Retro Water - Scripting API

Runtime types, profile control, sampling, ocean management, buoyancy state, enums, and code examples.

All runtime types are in:

```
using STLDynamics.RetroWater;
```

This document covers the intended public surface of the current source snapshot.

`RetroWaterProfile`

Source: `Runtime/Profiles/RetroWaterProfile.cs`

Public properties

```
string DisplayName
string Description
Material BaseMaterial
Material BakedMaterial
RetroWaterMaterialProfileSettings MaterialSettings
RetroWaterSurfaceProfileSettings SurfaceSettings
RetroWaterCausticsProfileSettings CausticsSettings
RetroWaterUnderwaterProfileSettings UnderwaterSettings
RetroWaterOceanProfileSettings OceanSettings
int Revision
```

Create a material instance

```
Material runtimeMaterial = profile.CreateMaterialInstance("Storm Water");
```

The method resolves the `STL Dynamics/Retro Water` shader, optionally clones the Base Material, applies all profile material settings, and returns a normal runtime `Material`.

The caller owns materials created directly through this method.

Apply to an existing material

```
profile.ApplyToMaterial(targetMaterial);
```

This applies texture, colors, mapping, motion, retro rendering, lighting, depth, render state, keywords, and render queue.

Capture a material

```
profile.CaptureFromMaterial(sourceMaterial);
```

Only properties present on the source material are copied.

Update profile identity and sources

```
profile.SetIdentity("Storm Ocean", "Dark high-energy ocean water.");
profile.SetBaseMaterial(templateMaterial);
profile.SetBakedMaterial(sharedMaterial);
profile.MarkChanged();
```

Call `MarkChanged()` after changing nested settings through code so profile-driven surfaces detect the revision change.

Example:

```
profile.MaterialSettings.WaveHeight = 0.5f;
profile.MaterialSettings.FlowSpeed = 0.9f;
profile.UnderwaterSettings.FogDensity = 0.25f;
profile.MarkChanged();
```

`RetroWaterSurface`

SOURCES

- `Runtime/Scripts/RetroWaterSurface.cs`
- `Runtime/Scripts/RetroWaterSurface.Profile.cs`

Public state

```
Renderer SurfaceRenderer
RetroWaterOcean OceanSource
bool FiniteSurface

bool UseProfile
RetroWaterProfile Profile
RetroWaterProfileMaterialMode ProfileMaterialMode
Material ManualMaterial
Material ActiveMaterial

bool CausticsEnabled
int CausticsPriority
bool UnderwaterEffectsEnabled
int UnderwaterPriority
```

Assign or swap a profile

```
water.SetProfile(profile);
water.SetProfile(profile, rebuildOcean: false);
```

`SetProfile(profile)` enables profile use, invalidates the cached revision, applies the profile, and rebuilds an attached ocean by default.

`UpdateProfile(profile)` is an alias intended for gameplay state changes.

```
water.UpdateProfile(stormProfile);
```

Reapply the current profile

```
water.UpdateProfile();
water.RefreshProfile();
water.ApplyProfile();
water.ApplyProfile(rebuildOcean: false);
```

The parameterless update/refresh methods rebuild an attached ocean.

Change material mode

```
water.SetProfileMaterialMode(
    RetroWaterProfileMaterialMode.BakedProfileMaterial);
```

Available values:

```
RuntimeInstance
BakedProfileMaterial
ManualMaterial
```

Assign a manual material

```
water.SetManualMaterial(manualWaterMaterial);
```

The material is applied immediately when the surface is not profile-driven or is using Manual Material mode.

Capture a surface into a profile

```
water.CaptureCurrentSettingsToProfile(
    targetProfile,
    captureMaterial: true,
    captureOcean: true);
```

This captures:

- Current compatible material values.
- Surface mapping and sampling.
- Caustics.
- Underwater effects.
- Ocean configuration when requested and present.

Refresh references and parameters

```
water.RefreshSurfaceData();
water.RefreshMaterialParameters();
```

`RefreshSurfaceData()` resolves references, applies stale profile data, rebuilds bounds, and refreshes wave material parameters.

Sample the visible surface

```
if (water.TrySample(worldPosition, out RetroWaterSample sample))
{
    Vector3 surfacePosition = sample.Position;
    Vector3 surfaceNormal = sample.Normal;
    Vector3 waterVelocity = sample.Velocity;
    float displacement = sample.Displacement;
}
```

A `false` result means the query is outside a finite surface, no valid mapping can be resolved, or the surface is otherwise unavailable.

Evaluate wave displacement

```
float displacement = water.EvaluateWave(uv, Time.time);
```

The UV must match the surface's CPU wave coordinate system. For general gameplay queries, prefer `TrySample()`.

Find the nearest registered surface

```
RetroWaterSurface surface =
    RetroWaterSurface.FindBestSurface(position, maximumVerticalDistance: 8f);
```

The method samples every active surface and chooses the closest valid result along gravity.

`RetroWaterSample`

Source: `Runtime/Scripts/RetroWaterSample.cs`

```
public readonly struct RetroWaterSample
{
    public readonly Vector3 Position;
    public readonly Vector3 Normal;
    public readonly Vector3 Velocity;
    public readonly float Displacement;
}
```

- `Position` - world-space animated surface position.
- `Normal` - world-space sampled surface normal.
- `Velocity` - world-space water velocity, including optional physical flow.
- `Displacement` - signed displacement along the local plane normal.

`RetroWaterOcean`

SOURCES

- `Runtime/Scripts/RetroWaterOcean.cs`
- `Runtime/Scripts/RetroWaterOcean.Profile.cs`

Public state

```
Material WaterMaterial
Renderer ReferenceRenderer
MeshFilter ReferenceMeshFilter
bool IsOceanReady
Vector2 OceanSize
float EffectiveChunkSize
int GeneratedChunkCount
Bounds LocalOceanBounds
Vector2 WaveUvPerWorldUnit
Vector2 WaveUvOffset
Transform Viewer
```

Set the viewer

```
ocean.SetViewer(gameplayCamera.transform);
```

This invalidates the cached viewer position so LODs update immediately.

Set the material

```
ocean.SetWaterMaterial(material, rebuild: true);
```

Build or clear

```
ocean.BuildOcean();
ocean.ClearGeneratedOcean();
```

`BuildOcean()` validates settings, recalculates effective chunk size, creates LOD meshes and chunks, performs the initial LOD update, and refreshes the attached surface.

Apply or capture profile settings

```
ocean.ApplyProfileSettings(
    profile.OceanSettings,
    water.ActiveMaterial,
    rebuild: true);

ocean.CaptureProfileSettings(profile.OceanSettings);
profile.MarkChanged();
```

Normally `RetroWaterSurface.ApplyProfile()` handles this bridge.

Wave coordinate helpers

```
Vector2 waveCoordinates = ocean.BuildWaveCoordinates(worldPosition);
Vector4 normalizedTransform = ocean.BuildNormalizedWaveUvTransform();
Vector2 delta = ocean.GetWaveUvDelta(localDistanceU, localDistanceV);
```

Bounds helpers

```
Bounds bounds = ocean.BuildWorldOceanBounds();
float squaredDistance = ocean.GetDistanceSquaredToBounds(worldPosition);
```

`RetroWaterBuoyant`

Source: `Runtime/Scripts/RetroWaterBuoyant.cs`

Public state

```
Rigidbody TargetRigidbody
RetroWaterSurface ActiveSurface
bool IsSubmerged
float SubmergedFraction
float ActiveBuoyancyFraction
float SinkDepth
```

`SubmergedFraction` reports physical contact before the optional sink offset.

`ActiveBuoyancyFraction` reports adjusted sample participation currently producing lift and water forces.

Change sink depth

```
buoyant.SinkDepth = 0.2f;
```

Positive values make the body settle deeper. This does not change maximum lift.

Rebuild automatic points

```
buoyant.RebuildAutomaticPoints();
```

Call this after runtime collider topology or dimensions change.

Runtime profile state example

```
using STLDynamics.RetroWater;
using UnityEngine;

public sealed class WaterWeatherController : MonoBehaviour
{
    [SerializeField] private RetroWaterSurface surface;
    [SerializeField] private RetroWaterProfile calm;
    [SerializeField] private RetroWaterProfile storm;

    public void ApplyWeather(bool storming)
    {
        surface.SetProfile(storming ? storm : calm);
    }
}
```

Runtime profile mutation example

Mutating a shared ScriptableObject changes every surface using it.

```
public void SetSharedWaveHeight(
    RetroWaterProfile profile,
    float waveHeight)
{
    profile.MaterialSettings.WaveHeight = waveHeight;
    profile.MarkChanged();
}
```

For per-instance runtime changes, create a cloned profile or use a private material rather than mutating a shared project asset.

```
RetroWaterProfile runtimeProfile =
    ScriptableObject.Instantiate(sourceProfile);

runtimeProfile.MaterialSettings.WaveHeight = 0.45f;
runtimeProfile.MarkChanged();
surface.SetProfile(runtimeProfile);
```

Destroy runtime-created ScriptableObjects when their owner is destroyed.

Direct current sampling example

```
using STLDynamics.RetroWater;
using UnityEngine;

public sealed class WaterCurrentProbe : MonoBehaviour
{
    [SerializeField] private RetroWaterSurface surface;

    private void Update()
    {
        if (!surface.TrySample(transform.position, out RetroWaterSample sample))
        {
            return;
        }

        Debug.DrawRay(
            sample.Position,
            sample.Velocity,
            Color.cyan);
    }
}
```

Public enum reference

`RetroWaterProfileMaterialMode`

```
RuntimeInstance
BakedProfileMaterial
ManualMaterial
```

`RetroWaterSurfaceMode`

```
Opaque
Cutout
Transparent
```

`RetroWaterTextureSpace`

```
ObjectAnchored
WorldLocked
```

`RetroWaterSnapSpace`

```
Off
Object
World
View
```

RetroWaterTextureWrap

```
Repeat
Clamp
```

RetroWaterHighlightMode

```
Off
RetroGlints
ClassicSpecular
```

RetroWaterPlaneAxes

```
Auto
XZ
XY
YZ
```

RetroWaterBuoyancyPointMode

```
CenterOfMass
AutomaticColliderBounds
LocalOffsets
Transforms
```

RetroWaterBuoyancyPointOrientation

```
GravityAligned
BodyLocal
```