Table of Contents

Quick Start

The example code in this quickstart guide is for educational and demonstration purposes only. It may not represent best practices for production use.

Most functionality in this package comes from two components: `DepthPreprocessor` and `DepthMesher`. Below is a quick guide to get them set up in your scene for runtime environment meshing on Meta Quest.

For full API details, see the reference documentation.

DepthPreprocessor

This component fetches depth frames from the Meta OpenXR API and exposes the depth texture to shaders. It's similar to Meta's `EnvironmentDepthManager` but adds exposure of intrinsic frame data (e.g., view and projection matrices). It also generates and updates two RenderTextures:

- **Worldspace Positions**: 3D positions for each pixel in the depth texture (in world coordinates).
- **Worldspace Normals**: Surface normals for the corresponding points.

These textures feed into `DepthMesher` for real-time mesh generation.

This script **directly conflicts** with `EnvironmentDepthManager`, `AROcclusionManager`, and any other script using `XROcclusionSubsystem.TryGetFrame()`. It maintains partial compatibility by setting global shader variables (e.g., `_EnvironmentDepthTexture`) for Meta's occlusion shaders.

See the API reference for more details.

DepthMesher

This consumes data from `DepthPreprocessor` to build a dynamic mesh using the Surface Nets algorithm. It can then assign the mesh to a `MeshFilter`, bake collision via the Jobs system and assign it to a `MeshCollider`, and bake a NavMesh with `NavMeshSurface`.

`DepthMesher` requires an instance of `DepthPreprocessor` in the same scene.

Main Editor Variables

These key settings are exposed in the Inspector for tuning mesh quality, performance, and scope:

Variable	Description	Default	Constraints
Volume Size	The dimensions of the TSDF volume grid (width x height x depth). Higher resolutions improve detail but increase memory usage and compute cost.	256x64x256	Vector3Int

Variable	Description	Default	Constraints
Meters Per Voxel	The real-world size represented by each voxel (in meters). Smaller values yield finer detail but require larger volumes.	0.1m	Min: 0
Min View Distance	The minimum distance from the camera at which depth data is considered for meshing (ignores closer user-occluded data).	1m	Min: 0
Max View Distance	The maximum distance from the camera at which depth data is considered for meshing.	4m	Min: 0
Max Mesh Update Distance	The maximum distance from the user's position to update the mesh (optimizes for local changes).	4m	Min: 0
Triangles Budget	The maximum number of triangles allowed in the generated mesh (caps GPU memory usage).	262144	Int
Target Volume Update Rate Hertz	The target update frequency for the TSDF volume (in Hz). Higher rates improve responsiveness but increase GPU load.	45	Min: 0
Target Mesh Refresh Rate Hertz	The target refresh frequency for the generated mesh (in Hz). Lower rates reduce CPU overhead in stable scenes.	1	Min: 0

Behavior Notes

- The TSDF volume is automatically cleared on recenter (via `OVRManager.display.RecenteredPose`). This ensures mesh accuracy, as recentering resets the tracking space offset and invalidates prior voxel data.

Utility Component: CPUDepthSampler

See the API reference for details on using `CPUDepthSampler` to asynchronously sample world-space positions from depth data (e.g., for raycasting or occlusion checks).

Breaking Changes Notice

If you've just updated the package, check the changelogs for information on breaking changes.