МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

імені ВОЛОДИМИРА ДАЛЯ

Звіт

з лабораторної роботи № 3

з теми: «**Разработка Графического приложения. Использование датчиков**»

Виконав студент:

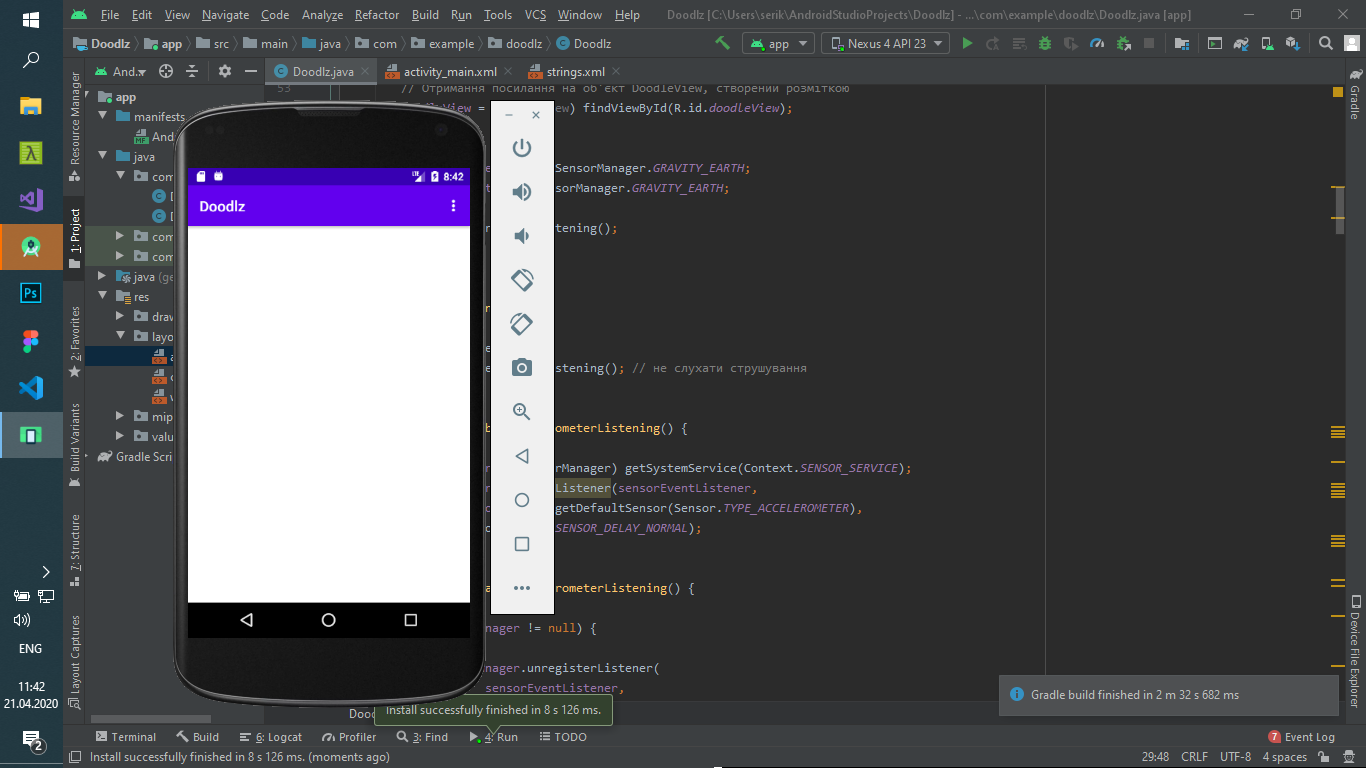
Група КН-18д

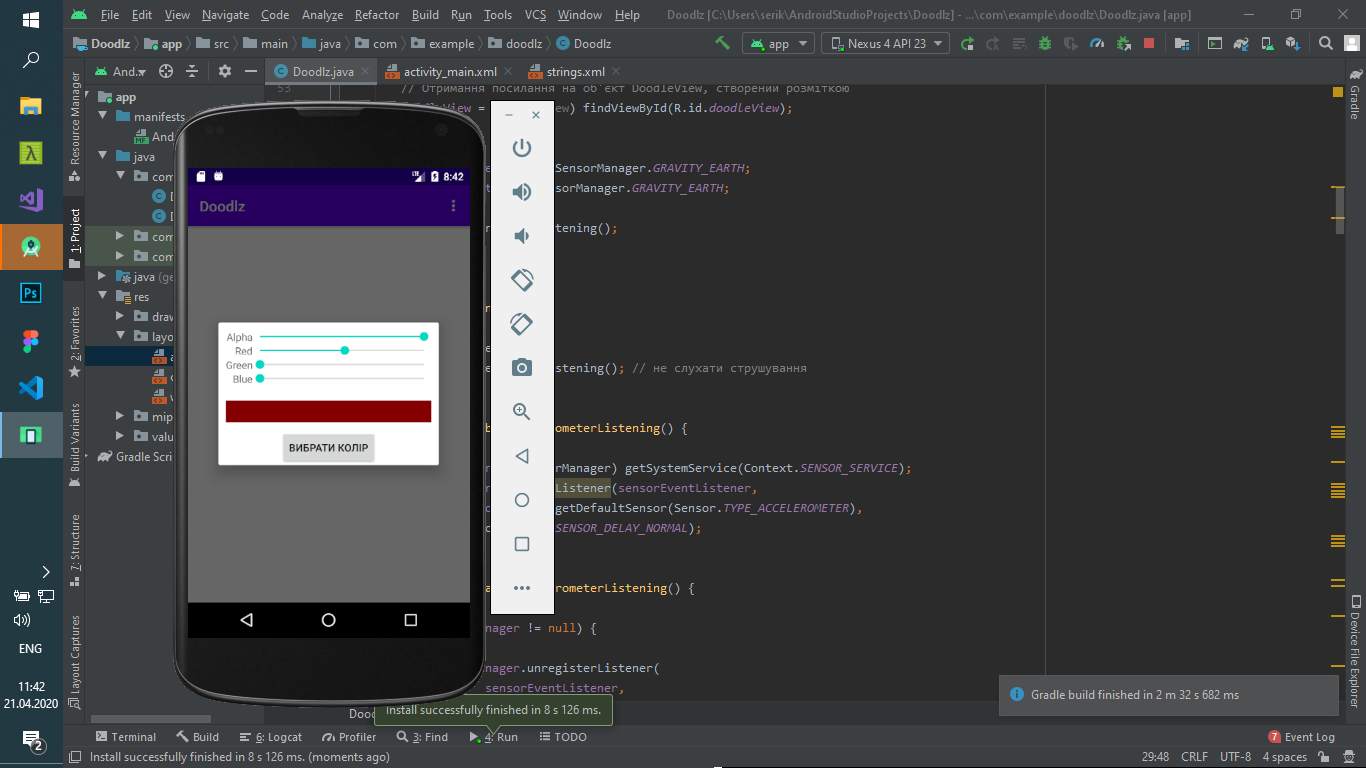
Скороход С.Г.

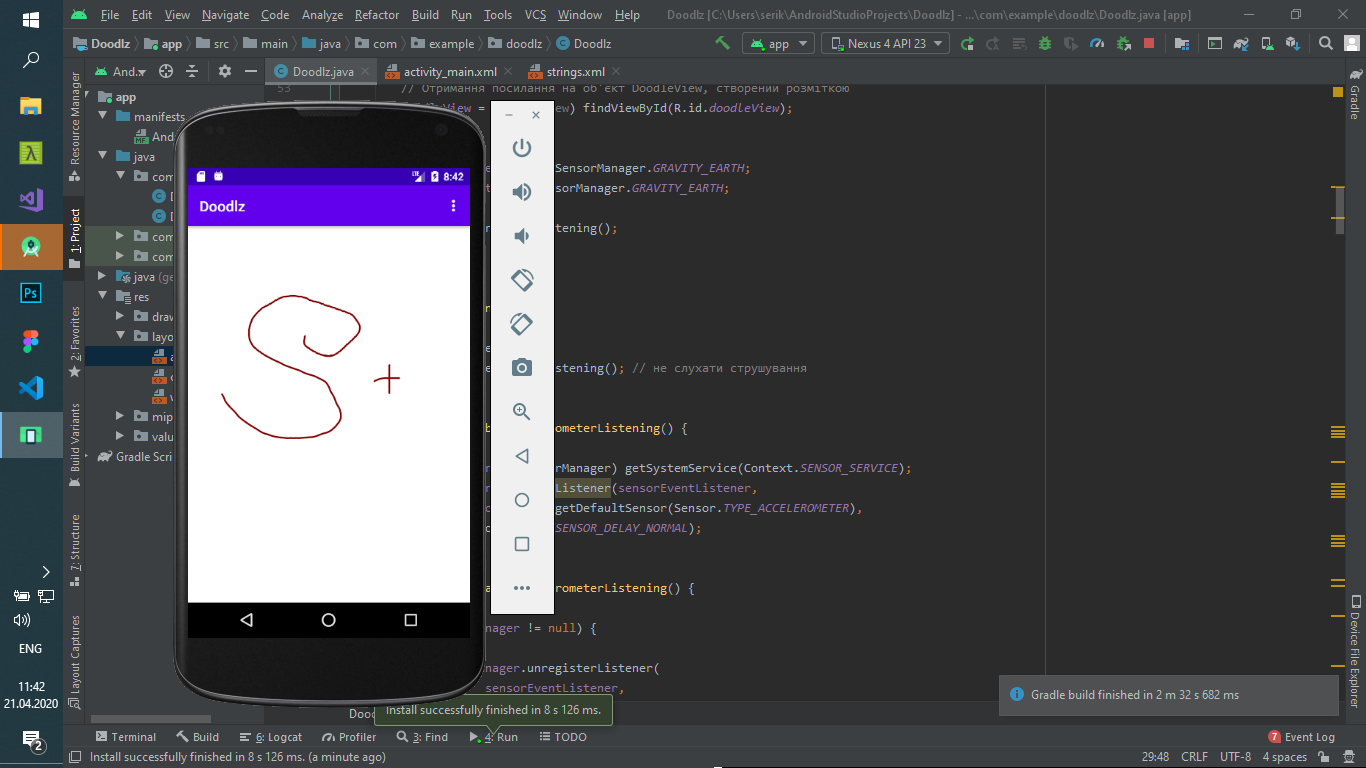
Дата 11.04.2020

**м. Северодонецьк**

2020







Doodle.java

package com.example.doodlz;  
  
import android.app.AlertDialog;  
import android.app.Dialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.hardware.Sensor;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.hardware.SensorManager;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.View.OnClickListener;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.SeekBar;  
import android.widget.SeekBar.OnSeekBarChangeListener;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import java.util.concurrent.atomic.AtomicBoolean;  
  
public class Doodle extends AppCompatActivity {  
  
 private DoodleView doodleView;  
 private SensorManager sensorManager;  
 private float acceleration;  
 private float currentAcceleration;  
 private float lastAcceleration;  
 private AtomicBoolean dialogIsVisible =  
 new AtomicBoolean(); // false  
  
 private static final int *COLOR\_MENU\_ID* = Menu.*FIRST*;  
 private static final int *WIDTH\_MENU\_ID* = Menu.*FIRST* + 1;  
 private static final int *ERASE\_MENU\_ID* = Menu.*FIRST* + 2;  
 private static final int *CLEAR\_MENU\_ID* = Menu.*FIRST* + 3;  
 private static final int *SAVE\_MENU\_ID* = Menu.*FIRST* + 4;  
 private static final int *ACCELERATION\_THRESHOLD* = 15000;  
 private Dialog currentDialog;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 // Отримання посилання на об'єкт DoodleView, створений розміткою  
 doodleView = (DoodleView) findViewById(R.id.*doodleView*);  
  
 acceleration = 0.00f;  
 currentAcceleration = SensorManager.*GRAVITY\_EARTH*;  
 lastAcceleration = SensorManager.*GRAVITY\_EARTH*;  
  
 enableAccelerometerListening();  
 }  
  
 @Override  
 protected void onPause() {  
  
 super.onPause();  
 disableAccelerometerListening(); // не слухати струшування  
 }  
  
 private void enableAccelerometerListening() {  
  
 sensorManager = (SensorManager) getSystemService(Context.*SENSOR\_SERVICE*);  
 sensorManager.registerListener(sensorEventListener,  
 sensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*),  
 SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
  
 private void disableAccelerometerListening() {  
  
 if (sensorManager != null) {  
  
 sensorManager.unregisterListener(  
 sensorEventListener,  
 sensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*));  
 sensorManager = null;  
 }  
 }  
  
 private SensorEventListener = new SensorEventListener() {  
  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
  
 if (!dialogIsVisible.get()) {  
  
 float x = event.values[0];  
 float y = event.values[1];  
 float z = event.values[2];  
  
 lastAcceleration = currentAcceleration;  
 currentAcceleration = x \* x + y \* y + z \* z;  
  
 // Обчислити зміну прискорення  
 acceleration = currentAcceleration \*  
 (currentAcceleration - lastAcceleration);  
  
 // Якщо прискорення вище певного порогу  
 if (acceleration > *ACCELERATION\_THRESHOLD*) {  
  
 // Створити об'єкт AlertDialog.Builder  
 builder =  
 new AlertDialog.Builder(Doodlz.this);  
  
 // Встановити повідомлення для вікна оповіщення  
 builder.setMessage(R.string.*message\_erase*);  
 builder.setCancelable(true);  
  
 // Додати кнопку підтвердження очищення  
 builder.setPositiveButton(R.string.*button\_erase*,  
 new DialogInterface.OnClickListener() {  
  
 public void onClick(DialogInterface dialog,  
 int id) {  
 dialogIsVisible.set(false);  
 doodleView.clear(); // очистка экрана  
 }  
 }  
 );  
  
 builder.setNegativeButton(R.string.*button\_cancel*, new DialogInterface.OnClickListener() {  
  
 public void onClick(DialogInterface dialog,  
 int id) {  
 dialogIsVisible.set(false);  
 dialog.cancel(); // закрити діалог  
 }  
 }  
 );  
  
 dialogIsVisible.set(true); // ознака, що діалог  
 // на екрані  
 builder.show(); // відображення діалогу  
 }  
 }  
 }  
  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
  
 }  
 };  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
  
 super.onCreateOptionsMenu(menu);  
 menu.add(Menu.*NONE*, *COLOR\_MENU\_ID*, Menu.*NONE*,  
 R.string.*menuitem\_color*);  
 menu.add(Menu.*NONE*, *WIDTH\_MENU\_ID*, Menu.*NONE*,  
 R.string.*menuitem\_line\_width*);  
 menu.add(Menu.*NONE*, *ERASE\_MENU\_ID*, Menu.*NONE*,  
 R.string.*menuitem\_erase*);  
 menu.add(Menu.*NONE*, *CLEAR\_MENU\_ID*, Menu.*NONE*,  
 R.string.*menuitem\_clear*);  
 menu.add(Menu.*NONE*, *SAVE\_MENU\_ID*, Menu.*NONE*,  
 R.string.*menuitem\_save\_image*);  
  
 return true; // створення меню виконано успішно  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
  
 switch (item.getItemId()) {  
  
 case *COLOR\_MENU\_ID*:  
 showColorDialog(); // діалог вибору кольору  
 return true; // погасити подію меню  
 case *WIDTH\_MENU\_ID*:  
 showLineWidthDialog(); // діалог товщини лінії  
 return true; // погасити подію меню  
 case *ERASE\_MENU\_ID*:  
 doodleView.setDrawingColor(Color.*WHITE*); // білий колір  
 return true; // погасити подію меню  
 case *CLEAR\_MENU\_ID*:  
 doodleView.clear(); // очищення екрану  
 return true; // погасити подію меню  
 case *SAVE\_MENU\_ID*:  
 doodleView.saveImage(); // збереження зображення  
 return true; // погасити подію меню  
 }  
 return super.onOptionsItemSelected(item); // виклик методу  
 // базового класу  
 }  
  
 private void showColorDialog() {  
 currentDialog = new Dialog(this);  
 currentDialog.setContentView(R.layout.*color\_dialog*);  
 currentDialog.setTitle(R.string.*title\_color\_dialog*);  
 currentDialog.setCancelable(true);  
 final SeekBar alphaSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*alphaSeekBar*);  
 final SeekBar redSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*redSeekBar*);  
 final SeekBar greenSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*greenSeekBar*);  
 final SeekBar blueSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*blueSeekBar*);  
  
 // Реєстрація інтерфейсу для обробки подій повзунків  
 alphaSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);  
 redSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);  
 greenSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);  
 blueSeekBar.setOnSeekBarChangeListener(colorSeekBarChanged);  
  
 final int color = doodleView.getDrawingColor();  
 alphaSeekBar.setProgress(Color.*alpha*(color));  
 redSeekBar.setProgress(Color.*red*(color));  
 greenSeekBar.setProgress(Color.*green*(color));  
 blueSeekBar.setProgress(Color.*blue*(color));  
  
 // Реєстрація інтерфейсу для обробки подій кнопки "Set Color"  
 Button setColorButton = (Button) currentDialog.findViewById(  
 R.id.*setColorButton*);  
 setColorButton.setOnClickListener(setColorButtonListener);  
 dialogIsVisible.set(true); // ознака, що діалог на екрані  
 currentDialog.show(); // відображення діалогу  
 }  
  
 private OnSeekBarChangeListener colorSeekBarChanged = new OnSeekBarChangeListener() {  
  
 @Override  
 public void onProgressChanged(SeekBar, int progress,  
 boolean fromUser) {  
  
 SeekBar alphaSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*alphaSeekBar*);  
 SeekBar redSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*redSeekBar*);  
 SeekBar greenSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*greenSeekBar*);  
 SeekBar blueSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*blueSeekBar*);  
 View colorView =  
 (View) currentDialog.findViewById(R.id.*colorView*);  
  
 colorView.setBackgroundColor(Color.*argb*(  
 alphaSeekBar.getProgress(), redSeekBar.getProgress(),  
 greenSeekBar.getProgress(), blueSeekBar.getProgress()));  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 };  
  
 private OnClickListener setColorButtonListener =  
 new OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 SeekBar alphaSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*alphaSeekBar*);  
 SeekBar redSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*redSeekBar*);  
 SeekBar greenSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*greenSeekBar*);  
 SeekBar blueSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*blueSeekBar*);  
  
 // Установка кольору лінії для малювання  
 doodleView.setDrawingColor(Color.*argb*(  
 alphaSeekBar.getProgress(), redSeekBar.getProgress(),  
 greenSeekBar.getProgress(), blueSeekBar.getProgress()));  
 dialogIsVisible.set(false);//ознака, діалогу на екрані немає  
 currentDialog.dismiss(); // закриття діалогу  
 currentDialog = null; // обнуління покажчика на діалог  
 }  
 };  
  
 private void showLineWidthDialog() {  
  
 // Створення об'єкта вікна і наповнення (роздування) його вмістом  
 currentDialog = new Dialog(this);  
 currentDialog.setContentView(R.layout.*width\_dialog*);  
 currentDialog.setTitle(R.string.*title\_line\_width\_dialog*);  
 currentDialog.setCancelable(true);  
  
 SeekBar widthSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*widthSeekBar*);  
 widthSeekBar.setOnSeekBarChangeListener(widthSeekBarChanged);  
 widthSeekBar.setProgress(doodleView.getLineWidth());  
 Button setLineWidthButton =  
 (Button) currentDialog.findViewById(R.id.*widthDialogDoneButton*);  
 setLineWidthButton.setOnClickListener(setLineWidthButtonListener);  
  
 dialogIsVisible.set(true); // ознака, що діалог на екрані  
 currentDialog.show(); // відображення діалогу  
 }  
  
 private OnSeekBarChangeListener widthSeekBarChanged =  
 new OnSeekBarChangeListener() {  
  
 Bitmap = Bitmap.*createBitmap*( // створити Bitmap  
 400, 100, Bitmap.Config.*ARGB\_8888*);  
 Canvas = new Canvas(bitmap); // пов'язати з Canvas  
  
 @Override  
 public void onProgressChanged(SeekBar, int progress,  
 boolean fromUser) {  
  
 ImageView widthImageView = (ImageView)  
 currentDialog.findViewById(R.id.*widthImageView*);  
  
 Paint p = new Paint();  
 p.setColor(doodleView.getDrawingColor());  
 p.setStrokeCap(Paint.Cap.*ROUND*);  
 p.setStrokeWidth(progress);  
 bitmap.eraseColor(Color.*WHITE*);  
 canvas.drawLine(30, 50, 370, 50, p);  
 widthImageView.setImageBitmap(bitmap);  
 }  
  
 // Порожня реалізація обов'язкового методу інтерфейсу  
 @Override  
 public void onStartTrackingTouch(SeekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar) {  
  
 }  
 };  
  
 private OnClickListener setLineWidthButtonListener =  
 new OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
  
 SeekBar widthSeekBar =  
 (SeekBar) currentDialog.findViewById(R.id.*widthSeekBar*);  
  
 // Установка ширини лінії  
 doodleView.setLineWidth(widthSeekBar.getProgress());  
 dialogIsVisible.set(false); // ознака, що діалогу  
 // немає на екрані  
 currentDialog.dismiss(); // закриття діалогу  
 currentDialog = null; // обнуління покажчика на діалог  
  
 }  
 };  
}

doodleView.java

package com.example.doodlz;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Path;  
import android.graphics.Point;  
import android.net.Uri;  
import android.provider.MediaStore.Images;  
import android.util.AttributeSet;  
import android.view.Gravity;  
import android.view.MotionEvent;  
import android.view.View;  
import android.widget.Toast;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.util.HashMap;  
  
public class DoodleView extends View {  
  
 private static final float *TOUCH\_TOLERANCE* = 10;  
  
 private Bitmap;  
 private Canvas bitmapCanvas;  
 private Paint paintScreen;  
 private Paint paintLine;  
 private HashMap<Integer, Path> pathMap;  
 private HashMap<Integer, Point> previousPointMap;  
  
 public DoodleView(Context, AttributeSet attrs) {  
  
 super(context, attrs);  
  
 paintScreen = new Paint();  
 paintLine = new Paint();  
 paintLine.setAntiAlias(true);  
 paintLine.setColor(Color.*BLACK*);  
 paintLine.setStyle(Paint.Style.*STROKE*);  
 paintLine.setStrokeWidth(5);  
 paintLine.setStrokeCap(Paint.Cap.*ROUND*);  
 pathMap = new HashMap<Integer, Path>();  
 previousPointMap = new HashMap<Integer, Point>();  
 }  
  
 @Override  
 public void onSizeChanged(int w, int h, int oldW, int oldH) {  
 bitmap = Bitmap.*createBitmap*(getWidth(), getHeight(),  
 Bitmap.Config.*ARGB\_8888*);  
 bitmapCanvas = new Canvas(bitmap);  
 bitmap.eraseColor(Color.*WHITE*);  
 }  
 public void clear(){  
  
 pathMap.clear();  
 previousPointMap.clear();  
 bitmap.eraseColor(Color.*WHITE*);  
 invalidate();  
 }  
  
 public void setDrawingColor ( int color){  
  
 paintLine.setColor(color);  
 }  
  
 public int getDrawingColor () {  
  
 return paintLine.getColor();  
 }  
  
 public void setLineWidth ( int width){  
  
 paintLine.setStrokeWidth(width);  
 }  
  
 public int getLineWidth () {  
  
 return (int) paintLine.getStrokeWidth();  
 }  
 @Override  
 protected void onDraw (Canvas canvas){  
 canvas.drawBitmap(bitmap, 0, 0, paintScreen);  
 for (Integer key : pathMap.keySet()) {  
 canvas.drawPath(pathMap.get(key), paintLine);  
 }  
 }  
  
 @Override  
 public boolean onTouchEvent (MotionEvent event){  
  
 int action = event.getActionMasked();  
 int actionIndex = event.getActionIndex();  
  
 if (action == MotionEvent.*ACTION\_DOWN* ||  
 action == MotionEvent.*ACTION\_POINTER\_DOWN*) {  
 touchStarted(event.getX(actionIndex),  
 event.getY(actionIndex),  
 event.getPointerId(actionIndex));  
 } else if (action == MotionEvent.*ACTION\_UP* ||  
 action == MotionEvent.*ACTION\_POINTER\_UP*) {  
 touchEnded(event.getPointerId(actionIndex));  
 } else {  
  
 touchMoved(event);  
 }  
  
 invalidate();  
 return true;  
 }  
 private void touchStarted ( float x, float y, int lineID){  
  
 Path path;  
 Point point;  
 if (pathMap.containsKey(lineID)) {  
  
 path = pathMap.get(lineID);  
 path.reset();  
 point = previousPointMap.get(lineID);  
 } else {  
  
 path = new Path();  
 pathMap.put(lineID, path);  
 point = new Point();  
 previousPointMap.put(lineID, point);  
 }  
  
 path.moveTo(x, y);  
 point.x = (int) x;  
 point.y = (int) y;  
 }  
 private void touchMoved (MotionEvent event) {  
 for (int i = 0; i < event.getPointerCount(); i++) {  
  
 int pointerID = event.getPointerId(i);  
 int pointerIndex = event.findPointerIndex(pointerID);  
  
 if (pathMap.containsKey(pointerID)) {  
  
 float newX = event.getX(pointerIndex);  
 float newY = event.getY(pointerIndex);  
  
 Path path = pathMap.get(pointerID);  
 Point point = previousPointMap.get(pointerID);  
  
 float deltaX = Math.*abs*(newX - point.x);  
 float deltaY = Math.*abs*(newY - point.y);  
  
 if (deltaX >= *TOUCH\_TOLERANCE* ||  
 deltaY >= *TOUCH\_TOLERANCE*) {  
  
 path.quadTo(point.x, point.y, (newX + point.x) / 2,  
 (newY + point.y) / 2);  
  
 point.x = (int) newX;  
 point.y = (int) newY;  
 }  
 }  
  
 }  
 }  
 private void touchEnded(int lineID){  
  
 Path path = pathMap.get(lineID);  
 assert path != null;  
 bitmapCanvas.drawPath(path, paintLine);  
 path.reset();  
 }  
 public void saveImage(){  
 String fileName = "Doodlzi" + System.*currentTimeMillis*() / 1000;  
 ContentValues values = new ContentValues();  
 values.put(Images.Media.*TITLE*, fileName);  
 values.put(Images.Media.*DATE\_MODIFIED*,  
 System.*currentTimeMillis*() / 1000);  
 values.put(Images.Media.*MIME\_TYPE*, "image/jpg");  
 values.put(Images.Media.*SIZE*, 50000);  
 Uri uri = getContext().getContentResolver().insert(  
 Images.Media.*EXTERNAL\_CONTENT\_URI*, values);  
  
 try {  
  
 OutputStream outStream = getContext().getContentResolver().openOutputStream(uri);  
 bitmap.compress(Bitmap.CompressFormat.*JPEG*, 100, outStream);  
 outStream.flush();  
 outStream.close();  
 Toast message = Toast.*makeText*(getContext(),  
 R.string.*message\_saved*, Toast.*LENGTH\_SHORT*);  
 message.setGravity(Gravity.*CENTER*, message.getXOffset() / 2,  
 message.getYOffset() / 2);  
 message.show();  
 } catch (IOException ex) {  
  
 Toast message = Toast.*makeText*(getContext(),  
 R.string.*message\_error\_saving*, Toast.*LENGTH\_SHORT*);  
 message.setGravity(Gravity.*CENTER*, message.getXOffset() / 2,  
 message.getYOffset() / 2);  
 message.show();  
 }  
 }  
 }

activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<com.example.doodlz.DoodleView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/doodleView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>

Color\_dialog.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/colorDialogLinearLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:minWidth="300dp"  
 android:orientation="vertical">  
  
 <TableLayout  
 android:id="@+id/tableLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:stretchColumns="1">  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:gravity="right"  
 android:text="@string/label\_alpha" />  
  
 <SeekBar  
 android:id="@+id/alphaSeekBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:max="255"  
 android:paddingLeft="10dp"  
 android:paddingRight="10dp" />  
 </TableRow>  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:gravity="right"  
 android:text="@string/label\_red" />  
  
 <SeekBar  
 android:id="@+id/redSeekBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:max="255"  
 android:paddingLeft="10dp"  
 android:paddingRight="10dp" />  
 </TableRow>  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:gravity="right"  
 android:text="@string/label\_green" />  
  
 <SeekBar  
 android:id="@+id/greenSeekBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:max="255"  
 android:paddingLeft="10dp"  
 android:paddingRight="10dp" />  
 </TableRow>  
  
 <TableRow  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:gravity="right"  
 android:text="@string/label\_blue" />  
  
 <SeekBar  
 android:id="@+id/blueSeekBar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:max="255"  
 android:paddingLeft="10dp"  
 android:paddingRight="10dp" />  
 </TableRow>  
 </TableLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:background="@android:color/white">  
  
 <View  
 android:id="@+id/colorView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="30dp" />  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/setColorButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="@string/button\_set\_color" />  
</LinearLayout>

Width\_dialog.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/widthDialogLinearLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:minWidth="300dp"  
 android:orientation="vertical">  
  
 <ImageView  
 android:id="@+id/widthImageView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:layout\_margin="10dp" />  
  
 <SeekBar  
 android:id="@+id/widthSeekBar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_margin="20dp"  
 android:max="50"  
 android:paddingLeft="20dp"  
 android:paddingRight="20dp" />  
  
 <Button  
 android:id="@+id/widthDialogDoneButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="@string/button\_set\_line\_width" />  
</LinearLayout>

String.xml

<resources>  
 <string name="app\_name">Doodlz</string>  
  
 <string name="action\_settings">Настройки</string>  
 <string name="button\_erase">Стерти</string>  
 <string name="button\_cancel">Відміна</string>  
 <string name="button\_set\_color">Вибрати колір</string>  
 <string name="button\_set\_line\_width">Вибрати товщину лінії</string>  
 <string name="label\_alpha">Alpha</string>  
 <string name="label\_red">Red</string>  
 <string name="label\_green">Green</string>  
 <string name="label\_blue">Blue</string>  
 <string name="menuitem\_clear">Стерти малюнок</string>  
 <string name="menuitem\_color">Колір</string>  
 <string name="menuitem\_erase">Гумка</string>  
 <string name="menuitem\_line\_width">Товщина лінії</string>  
 <string name="menuitem\_save\_image">Зберегти малюнок</string>  
 <string name="message\_erase">Стерти малюнок?</string>  
 <string name="message\_error\_saving">Не вдалося зберегти</string>  
 <string name="message\_saved">Малюнок збережено</string>  
 <string name="title\_color\_dialog">Виберіть колір</string>  
 <string name="title\_line\_width\_dialog">Виберіть товщину лінії</string>  
</resources>