

CSE 162 Mobile Computing Lab

Lab5b Media Recorder

Department of Computer Science and Engineering  
University of California, Merced, CA

# Goal: achieve the following features

- Control the media recording capabilities
- Learn the MediaRecorder API
- Learn the Camera API

# Outline

- create an app to shoot video
- Give it a name
- Push a button, the app begins to preview and record video
- Push the button again, save locally



# permission in the manifest file

```
<uses-feature  
    android:name="android.hardware.camera"  
    android:required="false" />  
  
<uses-permission android:name="android.permission.CAMERA" />  
<uses-permission android:name="android.permission.RECORD_AUDIO" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"  
    tools:ignore="ScopedStorage" />  
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

# other parts of the manifest

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Media"
    tools:targetApi="31">

    <meta-data
        android:name="androidx.camera.lifecycle.ProcessCameraProvider"
        android:value="androidx.camera.camera2.Camera2Config" />

    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

</application>

</manifest>
```

# Prepare the UI

- in `main_activity.xml`, structured as follows
  - FrameLayout
    - Textureview
    - LinearLayout
      - EditText
      - Button

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.camera.view.PreviewView
        android:id="@+id/preview_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <EditText
        android:id="@+id/video_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Video Name"
        android:background="@android:color/white"
        android:padding="8dp"
        android:layout_marginTop="16dp"
        android:layout_alignParentTop="true"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_centerHorizontal="true" />

    <Button
        android:id="@+id/button_capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="START"
        android:textSize="18sp"
        android:layout_marginBottom="32dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_alignParentBottom="true" />
```

prepare for video recording,  
Prepare for the file storage,  
videoCapture.getOutput()

```
181         currentRecording = videoCapture.getOutput() Recorder
182             .prepareRecording( context: this, mediaStoreOutputOptions) PendingRecording
183             .withAudioEnabled()
184             .start(ContextCompat.getMainExecutor( context: this), videoRecordEvent -> {
185                 if (videoRecordEvent instanceof VideoRecordEvent.Finalize) {
186                     if (!((VideoRecordEvent.Finalize) videoRecordEvent).hasError()) {
187                         Log.d(TAG, msg: "Video saved successfully");
188                     } else {
189                         Log.e(TAG, msg: "Video saving failed: " +
190                             ((VideoRecordEvent.Finalize) videoRecordEvent).getError());
191                     }
192                 }
193             });
194 }
```

# Configure the camera

```
private void startCamera() { 2 usages
    ListenableFuture<ProcessCameraProvider> cameraProviderFuture =
        ProcessCameraProvider.getInstance( context: this);

    cameraProviderFuture.addListener(() -> {
        try {
            ProcessCameraProvider cameraProvider = cameraProviderFuture.get();
            bindPreviewAndVideoCapture(cameraProvider);
        } catch (InterruptedException | ExecutionException e) {
            Log.e(TAG, msg: "Error starting camera: " + e.getMessage());
        }
    }, ContextCompat.getMainExecutor( context: this));
}

private void bindPreviewAndVideoCapture(@NonNull ProcessCameraProvider cameraProvider) { 1 usage
    Preview preview = new Preview.Builder().build();
    Recorder recorder = new Recorder.Builder().build();
    videoCapture = VideoCapture.withOutput(recorder);

    CameraSelector cameraSelector = CameraSelector.DEFAULT_BACK_CAMERA;

    preview.setSurfaceProvider(previewView.getSurfaceProvider());

    try {
        cameraProvider.unbindAll();
        cameraProvider.bindToLifecycle(
            (LifecycleOwner) this, cameraSelector, preview, videoCapture
        );
    } catch (Exception e) {
        Log.e(TAG, msg: "Use case binding failed", e);
    }
}
```

# Use the media recorder

```
public void onCaptureClick(View view) { 1usage
    if (isRecording) {
        // Stop recording
        currentRecording.stop();
        currentRecording = null;
        setCaptureButtonText("START");
        isRecording = false;
    } else {
        startRecording();
        setCaptureButtonText("STOP");
        isRecording = true;
    }
}
```

# startRecording()

```
148     private void startRecording() { 1usage
149         String videoName = editText.getText().toString();
150         String timeStamp = new SimpleDateFormat( pattern: "yyyy-MM-dd_HH:mm:ss", Locale.US).format(new Date());
151         String fileName = videoName + "_" + timeStamp + ".mp4";
152
153         ContentValues contentValues = new ContentValues();
154         contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, fileName);
155         contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "video/mp4");
156
157         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
158             contentValues.put(
159                 MediaStore.Video.Media.RELATIVE_PATH, "Movies/Lab3_Recordings"
160             );
161         }
162
163         MediaStoreOutputOptions mediaStoreOutputOptions =
164             new MediaStoreOutputOptions.Builder(
165                 getContentResolver(),
166                 MediaStore.Video.Media.EXTERNAL_CONTENT_URI
167             )
168                 .setContentValues(contentValues)
169                 .build();
170
171         if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.RECORD_AUDIO ) != PackageManager.PERMISSION_GRANTED)
172             Log.d( tag: "permission check", msg: "permission check failed");
173             return;
174         }
175         else
176         {
177             Log.d( tag: "permission check", msg: "permission check passed");
178         }
```

When the recording stops, release the camera and the mediarecorder

```
private void setCaptureButtonText(String title) { 2 usages
    captureButton.setText(title);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (currentRecording != null) {
        currentRecording.stop();
        currentRecording = null;
    }
}
```

# Overview of MainActivity

```
1 package yiliu.lab5b_yliu327.media;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5 import androidx.camera.core.CameraSelector;
6 import androidx.camera.core.Preview;
7 import androidx.camera.lifecycle.ProcessCameraProvider;
8 import androidx.camera.video.MediaStoreOutputOptions;
9 import androidx.camera.video.Recorder;
10 import androidx.camera.video.Recording;
11 import androidx.camera.video.VideoCapture;
12 import androidx.camera.video.VideoRecordEvent;
13 import androidx.core.app.ActivityCompat;
14 import androidx.core.content.ContextCompat;
15 import android.Manifest;
16 import android.content.ContentValues;
17 import android.content.pm.PackageManager;
18 import android.os.Build;
19 import android.os.Bundle;
20 import android.provider.MediaStore;
21 import android.util.Log;
22 import android.view.View;
23 import android.widget.Button;
24 import android.widget.EditText;
25 import androidx.camera.view.PreviewView;
26 import androidx.lifecycle.LifecycleOwner;
27 import com.google.common.util.concurrent.ListenableFuture;
28 import java.text.SimpleDateFormat;
29 import java.util.Date;
30 import java.util.Locale;
31 import java.util.concurrent.ExecutionException;
```

## Import Section

```
33 ></> public class MainActivity extends AppCompatActivity {
34
35     private PreviewView previewView; 2 usages
36     private Button captureButton; 3 usages
37     private EditText editText; 2 usages
38     private boolean isRecording = false; 3 usages
39     private VideoCapture<Recorder> videoCapture; 3 usages
40     private Recording currentRecording; 6 usages
41
42     private static final String TAG = "Recorder"; 4 usages
43     private static final int REQUEST_CODE_PERMISSIONS = 200; 2 usages
44     private final String[] REQUIRED_PERMISSIONS = new String[]{ 2 usages
45         Manifest.permission.CAMERA,
46         Manifest.permission.RECORD_AUDIO
47     };
48 }
```

## Declare your variables

# MainActivity (Continued)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    previewView = findViewById(R.id.preview_view);
    captureButton = findViewById(R.id.button_capture);
    editText = findViewById(R.id.video_name);

    if (allPermissionsGranted()) {
        startCamera();
    } else {
        ActivityCompat.requestPermissions(
            activity: this, REQUIRED_PERMISSIONS, REQUEST_CODE_PERMISSIONS
        );
    }

    captureButton.setOnClickListener(this::onCaptureClick);
}

private boolean allPermissionsGranted() { 2 usages
    for (String permission : REQUIRED_PERMISSIONS) {
        if (ContextCompat.checkSelfPermission(
            context: this, permission) != PackageManager.PERMISSION_GRANTED
        ) {
            return false;
        }
    }
    return true;
}
```

```
80
81 @Override 14 usages
82     public void onRequestPermissionsResult(
83         int requestCode,
84         @NonNull String[] permissions,
85         @NonNull int[] grantResults
86     ) {
87         super.onRequestPermissionsResult(requestCode, permissions, grantResults);
88         if (requestCode == REQUEST_CODE_PERMISSIONS) {
89             if (allPermissionsGranted()) {
90                 startCamera();
91             } else {
92                 // Permission not granted, exit the app
93                 finish();
94             }
95         }
96     }

97     private void startCamera() { 2 usages
98         ListenableFuture<ProcessCameraProvider> cameraProviderFuture =
99             ProcessCameraProvider.getInstance(context: this);
100

101        cameraProviderFuture.addListener(() -> {
102            try {
103                ProcessCameraProvider cameraProvider = cameraProviderFuture.get();
104                bindPreviewAndVideoCapture(cameraProvider);
105            } catch (InterruptedException | ExecutionException e) {
106                Log.e(TAG, msg: "Error starting camera: " + e.getMessage());
107            }
108        }, ContextCompat.getMainExecutor(context: this));
109    }
```

# MainActivity (Continued)

```
111     private void bindPreviewAndVideoCapture(@NonNull ProcessCameraProvider cameraProvider) { 1 usage
112         Preview preview = new Preview.Builder().build();
113         Recorder recorder = new Recorder.Builder().build();
114         videoCapture = VideoCapture.withOutput(recorder);
115
116         CameraSelector cameraSelector = CameraSelector.DEFAULT_BACK_CAMERA;
117
118         preview.setSurfaceProvider(previewView.getSurfaceProvider());
119
120         try {
121             cameraProvider.unbindAll();
122             cameraProvider.bindToLifecycle(
123                 (LifecycleOwner) this, cameraSelector, preview, videoCapture
124             );
125         } catch (Exception e) {
126             Log.e(TAG, msg: "Use case binding failed", e);
127         }
128     }
129
130     public void onCaptureClick(View view) { 1 usage
131         if (isRecording) {
132             // Stop recording
133             currentRecording.stop();
134             currentRecording = null;
135             setCaptureButtonText("START");
136             isRecording = false;
137         } else {
138             startRecording();
139             setCaptureButtonText("STOP");
140             isRecording = true;
141         }
142     }
```

# MainActivity (Continued)

```
144     private void startRecording() { 1 usage
145         String videoName = editText.getText().toString();
146         String timeStamp = new SimpleDateFormat( pattern: "yyyy-MM-dd_HH:mm:ss", Locale.US).format(new Date());
147         String fileName = videoName + "_" + timeStamp + ".mp4";
148
149         ContentValues contentValues = new ContentValues();
150         contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, fileName);
151         contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "video/mp4");
152
153         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
154             contentValues.put(
155                 MediaStore.Video.Media.RELATIVE_PATH, "Movies/Lab3_Recordings"
156             );
157         }
158
159         MediaStoreOutputOptions mediaStoreOutputOptions =
160             new MediaStoreOutputOptions.Builder(
161                 getContentResolver(),
162                 MediaStore.Video.Media.EXTERNAL_CONTENT_URI
163             )
164                 .setContentValues(contentValues)
165                 .build();
166
167         if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.RECORD_AUDIO) != PackageManager.PERMISSION_GRANTED)
168             Log.d( tag: "permission check", msg: "permission check failed");
169             return;
170         }
171         else
172         {
173             Log.d( tag: "permission check", msg: "permission check passed");
174         }
```

# MainActivity (Continued)

```
176     currentRecording = videoCapture.getOutput() Recorder
177         .prepareRecording( context: this, mediaStoreOutputOptions) PendingRecording
178         .withAudioEnabled()
179         .start(ContextCompat.getMainExecutor( context: this), videoRecordEvent -> {
180             if (videoRecordEvent instanceof VideoRecordEvent.Finalize) {
181                 if (!((VideoRecordEvent.Finalize) videoRecordEvent).hasError()) {
182                     Log.d(TAG, msg: "Video saved successfully");
183                 } else {
184                     Log.e(TAG, msg: "Video saving failed: " +
185                         ((VideoRecordEvent.Finalize) videoRecordEvent).getError());
186                 }
187             }
188         });
189     }
190
191     private void setCaptureButtonText(String title) { 2 usages
192         captureButton.setText(title);
193     }
194
195     @Override
196     protected void onDestroy() {
197         super.onDestroy();
198         if (currentRecording != null) {
199             currentRecording.stop();
200             currentRecording = null;
201         }
202     }
203 }
```

# activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.camera.view.PreviewView
        android:id="@+id/preview_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <EditText
        android:id="@+id/video_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Video Name"
        android:background="@android:color/white"
        android:padding="8dp"
        android:layout_marginTop="16dp"
        android:layout_alignParentTop="true"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_centerHorizontal="true" />
```

```
<Button
    android:id="@+id/button_capture"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="START"
    android:textSize="18sp"
    android:layout_marginBottom="32dp"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />
```

```
</RelativeLayout>
```

# Manifest

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <uses-feature
6         android:name="android.hardware.camera"
7         android:required="false" />
8
9     <uses-permission android:name="android.permission.CAMERA" />
10    <uses-permission android:name="android.permission.RECORD_AUDIO" />
11    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
12        tools:ignore="ScopedStorage" />
13    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
14
15    <application
16        android:allowBackup="true"
17        android:dataExtractionRules="@xml/data_extraction_rules"
18        android:fullBackupContent="@xml/backup_rules"
19        android:icon="@mipmap/ic_launcher"
20        android:label="@string/app_name"
21        android:roundIcon="@mipmap/ic_launcher_round"
22        android:supportsRtl="true"
23        android:theme="@style/Theme.Media"
24        tools:targetApi="31">
25
26        <meta-data
27            android:name="androidx.camera.lifecycle.ProcessCameraProvider"
28            android:value="androidx.camera.camera2.Camera2Config" />
29
```

# Gradle (app level) dependencies

```
dependencies {  
  
    implementation libs.appcompat  
    implementation libs.material  
    testImplementation libs.junit  
    androidTestImplementation libs.ext.junit  
    androidTestImplementation libs.espresso.core  
    implementation "androidx.camera:camera-core:1.3.0"  
    implementation "androidx.camera:camera-camera2:1.3.0"  
    implementation "androidx.camera:camera-lifecycle:1.3.0"  
    implementation "androidx.camera:camera-video:1.3.0"  
    implementation "androidx.camera:camera-view:1.3.0"  
}
```

## Extra credit

- Add one more button on the UI. Click and play the most recently recorded video in the preview.