ONWARD CUSTOM CONTENT GUIDE

1/12/2021 - Patch 1.8.7 - Guide/Package Version: 2.0.8

DISCLAIMER:

- All information in this guide is up to date as of **Onward patch 1.8.7** 01/12/2021 and is subject to change. Changes and updates will be noted in the *'Change log"* section below.
- Updates to the custom content package files may cause inconsistencies with this guide
- If you encounter any issues with this guide or the custom content package files, please contact us using the **Onward Discord channel.**

CHANGE LOG:

10/30/2019 Initial version 1.0 released. Happy content building! -DownpourDevs
11/6/2019 Version 1.1 released. Small bug fixes included. New *Set Physic Material* option added: *AI See Through* (Section 3)
11/7/2019 Version 1.1.2 released. Texture size of thumbnails increased to 512 x 256 (Required), additional performance tips.
11/20/2019 Version 1.1.3 released. Area Reverb zones added. Streamlined Escort setup.
1/12/2019 Version 1.1.4 released. Fixed a dev-side issue.
2/28/2020 Version 1.1.5 released. Added Shader Compliance tool to assist users in preparing for Patch 1.8.
Look for NEW. Upgrading for Unity 2019 section added. Prominent EULA link added.
4/10/2020 Clarification of Unity version added to guide.
10/13/2020 Updated to Guide/Package version 2.0.6. Look for NEW: UPDATING YOUR MAP FOR V1.8.5 pg. 50
1/12/2021 Updated to Guide/Package version 2.0.8. NEW: Custom audio added back in. See *OnwardCustomAudioSource* in "Additional Gameplay Components" on page 29.

REQUIREMENTS:

- Unity 2019.4.8f1 & 60-100GB free storage
- **Onward Custom Content Package**

RULES FOR CONTENT SUBMISSION (READ ME):

NEW: Supported Shaders:

Your map materials must use only the PreProcessing (prefixed; PP_xyz..) shaders included with the Onward Custom Content SDK v2.0.6+.
If your map currently does not contain ONLY these shaders, that's alright. **You can use our Shader Conversion steps and tools to assist you in converting your shaders.** *It is a simple process that can be carried out in a few minutes to an hour, depending on the complexity of your custom content scene.*

You may **NOT** use any copyrighted / unlicensed assets in your map. You may **NOT** copy existing assets from any copyrighted works (*IE directly-porting de_dust2, or Blood Gulch etc.*). Recreations of maps from other games and universes must be carried out **without** infringing source material. This means that re-creating a given map's layouts is fair game, but **copying** it's **textures/models/sounds or any form of asset used in it's creation** is **NOT**. **Using copyrighted / unlicensed assets in your map or infringing on copyrighted material in any way will result in your submission being DENIED**.

You **MUST** *lightmap* your map. Use this lightmapping document as a guide when you are baking lightmaps in your map. **Not lightmapping your map will result in your submission being DENIED.**

You may **NOT** place vulgar, inappropriate, racist, obscene or otherwise unpalatable or illicit content in your map. **Implementing illicit/vulgar content in audio, word or pictorial form will result in your submission being DENIED**.

You may **NOT** use the full name, likeness or representation of any person (alive or dead) in your scenes. Including **the likeness of anyone inside your scene will result in your submission being DENIED**. Non-inappropriate screen names/vague references are allowed.

Your map **MUST** support the FreeRoam game mode and at least one (1) other game mode. **If your map does not support FreeRoam and at least one (1) other game mode your submission will be DENIED**.

You are responsible for the testing and verification of your own map content

submissions as well as curating an acceptable user experience. DO NOT put any content in your map that would be deemed *hazardous or otherwise induce bodily harm to a player*. While using VR, disorientation is physically dangerous. Maps with extremely loud sounds, flashing/strobing visual aspects, nausea-inducing mechanics or otherwise hazardous gameplay will be DENIED.

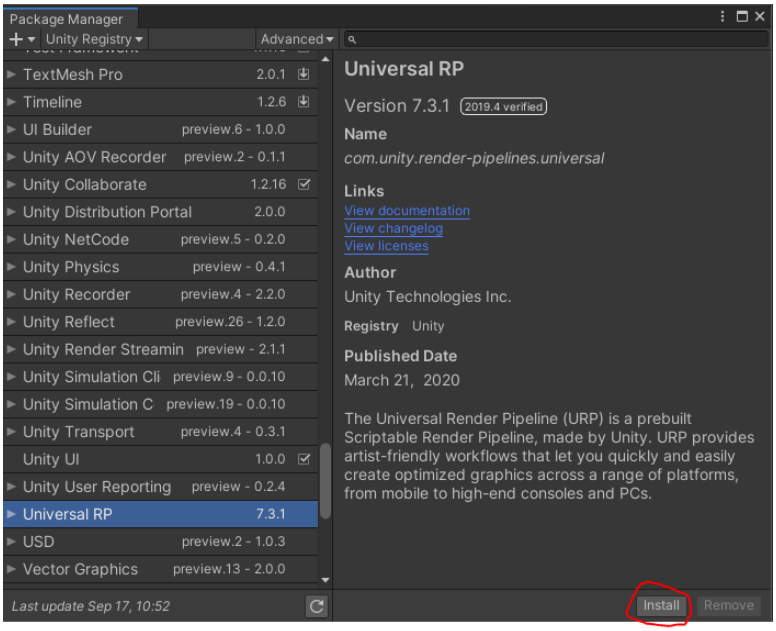
Your content/maps may be removed from our servers at any time post-approval if Downpour Interactive deems it has violated the rules for content submission.

Submission of custom content for play in Onward is a free service we offer to enrich the players, it is not a right. Please follow these rules and you will be on your way to share your creations with thousands of Onward players!

By submitting your map(s) to the Onward Workshop, you agree to the Custom Content End User Agreement terms.

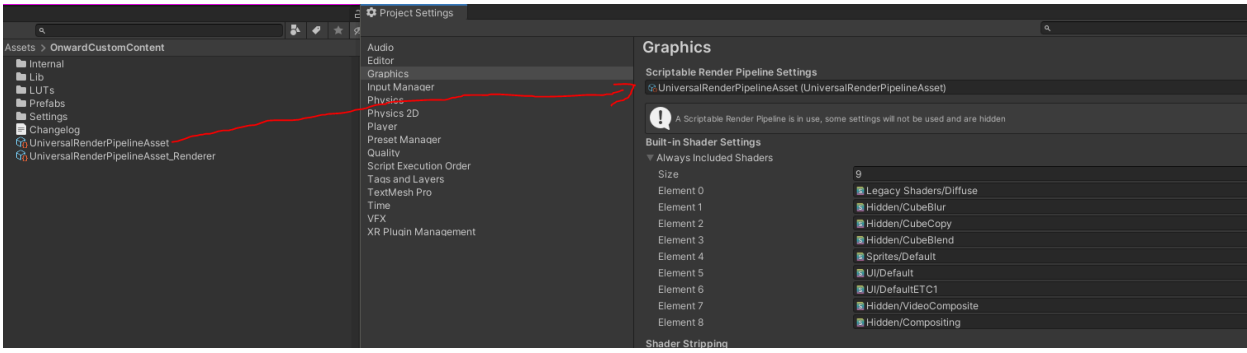
SETUP:

1. Ensure that you have enough disk space to store the custom project and associated data generated during map creation **(>60GB)**
2. Download and install **Unity 2019.4.8f1**
    a. Go to https://unity3d.com/get-unity/download/archive
    b. Select Unity 2019.x
    c. Locate and download **Unity 2019.4.8f1, install Android Build Dependencies when installing.**
    d. Open Unity and install Universal Render Pipeline from the Package Manager.
    Open the package manager with Window>Package Manager and find "Universal RP" in the available packages. Install it.



e. Open your **Project Settings** window under **Edit>Project Settings** and navigate to the **Graphics** tab.

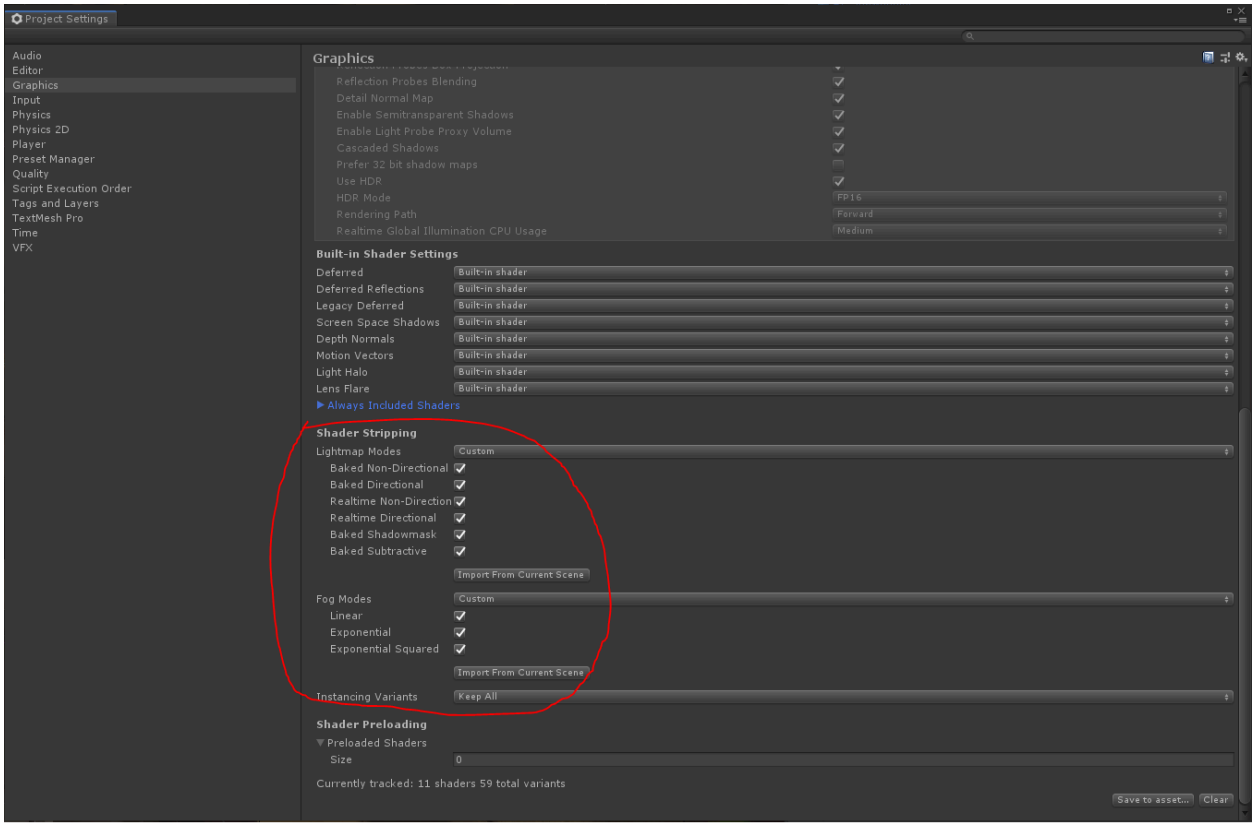
Once there, locate the **UniversalRenderPipelineAsset** file located in the root folder of **Assets>OnwardCustomContent** and drag it into the **Scriptable Render Pipeline Settings** field of the Graphics tab in the project settings window. You may now close the Project Settings window.



3. Download and add the [Onward Custom Content Package] to your Assets folder (drag and drop the package into your Unity window to import it).

BEGINNING YOUR FIRST PROJECT:

1. Locate the unpacked **Custom Content Template Project** in your Assets and open **ExampleScene**. You can use this scene as an **experimentation area and working example** for custom content *game mode* setups in Onward.
2. ➤ **IMPORTANT: Rename your Example Scene and save it as a separate file.** This will avoid any issues if the **ExampleScene** file is updated in the future.
3. ➤ **IMPORTANT: It is recommended to build your own map set-up by following the steps in this document so you have a full understanding of each component and how it is used.** You can still use the **Example Scene** as a reference of what a valid set-up looks like.
4. ➤ **IMPORTANT:** *Check that your **Shader Stripping / Fog modes are set to Custom and all checkboxes are checked**. These checkboxes can be found by going to **Edit > Project Settings > Graphics tab >** scroll down to **"Shader Stripping"** section.* ***Ensure all checkboxes are checked and both fields for Lightmap Modes and Fog Modes are set to 'Custom'. This screenshot illustrates correct settings.***

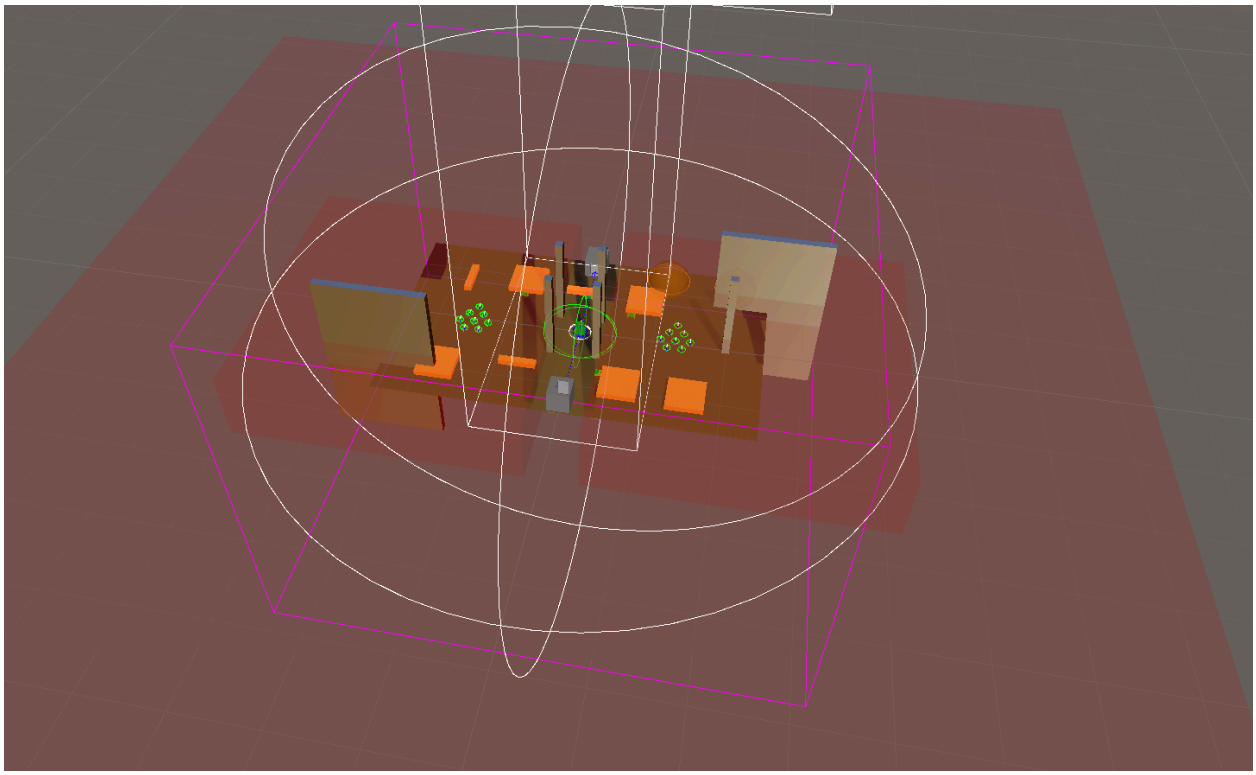


UPDATING AN EXISTING PROJECT: o

1. Make sure to backup your project before updating.
2. Confirm that you have renamed any test scene and that it is no longer named **ExampleScene**.
3. Drag the new Onward Custom Content Package into your Assets folder inside the editor and accept the import of new and changed files.

CREATING A CUSTOM MAP:

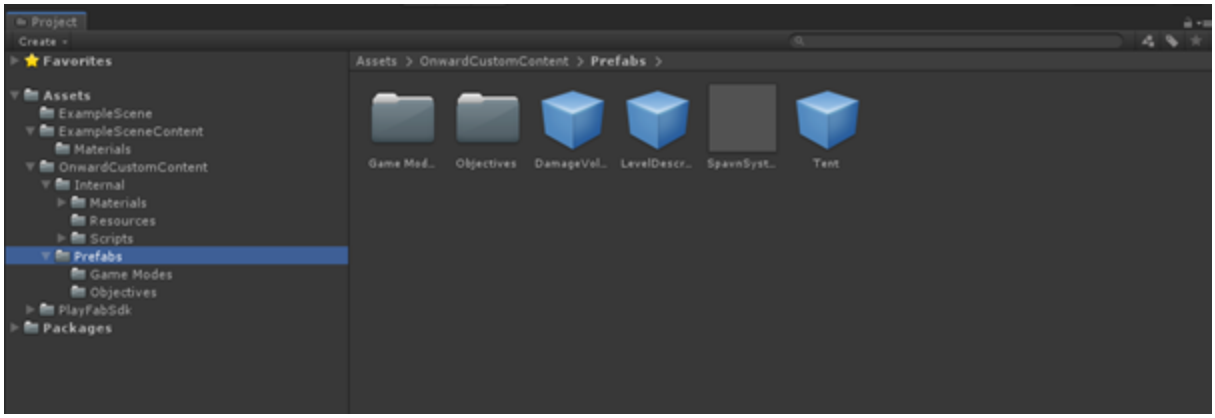
After loading the **ExampleScene**, you will see a simple map, as shown in the screenshot below.



➢ **IMPORTANT: Make sure you have already renamed this scene and saved it as a separate file. If you have not, do so now.** This will avoid any issues if the **ExampleScene** file is updated in the future.

- This scene is a valid map **FOR ALL GAME MODES**, which can be used as a starting point to experiment with your own custom map.

- All Unity prefabs for this project are located in ***Assets > OnwardCustomContent > Prefabs***



➢ **IMPORTANT: DO NOT modify the existing prefabs. Changing these prefabs will cause errors in your build.** *You may make modifications to the prefabs within the scene hierarchy instead.*

Now, we will begin to set up these game objects in a blank scene (or your own pre-existing map) to show you how they work and to allow you to build scenes from scratch. You can also follow along and inspect the game objects in the Example Scene to gain an understanding of how they work.

*You can always look to the original **Example Scene** for a reference on a working setup of a given game mode or feature.*

*To begin, every scene needs the following prefabs, you can find them in **OnwardCustomContent/Prefabs**:*

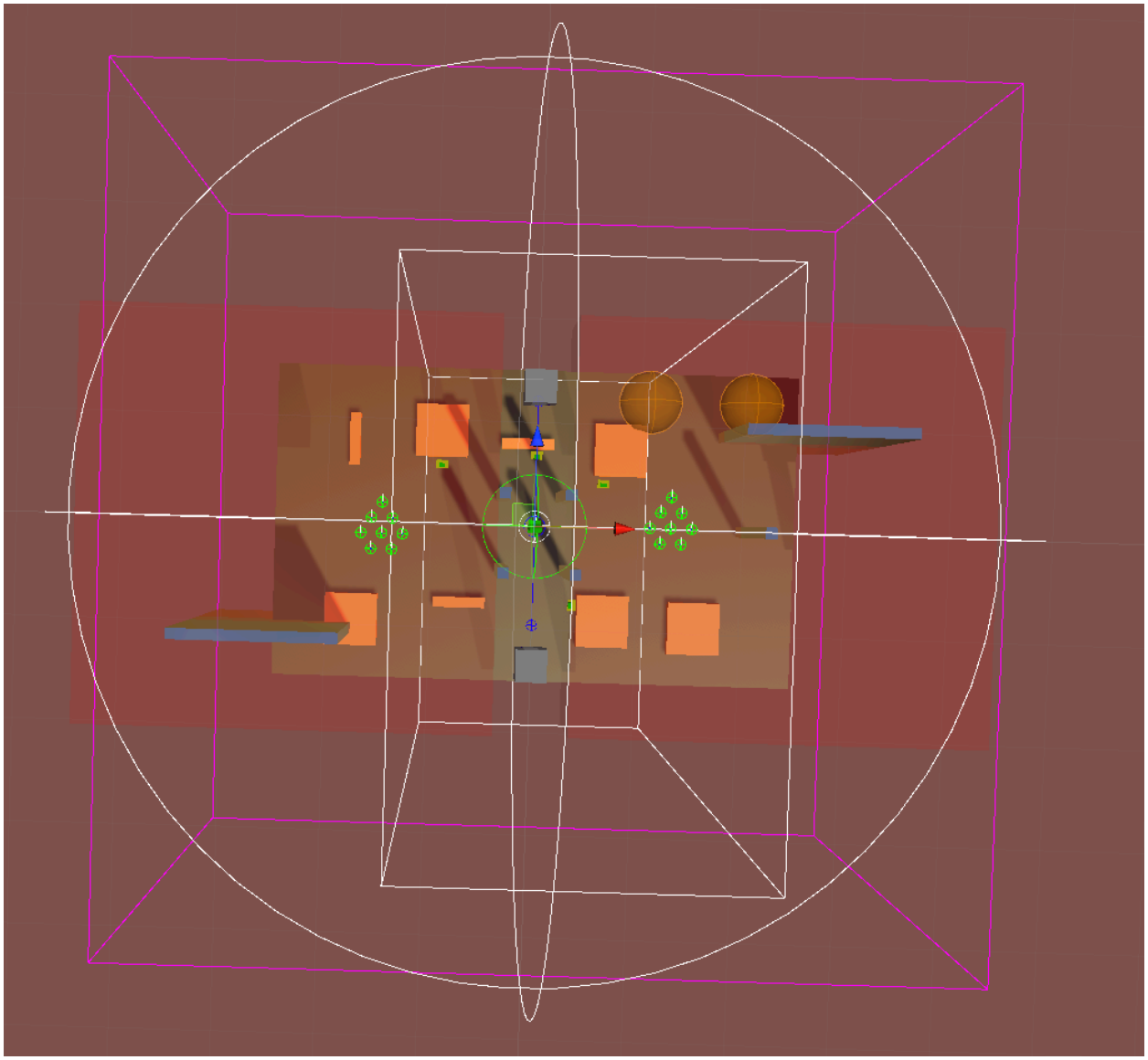
Tent
MapQuad
LevelDescriptor

Place these three Prefabs into your scene.

Ensure the Tent Prefab is moved to be situated well below the play area of the map, or players may see your level geometry through the tent's windows.

Ensure the MapQuad Prefab is appropriately-sized over the map, encapsulating its

playable-area within the magenta-pink wireframe box that is MapQuad's bounds. Use the size values in the inspector to shape the MapQuad to fit your map. Center the MapQuad over your map as shown in the picture below.



The LevelDescriptor's position does not matter, it will contain information about your map for later use.

*Now that you have essential gameplay objects in your map and have set them up properly, you can move onto adding **Game Modes** to your map.*

1. SET GAME MODES

In order to support a *game mode*, the Prefab for that *game mode* must be added to your custom map scene. We will be using the **Uplink** game mode for this first example, but you can add more/other game modes by following the **same steps** ***(some game modes have additional steps beyond Uplink, those will be detailed below this initial setup)***.

➢ IMPORTANT: Remember, every map is REQUIRED to support the Free Roam game mode and at least one other mode.
➢ IMPORTANT: If you are starting from scratch with a blank scene, create some geometry to serve as the 'ground' in your scene.

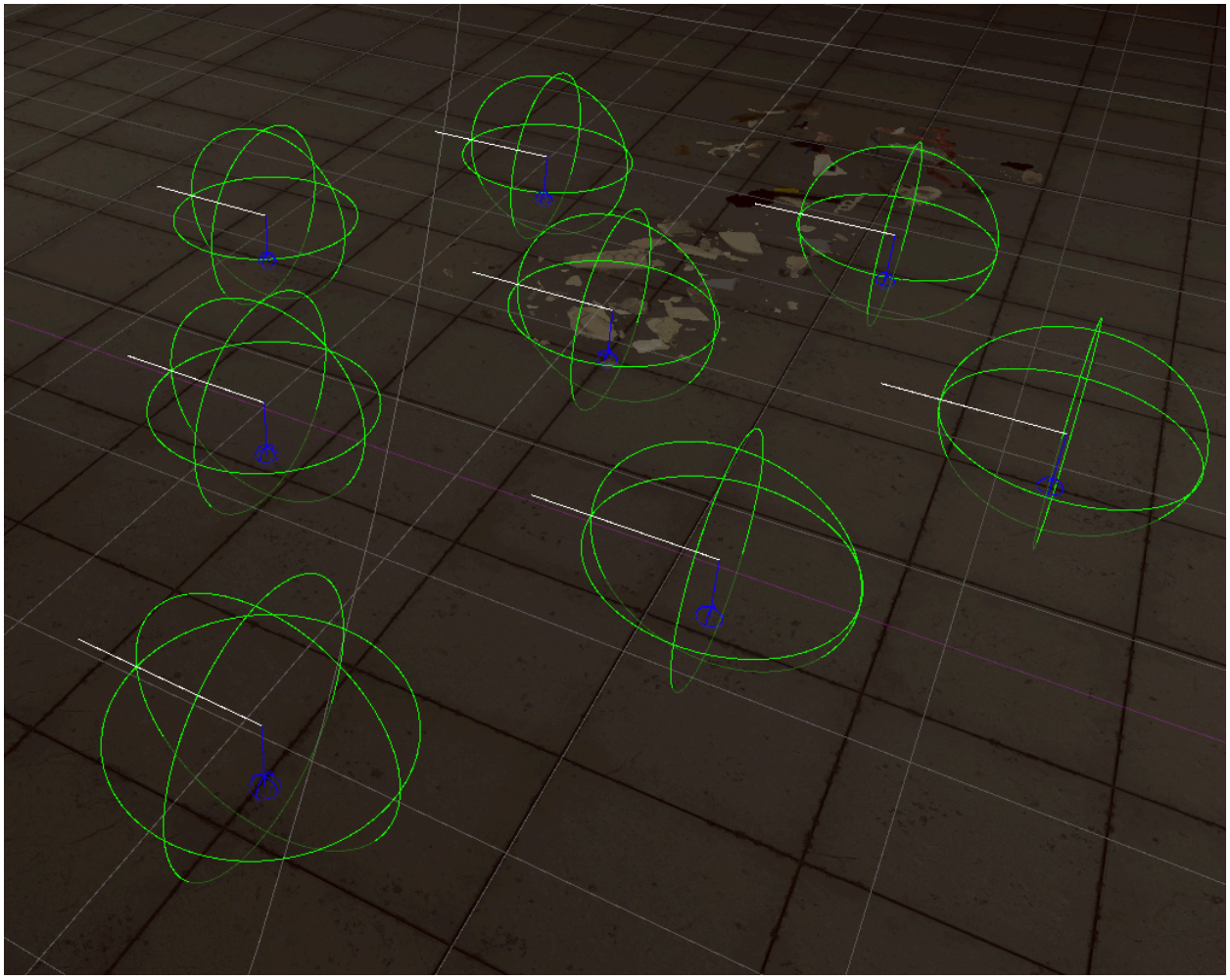
The **Example Scene** is designed to support **all game modes and thus has pre-existing setups**, but we will be replicating a new **Uplink** *game mode* in your own scene to start off.

To **replicate it's set up** begin by opening ***Assets > OnwardCustomContent > Prefabs > Game Modes*** and add **GM_Uplink** and the **GM_FreeRoam** prefabs to your scene. *(Game Mode prefabs will always start with **'GM_')***

2. SET OBJECTIVES & SPAWN POINTS

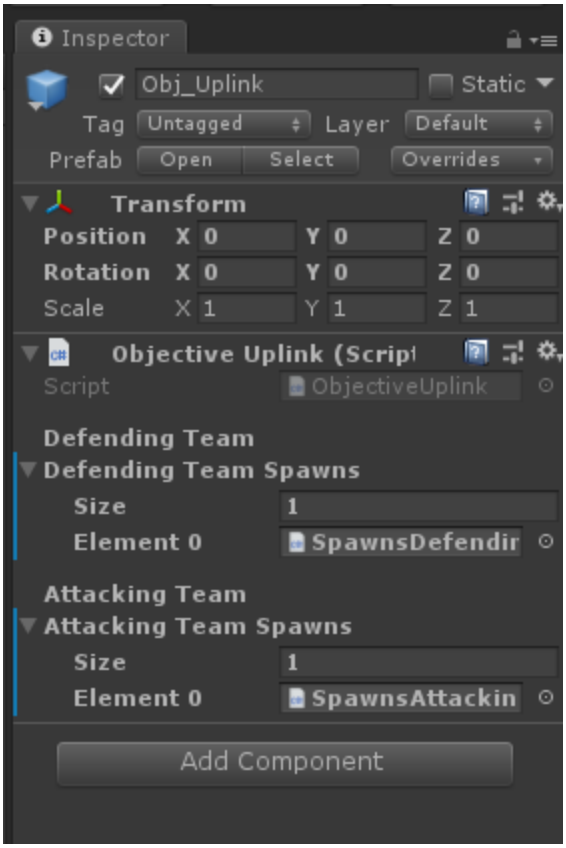
Once the game modes have been added, you will need to create *objectives* and *spawn points*. You can find the objective prefabs in ***Assets > OnwardCustomContent > Prefabs > Objectives***. In this case, drag the **Obj_Uplink** prefab into your scene. *(Objective prefabs will always start with '**Obj_')* Select the **Obj_Uplink** prefab in your scene hierarchy so you can see the information below in the inspector window.

Now it's time to add *spawn points*. *Open* ***Assets > OnwardCustomContent > Prefabs*** and drag **two (2**) **SpawnSystem** prefab into your scene. Rename each prefab (F2, or through the inspector window) to **SpawnsAttacking** and **SpawnsDefending**. <*These are example names, you can rename them to whatever you wish.*



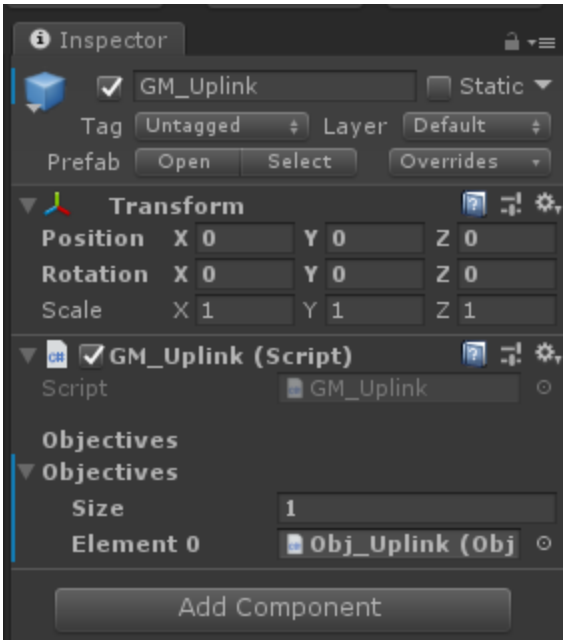
SpawnSystem prefab in scene.

Go back to the inspector window for the **Obj_Uplink prefab** and set the **Defending and Attacking Team sizes fields** to **1**, and then drag and drop the **SpawnSystem prefabs** that you have named **SpawnsAttacking** and **SpawnsDefending** in their respective fields for **attackers** and **defenders**. *Your settings on the Obj_Uplink should look like this:*



Correct spawns (you can have more than 1 spawn for Attackers to vary the gameplay)

The last item in this step is to make sure that the GM_ prefab is referencing the matching Obj_ prefab. Select **GM_Uplink** from the hierarchy to open it in the inspector. Increase the Size of the Objectives field to **one (1)**. Drag the **Obj_Uplink** prefab into the field of the **Objectives** dropdown. *Your settings should match the image below:*



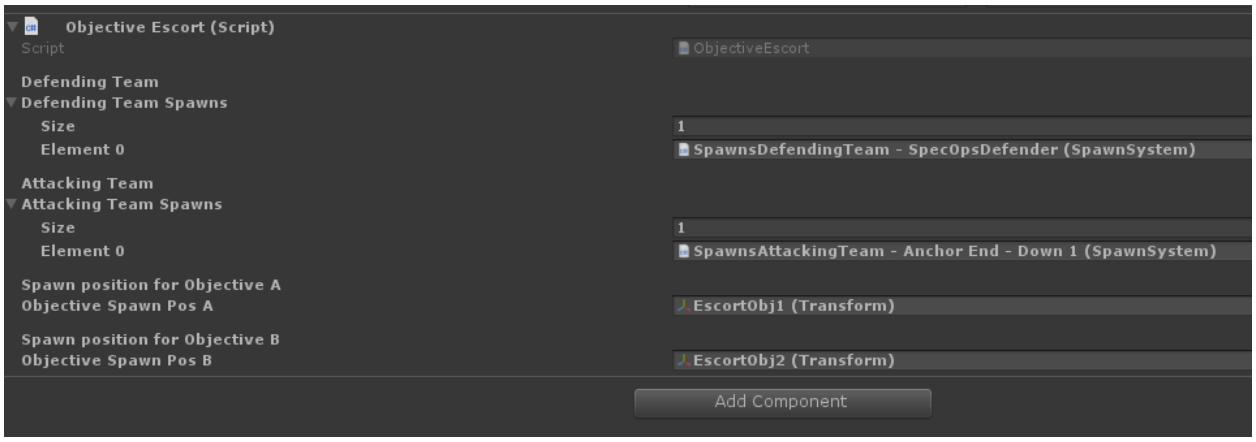
If you want to create an Uplink game with a single objective, that's all you need to do! Your Uplink game mode will now work. Refer to Section 7: Testing for testing information. *You can add more objectives and game modes using the steps above, just add more Obj_Uplink prefabs (that are set up with different spawns) to the GM_Uplink prefab.*

Be aware that game modes can share spawn points, and you can also modify *individual spawn points* within the **SpawnSystem** object in your hierarchy.

We recommend that you place a **DamageVolume** prefab below the map and scale it to cover the entire footprint of your map. If a player somehow falls through the map, this will prevent them from falling endlessly. This has already been done in the ExampleScene, and you can use this as a reference for your own maps. You can find **DamageVolume** in ***Assets > OnwardCustomContent > Prefabs***.

2A. ESCORT ADDITIONAL SETUP:

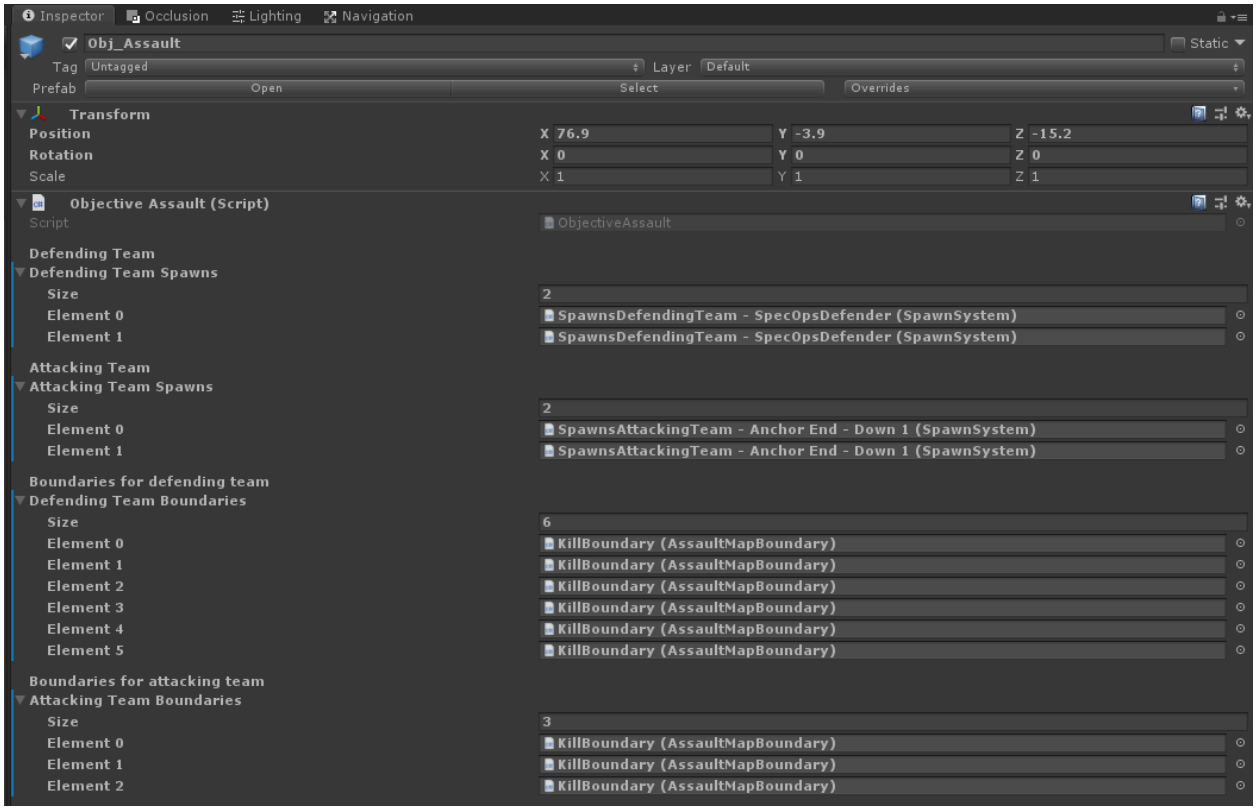
Escort is a game mode in which MARSOC has to escort 1 member of their team (The VIP) to one of two extraction points- marked with red smoke grenades. For this mode use **GM_Escort** and **Obj_Escort** prefabs. This mode allows you to define 2 different objective placements as dictated in the screenshot below:



When you place the **Obj_Escort** prefab you will have two objective children that are created that you must place in unique locations.
Take time to consider the implications of two objectives to reach and how that might influence your gameplay.

2B. UPLINK ASSAULT ADDITIONAL SETUP:

Uplink Assault is a game mode which has MARSOC and Volk fighting over an objective with ***respawns for both sides***. For this mode use the **GM_Assault** and **Obj_Assault** prefabs. To prevent spawn camping, additional no-go gameplay zones can be defined using the **AssaultMapBoundary** prefab. Place and scale these prefabs to fit around zones the opposing teams should not be able to encroach on (such as player spawns), then associate them with the **Obj_Assault** prefab. You can associate just a single zone or multiple to each team's limitations (as is done in this screenshot). **Place 2 SpawnSystem for each team, to allow them to choose between A and B spawn points in-game.**



Boundaries for **ATTACKERS** will penalize **ATTACKERS** from entering the zone.
Boundaries for **DEFENDERS** will penalize **DEFENDERS** from entering the zone.
You can also rename your **AssaultMapBoundary** prefabs to help distinguish them from each other.

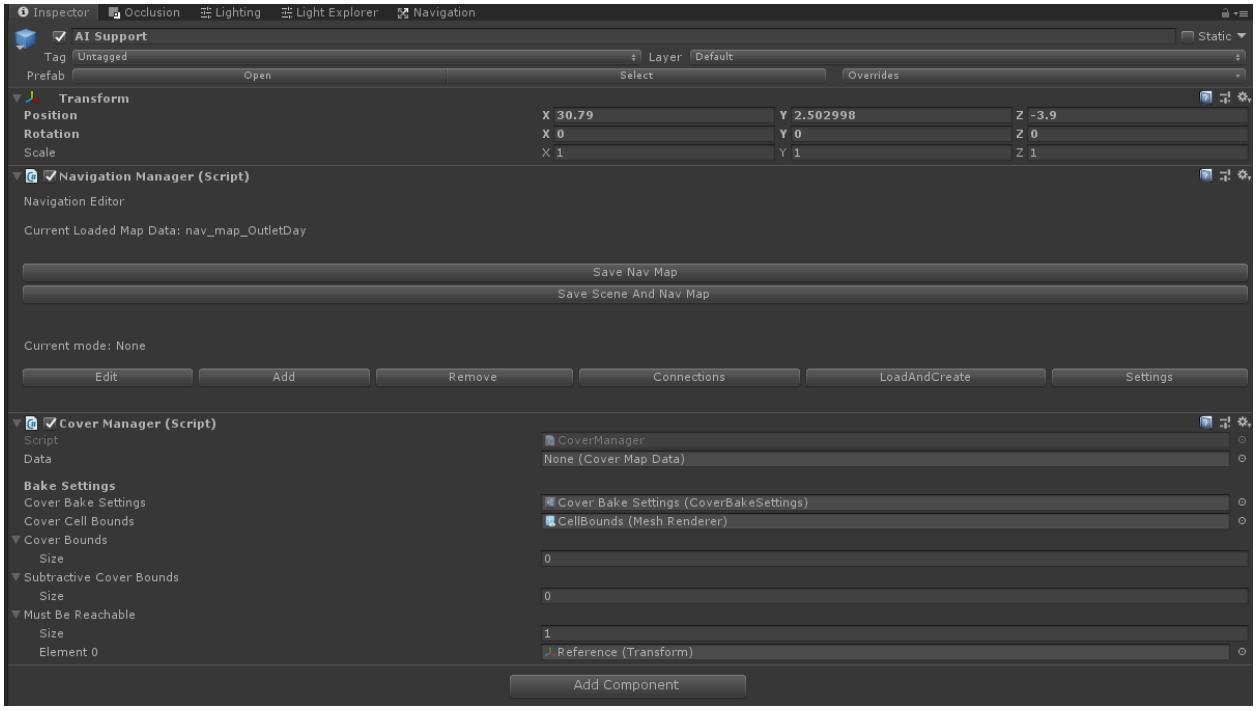
2C. AI MODES ADDITIONAL SETUP:

AI modes include **Hunt**, in which MARSOC search and destroy Volk forces and **Evac**, in which MARSOC forces seek to survive and eventually escape Volk forces by reaching a rescue helicopter.

These modes use the **GM_Hunt** and **Obj_Hunt** or **GM_Evac** and **Obj_Evac** prefabs.

➢ IMPORTANT: You must set up BOTH AI modes to gain functionality. It is not possible to set up just one or the other.

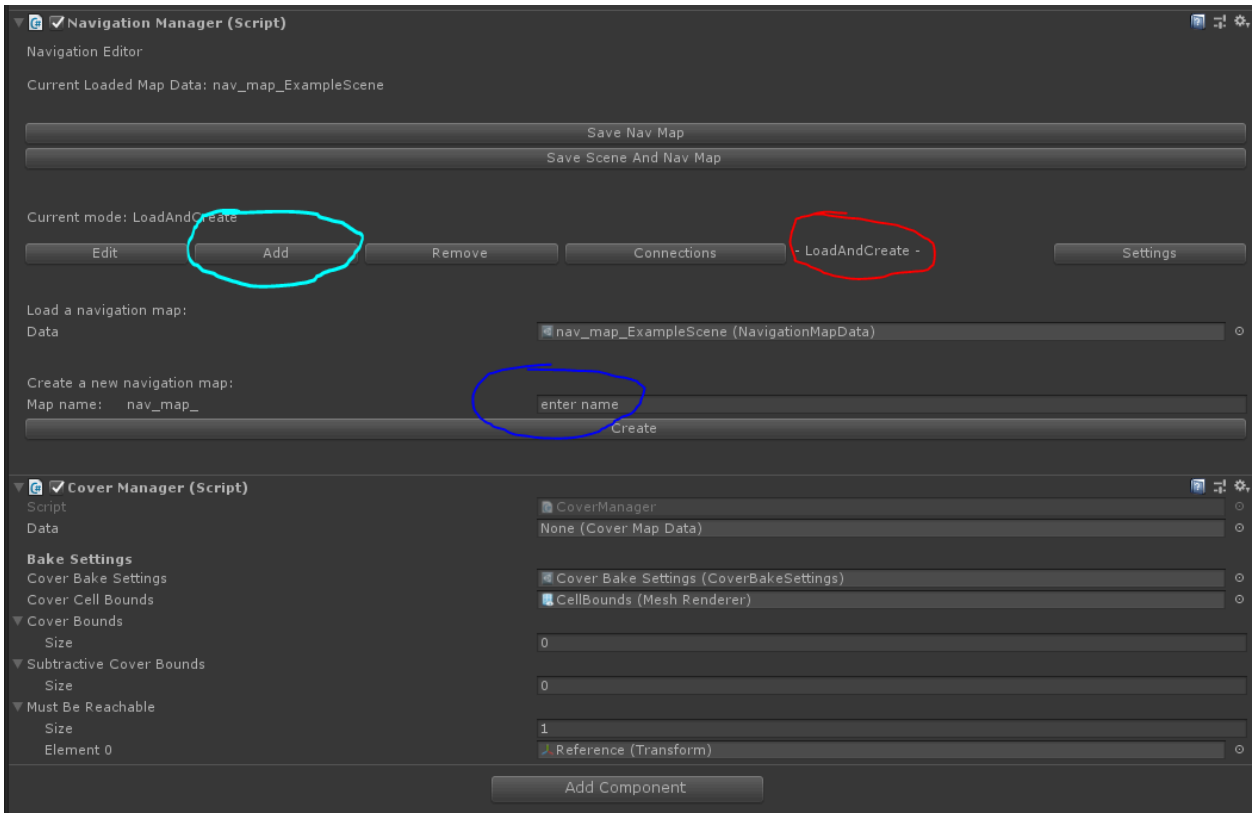
In addition to the normal game mode/objective objects, AI modes require the **AI_Support prefab.** *Drag the **AI_Support** prefab into the scene. You'll only need 1 for all modes.*



AI Support is a prefab that has some setup options for the navigation of the AI.

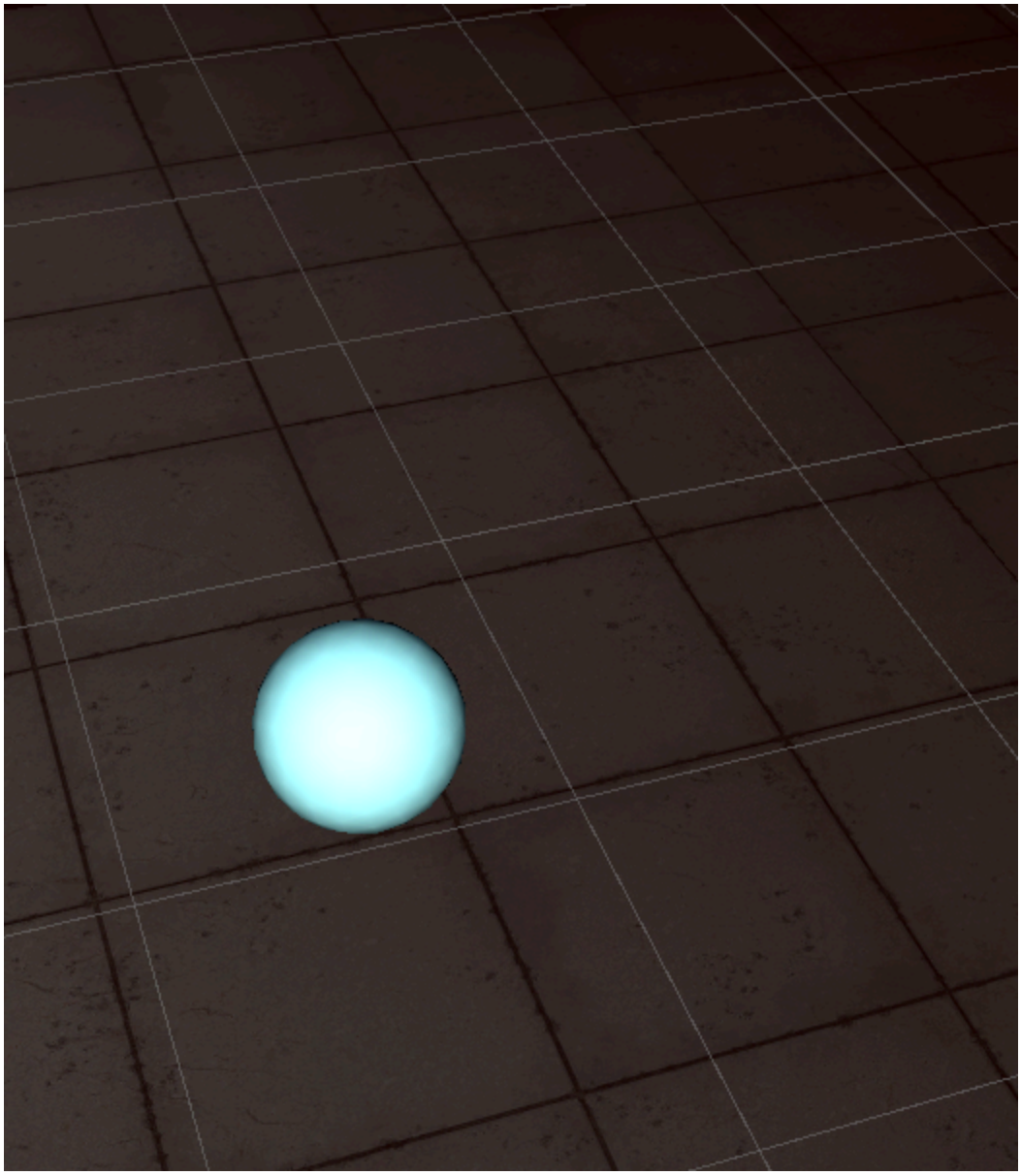
We will first use **AI_Support** to create a new **Navigation Map asset** to hold some data we will put into it.

Press "Load and Create" (circled in **red**) and then give our new Nav Map a name in the field circled in **blue.**

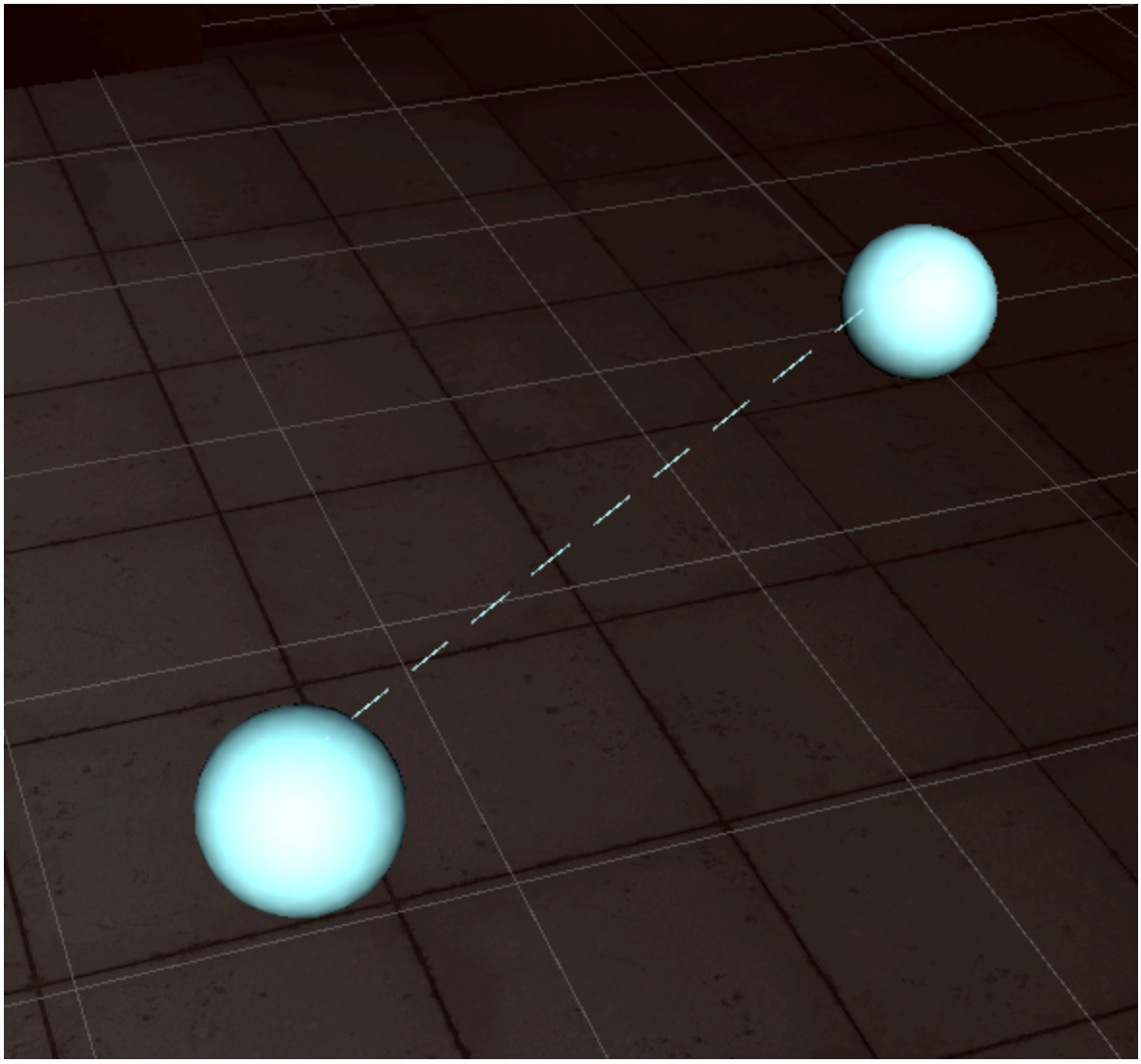


Press **"Create"** and a new **Nav Map asset** will be created and associated for your map.
*You can search for your named **nav map asset** to re-associate it if you decide to try different nav map profiles or it somehow becomes lost. It will always start with **nav_map_** (example: **nav_map_MyFirstNavMap**).*

Next, we will add **navigation points** to the scene. Press the **“Add”** Button (Circled in **TEAL**) and
click on an **unoccupied, playable floor space** in your map- you will see a white/blue dot generated where you’ve clicked your mouse in the scene.



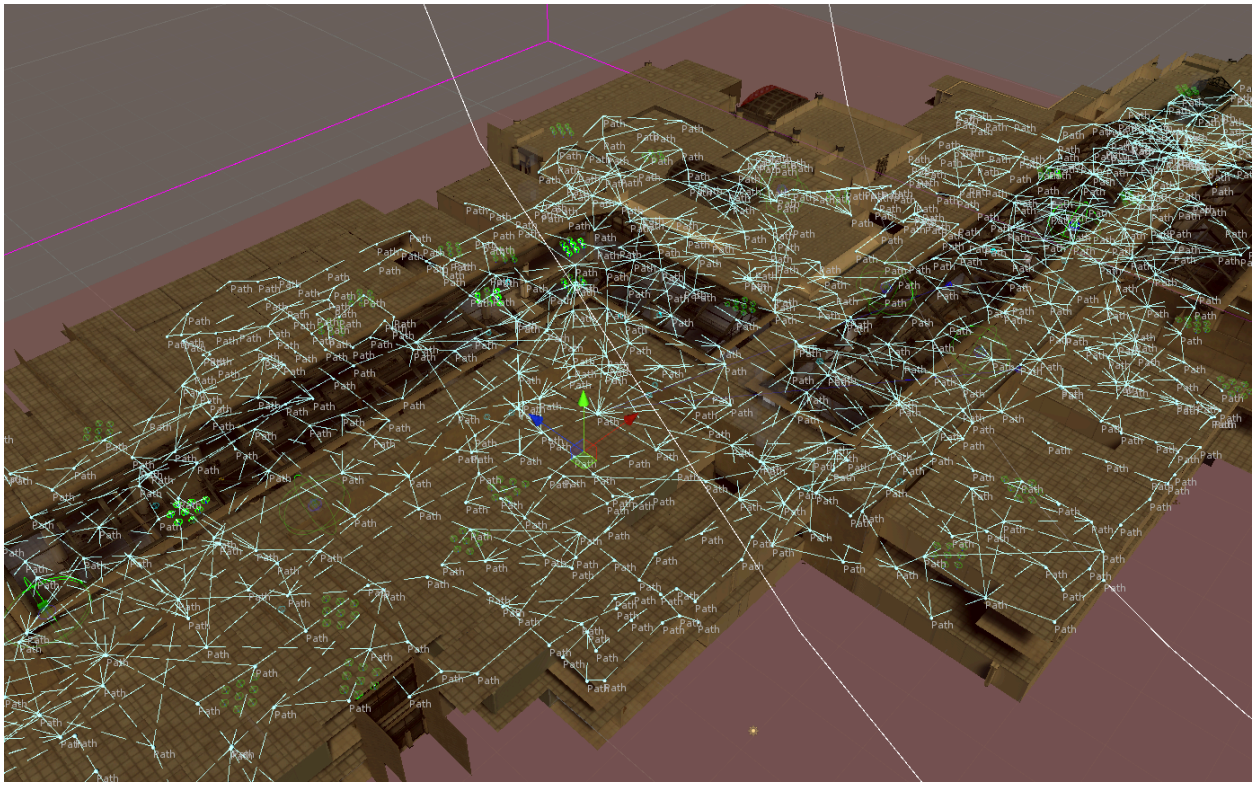
This is a ***navigation node***. Click in another nearby unoccupied space and they will link together.



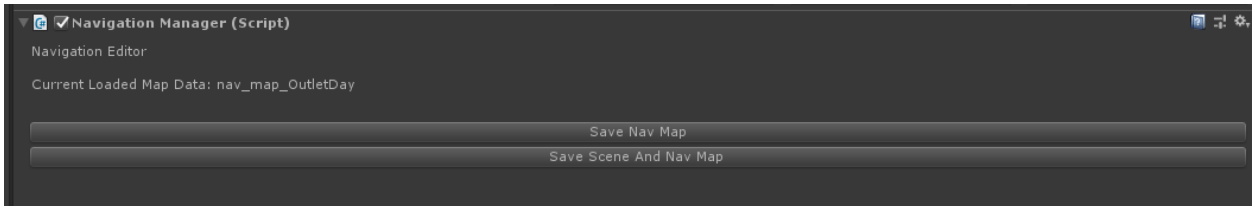
Navigation nodes will try to link to as many nearby nodes as they can within line of sight, thus creating a network of navigation nodes.

Now, begin clicking throughout your scene on the ground in player-walkable spaces to allow the **navigation nodes** to link to others that are in line of sight nearby, forming a web of connections that extends through all of the player-walkable areas in your scene. A demonstration of what this should look like when finished is below: **a network of interlinked navigation nodes that goes into each room/area of your map.**

IMPORTANT: ***Make sure navigation nodes are never intersecting with geometry and that they are on the ground in a spot where a player could reach and stand on top of said point.***



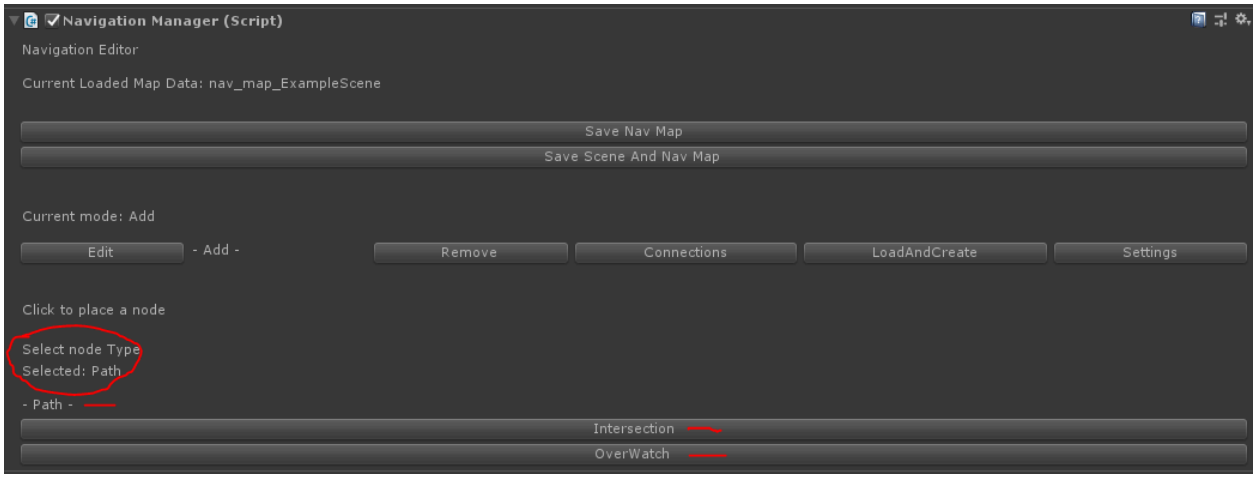
Once your nodes are set, ensure you press **“Save scene and navmap”** to save your navmap.



(Note: *You can also save your nav map individually*)

ADDITIONAL NAVIGATION NODE TYPES

You may have noticed that there are other types of **navigation nodes** that exist.



These can denote special nodes that clue the AI into information about their environment. For full understanding of these special nodes, you will need to read the ***next section*** about **AI_Spawners-** but here is a preface on what these nodes do to prepare you:

Path Node - This is the node type we have been using by default. It allows the AI a basic option for their navigational goals and will work fine by itself.

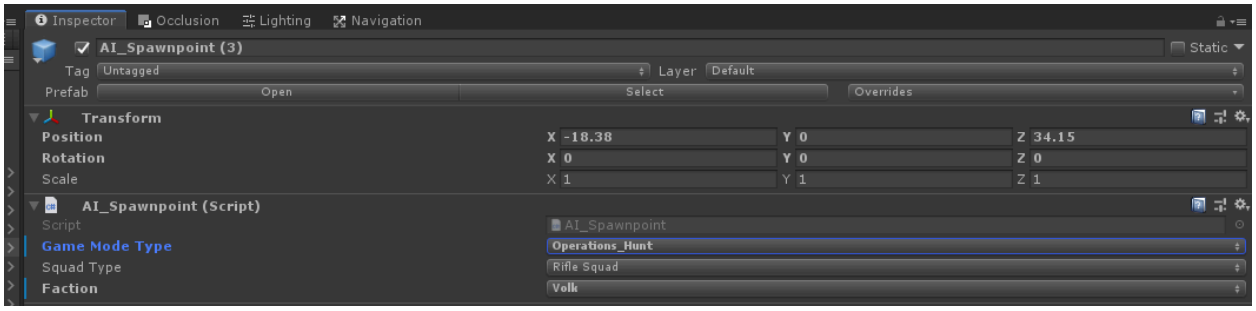
Intersection Node - A special node that tells AI there are many connections to other nodes on this node. Making an **Intersection Node** at every major node junction will help the AI choose a pathway with more options when they are nearby, further randomizing their behavior.

OverWatch Node - A special node that tells one type of AI (covered in the **AI_Spawner** section) that this node is right next to a good area to **look out or snipe** from- such as a window in a second story building, or a far-off crevice that is easily missed. **Sniper AI** will try to find these nodes and break off from their Squad to pick the player(s) off from an OverWatch node.

Now that you know about these nodes and how they interact, you can use the “Remove” button in the AI_Support prefab to strategically remove nodes so you can replace them with special nodes *if you desire*. **This step is not necessary for functioning AI, you will be fine if you do not add special nodes.**

AI_Spawners

Next, you will add some AI Spawners to the level. Place an **AI_Spawner** prefab in the scene and we will go over the settings it has.



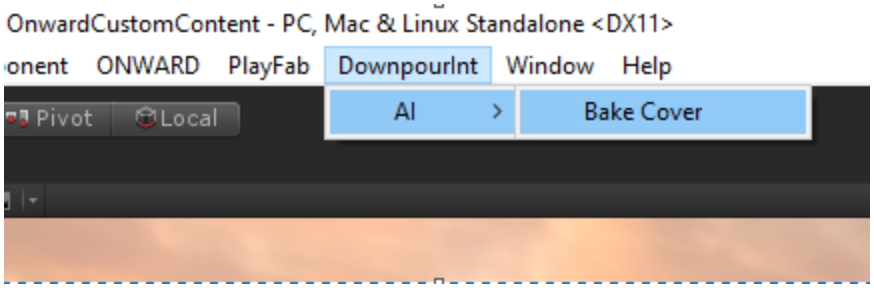
Game Mode Type: Set this to either **Hunt**, or **Evac** depending on which mode you want this spawner to work in. **You will need different spawners for both Evac AND Hunt modes**. **IMPORTANT: Your AI spawners will NOT work unless you set a game mode type for them.**

Squad Type: Different types of soldiers will react in different ways. *Rifle_Squad* is the default soldier equipped with Rifleman-like loadouts, you can experiment with the specialized classes such as *Sniper*. **NOTE:** The additional **Node types** discussed earlier for the navigation map may influence how some of the squads react, *but are not required.*

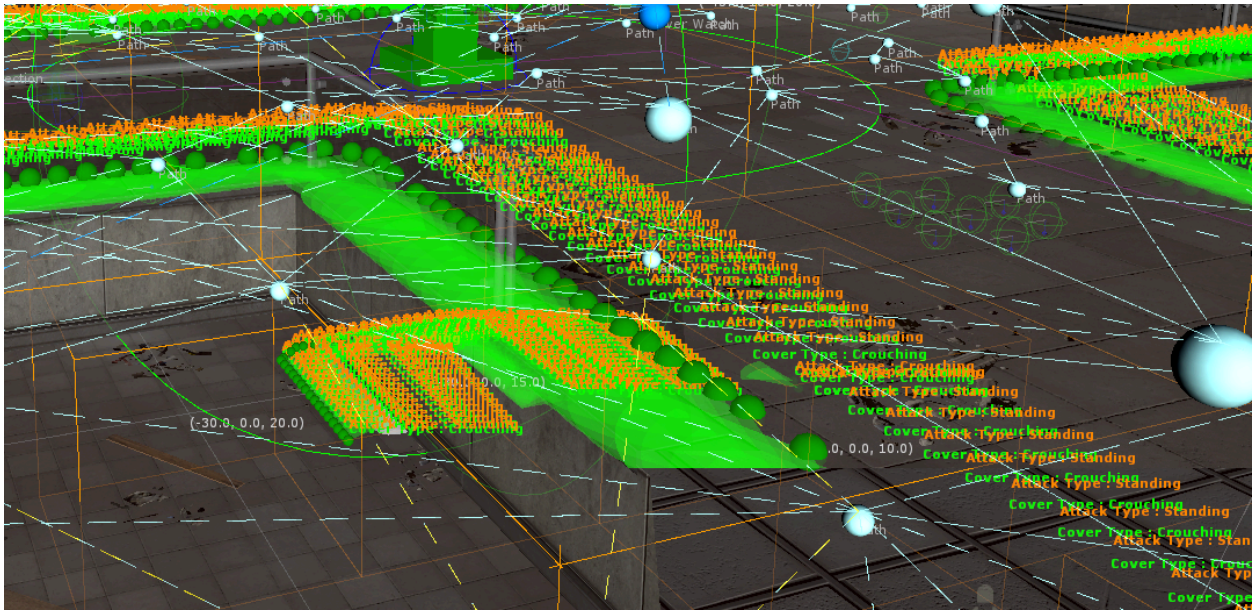
Set up these **AI_Spawner** throughout your map by placing them onto the ground (*holding **ctrl+shift** while grabbing the **Transform** gizmo of an object will make it snap to the floor. This can be useful for quickly placing AI_Spawner prefabs)*, making key decisions about where you want the action to be. Once you are satisfied you can move on.

Cover Manager

Next we will be setting up the second component on the **AI_Support** prefab and that is the cover manager. Go to the top menu marked ***"DownpourInt"*** and hover over *AI* then press **"Bake Cover"**



You will have to wait a moment, but your map will become populated with Cover Points. These are what your AI will use to determine where applicable cover exists during a firefight.

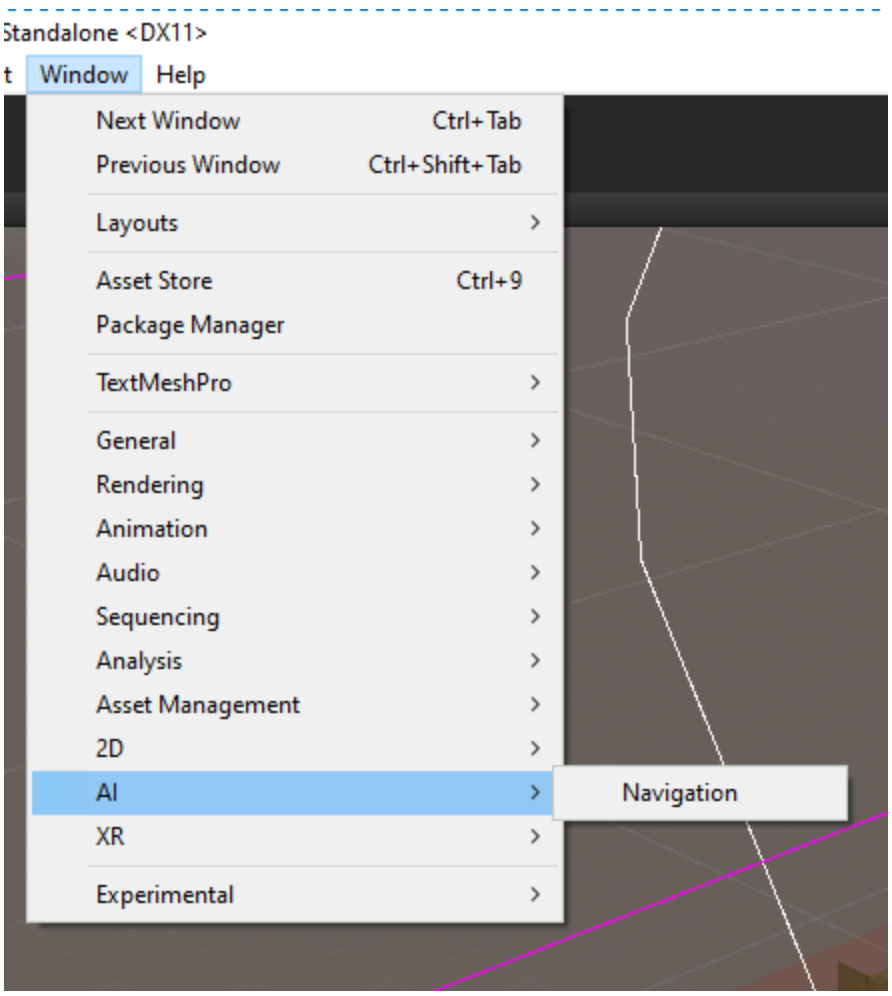


Cover Point examples

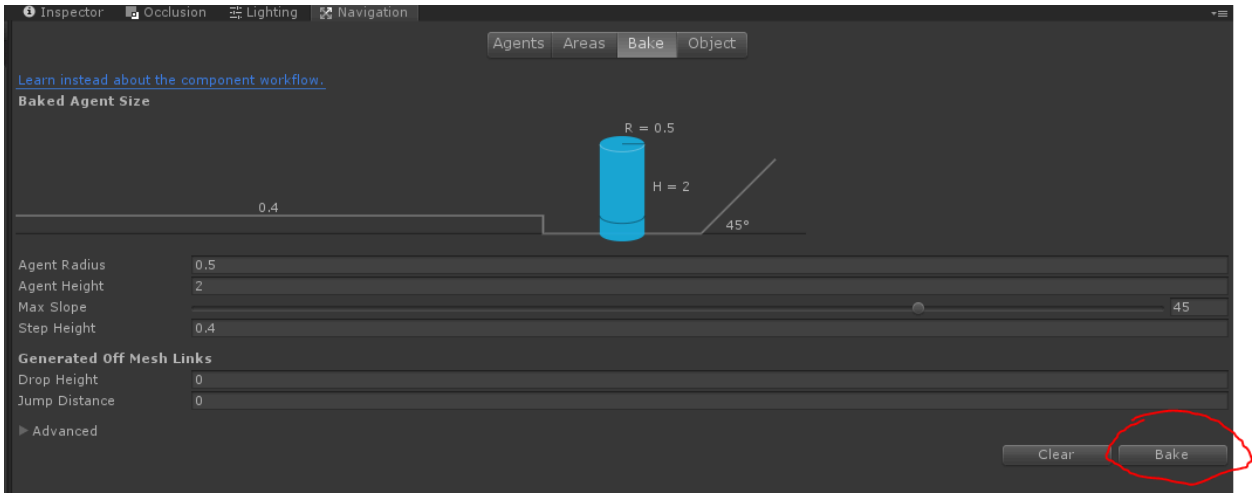
Ensure you **Save your scene** after you generate these cover points.

Finally, we will setup a **Navigation Mesh** and our *AI modes* will be functional.

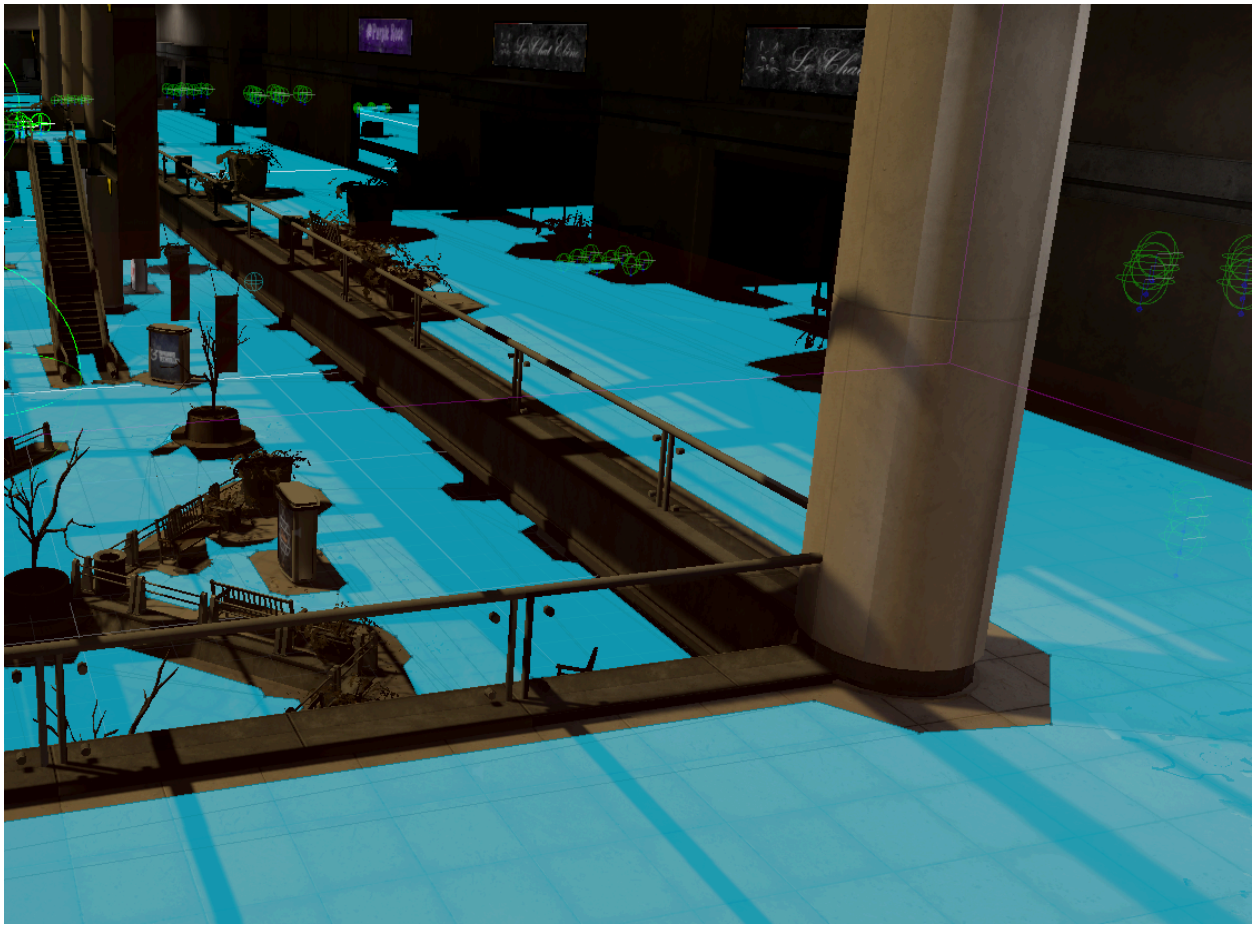
Open your **‘Window’** dropdown and go to the **‘AI’** dropdown and select **‘Navigation’**



In the *Navigation panel*, you will want to switch to the **"Bake"** tab and press **"Bake"**



This will take a moment and when its finished you will have generated a **Navigation Mesh** that your AI will use to walk upon.



This is what your finished **Navigation Mesh** will look like when you have the **'Bake'** tab of the *Navigation panel* open.

Now that you have created a **Navigation Mesh** and followed steps to set up *requisite components* (**GM_Hunt/GM_Evac linked to Obj_ Hunt/Obj_Evac prefabs and SpawnSystems linked to Obj_ prefabs, set up AI Support prefab, place AI Spawners and generated Navigation Mesh**), your *AI modes* should be functional.

SOCIAL MODES *(One in the Chamber, Gun Game, Spec Ops)* ADDITIONAL SET UP

In **Gun Game**, players must fight to be the first to achieve a kill with a list of weapons, receiving a new one each time they score a kill.

In **One in the Chamber**, players must fight to be the last one standing, their only weapon being a bolt-action rifle that contains a single round. You can find additional rounds throughout the game mode, but if you fail to kill on the first shot, you'll need to resort to melee tactics.

In **Spec Ops**, a *MARSOC* squad with **flashlight-equipped pistols and limited ammo** tries to survive a team of *Volk* assassins armed with **combat knives and night vision** until the MARSOC can escape after a set time period.
A dense fog covers the battlefield, **limiting MARSOC visibility**. The Volk assassins have ditched all unnecessary equipment, **greatly increasing their mobility and stealth**.

Ammo Boxes are scattered throughout the map- enticing each side with the tools to survive, or assassinate. The MARSOC will find **extra magazines for their sidearms** and the Volk will have a chance to find a **flashbang** that can give them the edge to close the gap. *Who will survive?*

How to set up Social Modes:

➢ **IMPORTANT:** ***Social Modes*** *are different from other* ***GM_*** *Prefabs because they do not require an* ***Obj_*** *Prefab to link into the* ***GM_*** *Prefab. They work just by adding a* ***GM_OneIntheChamber****,* ***GM_GunGame****, or* ***GM_SpecOps*** *Prefab into the scene and associating* ***SpawnSystem*** *Prefabs with them.*

Gun Game & One In the Chamber -

These modes make special use of a single ***SpawnSystem*** *prefab to allow free-for-all Deathmatch gameplay with respawns.*

Both of these game modes use a single **SpawnSystem** Prefab that is associated into the respective **GM_GunGame** and **GM_OneInTheChamber** Prefabs. You can use the same **SpawnSystem** for both Game Modes, or individualize them.

Initially, the **SpawnSystem** will have **8** spawn locations arranged next to each-other when you place this prefab. For these two *Free For All* modes, you will be **moving each individual spawn to a unique location**. If you desire more than **8** spawn points, you can duplicate any of the *SpawnPoint* children under the **SpawnSystem** parent and continue to place them in new locations until you are satisfied with the amount of spawns.

Once your **SpawnSystem** is linked to the **GM_GunGame** and **GM_OneInTheChamber** prefabs and you have moved the individual *SpawnPoints* in the **SpawnSystem** to unique locations (and possibly added additional ones through duplication), your setup is complete!

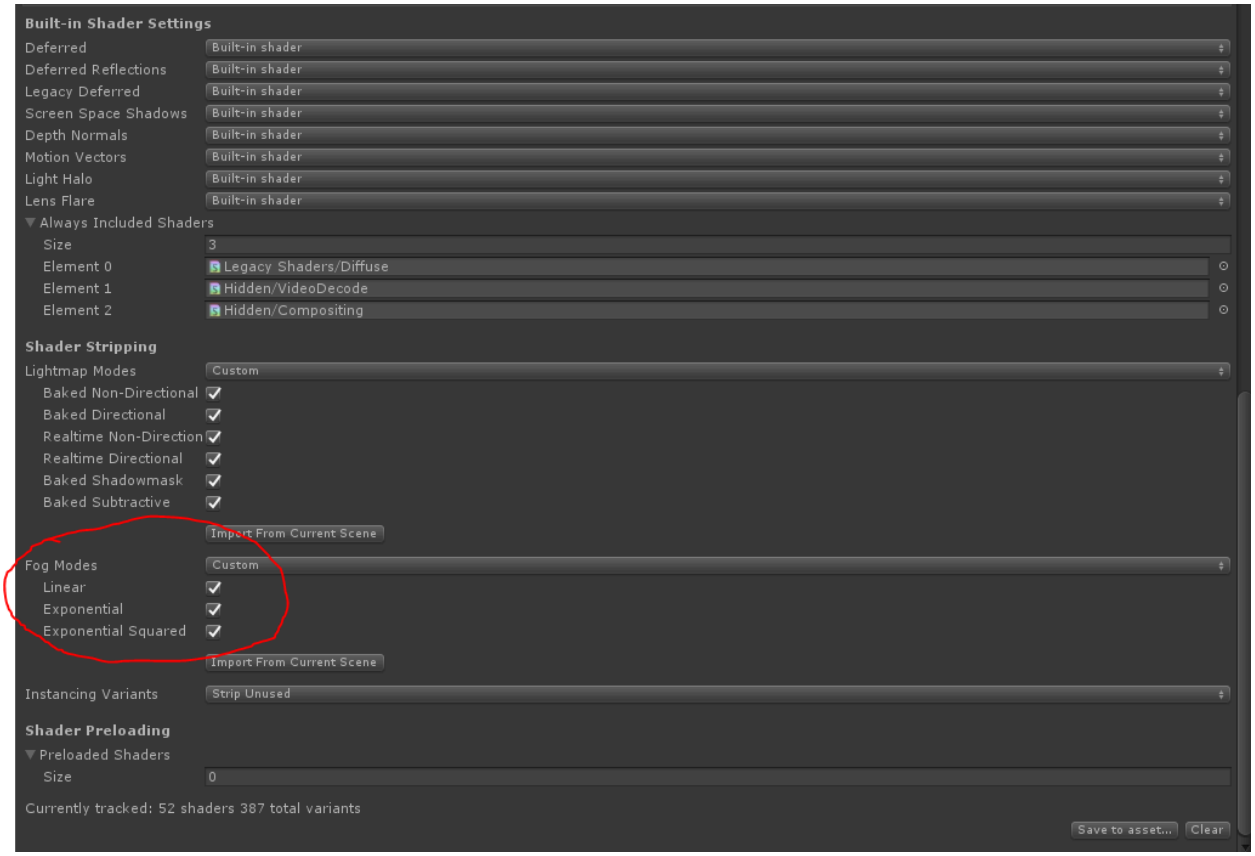
Spec Ops -
*This mode makes use of **SpawnSystems** for attackers (Volk) and defenders (MARSOC) as the regular non-FFA game modes do.*

Place **Spawn System** prefabs for both Attackers and Defenders in the fields of the **GM_SpecOps** prefab. Once you've moved your spawns into unique locations, your setup is complete. Note the section below on fog settings if you have not already verified your settings before.

➢ **IMPORTANT:** *If your map supports Spec Ops please double-check that your **Shader Stripping / Fog modes are set to Custom and all checkboxes are checked**. These checkboxes can be found by going to **Edit > Project Settings > Graphics tab > scroll down***

to “Shader Stripping” section.

Ensure all checkboxes are checked and both fields for Lightmap Modes and Fog Modes are set to ‘Custom’



This concludes the section on additional setup of extra game modes.

3. ADDITIONAL GAMEPLAY COMPONENTS

There are additional scripts, prefabs and settings available in the Onward Custom Content bundle that will allow you to further upgrade the quality and gameplay of your maps. These components and their use are covered in this section (**Note: You can find scripts by searching for them by name in the *"Add Component"* dropdown of the inspector**);

A.
'Set Physic Material' (Script) - Adding this **script** to any GameObject with a **collider** will allow you to choose a **physical material** to add to that collider. This will decide what type of **impact effect and penetration** a given surface has.

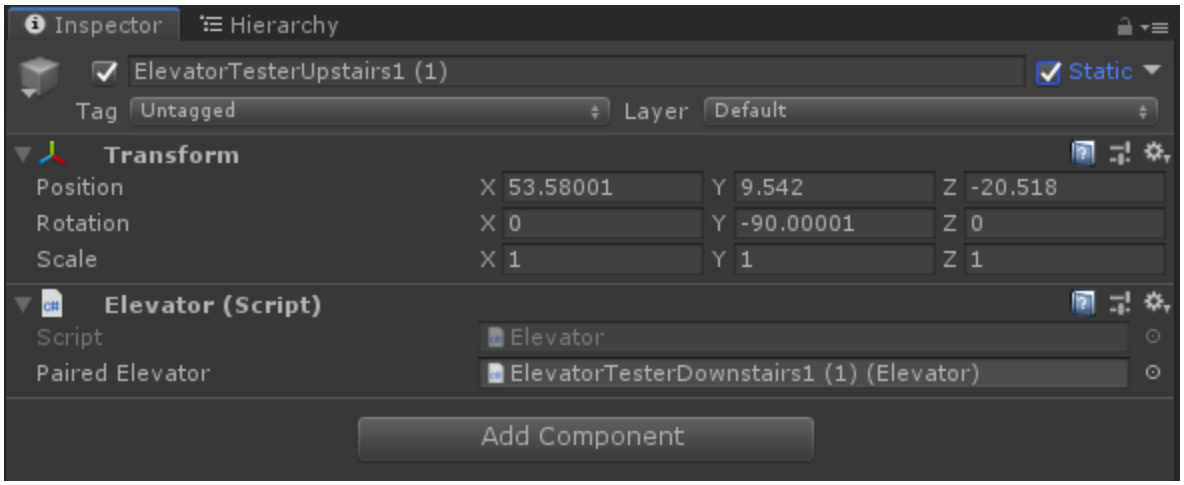
There are a few different *physical materials* available but the general overview is as follows:
Concrete - **Impenetrable**
Glass - **Penetrable**, reduces damage slightly
Glass Unpenetrable - **Impenetrable**
GlassVfx - **Penetrable**, does not reduce damage
Metal - **Penetrable**, reduces damage
Metal Unpenetrable - **Impenetrable**
Wood - **Penetrable**, reduces damage
Wood Unpenetrable - **Impenetrable**
Dirt - **Impenetrable**
Grass - **Impenetrable**
Cloth - **Penetrable**, reduces damage slightly
Cloth Unpenetrable - **Impenetrable**

ClothVfx - **Penetrable**, does not reduce damage
Water - **Penetrable**, reduces damage
Snow - **Penetrable**, reduces damage
SnowUnpenetrable - **Impenetrable**
Ice - **Penetrable**, reduces damage
IceUnpenetrable - **Impenetrable**

NEW **There is a checkbox called *'AI See Through'* for allowing AI to see through a material as well on a given object using Set Physic Material, use this for Glass or other see-through objects such as chain-link fences.**

Warning: If you have no SetPhysicMaterial, the materials of all objects with colliders will default to Concrete - Impenetrable.

B.
'Elevator' (Prefab) - This **Prefab** is a functional elevator that players can use to send themselves or equipment/items to a paired elevator. Two Elevator Prefabs will be able to be linked together by dropping each respective Elevator Prefab instance into the field of the linked elevator instance. (**Elevator A** reference goes in **Elevator B**'s field, and vice-versa.) Screenshot below shows a proper elevator set-up. *Elevators will start named as 'Elevator' but you should rename them to suit your organizational needs.*

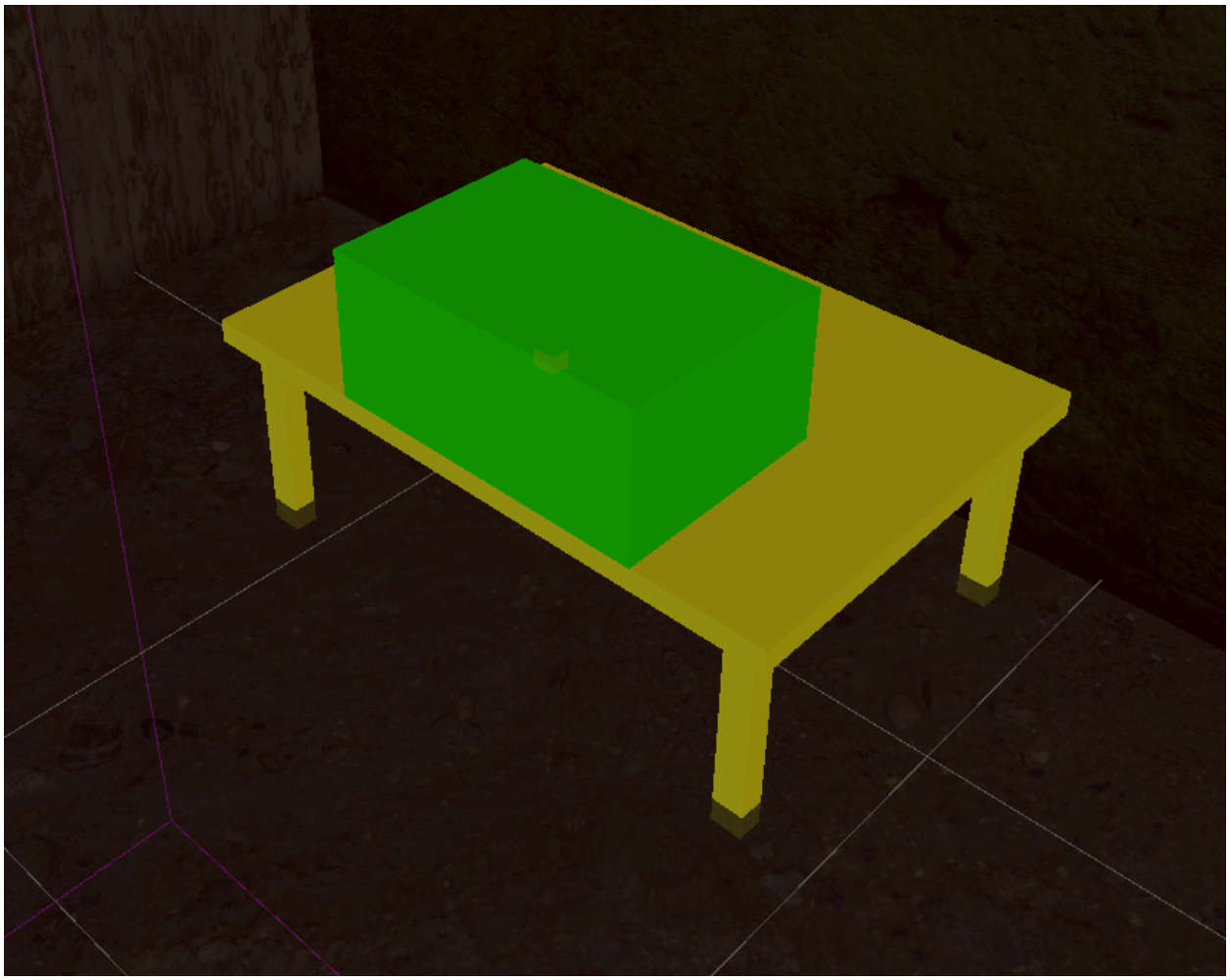


Note: *ElevatorTesterUpstairs1* has the connection to *ElevatorTesterDownstairs1* (its destination elevator), ElevatorTesterDownstairs1 has a setup that is a mirrored version (reverse) of this.

C.

‘Ammobox’ (Prefab) - This Prefab is a cache of ammunition and other supplies that can be found across the map in these game modes: **Hunt, Evac, One in the Chamber, Spec Ops**. *Its contents will change depending on the game mode that utilizes them.*

Ammoboxes open in one intended direction and that direction is on the edge of the table gizmo displayed when it is placed in the map. The edge of the table the ammo box is closest to is the “front”.



You are REQUIRED to place at least four (4) Ammobox throughout your map if your map has a Hunt/Evac, One in the Chamber or SpecOps game mode. *You may add more AmmoBoxes as desired.*

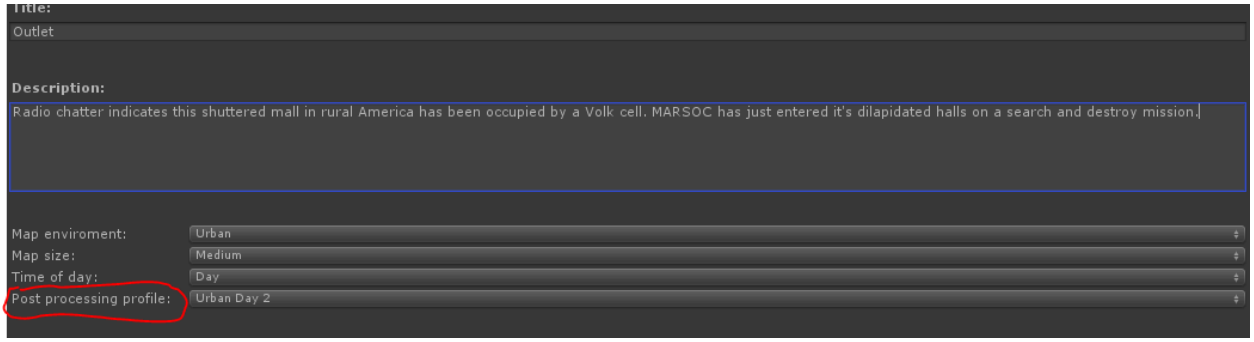
Hunt/Evac: The Ammobox spawns equipment such as magazines for weapons that a player has equipped, or grenades and syringes to support them in battle.

One in the Chamber: The Ammobox spawns a magazine containing one round for a bolt-action rifle.

Spec Ops: The Ammobox contains different tools for MARSOC and Volk. They are as follows: ***Pistol magazines*** *for MARSOC.* ***Combat knives*** *and a rare chance to spawn a* ***single flashbang*** *for VOLK.*

D.

PostProcessing Profile (optional setting in Publish page, details in section 6. PUBLISHING) - *Onward* uses post processing to further boost the quality of the image delivered to a player during gameplay and stylistically distinguish each map. There are several different profiles available to choose from for day and night maps of varying locales. *Experiment with the difference choices for your use-case and determine which you like the best for your map*. Alternatively, you can have no PostProcessing.



NOTE: You will only be able to see a PostProcessing setting change when you load into your map.

E.

OnwardCustomAudioSource is a prefab that allows you to play your own audio files in the game. It is recommended to not use too many as there are potential memory usage issues since the audio files are fully uncompressed during runtime, and the cpu usage will be higher than the game's built in audio.

Audio Source Type refers to how the audio is heard. Local Ambience will play from a 3D spatialized position. Global Ambience will play throughout the entire scene as if you are in the center of it.

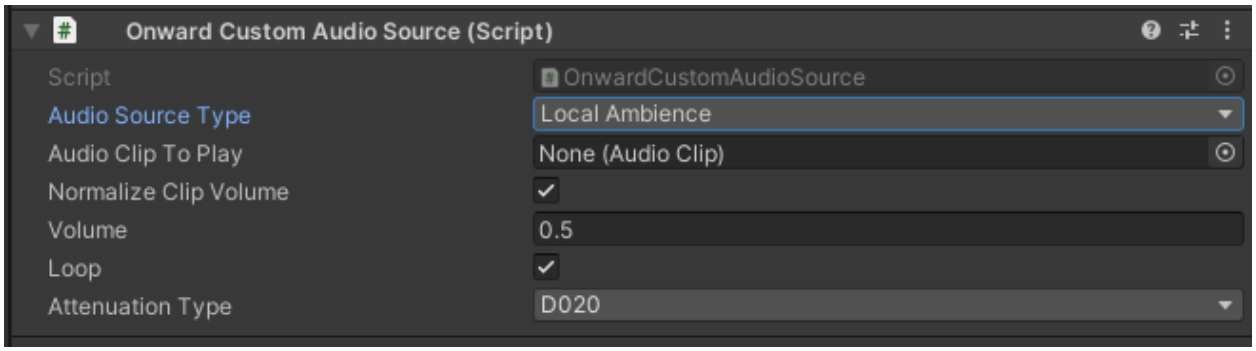
Audio Clip to Play is your own audio file you wish to play. Any audio type that is supported by Unity is allowed (.ogg, .wav, .mp3, .aiff/.aif, .mod, .it, .s3m, .xm). It is recommended to use .ogg since the compressed format will reduce the download size of the map.

Normalize Clip Volume normalizes the volume of the audio clip. This will essentially make the 0 to 1 volume setting consistent for all audio files.

Volume controls how loud the audio provided is.

Loop continues to loop the audio clip after it reaches the end.

Attenuation Type is the falloff distance for the audio source in meters.



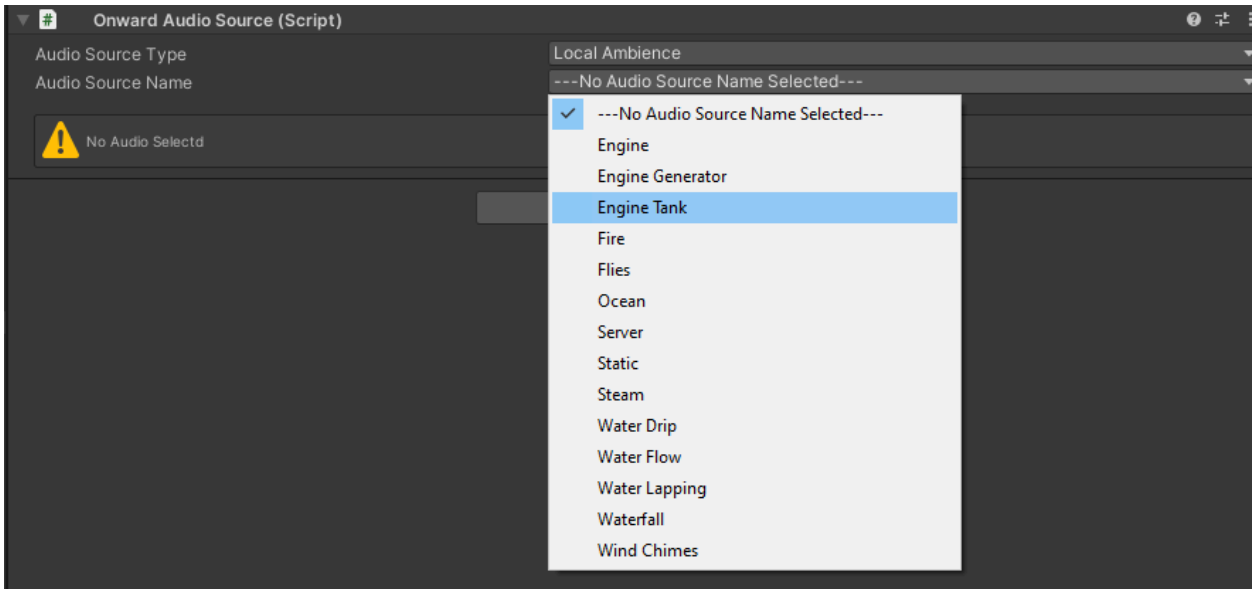
F.

OnwardAudioSource is a prefab that allows you to select audio used in the game and play it. Some good uses for it are to break up the silence in your level by adding ambience to it.

Audio Source Type refers to how the audio is heard. Local Ambience will play from a 3D spatialized position. Global Ambience will play throughout the entire scene as if you are in the center of it.

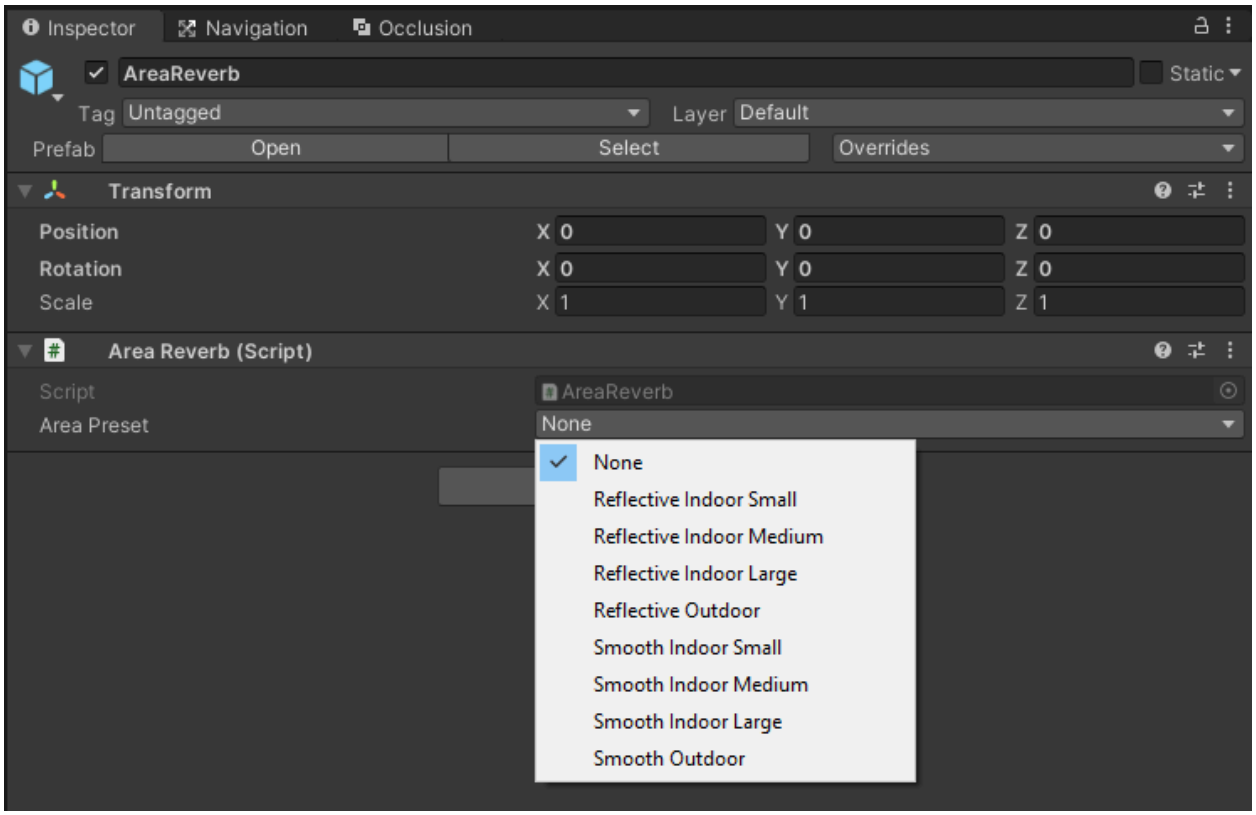
Audio Source Name is the type of effect you wish to play.

Use Onward Audio Source to give some character and ambience to your custom scene.

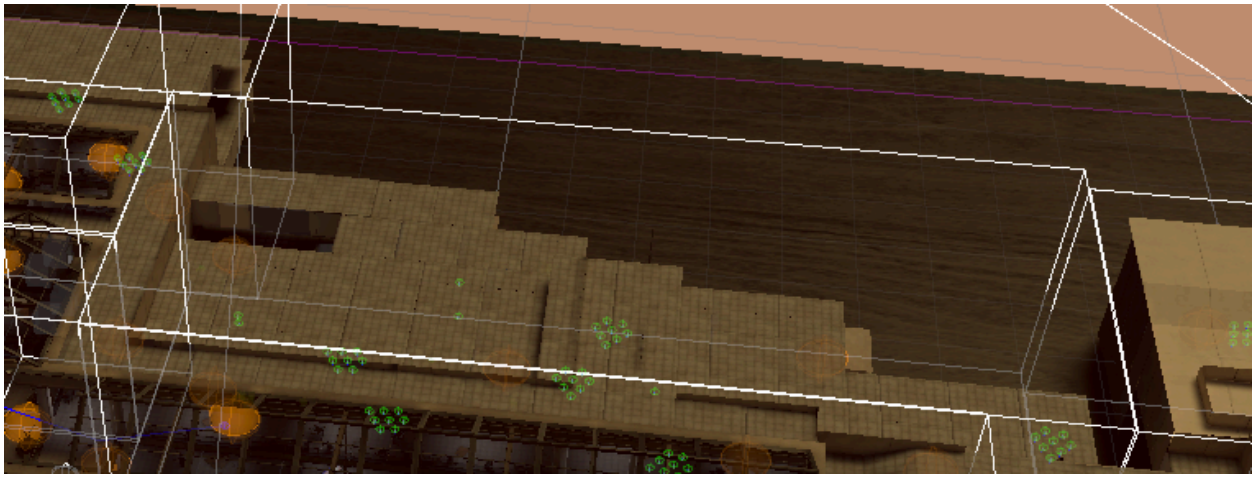


G.
AreaReverb

AreaReverb is a prefab that allows you to add different spatial effects to areas allowing for more realistic sound reverberations. Some examples are the Hallway preset for tight hallways, or the Auditorium preset for large open interior areas. Experiment with your local map versions to try different sound reverberations out and see what fits for you.



Place the AreaReverb prefab into your scene, then scale it to enclose the area you wish to have reverberations within and set the Area Preset to your desired effect.

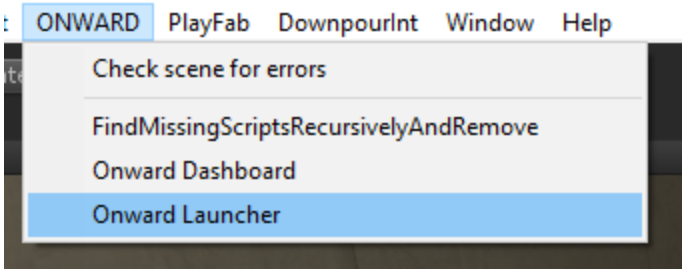


4. TESTING YOUR MAPS

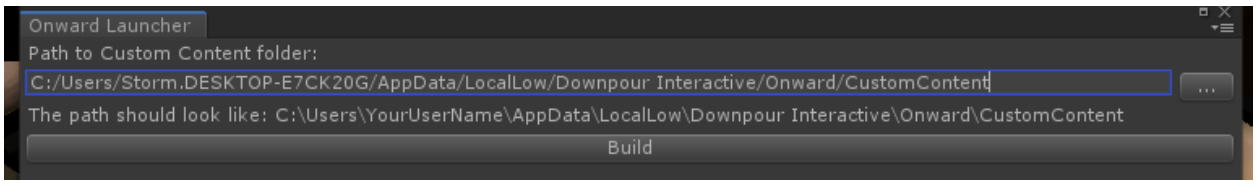
You can test your maps by using a local version and your installed Onward game client to test without uploading.

For publishing instructions, refer to *Section 7: Publishing*.
For local testing, refer to the instructions immediately below.

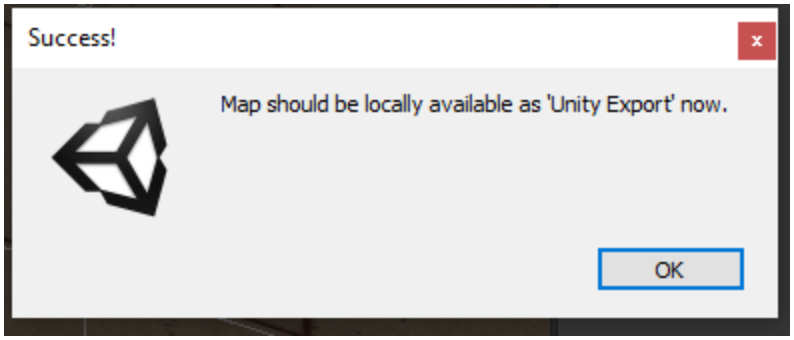
You will be able to test your maps without uploading them and waiting for approval. You can use the *Onward* dropdown for this. Select ***"Onward Launcher"*** and a new window will appear.



PC Testing -
Direct this window to the prompted file path (as shown in the screenshot) and press "Build"



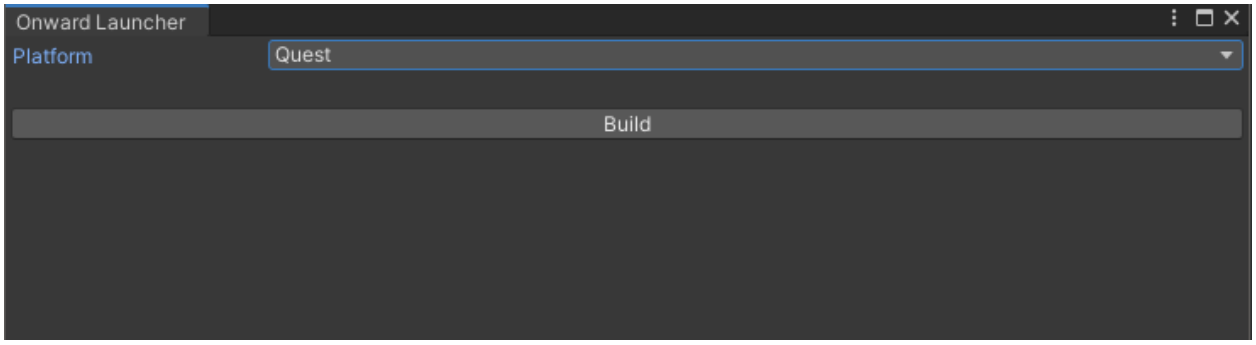
A progress bar will appear indicating the packaging of your map. Once it is complete, you will receive a success message.



After getting your success message, you can launch a locally-installed version of Onward and browse for your Custom Map using the server creator. It will be labelled as "Unity Export".

Locally-hosted Custom Content in the map selection menu.

Testing for Quest - Set your Platform to Quest and press “Build” You will be prompted where to save the files. Select a memorable folder location.

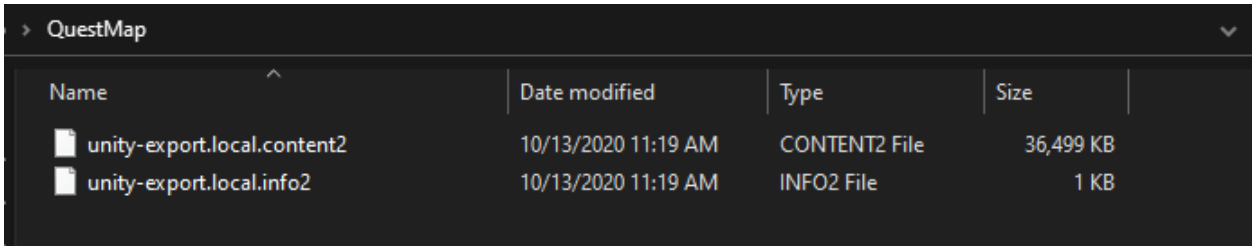


Upon Success you will see a message denoting where it is to be installed.

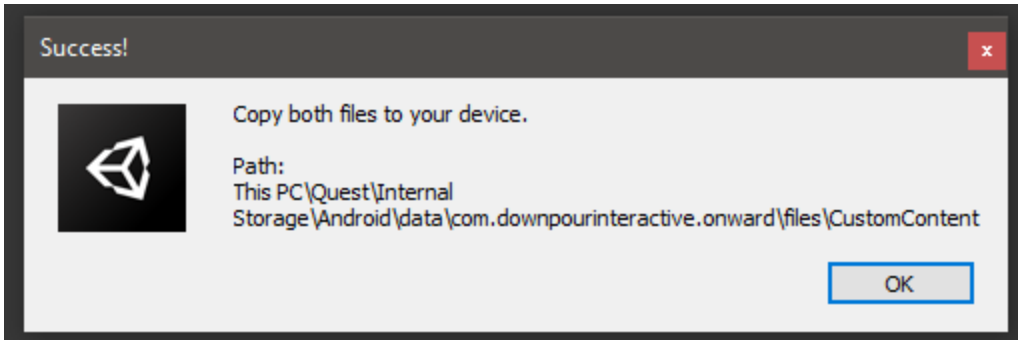
*To test on Quest you **MUST manually** add your map locally on the device.*

You MUST have Developer Mode enabled on your Quest device.

Plug the device into your PC with developer mode active, then browse the file structure and add the files created by the Onward Launcher Builder you saved to a memorable file location earlier to the directory below.



Add these files to:



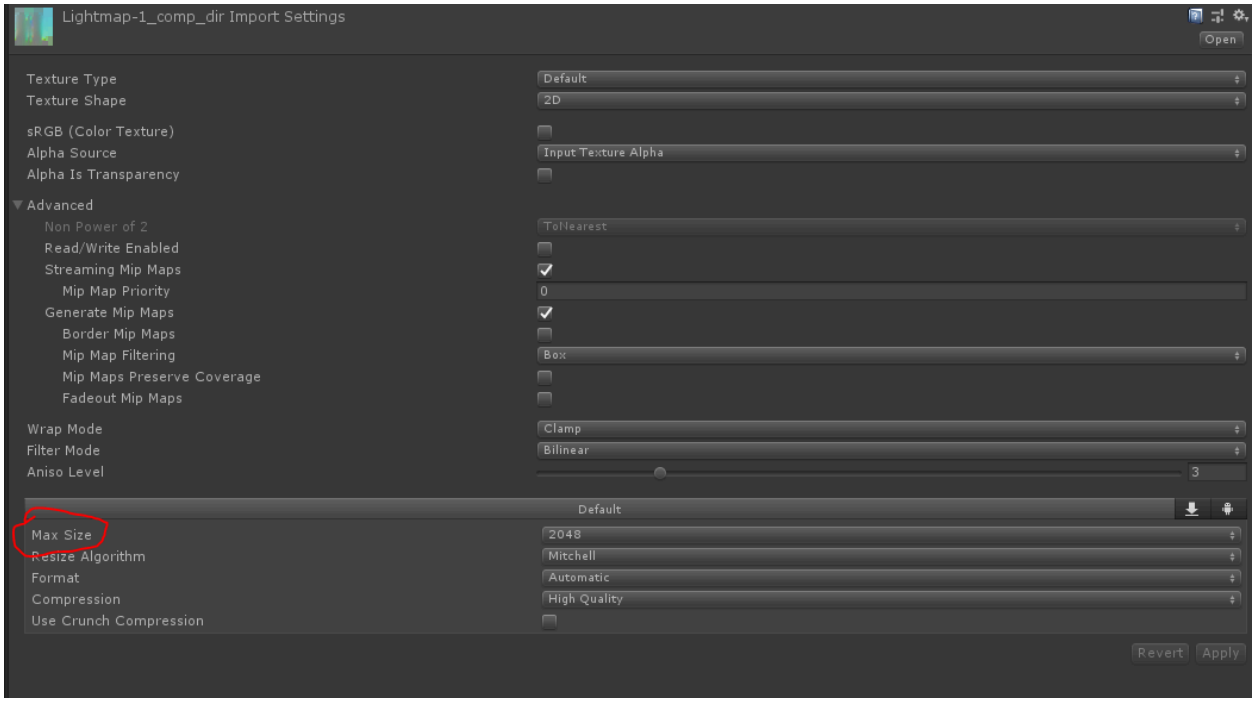
yourdirectory\Quest\InternalStorage\Android\data\com.downpour interactive.onward\files\CustomContent

From there, you can launch the map locally on your Quest device when the transfer is complete.

5. PERFORMANCE CONSIDERATIONS

PC performance considerations-

- Try to not exceed 750,000 vertices and 700 draw calls - the lower, the better.
- Make use of Occlusion Culling, Lightmapping and other performance tools in Unity.
- Resources for these tools and more can be found at Learn | Unity (https://unity.com/learn), or on YouTube
- If your submitted map suffers from major performance issues it may be denied.
- You should investigate your textures and scale them to a lower resolution if possible. The default resolution of many textures is in some cases egregious for the level of detail required and will only resulted in a bloated level size and slow loading times. You can inspect a given texture by clicking on it and choosing its max size. Reduce this from 4k to 2k then “Apply” and observe your texture in the scene. If the quality decrease was acceptable leave it, or even lower it further until its not acceptable- then raise it up one level from that.



Quest performance considerations-

QUEST ONLY USES (renders) LOD1+ models, LOD0 is excluded!

WARNING! Quest scenes must use extremely limited draw calls, on-screen triangle counts and texture memory to perform optimally. Creating a custom map for Quest is intended for advanced map creators. Your submission may be denied if it does not meet performance requirements and causes simulation sickness.

Do not exceed:
75 draw calls
85,000 triangles
350mb of texture usage (This means about 12 albedo+normal textures total)

These limitations account for players, objectives and other gameplay objects for you- so you can develop right to the edge of these specs if you wish.

Quick tips for developing Custom Content for Quest crossplay:
Usage of occlusion culling is *NOT* recommended for Quest scenes.
Rely on LoD and hierarchical LoD to cull geometry and keep performance manageable throughout your scenes.
Utilize clever line of sight to avoid large overview areas that will break your frame limits.
Atlas as many materials together as possible and combine meshes into groups to reduce draw calls.
Use the frame debugger to check out your draw calls and triangle counts at many given points throughout the scene by placing a camera in your scene and removing it before submission once you are satisfied with your results.
There are many tricks and techniques to make a lean, yet playable level and these are just a few of them. Respect the performance of the Quest platform and you will be rewarded with smooth gameplay and many highly-rated reviews of your custom content map.

6. DEVELOPMENT & BEST PRACTICES

Onward is a game with an established set of rules, based on realistic squad-level combat. You should keep this in mind while designing your maps and objectives to ensure an intuitive and fun gameplay experience for all players.

We highly recommend blocking out your maps with simple 3D shapes and playtesting them before finalizing your map design. This will give you a better understanding of how your map feels, and what sections need to be modified before you start polishing it visually.

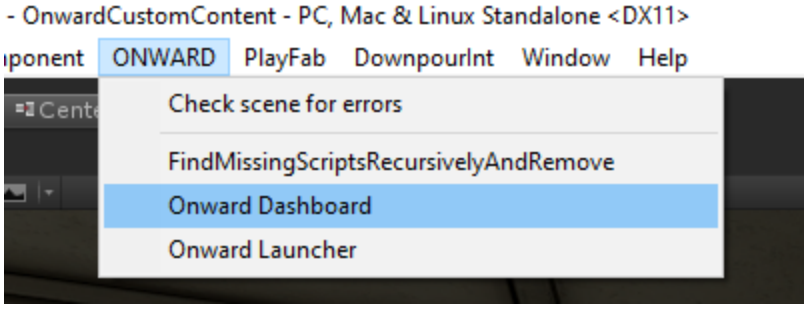
Regular playtests are a great way to see how your map evolves. They help you locate areas where players might have difficulty moving through the map, lines of sight that may be too short/long, and spots where lighting issues may occur.

Lastly, getting in touch with other map creators gives you the opportunity to share experiences. You can pick up tips and tricks from more experienced creators, or share your own ideas with others looking for help. Constructive feedback is one of your most useful tools.

7. PUBLISHING

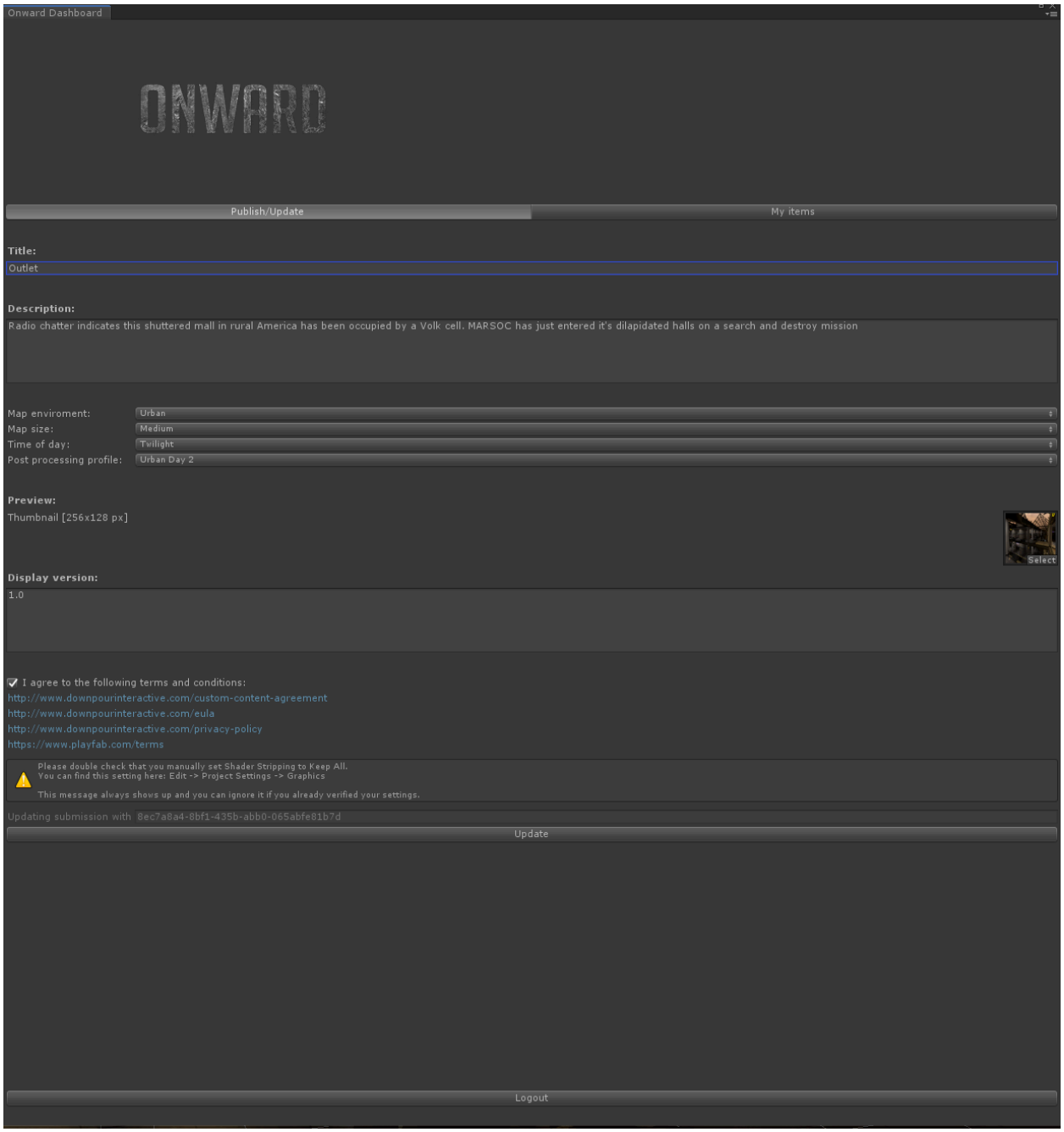
This is what you've been working for - once you've set up your map, built it in Unity and tested it thoroughly, it's time to publish it!

First, in the Unity top menu, open the **Onward Dashboard**.



Take a moment to read and accept the **Terms of Service**. If you have not already done so, you will need to create a new account. Choose **Create New Account**, then follow the necessary steps.

Once you've created your account, select **Existing Account and login**. Now you will be able to enter your map details in the window below.



Make sure you include a clear title and description. We recommend following the suggestions below to ensure that your information displays correctly:

- **Title:** Try not to exceed 24 characters - **~6-18 is ideal**
- **Description:** Try not to exceed 500 characters - **~250 is ideal**
  Map Environment: Change this to match your map.

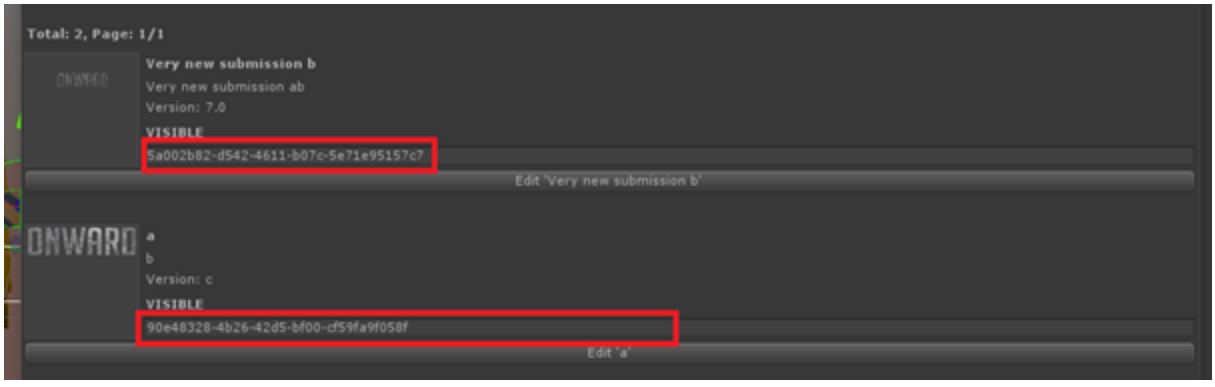
Map Size: Change this to match your map.
Time of Day: Change this to match your map.
Post Processing Profile: OPTIONAL setting, covered in *Section 3: Additional Gameplay Components*.

- **Thumbnail:** 512px wide by 256px tall. **No exceptions.**
- **Display Version:**
    - For a finished map, use 1.0
    - for maps still in development, we suggest *0.1, 0.2 etc.*

Once you have filled out the information, a **Publish** button will appear. **Click it to begin the submission process.** Depending on the size of your map, this may take some time. Once it is complete, you will be notified with a 'Success' message. *(If you receive errors in your console log or any pop-up error message, take time to review the message and determine the issue with your scene, take steps as advised to fix the issue, then try again. The Onward Custom Content bundle will detect errors in your setup automatically and try to help you fix them.)*

After your submission is sent, you will need to wait for approval. Approval may take up to 2 weeks. If you are denied, you will be notified.

If you encounter any issues with your submission that you cannot solve on your own, please provide us with the submission ID for your map. You can find it in My Items, just above the Edit button.



Once you've published your submission and been approved, you can update it to submit map changes and improvements. The button at the bottom of the **Publish/Update** window will now read **Update**. Click it to submit any revisions. **Once your map is available to the public, use local testing (Section 4: Testing your Maps) to test major changes before deploying them to the live servers. This will help prevent issues for users if you accidentally break some of your map's functionality. *Reserve uploading and publishing updates to your map for builds which you are certain function correctly.***

CONCLUSION:

Thank you for reading through this guide, and for submitting your custom Onward map(s). We're excited to present them to the community, and we appreciate the time and effort you invested to create new experiences for your fellow players.

If you have any issues submitting your custom content, or if you find any errors in this guide, please let us know via the Onward Discord channel so we can address them. Your feedback will help us provide the best experience for creators and players alike.

Thanks again for your support.

Onward!

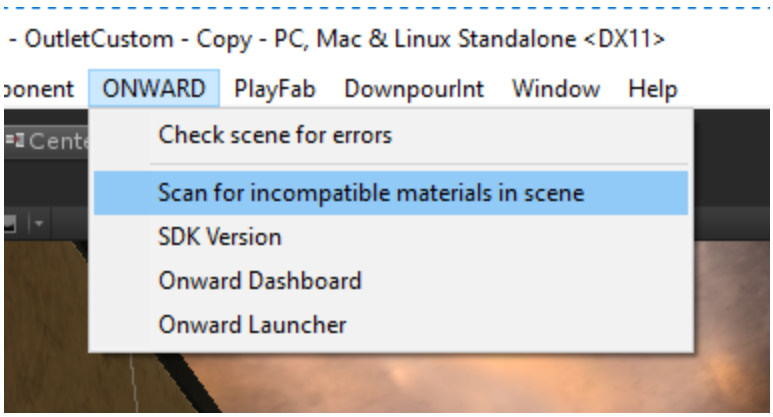
-Downpour Devs

UPGRADING YOUR MAP FOR UNITY 2019 and Universal Render Pipeline:

Using the **Shader Compliance Upgrade tool**:

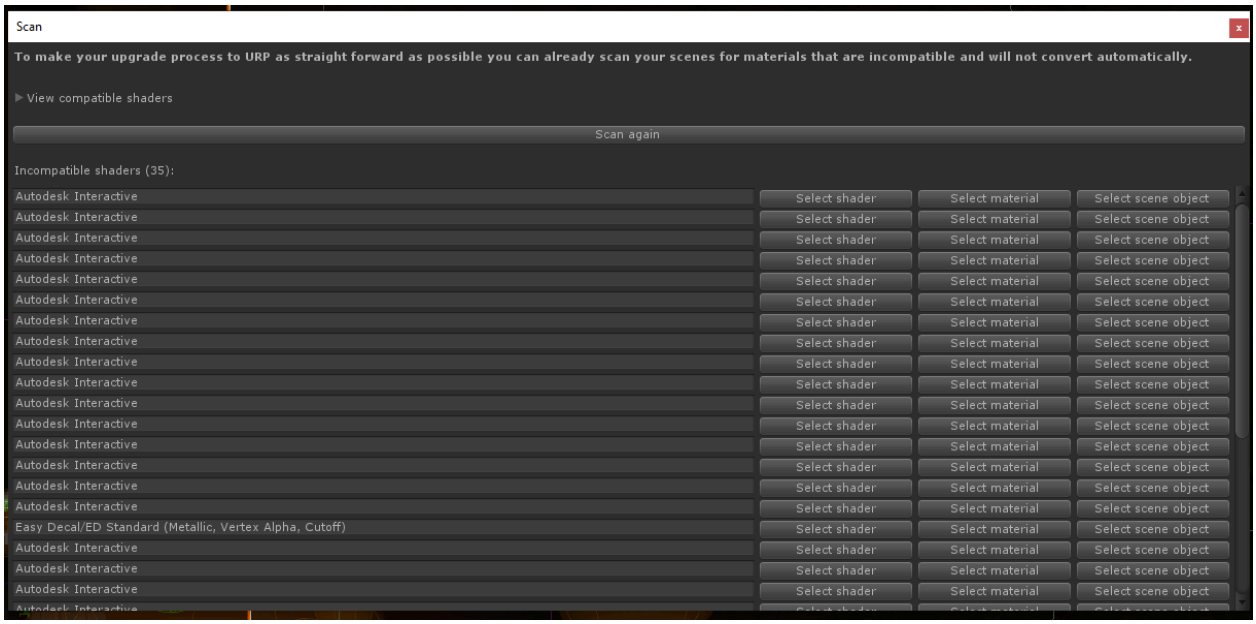
Making your scenes compliant for the upcoming Universal Render Pipeline upgrade with Unity 2019 is simple with our **Shader Compliance tool**. *It will allow you to locate shaders which are not compliant with being converted automatically to a URP shader in the later Unity 2019 upgrade that comes with* **Onward version 1.8.X** (***currently In development***).

After locating your problem shaders, you can easily swap your materials to a URP-convertible shader.



Go to *ONWARD > Scan for incompatible materials in scene*

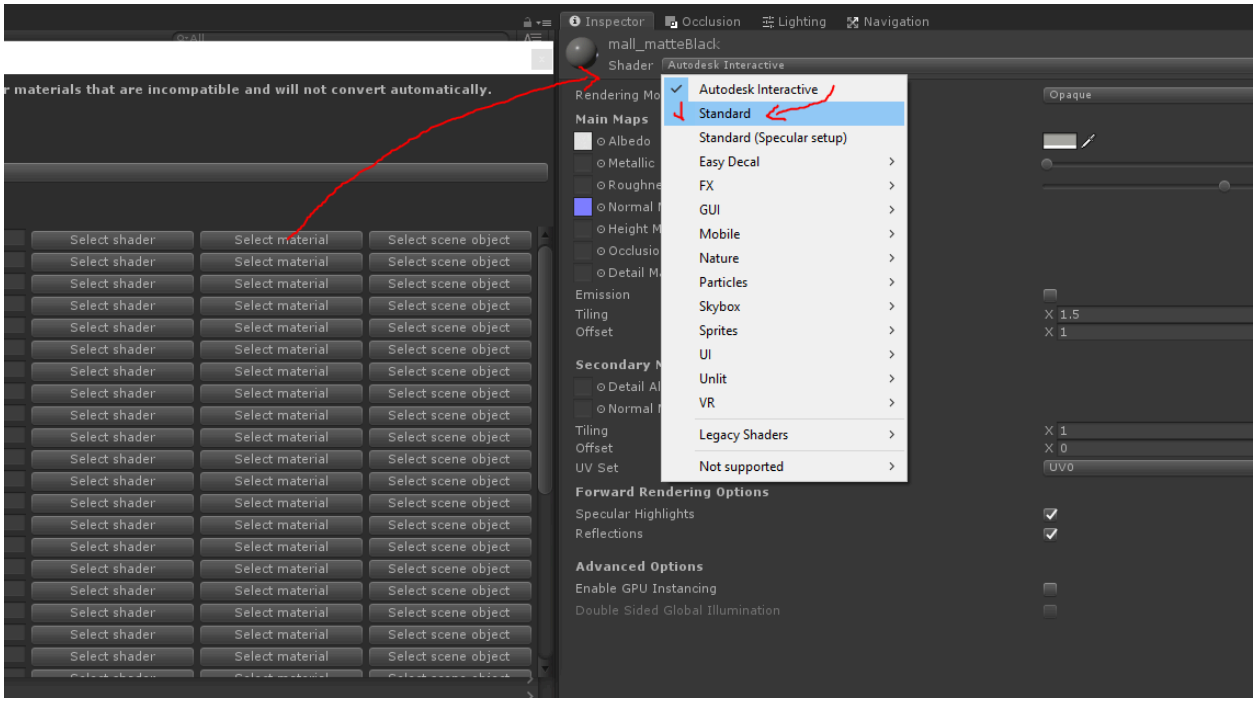
A new window will appear, press the **Scan** button to scan your currently opened scene for non-compliant shaders. You will get a list of them similar to what is seen below:



There are 3 options. **Select Scene Object**, **Select Material** and **Select Shader.**

Select Scene Object - *This will select the object in scene that uses a material with a non-compliant shader.* Using this you can focus on an object while you manipulate the shader on it's material.

Select Material - *This will select the Material of the given object, allowing you to change its shader.* **IMPORTANT: Change your shader on each material to a Standard or otherwise URP-compliant shader. See example below and further for a list of URP-convertible shaders you can use.**



*Change the material's current shader to **Standard**, or another type listed below.*

Here is a list of URP-convertible shaders:

Standard
Standard (Specular Setup)
Standard Terrain
Particles/Standard Surface
Particles/Standard Unlit
Mobile/Diffuse
Mobile/Bumped Specular
Mobile/Bumped Specular(1 Directional Light)
Mobile/Unlit (Supports Lightmap)
Mobile/VertexLit
Legacy Shaders/Diffuse
Legacy Shaders/Specular
Legacy Shaders/Bumped Diffuse
Legacy Shaders/Bumped Specular
Legacy Shaders/Self-Illumin/Diffuse
Legacy Shaders/Self-Illumin/Bumped Diffuse
Legacy Shaders/Self-Illumin/Specular
Legacy Shaders/Self-Illumin/Bumped Specular
Legacy Shaders/Transparent/Diffuse
Legacy Shaders/Transparent/Specular
Legacy Shaders/Transparent/Bumped Diffuse

Legacy Shaders/Transparent/Bumped Specular
Legacy Shaders/Transparent/Cutout/Diffuse
Legacy Shaders/Transparent/Cutout/Specular
Legacy Shaders/Transparent/Cutout/Bumped Diffuse
Legacy Shaders/Transparent/Cutout/Bumped Specular

Select Shader - *This will select the given shader asset itself.* Only use this if you plan to convert features of the non-compliant shader to URP-compatible versions on your own and you plan to support a custom URP-compliant shader. This is **NOT** for the average creator, **if you do not understand what you are doing when editing and re-assigning shaders in Unity 2019 with Universal Render Pipeline then ignore this option.**

Using this tool, work through the list replacing all non-URP-convertible shaders in your scene.

Once you've done this process, **save your scene** and *continue to work on it or upload it as you usually would*. **Thanks for your help in keeping your maps compliant with our upgrade, *Onward is still a work in progress* but we are anticipating that this is the last major upgrade process creators will have to participate in to remain compliant with our workshop.**

NEW: UPDATING YOUR MAP TO WORK WITH ONWARD WORKSHOP V1.8.5+

Updating your existing map to work with Onward Workshop v1.8.5+ is a relatively simple process and we have streamlined the steps so you can get back to what matters most - creating content and enjoying it with your fellow players.

Upgrading your project consists of 4 steps:
1. Upgrading your Unity version/opening your project.

2. Installing Universal Render Pipeline package, Custom Content SDK upgrade and assigning a render pipeline asset.

3. Converting your conversion-compliant materials to URP materials.

4. Converting your URP materials to Onward Shadergraph shaders.

5. Setting up Level Descriptor and adding scenes for upload.

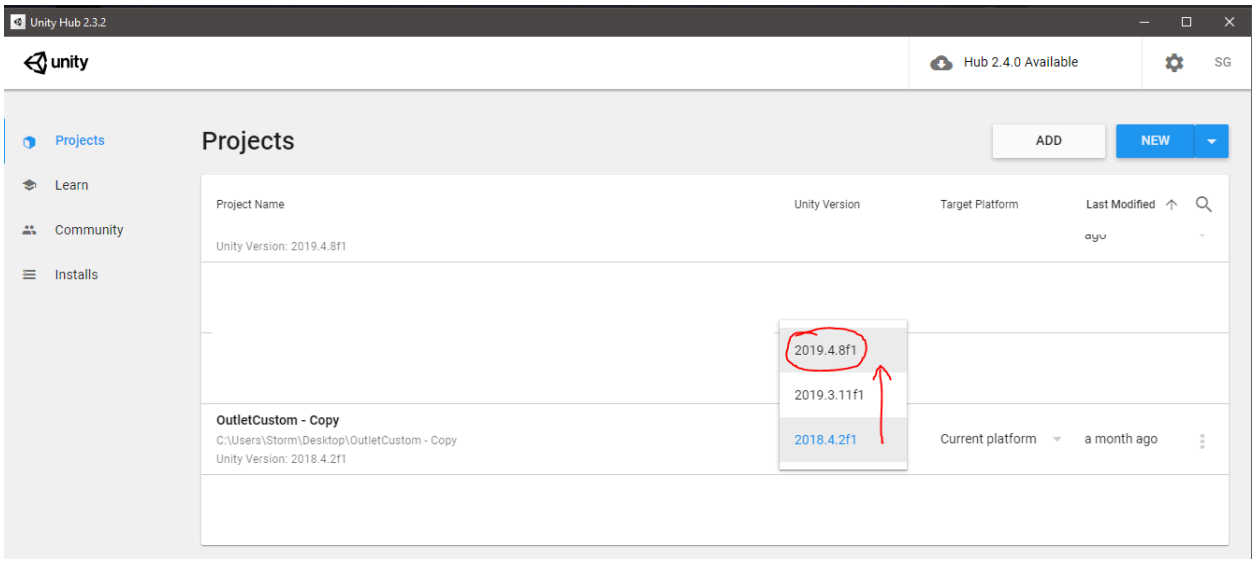
---

1. Upgrading your Unity version/opening your project -

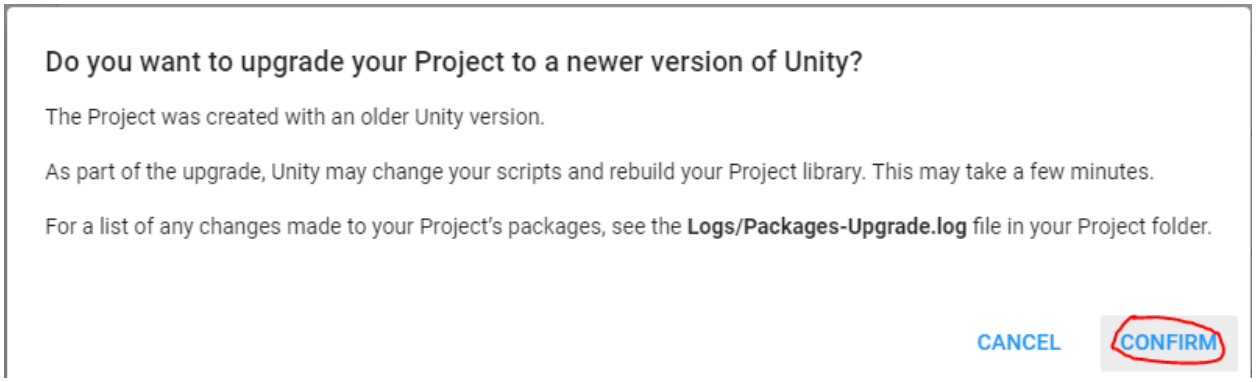
First, you will need to upgrade your Unity engine version to **2019.4.8f1**. This is the long-term support version of Unity and it is intended to be used for the rest of the Onward 1 lifecycle, so there will be no further upgrades to your engine anticipated in the future to continue supporting custom maps.

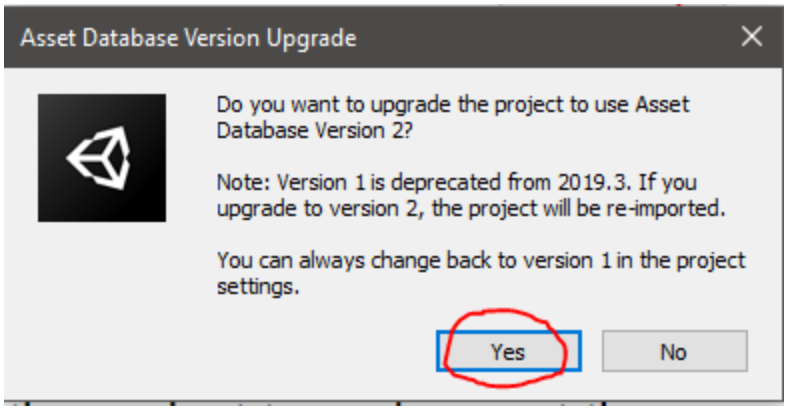
Update your Unity version to 2019.4.8f1 and be sure to INCLUDE Android Build Support when asked about packages to install with the editor.

Once you've updated, now you will be able to follow below to upgrade your project.

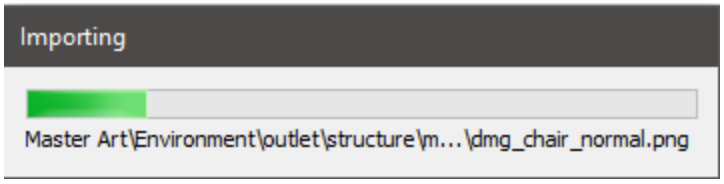


For example, we're using a copy of our official custom map: **OutletCustom - Copy**. Simply select the **Unity version 2019.4.8f1** and **accept** the upgrade prompt after you've selected 2019.4.8f1. Also **accept** the secondary prompt that asks about upgrading to Asset Database v2.



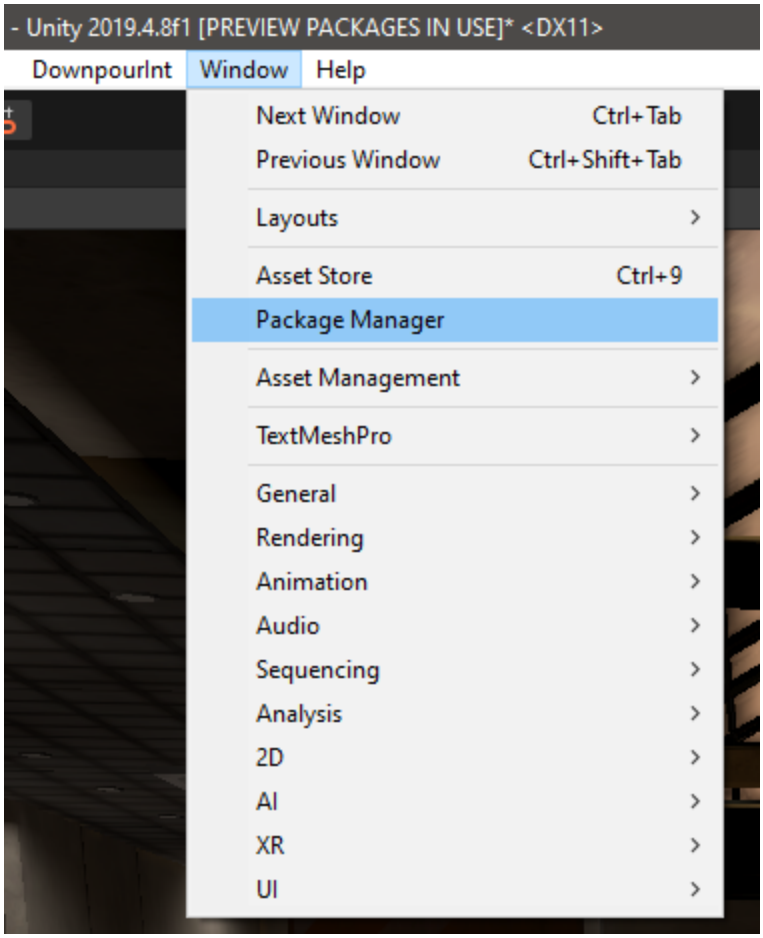


Your project may take some time to upgrade and re-import depending on its size, once it is complete Unity will open.

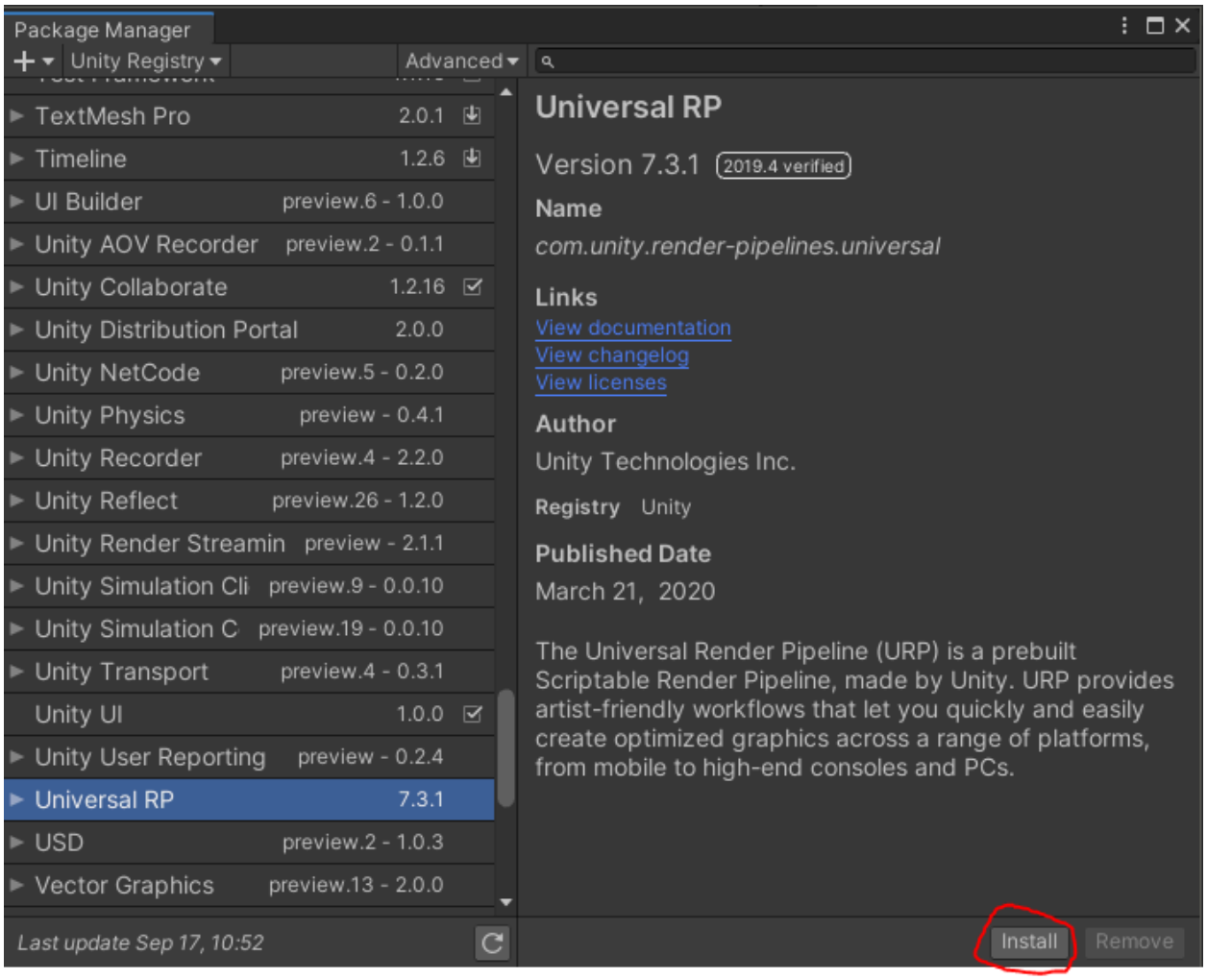


Now you're ready for step 2.

2. Installing Universal Rendering Pipeline, Custom Content SDK upgrade and configuring project.



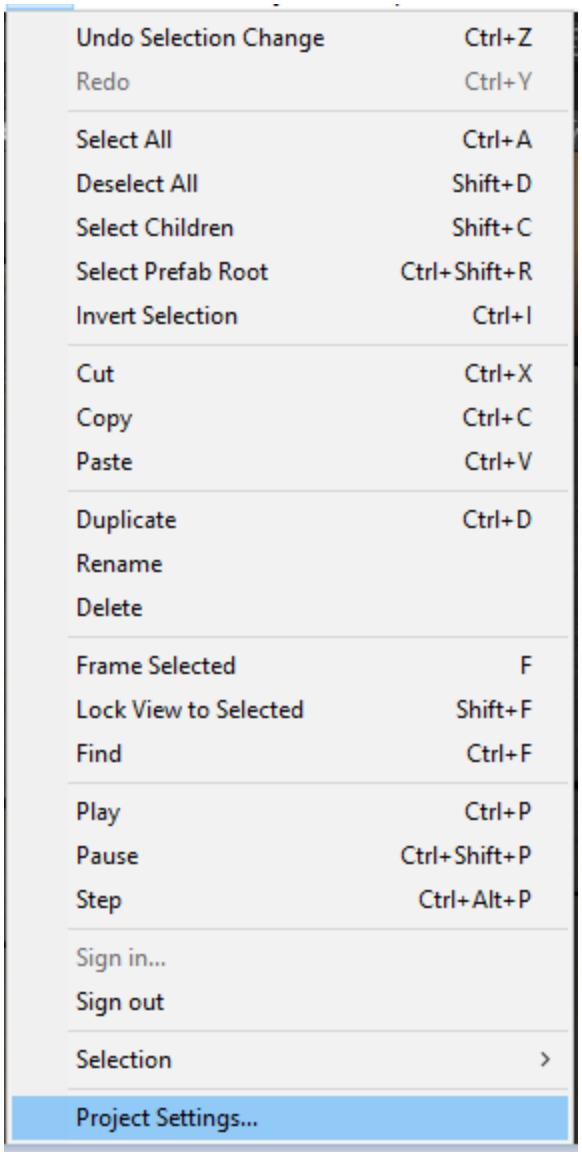
Open the package manager with Window>Package Manager and find “Universal RP” in the available packages. Install it.



Once Universal RP is installed, it will do a short material upgrade process, this is normal.

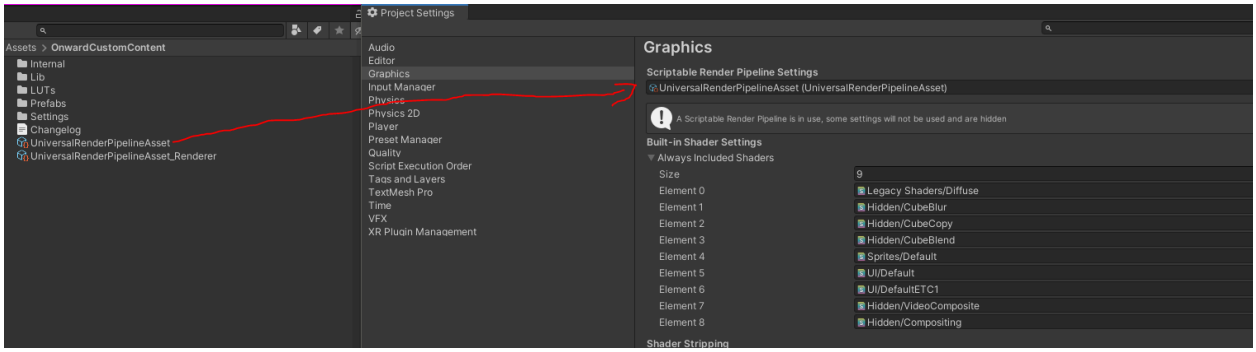
2b.
Now, import your latest version 2.0.0 of the Onward Custom Content Package to your project.

After the upgrade process, you will now need to set a few new graphical settings and swap some shaders to ones included with the Custom Content Package.



Open your **project settings** window under **Edit>Project Settings** and navigate to the **Graphics** tab.

Once there, locate the **UniversalRenderPipelineAsset** file located in the root folder of **Assets>OnwardCustomContent** and drag it into the **Scriptable Render Pipeline Settings** field of the Graphics tab in the project settings window.



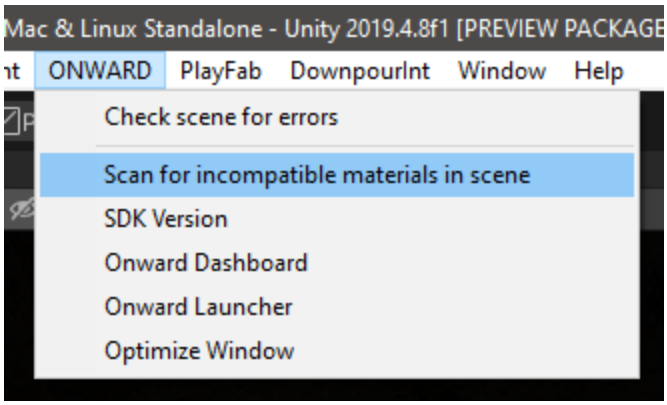
This will make your scene pink due to unsupported shaders- don't worry. We will change these shaders to proper ones now using Unity's built in URP converter tool.

WARNING! If you have a lightmap, you may see materials as odd colors until you remove and rebake the lightmap.

Now you're ready for step 3.

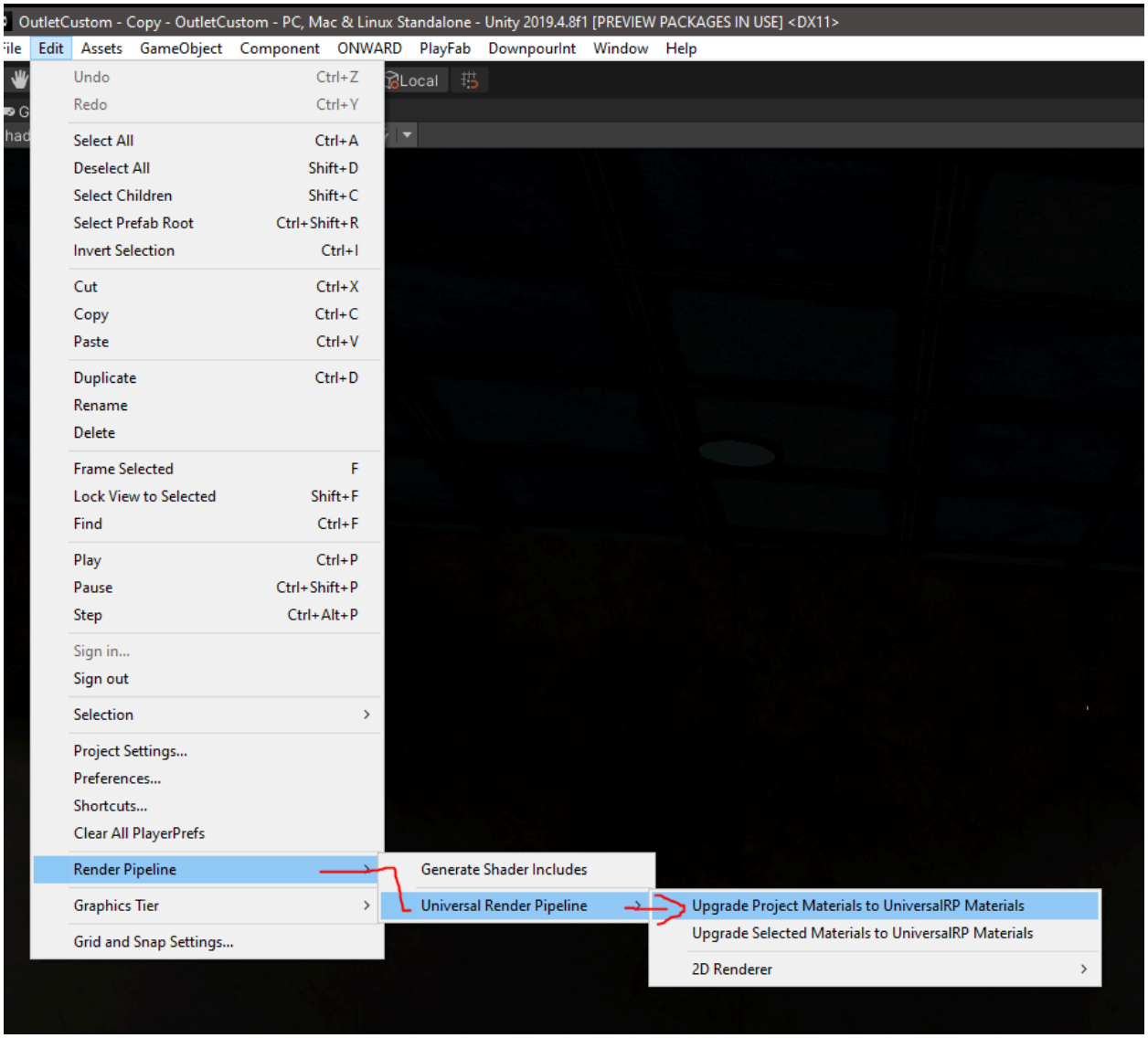
3. Upgrading your materials to URP versions.

WARNING! If you are using non-conversion-supported shaders, you may have missing texture references on your materials you will have to fix. Use the *Incompatible Material Scanner* to scan for incompatible materials before you use Unity's URP material upgrader tool or else you will have broken materials on those using non-supported shaders.



See *UPDATING FOR UNITY 2019 AND UNIVERSAL RENDERING PIPELINE* on page 42 for details on how to swap your non-convertible materials.

Once you've ensured your materials can be converted easily, use Unity's built in URP material upgrader and it will automatically update your materials in the project to URP compatible versions,. If you still have a pink texture here or there, that shader was not a URP-convertible shader and must be fixed--then you can redo the conversion step selecting individual materials.



Go to Edit>Render Pipeline>Universal Render Pipeline>Upgrade Project Materials to UniversalRP Materials.

After this completes, your scene should look as it did before once again. **Now you are ready for step 4.**

4. Assigning Onward ShaderGraph Shaders to URP materials.

Once your conversion is complete, now you must assign the shader of your new materials to an Onward-compliant version. **There are a number of Shadergraphs included with the Custom Content SDK. All of them with the prefix PP_.** *You can find them in Shader Graphs dropdown on the shader selector on a given material.*

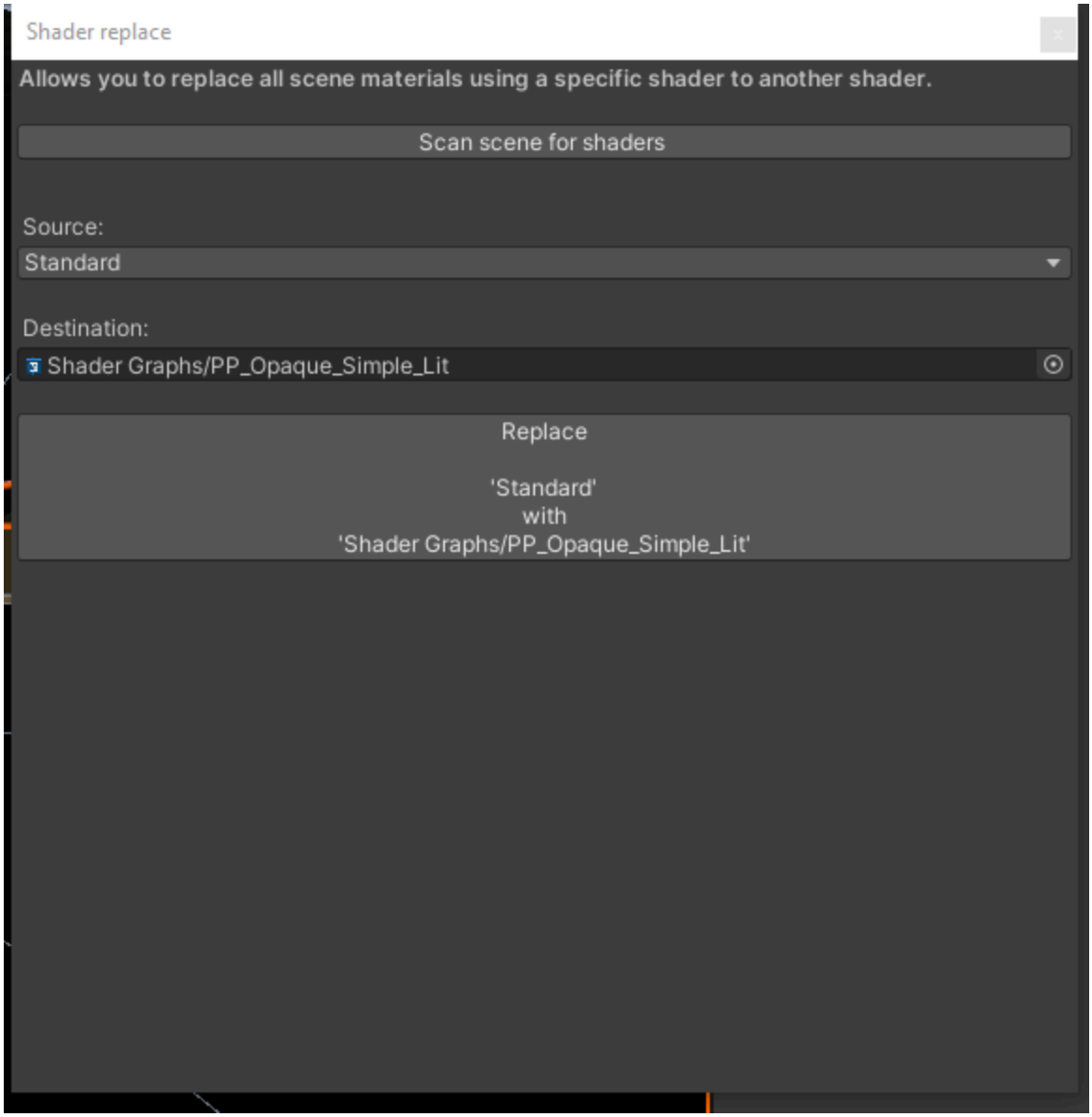
WARNING! *You **MUST** use Onward PP_ ShaderGraph shaders on your materials or they will not function correctly and post processing effects will not render properly. This is grounds for map removal from the workshop.*

Note: **Lit variants will be converted to Simple Lit variants for Quest.**

Lit for PC has support for Albedo, Normal, Specular and Metallic maps
Simple Lit for Quest only has support for Albedo and Normal maps.

PP_Terrain_Lit - Terrain shader.
PP_Opaque Lit - General purpose shader.
PP_Transparent Lit - General purpose transparency shader.
PP_Particle Lit - General purpose particle shader.
PP_Sprite Lit - General purpose sprite Shader.
PP_Cubemap_Skybox - to be used with the Skybox prefab for performant skyboxes on Quest. Takes a cubemap.

Included in the Custom Content SDK is a shader remapper tool.



Press “Scan scene for shaders” and you will get a dropdown list of shaders that exist in your scene. Under Destination you can select the shader you wish to convert to.

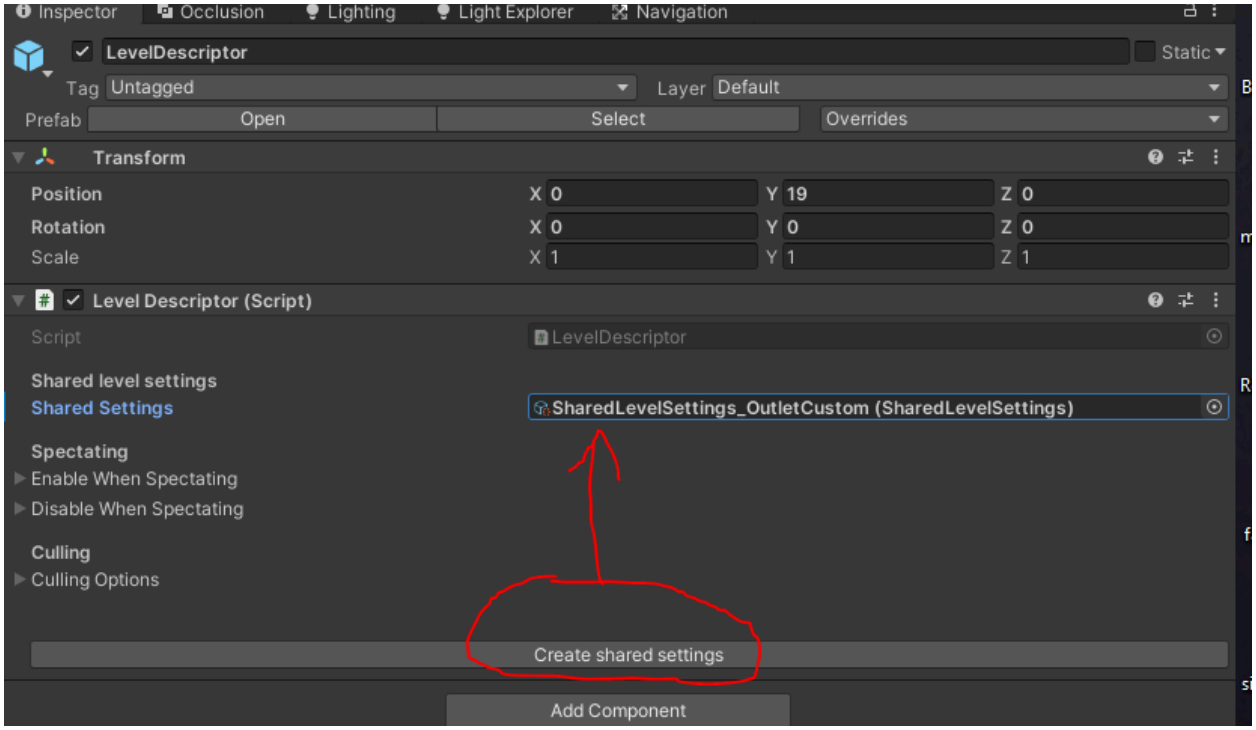
Once your fields are full, press “Replace” and your shaders will swap automatically.

Continue to use the Shader replace tool to remap your shaders until you are using all PP_XYZ Onward shadersgraph shaders. Once you are finished, you can now progress to step 5.

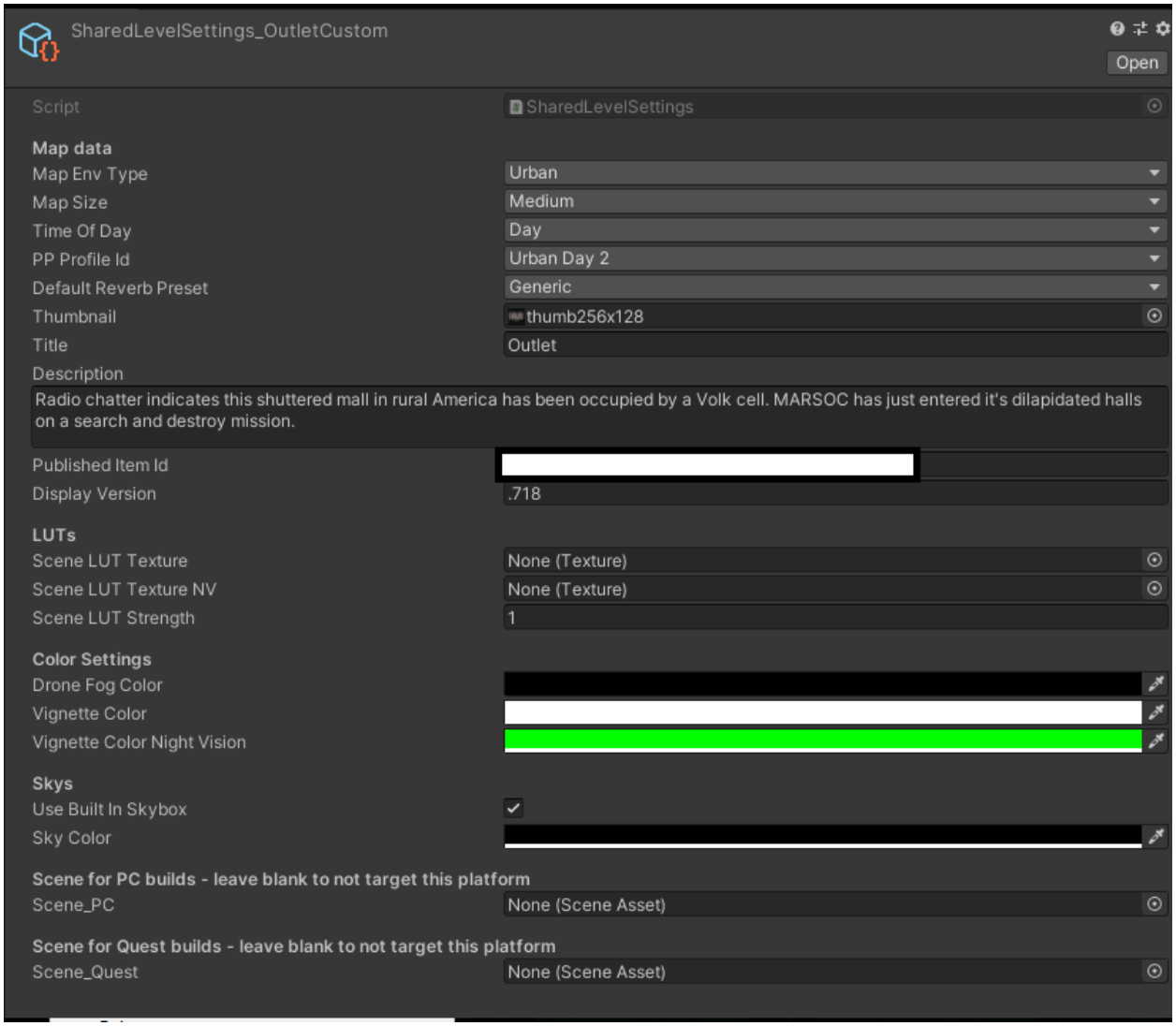
5. Setting up Level Descriptor and adding scenes for upload.

Now that your scene is converted, there are just a few more steps to submitting your map for PC and Quest.

Find your **LevelDescriptor prefab** in your scene. On it will be a new button called Create Shared Settings. Your Shared Settings field will be empty by default until you press the Create Shared Settings button. Press it.



After you've pressed the button and a new SharedLevelSettings_YourSceneName asset file has been created, click on the reference to be brought to the options panel for your Shared Level Settings as seen below in the example



This is where your LevelDescriptor parameters are now stored.

There is a final set of fields at the bottom which are the last step of restoring your map's functionality, here are some details about them:

Scene for PC builds - This is where you will drag a *scene reference* for your PC version of your map. PC Maps should use the **Universal Render Pipeline > Lit shader** on materials for full material map support. ***URP Lit supports Albedo, Normal, Specular and Occlusion maps + Emission.***

PC scenes can have more detailed lightmapping, texture use, high LoD models and more due to their higher headroom. If you are creating your first custom map, start off with submitting only a PC scene as they are more lenient on performance requirements than Quest.
Here is a quick rundown of specs you should try not to exceed if you wish to have solid performance on the PC platform:
700 draw calls
850,000 triangles
3.5gb of texture usage.

These limitations account for players, objectives and other gameplay objects for you- so you can develop right to the edge of these specs if you wish.

Scene for Quest builds - This is where you will drag a *scene reference* for your Quest version of your map. Quest Maps should use the **Universal Render Pipeline > Simple Lit shader** for performance support. ***URP Simple Lit only supports Albedo, Normal and Specular mapping + Emission.***

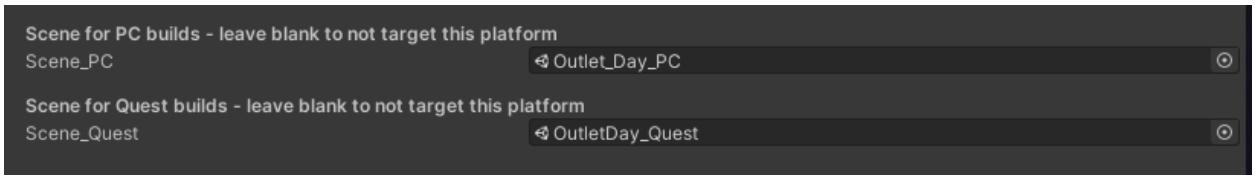
QUEST ONLY USES (renders) LOD1+ models, LOD0 is excluded!

NEVER USE A **PC-INTENDED SCENE** REFERENCE IN **'SCENE FOR QUEST BUILDS'** FIELD. *THIS WILL RESULT IN YOUR SCENE CRASHING THE DEVICE AND YOUR SUBMISSION WILL BE REMOVED.*

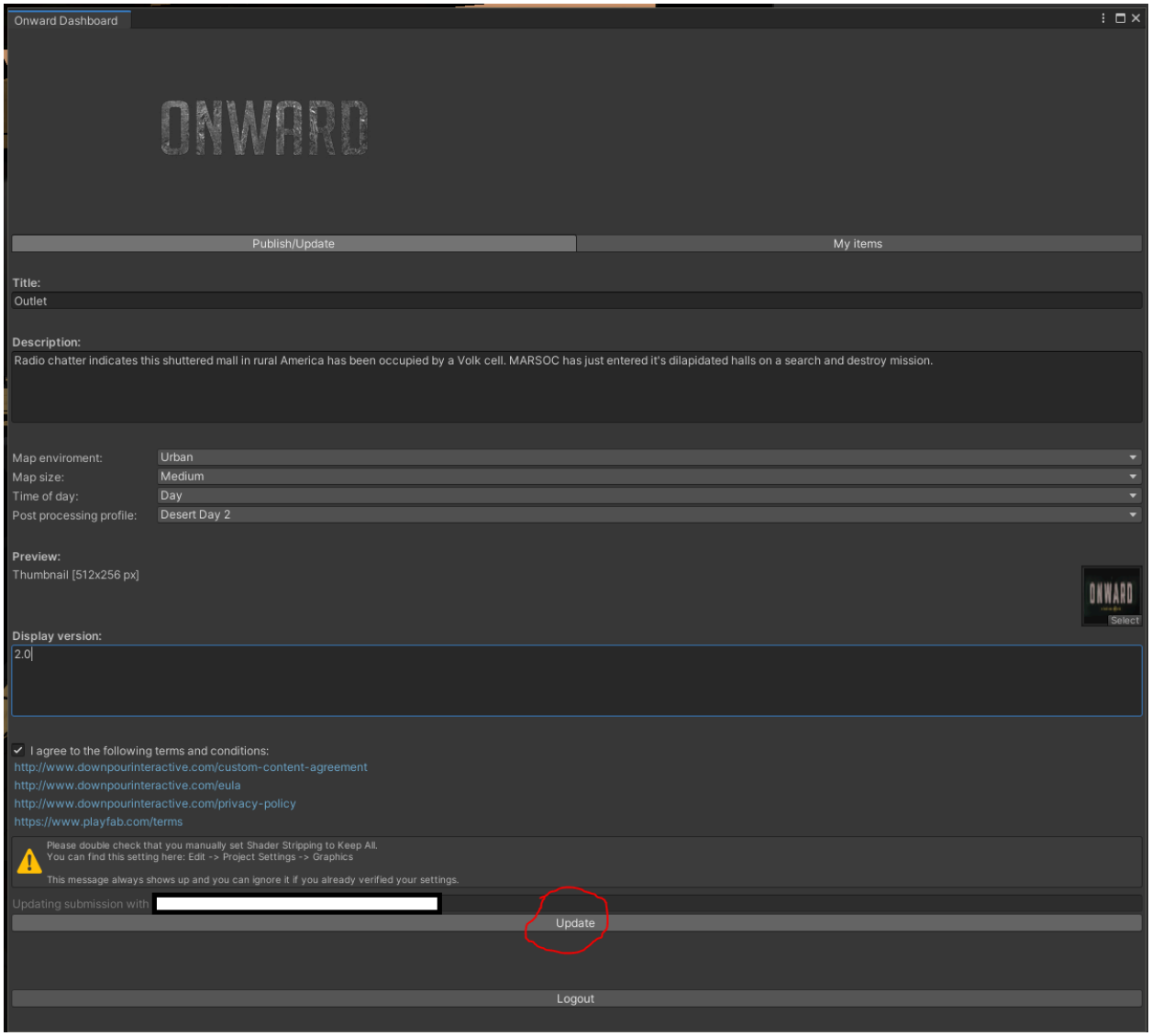
WARNING! Quest scenes must use extremely limited draw calls, on-screen triangle counts and texture memory to perform optimally. Creating a custom map for Quest is intended for advanced map creators. Your submission may be denied if it does not meet performance requirements and causes simulation sickness.
Here is a quick rundown of specs you should try not to exceed if you wish to have solid performance on the Quest platform:
75 draw calls
85,000 triangles
350mb of texture usage (This means about 12 albedo+normal textures total)

These limitations account for players, objectives and other gameplay objects for you- so you can develop right to the edge of these specs if you wish.

Quick tips for developing Custom Content for Quest/PC crossplay:
Usage of occlusion culling is *NOT* recommended.
Rely on LoD and hierarchical LoD to cull geometry and keep performance manageable throughout your scenes.
Utilize clever line of sight to avoid large overview areas that will break your frame limits.
Atlas as many materials together as possible and combine meshes into groups to reduce draw calls.
Use the frame debugger to check out your draw calls and triangle counts at many given points throughout the scene by placing a camera in your scene and removing it before submission once you are satisfied with your results.
There are many tricks and techniques to make a lean, yet playable level and these are just a few of them. Respect the performance of the Quest platform and you will be rewarded with smooth gameplay and many highly-rated reviews of your custom content map.



Once you have slotted one map or both maps, you can use the Onward Dashboard to re-submit your creation for release as you have in the past. Don’t forget to increment your version number to reflect this update.



WARNING: Submission may take longer than past iterations due to uploading 2 separate map versions.

Congratulations, you have converted and uploaded a v1.8 version of your maps for both PC and Quest!

Optimization tools and tricks for Quest:

Below are a few tools and features that will help you optimize your maps for Quest if you choose to publish for Quest.

OVRMetricsTool

This is a handy tool you can sideload on your device that allows you to monitor different performance metrics. Once it is installed, open it from your 'unknown sources' list and a new window will appear with settings for OVRMetrics. You will want to 'enable persistent overlay' and switch it to the 'advanced' view.

From there, the main metrics you will want to check out other than FPS are CPU/GPU usage and App T (GPU time per frame in Microseconds).

CPU Usage is NOT accurate to the performance you will see however, as when you are at ~70% CPU utilization you will begin to notice frame drops. This could be inferred as the cap. CPU Usage will get too high if you are processing too many draw calls. Reduce your draws to fix problems here. Complex shaders will also cost in CPU, so simplify custom ones where possible.

GPU however is more traditionally monitored and the danger zone(cap) is ~90-95% utilization. Hitting GPU limits means your scene is too geometrically complex or requires too much intensive shading. Reduce triangle counts and reduce shader complexity if using shader graph custom shaders. Another issue you can see is pixel fill issues from poor z-testing. You can alter the render queue of materials manually to help alleviate these issues. Use the frame debugger to see how your frame is drawing step-by-step and detect objects for special render queue ordering. Typically Unity does a good job of this by itself but there are always outliers.

App T (GPU time) above 12,700 is dangerous (12.7MS) and should not be maintained for long or else Fixed Foveated Rendering (FFR) will become enabled- which reduces the resolution of the images in the periphery of a user to reduce GPU workload. If your App T retreats below 13,000 after FFR is enabled, it will be disabled in a couple seconds of in-spec workload. Try to keep your App T constantly below 12.7MS to avoid triggering FFR unnecessarily.

Optimize Window

This is a handy tool we have added to version 2.0 of the custom content package that allows you to see high-triangle count meshes in a list that will help with identifying assets for decimation or further level of detail usage. Please use it as you see fit to help optimize your Quest scenes.