|  |  |
| --- | --- |
| Product name | Confidentiality level |
|  |  |
| Product version | Total 29 pages |
|  |

HUAWEI AR Engine Unity SDK Interface Manual

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by | Hu Chang/00422988  Qianpeng/00227536 | Date | 2018-05-03 |
| Reviewed by | Guozejin/00449655 | 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 |
| --- | --- | --- | --- | --- | --- |
| 2018/05/03 | 0.6 |  |  | Initialize this manual. | Huchang/00422988 |
| 2018/08/21 | 0.9.5 |  |  | Synchronize with Chinese version | Huchang /00422988 |

Catalog

1 Introduction 6

1.1 Intention 6

1.2 Scope 6

1.3 Constraint 6

2 HUAWEI AR Engine Unity SDK Interface 7

2.1 Process 7

2.2 Interface Introduction 7

2.2.1 Basic data class 7

2.2.1.1. ARTrackable 7

2.2.1.2. ARAnchor 8

2.2.1.3. ARPlane 8

2.2.1.4. ARPoint 10

2.2.1.5. ARBody 10

2.2.1.6. ARHand 12

2.2.1.7. ARPointCloud 14

2.2.1.8. AREnvironmentalLight 14

2.2.1.9. ARCameraMetadataValue 14

2.2.2 Functionality class 15

2.2.2.1. AREnginesSelector 15

2.2.2.2. AREnginesApk 15

2.2.2.3. ARConfigBase 16

2.2.2.4. ARWorldTrackingConfig 16

2.2.2.5. ARFaceTrackingConfig 17

2.2.2.6. ARBodyTrackingConfig 17

2.2.2.7. ARWorldBodyTrackingConfig 18

2.2.2.8. ARCameraConfig 18

2.2.2.9. ARHandTrackingConfig 18

2.2.2.10. ARCameraMetadata 18

2.2.2.11. ARSession 19

2.2.2.12. ARFrame 21

2.2.2.13. ARHitResult 22

2.2.2.14. ARPlaneHitResult [Obsolete] 23

2.2.2.15. ARPointCloudHitResult [Obsolete] 23

2.2.2.16. BackGroundRenderer 23

2.2.2.17. AsyncTask<T> 23

2.2.2.18. AndroidPermissionsRequest 24

2.2.2.19. AndroidPermissionsRequestResult 24

2.2.3 Exception 24

2.2.3.1. ARMissingGlContextException 24

2.2.3.2. ARNotTrackingException 24

2.2.3.3. ARSessionNotPausedException 24

2.2.3.4. ARSessionPausedException 24

2.2.3.5. ARTextureNotSetException 24

2.2.3.6. ARCameraPermissionDeniedException 24

2.2.3.7. ARUnSupportedConfigurationException 25

2.2.3.8. ARUnavailableException 25

2.2.3.9. ARUnavailableClientSdkTooOldException 25

2.2.3.10. ARUnavailableServiceApkTooOldException 25

2.2.3.11. ARUnavailableDeviceNotCompatibleException 25

2.2.3.12. ARUnavailableEmuiNotCompatibleException 25

2.2.3.13. ARUnavailableServiceNotInstalledException 25

2.2.3.14. ARUnavailableUserDeclinedInstallationException 25

2.2.3.15. ARUnavailableConnectServerTimeOutException 25

2.2.3.16. ARNotYetAvailableException 25

2.2.3.17. ARResourceExhaustedException 26

2.2.3.18. ARDeadlineExceededException 26

Figure List

Fig.1 Brief introduction of HUAWEI AR Engine process 7

# Introduction

## Intention

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

## Scope

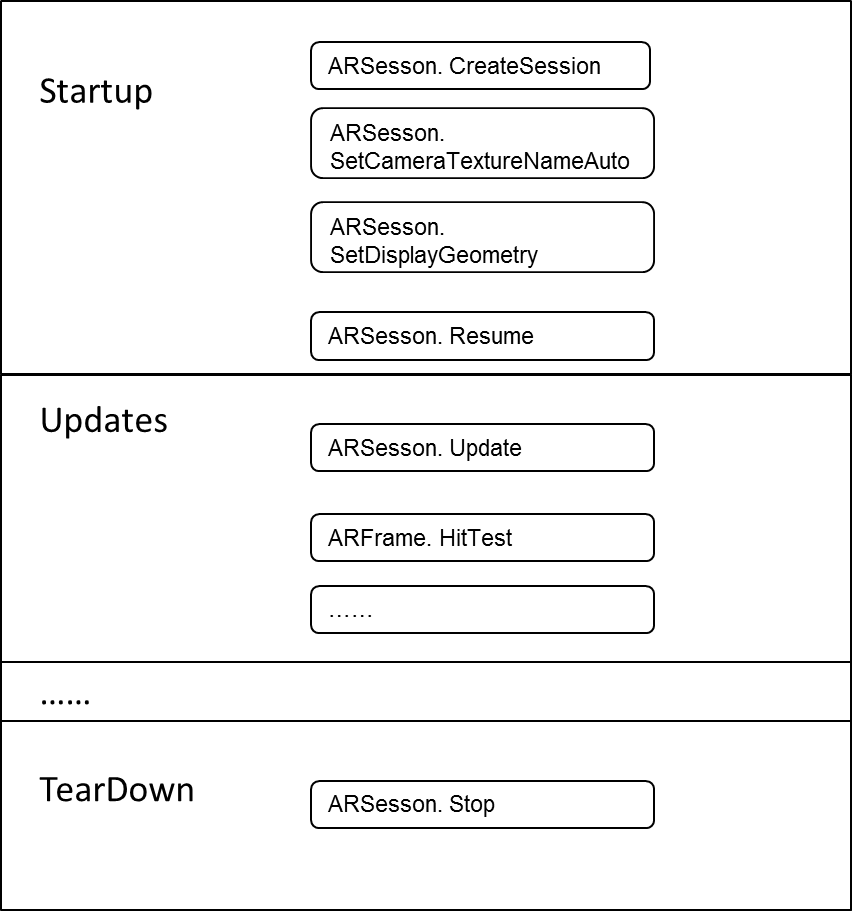
This manual includes the process of HUAWEI AR Engine Unity SDK, functionalities, parameters and returns.

## Constraint

The HUAWEI AR Engine Unity SDK is based on the NDK interface of HUAWEI AR Engine. As to the adoption of new features of Unity, Unity 2017.4 or higher version is recommended. In your building, please choose the build system as Internal, and **shutdown the Multithreaded Rendering in PlayerSettings**. Note: HandAR, BodyAR and Face AR is only used in Huawei AR Engine. If ARCore is set as the running time engine, these three features is not available.

# HUAWEI AR Engine Unity SDK Interface

## Process



Brief introduction of HUAWEI AR Engine process

## Interface Introduction

### Basic data class

#### ARTrackable

##### Description: This class is abstract, and is used to describe something in the real world that AREngine can track. It’s the superclass of ARPlane, ARPoint, ARHand, ARFace, and ARBody.

##### Methods:

###### public virtual TrackingState GetTrackingState()

Illustration: This method returns the tracking state of current trackable. *Enum TrackingState{ TRACKING, PAUSED, STOPPED}*.

|  |  |
| --- | --- |
| **TrackingState** | **Description** |
| TRACKING | This state means the object is being tracked and its 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. |

###### public virtual ARAnchor CreateAnchor(Pose pose)

Illustration: Creates an ARAnchor at the given Pose in the world coordinate space that is attached to this Trackable. Note that the relative offset between the Pose of multiple Anchors attached to a Trackable may adjust slightly over time as ARCore updates its model of the world.

###### public virtual void GetAllAnchors(List<ARAnchor> anchors)

Illustration: Gets the Anchors attached to this Trackable.

###### public override bool Equals(object obj)

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

###### public override int GetHashCode()

Illustration: Returns a hash code value for the object.

#### 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 Pose GetPose()

Illustration: Returns the Pose of the ARAnchor in the unity coordinate space. This Pose may change each time ARSession.*update()* is called. This ARPose should only be used for rendering if *getTrackingState()* returns TRACKING. Note that *Pose* is a struct in UnityEngine namespace.

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

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

###### public void Detach()

Illustration: Detaches this ARAnchor from its ARTrackable removes it from the Session. After calling, *ARAnchor.getTrackingState()* will return *STOPPED*.

###### 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.

#### ARPlane

##### Description: This class inherits from *ARTrackable*. 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*, which means B is the child plane of A. After that, B will continue behaving as if it were independently tracked.

##### Methods:

###### public Pose 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 void GetPlanePolygon(List<Vector3> polygonList)

Illustration: Returns the 3D vertices of a convex polygon approximating the detected plane, in the form [x1, 0,z1, x2,0, z2, ...]. These X-Z values are in the ARPlane's local x-z plane (y=0) and must be transformed by the Pose (*GetCenterPose()*) to get the boundary in unity coordinates.

###### public void GetPlanePolygon(List<Vector2> polygonList)

###### Illustration: Returns the 2D vertices of a convex 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 Pose (*GetCenterPose()*) to get the boundary in unity coordinates.

###### public ARPlane getSubsumedBy()

Illustration: If this plane has been subsumed, this method will return the plane this plane was merged into. In cases where a subsuming plane is itself subsumed, this function will always return the topmost non-subsumed plane. It will return null if there’s no parent plane.

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

Illustration: Gets this ARPlane's TrackingState. The ARPlane is valid only when the state is TRACKING.

###### 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 bool IsPoseInExtents(Pose pose)

Illustration: This method returns whether the given *pose* is in the extent of current plane.

###### public bool IsPoseInPolygon(Pose pose)

Illustration: Returns whether the given *pose* is in the plane’s polygon.

#### ARPoint

##### Description: This class is the subclass of ARTrackable, and it represents a point in the real world space that AREngine is tracking. This object usually comes from a hittest.

##### Methods:

###### public Pose GetPose()

Illustration: Returns the pose of current point.

###### public OrientationMode GetOrientationMode()

Illustration: Returns the orientation mode of current point.

|  |  |
| --- | --- |
| **OrientationMode** | **Description** |
| INITIALIZED\_TO\_IDENTITY | Same as world space. |
| ESTIMATED\_SURFACE\_NORMAL | Follow the behavior described in HitPose of ARHitResult. |

#### ARBody

##### Description: Subclass of ARTrackable. This class describe the result of body tracking, including posture and body skeleton.

##### Methods:

###### public ARCoordinateSystemType GetCoordinateSystemType()

Illustration: Returns the coordinate system type that is in use currently.

|  |  |
| --- | --- |
| **ARCoordinateSystemType** | **Description** |
| COORDINATE\_SYSTEM\_TYPE\_ UNKNOWN | Unknown coordinate. |
| COORDINATE\_SYSTEM\_TYPE\_3D\_WORLD | 3D World coordinate. |
| COORDINATE\_SYSTEM\_TYPE\_3D\_SELF | 3D coordinate of trackable itself. |
| COORDINATE\_SYSTEM\_TYPE\_2D\_IMAGE | 2D image coordinate. |
| COORDINATE\_SYSTEM\_TYPE\_3D\_CAMERA | 3D camera or eye coordinate. |

###### public int GetSkeletonPointCount()

Illustration: Returns the count of body skeleton.

###### public void GetSkeletons(Dictionary<SkeletonPointName, SkeletonPointEntry> outDic)

Illustration: Retrieve the data of each body skeleton, and put it into a dictionary. The *SkeletonPointName* is defined as follow. enum SkeletonPointName { Head\_Top, Neck, Right\_Shoulder, Right\_Elbow, Right\_Wrist, Left\_Shoulder, Left\_Elbow, Left\_Wrist, Right\_Hip, Right\_Knee, Right\_Ankle, Left\_Hip, Left\_Knee, Left\_Ankle, Body\_Center, SKELETON\_LENGTH }. And the *SkeletonPointEntry* is defined as: struct SkeletonPointEntry{

public bool Is2DValid // Is 2D data valid ?

public Vector3 Coordinate2D // 2D coordinate value, which is in NDC. The range of x, y is

[-1, 1] and z is 0.

public bool Is3DValid // Is 3D data valid?

public Vector3 Coordinate3D // 3D coordinate value, which is in body’s coordinate. The

range of x, y, z is [-1, 1].

}

###### public void GetSkeletonConnection(List<KeyValuePair<SkeletonPointName, SkeletonPointName>> outConnections)

Illustration: Retrieve the connections between skeleton points and put it into a list.

###### public int GetBodyAction()

Illustration: Returns the body actions.

|  |  |
| --- | --- |
| **Posture Type** | **int value** |
| Posture1 | 1 |
| Posture1 | 2 |
| Posture1 | 3 |
| Posture1 | 4 |
| Posture1 | 5 |
| Posture1 | 6 |
| Others | 0 |

posture1 posture2 posture3

posture4 posture5 posture6

###### public IntPtr GetSkeletonsConfidence()

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

###### public IntPtr GetMaskConfidence()

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

###### public IntPtr 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.](https://developer.android.com/reference/android/graphics/ImageFormat%23DEPTH16)

#### ARHand

##### Description: Subclass of ARTrackable. This class describe the result of hand tracking, including the hand type, gesture and skeleton.

##### Methods:

###### public ARCoordinateSystemType GetGestureCoordinateSystemType()

Illustration: Returns the coordinate system used in gesture.

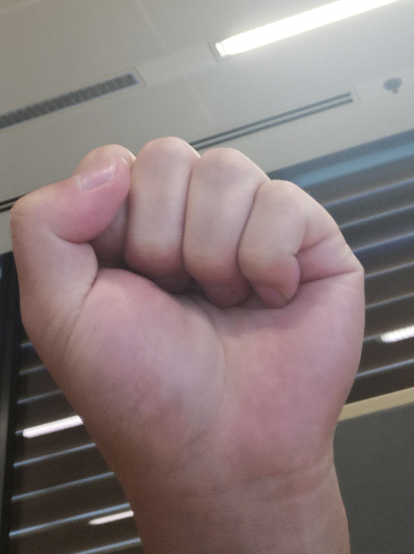
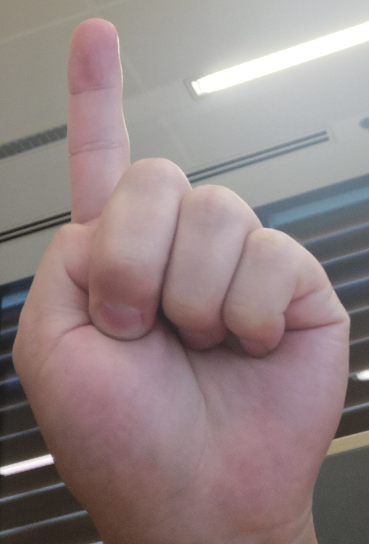
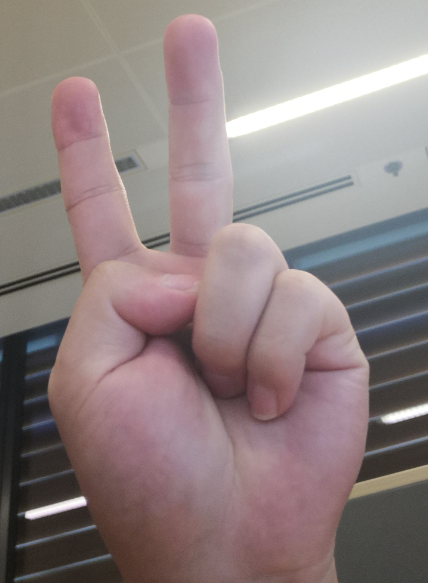
###### public HandType GetHandType()

Illustration: Returns the type of detected hand. enum HandType { UNKNOWN, RIGHT , LEFT }

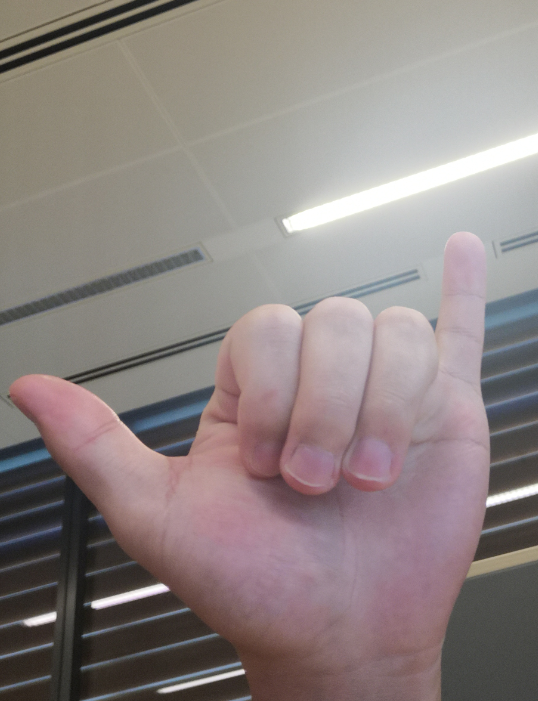
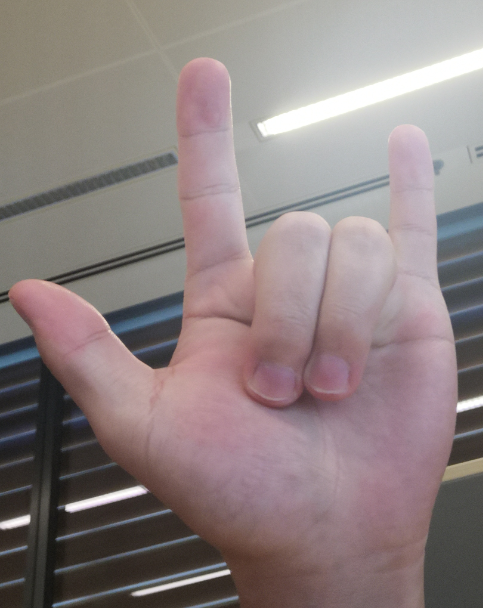
###### public int GetGestureType()

Illustration: Returns the type of detected gesture.

|  |  |  |  |
| --- | --- | --- | --- |
| **GestureType** | **int value** | **Support when depth is enable?** | **Support when depth is not enable?** |
| Gesture0 | 0 | Yes | Yes |
| Gesture1 | 1 | Yes | No |
| Gesture2 | 2 | No | Yes |
| Gesture5 | 5 | Yes | Yes |
| Gesture6 | 6 | Yes | Yes |
| Gesture7 | 7 | Yes | Yes |
| Gesture8 | 8 | No | Yes |
| Gesture10 | 10 | Yes | No |
| Other gestures | -1 | -- | -- |

Gesture0 Gesture1 Gesture2 Gesture5

Gesture6 Gesture7 Gesture8 Gesture10

###### public Vector3[] GetHandBox()

Illustration: Returns the left top and right bottom corner of the rectangle that cover the hand on phone’s screen. The return value is in NDC. The range of x, y is [-1, 1] and z is 0.

###### public Vector3 GetGestureCenter()

Illustration: Returns the center of the hand box.

###### public Vector4 GetGestureOrientation()

Illustration: Returns the gesture orientation with a quaternion.

###### public ARCoordinateSystemType GetSkeletonCoordinateSystemType()

Illustration: Returns the coordinate system of hand skeleton.

###### public int GetHandSkeletonCount()

Illustration: Returns the count of hand skeleton.

###### public void GetSkeletons(Dictionary<SkeletonPointName, SkeletonPointEntry> outSkeleton)

Illustration: Retrieve the data of each hand skeleton, and put it into a dictionary.

enum SkeletonPointName{

Root,

Pinky\_1, Pinky\_2 , Pinky\_3, Pinky\_4,

Ring\_1, Ring\_2, Ring\_3, Ring\_4,

Middle\_1, Middle\_2, Middle\_3, Middle\_4,

Index\_1, Index\_2, Index\_3, Index\_4,

Thumb\_1, Thumb\_2, Thumb\_3, Thumb\_4,

SKELETON\_LENGTH

};

###### public void GetSkeletonConnection(List<KeyValuePair<SkeletonPointName, SkeletonPointName>> outConnections)

Illustration: Retrieve the data of skeleton connections, and put it into a list.

#### ARPointCloud

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

##### Methods:

###### public void GetPoints(List<Vector3> pointList)

Illustration: Get the points’ coordinate in unity world coordinate space.

###### public long getTimestampNs()

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

###### public void Release()

Illustration: Release current pointcloud in case resource exhaust.

#### AREnvironmentalLight

##### Description: Inherits from MonoBehaviour. This class will set *\_GlobalLightEstimation* as a global ambient light intensity. Developers should use this global parameter in self-defined shaders.

#### ARCameraMetadataValue

##### Description: Struct to contain camera metadata's value. Application should retrieve the data according to value type. e.g. if *ValueType== typeof(byte)*, you should use ARCameraMetadataValue.AsByte() .

##### Methods:

###### public Type ValueType

Illustration: The type of meta data value.

###### public sbyte AsByte()

Illustration: Retrieve the byte value.

###### public int AsInt()

Illustration: Retrieve the int value.

###### public float AsFloat()

Illustration: Retrieve the float value.

###### public long AsLong()

Illustration: Retrieve the long value.

###### public double AsDouble()

Illustration: Retrieve the double value.

###### public ARCameraMetadataRational AsRational()

Illustration: Retrieve the rational value..

struct ARCameraMetadataRational { public int Numerator; public int Denominator; }

### Functionality class

#### AREnginesSelector

##### Description: Used to query the supported engines, Huawei AR Engine or ARCore. Also, you can set and get the engine at the running time. If you do not want to switch engines, just skip this class and the default engine is Huawei AR Engine.

##### Methods:

###### public static AREnginesSelector Instance

Illustration: Singleton of the class.

###### public AREnginesAvaliblity CheckDeviceExecuteAbility()

Illustration: Used to check the supported engines through the internet.

|  |  |
| --- | --- |
| **AREnginesAvaliblity** | **Description** |
| NONE\_SUPPORTED | None Engines are supported. |
| HUAWEI\_AR\_ENGINE | HUAWEI AR Engine is supported. |
| GOOGLE\_AR\_CORE | ARCore is supported. |
| ALL\_SUPPORTED | Both engines are supported. |

###### public AREnginesType SetAREngine(AREnginesType executor)

Illustration: Set the running time engine. The return value the engine that is actually used. Note: [*CheckDeviceExecuteAbility*](#_public_AREnginesAvaliblity_CheckDev)must be called ahead of this method.

|  |  |
| --- | --- |
| **AREnginesType** | **Description** |
| NONE | Invalid value. |
| HUAWEI\_AR\_ENGINE | HUAWEI AR Engine |
| GOOGLE\_AR\_CORE | ARCore |

###### public AREnginesType GetCreatedEngine()

Illustration: Returns the running time engine.

#### AREnginesApk

##### Description: Used to check the device compatible and to download the engines. If the running time engine is installed, the methods in this class will return immediately.

##### Methods:

###### public static AREnginesApk Instance

Illustration: Singleton of the class.

###### public ARInstallStatus RequestInstall(bool userRequestedInstall)

Illustration: Work if *userRequestedInstall* is set ture.If the running time engine is not installed or too old, this method will check compatible of the device. And, if the device is compatible, we will jump to the Huawei App market to download the engines. If the running time engine is installed and compatible, it will return *INSTALLED* immediately. We recommend you to call this method in *OnApplicationPause (unity method),* so that the app will run normally after the engine is installed.

|  |  |
| --- | --- |
| **ARInstallStatus** | **Description** |
| INSTALLED | Already installed. |
| INSTALL\_REQUESTED | Already requested install. |

This method may throw the following exceptions:

|  |  |
| --- | --- |
| **Exception Type** | **Description** |
| ARUnavailableDeviceNotCompatibleException | Device is not compatible. |
| ARUnavailableEmuiNotCompatibleException | Device is compatible, however EMUI is not compatible. |
| ARUnavailableUserDeclinedInstallationException | User decline installation. |
| ARUnavailableConnectServerTimeOutException | Network is not available. |

###### public ARAvailability CheckAvailability()

Illustration: Check the device availability asynchronously. You can call it in a coroutine.

|  |  |
| --- | --- |
| **ARAvailability** | **Description** |
| UNKNOWN\_ERROR | Unknow error. |
| UNKNOWN\_CHECKING | Is checking. |
| UNKNOWN\_TIMED\_OUT | Network time out. |
| UNSUPPORTED\_DEVICE\_NOT\_CAPABLE | Device is not compatible. |
| UNSUPPORTED\_EMUI\_NOT\_CAPABLE | Device is compatible, however EMUI is not compatible. |
| SUPPORTED\_NOT\_INSTALLED | Supported, but not installed. |
| SUPPORTED\_APK\_TOO\_OLD | Supported, but installed service is too old. |
| SUPPORTED\_INSTALLED | Supported and installed service is compatible. |

#### ARConfigBase

##### Description: Inherit from *ScriptableObject (unity class).* This is an abstract class. You should only use its subclasses, e.g. ARWorldTrackingConfig, ARWorldTrackingConfig , ARFaceTrackingConfig, ARBodyTrackingConfig, ARWorldBodyTrackingConfig, ARHandTrackingConfig, ARImageTrackingConfig.

#### ARWorldTrackingConfig

##### Description: A subclass of ARConfigBase, which is used to start world AR. You can set [*LightingMode*](#_LightingMode), [*PlaneFindingMode*](#_PlaneFindingMode), [*UpdateMode*](#_UpdateMode_1), *PowerMode*, *FocusMode,* *AugmetendImageDatabase* and *EnableCapability* through this config.

|  |  |
| --- | --- |
| **LightingMode** | **Description** |
| 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** | **Description** |
| 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** | **Description** |
| *BLOCKING* | In BLOCKING model, [ARSession.Update()](#_public_static_void) will wait until a new camera image is available. |
| *LATEST\_CAMERA\_IMAGE* | In this model, [ARSession.Update()](#_public_static_void)will return immediately without blocking. If no new camera image is available, then it will return the most recent ARFrme. |

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

|  |  |
| --- | --- |
| **FocusMode** | **Description** |
| HWAR\_FOCUS\_MODE\_FIXED | Fixed focus mode |
| HWAR\_FOCUS\_MODE\_AUTO | Auto focus mode |

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Constant value** | **Description** |
| EnableItem\_None | 0 | None |
| EnableItem\_Depth | 1 | Enable depth capability, enable by default |
| EnableItem\_Mask | 2 | Enable Mask capability |

#### ARFaceTrackingConfig

##### Description: A subclass of ARConfigBase, which is used to start face AR. You can set [*LightingMode*](#_LightingMode), [*UpdateMode*](#_UpdateMode_1), *PowerMode FocusMode* and *EnableCapability* through this config.

#### ARBodyTrackingConfig

##### Description: A subclass of ARConfigBase, which is used to start body AR. You can set [*LightingMode*](#_LightingMode), [*UpdateMode*](#_UpdateMode_1), *CameraLensFacing*, *PowerMode, FocusMode* and *EnableCapability* through this config.

#### ARWorldBodyTrackingConfig

##### Description: A subclass of ARConfigBase, which is used to start world AR and body AR at the same time. You can set [*LightingMode*](#_LightingMode), [*PlaneFindingMode*](#_PlaneFindingMode), [*UpdateMode*](#_UpdateMode_1), *PowerMode*, *FocusMode AugmetendImageDatabase* and *EnableCapability* through this config.

#### ARCameraConfig

##### Description: A configuration accessing the device's camera sensor.

##### Methods:

###### public Vector2Int GetImageDimensions()

Illustration: Gets the dimensions of the CPU-accessible image bytes for the camera configuration.

###### public Vector2Int GetTextureDimensions()

Illustration: Gets the dimensions of the GPU-accessible external texture for the camera configuration.

#### ARHandTrackingConfig

##### Description: A subclass of ARConfigBase, which is used to start hand AR. You can set [*LightingMode*](#_LightingMode), [*HandFindingMode*](#_PlaneFindingMode)*(*Obsolete*)*, *CameraLensFacing*, [*UpdateMode*](#_UpdateMode_1), *FocusMode,* *PowerMode* and *EnableCapability* through this config.

|  |  |
| --- | --- |
| **ARConfigCameraLensFacing** | **Description** |
| REAR | Use rear camera. |
| FRONT | Use front camera. |

|  |  |
| --- | --- |
| **ARConfigHandFindingMode** | **Description** |
| DISABLED | Disable the hand finding mode. |
| ENABLE\_2D | Enable 2D model, which uses 2D algorithm. |
| ENABLE\_3D | Enable 3D model, which uses 3D algorithm. |

Note: This Interface has already been Obsolete, use Enable\_Depth instead.

#### ARCameraMetadata

##### Description: This class is used to retrieve the metadata of camera.

##### Methods:

###### public List<ARCameraMetadataTag> GetAllCameraMetadataTags()

Illustration: Returns all the metadata tags. Note: The enumerations of *ARCameraMetadataTag* is the same as those in [NdkCameraMetadataTags.h](https://developer.android.com/ndk/reference/ndk_camera_metadata_tags_8h.html).

###### public List<ARCameraMetadataValue> GetValue(ARCameraMetadataTag cameraMetadataTag)

Illustration: Return the values of the specified tag.

#### ARSession

##### Description: Manages HUAWEI AR Engine system state. 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 static void CreateSession()

###### Illustration: Creates a new ARSession. This method may throw three types of exception:

|  |  |
| --- | --- |
| **Exception Type** | **Description** |
| ARUnavailableServiceNotInstalledException | If the HUAWEI AR Engine Service APK is not present. |
| ARUnavailableDeviceNotCompatibleException | This device is not supported. |
| 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 static void SetCameraTextureNameAuto ()

Illustration: Set the texture, which can holds the preview of camera, automatically.

###### public static void SetDisplayGeometry(float width, float height)

Illustration: Sets the display geometry with specified width and height in pixels. This width and height is usually the attribute of the view. If your app support screen rotation, you need to call this method after the screen rotate.

###### public static void Config(ARConfigBase config)

Illustration: Configure the ARSession. And the default configuration is [ARWorldTrackingConfig](#_ARWorldTrackingConfig). This method throws [ARUnsupportedConfigurationException](#_ARUnSupportedConfigurationException) if the *config* is not supported. Note that, if an EnableItem is not supported on current device, this item will be disabled after calling this function. Application can check this item after *Config()*.

###### public static void Resume()

Illustration: Resume the session. Note: If [ARSession.Stop](#_public_static_void_1) is called, you cannot resume this session.

###### public static void Pause ()

Illustration: Pause the current session. This method will stop the camera. The session can be resumed by calling [ARSession.*Resume*](#_public_static_void_2).

###### public static void Stop()

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

###### public static void 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 [UpdateMode](#_UpdateMode) is BLOCKING，this method will be blocked until a new camera image is available. Note: In order to have a better performance, this method should be called ahead of other methods in ARFrame. Typically, you set the script execute order of the script which contains this method ahead of default. This Method may throw the following exceptions:

|  |  |
| --- | --- |
| Exception Type | 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. SetCameraTextureNameAuto () is not called before Update() is called. |

###### public static ARAnchor AddAnchor(Pose pose)

Illustration: Adds a new tracking anchor with a specified pose into the system.

###### public static Matrix4x4 GetProjectionMatrix(float nearClipPlane, float farClipPlane)

Illustration: Returns the projection matrix, which is used to set the unity camera projection matrix. nearClipPlane and farClipPlane specifies the near and far clip plane in meters, respectively.

###### public static List<ARAnchor> GetAllAnchors() [Obsolete]

Illustration: Returns all known anchors. Recommend method is *ARFrame.GetAnchors(ARTrackableFilter.ALL).*

###### public static void RemoveAnchors(List<ARAnchor> anchors) [Obsolete]

Illustration: Removes all the specified anchors. Recommend method is [ARAnchor.Detach()](#_public_void_Detach()).

###### public static List<ARPlane> GetAllPlanes ()[Obsolete]

Illustration: Returns all the detected planes. Recommend methos is *ARFrame.GetTrackables<ARPlane>(ARTrackableFilter.ALL).*

###### public static boolean IsSupported(ARConfigBase config) [Obsolete]

Illustration: Returns true only.

###### public static void Resume(ARConfigBase config) [Obsolete]

Illustration: Starts or resumes the ARSession with the specified configuration. Recommend methos is [ARSession.Resume().](#_public_static_void_2)

###### public static ARCameraConfig GetCameraConfig()

Illustration: Gets the CameraConfig that the Session is currently using.

#### ARFrame

##### Description: This class is a snapshot of the HUAWEI AR Engine system*.*

##### Methods:

###### public static bool TextureIsAvailable()

Illustration: Check whether the texture is available. If it returns true, the texture can be used to render the background.

###### public static ARPointCloud AcquirePointCloud()

Illustration: Returns the point cloud in current frame.

###### public static Pose GetPose()

Illustration: Returns the Pose of the user's device in the world coordinate when current frame was captured.

###### public static long GetTimestampNs()

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

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

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

###### public static void GetTrackables<T>(List<T> trackableList, ARTrackableQueryFilter filter) where T:ARTrackable

Illustration: Retrieve the list according to the type of trackable and the filter. enum ARTrackableQueryFilter { ALL, NEW, UPDATED }.

###### public static List<ARAnchor> GetAnchors(ARTrackableQueryFilter filter)

Illustration: Returns the ARAnchors according to the filter. When filter is *ALL*, it returns all the anchors. If filter is *UPDATED*, it only returns the updated anchors. And this method will return null when filter is set *NEW,* since AR Engine will not add anchors automatically.

###### public List<ARHitResult> HitTest(Touch touch)

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.

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

Illustration: Same as *HitTest(Touch touch)*. *xPX* is the x coordinate in pixels, and *yPX* is the y coordinate in pixels. Note that the pixels is in unity screen coordinate space.

###### public static ARCameraMetadata GetCameraMetadata()

Illustration: Returns the metadata of Camera.

###### public static bool IsDisplayGeometryChanged()

Illustration: Returns true if the display geometry changed. If the return value is true, you should call *GetTransformDisplayUvCoords* to get the new texture coordinate.

###### public static float[] GetTransformDisplayUvCoords (float[] inUVCoords)

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()*.

###### public static ARPointCloud GetPointCloud()[Obsolete]

Illustration: Returns the point cloud in current frame. Recommend method is [ARFrame.AcquirePointCloud().](#_public_static_ARPointCloud)

###### public static List<ARPlane> GetPlanes(ARTrackableQueryFilter filter) [Obsolete]

Illustration: Returns the ARPlanes according to the filter. Recommend method is [*GetTrackables<ARPlane>()*](#_public_static_void_3)*.*

###### public static bool IsDisplayRotationChanged()[Obsolete]

Illustration: Checks if the display rotation or viewport geometry changed since the previous Frame. Recommend method is [*IsDisplayGeometryChanged*](#_public_static_bool) *.*

#### ARHitResult

##### Description: Defines an intersection between a ray and estimated real-world geometry.

##### Methods:

###### public Pose HitPose

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 an 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 forward 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 forward the user's device.

###### public float Distance

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

###### public ARTrackable GetTrackable()

Illustration: Returns the related trackable.

###### public ARAnchor CreateAnchor()

Illustration: Create an Anchor at the hit pose.

#### ARPlaneHitResult [Obsolete]

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

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

###### public ARPlane Plane [Obsolete]

Illustration: Returns the hit ARPlane. Recommend method is [ARHitResult. GetTrackable](#_public_ARTrackable_GetTrackable()).

###### public bool IsHitInExtents [Obsolete]

Illustration: Checks if the intersection is in the rectangular extents of ARPlane. Recommend method is [ARPlane.IsPoseInExtents](#_public_bool_IsPoseInExtents(Pose).

###### public bool IsHitInPolygon [Obsolete]

Illustration: Checks if the intersection is in the polygon of ARPlane. Recommend method is [ARPlane. IsPoseInPolygon .](#_public_bool_IsPoseInPolygon(Pose)

#### ARPointCloudHitResult [Obsolete]

##### 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 PointCloud [Obsolete]

Illustration: Returns the hit ARPointCloud.

#### BackGroundRenderer

##### Description: Inherits from MonoBehaviour. This class can be used to render the background automatically.

#### AsyncTask<T>

##### Description: A class used for monitoring the status of an asynchronous task.

##### Methods:

###### public bool IsTaskCompleted

Illustration: Gets a value indicating whether the task is complete.

###### public T TaskResult

Illustration: Gets the result of a completed task.

###### public AsyncTask<T> ThenAction(Action<T> actionAfterTask)

Illustration: Performs an action (callback) after task completion.

###### public CustomYieldInstruction GetWaitForCompletionYieldInstruction()

Illustration: Returns a yield instruction that monitors this task for completion within a coroutine.

#### AndroidPermissionsRequest

##### Description: Request android permission in the run time.

##### Methods:

###### public static AsyncTask<AndroidPermissionsRequestResult> RequestPermission(string[] permissionNames)

Illustration: Requests an Android permission from the user.

###### public static bool IsPermissionGranted(string permissionName)

Illustration: Checks if an Android permission is granted to the application.

#### AndroidPermissionsRequestResult

##### Description: The result of permission request.

##### Methods:

###### public bool IsAllGranted

Illustration: Gets a value indicating whether all permissions are granted.

### Exception

#### ARMissingGlContextException

##### Description: Thrown when the opengles context is missing in current thread.

#### ARNotTrackingException

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

#### ARSessionNotPausedException

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

#### ARSessionPausedException

##### Description: 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()*.

#### ARTextureNotSetException

##### Description: Thrown if a texture name was not set by calling *ARSession.SetCameraTextureNameAuto()* before the first call *ARSession.Update()*.

#### ARCameraPermissionDeniedException

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

#### ARUnSupportedConfigurationException

##### Description: Thrown if the configuration supplied to ARSession.Config() is unsupported.

#### ARUnavailableException

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

#### ARUnavailableClientSdkTooOldException

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

#### ARUnavailableServiceApkTooOldException

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

#### ARUnavailableDeviceNotCompatibleException

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

#### ARUnavailableEmuiNotCompatibleException

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

#### ARUnavailableServiceNotInstalledException

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

#### ARUnavailableUserDeclinedInstallationException

##### Description: Thrown when requestInstall is called after the user had previously cancelled installation.

#### ARUnavailableConnectServerTimeOutException

##### Description:Thrown when requestInstall connect server timeout.

#### ARNotYetAvailableException

##### Description:Acquire failed because the object being acquired is not yet available.

#### ARResourceExhaustedException

##### Description: Acquire failed because there are too many objects already acquired.

#### ARDeadlineExceededException

##### Description: Thrown while the current resource has been released.