Android Camera Application

Camera Application

- Most Android smart phones have a front and back facing camera
- Photos can be taken and then processed by Computer Vision Algorithms
- The following permission is required to be declared in manifest file:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

- Two important components:
  - SurfaceView
  - Camera class

SurfaceView

- SurfaceView class is used to present a live camera preview to the user

```
SurfaceView surfaceView;
surfaceView = (SurfaceView) findViewById(R.id.camera);
```

- SurfaceHolder interface providing access and control over the SurfaceView, you can get this interface by simply calling getHolder() method

```
SurfaceHolder surfaceHolder;
surfaceHolder = surfaceView.getHolder();
```

- To receive information about changes to the surface, we have to implement SurfaceHolder.Callback interface

```
public class MainActivity extends AppCompatActivity implements SurfaceHolder.Callback
surfaceHolder.addCallback(this);
```

Layout

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.xunhu.camera.MainActivity">
    <SurfaceView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/camera"
        />
```

Camera (Setting up)

- The Camera class is used to control your device camera
- open(int): obtain an instance of Camera class

```
Camera mCamera;
mCamera = Camera.open(Camera.CameraInfo.CAMERA_FACING_FRONT);
mCamera = Camera.open(Camera.CameraInfo.CAMERA_FACING_BACK);
```

- getParameters(): obtain existing default settings

```
Camera.Parameters parameters;
parameters = mCamera.getParameters();
parameters.setFlashMode(Camera.Parameters.FLASH_MODE_ON);
mCamera.setParameters(parameters);
```

Camera (Setting up)

- setDisplayOrientation(int): to ensure correct orientation of preview

```
mCamera.setDisplayOrientation(90);
```

- startPreview(): start updating the preview surface, this method should be called before you can take a picture

```
mCamera.startPreview();
```

- startPreviewDisplay(SurfaceHolder): pass your initialized Surface

```
mCamera.setPreviewDisplay(surfaceHolder);
```

- release(): release the camera for use by other applications

```
mCamera.release();
```

Camera (Switching Camera)

- Make sure you release your current camera object before switching

```
mCamera.stopPreview();
if (mCamera!=null){
    mCamera.release();
    mCamera=null;
}
mCamera = Camera.open(Camera.CameraInfo.CAMERA_FACING_FRONT);
try {
    mCamera.setPreviewDisplay(surfaceHolder);
} catch (IOException e) {
    e.printStackTrace();
}
mCamera.setDisplayOrientation(90);
mCamera.startPreview();
```

Camera (Taking photos)

- takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback postview, Camera.PictureCallback jpeg)
- shutter callback occurs after image is captured.

```
@Override
public void onShutter() {

}
```

- jpeg callback occurs when the compressed image is available

```
@Override
public void onPictureTaken(byte[] bytes, Camera camera) {

}
```

Saving Media Files

- Media files such as pictures and video should be stored into external storage
- The following permission should be added into manifest file

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

- Two standard locations to save media files on devices:

  Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICUTRES): return a standard and shared location for saving pictures and videos

  Context.getExternalFilesDir(Environment.DIRECTORY_PICTURES): return a standard location for saving pictures and video which are associated with your application.

Saving Media Files

- Step 1: create a directory to save the file

```
File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
        "MyCameraApplication");
if (!mediaStorageDir.exists()){
    mediaStorageDir.mkdirs();
}
```

- Step 2: create a jpg file

```
String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
File mediaFile;
mediaFile = new File(mediaStorageDir.getPath()+File.separator+"IMG_"+timeStamp+".jpg");
```

- Step 3: store your bitmap into the jpg file you created

```
FileOutputStream outputStream = null;
try {
    outputStream = new FileOutputStream(mediaFile);
    takenPhoto.compress(Bitmap.CompressFormat.JPEG, 100, outputStream);
```

- Step 4: update gallery

```
sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, Uri.fromFile(mediaFile)));
```

Camera (Face detection)

- Step 1: Create a face detection listener

```
mCamera.setFaceDetectionListener(new Camera.FaceDetectionListener() {
    @Override
    public void onFaceDetection(Camera.Face[] faces, Camera camera) {
        if (faces.length > 0){
            System.out.println("@ FaceDetection"+ "face detected: "+ faces.length +
                    " Face 1 Location X: " + faces[0].rect.centerX() +
                    "Y: " + faces[0].rect.centerY() );
        }
    }
});
```

Camera (Face detection)

- Step 2: start Face detection

```
parameters = mCamera.getParameters();
if (parameters.getMaxNumDetectedFaces()>0){
    mCamera.startFaceDetection();
    Toast.makeText(getApplicationContext(),"face detection start", Toast.LENGTH_LONG).show();
}
```

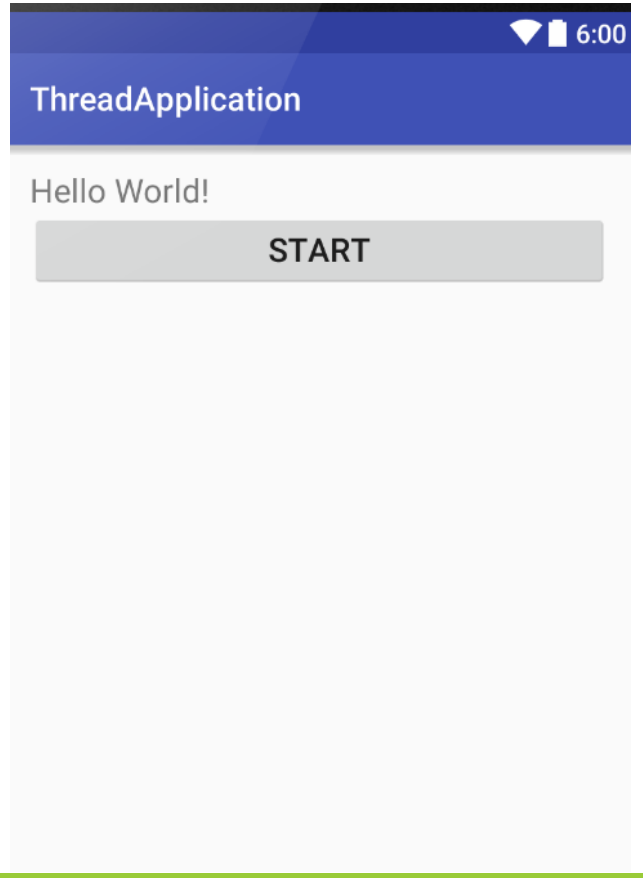
Thread in android

- How can we change/update UI if we obtain data in thread

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textView = (TextView) findViewById(R.id.tvOne);
    button = (Button)findViewById(R.id.btnOne);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    String text = "my name is xun";

                }
            }).start();
        }
    });

}
```



Thread in android

- You are not allowed to change/update UI within other threads
- Handler class: allows you to send and process Message and Runnable objects associated with a thread's MessageQueue

```
android.os.Handler handler = new android.os.Handler(new Handler.Callback() {
    @Override
    public boolean handleMessage(Message message) {
        if (message.what==MESSAGE_RETRIEVED){
            // update UI
        }
        return false;
    }
});
```

Thread in android

- Create a Message object and use handler to send this message

```
public void onClick(View view) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            String text = "my name is xun";
            Message message = new Message();
            message.what = MESSAGE_RETRIEVED;
            message.obj = text;
            handler.sendMessage(message);
        }
    }).start();
}
});
```

Thread in android

- Once the handler receive the message, we can update our UI

```
android.os.Handler handler = new android.os.Handler(new Handler.Callback() {
    @Override
    public boolean handleMessage(Message message) {
        if (message.what==MESSAGE_RETRIEVED){
            // update UI
            textView.setText(message.obj.toString());
        }
        return false;
    }
});
```