|  |  |
| --- | --- |
| Product name | Confidentiality level |
|  |  |
| Product version | Total 30 pages |
|  |

HUAWEI AR Engine SDK Interface Manual

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by | Hu Chang/00422988  Huangjiayi/00201448  Qianpeng/00227536 | Date | 2018-03-21 |
| Reviewed by | Guozejin/00168222 | Date | 2018-10-31 |
| Approved by |  | Date |  |



Huawei Technologies Co., Ltd.

All rights reserved

Revision Record

| Date | Revision Version | CR ID / Defect ID | Sec No. | Change Description | Author |
| --- | --- | --- | --- | --- | --- |
| 2017/12/22 | 0.1 |  |  | Initialize this manual. | Hu Chang/422988 |
| 2018/01/09 | 0.2 |  | All | Revise the class name with AR prefix | Hu Chang/422988 |
| 2018/02/02 | 0.3 |  |  | Implement the pointcloud hit test and the *getSubsumedBy* method of ARPlane, add *equals* & *hashCode* methods, remove *getNearestPointIndex* and *getNearestPoint* methods. | Hu Chang/422988 |
| 2018/03/16 | 0.5 |  |  | Add the description of the exceptions, and add the enumeration of the vertical plane. | Hu Chang/422988 |
| 2018/05/03 | 0.6 |  |  | Throw ARCameraPermissionDeniedException in ARSession.resume when it detects that the camera permission is missing. | Hu Chang/422988 |
| 2018/06/06 | 0.7 |  |  | Add preview of ARBody, ARHand | Guozejin/168222 |
| 2018/06/19 | 0.9 |  |  | Add class of ARCamera, ARConfigBase, ARImageMetadata,AREnginesSelector etc. | Huangjiayi/00201448 |
| 2018/07/12 | 0.9.2 |  |  | Add interface of ARBody, ARHand | Huangjiayi/00201448 |

Catalog

1 Introduction 5

1.1 Intention 5

1.2 Scope 5

2 HUAWEI AR Engine SDK Interface 6

2.1 Process 6

2.2 Interface Introduction 6

2.2.1 Functionality class 6

2.2.1.1. ARAnchor 6

2.2.1.2. ARBody 7

2.2.1.3. ARBodyTrackingConfig 8

2.2.1.4. ARCamera 8

2.2.1.5. ARCameraConfig 9

2.2.1.6. ARConfigBase 9

2.2.1.1. AREnginesApk 11

2.2.1.2. AREnginesSelector 12

2.2.1.3. ARFrame 12

2.2.1.4. ARHand 13

2.2.1.5. ARHandTrackingConfig 15

2.2.1.6. ARHitResult 16

2.2.1.7. ARLightEstimate 16

2.2.1.8. ARPlane 16

2.2.1.9. ARPoint 18

2.2.1.10. ARPointCloud 18

2.2.1.11. ARPose 18

2.2.1.12. ARSession 20

2.2.1.13. ARTrackable 23

2.2.1.14. ARTrackableBase 23

2.2.1.15. ARWorldBodyTrackingConfig 23

2.2.1.16. ARWorldTrackingConfig 24

2.2.1.17. ARConfig(Deprecated) 24

2.2.1.18. ARPlaneHitResult(Deprecated) 25

2.2.1.19. ARPointCloudHitResult(Deprecated) 25

2.2.2 Exception 25

2.2.2.1. ARFatalException 25

2.2.2.2. ARMissingGlContextException 25

2.2.2.3. ARNotTrackingException 26

2.2.2.4. ARSessionNotPausedException 26

2.2.2.5. ARSessionPausedException 26

2.2.2.6. ARTextureNotSetException 26

2.2.2.7. ARCameraPermissionDeniedException 26

2.2.2.8. ARUnSupportedConfigurationException 26

2.2.2.9. ARUnavailableException 27

2.2.2.10. ARUnavailableClientSdkTooOldException 27

2.2.2.11. ARUnavailableServiceApkTooOldException 27

2.2.2.12. ARUnavailableDeviceNotCompatibleException 27

2.2.2.13. ARUnavailableEmuiNotCompatibleException 27

2.2.2.14. ARUnavailableServiceNotInstalledException 28

Figure List

Fig.1 Brief introduction of HUAWEI AR Engine process 6

# Introduction

## Intention

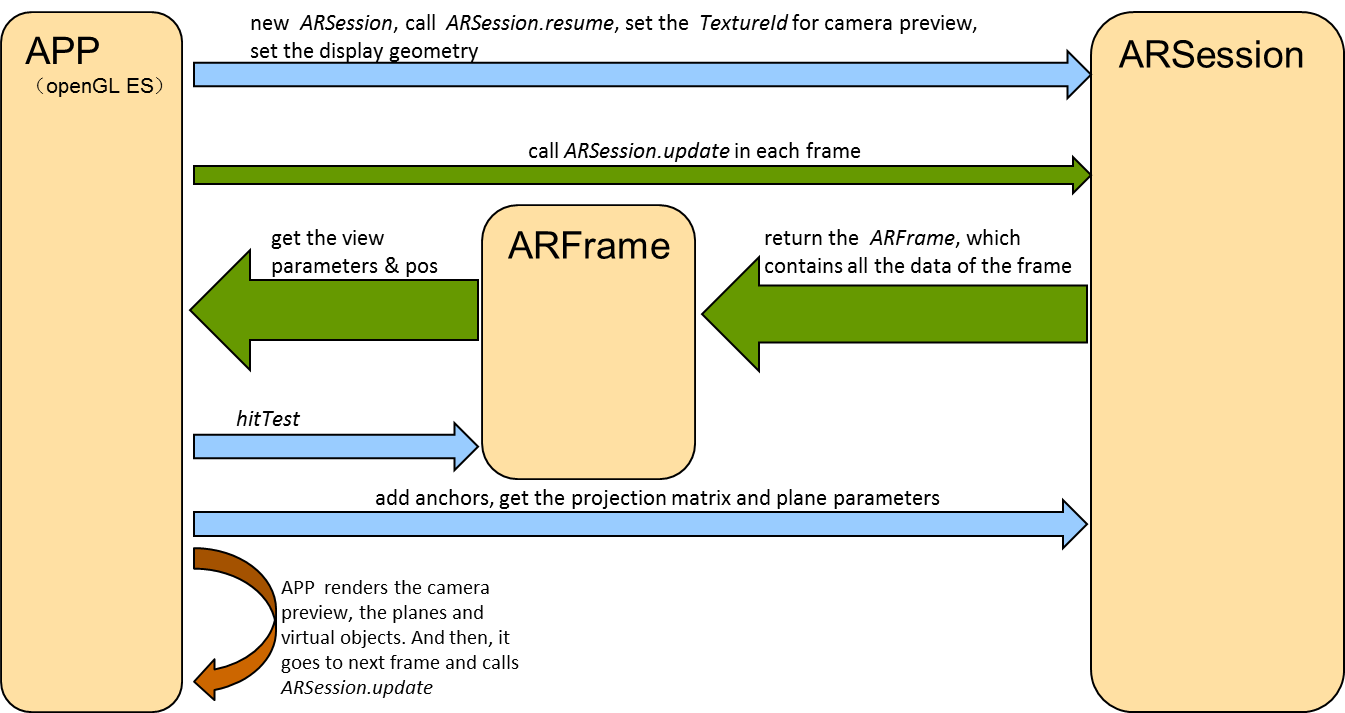
This manual will describe the interface of HUAWEI AR Engine SDK, and it is applicable to developers who will adopt the HUAWEI AR Engine service in their applications.

## Scope

This manual includes the process of HUAWEI AR Engine SDK, functionalities, parameters and returns.

# HUAWEI AR Engine SDK Interface

## Process



Brief introduction of HUAWEI AR Engine process

## Interface Introduction

### Functionality class

#### ARAnchor

##### Description: ARAnchor describes a fixed location and orientation in the real world. To stay at a fixed location in physical space, the numerical description of this position will update as HUAWEI AREngine's understanding of the space improves.

##### Methods:

###### public void detach()

Illustration:Used to tell AR Engine to stop tracking and delete the anchor.

###### public int getId()(Deprecated)

Illustration: Returns the identity of the ARAnchor.

###### public ARPose getPose()

Illustration: Returns the ARPose of the ARAnchor in the world coordinate space. This ARPose may change each time ARSession.*update()* is called. This ARPose should only be used for rendering if *getTrackingState()* returns TRACKING. Besides, the anchor’s local coordinate space is actually the coordinate of the virtual object which can be placed on this anchor.

###### public ARTrackable.TrackingState getTrackingState()

Illustration: Returns the current tracking state of this ARAnchor. If this state is anything but TRACKING, the ARPose should not be considered useful.

###### public boolean equals(Object obj)

Illustration: Indicates whether some other object is an ARAnchor referencing the same logical anchor as this one.

###### public int hashCode()

Illustration: Returns a hash code value for the object.

#### ARBody

##### Description:Return tracking results for body skeleton tracking,include body skeleton data,derived from ARTrackableBase.

##### Methods:

###### public int getBodyAction()

Illustration:Returns the body posture type.

###### public int getBodyID()

Illustration:Returns the body ID.

###### public int[] getBodySkeletonConnection()

Illustration: To obtain the body skeleton connection data,the sequence of the connection data is:[p0,p1;p0,p3;p0,p5;p1,p2;…]

###### public ARBodySkeletonType[] getBodySkeletonType()

Illustration:To obtain the body skeleton types.

###### public int getCoordinateSystemType()

Illustration: To obtain the coordinate system type.

|  |  |
| --- | --- |
| **ARCoordinateSystemType** | **Description** |
| COORDINATE\_SYSTEM\_TYPE\_3D\_WORLD | 3D world coordinate system |
| COORDINATE\_SYSTEM\_TYPE\_3D\_SELF | 3D coordinate system of the object itself |
| COORDINATE\_SYSTEM\_TYPE\_2D\_IMAGE | 2D image coordinate system |

###### public float[] getSkeletonPoint2D()

Illustration: To obtain the 2D coordinates of the human body skeleton points,the data format is :[x0,y0,x1,y1,…];

###### public float[] getSkeletonPoint3D()

Illustration: To obtain the 3D coordinates of the human body skeleton points, the data format is :[x0,y0,z0,x1,y1,z1…]

###### public int[] getSkeletonPointIsExist2D()

Illustration: To obtain the existences of human body skeleton points,the data format is:[0,1,0,0…]

###### public int[] getSkeletonPointIsExist3D()

Illustration: To obtain the existences of human body skeleton points,the data format is:[0,1,0,0…]

###### public float[] GetSkeletonConfidence ()

Illustration:get every confidence of Skeletons,the value is from 0 to 1.

###### public FloatBuffer GetMaskConfidence ()

###### Illustration: get every confidence of Mask Body,array size is TextureDimension\_width\* TextureDimension\_height, the value is from 0 to 1.

###### public ShortBuffer GetMaskDepth ()

Illustration: get every depth of Mask Body,array size is TextureDimension\_width\* TextureDimension\_height,the format of each value is depth16,refer to: https://developer.android.com/reference/android/graphics/ImageFormat#DEPTH16

#### ARBodyTrackingConfig

##### Description:Used to configure session for human body tracking,derived from ARConfigBase.

##### Methods:

###### public [ARConfigBase.CameraLensFacing](file:///E:\log\com\huawei\hiar\ARConfigBase.CameraLensFacing.html) getCameraLensFacing()

Illustration: Return the direction the camera faces relative to device screen.

###### public void setCameraLensFacing([ARConfigBase.CameraLensFacing](file:///E:\log\com\huawei\hiar\ARConfigBase.CameraLensFacing.html) lensFacing)

###### Illustration: Configure the direction the camera faces relative to device screen.

#### ARCamera

##### Description:Provides information about the camera that is used to capture images. The Camera is a long-lived object and the properties of Camera are updated every time ARSession.update() is called.

##### Methods:

###### public boolean equals(Object obj)

Illustration: Indicates whether some other object is a Camera referencing the same logical camera as this one.

###### public ARPose getDisplayOrientedPose()

Illustration: Returns the pose of the user's device in the world coordinate frame at the time of capture of the current camera texture. The pose is located at the camera of the device, and the direction matches the direction of the display (considering rotation).

###### public ARPose getPose()

Illustration: Returns the pose of the user's device in the world coordinate frame at the time of capture of the current camera texture. The position and direction of the pose follow the physical camera of the device (they are not affected by the direction of the display).

###### public void getProjectionMatrix

Illustration: Returns a projection matrix for rendering virtual content on top of the camera image.

###### public ARTrackable.TrackingState getTrackingState()

Illustration: Gets the current state of the pose of this camera.

###### public void getViewMatrix(float[] viewMatrix, int offset)

Illustration: Returns the view matrix for the camera for this frame.

###### public int hashCode()

###### Illustration: Returns a hash code value for the object.

#### ARCameraConfig

##### Description: Camera configuration

##### Method：

###### public Size GetImageDimensions()

###### Illustration: Obtains the camera image dimensions for the given camera config.

###### public Size GetTextureDimensions()

###### Illustration: Obtains the texture dimensions for the given camera config.

#### ARConfigBase

##### Description:Base classes of all ARConfigXXX classes.

##### Methods:

###### public ARAugmentedImageDatabase getAugmentedImageDatabase()

Illustration: Return the setted AugmentedImageDatabase.

###### public long getEnableItem()

Illustration:Return the capability of the camera.

|  |  |
| --- | --- |
| **EnableItem** | **Description** |
| ENABLE\_NULL = 0 | None |
| ENABLE\_DEPTH = 1 << 0 | Enable depth capability, enable by default |
| ENABLE\_MASK = 1 << 1 | Enable Mask capability |

###### public FocusMode getFocusMode()

Illustration:Return the focus mode. BodyAR, HandAR, FaceAR, AugmentedImageAR before version 1.4.0, default focus mode is fixed focus mode, after version 1.4.0, default focus mode is auto focus mode.

|  |  |
| --- | --- |
| **FocusMode** | **Description** |
| FIXED\_FOCUS = 0 | Fixed focus mode |
| AUTO\_FOCUS = 1 | Auto focus mode |

###### public ImageInputMode getImageInputMode ()

Illustration:Return the image input mode of AR session.

|  |  |
| --- | --- |
| **ImageInputMode** | **Description** |
| NON\_INPUT = 0 | No input. |
| EXTERNAL\_INPUT\_ALL = -1 | External input preview stream |

###### public List<Surface> getImageInputSurfaces()

Illustration: To obtain the surfaces for external input preview stream.

###### public List<SurfaceType> getImageInputSurfaceTypes()

Illustration: To obtain the surface types for external input preview stream.

|  |  |
| --- | --- |
| **SurfaceType** | **Description** |
| PREVIEW = 0 | Preview stream |
| VGA = 1 | Low resolution preview stream |
| METADATA=2 | METADATA stream |
| DEPTH=4 | Depth stream |

###### public LightingMode getLightingMode()

Illustration: To obtain the current lighting estimation model.

###### public PowerMode getPowerMode()

Illustration: Return the power mode.

|  |  |
| --- | --- |
| **PowerMode** | **Description** |
| NORMAL | Normal mode |
| POWER\_SAVING | Power saving mode |
| ULTRA\_POWER\_SAVING | Ultra Power Saving Mode |

###### public UpdateMode getUpdateMode()

Illustration: To obtain the current ARSession update model.

|  |  |
| --- | --- |
| **SurfaceType** | **Description** |
| BLOCKING | The update() method of ARsession returns only when a new frame is available. |
| LATEST\_CAMERA\_IMAGE | The update() method of ARsession returns immediately (if there is no new frame, it returns the previous frame). |

###### public void setAugmentedImageDatabase(ARAugmentedImageDatabase augImgDatabase)

Illustration:Set the AugmentedImageDatabase of the detecting and tracking images.

###### public void setEnableItem(long enableItem)

Illustration:Set the capability of the camera.

|  |  |
| --- | --- |
| **EnableItem** | **Description** |
| ENABLE\_NULL = 0 | None |
| ENABLE\_DEPTH = 1 << 0 | Enable depth capability, enable by default |
| ENABLE\_MASK = 1 << 1 | Enable Mask capability |

###### public FocusMode setFocusMode()

Illustration:Set the focus mode.

|  |  |
| --- | --- |
| **FocusMode** | **Description** |
| FIXED\_FOCUS = 0 | Fixed focus mode |
| AUTO\_FOCUS = 1 | Auto focus mode |

###### public void setImageInputMode(ImageInputMode mode)

Illustration: Set the image input mode of AR session.

###### public void setLightingMode(LightingMode lightingMode)

Illustration:Set the lighting estimation model.

|  |  |
| --- | --- |
| **LightingMode** | **Description** |
| DISABLED | Close light estimation |
| AMBIENT\_INTENSITY | Open light estimation,ARFrame. getLightEstimate()will return the ambient light intensity |

###### public void setPowerMode(PowerMode powerMode)

Illustration:Set the power mode.

|  |  |
| --- | --- |
| **PowerMode** | **Description** |
| NORMAL | Normal mode |
| POWER\_SAVING | Power saving mode |
| ULTRA\_POWER\_SAVING | Ultra Power Saving Mode |

###### public void setPreviewSize(int width,int height)

Illustration:Set preview size.

###### public void setUpdateMode(UpdateMode updateMode)

Illustration: Set the current ARSession update model.

|  |  |
| --- | --- |
| **UpdateMode** | **Description** |
| BLOCKING | The update() method of ARsession returns only when a new frame is available. |
| LATEST\_CAMERA\_IMAGE | The update() method of ARsession returns immediately (if there is no new frame, it returns the previous frame). |

#### AREnginesApk

##### Description:Used to manage ARCore server and HUAWEI AR Engine server.

##### Methods:

###### public static ARAvailability checkAvailability (Context applicationContext)

Illustration: Check if the device supports the currently set engine (set by the AREnginesSelector.setAREngine() method).

###### public static ARInstallStatus requestInstall (Activity applicationActivity, boolean userRequestedInstall)

Illustration: Initialize the installation function if you need to install the server. This method is called when userRequestedInstall is true.

#### AREnginesSelector

##### Description:Used to query and set the engine supported by this device.

##### Methods:

###### public static AREnginesAvaliblity checkAllAvailableEngines(Context context)

Illustration: Get the type of engine supported by this terminal.

|  |  |
| --- | --- |
| **AREnginesAvaliblity** | **AR engine capability** |
| HWAR\_NONE\_ENGINES\_SUPPORTED=0 | The device does not support any engine. |
| HWAR\_ENGINE\_SUPPORTED=1 | Only Huawei AREngine is supported. |
| GOOGLE\_ARCORE\_SUPPORTED=2 | Only Google ARCore is supported. |
| HWAR\_BOTH\_ENGINES\_SUPPORTED=3 | Support both Huawei AREngine and Google ARCore |

###### public static AREnginesType setAREngine(AREnginesType enginestype)

Illustration: Specify the engine to call.

|  |  |
| --- | --- |
| **AREnginesType** | **AR engine type** |
| HWAR\_NONE=0 | NO engine is supported |
| HWAR\_ENGINE=1 | Huawei AREngine |
| GOOGLE\_ARCORE=2 | Google ARCore |

###### public static AREnginesType getCreatedEngine()

Illustration: Get the type of engine currently in use.

#### ARFrame

##### Description: This class is a snapshot of the HUAWEI AR Engine system, should only be constructed by *ARSession.update().*

##### Methods:

###### public ARLightEstimate getLightEstimate()

Illustration: Returns the current ambient light estimate. If lighting estimation is not enabled in the configuration, the returned LightingEstimate will always returns false from *isValid()*.

###### public ARPointCloud getPointCloud()

Illustration: Returns the point cloud in current camera images.

###### public ARPose getPointCloudPose()

Illustration: Returns the pose of the point cloud. Due to the pose of each point in the cloud is in the world coordinate space, this method always return ARPose.IDENTITY.

###### public ARPose getPose()

Illustration: Returns the ARPose of the user's device in the world coordinate frame at the time of capture of the current camera texture. The position and orientation of the pose follow the device's physical camera and uses OpenGL camera conventions (+X right, +Y up, -Z in the direction the camera is looking).

###### public long getTimestampNs()

Illustration: Returns the timestamp in nanoseconds when this image was captured.

###### public ARTrackable.TrackingState getTrackingState()

Illustration: Gets the current state of the ARPose of this frame. If this state is anything but TRACKING, the ARPose should not be considered useful.

###### public Collection<ARAnchor> getUpdatedAnchors()

Illustration: Returns the ARAnchors that were changed by the *update()* that returned this ARFrame.

###### public Collection<ARPlane> getUpdatedPlanes()

Illustration: Returns the ARPlanes that were changed by the update() that returned this ARFrame.

###### public void getViewMatrix(float[] viewMatrix, int offset)

Illustration: Returns the view matrix for this frame. This is equivalent to: *getPose().inverse().asMatrix()*.

###### public List<ARHitResult> hitTest(float xPx, float yPx)

Illustration: 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. *xPX* is the x coordinate in pixels, and *yPX* is the y coordinate in pixels.

###### public List<ARHitResult> hitTest(MotionEvent motionEvent)

Illustration: Similar to *hitTest(float,float),* but will take values from Android MotionEvent. APP should ensure that the motionEvent is received from the same view that was used as the size for setDisplayGeometry.

###### public boolean isDisplayRotationChanged()

Illustration: Checks if the display rotation or viewport geometry changed since the previous Frame. The application should re-query *getProjectionMatrix()* and *transformDisplayUvCoords()* whenever this is true.

###### public void transformDisplayUvCoords(FloatBuffer uvCoords, FloatBuffer outUvCoords)

Illustration: Transform the given texture coordinates to correctly show the background image. This will account for the display rotation, and any additional required adjustment. This method should be called if *isDisplayRotationChanged()* returns true or resize the display view by *ARSession.setDisplayGeometry()*.

#### ARHand

##### Description:Return tracking results for hand tracking, including hand skeleton data and gesture recognition results,derived from ARTrackableBase.

##### Methods:

###### public int[] getGestureAction()（Not implemented）

Illustration: Get gestures (for example, left waving and right waving).

###### public float[] getGestureCenter()

Illustration: Get the coordinates of the center point of the hand.

###### public int getGestureCoordinateSystem ()

Illustration:Get the coordinate system.

|  |  |
| --- | --- |
| **ARCoordinateSystemType** | **Description** |
| COORDINATE\_SYSTEM\_TYPE\_3D\_WORLD | 3D world coordinate system |
| COORDINATE\_SYSTEM\_TYPE\_3D\_SELF | 3D coordinate system of the object itself |
| COORDINATE\_SYSTEM\_TYPE\_2D\_IMAGE | 2D image coordinate system |

###### public float[] getGestureHandBox()

Illustration: Get the Vertex coordinates of a box which indicating the position of hand, based on the OpenGL coordinate system.

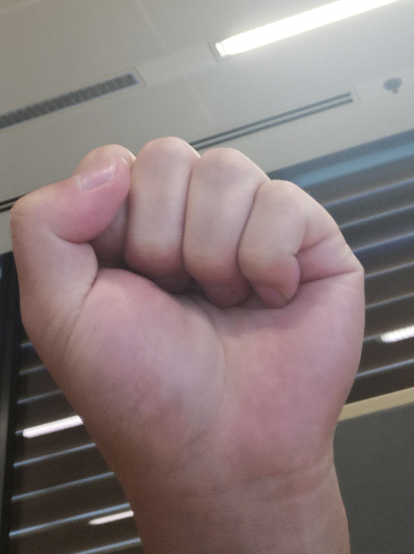
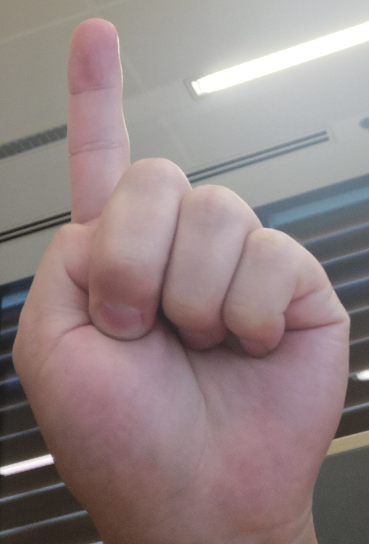
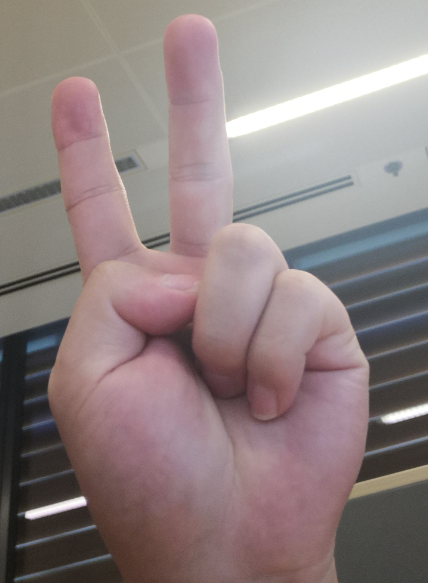
###### public float[] getGestureOrientation()

Illustration:Get the gesture orientations.

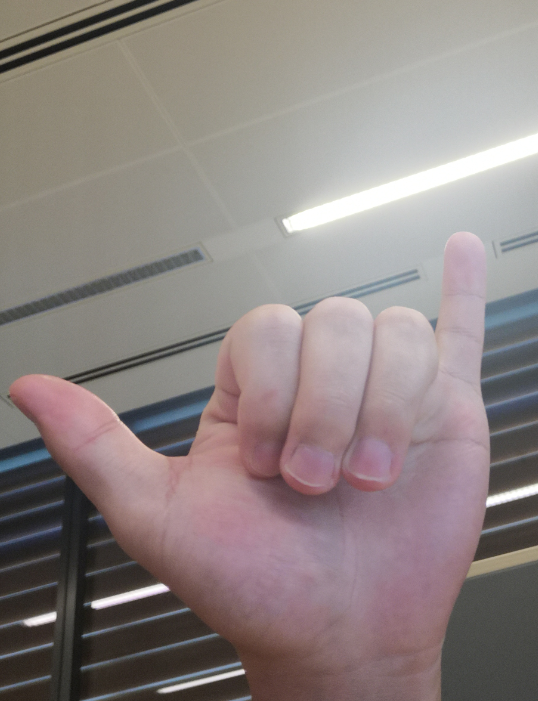
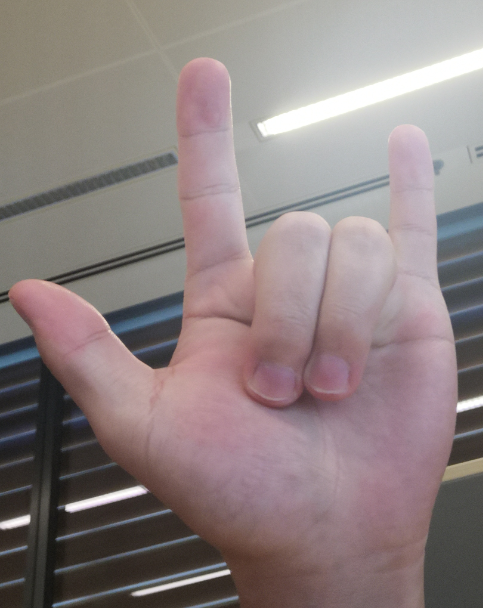
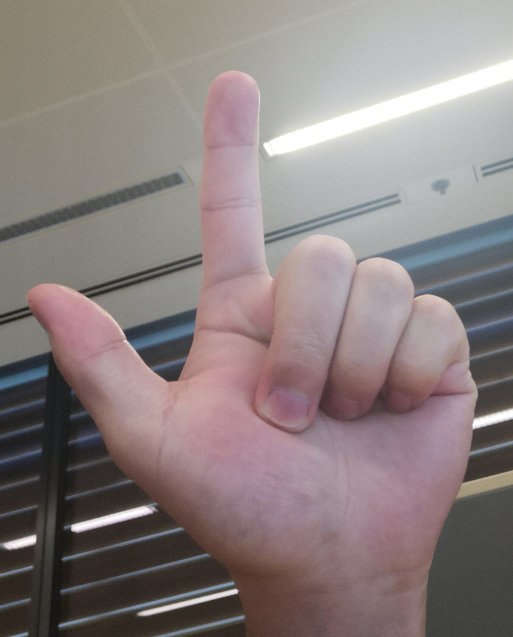
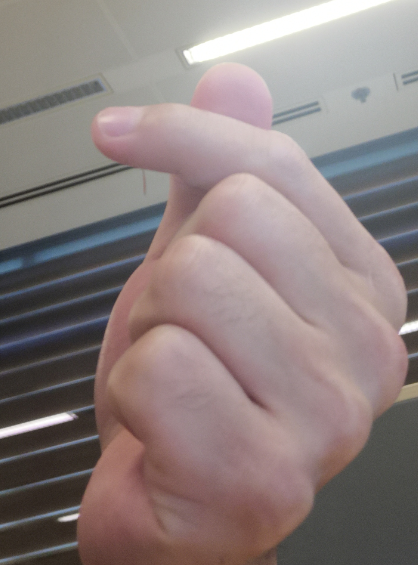
###### public int getGestureType ()

Illustration: Get the gesture type, the value is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **GestureType** | **Value** | **Support in disable depth** | **Support in enable depth** |
| Gesture 0（Make a fist） | 0 | Y | Y |
| Gesture 1（Erect index finger or middle finger） | 1 | Y | N |
| Gesture 2（Erect index finger and middle finger） | 2 | N | Y |
| Gesture 5（Five fingers open） | 5 | Y | Y |
| Gesture 6（Make a telephone call） | 6 | Y | Y |
| Gesture 7（I love you hand gesture） | 7 | Y | Y |
| Gesture 8（Erect thumb and middle finger） | 8 | N | Y |
| Gesture 10（Finger Heart） | 10 | Y | N |
| Other gestures | -1 | -- | -- |

Gesture0 Gesture 1 Gesture 2 Gesture 5

Gesture 6 Gesture 7 Gesture 8 Gesture 10

###### public int getHandID()（Deprecated）

Illustration:Get the hand ID

###### public float[] getHandskeletonArray()

Illustration: Returns the coordinates of the hand skeleton points.

###### public int[] getHandSkeletonConnection()

Illustration: Get the data of human skeleton point connection relationship, the data format is:[p0,p1;p0,p3;p0,p5;p1,p2;…]

###### public int getHandskeletonType()

Illustration: Get the type of hand skeleton.

###### public int getHandType ()

Illustration:Get the hand type,0 is the right hand, and 1 is the left hand.

###### public int getSkeletonCoordinateSystem()

Illustration: Returns the coordinate system for hand skeletons.

#### ARHandTrackingConfig

##### Description:Used to configuring session for gesture tracking,derived from ARConfigBase

##### Methods:

###### public void setCameraLensFacing

Illustration: Used to set camera orientation.

###### public CameraLensFacing getCameraLensFacing()

###### Illustration: Used to get camera orientation.

#### ARHitResult

##### Description: This is a abstract class, which defines an intersection between a ray and estimated real-world geometry.

##### Methods:

###### public ARAnchor createAnchor()

Illustration:Create a new anchor point in the hit position.

###### public ARPose getHitPose()

Illustration: Returns the pose of the intersection between a ray and detected real-world geometry. The position is the location in space where the ray intersected the geometry. The orientation’s definition differs depending on the object that was hit. When the ray hit a ARPlane, the local coordinate space is: X+ is perpendicular to the cast ray and parallel to the plane, Y+ points along the plane normal (up, for HORIZONTAL\_UPWARD\_FACING planes), and Z+ is parallel to the plane, pointing roughly toward the user's device. When it hit a point in ARPointCloud, the coordinate becomes: X+ is perpendicular to the cast ray and points right from the perspective of the user's device, Y+ points up, and Z+ points roughly toward the user's device.

###### public float getDistance()

Illustration: Returns the distance from the camera to the hit location, in meters.

###### public ARTrackable getTrackable()

Illustration:Return the tracked object that was hit.

#### ARLightEstimate

##### Description: Holds information about the estimated lighting of the real scene. Returned by ARFrame.*getLightEstimate()*.

##### Methods:

###### public State getState()

Illustration:Returns the validity of this light estimate.

###### public boolean isValid()

Illustration: The return value indicates whether the ARLightEstimate is valid.

###### public float getPixelIntensity()

Illustration: Returns the pixel intensity of the current camera view. Values are in the range (0.0, 1.0), with zero being black and one being white.

#### ARPlane

##### Description: Describes the current best knowledge of a real-world planar surface. Two or more planes may be automatically merged into a single parent plane. Assume that plane *A* and plane *B* will be merged. And *A* is recognized earlier than *B*. Then, *B* will be merged into *A*. After that, B will be removed in next ARSession.update().

##### Methods:

###### public ARPose getCenterPose()

Illustration: Returns the ARPose of the center of the detected ARPlane. The ARPose's transformed +Y axis will be point normal out of the plane, with the +X and +Z axes orienting the extents of the bounding rectangle.

###### public float getExtentX()

Illustration: Returns the length of this ARPlane's bounding rectangle measured along the local X-axis of the coordinate space centered on the plane.

###### public float getExtentZ()

Illustration: Returns the length of this ARPlane's bounding rectangle measured along the local Z-axis of the coordinate frame centered on the plane.

###### public FloatBuffer getPlanePolygon()

Illustration: Returns the 2D vertices of a concave polygon approximating the detected plane, in the form [x1, z1, x2, z2, ...]. These X-Z values are in the ARPlane's local x-z plane (y=0) and must be transformed by the ARPose (getCenterPose()) to get the boundary in world coordinates. Note that these polygon point are concave.

###### public ARPlane getSubsumedBy()

Illustration: Returns the ARPlane, which this ARPlane was merged into. It will return null if there’s no parent plane.

###### public PlaneType getType()

Illustration: Returns the type of this ARPlane.

|  |  |
| --- | --- |
| **PlaneType** | **Description** |
| HORIZONTAL\_UPWARD\_FACING | This type means a horizontal plane facing downward. |
| HORIZONTAL\_DOWNWARD\_FACING | This type indicates a horizontal plane facing upward. |
| VERTICAL\_FACING | This type indicates a vertical plane. |
| UNKNOWN\_FACING | This type means this plane is invalid. |

###### public boolean isPoseInExtents(ARPose pose)

Illustration:Return true if the positioned pose (usually obtained from HitResult) is in the rectangle of the plane.

###### public boolean isPoseInPolygon(ARPose pose)

Illustration:Return true if the positioned pose (usually obtained from HitResult) is in a flat polygon.

###### public boolean equals(Object obj)

Illustration: Indicates whether some other object is an ARPlane referencing the same logical plane as this one.

###### public int hashCode()

Illustration: Returns a hash code value for the object.

#### ARPoint

##### Description:Represents a point in space that AREngine is tracking, derived from ARTrackableBase.

##### Methods:

###### public OrientationMode getOrientationMode()

Illustration:

###### ARPose getPose()

Illustration: The current pose of the AR point.

#### ARPointCloud

##### Description: Contains a set of observed 3D points and confidence values.

##### Methods:

###### public FloatBuffer getPoints()

Illustration: Returns a buffer of point coordinates and confidence values. Each point is represented by four consecutive values in the buffer; first the X, Y, Z position coordinates, followed by a confidence value.

###### public long getTimestampNs()

Illustration: Returns the timestamp in nanoseconds when this point cloud was observed.

###### public void release()

Illustration:Release the resources of the point cloud

#### ARPose

##### Description: Represents an immutable rigid transformation from one coordinate space to another. As provided from HUAWEI AREngine, ARPoses always describe the transformation from object's local coordinate space to the world coordinate space. That is, ARPoses from HUAWEI AR Engine can be thought of as equivalent to OpenGL model matrices. The transformation is defined using a quaternion rotation about the origin followed by a translation. Coordinate system is right-handled.

##### Methods:

###### public static final ARPose IDENTITY

Illustration: The identity pose, which describe the transformation from the world coordinate space.

###### public ARPose(float[] translation, float[] rotation)

Illustration: Returns a new ARPose having the specified translation and rotation. Translation is the position vector from the destination (usually world) coordinate space to the local coordinate frame, expressed in destination (world) coordinates. Rotation is a quaternion following the Hamilton convention.

###### public ARPose compose(ARPose rhs)

Illustration: Returns the result of composing this with rhs. That is, transforming a point by the resulting pose will be equivalent to transforming that point first by rhs, and then transforming the result by this,which means that result.toMatrix() == this.toMatrix() \* rhs.toMatrix()

###### public ARPose extractRotation ()

Illustration: Returns a pose having the rotation of this pose but no translation.

###### public ARPose extractTranslation ()

Illustration: Returns a pose having the translation of this pose but no rotation.

###### public void getRotationQuaternion (float[] dest, int offset)

Illustration: Copies the rotation quaternion into a float array starting at offset. The values are written in the order {x, y, z, w}.

###### public void getTranslation (float[] dest, int offset)

Illustration: Copies the translation vector into a float array starting at offset.

###### public float[] getTransformedAxis (int axis, float scale)

Illustration: Returns the transformed direction of a local axis. The axis index is 0=X, 1=Y, 2=Z . And the scale is the length of the resulting vector.

public void getTransformedAxis (int axis, float scale, float[] dest, int offset)

Illustration: Computes the transformed direction of a local axis, outputting into a float array starting at offset. The axis index is 0=X, 1=Y, 2=Z . And the scale is the length of the resulting vector.

###### public ARPose inverse ()

Illustration: Returns a ARPose that performs the opposite transformation, which always means transforming from the world coordinate space to the object’s local coordinate space.

###### public float[] getXAxis ()

###### public float[] getYAxis ()

###### public float[] getZAxis ()

###### Illustration: Returns a 3-element array containing the direction of the transformed X, Y and Z axis, respectively.

###### public static ARPose makeInterpolated (ARPose a, ARPose b, float t)

###### Illustration: Returns a new pose that blends between two input poses. Linear and spherical-linear interpolation are performed on the translation and rotation respectively. result = (1-t)\*a+t\*b.t ranges in [0,1].

###### public static ARPose makeRotation (float[] quaternion)

public static ARPose makeRotation (float x, float y, float z, float w)

Illustration: Creates a rotation-only ARPose. The translation is 0.

###### public static ARPose makeTranslation (float[] translation)

public static ARPose makeTranslation (float tx, float ty, float tz)

Illustration: Creates a translation-only ARPose. The rotation is 0.

###### public float qw ()

public float qx ()

public float qx ()

public float qx ()

Illustration: Returns the W, X, Y, Z component of this ARPose's rotation quaternion, respectively.

###### public float tx ()

public float ty ()

public float tz ()

Illustration: Returns the X, Y, Z component of this ARPose's translation, respectively.

###### public void rotateVector (float[] vectorIn, int inOffset, float[] vectorOut, int outOffset)

Illustration: Rotates the provided vector by the ARPose's rotation. Does not apply translation. The *vectorIn* is the array containing the input vector. The *inOffset* is the location within *vectorIn* of the first element to read. The *vectorOut* is the array in which to write the output vector. And the *outOffset* is the location within vectorOut of the first element to write.

###### public float[] transformPoint (float[] x)

Illustration: Transforms the provided point by this ARPose, which means that result=this.toMatrix()\*x. *x* is a 3-element array containing the point to transform.

###### public void transformPoint (float[] pointIn, int inOffset, float[] pointOut, int outOffset)

Illustration: Transforms the provided point by the ARPose. Applies the ARPose's transformation to pointIn[inOffset..inOffset+2], placing the result in pointOut[outOffset..outOffset+2].

#### ARSession

##### Description: Manages HUAWEI AR Engine system state. This class is the main entry point to HUAWEI AR Engine. This class allows the user to create a session, configure it, start/stop it, and most importantly receive frames that can access to camera image and device pose.

##### Methods:

###### public ARSession(Activity activity)

Illustration: Creates a new ARSession, attached to the provided Activity. The activity is needed to track activity rotations and update the perspective matrix accordingly. This method may throw three types of exception:

|  |  |
| --- | --- |
| **Exception** | **Description** |
| ARUnavailableServiceNotInstalledException | If the HUAWEI AR Engine Service APK is not present. |
| ARUnavailableServiceApkTooOldException | If the installed HUAWEI AR Engine Service APK is too old for the HUAWEI AR Engine SDK with which this application was built. |
| ARUnavailableClientSdkTooOldException | If the HUAWEI AR Engine SDK that this application was built with is too old and no longer supported by this installed HUAWEI AR Engine Service APK. |

###### public ARAnchor addAnchor(ARPose pose) (Deprecated)

Illustration: Adds a new tracking anchor with a specified pose into the system. This pose always comes from *ARFrame.hitTest()*.

###### public void configure(ARConfigBase config)

Illustration:Configures the session..

###### public ARAnchor createAnchor(ARPose pose)

Illustration:Defines a tracked location in the physical world.

###### public Collection<ARAnchor> getAllAnchors()

Illustration: Returns all known anchors, including those whose states are PAUSED, TRACKING and STOPPED.APP should only use the valid anchors with the state of TRACKING, and delete the invalid anchors with the state of STOPPED.

###### public Collection<ARPlane> getAllPlanes()(Deprecated)

Illustration: Returns all the detected planes. If *ARConfig.PlaneFindingMode* in *ARSession.resume()* is DISABLED, this method returns null.

###### public <T extends ARTrackable> Collection<T> getAllTrackables(Class<T> filterType)

Illustration:Returns the collection of all known trackables.

###### public void getProjectionMatrix(float[] dest, int offset, float near, float far)

Illustration: Returns the projection matrix, which is used in the virtual content and plane rendering. The result outputs into the array *dest* which is the storage for at least 16 floats representing a 4x4 matrix in column major order. *offset* is the index of the beginning, *near* and *far* specifies the near and far clip plane in meters, respectively.

###### public static int getTextureId()

Illustration:Returns the textureId.

###### public boolean isSupported(ARConfig config) (Deprecated)

Illustration: Checks if the provided configuration is usable on the device. Note: This methods only returns true currently.

###### public void pause ()

Illustration: Pause the current session. This method will stop the camera feed and release resources. The session can be restarted again by calling *resume()*.

###### public void removeAnchors(Collection<ARAnchor> anchors)

Illustration: Removes all the specified anchors.

###### public void resume()

Illustration:Starts or resumes the AR Engine ARSession.

###### public void resume(ARConfig config)

Illustration: Starts or resumes the ARSession with the specified configuration. The session cannot be resumed after *stop()* is called. This method may throw two types of exception:

|  |  |
| --- | --- |
| **Exception** | **Description** |
| ARCameraPermissionDeniedException | If the client app fails to grant Camera Permission. |
| ARUnsupportedConfigurationException | If the config is not supported by HUAWEI AREngine. |

###### public void setCameraTextureName(int textureId)

Illustration: Sets the OpenGL texture id, which is created with *glGenTextures()* method by APP for rendering. The resulting texture must be bound to the *GL\_TEXTURE\_EXTERNAL\_OES* target for use. Shaders accessing this texture must use a *samplerExternalOES* sampler.

###### public void setDisplayGeometry(float width, float height) (Deprecated)

Illustration: Sets the display geometry with specified width and height in pixels. This width and height is usually the attribute of the view.

###### public void setDisplayGeometry(int displayRotation, int widthPx, int heightPx)

Illustration:Sets the aspect ratio, coordinate scaling, and display rotation.

###### public void stop()

Illustration: Stop the current session. This method will stop the camera feed and release resources. To restart, a new session is needed.

###### public ARFrame update()

Illustration: Updates the state of the HUAWEI AR Engine system. This includes: receiving a new camera frame, updating the location of the device, updating the location of tracking anchors, updating detected planes, etc. If the ARConfig.UpdateMode is BLOCKING,this method will be blocked until a new camera image is available. This Method may throw the following exceptions:

|  |  |
| --- | --- |
| **Exception** | **Description** |
| ARSessionPausedException | If ARSession.update() is called when the HUAWEI AR Engine is paused. |
| ARMissingGlContextException | If there is no opengl es context in current thread. |
| ARTextureNotSetException | If the ARSession.setCameraTextureName() is not called before update() is called. |

###### public ARCameraConfig GetCameraConfig()

###### Illustration: Gets the ArCameraConfig that the ArSession is currently using.

#### ARTrackable

##### Description: The ARTrackable is currently only used to describe the tracking state: enum TrackingState{ TRACKING, PAUSED, STOPPED}.

|  |  |
| --- | --- |
| **TrackingState** | **Description** |
| TRACKING | This state means the object is being tracked and it's state is valid. |
| PAUSED | This state indicates that HUAWEI AR Engine has paused tracking, and the related data is not accurate. |
| STOPPED | This state means that HUAWEI AR Engine has stopped tracking, and will never resume tracking. |

#### ARTrackableBase

##### Description:Base class for all traceable object classes,implements the ARTrackable interface.

##### Methods:

###### ARAnchor createAnchor(ARPose pose);

Illustration: Creates an Anchor at the given Pose in the world coordinate space that is attached to this Trackable.

###### public Collection<ARAnchor> getAnchors()

Illustration: Gets the Anchors attached to this Trackable.

###### ARTrackable.TrackingState getTrackingState();

Illustration: Gets the TrackingState of this Trackable.

#### ARWorldBodyTrackingConfig

##### Description:Used for configure session for motion tracking and Human body tracking, derived from ARConfigBase

##### Methods:

###### public void setCameraLensFacing(CameraLensFacing cameraLensFacing)

Illustration: Set to use front or rear camera lens.

###### public CameraLensFacing getCameraLensFacing()

Illustration: Returns the currently configured lens facing , front or rear.

###### public void setPlaneFindingMode(PlaneFindingMode findingMode)

Illustration: Sets the plane finding mode that should be used.

###### public PlaneFindingMode getPlaneFindingMode()

Illustration: Returns the currently configured plane finding mode.

#### ARWorldTrackingConfig

##### Description:Configuring session for motion tracking,derived from ARConfigBase.

##### Methods:

###### public void setPlaneFindingMode (PlaneFindingMode planeFindingMode)

Illustration: Sets the plane finding mode that should be used..

###### public PlaneFindingMode getPlaneFindingMode()

Illustration: Returns the currently configured plane finding mode.

#### ARConfig(Deprecated)

##### Description: This class is used to configure the HUAWEI AR Engine system,including the three modes: LightingMode, PlaneFindingMode and UpdateMode.

|  |  |
| --- | --- |
| **LightingMode** | **描述** |
| DISABLED | DISABLED means that lighting estimation is disabled. |
| AMBIENT\_INTENSITY | AMBIENT\_INTENSITY indicates the lighting estimation is enabled, generating a single-value intensity estimate. |

|  |  |
| --- | --- |
| **PlaneFindingMode** | **描述** |
| DISABLED | Disable the plane detection. |
| ENABLE | Enable the plane detection, including both horizontal and vertical planes. |
| HORIZONTAL\_ONLY | Enable the detection of only horizontal planes. |
| VERTICAL\_ONLY | Enable the detection of only vertical planes. |

|  |  |
| --- | --- |
| **UpdateMode** | **描述** |
| BLOCKING | In BLOCKING model, ARSession.update() will wait until a new camera image is available. |
| LATEST\_CAMERA\_IMAGE | In LATEST\_CAMERA\_IMAGE model, ARSession.update() will return immediately without blocking. If no new camera image is available, then it will return the most recent ARFrme. |

##### Note: Currently, these models cannot be modified and they are fixed: LightingMode is AMBIENT\_INTENSITY, PlaneFindingMode is HORIZONTAL, and UpdateMode is BLOCKING.

##### Methods:

###### public void setLightingMode(LightingMode lightingMode)

public LightingMode getLightingMode()

Illustration: Sets and gets the lighting estimation mode.

###### public void setPlaneFindingMode(PlaneFindingMode planeFindingMode)

public PlaneFindingMode getPlaneFindingMode()

Illustration: Sets and gets the lighting estimation mode.

###### public void setUpdateMode(UpdateMode updateMode)

public UpdateMode getUpdateMode()

Illustration: Sets and gets the update mode.

#### ARPlaneHitResult(Deprecated)

##### Description: Inherits from ARHitResult,and defines the intersection between a ray and a tracking ARPlane. This class should be constructed by *ARFrame.hitTest()*.

##### Methods: （Besides those in ARHitResult）

###### public ARPlane getPlane()

Illustration: Returns the hit ARPlane.

###### public boolean isHitOnFrontFace()

Illustration: Checks if the intersection is on the front face of the ARPlane.

###### public boolean isHitInExtents()

Illustration: Checks if the intersection is in the rectangular extents of ARPlane.

###### public boolean isHitInPolygon()

Illustration: Checks if the intersection is in the polygon of ARPlane.

#### ARPointCloudHitResult(Deprecated)

##### Description: Inherits from ARHitResult,and defines the intersection between a ray and a tracking ARPlane. This class should be constructed by ARFrame.hitTest().

##### Methods:

###### public ARPointCloud getPointCloud()

Illustration: Returns the hit ARPointCloud.

### Exception

#### ARFatalException

##### Description: Inherits from RuntimeException. Thrown when HUAWEI AR Engine experiences a non-recoverable internal error.

##### Methods:

###### public ARFatalException ()

###### public ARFatalException (String message)

###### public ARFatalException (String message, Throwable cause)

#### ARMissingGlContextException

##### Description: Inherits from IllegalStateException. Thrown when the opengl es context is missing in current thread.

##### Methods:

###### public ARMissingGlContextException()

#### ARNotTrackingException

##### Description: Inherits from IllegalStateException. Thrown if an operation requires the AR system to be TRACKING.

##### Methods:

###### public ARNotTrackingException()

#### ARSessionNotPausedException

##### Description: Inherits from IllegalStateException. Thrown if an operation requires HUAWEI AR Engine to be stopped.

##### Methods:

###### public ARSessionNotPausedException()

#### ARSessionPausedException

##### Description: Inherits from IllegalStateException. Thrown if an operation requires HUAWEI AR Engine to be running. For example, ARSession.update() is called without calling the ARSession.resume() firstly at the start up. Or ARSession.update() is called between the ARSeesion.pause() and ARSeesion.resume().

##### Methods:

###### public ARSessionPausedException()

#### ARTextureNotSetException

##### Description: Inherits from IllegalStateException. Thrown if a texture name was not set by calling setCameraTextureName() before the first call update().

##### Methods:

###### public ARTextureNotSetException()

#### ARCameraPermissionDeniedException

##### Description: Inherits from RuntimeException. Thrown if the camera permission is not granted when resume is called.

##### Methods:

###### public ARCameraPermissionDeniedException()

#### ARUnSupportedConfigurationException

##### Description: Inherits from RuntimeException. Thrown if the configuration supplied to resume() is unsupported. To avoid this exception, ensure that isSupported() returns true before calling resume.

##### Methods:

###### public ARUnSupportedConfigurationException()

#### ARUnavailableException

##### Description: Inherits from RuntimeException. This class and its subclasses is used to indicate that the HUAWEI AR Engine system is not available. The subclasses include: ARUnavailableClientSdkTooOldException, ARUnavailableServiceApkTooOldException, ARUnavailableDeviceNotCompatibleException, ARUnavailableEmuiNotCompatibleException, ARUnavailableServiceNotInstalledException.

##### Methods:

###### public ARUnavailableException()

#### ARUnavailableClientSdkTooOldException

##### Description: Inherits from ARUnavailableException. Thrown when the HUAWEI AR Engine SDK that this application was built with is too old for the installed HUAWEI AR Engine Service APK.

##### Methods:

###### public ARUnavailableClientSdkTooOldException()

#### ARUnavailableServiceApkTooOldException

##### Description: Inherits from ARUnavailableException. Thrown when the installed HUAWEI AR Engine Service APK is too old for the HUAWEI AR Engine SDK that this application was built with.

##### Methods:

###### public ARUnavailableServiceApkTooOldException()

#### ARUnavailableDeviceNotCompatibleException

##### Description: Inherits from ARUnavailableException. Thrown when the hardware of current device is not compatible with HUAWEI AREngine.

##### Methods:

###### public ARUnavailableDeviceNotCompatibleException()

#### ARUnavailableEmuiNotCompatibleException

##### Description: Inherits from ARUnavailableException. Thrown when the EMUI version of current device is not compatible with HUAWEI AREngine.

##### Methods:

###### public ARUnavailbleEmuiNotCompatibleException()

#### ARUnavailableServiceNotInstalledException

##### Description: Inherits from ARUnavailbleException. Thrown when HUAWEI AR Engine Service APK is not installed.

##### Methods:

###### public ARUnavailbleServiceNotInstalledException()