DOCUMENTATION

Shoreline Maker.

Procedural shoreline collider generator for Unity.

OneFoxStudio · version 1.1.0 · Unity 2021.3 LTS or newer

Table of Contents

01 Overview

02 Requirements

03 Installation

04 Demo Scenes

05 Quick Start

06 Components

07 Public API

08 Examples

09 Editor

10 Performance

11 Limitations

12 Troubleshooting

13 Changelog

14 Support

15 API Reference

Overview

ShorelineMaker generates collider walls along the boundary where a water surface meets a Unity terrain. It is designed for sandbox, survival, and adventure games where the player must be physically prevented from leaving an island, where you want to trigger gameplay (cold damage, splash VFX, fishing, etc.) when entering water, or where you simply want a fast, deterministic way to wall off water.

What's included:

- `ShorelineGenerator` — multi-terrain shoreline builder component.
- `ShoreZone` / `ShoreZoneTriggerRelay` — forward collider trigger events as UnityEvents.
- `ShorelineMaker` — static API used by the generator; also usable directly from your own scripts.
- `PerlinHeightmapGenerator` — utility for procedurally generating test terrains.

Requirements

- **Minimum Unity:** 2021.3 LTS.
- Render pipeline independent — generated walls are colliders; the optional wall mesh uses whatever material you assign.
- No third-party dependencies.

Installation

Import the `.unitypackage` via the Unity Asset Store, or via *Assets* > *Import Package* > *Custom Package…*

The plugin is fully contained under one root folder:

```
Assets/Plugins/OneFoxStudio/ShorelineMaker/
```

Sub-folders:

- `Core/Scripts/` — runtime and editor C# scripts.
- `Core/Scripts/Utilities/` — optional helpers (Perlin heightmap generator).
- `Demo/` — example scenes, materials, demo controllers, and the simple water shader.

Demo Scenes

Two scenes ship in `Demo/Scenes/`. Open either, hit Play, and walk around to see the wall colliders in action.

BasicTerrain STARTER

A single terrain generated by `PerlinHeightmapGenerator` — one Perlin-noise archipelago, one water plane, one `ShorelineGenerator`. The simplest end-to-end setup; a good place to tweak inspector values and watch the wall regenerate.

MultiTerrainDemo ADVANCED

A coastline that crosses multiple tiled terrains, exercising the edge-case path: chains span terrain seams instead of fragmenting per tile, terrain limits at the map border become real shore (no chord cutting across the interior), and the player can’t escape an island clipped by the map edge.

Quick Start

1. Create an empty GameObject in your scene and add the *Shoreline Generator* component.
2. Drag your target terrains into *Target Terrains*, or click *Add All Scene Terrains*.
3. Set the water source — assign a *Water Object* Transform (its world Y defines the surface) or leave it empty and enter *Water Y* manually.
4. Pick the collider mode:
   - **Box Segments** — overlapping `BoxCollider`s along the shoreline (can be a trigger).
   - **Solid Mesh** — one non-convex `MeshCollider` per chain (blocks only; cannot be a trigger).
   - **Convex Segments** — many small convex `MeshCollider`s per chain (mesh-trigger capable).
5. Hit *Generate Shoreline*.

Recommended starting values:

- Water Y: your water surface height in metres.
- Sample Spacing: 1–2 m. Min Point Spacing: 2–4 m.
- Wall Collider Mode: *Box Segments* (default).

Components

ShorelineGenerator

Component-based shoreline builder. Generates wall colliders for a flat water surface at a given world-space Y across one or more terrains.

- **Target Terrains** — Terrains to detect on (falls back to `Terrain.activeTerrain` when empty).
- **Water Source** — assign an optional *Water Object* Transform (its world Y drives Water Y), or leave empty and edit Water Y manually.
- **Output > Wall Parent** — optional Transform that the generated `Shore_NNN` objects are parented under.
- **Detection** — Sample Spacing, Min Point Spacing.
- **Wall** — Is Trigger, Wall Collider Mode, Draw Wall Mesh, Wall Height/Depth/Width, Wall Material, Close Loop, Max Segment Length.
- **Debug** — Show Gizmos, Log Summary.

Buttons: *Generate Shoreline*, *Clear*.

Wall Collider Modes

The *Wall Collider Mode* dropdown (`WallColliderMode`) selects how collision walls are built. A renderable wall mesh is built in every mode when *Draw Wall Mesh* is on; only the collider differs.

Box Segments *(default)* — One oriented `BoxCollider` per shoreline segment, each on its own child GameObject. Cheapest option: primitive narrow-phase, no mesh cooking. Can be a trigger.

Solid Mesh — One non-convex `MeshCollider` per chain. Fewest objects, exact shape. Cannot be a trigger.

Convex Segments — Many small convex `MeshCollider`s per chain. Supports mesh-based triggers. Most objects and most memory.

Rule of thumb: trigger or general use → **Box Segments**; pure static blocking with minimal object count → **Solid Mesh**; **Convex Segments** only for a niche mesh-trigger requirement.

ShoreZone and ShoreZoneTriggerRelay

`ShoreZone` forwards Unity collider trigger events to UnityEvent listeners (`onEnter` / `OnExit`) so you can wire splash VFX, fishing, cold damage, etc. without writing C#. One ShoreZone is added to each chain's `Shore_NNN` object.

Because Box Segments place each collider on a child GameObject, Unity delivers `OnTrigger*` messages to the child rather than to the parent ShoreZone. A `ShoreZoneTriggerRelay` is therefore added to each trigger box; it caches and forwards those events up to the parent ShoreZone.

API

All scripts live under the `OneFoxStudio.ShorelineMaker` namespace. The main scripting entry point is the static `ShorelineMaker` class:

```
using OneFoxStudio.ShorelineMaker;

var settings = new ShorelineMaker.Settings {
    WaterY            = 50f,
    Terrain           = myTerrain,
    FootprintTerrains = new[] { myTerrain },
    SampleSpacing     = 2f,
    MinPointSpacing   = 4f,
    ColliderMode      = WallColliderMode.BoxSegments,
    DrawWallMesh      = true,
    WallHeight        = 6f,
    IsTrigger         = true,
    CloseLoop         = true,
    MaxSegmentLength  = 100f,
};

ShorelineMaker.Clear(parent);
var result = ShorelineMaker.Build(settings, parent);
Debug.Log($"{result.Points.Count} points, {result.ChainCount} chains");

// Closest-point lookup against the detected shoreline:
if (ShorelineMaker.TryFindClosestPoint(result.Points, query, out var closest, out var
dist))
    Debug.Log($"Nearest shore point at {closest} ({dist:F2}m away)");
```

`ShorelineMaker.Result` includes diagnostic fields — `CellsAboveWater`, `CellsBelowWater`, `MinTerrainY`, `MaxTerrainY`, `WaterY` — useful when no shoreline is found.

Examples

Short, copy-pasteable recipes. Each block is self-contained — drop it into a MonoBehaviour and tweak the field names.

1. Generate from code

Build a shoreline at runtime without the inspector. Useful for procedural maps where the terrain is created on the fly.

```
using OneFoxStudio.ShorelineMaker;
using UnityEngine;

public class ShorelineBootstrap : MonoBehaviour
{
    [SerializeField] private Terrain terrain;
    [SerializeField] private Transform wallParent;
    [SerializeField] private float waterY = 50f;

    private void Start()
    {
        var result = ShorelineMaker.Build(new ShorelineMaker.Settings
        {
            Terrain          = terrain,
            WaterY           = waterY,
            SampleSpacing    = 2f,
            MinPointSpacing  = 4f,
            ColliderMode     = WallColliderMode.BoxSegments,
            WallHeight       = 6f,
            IsTrigger        = true,
            CloseLoop        = true,
            MaxSegmentLength = 100f,
            Layer            = gameObject.layer,
        }, wallParent);

        Debug.Log($"{result.Points.Count} pts, {result.ChainCount} chains");
    }
}
```

2. Closest shore point every frame

Display a UI hint when the player is within a few metres of the shoreline, e.g. for a “press F to fish” prompt.

```
using OneFoxStudio.ShorelineMaker;
using UnityEngine;

public class ShoreProximity : MonoBehaviour
{
    [SerializeField] private ShorelineGenerator shoreline;
    [SerializeField] private Transform player;
    [SerializeField] private float promptRange = 3f;

    private void Update()
    {
        if (!ShorelineMaker.TryFindClosestPoint(
                shoreline.ShorelinePoints, player.position,
                out var closest, out var dist))
            return;

        if (dist < promptRange)
            Debug.DrawLine(player.position, closest, Color.cyan);
    }
}
```

3. Multi-terrain with an explicit footprint

When several terrains tile together, set `FootprintTerrains` so the maker treats seams as land and the terrain edges as the real coast.

```
var all = Terrain.activeTerrains;
foreach (var t in all)
{
    var r = ShorelineMaker.Detect(new ShorelineMaker.Settings
    {
        Terrain           = t,
        FootprintTerrains = all,
        WaterY            = 50f,
        SampleSpacing     = 2f,
        MinPointSpacing   = 4f,
    });
    mergedPoints.AddRange(r.Points);
}

var unified = new List<Vector3>();
ShorelineMaker.ThinPoints(mergedPoints, 4f, unified);
ShorelineMaker.BuildWalls(buildSettings, unified, wallParent);
```

4. Iterate chains for custom gameplay logic

Use the chain output to drive non-collider features — placing buoys, spawning waves, computing perimeter length, etc.

```
var chains = ShorelineMaker.BuildChains(shoreline.ShorelinePoints, maxGap: 8f);

foreach (var chain in chains)
{
    float length = 0f;
    for (var i = 1; i < chain.Count; i++)
        length += Vector3.Distance(chain[i - 1], chain[i]);

    Debug.Log($"Chain of {chain.Count} pts, ~{length:F1}m around");
}
```

Editor

Components are added through Unity's *Add Component* menu under **OneFoxStudio > ShorelineMaker**. The package does not add items to Unity's main menu bar, run any `InitializeOnLoad` redirects, or open external URLs without consent.

- *Shoreline Generator* — custom inspector with *Generate Shoreline* and *Clear* buttons.
- *Perlin Heightmap Generator* — default inspector plus *Generate (Full Res)* and *Flatten*; supports an edit-mode live preview at reduced resolution.

Performance

- Prefer **Box Segments**: no mesh cooking, no mesh memory, fastest narrow-phase and raycasts.
- Use **Solid Mesh** when you only need blocking and want the fewest objects.
- Avoid **Convex Segments** on long, detailed shorelines unless you need mesh-based triggers.
- Raise *Sample Spacing* / *Min Point Spacing* / *Max Segment Length* to reduce the number of segments.
- Turn off *Draw Wall Mesh* for invisible collision-only walls.

Limitations

- **Solid Mesh** walls cannot be triggers (PhysX does not allow non-convex triggers).
- Box Segments cannot shear: on steep segments the box over-covers the vertical span slightly (acceptable for an invisible wall).
- Shoreline detection requires terrain both above and below the water level.

Troubleshooting

"Not enough shoreline points found." The diagnostic suffix tells you why:

- *No terrain above water Y=…* — lower Water Y, or raise the terrain.
- *No terrain below water Y=…* — raise Water Y, or lower the sea floor.
- Otherwise, try a smaller *Sample Spacing*.

Too many child objects: switch *Wall Collider Mode* to **Solid Mesh**, or raise *Sample Spacing* / *Min Point Spacing* / *Max Segment Length*.

ShoreZone events not firing: Solid Mesh never fires events. Switch to Box Segments or Convex Segments, ensure *Is Trigger* is on, and that the other object has a Rigidbody.

Changelog

1.1.0 — Multi-terrain shoreline (chains span terrain seams); GameObject-tracked water Y; drop legacy `ShorelineProxy` and `RiverShorelineGenerator`; Box / Solid Mesh / Convex Segments collider modes; public closest-point helper.

1.0.0 — Initial release.

Support

Publisher: OneFoxStudio **Contact:** info@onefoxstudio.com

API Reference

Auto-generated from `///` XML doc comments in the package source.

ShorelineGenerator CLASS

Detects the shoreline where a water surface meets one or more terrains, and builds collider walls along it (box or mesh, per `WallColliderMode`). Each chain's parent gets a `ShoreZone` component so users can hook trigger events.

SIGNATURE	DESCRIPTION
PROPERTY `IReadOnlyList<Vector3> ShorelinePoints`	Read-only view of the detected shoreline points from the most recent `Generate`.
PROPERTY `IReadOnlyList<Terrain> Terrains`	Read-only view of the configured terrains.
METHOD `void SetTerrains(params Terrain[] terrains)`	Replace the terrain list outright.
METHOD `void AddTerrain(Terrain terrain)`	Append a terrain if not already present.
METHOD `int AddAllSceneTerrains()`	Fill the terrain list with every active terrain in the open scene. Returns the count.
METHOD `void Generate()`	Detect points across every configured terrain, then build wall objects.
METHOD `void Clear()`	Remove any previously generated wall children and reset the cached point list.

ShorelineMaker STATIC

Static entry-point. Detects shoreline points and builds the wall hierarchy. Reused by `ShorelineGenerator` and safe to call directly from your own scripts.

SIGNATURE	DESCRIPTION
METHOD `Result Detect(Settings settings)`	Detect shoreline points for ONE terrain without building any walls.
METHOD `Result Build(Settings, Transform, int chainIdxOffset = 0)`	Build shoreline walls under `parent`. Does NOT clear existing children.
METHOD `int BuildWalls(Settings, List<Vector3>, Transform, int offset = 0)`	Chain the points and build wall objects under `parent`.
METHOD `void ThinPoints(List<Vector3> raw, float minSpacing, List<Vector3> out)`	Drop points within *minSpacing* of one already kept.
METHOD `List<List<Vector3>> BuildChains(List<Vector3>, float maxGap)`	Greedy nearest-neighbour chaining of raw points.
METHOD `bool IsLoop(List<Vector3> chain, bool closeLoop)`	Whether a chain should be closed into a loop.
METHOD `void AddBoxSegment(Transform, Vector3 a, Vector3 b, …)`	Creates one oriented `BoxCollider` segment.
METHOD `bool TryFindClosestPoint(IReadOnlyList<Vector3>, Vector3 q, out Vector3, out float)`	Nearest-point lookup, 3D Euclidean. O(n).
METHOD `bool TryFindClosestPoint(IReadOnlyList<Vector3>, Vector3 q, out Vector3)`	Convenience overload that drops the distance out-param.
METHOD `void Clear(Transform parent)`	Destroy all generated `Shore_*` children under `parent`.

ShoreZone CLASS

Trigger zone created by shoreline generators. Forwards collider events as UnityEvents so users can wire in their own gameplay.

SIGNATURE	DESCRIPTION
FIELD `ColliderEvent` **`onEnter`**	Fires on `OnTriggerEnter`.
FIELD `ColliderEvent` **`OnExit`**	Fires on `OnTriggerExit`.

ShoreZoneTriggerRelay CLASS

Box-collider shore wall segments live on their own child GameObjects (each needs an individual transform for orientation), so Unity delivers `OnTrigger*` messages to the child, not to the `ShoreZone` on the parent. This relay forwards those events up to the nearest ShoreZone in its parents.