package voss.TrafficAnalyzer;  
  
import android.Manifest;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.os.Build;  
import android.os.Handler;  
import android.support.v4.app.ActivityCompat;  
import android.support.v4.content.ContextCompat;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.LinearLayout;  
  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class SplashActivity extends AppCompatActivity {  
 private boolean permitted;  
 private String[] perms;  
 private LinearLayout splashFrame;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_splash*);  
  
 splashFrame = (LinearLayout)findViewById(R.id.*splashFrame*);  
  
 perms = new String[]{  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*,  
 Manifest.permission.*MOUNT\_UNMOUNT\_FILESYSTEMS*,  
 Manifest.permission.*CAMERA*,  
 Manifest.permission.*RECORD\_AUDIO*,  
 };  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
 ActivityCompat.*requestPermissions*(SplashActivity.this, perms, 101);  
 }  
 splashFrame.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent();  
 intent.setClass(SplashActivity.this, MainMenuActivity.class);  
 startActivity(intent);  
 }  
 });  
  
  
  
  
 }  
  
 @Override  
 public void onDestroy(){  
 finish();  
 super.onDestroy();  
 }  
}

package voss.TrafficAnalyzer;  
*//// TODO: 按钮美化，图表显示位置*import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Handler;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.LinearLayout;  
import android.widget.Toast;  
  
import java.util.Timer;  
  
public class MainMenuActivity extends AppCompatActivity {  
 private LinearLayout VidRecBtn, IntersecRecBtn, ChangeLogBtn, BrowseBtn;  
 private long exitTime;  
 private AlertDialog ChangeLogDialog, ChangeLogHistDialog;  
 private AlertDialog.Builder CLDBuilder, CLHDBuilder;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main\_menu*);  
  
 VidRecBtn = (LinearLayout)findViewById(R.id.*vidRecBtn*);  
 IntersecRecBtn = (LinearLayout)findViewById(R.id.*intersecRecBtn*);  
 ChangeLogBtn = (LinearLayout)findViewById(R.id.*changeLogBtn*);  
 BrowseBtn = (LinearLayout)findViewById(R.id.*dataRepBtn*);  
  
  
 VidRecBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent intent = new Intent();  
 intent.setClass(MainMenuActivity.this, VidRecordActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 IntersecRecBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent intent = new Intent();  
 intent.setClass(MainMenuActivity.this, IntersecRecActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 BrowseBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent intent = new Intent();  
 intent.setClass(MainMenuActivity.this, BrowseActivity.class);  
 startActivity(intent);  
 }  
 });  
  
  
 ChangeLogBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 CLHDBuilder = new AlertDialog.Builder(MainMenuActivity.this);  
 CLHDBuilder.setTitle("历史记录");  
 CLHDBuilder.setCancelable(true);  
 CLHDBuilder.setItems(getResources().getStringArray(R.array.*changeloghist*), new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
 });  
 CLHDBuilder.setNegativeButton("返回", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(MainMenuActivity.this, "返回主菜单", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 CLDBuilder = new AlertDialog.Builder(MainMenuActivity.this);  
 CLDBuilder.setTitle("更新记录");  
 CLDBuilder.setMessage(getResources().getString(R.string.*changelognew*));  
 CLDBuilder.setNegativeButton("返回", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(MainMenuActivity.this, "返回主菜单", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 CLDBuilder.setPositiveButton("历史记录", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 ChangeLogHistDialog = CLHDBuilder.create();  
 ChangeLogHistDialog.show();  
 }  
 });  
 CLDBuilder.setCancelable(true);  
 ChangeLogDialog = CLDBuilder.create();  
 ChangeLogDialog.show();  
  
 }  
 });  
 }  
  
 @Override  
 public void onBackPressed(){  
 if ((System.*currentTimeMillis*() - exitTime) > 2000) {  
 *// ToastUtil.makeToastInBottom("再按一次退出应用", MainMyselfActivity);* Toast.*makeText*(this, "再按一次退出", Toast.*LENGTH\_SHORT*).show();  
 exitTime = System.*currentTimeMillis*();  
 return;  
 } else {  
 finish();  
 this.onDestroy();  
 }  
 }  
}

package voss.TrafficAnalyzer;  
  
import android.Manifest;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.BitmapFactory;  
import android.hardware.Camera;  
import android.media.CamcorderProfile;  
import android.media.MediaRecorder;  
import android.os.Environment;  
import android.os.Handler;  
import android.os.PowerManager;  
import android.os.SystemClock;  
import android.support.v4.app.ActivityCompat;  
import android.support.v4.content.ContextCompat;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.SurfaceHolder;  
import android.view.SurfaceView;  
import android.view.View;  
import android.widget.Button;  
import android.widget.Chronometer;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.amap.api.location.AMapLocation;  
import com.amap.api.location.AMapLocationClient;  
import com.amap.api.location.AMapLocationClientOption;  
import com.amap.api.location.AMapLocationListener;  
import com.amap.api.maps.AMap;  
import com.amap.api.maps.AMapUtils;  
import com.amap.api.maps.CameraUpdateFactory;  
import com.amap.api.maps.TextureMapView;  
import com.amap.api.maps.model.BitmapDescriptorFactory;  
import com.amap.api.maps.model.LatLng;  
import com.amap.api.maps.model.Marker;  
import com.amap.api.maps.model.MarkerOptions;  
import com.amap.api.maps.model.MyLocationStyle;  
import com.amap.api.maps.model.Polyline;  
import com.amap.api.trace.TraceOverlay;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.text.DecimalFormat;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class VidRecordActivity extends AppCompatActivity implements SurfaceHolder.Callback {  
 private TextureMapView mMapView;  
 private AMap aMap;  
 private AMapLocationClientOption mLocationOption;  
 private AMapLocationClient mLocationClient;  
 private AMapLocation MapLocation;  
 private String userFileName;  
 private File FileName;  
 private File Folder;  
 private File VideoFile;  
 private List<LatLng> locList;  
 private AMapLocationListener mLocationListener;  
 private Handler TimerHandler = new Handler();  
 private Runnable Timer = null;  
 private Timer nTimer;  
 private Handler nHandler;  
 private JSONObject jsObj;  
 private JSONArray jsAry, pointsAry;  
 private FileOutputStream FOS;  
 private boolean Started, Noted, mStartedFlg;  
 private ImageButton StartBtn, CfgBtn, InfoBtn;  
 private double Lat, Lon, Dist;  
 private float Spd, Brn;  
 private String Time, dirDisp, spdDisp;  
 private int Note, tmpNote;  
 private int DataID, PointsID;  
 private int resCfgValue;  
 private byte[] output;  
 private SurfaceView nCameraView;  
 private Polyline nTraceLine;  
 private LatLng latLng;  
 private MarkerOptions markerOptions;  
 private Marker marker;  
 private EditText inputDialog;  
 private Camera nCamera;  
 private Camera.Parameters nParameters;  
 private Camera.AutoFocusCallback mAutoFocusCallback=null;  
 private SurfaceHolder mSurfaceHolder;  
 private MediaRecorder mRecorder;  
 private Button Note1Btn,Note2Btn, Note3Btn, Note4Btn, Note5Btn;  
 private List<Camera.Size> videoSizeList;  
 private Chronometer nChronometer;  
 private long exitTime;  
 private TextView dirText, spdText;  
 private TraceOverlay nTrace;  
 private PowerManager.WakeLock wakeLock;  
 private android.app.AlertDialog.Builder Dialog1B, Dialog2B, Dialog3B, Dialog4B, Dialog5B;  
 private String[] Dialog1List, Dialog2List, Dialog3List, Dialog4List, Dialog5List, CFGItems;  
 private CamcorderProfile vidQuality;  
  
  
 private android.app.AlertDialog ChangeLogDialog, ChangeLogHistDialog, infoDialog;  
 private android.app.AlertDialog.Builder CLDBuilder, CLHDBuilder, infoDialogBuilder;  
  
 private SimpleDateFormat TimeForm = new SimpleDateFormat("yyyy年MM月dd日\_HH时mm分ss秒");  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
  
 setContentView(R.layout.*activity\_vid\_record*);  
  
 *//if (Build.VERSION.SDK\_INT >= 23) {initPerm();}* nTimer = new Timer();  
 nHandler = new Handler();  
 mMapView = (TextureMapView) findViewById(R.id.*map*);  
 mMapView.onCreate(savedInstanceState);*// 此方法必须重写* if (aMap == null) {  
 aMap = mMapView.getMap();  
 }  
 StartBtn = (ImageButton) findViewById(R.id.*startBtn*);  
 CfgBtn = (ImageButton) findViewById(R.id.*cfgBtn*);  
 InfoBtn = (ImageButton)findViewById(R.id.*infoBtn*);  
 Note1Btn = (Button) findViewById(R.id.*note1Btn*);  
 Note2Btn = (Button) findViewById(R.id.*note2Btn*);  
 Note3Btn = (Button) findViewById(R.id.*note3Btn*);  
 Note4Btn = (Button) findViewById(R.id.*note4Btn*);  
 *//Note5Btn = (Button) findViewById(R.id.note5Btn);* nChronometer = (Chronometer) findViewById(R.id.*chronometer*);  
 dirText = (TextView) findViewById(R.id.*dirText*);  
 spdText = (TextView) findViewById(R.id.*spdText*);  
*//1:标线 2:标志 3:设施 4:特殊* Dialog1List = getResources().getStringArray(R.array.*dialoglist1*);  
 Dialog2List = getResources().getStringArray(R.array.*dialoglist2*);  
 Dialog3List = getResources().getStringArray(R.array.*dialoglist3*);  
 Dialog4List = getResources().getStringArray(R.array.*dialoglist4*);  
 *//Dialog5List = getResources().getStringArray(R.array.dialoglist5);* nCameraView = (SurfaceView)findViewById(R.id.*cameraView*);  
  
  
  
  
 mSurfaceHolder = nCameraView.getHolder();  
 mSurfaceHolder.setType(SurfaceHolder.*SURFACE\_TYPE\_PUSH\_BUFFERS*);  
  
 mSurfaceHolder.addCallback(this); *// holder加入回调接口* Folder = new File(Environment.*getExternalStorageDirectory*() + "/Surveyor/");  
 Started = false;  
 jsObj = null;  
 jsAry = null;  
 pointsAry = null;  
 FOS = null;  
 markerOptions = null;  
 FileName = null;  
  
  
  
 readCFG();  
 prepareInfo();  
  
 mAutoFocusCallback=new Camera.AutoFocusCallback() {  
 @Override  
 public void onAutoFocus(boolean success, Camera camera) {  
 if(success){  
 Log.*i*("focus", "true");  
 }  
 }  
 };  
  
 mLocationListener = new AMapLocationListener() {  
 @Override  
 public void onLocationChanged(AMapLocation aMapLocation) {  
 MapLocation = aMapLocation;  
  
 }  
 };  
  
  
  
 MyLocationStyle myLocationStyle;  
 myLocationStyle = new MyLocationStyle();*//初始化定位蓝点样式类myLocationStyle.myLocationType(MyLocationStyle.LOCATION\_TYPE\_LOCATION\_ROTATE);  
 // 连续定位、且将视角移动到地图中心点，定位点依照设备方向旋转，并且会跟随设备移动。（1秒1次定位）如果不设置myLocationType，默认也会执行此种模式。* myLocationStyle.interval(1000); *//设置连续定位模式下的定位间隔，只在连续定位模式下生效，单次定位模式下不会生效。单位为毫秒。* aMap.setMyLocationStyle(myLocationStyle);*//设置定位蓝点的Style* myLocationStyle.myLocationType(MyLocationStyle.*LOCATION\_TYPE\_FOLLOW*) ;*//连续定位、且将视角移动到地图中心点，定位蓝点跟随设备移动。（1秒1次定位）  
//aMap.getUiSettings().setMyLocationButtonEnabled(true);设置默认定位按钮是否显示，非必需设置。* aMap.setMyLocationEnabled(true);*// 设置为true表示启动显示定位蓝点，false表示隐藏定位蓝点并不进行定位，默认是false。* aMap.getUiSettings().setRotateGesturesEnabled(false);  
 aMap.getUiSettings().setTiltGesturesEnabled(false);  
  
  
 mLocationClient = new AMapLocationClient(getApplicationContext());  
*//设置定位回调监听* mLocationClient.setLocationListener(mLocationListener);  
  
 initLocOpt();  
  
 mLocationClient.startLocation();  
  
 aMap.moveCamera(CameraUpdateFactory.*zoomTo*(15));  
  
  
 mSurfaceHolder.setType(SurfaceHolder.*SURFACE\_TYPE\_PUSH\_BUFFERS*);  
  
 if (!Folder.exists()) {  
 Folder.mkdirs();  
 }  
  
*// 1 信号灯，2 让行，3 禁停，4 行人，5 摄像头* Note1Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if (Started && !Noted) {  
 Dialog1B = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 Dialog1B.setSingleChoiceItems(Dialog1List, -1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpNote = which;  
 }  
 });  
 Dialog1B.setCancelable(true);  
 Dialog1B.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Note = 10 + tmpNote;  
 Noted = true;  
 Toast.*makeText*(VidRecordActivity.this, "已标记为：" + Dialog1List[tmpNote], Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 Dialog1B.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "取消标记", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 android.app.AlertDialog tmpDialog = Dialog1B.create();  
 tmpDialog.show();  
 } else if (!Started){  
 Toast.*makeText*(VidRecordActivity.this, "未开始录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 Note2Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if (Started && !Noted) {  
 Dialog2B = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 Dialog2B.setSingleChoiceItems(Dialog2List, -1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpNote = which;  
 }  
 });  
 Dialog2B.setCancelable(true);  
 Dialog2B.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Note = 20 + tmpNote;  
 Noted = true;  
 Toast.*makeText*(VidRecordActivity.this, "已标记为：" + Dialog2List[tmpNote], Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 Dialog2B.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "取消标记", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 android.app.AlertDialog tmpDialog = Dialog2B.create();  
 tmpDialog.show();  
 } else if (!Started){  
 Toast.*makeText*(VidRecordActivity.this, "未开始录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 Note3Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (Started && !Noted) {  
 Dialog3B = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 Dialog3B.setSingleChoiceItems(Dialog3List, -1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpNote = which;  
 }  
 });  
 Dialog3B.setCancelable(true);  
 Dialog3B.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Note = 30 + tmpNote;  
 Noted = true;  
 Toast.*makeText*(VidRecordActivity.this, "已标记为：" + Dialog3List[tmpNote], Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 Dialog3B.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "取消标记", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 android.app.AlertDialog tmpDialog = Dialog3B.create();  
 tmpDialog.show();  
 } else if (!Started){  
 Toast.*makeText*(VidRecordActivity.this, "未开始录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 Note4Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if (Started && !Noted) {  
 Dialog4B = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 Dialog4B.setSingleChoiceItems(Dialog4List, -1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpNote = which;  
 }  
 });  
 Dialog4B.setCancelable(true);  
 Dialog4B.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Note = 20 + tmpNote;  
 Noted = true;  
 Toast.*makeText*(VidRecordActivity.this, "已标记为：" + Dialog4List[tmpNote], Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 Dialog4B.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "取消标记", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 android.app.AlertDialog tmpDialog = Dialog4B.create();  
 tmpDialog.show();  
 } else if (!Started){  
 Toast.*makeText*(VidRecordActivity.this, "未开始录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
  
  
  
 CfgBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if (!Started) {  
 Intent intent = new Intent();  
 intent.setClass(VidRecordActivity.this, VidRecCfgActivity.class);  
 startActivity(intent);  
 } else {  
 Toast.*makeText*(VidRecordActivity.this, "正在录制！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 InfoBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if (!Started) {  
 *//check prepareInfo* ChangeLogHistDialog = CLHDBuilder.create();  
 ChangeLogDialog = CLDBuilder.create();  
 infoDialog = infoDialogBuilder.create();  
  
 infoDialog.show();  
 } else {  
 Toast.*makeText*(VidRecordActivity.this, "正在录制！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 acquireWakeLock();  
 }  
  
  
 @Override  
 protected void onDestroy() {  
 *//在activity执行onDestroy时执行mMapView.onDestroy()，销毁地图* mMapView.onDestroy();  
 File tempfile = new File(Folder + "/temp.mp4");  
 if (tempfile.exists()){  
 tempfile.delete();  
 }  
  
 TimerHandler.removeCallbacks(Timer);  
 releaseWakeLock();  
 finish();  
 super.onDestroy();  
 }  
 @Override  
 protected void onResume() {  
 super.onResume();  
 *//在activity执行onResume时执行mMapView.onResume ()，重新绘制加载地图* mMapView.onResume();  
 TimerHandler.postDelayed(Timer, 2000);  
 acquireWakeLock();  
  
 }  
 @Override  
 protected void onPause() {  
 super.onPause();  
 *//在activity执行onPause时执行mMapView.onPause ()，暂停地图的绘制* mMapView.onPause();  
  
 TimerHandler.removeCallbacks(Timer);  
 releaseWakeLock();  
  
 }  
 @Override  
 protected void onSaveInstanceState(Bundle outState) {  
 super.onSaveInstanceState(outState);  
 *//在activity执行onSaveInstanceState时执行mMapView.onSaveInstanceState (outState)，保存地图当前的状态* mMapView.onSaveInstanceState(outState);  
 }  
  
 public void initPerm(){  
 boolean permitted = (ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_GRANTED* && ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_GRANTED* && ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED* && ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*MOUNT\_UNMOUNT\_FILESYSTEMS*) == PackageManager.*PERMISSION\_GRANTED* && ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED* && ContextCompat.*checkSelfPermission*(this,  
 Manifest.permission.*RECORD\_AUDIO*) == PackageManager.*PERMISSION\_GRANTED*);  
  
 if (!permitted) {  
 ActivityCompat.*requestPermissions*(VidRecordActivity.this, new String[]{  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*,  
 Manifest.permission.*MOUNT\_UNMOUNT\_FILESYSTEMS*,  
 Manifest.permission.*CAMERA*,  
 Manifest.permission.*RECORD\_AUDIO* }, 101);  
 }  
 }  
  
 public void initLocOpt(){  
 mLocationOption = new AMapLocationClientOption();  
 mLocationOption.setLocationMode(AMapLocationClientOption.AMapLocationMode.*Hight\_Accuracy*);  
 mLocationOption.setInterval(1000);  
 mLocationOption.setMockEnable(true);  
 mLocationClient.setLocationOption(mLocationOption);  
 }  
  
 private void startAction(){  
 if (!Started) {  
 Started = true;  
 locList = new ArrayList<LatLng>();  
 jsAry = new JSONArray();  
 pointsAry = new JSONArray();  
 jsObj = new JSONObject();  
 output = null;  
 FOS = null;  
 initTimer();  
 DataID = 0;  
 PointsID = 0;  
 Dist = 0;  
 if (marker != null){  
 marker.destroy();  
 }  
  
  
  
 if (mRecorder == null) {  
 mRecorder = new MediaRecorder(); *// Create MediaRecorder* }  
 try {  
 nCamera.unlock();  
 mRecorder.setCamera(nCamera);  
 *// Set audio and video source and encoder  
 // 这两项需要放在setOutputFormat之前* mRecorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);  
 mRecorder.setVideoSource(MediaRecorder.VideoSource.*CAMERA*);  
  
 mRecorder.setProfile(vidQuality);  
  
  
  
 mRecorder.setPreviewDisplay(mSurfaceHolder.getSurface());  
  
  
 Folder.mkdirs();  
 VideoFile = new File(Folder + "/temp.mp4");  
 mRecorder.setOutputFile(VideoFile.toString());  
 mRecorder.prepare();  
 mRecorder.start(); *// Recording is now started* mStartedFlg = true;  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 nChronometer.setBase(SystemClock.*elapsedRealtime*());  
 nChronometer.start();  
 }  
 }  
  
 private void stopAction(){  
 if (Started) {  
  
 try {  
 *//下面三个参数必须加，不加的话会奔溃，在mediarecorder.stop();  
 //报错为：RuntimeException:stop failed* mRecorder.setOnErrorListener(null);  
 mRecorder.setOnInfoListener(null);  
 mRecorder.setPreviewDisplay(null);  
 mRecorder.stop();  
 } catch (IllegalStateException e) {  
 Log.*i*("Exception", Log.*getStackTraceString*(e));  
 }catch (RuntimeException e) {  
 Log.*i*("Exception", Log.*getStackTraceString*(e));  
 }catch (Exception e) {  
 Log.*i*("Exception", Log.*getStackTraceString*(e));  
 }  
 mRecorder.reset();  
  
 inputDialog = new EditText(this);  
 inputDialog.setFocusable(true);  
 inputDialog.setText(Time);  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("请输入文件名");  
 builder.setView(inputDialog);  
 builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 userFileName = Time;  
 saveFile();  
 }  
 });  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 userFileName = inputDialog.getText().toString();  
 saveFile();  
 }  
 });  
 builder.setCancelable(true);  
 builder.show();  
  
 if (nTimer != null) {  
 nTimer.cancel();  
 nTimer.purge();  
 }  
  
 if (mRecorder != null) {  
 mRecorder.release();  
 mRecorder = null;  
 }  
 nChronometer.stop();  
 Started = false;  
 }  
 }  
  
 private void initTimer(){  
 nTimer = new Timer();  
 nTimer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 if (Started){  
 nHandler.post(new Runnable() {  
 @Override  
 public void run() {  
 getLoc();  
 dispInfo();  
 writeLog();  
 Note = 0;  
 Noted = false;  
  
 }  
 });  
 }  
 }  
 }, 0, 1000);  
  
 }  
  
 private void getLoc(){  
 if (MapLocation != null) {  
 if (MapLocation.getLocationType() != 0) {  
 Lat = MapLocation.getLatitude();  
 Lon = MapLocation.getLongitude();  
 Brn = MapLocation.getBearing();  
 Spd = MapLocation.getSpeed();  
 }  
 }  
 LatLng locTmp = latLng;  
  
 latLng = new LatLng(Lat, Lon);  
  
 Dist += AMapUtils.*calculateLineDistance*(locTmp, latLng);  
  
 locList.add(latLng);  
  
 */\*  
 if (nTraceLine != null){  
 nTraceLine.remove();  
 }  
  
 nTraceLine = aMap.addPolyline((new PolylineOptions())  
 .addAll(locList)  
 .width(10)  
 .color(Color.argb(255, 1, 1, 1)));  
\*/* nTrace = new TraceOverlay(aMap, locList);  
  
 if (Noted) {  
 placePoint();  
 }  
  
 }  
  
  
  
  
 private void writeLog(){  
 Time = TimeForm.format(new java.util.Date());  
 JSONObject tmpJS = new JSONObject();  
 JSONObject tmpJS2 = new JSONObject();  
  
 try {  
 tmpJS.put("ID", DataID++);  
 tmpJS.put("Lat", Lat);  
 tmpJS.put("Lon", Lon);  
 tmpJS.put("Bearing", Brn);  
 tmpJS.put("Speed", Spd);  
 tmpJS.put("Time", Time);  
 tmpJS.put("Distance", Dist);  
 jsAry.put(tmpJS);  
 if (Noted) {  
 tmpJS2.put("ID", PointsID++);  
 tmpJS2.put("Type", Note);  
 tmpJS2.put("Lat", Lat);  
 tmpJS2.put("Lon", Lon);  
 tmpJS2.put("Distance", Dist);  
 pointsAry.put(tmpJS2);  
 }  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 private void saveFile(){  
  
 if (!Folder.exists()) {  
 Folder.mkdirs();  
 }  
 FileName = new File(Folder + "/" + userFileName + ".vlog.json");  
 Toast.*makeText*(this, "已保存为：" + FileName, Toast.*LENGTH\_SHORT*).show();  
 try {  
 File file = VideoFile;  
 VideoFile = new File(Folder + "/" + userFileName + ".vlog.mp4");  
 file.renameTo(VideoFile);  
  
 FileName.createNewFile();  
 jsObj.put("MainTable", jsAry);  
 jsObj.put("PointsTable", pointsAry);  
 jsObj.put("VideoPath", "/Surveyor/" + userFileName + ".vlog.mp4");  
 output = jsObj.toString().getBytes();  
 FOS = new FileOutputStream(FileName);  
 FOS.write(output);  
 FOS.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 public void initCamera()  
 {  
  
 if(nCamera == null)  
 {  
 nCamera = Camera.*open*();  
 }  
 if(nCamera != null) {  
 try {  
 nParameters = nCamera.getParameters();  
 nParameters.setPreviewSize(960, 544);  
 nParameters.setFocusMode(Camera.Parameters.*FOCUS\_MODE\_CONTINUOUS\_VIDEO*);  
  
  
 nCamera.setParameters(nParameters);  
 nCamera.setPreviewDisplay(mSurfaceHolder);  
 nCamera.startPreview();  
  
 Camera.Parameters parameter=nCamera.getParameters();  
 videoSizeList = parameter.getSupportedVideoSizes();  
  
*//查找出最接近的视频录制分辨率* } catch (Exception e) {  
 e.printStackTrace();  
 Toast.*makeText*(VidRecordActivity.this, "初始化相机错误", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
  
 @Override  
 public void surfaceChanged(SurfaceHolder holder, int format, int width,  
 int height) {  
 mSurfaceHolder = holder;  
 }  
  
 @Override  
 public void surfaceCreated(SurfaceHolder holder) {  
 mSurfaceHolder = holder;  
 initCamera();  
 StartBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (Started){  
 stopAction();  
 StartBtn.setImageDrawable(getDrawable(android.R.drawable.*ic\_notification\_overlay*));  
 } else {  
 startAction();  
 StartBtn.setImageDrawable(getDrawable(R.drawable.*stop*));  
 }  
 }  
 });  
  
 }  
  
 @Override  
 public void surfaceDestroyed(SurfaceHolder holder) {  
 *// surfaceDestroyed的时候同时对象设置为null* nCameraView = null;  
 mSurfaceHolder = null;  
 if (mRecorder != null) {  
 mRecorder.release();  
 mRecorder = null;  
 }  
 }  
  
 @Override  
 public void onBackPressed(){  
 if (Started) {  
 Toast.*makeText*(this, "正在录制！", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 finish();  
 super.onBackPressed();  
 }  
  
 }  
  
 public void dispInfo(){  
 float convertedSpd = Spd \* 3.6f;  
 DecimalFormat decimalFormat=new DecimalFormat("000");  
 spdDisp = "速度：" + decimalFormat.format(convertedSpd) + "km/h";  
  
 dirDisp = Brn + "方向：";  
 if (Brn == -1){  
 dirDisp += "未确定方向";  
 } else if (Brn <= 22.5 || Brn > 337.5) {  
 dirDisp += "正北";  
 } else if (Brn > 22.5 && Brn <= 67.5) {  
 dirDisp += "东北";  
 } else if (Brn > 67.5 && Brn <= 112.5){  
 dirDisp += "正东";  
 } else if (Brn > 112.5 && Brn <= 157.5){  
 dirDisp += "东南";  
 } else if (Brn > 157.5 && Brn <= 202.5){  
 dirDisp += "正南";  
 } else if (Brn > 202.5 && Brn <= 247.5){  
 dirDisp += "西南";  
 } else if (Brn > 247.5 && Brn <= 292.5){  
 dirDisp += "正西";  
 } else if (Brn > 292.5 && Brn <= 337.5){  
 dirDisp += "西北";  
 }  
  
 spdText.setText(spdDisp);  
 dirText.setText(dirDisp);  
  
 }  
  
 public void acquireWakeLock()  
 {  
 if (wakeLock == null)  
 {  
 PowerManager pm = (PowerManager)this.getSystemService(Context.*POWER\_SERVICE*);  
 wakeLock = pm.newWakeLock(PowerManager.*FULL\_WAKE\_LOCK*|PowerManager.*ON\_AFTER\_RELEASE*, "PostLocationService");  
 if (null != wakeLock)  
 {  
 wakeLock.acquire();  
 }  
 }  
 }  
  
 public void releaseWakeLock()  
 {  
 if (wakeLock != null)  
 {  
 wakeLock.release();  
 wakeLock = null;  
 }  
 }  
  
 public void readCFG(){  
 File cfgFile = new File(Environment.*getExternalStorageDirectory*()  
 + "/Surveyor/Config/VidRec.cfg");  
 if (!cfgFile.exists()) {  
 resCfgValue = 0;  
 Toast.*makeText*(this, "视频质量默认为480P，请设置视频质量", Toast.*LENGTH\_LONG*).show();  
 vidQuality = CamcorderProfile.*get*(CamcorderProfile.*QUALITY\_480P*);  
 return;  
 }  
 CFGItems = getResources().getStringArray(R.array.*vidRecCfgList*);  
 String tempString = new String();  
 try {  
 InputStream is = new FileInputStream(cfgFile);  
 InputStreamReader streamReader = new InputStreamReader(is);  
 BufferedReader reader = new BufferedReader(streamReader);  
 String line = null;  
 StringBuilder stringBuilder = new StringBuilder();  
 while ((line = reader.readLine()) != null) {  
 stringBuilder.append(line);  
 }  
 tempString = stringBuilder.toString();  
 is.close();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 try {  
 JSONObject cfgObj = new JSONObject(tempString);  
  
 if (!cfgObj.getString("Version").equals(getResources().getString(R.string.*vercode*))){  
 resCfgValue = 0;  
 Toast.*makeText*(this, "视频质量默认为480P", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 resCfgValue = cfgObj.getInt(CFGItems[0]);  
 }  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 switch (resCfgValue){  
 case 0:  
 vidQuality = CamcorderProfile.*get*(CamcorderProfile.*QUALITY\_480P*);  
 Toast.*makeText*(this, "视频质量为480P", Toast.*LENGTH\_SHORT*).show();  
  
 break;  
 case 1:  
 vidQuality = CamcorderProfile.*get*(CamcorderProfile.*QUALITY\_720P*);  
 Toast.*makeText*(this, "视频质量为720P", Toast.*LENGTH\_SHORT*).show();  
 break;  
 case 2:  
 vidQuality = CamcorderProfile.*get*(CamcorderProfile.*QUALITY\_1080P*);  
 Toast.*makeText*(this, "视频质量为1080P", Toast.*LENGTH\_SHORT*).show();  
 break;  
 }  
  
 }  
  
 private void placePoint() {  
 markerOptions = new MarkerOptions().position(latLng).draggable(false);  
 switch (Note){  
 case 10:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s10*)));  
  
 break;  
 case 11:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s11*)));  
  
 break;  
 case 12:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s12*)));  
  
 break;  
 case 13:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s13*)));  
  
 break;  
 case 14:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s14*)));  
  
 break;  
 case 15:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s15*)));  
  
 break;  
 case 20:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s20*)));  
  
 break;  
 case 21:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s21*)));  
  
 break;  
 case 22:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s22*)));  
  
 break;  
 case 23:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s23*)));  
  
 break;  
 case 24:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s24*)));  
  
 break;  
 case 25:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s25*)));  
  
 break;  
 case 30:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s30*)));  
  
 break;  
 case 31:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s31*)));  
  
 break;  
 case 32:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s32*)));  
  
 break;  
 case 33:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s33*)));  
  
 break;  
 case 34:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s34*)));  
  
 break;  
 case 35:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s35*)));  
  
 break;  
 case 40:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s40*)));  
  
 break;  
 case 41:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s41*)));  
  
 break;  
 case 42:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s42*)));  
  
 break;  
 case 43:  
 markerOptions.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s43*)));  
  
 break;  
  
 }  
 marker = aMap.addMarker(markerOptions);  
 }  
  
 public void prepareInfo(){  
 CLHDBuilder = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 CLHDBuilder.setTitle("历史记录");  
 CLHDBuilder.setItems(getResources().getStringArray(R.array.*changeloghist*), new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
 });  
 CLHDBuilder.setNegativeButton("返回", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "返回录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 CLDBuilder = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 CLDBuilder.setTitle("更新内容");  
 CLDBuilder.setMessage(getResources().getString(R.string.*changelognew*));  
 CLDBuilder.setNegativeButton("返回", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "返回录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 CLDBuilder.setPositiveButton("历史更新", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 ChangeLogHistDialog.show();  
 }  
 });  
  
 infoDialogBuilder = new android.app.AlertDialog.Builder(VidRecordActivity.this);  
 infoDialogBuilder.setTitle("说明");  
 infoDialogBuilder.setCancelable(true);  
 infoDialogBuilder.setItems(getResources().getStringArray(R.array.*vidRecInfo*), null);  
 infoDialogBuilder.setNegativeButton("返回", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(VidRecordActivity.this, "返回录制", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 infoDialogBuilder.setPositiveButton("更新内容", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 ChangeLogDialog.show();  
 }  
 });  
  
 infoDialogBuilder.setCancelable(true);  
 CLDBuilder.setCancelable(true);  
 CLHDBuilder.setCancelable(true);  
  
  
  
 }  
}

package voss.TrafficAnalyzer;  
  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Environment;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
  
public class VidRecCfgActivity extends AppCompatActivity {  
 private Button posBtn, negBtn, L1Btn;  
 private TextView textL1;  
 private JSONObject cfgObj;  
 private AlertDialog resMenu;  
 private AlertDialog.Builder resMenuBuilder;  
 private String[] CFGItems, resItems;  
 private int tmpValue, resCfgValue;  
 private File filePos;  
 private StringBuilder stringBuilder;  
 private String tempString;  
 private FileOutputStream FOS;  
 private int[] cfgValues;  
 private boolean cfgVersion;  
 private byte[] output;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_vid\_rec\_cfg*);  
  
 posBtn = (Button)findViewById(R.id.*confirmVRC*);  
 negBtn = (Button)findViewById(R.id.*cancelVRC*);  
 textL1 = (TextView)findViewById(R.id.*textViewL1*);  
 L1Btn = (Button)findViewById(R.id.*buttonL1*);  
  
 resItems = getResources().getStringArray(R.array.*resList*);  
 CFGItems = getResources().getStringArray(R.array.*vidRecCfgList*);  
  
  
  
 filePos = new File(Environment.*getExternalStorageDirectory*() + "/Surveyor/Config/");  
 if (!filePos.exists()){  
 filePos.mkdirs();  
 }  
 filePos = new File(filePos + "/VidRec.cfg");  
 if (!filePos.exists()) {  
 createCFG();  
 } else {  
 readCFG();  
 }  
 if (!cfgVersion) {  
 createCFG();  
 }  
  
  
  
  
 L1Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 resCfg();  
 }  
 });  
  
  
  
  
 posBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 cfgValues = new int[]{resCfgValue};  
 writeCFG(cfgValues);  
  
 Intent intent = new Intent();  
 intent.setClass(VidRecCfgActivity.this, VidRecordActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 negBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent();  
 intent.setClass(VidRecCfgActivity.this, VidRecordActivity.class);  
 startActivity(intent);  
 }  
 });  
 }  
  
  
*//设置项：1.分辨率 2.* public void createCFG() {  
 int[] tmpValues;  
 tmpValues = new int[CFGItems.length-1];  
 for (int i = 0; i < tmpValues.length; i ++){  
 tmpValues[i] = 0;  
 }  
 writeCFG(tmpValues);  
 readCFG();  
  
  
 }  
  
 public void writeCFG(int[] Values){  
 try {  
 cfgObj = new JSONObject();  
 filePos.createNewFile();  
 for (int i = 0; i < Values.length; i++){  
 cfgObj.put(CFGItems[i], Values[i]);  
 }  
 cfgObj.put("Version", getResources().getString(R.string.*vercode*));  
  
 output = cfgObj.toString().getBytes();  
 FOS = new FileOutputStream(filePos);  
 FOS.write(output);  
 FOS.close();  
 } catch (IOException | JSONException e) {  
 e.printStackTrace();  
 }  
  
  
  
  
  
  
 }  
  
 public void readCFG(){  
 try {  
 InputStream is = new FileInputStream(filePos);  
 InputStreamReader streamReader = new InputStreamReader(is);  
 BufferedReader reader = new BufferedReader(streamReader);  
 String line = null;  
 stringBuilder = new StringBuilder();  
 while ((line = reader.readLine()) != null) {  
 stringBuilder.append(line);  
 }  
 tempString = stringBuilder.toString();  
 is.close();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 try {  
 cfgObj = new JSONObject(tempString);  
  
 if (!cfgObj.getString("Version").equals(getResources().getString(R.string.*vercode*))){  
 cfgVersion = false;  
 return;  
 } else {  
 cfgVersion = true;  
 }  
  
 resCfgValue = cfgObj.getInt(CFGItems[0]);  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 textL1.setText(resItems[resCfgValue]);  
  
 }  
  
 public void resCfg(){  
 resMenuBuilder = new AlertDialog.Builder(VidRecCfgActivity.this);  
 resMenuBuilder.setTitle("设置分辨率");  
 resMenuBuilder.setCancelable(true);  
 resMenuBuilder.setSingleChoiceItems(resItems, resCfgValue, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpValue = which;  
 }  
 });  
 resMenuBuilder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 resCfgValue = tmpValue;  
 Toast.*makeText*(VidRecCfgActivity.this, "已设置分辨率为：" + resItems[resCfgValue]  
 , Toast.*LENGTH\_SHORT*).show();  
 textL1.setText(resItems[resCfgValue]);  
 }  
 });  
 resMenu = resMenuBuilder.create();  
 resMenu.show();  
  
 }  
  
}

package voss.TrafficAnalyzer;  
  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.os.Environment;  
import android.os.Handler;  
import android.os.PowerManager;  
import android.os.SystemClock;  
import android.os.Vibrator;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.MotionEvent;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.amap.api.location.AMapLocation;  
import com.amap.api.location.AMapLocationClient;  
import com.amap.api.location.AMapLocationClientOption;  
import com.amap.api.location.AMapLocationListener;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.text.DecimalFormat;  
import java.text.SimpleDateFormat;  
import java.util.Arrays;  
import java.util.Date;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class IntersecRecActivity extends AppCompatActivity {  
 private TextView textName, textDirection, text00, text01, text10, text11, textCountDownMin,  
 textCountDownSec, dispL, dispN, dispR;  
 *// 00：直行 01：右转 10：左转 11：左转控制* private Button btnSave;  
 private ImageButton btnHelp, btnL, btnN, btnR;  
 private ImageView arrowImage;  
 private String mName, directionCode, fileName;  
 private int mDirection, Linfo, Ninfo, Rinfo, countL, countN, countR, CDmin, CDsec,  
 tmpSelection, recMin, recSec, timeMil, tmpMin,tmpSec,tmpL,tmpN,tmpR, timingL,  
 timingN, timingR;  
 private boolean turnControl, Counting, Saved, Located;  
 private EditText inputnameText, inputMin, inputSec, inputFileName;  
 private Timer timer;  
 private Handler handler;  
 private DecimalFormat timeFormat, countFormat;  
 private AlertDialog.Builder sbuilder,dbuilder;  
 private Integer[] DirectionUsed;  
 private File folder, outputFile;  
 private SimpleDateFormat TimeForm;  
 private JSONObject outputObj, tmpObj;  
 private JSONArray LtmpAry, NtmpAry, RtmpAry, LtimAry, NtimAry, RtimAry;  
 private long baseTime, timingL0, timingN0, timingR0;  
 private double Lat, Lon;  
  
 private Vibrator vibe;  
 private PowerManager.WakeLock wakeLock;  
  
 public AMapLocationClient mLocationClient;  
 public AMapLocationListener mLocationListener;  
 public AMapLocationClientOption mLocationOption;  
  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_intersec\_rec*);  
  
  
  
 textName = (TextView)findViewById(R.id.*textName*);  
 textDirection = (TextView)findViewById(R.id.*textDirection*);  
 text00 = (TextView)findViewById(R.id.*textL00*);  
 text01 = (TextView)findViewById(R.id.*textL01*);  
 text10 = (TextView)findViewById(R.id.*textL10*);  
 text11 = (TextView)findViewById(R.id.*textL11*);  
 textCountDownMin = (TextView)findViewById(R.id.*textCountDownMin*);  
 textCountDownSec = (TextView)findViewById(R.id.*textCountDownSec*);  
 dispL = (TextView)findViewById(R.id.*lTurnNum*);  
 dispN = (TextView)findViewById(R.id.*nTurnNum*);  
 dispR = (TextView)findViewById(R.id.*rTurnNum*);  
 btnSave = (Button)findViewById(R.id.*saveBtn*);  
 btnHelp = (ImageButton) findViewById(R.id.*helpBtn*);  
 btnL = (ImageButton) findViewById(R.id.*lTurnBtn*);  
 btnN = (ImageButton) findViewById(R.id.*nTurnBtn*);  
 btnR = (ImageButton) findViewById(R.id.*rTurnBtn*);  
 arrowImage = (ImageView)findViewById(R.id.*arrowImage*);  
  
 vibe = (Vibrator) getSystemService(*VIBRATOR\_SERVICE*);  
  
 timeFormat = new DecimalFormat("00");  
 countFormat = new DecimalFormat("000");  
 tmpObj = new JSONObject();  
 outputObj = new JSONObject();  
 folder = new File(Environment.*getExternalStorageDirectory*() + "/Surveyor/");  
 TimeForm = new SimpleDateFormat("yyyy年MM月dd日\_HH时mm分ss秒");  
 DirectionUsed = null;  
 directionCode = "N";  
 Saved = true;  
  
 mLocationListener = new AMapLocationListener(){  
 @Override  
 public void onLocationChanged(AMapLocation amapLocation) {  
 if (amapLocation != null) {  
 if (amapLocation.getErrorCode() == 0) {  
  
  
 Lat = amapLocation.getLatitude();  
 Lon = amapLocation.getLongitude();  
 Located = true;  
 }  
 }  
 }  
 };  
 mLocationClient = new AMapLocationClient(getApplicationContext());  
 mLocationClient.setLocationListener(mLocationListener);  
 mLocationOption = new AMapLocationClientOption();  
 mLocationOption.setLocationMode(AMapLocationClientOption.AMapLocationMode.*Hight\_Accuracy*);  
 mLocationOption.setOnceLocationLatest(true);  
  
  
  
  
 acquireWakeLock();  
  
  
 textName.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 inputnameText = new EditText(IntersecRecActivity.this);  
 if (mName!=null){  
 inputnameText.setText(mName);  
 }  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setTitle("请输入路口名");  
 builder.setView(inputnameText);  
 builder.setCancelable(true);  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (mName != inputnameText.getText().toString()){  
 Saved = false;  
 }  
 mName = inputnameText.getText().toString();  
 textName.setText(mName);  
 }  
 });  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 textDirection.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
  
 dbuilder = new AlertDialog.Builder(IntersecRecActivity.this);  
 dbuilder.setCancelable(true);  
 dbuilder.setTitle("此方向已统计过");  
  
 dbuilder.setPositiveButton("是", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 setDir();  
  
 outputObj.remove(directionCode);  
  
 countL = 0;  
 countN = 0;  
 countR = 0;  
 dispL.setText("000");  
 dispN.setText("000");  
 dispR.setText("000");  
 }  
 });  
 dbuilder.setNegativeButton("否", null);  
  
 sbuilder = new AlertDialog.Builder(IntersecRecActivity.this);  
 sbuilder.setCancelable(true);  
 sbuilder.setTitle("是否切换方向？");  
 sbuilder.setPositiveButton("是", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 setDir();  
  
  
 countL = 0;  
 countN = 0;  
 countR = 0;  
 dispL.setText("000");  
 dispN.setText("000");  
 dispR.setText("000");  
 }  
 });  
 sbuilder.setNegativeButton("否", null);  
  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setCancelable(true);  
 builder.setTitle("请选择方向");  
 builder.setSingleChoiceItems(R.array.*intersecDir*, tmpSelection, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpSelection = which;  
  
 }  
 });  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (DirectionUsed != null && Arrays.*asList*(DirectionUsed).contains(tmpSelection)) {  
  
 try {  
 String[] tmpary = getResources().getStringArray(R.array.*intersecDirS*);  
 tmpObj = new JSONObject();  
 tmpObj = outputObj.getJSONObject(tmpary[tmpSelection]);  
 tmpMin = tmpObj.getInt("Min");  
 tmpSec = tmpObj.getInt("Sec");  
 tmpL = tmpObj.getInt("LTurn");  
 tmpN = tmpObj.getInt("NTurn");  
 tmpR = tmpObj.getInt("RTurn");  
  
 dbuilder.setMessage("是否重新统计？（时长：" + tmpMin + "分 " + tmpSec +  
 "秒，左转：" + tmpL + "，直行：" + tmpN + ",右转：" + tmpR + " ）");  
 dbuilder.create().show();  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
  
 }else if (countL == 0 && countN == 0 && countR ==0){  
 setDir();  
 } else {  
 sbuilder.create().show();  
 }  
 }  
 }  
 );  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 *// 00：直行 01：右转 10：左转 11：左转控制* text00.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 tmpSelection = 0;  
 String[] tmpCount = {0 + "", 1 + "", 2 + "", 3 + "", 4 + "", 5 + "", 6 + ""};  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setCancelable(true);  
 builder.setTitle("请选择直行车道数量");  
 builder.setSingleChoiceItems(tmpCount, 1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpSelection = which;  
 }  
 });  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (Ninfo!=tmpSelection) {  
 Saved = false;  
 }  
 Ninfo = tmpSelection;  
 text00.setText(Ninfo + "");  
 }  
 });  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 text01.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 tmpSelection = 0;  
 String[] tmpCount = {0 + "", 1 + "", 2 + "", 3 + ""};  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setCancelable(true);  
 builder.setTitle("请选择右转车道数量");  
 builder.setSingleChoiceItems(tmpCount, 1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpSelection = which;  
 Toast.*makeText*(IntersecRecActivity.this, tmpSelection + "", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (Rinfo!=tmpSelection) {  
 Saved = false;  
 }  
 Rinfo = tmpSelection;  
 text01.setText(Rinfo + "");  
 }  
 });  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 text10.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 tmpSelection = 0;  
 String[] tmpCount = {0 + "", 1 + "", 2 + "", 3 + ""};  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setCancelable(true);  
 builder.setTitle("请选择左转车道数量");  
 builder.setSingleChoiceItems(tmpCount, 1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 tmpSelection = which;  
 }  
 });  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (Linfo!=tmpSelection) {  
 Saved = false;  
 }  
 Linfo = tmpSelection;  
 text10.setText(Linfo + "");  
 }  
 });  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 text11.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setCancelable(true);  
 builder.setTitle("请选择左转控制状态");  
 builder.setPositiveButton("有", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (!turnControl){  
  
 Saved = false;  
 }  
 turnControl = true;  
 text11.setText("有");  
 }  
 });  
 builder.setNeutralButton("取消", null);  
 builder.setNegativeButton("无", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if(turnControl){  
  
 Saved = false;  
 }  
 turnControl = false;  
 text11.setText("无");  
 }  
 });  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 *// 00：直行 01：右转 10：左转 11：左转控制* textCountDownMin.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 inputMin = new EditText(IntersecRecActivity.this);  
 inputMin.setHint("分钟（0~99）");  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setTitle("请输入倒计时分钟数");  
 builder.setView(inputMin);  
 builder.setCancelable(true);  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 try {  
 CDmin = Integer.*parseInt*(inputMin.getText().toString());  
 } catch (Exception e) {  
 CDmin = 0;  
 }  
 if (CDmin >= 0 && CDmin <= 99) {  
 textCountDownMin.setText("" + timeFormat.format(CDmin));  
 recMin = CDmin;  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "请输入正确的分钟数", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 textCountDownSec.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting) {  
 inputSec = new EditText(IntersecRecActivity.this);  
 inputSec.setHint("秒（0~59）");  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setTitle("请输入倒计时秒数");  
 builder.setView(inputSec);  
 builder.setCancelable(true);  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 try {  
 CDsec = Integer.*parseInt*(inputSec.getText().toString());  
 } catch (Exception e) {  
 CDsec = 0;  
 }  
 if (CDsec >= 0 && CDsec < 60) {  
 textCountDownSec.setText("" + timeFormat.format(CDsec));  
 recSec = CDsec;  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "请输入正确的秒数！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 builder.setNegativeButton("取消", null);  
 builder.create().show();  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
  
  
  
 btnSave.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting){  
 if (mName != null) {  
  
 fileName = TimeForm.format(new Date());  
  
 try {  
 tmpObj = new JSONObject();  
 tmpObj.put("Name", mName);  
 tmpObj.put("nLanes", Ninfo);  
 tmpObj.put("rLanes", Rinfo);  
 tmpObj.put("lLanes", Linfo);  
 tmpObj.put("turnConrtol", turnControl);  
 tmpObj.put("Date", fileName);  
 tmpObj.put("Lat", Lat);  
 tmpObj.put("Lon", Lon);  
  
 outputObj.put("Info", tmpObj);  
  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 fileName = mName + "\_" + fileName;  
  
 inputFileName = new EditText(IntersecRecActivity.this);  
 inputFileName.setText(fileName);  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setTitle("请输入文件名");  
 builder.setView(inputFileName);  
 builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 outputObj.remove("Info");  
 }  
 });  
 builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 fileName = inputFileName.getText().toString();  
 saveFile();  
 }  
 });  
 builder.setCancelable(true);  
 builder.show();  
  
  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "未输入路口名！", Toast.*LENGTH\_SHORT*).show();  
 }  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 btnHelp.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Counting){  
  
 AlertDialog.Builder builder = new android.app.AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setTitle("说明");  
 builder.setCancelable(true);  
 builder.setItems(getResources().getStringArray(R.array.*intersecRecInfo*), null);  
 builder.setNegativeButton("返回", null);  
 builder.create().show();  
  
 } else {  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 } }  
 });  
 btnL.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()){  
 case MotionEvent.*ACTION\_DOWN*:  
  
 startTiming();  
 if (Counting) {  
  
 timingL0 = SystemClock.*elapsedRealtime*();  
  
 timeMil = (int) (timingL0 - baseTime);  
  
 LtmpAry.put(timeMil);  
  
 countL++;  
 vibe.vibrate(75);  
  
 dispL.setText(countFormat.format(countL) + "");  
 }  
  
 break;  
 case MotionEvent.*ACTION\_UP*:  
 if (Counting) {  
 timingL = (int) (SystemClock.*elapsedRealtime*() - timingL0);  
 LtimAry.put(timingL);  
 }  
 break;  
 }  
  
 return false;  
 }  
 });  
  
 btnN.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()){  
 case MotionEvent.*ACTION\_DOWN*:  
  
 startTiming();  
 if (Counting) {  
  
 timingN0 = SystemClock.*elapsedRealtime*();  
  
 timeMil = (int) (timingN0 - baseTime);  
 NtmpAry.put(timeMil);  
  
 countN++;  
 vibe.vibrate(75);  
  
 dispN.setText(countFormat.format(countN) + "");  
 }  
  
 break;  
 case MotionEvent.*ACTION\_UP*:  
 if (Counting) {  
 timingN = (int) (SystemClock.*elapsedRealtime*() - timingN0);  
 NtimAry.put(timingN);  
 }  
 break;  
 }  
  
 return false;  
 }  
 });  
  
 btnR.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 switch (event.getAction()){  
 case MotionEvent.*ACTION\_DOWN*:  
 startTiming();  
 if (Counting) {  
  
 timingR0 = SystemClock.*elapsedRealtime*();  
  
 timeMil = (int) (timingR0 - baseTime);  
  
 RtmpAry.put(timeMil);  
  
 countR++;vibe.vibrate(75);  
  
 dispR.setText(countFormat.format(countR) + "");  
 }  
  
 break;  
 case MotionEvent.*ACTION\_UP*:  
 if (Counting) {  
 timingR = (int) (SystemClock.*elapsedRealtime*() - timingR0);  
 RtimAry.put(timingR);  
 }  
 break;  
 }  
  
 return false;  
 }  
 });  
  
 }  
  
 @Override  
 public void onBackPressed(){  
 if (Counting){  
 Toast.*makeText*(IntersecRecActivity.this, "正在统计中！", Toast.*LENGTH\_SHORT*).show();  
 }else if(!Saved){  
 releaseWakeLock();  
 AlertDialog.Builder builder = new AlertDialog.Builder(IntersecRecActivity.this);  
 builder.setTitle("是否退出？");  
 builder.setMessage("有未保存的变更，是否真的要退出？");  
 builder.setPositiveButton("是", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 IntersecRecActivity.this.finish();  
 }  
 });  
 builder.setCancelable(true);  
 builder.setNegativeButton("否", null);  
 builder.create().show();  
 } else {  
 super.onBackPressed();  
  
 }  
 }  
  
 public void startTiming(){  
 if (CDmin == 0 && CDsec ==0 && !Counting) {  
 Toast.*makeText*(this, "已经结束", Toast.*LENGTH\_SHORT*).show();  
 } else if (!Counting){  
 Counting = true;  
 Saved = false;  
 timer = new Timer();  
 handler = new Handler();  
 baseTime = SystemClock.*elapsedRealtime*();  
 LtmpAry = new JSONArray();  
 NtmpAry = new JSONArray();  
 RtmpAry = new JSONArray();  
 LtimAry = new JSONArray();  
 NtimAry = new JSONArray();  
 RtimAry = new JSONArray();  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 handler.post(new Runnable() {  
 @Override  
 public void run() {  
 if (CDmin == 0 && CDsec == 0) {  
 Counting = false;  
 vibe.vibrate(1500);  
  
 if (DirectionUsed != null) {  
 DirectionUsed = Arrays.*copyOf*(DirectionUsed, DirectionUsed.length + 1);  
 DirectionUsed[DirectionUsed.length - 1] = mDirection;  
 } else {  
 DirectionUsed = new Integer[]{mDirection};  
 }  
  
 try {  
  
 if (LtmpAry.length() > LtimAry.length()){  
 LtmpAry.remove(LtimAry.length());  
 } else if (LtmpAry.length() < LtimAry.length()){  
 LtimAry.remove(LtmpAry.length());  
 }  
 if (NtmpAry.length() != NtimAry.length()){  
 NtmpAry.remove(NtimAry.length());  
 } else if (NtmpAry.length() < NtimAry.length()){  
 NtimAry.remove(NtmpAry.length());  
 }  
 if (RtmpAry.length() != RtimAry.length()){  
 RtmpAry.remove(RtimAry.length());  
 } else if (RtmpAry.length() < RtimAry.length()){  
 RtimAry.remove(RtmpAry.length());  
 }  
  
 tmpObj = new JSONObject();  
 tmpObj.put("LTurn", countL);  
 tmpObj.put("NTurn", countN);  
 tmpObj.put("RTurn", countR);  
 tmpObj.put("Min", recMin);  
 tmpObj.put("Sec", recSec);  
 tmpObj.put("LPoints", LtmpAry);  
 tmpObj.put("LDetails",LtimAry);  
 tmpObj.put("NPoints", NtmpAry);  
 tmpObj.put("NDetails", NtimAry);  
 tmpObj.put("RPoints", RtmpAry);  
 tmpObj.put("RDetails", RtimAry);  
  
  
 outputObj.put(directionCode, tmpObj);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 recMin = 0;  
 recSec = 0;  
  
 timer.cancel();  
 } else if (CDsec == 0) {  
 CDmin--;  
 CDsec = 59;  
 textCountDownMin.setText(timeFormat.format(CDmin) + "");  
 textCountDownSec.setText(timeFormat.format(CDsec) + "");  
 } else {  
 CDsec--;  
 textCountDownSec.setText(timeFormat.format(CDsec) + "");  
 }  
  
 }  
 });  
 }  
 }, 0, 1000);  
  
 if (!Located) {  
 mLocationClient.setLocationOption(mLocationOption);  
 mLocationClient.startLocation();  
 }  
 }  
 }  
  
 public void saveFile(){  
 if (!folder.exists()) {  
 folder.mkdirs();  
 }  
 outputFile = new File(folder + "/" + fileName + ".ilog.json");  
 try {  
 FileOutputStream FOS = new FileOutputStream(outputFile);  
 FOS.write(outputObj.toString().getBytes());  
 FOS.close();  
 Toast.*makeText*(this, "已保存至：" + outputFile, Toast.*LENGTH\_SHORT*).show();  
 Saved = true;  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 public void setDir(){  
 mDirection = tmpSelection;  
 String[] tmpAry0 = getResources().getStringArray(R.array.*intersecDir*);  
 textDirection.setText(tmpAry0[mDirection]);  
 String[] tmpAry1 = getResources().getStringArray(R.array.*intersecDirS*);  
  
 switch (mDirection){  
 case 0:  
 arrowImage.setImageDrawable(getDrawable(R.drawable.*intersec\_arrow\_n*));  
 break;  
 case 1:  
 arrowImage.setImageDrawable(getDrawable(R.drawable.*intersec\_arrow\_e*));  
 break;  
 case 2:  
 arrowImage.setImageDrawable(getDrawable(R.drawable.*intersec\_arrow\_s*));  
 break;  
 case 3:  
 arrowImage.setImageDrawable(getDrawable(R.drawable.*intersec\_arrow\_w*));  
 break;  
 }  
 directionCode = tmpAry1[mDirection];  
 Saved = false;  
 tmpSelection = 0;  
  
 }  
  
  
 public void acquireWakeLock()  
 {  
 if (wakeLock == null)  
 {  
 PowerManager pm = (PowerManager)this.getSystemService(Context.*POWER\_SERVICE*);  
 wakeLock = pm.newWakeLock(PowerManager.*FULL\_WAKE\_LOCK*|PowerManager.*ON\_AFTER\_RELEASE*, "PostLocationService");  
 if (null != wakeLock)  
 {  
 wakeLock.acquire();  
 }  
 }  
 }  
  
 public void releaseWakeLock()  
 {  
 if (wakeLock != null)  
 {  
 wakeLock.release();  
 wakeLock = null;  
 }  
 }  
 @Override  
 public void onDestroy(){  
 finish();  
 super.onDestroy();  
 }  
  
}

package voss.TrafficAnalyzer;  
  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.drawable.Drawable;  
import android.os.Environment;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.BaseAdapter;  
import android.widget.Button;  
import android.widget.ImageButton;  
import android.widget.ListView;  
import android.widget.RadioButton;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Date;  
import java.util.List;  
  
public class BrowseActivity extends AppCompatActivity {  
 private Button BrowseTrueBtn;  
 private Button BrowseDeleteBtn;  
 private Button Filter0Btn;  
 private ImageButton Filter1Btn, Filter2Btn;  
 private ListView nListView;  
 private File nDirectory, videoFile;  
 private File[] nContents, mContents;  
 private Drawable nIcon;  
 private String[] filters, filterExt;  
 private String fileChosen, tempString, fileDuration, fileStart, fileEnd, IntersecName, IntersecDate;  
 private StringBuilder IntersecDirections;  
 private MyAdapter myAdapter;  
 private int selectedPosition, fileType, filter;  
 private StringBuilder stringBuilder;  
 private JSONObject jsOBJ, jsObj0;  
 private JSONArray jsARY;  
 private SimpleDateFormat formatter;  
 private boolean selected, IntersecDIRS;  
 private TextView preViewL1, preViewL2, preViewL3, preViewL4, preViewFilter;  
 private AlertDialog DelDialog;  
 private AlertDialog.Builder DelDialogB;  
 private SimpleDateFormat timeFormat;  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_browse*);  
  
 BrowseTrueBtn = (Button)findViewById(R.id.*browseConfirm*);  
 BrowseDeleteBtn = (Button)findViewById(R.id.*browseDelete*);  
 Filter0Btn = (Button)findViewById(R.id.*filter0Btn*);  
 Filter1Btn = (ImageButton)findViewById(R.id.*filter1Btn*);  
 Filter2Btn = (ImageButton)findViewById(R.id.*filter2Btn*);  
 nListView = (ListView)findViewById(R.id.*browseList*);  
 preViewL1 = (TextView)findViewById(R.id.*textL1*);  
 preViewL2 = (TextView)findViewById(R.id.*textL2*);  
 preViewL3 = (TextView)findViewById(R.id.*textL3*);  
 preViewL4 = (TextView)findViewById(R.id.*textL4*);  
 preViewFilter = (TextView)findViewById(R.id.*filterText*);  
 nDirectory = new File(Environment.*getExternalStorageDirectory*() + "/Surveyor/");  
 selectedPosition = -1;  
 formatter = new SimpleDateFormat("HH:mm:ss");  
 filters = getResources().getStringArray(R.array.*fileTypes*);  
 filterExt = getResources().getStringArray(R.array.*filterExt*);  
 timeFormat = new SimpleDateFormat("yyyy年MM月dd日\_HH时mm分ss秒");  
  
 nContents = nDirectory.listFiles();  
  
 listTheView();  
  
 BrowseTrueBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (fileChosen != null) {  
 Toast.*makeText*(BrowseActivity.this, "正在打开：" + fileChosen,  
 Toast.*LENGTH\_SHORT*).show();  
 switch (fileType) {  
 case 1:  
 Intent intent1 = new Intent();  
 intent1.setClass(BrowseActivity.this, VidReplayActivity.class);  
 intent1.putExtra("LogPath", fileChosen);  
 startActivity(intent1);  
 break;  
 case 2:  
 Intent intent2 = new Intent();  
 intent2.setClass(BrowseActivity.this, IntersecReplayActivity.class);  
 intent2.putExtra("LogPath", fileChosen);  
 startActivity(intent2);  
 break;  
 }  
 } else {  
 Toast.*makeText*(BrowseActivity.this, "未选择文件！", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 }  
 });  
  
  
 BrowseDeleteBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (selected){  
 DelDialogB = new AlertDialog.Builder(BrowseActivity.this);  
 DelDialogB.setCancelable(true);  
 DelDialogB.setTitle("是否删除？");  
 DelDialogB.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 if (videoFile != null) {  
 videoFile.delete();  
 }  
 File delFile = new File(Environment.*getExternalStorageDirectory*() + "/Surveyor/" + fileChosen);  
 delFile.delete();  
  
  
 nContents = nDirectory.listFiles();  
  
 listTheView();  
 selected = false;  
 }  
 });  
 DelDialogB.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
 });  
 DelDialog = DelDialogB.create();  
 DelDialog.show();  
 }  
 }  
 });  
  
 nListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view,  
 int position, long id) {  
 selectedPosition = position;  
 myAdapter.notifyDataSetChanged();  
 fileChosen = mContents[position].getName();  
 Toast.*makeText*(BrowseActivity.this, "已选定" + fileChosen, Toast.*LENGTH\_SHORT*).show();  
 if (fileChosen.endsWith(filterExt[1])) {  
 fileType = 1;  
 unpackVidJSON(new File(fileChosen));  
 } else if (fileChosen.endsWith(filterExt[2])){  
 fileType = 2;  
 unpackIntersecJSON(new File(fileChosen));  
 } else {  
 Toast.*makeText*(BrowseActivity.this, "不支持的JSON文件！", Toast.*LENGTH\_SHORT*).show();  
 }  
 showPreview(fileType);  
 selected = true;  
 }  
 });  
  
 Filter0Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 filter = 0;  
 listTheView();  
  
 }  
 });  
  
 Filter1Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 filter = 1;  
 listTheView();  
  
 }  
 });  
  
 Filter2Btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 filter = 2;  
 listTheView();  
  
 }  
 });  
 }  
  
 public void listTheView() {  
 List<String> listItems = new ArrayList<String>();  
 List<String> listTimes = new ArrayList<String>();  
  
 mContents = new File[0];  
  
 for (int i = 0; i < nContents.length; i++) {  
  
 if (nContents[i].getName().endsWith(filterExt[filter])) {  
 listItems.add(nContents[i].getName());  
 listTimes.add(timeFormat.format(nContents[i].lastModified()));  
 mContents = Arrays.*copyOf*(mContents, mContents.length + 1);  
 mContents[mContents.length - 1] = nContents[i];  
 }  
 }  
  
 myAdapter = new MyAdapter(this, listItems, listTimes);  
 nListView.setAdapter(myAdapter);  
  
 preViewFilter.setText("正在显示：" + filters[filter] + "记录文件");  
  
 }  
  
  
  
  
 public void unpackVidJSON(File file){  
 try {  
  
 InputStream is = new FileInputStream(Environment.*getExternalStorageDirectory*() + "/Surveyor/" +file);  
 InputStreamReader streamReader = new InputStreamReader(is);  
 BufferedReader reader = new BufferedReader(streamReader);  
 String line = null;  
 stringBuilder = new StringBuilder();  
 while ((line = reader.readLine()) != null) {  
 *// stringBuilder.append(line);* stringBuilder.append(line);  
 }  
 tempString = stringBuilder.toString();  
 is.close();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "读取失败！", Toast.*LENGTH\_SHORT*).show();  
 }  
 try {  
 jsOBJ = new JSONObject(tempString);  
 videoFile = new File(Environment.*getExternalStorageDirectory*() + jsOBJ.getString("VideoPath"));  
 jsARY = jsOBJ.getJSONArray("MainTable");  
 fileStart = jsARY.getJSONObject(0).getString("Time");  
 fileEnd = jsARY.getJSONObject(jsARY.length()-1).getString("Time");  
 } catch (JSONException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "JSON文件格式错误！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
  
 public void unpackIntersecJSON(File file){  
 try {  
  
 InputStream is = new FileInputStream(Environment.*getExternalStorageDirectory*() + "/Surveyor/" +file);  
 InputStreamReader streamReader = new InputStreamReader(is);  
 BufferedReader reader = new BufferedReader(streamReader);  
 String line = null;  
 stringBuilder = new StringBuilder();  
 while ((line = reader.readLine()) != null) {  
 *// stringBuilder.append(line);* stringBuilder.append(line);  
 }  
 tempString = stringBuilder.toString();  
 is.close();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "读取失败！", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 videoFile = null;  
  
 try {  
 jsOBJ = new JSONObject(tempString);  
 jsObj0 = new JSONObject();  
 jsObj0 = jsOBJ.getJSONObject("Info");  
 IntersecName = jsObj0.getString("Name");  
 IntersecDate = jsObj0.getString("Date");  
  
 IntersecDirections = new StringBuilder();  
 IntersecDIRS = false;  
  
 if (jsOBJ.has("N")){  
 IntersecDirections.append("北");  
 IntersecDIRS = true;  
 }  
 if (jsOBJ.has("E")){  
 if (IntersecDIRS){  
 IntersecDirections.append("，");  
 }  
 IntersecDirections.append("东");  
 IntersecDIRS = true;  
 }  
 if (jsOBJ.has("S")){  
 if (IntersecDIRS){  
 IntersecDirections.append("，");  
 }  
 IntersecDirections.append("南");  
 IntersecDIRS = true;  
 }  
 if (jsOBJ.has("W")){  
 if (IntersecDIRS){  
 IntersecDirections.append("，");  
 }  
 IntersecDirections.append("西");  
 }  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "JSON文件格式错误！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 public void showPreview(int type){  
 switch (fileType) {  
 case 1:  
 long fileDur;  
 fileDur = (jsARY.length() - 1 + 57600) \* 1000;  
 fileDuration = formatter.format(new Date(fileDur));  
  
 preViewL1.setText("长度：" + fileDuration);  
 preViewL2.setText("开始于：" + fileStart);  
 preViewL3.setText("结束于：" + fileEnd);  
  
 if (videoFile.exists()) {  
 preViewL4.setText("视频文件：有");  
 } else {  
 preViewL4.setText("视频文件：无");  
 }  
 break;  
 case 2:  
 preViewL1.setText("路口名：" + IntersecName);  
 preViewL2.setText("录制于：" + IntersecDate);  
 preViewL3.setText("包含方向：" + IntersecDirections.toString());  
  
 break;  
 }  
 }  
  
 private class MyAdapter extends BaseAdapter {  
 Context context;  
 List<String> filelist, timelist;  
 LayoutInflater mInflater;  
 MyAdapter(Context context, List<String> mList, List<String> nList){  
 this.context = context;  
 this.filelist = mList;  
 this.timelist = nList;  
 mInflater = (LayoutInflater)context.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  
 }  
 @Override  
 public int getCount() {  
 return filelist.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return position;  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
 @Override  
 public View getView(final int position, View convertView, ViewGroup parent) {  
  
 ViewHolder viewHolder = null;  
 if(convertView == null){  
 convertView = mInflater.inflate(R.layout.*browse\_item*,parent,false);  
 viewHolder = new ViewHolder();  
 viewHolder.name = (TextView)convertView.findViewById(R.id.*fileName*);  
 viewHolder.select = (RadioButton)convertView.findViewById(R.id.*radioButton*);  
 viewHolder.modified = (TextView)convertView.findViewById(R.id.*timeModified*);  
 convertView.setTag(viewHolder);  
 }else{  
 viewHolder = (ViewHolder)convertView.getTag();  
 }  
 viewHolder.name.setText(filelist.get(position));  
 viewHolder.modified.setText(timelist.get(position));  
 if(selectedPosition == position){  
 viewHolder.select.setChecked(true);  
 }  
 else{  
 viewHolder.select.setChecked(false);  
 }  
 return convertView;  
 }  
 }  
 private class ViewHolder{  
 TextView name;  
 RadioButton select;  
 TextView modified;  
 }  
  
 @Override  
 public void onDestroy(){  
 finish();  
 super.onDestroy();  
 }  
}

package voss.TrafficAnalyzer;  
  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.BitmapFactory;  
import android.os.Environment;  
import android.os.Handler;  
import android.os.PowerManager;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.ImageButton;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
import android.widget.VideoView;  
  
import com.amap.api.maps.AMap;  
import com.amap.api.maps.CameraUpdate;  
import com.amap.api.maps.CameraUpdateFactory;  
import com.amap.api.maps.TextureMapView;  
import com.amap.api.maps.model.BitmapDescriptorFactory;  
import com.amap.api.maps.model.CameraPosition;  
import com.amap.api.maps.model.LatLng;  
import com.amap.api.maps.model.Marker;  
import com.amap.api.maps.model.MarkerOptions;  
import com.amap.api.maps.model.Polyline;  
import com.amap.api.maps.utils.overlay.SmoothMoveMarker;  
import com.amap.api.trace.LBSTraceClient;  
import com.amap.api.trace.TraceListener;  
import com.amap.api.trace.TraceLocation;  
import com.amap.api.trace.TraceOverlay;  
import com.github.mikephil.charting.charts.LineChart;  
import com.github.mikephil.charting.components.XAxis;  
import com.github.mikephil.charting.data.Entry;  
import com.github.mikephil.charting.data.LineData;  
import com.github.mikephil.charting.data.LineDataSet;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class VidReplayActivity extends AppCompatActivity {  
 private File mLogFile, mVideoFile;  
 private TextureMapView RepMapView;  
 private AMap RepMap;  
 private VideoView RepVidView;  
 private Polyline RepTraceLine;  
 private Marker RepMarker, PosMarker;  
 private MarkerOptions RepMarkerOption, PosMarkerOption;  
 private ImageButton RepStartBtn;  
 *//private ImageButton RepMenuBtn;* private ImageButton RepCenterBtn;  
 private boolean Playing, Paused, FileReadiness, ChartPointed, validNote;  
 private List<LatLng> locList;  
 private JSONObject jsOBJ;  
 private JSONArray lineARY, pointsARY;  
 private LatLng markerLoc;  
 private SeekBar mSeekBar;  
 private StringBuilder stringBuilder;  
 private String tempString, spdDisp, dirDisp;  
 private int VidDur, TargetPos, PtsCount, currentPts, infoPos;  
 private Timer RepTimer, RepLocTimer;  
 private Handler RepTimerHandler, RepLocTimerHandler;  
 private LineChart RepChart;  
 private List<Entry> ChartEntries, markerEntries;  
 private LineDataSet ChartDataSet, markerDataSet;  
 private LineData ChartData;  
 private SmoothMoveMarker RepMoveMarker;  
 public CameraUpdate RepCameraUpdate;  
 public CameraUpdateFactory RepCameraUpdateFactory;  
 private TraceOverlay RecTrace;  
 private LBSTraceClient nTraceClient;  
 private List<TraceLocation> nTraceList;  
 private List<LatLng> markerlist;  
 private PowerManager.WakeLock wakeLock;  
 private TextView brnText, spdText;  
 private double rBearing, rSpeed;  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_vid\_replay*);  
  
 RepStartBtn = (ImageButton) findViewById(R.id.*repStartBtn*);  
 *//RepMenuBtn = (ImageButton) findViewById(R.id.repMenuBtn);* RepCenterBtn = (ImageButton)findViewById(R.id.*repCenterBtn*);  
 RepMapView = (TextureMapView)findViewById(R.id.*repMap*);  
 RepVidView = (VideoView) findViewById(R.id.*repVidView*);  
 mSeekBar = (SeekBar)findViewById(R.id.*seekBar*);  
 RepChart = (LineChart)findViewById(R.id.*repChart*);  
 brnText = (TextView)findViewById(R.id.*textRepBearing*);  
 spdText = (TextView)findViewById(R.id.*textRepSpeed*);  
  
 mSeekBar.setMax(1000);  
  
  
 RepMapView.onCreate(savedInstanceState);  
 if (RepMap == null) {  
 RepMap = RepMapView.getMap();  
 }  
  
 recvPath();  
 if (FileReadiness) {  
 unpackJSON(mLogFile);  
 drawMap();  
 markerTiming();  
 }  
 drawChart();  
  
  
 if (FileReadiness) {  
 RepStartBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Playing) {  
 if (Paused) {  
 Paused = false;  
 RepVidView.start();  
 *//RepVidView.resume();* } else {  
 VidDur = RepVidView.getDuration();  
 RepVidView.start();  
 initTimer();  
 }  
 Playing = true;  
  
 RepVidView.requestFocus();  
 RepStartBtn.setImageDrawable(getDrawable(android.R.drawable.*ic\_media\_pause*));  
 } else if (Playing){  
 Playing = false;  
 RepVidView.pause();  
 Paused = true;  
  
 RepStartBtn.setImageDrawable(getDrawable(android.R.drawable.*ic\_media\_play*));  
  
 }  
 }  
 });  
  
 RepCenterBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 RepMap.moveCamera(CameraUpdateFactory.*newCameraPosition*(  
 new CameraPosition(locList.get(currentPts),14,0,0)));  
 }  
 });  
 }  
 */\*RepMenuBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (!Playing){  
 Intent intent = new Intent();  
 intent.setClass(VidReplayActivity.this, BrowseActivity.class);  
 startActivity(intent);  
 } else {  
 Toast.makeText(VidReplayActivity.this, "正在播放！", Toast.LENGTH\_SHORT).show();  
 }  
  
  
 }  
 });\*/* if (FileReadiness) {  
 mSeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 TargetPos = VidDur \* progress / 1000;  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 if (mSeekBar.getProgress() != 1000) {  
 RepVidView.seekTo(TargetPos);  
 initTimer();  
 } else {  
 RepVidView.seekTo(TargetPos - 1);  
 RepVidView.pause();  
 RepStartBtn.setImageDrawable(getDrawable(android.R.drawable.*ic\_media\_pause*));  
 Playing = false;  
 }  
 }  
 });  
 }  
 acquireWakeLock();  
 }  
  
 private void initTimer() {  
 RepTimer = new Timer();  
 RepTimerHandler = new Handler();  
 RepTimer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 RepTimerHandler.post(new Runnable() {  
 @Override  
 public void run() {  
 if (Playing) {  
  
 if (mSeekBar.getProgress() < 1000) {  
 mSeekBar.setProgress(mSeekBar.getProgress() + 1);  
 } else {  
 RepVidView.pause();  
 Playing = false;  
 Toast.*makeText*(VidReplayActivity.this, "回放完毕！", Toast.*LENGTH\_SHORT*).show();  
  
 RepStartBtn.setImageDrawable(getDrawable(android.R.drawable.*ic\_media\_pause*));  
  
  
 RepVidView.seekTo(1);  
 mSeekBar.setProgress(0);  
  
 }  
  
  
 }  
 }  
 });  
  
 }  
 }, 0, VidDur / 1000);  
 }  
  
 public void recvPath(){  
 try {  
 Intent intent = getIntent();  
 mLogFile = new File(intent.getStringExtra("LogPath"));  
 } catch (Exception e) {  
 e.printStackTrace();  
 FileReadiness = false;  
 }  
  
 if (mLogFile == null){  
 Toast.*makeText*(this, "未打开文件", Toast.*LENGTH\_SHORT*).show();  
 FileReadiness = false;  
 } else {  
 Toast.*makeText*(this, "正在载入: " + mLogFile, Toast.*LENGTH\_SHORT*).show();  
 FileReadiness = true;  
 Log.*e*("file", mLogFile + "");  
 Log.*e*("isThere", mLogFile.exists() + "");  
 }  
 }  
  
 public void unpackJSON(File file){  
 try {  
  
 InputStream is = new FileInputStream(Environment.*getExternalStorageDirectory*() + "/Surveyor/" +file);  
 InputStreamReader streamReader = new InputStreamReader(is);  
 BufferedReader reader = new BufferedReader(streamReader);  
 String line = null;  
 stringBuilder = new StringBuilder();  
 while ((line = reader.readLine()) != null) {  
 *// stringBuilder.append(line);* stringBuilder.append(line);  
 }  
 tempString = stringBuilder.toString();  
 is.close();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "读取失败！", Toast.*LENGTH\_SHORT*).show();  
 }  
 try {  
 jsOBJ = new JSONObject(tempString);  
 mVideoFile = new File(jsOBJ.getString("VideoPath"));  
 lineARY = jsOBJ.getJSONArray("MainTable");  
 pointsARY = jsOBJ.getJSONArray("PointsTable");  
 } catch (JSONException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "JSON文件格式错误！", Toast.*LENGTH\_SHORT*).show();  
 }  
 RepVidView.setVideoPath(Environment.*getExternalStorageDirectory*() + mVideoFile.toString());  
 }  
  
 public void drawMap(){  
 locList = new ArrayList<LatLng>();  
  
 nTraceClient = new LBSTraceClient(this);  
 nTraceList = new ArrayList<TraceLocation>();  
 PtsCount = lineARY.length();  
 for (int i = 0; i < PtsCount; i++){  
 try {  
 JSONObject tmpPLObj = lineARY.getJSONObject(i);  
 TraceLocation tmpLocation = new TraceLocation();  
 tmpLocation.setLatitude(tmpPLObj.getDouble("Lat"));  
 tmpLocation.setLongitude(tmpPLObj.getDouble("Lon"));  
 tmpLocation.setSpeed((float)tmpPLObj.getDouble("Speed"));  
 tmpLocation.setBearing((float)tmpPLObj.getDouble("Bearing"));  
 tmpLocation.setTime(tmpPLObj.getInt("ID") \* 1000);  
 nTraceList.add(tmpLocation);  
 locList.add(new LatLng(tmpPLObj.getDouble("Lat"), tmpPLObj.getDouble("Lon")));  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 nTraceClient.queryProcessedTrace(1, nTraceList, LBSTraceClient.*TYPE\_AMAP*, new TraceListener() {  
 @Override  
 public void onRequestFailed(int i, String s) {  
 Toast.*makeText*(VidReplayActivity.this, "轨迹纠正失败，显示原始数据", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onTraceProcessing(int i, int i1, List<LatLng> list) {  
  
 }  
  
 @Override  
 public void onFinished(int i, List<LatLng> list, int i1, int i2) {  
 Toast.*makeText*(VidReplayActivity.this, "轨迹纠正成功，耗时：" + i2 +  
 "ms，显示处理后轨迹", Toast.*LENGTH\_SHORT*).show();  
 locList = list;  
 }  
 });  
  
 RecTrace = new TraceOverlay(RepMap, locList);  
  
 */\*  
 RepTraceLine = RepMap.addPolyline((new PolylineOptions())  
 .addAll(locList)  
 .width(8)  
 .color(Color.argb(255, 1, 1, 1)));  
\*/* for (int i = 0; i < pointsARY.length(); i++){  
 try {  
 validNote = true;  
 JSONObject tmpPTObj = pointsARY.getJSONObject(i);  
 markerLoc = new LatLng(tmpPTObj.getDouble("Lat"), tmpPTObj.getDouble("Lon"));  
 int Note = tmpPTObj.getInt("Type");  
 RepMarkerOption = new MarkerOptions().position(markerLoc).draggable(false);  
 switch (Note){  
 case 0:  
 validNote = false;  
 break;  
 case 10:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s10*)));  
  
 break;  
 case 11:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s11*)));  
  
 break;  
 case 12:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s12*)));  
  
 break;  
 case 13:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s13*)));  
  
 break;  
 case 14:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s14*)));  
  
 break;  
 case 15:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s15*)));  
  
 break;  
 case 20:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s20*)));  
  
 break;  
 case 21:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s21*)));  
  
 break;  
 case 22:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s22*)));  
  
 break;  
 case 23:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s23*)));  
  
 break;  
 case 24:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s24*)));  
  
 break;  
 case 25:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s25*)));  
  
 break;  
 case 30:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s30*)));  
  
 break;  
 case 31:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s31*)));  
  
 break;  
 case 32:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s32*)));  
  
 break;  
 case 33:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s33*)));  
  
 break;  
 case 34:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s34*)));  
  
 break;  
 case 35:  
 RepMarkerOption.icon(BitmapDescriptorFactory.*fromBitmap*(BitmapFactory  
 .*decodeResource*(getResources(),R.drawable.*s35*)));  
  
 break;  
 }  
 if (validNote) {  
 RepMarker = RepMap.addMarker(RepMarkerOption);  
 }  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
  
 RepCameraUpdateFactory = new CameraUpdateFactory();  
  
 RepCameraUpdate =RepCameraUpdateFactory.  
 *newCameraPosition*(new CameraPosition(locList.get(0),15,0,0));  
 RepMap.animateCamera(RepCameraUpdate);  
  
 RecTrace.zoopToSpan();  
 }  
  
 public void markerTiming(){  
 RepLocTimer = new Timer();  
 RepLocTimerHandler = new Handler();  
 RepLocTimer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 RepLocTimerHandler.post(new Runnable() {  
 @Override  
 public void run() {  
 if (Playing) {  
 if (currentPts != locList.size() \* mSeekBar.getProgress() / 1000) {  
 currentPts = locList.size() \* mSeekBar.getProgress() / 1000;  
  
  
 if (PosMarker != null) {  
 PosMarker.remove();  
 }  
 PosMarkerOption = new MarkerOptions().draggable(false).position(locList.get(currentPts));  
 PosMarker = RepMap.addMarker(PosMarkerOption);  
 }  
 dispInfo();  
 drawChartMarker();  
 }  
 }  
 });  
 }  
 }, 0, 1000);  
 }  
  
 public void dispInfo(){  
 infoPos = lineARY.length() \* mSeekBar.getProgress() / 1000 - 1;  
 JSONObject infoObj;  
 try {  
 infoObj = lineARY.getJSONObject(infoPos);  
 rBearing = infoObj.getDouble("Bearing");  
 rSpeed = infoObj.getDouble("Speed") \* 3.6;  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 DecimalFormat decimalFormat=new DecimalFormat("000.0");  
 spdDisp = "速度：" + decimalFormat.format(rSpeed) + "km/h";  
  
 dirDisp = "方向：";  
 if (rBearing == -1){  
 dirDisp += "未确定方向";  
 } else if (rBearing <= 22.5 || rBearing > 337.5) {  
 dirDisp += "正北";  
 } else if (rBearing > 22.5 && rBearing <= 67.5) {  
 dirDisp += "东北";  
 } else if (rBearing > 67.5 && rBearing <= 112.5){  
 dirDisp += "正东";  
 } else if (rBearing > 112.5 && rBearing <= 157.5){  
 dirDisp += "东南";  
 } else if (rBearing > 157.5 && rBearing <= 202.5){  
 dirDisp += "正南";  
 } else if (rBearing > 202.5 && rBearing <= 247.5){  
 dirDisp += "西南";  
 } else if (rBearing > 247.5 && rBearing <= 292.5){  
 dirDisp += "正西";  
 } else if (rBearing > 292.5 && rBearing <= 337.5){  
 dirDisp += "西北";  
 }  
  
 spdText.setText(spdDisp);  
 brnText.setText(dirDisp);  
 }  
  
 @Override  
 public void onDestroy() {  
 RepMap.clear();  
 if (RepTraceLine != null) {  
 RepTraceLine.remove();  
 }  
 RepMapView.onDestroy();  
 releaseWakeLock();  
 finish();  
 super.onDestroy();  
 }  
 @Override  
 public void onPause() {  
 super.onPause();  
 RepMapView.onPause();  
 releaseWakeLock();  
 }  
 @Override  
 public void onResume() {  
 super.onResume();  
 RepMapView.onResume();  
 acquireWakeLock();  
 }  
  
  
 public void drawChart(){  
 ChartEntries = new ArrayList<Entry>();  
  
 if (FileReadiness) {  
 for (int i = 0; i < PtsCount; i++) {  
 try {  
 JSONObject tmpChtObj = lineARY.getJSONObject(i);  
 Entry tmpEntry = new Entry(tmpChtObj.getInt("ID"), (float) tmpChtObj.getDouble("Distance"));  
 ChartEntries.add(tmpEntry);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 } else {  
 ChartEntries.add(new Entry(0,0));  
 }  
 ChartDataSet = new LineDataSet(ChartEntries,"距离");  
 ChartDataSet.setDrawCircles(false);  
 ChartDataSet.setLineWidth(5);  
 ChartData = new LineData(ChartDataSet);  
 RepChart.setData(ChartData);  
 RepChart.setDragDecelerationEnabled(false);  
 RepChart.getXAxis().setPosition(XAxis.XAxisPosition.*BOTTOM*);  
 RepChart.setBackgroundColor(getResources().getColor(R.color.*leaf*));  
 RepChart.getAxisLeft().setAxisMinimum(0);  
 RepChart.invalidate();  
  
 }  
  
 public void drawChartMarker(){  
 JSONObject tempObj;  
  
 if (ChartPointed) {  
 markerDataSet.removeLast();  
 ChartPointed = false;  
 }  
  
 markerEntries = new ArrayList<Entry>();  
 try {  
 tempObj = lineARY.getJSONObject(infoPos);  
 markerEntries.add(new Entry(tempObj.getInt("ID"), (float) tempObj.getDouble("Distance")));  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 markerDataSet = new LineDataSet(markerEntries,"");  
 markerDataSet.setCircleRadius(9);  
 markerDataSet.setCircleColor(R.color.*black*);  
 ChartDataSet.setDrawCircles(false);  
  
 ChartData.addDataSet(markerDataSet);  
 ChartPointed = true;  
 markerDataSet.notifyDataSetChanged();  
 ChartData.notifyDataChanged();  
 RepChart.notifyDataSetChanged();  
 RepChart.invalidate();  
  
 }  
  
 @Override  
 public void onBackPressed(){  
 if (Playing) {  
 Toast.*makeText*(this, "正在回放！", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 finish();  
 super.onBackPressed();  
 }  
 }  
  
  
 public void acquireWakeLock()  
 {  
 if (null == wakeLock)  
 {  
 PowerManager pm = (PowerManager)this.getSystemService(Context.*POWER\_SERVICE*);  
 wakeLock = pm.newWakeLock(PowerManager.*FULL\_WAKE\_LOCK*|PowerManager.*ON\_AFTER\_RELEASE*, "PostLocationService");  
 if (null != wakeLock)  
 {  
 wakeLock.acquire();  
 }  
 }  
 }  
  
 *//释放设备电源锁* public void releaseWakeLock()  
 {  
 if (null != wakeLock)  
 {  
 wakeLock.release();  
 wakeLock = null;  
 }  
 }  
  
}

package voss.TrafficAnalyzer;  
  
import android.content.Intent;  
import android.os.Environment;  
import android.support.annotation.NonNull;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.LinearLayout;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.amap.api.maps.AMap;  
import com.amap.api.maps.CameraUpdate;  
import com.amap.api.maps.CameraUpdateFactory;  
import com.amap.api.maps.TextureMapView;  
import com.amap.api.maps.model.CameraPosition;  
import com.amap.api.maps.model.LatLng;  
import com.amap.api.maps.model.Marker;  
import com.amap.api.maps.model.MarkerOptions;  
import com.github.mikephil.charting.charts.LineChart;  
import com.github.mikephil.charting.data.Entry;  
import com.github.mikephil.charting.data.LineData;  
import com.github.mikephil.charting.data.LineDataSet;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.Iterator;  
import java.util.List;  
import java.util.ListIterator;  
  
public class IntersecReplayActivity extends AppCompatActivity {  
  
 private File mLogFile;  
 private boolean FileReadiness, hasN, hasE, hasS, hasW;  
 private LinearLayout tabIllu, tabDiag, tabTabl, frameIllu, frameDiag, frameTabl;  
 private StringBuilder stringBuilder;  
 private JSONObject jsOBJ, nObj, eObj, sObj, wObj, iObj;  
 private int iNl, iNn, iNr, iEl, iEn, iEr, iSl, iSn, iSr, iWl, iWn, iWr,  
 oN, oE, oS, oW, iN, iE, iS, iW, timeN, timeE, timeS, timeW;  
 private JSONArray aNL0, aNL1, aNN0, aNN1, aNR0, aNR1, aEL0, aEL1, aEN0, aEN1, aER0, aER1,  
 aSL0, aSL1, aSN0, aSN1, aSR0, aSR1, aWL0, aWL1, aWN0, aWN1, aWR0, aWR1;  
 private TextView txtIN, txtINR, txtINN, txtINL, txtOW, txtIW, txtIWL, txtIWN, txtIWR,  
 txtOS, txtON, txtIER, txtIEN, txtIEL, txtIE, txtOE, txtISL, txtISN, txtISR,  
 txtIS, txtIntName, txtIntDate, tablINR, tablINN, tablINL, tablIN, tablON,  
 tablIER, tablIEN, tablIEL, tablIE, tablOE, tablISR, tablISN, tablISL, tablIS,  
 tablOS, tablIWR, tablIWN, tablIWL, tablIW, tablOW;  
 private TextureMapView intMapView;  
 private AMap intMap;  
 private LatLng coord;  
 private String name, date;  
 private List<Entry> lNL, lNN, lNR, lEL, lEN, lER, lSL, lSN, lSR, lWL, lWN, lWR;  
 private LineDataSet sNL, sNN, sNR, sEL, sEN, sER, sSL, sSN, sSR, sWL, sWN, sWR;  
 private Entry tmpEntry0, tmpEntry1, tmpEntry2, tmpEntry3;  
 private LineChart intersecChart;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_intersec\_replay*);  
  
 tabIllu = (LinearLayout)findViewById(R.id.*tabIllu*);  
 tabTabl = (LinearLayout)findViewById(R.id.*tabTabl*);  
 tabDiag = (LinearLayout)findViewById(R.id.*tabDiag*);  
 frameIllu = (LinearLayout)findViewById(R.id.*frameIllu*);  
 frameTabl = (LinearLayout)findViewById(R.id.*frameTabl*);  
 frameDiag = (LinearLayout)findViewById(R.id.*frameDiag*);  
 txtIN = (TextView)findViewById(R.id.*txtIN*);  
 txtINR = (TextView)findViewById(R.id.*txtINR*);  
 txtINN = (TextView)findViewById(R.id.*txtINN*);  
 txtINL = (TextView)findViewById(R.id.*txtINL*);  
 txtOW = (TextView)findViewById(R.id.*txtOW*);  
 txtIW = (TextView)findViewById(R.id.*txtIW*);  
 txtIWL = (TextView)findViewById(R.id.*txtIWL*);  
 txtIWN = (TextView)findViewById(R.id.*txtIWN*);  
 txtIWR = (TextView)findViewById(R.id.*txtIWR*);  
 txtOS = (TextView)findViewById(R.id.*txtOS*);  
 txtON = (TextView)findViewById(R.id.*txtON*);  
 txtIER = (TextView)findViewById(R.id.*txtIER*);  
 txtIEN = (TextView)findViewById(R.id.*txtIEN*);  
 txtIEL = (TextView)findViewById(R.id.*txtIEL*);  
 txtIE = (TextView)findViewById(R.id.*txtIE*);  
 txtOE = (TextView)findViewById(R.id.*txtOE*);  
 txtISL = (TextView)findViewById(R.id.*txtISL*);  
 txtISN = (TextView)findViewById(R.id.*txtISN*);  
 txtISR = (TextView)findViewById(R.id.*txtISR*);  
 txtIS = (TextView)findViewById(R.id.*txtIS*);  
 txtIntName = (TextView)findViewById(R.id.*textIntName*);  
 txtIntDate = (TextView)findViewById(R.id.*textIntDate*);  
 tablINR = (TextView)findViewById(R.id.*tablINR*);  
 tablINN = (TextView)findViewById(R.id.*tablINN*);  
 tablINL = (TextView)findViewById(R.id.*tablINL*);  
 tablIN = (TextView)findViewById(R.id.*tablIN*);  
 tablON = (TextView)findViewById(R.id.*tablON*);  
 tablIER = (TextView)findViewById(R.id.*tablIER*);  
 tablIEN = (TextView)findViewById(R.id.*tablIEN*);  
 tablIEL = (TextView)findViewById(R.id.*tablIEL*);  
 tablIE = (TextView)findViewById(R.id.*tablIE*);  
 tablOE = (TextView)findViewById(R.id.*tablOE*);  
 tablISR = (TextView)findViewById(R.id.*tablISR*);  
 tablISN = (TextView)findViewById(R.id.*tablISN*);  
 tablISL = (TextView)findViewById(R.id.*tablISL*);  
 tablIS = (TextView)findViewById(R.id.*tablIS*);  
 tablOS = (TextView)findViewById(R.id.*tablOS*);  
 tablIWR = (TextView)findViewById(R.id.*tablIWR*);  
 tablIWN = (TextView)findViewById(R.id.*tablIWN*);  
 tablIWL = (TextView)findViewById(R.id.*tablIWL*);  
 tablIW = (TextView)findViewById(R.id.*tablIW*);  
 tablOW = (TextView)findViewById(R.id.*tablOW*);  
 intMapView = (TextureMapView)findViewById(R.id.*intersecMap*);  
 intersecChart = (LineChart)findViewById(R.id.*intersecChart*);  
  
 intMapView.onCreate(savedInstanceState);  
 if (intMap == null) {  
 intMap = intMapView.getMap();  
 }  
  
  
 recvPath();  
 *//if (FileReadiness){* unpackJSON(mLogFile);  
 procData();  
 setDisp();  
 *//}* tabIllu.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 tabIllu.setBackgroundColor(getResources().getColor(R.color.*leaf*));  
 tabTabl.setBackgroundColor(getResources().getColor(R.color.*darkLeaf*));  
 tabDiag.setBackgroundColor(getResources().getColor(R.color.*darkLeaf*));  
 frameIllu.setVisibility(View.*VISIBLE*);  
 frameDiag.setVisibility(View.*GONE*);  
 frameTabl.setVisibility(View.*GONE*);  
  
 }  
 });  
  
 tabDiag.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 tabDiag.setBackgroundColor(getResources().getColor(R.color.*leaf*));  
 tabTabl.setBackgroundColor(getResources().getColor(R.color.*darkLeaf*));  
 tabIllu.setBackgroundColor(getResources().getColor(R.color.*darkLeaf*));  
 frameDiag.setVisibility(View.*VISIBLE*);  
 frameIllu.setVisibility(View.*GONE*);  
 frameTabl.setVisibility(View.*GONE*);  
 }  
 });  
  
 tabTabl.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 tabTabl.setBackgroundColor(getResources().getColor(R.color.*leaf*));  
 tabIllu.setBackgroundColor(getResources().getColor(R.color.*darkLeaf*));  
 tabDiag.setBackgroundColor(getResources().getColor(R.color.*darkLeaf*));  
 frameTabl.setVisibility(View.*VISIBLE*);  
 frameDiag.setVisibility(View.*GONE*);  
 frameIllu.setVisibility(View.*GONE*);  
 }  
 });  
 }  
  
  
 public void recvPath(){  
 try {  
 Intent intent = getIntent();  
 mLogFile = new File(intent.getStringExtra("LogPath"));  
 } catch (Exception e) {  
 e.printStackTrace();  
 FileReadiness = false;  
 }  
  
 if (mLogFile == null){  
 Toast.*makeText*(this, "未打开文件", Toast.*LENGTH\_SHORT*).show();  
 FileReadiness = false;  
 } else {  
 Toast.*makeText*(this, "正在载入: " + mLogFile, Toast.*LENGTH\_SHORT*).show();  
 FileReadiness = true;  
 Log.*e*("file", mLogFile + "");  
 Log.*e*("isThere", mLogFile.exists() + "");  
 }  
 }  
  
  
 public void unpackJSON(File file){  
 try {  
  
 InputStream is = new FileInputStream(Environment.*getExternalStorageDirectory*() + "/Surveyor/" +file);  
 InputStreamReader streamReader = new InputStreamReader(is);  
 BufferedReader reader = new BufferedReader(streamReader);  
 String line = null;  
 stringBuilder = new StringBuilder();  
 while ((line = reader.readLine()) != null) {  
 *// stringBuilder.append(line);* stringBuilder.append(line);  
 }  
 is.close();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "读取失败！", Toast.*LENGTH\_SHORT*).show();  
 }  
 try {  
 jsOBJ = new JSONObject(stringBuilder.toString());  
 iObj = jsOBJ.getJSONObject("Info");  
 if (jsOBJ.has("N")){  
 nObj = jsOBJ.getJSONObject("N");  
 hasN = true;  
 } else {  
 hasN = false;  
 }  
 if (jsOBJ.has("E")){  
 eObj = jsOBJ.getJSONObject("E");  
 hasE = true;  
 } else {  
 hasE = false;  
 }  
 if (jsOBJ.has("S")){  
 sObj = jsOBJ.getJSONObject("S");  
 hasS = true;  
 } else {  
 hasS = false;  
 }  
 if (jsOBJ.has("W")){  
 wObj = jsOBJ.getJSONObject("W");  
 hasW = true;  
 } else {  
 hasW = false;  
 }  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 Toast.*makeText*(this, "JSON文件格式错误！", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 public void procData(){  
 try {  
 name = iObj.getString("Name");  
 date = iObj.getString("Date");  
 coord = new LatLng(iObj.getDouble("Lat"), iObj.getDouble("Lon"));  
  
 if (hasN) {  
 timeN = nObj.getInt("Min") \* 60 + nObj.getInt("Sec");  
 iNl = (int) ((double) nObj.getInt("LTurn") / (double) timeN \* 3600);  
 iNn = (int) ((double) nObj.getInt("NTurn") / (double) timeN \* 3600);  
 iNr = (int) ((double) nObj.getInt("RTurn") / (double) timeN \* 3600);  
 iN = iNl + iNn + iNr;  
  
 aNL0 = nObj.getJSONArray("LPoints");  
 aNL1 = nObj.getJSONArray("LDetails");  
 aNN0 = nObj.getJSONArray("NPoints");  
 aNN1 = nObj.getJSONArray("NDetails");  
 aNR0 = nObj.getJSONArray("RPoints");  
 aNR1 = nObj.getJSONArray("RDetails");  
 }  
 if (hasE) {  
 timeE = eObj.getInt("Min") \* 60 + eObj.getInt("Sec");  
 iEl = (int) ((double) eObj.getInt("LTurn") / (double) timeE \* 3600);  
 iEn = (int) ((double) eObj.getInt("NTurn") / (double) timeE \* 3600);  
 iEr = (int) ((double) eObj.getInt("RTurn") / (double) timeE \* 3600);  
 iE = iEl + iEn + iEr;  
  
 aEL0 = eObj.getJSONArray("LPoints");  
 aEL1 = eObj.getJSONArray("LDetails");  
 aEN0 = eObj.getJSONArray("NPoints");  
 aEN1 = eObj.getJSONArray("NDetails");  
 aER0 = eObj.getJSONArray("RPoints");  
 aER1 = eObj.getJSONArray("RDetails");  
 }  
 if (hasS) {  
 timeS = sObj.getInt("Min") \* 60 + sObj.getInt("Sec");  
 iSl = (int) ((double) sObj.getInt("LTurn") / (double) timeS \* 3600);  
 iSn = (int) ((double) sObj.getInt("NTurn") / (double) timeS \* 3600);  
 iSr = (int) ((double) sObj.getInt("RTurn") / (double) timeS \* 3600);  
 iS = iSl + iSn + iSr;  
  
 aSL0 = sObj.getJSONArray("LPoints");  
 aSL1 = sObj.getJSONArray("LDetails");  
 aSN0 = sObj.getJSONArray("NPoints");  
 aSN1 = sObj.getJSONArray("NDetails");  
 aSR0 = sObj.getJSONArray("RPoints");  
 aSR1 = sObj.getJSONArray("RDetails");  
 }  
 if (hasW) {  
 timeW = wObj.getInt("Min") \* 60 + wObj.getInt("Sec");  
 iWl = (int) ((double) wObj.getInt("LTurn") / (double) timeW \* 3600);  
 iWn = (int) ((double) wObj.getInt("NTurn") / (double) timeW \* 3600);  
 iWr = (int) ((double) wObj.getInt("RTurn") / (double) timeW \* 3600);  
 iW = iWl + iWn + iWr;  
  
 aWL0 = wObj.getJSONArray("LPoints");  
 aWL1 = wObj.getJSONArray("LDetails");  
 aWN0 = wObj.getJSONArray("NPoints");  
 aWN1 = wObj.getJSONArray("NDetails");  
 aWR0 = wObj.getJSONArray("RPoints");  
 aWR1 = wObj.getJSONArray("RDetails");  
 }  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 oN = iSn + iEr + iWl;  
 oE = iWn + iSr + iNl;  
 oS = iNn + iWr + iEl;  
 oW = iEn + iNr + iSl;  
  
 }  
  
 public void setDisp(){  
 txtIN.setText(iN + "");  
 txtINL.setText(iNl + "");  
 txtINN.setText(iNn + "");  
 txtINR.setText(iNr + "");  
 txtOW.setText(oW + "");  
 txtIW.setText(iW + "");  
 txtIWL.setText(iWl + "");  
 txtIWN.setText(iWn + "");  
 txtIWR.setText(iWr + "");  
 txtOS.setText(oS + "");  
 txtON.setText(oN + "");  
 txtIER.setText(iEr + "");  
 txtIEN.setText(iEn + "");  
 txtIEL.setText(iEl + "");  
 txtIE.setText(iE + "");  
 txtOE.setText(oE + "");  
 txtISL.setText(iSl + "");  
 txtISN.setText(iSn + "");  
 txtISR.setText(iSr + "");  
 txtIS.setText(iS + "");  
  
 tablINL.setText(iNl + "");  
 tablINN.setText(iNn + "");  
 tablINR.setText(iNr + "");  
 tablIN.setText(iN + "");  
 tablON.setText(oN + "");  
 tablIEL.setText(iEl + "");  
 tablIEN.setText(iEn + "");  
 tablIER.setText(iEr + "");  
 tablIE.setText(iE + "");  
 tablOE.setText(oE + "");  
 tablISL.setText(iSl + "");  
 tablISN.setText(iSn + "");  
 tablISR.setText(iSr + "");  
 tablIS.setText(iS + "");  
 tablOS.setText(oS + "");  
 tablIWL.setText(iWl + "");  
 tablIWN.setText(iWn + "");  
 tablIWR.setText(iWr + "");  
 tablIW.setText(iW + "");  
 tablOW.setText(oW + "");  
  
 txtIntDate.setText(date);  
 txtIntName.setText(name);  
  
 MarkerOptions markerOption = new MarkerOptions();  
 markerOption.position(coord);  
 markerOption.draggable(false);  
 Marker marker = intMap.addMarker(markerOption);  
 CameraUpdateFactory cameraUpdateFactory = new CameraUpdateFactory();  
 CameraUpdate cameraUpdate = cameraUpdateFactory.*newCameraPosition*(new CameraPosition(coord,15,0,0));  
 intMap.animateCamera(cameraUpdate);  
  
  
 LineData lineData = new LineData();  
  
 if (hasN){  
 tmpEntry0 = new Entry(0,1);  
 lNL = new ArrayList<>();  
 lNL.add(tmpEntry0);  
 for (int i = 0; i < aNL0.length(); i++){  
 try {  
 int tmpi = aNL0.getInt(i)+aNL1.getInt(i);  
 tmpEntry0 = new Entry(aNL0.getInt(i), 1);  
 tmpEntry1 = new Entry(aNL0.getInt(i), (float)1.5);  
 tmpEntry2 = new Entry(tmpi, (float)1.5);  
 tmpEntry3 = new Entry(tmpi, 1);  
 lNL.add(tmpEntry0);  
 lNL.add(tmpEntry1);  
 lNL.add(tmpEntry2);  
 lNL.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeN\*1000, 1);  
 lNL.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,2);  
 lNN = new ArrayList<>();  
 lNN.add(tmpEntry0);  
 for (int i = 0; i < aNN0.length(); i++){  
 try {  
 int tmpi = aNN0.getInt(i)+aNN1.getInt(i);  
 tmpEntry0 = new Entry(aNN0.getInt(i), 2);  
 tmpEntry1 = new Entry(aNN0.getInt(i), (float)2.5);  
 tmpEntry2 = new Entry(tmpi, (float)2.5);  
 tmpEntry3 = new Entry(tmpi, 2);  
 lNN.add(tmpEntry0);  
 lNN.add(tmpEntry1);  
 lNN.add(tmpEntry2);  
 lNN.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeN\*1000, 2);  
 lNN.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,3);  
 lNR = new ArrayList<>();  
 lNR.add(tmpEntry0);  
 for (int i = 0; i < aNR0.length(); i++){  
 try {  
 int tmpi = aNR0.getInt(i)+aNR1.getInt(i);  
 tmpEntry0 = new Entry(aNR0.getInt(i), 3);  
 tmpEntry1 = new Entry(aNR0.getInt(i), (float)3.5);  
 tmpEntry2 = new Entry(tmpi, (float)3.5);  
 tmpEntry3 = new Entry(tmpi, 3);  
 lNR.add(tmpEntry0);  
 lNR.add(tmpEntry1);  
 lNR.add(tmpEntry2);  
 lNR.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeN\*1000, 3);  
 lNR.add(tmpEntry0);  
  
 sNL = new LineDataSet(lNL, "北左");  
 sNL.setDrawCircles(false);  
 sNL.setColor(getColor(R.color.*cirredmag*));  
 sNL.setDrawValues(false);  
 sNN = new LineDataSet(lNN, "北直");  
 sNN.setDrawCircles(false);  
 sNN.setColor(getColor(R.color.*cirred*));  
 sNN.setDrawValues(false);  
 sNR = new LineDataSet(lNR, "北右");  
 sNR.setDrawCircles(false);  
 sNR.setColor(getColor(R.color.*cirredorange*));  
 sNR.setDrawValues(false);  
 lineData.addDataSet(sNL);  
 lineData.addDataSet(sNN);  
 lineData.addDataSet(sNR);  
 }  
  
 if (hasE){  
 tmpEntry0 = new Entry(0,5);  
 lEL = new ArrayList<>();  
 lEL.add(tmpEntry0);  
 for (int i = 0; i < aEL0.length(); i++){  
 try {  
 int tmpi = aEL0.getInt(i)+aEL1.getInt(i);  
 tmpEntry0 = new Entry(aEL0.getInt(i), 5);  
 tmpEntry1 = new Entry(aEL0.getInt(i), (float)5.5);  
 tmpEntry2 = new Entry(tmpi, (float)5.5);  
 tmpEntry3 = new Entry(tmpi, 5);  
 lEL.add(tmpEntry0);  
 lEL.add(tmpEntry1);  
 lEL.add(tmpEntry2);  
 lEL.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeE\*1000, 5);  
 lEL.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,6);  
 lEN = new ArrayList<>();  
 lEN.add(tmpEntry0);  
 for (int i = 0; i < aEN0.length(); i++){  
 try {  
 int tmpi = aEN0.getInt(i)+aEN1.getInt(i);  
 tmpEntry0 = new Entry(aEN0.getInt(i), 6);  
 tmpEntry1 = new Entry(aEN0.getInt(i), (float)6.5);  
 tmpEntry2 = new Entry(tmpi, (float)6.5);  
 tmpEntry3 = new Entry(tmpi, 6);  
 lEN.add(tmpEntry0);  
 lEN.add(tmpEntry1);  
 lEN.add(tmpEntry2);  
 lEN.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeE\*1000, 6);  
 lEN.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,7);  
 lER = new ArrayList<>();  
 lER.add(tmpEntry0);  
 for (int i = 0; i < aER0.length(); i++){  
 try {  
 int tmpi = aER0.getInt(i)+aER1.getInt(i);  
 tmpEntry0 = new Entry(aER0.getInt(i), 7);  
 tmpEntry1 = new Entry(aER0.getInt(i), (float)7.5);  
 tmpEntry2 = new Entry(tmpi, (float)7.5);  
 tmpEntry3 = new Entry(tmpi, 7);  
 lER.add(tmpEntry0);  
 lER.add(tmpEntry1);  
 lER.add(tmpEntry2);  
 lER.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeE\*1000, 7);  
 lER.add(tmpEntry0);  
  
  
 sEL = new LineDataSet(lEL, "东左");  
 sEL.setDrawCircles(false);  
 sEL.setColor(getColor(R.color.*cirgreenyellow*));  
 sEL.setDrawValues(false);  
 sEN = new LineDataSet(lEN, "东直");  
 sEN.setDrawCircles(false);  
 sEN.setColor(getColor(R.color.*cirgreen*));  
 sEN.setDrawValues(false);  
 sER = new LineDataSet(lER, "东右");  
 sER.setDrawCircles(false);  
 sER.setColor(getColor(R.color.*cirgreendark*));  
 sER.setDrawValues(false);  
 lineData.addDataSet(sEL);  
 lineData.addDataSet(sEN);  
 lineData.addDataSet(sER);  
 }  
  
 if (hasS){  
 tmpEntry0 = new Entry(0,9);  
 lSL = new ArrayList<>();  
 lSL.add(tmpEntry0);  
 for (int i = 0; i < aSL0.length(); i++){  
 try {  
 int tmpi = aSL0.getInt(i)+aSL1.getInt(i);  
 tmpEntry0 = new Entry(aSL0.getInt(i), 9);  
 tmpEntry1 = new Entry(aSL0.getInt(i), (float)9.5);  
 tmpEntry2 = new Entry(tmpi, (float)9.5);  
 tmpEntry3 = new Entry(tmpi, 9);  
 lSL.add(tmpEntry0);  
 lSL.add(tmpEntry1);  
 lSL.add(tmpEntry2);  
 lSL.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeS\*1000, 9);  
 lSL.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,10);  
 lSN = new ArrayList<>();  
 lSN.add(tmpEntry0);  
 for (int i = 0; i < aSN0.length(); i++){  
 try {  
 int tmpi = aSN0.getInt(i)+aSN1.getInt(i);  
 tmpEntry0 = new Entry(aSN0.getInt(i), 10);  
 tmpEntry1 = new Entry(aSN0.getInt(i), (float)10.5);  
 tmpEntry2 = new Entry(tmpi, (float)10.5);  
 tmpEntry3 = new Entry(tmpi, 10);  
 lSN.add(tmpEntry0);  
 lSN.add(tmpEntry1);  
 lSN.add(tmpEntry2);  
 lSN.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeS\*1000, 10);  
 lSN.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,11);  
 lSR = new ArrayList<>();  
 lSR.add(tmpEntry0);  
 for (int i = 0; i < aSR0.length(); i++){  
 try {  
 int tmpi = aSR0.getInt(i)+aSR1.getInt(i);  
 tmpEntry0 = new Entry(aSR0.getInt(i), 11);  
 tmpEntry1 = new Entry(aSR0.getInt(i), (float)11.5);  
 tmpEntry2 = new Entry(tmpi, (float)11.5);  
 tmpEntry3 = new Entry(tmpi, 11);  
 lSR.add(tmpEntry0);  
 lSR.add(tmpEntry1);  
 lSR.add(tmpEntry2);  
 lSR.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeS\*1000, 11);  
 lSR.add(tmpEntry0);  
  
  
 sSL = new LineDataSet(lSL, "南左");  
 sSL.setDrawCircles(false);  
 sSL.setColor(getColor(R.color.*cirbluegreen*));  
 sSL.setDrawValues(false);  
 sSN = new LineDataSet(lSN, "南直");  
 sSN.setDrawCircles(false);  
 sSN.setColor(getColor(R.color.*cirblue*));  
 sSN.setDrawValues(false);  
 sSR = new LineDataSet(lSR, "南右");  
 sSR.setDrawCircles(false);  
 sSR.setColor(getColor(R.color.*cirbluedark*));  
 sSR.setDrawValues(false);  
 lineData.addDataSet(sSL);  
 lineData.addDataSet(sSN);  
 lineData.addDataSet(sSR);  
 }  
  
 if (hasW){  
 tmpEntry0 = new Entry(0,13);  
 lWL = new ArrayList<>();  
 lWL.add(tmpEntry0);  
 for (int i = 0; i < aWL0.length(); i++){  
 try {  
 int tmpi = aWL0.getInt(i)+aWL1.getInt(i);  
 tmpEntry0 = new Entry(aWL0.getInt(i), 13);  
 tmpEntry1 = new Entry(aWL0.getInt(i), (float)13.5);  
 tmpEntry2 = new Entry(tmpi, (float)13.5);  
 tmpEntry3 = new Entry(tmpi, 13);  
 lWL.add(tmpEntry0);  
 lWL.add(tmpEntry1);  
 lWL.add(tmpEntry2);  
 lWL.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeW\*1000, 13);  
 lWL.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,14);  
 lWN = new ArrayList<>();  
 lWN.add(tmpEntry0);  
 for (int i = 0; i < aWN0.length(); i++){  
 try {  
 int tmpi = aWN0.getInt(i)+aWN1.getInt(i);  
 tmpEntry0 = new Entry(aWN0.getInt(i), 14);  
 tmpEntry1 = new Entry(aWN0.getInt(i), (float)14.5);  
 tmpEntry2 = new Entry(tmpi, (float)14.5);  
 tmpEntry3 = new Entry(tmpi, 14);  
 lWN.add(tmpEntry0);  
 lWN.add(tmpEntry1);  
 lWN.add(tmpEntry2);  
 lWN.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeW\*1000, 14);  
 lWN.add(tmpEntry0);  
  
 tmpEntry0 = new Entry(0,15);  
 lWR = new ArrayList<>();  
 lWR.add(tmpEntry0);  
 for (int i = 0; i < aWR0.length(); i++){  
 try {  
 int tmpi = aWR0.getInt(i)+aWR1.getInt(i);  
 tmpEntry0 = new Entry(aWR0.getInt(i), 15);  
 tmpEntry1 = new Entry(aWR0.getInt(i), (float)15.5);  
 tmpEntry2 = new Entry(tmpi, (float)15.5);  
 tmpEntry3 = new Entry(tmpi, 15);  
 lWR.add(tmpEntry0);  
 lWR.add(tmpEntry1);  
 lWR.add(tmpEntry2);  
 lWR.add(tmpEntry3);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 tmpEntry0 = new Entry(timeW\*1000, 15);  
 lWR.add(tmpEntry0);  
  
 sWL = new LineDataSet(lWL, "西左");  
 sWL.setDrawCircles(false);  
 sWL.setColor(getColor(R.color.*cirpurpleblue*));  
 sWL.setDrawValues(false);  
 sWN = new LineDataSet(lWN, "西直");  
 sWN.setDrawCircles(false);  
 sWN.setColor(getColor(R.color.*cirpurple*));  
 sWN.setDrawValues(false);  
 sWR = new LineDataSet(lWR, "西右");  
 sWR.setDrawCircles(false);  
 sWR.setColor(getColor(R.color.*cirpurplered*));  
 sWR.setDrawValues(false);  
 lineData.addDataSet(sWL);  
 lineData.addDataSet(sWN);  
 lineData.addDataSet(sWR);  
 }  
  
 intersecChart.setData(lineData);  
 intersecChart.getLegend().setTextColor(getColor(R.color.*cyan*));  
 intersecChart.getXAxis().setGridColor(getColor(R.color.*black*));  
 intersecChart.getAxisLeft().setGridColor(getColor(R.color.*black*));  
 intersecChart.getAxisRight().setGridColor(getColor(R.color.*black*));  
 intersecChart.setBackgroundColor(getColor(R.color.*black*));  
 intersecChart.invalidate();  
  
 }  
  
  
 @Override  
 public void onDestroy(){  
 finish();  
 super.onDestroy();  
 }  
}