RoomPlan for Unity Kit Documentation

Silver Tau

Plugins

2023

Content

Overview **3**
RoomPlan Unity Kit **4**
Architecture 5
Actions 5
Properties 6
Functions 6
Session Camera **9**
Properties 9
Functions 9
Captured Room Snapshot **10**
Properties 10
Functions 11
Room Builder **11**
Properties 12
Functions 13
Captured Room **14**
Inspecting room details: 14
Captured Structure **15**
Inspecting structure details: 15
RoomPlan Object **16**
Properties 17
Functions 17
RoomPlan Unity Kit Settings **18**
Properties 18
Flashlight Manager **20**
Properties 20
Functions 20
Custom Session Instruction **21**
Properties 21
XR **22**
AR Foundation 22
Functions 23
Utilities **24**
Quick Look 24
CSGeometry 24
Screenshot 25
Screen Recorder 26
Share 26
OBJ 27
RPU Inspector 28

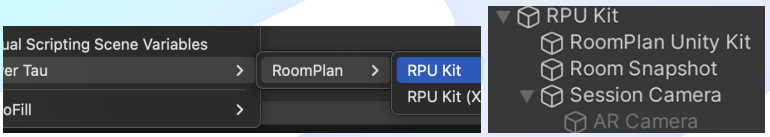
Overview

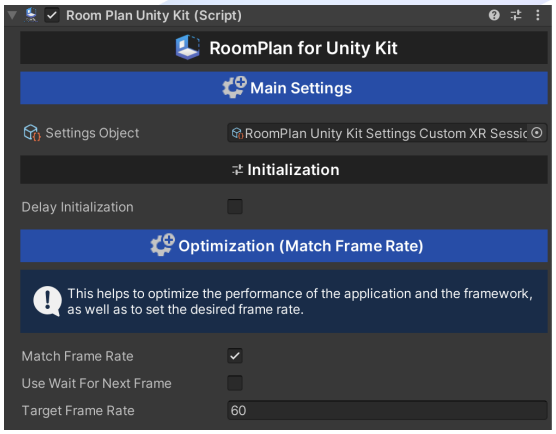
Feature	Description
RoomPlan Unity Kit	The main component that creates the connection between the Unity engine and the RoomPlan framework.
Session Camera	A component that communicates with the framework and enables the camera to navigate in real space.
Captured Room Snapshot	A component that interacts with the framework and builds scanned 3D objects in real time.
Room Builder	A component that allows you to build a room from the specified parameters.
Captured Room	A structure that provides the key details of a scanned room.
Captured Structure	An object that holds the results of the merger of multiple capture sessions.
RoomPlan Object	A component that has the main characteristics of the scanned object.
RoomPlan Unity Kit Settings	Scriptable object that has characteristics for configuring RoomPlan for Unity Engine.
Flashlight Manager	Flashlight Manager is a manager class that allows you to control the flashlight of a device during scanning or resting.
Custom Session Instruction	Custom Session Instruction is a class that allows you to manage hints from the RoomPlan API.

Adding a RPU (RoomPlan Unity Kit)

To add a basic module to a scene, you need to do the following:

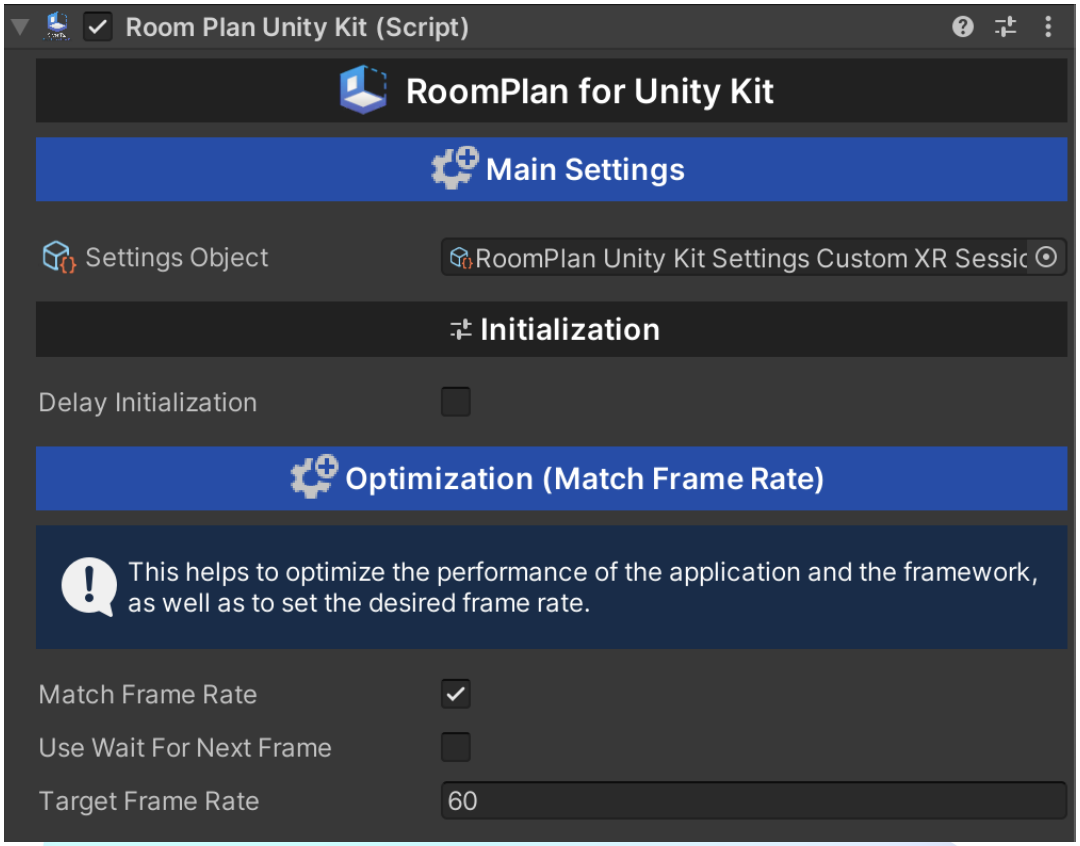
1. In the scene hierarchy, add a component from the quick menu (or manually).
2. In the Component inspector, add the basic elements.





RoomPlan Unity Kit

The main component that creates the connection between the Unity engine and the RoomPlan framework.



The RoomPlan Unity Kit framework uses the RoomPlan API, device sensors, trained ML models, and RealityKit rendering capabilities to render the physical environment of an indoor room in the Unity Engine. For example, the framework checks the device's camera image and LiDAR data to identify walls, windows, openings, and doors. The framework also receives functionality from RoomPlan, which recognizes room features, furniture, and appliances, such as a fireplace, bed, or refrigerator, and provides this information to the app.

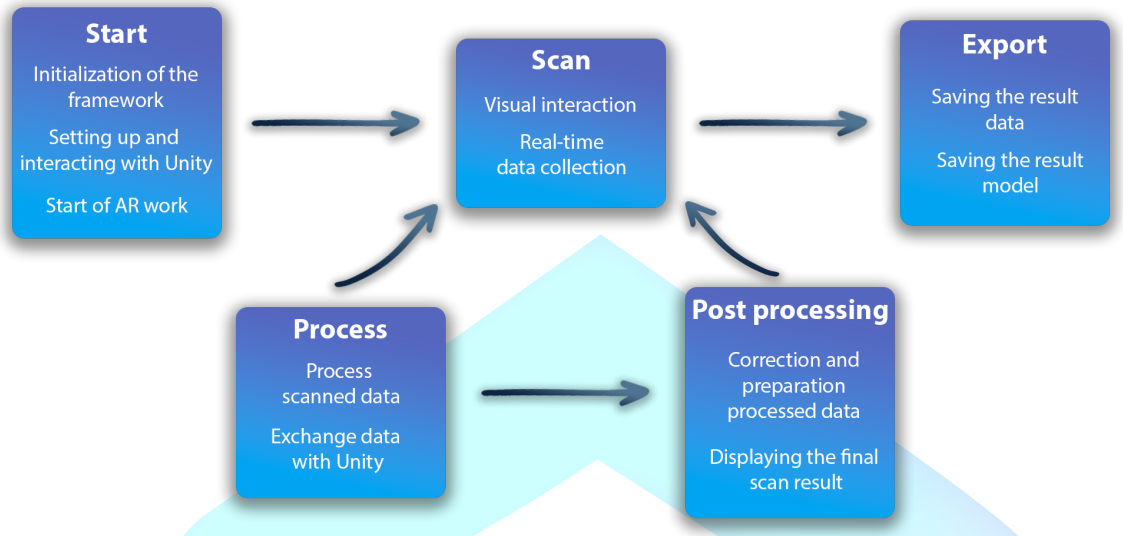
To start the capture, the app presents a view (RoomCaptureView) through which the user can see their room in augmented reality. The view displays virtual cues as the user moves around the room:

- Real-time graphical overlays are displayed on top of physical structures in the room to show the progress of the scan.
- If the framework requires a certain type of device movement or perspective to complete the scan, instructions are displayed in the interface to explain how to position the device.

When the program determines that the current scan is complete, a smaller version of the scanned room is displayed in the window for user approval.

Architecture

The following diagram shows framework architecture.



This diagram illustrates how the framework you use provides you or your development clients with an integrated and consistent development experience for building applications and experiences.

Next, we will describe the functionality and its purpose.

Actions

Use this to create some dynamic functionality in your scripts. Unity Actions allow you to dynamically call multiple functions.

Property	Description
didStart	An action called when the framework is initialized. With this action, you can track the status of the framework initialization.
didEnd	An action that is called when a framework is disposed of. With this action, you can track the status of the disposed of framework and you can given it additional actions.
captureSessionDidStart	Called when the scanning session starts.
captureSessionDidEnd	Called when the scan session ends.
captureSessionInstruction	Provides scan session instructions for the user. **Determining a coaching recommendation:** **normal** - An instruction that indicates scanning proceeds normally and the user needs no coaching. **moveCloseToWall** - An instruction that requests the user move closer to the wall. **moveAwayFromWall** - An instruction that requests the user move further from the wall. **turnOnLight** - An instruction that requests the user increase the amount of light in the room.

	slowDown - An instruction that requests that the user move slower. **lowTexture** - An instruction that indicates the framework doesn't detect distinguishable room features.
roomSnapshot<**String**>	Called when a session sends a snapshot to the Unity Engine. <**String**> - A data snapshot that is sent in JSON format.

captureStatus<**CaptureStatus**>	Called when the session status is changed.
	<**CaptureStatus**> is an enum of the session status.
	
	`enum CaptureStatus`
	`{`
	`None = 0,`
	`Processing = 1,`
	`Paused = 2,`
	`Scanning = 4,`
	`}`

Properties

Property	Description
delayInitialization	If the parameter is enabled, initialization will occur only when you initialize the framework manually.
matchFrameRate	This helps to optimize the performance of the application and the framework, as well as to set the desired frame rate.
CurrentCaptureStatus	A parameter that informs about the current status of the session.
CurrentSessionCamera	An option that lets you set or get the current session camera.
CurrentRoomPlanUnityKitSettings	An option that allows you to set or get the current settings of the framework.
ApplicationTargetFrameRate	Value that sets the target frame rate in real time. *If you use useWaitForNextFrame.*
RPUSupport	Values to check for RoomPlan support.
RPUActive	An option that allows you to set or get the current status of the session, whether it is active.
RPUCaptureSessionActive	An option that allows you to set or get the current scanning status of the session, whether it is active.

Functions

Property	Description
Initialize	A method that initializes a framework.
Dispose	A method that disposes of a framework.

UpdateRPUKitSettings	A method that updates the main RoomPlan settings in real time. By updating **RoomPlan Unity Kit Settings**, all settings will be applied
StartCaptureSession	A method that starts a RoomPlan scanning session.
StopCaptureSession	A method that stops a RoomPlan scanning session.
RunCaptureSession	Starts a room capture session with the specified configuration after the session was stopped with a pause (StopCaptureSessionWithPause).
StopCaptureSessionWithPause	Stops the room capture session with the specified AR session parameter. Used to be able to stop the session and add a new scanning room. <param name="**pauseARSession**">AR session status.</param>
ResumeCaptureSessionScanning	A method that resumes a RoomPlan scanning session. After using this method, the process of transferring snapshot data and creating objects in augmented reality continues.
PauseCaptureSessionScanning	A method that paused a RoomPlan scanning session. After using this method, the process of transferring snapshot data and creating objects in augmented reality is suspended.
SaveRoomPlanExperience	A method that allows you to preserve the scanning experience. To save the data, the last snapshot of the experience in augmented reality is taken. <param name="**scanName**">Scan name.</param> <param name="**directoryName**">The name of the directory. The main directory for saving files.</param>
TrySaveRoomPlanExperience	A method that allows you to save the scanning experience and returns a Boolean value for the success of the operation. To save the data, the last snapshot of the experience in augmented reality is taken. <param name="**scanName**">Scan name.</param> <param name="**directoryName**">The name of the directory. The main directory for saving files.</param>
Screenshot	A method that allows you to take a screenshot of an experience in augmented reality. **Example of use:** `RoomPlanUnityKit.Screenshot();`
ScreenshotShare	A method that allows you to share a screenshot. The last screenshot taken will be used for the share method. **Example of use:** `RoomPlanUnityKit.ScreenshotShare();`
StartScreenRecorder	A method that allows you to start recording a video screen. The recording process takes place using RPScreenRecorder specially customized for the RoomPlan for Unity Kit. **Example of use:**

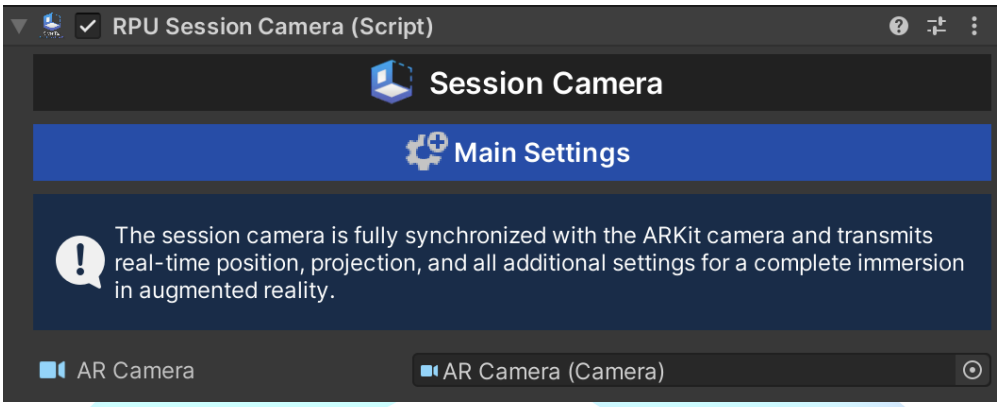
	`RoomPlanUnityKit.StartScreenRecorder();`
StopScreenRecorder	A method that allows you to stop recording a video screen. **Example of use:** `RoomPlanUnityKit.StopScreenRecorder();`
SetFlashlightStatus(bool status)	A method that allows you to set the state of the flashlight. It can be used even when running an AR Session. <param name="status">The condition of the flashlight.</param> **Example of use:** `RoomPlanUnityKit.SetFlashlightStatus(bool status);`
CombineRoomPlanUnityKitXRSession	This is a function for ARSession (subsystem) that combines work sessions in runtime.

Combine_RPU_XR_Session	This is an extending function for ARSession (subsystem) that combines work sessions. To use it, you need to initialize (enable) the ARSession and execute the extension.
	
	Example of use:
	
	`var aRSession = GetComponent<ARSession>();`
	`aRSession.enabled = true;`
	`aRSession.Combine_RPU_XR_Session();`

Session Camera

Device cameras and their configuration are an essential function of most AR apps. A component that communicates with the framework and enables the camera to navigate in real space.

The Session Camera is fully synchronized with the ARKit camera and transmits real-time positions, projection, and all additional settings for a complete immersion in augmented reality.



Properties

Property	Description
ARCamera	The current session camera.
CurrentARCamera	Customize the current session camera.
angle	An additional option that allows you to adjust the camera angle.
scale	An additional option that allows you to adjust the camera scale.
shearingX	An additional option that allows you to adjust the camera shearing along the X-axis.
shearingY	An additional option that allows you to adjust the camera shearing along the Y-axis.
translationX	An additional option that allows you to adjust the camera translation along the X-axis.
translationY	An additional option that allows you to adjust the camera translation along the Y-axis.

Functions

Property	Description
SetNewCamera	An additional way to install a new camera. This method helps you change the main camera of a session to another one if you need it.
UpdateCameraTRS	A method that updates the position, rotation, and scale of the camera in real time. The main purpose of the method is to stabilize and parallelize the camera's broadcast data frame by frame.

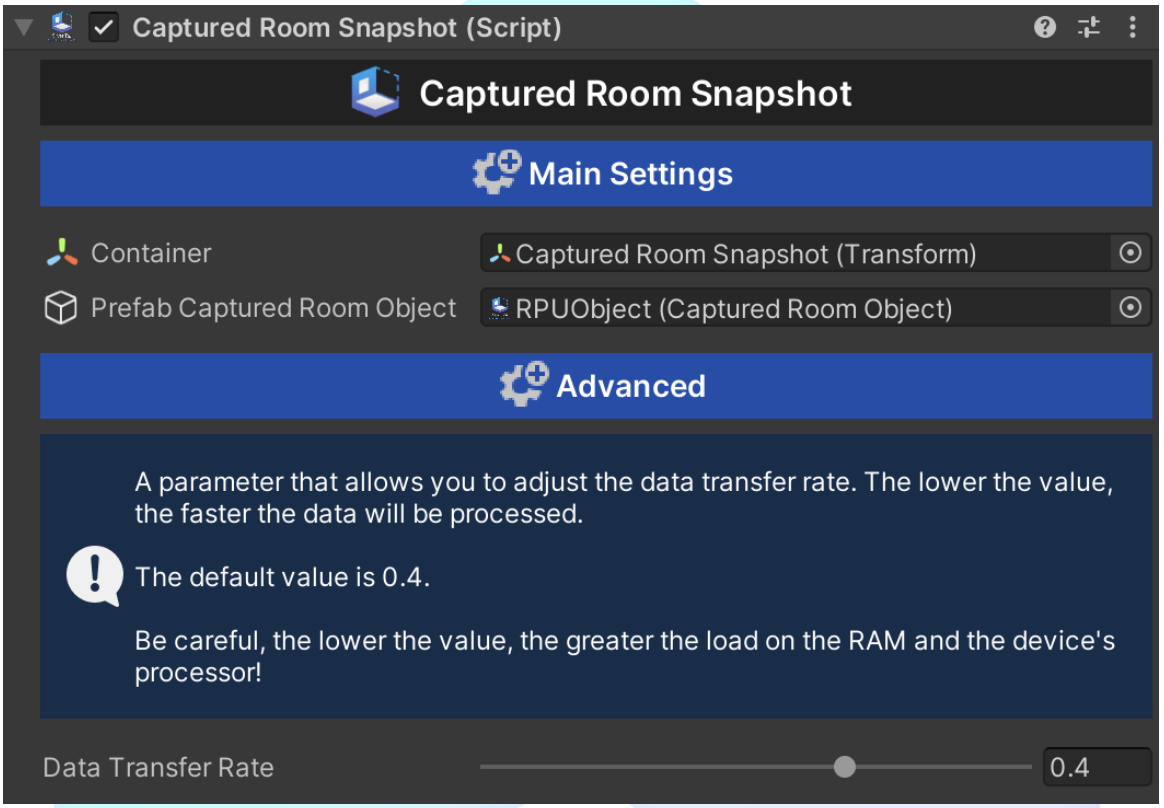
Captured Room Snapshot

This class is used only when creating an augmented reality session. A component that interacts with the framework and builds scanned 3D objects in real time. With the help of additional methods, the process of creating objects that are rendered in Unity is implemented and allows you to take separate data for each object during the experience.

This structure represents the processed-post-processed result of the room-scanning process.

Your app receives an instance of this structure through:

- With the room details, an app can provide custom features, such as rendering the room and enabling the user to modify the position of its objects.



Properties

Property	Description
container	Place the parent object where the scanned objects will be added.
prefabCapturedRoomObject	Prefab Captured Room Object. This object is used to create scanning objects in augmented reality.
ObjectContainer	Place, the parent object to which the scanned components of the object type are added.
SurfaceContainer	Place, the parent object to which the scanned components of the surface type are added.
GetCapturedRoom	Get a current captured room.
GetRoomObjects	Get a list of components of type object. This is a list of objects that were

	scanned during the session.
GetRoomSurfaces	Get a list of components of type surface. This is a list of surfaces that were scanned during the session.
GetContentRooms	Place, the parent object to which the scanned components of the object type are added.
createFloor	Enable or disable the variable to automatically create a floor. The floor is created by finding a point cloud and triangulating it.
typeFloorConstructor	The floor constructor type allows you to choose which main target objects will be used to build the floor.
resetObjectsIfNewRoomIsAdded	Reset scanned objects (visual display of objects) if a new room is added.
RoomBounds	A variable that returns the bounds of the room. Only during the experience, a session in augmented reality.
GetRaycastPlane	A variable that returns the raycast of the room. Only during the experience, a session in augmented reality.
DataTransferRate	A parameter that allows you to adjust the data transfer rate. The lower the value, the faster the data will be processed. The default value is 0.4. Be careful, the lower the value, the greater the load on the RAM and the device's processor!
TargetRoomPlanUnityKitSettings	Adds a target RoomPlan for Unity Ki settings to automatically synchronize settings.

Functions

Property	Description
RoomSnapshot	A method that creates objects from a real-time snapshot. Only during the experience, a session in augmented reality.
EditorRoomSnapshot	A method that creates objects from a snapshot in the Unity Engine editor.
AddedNewRoomToSnapshot	This is a function that performs the action of adding a new room during an AR object scanning session. Only for iOS 17+.
Dispose	A method that disposes of a Captured Room Snapshot.

Room Builder

This class can be used at any time, except during an augmented reality session (experience). A component that allows you to build a room from the specified parameters. With the help of additional methods, it implements the process of creating objects that are visualized in Unity and allows you to take separate data for each object during the experience.

Your app generates a detailed captured room object on the raw data. Your app can then inspect or modify this data before exporting the scanned room to a USDZ file or JSON snapshot.

Properties

Property	Description
container	Place the parent object where the scanned objects will be added.
prefabRoomPlanObject	Prefab RoomPlan Object. This object is used to create scanning objects.
GetContentRooms	Place, the parent object to which the scanned components of the object type are added.
ObjectContainer	Place, the parent object to which the scanned components of the object type are added.
SurfaceContainer	Place, the parent object to which the scanned components of the surface type are added.
GetCapturedRoom	Get a current captured room.
GetRoomObjects	Get a list of components of type object. This is a list of objects that were loaded from the input snapshot.
GetRoomSurfaces	Get a list of components of type surface. This is a list of surfaces that were loaded from the input snapshot.
createFloor	Enable or disable the variable to automatically create a floor. The floor is created by finding a point cloud and triangulating it.
typeFloorConstructor	The floor constructor type allows you to choose which main target objects will be used to build the floor.
concavity	Concavity is a value used to restrict the concave angles. It can go from -1 (not concave at all) ǀ 0 (no concavity - convex is used) ǀ to 1 (extreme concavity). Avoid concavity == 1 if you don't want 0° angles. In AR mode, the parameter will be 0.
TargetRoomPlanUnityKitSettings	Adds a target RoomPlan for Unity Ki settings to automatically synchronize settings.

Functions

Property	Description
CreateRoomFromSnapshot	A method that creates objects from a real-time snapshot. The creation process will take place from the uploaded input snapshot.
GetCapturedStructureFromSnapshot	A function that performs the action of converting a snapshot to a CapturedStructure class. <param name="**snapshot**">Input snapshot.</param> <param name="**error**">Called when an execution error occurs.</param>
CreateRoomFromSnapshot	A function that allows you to perform the action of creating a room with an input snapshot file. <param name="**snapshot**">Input snapshot.</param> <param name="**successfully**">Called on successful completion.</param> <param name="**error**">Called when an execution error occurs.</param>
CreateRoomFromCapturedStructure	A function that allows you to perform the action of creating a room with an input captured structure. <param name="**snapshot**">Input snapshot.</param> <param name="**successfully**">Called on successful completion.</param> <param name="**error**">Called when an execution error occurs.</param>
CreateRoomFromCapturedRoom	A function that allows you to perform the action of creating a room with an input captured room. <param name="**snapshot**">Input snapshot.</param> <param name="**successfully**">Called on successful completion.</param> <param name="**error**">Called when an execution error occurs.</param>
CreateRoomFromCapturedRooms	A function that allows you to perform the action of creating a room with an input captured rooms. <param name="**snapshot**">Input snapshot.</param> <param name="**successfully**">Called on successful completion.</param> <param name="**error**">Called when an execution error occurs.</param>
Dispose	A method that disposes of a Captured Room Snapshot.

Captured Room

This structure represents the post-processed result of the room-scanning process.
Your app receives an instance of this structure through:

- A room builder object (RoomBuilder) for an app that provides its own room-scanning view, or processes the saved results of a prior scan.
- With the room details, an app can provide custom features, such as rendering the room and enabling the user to modify the position of its objects.

Inspecting room details:

var identifier: UUID
A unique alphanumeric value that the framework assigns to the room.

var story: Int
The story (floor number) on which the captured room resides within a larger structure.

var floors: [CapturedRoom.Surface]
An array of floors that the framework identifies during a scan.

struct Surface
A 2D area in a room that the framework identifies as a surface.

var doors: [CapturedRoom.Surface]
An array of doors that the framework identifies during a scan.

var objects: [CapturedRoom.Object]
An array of objects that the framework identifies during a scan.

struct Object
A 3D area in a room that the framework identifies as an object.

var openings: [CapturedRoom.Surface]
An array of openings that the framework identifies during a scan.

var walls: [CapturedRoom.Surface]
An array of walls that the framework identifies during a scan.

var windows: [CapturedRoom.Surface]
An array of windows that the framework identifies during a scan.

var sections: [CapturedRoom.Section]
One or more room types that the framework observes in the room.

struct Section
An object that identifies a particular area in a captured room in relation to common types of room areas in a building.

enum Confidence
Levels of certainty in the classification of a particular detail in a scan.

var version: Int
A version number for the captured room.

Captured Structure

An object that holds the results of the merger of multiple capture sessions.

This structure combines the data from multiple CapturedRoom instances that a user scans in the same physical vicinity. The StructureBuilder class's function merges the captured rooms, which returns an object of this type.

Inspecting structure details:

var identifier: UUID
A unique alphanumeric value that the framework assigns the structure.

var rooms: [CapturedRoom]
An array of all the floors in the structure.

var floors: [CapturedRoom.Surface]
An array of all the floors in the structure.

struct Surface
The type a captured structure assigns to surfaces.

var doors: [CapturedRoom.Surface]
An array of all the doors in the structure.

var objects: [CapturedRoom.Object]
An array of all the objects in the structure.

struct Object
The type a captured structure assigns to objects.

var openings: [CapturedRoom.Surface]
An array of all the openings in the structure.

var walls: [CapturedRoom.Surface]
An array of all the walls in the structure.

var windows: [CapturedRoom.Surface]
An array of all the windows in the structure.

var sections: [CapturedRoom.Section]
One or more room types that the framework identifies in the structure.

struct Section
The type a captured structure assigns to distinct room areas.

var version: Int
A version number for the captured structure.

RoomPlan Object

A component that has the main characteristics of the scanned object.

This abstract class is used to serve and create your own objects. A component that has the main characteristics of the scanned object.

The structure of the object:

RoomPlanObject:

```
-- id;
-- confidence:
    - unknown;
    - low;
    - medium;
    - high;
-- type:
    - Unknown;
    - Object;
    - Surface;
-- category:
    - unknown;
    - bathtub;
    - bed;
    - chair;
    - dishwasher;
    - fireplace;
    - oven;
    - refrigerator;
    - sink;
    - sofa;
    - stairs;
    - storage;
    - stove;
    - table;
    - television;
    - toilet;
    - washerDryer;
    - window;
    - wall;
    - opening;
    - door;
    - doorOpen;
    - doorClose;
    - floor;
    - ceiling;
-- size;
-- InitObject(Vector3, Quaternion, Vector3);
-- UpdateObject(Vector3, Quaternion, Vector3);
-- RemoveObject();
```

Properties

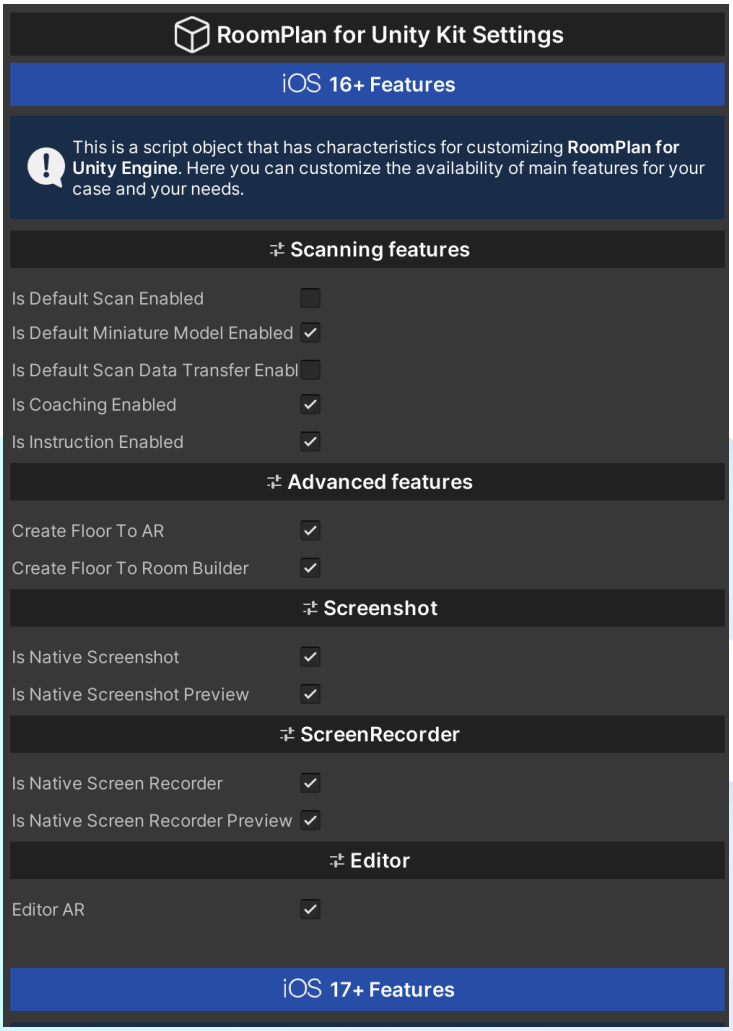
Property	Description
id	A unique object identifier.
confidence	The state of the scanned object.
type	Type of the scanned object.
category	The category of the scanned object.
size	The actual size of the scanned object.

Functions

Property	Description
InitObject	An abstract method that is executed when an object is initialized and passes certain data.
UpdateObject	An abstract method that is executed when an object is updated and passes certain data.
RemoveObject	An abstract method that is executed when an object is removed and passes certain data.

RoomPlan Unity Kit Settings

This is a script object that has characteristics for customizing RoomPlan for Unity Engine. Here you can customize the availability of main functions for your case and needs.



Properties

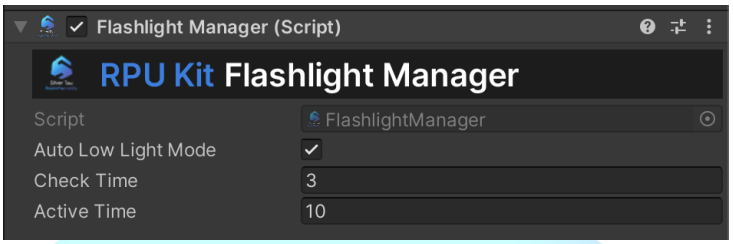
Property	Description
isDefaultScanEnabled	Enable or disable the standard RoomPlan API solution.
isDefaultMiniatureModelEnabled	Turn on or off the 3D mini-model for the default scan view.
isDefaultScanDataTransferEnabled	Enables/disables the transmission of scan data in the default mode.
isCoachingEnabled	Turn the coaching on or off for a view session.
isInstructionEnabled	Turn the instructions on or off for a view session.
createFloorToAR	Enable or disable the variable to automatically create a floor during an AR session.
createFloorToRoomBuilder	Enable or disable the variable to automatically create a floor when RoomBulder is running (outside of an AR session).

isNativeScreenshot	Turn the screenshot feature on or off.
isNativeScreenshotPreview	Turn the screenshot preview feature on or off.
isNativeScreenRecorder	Turn the screen recorder feature on or off.
isNativeScreenRecorderPreview	Turn the screen recorder preview feature on or off.
editorAR	Turn on or off the ability to edit a scan in the Unity Engine Editor.
useMeshBooleanOperationsForRoomBuilder	Enable or disable a variable to automatically apply Boolean operations (cut, merge, intersection) in RoomBuilder (outside of an AR session).
meshBooleanOperationsForRoomBuilder	Choose which category of objects you want to cut. (Mesh Boolean Operations) MeshBooleanOperations: - None; - Windows; - Openings; - Doors; - Open Doors; - Close Doors;
isCustomXREnabled	Enable or disable the ability to use custom XR Session for the RoomPlan API.

Flashlight Manager

Flashlight Manager is a manager class that allows you to control the flashlight of a device during scanning or resting. It performs an auxiliary function, additional lighting, in the real world for better scanning. The manager also has the function of manually turning on/off the device's flashlight.

The main purpose of this manager is to improve scanning in low-light conditions in the real world. It can be used during nighttime scanning.



Properties

Property	Description
autoLowLightMode	This option allows you to automatically turn on/off the phone's flashlight when the light is not sufficient for scanning.
checkTime	A parameter that indicates the time of the real-world light check. After how many seconds the device's flashlight will turn on.
activeTime	A parameter that indicates how long the flashlight will be on before the next check.

Functions

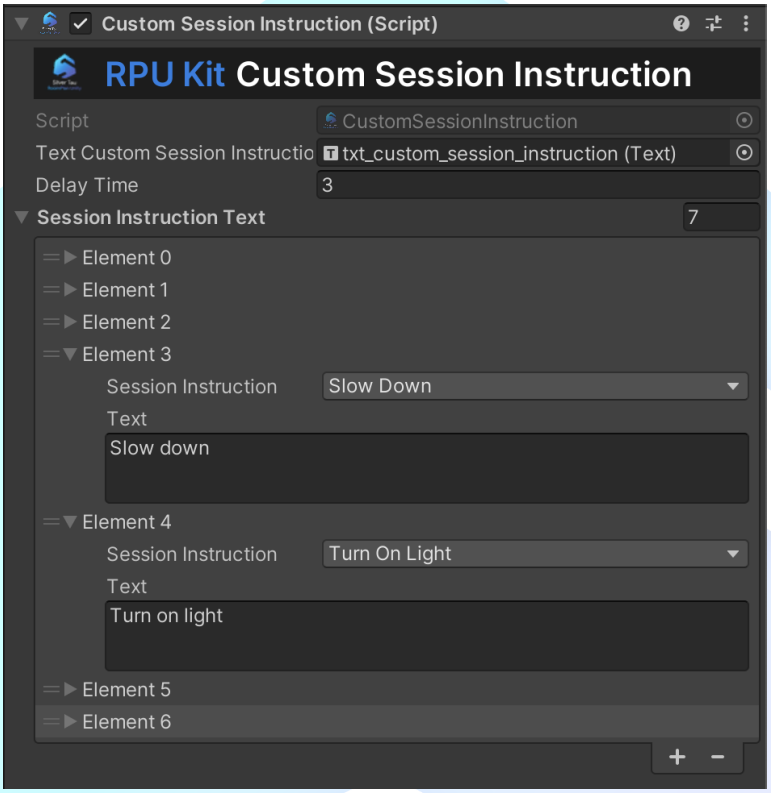
Property	Description
SetFlashlightStatus(bool status)	A method that allows you to set the state of the flashlight. It can be used even when running an AR Session. <param name="status">The condition of the flashlight.</param> **Example of use:** `RoomPlanUnityKit.SetFlashlightStatus(bool status);`
SetFlashlightStatus(bool status)	A method that allows you to automatically set the state of the flashlight (on/off). **Example of use:** `RoomPlanUnityKit.FlashlightOnOff();`

Custom Session Instruction

Custom Session Instruction is a class that allows you to manage hints from the RoomPlan API. It can be customized as you like. To get it working, don't forget to configure the RoomPlan Unity Kit settings file.

Determining a coaching recommendation:

- **normal** - An instruction that indicates scanning proceeds normally and the user needs no coaching.
- **moveCloseToWall** - An instruction that requests the user move closer to the wall.
- **moveAwayFromWall** - An instruction that requests the user move further from the wall.
- **turnOnLight** - An instruction that requests the user increase the amount of light in the room.
- **slowDown** - An instruction that requests that the user move slower.
- **lowTexture** - An instruction that indicates the framework doesn't detect distinguishable room features.



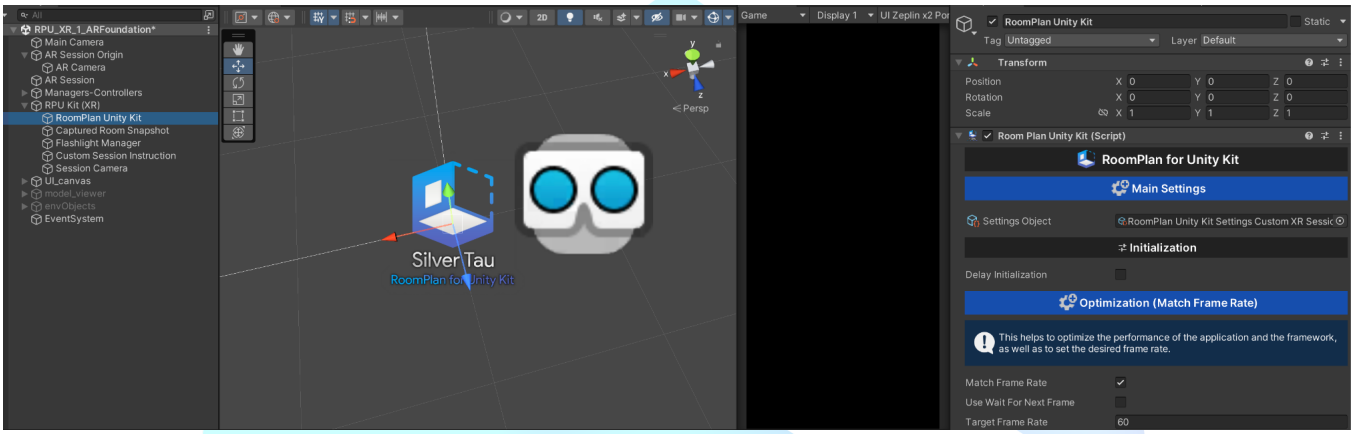
Properties

Property	Description
textCustomSessionInstruction	This is the UI text element to which the session instruction will be passed.
delayTime	This is a parameter that indicates how long the session instruction notification will be displayed.
sessionInstructionText	This is the email parameter where you can specify custom text for each category of instructions.

XR

The RoomPlan for Unity Kit (XR) uses the RoomPlan API, device sensors, trained ML models, XR Session (ARFoundation), and RealityKit rendering capabilities to visualize the physical environment of a room in the Unity Engine. The XR solution allows you to use XR Session, combining it with the ability to scan objects using the RoomPlan API.

When using a custom XR session, the RoomPlan plugin for the Unity Kit (XR) acts as a real-time object recognition module. The room capture process can be combined with the XR session at any time during the program execution, but the XR session must be initialized beforehand.



AR Foundation

The RoomPlan plugin for Unity Kit (XR) version 1.3.x+ is fully compatible with the AR Foundation. You can use all the available features and work with all the possibilities during your work.

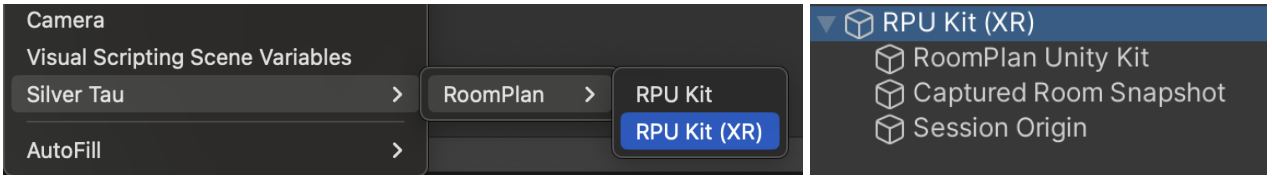
Instructions for setting up the work:

1. Create a new Unity project.
2. Add the ARFoundation and ARKit packages.
3. Configure ARFoundation and connect the ARKit provider (instructions).
4. Import the RoomPlan for Unity Kit package.
5. Open the demo scene "RPU_XR_1_ARFoundation".
6. The project is ready to use.

About the demo stage "RPU_XR_1_ARFoundation".:

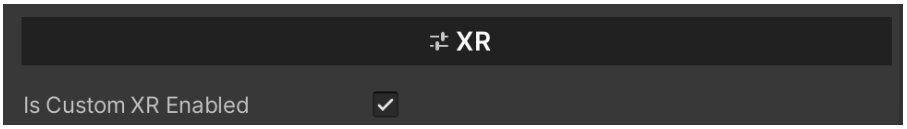
This demo scene provides an introduction to the XR solution for the RoomPlan API. Here we will describe the main points you need to set up a scene from scratch or what will help you in your work.

RPU Kit (XR) - is an object that contains the main components for the module to work.
You can add an object using the quick menu:



After you add an object, you need to make basic settings:
- add the SO configuration file;
- add the object prefab to the Captured Room Snapshot;
- add the target camera of the AR Session to the Session Camera;

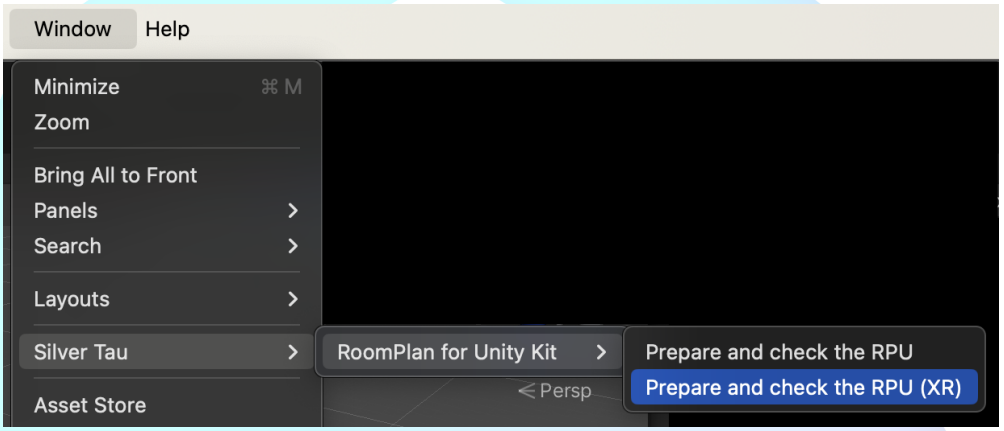
Note: in the settings file, you need to activate the "Is Custom XR Enabled" option for the XR session to work.



AR Session & AR Session Origin - are XR objects, which are completely identical to ARFoundation objects.
ARViewerXRSettings - is a script created to demonstrate XR session management. It performs the functions of enabling/disabling ARSession.
RPU_XRSessionService - is a script created to manage XR sessions for RoomPlan for Unity Kit. It performs the functions of combining a session with RoomPlan for Unity Kit.

Note: If you are using the XR solution of RoomPlan for Unity Kit, you do not need to run the scan in manual mode (RoomPlanUnityKit.StartCaptureSession()). The scan will automatically start when the ARSession (ARFoundation) is enabled.

You can also use the shortcut menu options to automatically check for the presence and readiness of all squeaks. These functions will automatically add the necessary objects to the scene.



Functions

Property	Description

Combine_And_Start_RPU_XR_Session	This is an extending function for ARSession (subsystem) that combines work sessions. To use it, you need to initialize (enable) the ARSession and execute the extension.
	
	Note: If you are using the XR solution of RoomPlan for Unity Kit, you do not need to run the scan in manual mode (RoomPlanUnityKit.StartCaptureSession()). The scan will automatically start when the ARSession (ARFoundation) is enabled.
	
	Example of use (for manual):
	
	`var aRSession = GetComponent<ARSession>();`
	`aRSession.enabled = true;`
	`aRSession.Combine_And_Start_RPU_XR_Session();`

Utilities

A utility is reusable and modular code that can be used in your various projects. These utilities are focused on solving specific, individual tasks that will help you create your application.

Quick Look

When showing files in your app, including the ability to quickly preview a file and its content can be helpful to your users. For example, you may want to allow users to zoom into a photo, play back an audio file, and so on. Use the Quick Look framework to show a preview of common file types in your app that allows basic interactions.

Quick Look can generate previews for common file types, including:

- iWork and Microsoft Office documents;
- Images;
- Live Photos;
- Text files;
- PDFs;
- Audio and video files;
- Augmented reality objects that use the USDZ file format (iOS and iPadOS only);

Note

The list of supported common file types may change between operating system releases.

Examples of use:

Quick Look:

```
public class CustomQuickLook : MonoBehaviour
{
  // A function that opens the object for quick viewing.
  public void QuickLook()
  {
    QuickLook.OpenQuickLook(string path);
  }
}
```

CSGeometry

Constructive solid geometry (formerly called computational binary solid geometry) is a technique used in solid modeling. Constructive solid geometry allows a modeler to create a complex surface or object by using Boolean operators to combine simpler objects, potentially generating visually complex objects by combining a few primitive ones.

Examples of use:

Additive:

```
public class CustomCSG : MonoBehaviour
{
  // A function that combines objects.
  public void Additive()
  {
    SilverTau.Utilities.CSGeometry.Additive(GameObject targetObject, GameObject[]
objects);
  }
}
```

Subtractive:

```
public class CustomCSG : MonoBehaviour
{
  // A function that subtractive objects.
  public void Subtractive()
  {
    SilverTau.Utilities.CSGeometry.Subtractive(GameObject targetObject, GameObject[]
objects);
  }
}
```

Intersect:

```
public class CustomCSG : MonoBehaviour
{
  // A function that intersect objects.
  public void Intersect()
  {
    SilverTau.Utilities.CSGeometry.Intersect(GameObject targetObject, GameObject[]
objects);
  }
}
```

DeIntersect:

```
public class CustomCSG : MonoBehaviour
{
  // A function that deintersect objects.
  public void DeIntersect()
  {
    SilverTau.Utilities.CSGeometry.DeIntersect(GameObject targetObject, GameObject[]
objects);
  }
}
```

Screenshot

Your app can take a screenshot in real time. The result is an image that is the same size as view, with view and its subviews drawn into it.

Examples of use:

Take a screenshot:

```
public class CustomScreenshot : MonoBehaviour
{
  // A function that takes a screenshot with certain parameters.
  public void TakeScreenshot()
  {
    Screenshot.TakeScreenshot(true, true);
  }
}
```

Share your screenshot:

When you use the share screenshot function, the target screenshot will be the last one you took.

```
public class CustomScreenshotShare : MonoBehaviour
{
  // The function that performs the action shares the last screenshot taken.
  public void ScreenshotShare()
  {
    Screenshot.ScreenshotShare();
  }
}
```

Screen Recorder

Your app can record the audio and video inside of the app, along with user commentary through the microphone. You get a reference to the recorder through the shared() function and use it to implement start-and-stop recording functionality. You can present a user interface (view controller) where a user can trim and preview recordings, and share them with other users.

Examples of use:

Take a video of the screen:

```
public class CustomScreenRecorder : MonoBehaviour
{
  // A function that starts recording the screen with certain parameters.
  public void StartScreenRecorder()
  {
    ScreenRecorder.StartScreenRecorder(true, true);
  }

  // A function that stops screen recording.
  public void StopScreenRecorder()
  {
    ScreenRecorder.StopScreenRecorder();
  }
}
```

Share

Your app can share any files and folders. Share utility is a simple way to share content from one app to another, even from different developers.

Examples of use:

Share a folder & file:

```
public class CustomShare : MonoBehaviour
{
  // A function that allows you to share a folder.
  public void ShareFolder()
  {
    Share.ShareFolder(string directoryPath);
  }

  // A function that allows you to share a file.
```

```
  public void ShareFile()
  {
    Share.ShareFile(string usdzPath);
  }
}
```

OBJ

Using the utilities developed and built into the plugin, you can easily import Unity models into both the Runtime and Editor.

Examples of use:

Import:

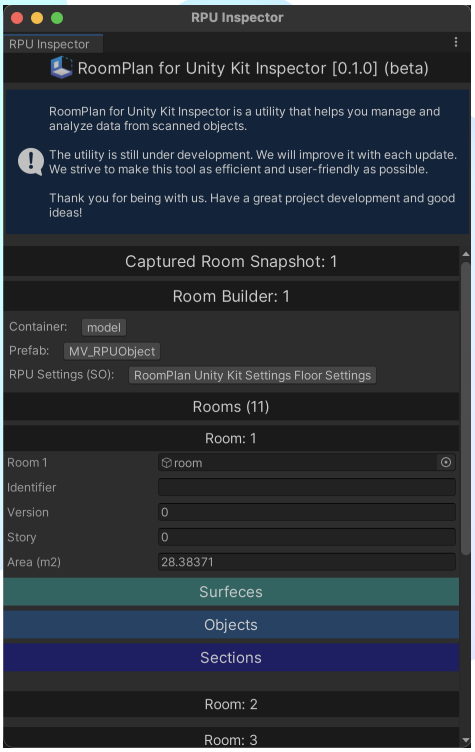
```
public class CustomOBJ : MonoBehaviour
{
  // Import an OBJ file from a file path. This function will also attempt to load the
MTL defined in the OBJ file.
  public void Import(string path, Shader shader)
  {
    var importObject = OBJUtility.Import(path, shader);
  }

  // Import an OBJ and MTL file from a file path.
  public void Import(string path, string mtlPath, Shader shader)
  {
    var importObject = OBJUtility.Import(path, mtlPath, shader);
  }

  // Import an OBJ file from a stream. No materials will be loaded, and will instead be
supplemented by a blank white material.
  public void Import(Stream input, Shader shader)
  {
    var importObject = OBJUtility.Import(input, shader);
  }

  // Import an OBJ and MTL file from a stream.
  public void Import(Stream input, Stream mtlInput Shader shader)
  {
    var importObject = OBJUtility.Import(input, mtlInput, shader);
  }
}
```

Export:

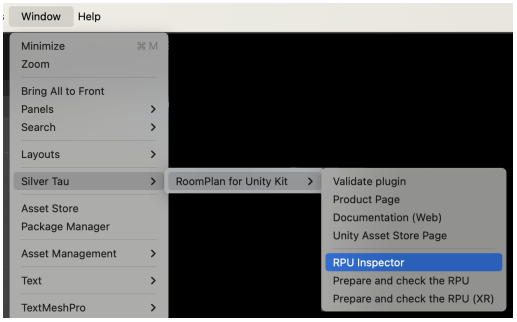
```
public class CustomOBJ : MonoBehaviour
{
  // Export game object to OBJ format model.
  public void Export(GameObject exportObject, string path, string modelName)
  {
    if(string.IsNullOrEmpty(path)) return;

    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }

    path = Path.Combine(path, modelName + ".obj");
    OBJUtility.Export(path, exportObject.transform);
  }
}
```

RPU Inspector

RoomPlan for Unity Kit Inspector is a utility that helps you manage and analyze data from scanned objects.



Use the shortcut menu to open the RPU Inspector:



Note: *For the Captured Room Snapshot (AR stage) object, tracking is automatic.*

Examples of use:

```
public class CustomRoomBuilder : MonoBehaviour
{
  // Create a new RoomBulder and add it to the RPU inspector.
  public void NewRoomBuilder()
  {
    // Create a new RoomBulder.
    var roomBuilder = new RoomBuilder
    {
       container = targetContainer,
       prefabRoomPlanObject = targetPrefab
    }

    // The RoomBuilder object is added to the inspector's RPU.
#if UNITY_EDITOR
    RPUInspector.AddRoomBulder(roomBuilder);
#endif
  }
}
```