Frame

`public class Frame`

Captures the state and changes to the AR system from a call to **`Session.update()`** (/ar/reference/java/com/google/ar/core/Session#update()).

Public Methods

`Image` (https://developer.android.com/reference/android/media/Image)	**`acquireCameraImage`** (/ar/reference Attempts to acquire an image from the
`Image` (https://developer.android.com/reference/android/media/Image)	**`acquireDepthImage`** (/ar/reference/j Attempts to acquire a depth Android Im
`PointCloud` (/ar/reference/java/com/google/ar/core/PointCloud)	**`acquirePointCloud`** (/ar/reference/j Acquires the current set of estimated 3
`Image` (https://developer.android.com/reference/android/media/Image)	**`acquireRawDepthConfidenceImag`** Attempts to acquire the confidence And
`Image` (https://developer.android.com/reference/android/media/Image)	**`acquireRawDepthImage`** (/ar/referen Attempts to acquire a "raw", mostly unfi
`long`	**`getAndroidCameraTimestamp`** (/ar/ Returns the (Android Camera timestam
`Pose` (/ar/reference/java/com/google/ar/core/Pose)	**`getAndroidSensorPose`** (/ar/referen Returns the pose of the Android Sensor
`Camera` (/ar/reference/java/com/google/ar/core/Camera)	**`getCamera`** (/ar/reference/java/com/g Returns the **`Camera`** (/ar/reference/java
`int`	**`getCameraTextureName`** (/ar/referen Returns the OpenGL ES camera texture
`ImageMetadata` (/ar/reference/java/com/google/ar/core/ImageMetadata)	**`getImageMetadata`** (/ar/reference/ja Returns the camera metadata for the cu
`LightEstimate` (/ar/reference/java/com/google/ar/core/LightEstimate)	**`getLightEstimate`** (/ar/reference/ja Returns the current ambient light estim
`long`	**`getTimestamp`** (/ar/reference/java/co Returns the timestamp in nanoseconds
`Collection` (https://developer.android.com/reference/java/util/Collection) <**`Anchor`** (/ar/reference/java/com/google/ar/core/Anchor)>	**`getUpdatedAnchors`** (/ar/reference/j Returns the anchors that were changed

Collection (https://developer.android.com/reference/java/util/Collection) <TrackData (/ar/reference/java/com/google/ar/core/TrackData)>	getUpdatedTrackData (/ar/referenc Retrieve all track data that was written t
<T extends Trackable (/ar/reference/java/com/google/ar/core/Trackable)> Collection (https://developer.android.com/reference/java/util/Collection) <T>	getUpdatedTrackables (/ar/referen filterType) Returns the trackables of a particular ty
boolean	hasDisplayGeometryChanged (/ar/ Checks if the display rotation or viewpo
List (https://developer.android.com/reference/java/util/List) <HitResult (/ar/reference/java/com/google/ar/core/HitResult) >	hitTest (/ar/reference/java/com/goo Similar to hitTest(float, float) (https://developer.android.com/referer
List (https://developer.android.com/reference/java/util/List) <HitResult (/ar/reference/java/com/google/ar/core/HitResult) >	hitTest (/ar/reference/java/com/goo directionOffset) Similar to hitTest(float, float)
List (https://developer.android.com/reference/java/util/List) <HitResult (/ar/reference/java/com/google/ar/core/HitResult) >	hitTest (/ar/reference/java/com/goo Performs a ray cast from the user's dev
List (https://developer.android.com/reference/java/util/List) <HitResult (/ar/reference/java/com/google/ar/core/HitResult) >	hitTestInstantPlacement (/ar/ref approximateDistanceMeters) Performs a ray cast that can return a re
void	recordTrackData (/ar/reference/java trackId, ByteBuffer (https://deve Writes a data sample in the specified tr
void	transformCoordinates2d (/ar/reference/java/com/google/ar/co (/ar/reference/java/com/google/ar/co outputCoordinates, float[] ou Transforms a list of 2D coordinates fro
void	transformCoordinates2d (/ar/reference/java/com/google/ar/co (Coordinates2d (/ar/reference/java/ inputVertices2d, Coordinates2 (https://developer.android.com/referer Transforms a list of 2D coordinates fro
void	transformDisplayUvCoords (/ar/re (https://developer.android.com/referer *This method is deprecated. Replaced by*

Inherited Methods

From `class java.lang.Object`

Object (https://developer.android.com/reference/java/lang/Object)	`clone()`
`boolean`	`equals(`**Object** (https://developer.android.com/reference/ja `arg0)`
`void`	`finalize()`
`final` **Class** (https://developer.android.com/reference/java/lang/Class) `<?>`	`getClass()`
`int`	`hashCode()`
`final void`	`notify()`
`final void`	`notifyAll()`
String (https://developer.android.com/reference/java/lang/String)	`toString()`
`final void`	`wait(long arg0, int arg1)`
`final void`	`wait(long arg0)`
`final void`	`wait()`

Public Methods

acquireCameraImage

`public` **Image** (https://developer.android.com/reference/android/media/Image) `acquireCameraIma`

Attempts to acquire an image from the camera that corresponds to the current frame. Depending on device performance, can throw **`NotYetAvailableException`**

(/ar/reference/java/com/google/ar/core/exceptions/NotYetAvailableException) for several frames after session start, and for a few frames at a time while the session is running.

Details	
Returns	an Android Image object (https://developer.android.com/reference/android/media/Image) that contains the image data from the camera. The returned image object format is **`AIMAGE_FORMAT_YUV_420_888`** (https://developer.android.com/ndk/reference/group/media#group___media_1gga9c3dace30485a0f28163a882a5d65a19aea9797f9b5db5d26a2055a43d8491890) .
Throws	**`NullPointerException`** (https://developer.android.com/reference/java/lang/NullPointerEx
	`DeadlineExceededException` (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc
	`ResourceExhaustedException` (/ar/reference/java/com/google/ar/core/exceptions/ResourceExh
	`NotYetAvailableException` (/ar/reference/java/com/google/ar/core/exceptions/NotYetAvaila

acquireDepthImage

`public` **`Image`** (https://developer.android.com/reference/android/media/Image) `acquireDepthImag`

Attempts to acquire a depth Android Image object (https://developer.android.com/reference/android/media/Image) that corresponds to the current frame.

The depth image has a single 16-bit plane at index 0, stored in little-endian format. Each pixel contains the distance in millimeters to the camera plane. Currently, the three most significant bits are always set to 000. The remaining thirteen bits express values from 0 to 8191, representing depth in millimeters. To extract distance from a depth map, see the

Depth API developer guide (https://developers.google.com/ar/develop/java/depth/developer-guide#extract-distance).

The actual size of the depth image depends on the device and its display aspect ratio. The size of the depth image is typically around 160x120 pixels, with higher resolutions up to 640x480 on some devices. These sizes may change in the future. The outputs of `acquireDepthImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireDepthImage()), `acquireRawDepthImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthImage()) and `acquireRawDepthConfidenceImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthConfidenceImage()) will all have the exact same size.

Optimal depth accuracy occurs between 500 millimeters (50 centimeters) and 5000 millimeters (5 meters) from the camera. Error increases quadratically as distance from the camera increases.

Depth is estimated using data from the world-facing cameras, user motion, and hardware depth sensors such as a time-of-flight sensor (or ToF sensor) if available. As the user moves their device through the environment, 3D depth data is collected and cached which improves the quality of subsequent depth images and reducing the error introduced by camera distance.

If an up-to-date depth image isn't ready for the current frame, the most recent depth image available from an earlier frame will be returned instead. This is expected only to occur on compute-constrained devices. An up-to-date depth image should typically become available again within a few frames.

The image must be released via `Image.close()` (https://developer.android.com/reference/android/media/Image#close()) once it is no longer needed.

Details	
Returns	The depth image corresponding to the frame.

Details	
Throws	**NotYetAvailableException** (/ar/reference/java/com/google/ar/core/exceptions/NotYetAvaila
	NotTrackingException (/ar/reference/java/com/google/ar/core/exceptions/NotTracking
	IllegalStateException (https://developer.android.com/reference/java/lang/IllegalStateEx
	ResourceExhaustedException (/ar/reference/java/com/google/ar/core/exceptions/ResourceExh
	DeadlineExceededException (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc

acquirePointCloud

`public` **`PointCloud`** (/ar/reference/java/com/google/ar/core/PointCloud) `acquirePointCloud()`

Acquires the current set of estimated 3d points attached to real-world geometry. **`PointCloud.release()`** (/ar/reference/java/com/google/ar/core/PointCloud#release()) must be called after application is done using the PointCloud object.

Note: This information is for visualization and debugging purposes only. Its characteristics and format are subject to change in subsequent versions of the API.

Details	
Throws	**ResourceExhaustedException** (/ar/reference/java/com/google/ar/core/exceptions/ResourceExh
	DeadlineExceededException (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc

acquireRawDepthConfidenceImage

`public` **`Image`** (https://developer.android.com/reference/android/media/Image) `acquireRawDepthC`

Attempts to acquire the confidence Android Image object (https://developer.android.com/reference/android/media/Image) corresponding to the raw depth image of the current frame.

The image must be released via `Image.close()` (https://developer.android.com/reference/android/media/Image#close()) once it is no longer needed.

Each pixel is an 8-bit unsigned integer representing the estimated confidence of the corresponding pixel in the raw depth image. The confidence value is between 0 and 255, inclusive, with 0 representing the lowest confidence and 255 representing the highest confidence in the measured depth value. Pixels without a valid depth estimate have a confidence value of 0 and a corresponding depth value of 0 (see `acquireRawDepthImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthImage())).

The scaling of confidence values is linear and continuous within this range. Expect to see confidence values represented across the full range of 0 to 255, with values increasing as better observations are made of each location. If an application requires filtering out low-confidence pixels, removing depth pixels below a confidence threshold of half confidence (128) tends to work well.

The actual size of the depth image depends on the device and its display aspect ratio. The size of the depth image is typically around 160x120 pixels, with higher resolutions up to 640x480 on some devices. These sizes may change in the future. The outputs of `acquireDepthImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireDepthImage()), `acquireRawDepthImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthImage()) and `acquireRawDepthConfidenceImage()` (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthConfidenceImage()) will all have the exact same size.

Details	
Returns	The confidence image corresponding to the raw depth of the frame.

Details	
Throws	**NotYetAvailableException** (/ar/reference/java/com/google/ar/core/exceptions/NotYetAvaila
	NotTrackingException (/ar/reference/java/com/google/ar/core/exceptions/NotTracking
	IllegalStateException (https://developer.android.com/reference/java/lang/IllegalStateEx
	ResourceExhaustedException (/ar/reference/java/com/google/ar/core/exceptions/ResourceExh
	DeadlineExceededException (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc

acquireRawDepthImage

`public` **`Image`** (https://developer.android.com/reference/android/media/Image) `acquireRawDepthI`

Attempts to acquire a "raw", mostly unfiltered, depth Android Image object (https://developer.android.com/reference/android/media/Image) that corresponds to the current frame.

The raw depth image is sparse and does not provide valid depth for all pixels. Pixels without a valid depth estimate have a pixel value of 0 and a corresponding confidence value of 0 (see **`acquireRawDepthConfidenceImage()`** (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthConfidenceImage())).

The depth image has a single 16-bit plane at index 0, stored in little-endian format. Each pixel contains the distance in millimeters to the camera plane. Currently, the three most significant bits are always set to 000. The remaining thirteen bits express values from 0 to 8191, representing depth in millimeters. To extract distance from a depth map, see the Depth API developer guide (https://developers.google.com/ar/develop/java/depth/developer-guide#extract-distance).

The actual size of the depth image depends on the device and its display aspect ratio. The size of the depth image is typically around 160x120 pixels, with higher resolutions up to

640x480 on some devices. These sizes may change in the future. The outputs of **`acquireDepthImage()`** (/ar/reference/java/com/google/ar/core/Frame#acquireDepthImage()), **`acquireRawDepthImage()`** (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthImage()) and **`acquireRawDepthConfidenceImage()`** (/ar/reference/java/com/google/ar/core/Frame#acquireRawDepthConfidenceImage()) will all have the exact same size.

Optimal depth accuracy occurs between 500 millimeters (50 centimeters) and 5000 millimeters (5 meters) from the camera. Error increases quadratically as distance from the camera increases.

Depth is primarily estimated using data from the motion of world-facing cameras. As the user moves their device through the environment, 3D depth data is collected and cached, improving the quality of subsequent depth images and reducing the error introduced by camera distance. Depth accuracy and robustness improves if the device has a hardware depth sensor, such as a time-of-flight (ToF) camera.

Not every raw depth image contains a new depth estimate. Typically there are about 10 updates to the raw depth data per second. The depth images between those updates are a 3D reprojection which transforms each depth pixel into a 3D point in space and renders those 3D points into a new raw depth image based on the current camera pose. This effectively transforms raw depth image data from a previous frame to account for device movement since the depth data was calculated. For some applications it may be important to know whether the raw depth image contains new depth data or is a 3D reprojection (for example, to reduce the runtime cost of 3D reconstruction). To do that, compare the current raw depth image timestamp, obtained via **`Image.getTimestamp()`** (https://developer.android.com/reference/android/media/Image#getTimestamp()), with the previously recorded raw depth image timestamp. If they are different, the depth image contains new information.

The image must be released via **`Image.close()`** (https://developer.android.com/reference/android/media/Image#close()) once it is no longer needed.

Details	
Returns	The raw depth image corresponding to the frame.

Details	
Throws	**NotYetAvailableException** (/ar/reference/java/com/google/ar/core/exceptions/NotYetAvail
	NotTrackingException (/ar/reference/java/com/google/ar/core/exceptions/NotTracking
	IllegalStateException (https://developer.android.com/reference/java/lang/IllegalStateE
	ResourceExhaustedException (/ar/reference/java/com/google/ar/core/exceptions/ResourceExl
	DeadlineExceededException (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc

getAndroidCameraTimestamp

```
public long getAndroidCameraTimestamp()
```

Returns the (Android Camera timestamp (https://developer.android.com/reference/android/hardware/camera2/CaptureResult#SENSOR_TIMESTAMP)) of the image.

getAndroidSensorPose

`public` **Pose** (/ar/reference/java/com/google/ar/core/Pose) `getAndroidSensorPose()`

Returns the pose of the Android Sensor Coordinate System (https://developer.android.com/guide/topics/sensors/sensors_overview#sensors-coords) in the world coordinate space for this frame. The orientation follows the device's "native" orientation (it is not affected by display rotation) with all axes corresponding to those of the Android sensor coordinates.

See Also:

- **`Camera.getPose()`** (/ar/reference/java/com/google/ar/core/Camera#getPose()) for the pose of the physical camera.
- **`Camera.getDisplayOrientedPose()`** (/ar/reference/java/com/google/ar/core/Camera#getDisplayOrientedPose()) for the pose of the virtual camera.

Note: This pose is only useful when **`Camera.getTrackingState()`** (/ar/reference/java/com/google/ar/core/Camera#getTrackingState()) returns **`TrackingState.TRACKING`** (/ar/reference/java/com/google/ar/core/TrackingState#TRACKING) and otherwise should not be used.

getCamera

`public` **`Camera`** (/ar/reference/java/com/google/ar/core/Camera) `getCamera()`

Returns the **`Camera`** (/ar/reference/java/com/google/ar/core/Camera) object for the session. Note that this Camera instance is long-lived so the same instance is returned regardless of the frame object this method was called on.

getCameraTextureName

```
public int getCameraTextureName()
```

Returns the OpenGL ES camera texture name (id) associated with this frame. This is guaranteed to be one of the texture names previously set via **`Session.setCameraTextureNames(int[])`** (/ar/reference/java/com/google/ar/core/Session#setCameraTextureNames(int[])) or **`Session.setCameraTextureName(int)`** (/ar/reference/java/com/google/ar/core/Session#setCameraTextureName(int)). Texture names (ids) are returned in a round robin fashion in sequential frames.

Details	
Returns	the OpenGL ES texture name (id).

getImageMetadata

`public` **`ImageMetadata`** (/ar/reference/java/com/google/ar/core/ImageMetadata) `getImageMetad`

Returns the camera metadata for the current camera image, if available. Throws **`NotYetAvailableException`** (/ar/reference/java/com/google/ar/core/exceptions/NotYetAvailableException) when metadata is not yet available due to sensors data not yet being available.

If the AR session was created for shared camera access, this method will throw **`IllegalStateException`** (https://developer.android.com/reference/java/lang/IllegalStateException) . To retrieve image metadata in shared camera mode, use **`SharedCamera.setCaptureCallback(CameraCaptureSession.CaptureCallback, Handler)`** (/ar/reference/java/com/google/ar/core/SharedCamera#setCaptureCallback(android.hardware.camera2.CameraCaptureSession.CaptureCallback,%20android.os.Handler)) , then use **`getAndroidCameraTimestamp()`** (/ar/reference/java/com/google/ar/core/Frame#getAndroidCameraTimestamp()) to correlate the frame to metadata retrieved from **`CameraCaptureSession.CaptureCallback`** (https://developer.android.com/reference/android/hardware/camera2/CameraCaptureSession.CaptureCallback) .

Details	
Throws	**`NotYetAvailableException`** (/ar/reference/java/com/google/ar/core/exceptions/NotYetAvail
	`DeadlineExceededException` (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc
	`ResourceExhaustedException` (/ar/reference/java/com/google/ar/core/exceptions/ResourceExl

getLightEstimate

`public` **`LightEstimate`** (/ar/reference/java/com/google/ar/core/LightEstimate) `getLightEstimat`

Returns the current ambient light estimate, if light estimation was enabled.

If lighting estimation is not enabled in the session configuration, the returned LightingEstimate will always return **`LightEstimate.State.NOT_VALID`** (/ar/reference/java/com/google/ar/core/LightEstimate.State#NOT_VALID) from **`LightEstimate.getState()`** (/ar/reference/java/com/google/ar/core/LightEstimate#getState()).

getTimestamp

```
public long getTimestamp()
```

Returns the timestamp in nanoseconds when this image was captured. This can be used to detect dropped frames or measure the camera frame rate. The time base of this value is specifically **not** defined, but it is likely similar to `System.nanoTime()`.

getUpdatedAnchors

public **Collection** (https://developer.android.com/reference/java/util/Collection)<**Anchor** (/ar/refe

Returns the anchors that were changed by the **`Session.update()`** (/ar/reference/java/com/google/ar/core/Session#update()) that returned this Frame.

getUpdatedTrackData

public **Collection** (https://developer.android.com/reference/java/util/Collection)<**TrackData** (/ar/
  UUID (https://developer.android.com/reference/java/util/UUID) trackUuid
)

Retrieve all track data that was written to the specified track during the current frame. If frames are skipped during playback, which can happen when the device is under load, played back track data will be attached to a later frame in order.

Each call to **`recordTrackData(UUID, ByteBuffer)`** (/ar/reference/java/com/google/ar/core/Frame#recordTrackData(java.util.UUID,%20java.nio.ByteBuffer))
at recording time will be returned as a separate **`TrackData`** (/ar/reference/java/com/google/ar/core/TrackData) entry in the collection.

Details

Details	
Parameters	`trackUuid`
Throws	**`DeadlineExceededException`** (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc

getUpdatedTrackables

public **Collection** (https://developer.android.com/reference/java/util/Collection)<T> getUpdated
  Class (https://developer.android.com/reference/java/lang/Class)<T> filterType
)

Returns the trackables of a particular type that were changed by the **`Session.update()`** (/ar/reference/java/com/google/ar/core/Session#update()) that returned this Frame. `filterType` may be **`Plane.class`** (/ar/reference/java/com/google/ar/core/Plane) or **`Point.class`** (/ar/reference/java/com/google/ar/core/Point), or `Trackable.class` to retrieve all changed trackables.

Details	
Parameters	`filterType`

hasDisplayGeometryChanged

```
public boolean hasDisplayGeometryChanged()
```

Checks if the display rotation or viewport geometry changed since the previous `Frame`. The application should re-query **`Camera.getProjectionMatrix(float[], int, float, float)`** (/ar/reference/java/com/google/ar/core/Camera#getProjectionMatrix(float[],%20int,%20float,%20float)) and **`transformCoordinates2d(Coordinates2d, float[], Coordinates2d, float[])`** (/ar/reference/java/com/google/ar/core/Frame#transformCoordinates2d(com.google.ar.core.Coordinates2d,%20float[],%20com.google.ar.core.Coordinates2d,%20float[])) whenever this is true.

hitTest

```
public List (https://developer.android.com/reference/java/util/List)<HitResult (/ar/reference/java/
  MotionEvent (https://developer.android.com/reference/android/view/MotionEvent) motionEvent
)
```

Similar to **`hitTest(float, float)`** (/ar/reference/java/com/google/ar/core/Frame#hitTest(float,%20float)), but will take values from Android **`MotionEvent`** (https://developer.android.com/reference/android/view/MotionEvent). It is assumed that the `MotionEvent` is received from the same view that was used as the size for **`Session.setDisplayGeometry(int, int, int)`** (/ar/reference/java/com/google/ar/core/Session#setDisplayGeometry(int,%20int,%20int)).

Note: this method does not consider the **`action`** (https://developer.android.com/reference/android/view/MotionEvent#getAction()) of the `MotionEvent`. The caller must check for appropriate action, if needed, before calling this method.

Note: When using **`Session.Feature.FRONT_CAMERA`** (/ar/reference/java/com/google/ar/core/Session.Feature#FRONT_CAMERA), the returned hit result list will always be empty, as the camera is not **`TrackingState.TRACKING`** (/ar/reference/java/com/google/ar/core/TrackingState#TRACKING). Hit testing against tracked faces is not currently supported.

Details		
Parameters	`motionEvent`	an event containing the x,y coordinates to hit test

hitTest

```
public List (https://developer.android.com/reference/java/util/List)<HitResult (/ar/reference/java/
  float[] origin3,
  int originOffset,
  float[] direction3,
  int directionOffset
)
```

Similar to **`hitTest(float, float)`** (/ar/reference/java/com/google/ar/core/Frame#hitTest(float,%20float)), but takes an arbitrary ray in world space coordinates instead of a screen-space point.

Note: When using **`Session.Feature.FRONT_CAMERA`** (/ar/reference/java/com/google/ar/core/Session.Feature#FRONT_CAMERA), the returned hit result list will always be empty, as the camera is not **`TrackingState.TRACKING`** (/ar/reference/java/com/google/ar/core/TrackingState#TRACKING). Hit testing against tracked faces is not currently supported.

Details		
Parameters	`origin3`	an array of 3 floats containing ra origin in world space coordinate
	`originOffset`	the offset into `origin3` array.
	`direction3`	an array of 3 floats containing ra direction in world space coordinates. Does not have to b normalized.
	`directionOffset`	the offset into `direction3` arr
Returns	an ordered list of intersections with scene geometry, nearest hit first.	

hitTest

```
public List (https://developer.android.com/reference/java/util/List)<HitResult (/ar/reference/java/
  float xPx,
  float yPx
)
```

Performs a ray cast from the user's device in the direction of the given location in the camera view. Intersections with detected scene geometry are returned, sorted by distance from the device; the nearest intersection is returned first.

Note: Significant geometric leeway is given when returning hit results. For example, a plane hit may be generated if the ray came close, but did not actually hit within the plane extents or plane bounds (**`Plane.isPoseInExtents(Pose)`** (/ar/reference/java/com/google/ar/core/Plane#isPoseInExtents(com.google.ar.core.Pose)) and

`Plane.isPoseInPolygon(Pose)` (/ar/reference/java/com/google/ar/core/Plane#isPoseInPolygon(com.google.ar.core.Pose)) can be used to determine these cases). A point (point cloud) hit is generated when a point is roughly within one finger-width of the provided screen coordinates.

Note: When using **`Session.Feature.FRONT_CAMERA`** (/ar/reference/java/com/google/ar/core/Session.Feature#FRONT_CAMERA), the returned hit result list will always be empty, as the camera is not **`TrackingState.TRACKING`** (/ar/reference/java/com/google/ar/core/TrackingState#TRACKING). Hit testing against tracked faces is not currently supported.

Note: In ARCore 1.24.0 or later on supported devices, if depth is enabled by calling **`Config.setDepthMode(Config.DepthMode)`** (/ar/reference/java/com/google/ar/core/Config#setDepthMode(com.google.ar.core.Config.DepthMode))
with the value **`Config.DepthMode.AUTOMATIC`** (/ar/reference/java/com/google/ar/core/Config.DepthMode#AUTOMATIC), the returned list includes **`DepthPoint`** (/ar/reference/java/com/google/ar/core/DepthPoint) values sampled from the latest computed depth image.

Details		
Parameters	**xPx**	x coordinate in pixels
	yPx	y coordinate in pixels
Returns	an ordered list of intersections with scene geometry, nearest hit first	

hitTestInstantPlacement

```
public List (https://developer.android.com/reference/java/util/List)<HitResult (/ar/reference/java/
  float xPx,
  float yPx,
  float approximateDistanceMeters
)
```

Performs a ray cast that can return a result before ARCore establishes full tracking.

The pose and apparent scale of attached objects depends on the **`InstantPlacementPoint`** (/ar/reference/java/com/google/ar/core/InstantPlacementPoint) tracking method and the provided approximateDistanceMeters. A discussion of the different tracking methods and the effects of apparent object scale are described in **`InstantPlacementPoint`** (/ar/reference/java/com/google/ar/core/InstantPlacementPoint).

This function will succeed only if **`Config.InstantPlacementMode`** (/ar/reference/java/com/google/ar/core/Config.InstantPlacementMode) is **`Config.InstantPlacementMode.LOCAL_Y_UP`** (/ar/reference/java/com/google/ar/core/Config.InstantPlacementMode#LOCAL_Y_UP) in the ARCore session configuration, the ARCore session tracking state is **`TrackingState.TRACKING`** (/ar/reference/java/com/google/ar/core/TrackingState#TRACKING), and there are sufficient feature points to track the point in screen space.

Details		
Parameters	`xPx`	x screen coordinate in pixels
	`yPx`	y screen coordinate in pixels
	`approximateDistanceMeters`	the distance at which to create a This is only used while the track **SCREENSPACE_WITH_APPROX** (/ar/reference/java/com/googl .
Returns	if successful a list containing a single **`HitResult`** (/ar/reference/java/com/google/ar/core/HitResult), otherwise an empty list. The **`HitResult`** (/ar/reference/java/com/google/ar/core/HitResult) will have a trackable of type **`InstantPlacementPoint`** (/ar/reference/java/com/google/ar/core/InstantPlacementPoint) .	

recordTrackData

```
public void recordTrackData(
   UUID (https://developer.android.com/reference/java/util/UUID) trackId,
   ByteBuffer (https://developer.android.com/reference/java/nio/ByteBuffer) sample
)
```

Writes a data sample in the specified track. The samples recorded using this API are muxed into the recorded MP4 dataset as an additional MP4 stream.

Multiple samples can be recorded to the same frame and will be played back together.

For smooth playback of the MP4 on video players and for future compatibility of the MP4 datasets with ARCore's playback of data tracks it is recommended that the samples are recorded at a frequency no higher than 90kHz.

Additionally, if the samples are recorded at a frequency lower than 1Hz, empty (zero byte) padding samples will be automatically recorded at approximately one second intervals to fill in the gaps.

Recording samples introduces additional CPU and/or I/O overhead and may affect app performance.

Details		
Parameters	`trackId`	The **`UUID`** (https://developer.android.com the track.
	`sample`	The **`ByteBuffer`** (https://developer.android.com representation of the sample to
Throws	**`IllegalStateException`** (https://developer.android.com/reference/java/lang/IllegalStateEx	
	`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgum	
	`DeadlineExceededException` (/ar/reference/java/com/google/ar/core/exceptions/DeadlineExc	

transformCoordinates2d

```
public void transformCoordinates2d(
  Coordinates2d (/ar/reference/java/com/google/ar/core/Coordinates2d) inputCoordinates,
  float[] inputVertices2d,
  Coordinates2d (/ar/reference/java/com/google/ar/core/Coordinates2d) outputCoordinates,
  float[] outputVertices2d
)
```

Transforms a list of 2D coordinates from one 2D coordinate system to another 2D coordinate system.

Same as **`transformCoordinates2d(Coordinates2d, FloatBuffer, Coordinates2d, FloatBuffer)`** (/ar/reference/java/com/google/ar/core/Frame#transformCoordinates2d(com.google.ar.core.Coordinates2d,%20java.nio.FloatBuffer,%20com.google.ar.core.Coordinates2d,%20java.nio.FloatBuffer)) , but taking float arrays.

Details		
Parameters	`inputCoordinates`	The coordinate system used by `inputVertices2d`.
	`inputVertices2d`	Input 2D vertices to transform.
	`outputCoordinates`	The coordinate system to conve to.
	`outputVertices2d`	Buffer to put the transformed 2I vertices into.

transformCoordinates2d

```
public void transformCoordinates2d(
  Coordinates2d (/ar/reference/java/com/google/ar/core/Coordinates2d) inputCoordinates,
  FloatBuffer (https://developer.android.com/reference/java/nio/FloatBuffer) inputVertices2d,
  Coordinates2d (/ar/reference/java/com/google/ar/core/Coordinates2d) outputCoordinates,
  FloatBuffer (https://developer.android.com/reference/java/nio/FloatBuffer) outputVertices2d
)
```

Transforms a list of 2D coordinates from one 2D coordinate system to another 2D coordinate system.

For Android view coordinates (**`Coordinates2d.VIEW`** (/ar/reference/java/com/google/ar/core/Coordinates2d#VIEW), **`Coordinates2d.VIEW_NORMALIZED`** (/ar/reference/java/com/google/ar/core/Coordinates2d#VIEW_NORMALIZED)), the view information is taken from the most recent call to **`Session.setDisplayGeometry(int, int, int)`** (/ar/reference/java/com/google/ar/core/Session#setDisplayGeometry(int,%20int,%20int)).

Must be called on the most recently obtained **`Frame`** (/ar/reference/java/com/google/ar/core/Frame) object. If this function is called on an older frame, a log message will be printed and `outputVertices2d` will remain unchanged.

Some examples of useful conversions:

- To transform from `[0,1]` range to screen-quad coordinates for rendering: `Coordinates2d.VIEW_NORMALIZED` (/ar/reference/java/com/google/ar/core/Coordinates2d#VIEW_NORMALIZED) -> `Coordinates2d.TEXTURE_NORMALIZED` (/ar/reference/java/com/google/ar/core/Coordinates2d#TEXTURE_NORMALIZED)
- To transform from `[-1,1]` range to screen-quad coordinates for rendering: `Coordinates2d.OPENGL_NORMALIZED_DEVICE_COORDINATES` (/ar/reference/java/com/google/ar/core/Coordinates2d#OPENGL_NORMALIZED_DEVICE_COORDINATES) -> `Coordinates2d.TEXTURE_NORMALIZED` (/ar/reference/java/com/google/ar/core/Coordinates2d#TEXTURE_NORMALIZED)
- To transform a point found by a computer vision algorithm in a CPU image into a point on the screen that can be used to place an Android View (e.g. Button) at that location: `Coordinates2d.IMAGE_PIXELS` (/ar/reference/java/com/google/ar/core/Coordinates2d#IMAGE_PIXELS) -> `Coordinates2d.VIEW` (/ar/reference/java/com/google/ar/core/Coordinates2d#VIEW)
- To transform a point found by a computer vision algorithm in a CPU image into a point to be rendered using GL in clip-space (`[-1,1]` range): `Coordinates2d.IMAGE_PIXELS` (/ar/reference/java/com/google/ar/core/Coordinates2d#IMAGE_PIXELS) -> `Coordinates2d.OPENGL_NORMALIZED_DEVICE_COORDINATES` (/ar/reference/java/com/google/ar/core/Coordinates2d#OPENGL_NORMALIZED_DEVICE_COORDINATES)

Read-only array-backed buffers are not supported by `inputVertices2d` for performance reasons.

If `inputCoordinates` is same as `outputCoordinates`, the input vertices will be copied to the output vertices unmodified.

Details

Details		
Parameters	`inputCoordinates`	The coordinate system used by `inputVertices2d`.
	`inputVertices2d`	Input 2D vertices to transform.
	`outputCoordinates`	The coordinate system to conve to.
	`outputVertices2d`	Buffer to put the transformed 2I vertices into.
Throws	`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgum	
	`ReadOnlyBufferException` (https://developer.android.com/reference/java/nio/ReadOnlyBuff	

transformDisplayUvCoords

```
public void transformDisplayUvCoords(
  FloatBuffer (https://developer.android.com/reference/java/nio/FloatBuffer) uvCoords,
  FloatBuffer (https://developer.android.com/reference/java/nio/FloatBuffer) outUvCoords
)
```

This method was deprecated.
Replaced by **`frame.transformCoordinates2d(Coordinates2d.VIEW_NORMALIZED, ..,`** **`Coordinates2d.TEXTURE_NORMALIZED, ..)`**.

Transform the given texture coordinates to correctly show the background image. This will account for the display rotation, and any additional required adjustment. For performance, this function should be called only if `hasDisplayGeometryChanged()` (/ar/reference/java/com/google/ar/core/Frame#hasDisplayGeometryChanged()) returns true.

Usage Notes / Bugs:

- Both input and output buffers must be direct and native byte order.
- Position and limit of buffers is ignored.
- Capacity of both buffers must be identical.

- Capacity of both buffers must be a multiple of 2.

Note: both buffer positions will remain unmodified after this call.

Details		
Parameters	`uvCoords`	The uv coordinates to transforn
	`outUvCoords`	The buffer to hold the transforn uv coordinates. Must have enou remaining elements to fit the inp uvCoords.