基于Android的绘图分享软件 V1.0

源程序

武汉科技大学

2016-11-25

package com.example.activity;

import java.io.File;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import cn.bmob.v3.BmobUser;

import cn.bmob.v3.datatype.BmobFile;

import cn.bmob.v3.listener.SaveListener;

import cn.bmob.v3.listener.UploadFileListener;

import com.example.entity.Post;

import com.example.quick.R;

import com.example.utils.GetPath;

import com.example.utils.ImageUtils;

import com.example.utils.RecordTopic;

import com.example.utils.Toasts;

/\*\*

\* 添加说说界面

\*/

public class AddPostActivity extends BaseActivity implements OnClickListener {

private TextView back;

private TextView publish;

private EditText title;

private EditText content;

private LinearLayout take;

private LinearLayout open;

private ImageView img;

private Uri uri = null;

private Post post;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_post);

initView();

}

private void initView() {

back = (TextView) findViewById(R.id.tv\_back);

back.setOnClickListener(this);

publish = (TextView) findViewById(R.id.tv\_publish);

publish.setOnClickListener(this);

title = (EditText) findViewById(R.id.et\_title);

content = (EditText) findViewById(R.id.et\_content);

take = (LinearLayout) findViewById(R.id.take\_layout);

take.setOnClickListener(this);

open = (LinearLayout) findViewById(R.id.open\_layout);

open.setOnClickListener(this);

img = (ImageView) findViewById(R.id.img\_post);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.tv\_back:

finish();

break;

case R.id.tv\_publish:

publish();

break;

case R.id.take\_layout:

ImageUtils.PickImageFromCamera(this);

break;

case R.id.open\_layout:

ImageUtils.PickImageFromAlbum(this);

break;

default:

break;

}

}

/\*\*

\* 上传内容

\*/

private void publish() {

if (title.getText().toString().equals("")

|| content.getText().toString().equals("")) {

Toasts.toastShort(getApplicationContext(), "内容不能为空");

} else {

if (uri != null) {

String path = GetPath.getPath(getApplicationContext(), uri);

final BmobFile file = new BmobFile(new File(path));

file.upload(this, new UploadFileListener() {

@Override

public void onSuccess() {

post = new Post();

post.setTitle(title.getText().toString());

post.setContent(content.getText().toString());

post.setUserID(BmobUser.getCurrentUser(

getApplicationContext()).getObjectId());

post.setHavaPic(true);

post.setTopic(RecordTopic

.getTopic(getApplicationContext()));

post.setLoveCount(0);

post.setPic(file);

post.save(getApplicationContext(), new SaveListener() {

@Override

public void onSuccess() {

Toasts.toastShort(getApplicationContext(),

"发布成功");

Intent i = new Intent(AddPostActivity.this,

PostActivity.class);

startActivity(i);

}

@Override

public void onFailure(int arg0, String arg1) {

Toasts.toastShort(getApplicationContext(),

"error");

}

});

}

@Override

public void onFailure(int arg0, String arg1) {

}

});

} else {

post = new Post();

post.setTitle(title.getText().toString());

post.setContent(content.getText().toString());

post.setUserID(BmobUser.getCurrentUser(getApplicationContext())

.getObjectId());

post.setHavaPic(false);

post.setTopic(RecordTopic.getTopic(getApplicationContext()));

post.setUserName(BmobUser.getCurrentUser(

getApplicationContext()).getUsername());

post.setLoveCount(0);

post.setPic(null);

post.save(getApplicationContext(), new SaveListener() {

@Override

public void onSuccess() {

Toasts.toastShort(getApplicationContext(), "发布成功");

startActivity(new Intent(AddPostActivity.this,

PostActivity.class));

}

@Override

public void onFailure(int arg0, String arg1) {

Toasts.toastShort(getApplicationContext(), "error");

}

});

}

}

}

/\*\*

\* 处理拍照或相册返回结果

\*/

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

// TODO Auto-generated method stub

super.onActivityResult(requestCode, resultCode, data);

switch (requestCode) {

case ImageUtils.REQUEST\_CODE\_FROM\_ALBUM: // 相册

if (resultCode == RESULT\_CANCELED) {

return;

}

uri = data.getData();

img.setImageURI(uri);

break;

case ImageUtils.REQUEST\_CODE\_FROM\_CAMERA: // 相机

if (resultCode == RESULT\_CANCELED) {

ImageUtils.deleteImageUri(getApplicationContext(),

ImageUtils.imageUriFromCamera);

} else {

uri = ImageUtils.imageUriFromCamera;

img.setImageURI(uri);

}

break;

default:

break;

}

}

}

package com.example.activity;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.EditText;

import android.widget.TextView;

import cn.bmob.v3.BmobUser;

import cn.bmob.v3.listener.SaveListener;

import com.example.entity.Topic;

import com.example.quick.R;

import com.example.utils.Toasts;

/\*\*

\* 添加话题界面

\*/

public class AddTopicActivity extends BaseActivity implements OnClickListener {

private EditText topic;

private TextView back;

private TextView addTopic;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_topic);

initView();

}

private void initView() {

topic = (EditText) findViewById(R.id.et\_topic);

back = (TextView) findViewById(R.id.tv\_back);

back.setOnClickListener(this);

addTopic = (TextView) findViewById(R.id.tv\_add\_topic);

addTopic.setOnClickListener(this);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.tv\_back:

finish();

break;

case R.id.tv\_add\_topic:

addTopic();

break;

default:

break;

}

}

/\*\*

\* 添加话题操作

\*/

private void addTopic() {

if (topic.getText().toString().equals("")) {

Toasts.toastShort(getApplicationContext(), "话题不能为空");

} else {

Topic t = new Topic();

t.setTopic(topic.getText().toString());

t.setOwnerID(BmobUser.getCurrentUser(getApplicationContext())

.getObjectId());

t.save(getApplicationContext(), new SaveListener() {

@Override

public void onSuccess() {

Toasts.toastShort(getApplicationContext(), "创建话题成功");

finish();

}

@Override

public void onFailure(int arg0, String arg1) {

}

});

}

}

}

package com.example.activity;

import com.example.utils.ActivityManager;

import android.app.Activity;

import android.os.Bundle;

/\*\*

\* 基础Activity

\*/

public class BaseActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

ActivityManager.getInstance().addActivity(this);// 添加新activity

super.onCreate(savedInstanceState);

}

@Override

protected void onDestroy() {

super.onDestroy();

ActivityManager.getInstance().remove(this); // 移除activity

}

}

package com.example.activity;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import android.app.Activity;

import android.content.ContentResolver;

import android.content.Intent;

import android.content.IntentFilter;

import android.content.SharedPreferences;

import android.content.res.Configuration;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.drawable.BitmapDrawable;

import android.media.MediaScannerConnection;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import android.os.Environment;

import android.util.Log;

import android.view.Gravity;

import android.view.KeyEvent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.CompoundButton;

import android.widget.CompoundButton.OnCheckedChangeListener;

import android.widget.ImageButton;

import android.widget.LinearLayout;

import android.widget.LinearLayout.LayoutParams;

import android.widget.PopupWindow;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import com.example.interfaces.EditTextDialogListener;

import com.example.interfaces.OnClickOkListener;

import com.example.interfaces.PaintViewCallBack;

import com.example.quick.R;

import com.example.utils.AlphaSet;

import com.example.utils.BitMapUtils;

import com.example.utils.GetPath;

import com.example.utils.ImageButtonTools;

import com.example.utils.PaintConstants;

import com.example.utils.PenSizeSet;

import com.example.utils.PaintConstants.ERASER\_SIZE;

import com.example.utils.PaintConstants.PEN\_SIZE;

import com.example.utils.PaintConstants.PEN\_TYPE;

import com.example.utils.PaintConstants.SHAP;

import com.example.utils.SavePathSet;

import com.example.utils.Toasts;

import com.example.view.ColorPickerDialog;

import com.example.view.ColorPickerDialog.OnColorChangedListener;

import com.example.view.ColorView;

import com.example.view.OkCancleDialog;

import com.example.view.OkDialog;

import com.example.view.PaintView;

import com.example.view.SaveDialog;

/\*\*

\* 画板

\*/

public class DrawActivity extends Activity implements OnClickListener {

private static final int SELECT\_PIC\_KITKAT = 0;

private static final int SELECT\_PIC = 1;

// PaintView

private PaintView mPaintView = null;

// button 界面上的各个按钮

private ImageButton saveButton = null;

private ImageButton loadButton = null;

private ImageButton clearButton = null;

private ImageButton eraserButton = null;

private ImageButton colorSelectButton = null;

private ImageButton penSizeButton = null;

private ImageButton undoButton = null;

private ImageButton redoButton = null;

private ImageButton toButtonLayoutButton = null;

private ImageButton toColorLayoutButton = null;

private ImageButton toolButton = null;

// 点击工具按钮弹出浮动菜单上的按钮

private Button backGroundColorButton = null;

private Button plainPaintButton = null;

private Button blurPaintButton = null;

private Button embossButton = null;

// 点击Menu弹出的功能菜单

private ImageButton exitButton = null;

private ImageButton aboutButton = null;

private ImageButton helpButton = null;

// 两个PopWindow

private PopupWindow mPopupWindow = null;

private PopupWindow toolsPopupWindow = null;

// 一共4个ColorView

private ColorView colorView1 = null;

private ColorView colorView2 = null;

private ColorView colorView3 = null;

private ColorView colorView4 = null;

// 通过控制Layout来控制某些变化

private LinearLayout colorLayout = null;

private LinearLayout buttonLayout = null;

private LinearLayout paintViewLayout = null;

private LinearLayout eraserSizeLayout = null;

private LinearLayout penSizeLayout = null;

private LinearLayout shapLayout = null;

private LinearLayout shapLayoutf = null;

// 一些RadioGroup和重要的(也就是默认的)RadioButton

private RadioGroup colorRadioGroup = null;

private RadioGroup eraserSizeRadioGroup = null;

private RadioButton eraserSizeRadio01 = null;

private RadioGroup penSizeRadioGroup = null;

private RadioButton penSizeRadio1 = null;

private RadioGroup shapRadioGroup = null;

private RadioGroup shapRadioGroupf = null;

private RadioButton curvRadioButton = null;

// 用于两个SizeRadioGroup的一些操作

private boolean clearCheckf = false;

private boolean clearCheck = false;

private List<ColorView> mColorViewList = null;

// 使用PenType临时存储选择的变量，当创建时再传给PaintView

private int mPenType = PEN\_TYPE.PLAIN\_PEN;

private String file;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_draw);

init();

}

private void init() {

initLayout();

initButtons();

initColorViews();

initPaintView();

initPopUpWindow();

initCallBack();

initShapRadioGroups();

mPaintView.redo();

}

/\*\*

\* 初始化所有的RadioGroup

\*/

private void initShapRadioGroups() {

shapRadioGroup = (RadioGroup) findViewById(R.id.shapRadioGroup);

curvRadioButton = (RadioButton) findViewById(R.id.RadioButtonShapCurv);

shapRadioGroupf = (RadioGroup) findViewById(R.id.shapRadioGroupf);

initEraseSizeGroup();

initPenSizeGroup();

initShapRadioGroup();

initShapRadioGroupf();

}

private void initShapRadioGroupf() {

shapRadioGroupf

.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

// 如果当前没有选中

if (checkedId == -1) {

return;

}

if (clearCheck == false) {

clearCheckf = true;

shapRadioGroup.clearCheck();

}

mPaintView.setPenStyle(Paint.Style.FILL);

switch (checkedId) {

case R.id.RadioButtonShapLine:

mPaintView.setCurrentShapType(SHAP.LINE);

mPaintView.setPenStyle(Paint.Style.STROKE);

break;

case R.id.RadioButtonShapRectf:

mPaintView.setCurrentShapType(SHAP.RECT);

break;

case R.id.RadioButtonShapCirclef:

mPaintView.setCurrentShapType(SHAP.CIRCLE);

break;

case R.id.RadioButtonShapOvalf:

mPaintView.setCurrentShapType(SHAP.OVAL);

break;

case R.id.RadioButtonShapSquaref:

mPaintView.setCurrentShapType(SHAP.SQUARE);

break;

default:

break;

}

clearCheckf = false;

}

});

}

/\*\*

\* 初始化第一个ShapRadioGroup

\*/

private void initShapRadioGroup() {

curvRadioButton.setChecked(true);

shapRadioGroup

.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

// 如果当前没有选中

if (checkedId == -1) {

return;

}

/\*\*

\* 不需要每次都调用

\*/

if (clearCheckf == false) {

clearCheck = true;

shapRadioGroupf.clearCheck();

}

mPaintView.setPenStyle(Paint.Style.STROKE);

switch (checkedId) {

case R.id.RadioButtonShapCurv:

mPaintView.setCurrentShapType(SHAP.CURV);

break;

case R.id.RadioButtonShapRect:

mPaintView.setCurrentShapType(SHAP.RECT);

break;

case R.id.RadioButtonShapCircle:

mPaintView.setCurrentShapType(SHAP.CIRCLE);

break;

case R.id.RadioButtonShapOval:

mPaintView.setCurrentShapType(SHAP.OVAL);

break;

case R.id.RadioButtonShapSquare:

mPaintView.setCurrentShapType(SHAP.SQUARE);

break;

default:

break;

}

clearCheck = false;

}

});

}

/\*\*

\* 初始化负责确定Pensize的radioGroup

\*/

private void initPenSizeGroup() {

penSizeRadioGroup = (RadioGroup) findViewById(R.id.penRaidoGroup);

penSizeRadio1 = (RadioButton) findViewById(R.id.RadioButtonPen01);

penSizeRadio1.setChecked(true);

penSizeRadioGroup

.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

switch (checkedId) {

case R.id.RadioButtonPen01:

mPaintView.setPenSize(PEN\_SIZE.SIZE\_1);

PenSizeSet.recordSize(getApplicationContext(), 1);

break;

case R.id.RadioButtonPen02:

mPaintView.setPenSize(PEN\_SIZE.SIZE\_2);

break;

case R.id.RadioButtonPen03:

mPaintView.setPenSize(PEN\_SIZE.SIZE\_3);

PenSizeSet.recordSize(getApplicationContext(), 4);

break;

case R.id.RadioButtonPen04:

mPaintView.setPenSize(PEN\_SIZE.SIZE\_4);

break;

case R.id.RadioButtonPen05:

mPaintView.setPenSize(PEN\_SIZE.SIZE\_5);

PenSizeSet.recordSize(getApplicationContext(), 6);

break;

default:

break;

}

}

});

}

/\*\*

\* 初始化EraserSize选择布局

\*/

private void initEraseSizeGroup() {

eraserSizeRadioGroup = (RadioGroup) findViewById(R.id.eraseRaidoGroup);

eraserSizeRadio01 = (RadioButton) findViewById(R.id.RadioButtonEraser01);

eraserSizeRadio01.setChecked(true);

eraserSizeRadioGroup

.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(RadioGroup group, int checkedId) {

switch (checkedId) {

case R.id.RadioButtonEraser01:

mPaintView.setEraserSize(ERASER\_SIZE.SIZE\_1);

break;

case R.id.RadioButtonEraser02:

mPaintView.setEraserSize(ERASER\_SIZE.SIZE\_2);

break;

case R.id.RadioButtonEraser03:

mPaintView.setEraserSize(ERASER\_SIZE.SIZE\_3);

break;

case R.id.RadioButtonEraser04:

mPaintView.setEraserSize(ERASER\_SIZE.SIZE\_4);

break;

case R.id.RadioButtonEraser05:

mPaintView.setEraserSize(ERASER\_SIZE.SIZE\_5);

break;

default:

break;

}

}

});

}

/\*\*

\* 初始化paintView的回调函数

\*/

private void initCallBack() {

mPaintView.setCallBack(new PaintViewCallBack() {

// 当画了之后对Button进行更新

@Override

public void onHasDraw() {

enableUndoButton();

disableRedoButton();

}

// 当点击之后让各个弹出的窗口都消失

@Override

public void onTouchDown() {

setAllLayoutInvisable();

}

});

}

/\*\*

\* 初始化popUpWindow

\*/

private void initPopUpWindow() {

initMenuPopup();

initToolPopup();

}

/\*\*

\* 初始化Tool的PopupWidow

\*/

private void initToolPopup() {

View toolsPopup = initToolPopWindowLayout();

initToolPopButtons(toolsPopup);

}

/\*\*

\* 初始化ToolsPopupWindow的布局

\*/

private View initToolPopWindowLayout() {

LayoutInflater mLayoutInflater = (LayoutInflater) this

.getSystemService(LAYOUT\_INFLATER\_SERVICE);

View toolsPopup = mLayoutInflater.inflate(R.layout.tools, null);

toolsPopupWindow = new PopupWindow(toolsPopup,

LayoutParams.FILL\_PARENT, LayoutParams.WRAP\_CONTENT, true);

toolsPopupWindow.setBackgroundDrawable(new BitmapDrawable());

return toolsPopup;

}

/\*\*

\* 初始化ToolsPopupWindows上的Buttons

\*/

private void initToolPopButtons(View toolsPopup) {

plainPaintButton = (Button) toolsPopup

.findViewById(R.id.buttonPlainPen);

setToolButton(plainPaintButton);

blurPaintButton = (Button) toolsPopup.findViewById(R.id.buttonBlurPen);

setToolButton(blurPaintButton);

backGroundColorButton = (Button) toolsPopup

.findViewById(R.id.buttonSelectBackGroundColor);

setToolButton(backGroundColorButton);

embossButton = (Button) toolsPopup.findViewById(R.id.buttonEmboss);

setToolButton(embossButton);

}

/\*\*

\* 初始化Menu的popupWindow

\*/

private void initMenuPopup() {

View menuPopup = initPopLayout();

initMenuPopButtons(menuPopup);

}

/\*\*

\* 初始化MenuPopupWindow的布局

\*/

private View initPopLayout() {

LayoutInflater mLayoutInflater = (LayoutInflater) this

.getSystemService(LAYOUT\_INFLATER\_SERVICE);

View menuPopup = mLayoutInflater.inflate(R.layout.translucent\_button,

null);

mPopupWindow = new PopupWindow(menuPopup, LayoutParams.FILL\_PARENT,

LayoutParams.WRAP\_CONTENT, true);

mPopupWindow.setBackgroundDrawable(new BitmapDrawable());

return menuPopup;

}

/\*\*

\* 初始化MenuPopupWindows上的Buttons

\*/

private void initMenuPopButtons(View menuPopup) {

exitButton = (ImageButton) menuPopup.findViewById(R.id.exitButton);

exitButton

.setImageDrawable(getResources().getDrawable(R.drawable.quit));

exitButton.setOnClickListener(this);

aboutButton = (ImageButton) menuPopup.findViewById(R.id.aboutButton);

aboutButton.setImageDrawable(getResources().getDrawable(

R.drawable.about));

aboutButton.setOnClickListener(this);

helpButton = (ImageButton) menuPopup.findViewById(R.id.helpButton);

helpButton

.setImageDrawable(getResources().getDrawable(R.drawable.help));

helpButton.setOnClickListener(this);

}

/\*\*

\* 设置popupWindow所用到的按钮的格式

\*/

private void setToolButton(Button button) {

button.setOnClickListener(this);

button.setTextColor(Color.WHITE);

}

/\*\*

\* 初始化画画所用的paintView

\*/

private void initPaintView() {

mPaintView = new PaintView(this);

mPaintView.setScaleX(1);

mPaintView.setScaleY(1);

paintViewLayout.addView(mPaintView);

}

/\*\*

\* 初始化所用到的Layout

\*/

private void initLayout() {

colorLayout = (LinearLayout) findViewById(R.id.LinearLayoutColor);

buttonLayout = (LinearLayout) findViewById(R.id.buttonLayout);

colorRadioGroup = (RadioGroup) findViewById(R.id.radioGroupColor);

paintViewLayout = (LinearLayout) findViewById(R.id.paintViewLayout);

eraserSizeLayout = (LinearLayout) findViewById(R.id.sizeSelectLayout);

penSizeLayout = (LinearLayout) findViewById(R.id.sizeSelectLayoutPen);

eraserSizeLayout.setBackgroundDrawable(getResources().getDrawable(

R.drawable.popbackground4));

penSizeLayout.setBackgroundDrawable(getResources().getDrawable(

R.drawable.popbackground4));

shapLayout = (LinearLayout) findViewById(R.id.shapSelectLayout1);

shapLayoutf = (LinearLayout) findViewById(R.id.shapSelectLayout2);

}

/\*\*

\* 初始化颜色选择的RadioGroup

\*/

private void initColorRadioGroup() {

mColorViewList = new ArrayList<ColorView>();

mColorViewList.add(colorView1);

mColorViewList.add(colorView2);

mColorViewList.add(colorView3);

mColorViewList.add(colorView4);

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(

PaintConstants.COLOR\_VIEW\_SIZE, PaintConstants.COLOR\_VIEW\_SIZE);

params.setMargins(10, 5, 10, 5);

for (ColorView colorView : mColorViewList) {

colorRadioGroup.addView(colorView, params);

colorView.setOnCheckedChangeListener(new OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton buttonView,

boolean isChecked) {

for (ColorView colorView : mColorViewList) {

if (buttonView.equals(colorView)

&& buttonView.isChecked()) {

setToLastPenTeype();

mPaintView.setPenColor(colorView.getColor());

}

}

}

});

}

}

/\*\*

\* 如果是Eraser则Set到上一个PenType

\*/

private void setToLastPenTeype() {

if (mPaintView.getCurrentPainter() == PEN\_TYPE.ERASER) {

mPaintView.setCurrentPainterType(mPenType);

}

}

/\*\*

\* 初始化颜色选择的View

\*/

private void initColorViews() {

// 读取preference

SharedPreferences settings = getPreferences(Activity.MODE\_PRIVATE);

// 如果配置文件不存在，则使用默认值

colorView1 = new ColorView(this, settings.getInt("color1",

PaintConstants.COLOR1));

colorView2 = new ColorView(this, settings.getInt("color2",

PaintConstants.COLOR2));

colorView3 = new ColorView(this, settings.getInt("color3",

PaintConstants.COLOR3));

colorView4 = new ColorView(this, settings.getInt("color4",

PaintConstants.COLOR4));

initColorRadioGroup();

}

/\*\*

\* 初始化所有的Button

\*/

private void initButtons() {

findButtonById();

setBackGroundDrawable();

List<ImageButton> list = initButtonList();

for (ImageButton imageButton : list) {

ImageButtonTools.setButtonFocusChanged(imageButton);

imageButton.setOnClickListener(this);

}

}

/\*\*

\* 将需要处理的ImageButton加入到List中

\*/

private List<ImageButton> initButtonList() {

List<ImageButton> list = new ArrayList<ImageButton>();

list.add(loadButton);

list.add(clearButton);

list.add(eraserButton);

list.add(saveButton);

list.add(penSizeButton);

list.add(colorSelectButton);

list.add(undoButton);

list.add(redoButton);

list.add(toButtonLayoutButton);

list.add(toColorLayoutButton);

list.add(toolButton);

return list;

}

/\*\*

\* 找到所有的通过所有的button

\*/

private void findButtonById() {

saveButton = (ImageButton) findViewById(R.id.imageButtonSave);

loadButton = (ImageButton) findViewById(R.id.imageButtonLoadPicture);

clearButton = (ImageButton) findViewById(R.id.imageButtonClear);

eraserButton = (ImageButton) findViewById(R.id.imageButtonEraser);

penSizeButton = (ImageButton) findViewById(R.id.imageButtonPen);

colorSelectButton = (ImageButton) findViewById(R.id.imageButtonColorSelect);

undoButton = (ImageButton) findViewById(R.id.imageButtonUndo);

redoButton = (ImageButton) findViewById(R.id.imageButtonRedo);

toolButton = (ImageButton) findViewById(R.id.imageButtonTools);

toButtonLayoutButton = (ImageButton) findViewById(R.id.imageButtonToButtonLayout);

toColorLayoutButton = (ImageButton) findViewById(R.id.imageButtonToColorLayout);

}

/\*\*

\* 初始化所有Button的Drawable

\*/

@SuppressWarnings("deprecation")

private void setBackGroundDrawable() {

clearButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.newfile));

eraserButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.eraser));

loadButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.loadpicture));

saveButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.save));

penSizeButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.pen\_default));

colorSelectButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.colorselect));

redoButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.cantredo));

undoButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.cantundo));

toButtonLayoutButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.changetobuttonlayout));

toColorLayoutButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.changetocolorlayout));

toolButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.tools));

}

/\*\*

\* onClick函数

\*/

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.imageButtonSave:

onClickButtonSave();

break;

case R.id.imageButtonLoadPicture:

onClickButtonLoad();

break;

case R.id.imageButtonClear:

onClickButtonClear();

break;

case R.id.imageButtonEraser:

onClickButtonEraser();

break;

case R.id.imageButtonPen:

onCLickButtonPen();

break;

case R.id.imageButtonColorSelect:

onClickButtonColorSelect();

break;

case R.id.imageButtonUndo:

onClickButtonUndo();

break;

case R.id.imageButtonRedo:

onClickButtonRedo();

break;

case R.id.imageButtonToColorLayout:

onClickButtonToColorLayout();

break;

case R.id.imageButtonToButtonLayout:

onClickButtonToButtonLayout();

break;

case R.id.buttonSelectBackGroundColor:

onClickButtonBackGround();

break;

case R.id.imageButtonTools:

onClickButtonTools();

break;

case R.id.buttonBlurPen:

onClickButtonBlurPen();

break;

case R.id.buttonPlainPen:

onClickButtonPlainPen();

break;

case R.id.exitButton:

onClickButtonExit();

break;

case R.id.buttonEmboss:

onClickButtonEmboss();

break;

case R.id.aboutButton:

break;

case R.id.helpButton:

break;

default:

break;

}

}

/\*\*

\* 点击浮动才当设置工具类型

\*/

private void setToolTyle(int type) {

mPaintView.setCurrentPainterType(type);

mPenType = type;

toolsPopupWindow.dismiss();

}

/\*\*

\* 透明度

\*/

private void onClickButtonEmboss() {

AlphaSet.recordAlpha(getApplicationContext(), 127);

Toasts.toastShort(getApplicationContext(), "透明度已调整为50%");

}

/\*\*

\* 退出程序

\*/

private void onClickButtonExit() {

this.finish();

}

/\*\*

\* 点击铅笔功能

\*/

private void onClickButtonPlainPen() {

AlphaSet.recordAlpha(getApplicationContext(), 191);

Toasts.toastShort(getApplicationContext(), "透明度已调整为75%");

}

/\*\*

\* blurPen

\*/

private void onClickButtonBlurPen() {

AlphaSet.recordAlpha(getApplicationContext(), 67);

Toasts.toastShort(getApplicationContext(), "透明度已调整为25%");

}

/\*\*

\* 点击工具，弹出工具选项

\*/

private void onClickButtonTools() {

setAllLayoutInvisable();

toolsPopupWindow.showAtLocation(findViewById(R.id.mainLayout),

Gravity.RIGHT | Gravity.BOTTOM, 0, 0);

}

/\*\*

\* 保存ColorViews

\*/

@Override

protected void onPause() {

SharedPreferences currentState = getPreferences(MODE\_PRIVATE);

SharedPreferences.Editor editor = currentState.edit();

editor.putInt("color1", colorView1.getColor());

editor.putInt("color2", colorView2.getColor());

editor.putInt("color3", colorView3.getColor());

editor.putInt("color4", colorView4.getColor());

editor.commit();

super.onPause();

}

/\*\*

\* 当点击menu的时候将popupwindow伪装成menu显示

\*/

@Override

public boolean onKeyUp(int keyCode, KeyEvent event) {

// 点击menu

if (keyCode == KeyEvent.KEYCODE\_MENU) {

setAllLayoutInvisable();

mPopupWindow.showAtLocation(findViewById(R.id.mainLayout),

Gravity.RIGHT | Gravity.BOTTOM, 0, 0);

}

// 点击返回

if (keyCode == KeyEvent.KEYCODE\_BACK) {

OkCancleDialog returnDialog = new OkCancleDialog(this,

new OnClickOkListener() {

@Override

public void onClickOk() {

finish();

}

});

returnDialog.show();

returnDialog.setMessage("确定要退出么？");

return true;

}

return super.onKeyUp(keyCode, event);

}

/\*\*

\* 改变背景颜色

\*/

private void onClickButtonBackGround() {

AlphaSet.recordAlpha(getApplicationContext(), 255);

Toasts.toastShort(getApplicationContext(), "透明度已调整为100%");

}

/\*\*

\* 去ButtonLayout（主界面）

\*/

private void onClickButtonToButtonLayout() {

buttonLayout.setVisibility(View.VISIBLE);

colorLayout.setVisibility(View.GONE);

}

/\*\*

\* 去颜色选择界面

\*/

private void onClickButtonToColorLayout() {

setAllLayoutInvisable();

setToLastPenTeype();

buttonLayout.setVisibility(View.INVISIBLE);

colorLayout.setVisibility(View.VISIBLE);

}

/\*\*

\* redo

\*/

private void onClickButtonRedo() {

setAllLayoutInvisable();

mPaintView.redo();

upDateUndoRedo();

}

/\*\*

\* undo

\*/

private void onClickButtonUndo() {

setAllLayoutInvisable();

mPaintView.undo();

upDateUndoRedo();

}

/\*\*

\* 更新UndoRedo Button

\*/

private void upDateUndoRedo() {

if (mPaintView.canUndo()) {

enableUndoButton();

} else {

disableUndoButton();

}

if (mPaintView.canRedo()) {

enableRedoButton();

} else {

disableRedoButton();

}

}

private void enableRedoButton() {

redoButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.redo));

}

private void disableUndoButton() {

undoButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.cantundo));

}

/\*\*

\* color选择界面 如果是Eraser则改为前一个画笔

\*/

private void onClickButtonColorSelect() {

new ColorPickerDialog(this, new OnColorChangedListener() {

@Override

public void colorChanged(int color) {

mPaintView.setPenColor(color);

for (ColorView colorView : mColorViewList) {

if (colorView.isChecked()) {

setToLastPenTeype();

colorView.setColor(color);

Log.e("aaa", "" + color);

}

}

}

}, mPaintView.getPenColor()).show();

}

/\*\*

\* 改变画笔的大小

\*/

private void onCLickButtonPen() {

mPaintView.setCurrentPainterType(mPenType);

if (penSizeLayout.isShown()) {

setAllLayoutInvisable();

} else {

penSizeLayout.setVisibility(View.VISIBLE);

shapLayout.setVisibility(View.VISIBLE);

shapLayoutf.setVisibility(View.VISIBLE);

eraserSizeLayout.setVisibility(View.INVISIBLE);

}

}

/\*\*

\* 将所有的布局全都隐藏

\*/

private void setAllLayoutInvisable() {

shapLayout.setVisibility(View.INVISIBLE);

shapLayoutf.setVisibility(View.INVISIBLE);

penSizeLayout.setVisibility(View.INVISIBLE);

eraserSizeLayout.setVisibility(View.INVISIBLE);

}

/\*\*

\* 橡皮

\*/

private void onClickButtonEraser() {

if (eraserSizeLayout.isShown()) {

setAllLayoutInvisable();

} else {

eraserSizeLayout.setVisibility(View.VISIBLE);

penSizeLayout.setVisibility(View.INVISIBLE);

shapLayout.setVisibility(View.INVISIBLE);

shapLayoutf.setVisibility(View.INVISIBLE);

}

mPaintView.setCurrentPainterType(PEN\_TYPE.ERASER);

}

/\*\*

\* 清空

\*/

private void onClickButtonClear() {

setAllLayoutInvisable();

clearDialog();

}

/\*\*

\* 清空对话框

\*/

private void clearDialog() {

OkCancleDialog dialog = new OkCancleDialog(this,

new OnClickOkListener() {

@Override

public void onClickOk() {

mPaintView.clearAll();

mPaintView.resetState();

upDateUndoRedo();

upDateColorView();

resetSizeView();

}

});

dialog.show();

dialog.setCanceledOnTouchOutside(true);

dialog.setMessage("确定要清空当前图画么？\n您未保存的修改将丢失");

}

/\*\*

\* 将ColorView的Check清空

\*/

private void upDateColorView() {

colorRadioGroup.clearCheck();

}

/\*\*

\* 重置sizeView

\*/

private void resetSizeView() {

penSizeRadio1.setChecked(true);

eraserSizeRadio01.setChecked(true);

curvRadioButton.setChecked(true);

}

/\*\*

\* 载入图片

\*/

private void onClickButtonLoad() {

setAllLayoutInvisable();

// 点击Load时要对数据库进行更新

sendUpdateBroadCast(file);

startLoadActivity();

}

/\*\*

\* 发送广播，更新sd卡中的数据库

\*/

private void sendUpdateBroadCast(String file) {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.KITKAT) {

MediaScannerConnection.scanFile(

this,

new String[] { Environment

.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_DCIM).getPath()

+ "/" + file }, null, null);

} else {

sendBroadcast(new Intent(Intent.ACTION\_MEDIA\_MOUNTED,

Uri.parse("file://"

+ Environment.getExternalStorageDirectory())));

}

}

/\*\*

\* 保存

\*/

private void onClickButtonSave() {

setAllLayoutInvisable();

boolean sdCardIsMounted = android.os.Environment

.getExternalStorageState().equals(

android.os.Environment.MEDIA\_MOUNTED);

if (!sdCardIsMounted) {

OkDialog okDialog = new OkDialog(this, new OnClickOkListener() {

@Override

public void onClickOk() {

}

});

okDialog.show();

okDialog.setMessage("请插入存储卡");

} else {

SaveDialog dialog = new SaveDialog(this,

new EditTextDialogListener() {

// 当点击确定的时候自动调用 getDialogText接口

@Override

public void getDialogText(String string) {

String sdDir = getDirPath();

file = sdDir + string + ".png";

Bitmap bitmap = mPaintView.getSnapShoot();

BitMapUtils.saveToSdCard(file, bitmap);

sendUpdateBroadCast(file);

}

});

dialog.show();

}

}

/\*\*

\* 开始载入activiy, 发送imageIntent，寻找合适的程序载入

\*/

private void startLoadActivity() {

Intent intent = new Intent(Intent.ACTION\_GET\_CONTENT);// ACTION\_OPEN\_DOCUMENT

intent.addCategory(Intent.CATEGORY\_OPENABLE);

intent.setType("image/jpeg/png");

if (android.os.Build.VERSION.SDK\_INT >= android.os.Build.VERSION\_CODES.KITKAT) {// 4.4系统以上

startActivityForResult(intent, SELECT\_PIC\_KITKAT);

} else {

startActivityForResult(intent, SELECT\_PIC);

}

}

/\*\*

\* 载入之后得到路径

\*/

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

switch (requestCode) {

case SELECT\_PIC\_KITKAT:

if (resultCode != Activity.RESULT\_OK) {

return;

}

if (data != null) {

Uri uri = data.getData();

Bitmap bitmap = comp(GetPath.getPath(getApplicationContext(),

uri));

mPaintView.setForeBitMap(bitmap, DrawActivity.this);

mPaintView.resetState();

upDateUndoRedo();

if (bitmap != null && !bitmap.isRecycled()) {

bitmap.recycle();

bitmap = null;

}

}

break;

case SELECT\_PIC:

if (resultCode != Activity.RESULT\_OK) {

return;

}

if (data != null) {

Uri uri = data.getData();

Bitmap bitmap = comp(GetPath.getPath(getApplicationContext(),

uri));

mPaintView.setForeBitMap(bitmap, DrawActivity.this);

mPaintView.resetState();

upDateUndoRedo();

if (bitmap != null && !bitmap.isRecycled()) {

bitmap.recycle();

bitmap = null;

}

}

break;

default:

break;

}

super.onActivityResult(requestCode, resultCode, data);

}

/\*\*

\* 得到存储路径

\*/

private String getDirPath() {

// File sdcarddir =

// android.os.Environment.getExternalStorageDirectory();

String dirString = SavePathSet.getSavePath(getApplicationContext());

// File filePath = new File(dirString);

// if (!filePath.exists()) {

// // 如果无法创建

// if (!filePath.mkdirs()) {

// OkDialog dialog = new OkDialog(this, new OnClickOkListener() {

// @Override

// public void onClickOk() {

//

// }

// });

// dialog.show();

// dialog.setMessage("无法在sd卡中创建目录/paintPad, \n请检查SDCard");

// }

// }

return dirString;

}

private void enableUndoButton() {

undoButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.undo));

}

private void disableRedoButton() {

redoButton.setBackgroundDrawable(getResources().getDrawable(

R.drawable.cantredo));

}

public static Bitmap comp(String srcPath) {

BitmapFactory.Options newOpts = new BitmapFactory.Options();

/\*\* 开始读入图片，此时把options.inJustDecodeBounds 设回true了 \*\*/

newOpts.inJustDecodeBounds = true;

Bitmap bitmap = BitmapFactory.decodeFile(srcPath, newOpts);

/\*\* 此时返回bm为空 \*\*/

newOpts.inJustDecodeBounds = false;

int w = newOpts.outWidth;

int h = newOpts.outHeight;

/\*\* 现在主流手机比较多是800\*480分辨率，所以高和宽我们设置为 \*\*/

float hh = 800f;

/\*\* 这里设置高度为800f \*\*/

float ww = 400f;

/\*\* 这里设置宽度为480f \*\*/

/\*\* 缩放比。由于是固定比例缩放，只用高或者宽其中一个数据进行计算即可 \*\*/

int be = 1;

/\*\* be=1表示不缩放 \*\*/

if (w > h && w > ww) {

/\*\* 如果宽度大的话根据宽度固定大小缩放 \*\*/

be = (int) (newOpts.outWidth / ww);

} else if (w < h && h > hh) {

/\*\* 如果高度高的话根据宽度固定大小缩放 \*\*/

be = (int) (newOpts.outHeight / hh);

}

if (be <= 0)

be = 1;

newOpts.inSampleSize = be;

/\*\* 设置缩放比例 \*\*/

/\*\* 重新读入图片，注意此时已经把options.inJustDecodeBounds 设回false了 \*\*/

bitmap = BitmapFactory.decodeFile(srcPath, newOpts);

return compressImage(bitmap);

/\*\* 压缩好比例大小后再进行质量压缩 \*\*/

}

/\*\*

\* @方法说明:质量压缩方法

\* @方法名称:compressImage

\* @param image

\* @return

\* @返回值:Bitmap

\*/

public static Bitmap compressImage(Bitmap image) {

ByteArrayOutputStream baos = new ByteArrayOutputStream();

image.compress(Bitmap.CompressFormat.JPEG, 100, baos);// 质量压缩方法，这里100表示不压缩，把压缩后的数据存放到baos中

int options = 100;

while (baos.toByteArray().length / 1024 > 100) { // 循环判断如果压缩后图片是否大于100kb,大于继续压缩

baos.reset();// 重置baos即清空baos

image.compress(Bitmap.CompressFormat.JPEG, options, baos);// 这里压缩options%，把压缩后的数据存放到baos中

options -= 10;// 每次都减少10

}

ByteArrayInputStream isBm = new ByteArrayInputStream(baos.toByteArray());// 把压缩后的数据baos存放到ByteArrayInputStream中

Bitmap bitmap = BitmapFactory.decodeStream(isBm, null, null);// 把ByteArrayInputStream数据生成图片

return bitmap;

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

// TODO Auto-generated method stub

super.onConfigurationChanged(newConfig);

if (newConfig.orientation == this.getResources().getConfiguration().ORIENTATION\_LANDSCAPE) {

mPaintView.redo();

}

}

}

/\*\*

\* 主界面

\*/

public class MainActivity extends FragmentActivity implements OnClickListener {

private RadioButton home;// 主页

private RadioButton user;// 用户

private RadioButton discover;// 话题

private Fragment homeFragment;

private Fragment userFragment;

private Fragment discoverFragment;

@Override

protected void onCreate(Bundle savedInstanceState) {

ActivityManager.getInstance().addActivity(this);

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

initView();

setSelect(0);

// 初始化 Bmob SDK

Bmob.initialize(getApplicationContext(),

"0985f7f9dd3a8641c766ffbc56b7c632");

UmengUpdateAgent.update(this);

}

private void initView() {

home = (RadioButton) findViewById(R.id.rb\_home);

home.setOnClickListener(this);

user = (RadioButton) findViewById(R.id.rb\_user);

user.setOnClickListener(this);

discover = (RadioButton) findViewById(R.id.rb\_discover);

discover.setOnClickListener(this);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.rb\_home:

setSelect(0);

break;

case R.id.rb\_discover:

if (BmobUser.getCurrentUser(getApplicationContext()) == null)

startActivity(new Intent(MainActivity.this, LoginActivity.class));

else

setSelect(1);

break;

case R.id.rb\_user:

if (BmobUser.getCurrentUser(getApplicationContext()) == null)

startActivity(new Intent(MainActivity.this, LoginActivity.class));

else

setSelect(2);

break;

default:

break;

}

}

/\*\*

\* 显示选中的内容

\*

\* @param i

\*/

private void setSelect(int i) {

FragmentManager fm = getSupportFragmentManager();

FragmentTransaction transaction = fm.beginTransaction();

hideFragment(transaction);

switch (i) {

case 0:

if (homeFragment == null) {

homeFragment = new HomeFragment();

transaction.add(R.id.fl\_content, homeFragment);

} else {

transaction.show(homeFragment);

}

break;

case 1:

if (discoverFragment == null) {

discoverFragment = new TopicFragment();

transaction.add(R.id.fl\_content, discoverFragment);

} else {

transaction.show(discoverFragment);

}

break;

case 2:

if (userFragment == null) {

userFragment = new UserFragment();

transaction.add(R.id.fl\_content, userFragment);

} else {

transaction.show(userFragment);

}

break;

default:

break;

}

transaction.commit();

}

/\*\*

\* 隐藏内容

\*

\* @param transaction

\*/

private void hideFragment(FragmentTransaction transaction) {

if (homeFragment != null) {

transaction.hide(homeFragment);

}

if (discoverFragment != null) {

transaction.hide(discoverFragment);

}

if (userFragment != null) {

transaction.hide(userFragment);

}

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_BACK && event.getRepeatCount() == 0) {

exitDialog(1);

return true;

} else

return super.onKeyDown(keyCode, event);

}

protected void exitDialog(int id) {

if (id == 1) {

final OkCancleDialog dialog = new OkCancleDialog(this,

new OnClickOkListener() {

@Override

public void onClickOk() {

ActivityManager.getInstance().exit();

finish();

}

});

dialog.show();

dialog.setCanceledOnTouchOutside(true);

dialog.setMessage("确定退出程序吗");

}

}

@Override

protected void onDestroy() {

super.onDestroy();

ActivityManager.getInstance().remove(this);

}

}

package com.example.activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.EditText;

import android.widget.TextView;

import cn.bmob.v3.BmobUser;

import cn.bmob.v3.listener.UpdateListener;

import com.example.entity.User;

import com.example.quick.R;

import com.example.utils.Toasts;

/\*\*

\* 修改用户名

\*/

public class ModifyNameActivity extends BaseActivity implements OnClickListener {

private TextView cancel;

private TextView finish;

private EditText name;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_modify\_name);

initView();

}

private void initView() {

cancel = (TextView) findViewById(R.id.tv\_cancel);

cancel.setOnClickListener(this);

finish = (TextView) findViewById(R.id.tv\_finish);

finish.setOnClickListener(this);

name = (EditText) findViewById(R.id.et\_name);

name.setText(BmobUser.getCurrentUser(getApplicationContext())

.getUsername());

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.tv\_cancel:

finish();

break;

case R.id.tv\_finish:

updateName();

break;

default:

break;

}

}

/\*\*

\* 更新姓名操作

\*/

private void updateName() {

if (name.getText().toString().equals(""))

Toasts.toastShort(getApplicationContext(), "请填写您的帐户名");

else {

User newUser = new User();

newUser.setUsername(name.getText().toString());

BmobUser user = BmobUser.getCurrentUser(this);

newUser.update(this, user.getObjectId(), new UpdateListener() {

@Override

public void onSuccess() {

Toasts.toastShort(getApplicationContext(), "更新成功");

Intent i = new Intent();

i.putExtra("name", name.getText().toString());

setResult(RESULT\_OK, i);

finish();

}

@Override

public void onFailure(int arg0, String arg1) {

Toasts.toastShort(getApplicationContext(), "error");

}

});

}

}

}

package com.example.activity;

import java.util.List;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.TextView;

import cn.bmob.v3.BmobQuery;

import cn.bmob.v3.BmobQuery.CachePolicy;

import cn.bmob.v3.listener.FindListener;

import com.example.adapter.PostAdapter;

import com.example.entity.Post;

import com.example.interfaces.OnClickOkListener;

import com.example.quick.R;

import com.example.utils.ActivityManager;

import com.example.utils.RecordTopic;

import com.example.utils.Toasts;

import com.example.view.LoadingDialog;

import com.example.view.ReFlashListView;

import com.example.view.ReFlashListView.IReflashListener;

/\*\*

\* 说说界面

\*/

public class PostActivity extends BaseActivity implements OnItemClickListener,

IReflashListener, OnClickListener {

private TextView back;

private TextView addPost;

private ReFlashListView lv;

private PostAdapter adapter;

private List<Post> data;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_post);

initView();

getPost();

}

/\*\*

\* 获取说说

\*/

private void getPost() {

BmobQuery<Post> query = new BmobQuery<Post>();

query.order("-createdAt");

query.addWhereEqualTo("topic",

RecordTopic.getTopic(getApplicationContext()));

query.findObjects(getApplicationContext(), new FindListener<Post>() {

@Override

public void onSuccess(List<Post> arg0) {

if (arg0.size() == 0)

lv.setAdapter(null);

else {

data = arg0;

adapter = new PostAdapter(getApplicationContext(), data,

R.layout.item\_post);

lv.setAdapter(adapter);

}

}

@Override

public void onError(int arg0, String arg1) {

}

});

}

private void initView() {

back = (TextView) findViewById(R.id.tv\_back);

back.setOnClickListener(this);

addPost = (TextView) findViewById(R.id.tv\_add\_post);

addPost.setOnClickListener(this);

lv = (ReFlashListView) findViewById(R.id.lv\_post);

lv.setInterface(this);

lv.setOnItemClickListener(this);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.tv\_back:

startActivity(new Intent(PostActivity.this,MainActivity.class));

break;

case R.id.tv\_add\_post:

Intent i = new Intent(PostActivity.this, AddPostActivity.class);

startActivityForResult(i, 1);

break;

default:

break;

}

}

/\*\*

\* 刷新数据

\*/

@Override

public void onReflash() {

Handler handler = new Handler();

handler.postDelayed(new Runnable() {

@Override

public void run() {

// 获取最新数据

getPost();

// 通知listview 刷新数据完毕；

lv.reflashComplete();

}

}, 2000);

}

/\*\*

\* 列表单机事件

\*/

@Override

public void onItemClick(AdapterView<?> parent, View view, int position,

long id) {

Intent i = new Intent(PostActivity.this, PostDetailActivity.class);

i.putExtra("id", data.get(position - 1).getObjectId());

i.putExtra("number", "2");

startActivity(i);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if(requestCode == 1&&resultCode == RESULT\_OK){

Toasts.toastShort(getApplicationContext(), "收到");

}

}

}

package com.example.activity;

import java.io.File;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.os.Environment;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.RelativeLayout;

import android.widget.TextView;

import cn.bmob.v3.BmobQuery;

import cn.bmob.v3.BmobUser;

import cn.bmob.v3.datatype.BmobFile;

import cn.bmob.v3.listener.GetListener;

import cn.bmob.v3.listener.UpdateListener;

import cn.bmob.v3.listener.UploadFileListener;

import com.bmob.BmobProFile;

import com.example.entity.User;

import com.example.fragment.UserFragment;

import com.example.quick.R;

import com.example.utils.GetPath;

import com.example.utils.ImageUtils;

import com.example.utils.PushState;

import com.example.utils.SavePathSet;

import com.example.utils.Toasts;

import com.squareup.picasso.Picasso;

import com.umeng.message.PushAgent;

/\*\*

\* 设置

\* @author shi

\*/

public class SettingsActivity extends BaseActivity implements OnClickListener {

private TextView back;

private TextView name;

private TextView dir;

private TextView state;

private TextView modify\_state;

private RelativeLayout rl\_name;

private RelativeLayout rl\_dir;

private RelativeLayout rl\_icon;

private RelativeLayout rl\_state;

private Button logout;

private PushAgent mPushAgent;

private ImageView img\_userIcon;

private Uri uri;

private static final int REQUEST\_CODE = 1; // 请求码

public static final String EXTRA\_FILE\_CHOOSER = "file\_chooser";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_settings);

initView();

}

private void initView() {

mPushAgent = PushAgent.getInstance(getApplicationContext());

state = (TextView) findViewById(R.id.tv\_state);

modify\_state = (TextView) findViewById(R.id.tv\_modify\_state);

back = (TextView) findViewById(R.id.tv\_back);

back.setOnClickListener(this);

name = (TextView) findViewById(R.id.tv\_name);

dir = (TextView) findViewById(R.id.tv\_dir);

dir.setText(SavePathSet.getSavePath(getApplicationContext()));

rl\_name = (RelativeLayout) findViewById(R.id.rl\_name);

rl\_name.setOnClickListener(this);

rl\_dir = (RelativeLayout) findViewById(R.id.rl\_dir);

rl\_dir.setOnClickListener(this);

rl\_state = (RelativeLayout) findViewById(R.id.rl\_state);

rl\_state.setOnClickListener(this);

rl\_icon = (RelativeLayout) findViewById(R.id.rl\_icon);

rl\_icon.setOnClickListener(this);

logout = (Button) findViewById(R.id.btn\_logout);

logout.setOnClickListener(this);

img\_userIcon = (ImageView) findViewById(R.id.img\_usericon);

name.setText(BmobUser.getCurrentUser(getApplicationContext())

.getUsername());

if (PushState.getPush(getApplicationContext())) {

mPushAgent.enable();

modify\_state.setText("点击关闭");

state.setText("开");

} else {

mPushAgent.disable();

modify\_state.setText("点击打开");

state.setText("关");

}

loadImage();

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.tv\_back:

finish();

break;

case R.id.rl\_dir:

chooseSavePath();

break;

case R.id.rl\_state:

changeState();

break;

case R.id.rl\_icon:

ImageUtils.showImagePickDialog(SettingsActivity.this);

break;

case R.id.rl\_name:

Intent i = new Intent(SettingsActivity.this,

ModifyNameActivity.class);

startActivityForResult(i, 2);

break;

case R.id.btn\_logout:

BmobUser.logOut(getApplicationContext());

startActivity(new Intent(SettingsActivity.this, LoginActivity.class));

break;

default:

break;

}

}

/\*\*

\* 加载用户头像

\*/

private void loadImage() {

BmobQuery<User> query = new BmobQuery<User>();

query.getObject(getApplicationContext(),

BmobUser.getCurrentUser(getApplicationContext()).getObjectId(),

new GetListener<User>() {

@Override

public void onSuccess(User arg0) {

if (arg0.getPic() == null) {

img\_userIcon

.setBackgroundResource(R.drawable.user\_icon\_default);

} else {

BmobFile icon = arg0.getPic();

String fileUrl = icon

.getFileUrl(getApplicationContext());

String fileName = icon.getFilename();

String URL = BmobProFile.getInstance(

getApplicationContext()).signURL(fileName,

fileUrl,

"54ff111e20b202ccad6262e519c25cc3", 100,

"123456");

Picasso.with(getApplicationContext()).load(URL)

.into(img\_userIcon);

}

}

@Override

public void onFailure(int arg0, String arg1) {

}

});

}

/\*\*

\* 改变消息接收状态

\*/

private void changeState() {

if (PushState.getPush(this)) {

modify\_state.setText("点击打开");

state.setText("关");

mPushAgent.disable();

PushState.recordPush(getApplicationContext(), false);

Toasts.toastShort(getApplicationContext(), "推送已关闭");

} else {

modify\_state.setText("点击关闭");

state.setText("开");

mPushAgent.enable();

PushState.recordPush(getApplicationContext(), true);

Toasts.toastShort(getApplicationContext(), "推送已打开");

}

}

/\*\*

\* 选择文件保存路径

\*/

private void chooseSavePath() {

Intent i = new Intent(this, FileChooserActivity.class);

if (Environment.getExternalStorageState().equals(

Environment.MEDIA\_MOUNTED))

startActivityForResult(i, REQUEST\_CODE);

else

Toasts.toastShort(getApplicationContext(), "检查存储卡是否存在");

}

public void onActivityResult(int requestCode, int resultCode, Intent data) {

// 文件选择失败回调

if (resultCode == RESULT\_CANCELED) {

Toasts.toastShort(getApplicationContext(), "没有选择文件夹");

return;

}

// 文件选择成功回调

if (resultCode == RESULT\_OK && requestCode == REQUEST\_CODE) {

// 获取路径名

// String pptPath = data.getStringExtra(EXTRA\_FILE\_CHOOSER);

String pptPath = data.getExtras().getString(EXTRA\_FILE\_CHOOSER);

if (pptPath != null) {

dir.setText(pptPath);

} else

Toasts.toastShort(getApplicationContext(), "打开文件失败");

}

// 修改名字回调

if (resultCode == RESULT\_OK && requestCode == 2) {

name.setText(data.getExtras().getString("name"));

}

// 相册回调

if (requestCode == ImageUtils.REQUEST\_CODE\_FROM\_ALBUM) {

if (resultCode == RESULT\_CANCELED) {

return;

} else {

uri = data.getData();

img\_userIcon.setImageURI(uri);

uploadUserIcon();

}

}

// 相机回调

if (requestCode == ImageUtils.REQUEST\_CODE\_FROM\_CAMERA) {

if (resultCode == RESULT\_CANCELED) {

return;

} else {

uri = ImageUtils.imageUriFromCamera;

img\_userIcon.setImageURI(uri);

uploadUserIcon();

}

}

}

/\*\*

\* 上传用户头像

\*/

private void uploadUserIcon() {

String path = GetPath.getPath(getApplicationContext(), uri);

final BmobFile file = new BmobFile(new File(path));

file.upload(getApplicationContext(), new UploadFileListener() {

@Override

public void onSuccess() {

User newUser = new User();

newUser.setPic(file);

BmobUser user = BmobUser.getCurrentUser(getApplicationContext());

newUser.update(getApplicationContext(), user.getObjectId(), new UpdateListener() {

@Override

public void onSuccess() {

Toasts.toastShort(getApplicationContext(), "更新成功");

Intent i = new Intent(SettingsActivity.this,UserFragment.class);

setResult(RESULT\_OK, i);

}

@Override

public void onFailure(int arg0, String arg1) {

Toasts.toastShort(getApplicationContext(), "未知错误");

}

});

}

@Override

public void onFailure(int arg0, String arg1) {

Toasts.toastShort(getApplicationContext(), "error");

}

});

}

}

package com.example.view;

import java.util.ArrayList;

import android.app.Activity;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.Paint.Style;

import android.util.AttributeSet;

import android.util.DisplayMetrics;

import android.view.MotionEvent;

import android.view.View;

import android.view.ViewTreeObserver.OnGlobalLayoutListener;

import com.example.interfaces.PaintViewCallBack;

import com.example.interfaces.Shapable;

import com.example.interfaces.ShapesInterface;

import com.example.interfaces.ToolInterface;

import com.example.interfaces.UndoCommand;

import com.example.painttools.BlurPen;

import com.example.painttools.EmbossPen;

import com.example.painttools.Eraser;

import com.example.painttools.PlainPen;

import com.example.shapes.Circle;

import com.example.shapes.Curv;

import com.example.shapes.DottedLine;

import com.example.shapes.Line;

import com.example.shapes.Oval;

import com.example.shapes.Rectangle;

import com.example.shapes.Square;

import com.example.utils.AlphaSet;

import com.example.utils.BitMapUtils;

import com.example.utils.MyApplication;

import com.example.utils.PaintConstants;

import com.example.utils.PaintConstants.DEFAULT;

import com.example.utils.PaintConstants.ERASER\_SIZE;

import com.example.utils.PaintConstants.PEN\_SIZE;

import com.example.utils.PaintConstants.PEN\_TYPE;

import com.example.utils.PaintConstants.SHAP;

import com.example.utils.PenSizeSet;

public class PaintView extends View implements UndoCommand,OnGlobalLayoutListener{

boolean canvasIsCreated = false;

private Canvas mCanvas = null;

private ToolInterface mCurrentPainter = null;

private float mInitScale;

private float mMidScale;

private float mMaxScale;

/\* Bitmap 的配置 \*/

private Bitmap mBitmap = null;

private Bitmap mOrgBitMap = null;

private int mBitmapWidth = 0;

private int mBitmapHeight = 0;

private int mBackGroundColor = DEFAULT.BACKGROUND\_COLOR;

/\* paint 的配置 \*/

private Paint mBitmapPaint = null;

private paintPadUndoStack mUndoStack = null;

private int mPenColor = DEFAULT.PEN\_COLOR;;

private int mPenSize = PEN\_SIZE.SIZE\_1;

private int mEraserSize = ERASER\_SIZE.SIZE\_1;

int mPaintType = PEN\_TYPE.PLAIN\_PEN;

private PaintViewCallBack mCallBack = null;

private int mCurrentShapeType = 0;

private ShapesInterface mCurrentShape = null;

private Paint.Style mStyle = Paint.Style.STROKE;

/\* 其他 的配置 \*/

private boolean isTouchUp = false;

private int mStackedSize = PaintConstants.UNDO\_STACK\_SIZE;

public PaintView(Context context) {

this(context, null);

}

public PaintView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PaintView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

/\*\*

\* 初始化

\*/

private void init() {

mCanvas = new Canvas();

mBitmapPaint = new Paint(Paint.DITHER\_FLAG);

mUndoStack = new paintPadUndoStack(this, mStackedSize);

mPaintType = PEN\_TYPE.PLAIN\_PEN;

mCurrentShapeType = SHAP.CURV;

creatNewPen();

}

/\*\*

\* 回调主函数的onHasDraw函数

\*/

public void setCallBack(PaintViewCallBack callBack) {

mCallBack = callBack;

}

@Override

public boolean onTouchEvent(MotionEvent event) {

float x = event.getX();

float y = event.getY();

isTouchUp = false;

switch (event.getAction()) {

case MotionEvent.ACTION\_DOWN:

mCanvas.setBitmap(mBitmap);

creatNewPen();

mCurrentPainter.touchDown(x, y);

mUndoStack.clearRedo();

mCallBack.onTouchDown();

invalidate();

break;

case MotionEvent.ACTION\_MOVE:

mCurrentPainter.touchMove(x, y);

if (mPaintType == PEN\_TYPE.ERASER) {

mCurrentPainter.draw(mCanvas);

}

invalidate();

break;

case MotionEvent.ACTION\_UP:

if (mCurrentPainter.hasDraw()) {

mUndoStack.push(mCurrentPainter);

if (mCallBack != null) {

// 控制undo\redo的现实

mCallBack.onHasDraw();

}

}

mCurrentPainter.touchUp(x, y);

// 只有在up的时候才在bitmap上画图，最终显示在view上

mCurrentPainter.draw(mCanvas);

invalidate();

isTouchUp = true;

break;

}

return true;

}

/\*\*

\* 设置具体形状，需要注意的是构造函数中的Painter必须是新鲜出炉的

\*/

private void setShape() {

if (mCurrentPainter instanceof Shapable) {

switch (mCurrentShapeType) {

case SHAP.CURV:

mCurrentShape = new Curv((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()));

break;

case SHAP.LINE:

mCurrentShape = new Line((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()));

break;

case SHAP.SQUARE:

mCurrentShape = new Square((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()));

break;

case SHAP.RECT:

mCurrentShape = new Rectangle((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()));

break;

case SHAP.CIRCLE:

mCurrentShape = new Circle((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()));

break;

case SHAP.OVAL:

mCurrentShape = new Oval((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()),

PenSizeSet.getSize(MyApplication.getContextObject()));

break;

case SHAP.DottedLine:

mCurrentShape = new DottedLine((Shapable) mCurrentPainter,

AlphaSet.getAlpha(MyApplication.getContextObject()));

default:

break;

}

((Shapable) mCurrentPainter).setShap(mCurrentShape);

}

}

@Override

public void onDraw(Canvas cv) {

cv.drawColor(mBackGroundColor);

// 在外部绘制的方法只有一种，就是先在bitmap上绘制，然后加载到cv

cv.drawBitmap(mBitmap, 0, 0, mBitmapPaint);

// TouchUp使用BitMap的canvas进行绘制，也就不用再View上绘制了

if (!isTouchUp) {

// 平时都只在view的cv上临时绘制

// earaser不能再cv上绘制，需要直接绘制在bitmap上

if (mPaintType != PEN\_TYPE.ERASER) {

mCurrentPainter.draw(cv);

}

}

}

/\*\*

\* 创建一个新的画笔

\*/

void creatNewPen() {

ToolInterface tool = null;

switch (mPaintType) {

case PEN\_TYPE.PLAIN\_PEN:

tool = new PlainPen(mPenSize, mPenColor, mStyle);

break;

case PEN\_TYPE.ERASER:

tool = new Eraser(mEraserSize);

break;

case PEN\_TYPE.BLUR:

tool = new BlurPen(mPenSize, mPenColor, mStyle);

break;

case PEN\_TYPE.EMBOSS:

tool = new EmbossPen(mPenSize, mPenColor, mStyle);

break;

default:

break;

}

mCurrentPainter = tool;

setShape();

}

/\*\*

\* 当此事件发生时，创建Bitmap并setCanvas

\*/

@Override

protected void onSizeChanged(int w, int h, int oldw, int oldh) {

super.onSizeChanged(w, h, oldw, oldh);

if (!canvasIsCreated) {

mBitmapWidth = w;

mBitmapHeight = h;

creatCanvasBitmap(w, h);

canvasIsCreated = true;

}

}

/\*\*

\* 创建画布上的背景图片，同时留有备份

\*/

public void setForeBitMap(Bitmap bitmap, Activity activity) {

if (bitmap != null) {

recycleMBitmap();

recycleOrgBitmap();

}

DisplayMetrics dm = new DisplayMetrics();

// 获取屏幕信息

((Activity) activity).getWindowManager().getDefaultDisplay()

.getMetrics(dm);

int screenWidth = dm.widthPixels;

int screenHeigh = dm.heightPixels;

mBitmap = BitMapUtils.duplicateBitmap(bitmap, screenWidth, screenHeigh);

mOrgBitMap = BitMapUtils.duplicateBitmap(mBitmap);

if (bitmap != null && !bitmap.isRecycled()) {

bitmap.recycle();

bitmap = null;

}

invalidate();

}

/\*\*

\* 回收原始图片

\*/

private void recycleOrgBitmap() {

if (mOrgBitMap != null && !mOrgBitMap.isRecycled()) {

mOrgBitMap.recycle();

mOrgBitMap = null;

}

}

/\*\*

\* 回收图片

\*/

private void recycleMBitmap() {

if (mBitmap != null && !mBitmap.isRecycled()) {

mBitmap.recycle();

mBitmap = null;

}

}

/\*\*

\* 得到当前view的截图

\*/

public Bitmap getSnapShoot() {

// 获得当前的view的图片

setDrawingCacheEnabled(true);

buildDrawingCache(true);

Bitmap bitmap = getDrawingCache(true);

Bitmap bmp = BitMapUtils.duplicateBitmap(bitmap);

if (bitmap != null && !bitmap.isRecycled()) {

bitmap.recycle();

bitmap = null;

}

// 将缓存清理掉

setDrawingCacheEnabled(false);

return bmp;

}

/\*\*

\* 清空屏幕

\*/

public void clearAll() {

recycleMBitmap();

recycleOrgBitmap();

mUndoStack.clearAll();

creatCanvasBitmap(mBitmapWidth, mBitmapHeight);

invalidate();

}

/\*\*

\* 创建bitMap同时获得其canvas

\*/

private void creatCanvasBitmap(int w, int h) {

mBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB\_8888);

mCanvas.setBitmap(mBitmap);

}

/\*\*

\* 改变当前画笔的类型

\*/

public void setCurrentPainterType(int type) {

switch (type) {

case PEN\_TYPE.BLUR:

case PEN\_TYPE.PLAIN\_PEN:

case PEN\_TYPE.EMBOSS:

case PEN\_TYPE.ERASER:

mPaintType = type;

break;

default:

mPaintType = PEN\_TYPE.PLAIN\_PEN;

break;

}

}

/\*\*

\* 改变当前的Shap

\*/

public void setCurrentShapType(int type) {

switch (type) {

case SHAP.CURV:

case SHAP.LINE:

case SHAP.RECT:

case SHAP.CIRCLE:

case SHAP.OVAL:

case SHAP.SQUARE:

mCurrentShapeType = type;

break;

default:

mCurrentShapeType = SHAP.CURV;

break;

}

}

/\*\*

\* 得到当前画笔的类型

\*/

public int getCurrentPainter() {

return mPaintType;

}

/\*\*

\* 改变当前画笔的大小

\*/

public void setPenSize(int size) {

mPenSize = size;

}

/\*\*

\* 改变当前Eraser的大小

\*/

public void setEraserSize(int size) {

mEraserSize = size;

}

/\*\*

\* 得到当前画笔的大小

\*/

public int getPenSize() {

return mPenSize;

}

/\*\*

\* 重置状态

\*/

public void resetState() {

setCurrentPainterType(PEN\_TYPE.PLAIN\_PEN);

setPenColor(DEFAULT.PEN\_COLOR);

setBackGroundColor(DEFAULT.BACKGROUND\_COLOR);

mUndoStack.clearAll();

}

/\*\*

\* 更改背景颜色

\*/

public void setBackGroundColor(int color) {

mBackGroundColor = color;

invalidate();

}

/\*\*

\* 得到背景颜色

\*/

public int getBackGroundColor() {

return mBackGroundColor;

}

/\*\*

\* 改变画笔的颜色，在创建新笔的时候就能使用了

\*/

public void setPenColor(int color) {

mPenColor = color;

}

/\*\*

\* 得到penColor

\*/

public int getPenColor() {

return mPenColor;

}

/\*\*

\* 创建临时背景，没有备份

\*/

protected void setTempForeBitmap(Bitmap tempForeBitmap) {

if (null != tempForeBitmap) {

recycleMBitmap();

mBitmap = BitMapUtils.duplicateBitmap(tempForeBitmap);

if (null != mBitmap && null != mCanvas) {

mCanvas.setBitmap(mBitmap);

invalidate();

}

}

}

public void setPenStyle(Style style) {

mStyle = style;

}

public byte[] getBitmapArry() {

return BitMapUtils.bitampToByteArray(mBitmap);

}

@Override

public void undo() {

if (null != mUndoStack) {

mUndoStack.undo();

}

}

@Override

public void redo() {

if (null != mUndoStack) {

mUndoStack.redo();

} else {

}

}

@Override

public boolean canUndo() {

return mUndoStack.canUndo();

}

@Override

public boolean canRedo() {

return mUndoStack.canRedo();

}

@Override

public String toString() {

return "mPaint" + mCurrentPainter + mUndoStack;

}

/\*

\* ===================================内部类开始=================================

\* 内部类，负责undo、redo

\*/

public class paintPadUndoStack {

private int m\_stackSize = 0;

private PaintView mPaintView = null;

private ArrayList<ToolInterface> mUndoStack = new ArrayList<ToolInterface>();

private ArrayList<ToolInterface> mRedoStack = new ArrayList<ToolInterface>();

private ArrayList<ToolInterface> mOldActionStack = new ArrayList<ToolInterface>();

public paintPadUndoStack(PaintView paintView, int stackSize) {

mPaintView = paintView;

m\_stackSize = stackSize;

}

/\*\*

\* 将painter存入栈中

\*/

public void push(ToolInterface penTool) {

if (null != penTool) {

// 如果undo已经存满

if (mUndoStack.size() == m\_stackSize && m\_stackSize > 0) {

// 得到最远的画笔

ToolInterface removedTool = mUndoStack.get(0);

// 所有的笔迹增加

mOldActionStack.add(removedTool);

mUndoStack.remove(0);

}

mUndoStack.add(penTool);

}

}

/\*\*

\* 清空所有

\*/

public void clearAll() {

mRedoStack.clear();

mUndoStack.clear();

mOldActionStack.clear();

}

/\*\*

\* undo

\*/

public void undo() {

if (canUndo() && null != mPaintView) {

ToolInterface removedTool = mUndoStack

.get(mUndoStack.size() - 1);

mRedoStack.add(removedTool);

mUndoStack.remove(mUndoStack.size() - 1);

if (null != mOrgBitMap) {

// Set the temporary fore bitmap to canvas.

// 当载入文件时保存了一份,现在要重新绘制出来

mPaintView.setTempForeBitmap(mPaintView.mOrgBitMap);

} else {

// 如果背景不存在，则重新创建一份背景

mPaintView.creatCanvasBitmap(mPaintView.mBitmapWidth,

mPaintView.mBitmapHeight);

}

Canvas canvas = mPaintView.mCanvas;

// First draw the removed tools from undo stack.

for (ToolInterface paintTool : mOldActionStack) {

paintTool.draw(canvas);

}

for (ToolInterface paintTool : mUndoStack) {

paintTool.draw(canvas);

}

mPaintView.invalidate();

}

}

/\*\*

\* redo

\*/

public void redo() {

if (canRedo() && null != mPaintView) {

ToolInterface removedTool = mRedoStack

.get(mRedoStack.size() - 1);

mUndoStack.add(removedTool);

mRedoStack.remove(mRedoStack.size() - 1);

if (null != mOrgBitMap) {

// Set the temporary fore bitmap to canvas.

mPaintView.setTempForeBitmap(mPaintView.mOrgBitMap);

} else {

// Create a new bitmap and set to canvas.

mPaintView.creatCanvasBitmap(mPaintView.mBitmapWidth,

mPaintView.mBitmapHeight);

}

Canvas canvas = mPaintView.mCanvas;

// 所有以前的笔迹都存放在removedStack中

// First draw the removed tools from undo stack.

for (ToolInterface sketchPadTool : mOldActionStack) {

sketchPadTool.draw(canvas);

}

// 不管怎样都是从撤销里面绘制，重做只是暂时的存储

for (ToolInterface sketchPadTool : mUndoStack) {

sketchPadTool.draw(canvas);

}

mPaintView.invalidate();

}

}

public boolean canUndo() {

return (mUndoStack.size() > 0);

}

public boolean canRedo() {

return (mRedoStack.size() > 0);

}

public void clearRedo() {

mRedoStack.clear();

}

@Override

public String toString() {

return "canUndo" + canUndo();

}

}

/\* ==================================内部类结束 ================================= \*/

@Override

protected void onAttachedToWindow() {

// TODO Auto-generated method stub

super.onAttachedToWindow();

getViewTreeObserver().addOnGlobalLayoutListener(this);

}

@Override

protected void onDetachedFromWindow() {

// TODO Auto-generated method stub

super.onDetachedFromWindow();

getViewTreeObserver().removeOnGlobalLayoutListener(this);

}

@Override

public void onGlobalLayout() {

}

}

package com.example.view;

import java.text.SimpleDateFormat;

import java.util.Date;

import android.content.Context;

import android.util.AttributeSet;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.MotionEvent;

import android.view.View;

import android.view.ViewGroup;

import android.view.animation.RotateAnimation;

import android.widget.AbsListView;

import android.widget.AbsListView.OnScrollListener;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.ProgressBar;

import android.widget.TextView;

import com.example.quick.R;

/\*\*

\* 自定义ListView

\*

\*/

public class ReFlashListView extends ListView implements OnScrollListener {

View header;// 顶部布局文件；

int headerHeight;// 顶部布局文件的高度；

int firstVisibleItem;// 当前第一个可见的item的位置；

int scrollState;// listview 当前滚动状态；

boolean isRemark;// 标记，当前是在listview最顶端摁下的；

int startY;// 摁下时的Y值；

int state;// 当前的状态；

final int NONE = 0;// 正常状态；

final int PULL = 1;// 提示下拉状态；

final int RELESE = 2;// 提示释放状态；

final int REFLASHING = 3;// 刷新状态；

IReflashListener iReflashListener;//刷新数据的接口

public ReFlashListView(Context context) {

super(context);

// TODO Auto-generated constructor stub

initView(context);

}

public ReFlashListView(Context context, AttributeSet attrs) {

super(context, attrs);

// TODO Auto-generated constructor stub

initView(context);

}

public ReFlashListView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

// TODO Auto-generated constructor stub

initView(context);

}

/\*\*

\* 初始化界面，添加顶部布局文件到 listview

\*

\* @param context

\*/

private void initView(Context context) {

LayoutInflater inflater = LayoutInflater.from(context);

header = inflater.inflate(R.layout.header\_layout, null);

measureView(header);

headerHeight = header.getMeasuredHeight();

Log.i("tag", "headerHeight = " + headerHeight);

topPadding(-headerHeight);

this.addHeaderView(header);

this.setOnScrollListener(this);

}

/\*\*

\* 通知父布局，占用的宽，高；

\*

\* @param view

\*/

private void measureView(View view) {

ViewGroup.LayoutParams p = view.getLayoutParams();

if (p == null) {

p = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT);

}

int width = ViewGroup.getChildMeasureSpec(0, 0, p.width);

int height;

int tempHeight = p.height;

if (tempHeight > 0) {

height = MeasureSpec.makeMeasureSpec(tempHeight,

MeasureSpec.EXACTLY);

} else {

height = MeasureSpec.makeMeasureSpec(0, MeasureSpec.UNSPECIFIED);

}

view.measure(width, height);

}

/\*\*

\* 设置header 布局 上边距；

\*

\* @param topPadding

\*/

private void topPadding(int topPadding) {

header.setPadding(header.getPaddingLeft(), topPadding,

header.getPaddingRight(), header.getPaddingBottom());

header.invalidate();

}

@Override

public void onScroll(AbsListView view, int firstVisibleItem,

int visibleItemCount, int totalItemCount) {

// TODO Auto-generated method stub

this.firstVisibleItem = firstVisibleItem;

}

@Override

public void onScrollStateChanged(AbsListView view, int scrollState) {

// TODO Auto-generated method stub

this.scrollState = scrollState;

}

@Override

public boolean onTouchEvent(MotionEvent ev) {

// TODO Auto-generated method stub

switch (ev.getAction()) {

case MotionEvent.ACTION\_DOWN:

if (firstVisibleItem == 0) {

isRemark = true;

startY = (int) ev.getY();

}

break;

case MotionEvent.ACTION\_MOVE:

onMove(ev);

break;

case MotionEvent.ACTION\_UP:

if (state == RELESE) {

state = REFLASHING;

// 加载最新数据；

reflashViewByState();

iReflashListener.onReflash();

} else if (state == PULL) {

state = NONE;

isRemark = false;

reflashViewByState();

}

break;

}

return super.onTouchEvent(ev);

}

/\*\*

\* 判断移动过程操作；

\*

\* @param ev

\*/

private void onMove(MotionEvent ev) {

if (!isRemark) {

return;

}

int tempY = (int) ev.getY();

int space = tempY - startY;

int topPadding = space - headerHeight;

switch (state) {

case NONE:

if (space > 0) {

state = PULL;

reflashViewByState();

}

break;

case PULL:

topPadding(topPadding);

if (space > headerHeight + 30

&& scrollState == SCROLL\_STATE\_TOUCH\_SCROLL) {

state = RELESE;

reflashViewByState();

}

break;

case RELESE:

topPadding(topPadding);

if (space < headerHeight + 30) {

state = PULL;

reflashViewByState();

} else if (space <= 0) {

state = NONE;

isRemark = false;

reflashViewByState();

}

break;

}

}

/\*\*

\* 根据当前状态，改变界面显示；

\*/

private void reflashViewByState() {

TextView tip = (TextView) header.findViewById(R.id.tip);

ImageView arrow = (ImageView) header.findViewById(R.id.arrow);

ProgressBar progress = (ProgressBar) header.findViewById(R.id.progress);

RotateAnimation anim = new RotateAnimation(0, 180,

RotateAnimation.RELATIVE\_TO\_SELF, 0.5f,

RotateAnimation.RELATIVE\_TO\_SELF, 0.5f);

anim.setDuration(500);

anim.setFillAfter(true);

RotateAnimation anim1 = new RotateAnimation(180, 0,

RotateAnimation.RELATIVE\_TO\_SELF, 0.5f,

RotateAnimation.RELATIVE\_TO\_SELF, 0.5f);

anim1.setDuration(500);

anim1.setFillAfter(true);

switch (state) {

case NONE:

arrow.clearAnimation();

topPadding(-headerHeight);

break;

case PULL:

arrow.setVisibility(View.VISIBLE);

progress.setVisibility(View.GONE);

tip.setText("下拉可以刷新！");

arrow.clearAnimation();

arrow.setAnimation(anim1);

break;

case RELESE:

arrow.setVisibility(View.VISIBLE);

progress.setVisibility(View.GONE);

tip.setText("松开可以刷新！");

arrow.clearAnimation();

arrow.setAnimation(anim);

break;

case REFLASHING:

topPadding(50);

arrow.setVisibility(View.GONE);

progress.setVisibility(View.VISIBLE);

tip.setText("正在刷新...");

arrow.clearAnimation();

break;

}

}

/\*\*

\* 获取完数据；

\*/

public void reflashComplete() {

state = NONE;

isRemark = false;

reflashViewByState();

TextView lastupdatetime = (TextView) header

.findViewById(R.id.lastupdate\_time);

SimpleDateFormat format = new SimpleDateFormat("yyyy年MM月dd日 hh:mm:ss");

Date date = new Date(System.currentTimeMillis());

String time = format.format(date);

lastupdatetime.setText(time);

}

public void setInterface(IReflashListener iReflashListener){

this.iReflashListener = iReflashListener;

}

/\*\*

\* 刷新数据接口

\* @author Administrator

\*/

public interface IReflashListener{

public void onReflash();

}

}

package com.example.utils;

import android.annotation.SuppressLint;

import android.content.ContentUris;

import android.content.Context;

import android.database.Cursor;

import android.net.Uri;

import android.os.Build;

import android.os.Environment;

import android.provider.DocumentsContract;

import android.provider.MediaStore;

public class GetPath {

/\*\*

\* Uri转为文件路径

\* @param contentUri

\* @return

\*/

@SuppressLint("NewApi")

public static String getPath(final Context context, final Uri uri) {

final boolean isKitKat = Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.KITKAT;

// DocumentProvider

if (isKitKat && DocumentsContract.isDocumentUri(context, uri)) {

if (isExternalStorageDocument(uri)) {

final String docId = DocumentsContract.getDocumentId(uri);

final String[] split = docId.split(":");

final String type = split[0];

if ("primary".equalsIgnoreCase(type)) {

return Environment.getExternalStorageDirectory() + "/"

+ split[1];

}

// TODO handle non-primary volumes

}

// DownloadsProvider

else if (isDownloadsDocument(uri)) {

final String id = DocumentsContract.getDocumentId(uri);

final Uri contentUri = ContentUris.withAppendedId(

Uri.parse("content://downloads/public\_downloads"),

Long.valueOf(id));

return getDataColumn(context, contentUri, null, null);

}

// MediaProvider

else if (isMediaDocument(uri)) {

final String docId = DocumentsContract.getDocumentId(uri);

final String[] split = docId.split(":");

final String type = split[0];

Uri contentUri = null;

if ("image".equals(type)) {

contentUri = MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI;

} else if ("video".equals(type)) {

contentUri = MediaStore.Video.Media.EXTERNAL\_CONTENT\_URI;

} else if ("audio".equals(type)) {

contentUri = MediaStore.Audio.Media.EXTERNAL\_CONTENT\_URI;

}

final String selection = "\_id=?";

final String[] selectionArgs = new String[] { split[1] };

return getDataColumn(context, contentUri, selection,

selectionArgs);

}

}

// MediaStore (and general)

else if ("content".equalsIgnoreCase(uri.getScheme())) {

return getDataColumn(context, uri, null, null);

}

// File

else if ("file".equalsIgnoreCase(uri.getScheme())) {

return uri.getPath();

}

return null;

}

public static String getDataColumn(Context context, Uri uri,

String selection, String[] selectionArgs) {

Cursor cursor = null;

final String column = "\_data";

final String[] projection = { column };

try {

cursor = context.getContentResolver().query(uri, projection,

selection, selectionArgs, null);

if (cursor != null && cursor.moveToFirst()) {

final int column\_index = cursor.getColumnIndexOrThrow(column);

return cursor.getString(column\_index);

}

} finally {

if (cursor != null)

cursor.close();

}

return null;

}

public static boolean isExternalStorageDocument(Uri uri) {

return "com.android.externalstorage.documents".equals(uri

.getAuthority());

}

public static boolean isDownloadsDocument(Uri uri) {

return "com.android.providers.downloads.documents".equals(uri

.getAuthority());

}

public static boolean isMediaDocument(Uri uri) {

return "com.android.providers.media.documents".equals(uri

.getAuthority());

}

}

package com.example.utils;

import android.content.Context;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

public class AlphaSet {

private static SharedPreferences sp;

private static Editor editor;

/\*\*

\* 记录透明度

\* @param context

\* @param topic

\*/

public static void recordAlpha(Context context,int alpha){

sp = context.getSharedPreferences("alpha", context.MODE\_WORLD\_READABLE);

editor = sp.edit();

editor.clear();

editor.commit();//清除之前的记录

editor.putInt("alpha", alpha);

editor.commit();

}

/\*\*

\* 获取透明度

\* @param context

\*/

public static int getAlpha(Context context){

sp = context.getSharedPreferences("alpha", context.MODE\_WORLD\_READABLE);

return sp.getInt("alpha", 255);

}

}

package com.example.utils;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

import android.util.Log;

import com.example.utils.PaintConstants.PATH;

public class SDCardFiles {

public static List<File> getPaintPadPicFiles() {

List<File> imageFiles = new ArrayList<File>();

File f = new File(PATH.SAVE\_PATH);

// 如果当前路径不存在，则创建路径

if (!f.exists()) {

f.mkdirs();

}

File[] files = f.listFiles();

// 如果文件存在

if (files.length != 0) {

/\* 将所有文件存入ArrayList中 \*/

for (int i = 0; i < files.length; i++) {

// 如果是图片文件，则加入file

if (isImageFile(files[i].getPath())) {

imageFiles.add(files[i]);

}

}

}

return imageFiles;

}

public static List<String> getPaintPadPicPaths() {

List<File> imageFiles = getPaintPadPicFiles();

List<String> filePathList = new ArrayList<String>();

for (File file : imageFiles) {

filePathList.add(file.getPath());

}

return filePathList;

}

public static List<String> getPaintPadPicNames() {

List<String> nameList = new ArrayList<String>();

List<File> imageFiles = getPaintPadPicFiles();

for (File file : imageFiles) {

nameList.add(file.getName());

}

return nameList;

}

public static boolean fileNameExists(String name) {

List<String> nameList = getPaintPadPicNames();

if (nameList.contains(name)) {

return true;

}

return false;

}

private static boolean isImageFile(String fName) {

boolean imageExist;

/\* 取得扩展名 \*/

String end = fName

.substring(fName.lastIndexOf(".") + 1, fName.length())

.toLowerCase();

/\* 按扩展名的类型决定MimeType \*/

if (end.equals("jpg") || end.equals("gif") || end.equals("png")

|| end.equals("jpeg") || end.equals("bmp")) {

imageExist = true;

} else {

imageExist = false;

}

return imageExist;

}

}