ViveSR Unity API Reference

Use [Ctrl] + [F] to search the detail description.

Classes	
ViveSR	Manage all features.
ViveSR_Framework	C API wrapper for the framework.
ViveSR_DualCameraRig	Manage the feature of See-Through.
ViveSR_DualCameraImageCapture	C API wrapper for the See-Through.
ViveSR_DualCameraImageRenderer	Manage the texture and material of See-Through.
ViveSR_TrackedCamera	Mediators between ViveSR_DualCameraRig and image planes.
ViveSR_HMDCameraShifter	Align the HMD's position with the dual camera.
ViveSR_DualCameraDepthExtra	C API wrapper for extra usages of Depth.
ViveSR_DualCameraDepthCollider	Manage the dynamic meshes and colliders generated by depth.
ViveSR_DualCameraCalibrationTool	Calibrate displayed plane of the dual camera.
ViveSR_RigidReconstruction	C API wrapper for the RECONSTRUCTION.
ViveSR_RigidReconstructionRenderer	Manage the feature of RECONSTRUCTION.
ViveSR_SceneUnderstanding	Manage scene understanding features of RECONSTRUCTION.
ViveSR_StaticColliderInfo	Auto-Generated collider info from the imported/loaded collider file.
ViveSR_StaticColliderPool	Manage the imported/loaded collider info
ViveSR_FileTool	Save and Load files.
ViveSR_Timer	Record the time of specified block and analyze the data.
ViveSR_Enums	Enumerations used by ViveSR plugin.

ViveSR

Manage all features.

Summary

Control the lifecycle of the framework.

Public attributes	
EnableAutomatically	bool Enable this to launch the framework when this component start.
EnableSeeThroughModule	bool Enable this to launch the SeeThrough module before ViveSR_InitialFramework() start.
EnableDepthModule	bool Enable this to launch the Depth module when before ViveSR_InitialFramework () start.
EnableReconstructionModule	bool Enable this to launch the Reconstruction module before ViveSR_InitialFramework() start.
EnableAIModule	bool Enable this to launch the AI module before ViveSR_InitialFramework() start.
DualCameraRig	ViveSR_DualCameraRig
OnStartFailed	List<UnityAction> Execute those Actions right after failing to start the SR Framework.
OnStartComplete	List<UnityAction> Execute those Actions right after completing starting SR Framework.

Public static attributes	
FrameworkStatus	Status The status of the framework.
LastError	string The error message of the latest operation of this script.
Instance	ViveSR Return the instance of this singleton class.

Public functions	
StartFramework()	void Run the initialization of SR Framework and plugin in the background.
StopFramework()	void Turn off and release the SR Framework and plugin.

Protected functions	
ViveSR_InitialFramework ()	int Initialize specific modules of SR Framework.
ViveSR_StartFramework ()	int Start specific modules of SR Framework.
ViveSR_StopFramework()	int

	Stop and release all modules of SR Framework.
StartFrameworkCoroutine()	IEnumerator Do initialization of SR Framework and plugin.

Public attributes

Status FrameworkStatus

The status of the framework.

bool EnableAutomatically

Enable this to launch the framework when this component start.
Enable by default.

bool `EnableSeeThroughModule`

Enable this to launch the SeeThrough module before ViveSR_InitialFramework() start.
Enable by default.

bool `EnableDepthModule`

Enable this to launch the Depth module when before ViveSR_InitialFramework () start.
Enable by default.

bool `EnableReconstructionModule`

Enable this to launch the Reconstruction module before ViveSR_InitialFramework() start.
Enable by default.

bool `EnableAIModule`

Enable this to launch the AI module before ViveSR_InitialFramework() start.
Enable by default.

Public static attributes

ViveSR Instance

Return the instance of this singleton class.

ViveSR_Framework

C API wrapper for the framework.

Summary

Control the lifecycle of the framework.

Public static functions	
Initial()	int Initialize necessary components.
Stop()	int Stop and release the whole SR framework.
CreateModule(int ModuleType, ref int moduleID)	int Create the specified module.
StartModule(int moduleID)	int Start the specified module.
ModuleLink(int moduleIDfrom, int moduleIDto, int mode)	int Let moduleIDto be able to receive the output from moduleIDfrom.
GetMultiDataSize(int moduleID, DataInfo[] data, int size)	int Get specified data size from specified module.
GetMultiData(int moduleID, DataInfo[] data, int size)	int Get specified data from specified module.
RegisterCallback(int moduleID, int type, System.IntPtr callback)	int Start to listen the output from the specified module and type.
UnregisterCallback(int moduleID, int type, System.IntPtr callback)	int Stop listening the output from the specified module and type.
GetParameterBool(int moduleID, int type, ref bool parameter)	int Get the parameter from the specified module.
SetParameterBool(int moduleID, int type, bool parameter)	int Set the parameter to the specified module and type.
GetParameterInt(int moduleID, int type, ref int parameter)	int Get the parameter from the specified module.
SetParameterInt(int moduleID, int type, int parameter)	int Set the parameter to the specified module and type.
GetParameterFloat(int moduleID, int type, ref float parameter)	int Get the parameter from the specified module.
SetParameterFloat(int moduleID, int type, float parameter)	int Set the parameter to the specified module and type.
GetParameterDouble(int moduleID, int type, ref double parameter)	int Get the parameter from the specified module.
SetParameterDouble(int moduleID, int type, double parameter)	int Set the parameter to the specified module and type.
SetParameterString(int moduleID, int type, string parameter)	int Get the parameter from the specified module.
GetParameterStruct(int moduleID, int type, ref System.IntPtr parameter)	int Get the parameter from the specified module.
SetParameterStruct(int moduleID, int type, System.IntPtr parameter)	int Set the parameter to the specified module and type.
GetParameterNativePtr(int moduleID, int type, ref System.IntPtr parameter)	int Get the parameter from the specified module.
SetParameterNativePtr(int moduleID, int type, System.IntPtr parameter)	int Set the parameter to the specified module and type.

GetParameterFloatArray(int moduleID, int type, ref float[] parameter)	int Get the parameter from the specified module.
SetParameterFloatArray(int moduleID, int type, float[] parameter)	int Set the parameter to the specified module and type.
SetCommandBool(int moduleID, int type, bool content)	int Send a Boolean to the specified module.
SetCommandInt(int moduleID, int type, int content)	int Send an integer to the specified module.
SetCommandFloat(int moduleID, int type, float content)	int Send a float to the specified module.
SetCommandString(int moduleID, int type, string content)	int Send a string to the specified module.
SetCommandFloat3(int moduleID, int type, float content0, float content1, float content2)	int Send three float to the specified module.
ChangeModuleLinkStatus(int from, int to, int mode)	int Change the linked status between two modules.
GetPointer(int key, int type, ref System.IntPtr ptr)	int Get pointer of specified type by key.
SetLogLevel(int level)	int Set the log level of SR framework.

ViveSR_DualCameraRig

Manage the feature of See-Through.

Summary

Control the status of camera and display method.

Public attributes	
OriginalCamera	Camera Keep the original main camera before changing the display mode.
VirtualCamera	Camera A camera responsible for rendering virtual game objects.
DualCameraLeft	Camera A camera responsible for rendering the left-eye image plane.
DualCameraRight	Camera A camera responsible for rendering the right-eye image plane.
DualCameraImageRenderer	ViveSR_DualCameraImageRenderer Return the reference of the ViveSR_DualCameraImageRenderer.
DualCameraCalibration	ViveSR_DualCameraCalibrationTool Return the reference of the ViveSR_DualCameraCalibrationTool.
TrackedCameraLeft	ViveSR_TrackedCamera Return the reference of the left trackedCamera.
TrackedCameraRight	ViveSR_TrackedCamera Return the reference of the right trackedCamera.
HMDCameraShifter	ViveSR_HMDCameraShifter Keep the distance
Mode	DualCameraDisplayMode Return current display mode of see-through.
OnInitialComplete	List<UnityAction> Execute those Actions right after completing initialization.
OnInitialFailed	List<UnityAction> Execute those Actions right after failing to initialize dual camera.

Public static attributes	
DualCameraStatus	DualCameraStatus Return the status of dual camera device.
Instance	ViveSR Return the instance of this singleton class.

Public functions	
Initial()	void Initialize the dual camera settings.
Release()	void release the dual camera settings.
SetMode(DualCameraDisplayMode mode)	void Change modes between VR and MR.

ViveSR_DualCameraImageCapture

C API wrapper for the See-Through.

Summary

Get dual camera data from SR framework. Operate camera and depth settings.

Public static attributes	
DistortedPose	Matrix4x4 Return the last posture at the moment of capturing frame.
UndistortedPose	Matrix4x4 Return the last posture at the moment of capturing frame.
DepthPose	Matrix4x4 Return that last posture at the moment of capturing frame.
FocalLength	float[] Return the focal length. [0 left, 1 right]
DistortedImageWidth	int Return the width of the source distorted image.
DistortedImageHeight	int Return the height of the source distorted image.
DistortedImageChannel	int Return the channel of the source distorted image.
UndistortedImageWidth	int Return the width of the undistorted image.
UndistortedImageHeight	int Return the height of the undistorted image.
UndistortedImageChannel	int Return the channel of the undistorted image.
DepthImageWidth	int Return the width of the depth image.
DepthImageHeight	int Return the height of the depth image.
DepthImageChannel	int Return the channel of the depth image.
DepthDataSize	int Return the size of pixel element data type.
DistortedFrameIndex	int Return the frame index of the source distorted image.
DistortedTimeIndex	int Return the time index of the source distorted image.
UndistortedFrameIndex	int Return the frame index of the undistorted image.
UndistortedTimeIndex	int Return the time index of the source undistorted image.
DepthFrameIndex	int Return the frame index of the depth image.
DepthTimeIndex	int Return the time index of the source depth image.
DepthProcessing	bool Return true when depth module is processing.
DepthRefinement	bool Return true when depth refinement is enable.

DepthEdgeEnhance	bool Return true when depth edge enhance is enable.
DepthCase	DepthCase Current depth case.
DepthConfidenceThreshold	float Return the threshold for confidence in depth.
DepthDenoiseGuidedFilter	float Return the level for guided filter.
DepthDenoiseMedianFilter	float Return the level for median filter.

Public static functions	
Initial()	int Initial settings.
Release()	void Release resources.
SetMode(DualCameraDisplayMode mode)	void
RegisterDistortedCallback()	int Register callback to listen distorted image outputs.
RegisterUndistortedCallback()	int Register callback to listen un-distorted image outputs.
RegisterDepthCallback()	int Register callback to listen depth image outputs.
UnregisterDistortedCallback()	int Unregister callback to stop listening distorted image outputs.
UnregisterUndistortedCallback()	int Unregister callback to stop listening un-distorted image outputs.
UnregisterDepthCallback()	int Unregister callback to stop listening depth image outputs.
GetDistortedTexture(out Texture2D imageLeft, out Texture2D imageRight, out int frameIndex, out int timeIndex, out Matrix4x4 poseLeft, out Matrix4x4 poseRight)	void Return latest distorted image data stored in local memory.
GetUndistortedTexture(out Texture2D imageLeft, out Texture2D imageRight, out int frameIndex, out int timeIndex, out Matrix4x4 poseLeft, out Matrix4x4 poseRight)	void Return latest un-distorted image data stored in local memory.
GetDepthTexture(out Texture2D imageDepth, out int frameIndex, out int timeIndex, out Matrix4x4 pose)	void Return latest depth image data stored in local memory.
UpdateDistortedImage()	void Copy distorted data from SDK to local memory.
UpdateUndistortedImage()	void Copy un-distorted data from SDK to local memory.
UpdateDepthImage()	void Copy depth data from SDK to local memory.
Rotation(Matrix4x4 m)	Quaternion

	Extract a rotation from the input matrix.
Position(Matrix4x4 m)	Vector3 Extract a position from the input matrix.
EnableDepthProcess(bool active)	int Enable or disable the depth processing.
EnableDepthRefinement(bool active)	int Enable or disable the depth refinement.
EnableDepthEdgeEnhance(bool active)	int Enable or disable the depth edge enhancement.
SetDefaultDepthCase(DepthCase depthCase)	int Set the default depth case before starting the depth module.
ChangeDepthCase(DepthCase depthCase)	int Change the depth case after starting the depth module.
GetCameraQualityInfo(CameraQuality item, ref CameraQualityInfo paramInfo)	int Get camera quality information.
SetCameraQualityInfo(CameraQuality item, CameraQualityInfo paramInfo)	int Set camera quality information.

Public types	

CameraQualityInfo
{
public Int32 Status;
public Int32 DefaultValue;
public Int32 Min;
public Int32 Max;
public Int32 Step;
public Int32 DefaultMode;
public Int32 Value;
public Int32 Mode;
}
	struct
	Mode: AUTO = 1, MANUAL = 2

CameraQuality
{
BRIGHTNESS = 100,
CONTRAST = 101,
HUE = 102,
SATURATION = 103,
SHARPNESS = 104,
GAMMA = 105,
WHITE_BALANCE = 107,
BACKLIGHT_COMPENSATION = 108,
GAIN = 109,
}
	enum

Public static functions	
Initial()	int Initialize the image capturing tool.
CheckNecessayFile()	bool Check the necessary file and copy it to the correct path.
RegisterDistortedCallback()	int Register the callback function of receiving source image.
RegisterUndistortedCallback()	int Register the callback function of receiving undistorted image.
RegisterDepthCallback()	int Register the callback function of receiving depth image.
UnregisterDistortedCallback()	int Unregister the callback function of receiving source image.
UnregisterUndistortedCallback()	int Unregister the callback function of receiving undistorted image.
UnregisterDepthCallback()	int Unregister the callback function of receiving depth image.
GetDistortedTexture(out Texture2D imageLeft, out Texture2D imageRight, out int frameIndex, out int timeIndex)	void Get the source image texture from the buffer of this class.
GetUndistortedTexture(out Texture2D imageLeft, out Texture2D imageRight, out int frameIndex, out int timeIndex)	void Get the undistorted image texture from the buffer of this class.
GetDepthTexture(out Texture2D imageDepth, out int frameIndex, out int timeIndex)	void Get the depth image texture from the buffer of this class.
UpdateDistortedImage()	void Get the source image from framework and save the data in this class.
UpdateUndistorted()	void Get the undistorted image from framework and save the data in this class.
UpdateDepth()	void Get the depth image from framework and save the data in this class.
UpdateDistortedCallback(IntPtr left, IntPtr right, int frame, int time)	void Be called when framework generate a new distorted image.
UpdateUndistortedCallback(IntPtr left, IntPtr right, int frame, int time)	void Be called when framework generate a new undistorted image.
UpdateDepthCallback(IntPtr left, IntPtr depth, int frame, int time)	void Be called when framework generate a new depth image.
EnableDepthProcess(bool active)	int Enable or disable the depth processing.

ViveSR_DualCameraImageRenderer

Manage the texture and material of See-Through.

Summary

It would control whether to get texture and update material.

Public attributes	
DistortedLeft	List<Material> Assign the material which needed to update source image texture.
DistortedRight	List<Material> Assign the material which needed to update source image texture.
UndistortedLeft	List<Material> Assign the material which needed to update undistorted image texture.
UndistortedRight	List<Material> Assign the material which needed to update undistorted image texture.
Depth	List<Material> Assign the material which needed to update depth image texture.

Public static attributes	
UpdateDistortedMaterial	bool Set true if you want to update source image.
UpdateUndistortedMaterial	bool Set true if you want to update undistorted image.
UpdateDepthMaterial	bool Set true if you want to update depth image.
CallbackMode	bool Change the method of capturing new image.

ViveSR_TrackedCamera

Mediators between ViveSR_DualCameraRig and image planes.

Summary

represent the dual camera real posture and store references of the image plane.

Public attributes	
DualCameraIndex	DualCameraIndex Indicate which eye it represent.
Anchor	Transform Eliminate the offset between the camera and the head.
ImagePlane	ViveSR_DualCameraImagePlane See-through image plane.
ImagePlaneCalibration	ViveSR_DualCameraImagePlane See-through image plane for calibration.

ViveSR_HMDCameraShifter

Align the HMD's position with the dual camera.

Summary

In order to let the position of HMD is same as the center of dual camera. Therefore, the virtual things can align with the real things.

Public attributes	
CameraShiftZ	float The distance between head and the center of dual camera.

ViveSR_DualCameraDepthExtra

C API wrapper for extra usages of Depth.

Summary

Using depth frame data generates real-time mesh collider for real-time interaction.

Public static attributes	
DepthColliderTimeIndex	int Return time index of the source depth mesh collider.
ColliderVerticeNum	int Return number of mesh vertices
ColliderBytePervert	int Return data size per vertices (current only support XYZ)
ColliderIndicesNum	int number of mesh vertices indices

Public static functions	
InitialDepthCollider(int depthImageWidth, int depthImageHeight)	void Initial collider mesh data memory depending on depth size
GetDepthColliderFrameInfo()	bool Return frame information (ex. Time index, frame index and data size per mesh)
GetDepthColliderData(ref int verticesNum, out float[] verticesBuff, ref int indicesNum, out int [] indicesBuff)	bool input: number of vertices and indices Return: vertices and indices mesh data buffer

ViveSR_DualCameraDepthCollider

Manage the dynamic meshes and colliders generated by depth.

Summary

Provide different specific setting (distance truncation / mesh quality / mesh visibility) for custom adjustment.

Public static attributes	
UpdateColliderNearDistance	float Return and set near distance truncation
UpdateColliderFarDistance	float Return and set far distance truncation
UpdateDepthCollider	bool Return and set turning on/off depth collider
UpdateDepthColliderRange	bool Return and set turning on/off distance truncation
UpdateDepthColliderHoleFilling	bool Return and set turning on/off collider mesh hole filling
ColliderMeshVisibility	bool Return mesh visibility status
ColliderDefaultMaterial	Material Return default collider mesh material

Public static functions	
ChangeColliderMaterial(Material mat)	bool Change different wireframe material
SetColliderProcessEnable(bool value)	bool Set turning on/off collider mesh generation
SetDepthColliderHoleFilling(bool value)	bool Set turning on/off collider mesh hole filling
SetColliderRangeEnable(bool value)	bool Set turning on/off mesh distance truncation
SetLiveMeshVisibility(bool value)	bool Set showing or hiding mesh
SetColliderEnable(bool value)	bool Set turning on/off unity physical collider
GetQualityScale(out int value)	bool Return down-sampling number
SetQualityScale(out int value)	bool Set down-sampling number (Suggestion: 8)
SetDepthColliderNearDistance(double value)	bool Set near distance truncation
SetDepthColliderFarDistance(double value)	bool Set far distance truncation

ViveSR_DualCameraCalibrationTool

Calibrate displayed plane of the dual camera.

Summary

Calibration the displayed plane position and reuse the settings.

Public static attributes	
IsCalibrating	bool Return true when calibration tool is enable.
CurrentCalibrationType	CalibrationType Return the current type of calibration tool.

Public functions	
SetCalibrationMode(bool active, CalibrationType calibrationType = CalibrationType. ABSOLUTE)	void Enable/Disable calibration tool and determine the calibration type.
Calibration(CalibrationAxis axis, float angle)	void Calibration the angle on the axis.
ResetCalibration()	void Reset all of calibration parameters.
LoadDeviceParameter()	void Load the parameters of calibration from registry.
SaveDeviceParameter()	void Save the parameters of calibration into registry.

ViveSR_RigidReconstruction

C API wrapper for the rigid reconstruction.

Summary

The global command API of rigid reconstruction.

Public static attributes	
IsScanning	bool Returns true if scanning scene data currently.
IsExportingMesh	bool Returns true if exporting model currently.

Public functions	
InitRigidReconstructionParamFromFile(string configFile)	bool Set the path of configure file of rigid reconstruction.
GetRigidReconstructionIntParameter(int type)	int Get the parameter according to the type.
AllocOutputDataMemory()	void Initialize and allocate every necessary data.
GetRigidReconstructionFrame(ref int frame)	bool Return true after get the latest frame index.
GetRigidReconstructionData (ref int frame, out float[] pose, ref int verticesNum, out float[] verticesBuff, ref int vertStrideInFloat, out int[] sectorIDList, ref int sectorNum, out int[] sectorVertNum, out int[] sectorMeshIdNum, ref int indicesNum, out int[] indicesBuff)	bool Return true if the reconstructed data is updated. **Pose**: The tracked camera pose matrix **verticesNum**: Number of the updated vertices in the buffer, if reconstruction is not updated, the value will be the same as the previous one. **verticesBuff**: The vertices data **vertStridInFloat**: The length of each vertex, how many sizeof(float) for each vertex. When "EnableSector" is on, the following data will have value. **sectorIDList**: Updated sector IDs in this frame. **sectorNum**: The number of updated sectors in this frame. **sectorVertNum**: The number of vertices in each sector **sectorMeshIdNum**: The number of indices in each sector. If display mode is not mesh, it is all zeros. When DisplayMode is "Adaptive Mesh", the following data will have value. **indicesNum**: Number of valid indices data in the buffer. If reconstruction is not updated, the value will be the same as the previous one. **indicesBuff**: The indices data
RegisterReconstructionCallback()	void Register the default callback function to rigid reconstruction module.
UnregisterReconstructionCallback()	void Unregister the default callback function from rigid reconstruction

	module.
ExportModel(string filename)	void Start to export the model and save with the specified name in the folder “\Recons3DAsset\”
GetExportProgress(ref int stage, ref int percentage)	void Get the current exporting stage and stage percentage of exporting model.
GetExportProgress(ref int percentage)	void Get the whole percentage of exporting model.
EnableLiveMeshExtraction(bool enable)	void Enable the live mesh extraction.
SetLiveMeshExtractionMode (ReconstructionLiveMeshExtractMode mode)	void Set the mode of live mesh extraction.
StartRunning()	void Start to scan scene for reconstruction.
StopRunning()	void Stop to scan scene for reconstruction.

ViveSR_RigidReconstructionRenderer

Manage the feature of RECONSTRUCTION.

Summary

Control the reconstruction life cycle.

Public attributes	
ConfigFilePath	string The path of configure file for rigid reconstruction.
FullSceneQuality	ReconstructionQuality The quality setting, it affects Full Scene Point and the exported color quality and need to set before starting.
LiveMeshMode	ReconstructionLiveMeshExtractMode The extraction mode of live mesh.
RefreshIntervalMS	int The reconstruction refreshing interval time in millisecond
EnableSector	bool Enable or disable making the scanned mesh divided into sector, it can keep the sectors on the screen during live mesh extraction.
MaxActiveGO	Int The max number of visible sectors when EnableSector is on.
NumOfActiveGO	Int The number of rendered sectors. (ReadOnly)

Public static attributes	
LiveMeshDisplayMode	ReconstructionDisplayMode Set the display mode of live extraction.
Instance	ViveSR_RigidReconstructionRenderer Return the instance of this singleton class.

Public functions	
InitRigidReconstructionParam()	void Set the parameter to rigid reconstruction for initialization.

ViveSR_SceneUnderstanding

Manage scene understanding features of RECONSTRUCTION.

Summary

Scene Understanding detects objects of interest in player's surroundings using advanced machine learning capabilities.

Public helper class	
SceneUnderstandingObjects	class Helper class that loads scene descriptions comprising objects of interest and retrieve specific type of objects

Element
{
public string tag;
public int id;
public string objfilename;
public string cldfilename;
public List<Vector3> position;
public Vector3 forward;
public Vector3 bBoxMinPoint;
public Vector3 bBoxMaxPoint;
}
	Struct
	tag: target scene type
	id: scene object order id
	objfilename: mesh obj filename
	cldfilename: collider mesh obj filename
	position: array of usable position in the target scene object (ex. Chair seat position)
	forward: normal of scene object
	bBoxMinPoint: minimal point of the bounding box
	bBoxMaxPoint: maximal point of the bounding box

SceneUnderstandingObjects(string fileDir)	constructor Load up previously exported XML file stored in /Recons3DAsset/ SemanticIndoorObj folder which records the objects (.xml, mesh.obj and mesh_cld.obj) detected by Scene Understanding
GetElementsBoundingBoxMeshes(int tagObj, Element tagIdElement, ref List<GameObject> boxObj)	void Get bounding box gameOject according to targeted scene object type
GetElementsIcons (int tagObj, Element tagIdElement, ref List<GameObject> iconObj)	void Get scene object icon gameOject according to targeted scene object type
GetElements(int tag)	Elements[] Get detected object elements via enum SceneUnderstandingObjectType
GetElements()	Elements[] Get all detected object elements
GetElements(string tag)	Elements[] Get targeted type name object elements
GetElementName(int tag)	string Get detected object elements name string
GetNElement ()	int Get detected object elements total number

Public helper struct	

SceneUnderstandingConfig
{
public int nFloorMaxInst;
public int nWallMaxInst;
public int nCeilingMaxInst;
	struct
	Sets to the engine the maximum number of objects to detect. Can be set to zero to avoid certain object types.

public int nChairMaxInst;
public int nTableMaxInst;
public int nBedMaxInst;
}	

SceneObject
{
public int objTypeID;
public int objID;
public string objFileName;
public string cldFileName;
public List<GameObject> BoundingBoxGameObj;
public List<GameObject> IconGameObj;
}
	struct
	Sets to the engine the maximum number of objects to detect. Can be set to zero to avoid certain object types.

Public static attributes	
IsEnabledSceneUnderstanding	bool Returns true if scene understanding is enabled
IsEnabledSceneUnderstandingRefinement	bool Returns true if scene understanding refinement is enabled
IsEnabledSceneUnderstandingView	bool Returns true if real-time scene understanding view is enabled
IsExportingSceneUnderstandingInfo	bool Returned true when engine is exporting objects of interest as XML file, which afterwards can be loaded into Unity via class SceneUnderstandingObjects
ShowSceneObjects	List<SceneObject> Detected scene objects list for game object operation

Public static methods	
EnableSceneUnderstanding(bool enable)	void Instructs rigid reconstruction engine to start analyze player's surroundings. For supported objects please refer to enum SceneUnderstandingObjectType
EnableSceneUnderstandingRefinement (bool enable)	void Enables scene understanding refinement. Note it requires some warmup time to be complete this operation
EnableSceneUnderstandingView(bool enable)	void Enables AI vision-like view of current surroundings. Note it requires some warmup time to be complete this operation
ExportSceneUnderstandingInfo(string filename)	void Exports a human-readable XML file containing objects of interest to the folder "\Recons3DAsset\ SemanticIndoorObj". To easily get objects of interest, use the provided helper class SceneUnderstandingObjects
GetSceneUnderstandingProgress()	int Get the progress of Scene Understanding Info or Scene Understanding View whose range is from 0 until completion 100
GetSceneUnderstandingConfig(ref SceneUnderstandingConfig config)	SceneUnderstandingConfig Returns current scene understanding configurations; refer to

	respective struct SceneUnderstandingConfig for available parameters
SetSceneUnderstandingConfig(SceneUnderstandingConfig config)	void Set understanding configurations
SetCustomSceneUnderstandingConfig(int objectType, int objectMaxNum, bool isOn)	void Set custom understanding configurations for each object type
SetAllCustomSceneUnderstandingConfig(int objectMaxNum, bool isOn)	void Set custom understanding configurations for all same parameter
DestroySceneObjects()	void Destroy SceneObject list
SetIconLookAtPlayer(Transform player)	void Let SceneObject's icons frontal orientation with player
ShowSemanticBoundingBoxAndIconWithType(int objType, bool boxIsVisible, bool iconIsVisible)	bool Set targeted bounding box and icon with scene type to enable visible Returns true if found scene object
SetAllSemanticBoundingBoxAndIconVisible(bool boxIsVisible, bool iconIsVisible)	void Set all bounding boxes and icons to enable visible Returns true if found scene object
ShowAllSemanticBoundingBoxAndIcon()	void Show all semantic bounding boxes And Icons
HideAllSemanticBoundingBoxAndIcon()	void Hide all semantic bounding boxes And Icons
ShowSemanticBoundingBoxAndIconWithId(int objType, int objId, bool IsShowingBox, bool IsShowingIcon)	void Set targeted bounding box and icon with scene type and object id to enable visible
ImportSceneObjectsByType(string dirPath, int objType)	void Import targeted scene object type to SceneObject list from .xml
ImportSceneObjects (string dirPath)	Void Import all scene objects to SceneObject list from .xml
GetColliderFileNames()	string[] Get all collider mesh filename from SceneObject list
GetPlacedPositionsByID (int objType, int objId)	List<Vector3> Get placed position list of targeted ID and type objects from SceneObject list
GetPlacedPositionsByType(int objType)	List<Vector3> Get placed position list of targeted type objects from SceneObject list
GetAllPlacedPositions	List<Vector3> Get all placed position list of scene objects from SceneObject list

ViveSR_StaticColliderInfo

Summary

The information of each collider in the reconstruction data.

Public types	

ColliderShapeType
{
UNDEFINED = 0,
CONVEX_SHAPE = 1,
BOUND_RECT_SHAPE = 2,
MESH_SHAPE = 4,
}
	enum
	Shape types of the collider.

PlaneOrientation
{
UNDEFINED = 0,
HORIZONTAL = 8,
VERTICAL = 16,
OBLIQUE = 32,
FRAGMENT = 64,
}
	Enum
	Orientation of the collider.

ColliderCondition
{
NONE,
LARGEST,
CLOSEST,
FURTHEST
}
	Enum
	Conditions of plane,

Public attributes	
ApproxArea	float Return the approximate area of the collider.
GroupNormal	Vector3 Return the approximate normal vector of the collider.
RectWidth	Float Return the width of the bounding rectangle of the collider.
RectHeight	Float Return the height of the bounding rectangle of the collider.
RectRightAxis	Vector3 Return the right axis of the bounding rectangle of the collider.
ID	Int Return the ID of the collider.

Public functions	
SetBit(uint bit)	bool Set the corresponding bit of the shape type and the orientation to the collider.
CheckHasAllBit(uint bit)	bool Check if the collider matches the corresponding bit of the

	shape type and the orientation.
SetCorrespondingColliderOfType(ColliderShapeType type, ViveSR_StaticColliderInfo info)	Void Set corresponding ColliderInfo with type.
GetCorrespondingColliderOfType(ColliderShapeType type)	ViveSR_StaticColliderInfo Get corresponding ColliderInfo with type
GetColliderUsableLocations(float intervalW, float intervalH, float surf_shift, List<Vector3> outLocations, out Quaternion rotation)	Void Set the distance between the center of each placed object with intervalW and intervalH. Set the distance between the center of placed object and this collider with surf_shift. Return object-placed locations and the orientation for placed objects.
GetColliderUsableLocationsWithRightAxis(float intervalW, float intervalH, float surf_shift, List<Vector3> outLocations, out Quaternion rotation, ref Vector3 rightVec)	Void Set the distance between the center of each placed object with intervalW and intervalH. Set the distance between the center of placed object and this collider with surf_shift. Set right vector of placed object with rightVec. Return object-placed locations in outLocations and the rotation for placed objects. rightVec will be modified to align collider’s orientation.

ViveSR_StaticColliderPool

Summary
The collider pool and manager of the ColliderInfo in the reconstruction data.

Public static functions	
ProcessDataAndGenColliderInfo(GameObject go)	bool Import the reconstruction collider from the saved file and generate ColliderInfo.

Public functions	
OrganizeHierarchy()	void Organize and category the hierarchy of the ColliderInfo.
GetClosestColliderWithProps(Vector3 testPos, uint[] props)	ViveSR_StaticColliderInfo Return the closest ColliderInfo that matches the corresponding bit of the shape type and the orientation.
GetFurthestColliderWithProps(Vector3 testPos, uint[] props)	ViveSR_StaticColliderInfo Return the furthest ColliderInfo that matches the corresponding bit of the shape type and the orientation.
GetLargestCollider(uint[] props)	ViveSR_StaticColliderInfo Return the largest ColliderInfo that matches the corresponding bit of the shape type and the orientation.
GetColliderByHeightRange(ColliderShapeType shapeType, float lowestHeight, float highestHeight)	ViveSR_StaticColliderInfo Return all ColliderInfo in the range of lowestHight to highestHeight that match the corresponding bit of the shape type.
GetAllColliderHasProps(uint[] props)	ViveSR_StaticColliderInfo[] Return all ColliderInfo that match the corresponding bit of the shape type and the orientation.
GetColliderWithPropsAndCondition(uint[] props, ColliderCondition condition, Vector3 testPos = new Vector3())	ViveSR_StaticColliderInfo[] Return all ColliderInfo that match the corresponding bit of the shape type and the orientation and the condition.
ShowAllColliderWithPropsAndCondition(uint[] props, ColliderCondition condition = ColliderCondition.NONE, Vector3 testPos = new Vector3())	void Render all ColliderInfo that match the corresponding bit of the shape type and the orientation and the condition and is active now.
DrawAllExtractedPlacerLocations(ViveSR_StaticColliderInfo[] infoArray)	void Render object-placed locations on all ColliderInfo in infoArray.
HideAllColliderRenderers()	void Hide all colliders.
ClearPlacerList()	void Clear all placed objects.

ViveSR_Enums

Enumerations used by ViveSR plugin.

Summary

Public types	

ModuleStatus
{
    SR_DISABLE = 0,
    SR_ENABLE = 1,
    SR_UPDATE = 2,
    SR_NOT_UPDATE = 3
}
	enum

ModuleType
{
ENGINE_SEETHROUGH = 1,
DEVICE_VIVE_HMD_DUALCAM = 2,
ENGINE_UNDISTORTION = 3,
ENGINE_DEPTH = 4,
ENGINE_RIGID_RECONSTRUCTION = 5,
ENGINE_CHAPERONE = 6,
};
	enum

WorkLinkMethod
{
    NONE = -1,
    PASSIVE = 0,
    ACTIVE = 1,
}
	enum

FrameworkStatus
{
    WORKING,
    STOP,
    ERROR
}
	enum
	Framework Status.

DualCameraMode
{
    REAL,
    MIX
}
	enum
	Dual camera mode.

DualCameraIndex
{
    LEFT,
    RIGHT
}
	enum

CalibrationType
{
 RELATIVE,
 ABSOLUTE
}
	enum
	Calibration type.

CalibrationAxis
{
 X, Y, Z
}
	enum
	Calibration axis.

DualCameraDisplayMode
{
 VIRTUAL,
 REAL,
 MIX
}
	enum
	Dual camera display mode.

DualCameraStatus
{
 NOT_FOUND,
 IDLE,
 WORKING,
 ERROR
}
	enum
	Dual camera staus.

Error
{
 FAILED = -1,
 WORK = 0,
 INVAILD_INPUT = 1,
 FILE_NOT_FOUND = 2,
 DATA_NOT_FOUND = 13,
 INITIAL_FAILED = 1001,
 NOT_IMPLEMENTED = 1003,
 NULL_POINTER = 1004,
 OVER_MAX_LENGTH = 1005,
 FILE_INVALID = 1006,
 UNINSTALL_STEAM = 1007,
 MEMCPY_FAIL = 1008,
 NOT_MATCH = 1009,
 NODE_NOT_EXIST = 1010,
 UNKONW_MODULE = 1011,
 MODULE_FULL = 1012,
 UNKNOW_TYPE = 1013,
 INVALID_MODULE = 1014,
 INVALID_TYPE = 1015,
 MEMORY_NOT_ENOUGH = 1016,
 BUZY = 1017,
 NOT_SUPPORT = 1018,
}
	enum

LogLevel
{
_0 = 3
_1 = 4,
_2 = 5,
_3 = 6,
_MAX = 10
};
	enum
	_0: turn-off any log except error & warning
	_1: the lower level, the less log, default value
	_MAX: turn-on all log message

CameraParam
{
CX_L,
CX_R,
CY_L,
CY_R,
FOCAL_LENGTH_L,
FOCAL_LENGTH_R,
R_M0,
R_M1,
R_M2,
R_M3,
R_M4,
R_M5,
R_M6,
R_M7,
R_M8,
T_M0,
T_M1,
T_M2
}
	enum

SeeThroughParam { VR_INIT = 0, VR_INIT_TYPE, OUTPUT_DISTORTED_WIDTH, OUTPUT_DISTORTED_HEIGHT, OUTPUT_DISTORTED_CHANNEL, OUTPUT_UNDISTORTED_WIDTH, OUTPUT_UNDISTORTED_HEIGHT, OUTPUT_UNDISTORTED_CHANNEL, OUTPUT_FPS, OFFSET_HEAD_TO_CAMERA PLAY_AREA_RECT VIDEO_RES_NATIVE_PTR, VIDEO_RES_VIEW_NATIVE_PTR, IMAGE_NATIVE_TEXTURE_PTR_L, IMAGE_NATIVE_TEXTURE_PTR_R, CAMERA_BRIGHTNESS = 100, CAMERA_CONTRAST, CAMERA_HUE, CAMERA_SATURATION, CAMERA_SHARPNESS, CAMERA_GAMMA,	enum

CAMERA_COLOR_ENABLE, CAMERA_WHITE_BALANCE, CAMERA_BACKLIGHT_COMPENSATION, CAMERA_GAIN, CAMERA_PAN, CAMERA_TILT, CAMERA_ROLL, CAMERA_ZOOM, CAMERA_EXPOSURE, CAMERA_IRIS, CAMERA_FOCUS, UNDISTORTION_MODE = 200, UNDISTORTION_CX, UNDISTORTION_CY, UNDISTORTION_FOCAL_LENGTH, UNDISTORTION_FMAT_RM_L, UNDISTORTION_FMAT_RM_R, UNDISTORTION_INTRINSIC_L, UNDISTORTION_INTRINSIC_R, UNDISTORTION_R_RECTIFY_L, UNDISTORTION_R_RECTIFY_R, UNDISTORTION_COEFFS_L, UNDISTORTION_COEFFS_R, UNDISTORTION_P_NEWPROJ_L, UNDISTORTION_P_NEWPROJ_R, UNDISTORTION_MAP_SIZE, UNDISTORTION_MAP_L, UNDISTORTION_MAP_R, UNDISTORTION_CENTER, }	

SeeThroughDataMask
{
DISTORTED_FRAME_LEFT = 0
DISTORTED_FRAME_RIGHT = 1,
UNDISTORTED_FRAME_LEFT = 2
UNDISTORTED_FRAME_RIGHT = 3,
FRAME_SEQ = 4,
TIME_STP = 5,
POSE_LEFT = 6,
POSE_RIGHT = 7,
LEFT_AEINDEX = 8,
RIGHT_AEINDEX = 9,
}
	enum
	DISTORTED_FRAME_LEFT: sizeof(char) * 612 * 460 * 4
	DISTORTED_FRAME_RIGHT: sizeof(char) * 612 * 460 * 4
	UNDISTORTED_FRAME_LEFT: sizeof(char) * 1150 * 750 * 4
	UNDISTORTED_FRAME_RIGHT: sizeof(char) * 1150 * 750 * 4
	FRAME_SEQ: sizeof(unsigned int)
	TIME_STP: sizeof(unsigned int)
	POSE_LEFT: sizeof(float) * 16
	POSE_RIGHT: sizeof(float) * 16
	LEFT_AEINDEX: sizeof(int)
	RIGHT_AEINDEX: sizeof(int)

SeeThroughCallback
{
BASIC = 1001,
}
	enum

DepthParam { OUTPUT_WIDTH, OUTPUT_HEIGHT, OUTPUT_CHAANEL_0,	enum DENOISE_M: range : 1, 3, 5; (default: 3) CONFIDENCE_THRESHOLD: range : 0 ~ 5; (default: 0.05) DENOISE_G: range : 1 ~ 7; (default: 3)

OUTPUT_CHAANEL_1, TYPE, FOCULENS, BASELINE, COLLIDER_QUALITY, MESH_NEAR_DISTANCE, MESH_FAR_DISTANCE, DENOISE_M, CONFIDENCE_THRESHOLD, DENOISE_G, DEPTH_USING_CASE, KEEP_ONFLY_RESULT_AT_END, }	

DepthCase
{
DEFAULT,
CLOSE_RANGE,
};
	enum

DepthDataMask
{
LEFT_FRAME = 0,
DEPTH_MAP = 1,
FRAME_SEQ = 2,
TIME_STP = 3,
POSE = 4,
NUM_VERTICES = 5,
BYTEPERVERT = 6,
VERTICES = 7,
NUM_INDICES = 8,
INDICES = 9,
}
	enum
	LEFT_FRAME: sizeof(char) * 640 * 480 * 4
	DEPTH_MAP: sizeof(float) * 640 * 480 * 1
	FRAME_SEQ: sizeof(unsigned int)
	TIME_STP: sizeof(unsigned int)
	POSE: sizeof(float) * 16
	NUM_VERTICES: sizeof(unsigned int)
	BYTEPERVERT: sizeof(unsigned int)
	VERTICES: sizeof(float) * 640 * 480 * 3
	NUM_INDICES: sizeof(unsigned int)
	INDICES: sizeof(int) * 640 * 480 * 6

DepthCmd
{
EXTRACT_DEPTH_MESH = 0,
ENABLE_SELECT_MESH_DISTANCE_RANGE,
ENABLE_REFINEMENT,
ENABLE_EDGE_ENHANCE,
CHANGE_DEPTH_CASE,
ENABLE_ONFLY,
}
	enum

DepthCallback
{
BASIC = 1001,
}
	enum

ReconstructionParam
{
VOXEL_SIZE = 0,
COLOR_SIZE = 1,
DATA_SOURCE = 2,
DATASET_PATH = 3,
RGB_IMAGE_EXT = 4,
DATASET_FRAME = 5,
	enum
	SCENE_UNDERSTANDING_ENABLE allows to analyze the surroundings powered by machine learning

MAX_DEPTH = 6,
MIN_DEPTH = 7,
POINTCLOUD_POINTSIZE = 9,
EXPORT_ADAPTIVE_MODEL = 10,
ADAPTIVE_MAX_GRID = 11,
ADAPTIVE_MIN_GRID = 12,
ADAPTIVE_ERROR_THRES = 13,
SECTOR_SIZE = 15,
SECTOR_NUM_PER_SIDE = 16,
ENABLE_FRUSTUM_CULLING = 20,
CONFIG_FILEPATH = 21,
CONFIG_QUALITY,
CONFIG_EXPORT_COLLIDER,
CONFIG_EXPORT_TEXTURE,
DATA_CURRENT_POS = 31,
LITE_POINT_CLOUD_MODE,
FULL_POINT_CLOUD_MODE,
LIVE_ADAPTIVE_MODE,
MESH_REFRESH_INTERVAL = 37,
ENABLE_SECTOR_GROUPER = 38,
SCENE_UNDERSTANDING_ENABLE = 40,
SCENE_UNDERSTANDING_MACHINE_VISION = 41,
SCENE_UNDERSTANDING_CONFIG = 42,
SCENE_UNDERSTANDING_REFINEMENT = 43,
VERTEX_BUFFER_NATIVE_PTR = 99,
INDEX_BUFFER_NATIVE_PTR = 100,
}	

ReconstructionDataMask
{
FRAME_SEQ = 0,
POSEMTX44 = 1,
NUM_VERTICES = 2,
BYTEPERVERT = 3,
VERTICES = 4,
NUM_INDICES = 5,
INDICES = 6,
CLDTYPE = 7,
COLLIDERNUM = 8,
CLD_NUM_VERTS = 9,
CLD_NUMIDX = 10,
CLD_VERTICES = 11,
CLD_INDICES = 12,
SECTOR_NUM = 13,
SECTOR_ID_LIST = 14,
SECTOR_VERT_NUM = 15,
SECTOR_IDX_NUM = 16,
}
	enum

SceneUnderstandingObjectType
{
FLOOR = 0,
WALL = 1,
	enum

CEILING = 2,
CHAIR = 3,
TABLE = 4,
BED = 5
}	

ReconstructionCmd
{
START = 0,
STOP = 1,
SHOW_INFO = 2,
EXTRACT_POINT_CLOUD = 3,
EXTRACT_VERTEX_NORMAL = 4,
EXPORT_MODEL_RIGHT_HAND = 5,
EXPORT_MODEL_FOR_UNITY = 6,
EXPORT_SCENE_UNDERSTANDING_RIGHT_HAND = 7,
EXPORT_SCENE_UNDERSTANDING_FOR_UNITY = 8,
}
	enum

ReconstructionCallback
{
EXPORT_PROGRESS = 1,
SCENE_UNDERSTANDING_PROGRESS = 2,
BASIC = 1001,
}
	enum

ReconstructionDataSource
{
HMD = 0,
DATASET = 1
}
	enum
	Reconstruction data source

ReconstructionQuality
{
LOW = 2,
MID = 3,
HIGH = 4,
}
	enum

ReconstructionLiveMeshExtractMode
{
VERTEX_WITHOUT_NORMAL = 0,
VERTEX_WITH_NORMAL = 1,
FACE_NORMAL = 2,
}
	enum

ReconstructionLiveColliderType
{
CONVEX_SHAPE = 0,
BOUNDING_BOX_SHAPE = 1,
}
	enum

ReconstructionExportStage
{
STAGE_EXTRACTING_MODEL = 0x0017,
STAGE_COMPACTING_TEXTURE = 0x0018,
STAGE_SAVING_MODEL_FILE = 0x0019,
	enum

STAGE_EXTRACTING_COLLIDER = 0x001A,
SCENE_UNDERSTANDING_PASS_1 = 0x0030,
SCENE_UNDERSTANDING_PASS_2 = 0x0031,
}	

ReconstructionDisplayMode
{
    FULL_SCENE = 0,
    FIELD_OF_VIEW = 1,
    ADAPTIVE_MESH = 2,
}
	enum

ReconstructionQuality
{
    LOW = 2,
    MID = 3,
    HIGH = 4,
}
	enum
	Quality of the reconstructed model.

ReconstructionCmd
{
    START = 0,
    STOP = 1,
    SHOW_INFO = 2,
    EXTRACT_LIVE_MESH = 3,
    UNUSED_IN_UNITY = 4,
    EXPORT_MODEL = 5,
}
	enum
	Commands of Reconstruction.

ReconstructionParamID
{
    PARAM_DATA_SOURCE = 2,
    PARAM_DATASET_PATH = 3,
    PARAM_POINT_CLOUD_LOD = 9,
}
	enum
	Parameter index of reconstruction.