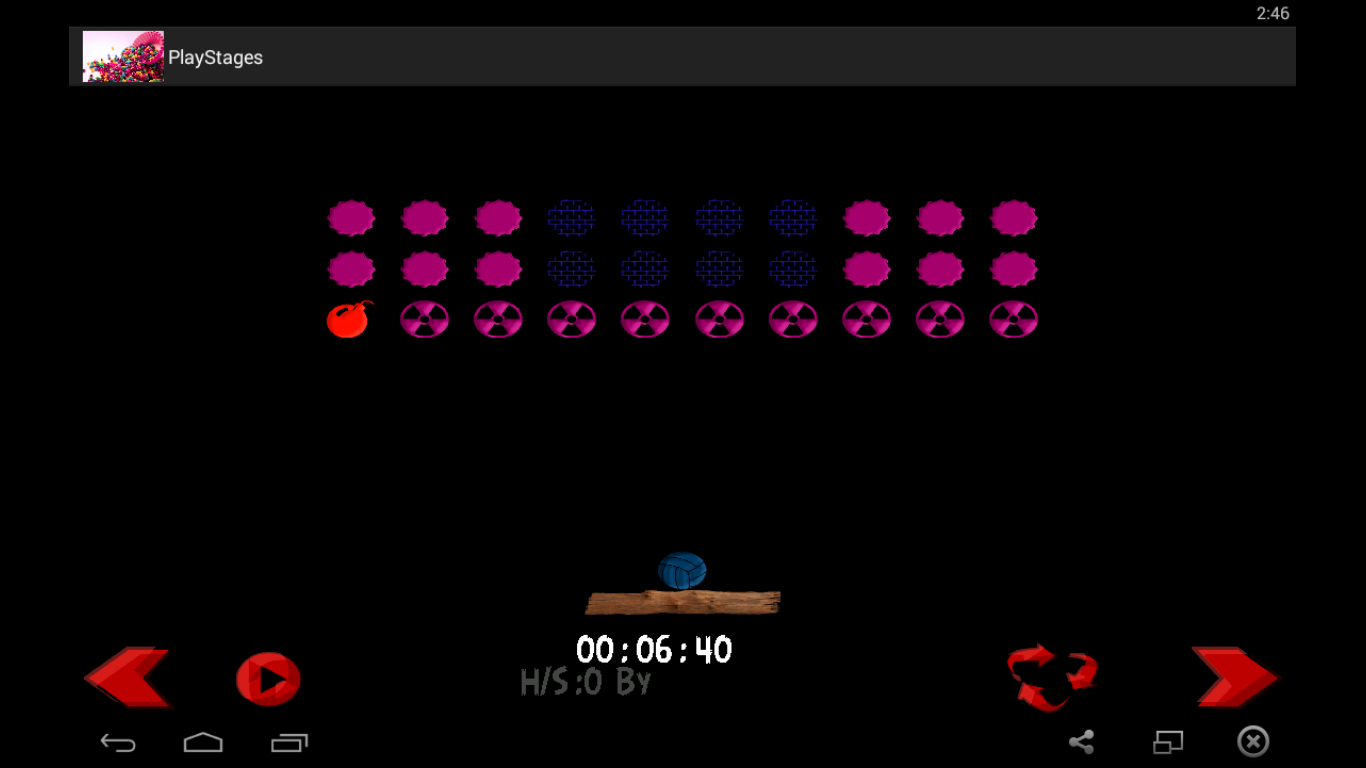
**Level 6**



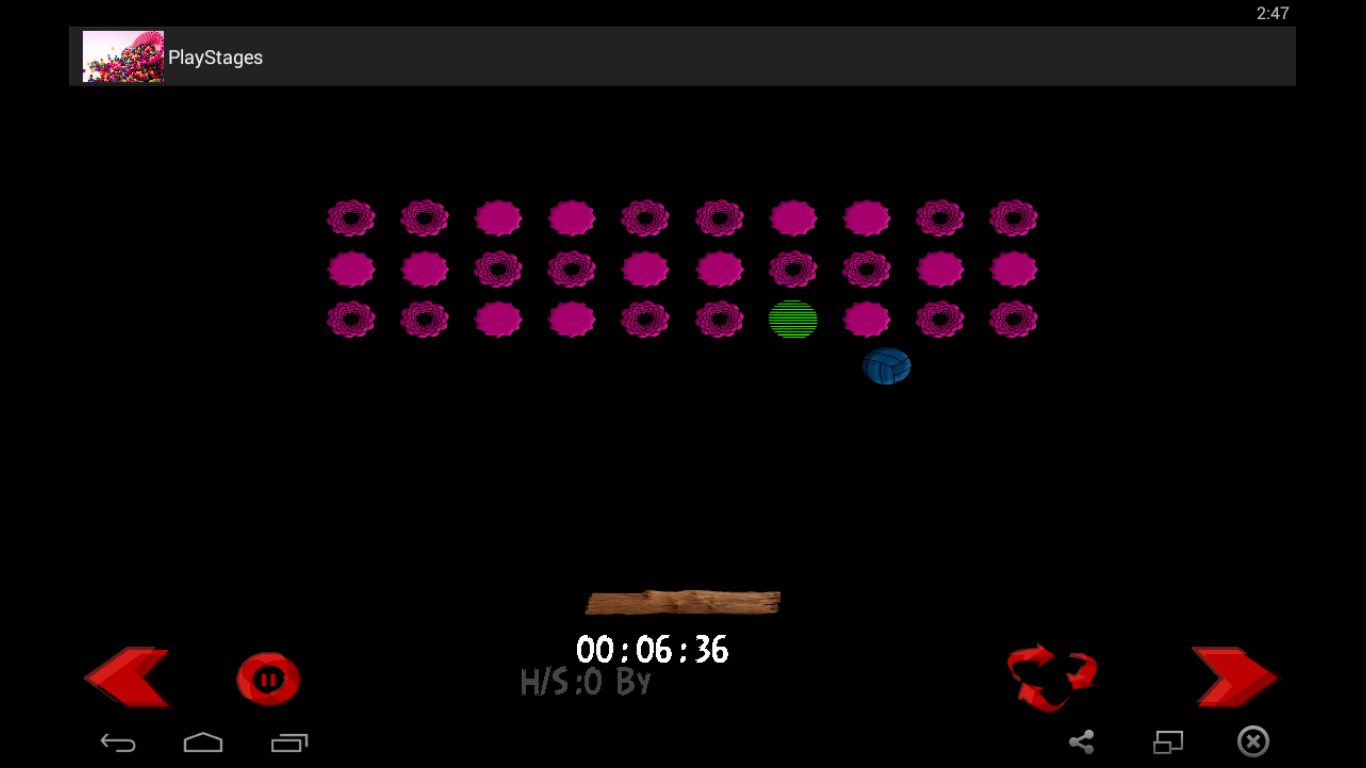
**Level 7**



**Level 8**



**Level 9**



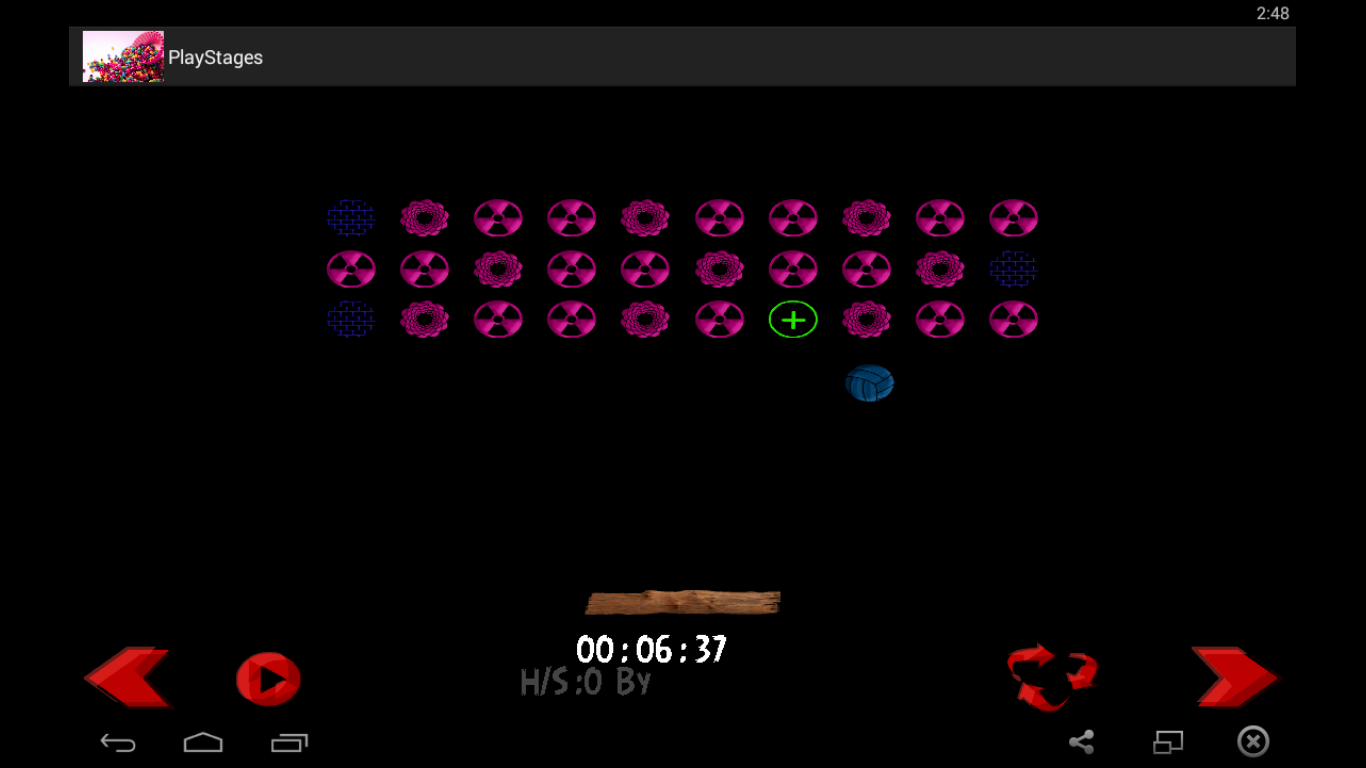
**Level 10**



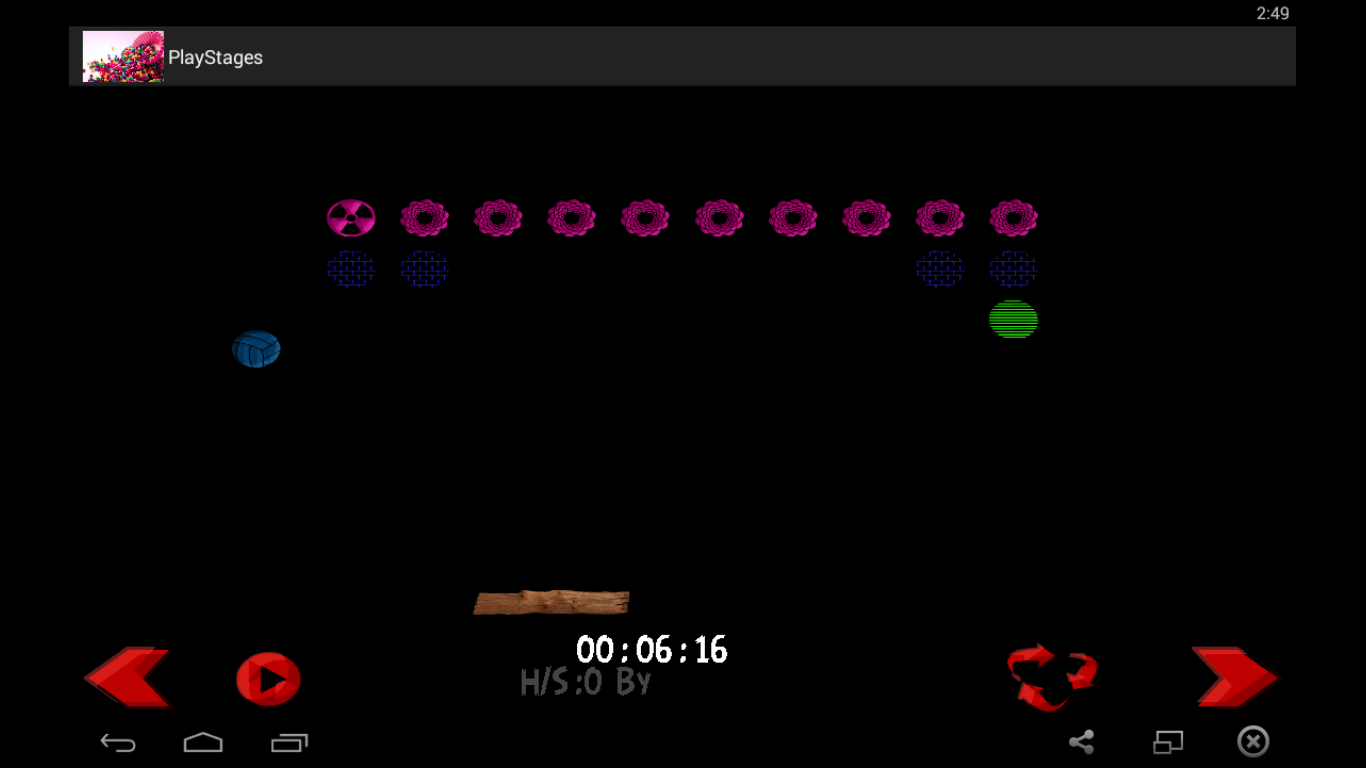
**Level 11**



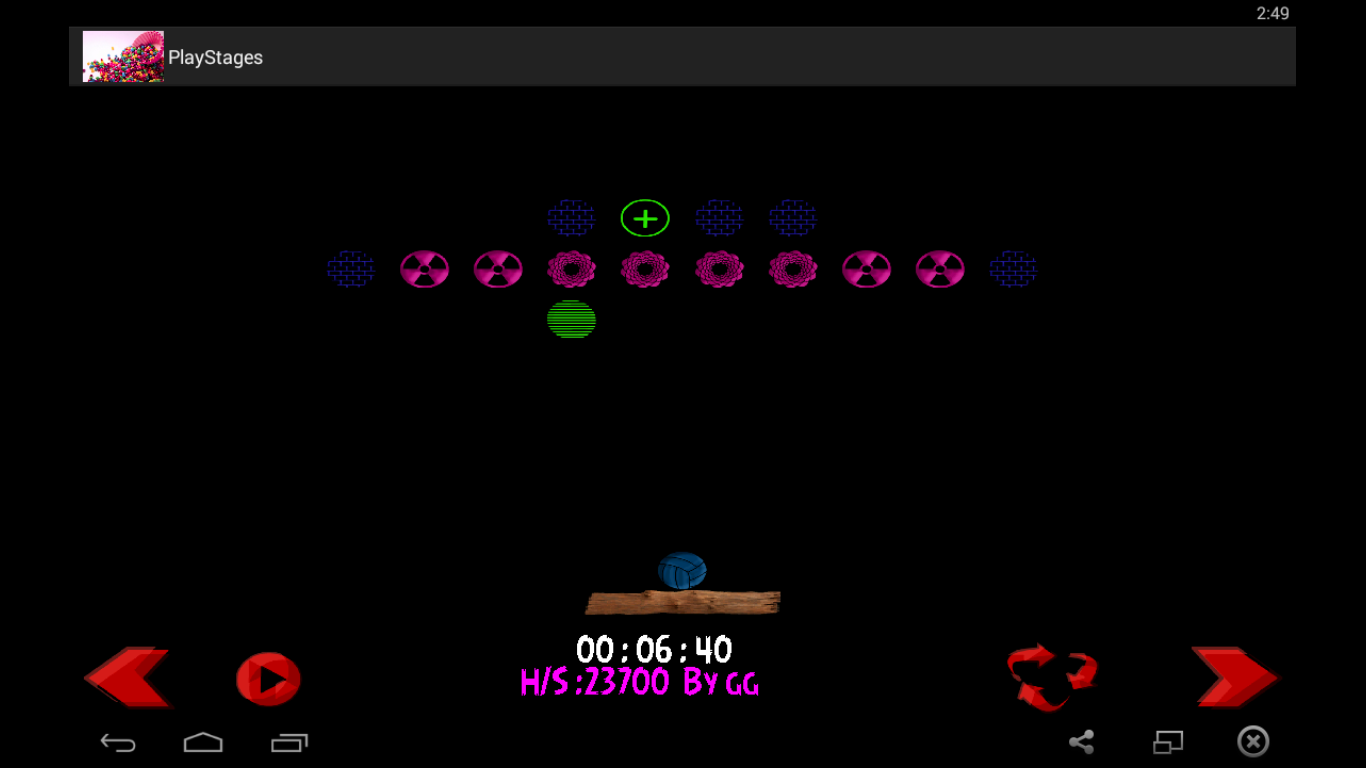
**Level 12**



**Level 13**



**Level 14**



**Level 15**



**Level Completed**



**All Level Completed**



**Game Over**

