Introdução aos dispositivos móveis

Alguns sensores

Marcelo Zamith

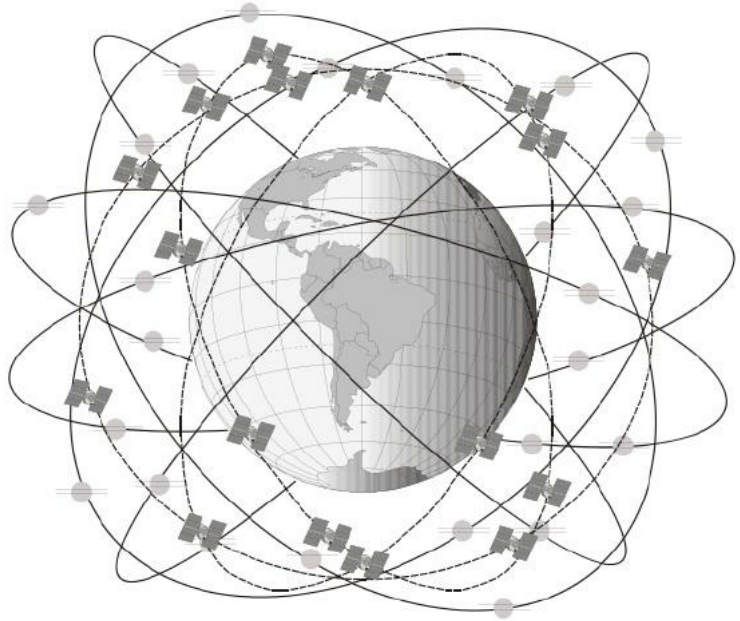
e-mail: mzamith@ufrrj.br
https://www.dcc.ufrrj.br/~marcelo/

Universidade Federal Rural do Rio de Janeiro - DCC

Introdução

Introdução

- Um dispositivo que recebe um sinal e responde a um estímulo
- É uma forma de perceber o mundo



UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
UFRRJ

Introdução

- Permissões versus versão do SDK
- Arquivo AndroidManifest.xml **E** no código
- Presença do sensor ou hardware
- Variedade de dispositivos e diferentes configurações
- Configuração necessária versus permissão

Introdução

- Câmeras:
  - Frontal
  - Traseira
- Comunicação:
  - Bluetooth
  - Wi-Fi
- Movimento:
  - Acelerômetro (força)
  - Gravidade
  - Giroscópio (taxa de rotação)
  - Aceleração linear
  - Vetor de rotação
  - Contator de passos
- Posição:
  - Vetor de rotação de jogo
  - Vetor de rotação geomagnético
  - Campo magnético
  - Orientação
  - Proximidade
  - GPS
- Sensores do ambiente:
  - Temperatura do ambiente - ar (Celsius)
  - Temperatura do aparelho (Celsius)
  - Iluminação
  - Pressão ambiente (hPa) ou (mbar)
  - Umidade relativa %

Introdução

- O que é necessário para usar um sensor:

Introdução

- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware

Introdução

- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware
  2. Permissões

Introdução

- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware
  2. Permissões
  3. Configurações e inicializações

Introdução

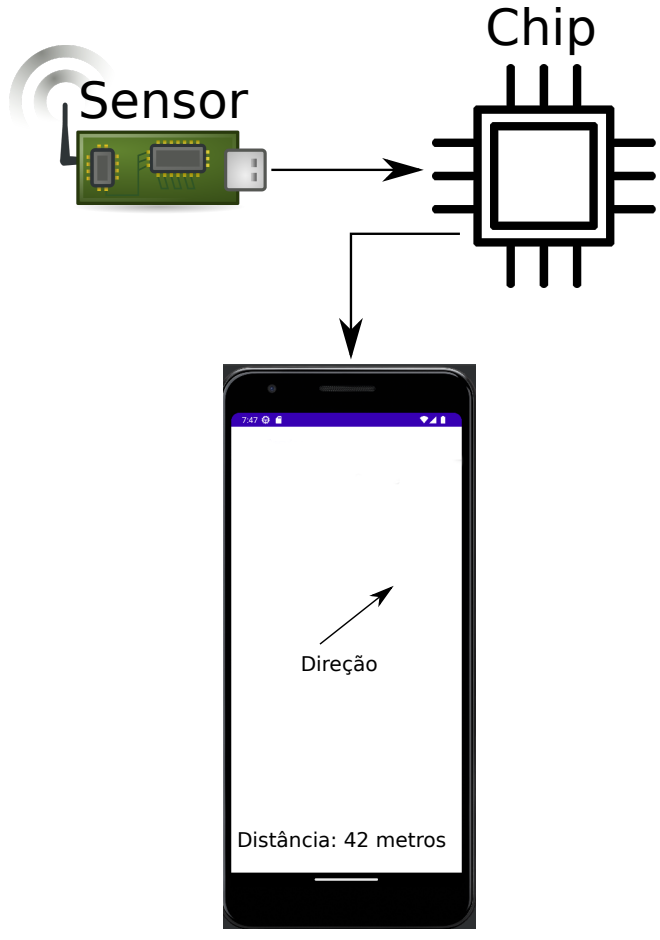
- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware
  2. Permissões
  3. Configurações e inicializações
  4. `Callbacks` e eventos

Introdução

- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware
  2. Permissões
  3. Configurações e inicializações
  4. `Callbacks` e eventos
  5. Desalocação dos recursos - bateria

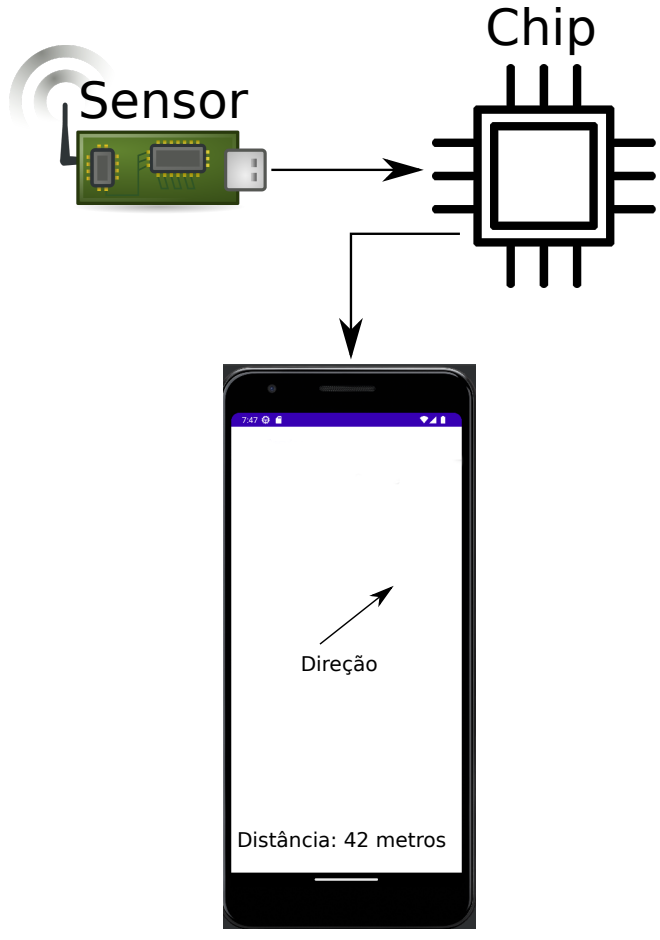
Introdução

- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware
  2. Permissões
  3. Configurações e inicializações
  4. `Callbacks` e eventos
  5. Desalocação dos recursos - bateria



Introdução

- O que é necessário para usar um sensor:
  1. Aplicação e disponibilidade do hardware
  2. Permissões
  3. Configurações e inicializações
  4. `Callbacks` e eventos
  5. Desalocação dos recursos - bateria



IMPORTANTE

Passos necessários para cada sensor

GPS - Global Positioning System

GPS

- Localização em ambientes abertos
- Uso de satélites para posicionamento do dispositivo
- Não são os mapas e nem as rotas - coordenadas do globo
- Sistema de coordenadas geográficas WGS84
  - Longitude - meridiano de Greenwich (0^o)
  - Latitude - linha do Equador (0^o)
- Exemplo - UFRRJ - IM - DCC
  - Coordenadas: $22^o44'32.4''S$, $43^o27'31.6''W$
  - Projeção: -22.74234542558444, -43.458787698966354

GPS

Disponibilidade do hardware

- Todo o dispositivo possui
- Não há necessidade

GPS

Permissões

- `AndroidManifest.xml`:

```
1    ...
2    <uses-permission android:name="android.permission.INTERNET" />
3    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
4    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
5    ...
```

- Classe `Acitivity`:

```
1   ...
2     if (ActivityCompat.checkSelfPermission(...) != PackageManager.PERMISSION_GRANTED &&
3         ActivityCompat.checkSelfPermission(...) != PackageManager.PERMISSION_GRANTED) {
4
5       ActivityCompat.requestPermissions(SecondActivity.this, new String[]{..);
6     }
7   ...
```

- Tela de dialogo
- Usuário nega a permissão

GPS

Configurações e inicializações

- No caso do GPS:
  - Dados do GPS
  - Intervalo de tempo - MIN_TIME_MS
  - Intervalo de distância - MIN_DISTANCE_M
  - A `callback`

```
1      ...
2   mLocationManager = (LocationManager) this.getSystemService(LOCATION_SERVICE);
3      ...
4   mMyLocationListener =  new MyLocationListener();
5   mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
6   MIN_TIME_MS, MIN_DISTANCE_M, mMyLocationListener);
7   Location location = mLocationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
8   if (location != null)  mMyLocationListener.onLocationChanged(location);
9      ...
```

GPS

Callbacks

```
1         ...
2         mMyLocationListener =  new MyLocationListener();
3         ...
```

```
1     ...
2     private class MyLocationListener implements LocationListener{
3         @Override
4         public void onLocationChanged(@NonNull Location location) {
5             ...
6         }
7     ...
```

GPS

- O que fazer com o GPS ?
  - Sua posição: `onLocationChanged(@NonNull Location location)`
    - `Location.getLongitude()`: $[-180^o, 180^o]$ - `double`
    - `Location.getLatitude()`: $[-90^o, 90^o]$ - `double`
    - `Location.getSpeed()`: Velocidade em m/s - `float`
    - `Location.getAltitude()`: Altitude em m - `double`
    - `Location.distanceTo(..)`: Distância em m - `float`

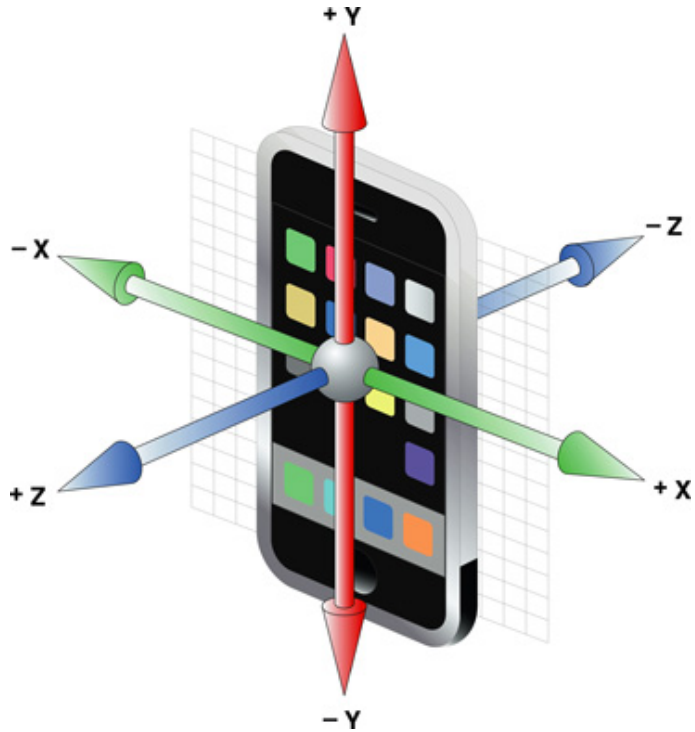
ATENÇÃO

Atenção para as unidades! ←

Acelerômetro

Acelerômetro

- O que é o acelerômetro:
  - Aceleração em m/s^2
  - 3 eixos: x, y e z
  - Gravidade
  - Calibração
  - Orientação do aparelho



Acelerômetro

Aplicação e disponibilidade do hardware

- Presente em todos os dispositivos
- Verificar via API:
  - `PackageManager.hasSystemFeature(PackageManager.FEATURE_SENSOR_ACCELEROMETER)`
  - `PackageManager.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS)`
- Grupo de sensores:
  - Movimento
  - Posição
  - Ambiente

Acelerômetro

Permissões

- Não precisa

Acelerômetro

Configurações e inicializações

```
1   ...
2   PackageManager m = getPackageManager();
3   if(!m.hasSystemFeature(PackageManager.FEATURE\_SENSOR\_ACCELEROMETER)) {
4       ...
5   }
6   ...
7   mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
8   ...
9   Sensor accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
10  ...
11
12  protected void onResume() {
13      super.onResume();
14      mSensorManager.registerListener(callback, accelerometer, SensorManager.SENSOR_DELAY_NORMAL
15  }
```

- Sensor Manager
- callbak
- Tempo do evento:
  - `SensorManager.SENSOR_DELAY_NORMAL`: tela - padrão
  - `SensorManager.SENSOR_DELAY_FASTEST`: mais rápido possível !
  - `SensorManager.SENSOR_DELAY_GAME`: do jogo - laço do jogo
  - `SensorManager.SENSOR_DELAY_UI`: interface
- Consumo versus resposta

Acelerômetro

Callbacks e eventos

```
1   private class AccelerometerSensorListener implements SensorEventListener {
2       @Override
3       public void onSensorChanged(SensorEvent event) {
4           String aux = "(" + Float.toString(event.values[0]) + "," +
5                              Float.toString(event.values[1]) + "," +
6                              Float.toString(event.values[2]) + ")";
7           mAccele.setText(aux);
8       }
9
10      @Override
11      public void onAccuracyChanged(Sensor sensor, int accuracy) {
12
13      }
14  }
```

- É um dos parâmetros: `sensorManager.registerListener`
- 3 valores: x, y e z
- Unidade
- Ruídos
- Calibração

Acelerômetro

Desalocação dos recursos

- Libera recurso e economiza energia

```
1    protected void onPause() {
2      super.onPause();
3      mSensorManager.unregisterListener(this);
4    }
```

Câmeras

Câmeras

- O hardware
- Permissão & segurança
- Orientação da imagem
- Um frame e um fluxo de frames
- Projeção da imagem
- Formato do arquivo (jpg)
- Gravação da imagem

Câmeras

Disponibilidade do hardware

- Grande variedade de fabricantes
- Diferentes câmeras e configurações:
  - FPS
  - Imagem stereo
- Recursos:
  - flash
  - distância focal

Câmeras

Configurações e inicializações

- Disponibilidade do hardware - `AndroidManifest.xml`

```
1  ...
2  <uses-feature android:name="android.hardware.camera" android:required="false" />
3  <uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />
4  <uses-feature android:name="android.hardware.camera.flash" android:required="false" />
5  ...
```

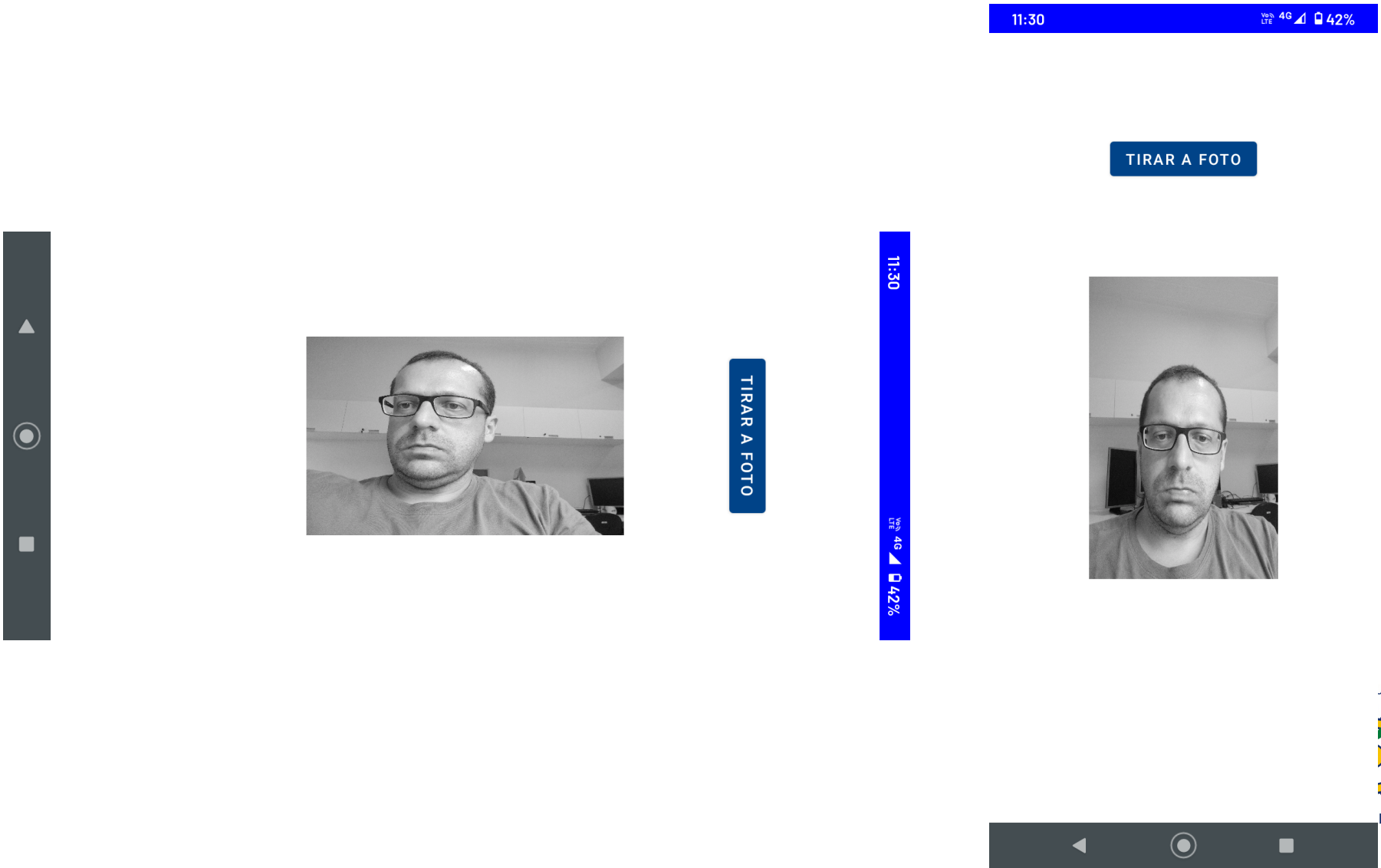
- permissão do hardware - `AndroidManifest.xml`

```
1  ...
2  <uses-permission android:name="android.permission.CAMERA" />
3  ...
```

Câmeras

Configurações e inicializações

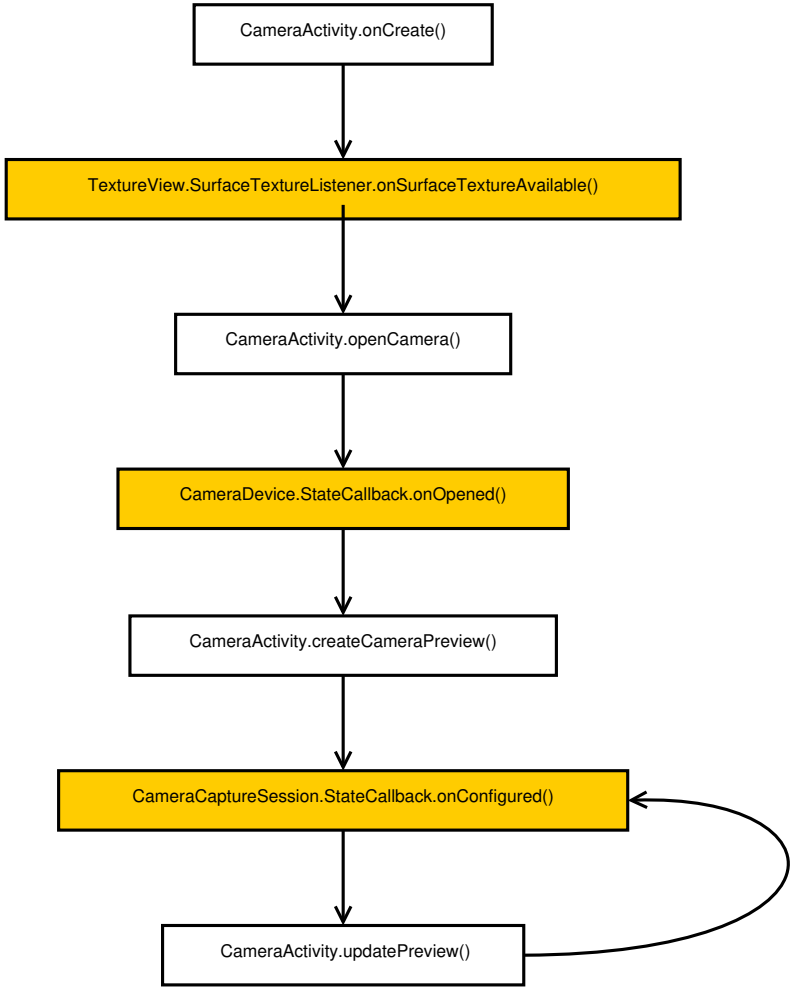
- Orientação do sensor da câmera do celular
- Câmeras disponíveis, imagem stereo e escolha da câmera



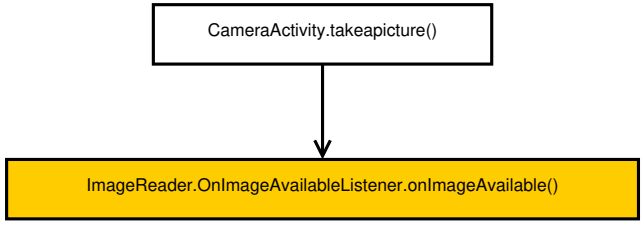
Câmeras

Callbacks e eventos

- Fluxo de eventos para abrir a câmera no celular - fluxo de vídeo



- Fluxo de eventos para tirar uma foto



Câmeras

`Callbacks` e eventos: `CameraActivity.onCreate()`

- Cria as instâncias dos objetos
- Os eventos dos objetos
  - `TextureView`
  - `TextureView.setSurfaceTextureListener(new TextureView.SurfaceTextureListener)`

Câmeras

`Callbacks` e eventos: `TextureView.SurfaceTextureListener.onSurfaceTextureAvailable()`

- Evento disparado pela `TextureView`
- `onSurfaceTextureAvailable(..)`:
  - `CameraActivity.openCamera()`

Câmeras

`Callbacks` e eventos: `CameraActivity.openCamera()`

- Acesso ao serviço de câmera
- Obtém as características da câmera (já sabe qual é a câmera)
- Associa o stream de imagem da câmera com a Texture Surface
- Necessidade de pedir permissão durante o uso (segurança)
- Serviço câmera ativa a hardware
  - Callback: `CameraDevice.StateCallback.onOpened()`

Câmeras

`Callbacks` e **eventos:** `CameraDevice.StateCallback.onOpened()`

- Evento associado a definição da câmera (CameraDevice)
- No exemplo, chamar `CameraActivity.createCameraPreview()`

Câmeras

`Callbacks` e eventos: `CameraActivity.createCameraPreview()`

- Cria buffer de textura
- Configura a câmera:
  - Tipo de compressão da imagem (jpg)
  - Imagem preto e branco
  - Orientação de captura da imagem

Câmeras

`Callbacks` e eventos: `CameraCaptureSession.StateCallback.onConfigured()`

- Callback - previa da imagem, atualização do fluxo da câmera
- Defina a sessão de aquisição das imagens
- Chamada `CameraActivity.updatePreview()` em loop até o fim da `activity`

Câmeras

`Callbacks` e **eventos:** `CameraActivity.updatePreview()`

- Mantém o fluxo da câmera para a surface
- FPS

Câmeras

`Callbacks` e eventos: `CameraActivity.takeapicture()`

- Evento disparado pelo usuário - botão
- Configuração de um buffer para ler os dados da câmera
- Configura a câmera para a foto:
  - Tipo de compressão da imagem (jpg)
  - Imagem preto e branco
  - Orientação de captura da imagem
  - Outras configurações
  - Pode ser diferente da prévia
- Evento de leitura da câmera, gravação da imagem

Câmeras

`Callbacks` e **eventos:** `ImageReader.OnImageAvailableListener.onImageAvailable()`

- Finalização da aquisição da imagem
- Passo seguinte é: o que fazer com a imagem?
- O fluxo da câmera continua - atualização da Texture Surface

Câmeras

Desalocação dos recursos

- Feita de forma automática

Considerações finais

- Sensores:
  - Programação baseada nos eventos
  - Diferentes fabricantes e características
  - Versões do SDK e do SO
  - Erro na aquisição dos dados
  - Interpretar os resultados
  - Pensar fora da caixa
  - Disponibilidade dos sensores - restringir ?
  - Sensores mais comuns
  - Câmeras e imagens:
    - Orientação
    - Uso de matrizes de rotação

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
UFRRJ

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
UFRRJ