ackage ch.fraise;  
  
import java.io.IOException;  
import android.app.Activity;  
import android.hardware.Camera;  
import android.hardware.Camera.PreviewCallback;  
import android.media.MediaRecorder;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.Menu;  
import android.view.MenuInflater;  
import android.view.MenuItem;  
import android.view.SurfaceHolder;  
import android.view.SurfaceView;  
  
public class CameraActivity extends Activity implements SurfaceHolder.Callback,  
    Camera.AutoFocusCallback {  
  
private SurfaceView preview;  
private SurfaceHolder previewHolder;  
  
private MediaRecorder mRecorder;  
private Camera mCamera;  
private boolean mPreviewRunning = false;  
private boolean mCaptureFrame = false;  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Log.e("", "Begin onCreate");  
    setContentView(R.layout.main);  
  
    preview = (SurfaceView) findViewById(R.id.surfaceView1);  
    previewHolder = preview.getHolder();  
    previewHolder.addCallback(this);  
    previewHolder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);  
  
    mRecorder = new MediaRecorder();  
}  
  
@Override  
public void onResume() {  
    super.onResume();  
}  
  
@Override  
public void onPause() {  
    super.onPause();  
}  
  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.capture\_menu, menu);  
    return true;  
}  
  
public void startRecording() {  
    Log.e("", "Begin StartRecording");  
    mCaptureFrame = true;  
    mRecorder.start();  
}  
  
public void stopRecording() {  
    Log.e("", "Begin StopChange");  
    mRecorder.stop();  
}  
  
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle item selection  
    switch (item.getItemId()) {  
    case R.id.startRecording:  
        startRecording();  
        return true;  
    case R.id.stopRecording:  
        stopRecording();  
        return true;  
    default:  
        return super.onOptionsItemSelected(item);  
    }  
}  
  
@Override  
public void surfaceCreated(SurfaceHolder holder) {  
    Log.e("", "Begin surfaceDestroy");  
    mCamera = Camera.open();  
}  
  
@Override  
public void surfaceDestroyed(SurfaceHolder holder) {  
    mCamera.stopPreview();  
    mPreviewRunning = false;  
    mCamera.release();  
  
    mRecorder.reset();  
    mRecorder.release();  
}  
  
@Override  
public void onAutoFocus(boolean success, Camera camera) {  
    // TODO Auto-generated method stub  
  
}  
  
/\*  
 \* PreviewCallback()  
 \*   
 \* this callback captures the preview at every frame and puts it in a byte  
 \* buffer. we will evaluate if this is a frame that we want to process, and  
 \* if so, we will send it to an asynchronous thread that will process it to  
 \* an ARGB Bitmap and POST it to the server  
 \*/  
PreviewCallback previewCallback = new PreviewCallback() {  
    public void onPreviewFrame(byte[] data, Camera camera) {  
        Log.e("", "onPreviewFrame pass");  
        if (mCaptureFrame) {  
            mCaptureFrame = false;  
            // new FrameHandler().execute(data);  
        }  
    }  
};  
  
@Override  
public void surfaceChanged(SurfaceHolder holder, int format, int width,  
        int height) {  
    Log.e("", "Begin SurfaceChange");  
  
    mRecorder.reset();  
    mRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);  
    mRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG\_4);  
    mRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG\_4\_SP);  
    mRecorder.setOutputFile("/sdcard/videotest2.mp4");  
    mRecorder.setVideoFrameRate(30);  
  
    mRecorder.setPreviewDisplay(previewHolder.getSurface());  
    try {  
        mRecorder.prepare();  
    } catch (IllegalStateException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
  
    if (mPreviewRunning)  
        mCamera.stopPreview();  
  
    Camera.Parameters p = mCamera.getParameters();  
    // p.setPreviewSize(width, height);  
    mCamera.setParameters(p);  
  
    try {  
        mCamera.setPreviewDisplay(holder);  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
  
    mCamera.setPreviewCallback(previewCallback);  
  
    mCamera.startPreview();  
    mPreviewRunning = true;  
  
}  
  
}