Table of Contents

Namespace Uralstech.UXR.QuestMeshing

Classes

CPUDepthSampler

A utility for asynchronously sampling world-space positions from normalized device coordinates (NDC) in the depth texture using a compute shader. This class batches requests for efficiency and ensures main-thread safety for XR frame access.

DepthMesher

Converts the Meta Quest's Depth API textures into a 3D mesh in worldspace using the surface nets algorithm.

DepthPreprocessor

Script that will preprocess depth data for meshing.

XROcclusionFrameExtensions

Extensions for Unity's XROcclusionFrame.

Structs

ComputeShaderKernel

A light wrapper for a compute shader kernel.

DepthMesher.MeshColliderBakeJob

Enums

DepthEye

Specifies the eye for depth sampling operations.

Class CPUDepthSampler

Namespace: Uralstech.UXR.QuestMeshing

A utility for asynchronously sampling world-space positions from normalized device coordinates (NDC) in the depth texture using a compute shader. This class batches requests for efficiency and ensures main-thread safety for XR frame access.

```
public class CPUDepthSampler : DontCreateNewSingleton<CPUDepthSampler>
```

Inheritance

object ← CPUDepthSampler

Remarks

This sampler relies on DepthPreprocessor for depth texture and reprojection data. It is designed for CPU-side depth queries, such as raycasting against environment depth or occlusion checks in custom XR interactions.

Usage Notes:

- Requests are batched and dispatched at the end of the frame via Uralstech.UXR.QuestMeshing.CPUDepthSampler.DispatchForSamplingAsync(System.Collections.Generic.List{Vector3},Uralstech.UXR.QuestMeshing.ComputeShaderKernel,System.Int32,System.Int32,System.Action) to minimize GPU overhead.
- All public sampling methods require main-thread execution for XR subsystem access and yield to the main thread if called from elsewhere.
- Invalid NDC positions (outside [0,1]) will return invalid results—validate with IsValidNDC(Vector2) before sampling.
- Errors (e.g., GPU readback failures) return null; check return values before use.

Single Sample:

```
if (CPUDepthSampler.Instance.ConvertToNDCPosition(worldPos, DepthEye.Left, out Vector2? ndc)
    && CPUDepthSampler.IsValidNDC(ndc.Value))
{
    Vector3? depthPos = await CPUDepthSampler.Instance.SampleDepthAsync(ndc.Value);
    if (depthPos.HasValue) { /* Process data */ }
}
```

Batch Sample:

```
ArraySegment<Vector3>? results = await sampler.BatchSampleDepthAsync(new Vector2[] { ndc1,
ndc2 });
if (results.HasValue)
{
    foreach (Vector3 pos in results.Value) { /* Process data */ }
}
```

Methods

Awake()

```
protected override void Awake()
```

ConvertToNDCPosition(Vector3, DepthEye, out Vector2?)

Converts a world-space position to normalized device coordinates (NDC) in the depth texture for the specified eye.

```
public bool ConvertToNDCPosition(Vector3 worldPosition, DepthEye eye, out
Vector2? ndcPosition)
```

Parameters

`worldPosition` Vector3

The world-space position to project.

`eye` DepthEye

The eye for which to compute the NDC (affects reprojection matrix).

`ndcPosition` Vector2?

The resulting NDC position (x,y in [0,1] range), or null if preprocessor data is unavailable.

Returns

bool

true if conversion succeeded (preprocessor data available); otherwise false.

Remarks

This uses the current frame's reprojection matrix from DepthReprojectionMatrices.

IsValidNDC(Vector2)

Determines if an NDC position is valid for depth texture sampling.

```
public static bool IsValidNDC(Vector2 position)
```

Parameters

`position` Vector2

The NDC position to validate.

Returns

bool

true if the position is within the [0,1] range for both x and y; otherwise false.

SampleNormalAsync(Vector2)

Asynchronously samples the world-space depth normal at the given NDC coordinates in the depth texture.

```
public Awaitable<Vector3?> SampleNormalAsync(Vector2 ndcPosition)
```

Parameters

`ndcPosition` Vector2

The NDC position (x,y in [0,1] range) to sample.

Returns

Awaitable<Vector3?>

The world-space depth normal at the sample point, or null if data is unavailable or sampling fails.

Remarks

This queues the request and batches it with others for end-of-frame dispatch.

SampleNormalsAsync(IEnumerable<Vector2>)

Asynchronously samples world-space depth normals for a batch of NDC coordinates in the depth texture.

```
public Awaitable<ArraySegment<Vector3>?> SampleNormalsAsync(IEnumerable<Vector2>
ndcPositions)
```

Parameters

`ndcPositions` IEnumerable<Vector2>

The enumerable of NDC positions (x,y in [0,1] range) to sample.

Returns

Awaitable<ArraySegment<Vector3>?>

An ArraySegment of results in input order, or null if data is unavailable or sampling fails.

Remarks

This queues the batch and dispatches at end-of-frame for efficiency. Ideal for multiple queries. The returned segment shares the underlying results array (do not modify it externally).

SamplePositionAsync(Vector2)

Asynchronously samples the world-space depth position at the given NDC coordinates in the depth texture.

```
public Awaitable<Vector3?> SamplePositionAsync(Vector2 ndcPosition)
```

Parameters

`ndcPosition` Vector2

The NDC position (x,y in [0,1] range) to sample.

Returns

Awaitable<Vector3?>

The world-space depth position at the sample point, or null if data is unavailable or sampling fails.

Remarks

This queues the request and batches it with others for end-of-frame dispatch.

SamplePositionsAsync(IEnumerable<Vector2>)

Asynchronously samples world-space depth positions for a batch of NDC coordinates in the depth texture.

```
public Awaitable<ArraySegment<Vector3>?> SamplePositionsAsync(IEnumerable<Vector2>
ndcPositions)
```

Parameters

`ndcPositions` IEnumerable<Vector2>

The enumerable of NDC positions (x,y in [0,1] range) to sample.

Returns

Awaitable<ArraySegment<Vector3>?>

An ArraySegment of results in input order, or null if data is unavailable or sampling fails.

Remarks

This queues the batch and dispatches at end-of-frame for efficiency. Ideal for multiple queries. The returned segment shares the underlying results array (do not modify it externally).

Start()

```
protected void Start()
```

Struct ComputeShaderKernel

Namespace: Uralstech.UXR.QuestMeshing

A light wrapper for a compute shader kernel.

```
public readonly struct ComputeShaderKernel
```

Constructors

ComputeShaderKernel(ComputeShader, string)

```
public ComputeShaderKernel(ComputeShader shader, string name)
```

Parameters

`shader` ComputeShader

`name` string

Fields

KernelIndex

The index of the kernel.

```
public readonly int KernelIndex
```

Field Value

int

ThreadGroupSizes

The thread group sizes of the kernel as defined in its numthreads attribute.

```
public readonly (uint x, uint y, uint z) ThreadGroupSizes
```

Field Value

(uint x, uint y, uint z)

Methods

Dispatch(int, int, int)

Dispatches the shader kernel.

```
public void Dispatch(int threadsX, int threadsY = 1, int threadsZ = 1)
```

Parameters

`threadsX` int

The total threads for computation in the X dimension.

`threadsY` int

The total threads for computation in the Y dimension.

`threadsZ` int

The total threads for computation in the Z dimension.

Dispatch(Vector3Int)

Dispatches the shader kernel.

```
public void Dispatch(Vector3Int threads)
```

Parameters

`threads` Vector3Int

The total threads for computation in 3 dimensions.

SetBuffer(int, ComputeBuffer)

Sets a buffer for the kernel.

```
public void SetBuffer(int id, ComputeBuffer buffer)
```

Parameters

`id` int

The parameter ID of the buffer as defined in the shader.

`buffer` ComputeBuffer

The buffer to set.

SetBuffer(int, GraphicsBuffer)

Sets a buffer for the kernel.

```
public void SetBuffer(int id, GraphicsBuffer buffer)
```

Parameters

`id` int

The parameter ID of the buffer as defined in the shader.

`buffer` GraphicsBuffer

The buffer to set.

SetTexture(int, Texture)

Sets a texture for the kernel.

```
public void SetTexture(int id, Texture texture)
```

Parameters

`id` int

The parameter ID of the texture as defined in the shader.

`texture` Texture

The texture to set.

Enum DepthEye

Namespace: Uralstech.UXR.QuestMeshing

Specifies the eye for depth sampling operations.

```
public enum DepthEye
```

Fields

`Left = 0`

The left eye.

`Right = 1`

The right eye.

Class DepthMesher

Namespace: Uralstech.UXR.QuestMeshing

Converts the Meta Quest's Depth API textures into a 3D mesh in worldspace using the surface nets algorithm.

```
public class DepthMesher : DontCreateNewSingleton<DepthMesher>
```

Inheritance
object ← DepthMesher

Fields

TargetMeshRefreshRateHertz

The target refresh frequency for the generated mesh (in Hz). Lower rates reduce CPU overhead for stable scenes.

```
public float TargetMeshRefreshRateHertz
```

Field Value

float

TargetVolumeUpdateRateHertz

The target update frequency for the TSDF volume (in Hz). Higher rates improve responsiveness but increase GPU load; lower rates reduce overhead in stable scenes.

```
public float TargetVolumeUpdateRateHertz
```

Field Value

float

Properties

Mesh

The generated mesh.

```
public Mesh Mesh { get; }
```

Property Value

Mesh

Volume

The TSDF volume used for the surface nets operation.

```
public RenderTexture Volume { get; }
```

Property Value

RenderTexture

Methods

Awake()

```
protected override void Awake()
```

Clear()

Clears the TSDF volume and vertex-index buffer, essentially resetting the system.

```
public void Clear()
```

OnDestroy()

```
protected void OnDestroy()
```

OnDisable()

```
protected void OnDisable()
```

OnEnable()

```
protected void OnEnable()
```

Start()

```
protected void Start()
```

Events

OnMeshRefreshed

Invoked after the Mesh is updated and all optional collision and NavMesh baking completes.

```
public event Action? OnMeshRefreshed
```

Event Type

Action

Struct DepthMesher.MeshColliderBakeJob

Namespace: Uralstech.UXR.QuestMeshing

```
public readonly struct DepthMesher.MeshColliderBakeJob : IJob
```

Implements
IJob

Constructors

MeshColliderBakeJob(int)

```
public MeshColliderBakeJob(int meshId)
```

Parameters

`meshId` int

Fields

MeshId

```
public readonly int MeshId
```

Field Value

int

Methods

Execute()

```
public void Execute()
```

Class DepthPreprocessor

Namespace: Uralstech.UXR.QuestMeshing

Script that will preprocess depth data for meshing.

```
public class DepthPreprocessor : DontCreateNewSingleton<DepthPreprocessor>
```

Inheritance

object ← DepthPreprocessor

Remarks

This script by itself does very little. It is meant to be used with shaders which utilize the data exposed by this script and with other scripts from this package, like DepthMesher and CPUDepthSampler.

This **directly conflicts** with Meta's EnvironmentDepthManager and any other script that acquires the depth textures using XROcclusionSubsystem.TryGetFrame, potentially including Unity's AROcclusionManager. This script automatically populates the following global shader variables, to keep some compatibility with EnvironmentDepthManager:

- _EnvironmentDepthTexture
- _EnvironmentDepthReprojectionMatrices
- _EnvironmentDepthZBufferParams
- _PreprocessedEnvironmentDepthTexture

The following custom global shader variables are also defined by this script:

- **_EnvironmentDepthInverseReprojectionMatrices** - The inverse of _EnvironmentDepthReprojectionMatrices, used to convert points from the current depth texture frame into world space.
- **_EnvironmentDepthProjectionMatrices** - Projection matrices for the current depth texture frame.
- **_EnvironmentDepthViewMatrices** - View matrices for the current depth texture frame.
- **_EnvironmentDepthWorldPositionsTexture** - Texture containing world-space positions from the XR depth texture (computed for the left eye).
- **_EnvironmentDepthWorldNormalsTexture** - Texture containing world-space normals from the XR depth texture (computed for the left eye).

Fields

DepthInverseReprojectionMatrices

The inverse of DepthReprojectionMatrices, used to convert points from the current depth texture frame into world space.

```
public readonly Matrix4x4[] DepthInverseReprojectionMatrices
```

Field Value

Matrix4x4[]

DepthProjectionMatrices

Projection matrices for the current depth texture frame.

```
public readonly Matrix4x4[] DepthProjectionMatrices
```

Field Value

Matrix4x4[]

DepthReprojectionMatrices

Matrices used to convert points in world space to points in the current depth texture frame.

```
public readonly Matrix4x4[] DepthReprojectionMatrices
```

Field Value

Matrix4x4[]

DepthViewMatrices

View matrices for the current depth texture frame.

```
public readonly Matrix4x4[] DepthViewMatrices
```

Field Value

Matrix4x4[]

EnableSoftOcclusion

```
public bool EnableSoftOcclusion
```

Field Value

bool

OnDataAvailable

Invoked when this script processes its first depth frame after enabling.

```
public UnityEvent? OnDataAvailable
```

Field Value

UnityEvent?

Properties

IsDataAvailable

Flag set when this script processes its first depth frame after being enabled.

```
public bool IsDataAvailable { get; }
```

Property Value

bool

IsSupported

Is the Meta Quest Depth API supported on the current device?

```
public bool IsSupported { get; }
```

Property Value

bool

Remarks

Do not use this from Awake.

NormalsTexture

Texture containing the normal vectors of the points in the XR depth texture converted to world space.

```
public RenderTexture? NormalsTexture { get; }
```

Property Value

RenderTexture?

PositionsTexture

Texture containing the points in the XR depth texture converted to world space.

```
public RenderTexture? PositionsTexture { get; }
```

Property Value

RenderTexture?

Methods

Awake()

```
protected override void Awake()
```

OnDestroy()

```
protected void OnDestroy()
```

OnDisable()

```
protected void OnDisable()
```

OnEnable()

```
protected void OnEnable()
```

Class XROcclusionFrameExtensions

Namespace: Uralstech.UXR.QuestMeshing

Extensions for Unity's XROcclusionFrame.

```
public static class XROcclusionFrameExtensions
```

Inheritance

object ← XROcclusionFrameExtensions

Methods

GetBaseFrameData(XROcclusionFrame, Matrix4x4[], Matrix4x4[], out Vector4)

Constructs:

- A perspective projection matrix for each eye
- The view matrix for each eye from its Pose
- Parameters used for handling the depth buffer for shaders using "_EnvironmentDepthZBufferParams".
  for a given XROcclusionFrame.

```
public static bool GetBaseFrameData(this XROcclusionFrame frame, Matrix4x4[]
projectionMatrices, Matrix4x4[] viewMatrices, out Vector4 zBufferParams)
```

Parameters

`frame` XROcclusionFrame

`projectionMatrices` Matrix4x4[]

The array to populate the projection matrices in, must be of length 2.

`viewMatrices` Matrix4x4[]

The array to populate the view matrices in, must be of length 2.

`zBufferParams` Vector4

Resulting depth buffer parameters for "_EnvironmentDepthZBufferParams".

Returns

bool

A boolean representing the success of the operation.

GetZBufferParams(XRNearFarPlanes)

Calculates parameters used for handling the depth buffer for shaders using "_EnvironmentDepthZBufferParams".

```
public static Vector4 GetZBufferParams(this XRNearFarPlanes planes)
```

Parameters

`planes` XRNearFarPlanes

Returns

Vector4

The depth buffer parameters.

TryGetReprojectionMatrices(XROcclusionFrame, Matrix4x4, Matrix4x4[], Matrix4x4[], Matrix4x4[], out Vector4)

Uses data from GetBaseFrameData(XROcclusionFrame, Matrix4x4[], Matrix4x4[], out Vector4) to construct reprojection matrices which can be used to convert world space points to points in the depth texture.

```
public static bool TryGetReprojectionMatrices(this XROcclusionFrame frame, Matrix4x4
worldToLocalMatrix, Matrix4x4[] reprojectionMatrices, Matrix4x4[] projectionMatrices,
Matrix4x4[] viewMatrices, out Vector4 zBufferParams)
```

Parameters

`frame` XROcclusionFrame

`worldToLocalMatrix` Matrix4x4

A matrix used to convert world space points into the local space of the XR session.

`reprojectionMatrices` Matrix4x4[]

The array to populate the reprojection matrices in, must be of length 2.

`projectionMatrices` Matrix4x4[]

The array to populate the projection matrices in, must be of length 2.

`viewMatrices` Matrix4x4[]

The array to populate the view matrices in, must be of length 2.

`zBufferParams` Vector4

Resulting depth buffer parameters for "_EnvironmentDepthZBufferParams".

Returns

bool

A boolean representing the success of the operation.