CameraX: Zoom API

Status: Draft
Created: 2019-6-13

Objective

Provides a zoom API that is simple and intuitive for developers. Developers should be able to implement Pinch-To-Zoom or zoom slider with just a few lines of code using this API. Currently it only supports digital zoom. CameraX will expand the API to have optical zoom support in the future.

Background

Camera2 uses SCALER_CROP_REGION for digital zoom in order to have off-axis zoom capability. The crop region is in sensor active array coordinates, which is consistent across all the coordinates in the Camera2 API. While this is powerful, precise and consistent, it is not intuitive for developers to understand the correct usage.

CameraX solves the problem by providing an intuitive high level zoom API that allows for zoom UIs to be implemented quickly.

Overview

This API will be exposed at Camera Control. Developer can get the CameraControl via

```
CameraX.getCameraControl(LensFacing);
```

CameraX zoom API is around the concept of "**Zoom Multiplier**". It is equal to the X-number in camera terminology. In optical zoom camera, it is the ratio of current focal length and minimal focal length. It also means the multiplier of the image magnification. For example, 2.0 = 2X magnification and 4.0 = 4X magnification. Please note that currently the API only supports digital zoom, which means there is no focal length concept.

Below are all zoom APIs:

```
// Sets current zoom multiplier , default is 1.0.
// if the multiplier is larger than max zoom , it'll set the max zoom.
// if the multiplier is smaller than min zoom, it'll set the min zoom.
// This method returns a ListenableFuture, apps can get the result asynchronously.
```

```
public ListenableFuture<Boolean> setZoom(float multiplier);
// Gets maximum zoom multiplier.
public float getMaxZoom();
public LiveData<float> getMaxZoomLiveData();

// Gets minimum zoom multiplier.
public float getMinZoom();
public LiveData<float> getMinZoomLiveData();
// Gets current zoom multiplier.
public float getZoom();
public LiveData<float> getZoomLiveData();
```

Detailed Design

setZoom/getZoom

Apps call setZoom() to set the zoom multiplier. It returns a ListenableFuture<Boolean>. If apps need to access the results, they can process the result asynchronously. The ListenableFuture is set when the capture result in repeating request contains the requested crop region.

getZoom() is used to get current zoom multiplier. While getZoom() returns immediate value, the alternative getZoomLiveData() returns a LiveData<float> which allows apps to be notified each time the value changes.

Max/Min zoom multiplier

getMaxZoom() / getMinZoom() will return the max and min zoom multipliers. The min zoom multiplier should be 1.0 for now, and max zoom multiplier is equal to the Camera2 characteristic SCALER_AVAILABLE_MAX_DIGITAL_ZOOM.

Just like getZoomLiveData(), getMaxZoomLiveData() / getMinZoomLiveData() returns a LiveData<float> which apps can observe with their activity/fragment to update the UI every time the value changes.

On devices that don't support zoom, getMaxZoom() returns the same value as getMinZoom().

Uniform zoom

There are concerns that using multiplier will cause FOV(Field of View) width/height to not be able to be changed linearly as the multiplier changes. For example, suppose that sensor region width is 1000. When the multiplier changes from 1.0 to 2.0, the crop region's width changes from 1000 to 500. But when multiplier changes from 4.0 to 5.0, the crop region's width changes from 250 to 200. The width changes are varied significantly.

The issue happens in zoom slider UI. To fix the problem, instead of calculating multiplier by interpolating between max and min zoom, apps should calculate the FOV width/height first to guarantee the FOV width/height is distributed linearly , and then get the multiplier and set the zoom. Below are sample codes for getting uniform zoom multiplier.

```
// ratio is between 0 to 1 ,  0 = min zoom, 1 = max zoom
public float getUniformZoomMultiplier(float ratio) {
    // This crop width is proportional to the real crop width.
    // The real crop with = sensorWidth/ zoomMultiplier,  but we don't need the
    // the real crop with, so we can assume sensorWidth as 1.0f.
    float cropWidthInMaxZoom = 1.0f / getMaxZoom();
    float cropWidthInMinZoom = 1.0f / getMinZoom();

    float cropWidth = cropWidthInMinZoom + (cropWidthInMaxZoom -cropWidthInMinZoom)
                      * (ratio);

    float multiplier = 1.0f / cropWidth;

    Return multiplier;
}
```

Below is sample code that gets the adjusted ratio by given zoom multiplier by which an app can get the position of the slider by the given multiplier.

```
public float getUniformRatioByMultiplier(float multiplier) {
    float cropWidth = 1.0f / multiplier;
    float cropWidthInMaxZoom = 1.0f / getMaxZoom();
    float cropWidthInMinZoom = 1.0f / getMinZoom();

    return (cropWidth - cropWidthInMinZoom) / (cropWidthInMaxZoom -
           cropWidthInMinZoom);
}
```

Although adding these methods to our API helps developers implement the uniform zoom slider quickly, it comes at a cost that some developers are confused why there are two ways to specify the zoom especially for those who don't use slider for zoom. Hence we remove it from our API to

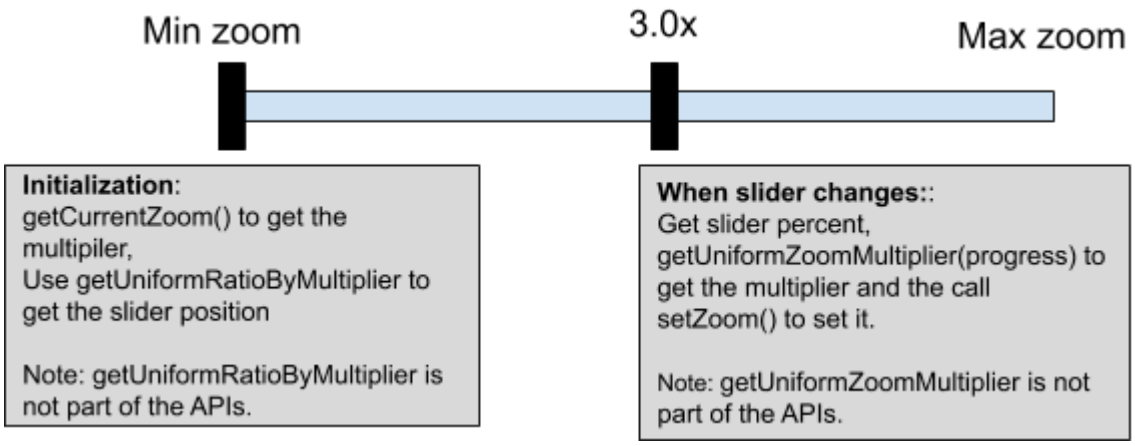
keep it simple. We can consider using blogs / sample code / Stack Overflow to communicate the tricks.

ZOOM UI

One of the goals of this new zoom API is to help application implement below 2 zoom UIs easily:

Zoom UI 1: using slider

To implement slider with the uniform/linear zoom effect, apps can use the uniform zoom concept or sample code from the previous section. Below illustrates how an app can implement it by our API and the sample code.



Zoom UI 2: pinch-to-zoom

Developers can use *ScaleGestureDetector* to implement pinch-to-zoom as below: detector.getScaleFactor() returns the scaling factor from the previous scale event to the current Event. Apps can simply multiply the detector.getScaleFactor() with current zoom multiplier and set that as the new zoom multiplier.

```
new ScaleGestureDetector.OnScaleGestureListener() {
     @Override
     public boolean onScale(ScaleGestureDetector detector) {
         float multiplier = cameraControl.getZoomMultiplier() *
               detector.getScaleFactor();
         cameraControl.setZoomMultiplier(multiplier);
     }
}
```

Revision History

version	date	description
1.0	2019/6/13	Initial version
2.0	2019/6/21	● Removed uniform zoom API. Adds uniform zoom sample codes ● Shortened the method name ● setZoom() now returns a ListenableFuture ● getCurrentZoom()/getMaxZoom()/getMinZoom() can now returns a LiveData<Float> for apps to observe.