Android Camera Manager

Application example to use custom `Camera1Capturer` interface to capture images from camera

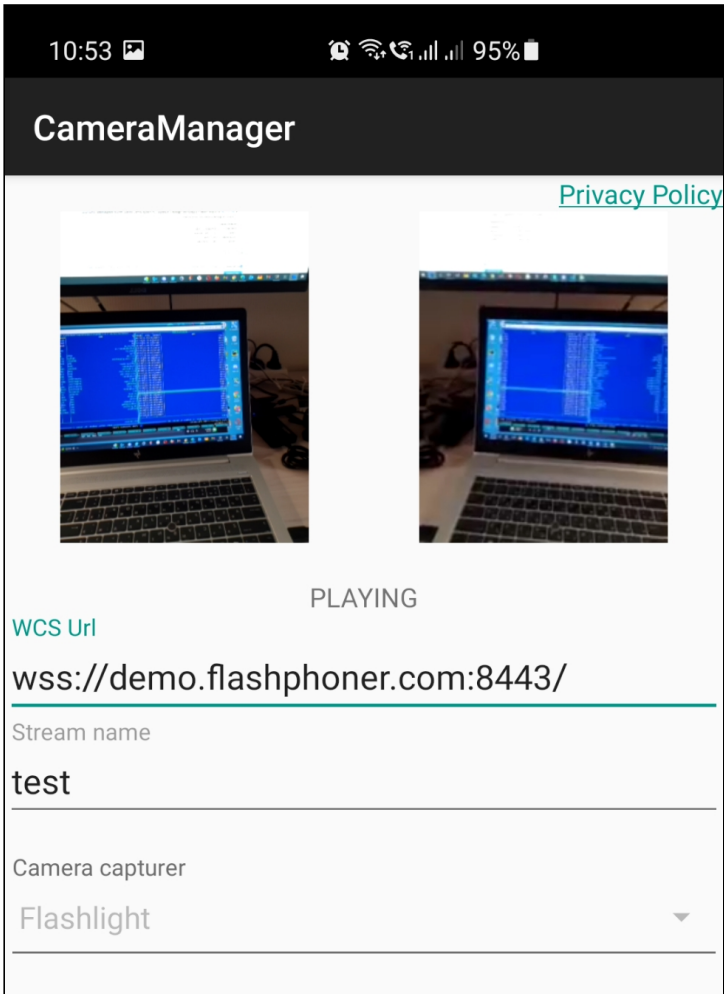
This example shows a different ways to use custom image capturer in one Android application. The example works since Android SDK build 1.1.0.42

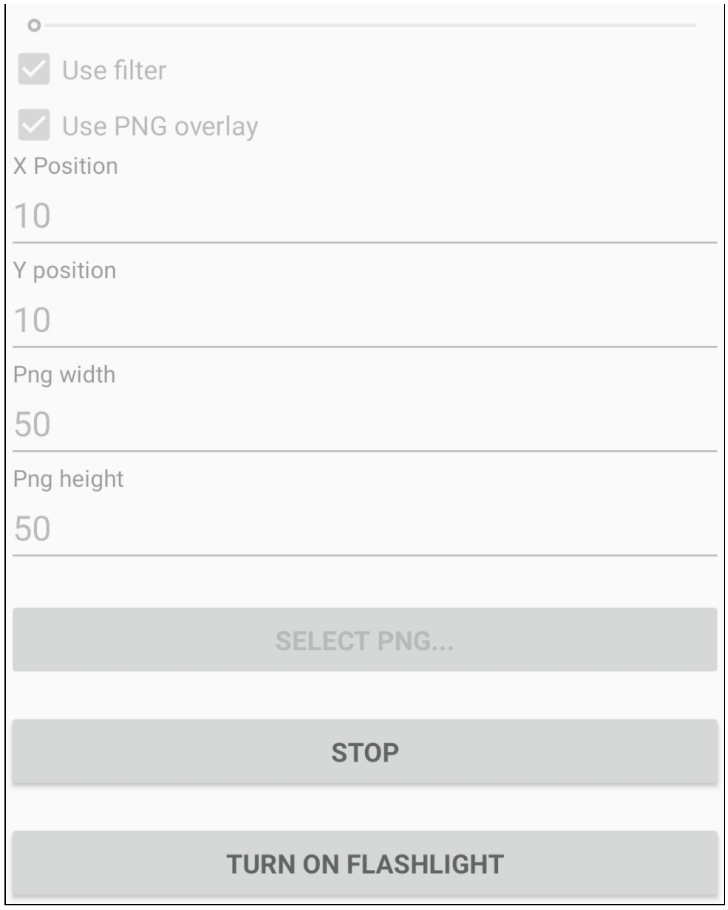
For all the screenshots:

- `WCS Url` - WCS address to establuish websocket connection
- `Stream name` - stream name to publish and play
- `Camera capturer` - choose a custom capturer example

Flashlight management screenshot:

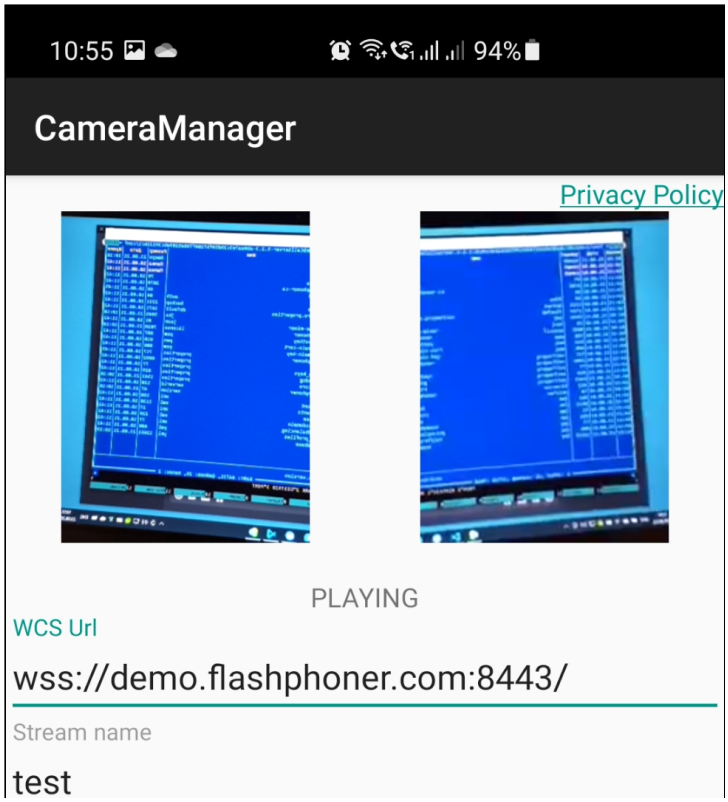
- `Turn on flashlight` - button to turn flashlight on and off

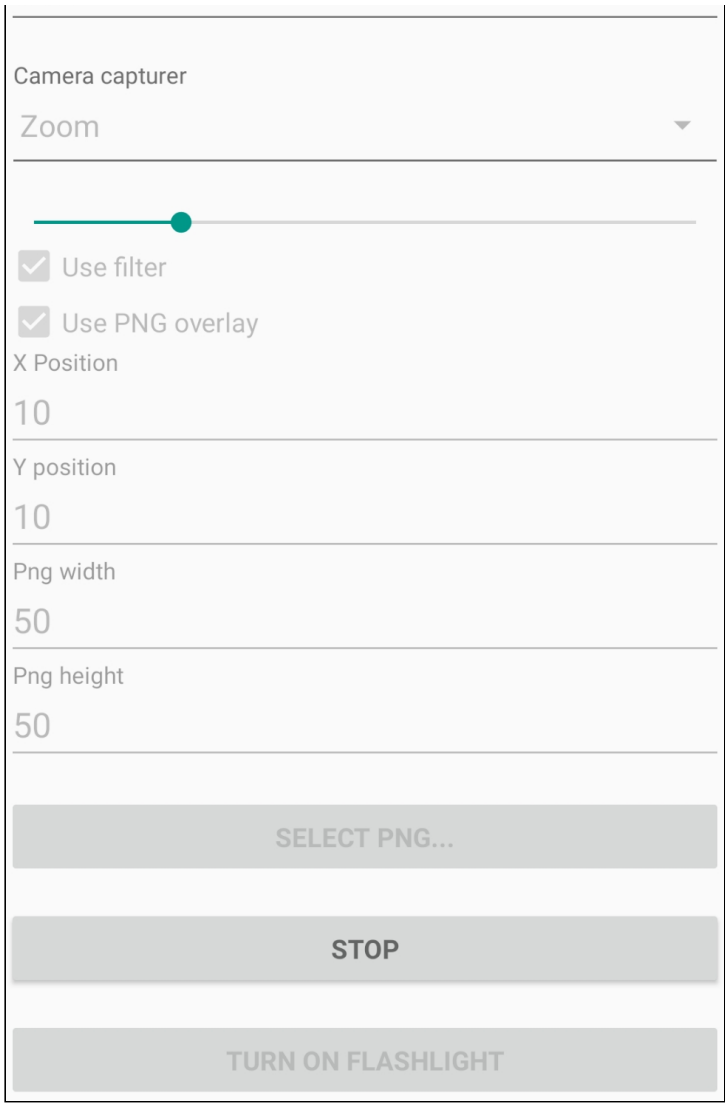




Zoom in/out management screenshot:

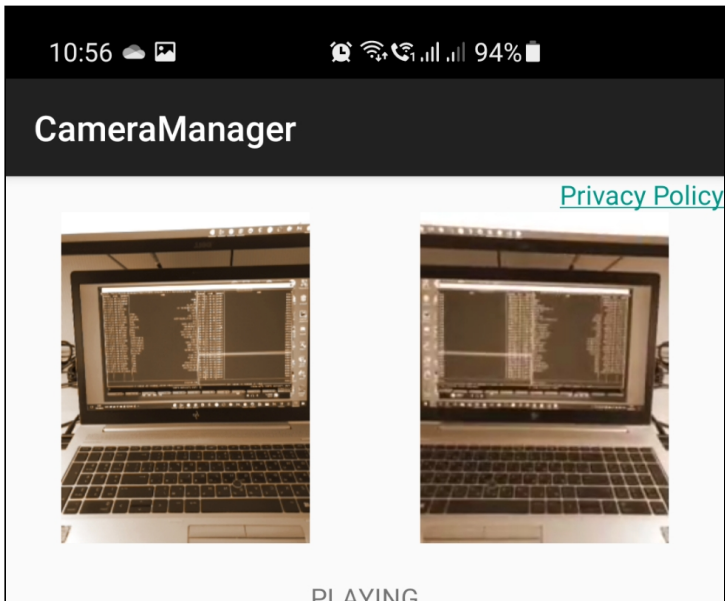
- use slider to zoom in and out

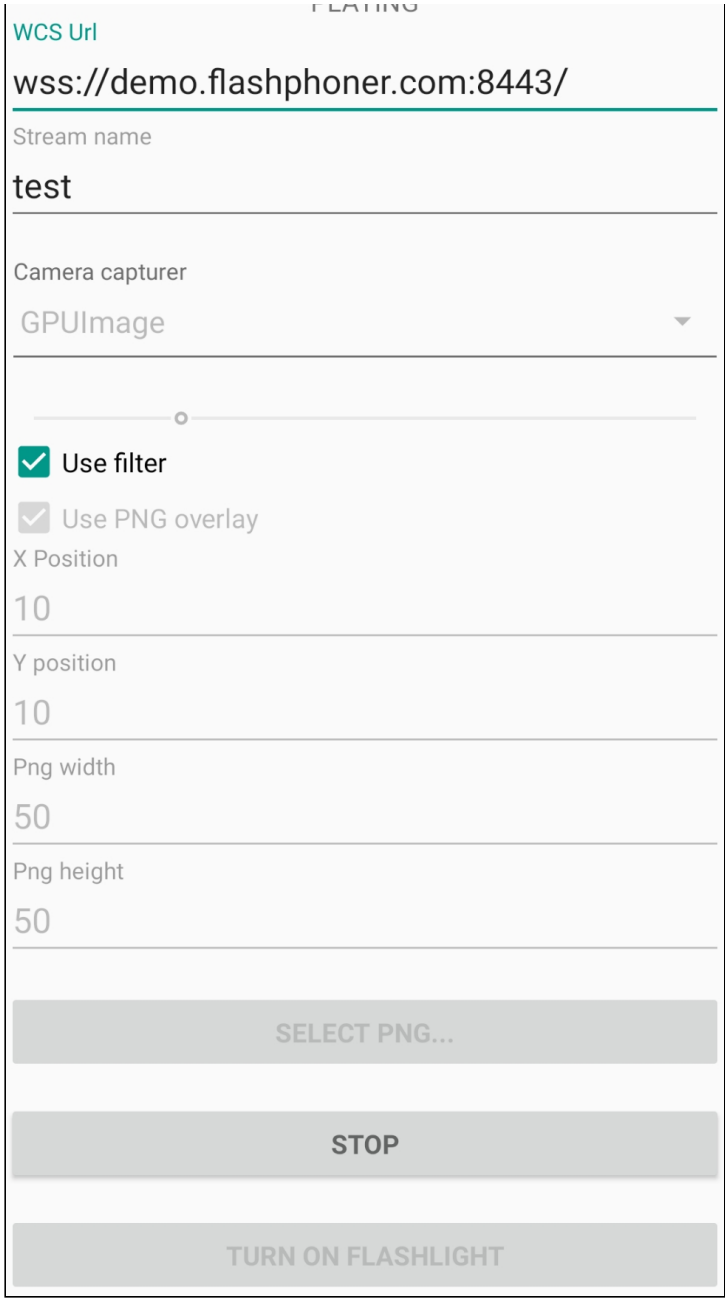




GPUImage `sepia` filter screenshot:

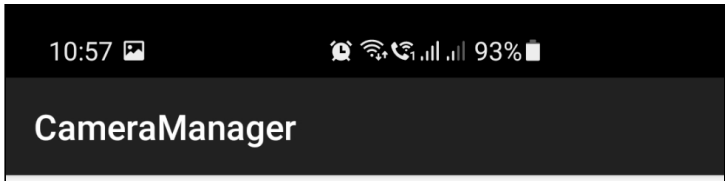
- `Use filter` - apply filter to the stream captured

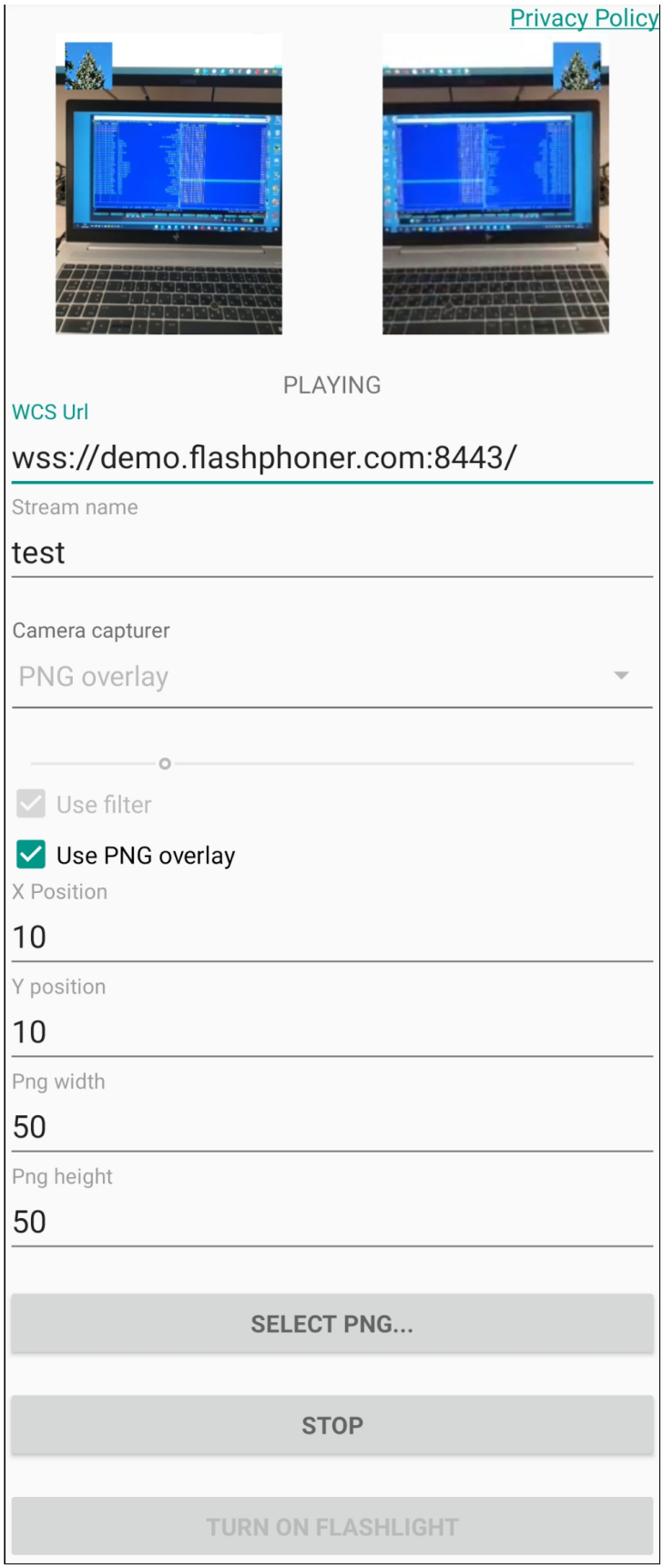




PNG overlay screenshot:

- `Select PNG` - button to select image from device gallery
- `Use PNG overlay` - apply PNG image to stream published
- `X Position`, `Y position` - top left corner coordinates to overlay image to, in pixels
- `Png width` - PNG picture width in frame, in pixels
- `Png height` - PNG picture height in frame, in pixels





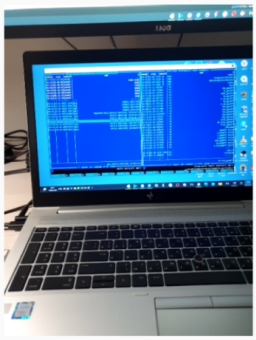
Resolution selection screenshot:

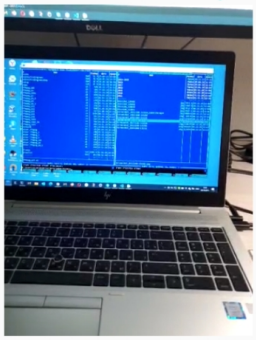
- `Camera resolution` - camera supported resolutions spinner

13:22 96%

CameraManager

Privacy Policy





PLAYING

WCS Url

wss://demo.flashphoner.com:8443/

Stream name

test

Camera capturer

Resolution

Width

640

Height

480

Camera resolution

640x480

Use filter

Use PNG overlay

X Position

10

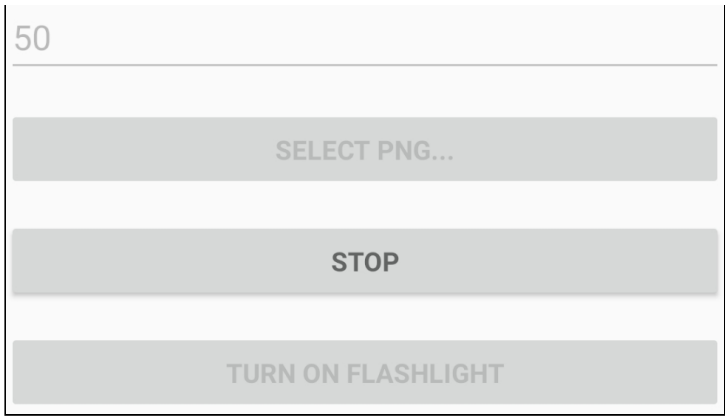
Y position

10

Png width

50

Png height



Analyzing the example code

To analyze the code use the following classes of `camera-manager` example which is available to download in build 1.1.0.47:

- main application activity class CameraManagerActivity.java
- `Camera1Capturer` implementation class for Zoom example ZoomCameraCapturer.java
- `Camera1Enumerator` implementation class for Zoom example ZoomCameraEnumerator.java
- `CameraSession` implementation class for Zoom example ZoomCameraSession.java
- `Camera1Capturer` implementation class for GPUImage example GPUImageCameraCapturer.java
- `Camera1Enumerator` implementation class for GPUImage example GPUImageCameraEnumerator.java
- `CameraSession` implementation class for GPUImage example GPUImageCameraSession.java
- `Camera1Capturer` implementation class for PngOverlay example PngOverlayCameraCapturer.java
- `Camera1Enumerator` implementation class for PngOverlay example PngOverlayCameraEnumerator.java
- `CameraSession` implementation class for PngOverlay example PngOverlayCameraSession.java
- `Camera1Capturer` implementation class for Resolution example ResolutionCameraCapturer.java
- `Camera1Enumerator` implementation class for Resolution example ResolutionCameraEnumerator.java
- `CameraSession` implementation class for Resolution example ResolutionCameraSession.java

Note that interfaces implementation classes are placed to `org.webrtc` package, it is necessary to access camera capturing and management functions

1. API initializing

`Flashphoner.init()` code

```
Flashphoner.init(this);
```

2. Websocket session creation

`Flashphoner.createSession()` code

The `SessionOptions` object with the following parameters is passing to the method:

- URL of WCS server
- `SurfaceViewRenderer localRenderer` to use to display a stream publishing (with changes applyed)
- `SurfaceViewRenderer remoteRenderer` to use to display the stream playing

```
sessionOptions = new SessionOptions(mWcsUrlView.getText().toString());
sessionOptions.setLocalRenderer(localRender);
sessionOptions.setRemoteRenderer(remoteRender);

/**
  * Session for connection to WCS server is created with method
createSession().
  */
session = Flashphoner.createSession(sessionOptions);
```

3. Connection establishing

`Session.connect()` code

```
session.connect(new Connection());
```

4. Receiving the event confirming successful connection

`session.onConnected()` code

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mStatusView.setText(connection.getStatus());
```

```
            ...
        }
    });
});
```

5. Getting back camera Id

`Flashphoner.getMediaDevices().getVideoList()`, `Flashphoner.getCameraEnumerator().isBackFacing()` code

```
int cameraId = 0;
List<MediaDevice> videoList = Flashphoner.getMediaDevices().getVideoList();
for (MediaDevice videoDevice : videoList) {
    String videoDeviceName = videoDevice.getLabel();
    if (Flashphoner.getCameraEnumerator().isBackFacing(videoDeviceName)) {
        cameraId = videoDevice.getId();
        break;
    }
}
```

6. Constraints setting and stream creation

`StreamOptions.setConstraints()`, `Session.createStream()` code

```
StreamOptions streamOptions = new StreamOptions(streamName);
VideoConstraints videoConstraints = new VideoConstraints();
videoConstraints.setVideoFps(25);
videoConstraints.setCameraId(cameraId);
Constraints constraints = new Constraints(true, true);
constraints.setVideoConstraints(videoConstraints);
streamOptions.setConstraints(constraints);

/**
 * Stream is created with method Session.createStream().
 */
publishStream = session.createStream(streamOptions);
```

7. Permissions requesting

`ActivityCompat.requestPermissions()` code

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            ...
            ActivityCompat.requestPermissions(StreamingMinActivity.this,
                  new String[]{Manifest.permission.RECORD_AUDIO,
Manifest.permission.CAMERA},
```

```
                    PUBLISH_REQUEST_CODE);
            ...
        }
        ...
    });
});
```

8. Stream publishing after permissions are granted

`Stream.publish()` code

```
@Override
 public void onRequestPermissionsResult(int requestCode,
                                        @NonNull String permissions[],
@NonNull int[] grantResults) {
     switch (requestCode) {
         case PUBLISH_REQUEST_CODE: {
             if (grantResults.length == 0 ||
                     grantResults[0] != PackageManager.PERMISSION_GRANTED ||
                     grantResults[1] != PackageManager.PERMISSION_GRANTED) {
                 muteButton();
                 session.disconnect();
                 Log.i(TAG, "Permission has been denied by user");
             } else {
                 /**
                  * Method Stream.publish() is called to publish stream.
                  */
                 publishStream.publish();
                 Log.i(TAG, "Permission has been granted by user");
             }
             break;
         }
         ...
     }
}
```

9. Stream playback after successful stream publishing

`Session.createStream()`, `Stream.play()` code

```
publishStream.on(new StreamStatusEvent() {
    @Override
    public void onStreamStatus(final Stream stream, final StreamStatus
streamStatus) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (StreamStatus.PUBLISHING.equals(streamStatus)) {
                    ...
                    /**
                     * The options for the stream to play are set.
                     * The stream name is passed when StreamOptions object is
created.
```

```
                     */
                    StreamOptions streamOptions = new
StreamOptions(streamName);
                    streamOptions.setConstraints(new Constraints(true,
true));

                    /**
                     * Stream is created with method Session.createStream().
                     */
                    playStream = session.createStream(streamOptions);
                    ...
                    /**
                     * Method Stream.play() is called to start playback of
the stream.
                     */
                    playStream.play();
                } else {
                    Log.e(TAG, "Can not publish stream " + stream.getName() +
" " + streamStatus);
                    onStopped();
                }
                mStatusView.setText(streamStatus.toString());
            }
        });
    }
});
```

10. Close connection

`Session.disconnect()` code

```
mStartButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        muteButton();
        if (mStartButton.getTag() == null ||
Integer.valueOf(R.string.action_start).equals(mStartButton.getTag())) {
            ...
        } else {
            /**
             * Connection to WCS server is closed with method
Session.disconnect().
             */
            session.disconnect();
        }
        ...
    }
});
```

11. Receiving the event confirming disconnection

`session.onDisconnection()` code

```
@Override
public void onDisconnection(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mStatusView.setText(connection.getStatus());
            mStatusView.setText(connection.getStatus());
            onStopped();
        }
    });
}
```

12. Example choosing

code

```
mCameraCapturer.setOnItemChosenListener(new
LabelledSpinner.OnItemChosenListener() {
    @Override
    public void onItemChosen(View labelledSpinner, AdapterView<?>
adapterView, View itemView, int position, long id) {
        String captureType =
getResources().getStringArray(R.array.camera_capturer)[position];
        switch (captureType) {
            case "Flashlight":
                changeFlashlightCamera();
                break;
            case "Zoom":
                changeZoomCamera();
                break;
            case "GPUImage":
                changeGpuImageCamera();
                break;
            case "PNG overlay":
                changePngOverlayCamera();
                break;
        }
    }

    @Override
    public void onNothingChosen(View labelledSpinner, AdapterView<?>
adapterView) {

    }
});
```

13. Camera type and custom camera options setting

code

```
private void changeFlashlightCamera() {

CameraCapturerFactory.getInstance().setCameraType(CameraCapturerFactory.CameraTy
```

```
    ...
}

private void changeZoomCamera() {

CameraCapturerFactory.getInstance().setCustomCameraCapturerOptions(zoomCameraCap


CameraCapturerFactory.getInstance().setCameraType(CameraCapturerFactory.CameraTy

    ...
}

private void changePngOverlayCamera() {

CameraCapturerFactory.getInstance().setCustomCameraCapturerOptions(pngOverlayCam


CameraCapturerFactory.getInstance().setCameraType(CameraCapturerFactory.CameraTy

    ...
}

private void changeGpuImageCamera() {

CameraCapturerFactory.getInstance().setCustomCameraCapturerOptions(gpuImageCamer


CameraCapturerFactory.getInstance().setCameraType(CameraCapturerFactory.CameraTy

    ...
}
```

14. Custom camera options for Zoom example

code

```
private CustomCameraCapturerOptions zoomCameraCapturerOptions = new
CustomCameraCapturerOptions() {

    private String cameraName;
    private CameraVideoCapturer.CameraEventsHandler eventsHandler;
    private boolean captureToTexture;

    @Override
    public Class<?>[] getCameraConstructorArgsTypes() {
        return new Class<?>[]{String.class,
CameraVideoCapturer.CameraEventsHandler.class, boolean.class};
    }

    @Override
    public Object[] getCameraConstructorArgs() {
        return new Object[]{cameraName, eventsHandler, captureToTexture};
    }
```

```
    @Override
    public void setCameraName(String cameraName) {
        this.cameraName = cameraName;
    }

    @Override
    public void setEventsHandler(CameraVideoCapturer.CameraEventsHandler
eventsHandler) {
        this.eventsHandler = eventsHandler;
    }

    @Override
    public void setCaptureToTexture(boolean captureToTexture) {
        this.captureToTexture = captureToTexture;
    }

    @Override
    public String getCameraClassName() {
        return "org.webrtc.ZoomCameraCapturer";
    }

    @Override
    public Class<?>[] getEnumeratorConstructorArgsTypes() {
        return new Class[0];
    }

    @Override
    public Object[] getEnumeratorConstructorArgs() {
        return new Object[0];
    }

    @Override
    public String getEnumeratorClassName() {
        return "org.webrtc.ZoomCameraEnumerator";
    }
};
```

15. Custom camera options for PngOverlay example

code

```
private CustomCameraCapturerOptions pngOverlayCameraCapturerOptions = new
CustomCameraCapturerOptions() {

    private String cameraName;
    private CameraVideoCapturer.CameraEventsHandler eventsHandler;
    private boolean captureToTexture;

    @Override
    public Class<?>[] getCameraConstructorArgsTypes() {
        return new Class<?>[]{String.class,
CameraVideoCapturer.CameraEventsHandler.class, boolean.class};
    }
```

```
    @Override
    public Object[] getCameraConstructorArgs() {
        return new Object[]{cameraName, eventsHandler, captureToTexture};
    }

    @Override
    public void setCameraName(String cameraName) {
        this.cameraName = cameraName;
    }

    @Override
    public void setEventsHandler(CameraVideoCapturer.CameraEventsHandler
eventsHandler) {
        this.eventsHandler = eventsHandler;
    }

    @Override
    public void setCaptureToTexture(boolean captureToTexture) {
        this.captureToTexture = captureToTexture;
    }

    @Override
    public String getCameraClassName() {
        return "org.webrtc.PngOverlayCameraCapturer";
    }

    @Override
    public Class<?>[] getEnumeratorConstructorArgsTypes() {
        return new Class[0];
    }

    @Override
    public Object[] getEnumeratorConstructorArgs() {
        return new Object[0];
    }

    @Override
    public String getEnumeratorClassName() {
        return "org.webrtc.PngOverlayCameraEnumerator";
    }
};
```

16. Custom camera options for GPUImage example

code

```
private CustomCameraCapturerOptions gpuImageCameraCapturerOptions = new
CustomCameraCapturerOptions() {

    private String cameraName;
    private CameraVideoCapturer.CameraEventsHandler eventsHandler;
    private boolean captureToTexture;

    @Override
    public Class<?>[] getCameraConstructorArgsTypes() {
```

```
        return new Class<?>[]{String.class,
CameraVideoCapturer.CameraEventsHandler.class, boolean.class};
    }

    @Override
    public Object[] getCameraConstructorArgs() {
        return new Object[]{cameraName, eventsHandler, captureToTexture};
    }

    @Override
    public void setCameraName(String cameraName) {
        this.cameraName = cameraName;
    }

    @Override
    public void setEventsHandler(CameraVideoCapturer.CameraEventsHandler
eventsHandler) {
        this.eventsHandler = eventsHandler;
    }

    @Override
    public void setCaptureToTexture(boolean captureToTexture) {
        this.captureToTexture = captureToTexture;
    }

    @Override
    public String getCameraClassName() {
        return "org.webrtc.GPUImageCameraCapturer";
    }

    @Override
    public Class<?>[] getEnumeratorConstructorArgsTypes() {
        return new Class[0];
    }

    @Override
    public Object[] getEnumeratorConstructorArgs() {
        return new Object[0];
    }

    @Override
    public String getEnumeratorClassName() {
        return "org.webrtc.GPUImageCameraEnumerator";
    }
};
```

17. Custom camera options for Resolution example

code

```
private CustomCameraCapturerOptions resolutionCameraCapturerOptions = new
CustomCameraCapturerOptions() {

    private String cameraName;
    private CameraVideoCapturer.CameraEventsHandler eventsHandler;
```

```
    private boolean captureToTexture;

    @Override
    public Class<?>[] getCameraConstructorArgsTypes() {
        return new Class<?>[]{String.class,
CameraVideoCapturer.CameraEventsHandler.class, boolean.class};
    }

    @Override
    public Object[] getCameraConstructorArgs() {
        return new Object[]{cameraName, eventsHandler, captureToTexture};
    }

    @Override
    public void setCameraName(String cameraName) {
        this.cameraName = cameraName;
    }

    @Override
    public void setEventsHandler(CameraVideoCapturer.CameraEventsHandler
eventsHandler) {
        this.eventsHandler = eventsHandler;
    }

    @Override
    public void setCaptureToTexture(boolean captureToTexture) {
        this.captureToTexture = captureToTexture;
    }

    @Override
    public String getCameraClassName() {
        return "org.webrtc.ResolutionCameraCapturer";
    }

    @Override
    public Class<?>[] getEnumeratorConstructorArgsTypes() {
        return new Class[0];
    }

    @Override
    public Object[] getEnumeratorConstructorArgs() {
        return new Object[0];
    }

    @Override
    public String getEnumeratorClassName() {
        return "org.webrtc.ResolutionCameraEnumerator";
    }
};
```

18. Turning on flashlight

`Flashphoner.turnOnFlashlight()` code

```
private void turnOnFlashlight() {
    if (Flashphoner.turnOnFlashlight()) {

mSwitchFlashlightButton.setText(getResources().getString(R.string.turn_off_flash

        flashlight = true;
    }
}
```

19. Turning off flashlight

`Flashphoner.turnOffFlashlight()` code

```
private void turnOffFlashlight() {
    Flashphoner.turnOffFlashlight();

mSwitchFlashlightButton.setText(getResources().getString(R.string.turn_on_flashl

    flashlight = false;
}
```

20. Zoom in/out management with slider

`ZoomCameraCapturer.setZoom()` code

```
mZoomSeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
{
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        CameraVideoCapturer cameraVideoCapturer =
CameraCapturerFactory.getInstance().getCameraVideoCapturer();
        if (cameraVideoCapturer instanceof ZoomCameraCapturer) {
            ((ZoomCameraCapturer) cameraVideoCapturer).setZoom(progress);
        }
    }
    ...
});
```

21. Overlaying a pucture to a stream with permission request.

`PngOverlayCameraCapturer.setPicture()` code

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        InputStream inputStream = null;
```

```
        try {
            inputStream =
CameraManagerActivity.this.getBaseContext().getContentResolver().openInputStream

        } catch (FileNotFoundException e) {
            Log.e(TAG, "Can't select picture: " + e.getMessage());
        }
        picture = BitmapFactory.decodeStream(inputStream);
    }

    CameraVideoCapturer cameraVideoCapturer =
CameraCapturerFactory.getInstance().getCameraVideoCapturer();
    if (cameraVideoCapturer instanceof PngOverlayCameraCapturer && picture !=
null) {
        ((PngOverlayCameraCapturer) cameraVideoCapturer).setPicture(picture);
    }
}
```

22. Choosing camera resolution

code

```
mCameraResolutionSpinner = (LabelledSpinner)
findViewById(R.id.camera_resolution_spinner);
mCameraResolutionSpinner.setOnItemChosenListener(new
LabelledSpinner.OnItemChosenListener() {

    @Override
    public void onItemChosen(View labelledSpinner, AdapterView<?>
adapterView, View itemView, int position, long id) {
        String resolution = adapterView.getSelectedItem().toString();
        if (resolution.isEmpty()) {
            return;
        }
        setResolutions(resolution);
    }

    @Override
    public void onNothingChosen(View labelledSpinner, AdapterView<?>
adapterView) {

    }
});

...
private void setResolutions(String resolutionStr) {
    String[] resolution = resolutionStr.split("x");
    mWidth.setText(resolution[0]);
    mHeight.setText(resolution[1]);
}
```

23. Camera session creation in `ZoomCameraCapturer` class

`CameraSession.create()` code

```
@Override
protected void createCameraSession(CameraSession.CreateSessionCallback
createSessionCallback, CameraSession.Events events, Context
applicationContext, SurfaceTextureHelper surfaceTextureHelper, String
cameraName, int width, int height, int framerate) {
    CameraSession.CreateSessionCallback myCallback = new
CameraSession.CreateSessionCallback() {
        @Override
        public void onDone(CameraSession cameraSession) {
            ZoomCameraCapturer.this.cameraSession = (ZoomCameraSession)
cameraSession;
            createSessionCallback.onDone(cameraSession);
        }

        @Override
        public void onFailure(CameraSession.FailureType failureType, String
s) {
            createSessionCallback.onFailure(failureType, s);
        }
    };

    ZoomCameraSession.create(myCallback, events, captureToTexture,
applicationContext, surfaceTextureHelper,
Camera1Enumerator.getCameraIndex(cameraName), width, height, framerate);
}
```

24. Zoom in/out value setting in `ZoomCameraCapturer` class

`CameraSession.setZoom()` code

```
public boolean setZoom(int value) {
    return cameraSession.setZoom(value);
}
```

25. Byte buffer allocation for camera images data in `ZoomCameraSession` class

code

```
if (!captureToTexture) {
    int frameSize = captureFormat.frameSize();

    //The implementation is taken from the WebRTC library, so the purpose of
the three buffers is not entirely known
    for(int i = 0; i < 3; ++i) {
        ByteBuffer buffer = ByteBuffer.allocateDirect(frameSize);
        camera.addCallbackBuffer(buffer.array());
    }
}
```

26. Реализация изменение масштаба в классе `ZoomCameraSession`

code

```
public boolean setZoom(int value) {
    if (!isCameraActive() && camera.getParameters().isZoomSupported()) {
        return false;
    }

    Camera.Parameters parameters = camera.getParameters();
    parameters.setZoom(value);
    camera.setParameters(parameters);
    return true;
}
```

27. Setting to apply filter in `GPUImageCameraSession` class

code

```
public void setUsedFilter(boolean usedFilter) {
    isUsedFilter = usedFilter;
}
```

28. Filter applying to image data from camera buffer

code

```
private void listenForBytebufferFrames() {
    this.camera.setPreviewCallbackWithBuffer(new Camera.PreviewCallback() {
        public void onPreviewFrame(byte[] data, Camera callbackCamera) {
            GPUImageCameraSession.this.checkIsOnCameraThread();
            if (callbackCamera != GPUImageCameraSession.this.camera) {
                Logging.e(TAG,
CALLBACK_FROM_A_DIFFERENT_CAMERA_THIS_SHOULD_NEVER_HAPPEN);
            } else if (GPUImageCameraSession.this.state !=
GPUImageCameraSession.SessionState.RUNNING) {
                Logging.d(TAG,
BYTEBUFFER_FRAME_CAPTURED_BUT_CAMERA_IS_NO_LONGER_RUNNING);
            } else {
                ...
                applyFilter(data,
GPUImageCameraSession.this.captureFormat.width,
GPUImageCameraSession.this.captureFormat.height);

                VideoFrame.Buffer frameBuffer = new NV21Buffer(data,
GPUImageCameraSession.this.captureFormat.width,
GPUImageCameraSession.this.captureFormat.height, () -> {
                    GPUImageCameraSession.this.cameraThreadHandler.post(() ->
{
                        if (GPUImageCameraSession.this.state ==
GPUImageCameraSession.SessionState.RUNNING) {

GPUImageCameraSession.this.camera.addCallbackBuffer(data);
                        }
```

```
                    });
                });
                VideoFrame frame = new VideoFrame(frameBuffer,
GPUImageCameraSession.this.getFrameOrientation(), captureTimeNs);

GPUImageCameraSession.this.events.onFrameCaptured(GPUImageCameraSession.this,
frame);
                frame.release();
            }
        }
    });
}
```

29. Filter implementation

code

```
private void initFilter(int width, int height) {
    filter = new GPUImageMonochromeFilter();
    filter.setColor(0,0,0);

    renderer = new GPUImageRenderer(filter);
    renderer.setRotation(Rotation.NORMAL, false, false);
    renderer.setScaleType(GPUImage.ScaleType.CENTER_INSIDE);

    buffer = new PixelBuffer(width, height);
    buffer.setRenderer(renderer);
}

private void destroyFilter() {
    filter.destroy();
    buffer.destroy();
}

private void applyFilter(byte[] data, int width, int height) {
    if (!isUsedFilter) {
        return;
    }

    renderer.onPreviewFrame(data, width, height);
    Bitmap newBitmapRgb = buffer.getBitmap();
    byte[] dataYuv = Utils.getNV21(width, height, newBitmapRgb);
    System.arraycopy(dataYuv, 0, data, 0, dataYuv.length);
}
```

30. Setting picture bitmap to overlay in `PngOverlayCameraCapturer` class

code

```
public void setPicture(Bitmap picture) {
    if (cameraSession != null) {
        cameraSession.setPicture(picture);
```

```
    }
}
```

31. Picture data overlaying to camera image data

code

```
private void listenForBytebufferFrames() {
    this.camera.setPreviewCallbackWithBuffer(new Camera.PreviewCallback() {
        public void onPreviewFrame(byte[] data, Camera callbackCamera) {
            PngOverlayCameraSession.this.checkIsOnCameraThread();
            if (callbackCamera != PngOverlayCameraSession.this.camera) {
                Logging.e(TAG,
CALLBACK_FROM_A_DIFFERENT_CAMERA_THIS_SHOULD_NEVER_HAPPEN);
            } else if (PngOverlayCameraSession.this.state !=
PngOverlayCameraSession.SessionState.RUNNING) {
                Logging.d(TAG,
BYTEBUFFER_FRAME_CAPTURED_BUT_CAMERA_IS_NO_LONGER_RUNNING);
            } else {
                ...
                insertPicture(data,
PngOverlayCameraSession.this.captureFormat.width,
PngOverlayCameraSession.this.captureFormat.height);

                VideoFrame.Buffer frameBuffer = new NV21Buffer(data,
PngOverlayCameraSession.this.captureFormat.width,
PngOverlayCameraSession.this.captureFormat.height, () -> {
                    PngOverlayCameraSession.this.cameraThreadHandler.post(()
-> {
                        if (PngOverlayCameraSession.this.state ==
PngOverlayCameraSession.SessionState.RUNNING) {

PngOverlayCameraSession.this.camera.addCallbackBuffer(data);
                        }

                    });
                });
                VideoFrame frame = new VideoFrame(frameBuffer,
PngOverlayCameraSession.this.getFrameOrientation(), captureTimeNs);

PngOverlayCameraSession.this.events.onFrameCaptured(PngOverlayCameraSession.this
 frame);
                frame.release();
            }
        }
    });
}
```

32. Picture overlaying implementation

code

```
private void insertPicture(byte[] data, int width, int height) {
    if (picture == null || !isUsedPngOverlay) {
        return;
    }

    Bitmap scaledPicture = rescalingPicture();

    int [] pngArray = new int[scaledPicture.getHeight() *
scaledPicture.getWidth()];
    scaledPicture.getPixels(pngArray, 0, scaledPicture.getWidth(), 0, 0,
scaledPicture.getWidth(), scaledPicture.getHeight());

    int [] rgbData = new int [width * height];
    GPUImageNativeLibrary.YUVtoARBG(data, width, height, rgbData);

    int pictureW = scaledPicture.getWidth();
    int pictureH = scaledPicture.getHeight();

    for (int c = 0; c < pngArray.length; c++) {
        int pictureColumn = c / pictureW;
        int pictureLine = c - pictureColumn * pictureW;
        int index = (pictureLine * width) + pictureColumn + startX * width +
startY;

        if (index >= data.length) {
            break;
        }
        rgbData[index] = pngArray[c];
    }

    byte[] yuvData = Utils.getNV21(width, height, rgbData);
    System.arraycopy(yuvData, 0, data, 0, yuvData.length);
}
```

33. Getting resolutions supported list

`ResolutionCameraCapturer.getSupportedResolutions()` code

```
public List<Camera.Size> getSupportedResolutions() {
    Camera camera =
Camera.open(Camera1Enumerator.getCameraIndex(cameraName));
    List ret = Collections.EMPTY_LIST;
    if (camera != null) {
        ret = camera.getParameters().getSupportedVideoSizes();
        camera.release();
    }

    return ret;
}
```