Map Editor Manual

You can create your own Skirmish and Multiplayer maps for the game using the Unreal Editor and the `MapEditorProject` Unreal project.

Custom maps are packaged into Unreal mod plugins that can be placed into the `Tempest Rising/Tempest/Mods/` directory or downloaded from the Steam Workshop.

Other kinds of mods and custom campaigns are not properly supported. Custom maps can include custom assets and blueprints (both level and standalone, though you cannot extend most existing ones), but they can't contain code. They can only be played in Skirmish and Custom/LAN multiplayer games.

Note that this guide assumes that you are already at least somewhat familiar with designing levels in Unreal Editor. If you aren't, refer to the **Unreal Beginner's Guide**.

The Editor project includes two example maps:

- `ZooExampleMap`: A basic example map that demonstrates most of the features mentioned in this guide.
- `SetdressingExampleMap`: A complete example map that demonstrates some set dressing techniques used in Tempest Rising levels.
    - Please refer to the **Tempest Rising Environment Art Guide** for more information.

Table of Contents

Prerequisites

- **Unreal Engine 5.4.4**
    - You'll have to install it from the Epic Games Launcher, as you normally would.
- Tempest Rising Map Editor Project
    - Install the "**Tempest Rising Mod Kit**" free DLC on Steam. You need to own the game for this.
        - You can find the DLC menu by right clicking on Tempest Rising in your Steam Library, then selecting “DLC”.
    - The project will be located in the `MapEditorProject` directory within the game directory (usually `steamapps/Tempest Rising`).
        - This directory will also contain the maps you're going to be working on, so make sure there is enough space, or copy it somewhere else.
- SteamCMD if you want to publish your map to the Steam Workshop
    - Download it from the Valve Developer Wiki.

Getting started

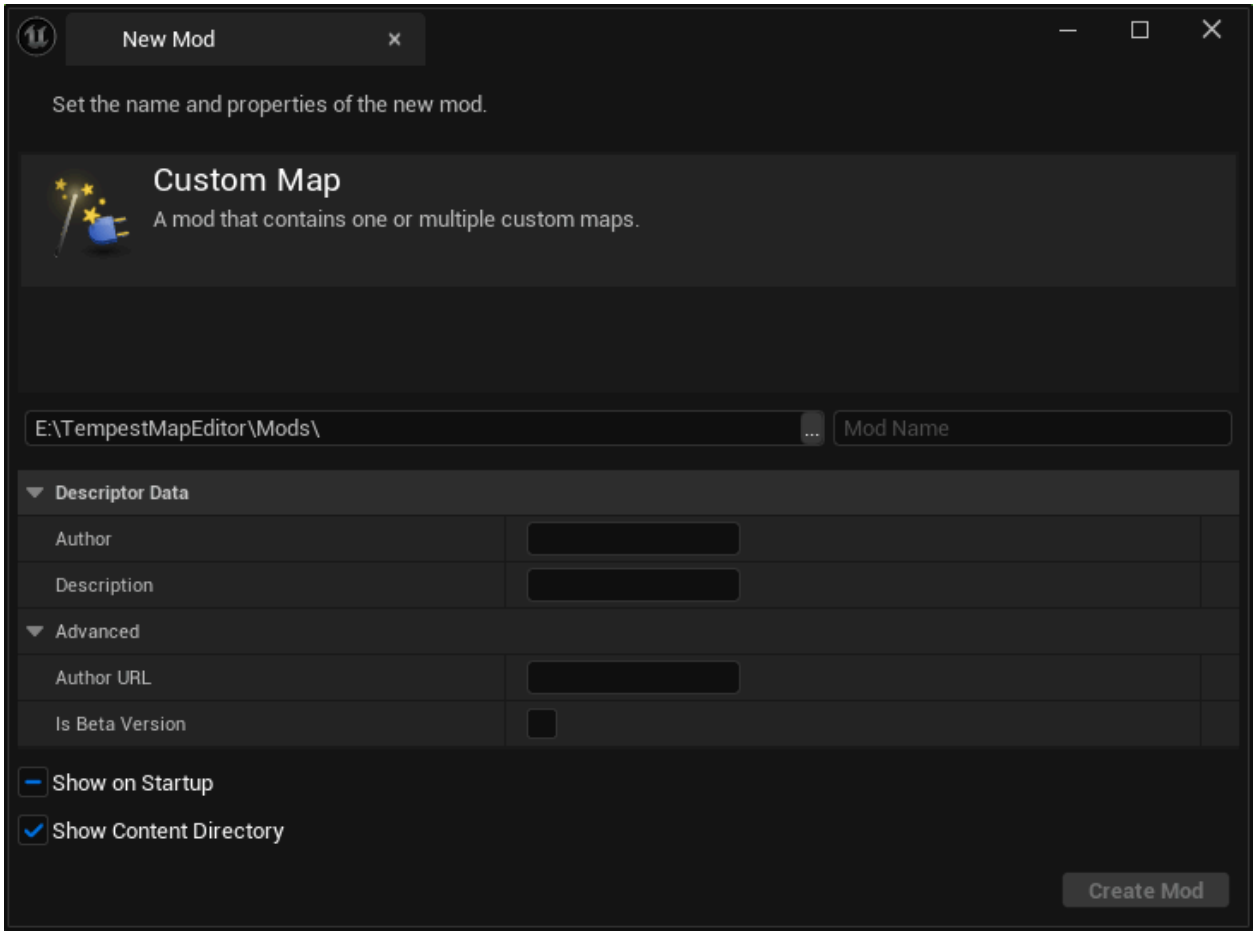
Before you do anything else, set up and open the Editor:

1. Install UE5.4.4.
2. Launch UE5.4.4 from the Unreal Engine / Library tab of the Epic Games Launcher.
3. Click *Browse* and navigate to the `MapEditorProject` folder mentioned above. Select `Tempest.uproject`.

Creating a New Map

First you need to create a new mod that will contain your custom map. A single mod can technically contain multiple custom maps, but it is best to limit it to one map per mod for organization purposes.

Select `Tempest -> Create New Mod...` in the menu bar at the top of the Editor window. This will pop up the `New Mod` dialog:



Select the `Custom Map` template and fill in the `Mod Name` field. Other fields like `Author` and `Description` are optional. By default the new mod will be placed into `MapEditorProject/Mods/<ModName>`. You can name your mod anything you'd like, but make sure the name is not too long (less than 32 characters or so).

Click `Create Mod`. This will create a new mod plugin and navigate to its `Content` directory in the Content Browser. Your custom map and any custom assets you use in it should be placed inside this directory or one of its subdirectories. You can find it again by finding your mod in the `Plugins` list in the `Content Browser`.

The editor should already be open on one of the bundled template maps. This template the basic elements and world settings necessary for a working Tempest Rising map. You can start from here or make an empty map, but make sure to include the necessary elements described in the rest of the guide. You can also create a new map from one of the other Tempest template maps (`MAP_Template_Small/Medium/Large`).

Save your new map in your mod's `Content/Maps` directory according to the naming/path conventions outlined in the section below.

If you want to delete a mod, you can just delete its directory while the Editor is not running.

Official Tempest Rising maps often consist of one persistent level and multiple sublevels: one for set dressing, one for scripting, etc.

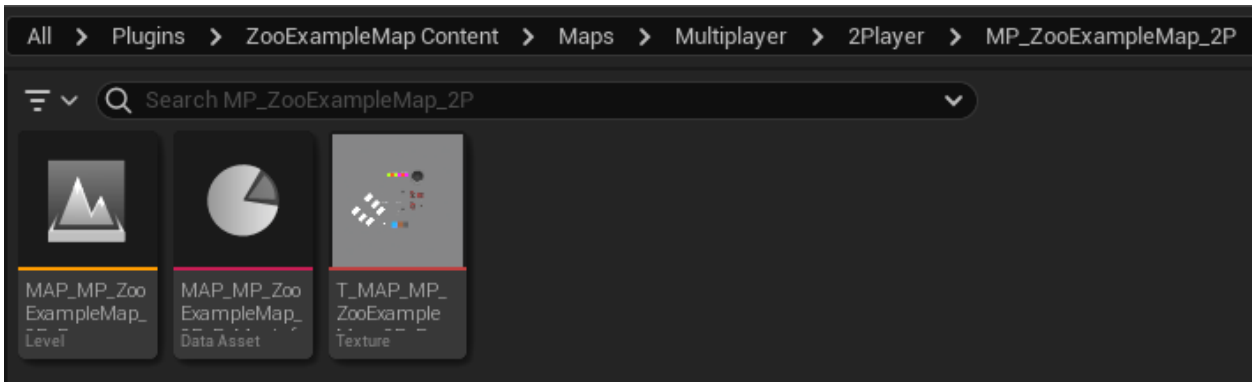
Name and Path Conventions

Your custom map and all its custom assets and blueprints should be contained within a single mod.

You can place assets into any subdirectories within the mod's `Content` directory and name them anything you want, but maps have special requirements.

Maps should be named `MAP_MP_<MapName>_#P`, where `<MapName>` is your chosen map name and `#` is the amount of players the map is for, for example `MAP_MP_ExampleMap_2P`. Persistent levels (the main or maybe only level of your map that is always loaded) should have the postfix `_P`, for example `MAP_MP_ExampleMap_2P_P`. Any sublevels should be in the same directory as the corresponding persistent level, but should have different suffixes, for example `MAP_MP_MultiLevelMap_4P_Environment`.

Maps should be placed into `Maps/Multiplayer/#Player/MP_<MapName>_#P/` within your mod's `Content` directory, where `#` is the amount of players the map is for and `<MapName>` is your chosen map name. For example the full path for the included Zoo Example Map package is `Content/Maps/Multiplayer/2Player/MP_ZooExampleMap_2P/MAP_MP_ZooExampleMap_2P_P`:



It might be a good idea to place each type of asset into its own subdirectory: custom blueprints should go into `Content/Blueprints`, static meshes into `Content/StaticMeshes`, etc.

DO NOT place any custom assets into the `Tempest` directory or the `TempestEditor` directory, or in general anywhere that's *not* inside a mod's `Content` directory. The main `Content` directory of the `MapEditorProject` project should generally only contain the resources that come with the Editor project. Nothing in that directory is included with your map when you package it for distribution.

Map Settings and Metadata

World Settings

Tempest Rising levels have several additional World Settings:

- `World Bounds`: Which `WorldBoundsActor` to use to determine the map boundaries for camera movement and fog of war purposes.
    - See the *World Bounds* section for more information.
- `Fog Of War Init Mode`: Determines the starting state of fog of war on the map. Has the following options:
    - `Default`: Map will start with 100% black fog of war over everything, like in campaign maps.
    - `Visible Terrain`: All map terrain will be visible under dimmed (transparent) fog of war, like in Skirmish/MP maps.
    - `Visible All`: Everything except what's beyond the World Bounds will be completely visible.
    - `Disabled`: Completely disables fog of war. Everything will be visible.
- `Terrain Level Z Span`: Z size of one terrain level.
    - See the Terrain section for more information.
- `Terrail Level Z Min/Max`: If enabled, clamps the Z of the lowest/highest terrain level.
    - See the Terrain section for more information.
- `Minimap State Settings`: Force enable/disable the minimap.
- `Minimap Ignores Permanent FoW`: If enabled, only the temporary/dimmed fog of war will be displayed on the minimap, the landscape will be always visible.
- `Allow Player Commands`: Enables/disables Support Powers and other commands that don't require a unit.
- `Use Map Info`: If enabled, this map will automatically generate a `Map Info` asset upon saving, if it doesn't exist already.
    - This should be checked **only** for the persistent (i.e. main) level in your map. Any sublevels should have this disabled.
    - This is **disabled** by default, so **don't forget to enable it** for persistent levels.
    - See the MapInfo section below.
- `Map Info` and other related settings: See the MapInfo section below.

If you're making a map from the Empty Level template, make sure that the `Navigation System Class` in `World Settings` is set to `TedNavigationSystem`.

MapInfo

Each Tempest Rising map should have one corresponding `TedMapInfo` asset. These assets contain various settings (which minimap texture to use, how many players the map supports, etc.) and metadata (map name, author and description).

Whenever you save a level with `Use Map Info` enabled in its `World Settings` for the first time, a `TedMapInfo` will be generated for it automatically. You can then change the settings within it either through the `Map Info` section in the `World Settings` panel of the level or by editing the asset itself.

If your map consists of multiple levels, only the persistent (i.e. main) level should have `Use Map Info` enabled. This setting is enabled by default, so don't forget to enable it for that level and leave it disabled for sublevels. There should be **only one `TedMapInfo` asset for each map**, even if the map contains multiple sublevels.

`TedMapInfo` has the following properties:

- `Display Name`: Map name as it displays in the map select menu in the game. Max 32 characters.
- `Location`: Location that the map takes place in as displayed in the map select menu.
- `Description`: Map description as displayed in the game.
- `Author`: Your name. This will display next to the map name in the game. Max 32 characters.
- `Minimap`: The Texture to use for the minimap. This can be generated using the provided tools.
    - See the Minimap section for more information.
- `Players`: Array of settings for each player slot. The game supports up to 8 players, plus two fake players for neutral and always-hostile structures/units, but your map can have a smaller limit by disabling all player slots except the required amount. Each player slot has the following settings:
    - `Control`: Who can control this player.
        - `None`: Player slot is disabled.
        - `Human`: Human or Skirmish AI players can use this slot.
        - `Bot`: Always controlled by AI.
        - `Uncontrollable`: Can't be used by actual human or AI players. Used for the Neutral and Hostile players.
    - `Player Start`: The `TedPlayerStart` actor to spawn this player at.
        - See the Player Starts section below for more information.
    - `Player Faction`: Default faction for this player. Will be overridden by Skirmish/MP settings for actual players.

        - `Player Color`: Color for this player. Will be overridden by Skirmish/MP settings for actual players.
        - `Additional Pawn GameplayTags`: GameplayTags to apply to all of this player's units and structures when they spawn.
        - `Player Data`: What resources/unlocked tech the player starts with. Will be overridden by Skirmish/MP settings for actual players.
- `Custom Attitudes`: Set up which players are always hostile/friendly/neutral to each other. Mostly useful for Uncontrollable/Bot players.
- `Mission Tracks`: Music config for this map. All the settings here use `TedJukeboxTrackAsset`s. These correspond to Tempest Rising music tracks, and you cannot add your own. This category has the following settings:
    - `Map Sound Tracks`: List of tracks that will play during gameplay in this map if Mission Tracks mode is enabled in the Jukebox. You can turn this mode on using the `JukeboxEnableMissionTracks` Blueprint node. The user can still turn it back off in the Jukebox menu.
    - `Victory Track`: Music that you hear when you win. The default track for this is `Track_5_Redemption`.
    - `Lose Track`: Music that you hear when you lose. The default track for this is `Track_7_Fallout`.
- `Supports MP`: Whether this map will be available in Custom games.
- `Supports SP`: Whether this map will be available in Skirmish.
- `Sandbox Only`: If enabled, forces this map to always use the `Sandbox` game mode, where there is no win or lose condition.
    - If this is enabled, you will have to use the `Win Game` or `Defeat Player` nodes from the Mod API to ensure a player can win the map.
- `Starting Armies`: Specifies the unit lists for different `Starting Army` settings in Skirmish/MP.
    - If you want to force players to start with nothing, reset all the settings to None.

Before packaging your map, ensure that its MapInfo has been created and set up to your liking. Maps without a MapInfo will not be visible in game.

Map Elements

Player Starts

There should be one `TedPlayerStart` for each Human or Bot player slot in the map. They can be found in the *Place Actors* panel. Keep in mind that the Player Starts mark the location where the player's Construction Yard and starting army will spawn, so make sure that the entire area covered by the `TedPlayerStart` is on valid navmesh.

Do not use the normal Unreal Engine `PlayerStart` actors.

World Bounds

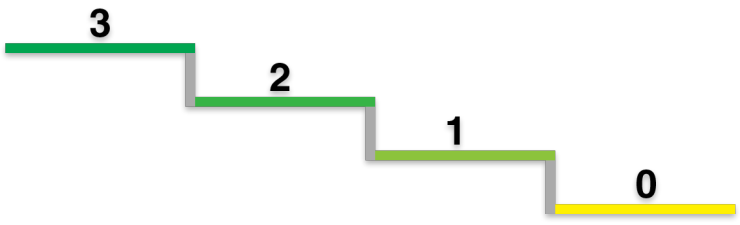
Your map should contain a `WorldBoundsActor` that covers the playable area (horizontally and vertically). You can find one in the *Place Actors* panel. Extend it a little bit under the ground. After setting it up select it in *World Settings*.

Make sure that the `WorldBoundsActor` is **perfectly square** on the horizontal plane (X size = Y size), otherwise your minimap texture won't generate correctly.

Terrain

Terrain or ground is basically anything units can move on (or air units can fly above) that's not a destructible bridge.

Tempest Rising separates terrain into several discrete height levels, by default every 700 Z units:



This is done to limit unit vision. When a unit is on level 1 for example, it cannot see levels 2 and 3, but can see level 0. Note that the level numbers in the image are purely for illustration purposes and a map can have any number of them.

The lowest height level is at the Z coordinate of the lowest walkable surface (i.e. navmesh) in the map. The thickness of each level is 700 by default and can be configured in `World Settings` under `Terrain Level Z Span`. The Z of the lowest and highest levels can be clamped by setting `Terrain Level Z Min/Max` in the same section.

Make sure that all terrain that units and structures should be able to exist on is set to Collision Preset `Landscape` and has `Used for Navigation` enabled. Normally Tempest Rising maps have an actual `Landscape` actor as well, but that's not strictly necessary. The template maps do include a `Landscape` actor that already has its collision preset set to `Landscape`.

See also the Vertical limitations subsection in the Navigation section.

Navigation

All units and structures in Tempest exist and move on a navmesh. There are 5 navmeshes in any map:

- `Infantry`: Used for infantry units.
- `Medium`, `Large`: Used for ground vehicles.
- `Square`: Square grid that's used for structure placement.
- `Air`: Used for airborne units.

The navmeshes are usually generated automatically or when building the level or paths, but if you accidentally click `Build All` or delete the navmesh actors, only one might be left on the map. You can check for this by enabling `Show Navigation` in the editor viewport (press **P** and **Alt+N** to cycle between meshes). If the debug text that shows up in the viewport says `NavData count: 5` and you see the navmeshes covering the ground, then you're good. If not, click `Force Rebuild Navigation` in the `Tempest` menu and it should build the missing meshes.

The navmeshes will **only** be built on top of actors that have the `Landscape` collision preset. Even for air units.

Vertical Limitations

Note that all navigation in Tempest Rising is projected and generated from the top-down (vertically). It is not possible to have a scenario where there is a bridge going over a valley and units are able to both use the bridge but also go under it. If the bridge has collision, then it will take precedence over the collision under it when the navigation mesh is generated. Conversely, if the bridge doesn't have collision and is purely for decoration, then units can navigate under it, as the navigation mesh will be generated under the bridge, but not on it.

Modifying the Navmesh

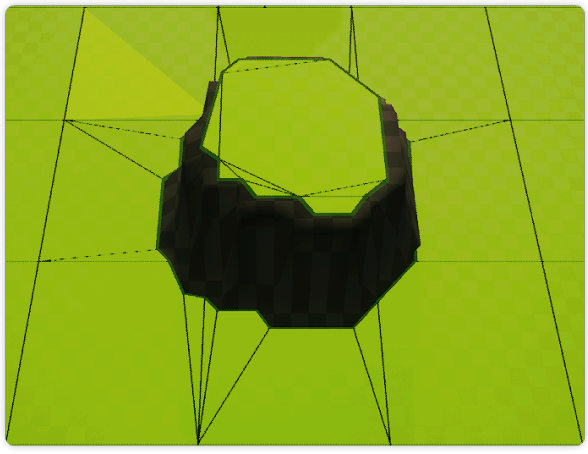
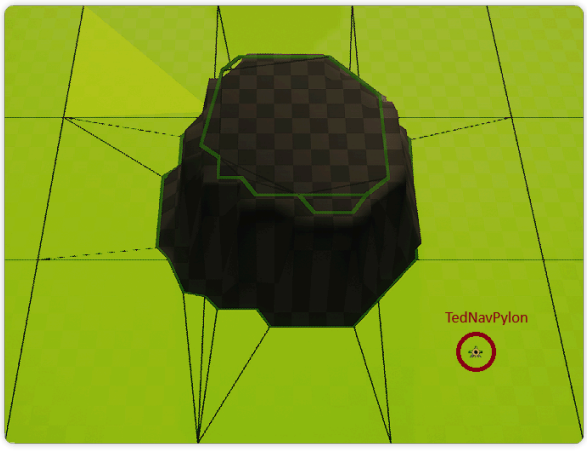
You can alter generated navmeshes by placing `NavModifierVolume` actors. Drag and drop them into the level from the *Place Actors* panel and adjust their scale so that they cover the area you want to tweak the navmesh for. In the *Details* panel for the actor set the `Area Class` parameter to one of the following:

- `NavArea_Default`: Generate default navigation mesh.
- `NavArea_Null`: Prevent any navigation meshes from being generated here. Units and structures cannot exist in this area.
- `NavArea_Obstacle`: Units will avoid this area if possible.
- `TedNavArea_NoMines`: Mines cannot be placed here.
- `TedNavArea_NoStructures`: Structures cannot be placed here.
- `TedNavArea_NoGroundUnits`: Ground units (infantry and vehicles, etc.) can't walk in this area.
- `TedNavArea_NoAirUnits`: Air units (aircraft except airstrike planes) can't move over this area.

Make sure that `Can Ever Affect Navigation` is **disabled** on these.

Navmesh Islands

In some scenarios you will end up with situations where you have navigation mesh in a deep hole or on top of a plateau that is inaccessible for units because it's disconnected from the rest of the navmesh. In those cases you might want to get rid of the navigation mesh in those areas to prevent issues and confusion. To do that, you can place a `TedNavPylon` (found in the *Place Actors* panel) somewhere in the accessible gameplay area surrounding the disconnected island and rebuild navigation.

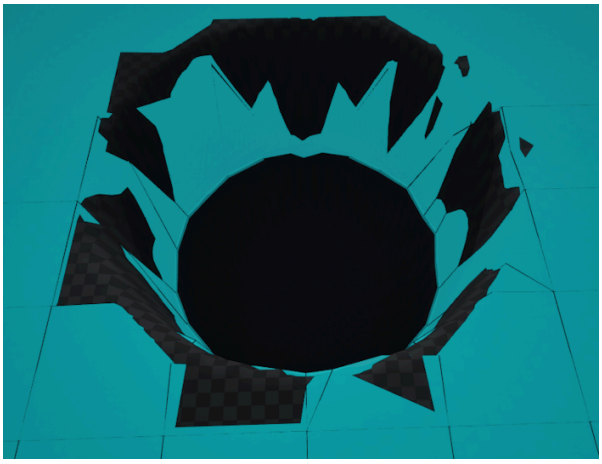
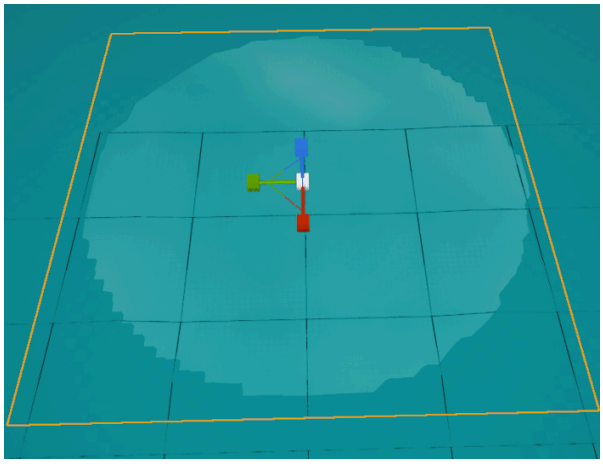
	
Without a `TedNavPylon`	With a `TedNavPylon`

Notice how the navmesh island shows only the outline after placing the pylon. This means that, while it is still generated, it is functionally disabled and will not be used by units in game. During navmesh build the editor will mark all areas that are reachable (i.e. directly connected by navmesh) from any `TedNavPylon`s as valid navmesh and all other areas as invalid, which will disable any isolated islands.

Air Unit Navigation

Air units (aircraft other than airstrike planes, etc.) also use navmeshes for navigation. Moreover, their navmesh is also generated on the ground, they just fly some distance above it. This means that they will closely follow the curve of the ground, at least by default.

If you want them to fly over pits and rivers instead of dipping into them, you can place invisible planes over them. Get a `Plane` from the *Place Actors* panel and set its `Collision Preset` to `Navigation` or `Landscape`, enable `Actor Hidden in Game`, then stretch it over the pit. Make sure that this plane does not cover any ground below itself that you want to keep walkable.

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Without a plane</td><td>With a plane</td></tr>
</table>

You can cover the plane up with a `TedNavArea_NoGroundUnits` if you want to prevent ground units from walking on it.

When following some steep slopes air units will fly diagonally down/up very fast, closely following the slope, which doesn't look very good. If you want to smooth this out, you can select `TedNavMesh-Air` in the Actors list (after building paths at least once) and set these properties on it:

- `Enable Smoothing`: enabled

- `Min Height`: change this if the lowest walkable terrain in your map is lower than Z=0
- `Nav Mesh Resolution Params`:
    - `Low`: Set `Cell Size` to `100`, `Cell Height` to `20`
    - `Default`: Set `Cell Size` to `50`, `Cell Height` to `20`
    - `High`: Set both `Cell Size` and `Cell Height` to `20`

Then rebuild navigation again.

Structures and Units

Pawns, such as structures (neutral buildings and bridges) or units can be placed on the map by adding instances of the Spawner blueprints available in the `Content/Tempest/Blueprints/Spawners` directory. Neutral buildings are also called Conformations. They should belong to the Neutral player, which will make them capturable by other players.

DO NOT place the actual Blueprint Actors for structures on the map, as they will not work. Only use Spawners.

All Spawners have the following common parameters:

- `Initial Upgrade Flags`: Set these to automatically upgrade the spawned pawns.
- `Pawn Owner`: Which player will own the spawned pawns. Use `Neutral` for neutral structures.
- `Activator Is Owner`: If enabled, ignores the `Pawn Owner` property and instead assigns the spawned pawns to whichever player was passed into the `Spawn` function. Pawns spawned on startup will still be assigned to `Pawn Owner`.
- `Auto Spawn`: Whether this Spawner should activate on map start (i.e. `BeginPlay`). Defaults to enabled.
    - If disabled, you can call the `Spawn` function of the Spawner from the Level Blueprint to trigger it.
    - Pass the ID of the player that activated the spawner as the argument if you want the spawned pawns to be assigned to that player and `Activator Is Owner` is enabled, otherwise pass Neutral or None.
- `OnSpawnedPawnEvent`: Blueprint event that triggers (in Skirmish or on the server) every time the Spawner spawns something.

Trying to assign pawns to a non-existent or defeated player will fail to spawn the pawns, unless that player is Neutral or Hostile. This goes for both the `Pawn Owner` property and the `Spawn` function argument. Spawn will, however, succeed even if there's no valid navmesh under the pawn, in which case it will be stuck in place.

Neutral Buildings

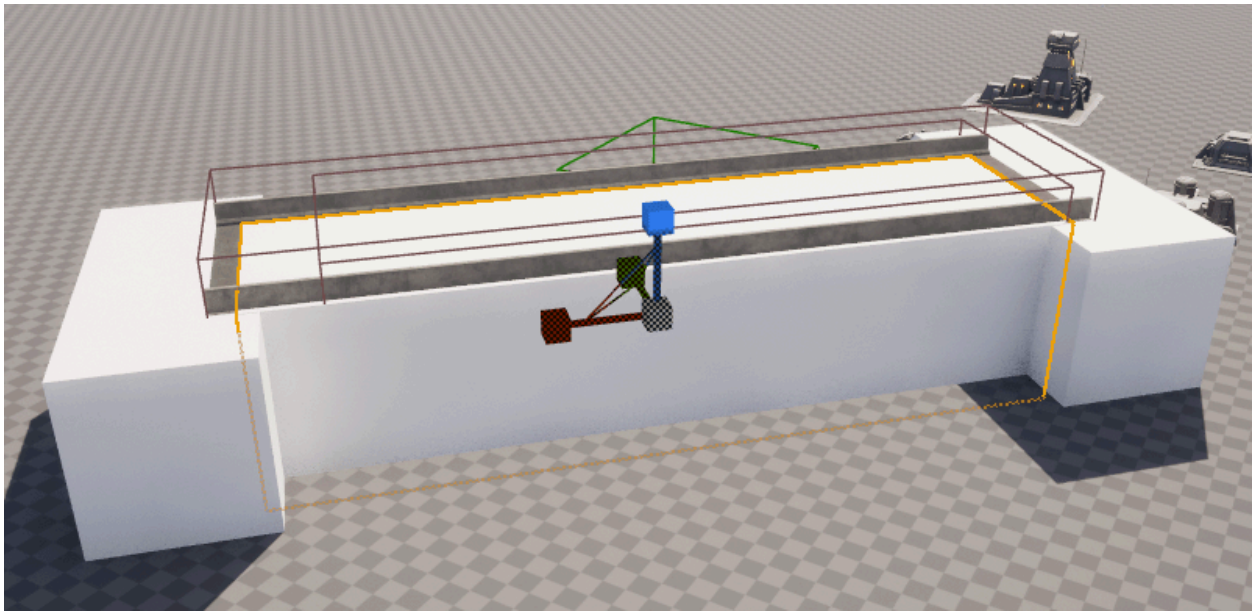
The Spawners in the `Content/Tempest/Blueprints/Spawners/Conformations` directory spawn neutral structures, such as bunkers and bridges. They have the following properties, in addition to the common properties listed above:

- `Structure Alignment`: Which grid-aligned direction the structure will face when placed. Note that you **should not** rotate them using the regular Rotation property, and instead you should use this property.

Repairable bridges

You can add bridges to your map that can be destroyed and repaired by an Engineer. They are spawned using the `Bridge` Spawner blueprints in the `Content/Tempest/Blueprints/Spawners/Conformations` directory.

Note that a bridge by itself is **not** considered walkable ground. It serves just as a visual and to mark the points where the Engineers will go to repair it. You should place something like a Plane or a Cube marked `Hidden In Game` with the `Landscape` collision preset under the bridge:



When the bridge is broken, it will generate a navigation blocker that will prevent units from traversing it. Repairing the bridge will remove the blocker.

The bridge itself does not need to be set to the `Landscape` collision preset.

Bridge Spawners have the following properties in addition to the Structure properties described above:

- `Start Destroyed`: Whether the bridge will be destroyed at the start of the map.
- `ActivationPoint1/2`: Points at the ends of the bridge where Engineers will go to to repair it. They are represented as little diamond shapes in the Editor viewport that you can select and drag around when the bridge is selected:

Destructibles

`Content/Tempest/Blueprints/Spawners/Destructibles` contains a few Spawner Blueprints for destructible objects, such as explosive barrels. These don't respawn in Skirmish or Multiplayer, so their use is a bit limited, unless you hook up a Spawner to respawn them manually using the Level Blueprint.

Units

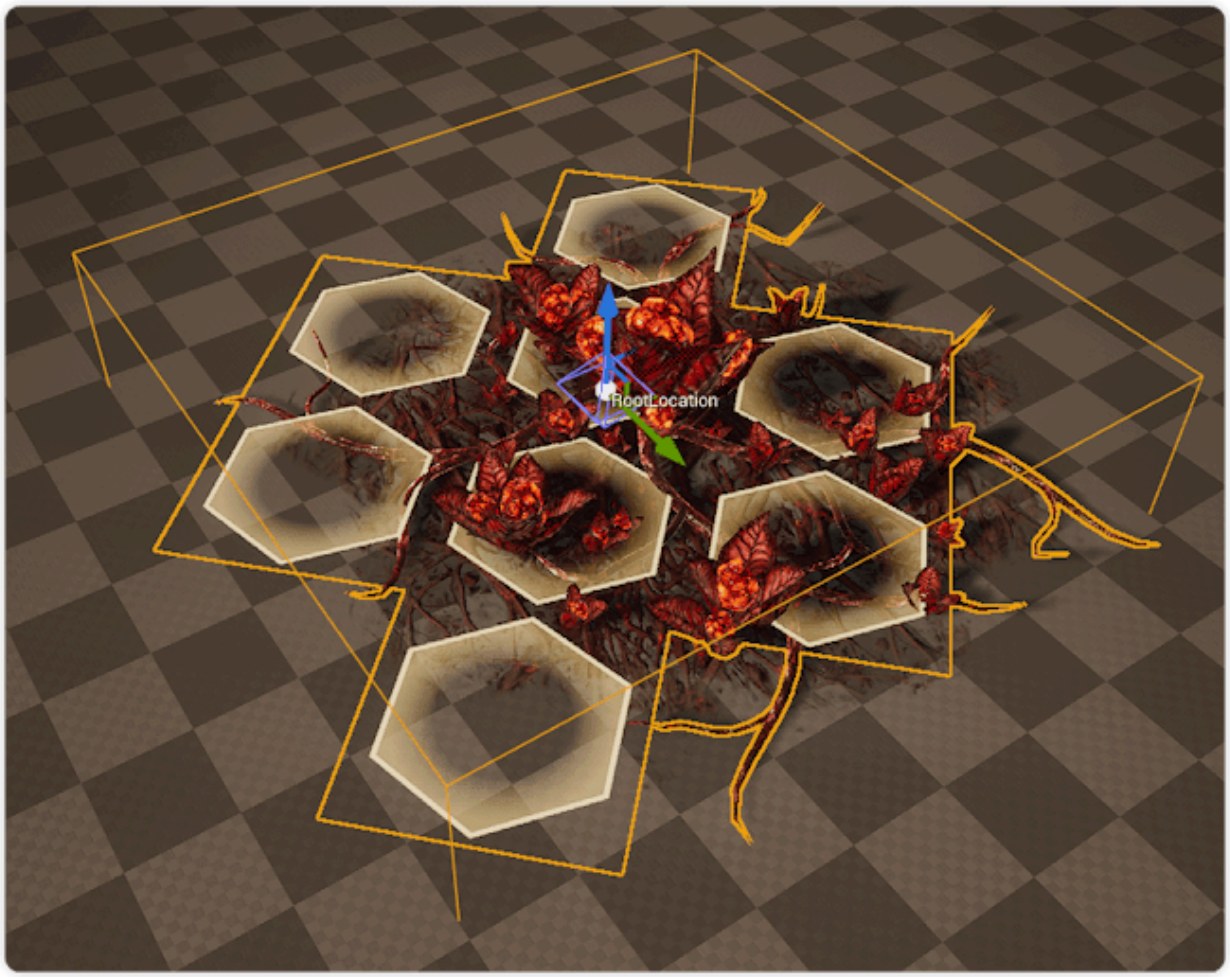
You can use `BP_Spawner_Unit` to spawn units or groups of units. They can be assigned to `Neutral`, `Hostile` or one of the used player slots.

This blueprint has the following additional properties:

- `Pawn Definition Id`: The `GameplayTag` that identifies the class of the unit you want to spawn. Make sure this is not empty.
- `Pawn Count`: How many units to spawn whenever the Spawner activates (either at map start or when `Spawn` is called).
- `OnSpawnedAllPawnsEvent`: Blueprint event that activates when *all* `Pawn Count` units have been spawned.

Tempest

Tempest fields (i.e. the main resource in the game) are placed on the map using the `TempestFieldVolume` actor, which can be found in the *Place Actors* panel. These will automatically generate a bunch of Tempest vines within the volume when placed or moved. Each cell of the nav grid covered by this volume will have its own resource value. You can see them as hexagonal cells in the Editor viewport:

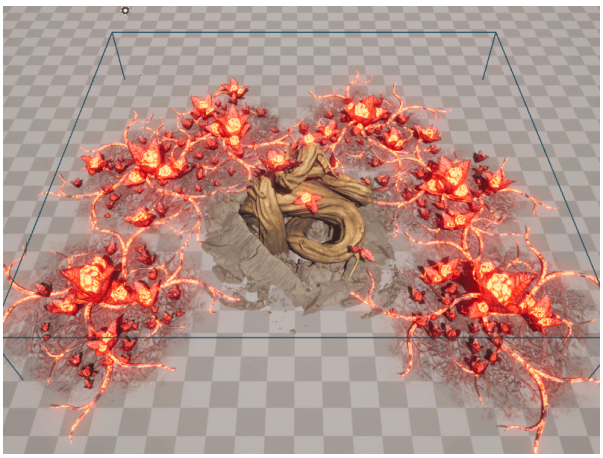
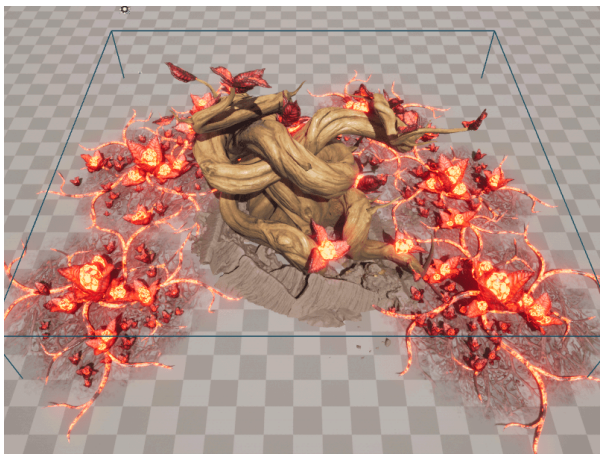


The default size of the volume is too small to generate even a single cell, so make sure to scale it up. You can use the *Brush Editing* mode for this, selectable in the *Mode* dropdown in the Editor viewport, or with **SHIFT + 8**. You can also use the Actor scale, but changing the brush size is preferred. If Tempest doesn't appear on the field even after scaling it up, click *Reset Volume* in the Tempest Field Volume's *Details* panel.

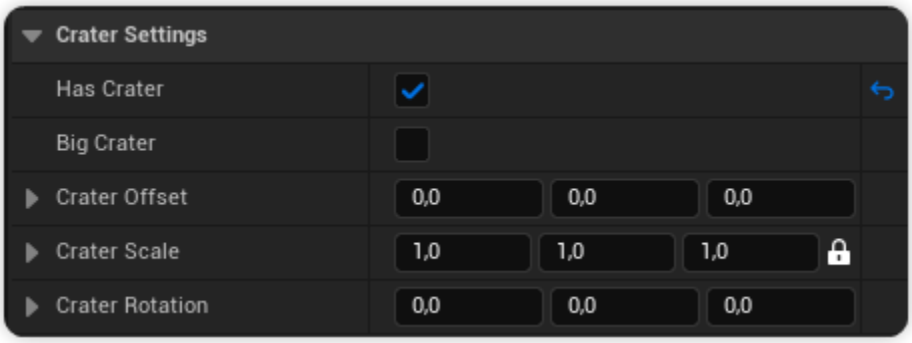
You can change the `Root Location` of the Volume by selecting and moving the blue diamond shape in the Editor viewport while the volume is selected. This specifies the location from which Tempest vines spread horizontally.

Craters

Craters are the source of Tempest and serve as spawn points for additional Tempest if a field is depleted. This means that fields without a crater can't regrow and are single-use. You have the option to have no crater, a regular crater or a big crater for each `TempestFieldVolume`:

	
Regular crater	Big crater

You can enable/disable the crater, select its size and adjust some other settings in the *Details* panel of the Volume:



The crater will spawn at the `Root Location` described above. Make sure that the `Root Location` is somewhere away from the borders of the Volume, so that there is enough space for the crater to spawn. Note that no navigation mesh will be generated on the crater. If you adjust offset, scale or rotation, check the resulting navigation mesh to ensure that the crater didn't cover something important. After moving the crater, it might be required to click `Reset Volume` on the Volume you adjusted to regenerate affected navigation.

Do not overlap craters with `TedNavArea_NoGroundUnits`. You can place them on top of the other parts of the Volume, though. If you do, make sure that you click `Reset Volume` on the affected Volumes to regenerate affected navigation.

Other Properties and Notes

If two Tempest fields are overlapping, you can change the `Build Priority` properties on them to specify which field should take precedence for shared tiles. Higher priority fields take precedence over lower priority fields.

You can adjust the `Random Seed` on a Volume to make it generate a different looking field.

Clicking `Reset All Volumes` in any Volume's *Details* panel will regenerate all `TempestFieldVolume`s on the level.

If you want a Tempest Field to start without any Tempest on it, set its `Growth Start Stage` property to `Initial`. Remember to also enable the crater, otherwise nothing will grow on the field.

Sky and Weather

Skydome, skylight, and weather settings on the map are controlled by the `BP_DayAndNight` actor. This actor is pre-placed in the Tempest template maps. If you're using the `Empty Map` template, make sure to place this actor on your map, along with a `PostProcessVolume`.

A lot of the properties in this blueprint are deprecated and won't work properly. Due to how UE displays the *Details* panel of cooked blueprints in the Editor, they can't be hidden. You should only use the properties explicitly mentioned in this section.

The same actor also contains the standard Unreal Sky Light Component (`SkyLight`), as well as a Directional Light Component for controlling daylight (`DayLight`), a Post Process Component (`PostProcess`) and a skydome mesh (`SkySphereMesh`). You can adjust these as you see fit.

Weather

You can choose between no weather, snow and rain by setting the `Weather Type` property of the `BP_DayAndNight` actor placed in the map. Note that when you turn on snow or rain you will also have to set `Snow Intensity Preset` or `Rain Intensity Preset` for it to start working.

Falling snow or rain is affected by the wind settings. Set `Wind Type` to something other than `No wind` and adjust the angle and other parameters to your liking if you want to enable wind.

When snow or rain is enabled, in addition to falling raindrops and snowflakes, rain splashes or snow will be generated on the surface of meshes that have materials derived from any of the `MASTER` materials from the `/Tempest/MaterialLibrary` directory, or otherwise support snow/rain.

The following weather properties are usable:

- Wind:
  - `Wind Type`: Strength of the wind. Set to `No wind` to disable wind.
  - `Wind Base Angle`: Direction of the wind (angle in degrees).
- Snow/Rain:
  - `Enable Rain - Snow`: Enable rain and snow system.
  - `Weather Type`: Select whether to enable snow, rain or dust.
  - `Snow Intensity Preset`: Select snow strength, if `Weather Type` is set to `Snow`.
  - `Rain Intensity Preset`: Select rain strength, if `Weather Type` is set to `Rain`.
  - `Snow Base Amount`: Amount of snow on the ground (i.e. meshes with materials that support snow).
  - `Set Weather Random Seed`: If checked, will allow you to set your own random seed for the weather generation.
  - `Weather Random Seed`: Current random seed for the weather generation.

If you want to add your own materials with snow/rain support, make sure to use one of the included `MASTER` materials as the base (e.g. **`M_MASTER`**`_Environment` or `M_MASTER_Props`).

Sun and Sky

You can adjust the angle and size of the sun in the sky. The angle will affect the skylight and the color of the skydome by default.

The following sky-related properties are usable:

- `Sun Brightness`: Sets the size of the sun. Does not affect lighting.
- `Stars Brightness`: Sets the brightness of the stars in the sky. Not that you will ever see them under normal circumstances.
- `Sun Horizontal/Vertical/Tilt Angle`: Sets the position of the sun in the sky. Affects skylight direction and sky color (if enabled).
- `Colors Determined By Sun Position`: If enabled, sky color will be modified according to the sun's position.
- `Zenith/Horizon/Cloud/Overall Color`: Allows you to explicitly set the color of the sky, if the above setting is disabled.
- `Night Light`: Set to enable night time lighting mode.
- `Cloud Opacity`: Sets the opacity/alpha of the clouds in the sky.
- `Cloud Speed`: Sets the speed at which the clouds move.

Lighting

Most of the lighting in Tempest maps is provided by the skylight/daylight attached to the `BP_DayAndNight` actor. You can use regular Spot and Point Lights if you need to. Keep in mind that all lighting in official Tempest maps is dynamic, so you might have to use Movable or Stationary lights for some materials to receive light correctly.

Foliage

If you're going to use any foliage in your map, you can place it using the `Foliage` mode in the Editor as normal.

Tempest Rising maps additionally should have a `FoliageInteractionManager` actor in them if there is any foliage on them. This actor enables interactions between pawns and foliage (e.g. vehicles trampling plants, etc.). After doing any foliage work, make sure to select `Build Foliage Clusters` in the `Tempest` menu before saving the map. This will add a `FoliageInteractionManager` to the level if it doesn't already exist and generate the data it needs to function properly.

See the Set Dressing Guide for more information.

Minimap

The minimap usually displays a texture with a top-down capture of the map's terrain. This texture can be generated automatically by selecting one of the `Generate Minimap Texture` options in the `Tempest` menu at the top of the Editor window. It will be saved into the same directory as the map and automatically selected in the map's MapInfo.

Usually a 512x512 texture is enough, but for large enough maps you might want to use the bigger options (1024x1024 or 2048x2048).

By default the minimap generator will automatically hide all Pawns (i.e. units and buildings) and Player Starts when capturing the minimap texture. You might want to hide additional actor classes. To do that, manually add a `BP_Minimap_Capturer` actor (located in `/TempestEditor/Blueprints/`) to the map. You can then adjust the following properties in its *Details* panel:

- `Classes To Hide`: List of actor classes to hide when rendering the minimap.
- `Classes To Show`: List of actor classes to explicitly not hide when rendering the minimap. Can be used to override the above setting.

When you're done setting the actor up, you can use the same `Generate Minimap Texture` menu options as mentioned above. This will use the pre-placed actor. After you're done generating the texture you can remove the actor from the level. Make sure to save the level and the map's MapInfo asset (or just hit `Save All`).

Trigger Volumes

Stock Unreal `TriggerVolume`s **can't** detect Tempest Rising units, so you **shouldn't** use them. Use `TedTriggerVolume`s instead, which work the same way as the regular volumes, but can be used with Tempest units.

Custom Assets and Blueprints

In general, any custom assets you have added to use in your map should be contained within the mod plugin's `Content` directory. You can add any custom assets supported by Unreal Engine, as well as custom blueprints.

Audio

If you want your custom audio assets to obey the in-game volume settings, assign them to the corresponding Sound Classes in the `Tempest/Audio/Classes` directory.

You cannot add custom music to the Jukebox, but you can play your own music via standard Unreal Engine methods. If you decide to do so, you can disable the Jukebox music with the `JukeboxSetPaused` or `JukeboxFadeOut` Blueprint nodes.

Materials

Custom Materials should inherit from one of the `MASTER` materials provided in the `Tempest/MaterialLibrary` directory. This way they should respond properly to weather and other Tempest Rising effects.

Gameplay Effects and Attributes

Tempest Rising uses the Gameplay Ability System to implement unit and structure attributes (such as health, speed, etc.) and commands.

All Tempest Rising-specific attributes are contained within the `TedAttributeSet` attribute set. You can create custom Gameplay Effects to modify the following pawn attributes:

- `Health`, `HealthMax`: Current health and max health.
- `AmmoRechargeTimeMultiplier`: Multiplier for the time it takes to restock ammo. Applies only if the weapon can recharge automatically.
- `AmmoCapacityMultiplier`: Multiplier for max ammo count.
- `GunResistance`: Resistance to gun damage.

- `FireResistance`: Resistance to fire damage.
- `CannonResistance`: Resistance to cannon damage.
- `RocketResistance`: Resistance to rocket damage.
- `SniperResistance`: Resistance to sniper damage.
- `ExplosiveResistance`: Resistance to explosive damage.
- `AddResistance`: Additional resistance added on top of damage specific resistance
- `DebuffDurationMod`: Multiplier for the durations of all debuffs.
- `DebuffBuildupSpeedMod`: Multiplier for the speed at which debuffs build up.
- `HealEfficiencyMod`: Multiplier for all healing.
- `RepairEfficiencyMod`: Multiplier for repair speed.
- `StealthRevealRadius`: Stealth reveal radius.
- `FoWRevealRadius`: Fog of war reveal radius.
- `FoWVisionRadius`: Fog of war vision radius.
- `FoWRadiusAttributesMultiplier`: Multiplier for FoWRevealRadius and FoWVisionRadius.
- `FoWDisableHeightCheck`: If set, this unit will see all terrain Z levels regardless of its current position.
- `BuildRadius`: Build radius.
- `Damage`: Base damage.
- `AttackSpeed`: Base delay between attacks.
- `AttackSpeedMod`: Multiplier for AttackSpeed. Note that AttackSpeed actually specifies the delay between attacks, so if you want to make the attack speed higher, you need to set this to a number smaller than 1.
- `AttackRange`: Attack range.
- `AttackGraceRange`: Attack grace range.
- `MovementSpeedMod`: Movement speed multiplier.
- `MovementSpeed`: Base movement speed.
- `ProductionTimeMultiplier`: Multiplier for production time, if this is a factory.
- `ProductionCostMultiplier`: Multiplier for all production costs, if this is a factory.
- `ResourceProductionMultiplier`: Multiplier for all resource production rates, if this is a structure that produces resources.
- `Power`: Base power. Negative amounts consume power, positive amounts produce power.

Each `TedPawn` has an `AbilitySystemComponent`, which you can retrieve using the `GetAbilitySystemComponent` blueprint node.

Refer to the Gameplay Ability System Documentation for more on how Gameplay Effects work. `ZooExampleMap` also contains an example of a custom gameplay effect that changes units' movement speed multiplier (`Blueprints/GameplayEffects/GE_Example`).

NOTE: Functions that apply Gameplay Effects or execute Commands should only be called on the server (or in single player). Do not call them if you don't have network authority.

Using Tempest Rising Assets

The `MapEditorProject` project comes with some (but not all) assets from the game in cooked form. They are located in the `Content/Tempest` directory. You can use them in your map without having to copy and paste them into your mod, because their full paths are the same in both the Editor project and the actual game. This means that references to them remain valid even if they aren't included in the cooked mod content.

The same rule will apply to any other assets you put in that directory. This means that you can, for example, create placeholders for game assets that aren't part of the Editor project already. If you then use these placeholders in your map, they will be replaced by the actual asset when the map is loaded in the game, as long as their paths match. You should not put any custom assets into the `Tempest` directory, as the base game does not have them.

Cooking, Packaging and Distributing the Map

Once you're done working on the current iteration of your map, make sure to do the following:

1. `Build -> Build Paths`.
2. `Build -> Build Lighting` (if you're using static lights)
3. `Tempest -> Build Foliage Clusters` (if you're using any foliage)
4. Save all changed levels and the MapInfo (or just hit `Save All`).

Note: If you use `Build All Levels` or `Build Geometry`, the navigation mesh will get nuked and you won't be able to rebuild it using `Build Paths`. You will need to use the `Force Rebuild All Navigation` option from the `Tempest` menu if that happens.

Error Checking

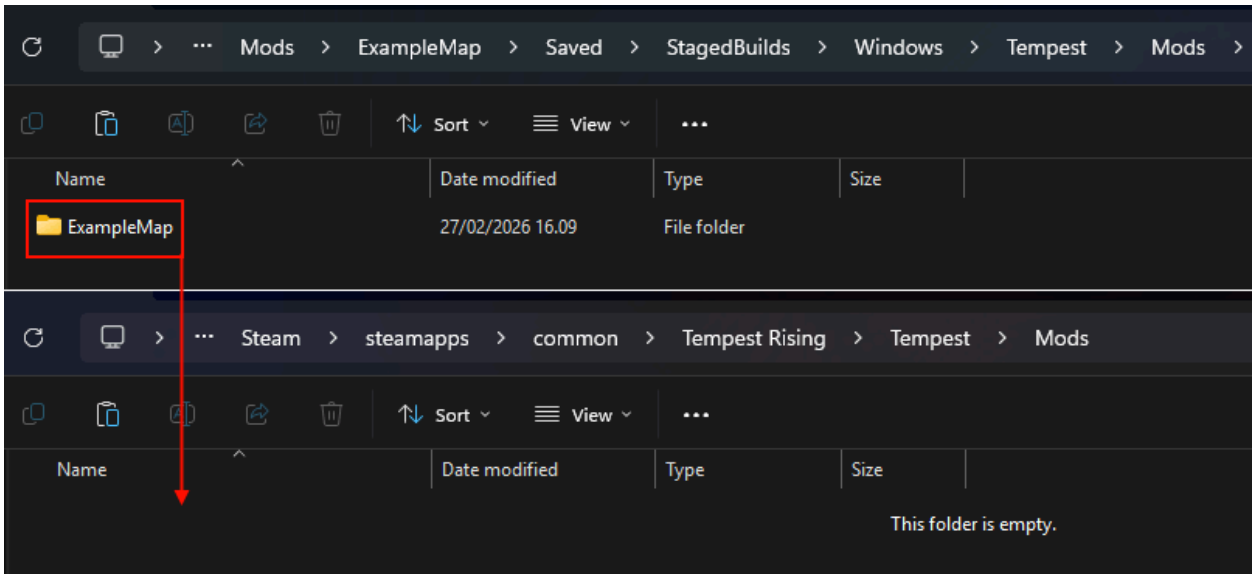
Whenever you build or package your map, it will be validated in an attempt to catch common mistakes. If there are any, you'll see a window with the error list. These errors will usually have information on how to fix them, so ensure you do that before saving the map again.

Packaging

To package the map use the `Package Current Mod` option in the `Tempest` menu. You can also use the `Package a Mod...` option, which will prompt you to select which mod you want to package first.

All assets and maps contained within the mod that your map is part of will be cooked, packaged and copied into `MapEditorProject/Mods/<ModName>/Saved/StagedBuilds/Windows/Tempest/Mods/<ModName>`. This directory will automatically open in Explorer once Unreal is done packaging the mod.

To distribute your mod without the use of Steam Workshop just package the mod and copy the resulting `<ModName>` directory to `Tempest Rising/Tempest/Mods/`:

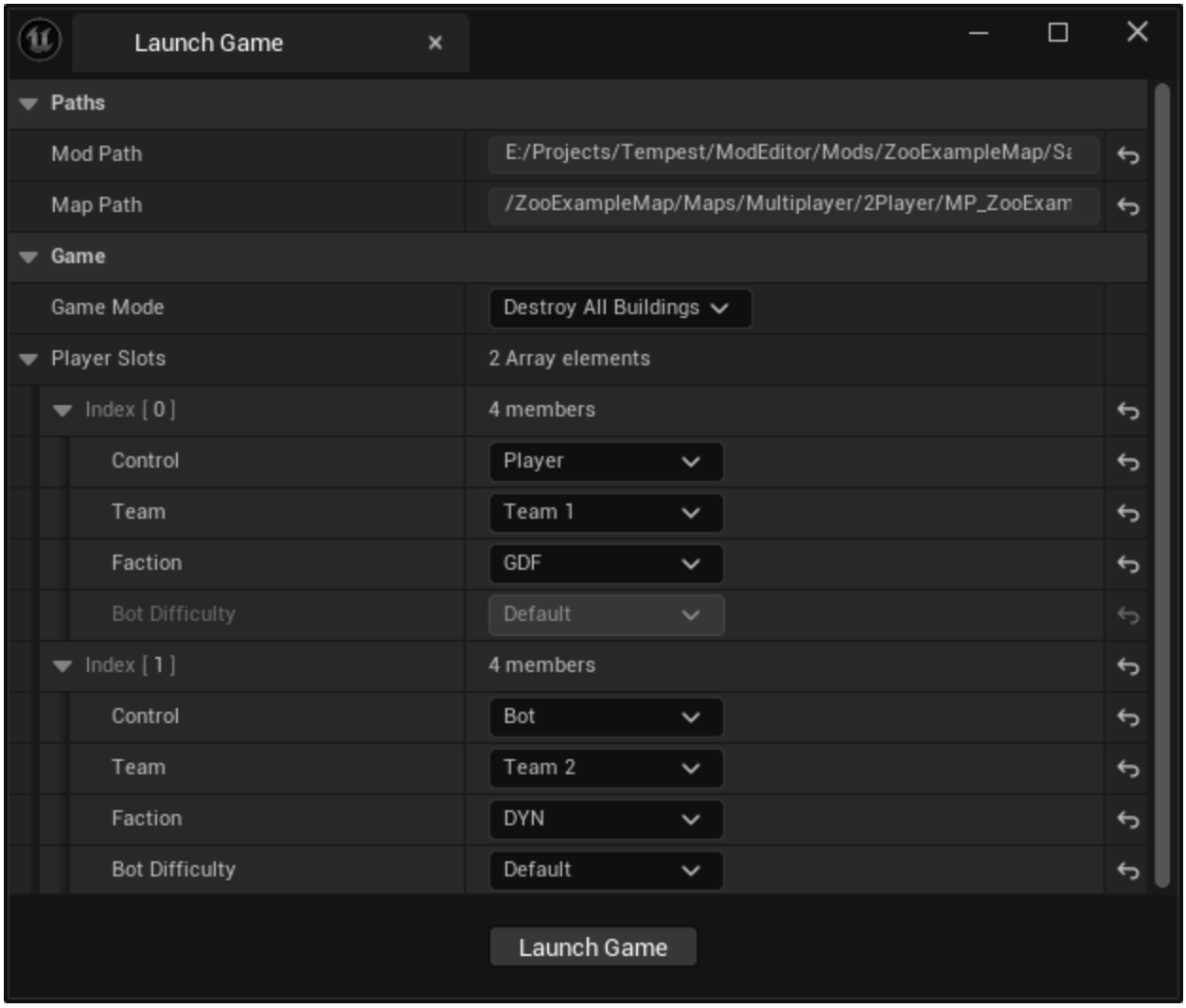


The maps contained in that mod will appear in Skirmish and Custom Game map lists next time you launch the game. Note that only maps distributed through Steam Workshop can be downloaded automatically in Multiplayer.

Testing Your Map

The map editor project does not actually include any gameplay-related code, so the Play In Editor mode does not let you play your map, but it does act as a preview mode that lets you see how the map will look in-game. There you have access to the same camera controls as in the game.

You can test your map in game using the `Test Current Level` option in the `Tempest` menu. This will cook and package the mod you are currently working on in the Editor and then run it in the game in Skirmish mode. You can use the `Test Settings...` menu option to change the game mode and the amount of bots the game will run with next time you use the Test option:



By default, the editor will try to run the game through Steam. If that does not work for whatever reason, you can specify the path to the game's executable (`Tempest.exe`) in `Editor Preferences -> Tempest Rising Settings`.

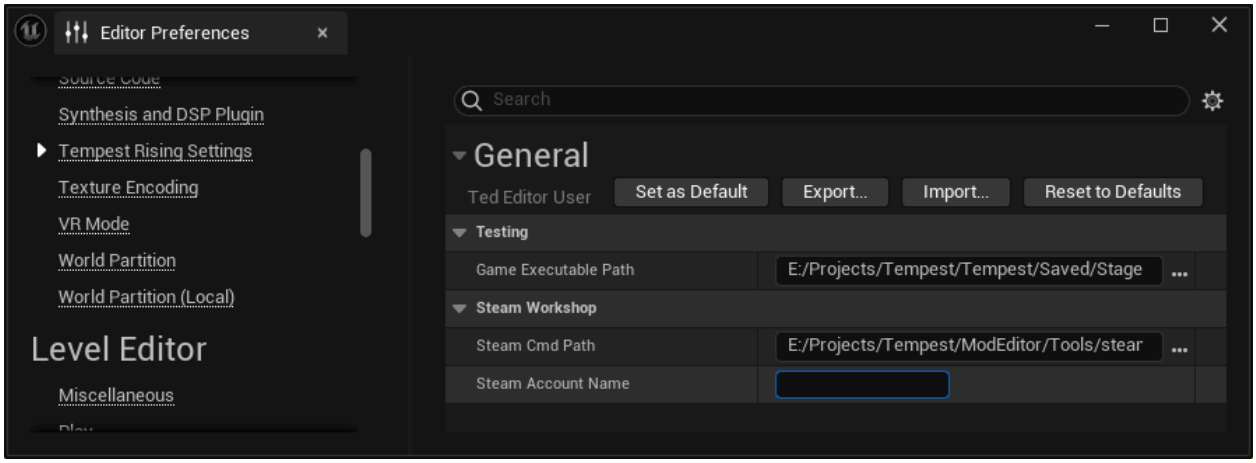
The editor will minimize itself before launching the game. You can only run one instance of the game at a time.

Note that debug Blueprint nodes like `Print String` and `Log String` **will not work** in the game. You can use the `Send Chat Message` node for printing debug information instead.

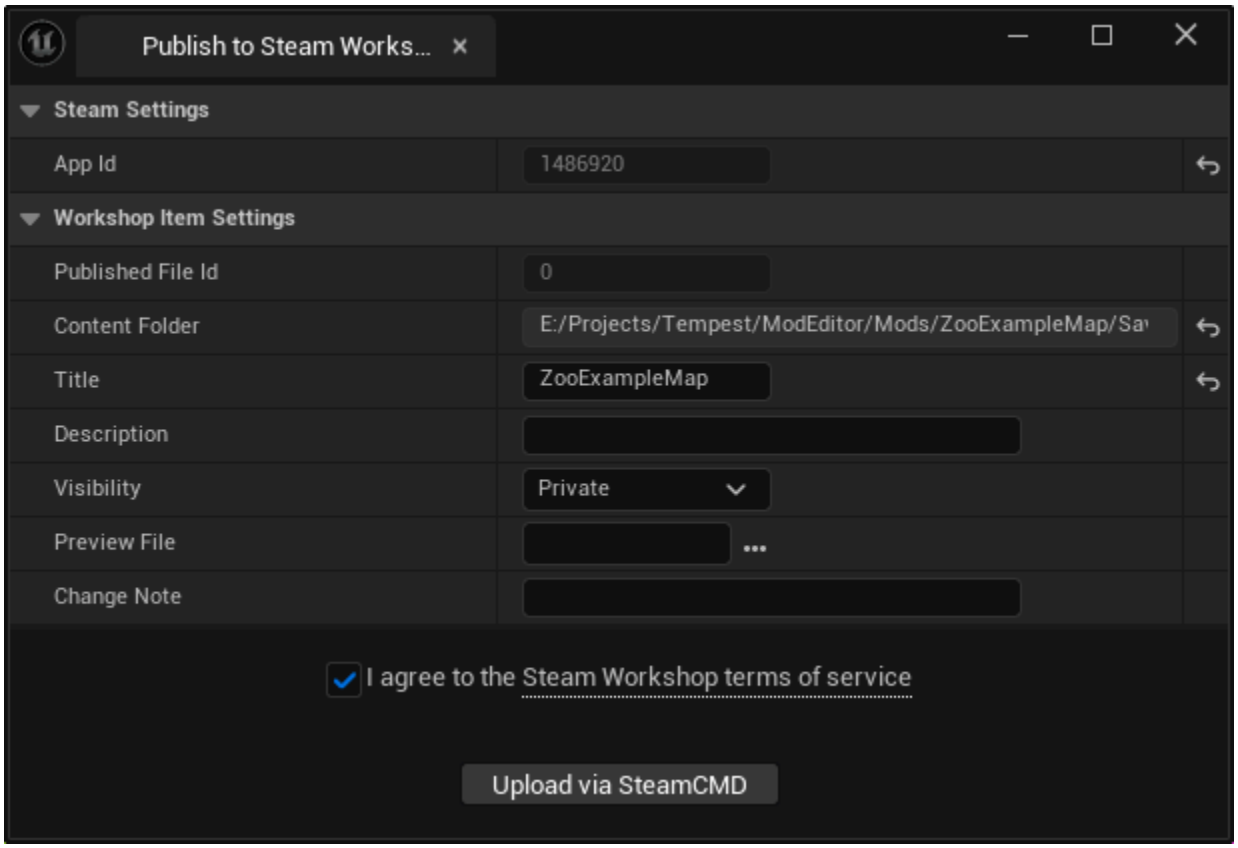
Uploading to Steam Workshop

You can use the Map Editor to publish your maps to Steam Workshop. You will need a valid Steam account and SteamCMD installed.

First, set your Steam account name (**NOTE:** this is **not** your nickname but the name of your account, you can see it in `Account Details`) and the path to `steamcmd.exe` in `Editor Preferences -> Tempest Rising Settings`:



Once your map is ready for release, use the `Publish Current Mod...` option in the `Tempest` menu:



When publishing the map for the first time, you will be able to change the following options:

- `Title`: The title of the map will be shown on Steam Workshop. This can be different from both the map display name and the name of the actual asset.
- `Description`: The description of the map on Steam Workshop.
- `Visibility`: Whether the map will be public on the Workshop right away or not. This should probably be set to `Private` first and then changed in the Steam Workshop UI once you can confirm that everything is set up correctly.
- `Preview File`: The preview image that will be shown on the Workshop for your mod. This can be any JPEG or PNG file.
    - This can also be changed on subsequent uploads.
- `Change Note`: The change note that will display in the changelist for the first version of your map on the Workshop.

Note that all these settings except the preview image can be later changed on the Steam Workshop page for your map. You can change the preview image when updating the mod from the uploader (see below).

Single (`'`) and double (`"`) quotation marks are not allowed in the string fields in this window, but you can later use any character when editing them on the Steam Workshop page.

When you've set everything up, ensure that you have read and agreed to the Steam Workshop terms of service by clicking the corresponding link in the `Publish to Steam Workshop` window and checking the checkbox next to it, then click the `Upload via SteamCMD` button. This will cook the map and try to upload it using SteamCMD. Note that the first time you upload the map, SteamCMD will ask you to log in by inputting your password and Steam Guard code into the window that will pop up after hitting the `Upload` button.

After the upload is finished, the editor will open the map's Workshop page in Steam. There you can review and change all the details and upload additional screenshots.

Updating an Already Published Map

To upload a new version of the map, just open the `Publish Current Mod...` dialog again and hit `Upload via SteamCMD`. Note that when you upload updates you will be unable to change any Workshop item properties in the Map Editor other than `Change Note`, which can be used to set the change note for the new version. All the other settings can still be changed on the Steam Workshop page of your map if needed.

Uploading Maps Manually using SteamCMD

The Map Editor automatically generates a VDF file for SteamCMD called `Workshop.vdf` in the `Mods/<ModName>/Resources/` directory of every mod. You can feed this script into SteamCMD yourself to upload the map to Workshop by running it like so:

```
None
steamcmd.exe +login ACCOUNT_NAME PASSWORD +workshop_build_item
Path/To/Mods/<ModName>/Resources/Workshop.vdf +quit
```

This is exactly what the Map Editor does under the hood, except it does not specify the password.