World Partition Re-Evaluation (April ‘24 - UE5.4)

Owner	Matt Canei
Tags	

TL;DR

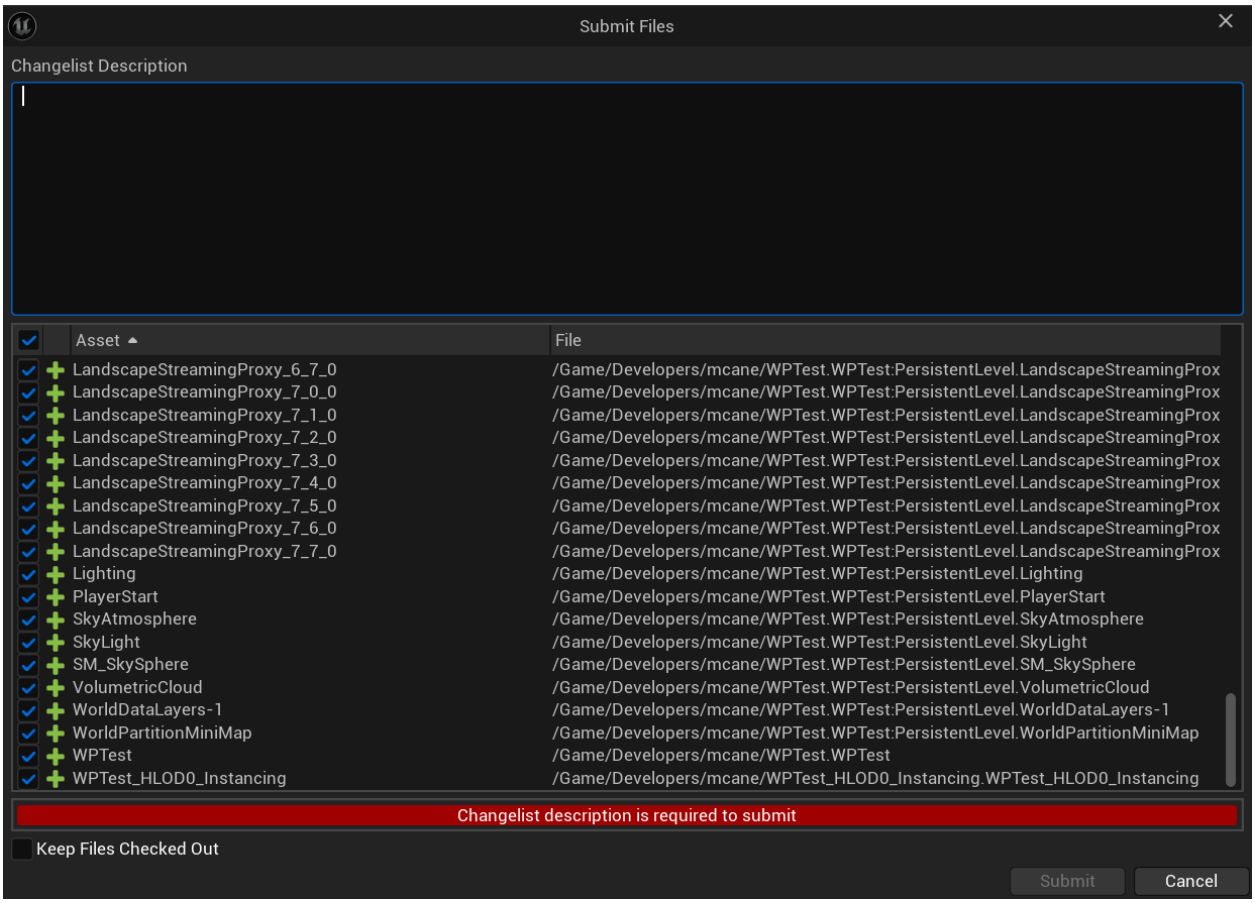
World Partition is mostly ready for everyday use in UE5. It has some quirks to work around and we definitely need to make some custom improvements, especially OFPA GUID names on disk/friendshipper, and certain organizational tools to really make it work for us at scale. Organizational discipline is paramount to having huge worlds be workable for everyone. Most of the shortcomings of this workflow are ALSO shortcomings of the legacy workflow (or require heavier lifts such as custom level streaming).

The goal of this document is to re-evaluate the current state of World Partition in UE5.4 to see if maturity of feature set is at a spot to begin tooling and use of this workflow.

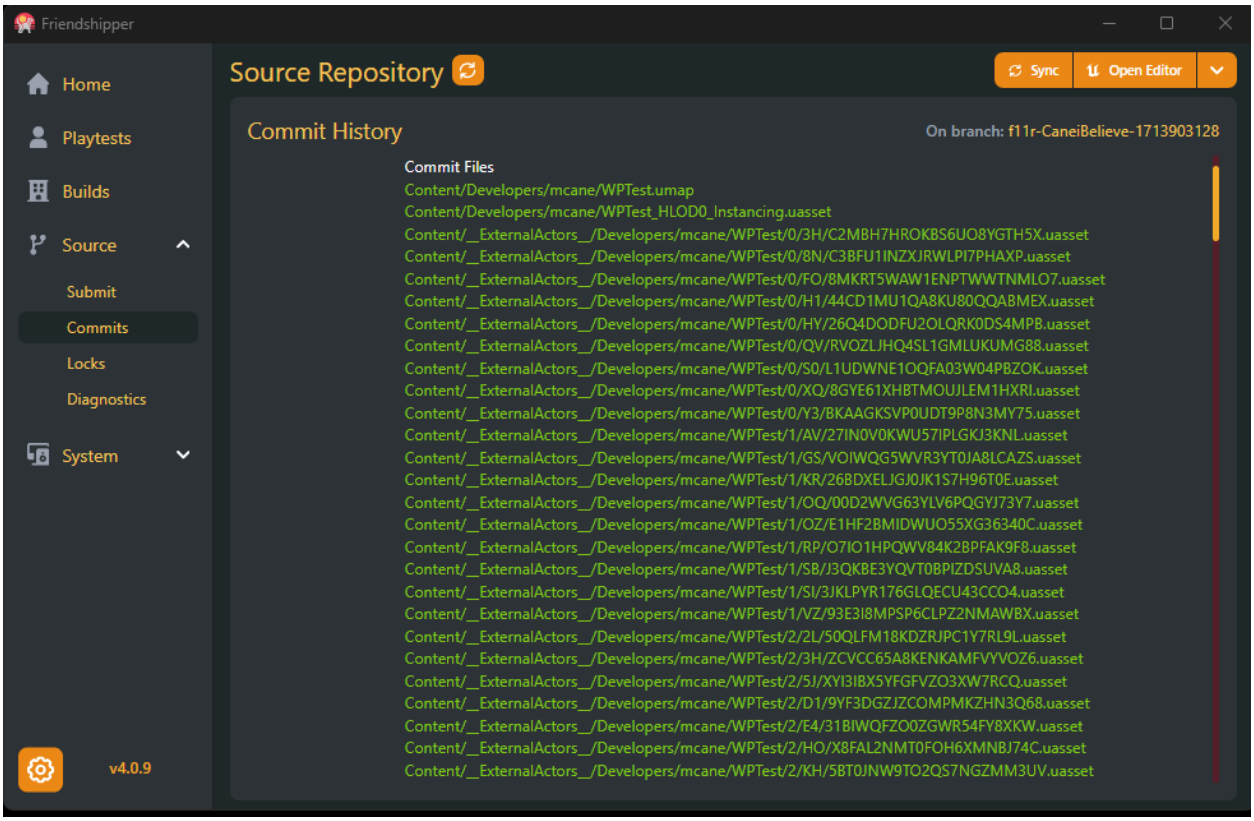
Basic Source Control Interfacing

Creating + Saving + Initial Submission

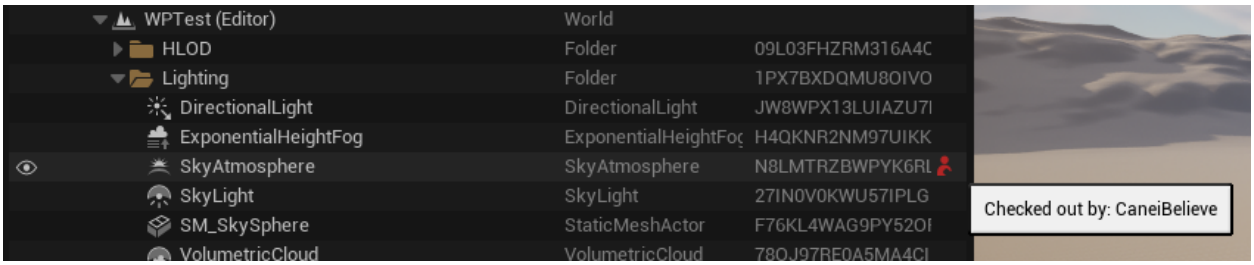
The initial save + submit dialog of a new open world map presents the following dialog with friendly names:



The actual on-disk/git submission is still terrible GUID names and random generated folders:



Multi-User Status

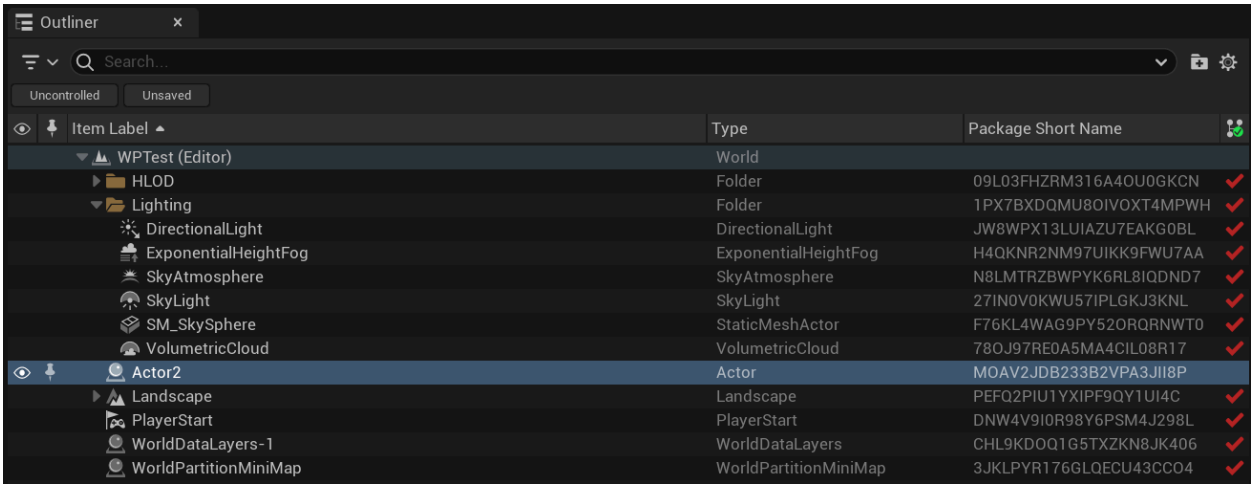


You can hover over a revision control icon for files checked out by others

Status Reporting In World Outliner

There is no source control reporting by default, but you can rightclick on any header tab and add a Revision Control column/tab. This does properly report source control status, we should seek to make this default.

Also Note: The GUID name (the file name on disk) is displayed as the *Package Short Name* column (presently the only way to connect friendly instance name to on-disk name)

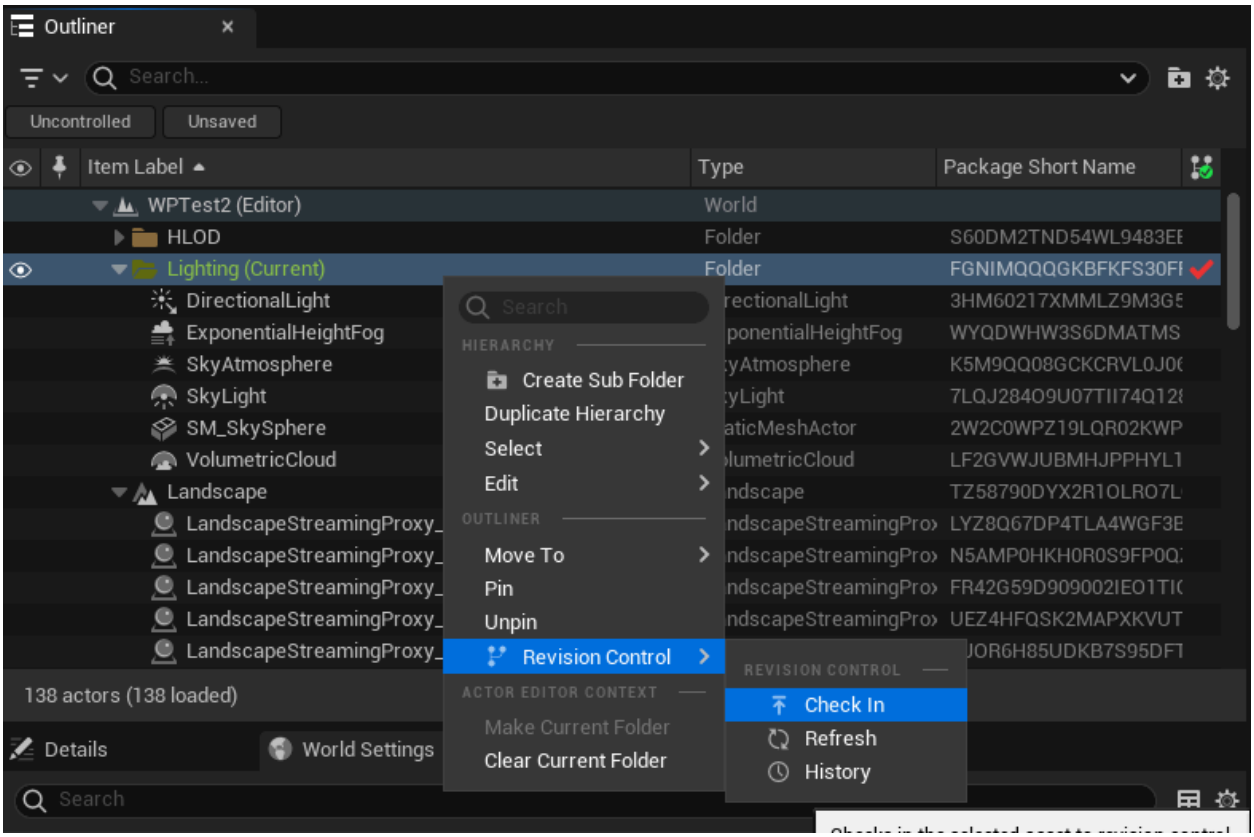


Doing a Submit Content (quick submit flow) locks you onto a temp branch, which causes all submitted files to falsely appear in a checked-out state because its no longer refreshing against main until the merge back to main if fully complete. You will see check marks disappear as OFPA's get merged.

FEATURE REQUEST: If you submit up-to-date, either bypass the temp branch or allow resetting to main without closing the editor since no files changed.

Workflow Features

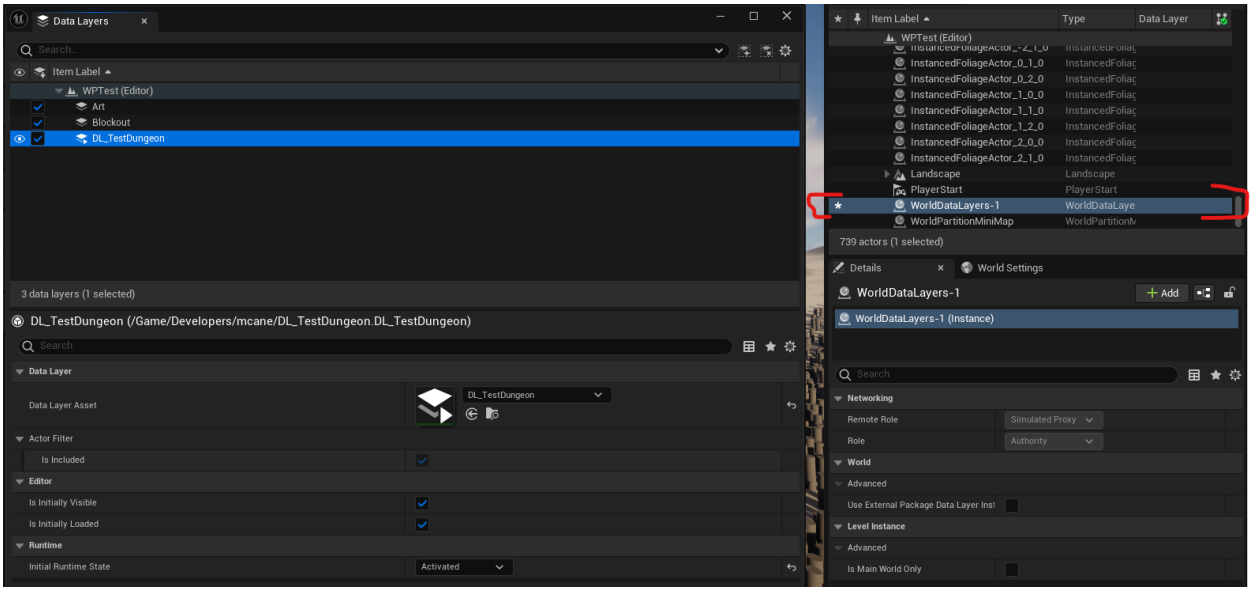
Folders



Folders now function fairly similar to sub-levels in a sense. They generate .uassets (They used to not to) on disk that can be individually checked out. Folders can also be “made current” which means all created content will automatically end up in the current folder (similar to making a sublevel current).

Folders being saved/submitted/updated does not store/impact references to the files in it. It seems like its just a container so folders don’t have to save to the persistent level. Individual actors/instances self-store their own folder organization (ie. moving SkyLight from Lighting into another folder would require saving SkyLight (NOT any folders) to update/save that folder move and nothing else. Actions done to a folder DO NOT impact the children (Folders in unreal used to, now they don’t) other than Select options.

Data Layers

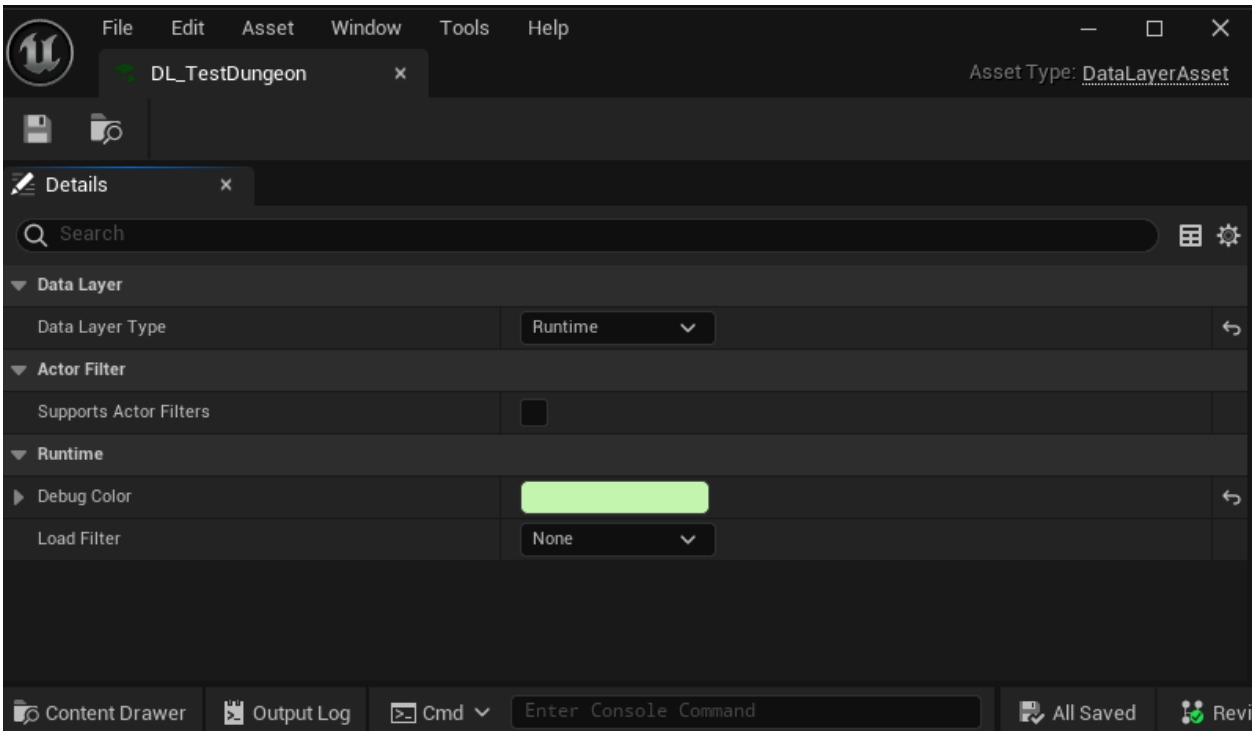


Data Layers are an organizational tool similar to folders, but with their own UI. They also contain additional functionality to allow actors that are members to receive certain editor time or runtime streaming capabilities.

DataLayers can be loaded/unloaded and/or visible/invisible by default or by any given runtime logic to solve for edge cases where the WP grid doesn't properly handle loading. They also can have that behavior for editor testing purposes, so a blockout DataLayer for example, could be toggled off once we are done with the Blockout stage.

Private DataLayers are purely an organizational tool that let teams split up levels by a data layer and filter visibility in an easy way.

Runtime data layers actually save an additional ASSET on disk that contains additional settings:



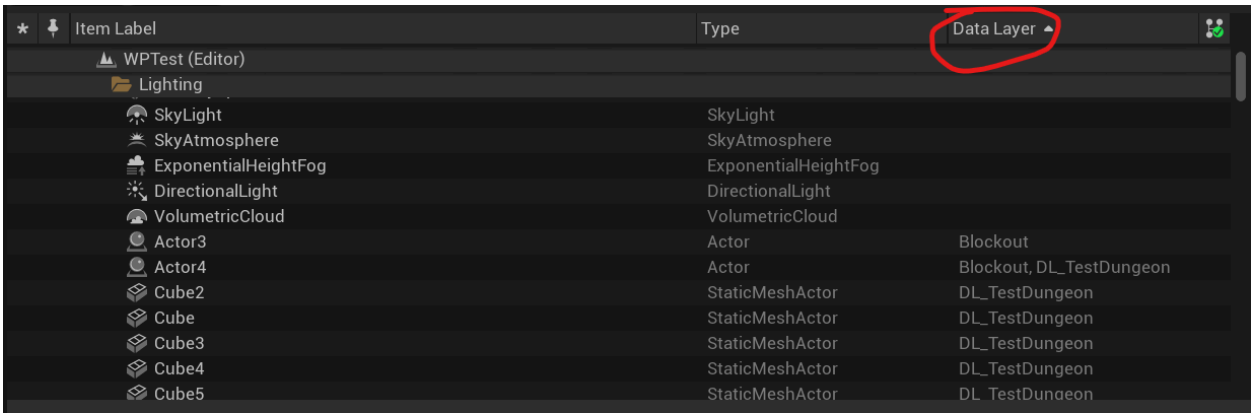
Runtime data layers contain typical streaming states (Unloaded; Loaded; Activated) and an additional Load Filter for Client Only or Server Only (or both which is the default; listed as *None).*

DataLayers can be double-clicked to be “Made Current” which automatically puts every placed actor afterwards into that Data Layer.

DataLayers are created in the Data Layers panel (Window > World Partition > Data Layers) and the creation of them is stored in the *WorldDataLayers* object in the world. This is somewhat problematic because this is a single OFPA that only one person can control at a time. So if multiple devs need to do ANY of the following at the same time, they would need to pass the file off:

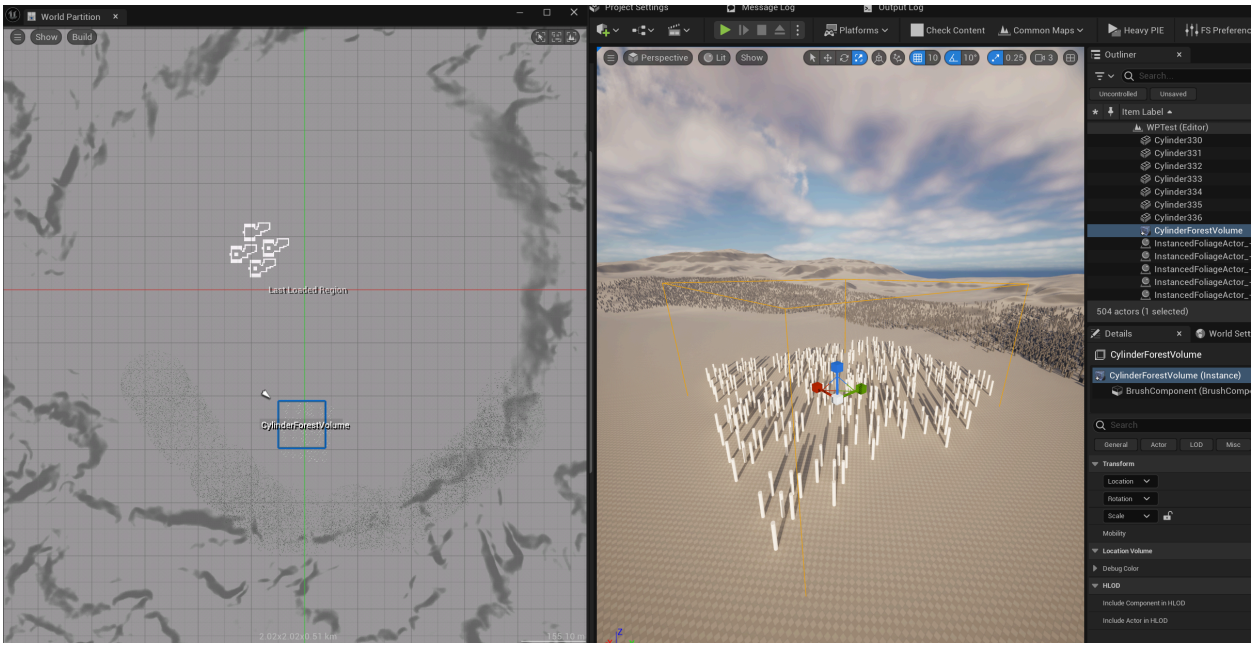
- Create a data layer (any type)
- Change ANY setting in the data layer settings (shown in the bottom section of the image above).
- Rename a layer.

DataLayers can be sorted/observed in the World Outliner by right-clicking on the header tab and adding a column:



Note that actors CAN be members of multiple data layers (but NOT multiple folders).

World Partition Editor

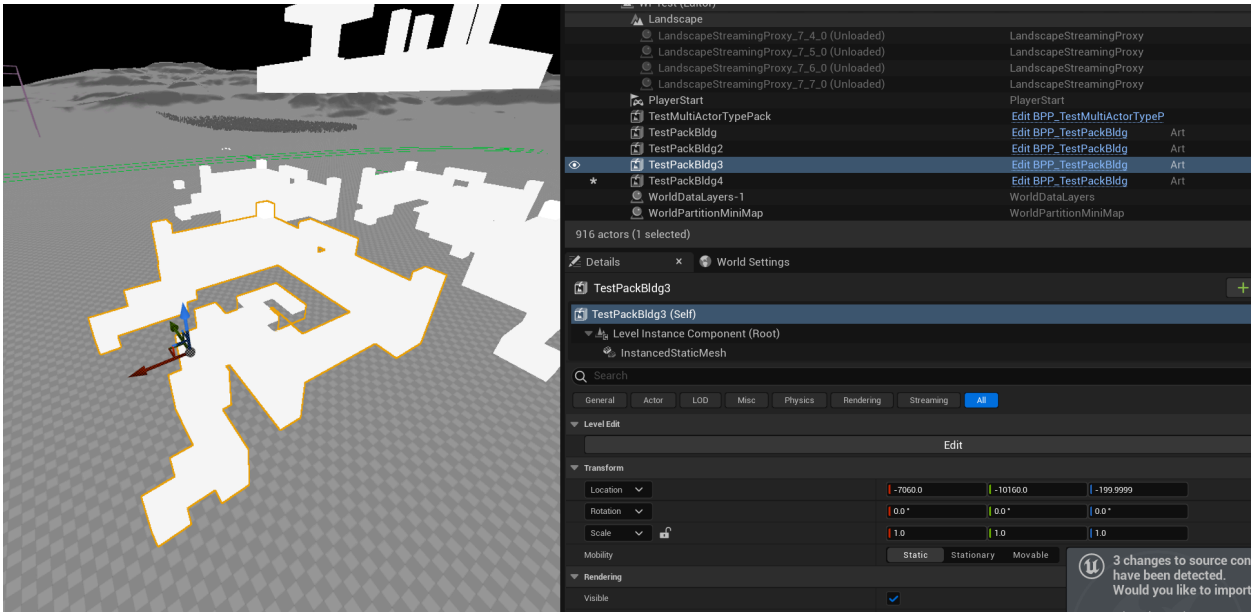


The World Partition Editor is a minimap-driven GUI that lets teams setup Editor-time loading zones via LocationVolumes, selections, etc. to load/unload portions of worlds so if maps are particularly huge, you only have to load what you're working on. This is potentially an optional feature until map sizes are extremely large and/or have a tremendous amount of content density.

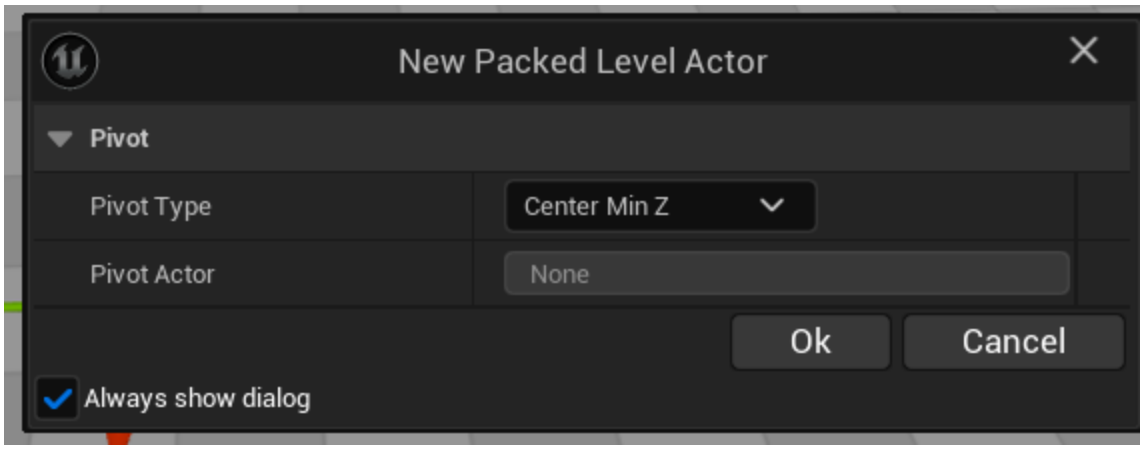
The minimap view has to be baked after content is added to update the view, and they can take a very long time but this editor window can serve a dual purpose to let you quickly navigate around the map, snap the camera to spots you're clicking and sanity check "paper" level design.

HLOD and Landscape Spline builds are initiated in this window as well. LandscapeSplines are individual actors

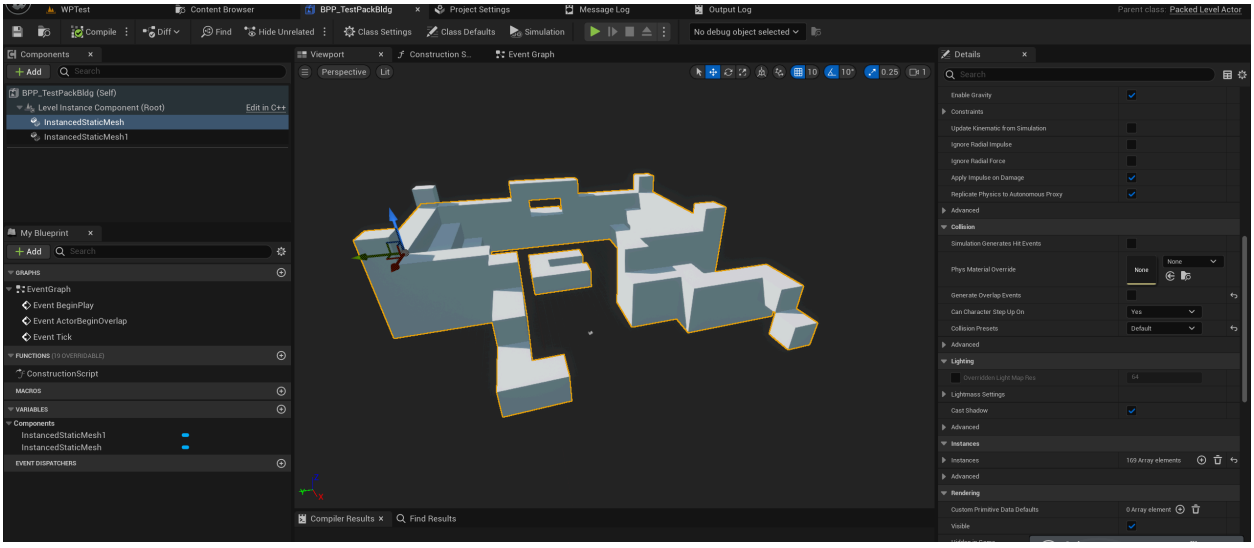
Packed Level Instance



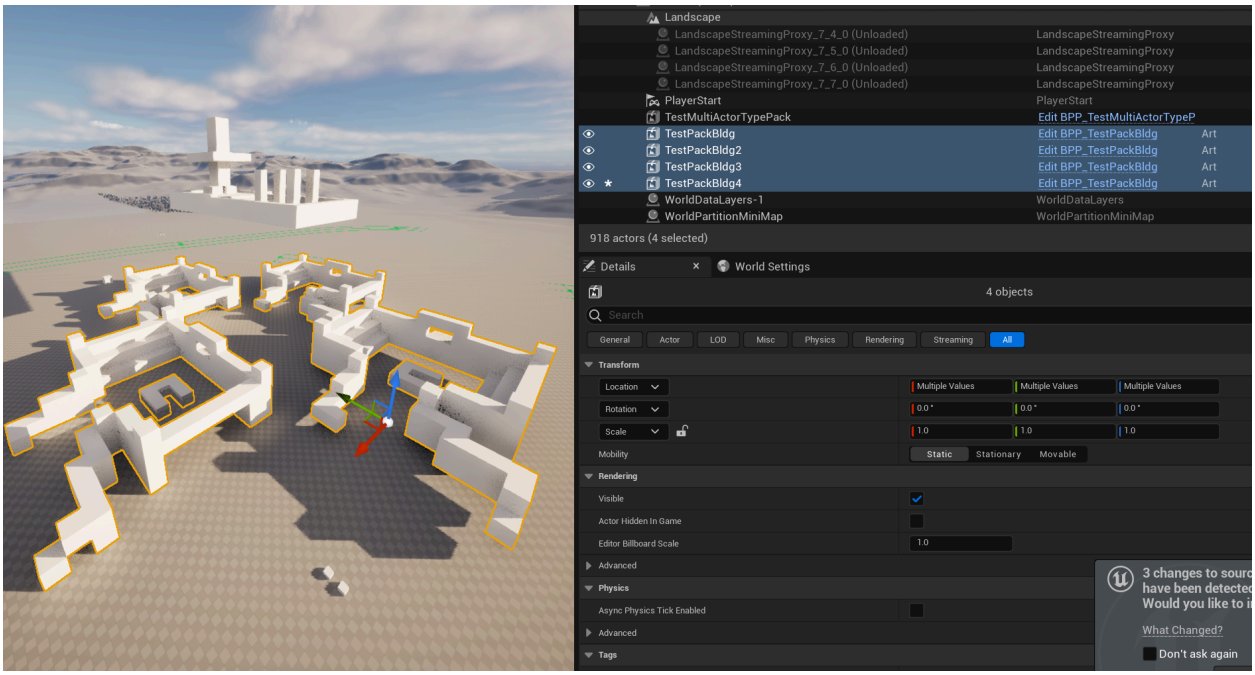
Creating a prefab of a set of re-usable static meshes can be done by selecting a group of StaticMeshes, right clicking in the viewport and choosing ***Level > Create Packed Level Actor*** which creates the below dialog:



Center Min Z will center the pivot at the “floor” of the selection; which is the typical ideal default. After confirming Pivot Type, you will be prompted to save a Level AND save a blueprint (this is the prefab blueprint). We should in general save these in the proper subfolder within the same biome/world folder as a majority of the source assets.



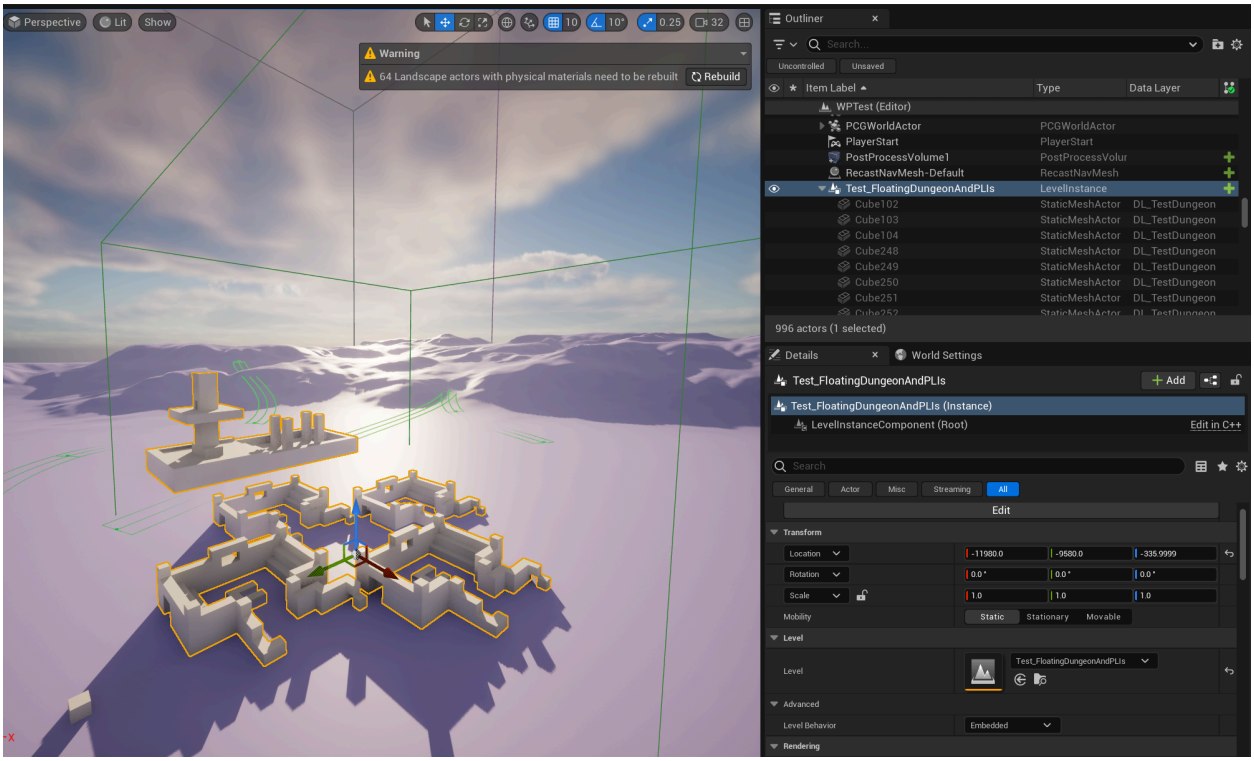
This shows the PackedLevelActor blueprint: It creates ISM components of all the static meshes to instance where it can instance to help with performance. This created Blueprint is what is used to place a version of the prefab/building/etc. in the world as many times as you need.



If you want to edit your PackedLevelActor/Prefab there is an edit button on any instance that will gray out the rest of the map and let you edit. When done, click ***Commit Changes***. **NOTE: CHANGES MADE IN EDIT MODE WILL APPLY TO ALL INSTANCES. If you want variants you would need to create the duplicate from the original meshes and add/remove instances specifically for that variant.**

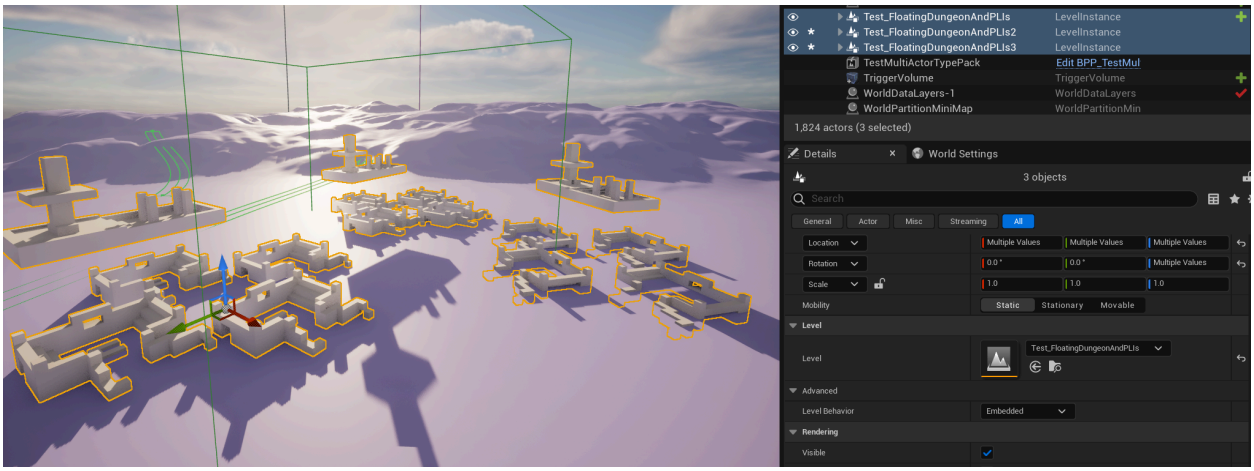
Editing a PLI-generated level (that holds all the OFPA for the prefab/PLI) and saving it did not propagate to the BPP or any placed PLI's back in the main World map. This is unfortunate.

Level Instances



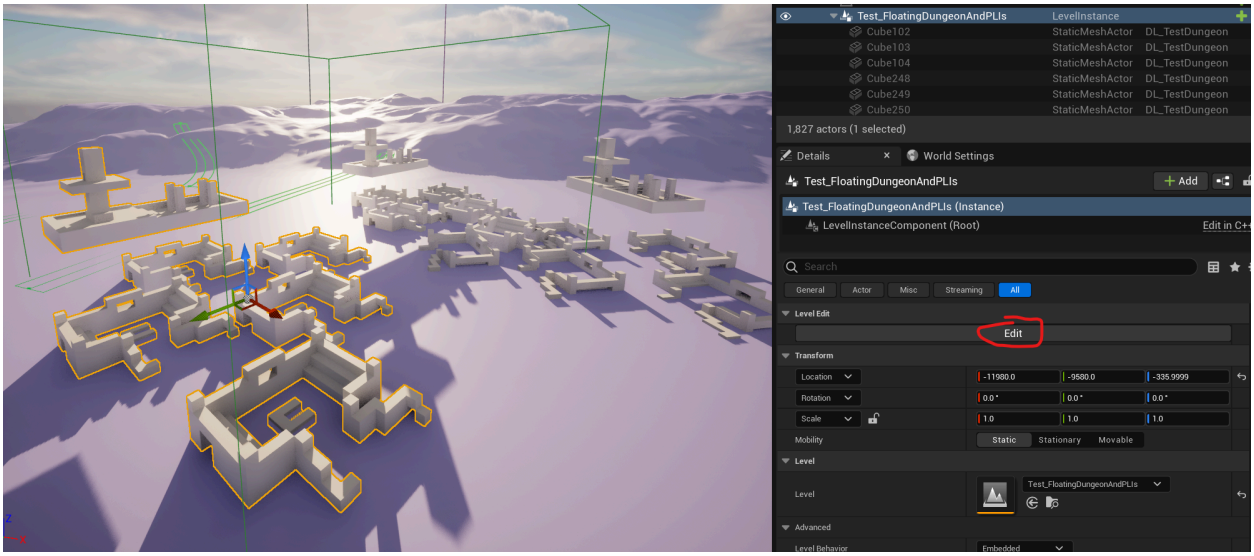
Using a few PLIs from the previous section and a floating set of cubes - a Level Instance was created. A Level Instance is similar to a PLI except it can contain ANY actor type.

Once established, the contents are part of the level instance, similar to a sub-level in the legacy UE workflow. The primary difference is you can duplicate these instances and have as many as you want:

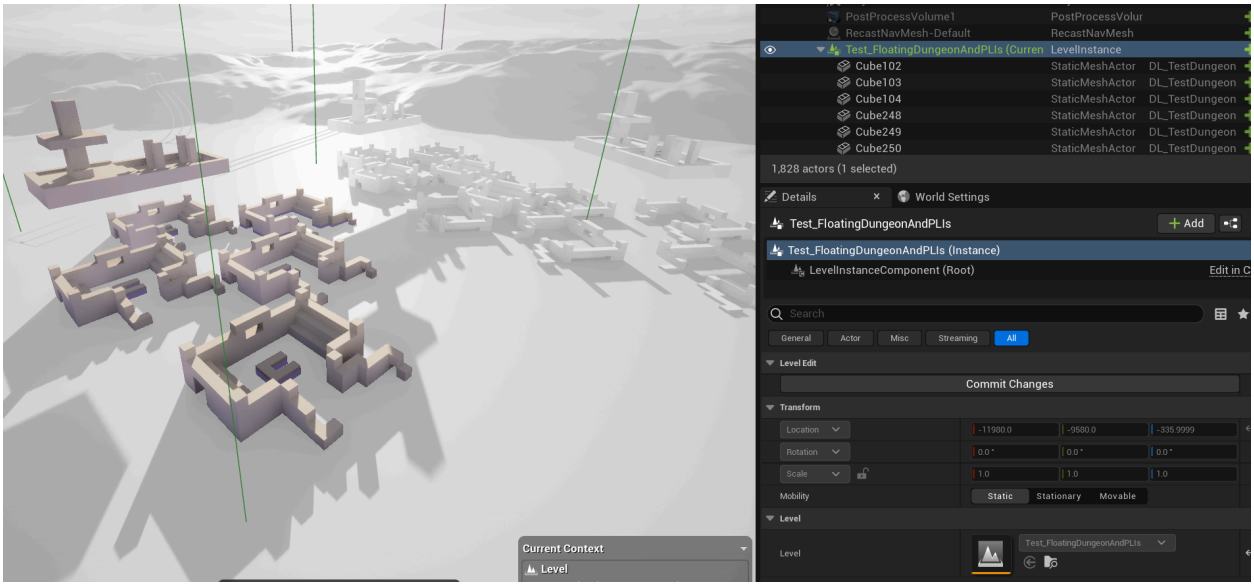


A Level Instance can be edited in one of two ways:

1) In-line in the editor



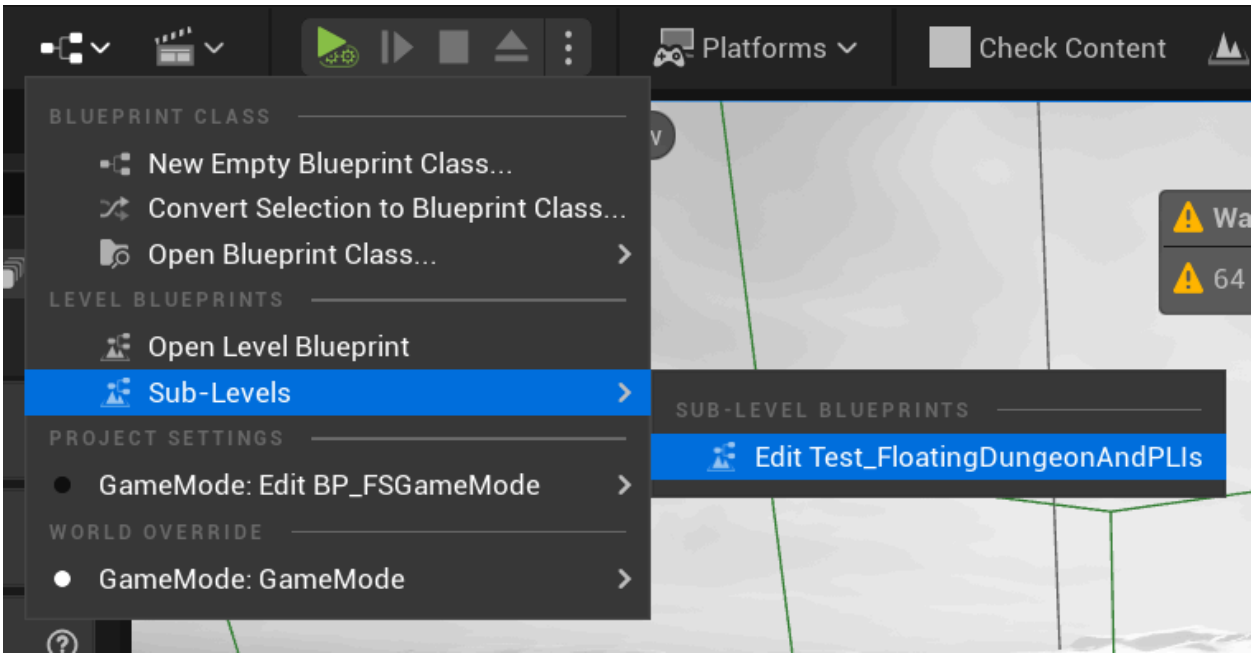
This puts the editor in a “grayed out” mode where all contents OUTSIDE of the Level Instance cannot be touched/edited. Any additions/subtractions to this Level Instance will propagate to all of the instances. If you want them to be different they must have a unique level reference.



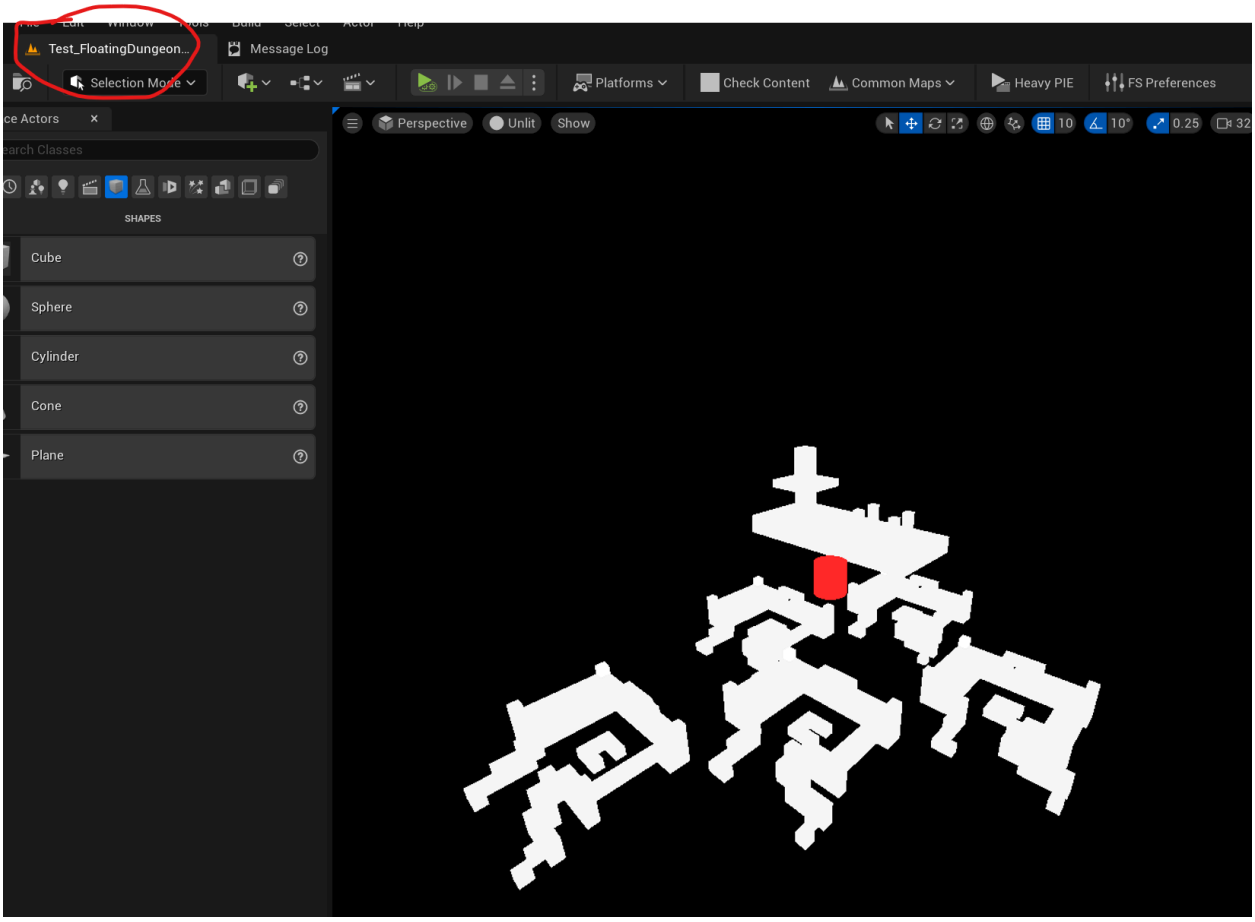
When done editing hit ***Commit Changes*** to update all level instances using the same Level Instance.

Important Workflow Note

Each level instance will add an entry to Sub-levels for Level Blueprinting purposes. This means that level instances could execute their own logic. You HAVE to be actively editing a Level Instance to edit the level blueprint for it:



2) Opening the actual level separately from the main world and editing it there:



This creates an opportunity for developers possibly focused on bunkers/dungeons/caves/etc. to work isolated in a level instance that the larger world contains one or more of.

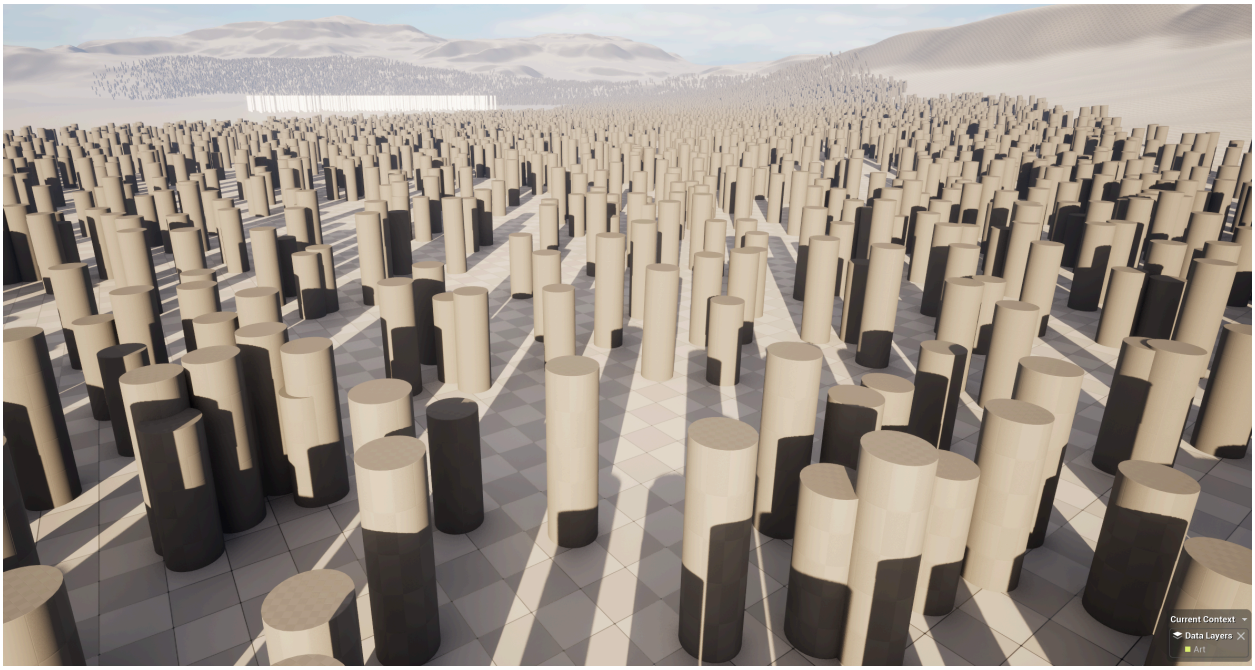
Experiments

Experiment #1: Simultaneously submitting 200-300 OFPAs at the same time (2 users)

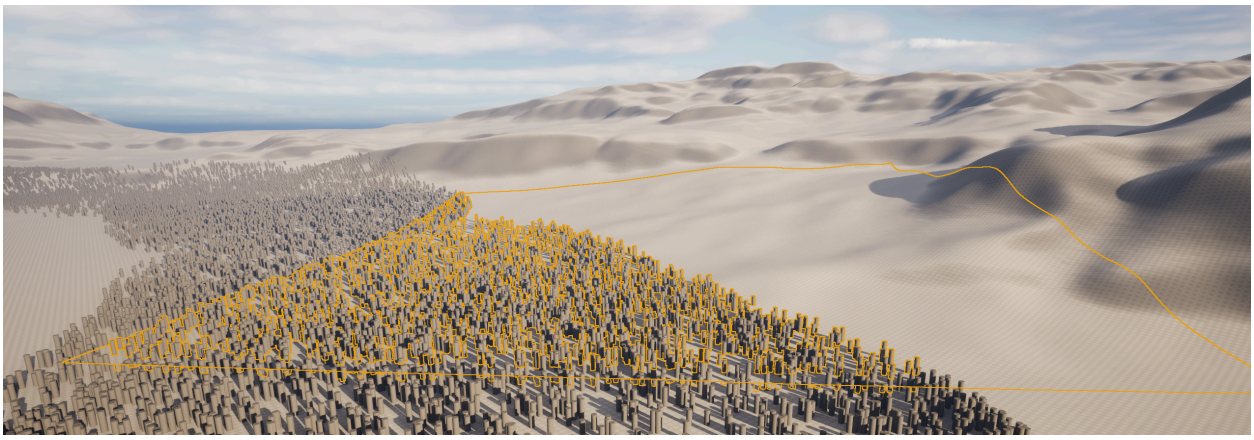
The first submit fully submitted of 200 files (Jed). My submission of ~338 appeared to time out in the editor but did successfully submit and enter merge queue in friendshipper. You can live view revision control status in the World Outliner (checkout check icons disappear as those actors are individually merged) so you can tell if your merge is done both inside the editor AND in friendshipper.

The 338 file OFPA submission (just a bunch of cylinders) took roughly 8 minutes for successful merge (this accounts for time for the ~200 file submission to also finish merging).

Experiment #2: Painting tens of thousands of foliage instances across the entire world.



We wanted to discover what World Partition does with InstancedFoliageActors. In legacy workflows, The InstancedFoliageActors would get put in the same sub-level as the Landscape. In World Partition, the IFA's are separated by LandscapeStreamingProxy boundaries and NOT the WorldPartition grid. This is preferred because someone working on a LSP and IFA at the same time could do so with only those two OFPAs. This also allows the WP streaming grid to be continuously iterated on without having to re-evaluate how IFAs are split up.



Example of a contiguous foliage "forest" being split up by the bounds of a LandscapeStreamingProxy (just a Landscape Component square).

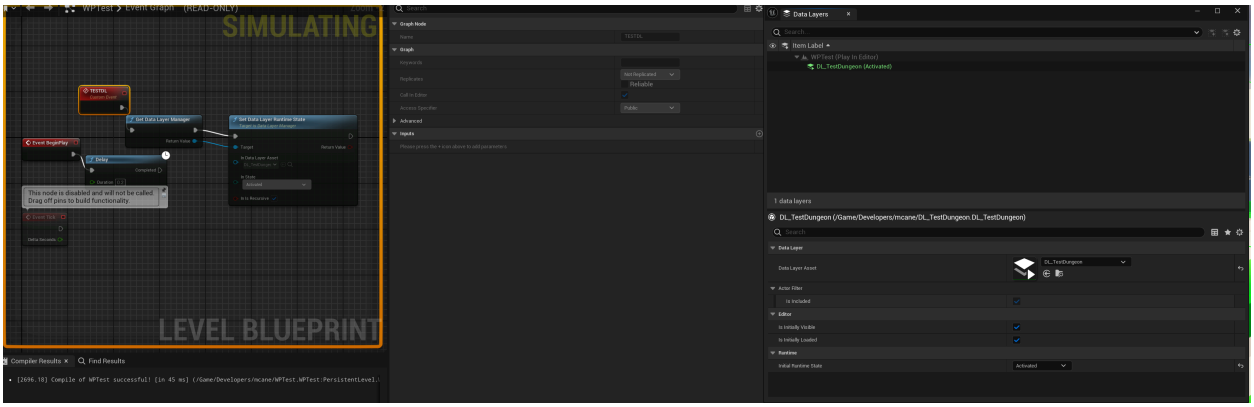
Experiment #3: Objects that span across multiple cells & floating objects

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/eb1bc295-736e-4e65-b35b-7a7c278083d9/WPTest_AdjacentCells.mp4

This experiment has a bunch of PackedLevelActors (prefabs) simulating buildings placed across different cell boundary lines. There is also a floating "building" of cubes that are individual cubes. The WP grid does appear to be fully 3d at runtime and you HAVE to encompass the entire loading range of a cell for the objects inside to load/render. **The solution to boundary crossover appears to ALSO be "make those assets part of a bigger partition grid name".**

It appeared that you could solve some of the issues in the first part of the experiment by creating a Runtime DataLayer and changing its loading state to circumvent the gridsize to keep things visible. This did not work. Loading/unloading/setting activated states at runtime on DataLayers only matters if you are INSIDE the cell already. So if we had large cells and wanted to load small

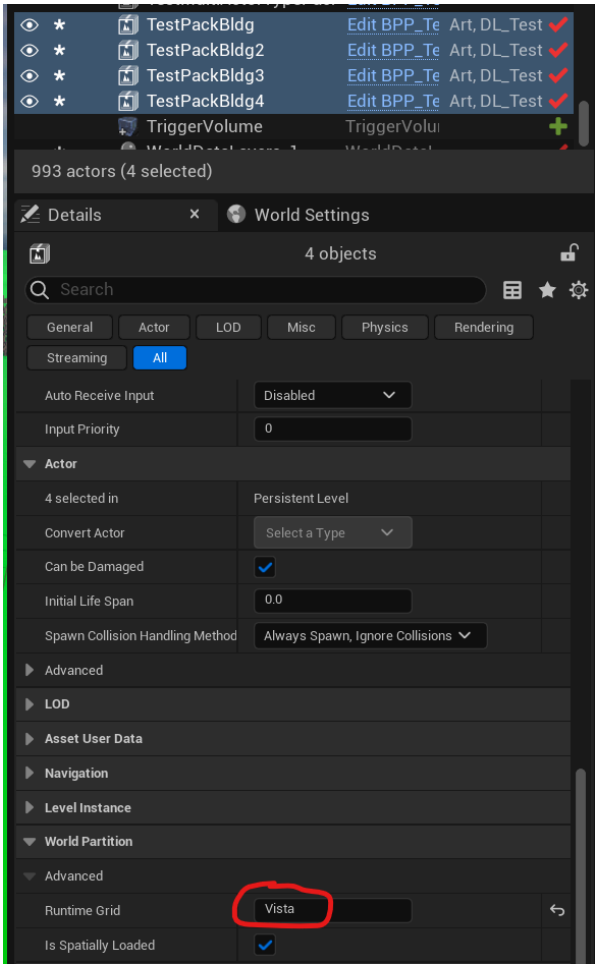
props or something after a gameplay event we could do so, but it is not the proper workflow for distant/floating objects.

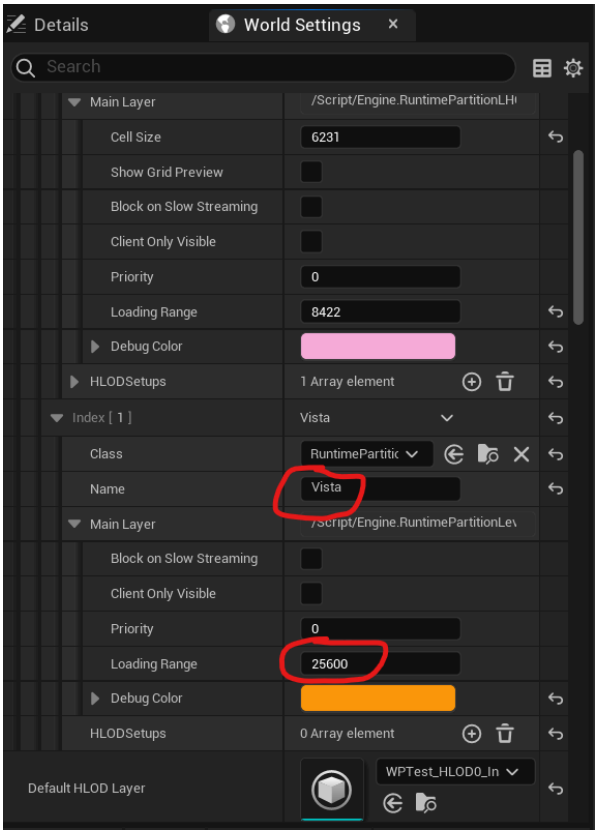


What was tested with DataLayer state changes at runtime:

1. Started off data layer containing the PLIs and the floating cubes to visible/loaded AND activated.
    a. Expected Result: Objects in DL would be visible/loaded until the DL state was changed.
    b. Actual Result: The grid took priority and the DL logic did not change behavior.
2. Changed the DL settings to be unloaded and initially unloaded and not visible.
    a. Expected Result: Objects would be invisible until you entered the loading range.
    b. Actual Result: Objects stayed unloaded/invisible no matter what.

The actual fix for “I want to see far away floating things” was to make a second grid with a much larger Loading Range and set that grid in all the actors you want to see far away:



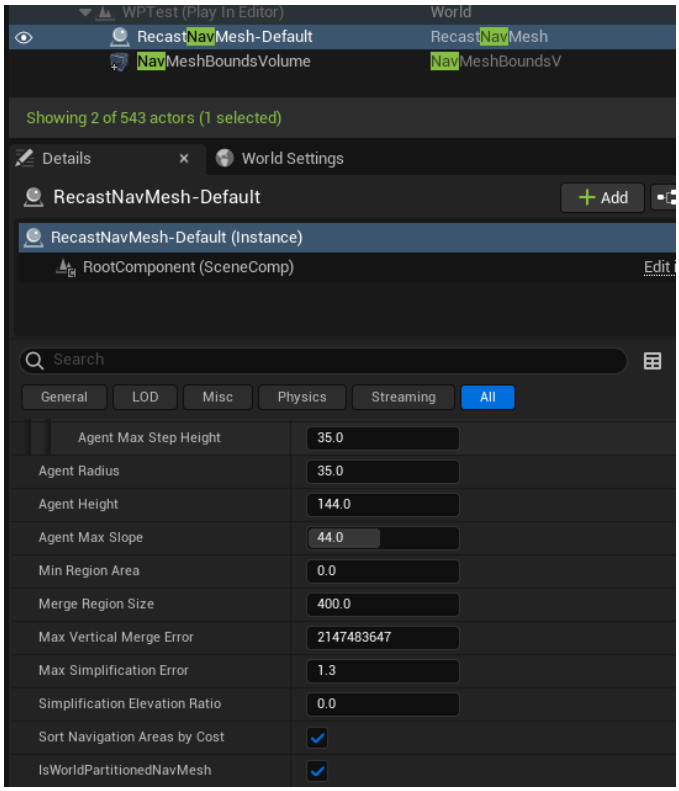


Note the loading range is about 3x of the main grid used before.

The end result of the additional grid:

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/07b446fe-11b0-4d22-9569-21d0e8210efe/WPTest_Vista_Floating.mp4

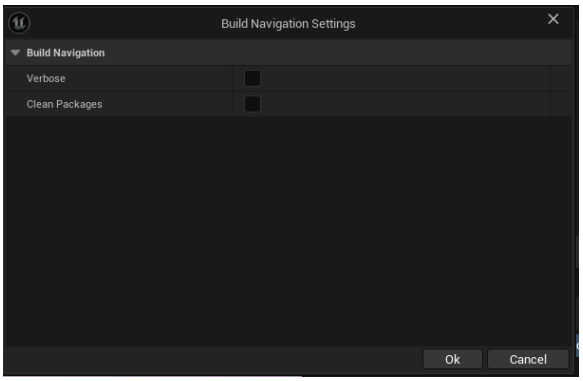
Experiment #4: Adding NavMesh to see what happens



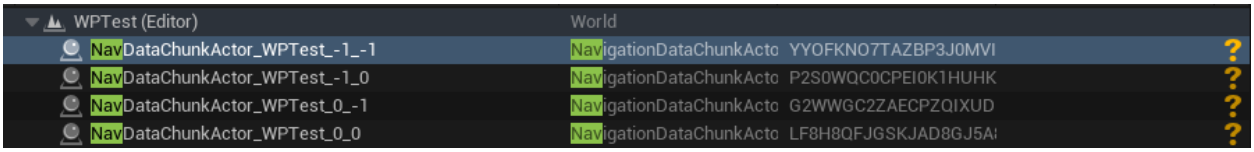
NavMesh has a different workflow that requires some configuration for UE5.4.

In your RecastNavMesh (requires one Build Paths) actor you need to check ***IsWorldPartitionedNavMesh*** and then do the following steps:

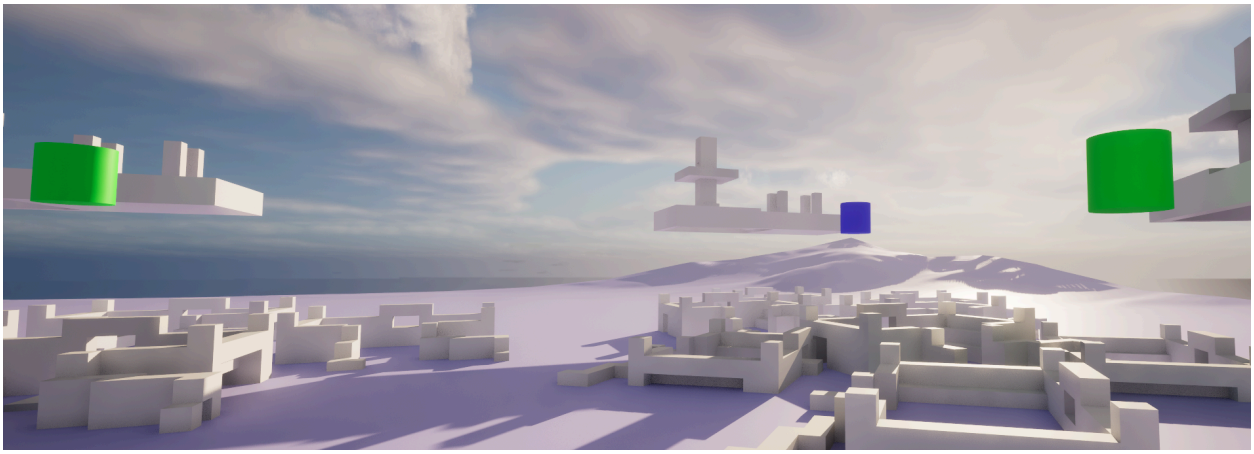
1. In Editor Preferences > Level Editor > Miscellaneous UNCHECK ***Updated Navigation Automatically***
2. Press ~ to focus Console and enter: n.bNavmeshAllowPartitionedBuildingFromEditor 1
3. Now Build > Build Paths will compile WP-supported NavMesh instead of legacy NavMesh. You will get the following dialog each time, if the command worked properly:
    a. Checking Clean Packages will remove all NavDataChunkActors. You will need to run Build Paths again to regenerate them.



The end result, instead of ONE RecastNavMesh volume you get multiple chunks:



Experiment #5: Randomization inside a LevelInstance



I wanted to see if a LevelInstance could function as essentially a “dungeon generator” in a sense that randomize logic would be different between different instances. Here is what was attempted:

1. Putting randomized logic to change the cylinder color in the level instance level blueprint.

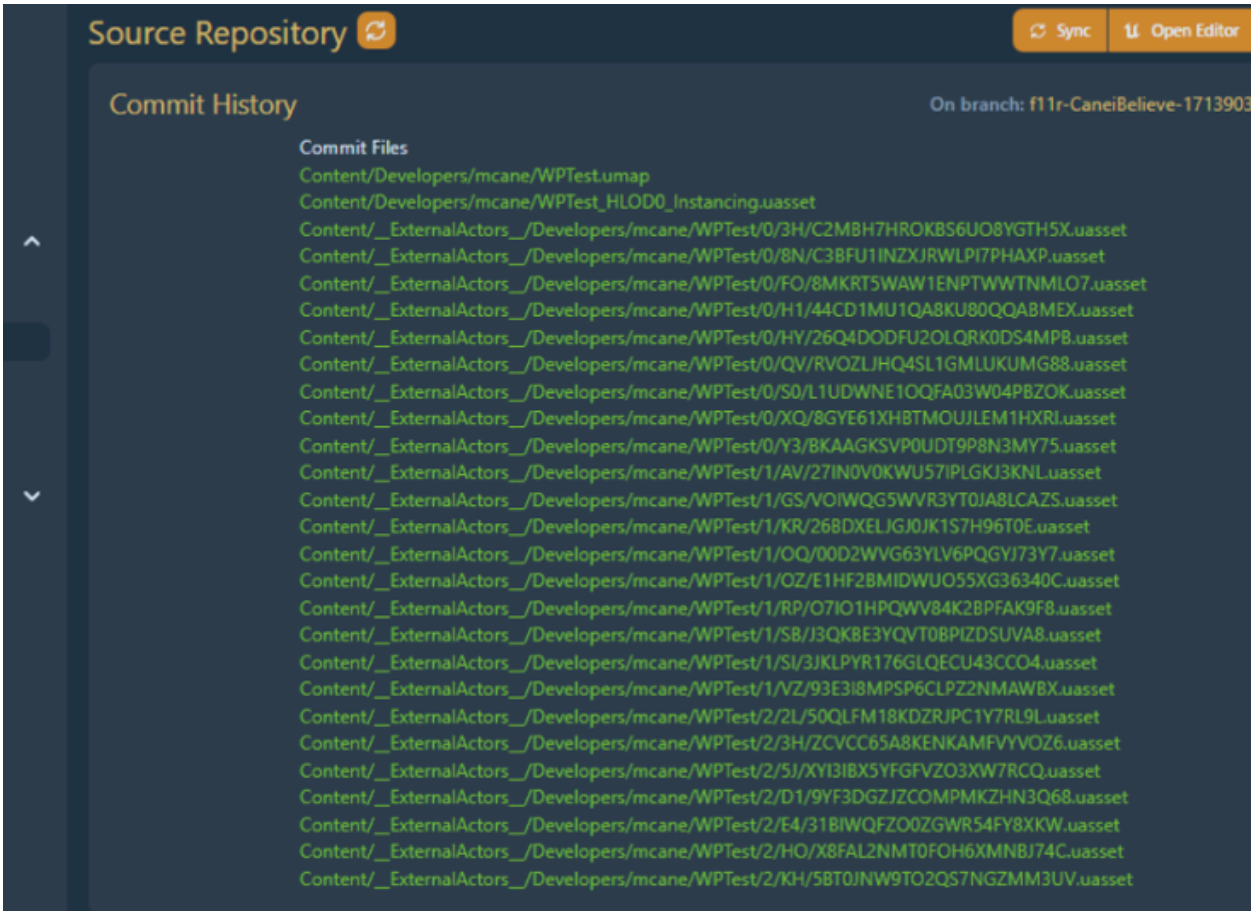
    a. Expected Result: It would randomize between 3 colors.
    b. Actual Result: Nothing, logic did not randomize.
2. Putting randomized logic inside a blueprint that contained the cylinder.
    a. Expected Result: It would randomize between 3 colors.
    b. Actual Result: It worked properly.

The conclusion for this experiment was that we can do randomized behavior within level instances, where even if the level instance is used 20 times - as long as the logic exists inside a blueprint, and not the level blueprint, it appears to work.

Tool & Workflow Requests

Based on the evaluation, these are improvements we need as a team to increase confidence in adoption World Partition in the short-term and at scale.

Friendly/Identifiable Names on Disk -or- Displayed in Friendshipper



This example is one of the worst aspects of OFPA/WP in its current state. There is no way to know what any of these are without having the editor open. We need to encode some sort of friendly name or ID Name identifier to display in commit histories on Friendshipper so devs never see the above view. This will also reduce support burden when backend needs to track down problem files when the person requesting help may not have editor access to troubleshoot.

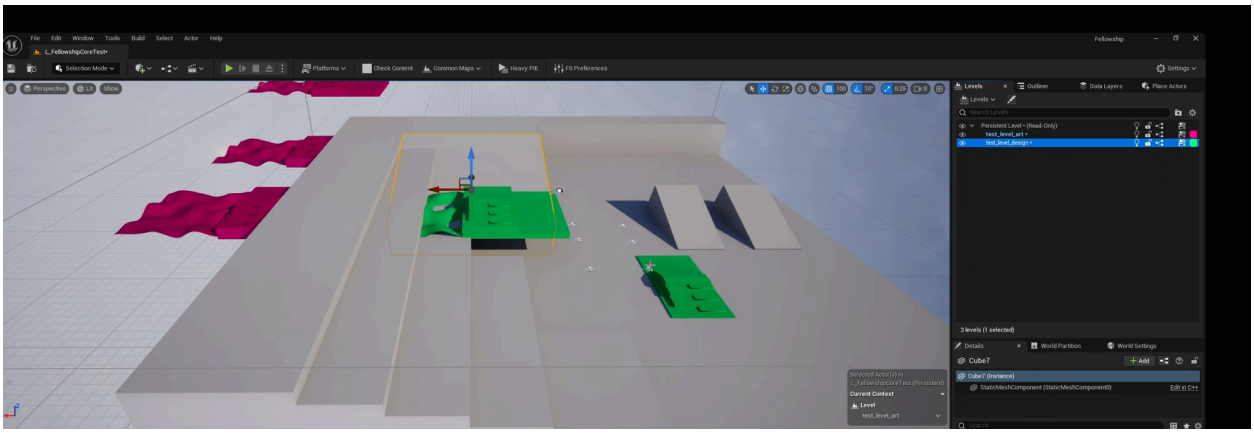
FSLevelEditorSubsystem Extensions + Editor Shortcut Extensions

There is no Outliner subsystem or Outliner calls for Editor Utility Widget use, so I'd like to request the following methods be exposed/created for BP tools use:

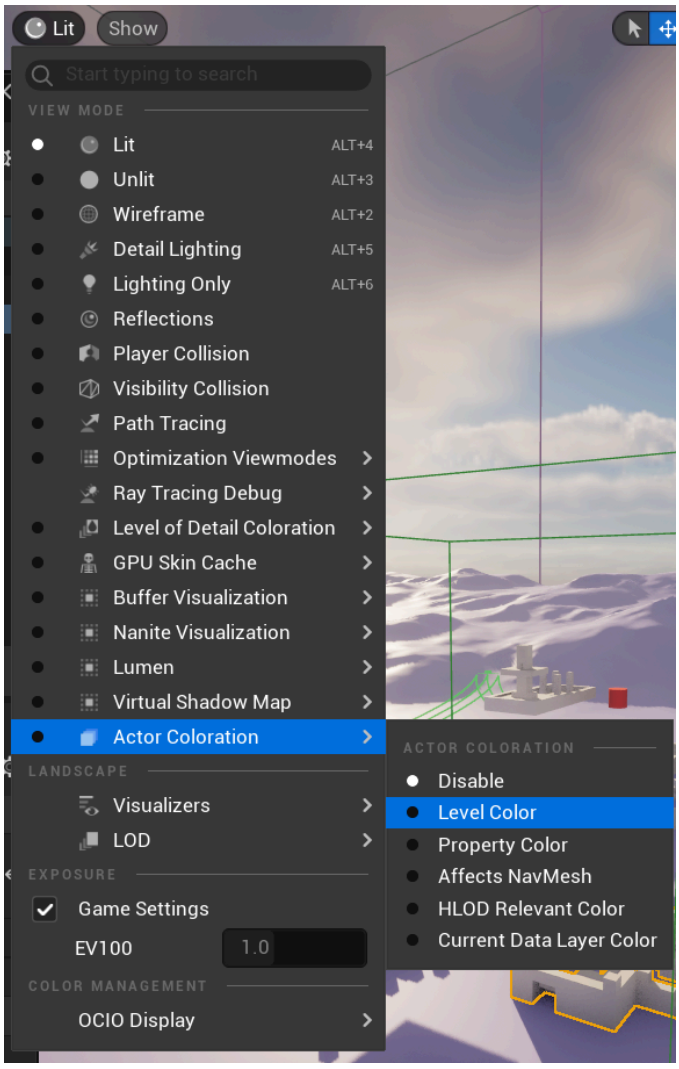
- Get Outliner Folders (outputs Array of folders)

- Get Folder Children (outputs array of actors in a folder passed in, bool for recursive [outliner calls it immediate children or all descendents])
- Create Folder (input: folder name, folder color)
- Set Folder Color (linear color)
    - Keyboard shortcut added
- Make Current Folder (folder name)
- Get Current Selection (outputs array of selected actors in the outliner)
- Get Current Folder (outputs folder name that is current)
- Move To Current Folder (input array of actors)
    - Keyboard shortcut added for this, uses current selection of actors as input array by default
- Move to Folder (input: folder name and array of actors)
- Get DataLayers (input: array of actors; outputs array of datalayer names)
- Get Source Control Status
    - There was a similar call in the sub-level tool that output a struct of its source control status, who has it checked out, etc.
    - Outputs reference of original actor (even in async)
- Set folder Name(name)
- Bind to OnSelectionChanged (editor selection)
    - Presently you have to use tick/timer with EditorActorSubsystem.GetSelectedActors to do things on select, it would be a major editor performance improvement to bind to selection change when the tool is initialized and output the updated selection array when it changes.

Folder Coloration



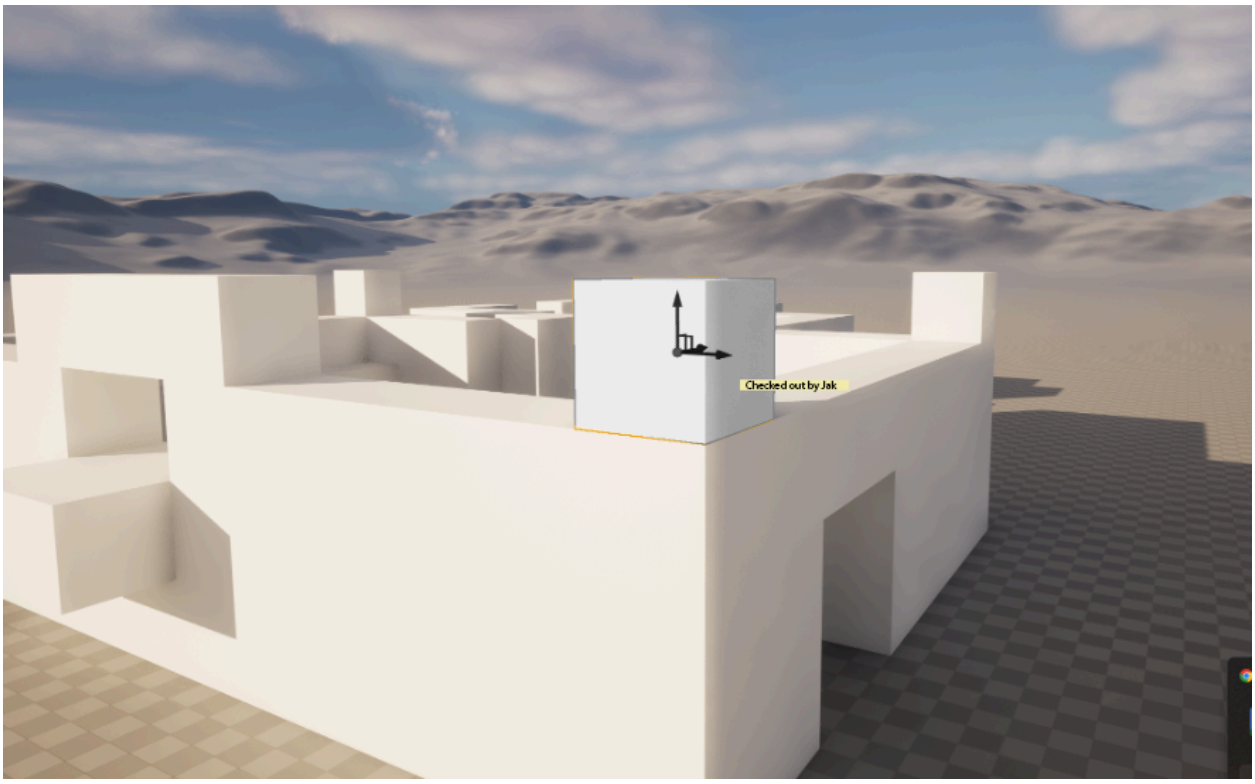
Shown above is Level Coloration from the Sub-level workflow. Devs can set an explicit color for sub-level and visualize all members of the level in 3D via the viewport flag below:



We want this functionality on Folders in the outliner for World Partition. Content Browser already lets you set folder colors so ideally the tool would function:

- Right click on an Outliner folder > Set Color > Choose from color picker
- From Lit > Actor Coloration > Folder Color (new option), activate the visualizer like the sub-level example to color-code all actors in that folder in the 3d viewport.

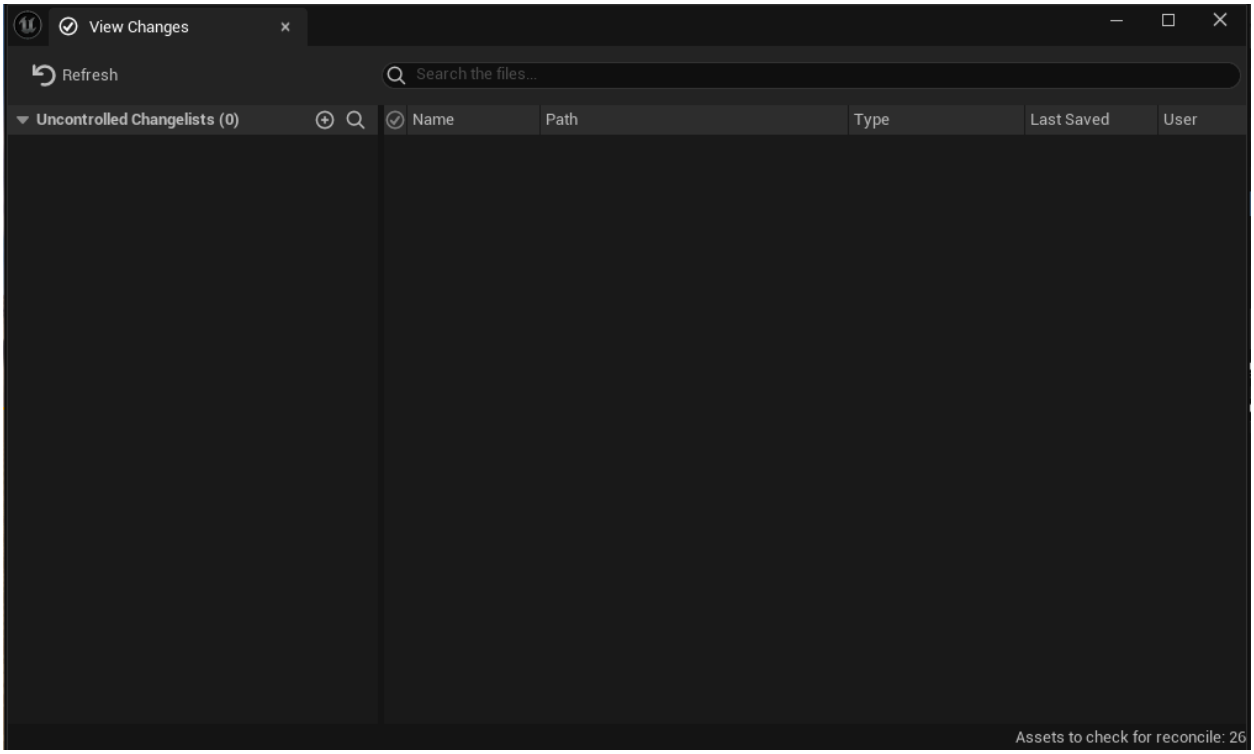
Viewport UX for Actors Checked Out By [Not You]



If the selected cube is checked out by someone else there should be some workflow tooltip, desaturation or something to indicate that the object is checked out by someone else, who that is and lock transforms of that object to reduce the chance of accidental conflicts.

Bugs/General Areas of Improvement

- Submitting a large list of OFPA appeared to timeout in the editor; but did successfully submit/merge. Timer needs to be longer for timeouts?
- When working with a ton of OFPA's its easy to lose track of when ones get marked for add/delete/rename/etc. and there are a lot of complex file actions that Unreal takes care of. Trying quick submit in Friendshipper failed because the file list was hundreds of OFPAs with various delete/add/modified states. However, submit content from the plugin in Editor let Unreal do the required revert/add/submit chains. **We may need to require devs to only submit from Unreal for World-related changes.**
- The Revision Control > View Changes dialog does not actually preview your changes. This is a Friendshipper/UE5.4 plugin bug; it has nothing to do with World Partition.



- Data Layers have to be created/stored/updated in the WorldDataLayers object, that is one OFPA. This means only one developer can make those changes at a time; whereas folders are one OFPA PER folder. This makes WorldDataLayer objects a similar issue as Persistent Levels in legacy workflows (To add/remove/change a sub-level you have to checkout/save Persistent).

- Data Layers cannot be used to manage loading/visibility if you are outside of a grid square, only inside UNLESS you are trying to turn things off (but not ON). This means you have to make multiple WP grid sizes and apply them to each placed Actor by name (in the World Partition settings block).
- All WP partition grids are manage/saved in the persistent world, meaning only one developer could ever add/change/remove a runtime partition grid.
- PackedLevelActors can only have Static Meshes, having multiple actor types did not transfer. Since PLA's are optimized for a "fast render path" (really they just use ISMs) they have that specific use case only.
- Editing a PackedLevelActor-generated level (that holds all the OFPA for the prefab/PLI) and saving it did not propagate to the BPP or any placed PLI's back in the main World map. This is unfortunate.
- Level Blueprint only exists for the Persistent level, so only one developer could ever world in LevelBP at a time. This signals a move towards self-contained Volume logic at the BP level, versus placing generic triggers/volumes in world and doing the logic in the Level BP. This isn't the worst workflow thing, but it will certainly be an adjustment for devs used to Level BP.
- NavMesh must be built by one dev, it is stored in the Persistent World (legacy workflow worked this way too).

Questions for Design/Art

1. How do we want to organize content? (ie. Folders for X; DataLayers for Y)
2. Who owns Runtime Partition Grid Settings? (Proposed: TA)
3. Who owns/manages DataLayers?
4. Are the shortcomings of World Partition addressable to a level that we feel this workflow is better than the legacy sub-level workflow?

Questions for Engineering

1. Does NavMesh actually split up by Partitions, its own cells or not at all?
   Answered by Reuben (see NavMesh experiment)

Proposed Workflow & Conclusion

🌐 World Partition Workflow

Based on the above evaluation and workflow proposal, it is fair to conclude that adopting World Partition as we transition to 5.4 and begin pre-production now is the best move for our studio. This allows us the time to familiarize ourselves with the new workflow, create new training and tools and by the time we're at EOY 2024 we should have a mature workflow, especially as we gain additional features and support from Epic.

However, the highest priority tools requests (namely OFPA friendly naming in Friendshipper/on-disk) should be done ***BEFORE*** the team adopts World Partition to reduce support time for the Dev Workflow team. Ideally, some of the folder tools and fool-proofing too.

Why World Partition Now?

- WP's flaws are mostly organizational, its strengths are in 3d level streaming tech. Workflow/tools support for this is a much smaller lift. Supporting custom streaming is many times more complex/time consuming for engineering than the former.
- A lot of the pain points of World Partition also exist in the Sublevel workflow.
- The editor tools and our own source control have progressed to a point that we are ready for the adjustment period.
- Epic is heavily pushing for studios to adopt WP and doing so may help with future support.
- It has improved nicely since the initial evaluation in 5.2.
- The longer we wait the more difficult transitioning content/maps to the new workflow may become.
- This workflow does not restrict our world size, should that matter, where sublevel workflow has some limitations there.

Pain Points/Risks

- Working with OFPA, even with stress tests, will cause A LOT of source control headaches to work through early on. This will increase support needs for Dev Workflow potentially a lot at first.
- The team that learned and worked in the sub-level workflow will need some time to adjust to World Partition. Training materials/calls will be provided to aid with this transition.
- We do not know how WP will function on different platforms, where hundreds of games have shipped in the sub-level workflow.