
gcap
games
connect
asia pacific

Streaming & Serialisation For Unreal Open Worlds

Who am I?

Who am I?

ANIMALLOGIC

Pipeline Engineer

Who am I?

ANIMALLOGIC

Pipeline Engineer

Indie Developer

Who am I?

ANIMALLOGIC

Pipeline Engineer

Indie Developer

GAMELOFT

Gameplay Engineer

Who am I?

ANIMALLOGIC

Pipeline Engineer

Indie Developer

GAMELOFT

Gameplay Engineer

?

Why this talk?

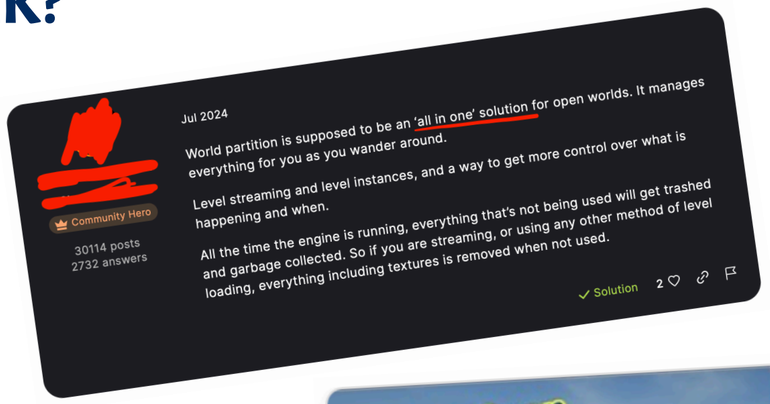
- I didn't know anything about how to make open worlds

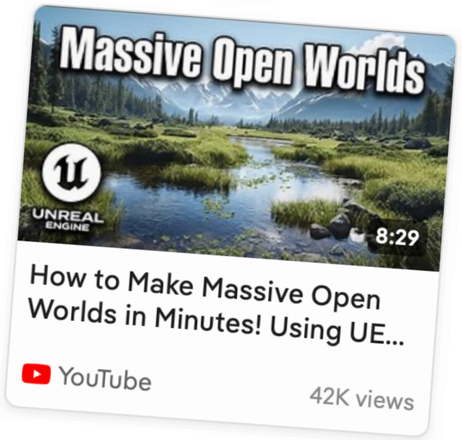
Why this talk?

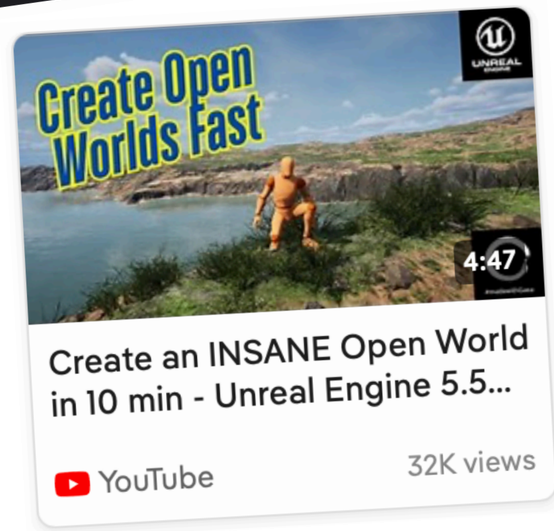
- I didn't know anything about how to make open worlds
- Unreal's World Partition was exactly the solution we needed
- So much incredible functionality for free
- Very little documentation on its features
- NOT an "all in one" solution

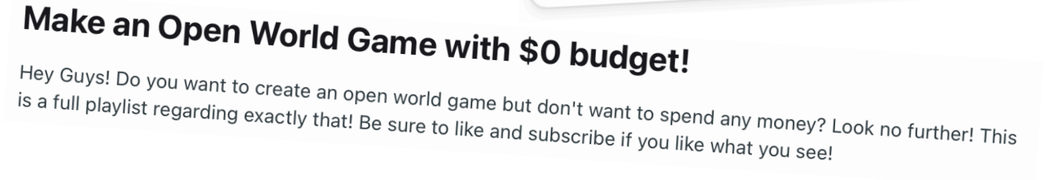
Why this talk?

- I didn't know anything about how to make open worlds
- Unreal's World Partition was exactly the solution we needed
- So much incredible functionality for free
- Very little documentation on its features
- NOT an "all in one" solution
- *People were wrong on the internet*









What isn't this talk?

- **NOT** a tutorial on "how to get started"
- **NOT** a step-by-step guide with code-snippets
- **NOT** a talk about asset streaming or to-disk serialisation
- **NOT** a deep dive on engine functionality

What isn't this talk?

- **NOT** a tutorial on "how to get started"
- **NOT** a step-by-step guide with code-snippets
- **NOT** a talk about asset streaming or to-disk serialisation
- **NOT** a deep dive on engine functionality

"Why isn't my quest location showing up?"

What isn't this talk?

- **NOT** a tutorial on "how to get started"
- **NOT** a step-by-step guide with code-snippets
- **NOT** a talk about asset streaming or to-disk serialisation
- **NOT** a deep dive on engine functionality

"Why isn't my quest location showing up?"

"Why isn't the navmesh generating in builds?"

What isn't this talk?

- **NOT** a tutorial on "how to get started"
- **NOT** a step-by-step guide with code-snippets
- **NOT** a talk about asset streaming or to-disk serialisation
- **NOT** a deep dive on engine functionality

"Why isn't my quest location showing up?"

"Why isn't the navmesh generating in builds?"

Don't worry, there will be a link to the slides provided!

gcap games connect asia pacific

What to expect over the next 50 minutes

What are Unreal's World Partition systems?

Streaming systems for game state management

Serialisation systems for game state persistence

Wrap-up, summary and Q&A

gcap games connect asia pacific

What to expect over the next 50 minutes

What are Unreal's World Partition systems?

- OFPA
- HLODs
- Grids
- Level Instances
- Packed Level Actors
- Smart Objects
- Data Layers

Streaming systems for game state management

Serialisation systems for game state persistence

Wrap-up, summary and Q&A

gcap games connect asia pacific

What to expect over the next 50 minutes

What are Unreal's World Partition systems?

- OFPA
- HLODs
- Grids
- Level Instances
- Packed Level Actors
- Smart Objects
- Data Layers

Streaming systems for game state management

- Schedules & Quests
- Interior/Exterior Spaces
- Markers
- Lighting & Environment Changes

Serialisation systems for game state persistence

Wrap-up, summary and Q&A

gcap games connect asia pacific

What to expect over the next 50 minutes

What are Unreal's World Partition systems?

- OFPA
- HLODs
- Grids
- Level Instances
- Packed Level Actors
- Smart Objects
- Data Layers

Streaming systems for game state management

- Schedules & Quests
- Interior/Exterior Spaces
- Markers
- Lighting & Environment Changes

Serialisation systems for game state persistence

- World State Representation
- Live Data Capture
- Handling Actors
- Checkpointing

Wrap-up, summary and Q&A

gcap games connect asia pacific

What to expect over the next 50 minutes

What are Unreal's World Partition systems?

- OFPA
- HLODs
- Grids
- Level Instances
- Packed Level Actors
- Smart Objects
- Data Layers

Streaming systems for game state management

- Schedules & Quests
- Interior/Exterior Spaces
- Markers
- Lighting & Environment Changes

Serialisation systems for game state persistence

- World State Representation
- Live Data Capture
- Handling Actors
- Checkpointing

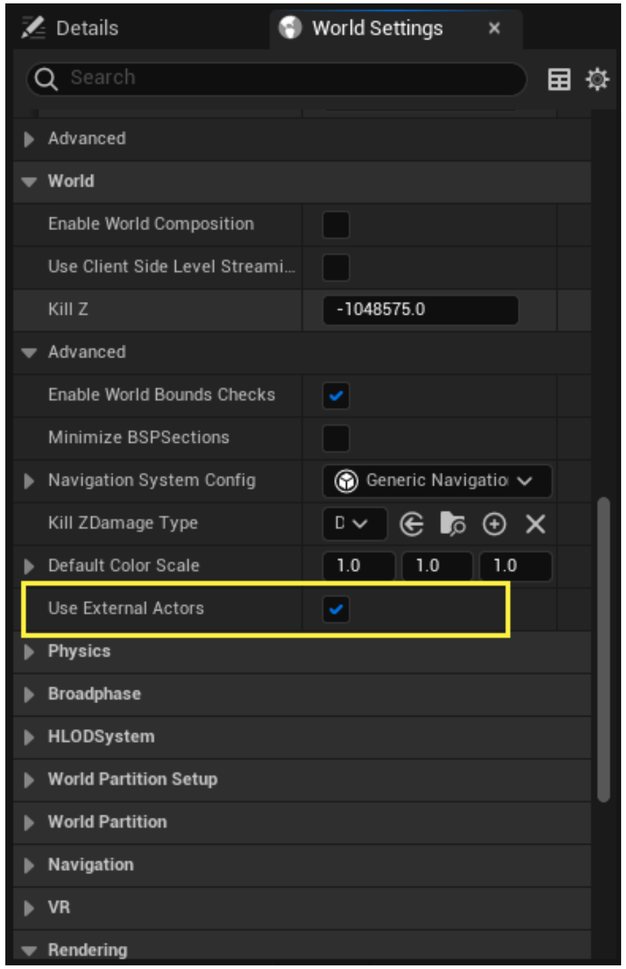
Wrap-up, summary and Q&A

- Concluding Thoughts
- Other people talk for a bit

What Are Unreal's World Partition Systems?

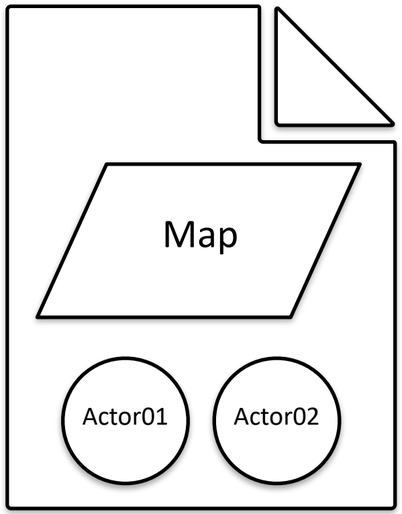
One File Per-Actor (OFPA)

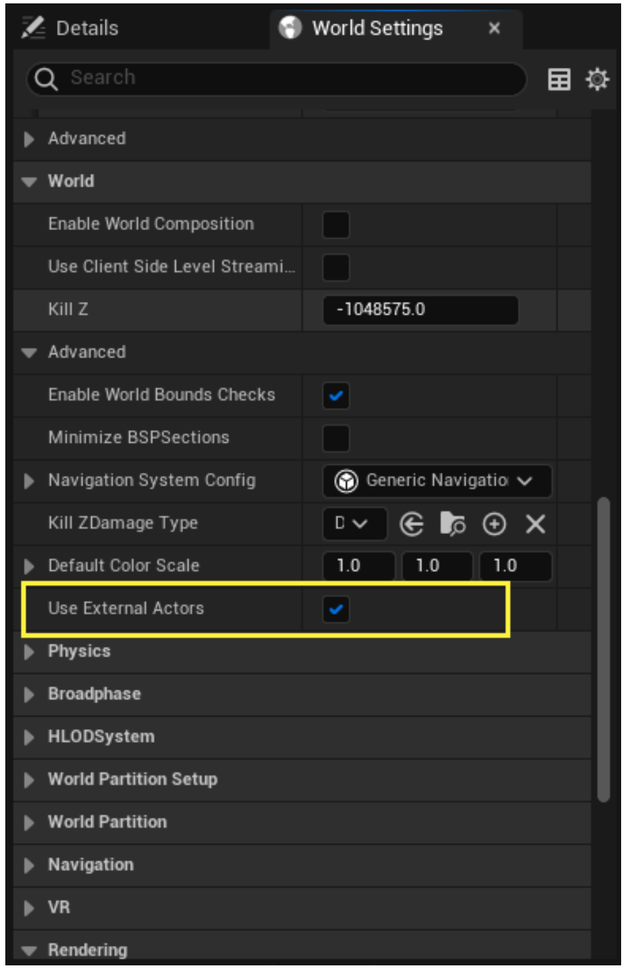
- World Partition's multi-user editing solution
- Enabled by default for World Partition levels
- Inverts the relationship between maps and actors
- Maps typically contain:
  - Settings
  - Metadata
  - Actors



One File Per-Actor (OFPA)

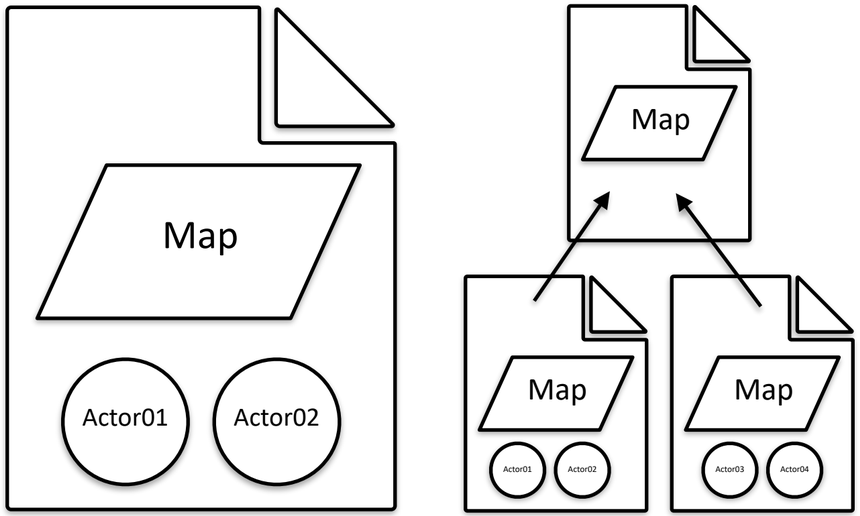
- World Partition's multi-user editing solution
- Enabled by default for World Partition levels
- Inverts the relationship between maps and actors
- Maps typically contain:
  - Settings
  - Metadata
  - Actors

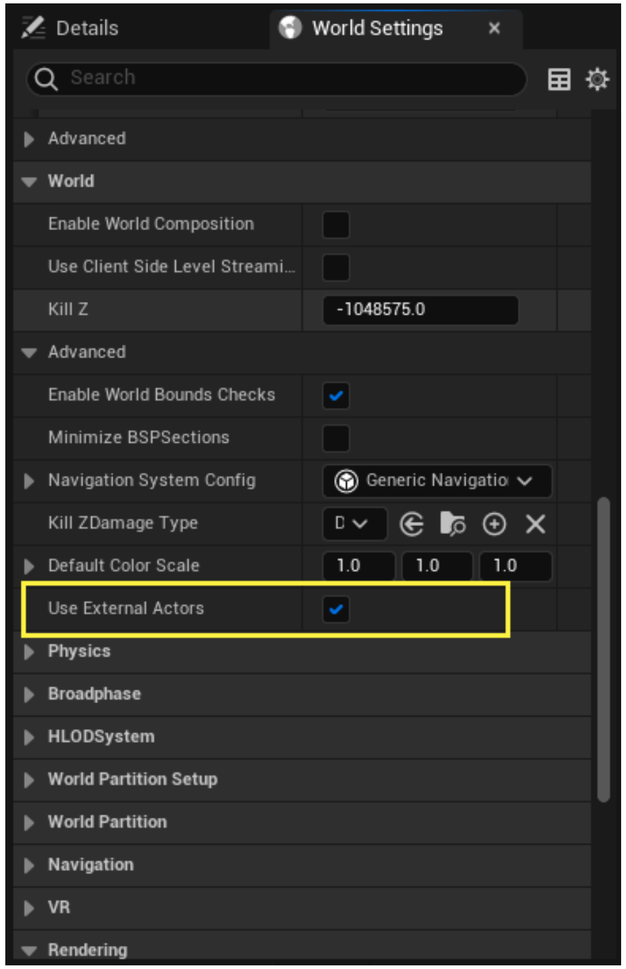




One File Per-Actor (OFPA)

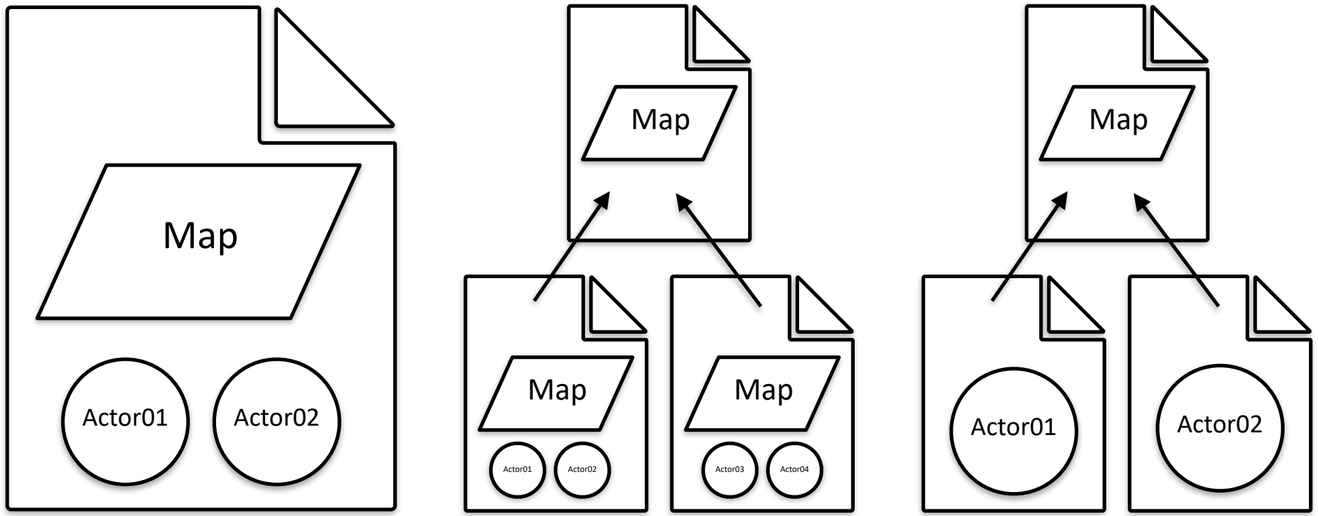
- World Partition's multi-user editing solution
- Enabled by default for World Partition levels
- Inverts the relationship between maps and actors
- Maps typically contain:
  - Settings
  - Metadata
  - Actors

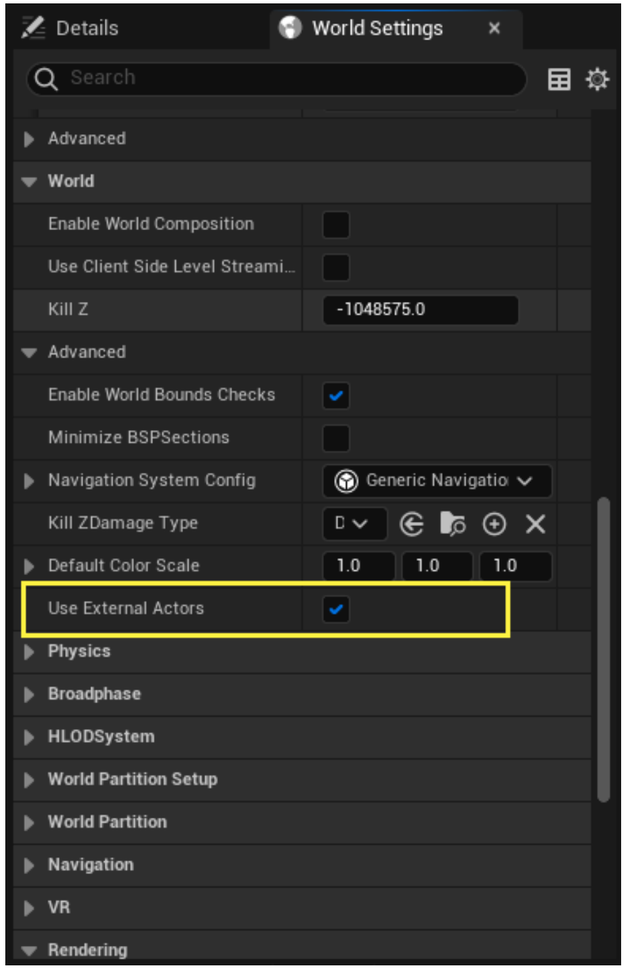




One File Per-Actor (OFPA)

- World Partition's multi-user editing solution
- Enabled by default for World Partition levels
- Inverts the relationship between maps and actors
- Maps typically contain:
  - Settings
  - Metadata
  - Actors





Anatomy Of An External Actor File

Anatomy Of An External Actor File

Map

/Game/Content/Levels/Main/L_Main.uasset

Anatomy Of An External Actor File

Map

/Game/Content/Levels/Main/L_Main.uasset

Actor

/Game/Content/__ExternalActors__/Levels/Main/L_Main/0/0C/ZNNHNWCBB9FG9K4BCQBQ4G.uasset

Anatomy Of An External Actor File

Map

/Game/Content/Levels/Main/L_Main.uasset

Actor

/Game/Content/__ExternalActors__/Levels/Main/L_Main/0/0C/ZNNHNWCBB9FG9K4BCQBQ4G.uasset

Object

/Game/Content/__ExternalObjects__/Levels/Main/L_Main/F/04/6U6DBM0R72CE8RPNB95JTO.uasset

Anatomy Of An External Actor File

Map

/Game/Content/Levels/Main/L_Main.uasset

Actor

/Game/Content/__ExternalActors__/Levels/Main/L_Main/0/0C/ZNNHNWCBB9FG9K4BCQBQ4G.uasset

Anatomy Of An External Actor File

Map

/Game/Content/Levels/Main/L_Main.uasset

Actor

/Game/Content/__ExternalActors__/Levels/Main/L_Main/0/0C/ZNNHNWCBB9FG9K4BCQBQ4G.uasset

External Actor Directory

Anatomy Of An External Actor File

Map

/Game/Content/Levels/Main/L_Main.uasset

Actor

/Game/Content/__ExternalActors__/Levels/Main/L_Main/0/0C/ZNNHNWCBB9FG9K4BCQBQ4G.uasset

External Actor Directory

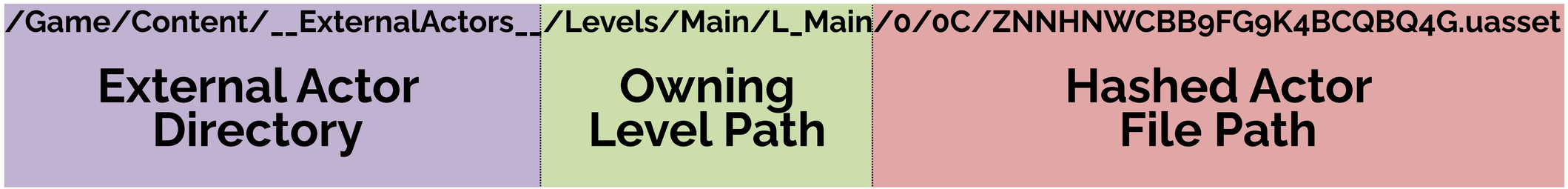
Owning Level Path

Anatomy Of An External Actor File

Map

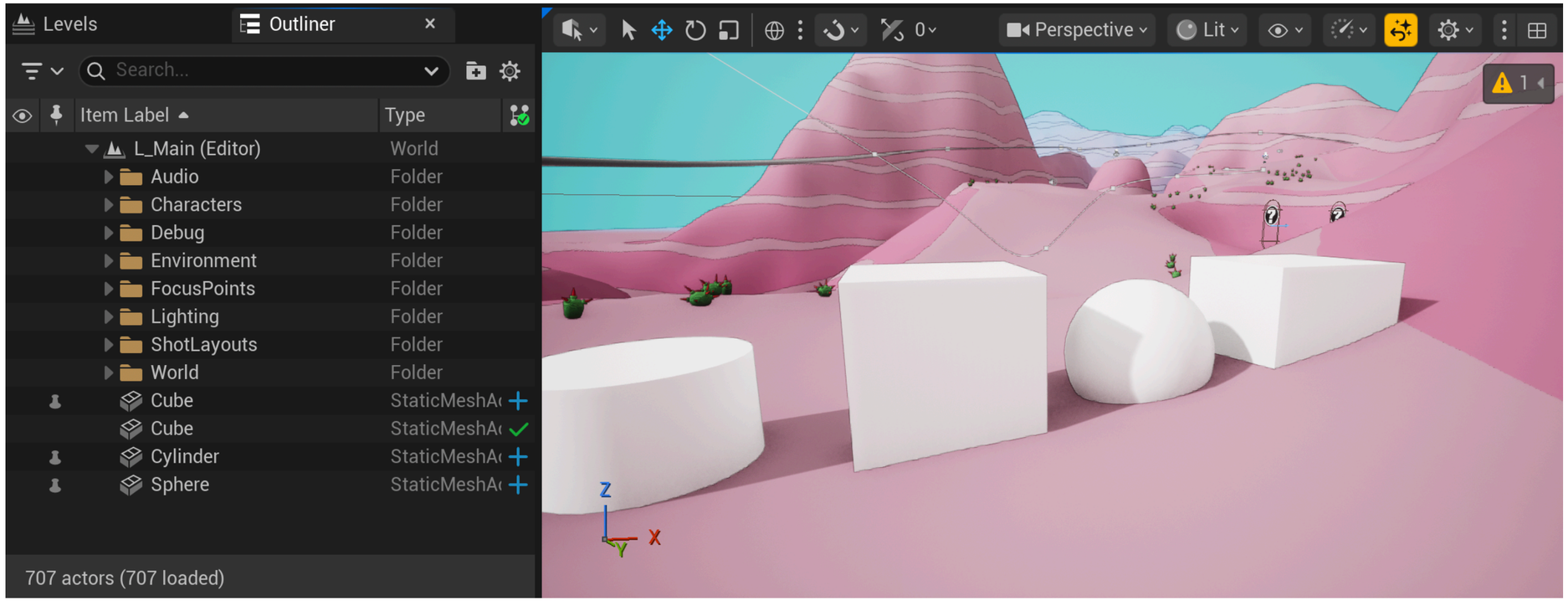
/Game/Content/Levels/Main/L_Main.uasset

Actor



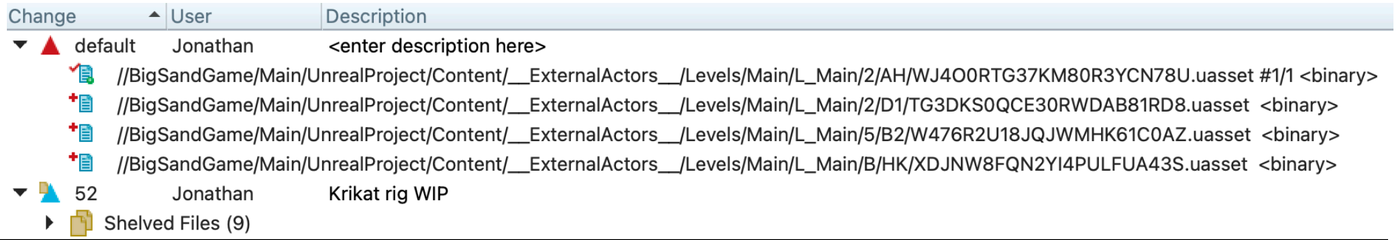
Managing External Actors

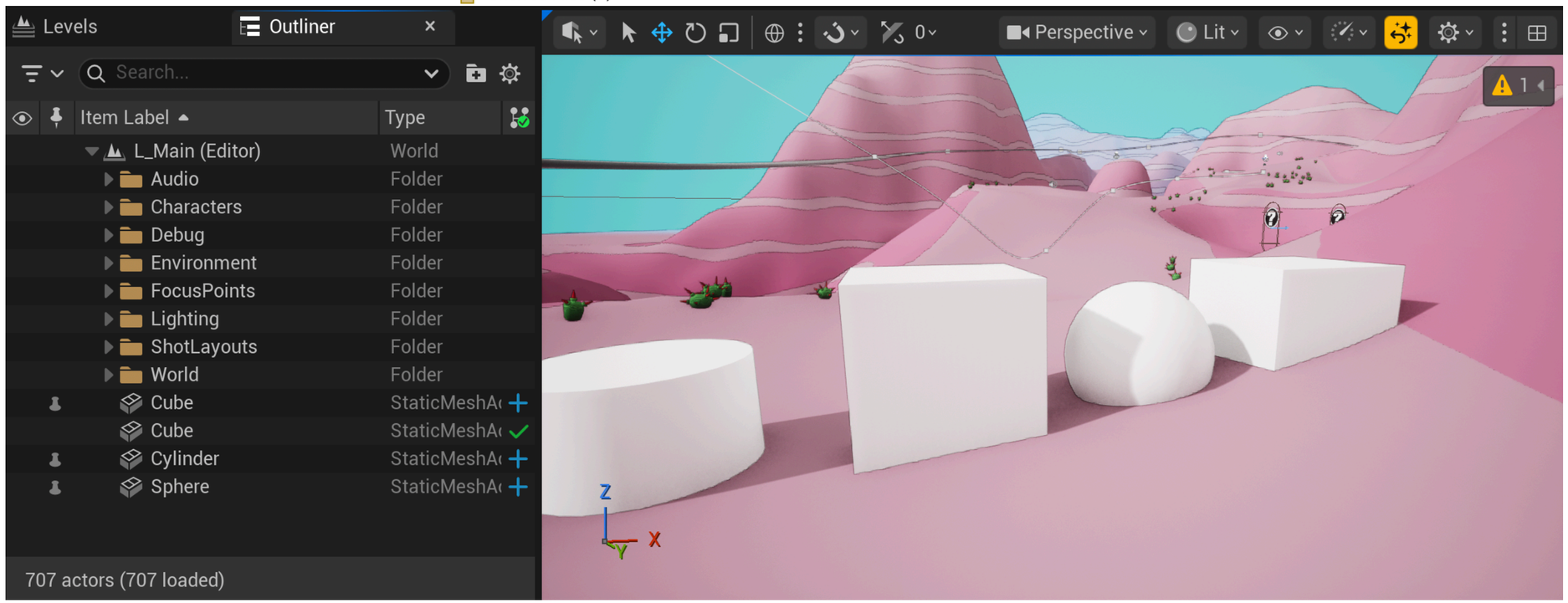
- Keeping your comfortable p4 or Git workflow can be painful
- Seems worse than it is though



Managing External Actors

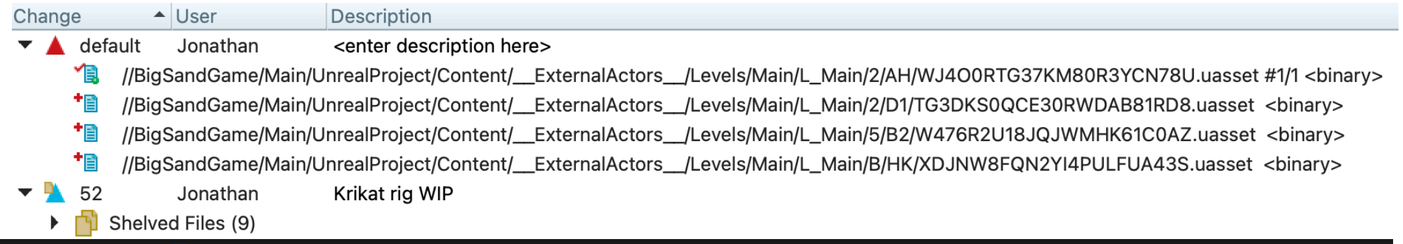
- Keeping your comfortable p4 or Git workflow can be painful
- Seems worse than it is though

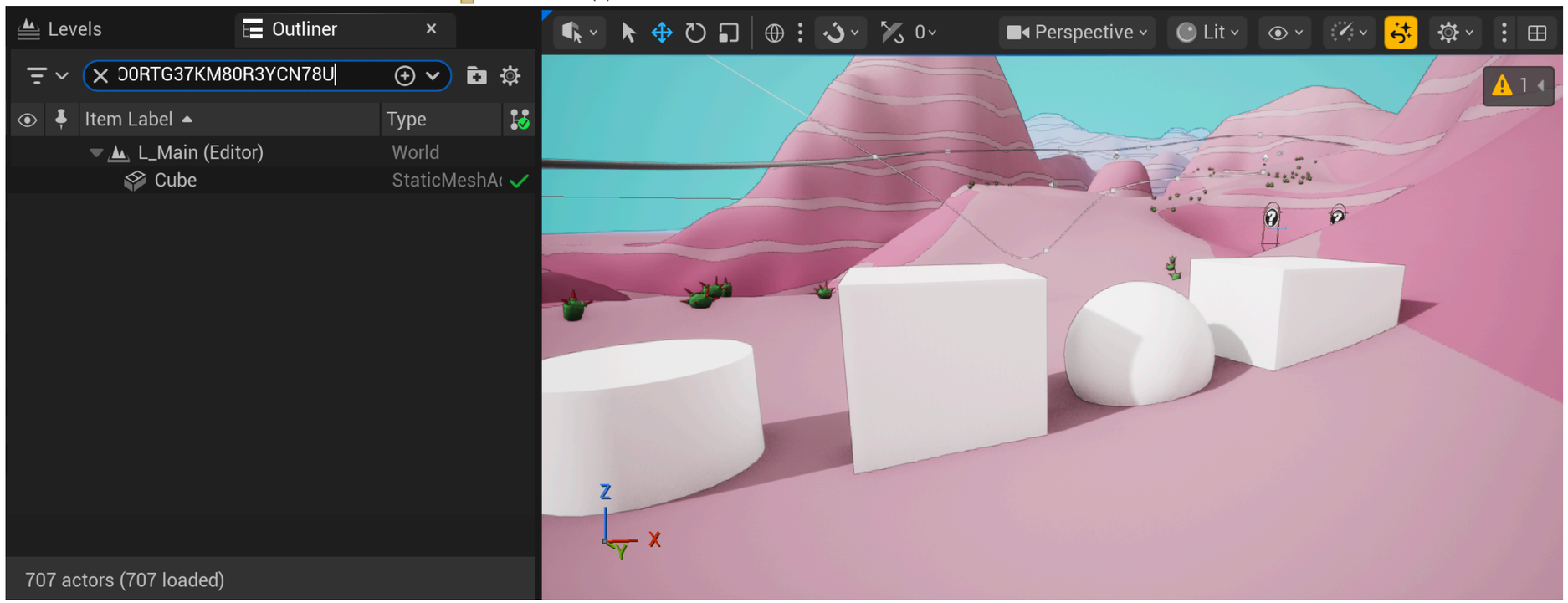




Managing External Actors

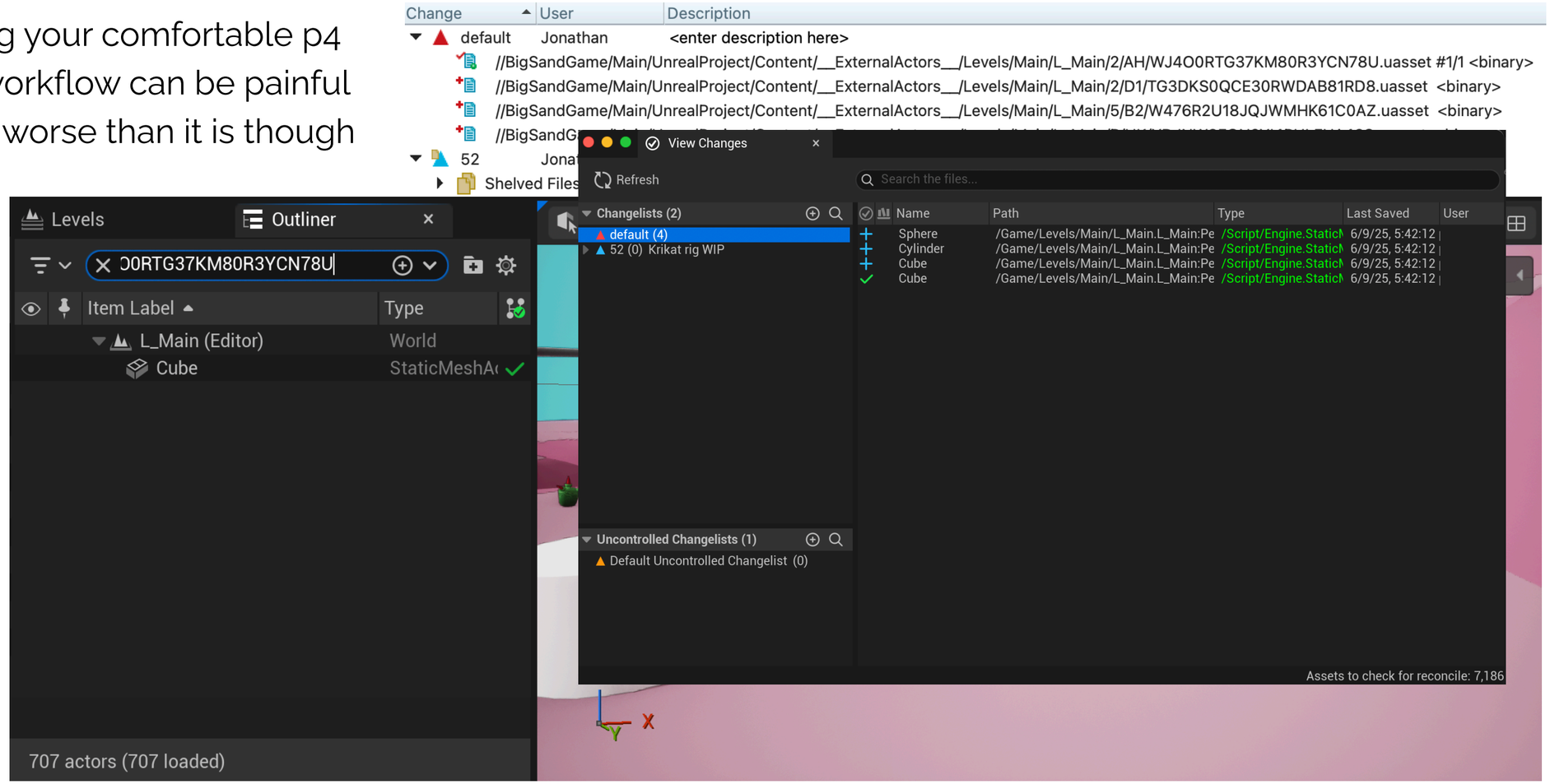
- Keeping your comfortable p4 or Git workflow can be painful
- Seems worse than it is though





Managing External Actors

- Keeping your comfortable p4 or Git workflow can be painful
- Seems worse than it is though



Workflow Considerations When Working With OFPA

Workflow Considerations When Working With OFPA

- ✓ Reduced Checkout Collisions

Workflow Considerations When Working With OFPA

Reduced Checkout Collisions

More Complicated Checkout Collisions

Workflow Considerations When Working With OFPA

- ✓ Reduced Checkout Collisions
- ✓ More Granular Checkins
- ✗ More Complicated Checkout Collisions

Workflow Considerations When Working With OFPA

- ✓ Reduced Checkout Collisions
- ✓ More Granular Checkins
- ✗ More Complicated Checkout Collisions
- ✗ Nonsense Changelist Contents

Workflow Considerations When Working With OFPA

- ✓ Reduced Checkout Collisions
- ✓ More Granular Checkins
- ✓ Great In-Editor VCS Support
- ✗ More Complicated Checkout Collisions
- ✗ Nonsense Changelist Contents

Workflow Considerations When Working With OFPA

- ✓ Reduced Checkout Collisions
- ✓ More Granular Checkins
- ✓ Great In-Editor VCS Support
- ✗ More Complicated Checkout Collisions
- ✗ Nonsense Changelist Contents
- ✗ Required In-Editor Workflows

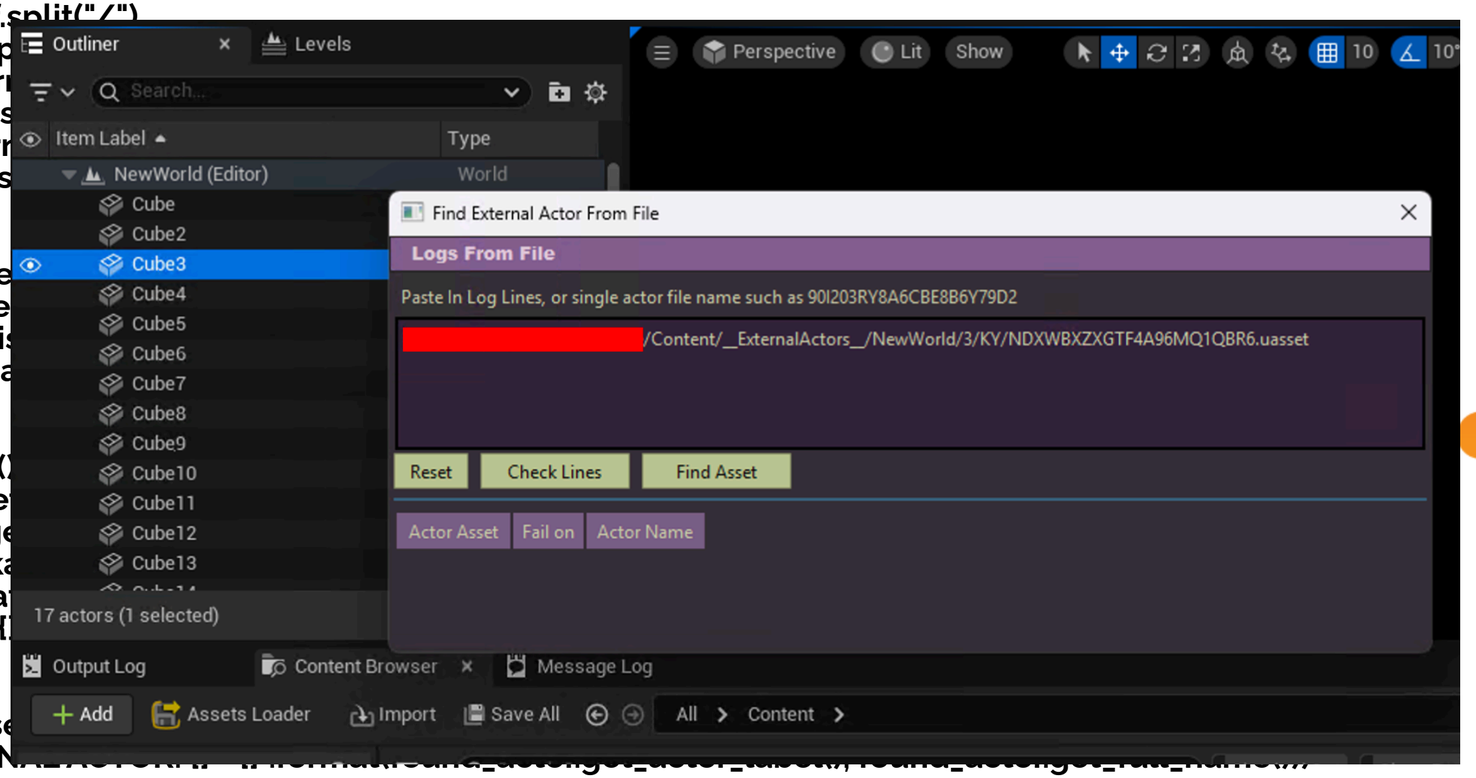
A Tool For OFPA Actor Lookup

```
def find_external_actor(search_file, log_file):
	def get_map_filename(f):
		search_file_components = f.split("/")
		map_file = search_file_components[search_file_components.index("__ExternalActors__")+1:len(search_file_components)-3]
		return "/Game/{}.umap".format("/".join(map_file))
	map_file_path = get_map_filename(search_file)
	unreal.log_warning("Deduced external actor from map: {}".format(map_file_path))
	unreal.EditorLoadingAndSavingUtils.load_map(map_file_path)

	def get_external_actor_map():
		ed_actor_subsys = unreal.get_editor_subsystem(unreal.EditorActorSubsystem)
		actors = ed_actor_subsys.get_all_level_actors()
		return {str(unreal.AssetRegistryHelpers.create_asset_data(a).package_name): a for a in actors}
	external_actor_map = get_external_actor_map()

	f = open(log_file, "w")
	for k, v in external_actor_map.items():
		package_path = unreal.AssetRegistryHelpers.create_asset_data(v).package_name
		unreal.log_warning(package_path)
		path = pathlib.Path(str(package_path).replace("/Game", unreal.Paths.project_content_dir()) + ".uasset")
		last_modified = datetime.datetime.fromtimestamp(path.stat().st_mtime)
		f.write("{}: {} - {} modified at {}\n".format(k, v.get_actor_label(), v.get_full_name(), last_modified))
	f.close()

	found_actor = external_actor_map[search_file.replace("Content", "/Game").replace(".uasset", "")]
	unreal.log_warning("FOUND EXTERNAL ACTOR: {} - {}".format(found_actor.get_actor_label(), found_actor.get_full_name()))

find_external_actor(sys.argv[1])
```

"C:\PathToUnreal\UnrealEditor-Cmd.exe" "C:\PathToProject\Project.uproject" -stdout -FullStdOutLogOutput -run=pythonscript
-script="C:\PathToScript\ofpa_find.py Content/__ExternalActors__/Levels/Main/L_Main/7/3N/J2FSD2Q4GFYXLPL134D4HJ.uasset"

A Tool For OFPA Actor Lookup

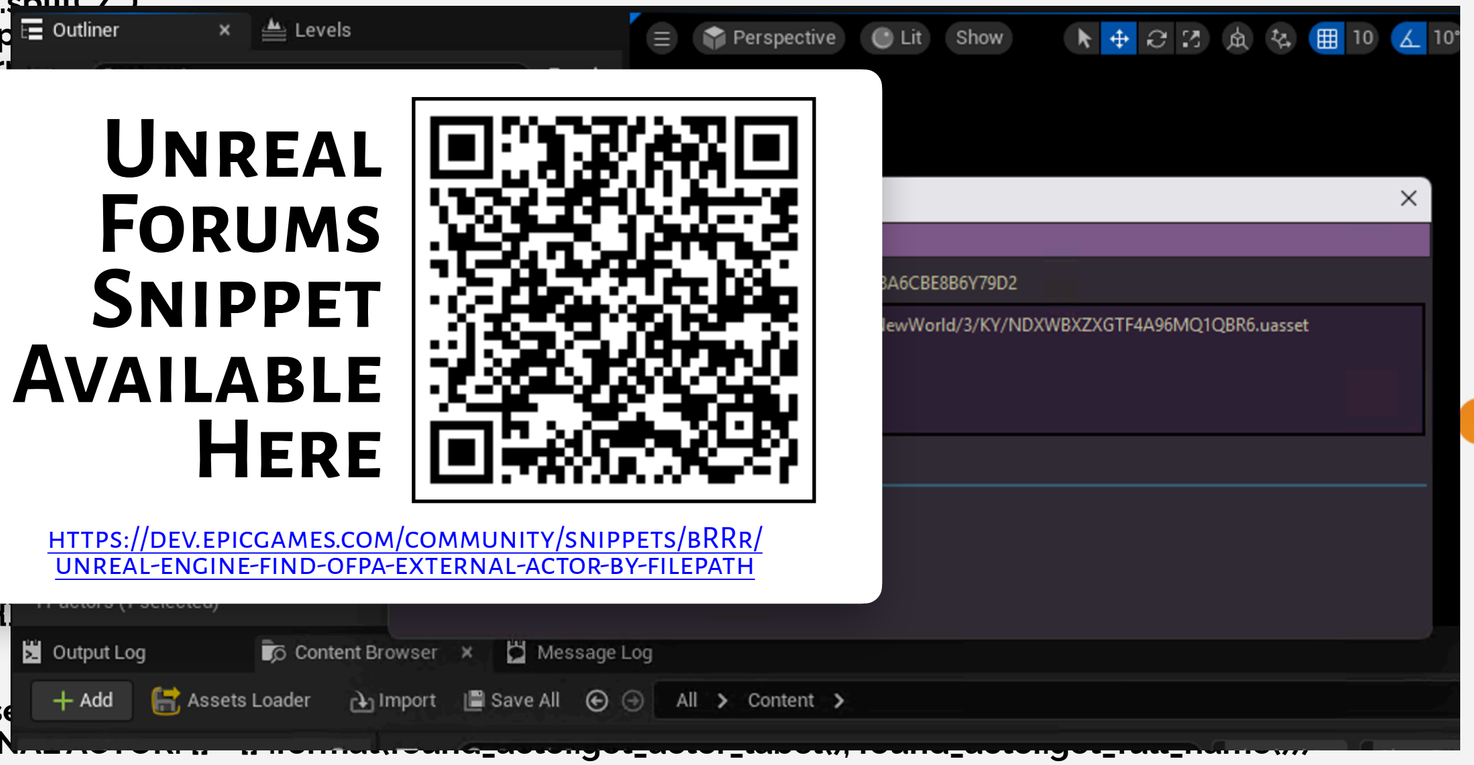
```
def find_external_actor(search_file, log_file):
        def get_map_filename(f):
                search_file_components = f.split("/")
                map_file = search_file_comp
                return "/Game/{}.umap".for
        map_file_path = get_map_filename(s
        unreal.log_warning("Deduced extern
        unreal.EditorLoadingAndSavingUtils

        def get_external_actor_map():
                ed_actor_subsys = unreal.ge
                actors = ed_actor_subsys.ge
                return {str(unreal.AssetRegis
        external_actor_map = get_external_a

        f = open(log_file, "w")
        for k, v in external_actor_map.items()
                package_path = unreal.Asse
                unreal.log_warning(package
                path = pathlib.Path(str(packa
                last_modified = datetime.da
                f.write("{}: {} - {} modified at {
        f.close()

        found_actor = external_actor_map[se
        unreal.log_warning("FOUND EXTERN

find_external_actor(sys.argv[1])
```



"C:\PathToUnreal\UnrealEditor-Cmd.exe" "C:\PathToProject\Project.uproject" -stdout -FullStdOutLogOutput -run=pythonscript
-script="C:\PathToScript\ofpa_find.py Content/__ExternalActors__/Levels/Main/L_Main/7/3N/J2FSD2Q4GFYXLPL134D4HJ.uasset"

A Tool For OFPA Actor Lookup

```
def find_external_actor(search_file, log_file):
        def get_map_filename(f):
                search_file_components = f.split("/")
                map_file = search_file_comp
                return "/Game/{}.umap" for
        map_file_path = get_map_filena
        unreal.log_warning("Deduced
        unreal.EditorLoadingAndSaving

        def get_external_actor_map():
                ed_actor_subsys = unre
                actors = ed_actor_subs
                return {str(unreal.Asse
        external_actor_map = get_exte

        f = open(log_file, "w")
        for k, v in external_actor_map.it
                package_path = unreal
                unreal.log_warning(pa
                path = pathlib.Path(str(
                last_modified = datetim
                f.write("{}: {} - {} modified at {
        f.close()

        found_actor = external_actor_map[se
        unreal.log_warning("FOUND EXTERN

find_external_actor(sys.argv[1])
```



Unreal Forums Snippet Available Here

https://dev.epicgames.com/community/snippets/bRRr/unreal-engine-find-ofpa-external-actor-by-filepath

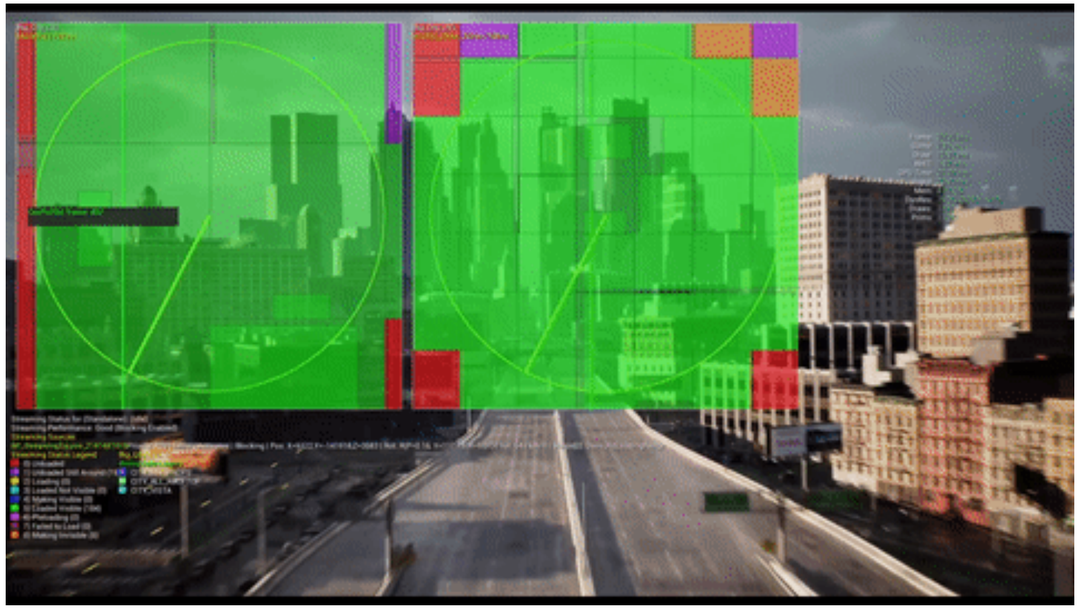
```
"C:\PathToUnreal\UnrealEditor-Cmd.exe" "C:\PathToProject\Project.uproject" -stdout -FullStdOutLogOutput -run=pythonscript
                                        -script="C:\PathToScript\ofpa_find.py Content/__ExternalActors__/Levels/Main/L_Main/7/3N/J2FSD2Q4GFYXLPL134D4HJ.uasset"
```

Grids

- Open worlds typically divvy up the world into “chunks” that can be streamed in and out as the player navigates the game world

- A world partitioned map may contain one or more **grids**, which each contain automatically portioned **cells** across which content is divided

- Thought should be put into the initial configuration which should account for:
  - Content density
  - Available memory
  - World layout
  - Player movement speed
- Avoid micro-management



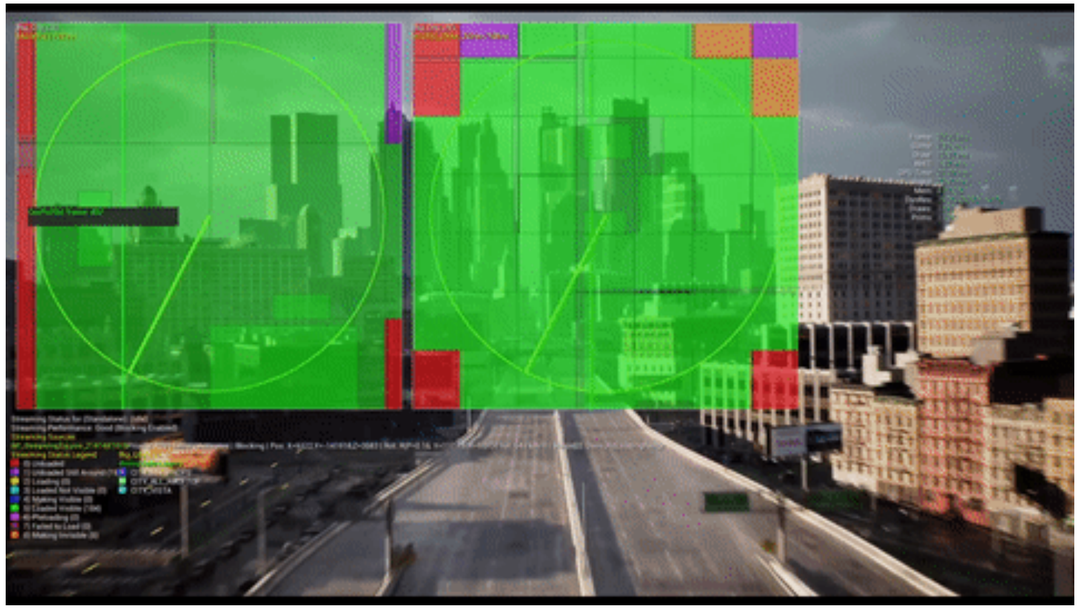
“World Partition is an automatic data management and distance-based level streaming system … storing your world in a single persistent level separated into grid cells” - Epic’s World Building Guide

Grids

- Open worlds typically divvy up the world into “chunks” that can be streamed in and out as the player navigates the game world

- A world partitioned map may contain one or more **grids**, which each contain automatically portioned **cells** across which content is divided

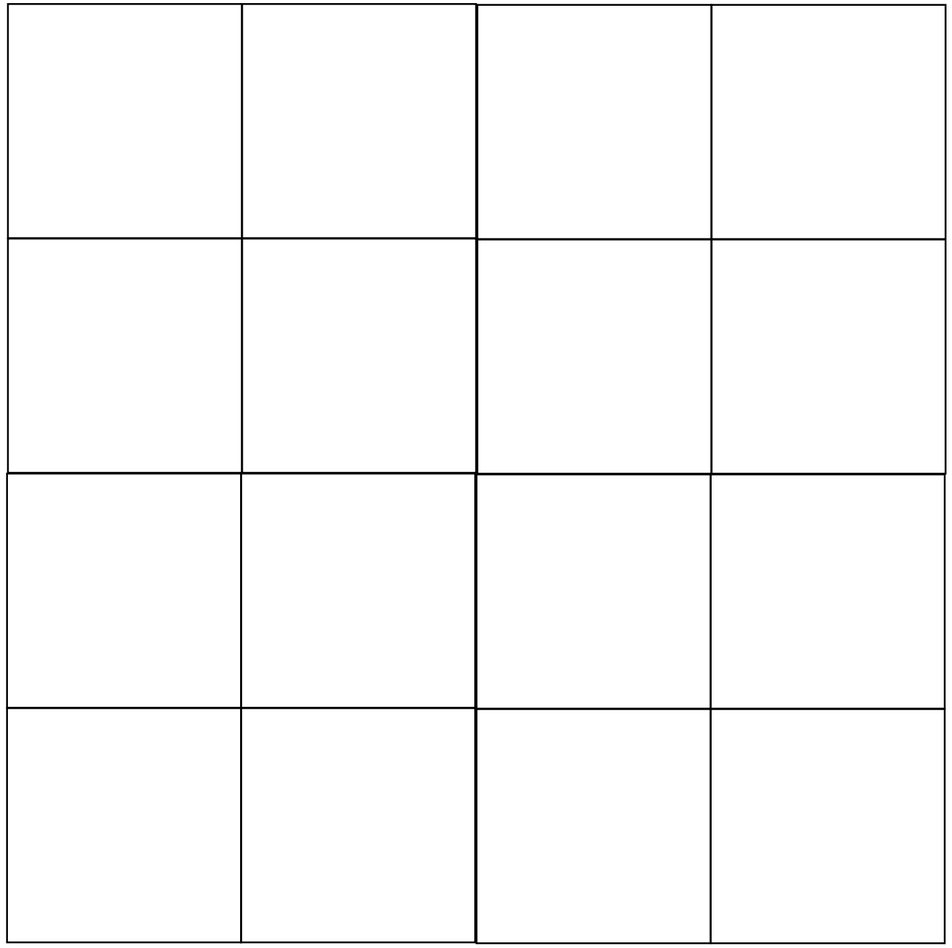
- Thought should be put into the initial configuration which should account for:
  - Content density
  - Available memory
  - World layout
  - Player movement speed
- Avoid micro-management



“World Partition is an automatic data management and distance-based level streaming system … storing your world in a single persistent level separated into grid cells” - Epic’s World Building Guide

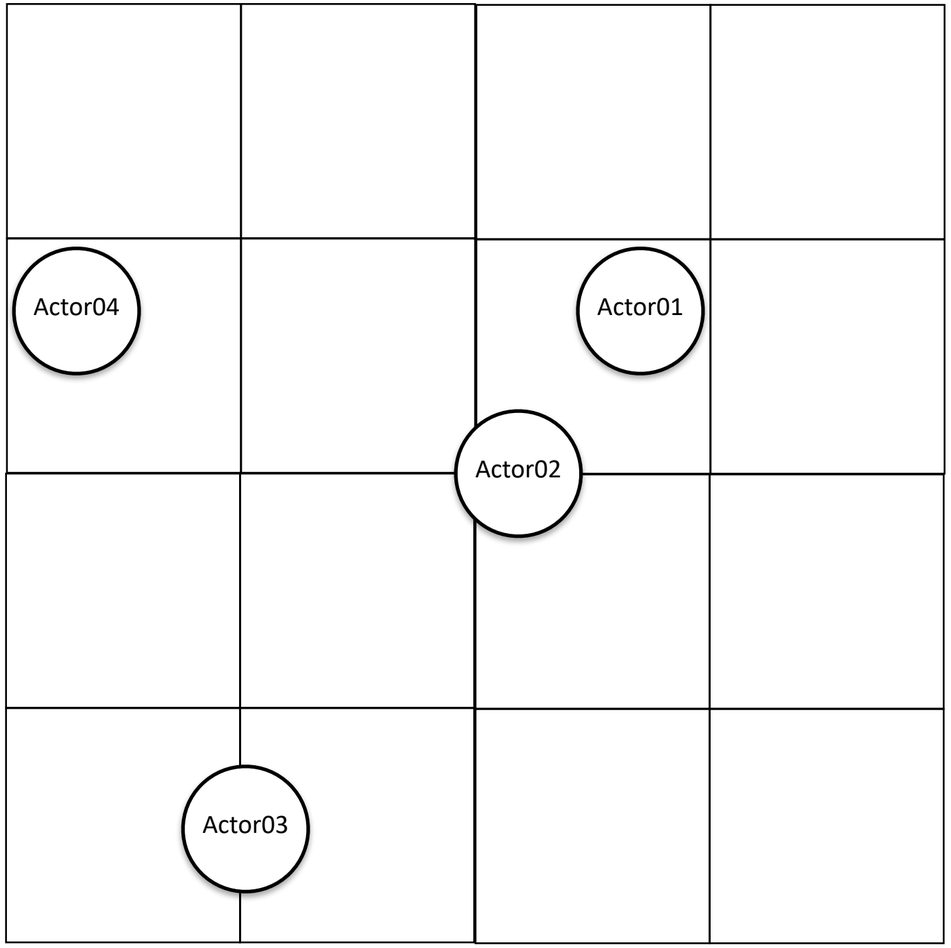
Streaming Generation

- Process of building the data structures to assign actors into logical groups for streaming purposes
- All actors will be assigned to a streaming level based a number of factors, including the cell of the runtime grid they fall within
- The resulting work is used to marshal actors to the “streaming levels” used at runtime
- Occurs during cook in builds and on-the-fly during PIE sessions



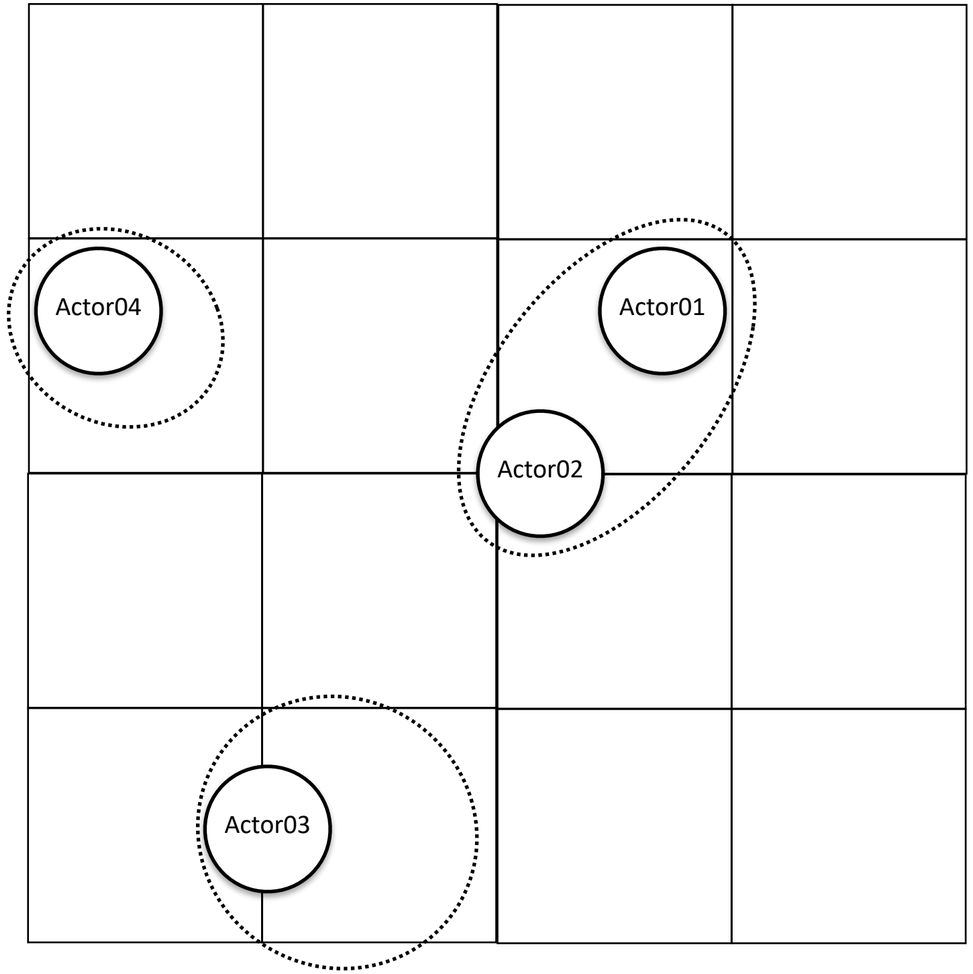
Streaming Generation

- Process of building the data structures to assign actors into logical groups for streaming purposes
- All actors will be assigned to a streaming level based a number of factors, including the cell of the runtime grid they fall within
- The resulting work is used to marshal actors to the “streaming levels” used at runtime
- Occurs during cook in builds and on-the-fly during PIE sessions



Streaming Generation

- Process of building the data structures to assign actors into logical groups for streaming purposes
- All actors will be assigned to a streaming level based a number of factors, including the cell of the runtime grid they fall within
- The resulting work is used to marshal actors to the “streaming levels” used at runtime
- Occurs during cook in builds and on-the-fly during PIE sessions

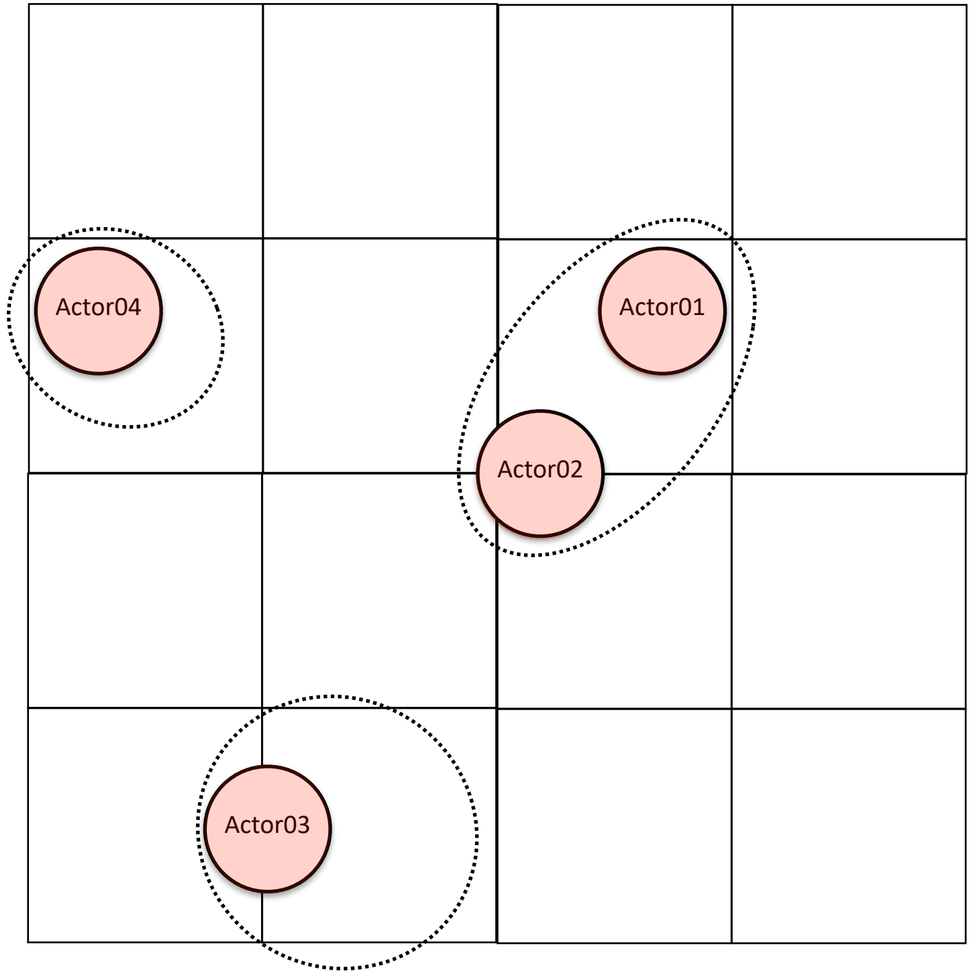


Runtime Grid Streaming Rules

- Streaming levels are brought up for consideration when their requirements are met

- Streaming states can be one of:
  Unloaded, **Loaded**, or **Activated**

- **BeginPlay** will be fired as soon as they are streamed into the world
- **EndPlay** will be fired as they are streamed out (with the reason of **RemovedFromWorld**)

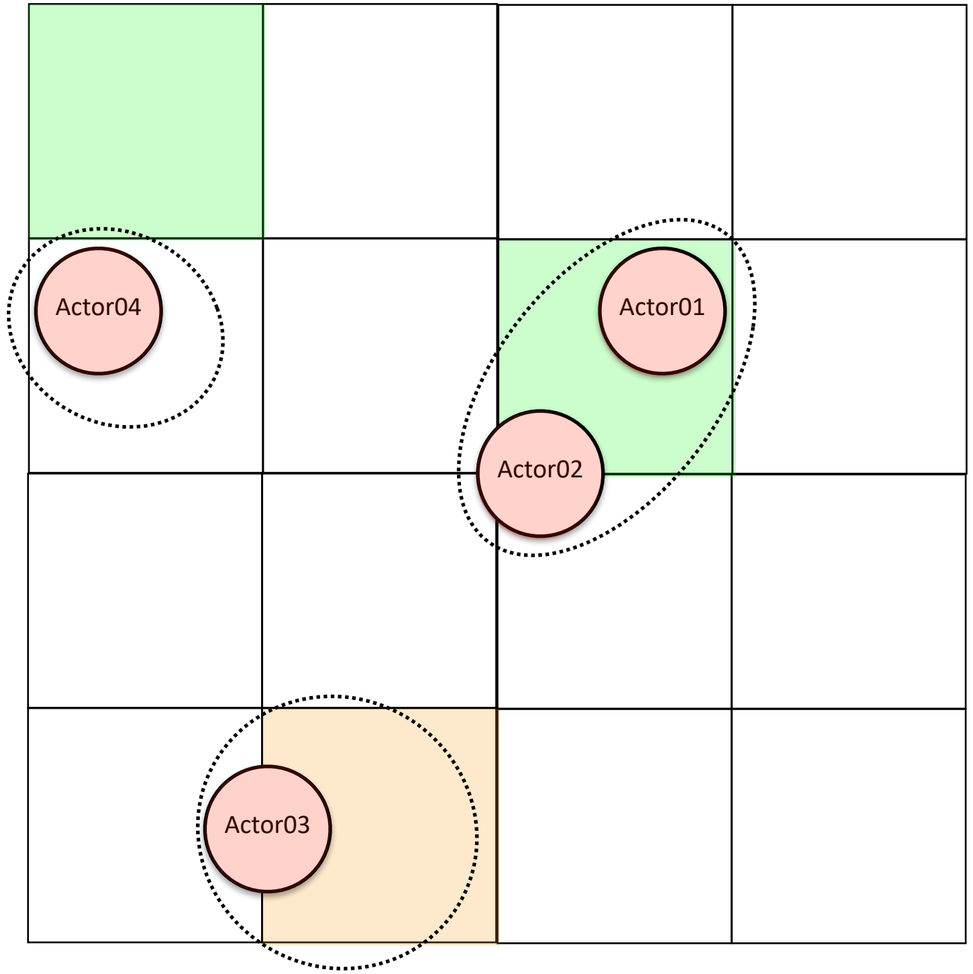


Runtime Grid Streaming Rules

- Streaming levels are brought up for consideration when their requirements are met

- Streaming states can be one of:
  Unloaded, **Loaded**, or **Activated**

- **BeginPlay** will be fired as soon as they are streamed into the world
- **EndPlay** will be fired as they are streamed out (with the reason of **RemovedFromWorld**)

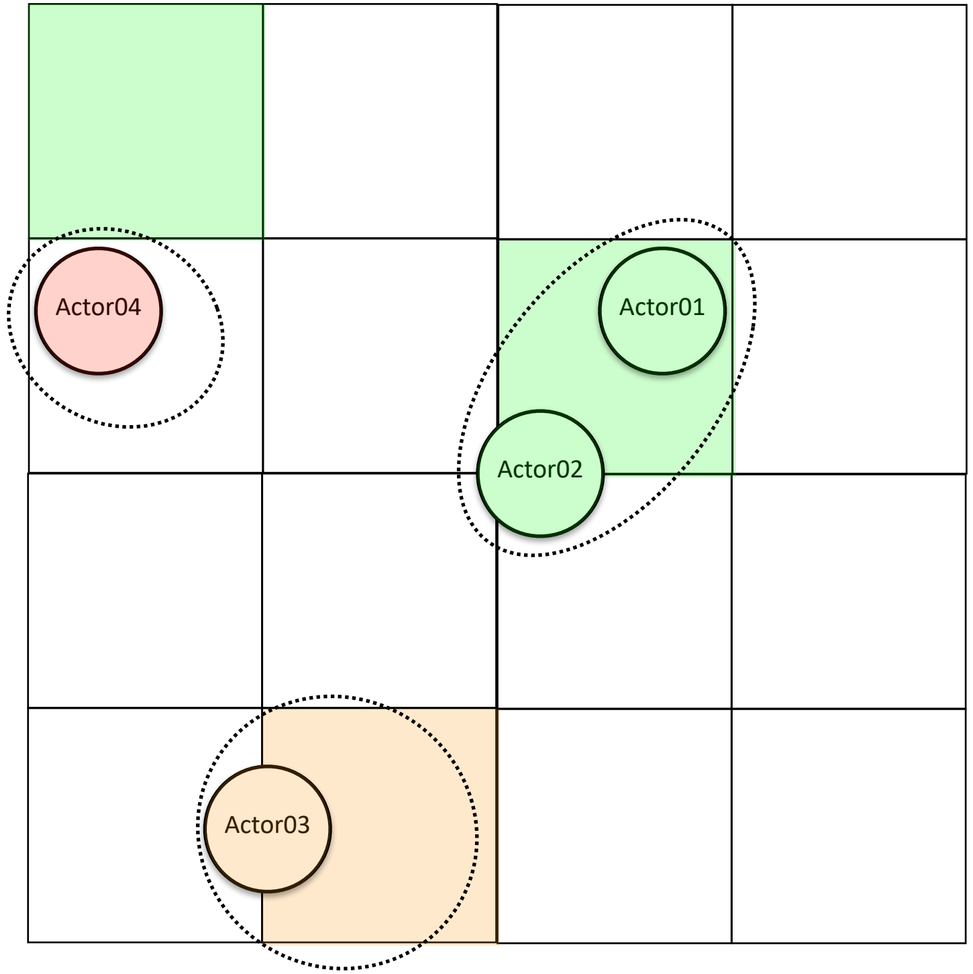


Runtime Grid Streaming Rules

- Streaming levels are brought up for consideration when their requirements are met

- Streaming states can be one of:
  Unloaded, **Loaded**, or **Activated**

- **BeginPlay** will be fired as soon as they are streamed into the world
- **EndPlay** will be fired as they are streamed out (with the reason of **RemovedFromWorld**)

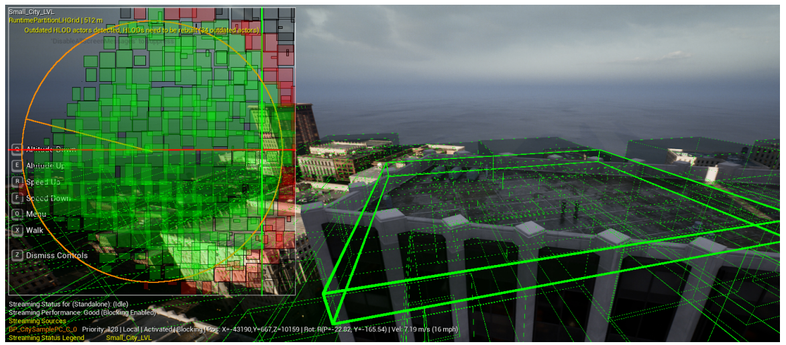


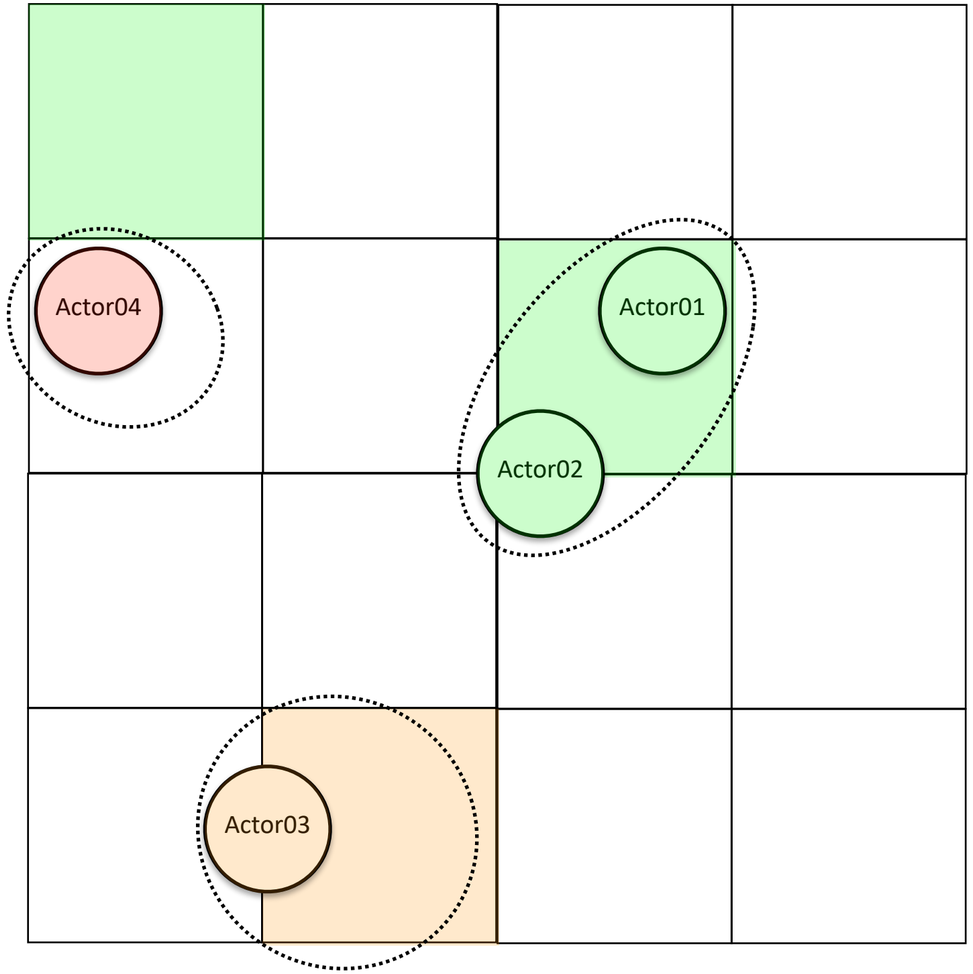
Runtime Grid Streaming Rules

- Streaming levels are brought up for consideration when their requirements are met

- Streaming states can be one of:
  Unloaded, **Loaded**, or **Activated**

- **BeginPlay** will be fired as soon as they are streamed into the world
- **EndPlay** will be fired as they are streamed out (with the reason of **RemovedFromWorld**)





Dehydrating Actors For Streaming

right up there with other programmer phrases like “ensure all children are orphaned before destroying world”

- Actor descriptors are used to represent actors within the World Partition system
- They are managed as: **FWorldPartitionActorDescriptor** and **FWorldPartitionActorDescriptorInstance**
- Effectively handled separately from the actors that are managed by the world itself
- They retain a relatively limited set of instance information

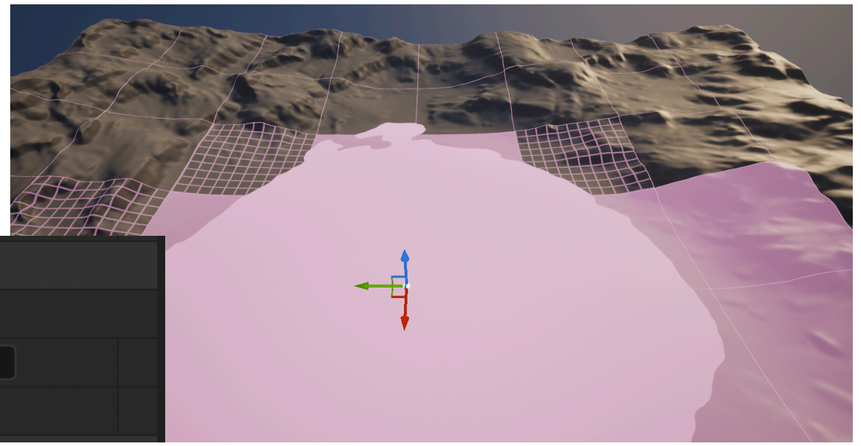
```
// Persistent
FGuid                       Guid;
FTopLevelAssetPath          BaseClass;
FTopLevelAssetPath          NativeClass;    // Not serialized, comes from initialization data
FName                       ActorPackage;   // Not serialized, comes from initialization data
FSoftObjectPath             ActorPath;      // Not serialized, comes from initialization data
FName                       ActorLabel;
FTransform                  ActorTransformRelative;        // Relative transform for actors with parent actor, actor-to-world transform otherwise
FWorldPartitionRelativeBounds RuntimeBoundsRelative;  // Runtime bounds relative to actor-to-world transform
FWorldPartitionRelativeBounds EditorBoundsRelative;   // Editor bounds relative to actor-to-world transform
FName                       RuntimeGrid;
bool                        bIsSpatiallyLoaded;
bool                        bActorIsEditorOnly;
bool                        bActorIsRuntimeOnly;
bool                        bActorIsMainWorldOnly;
bool                        bActorIsHLODRelevant;
bool                        bActorIsListedInSceneOutliner;
bool                        bIsUsingDataLayerAsset; // Used
FSoftObjectPath             HLODLayer;
TArray<FName>               DataLayers;
FSoftObjectPath             ExternalDataLayerAsset;
TArray<FGuid>               References;
TArray<FGuid>               EditorOnlyReferences; // Refere
TArray<FName>               Tags;
FPropertyPairsMap           Properties;
FName                       FolderPath;
FGuid                       FolderGuid;
FGuid                       ParentActor; // Used to validat
FGuid                       ContentBundleGuid;

// Transient
UClass*                     ActorNativeClass;
FName                       ActorName;
FString                     ActorNameString;
FString                     ActorLabelString;
FString                     ActorDisplayClassNameString;
UActorDescContainer*        Container;
FTransform                  ActorTransform; // Current acto
FBox                        RuntimeBounds;  // Current runt
FBox                        EditorBounds;   // Current edit
bool                        bIsDefaultActorDesc;
bool                        bHasValidRelativeBounds;
```

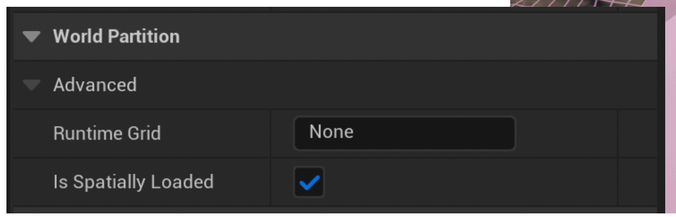
```
protected:
    TObjectPtr<UActorDescContainerInstance>     ContainerInstance;

    mutable uint32                              SoftRefCount;
    mutable uint32                              HardRefCount;
    TOptional<FDataLayerInstanceNames>          ResolvedDataLayerInstanceNames;
    bool                                        bIsForcedNonSpatiallyLoaded;
    bool                                        bIsRegisteringOrUnregistering;
    mutable FText*                              UnloadedReason;
    mutable int32                               AsyncLoadID;

    // Instancing Path if set
    TOptional<FSoftObjectPath>                  ActorPath;

    mutable TWeakObjectPtr<AActor>              ActorPtr;
    FWorldPartitionActorDesc*                   ActorDesc;

    TObjectPtr<UActorDescContainerInstance>     ChildContainerInstance;
```

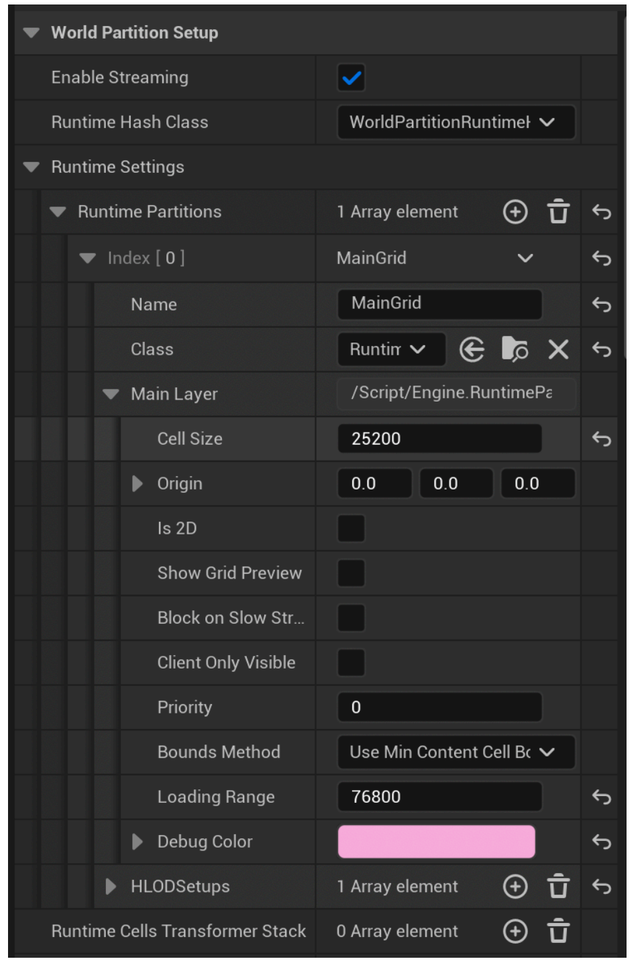
Grids & Actor Streaming Configuration

- **World Settings:**
  - Runtime Hash Class
  - Runtime Partitions:
    - Cell size, origin, priority, loading range, slow streaming block, etc.
    - Top partition is considered default grid
- **Actor Settings:**
  - Runtime grid property (WHY IS IT FREE TEXT?? *map check helps*)
  - IsSpatiallyLoaded (hard to keep track of, can create ref chain)

- Use grid previews and block on slow streaming to debug setup

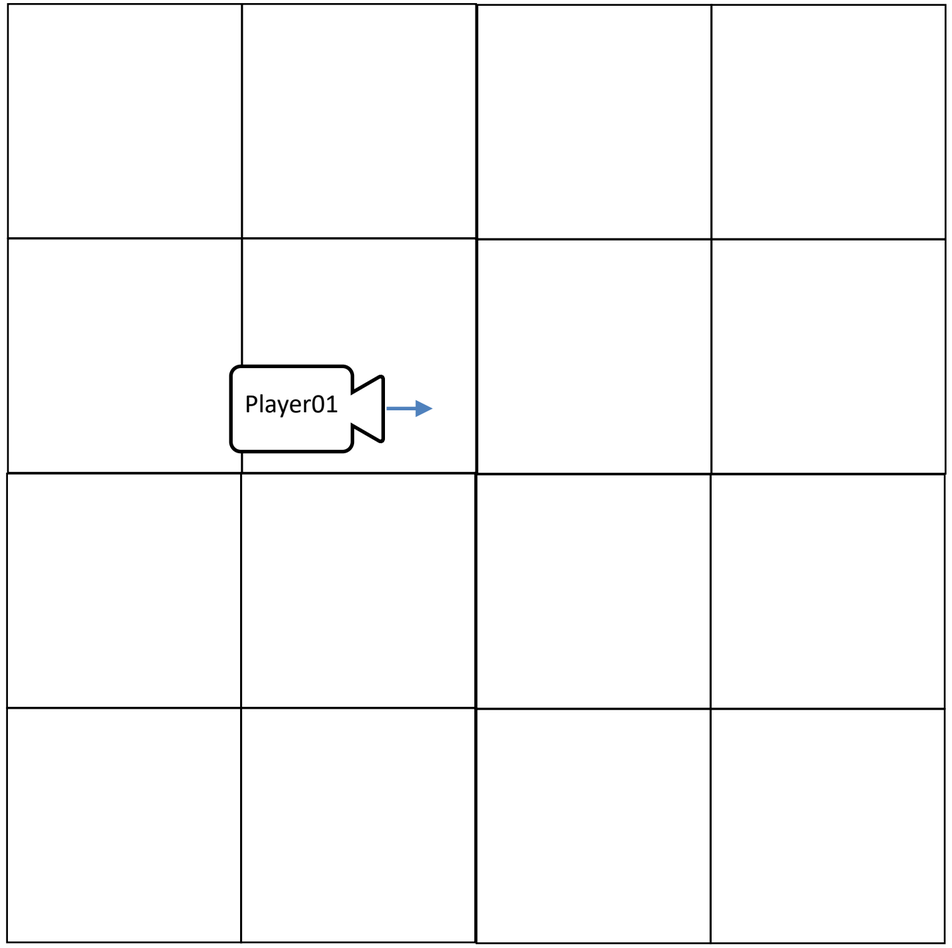






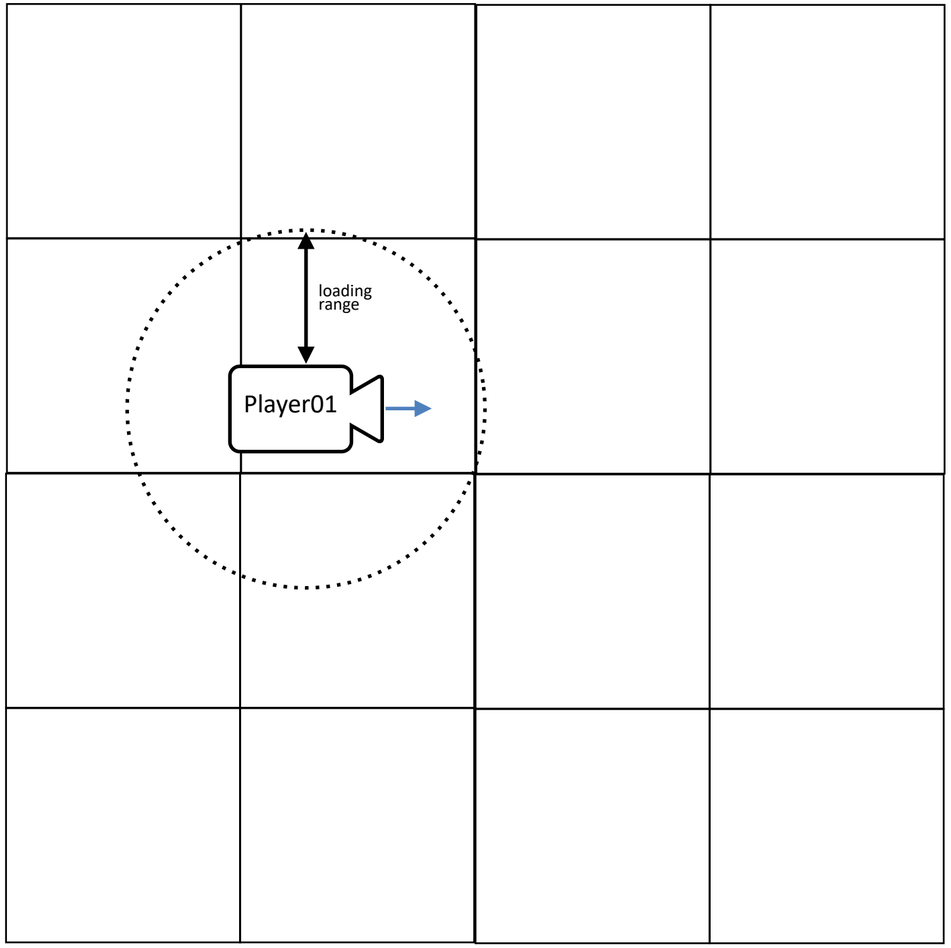
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



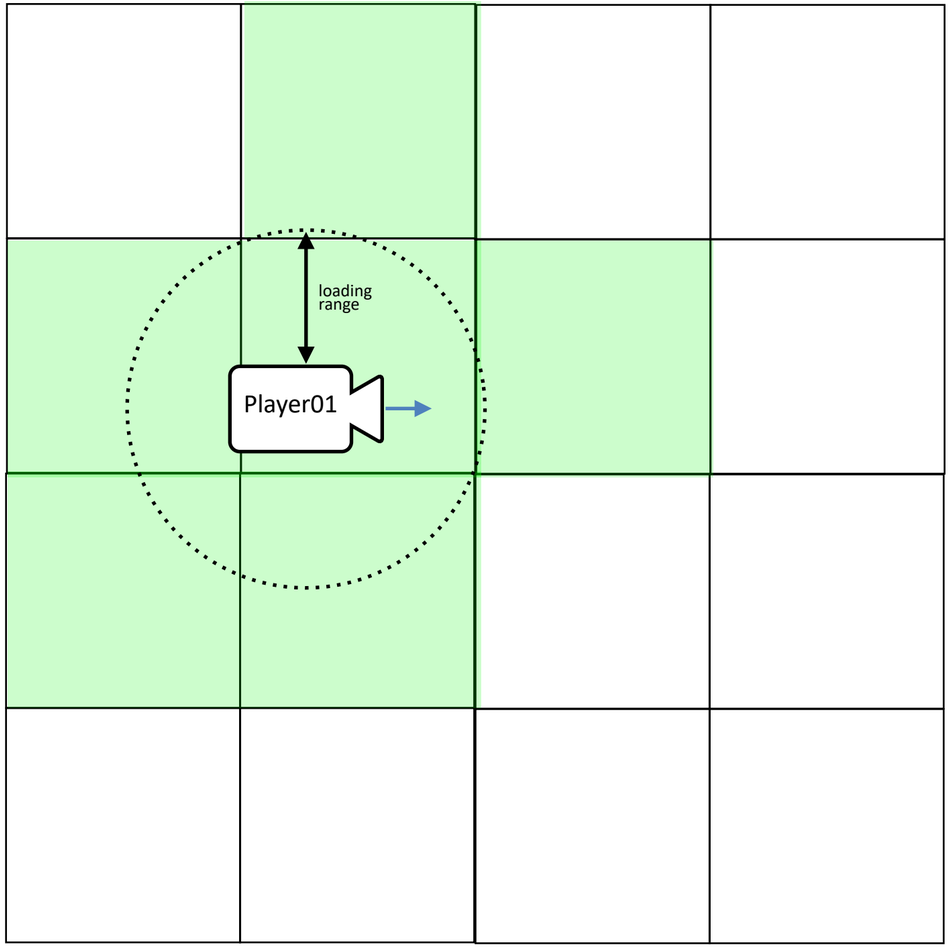
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



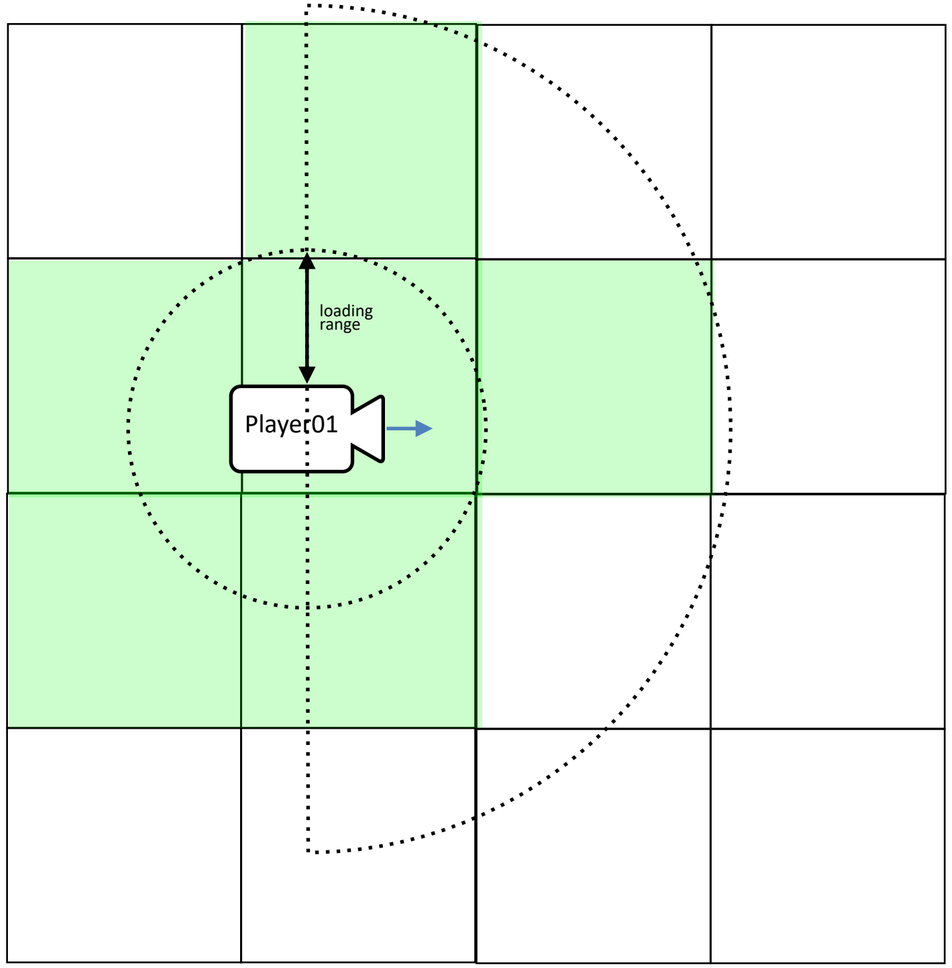
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



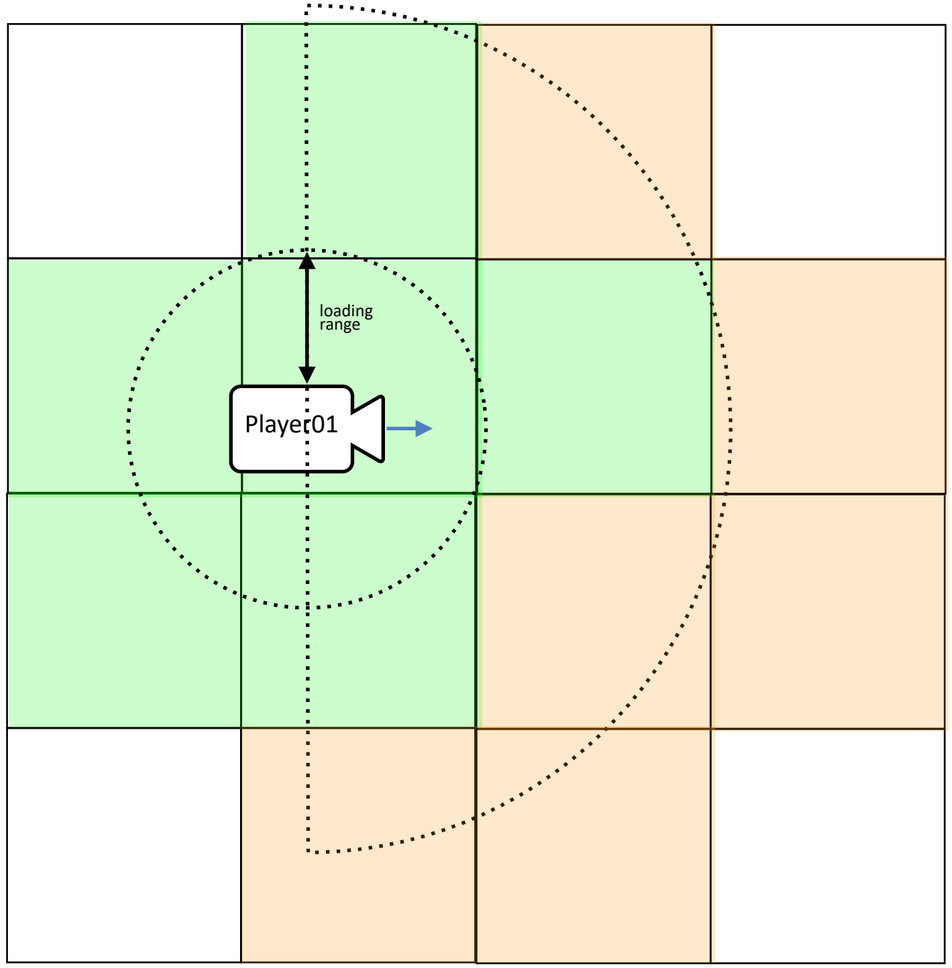
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



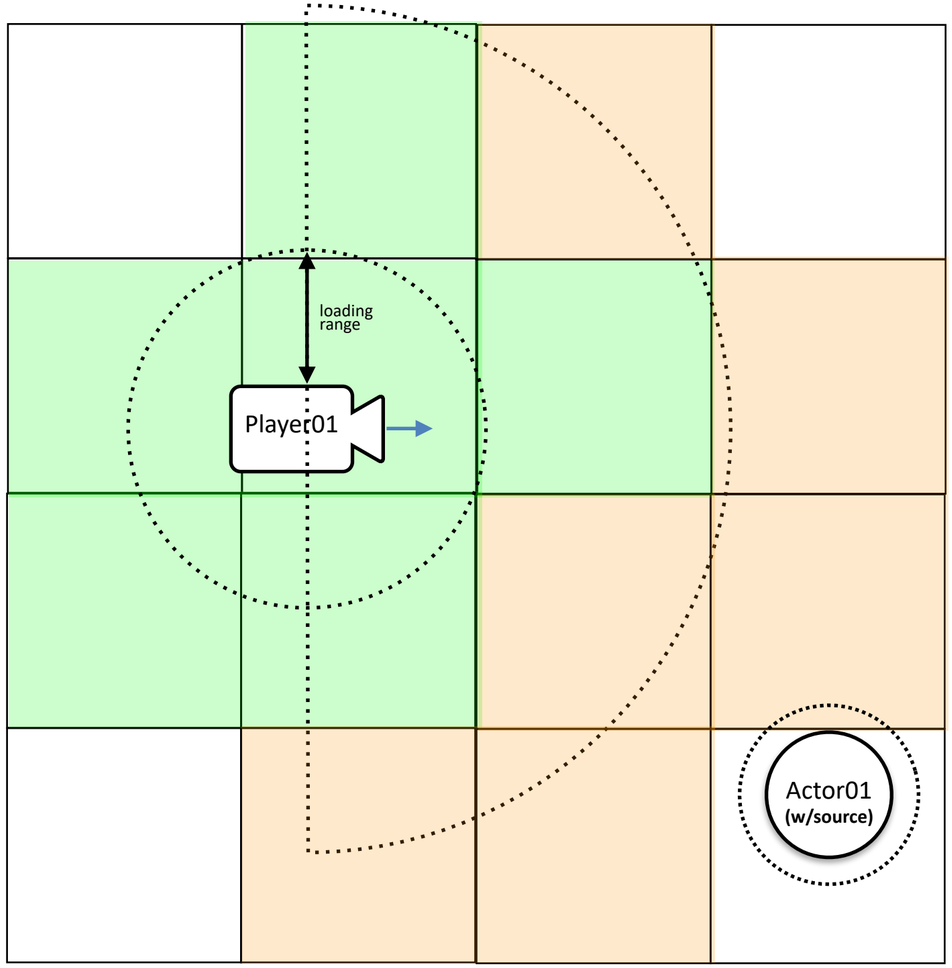
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



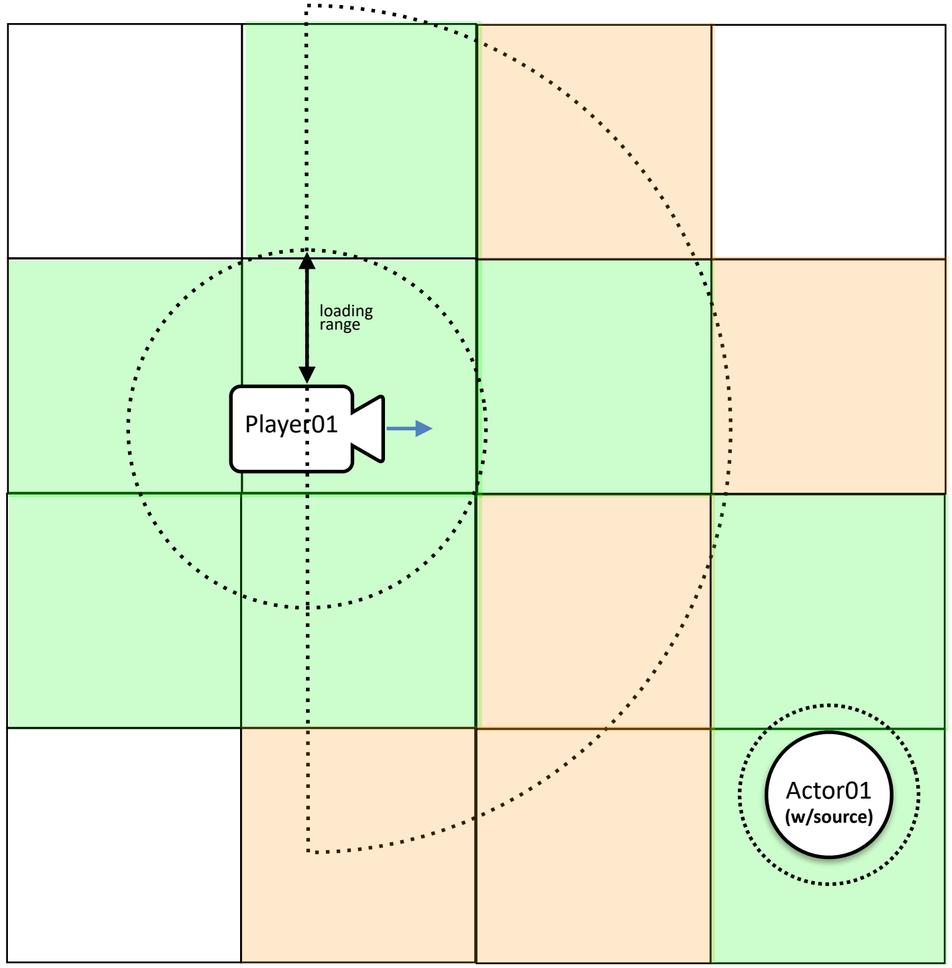
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



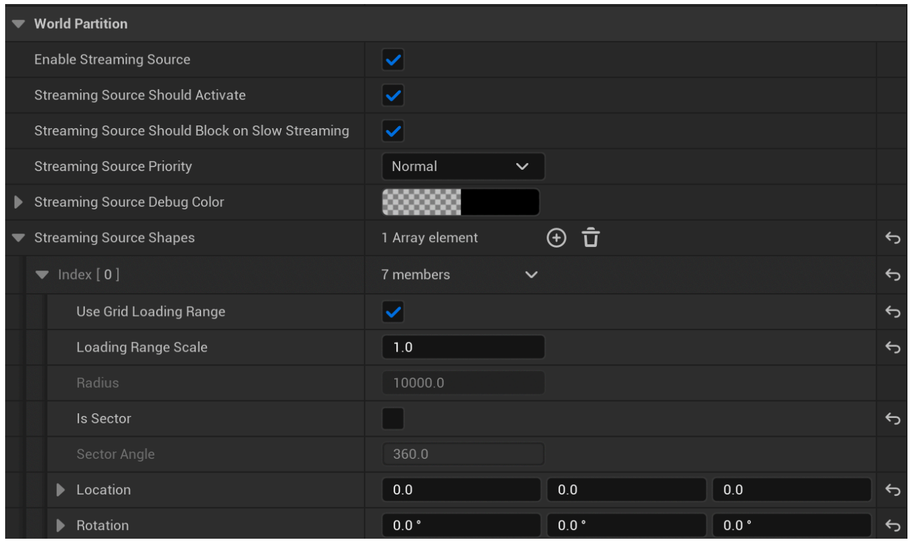
Streaming Sources

- Streaming sources define the range of grid cells to consider at runtime for streaming
- Streaming sources must register to the World Partition subsystem as a streaming source provider
- Sources can override default grid loading ranges and define streaming source shapes for more complex behaviour management
- Can be setup to target different grids and states
- Having multiple streaming sources can be useful when preloading areas of the world such as:
  - Warming up an impending teleport
  - Faraway sequence cuts



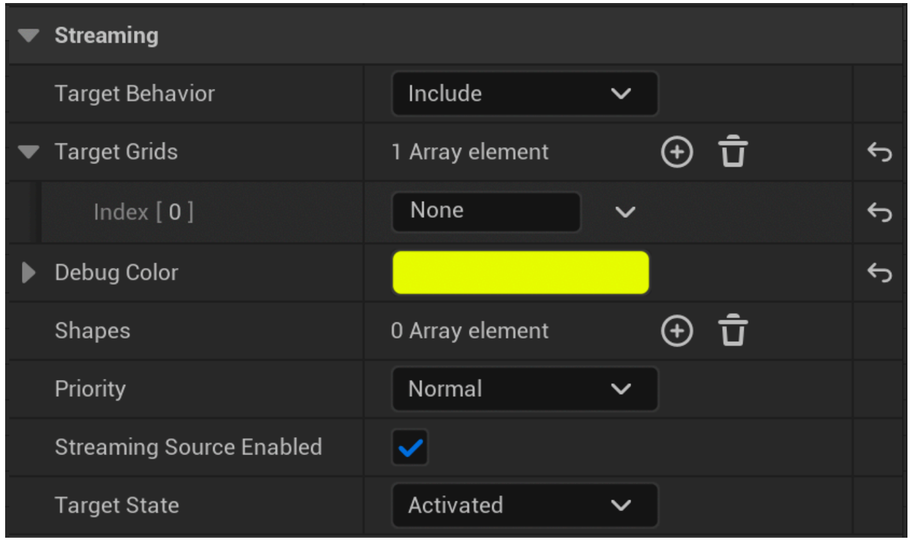
Configuring Streaming Sources

- **APlayerController** registers itself as a streaming source by default using the player camera manager's view target actor's "eyes"
- **UWorldPartitionStreamingSourceComponent** provides its owning actor's location/rotation



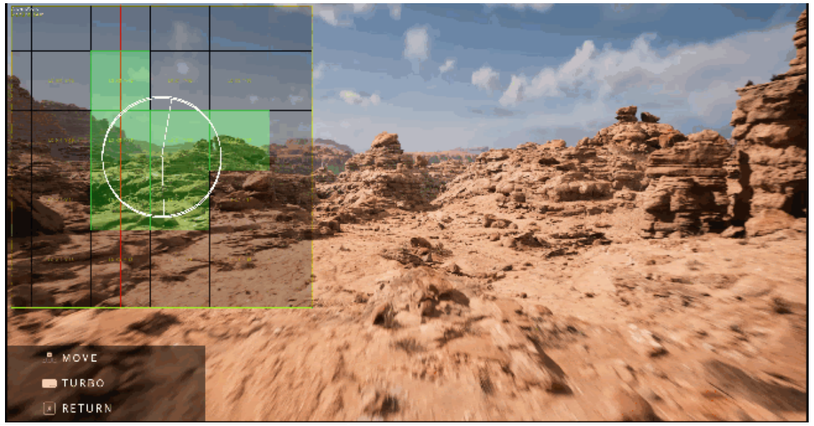
Configuring Streaming Sources

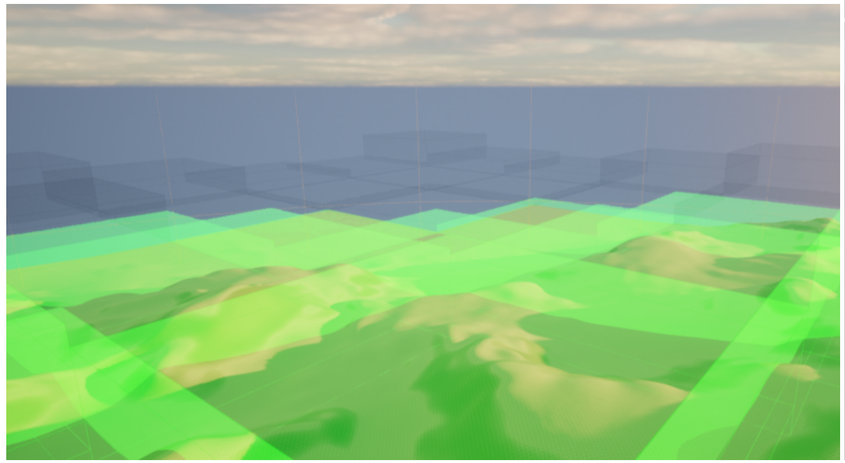
- **APlayerController** registers itself as a streaming source by default using the player camera manager's view target actor's "eyes"
- **UWorldPartitionStreamingSourceComponent** provides its owning actor's location/rotation

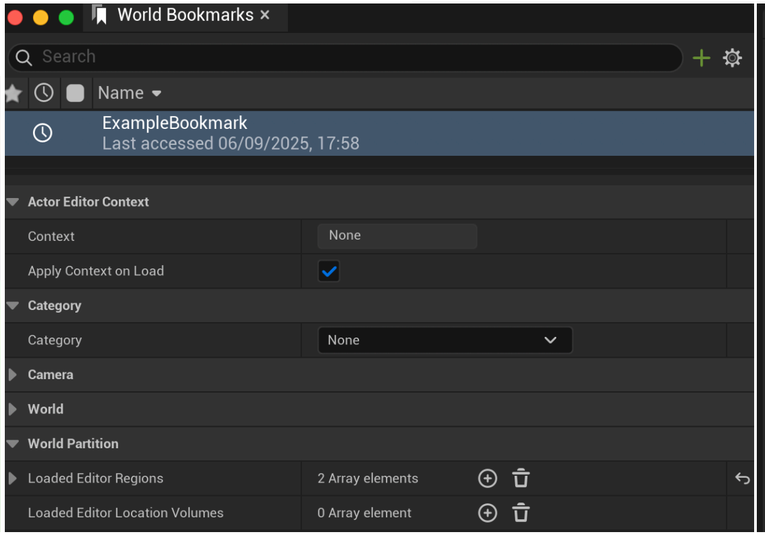


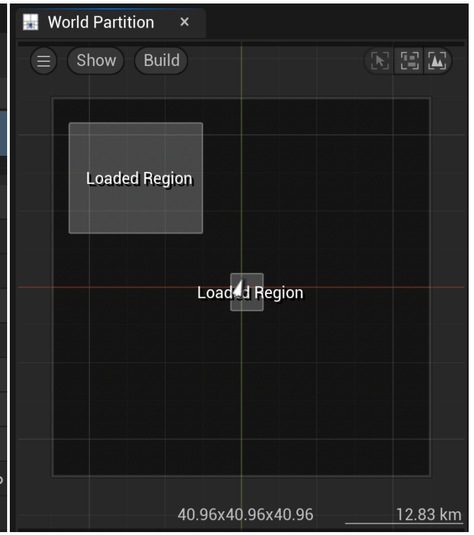
Tips For Working With Grids

- Make use of the inbuilt 2D/3D visualisation for debugging
- Use “block on slow streaming” to pinpoint streaming problems
- Sub-World Partitions can be used for complex world setups
- World Partition Editor & Bookmarks help with editing
- Foliage & landscape auto-partition with streaming proxies
- Actors will not persist data on streaming out completely
- Actors are streamed in and out with their original cell
- Keep actor bounds to a minimum



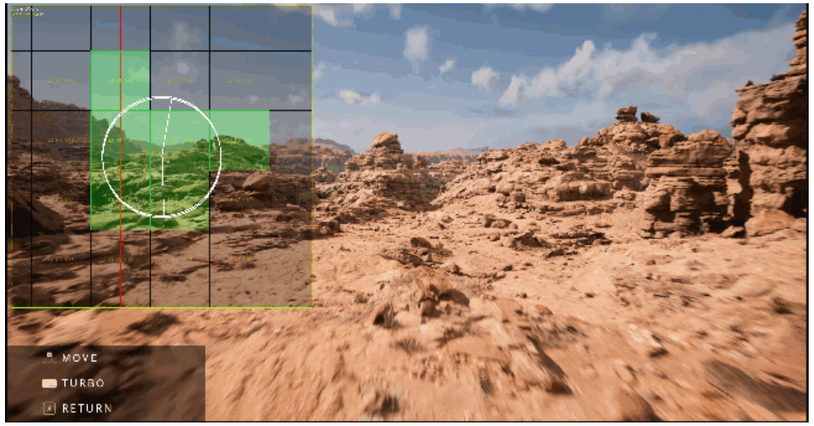


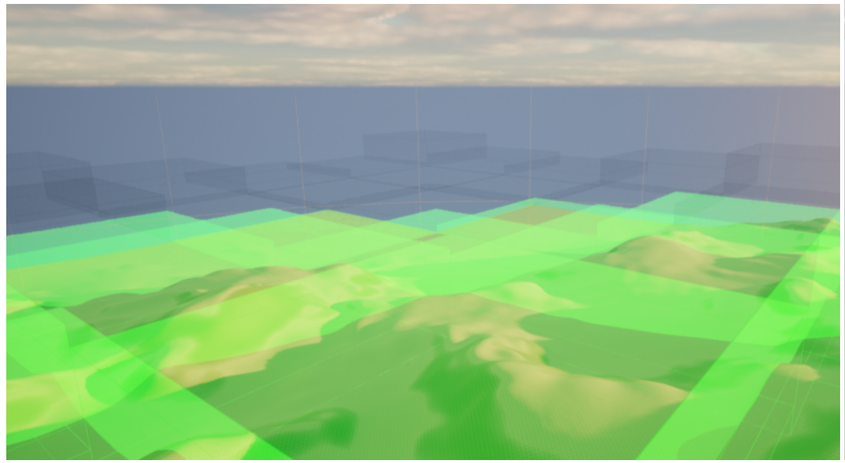


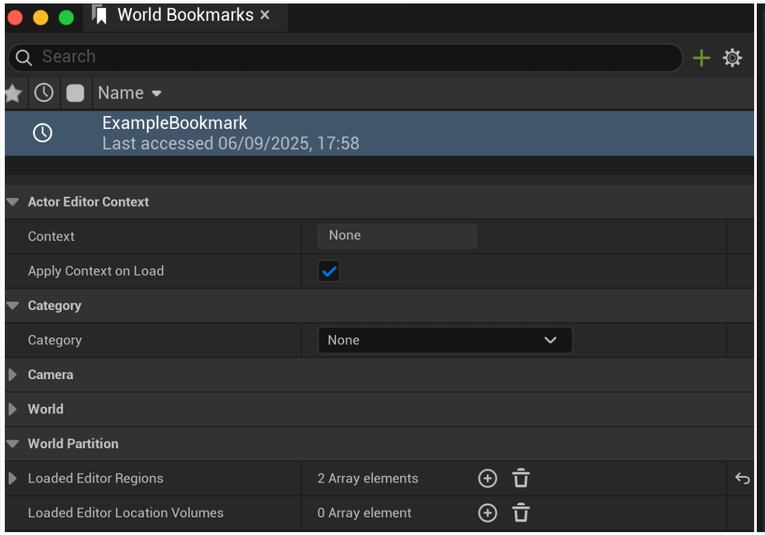


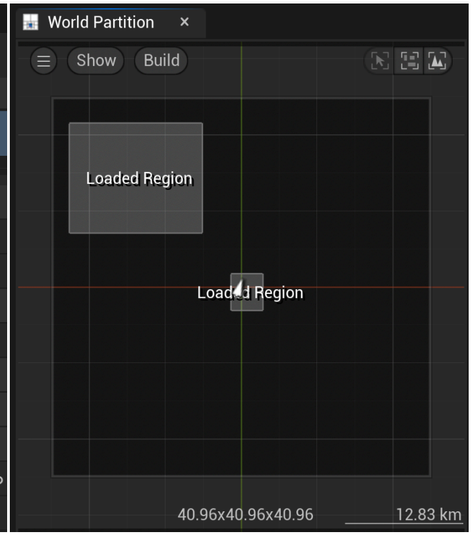
Tips For Working With Grids

- Make use of the inbuilt 2D/3D visualisation for debugging
- Use “block on slow streaming” to pinpoint streaming problems
- Sub-World Partitions can be used for complex world setups
- World Partition Editor & Bookmarks help with editing
- Foliage & landscape auto-partition with streaming proxies
- Actors will not persist data on streaming out completely
- Actors are streamed in and out with their original cell
- Keep actor bounds to a minimum



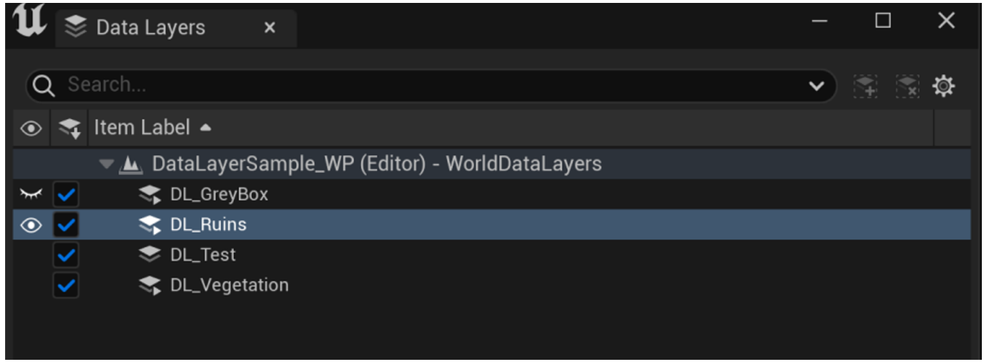


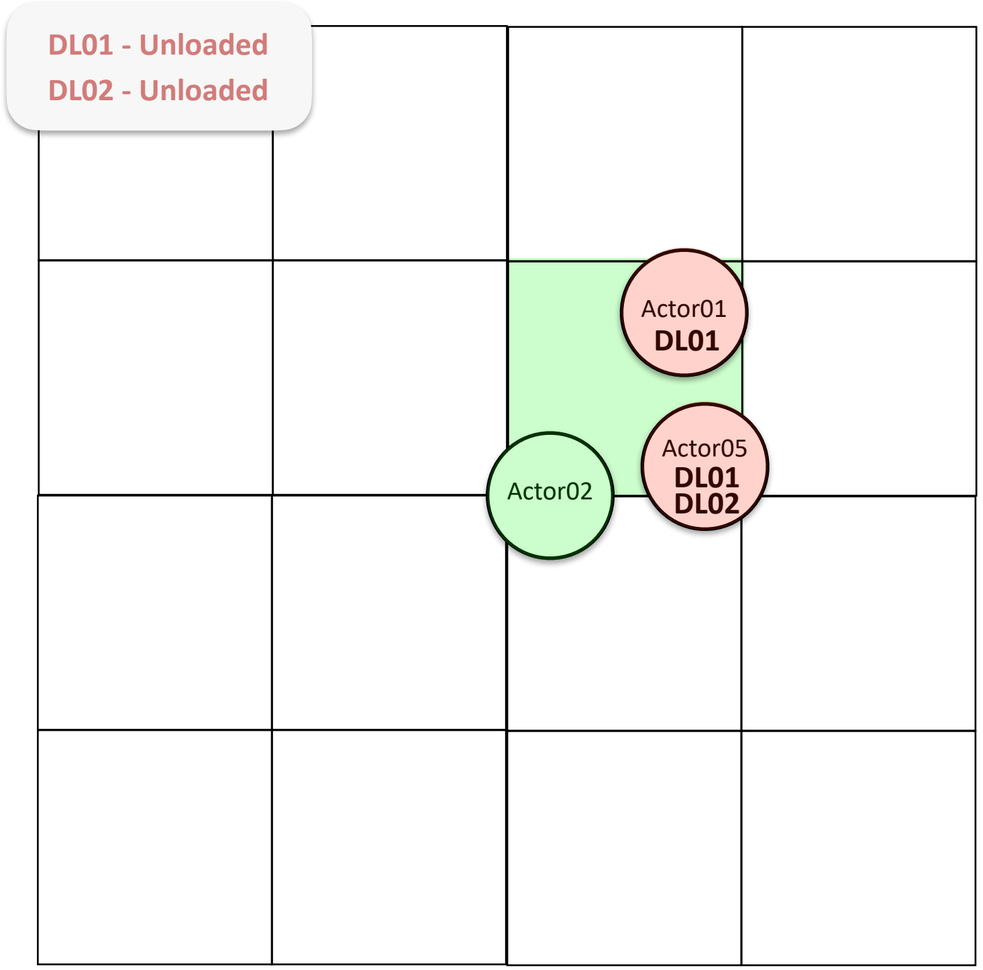




Data Layers

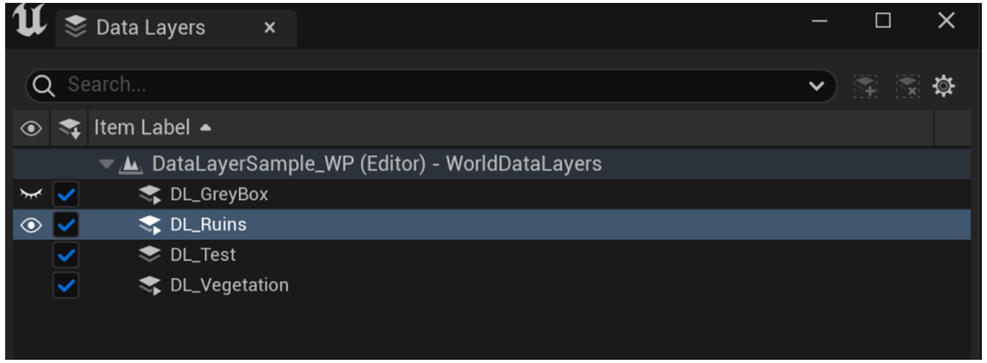
- “Layer” is a misnomer; it’s actually a filter to assign runtime streaming state
- Actors may be assigned to zero or more data layers
- Filtering conditions by default form an OR operation on the highest streaming state
- Actors without are not additionally filtered

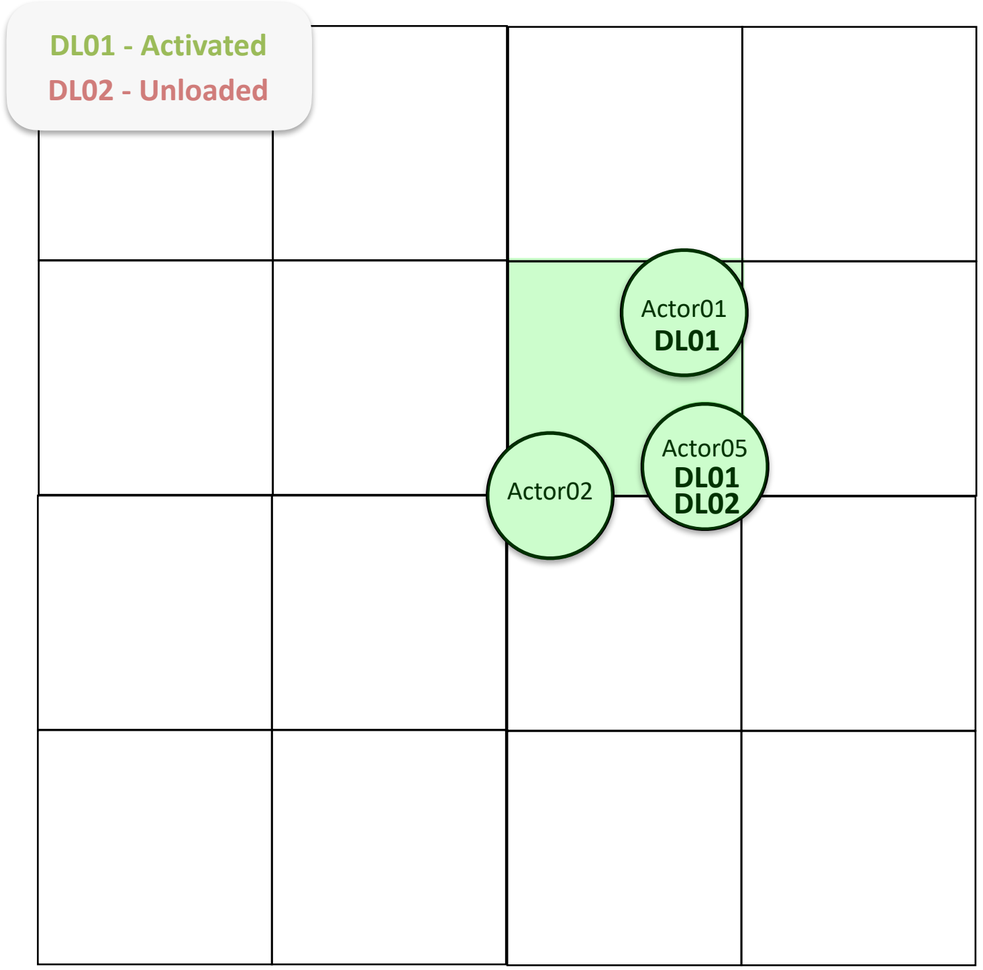




Data Layers

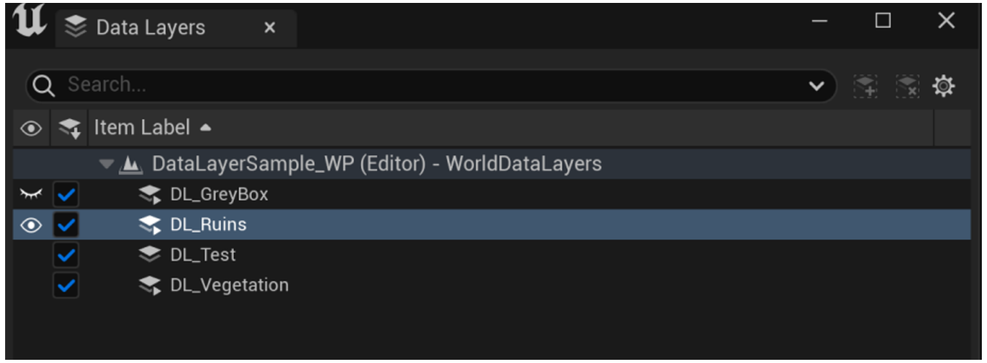
- “Layer” is a misnomer; it’s actually a filter to assign runtime streaming state
- Actors may be assigned to zero or more data layers
- Filtering conditions by default form an OR operation on the highest streaming state
- Actors without are not additionally filtered

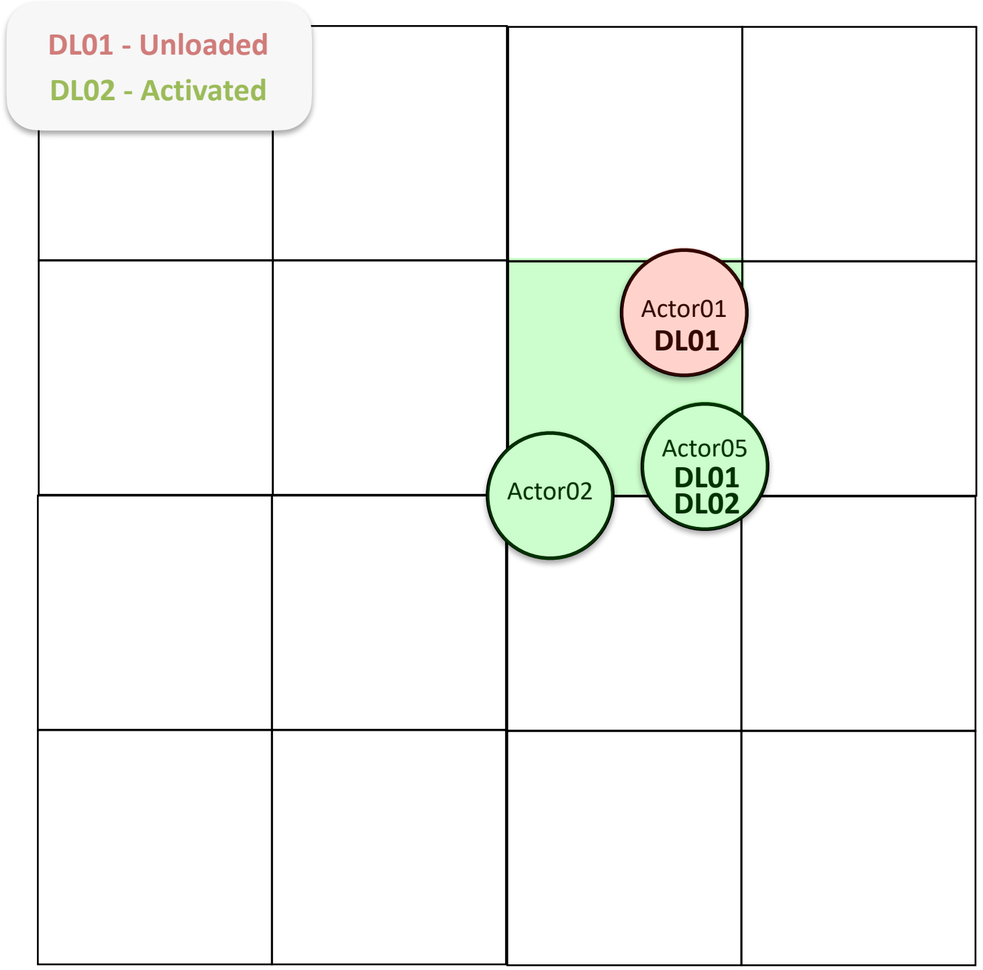




Data Layers

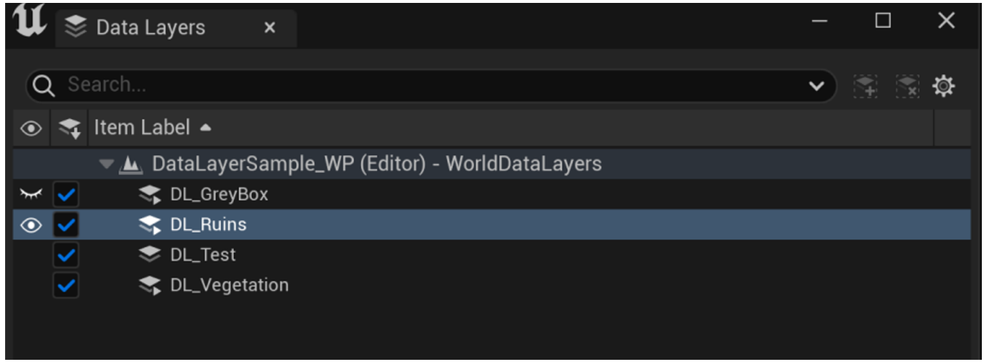
- “Layer” is a misnomer; it’s actually a filter to assign runtime streaming state
- Actors may be assigned to zero or more data layers
- Filtering conditions by default form an OR operation on the highest streaming state
- Actors without are not additionally filtered

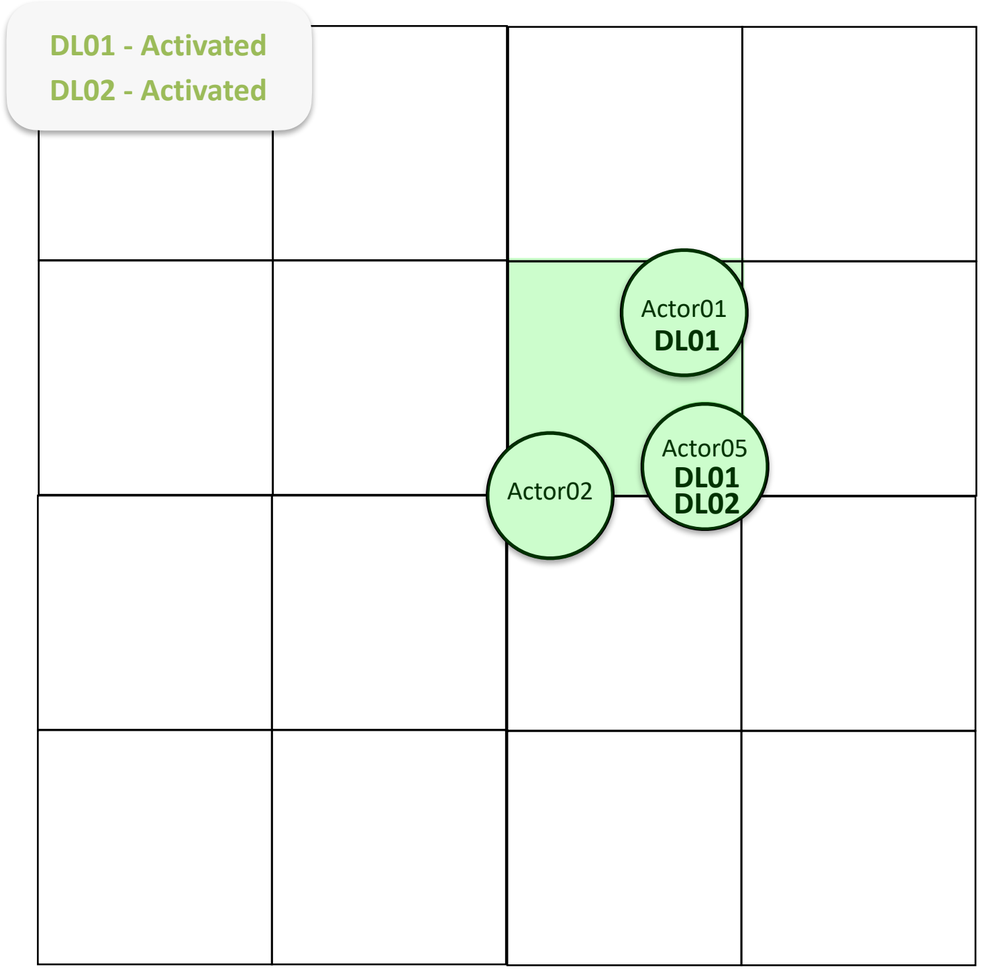




Data Layers

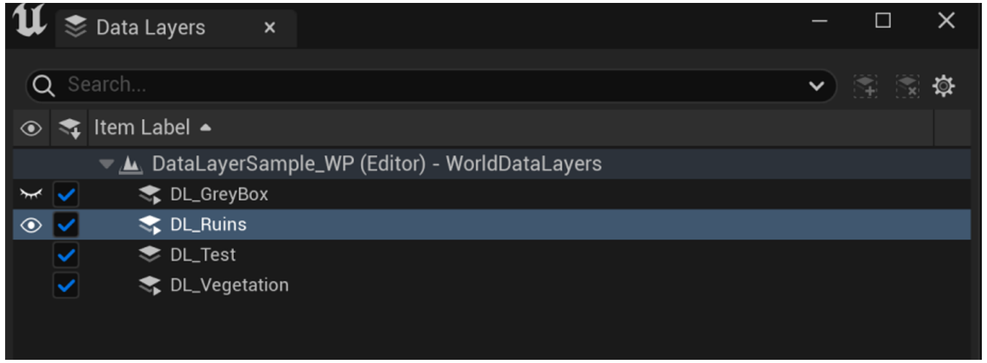
- “Layer” is a misnomer; it’s actually a filter to assign runtime streaming state
- Actors may be assigned to zero or more data layers
- Filtering conditions by default form an OR operation on the highest streaming state
- Actors without are not additionally filtered

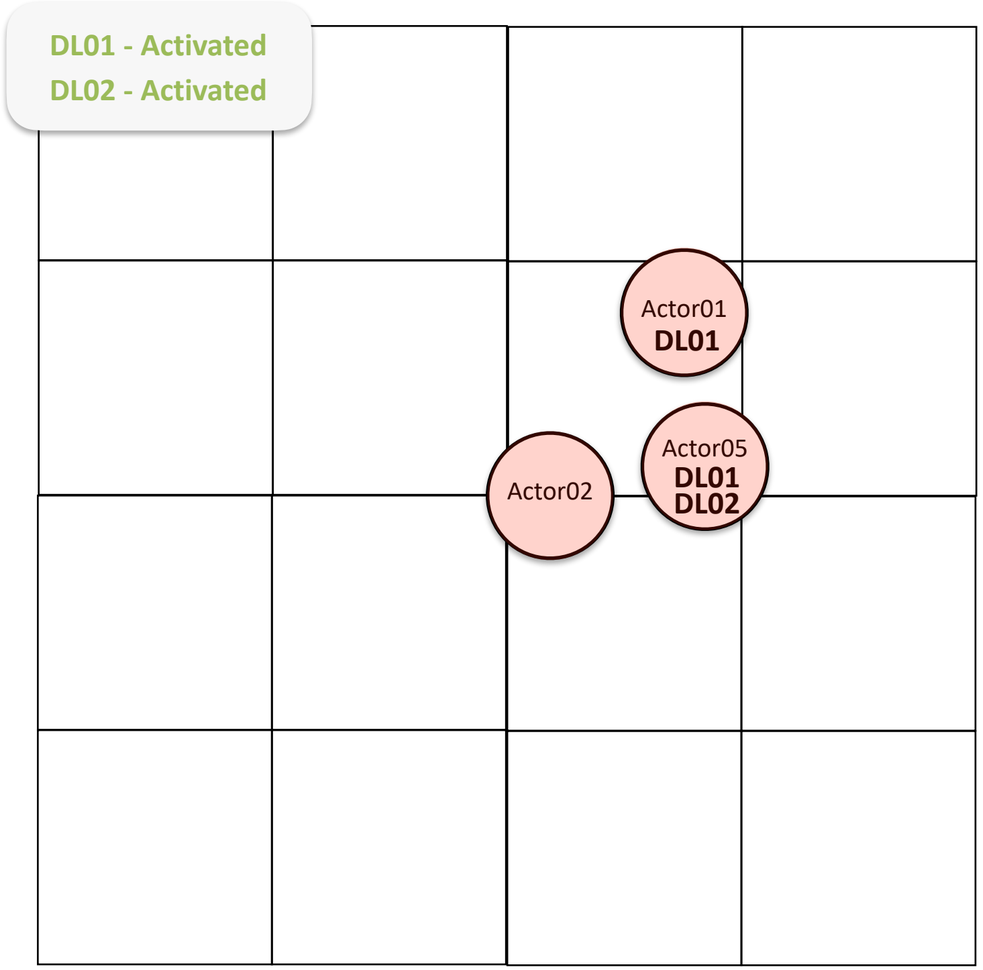




Data Layers

- "Layer" is a misnomer; it's actually a filter to assign runtime streaming state
- Actors may be assigned to zero or more data layers
- Filtering conditions by default form an OR operation on the highest streaming state
- Actors without are not additionally filtered



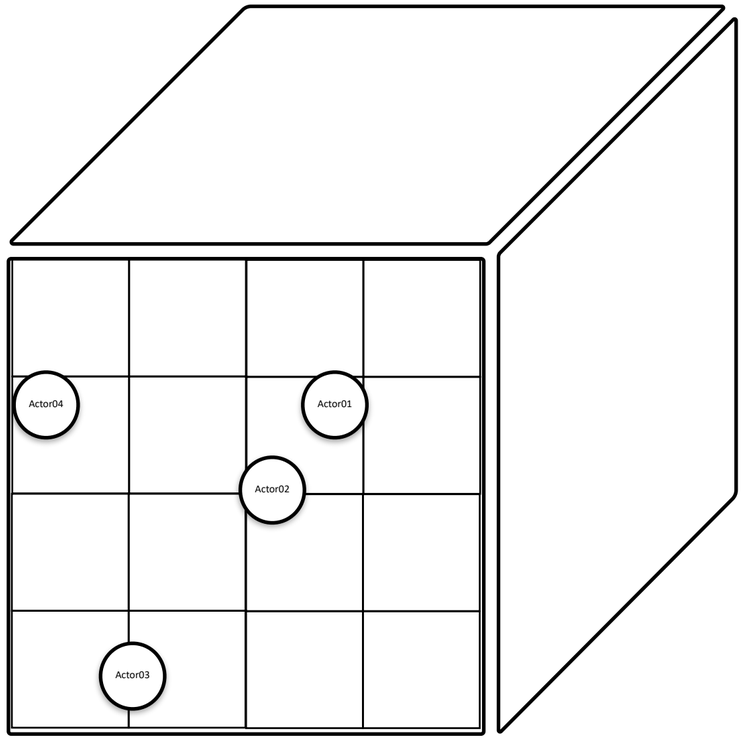


Data Layers & Streaming Generation

- Data Layers further expand the combinatorics of streaming generation rules

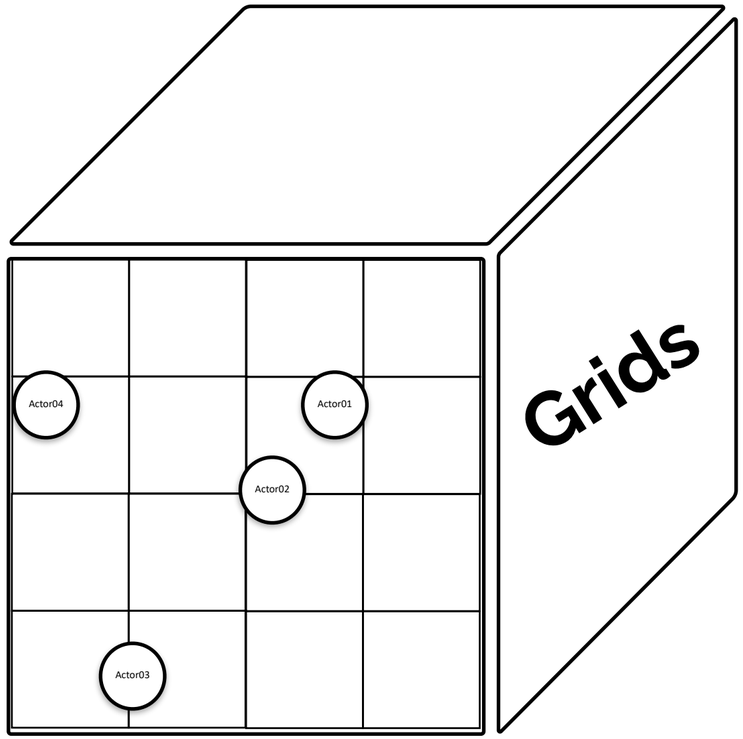
Data Layers & Streaming Generation

- Data Layers further expand the combinatorics of streaming generation rules



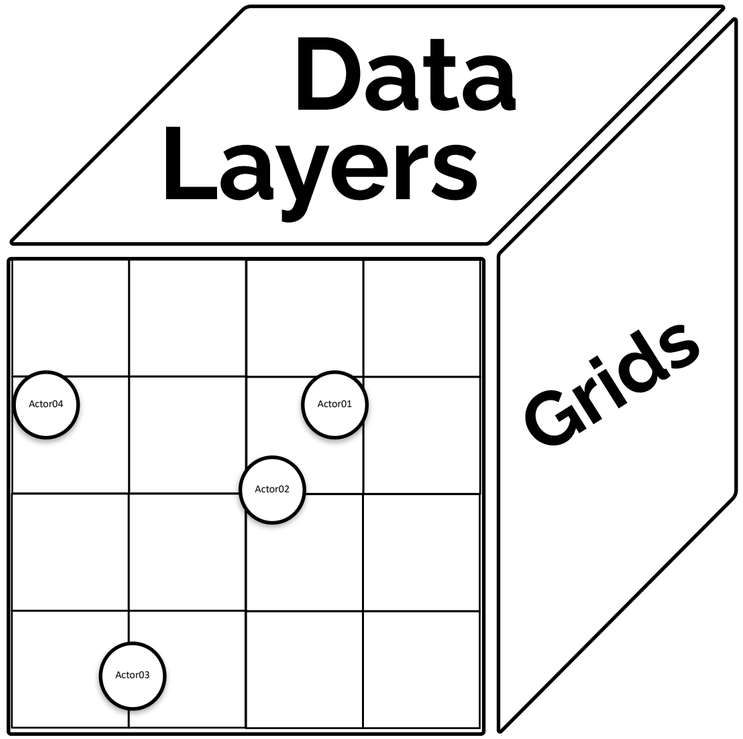
Data Layers & Streaming Generation

- Data Layers further expand the combinatorics of streaming generation rules



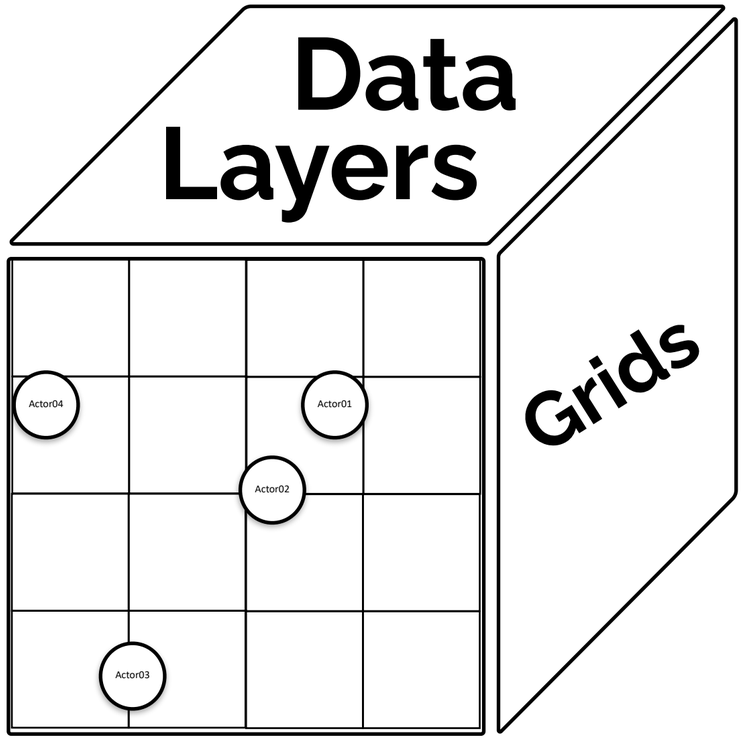
Data Layers & Streaming Generation

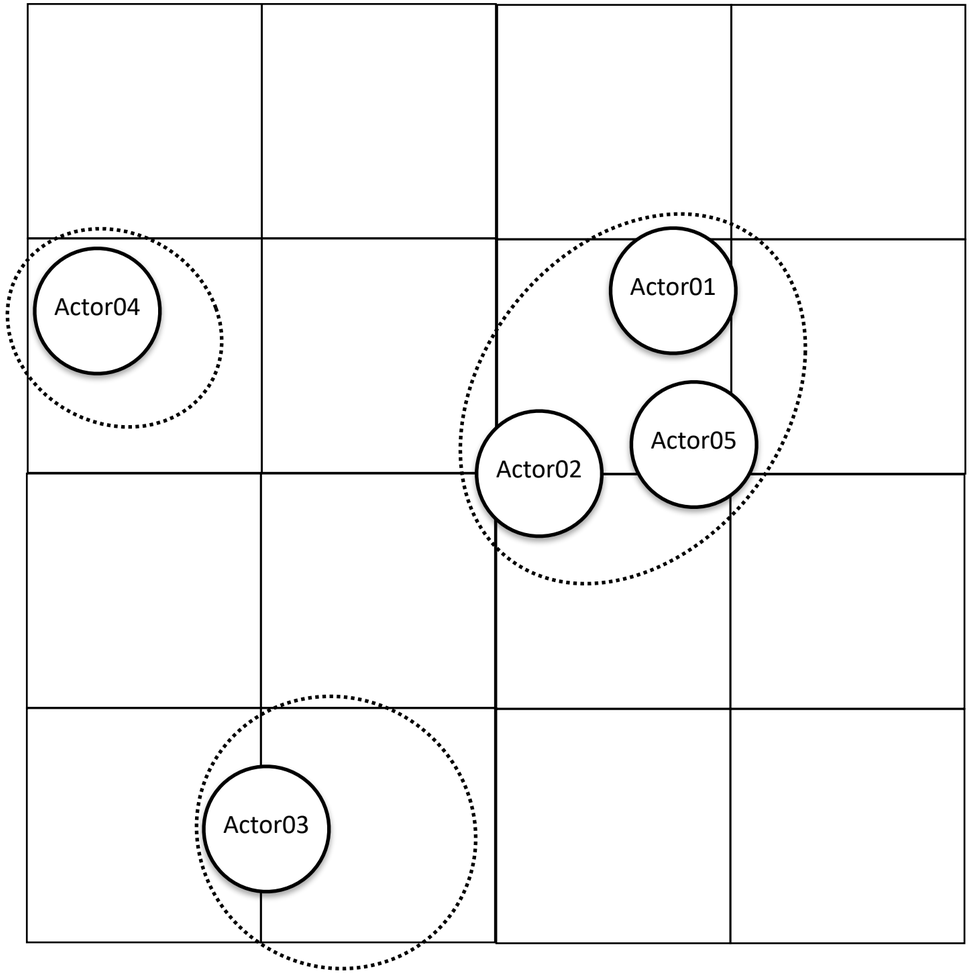
- Data Layers further expand the combinatorics of streaming generation rules



Data Layers & Streaming Generation

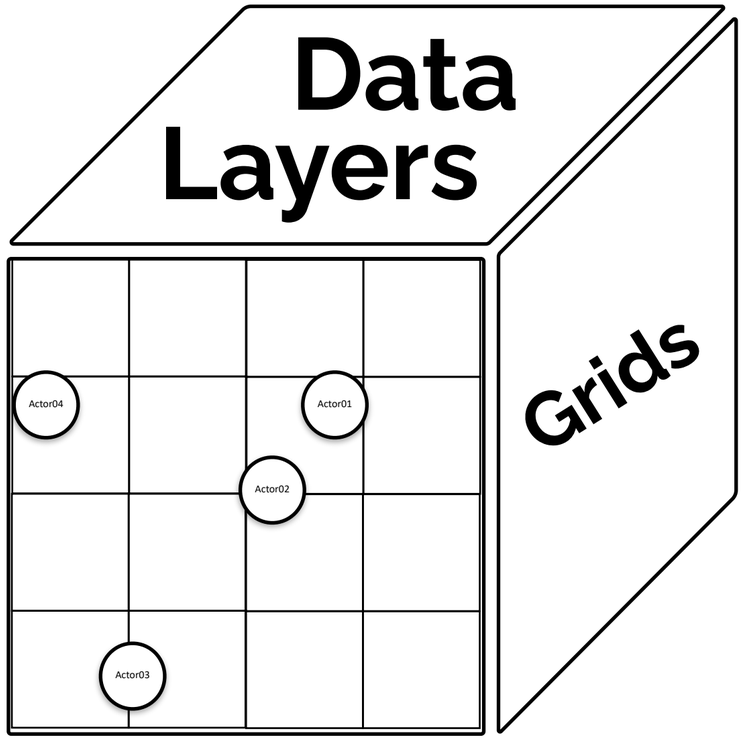
- Data Layers further expand the combinatorics of streaming generation rules

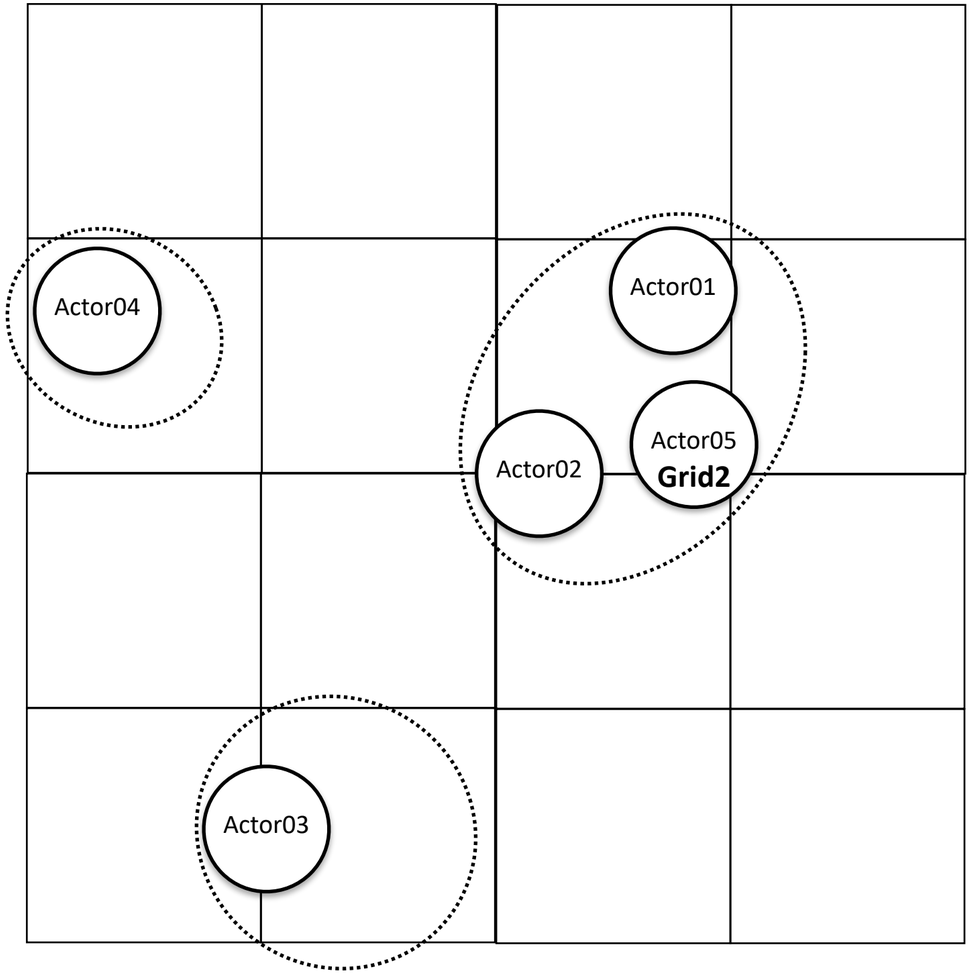




Data Layers & Streaming Generation

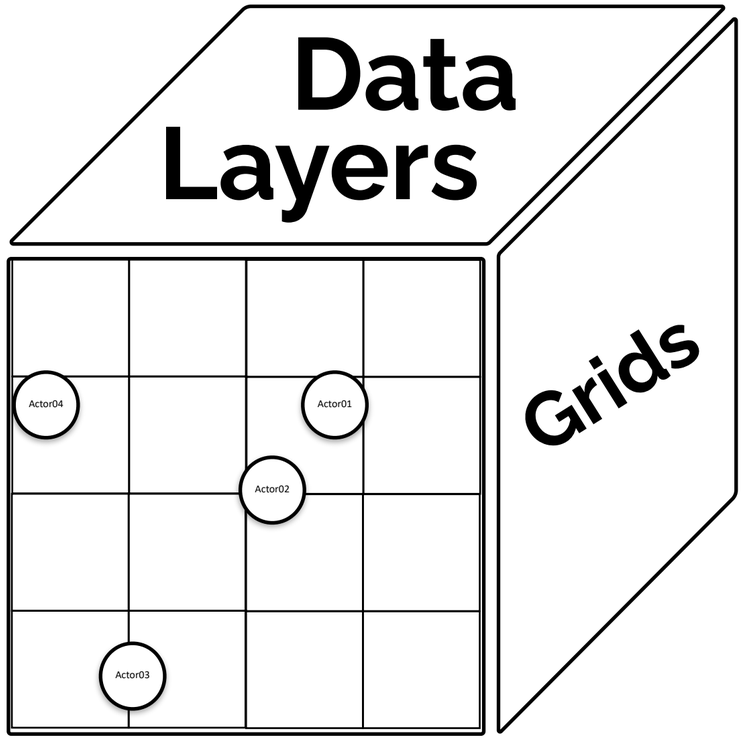
- Data Layers further expand the combinatorics of streaming generation rules

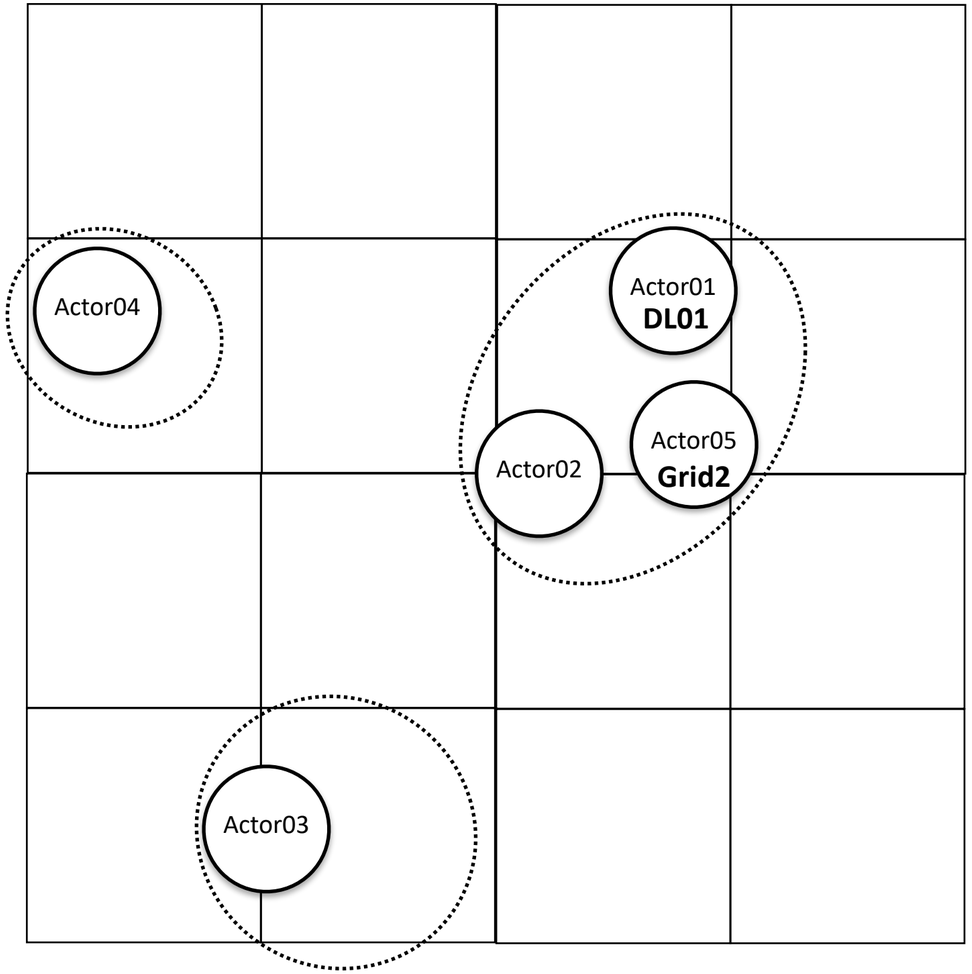




Data Layers & Streaming Generation

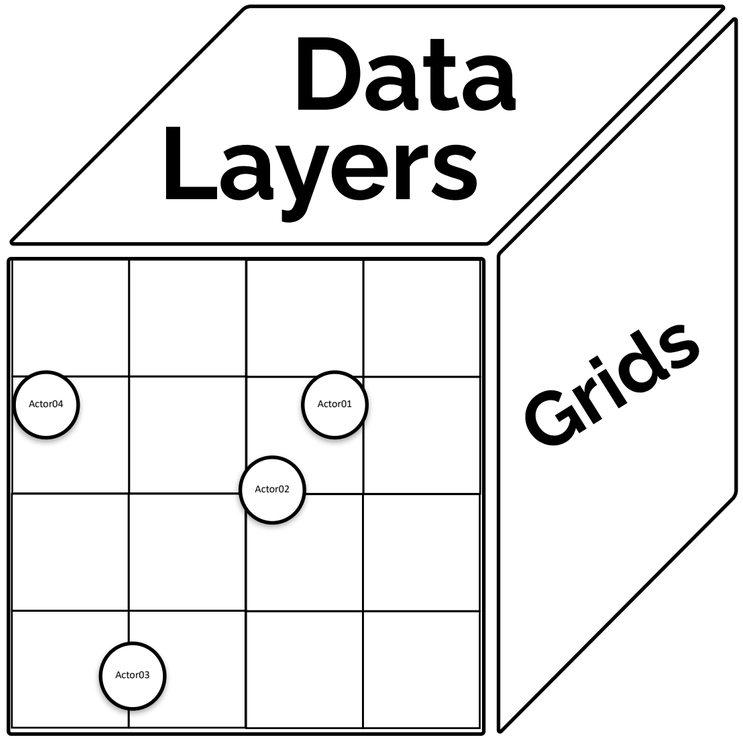
- Data Layers further expand the combinatorics of streaming generation rules

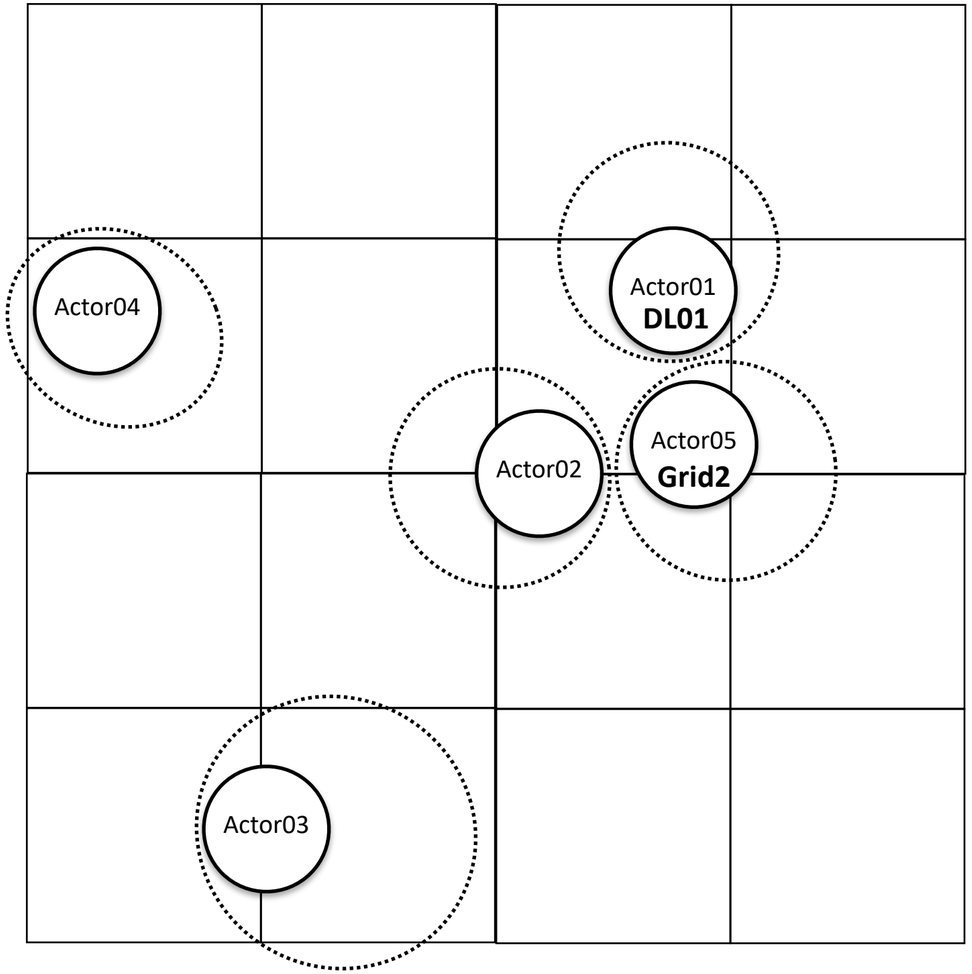




Data Layers & Streaming Generation

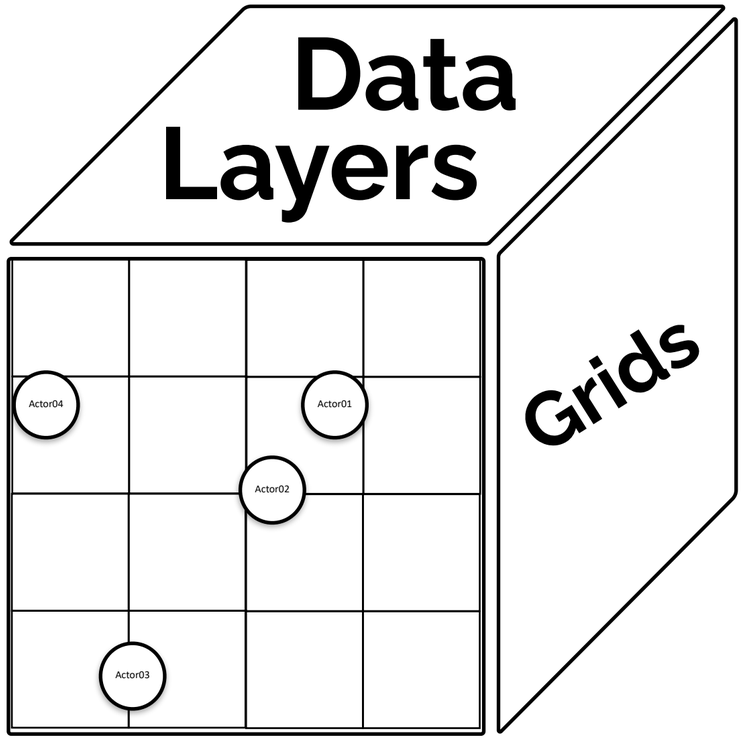
- Data Layers further expand the combinatorics of streaming generation rules





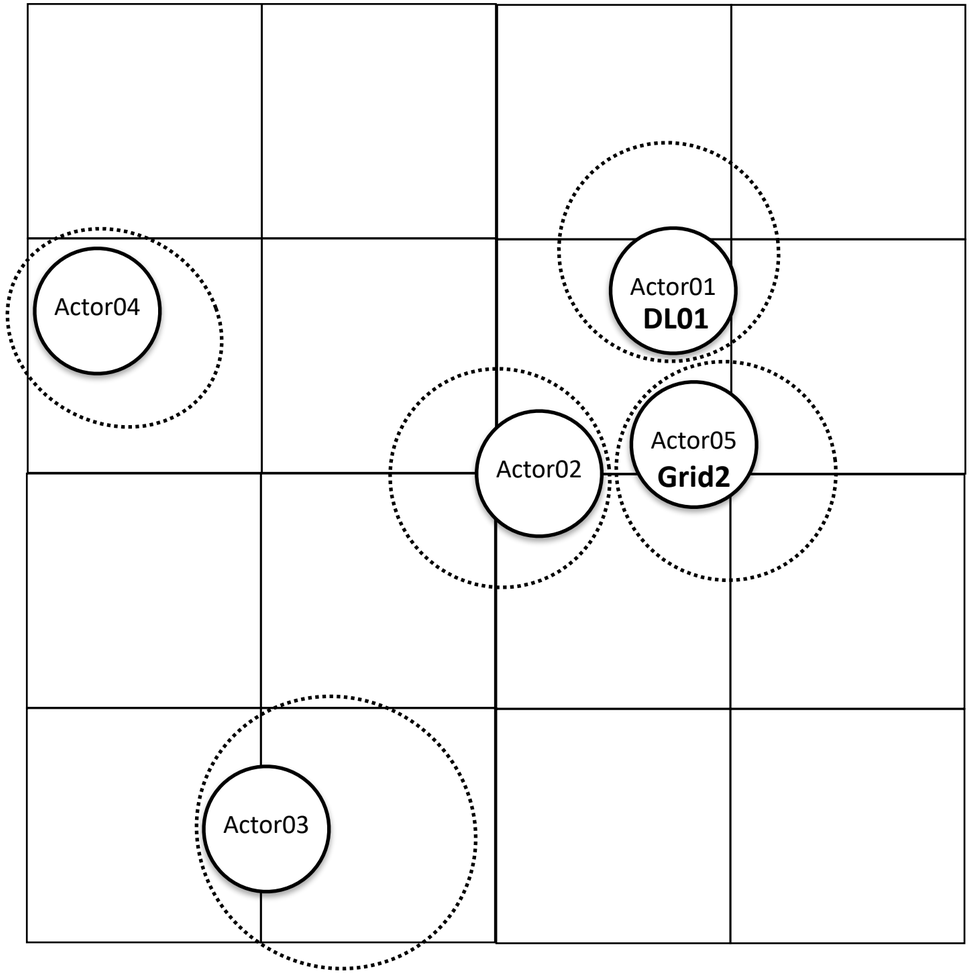
Data Layers & Streaming Generation

- Data Layers further expand the combinatorics of streaming generation rules



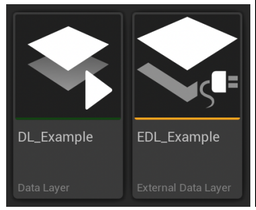
`streaming levels = cells * grids * data layers`

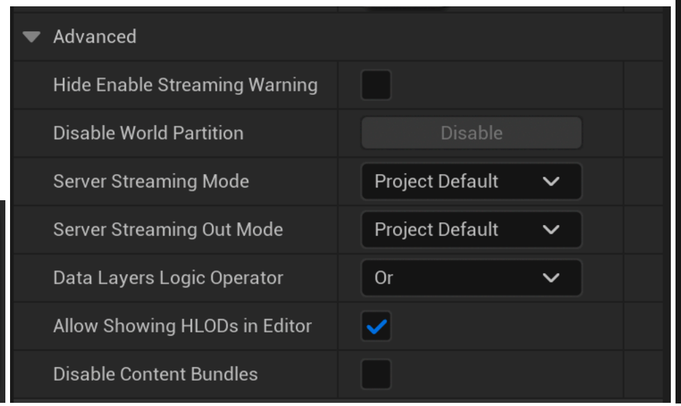
(for each unique combination used by actors)

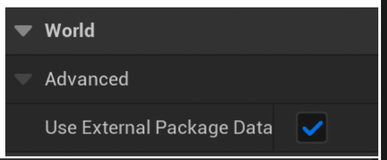


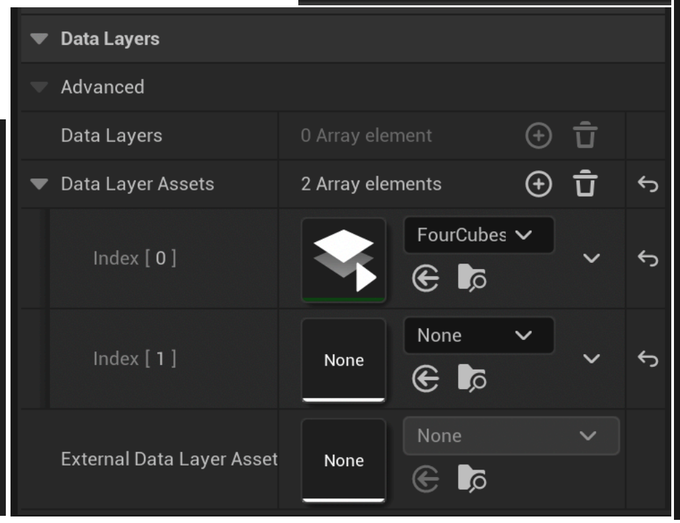
Configuring Data Layers

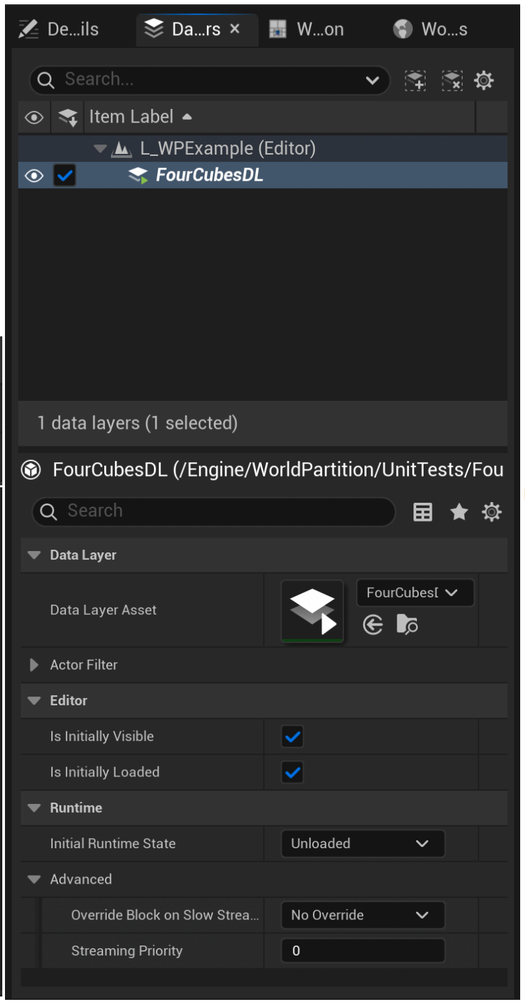
- World Partition levels automatically spawn **AWorldDataLayers** actor
  - This owns all data layer configurations for the persistent level
- Data Layers are actually configured via the Data Layer Outliner
  - Tip: enable **bUseExternalPackageData** to reduce checkout contention
- Data Layer Assets can be set to runtime or editor
- Data Layer Instances be placed in a hierarchy, where relationships reduce to an effective “min” streaming state
- Can be reconfigured from OR to AND
- External layers for game feature plugins







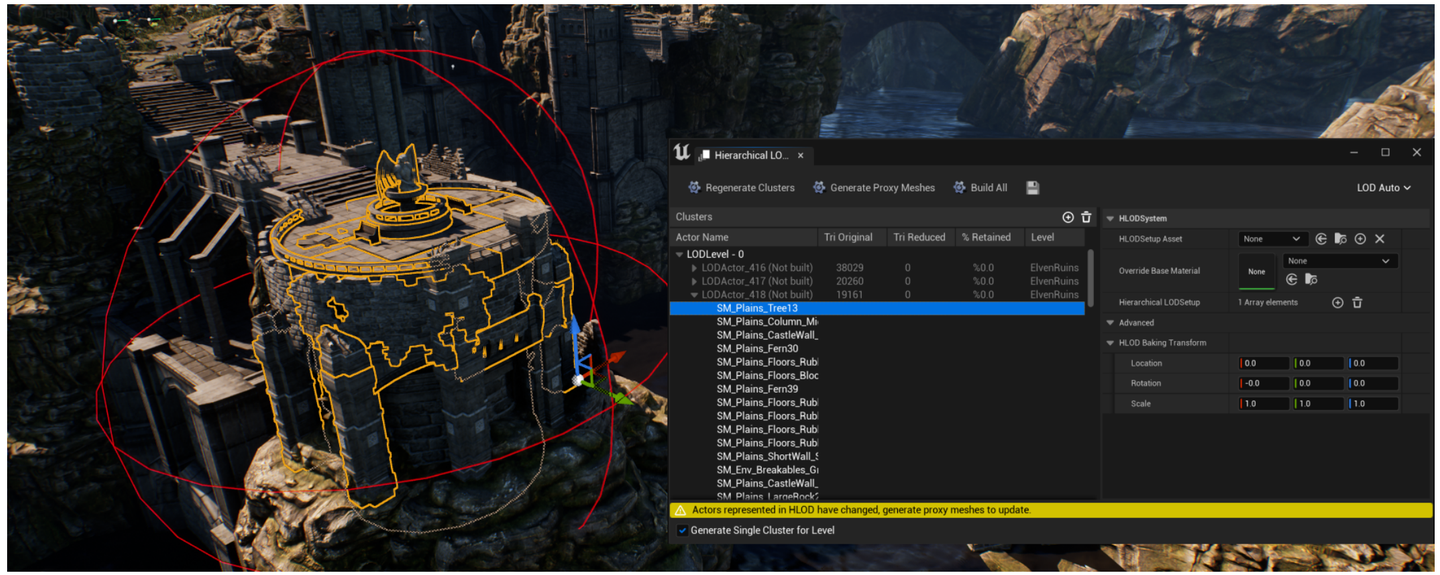




Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

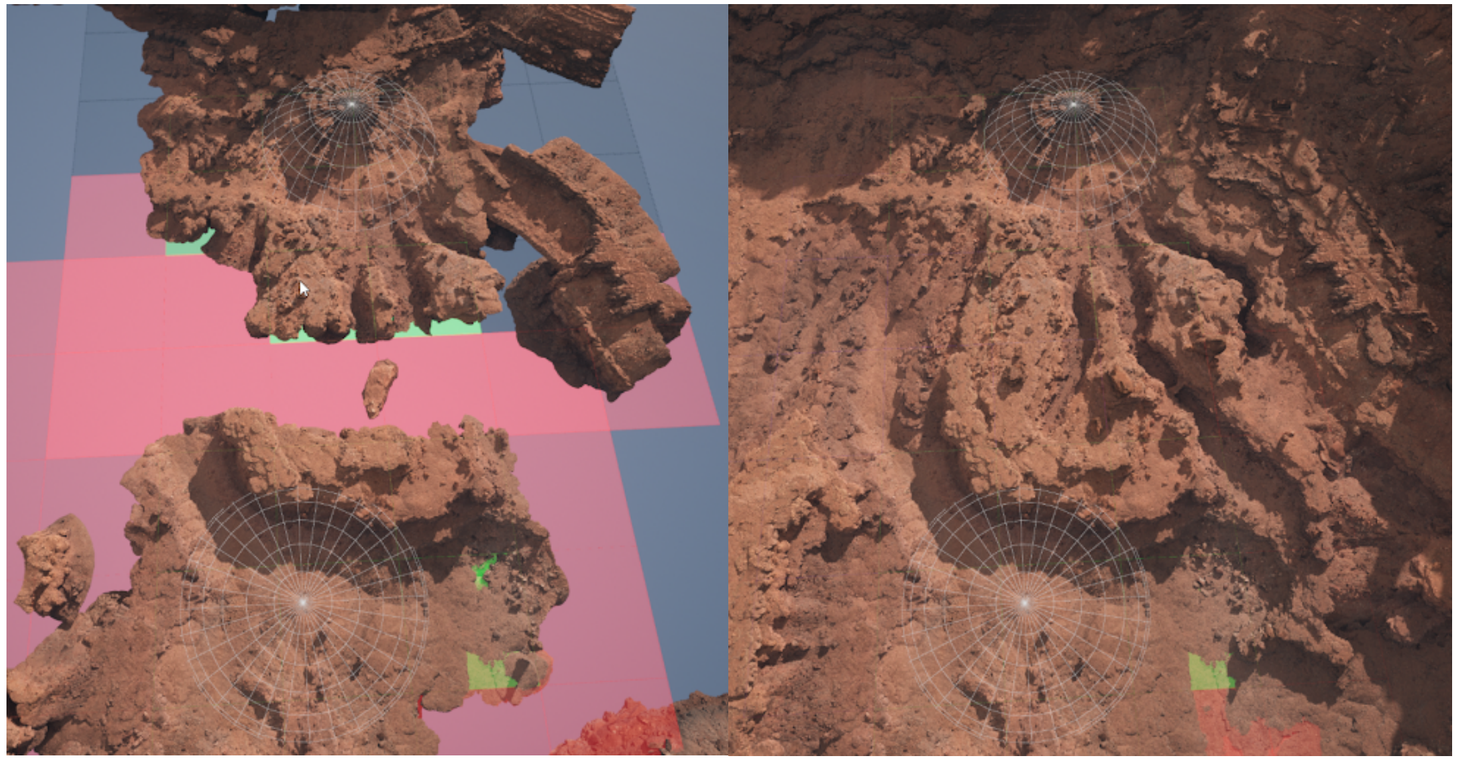
- Supports instanced, merged, simplified, and approximated static mesh transformations



Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

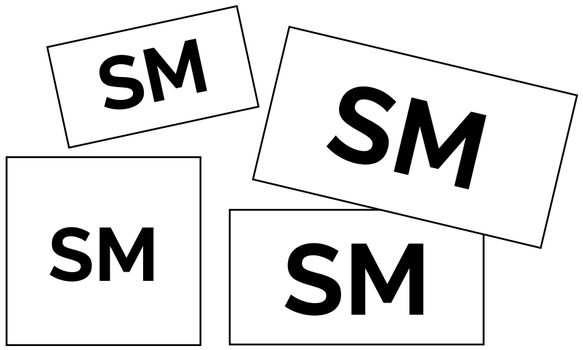
- Supports instanced, merged, simplified, and approximated static mesh transformations

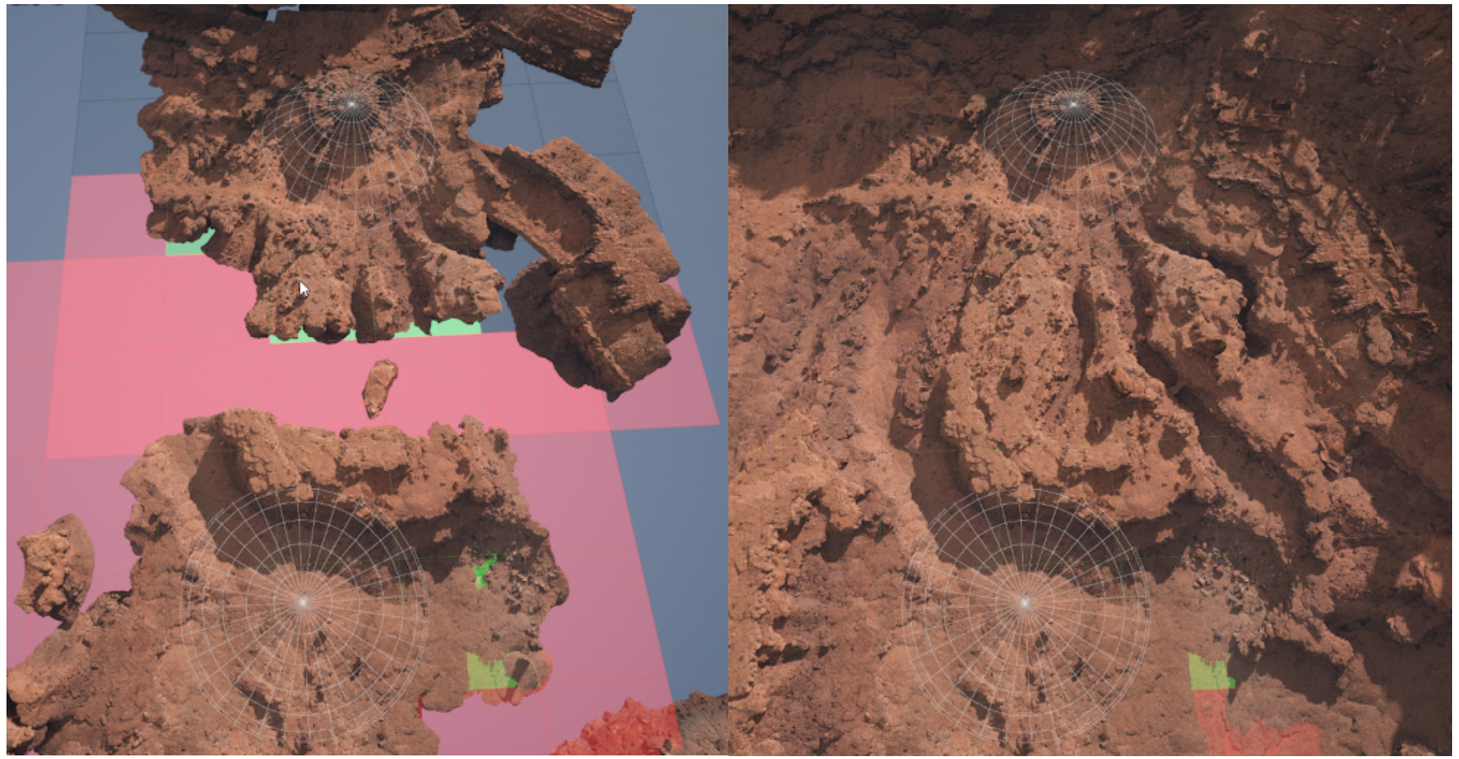


Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

- Supports instanced, merged, simplified, and approximated static mesh transformations

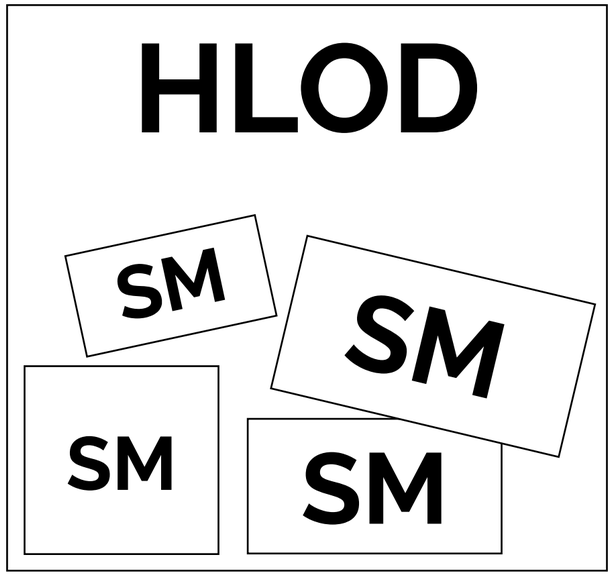


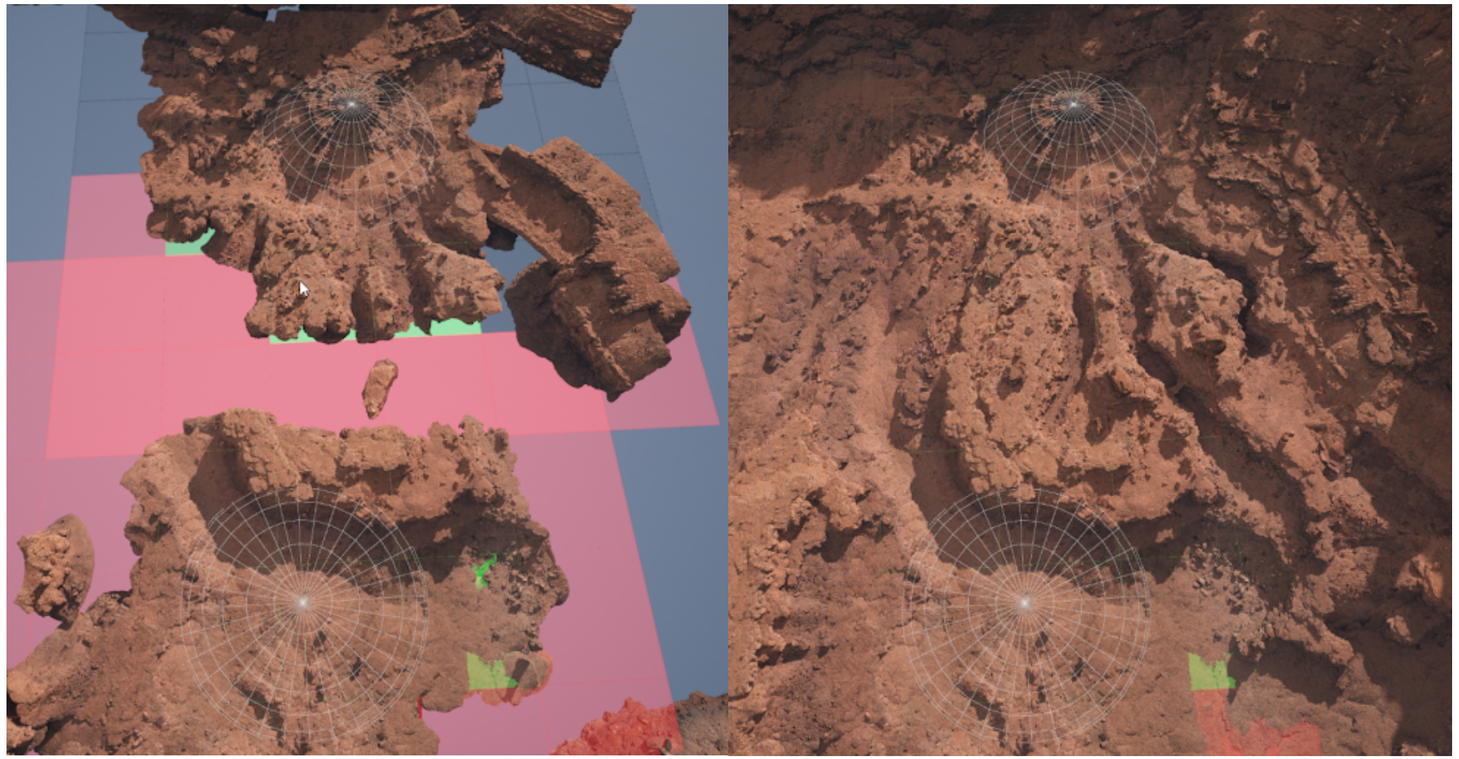


Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

- Supports instanced, merged, simplified, and approximated static mesh transformations

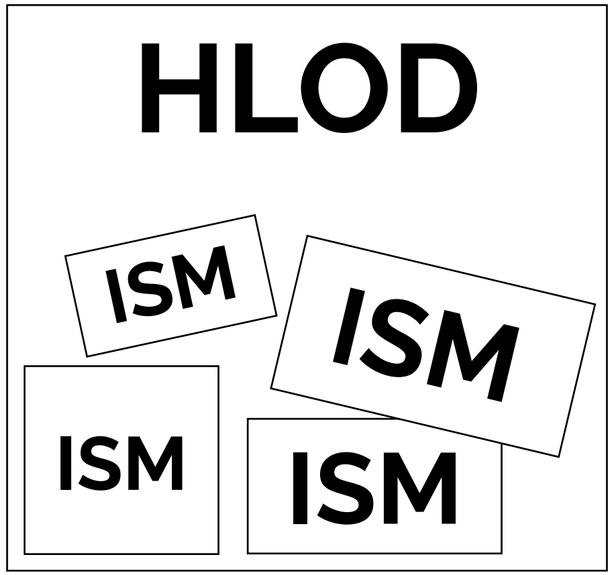


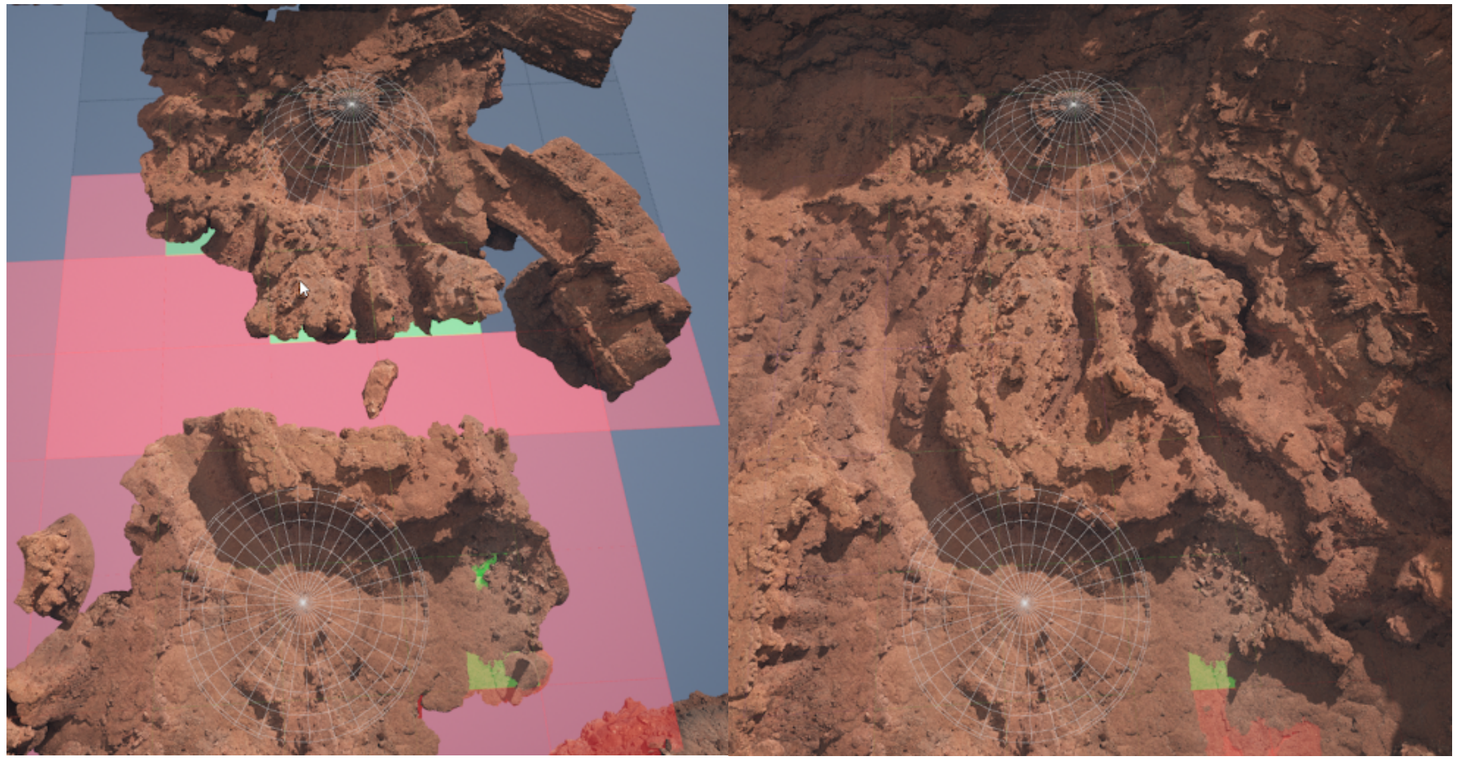


Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

- Supports instanced, merged, simplified, and approximated static mesh transformations

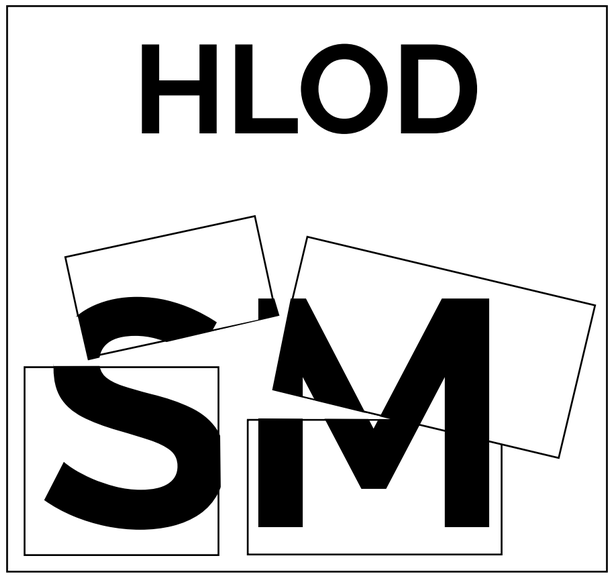


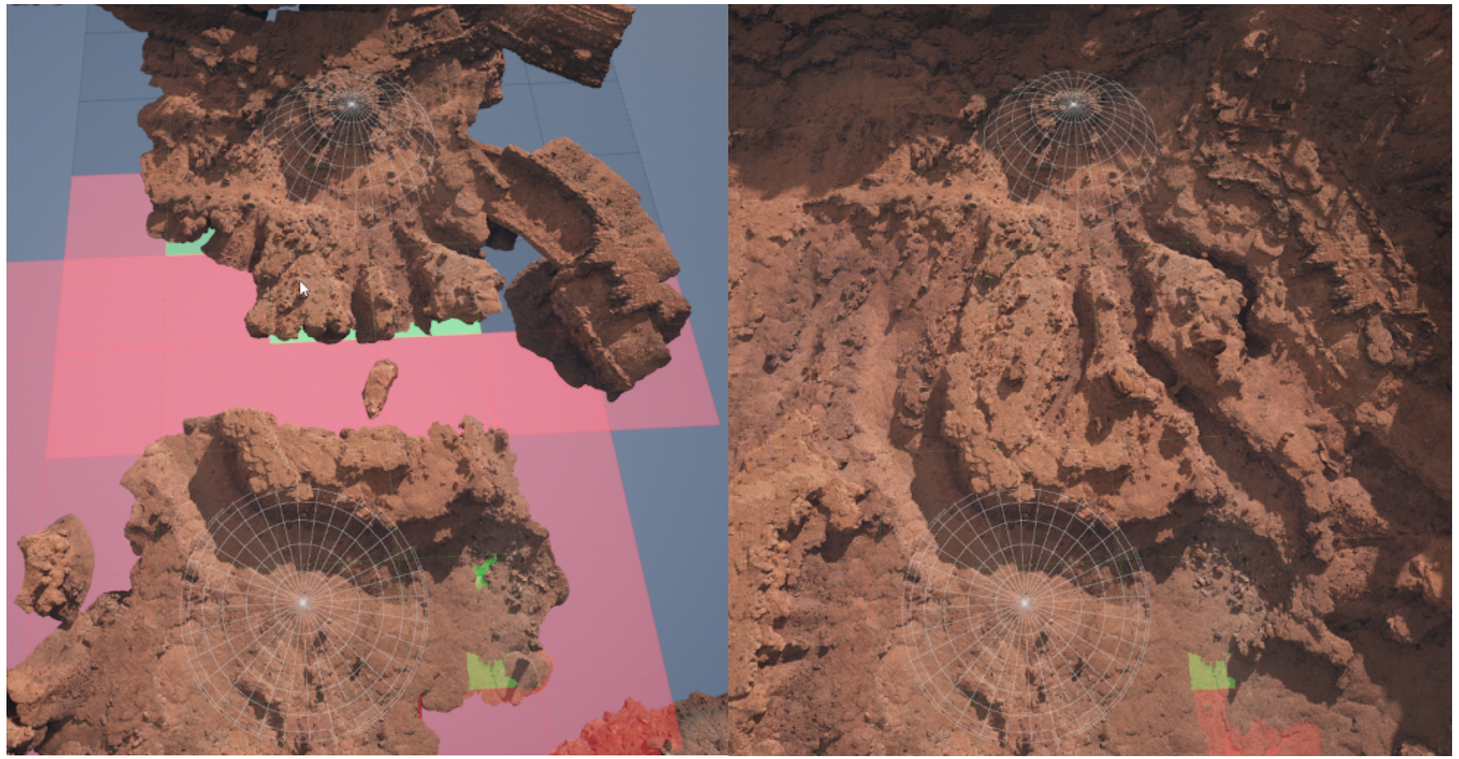


Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

- Supports instanced, merged, simplified, and approximated static mesh transformations

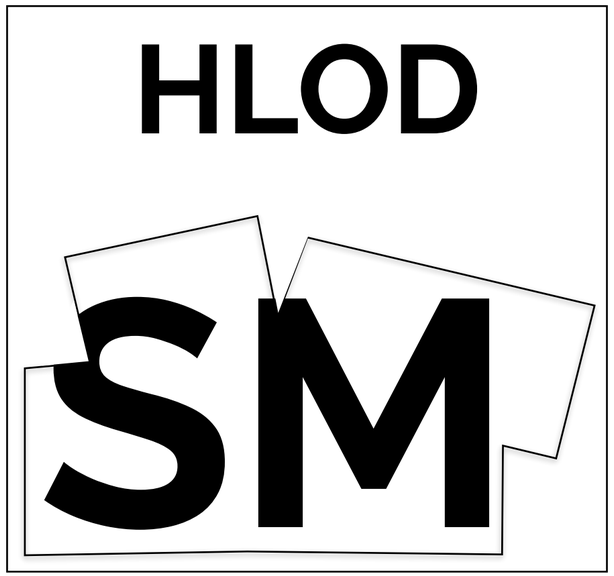


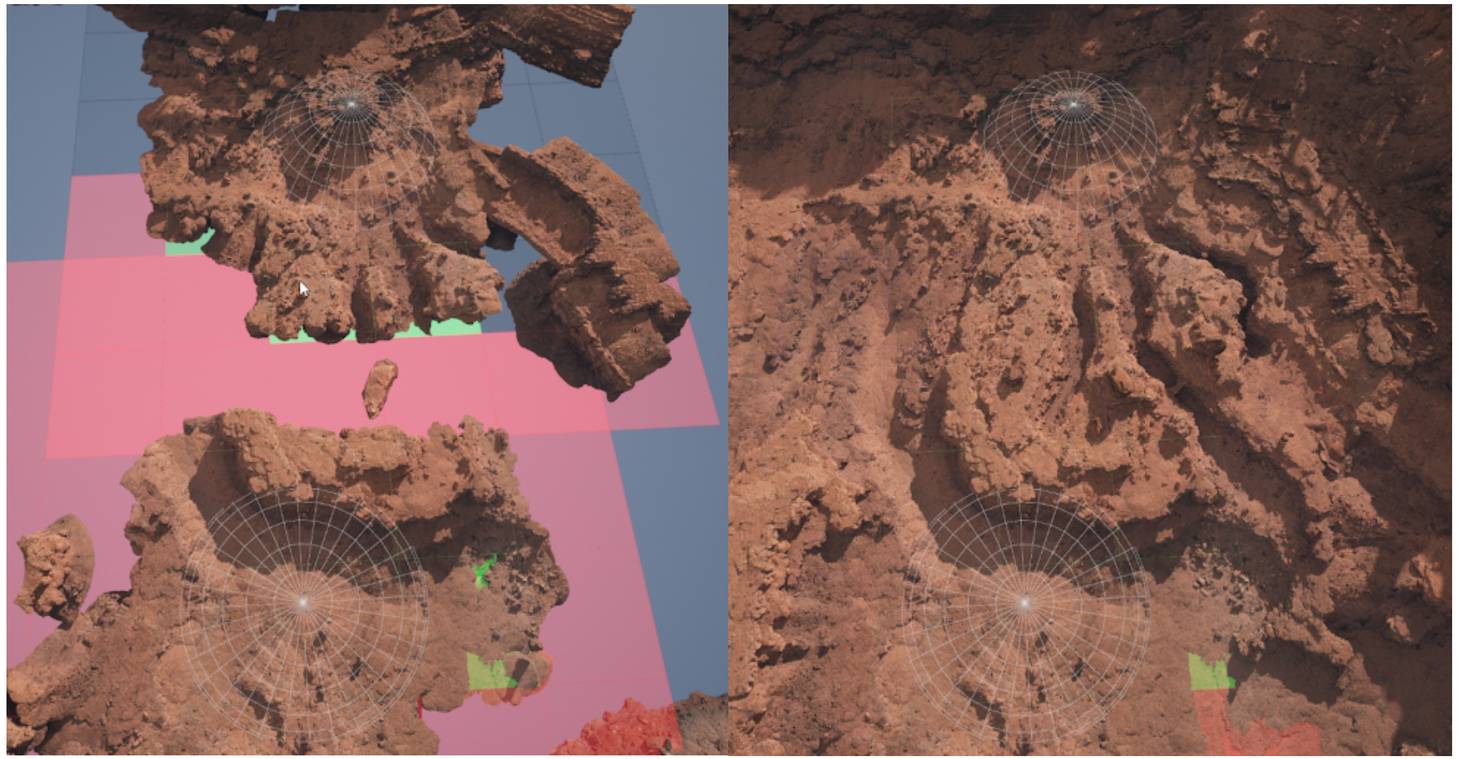


Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

- Supports instanced, merged, simplified, and approximated static mesh transformations

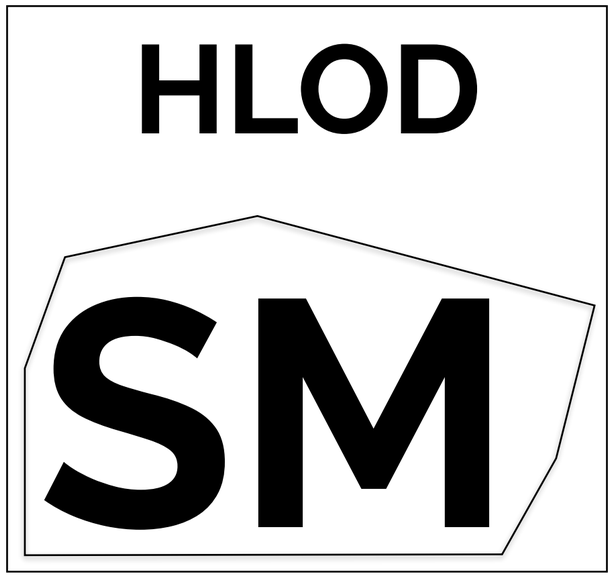


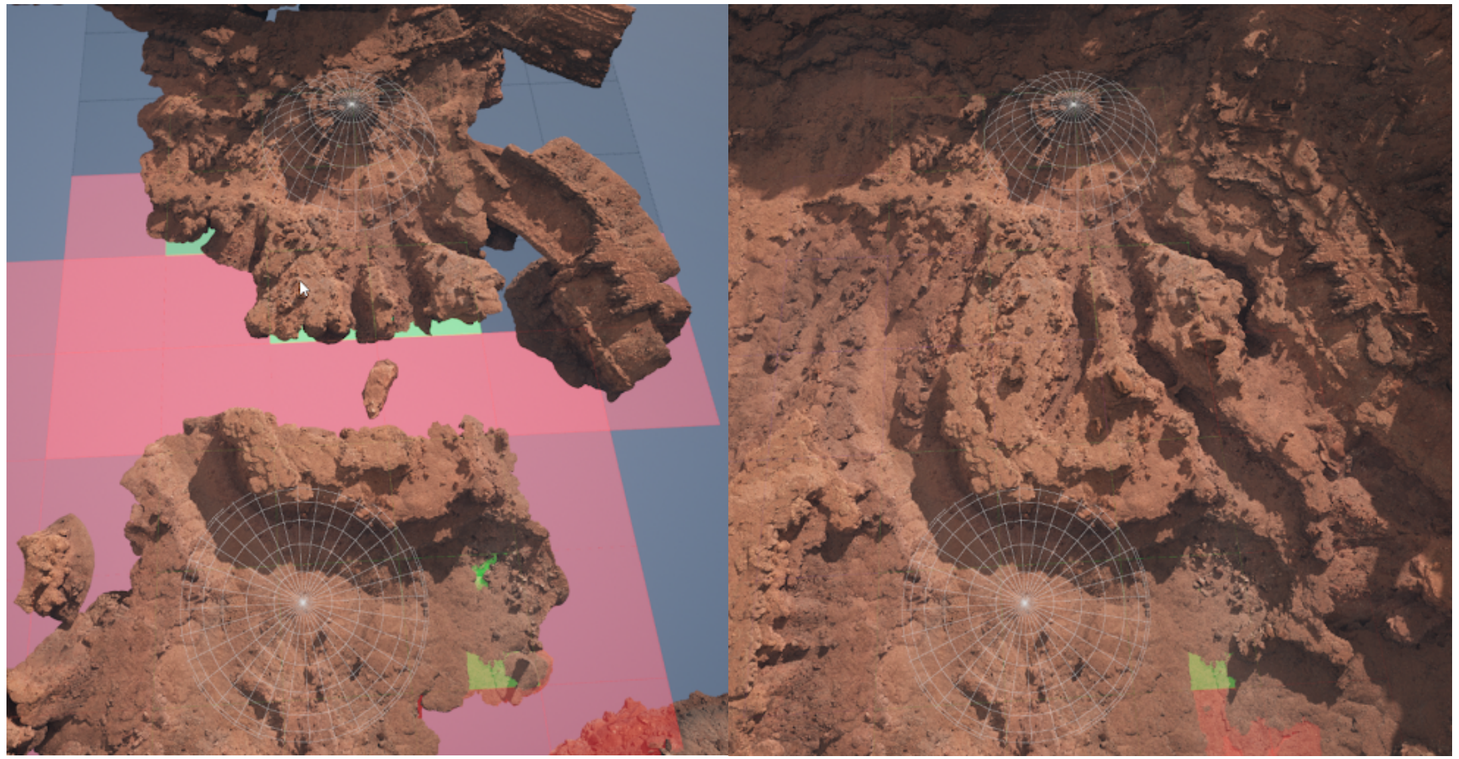


Hierarchical Level of Detail (HLODs)

- Tool to generate “proxy meshes” that can be swapped out at different LOD levels
- Can load in HLODs as distant grid cells are streamed out, providing lower cost distant geometry

- Supports instanced, merged, simplified, and approximated static mesh transformations





Level Instances

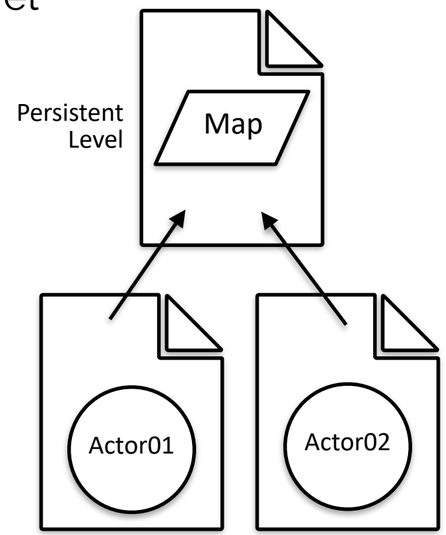
- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically

Level Instances

- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically

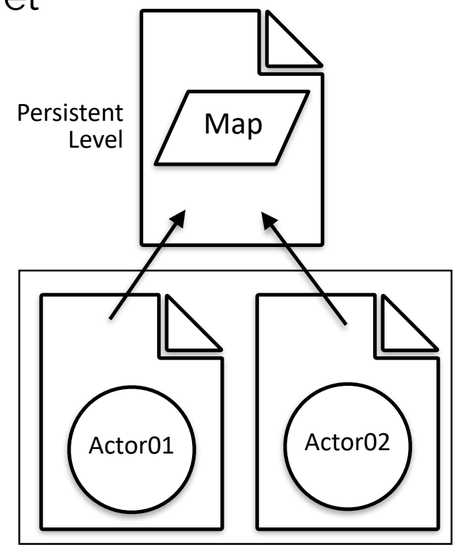
Level Instances

- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically



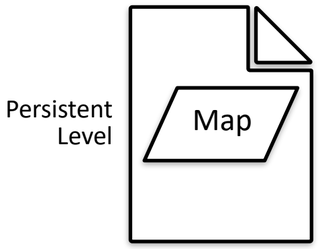
Level Instances

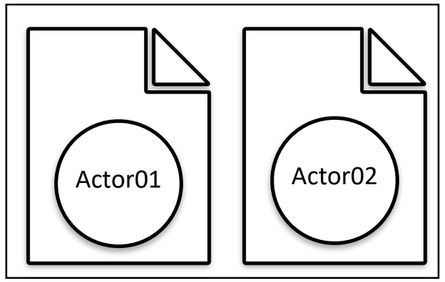
- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically



Level Instances

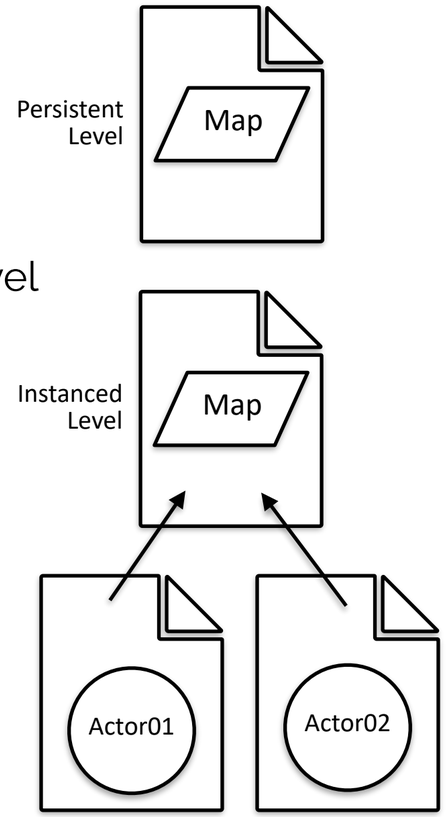
- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically





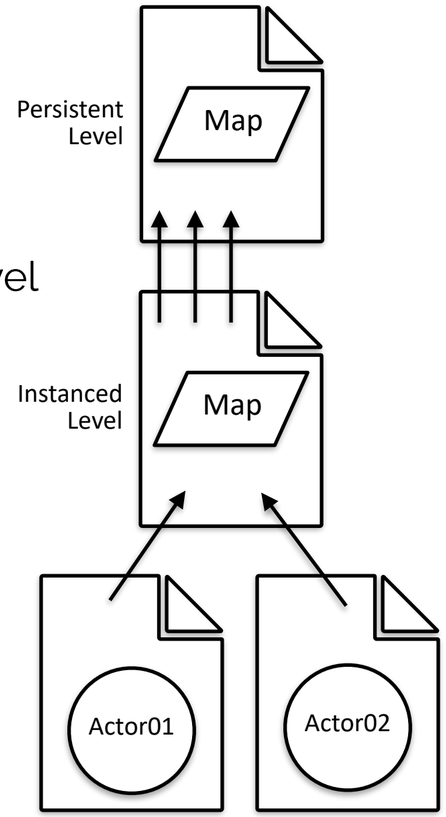
Level Instances

- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically



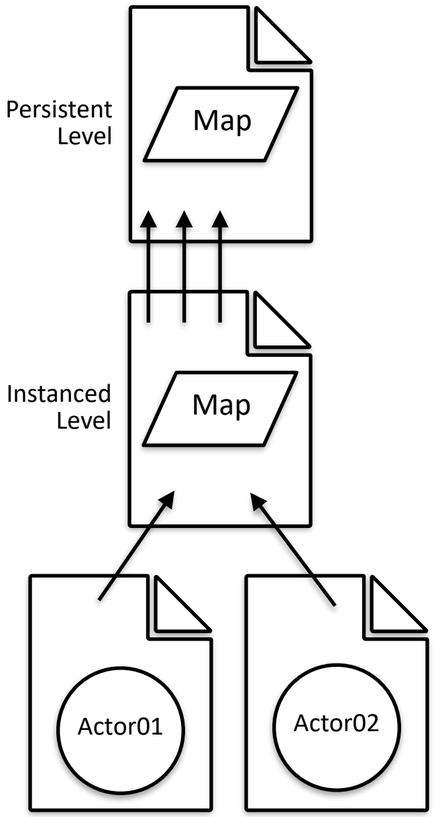
Level Instances

- Provide Unreal with its own version of a “prefab”
- Essentially functions as a sub-level
- Changes are propagated across all instances
- Editing can be done in-situ
- Unpacked at at PIE/Cook time
- Data layers can be assigned to level instances but will be inherited by all owned actors
- Instances can be nested
- Cannot reference actors from outside the level instance without EditorPaths
- Try to organise world content vertically

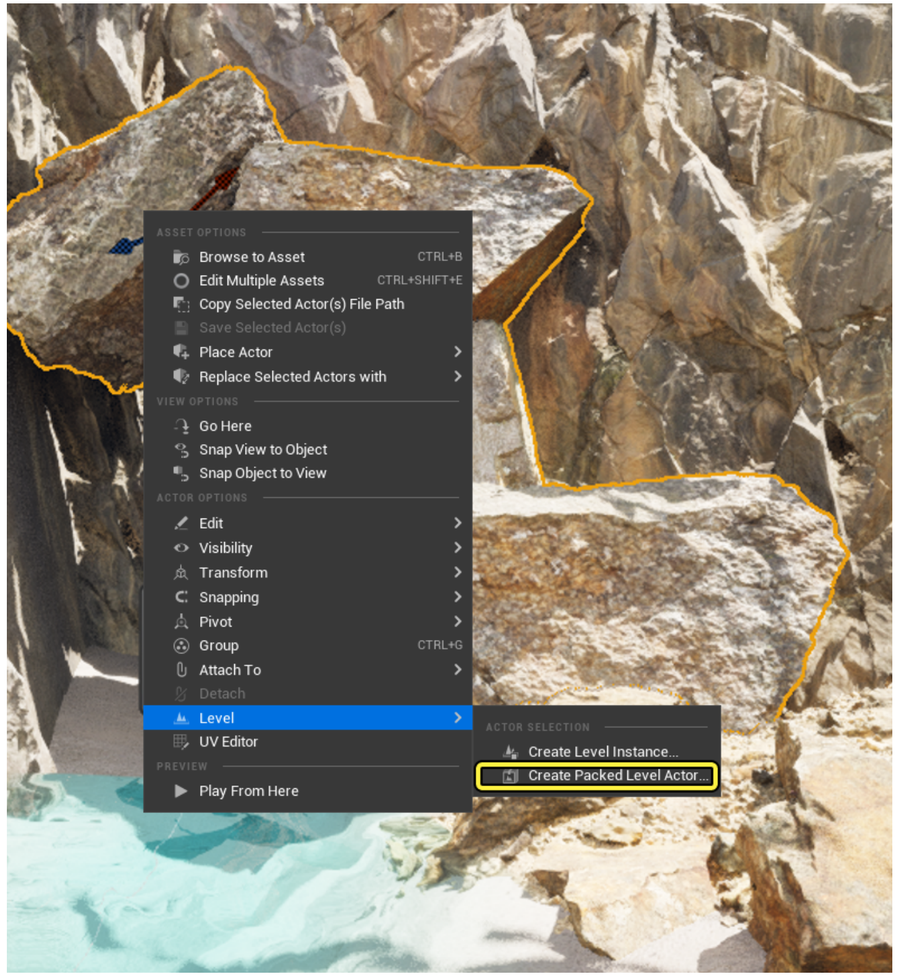


Packed Level Actors

- Discards all components that are not static meshes
- Automatically generates instanced static meshes
- Generates two files
- Avoid editing in the BP
- Be careful moving files around
- Bounds affect streaming

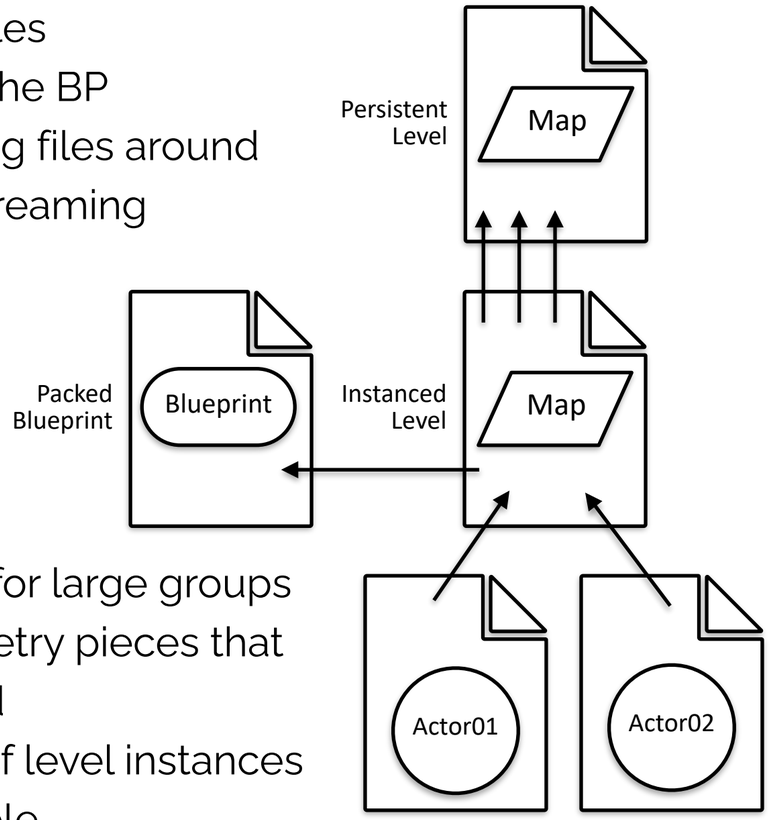


- These are great for large groups of smaller geometry pieces that may be repeated
- Try use instead of level instances whenever possible

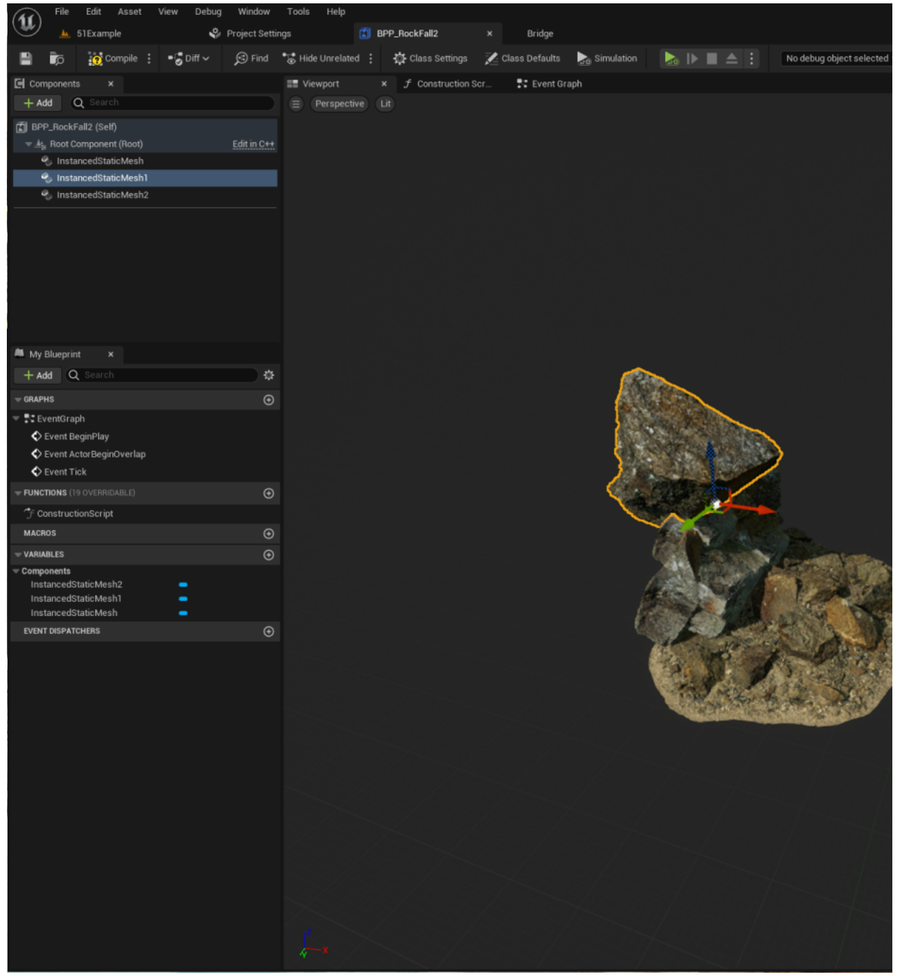


Packed Level Actors

- Discards all components that are not static meshes
- Automatically generates instanced static meshes
- Generates two files
- Avoid editing in the BP
- Be careful moving files around
- Bounds affect streaming

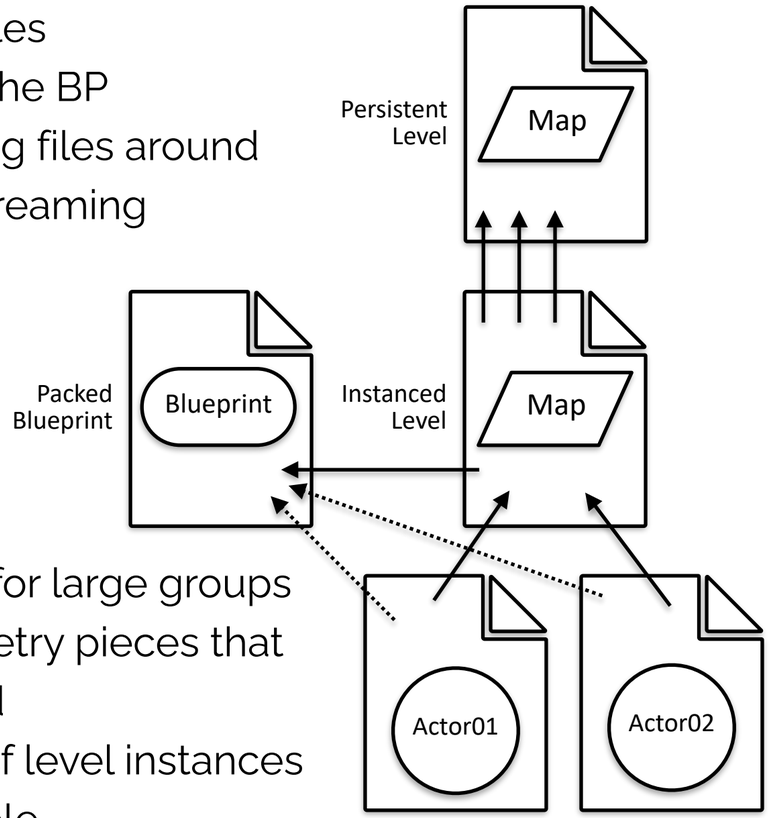


- These are great for large groups of smaller geometry pieces that may be repeated
- Try use instead of level instances whenever possible

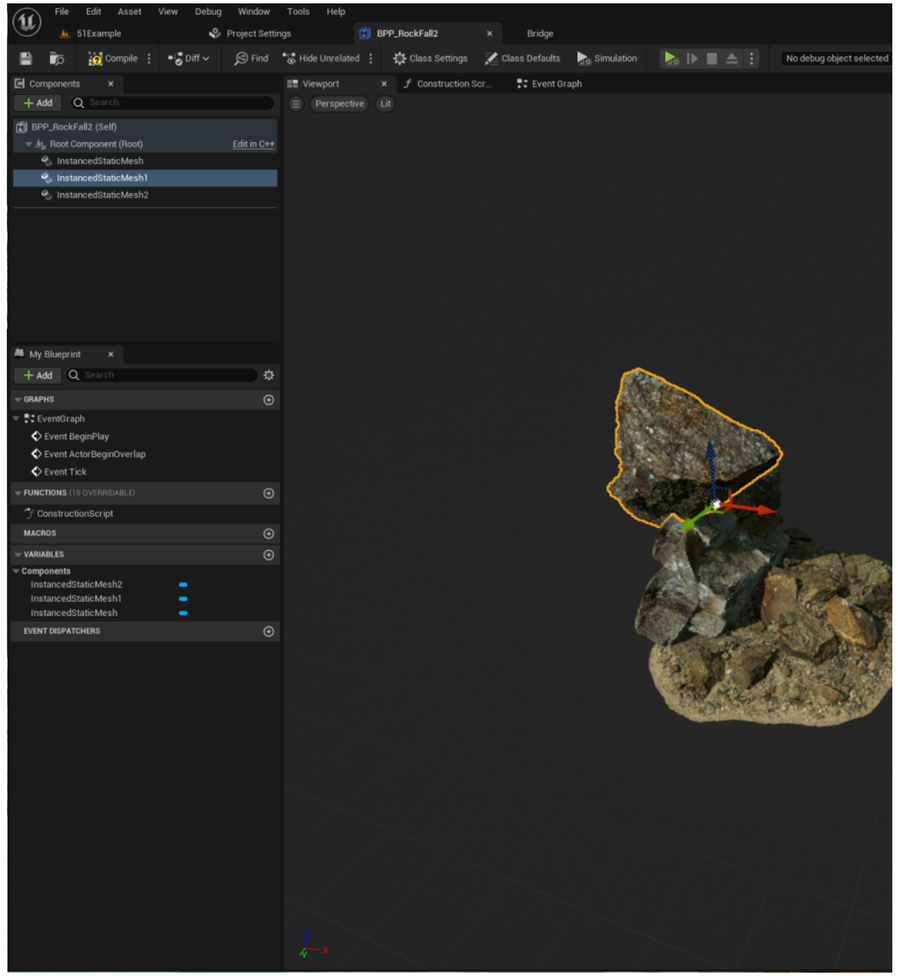


Packed Level Actors

- Discards all components that are not static meshes
- Automatically generates instanced static meshes
- Generates two files
- Avoid editing in the BP
- Be careful moving files around
- Bounds affect streaming

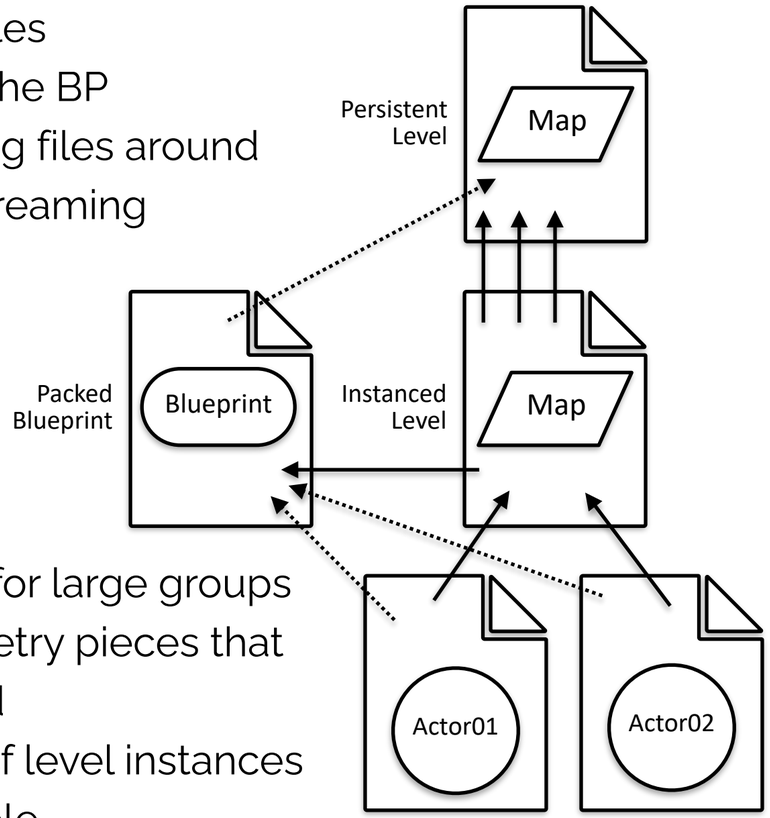


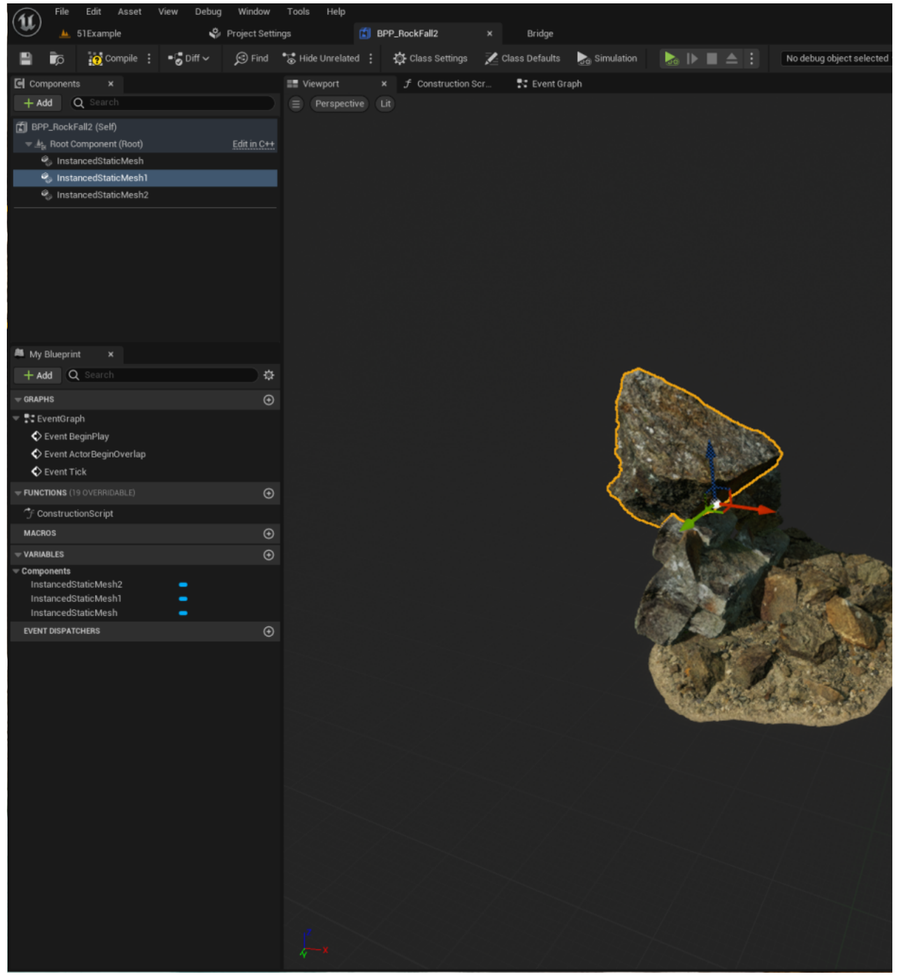
- These are great for large groups of smaller geometry pieces that may be repeated
- Try use instead of level instances whenever possible



Packed Level Actors

- Discards all components that are not static meshes
- Automatically generates instanced static meshes
- Generates two files
- Avoid editing in the BP
- Be careful moving files around
- Bounds affect streaming

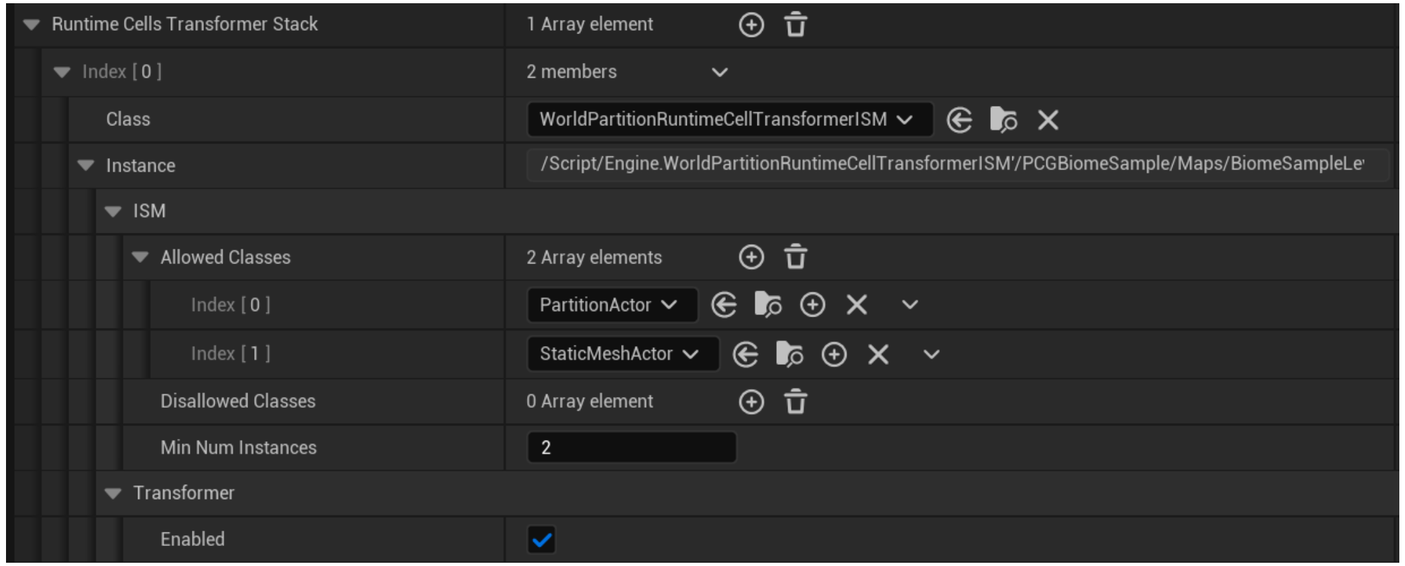




- These are great for large groups of smaller geometry pieces that may be repeated
- Try use instead of level instances whenever possible

Runtime Cell Transformers

- Load and apply arbitrary transformations for actors on a cell-by-cell basis
- Only occurs during streaming generation, not saved
- While for build this is done for all cells on all grids at cook time, it is done on-the-fly during PIE sessions as grid cells stream in



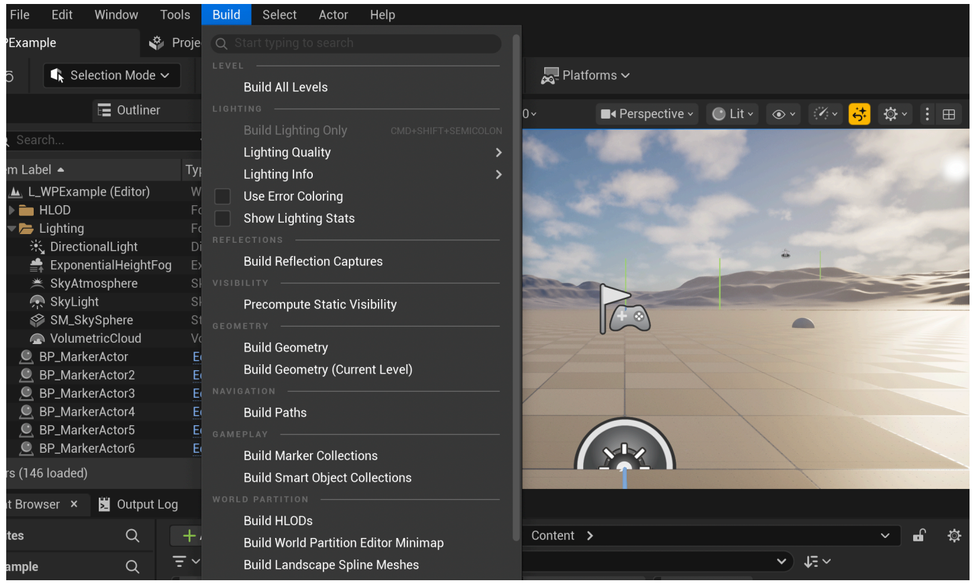
"Runtime Cell Transformers are non-destructive stackable data transformation processes that are applied during the streaming generation phase when cooking and launching into PIE." - Epic's World Building Guide

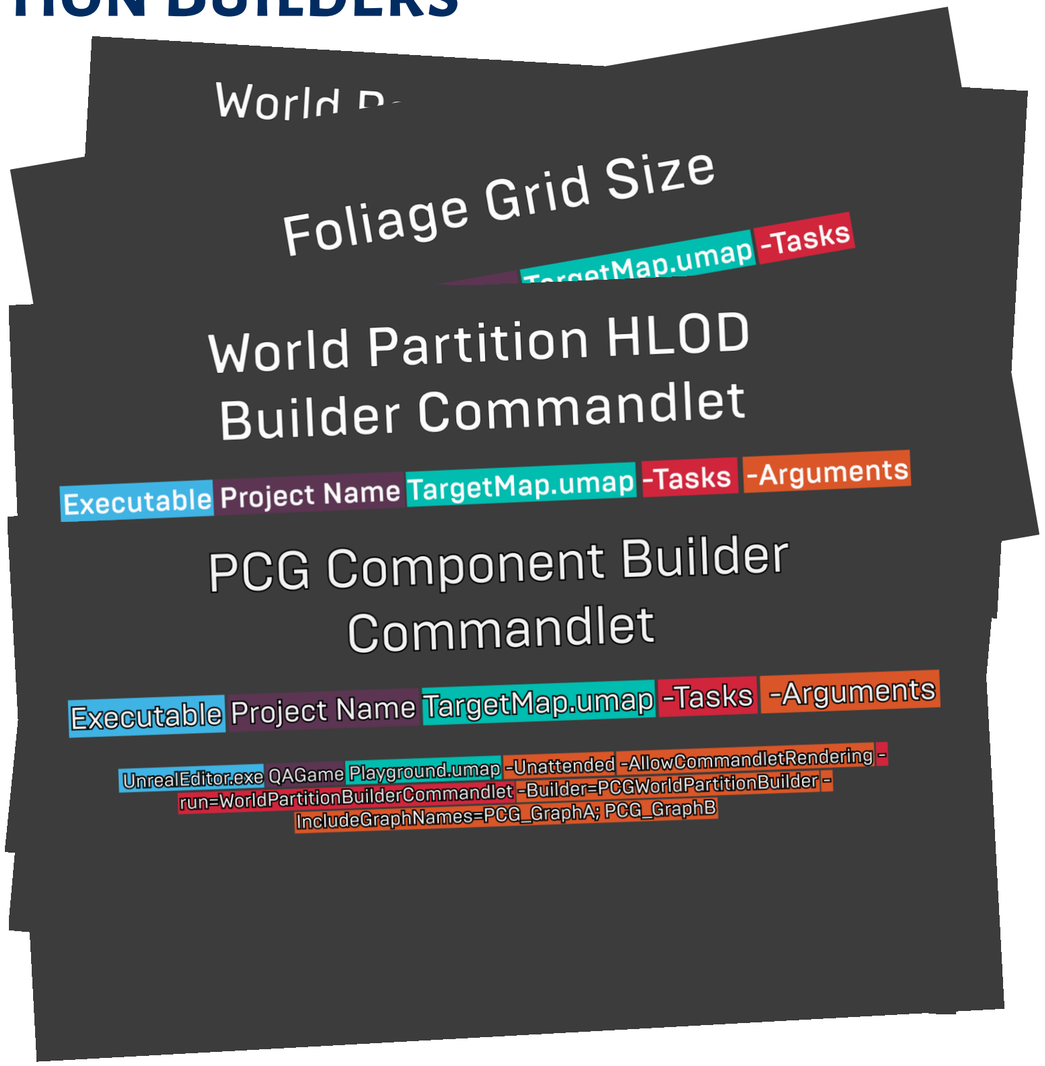
World Partition Builders

- Arbitrary jobs that can be run incrementally across a World Partition level by cell
- Epic provides a number of these exposed by default under the “Build” menu
- Allows for definition and registration of custom builder commandlets

World Partition Builders

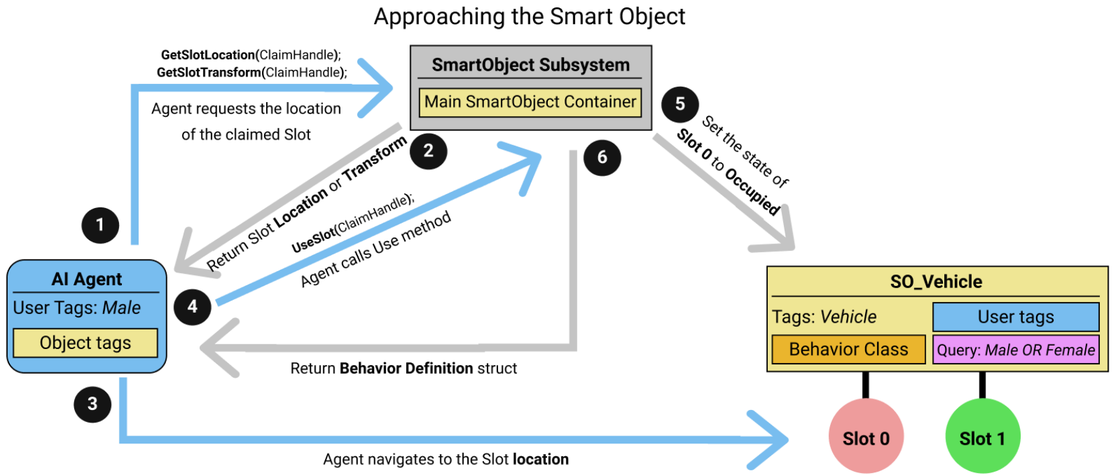
- Arbitrary jobs that can be run incrementally across a World Partition level by cell
- Epic provides a number of these exposed by default under the “Build” menu
- Allows for definition and registration of custom builder commandlets





Smart Objects

- Provide tools for agents to interact with “slots” via a reservation system
- Actors with **USmartObjectComponent**s can be configured with definitions that can hold arbitrary data, or even take control interacting agent behaviour on each slot
- When searching via the subsystem, a limited subset of slot data for currently unloaded Smart Objects can still be retrieved
- Slots held in spatially-partitioned container:
  - Spatial Hash Grid
  - Octree



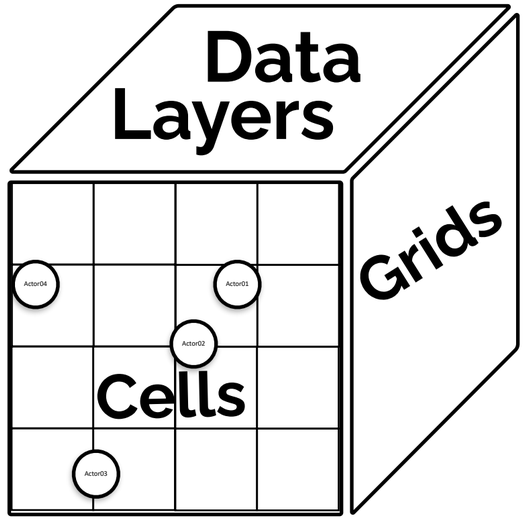
Streaming Systems For Game State Management

A Typical Open World

- Non-linear progression
- Main quest-line with lasting world effects
- Side quests with player ability to drop-in/drop-out
- Interactive map with point of interest markers
- Party system for companion NPCs
- Scheduled world changes
- Time-of-day mechanics

Evaluating Gameplay Requirements

- Assess which systems of World Partition can be leveraged to build out gameplay features with respect to the needs of your game
- Keep in mind possible dependencies and combinatorics involved with both runtime management and markup



Potential Data Layer Feature Compatibility Matrix

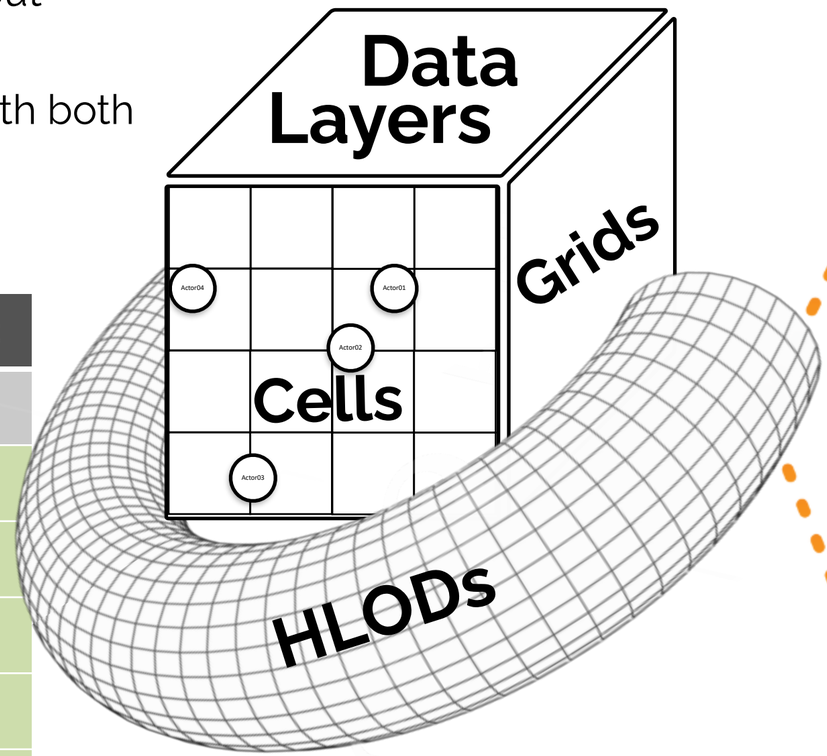
	Common	Scheduling	Quests	Locality	Interiors	Quality	Lighting
Common		—	—	—	—	—	—
Scheduling			✔	✘	✘	✘	✔
Quests				✘	✘	✘	✔
Locality					✔	✘	✔
Interiors						✘	✔
Quality							✔
Lighting							

Evaluating Gameplay Requirements

- Assess which systems of World Partition can be leveraged to build out gameplay features with respect to the needs of your game
- Keep in mind possible dependencies and combinatorics involved with both runtime management and markup

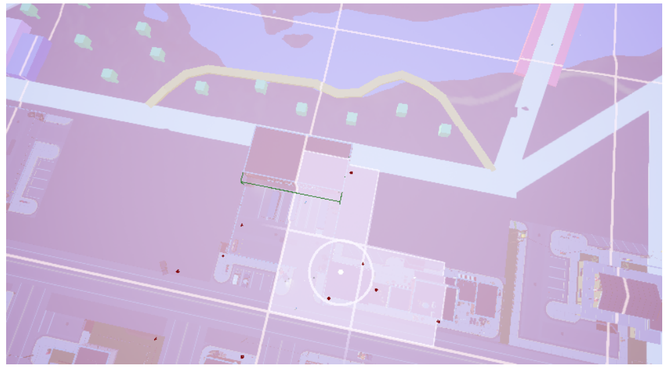
Potential Data Layer Feature Compatibility Matrix

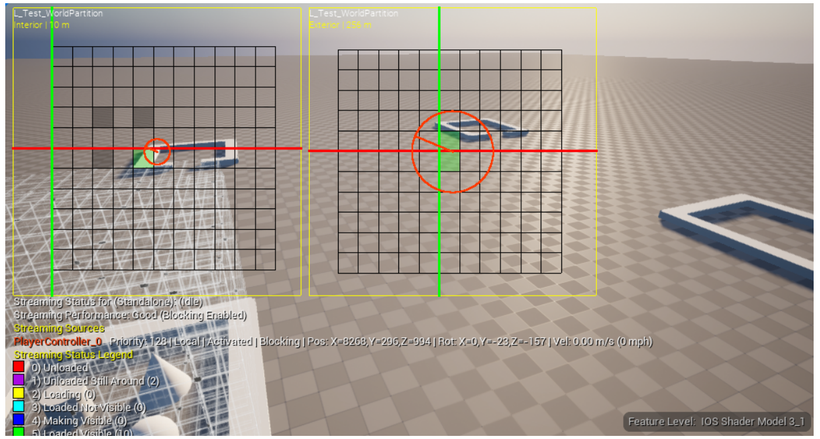
	Common	Scheduling	Quests	Locality	Interiors	Quality	Lighting
Common		—	—	—	—	—	—
Scheduling			✔	✘	✘	✘	✔
Quests				✘	✘	✘	✔
Locality					✔	✘	✔
Interiors						✘	✔
Quality							✔
Lighting							

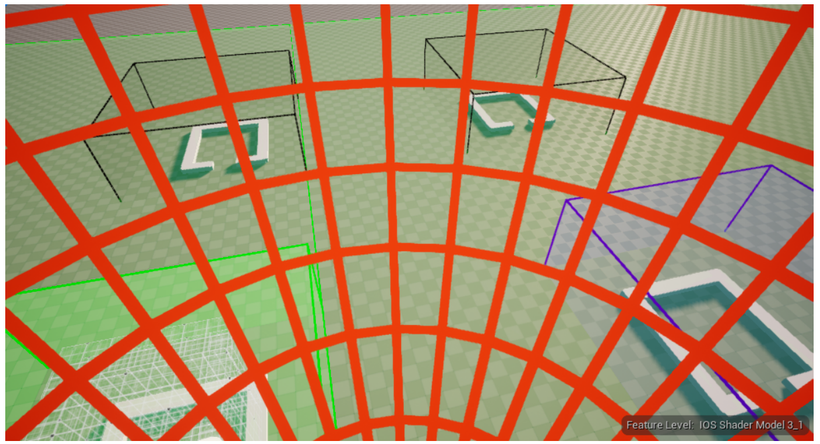


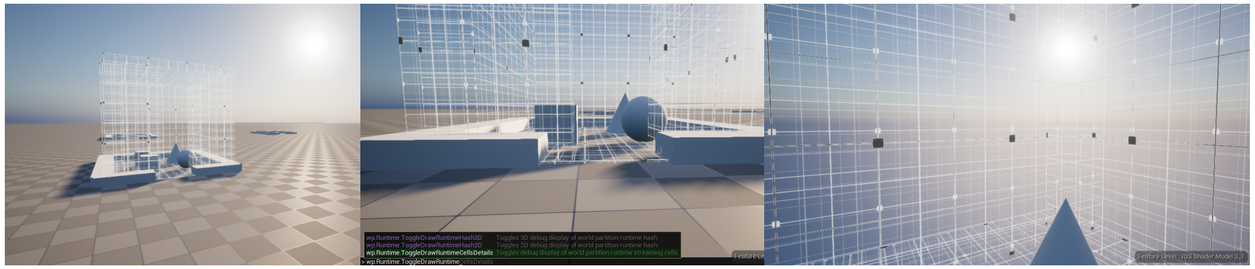
Grids As Interiors, Exteriors, & Performance Settings

- Grids used for handling interior/exterior streaming
- **UWorldPartitionMultiStreamingSourceComponent** was added to support multiple source shapes across presets
- Many interior grids is ok for level streaming
- Fully unloading exterior is not recommended unless applied to large interiors





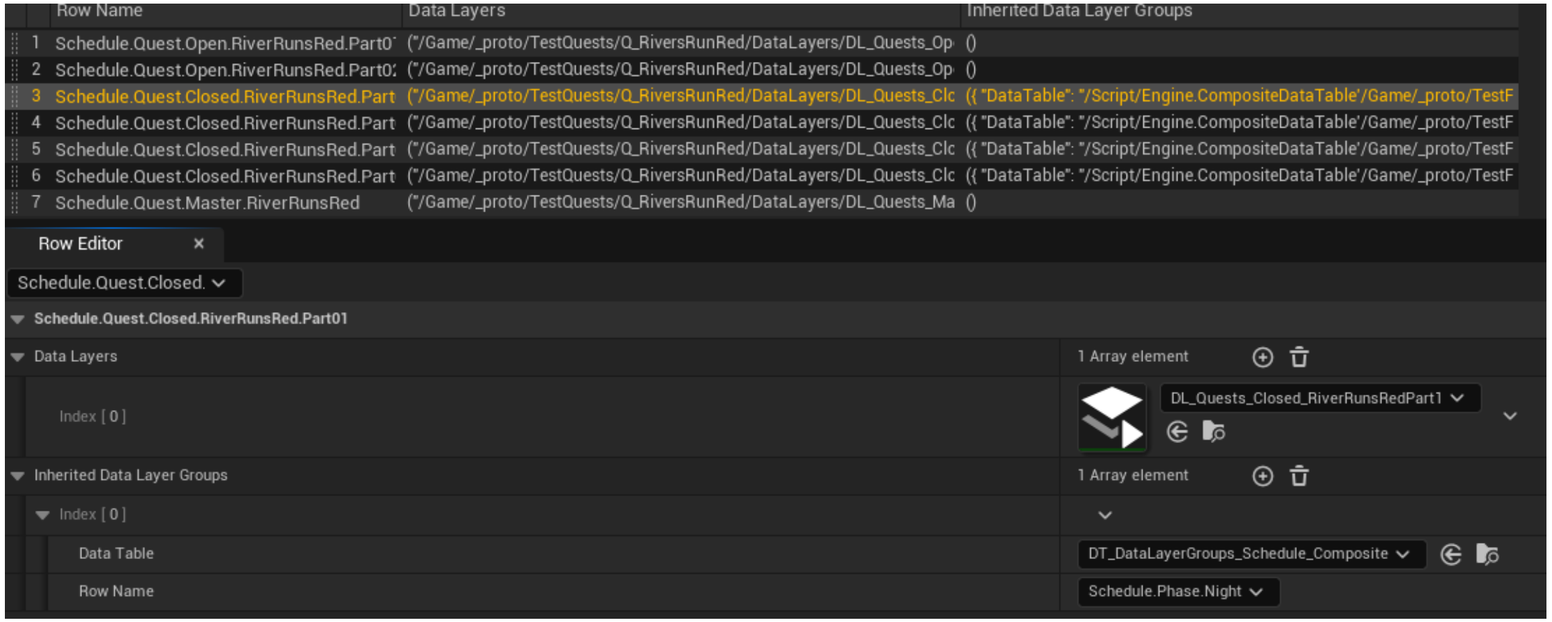


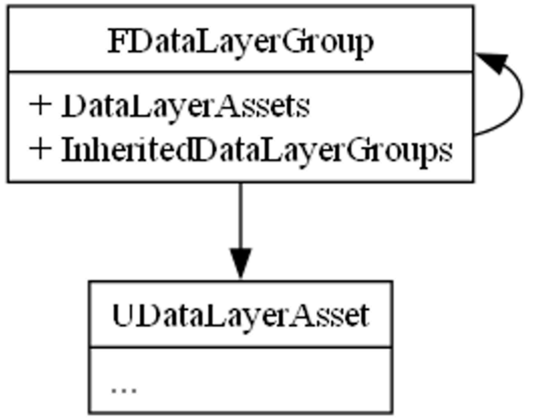


Data Layers As Schedules, Quests, & World States

- OR behaviour makes these concepts reasonably compatible
- Data Layers make for a powerful tool in manipulating scheduling

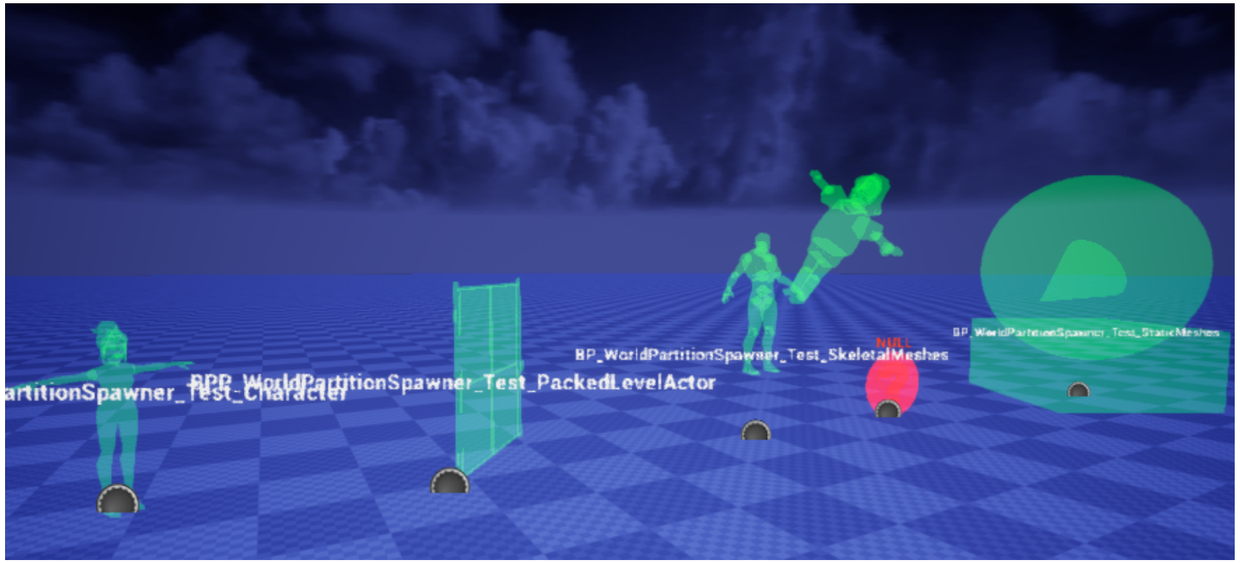
- **FDataLayerGroup** provides hierarchical specification of inherited groups
  - Multiple schedule groups can be enabled simultaneously

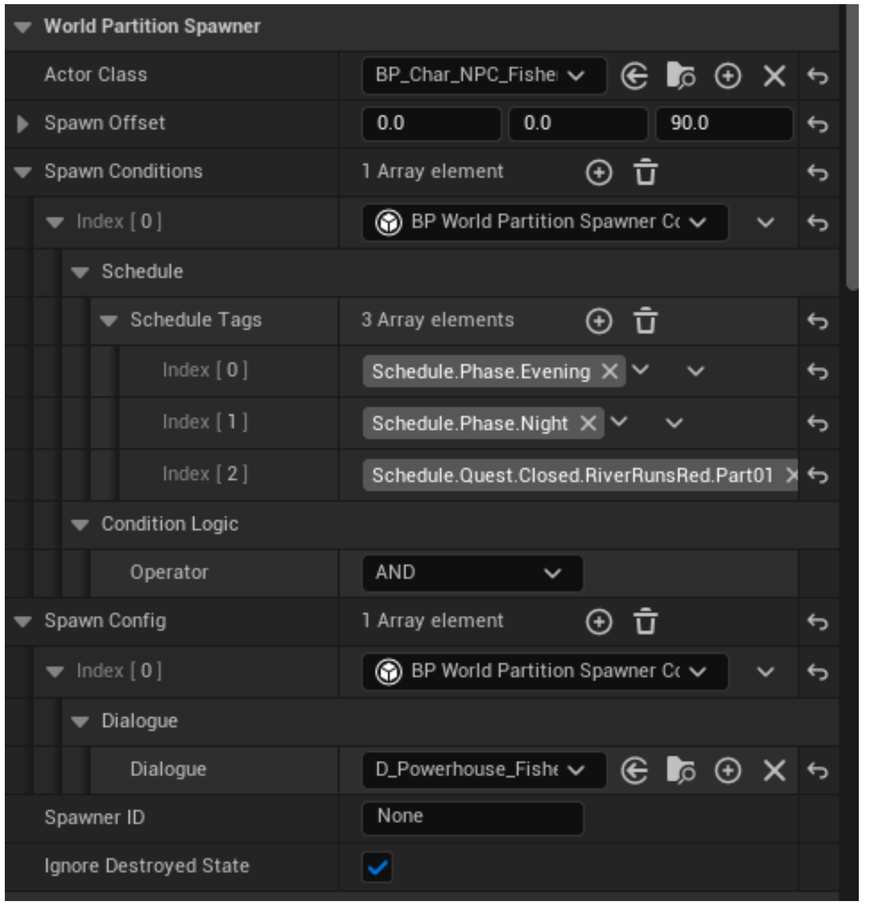




World Partition Spawners

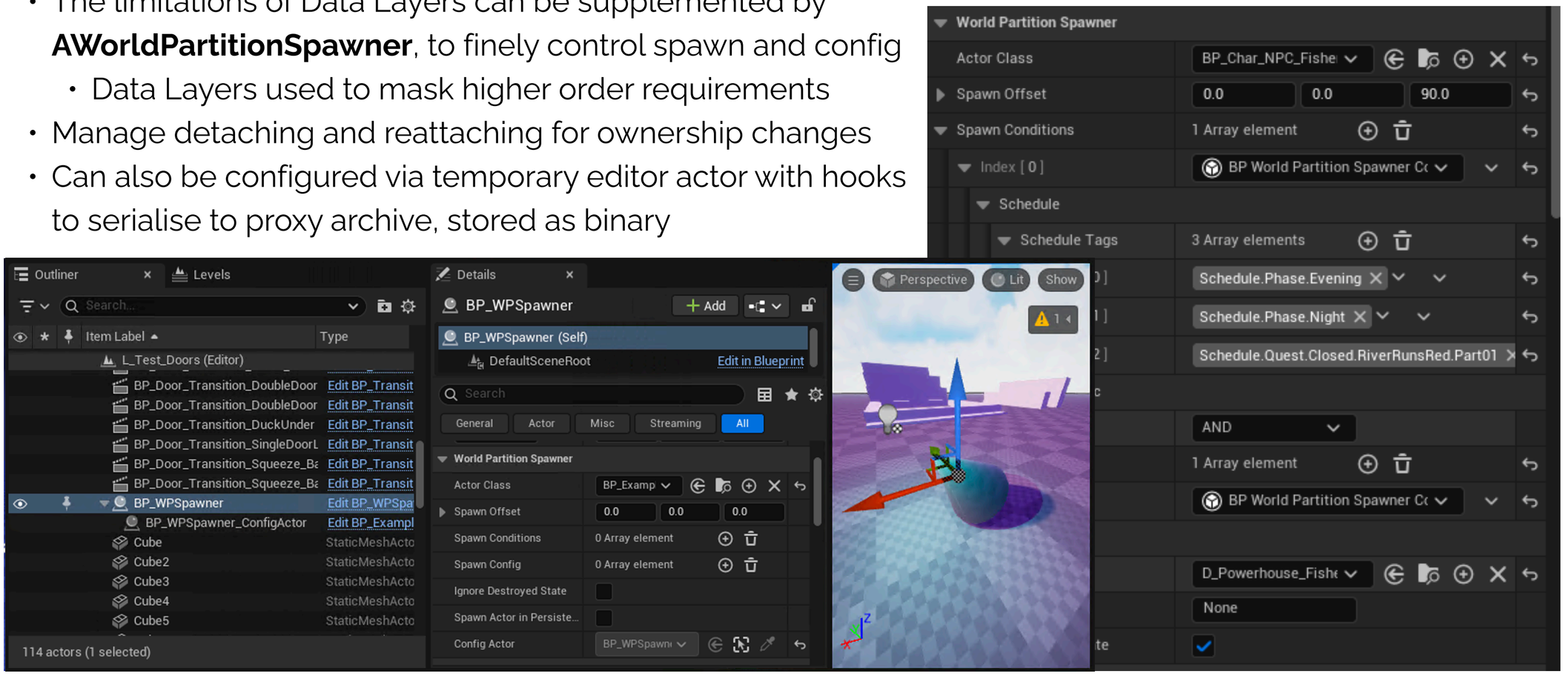
- The limitations of Data Layers can be supplemented by **AWorldPartitionSpawner**, to finely control spawn and config
  - Data Layers used to mask higher order requirements
- Manage detaching and reattaching for ownership changes
- Can also be configured via temporary editor actor with hooks to serialise to proxy archive, stored as binary





World Partition Spawners

- The limitations of Data Layers can be supplemented by **AWorldPartitionSpawner**, to finely control spawn and config
  - Data Layers used to mask higher order requirements
- Manage detaching and reattaching for ownership changes
- Can also be configured via temporary editor actor with hooks to serialise to proxy archive, stored as binary



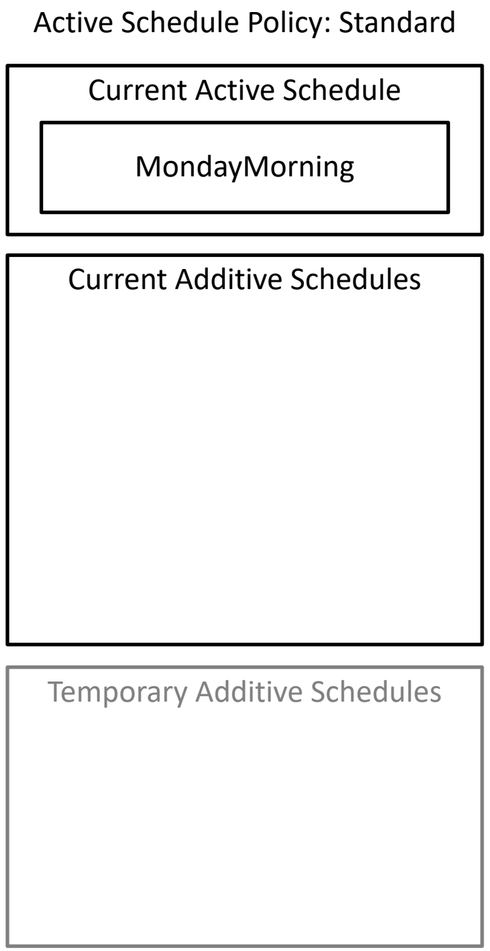
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



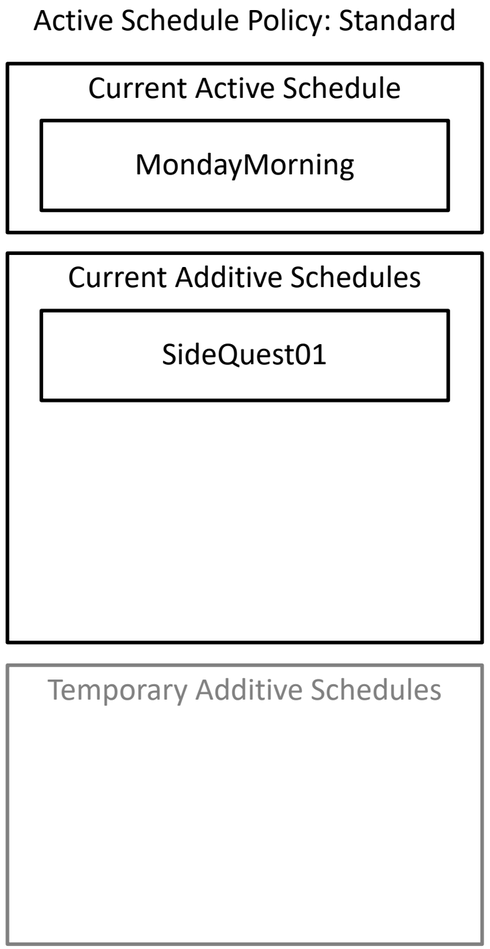
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



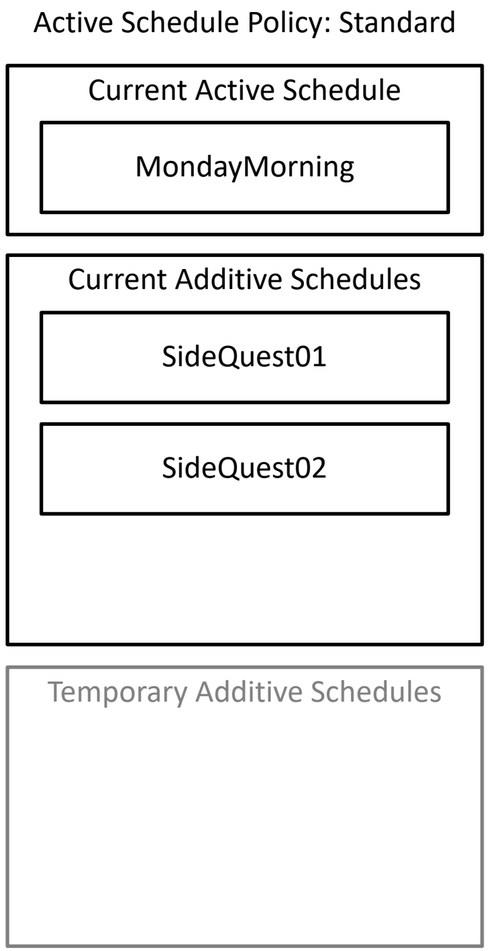
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



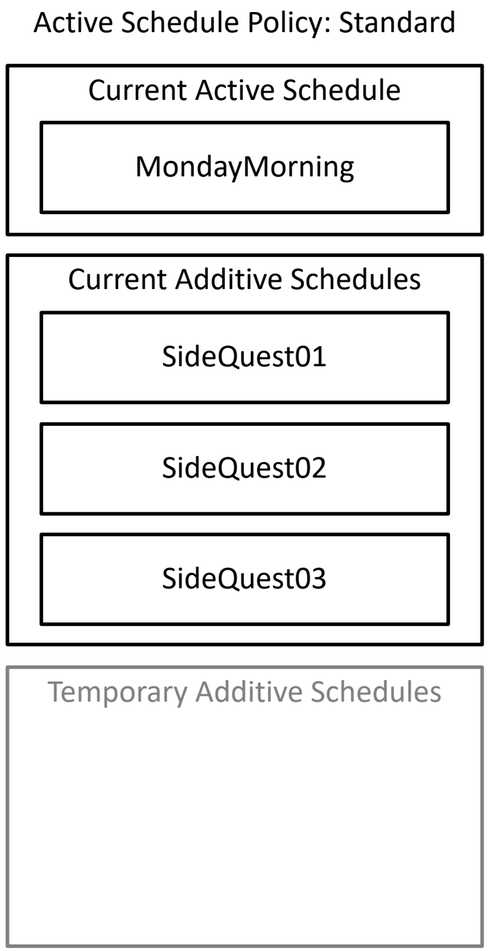
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



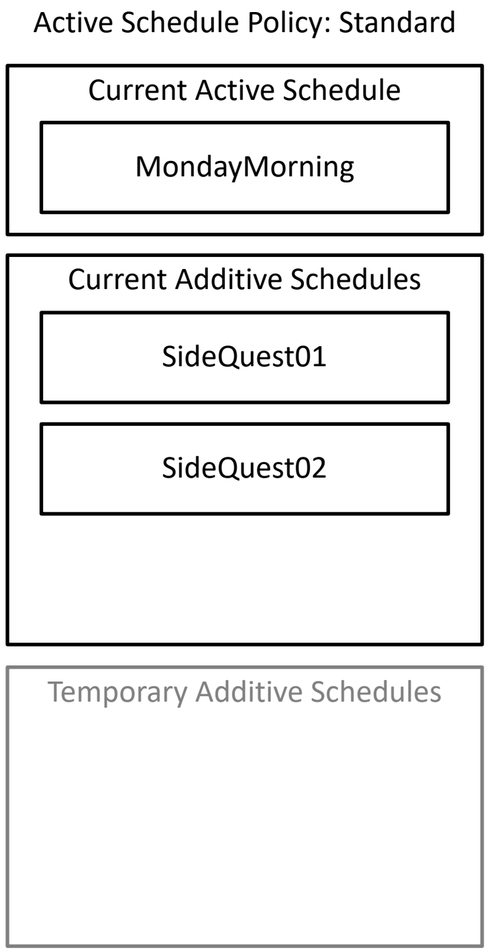
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



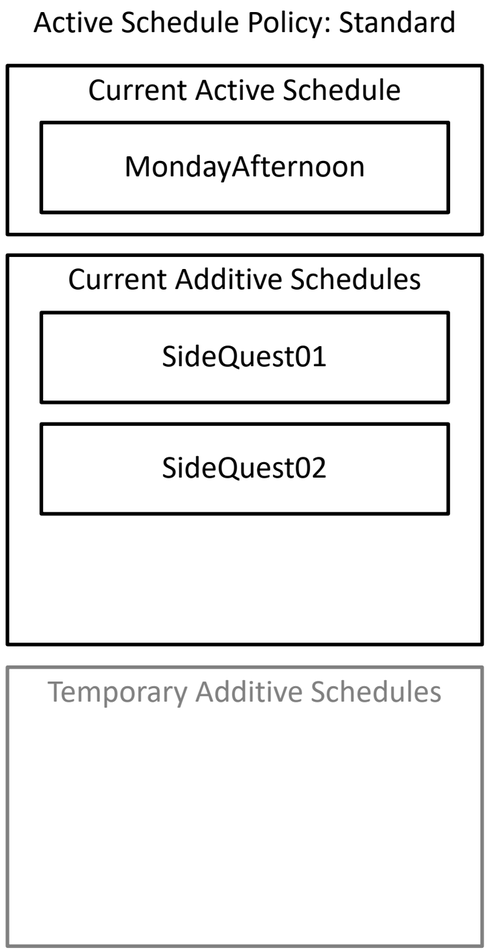
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



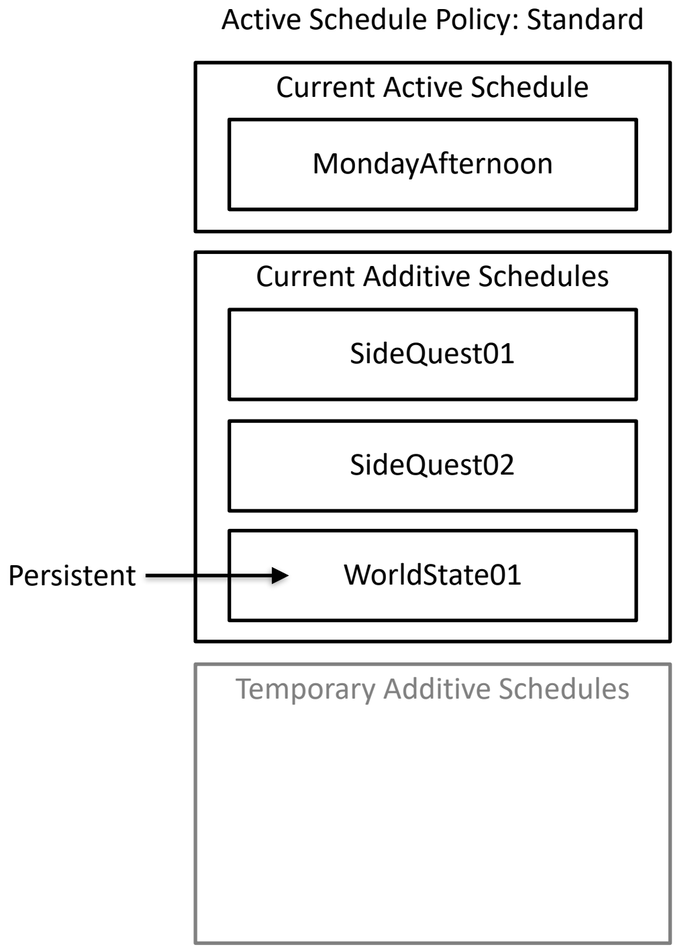
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



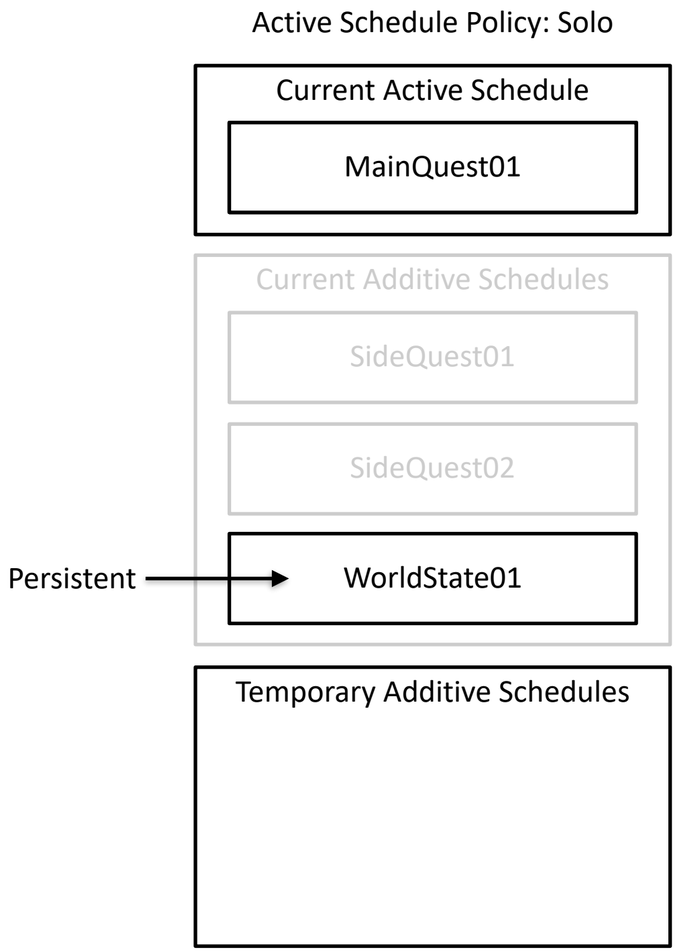
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



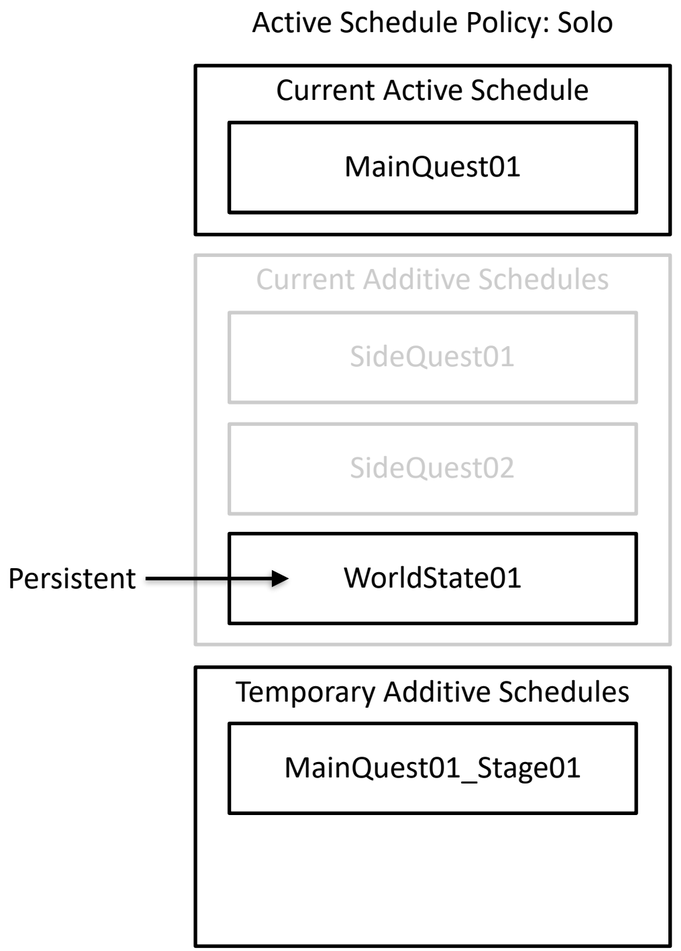
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



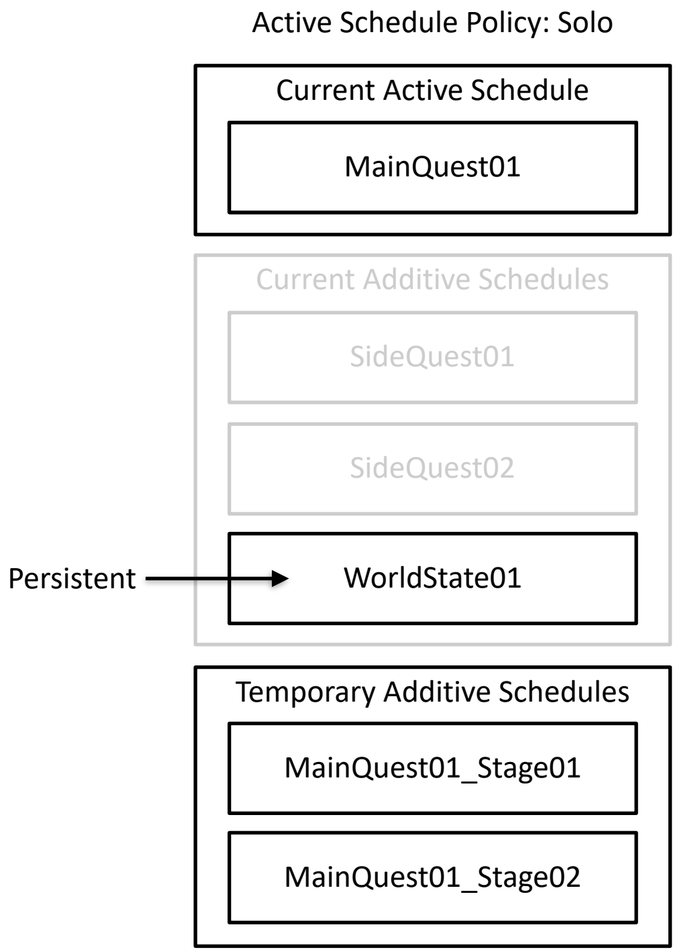
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions



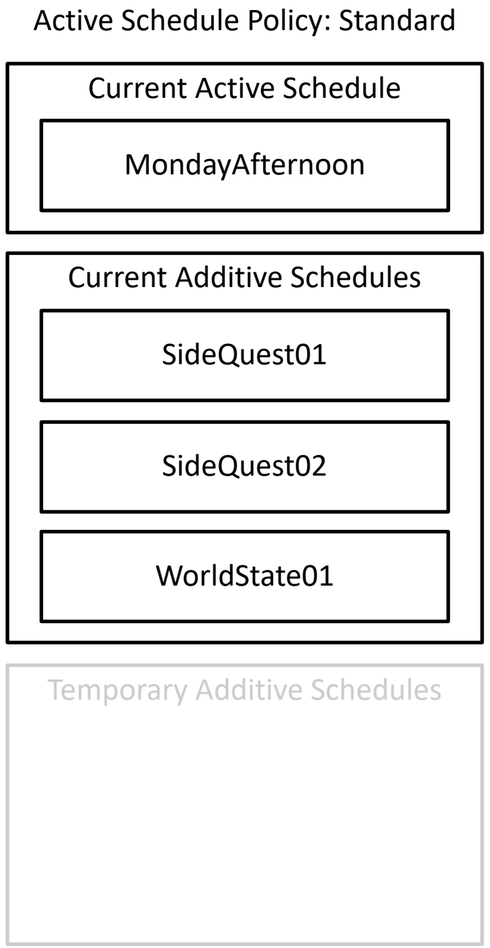
Open & Closed Schedules

- Use “open” and “closed” schedules for controlling additive world state with managed preloading

- Schedule can be modified with activation policies of:
  - **Standard:** Replace the current active schedule, keep additives
  - **Additive:** Add the schedule into the current additive overlay
  - **Solo:** Replace the current active schedule, ignore additives

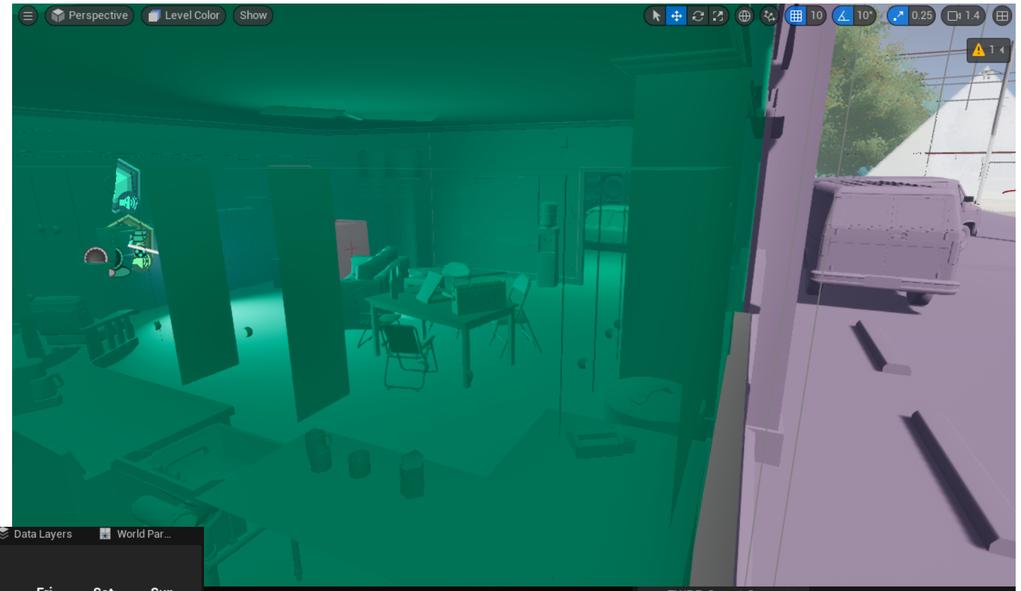
- Temporary additives can be overlayed during solos for finer controls - will be discarded on policy change
- Persistent schedules allow world state to affect all policies

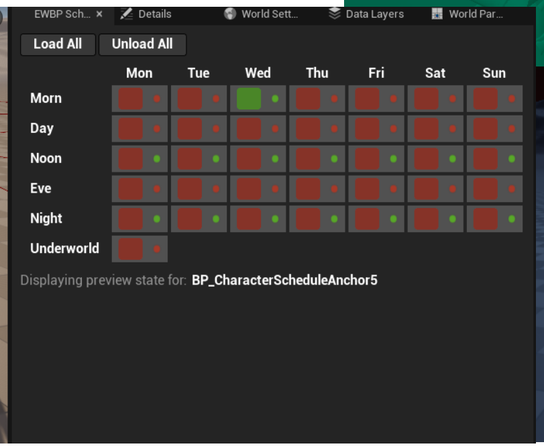
- Can be linked to the lighting and environment systems to force certain properties such as time-of-day and weather conditions

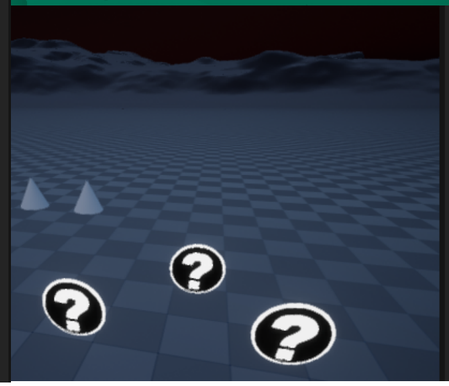


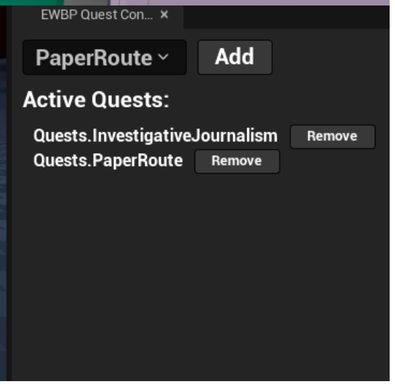
Custom Markup & Tooling Workflows

- Runtime Streaming Grid Actor Colouration
- Character Appointment Spawners
- Lighting previews alongside schedule in-editor
- Schedule preview & management controls



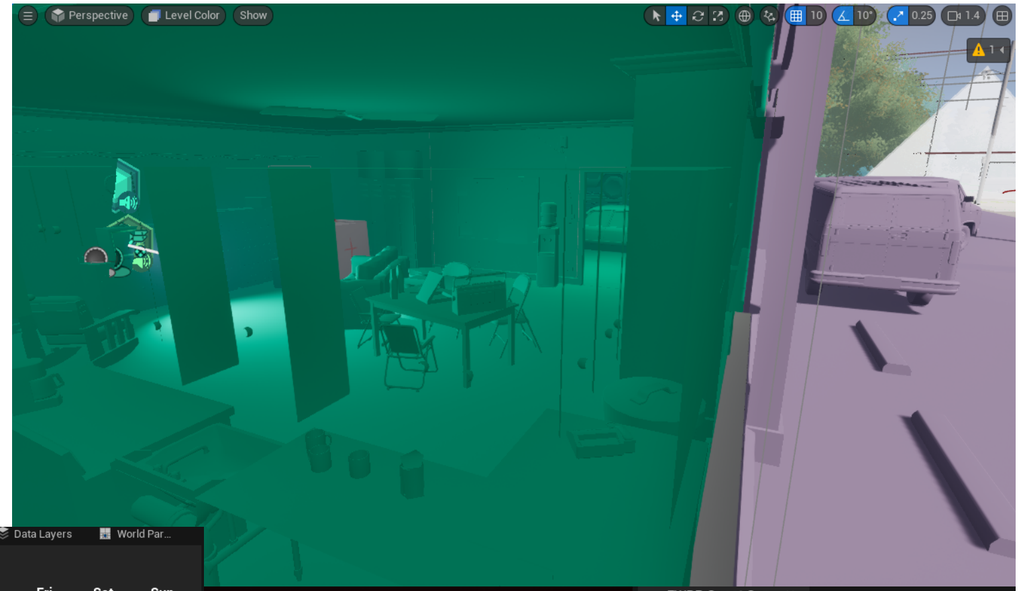


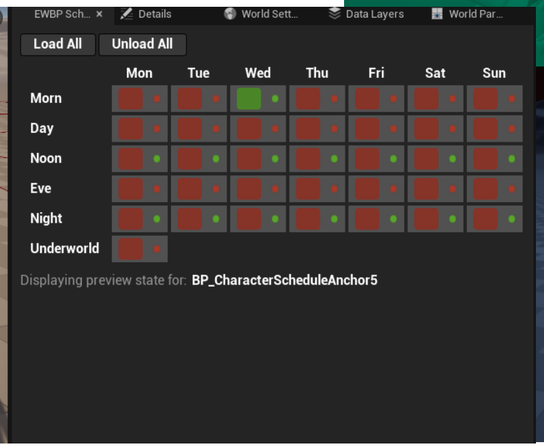


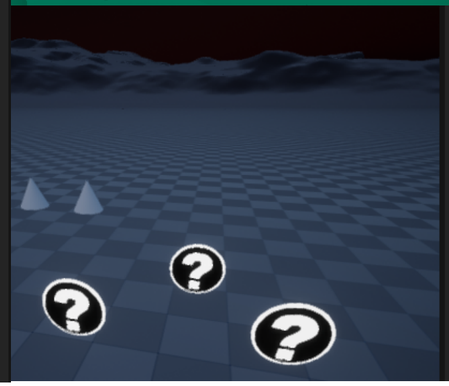


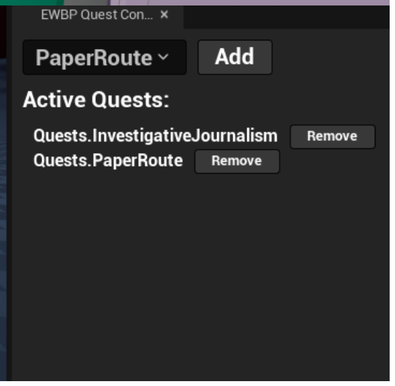
Custom Markup & Tooling Workflows

- Runtime Streaming Grid Actor Colouration
- Character Appointment Spawners
- Lighting previews alongside schedule in-editor
- Schedule preview & management controls









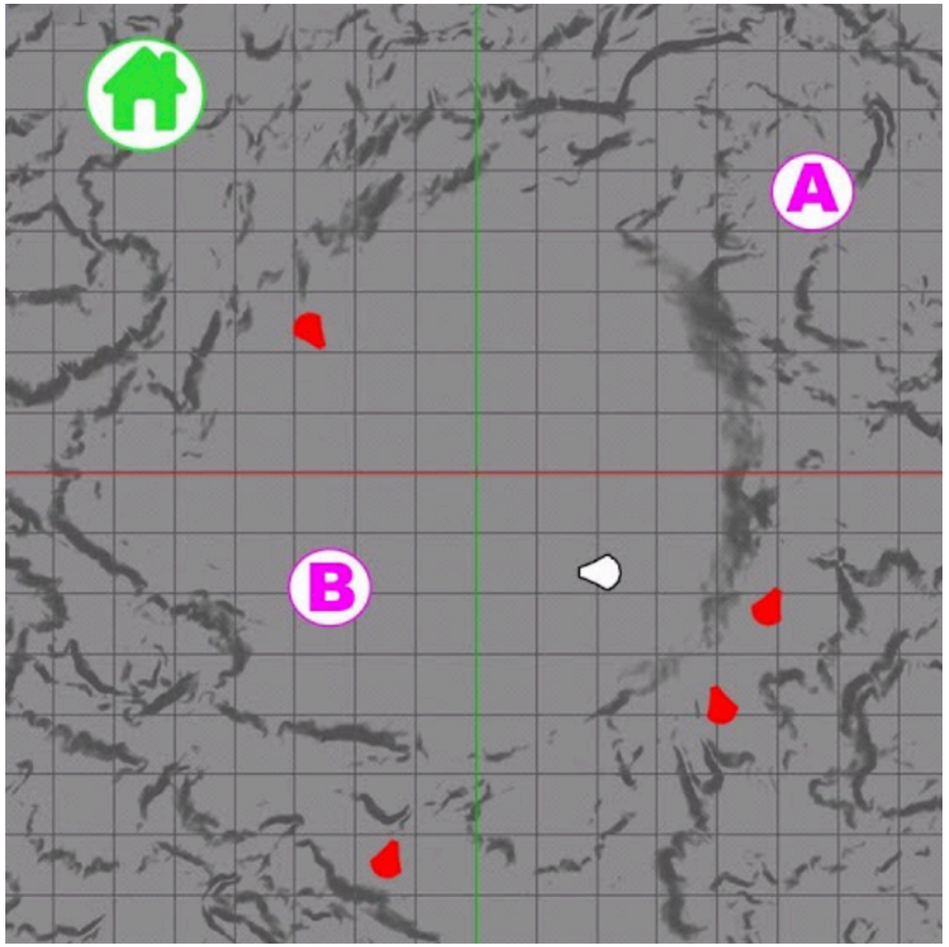
Level Streaming Watch

- The **ULevelStreamingWatch** object allows owners to easily bind to and query world streaming state for either the whole world or just for a specific source
  - Helps handle slow streaming and provides feedback for dependent features

- Used across several systems to manage streaming-aware functionality:
  - World startup and configuration
  - Teleportation warm-up and execution
  - Interior/exterior transitions
  - Managing airlocks between scripted level segments

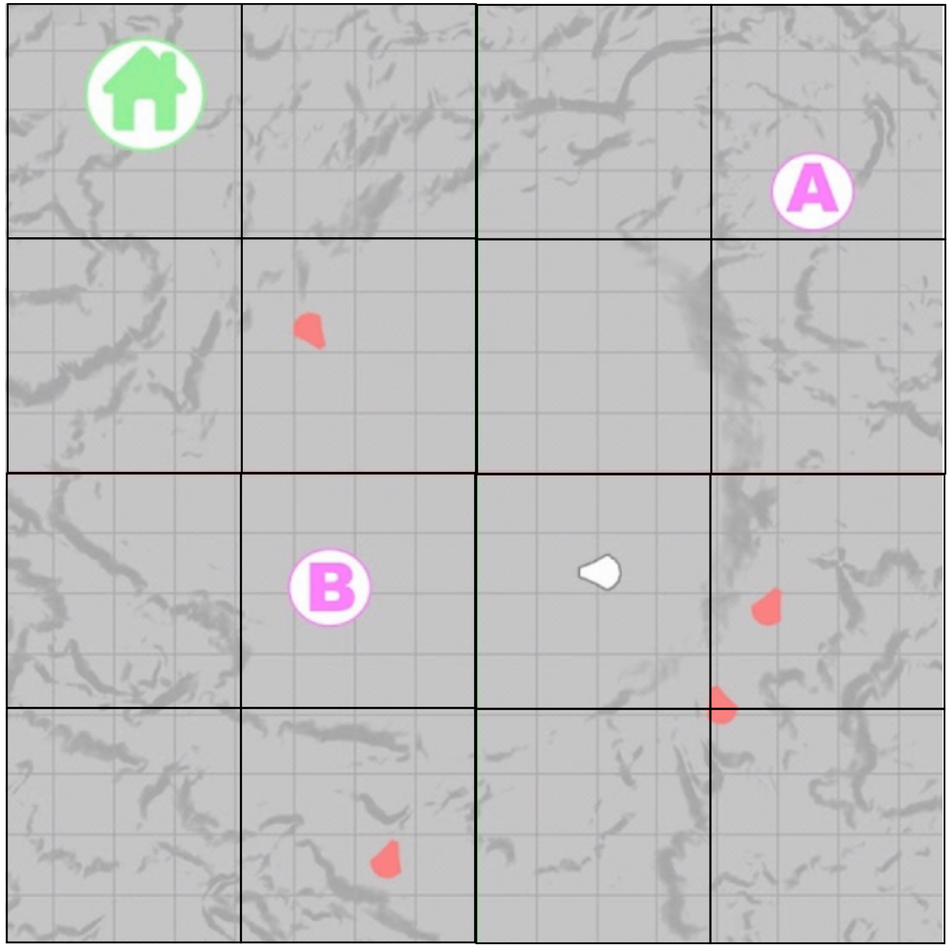
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



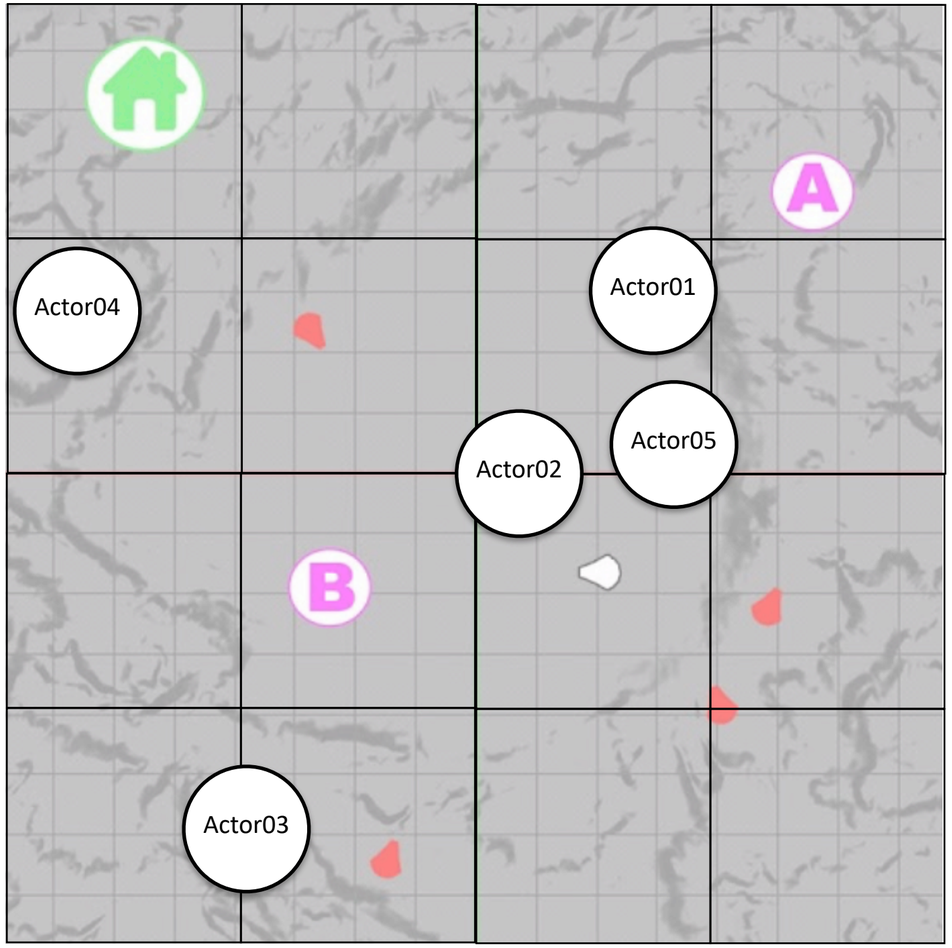
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



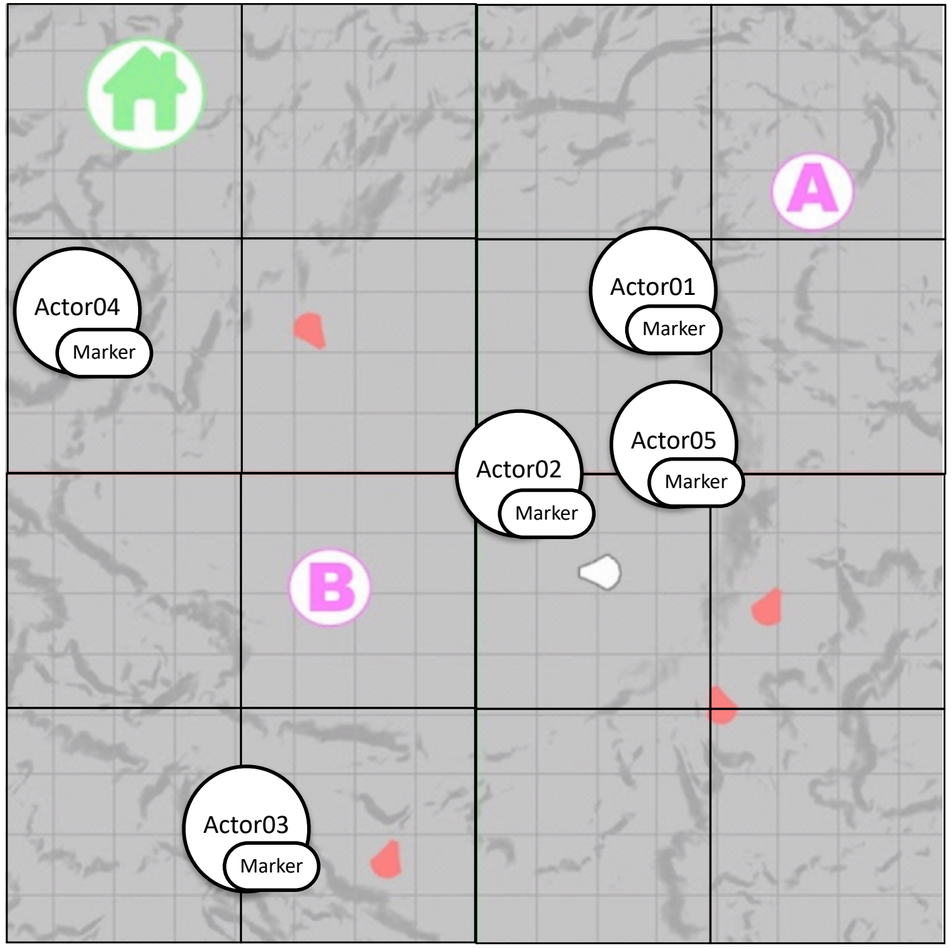
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



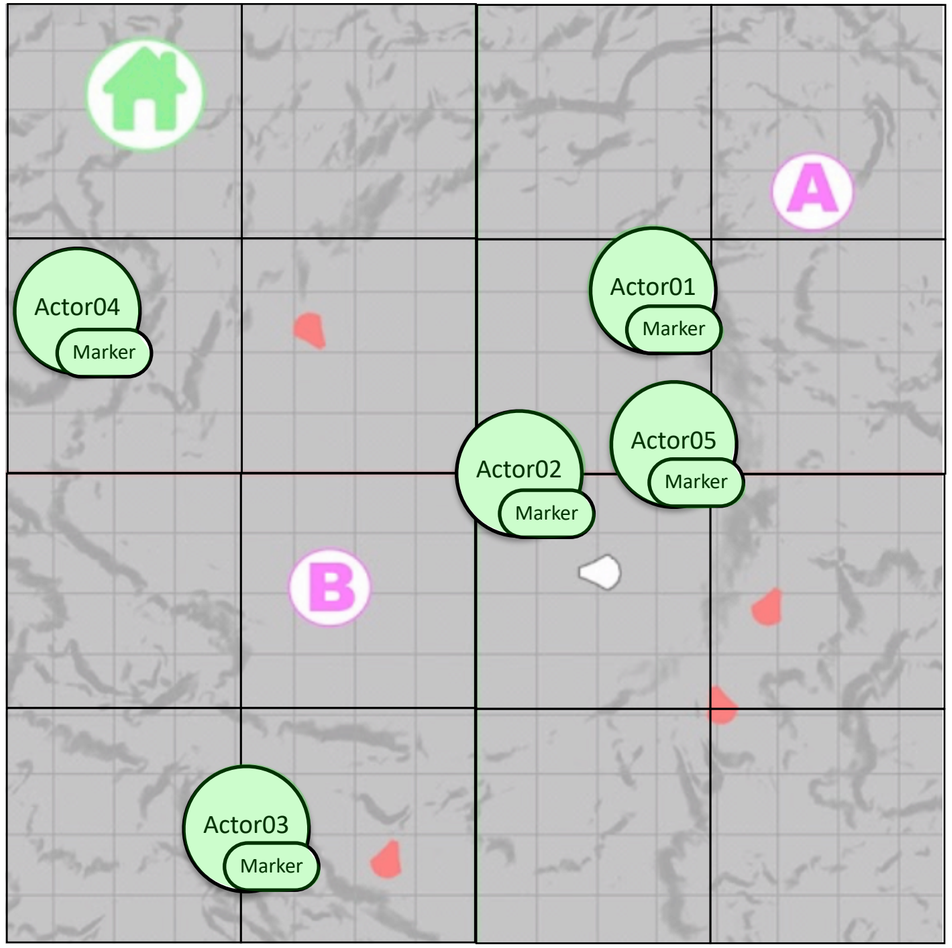
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



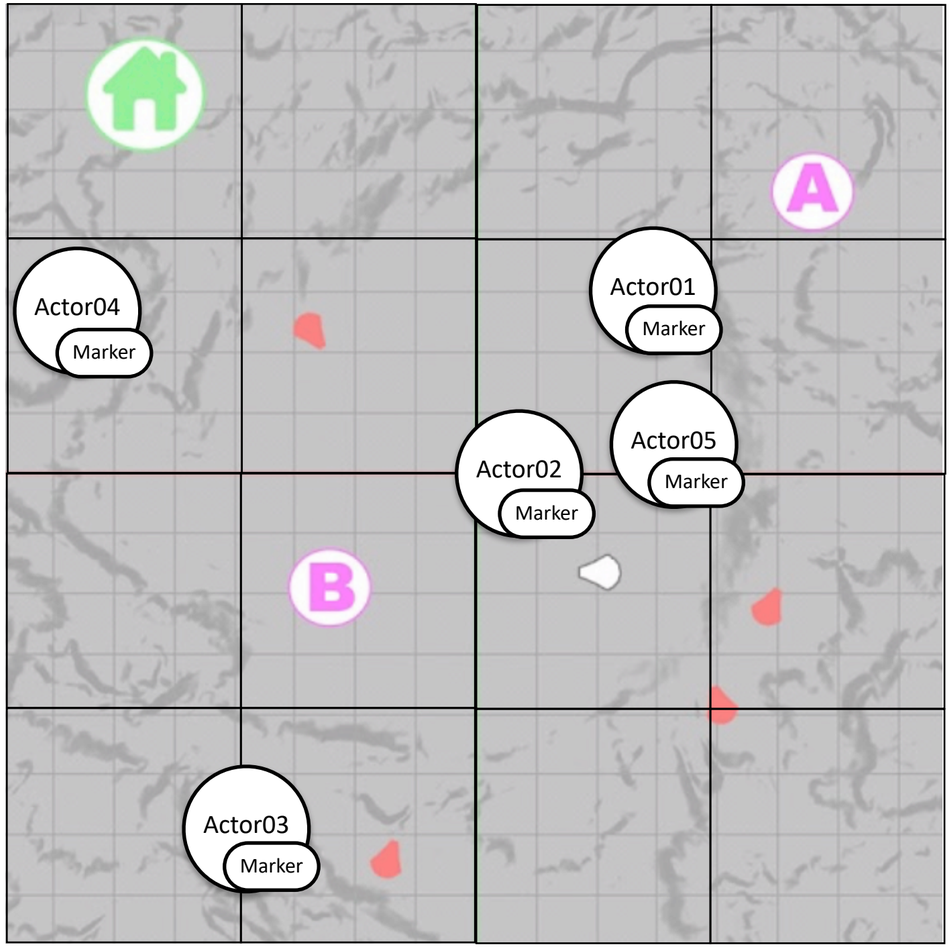
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



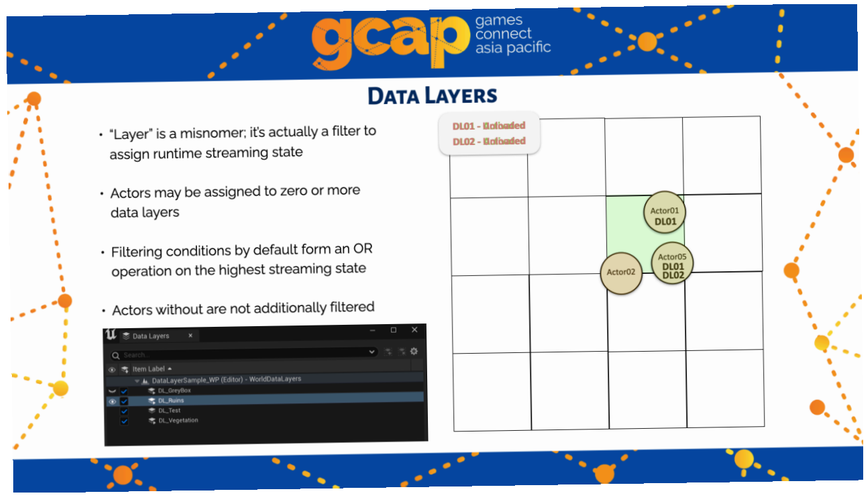
A Marker System

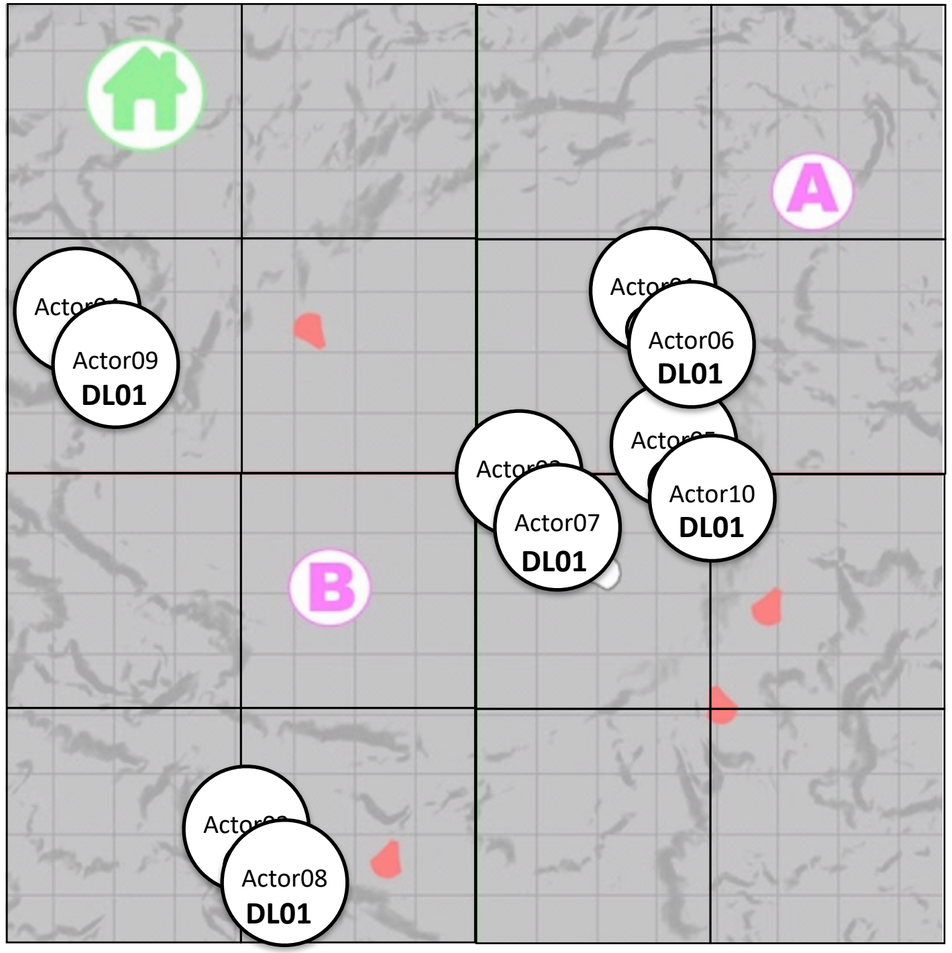
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



A Marker System

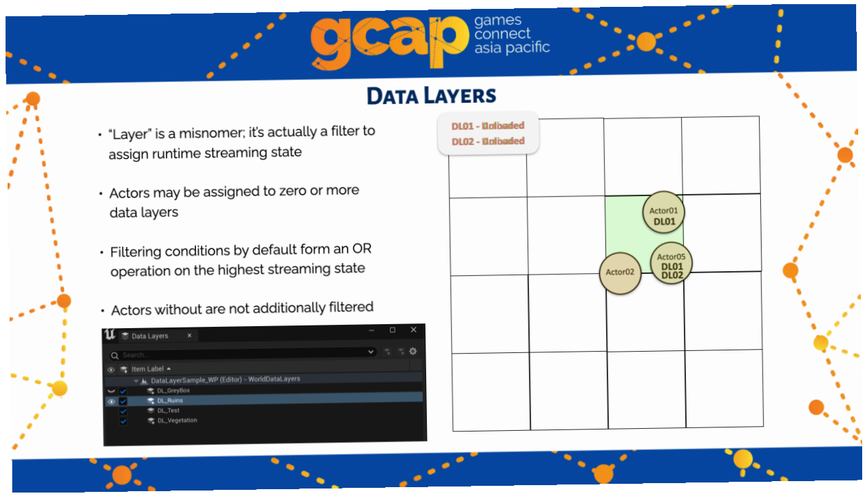
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results

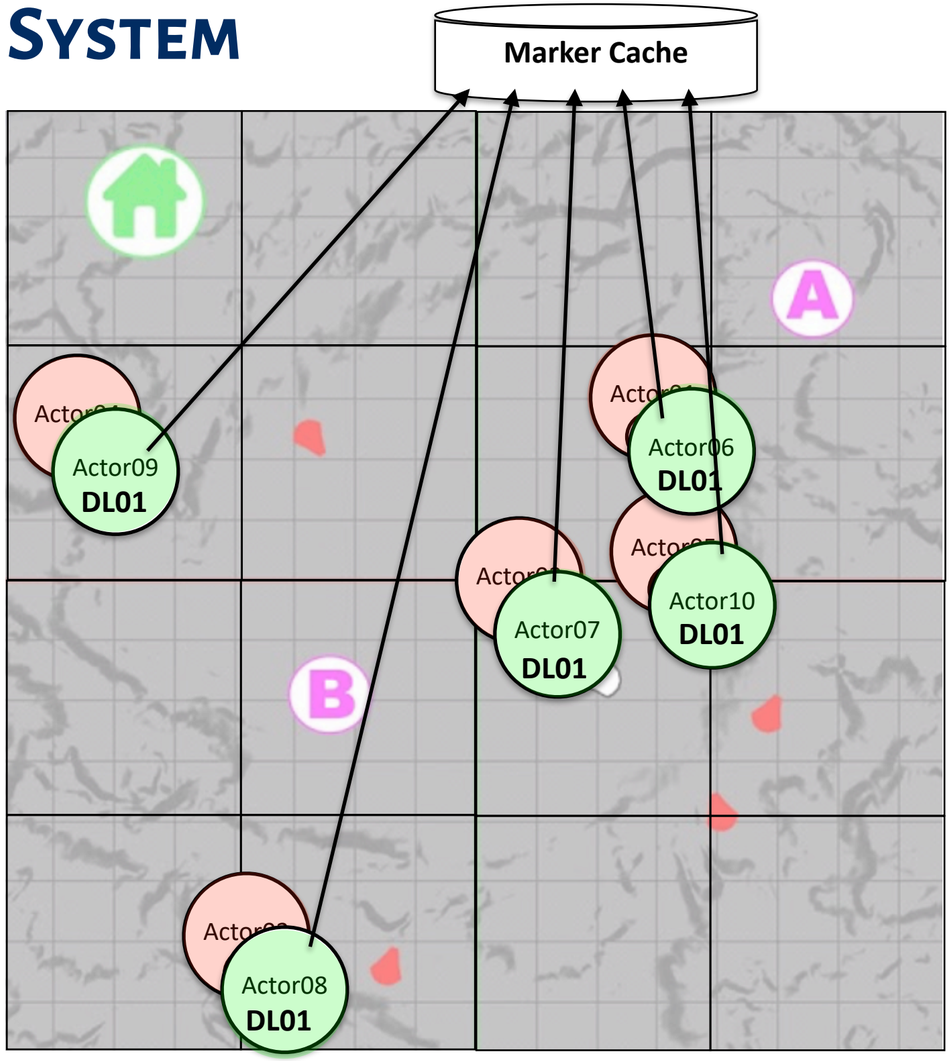




A Marker System

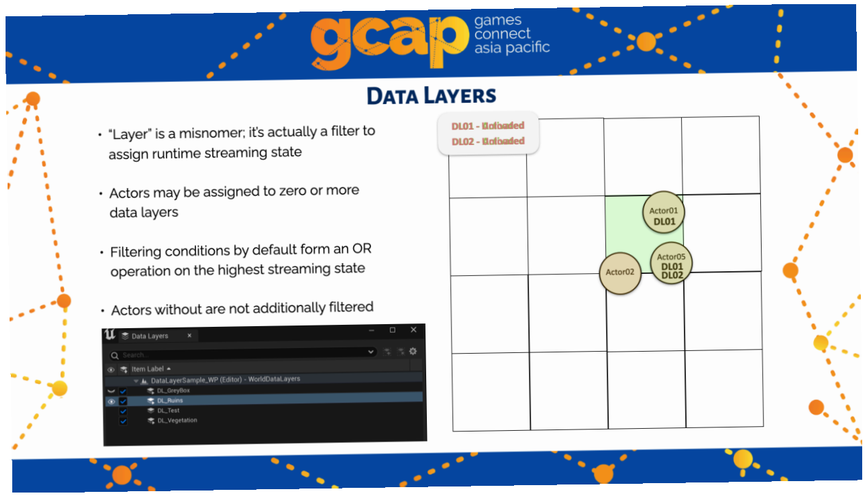
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results

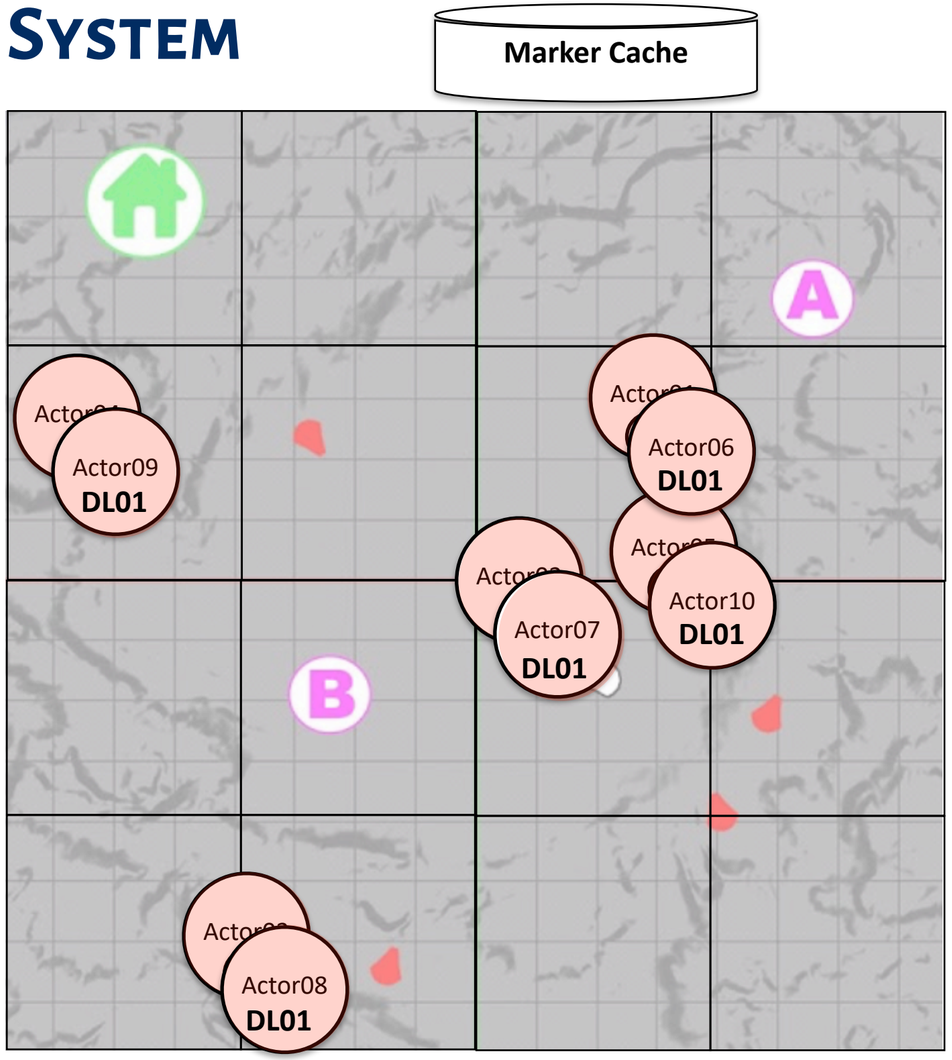




A Marker System

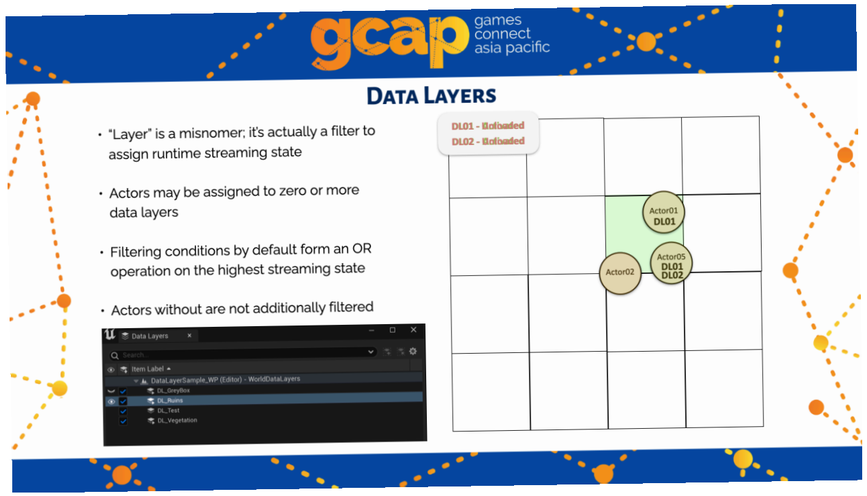
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results

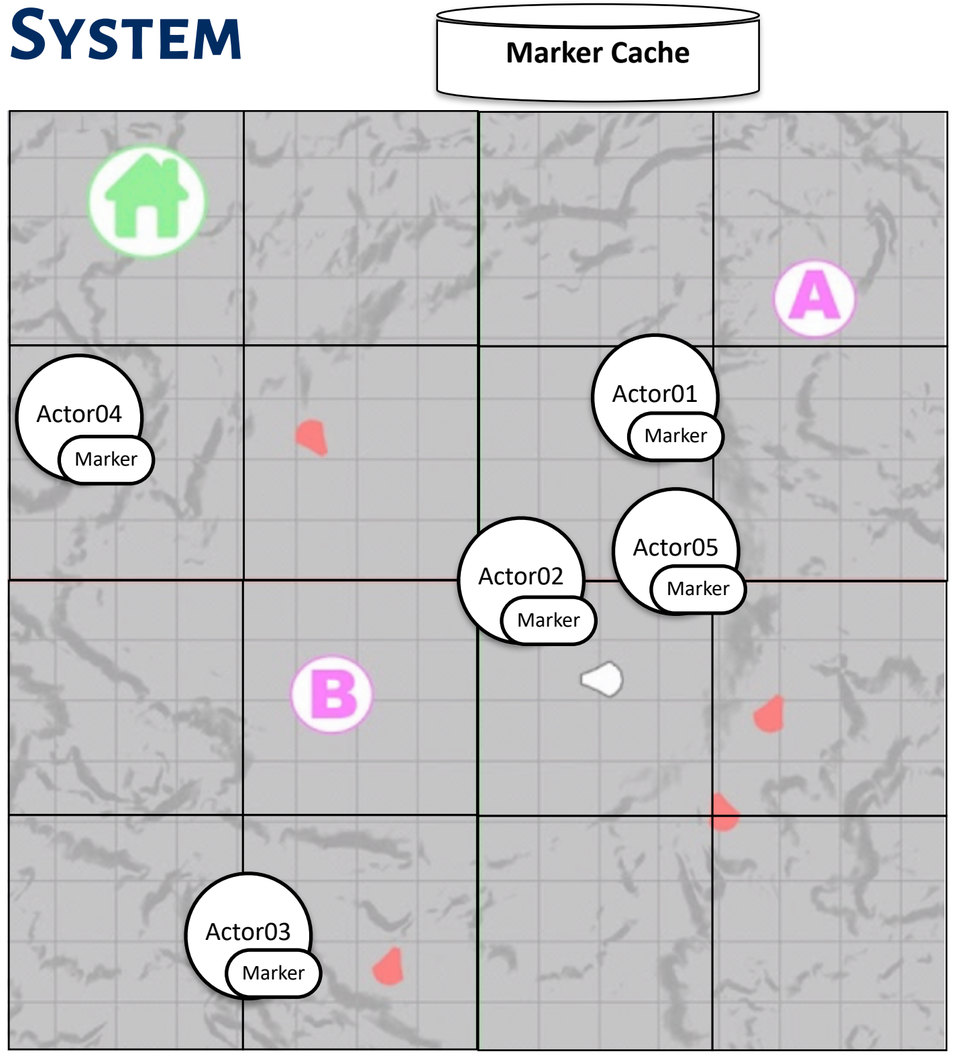




A Marker System

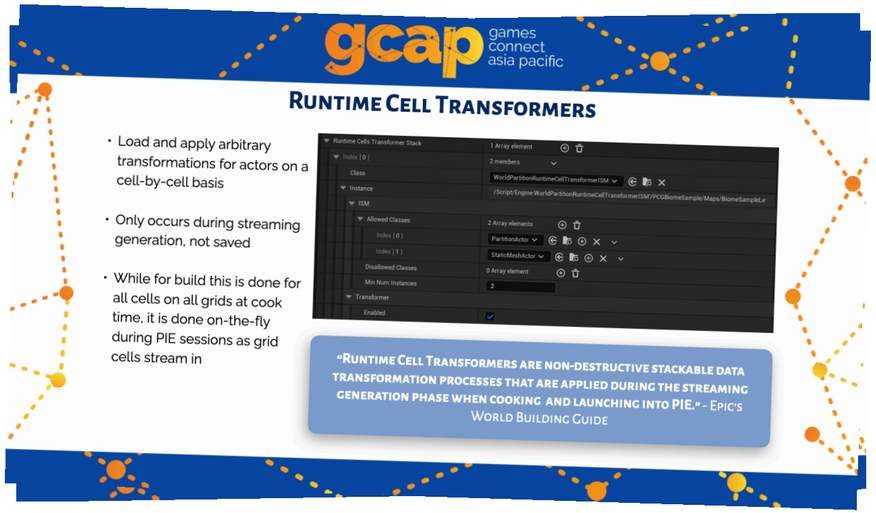
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results

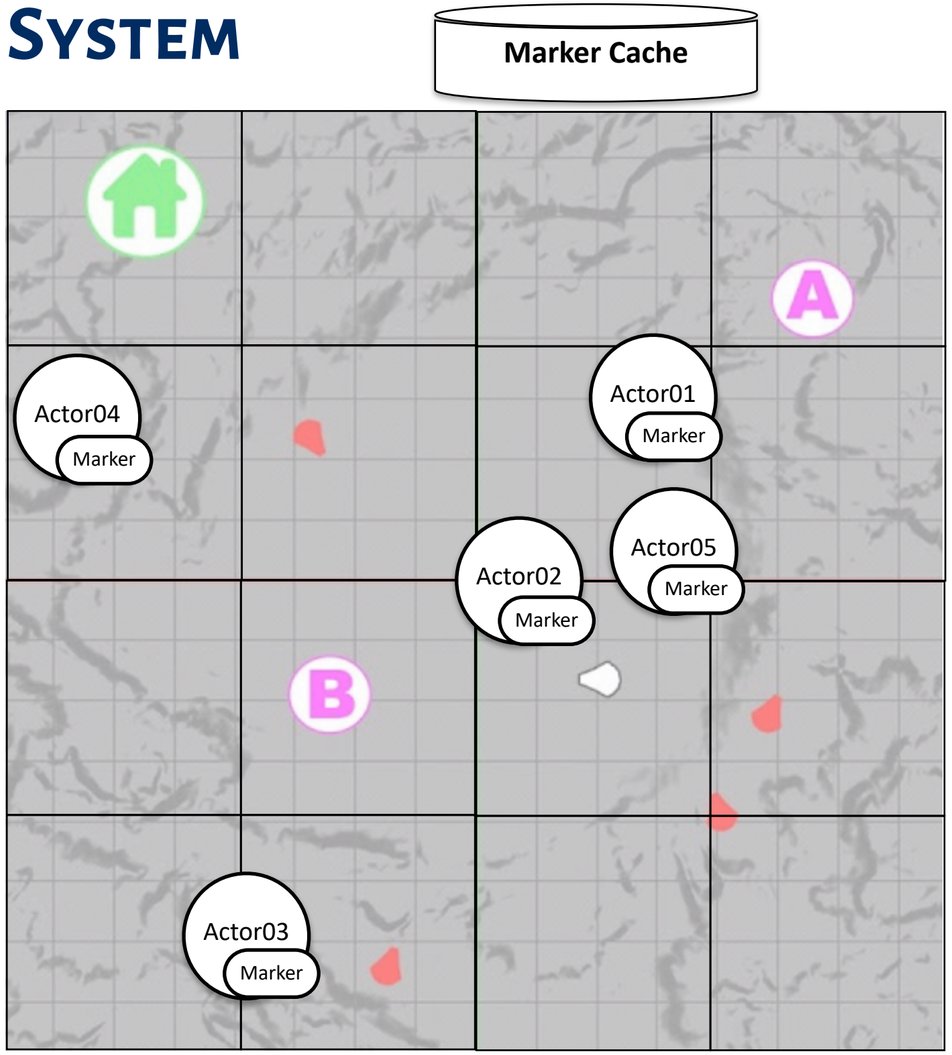




A Marker System

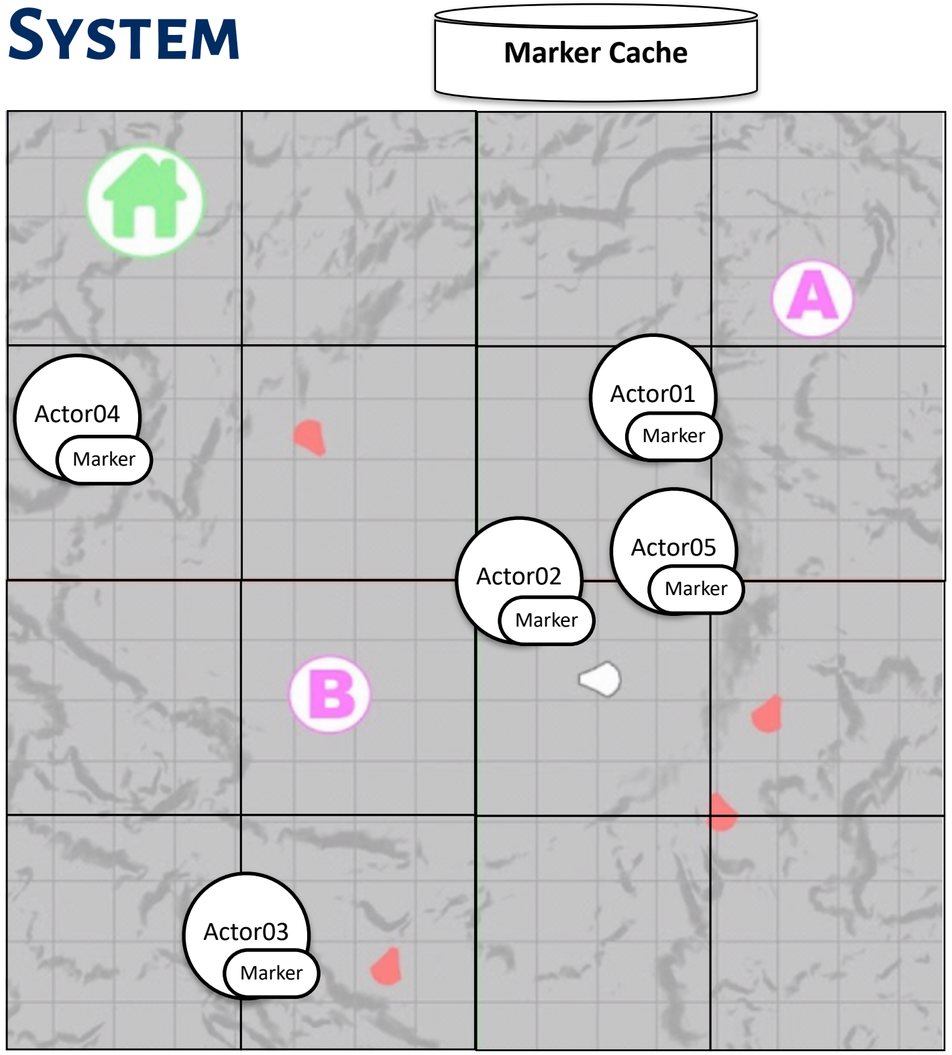
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results





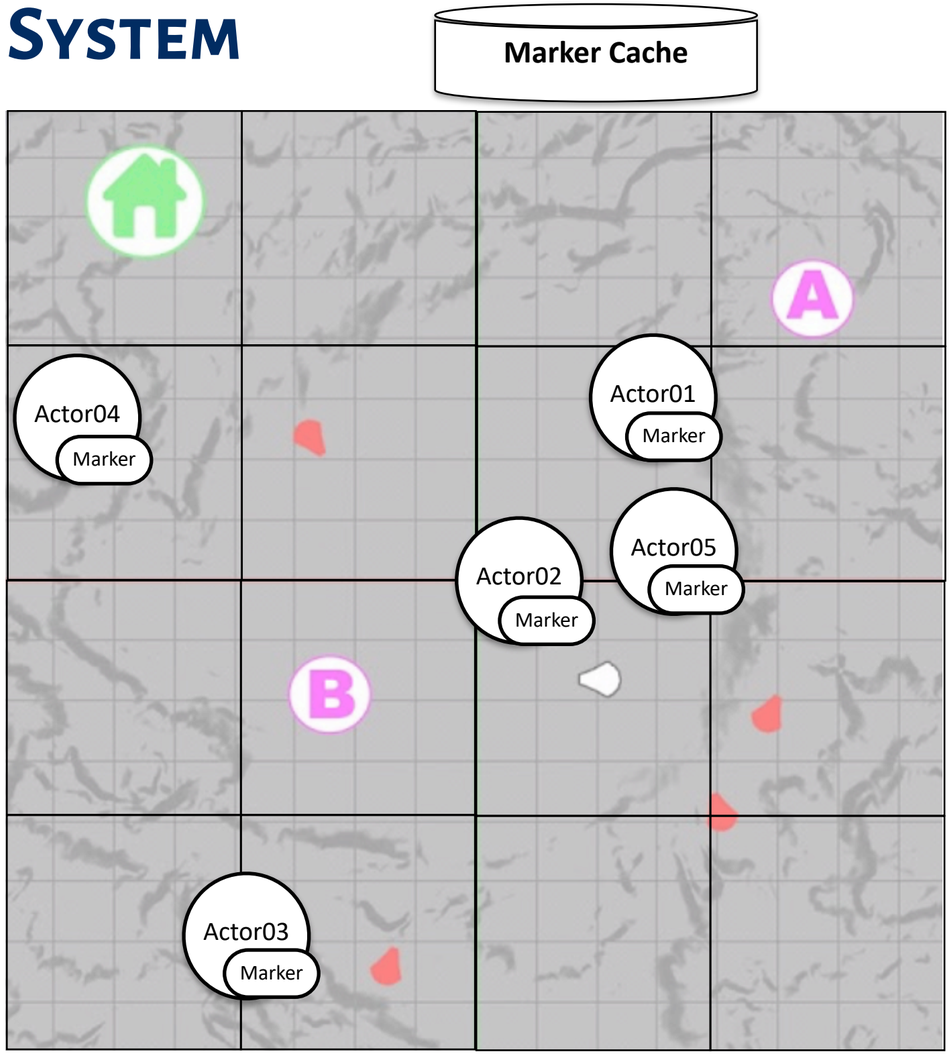
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



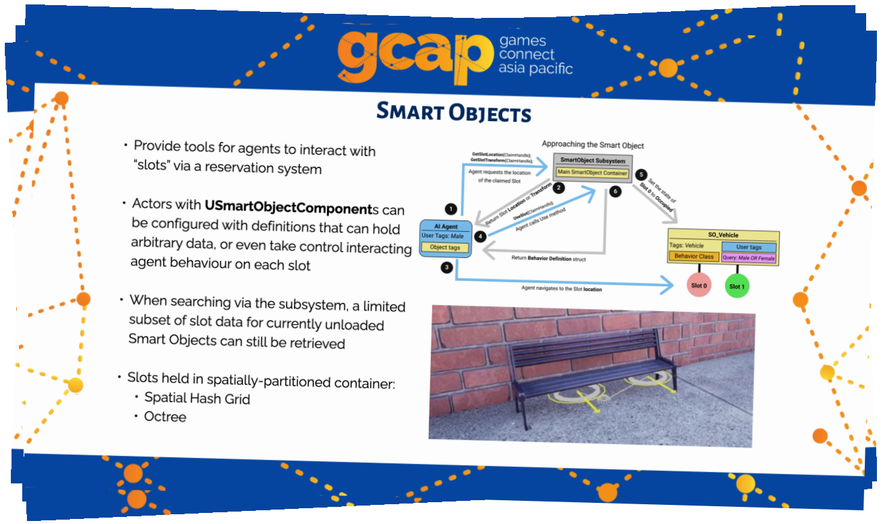
A Marker System

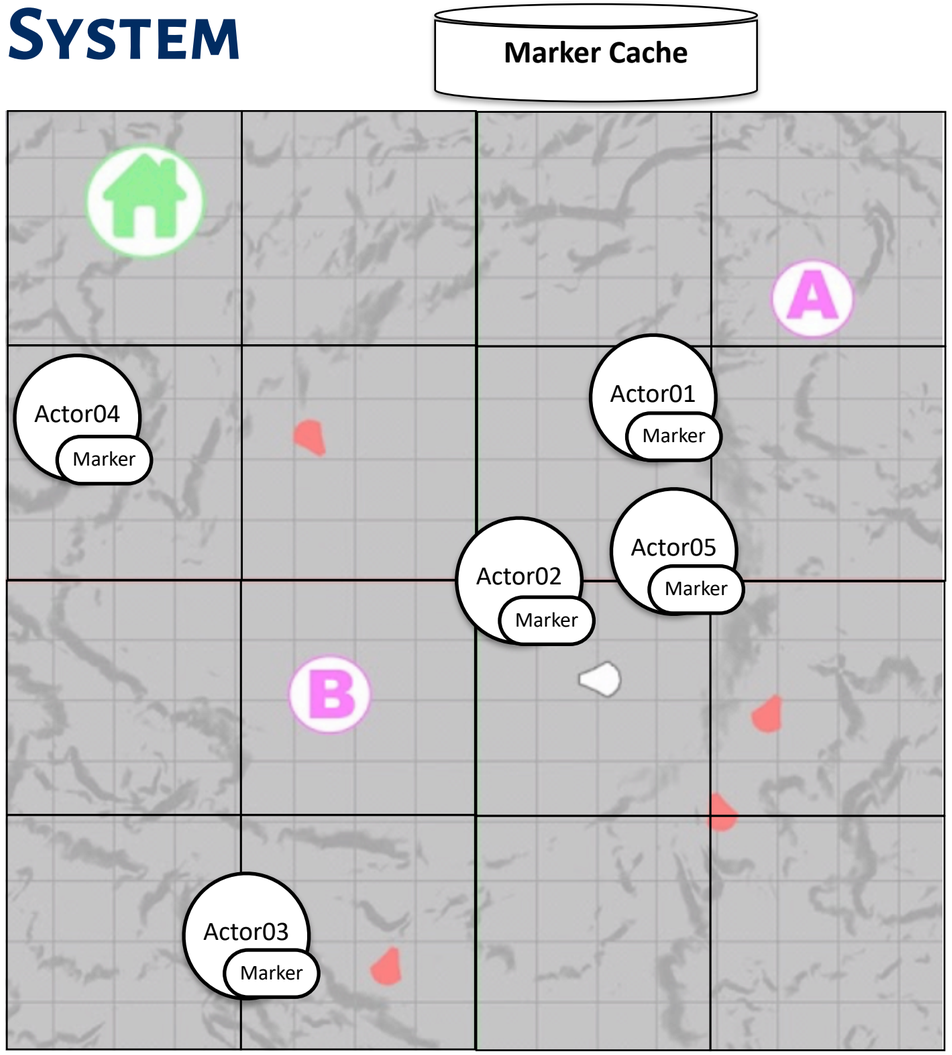
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results



A Marker System

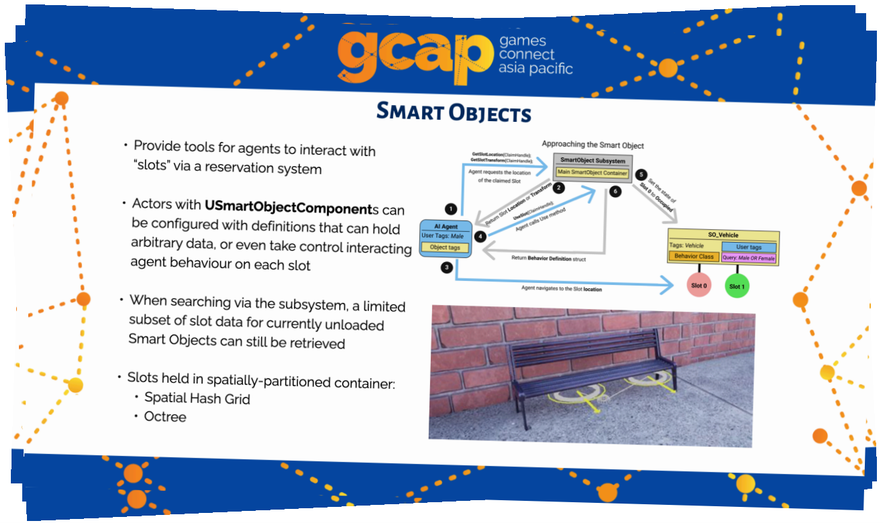
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results

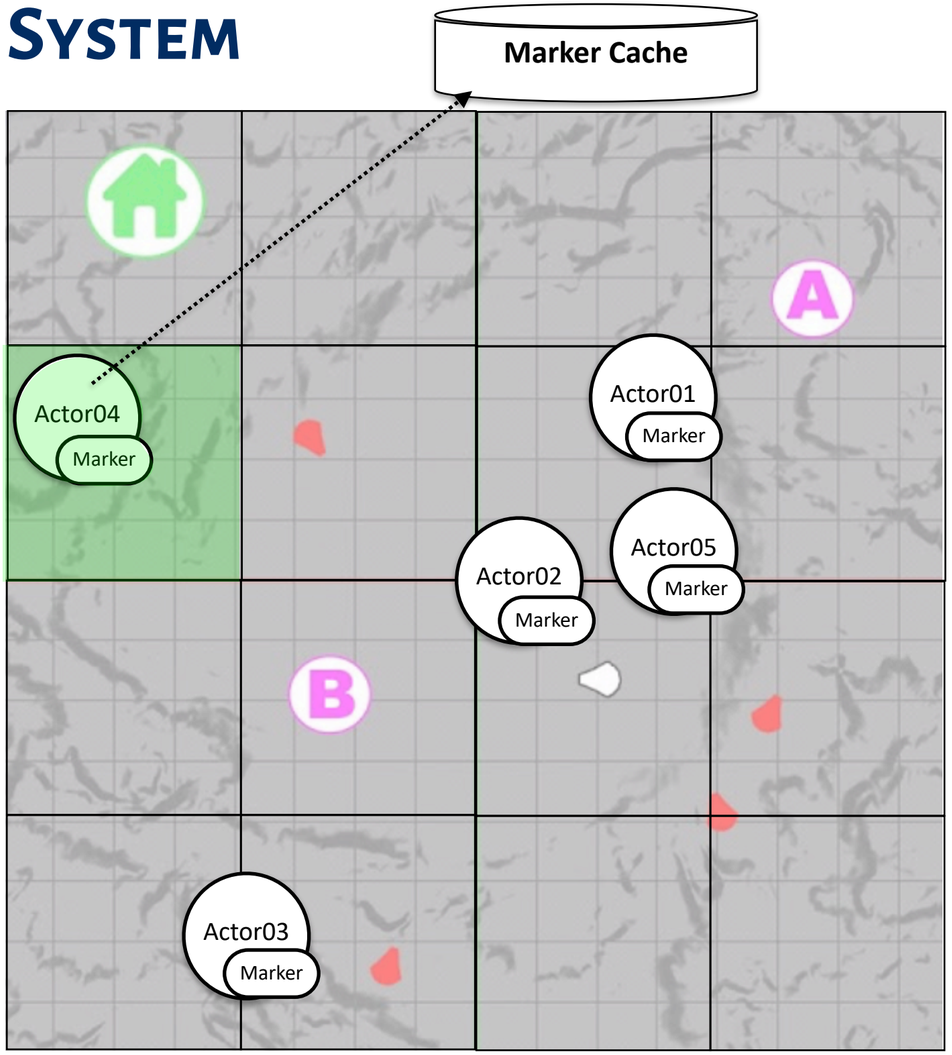




A Marker System

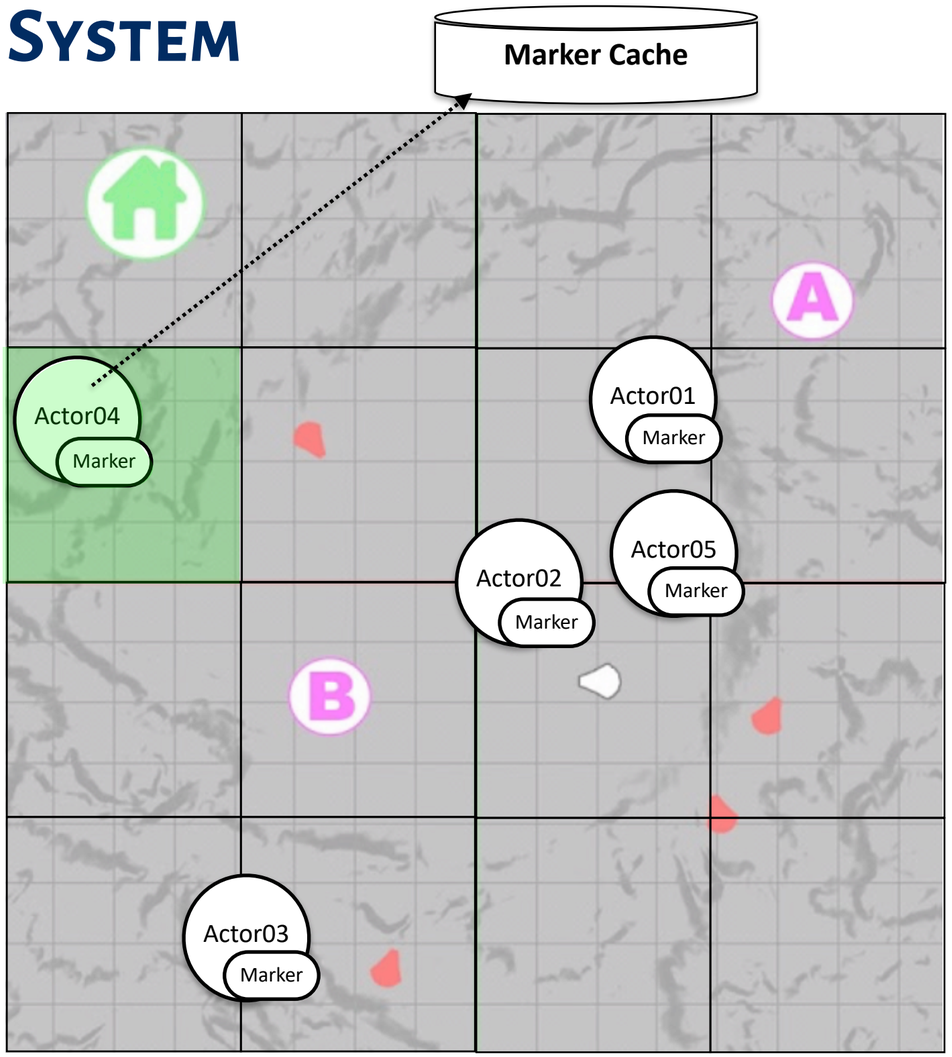
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results





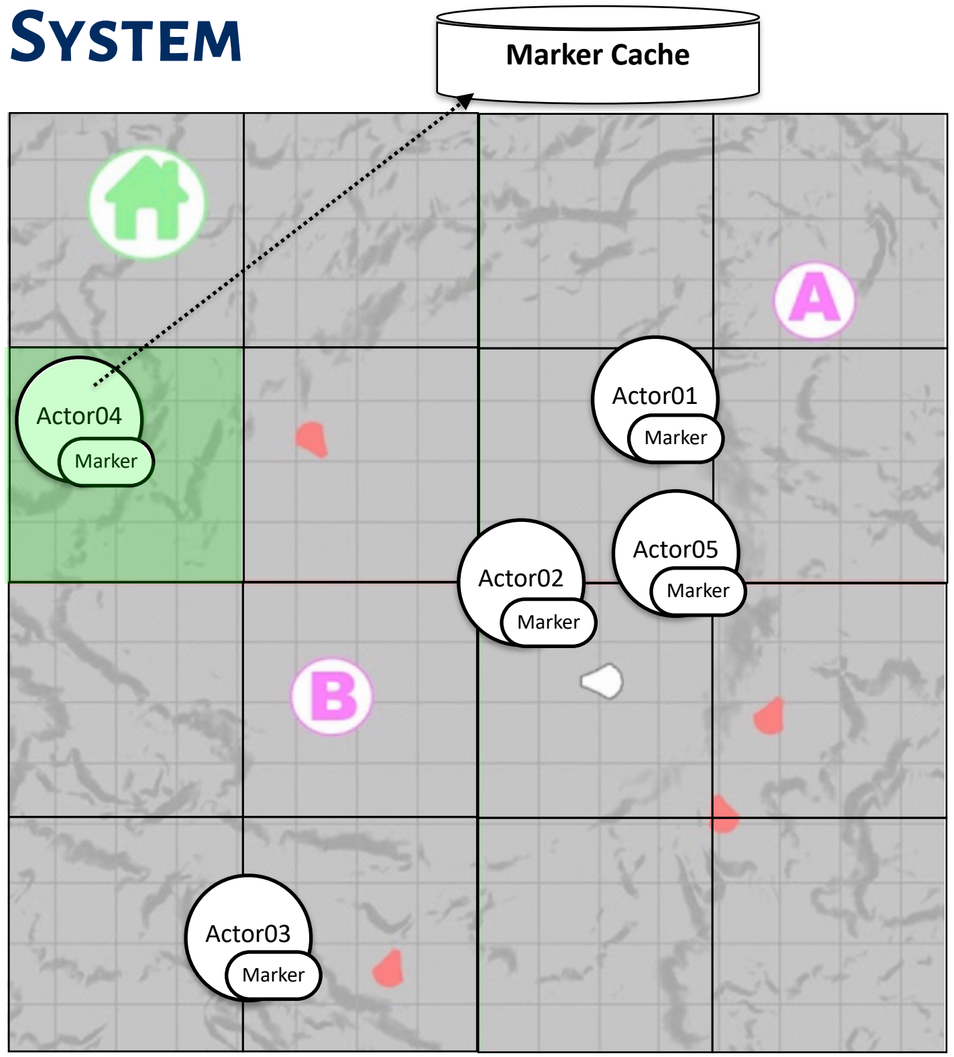
A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results
- Interior/exterior handling leverage grid info
  - Pull runtime grid info on marker data
  - Create virtual mapping of spaces
  - Add marker “portals” as delegates



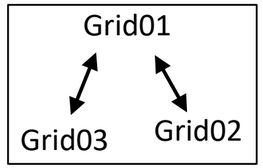
A Marker System

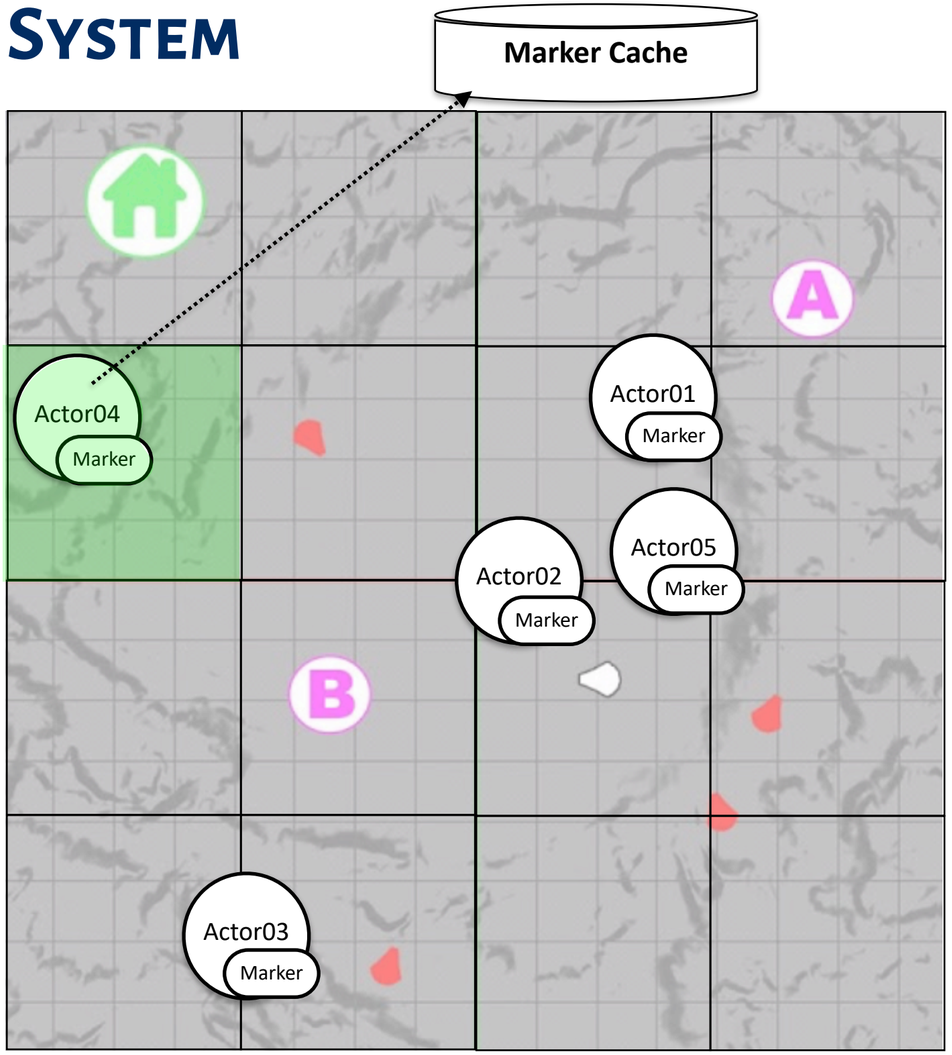
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results
- Interior/exterior handling leverage grid info
  - Pull runtime grid info on marker data
  - Create virtual mapping of spaces
  - Add marker “portals” as delegates



A Marker System

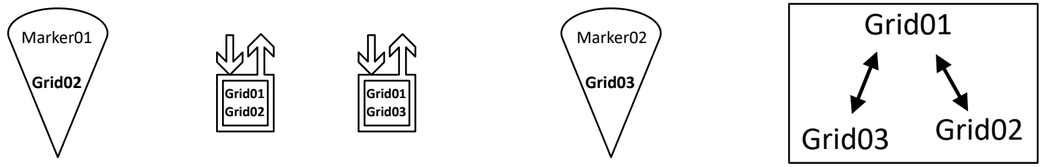
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results
- Interior/exterior handling leverage grid info
  - Pull runtime grid info on marker data
  - Create virtual mapping of spaces
  - Add marker “portals” as delegates

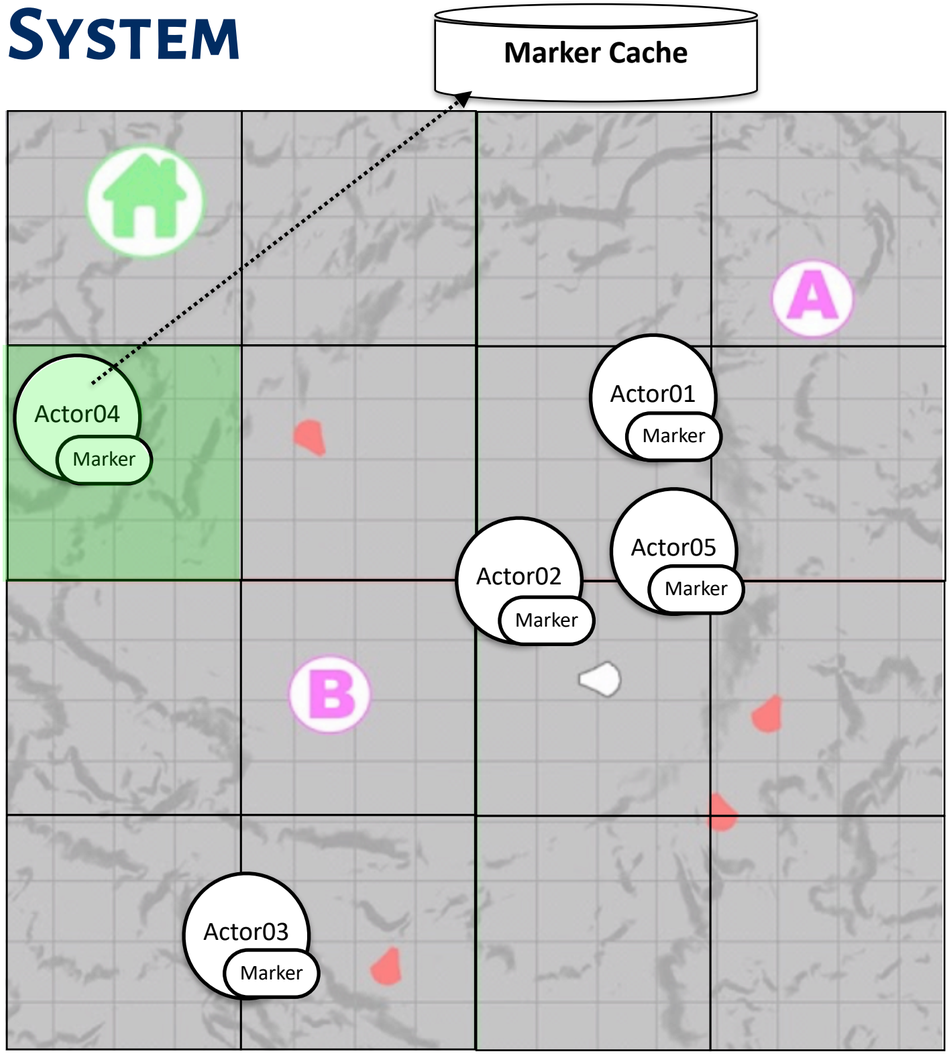




A Marker System

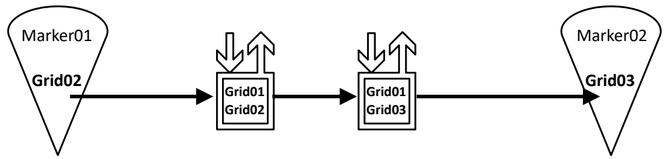
- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results
- Interior/exterior handling leverage grid info
  - Pull runtime grid info on marker data
  - Create virtual mapping of spaces
  - Add marker “portals” as delegates

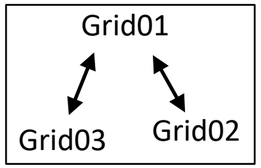


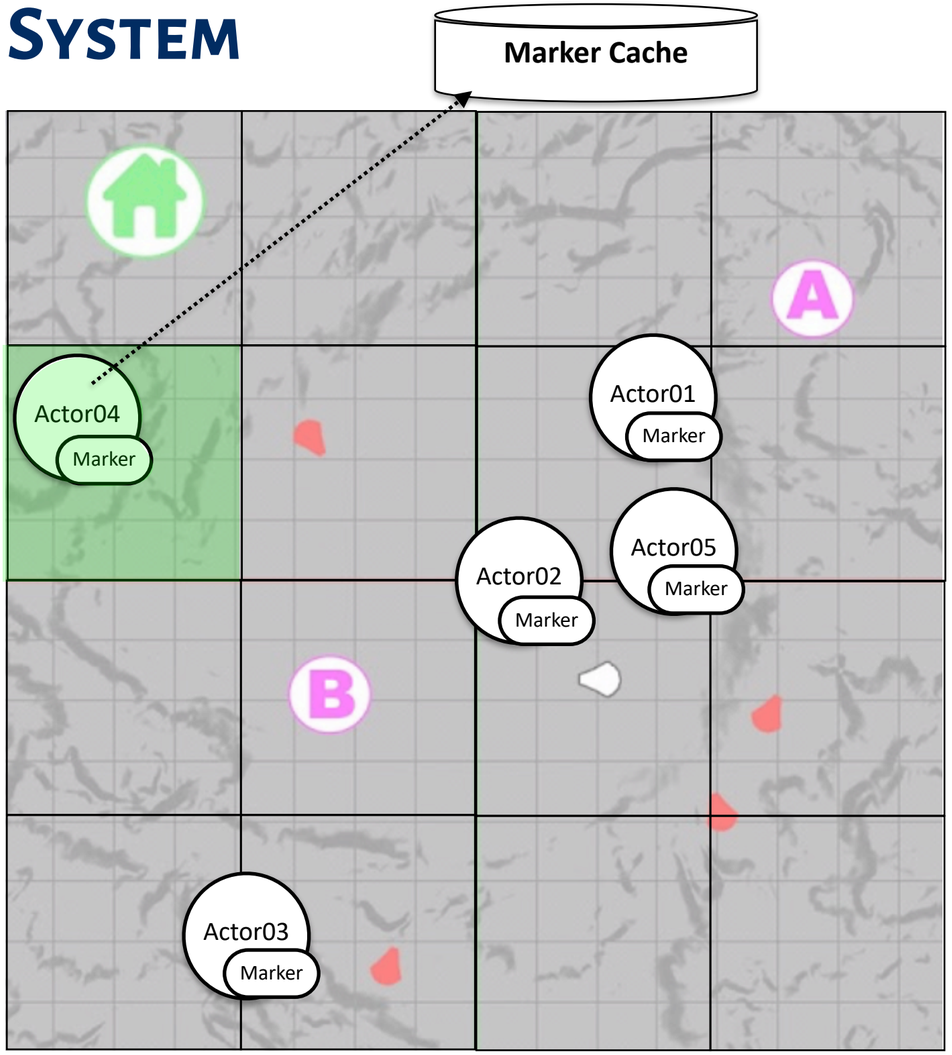


A Marker System

- Keep all marker actors persistently loaded
  - Easy to understand, incredibly wasteful
- Cache markers on world startup with Data Layers
  - Easy in PIE, painful and fiddly to manage
- Cache markers on cook for offline lookup, track live
  - Painful for iteration with PIE, consistent results
- Interior/exterior handling leverage grid info
  - Pull runtime grid info on marker data
  - Create virtual mapping of spaces
  - Add marker “portals” as delegates



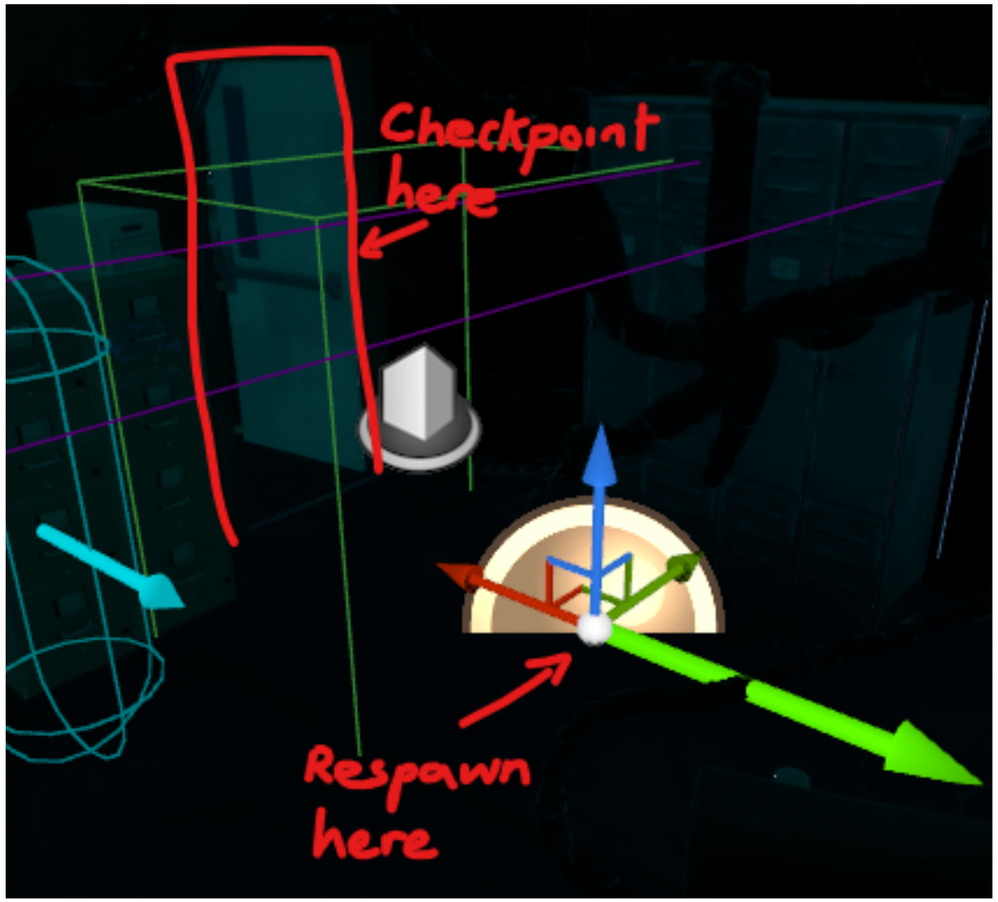




Serialisation Systems For Game State Persistence

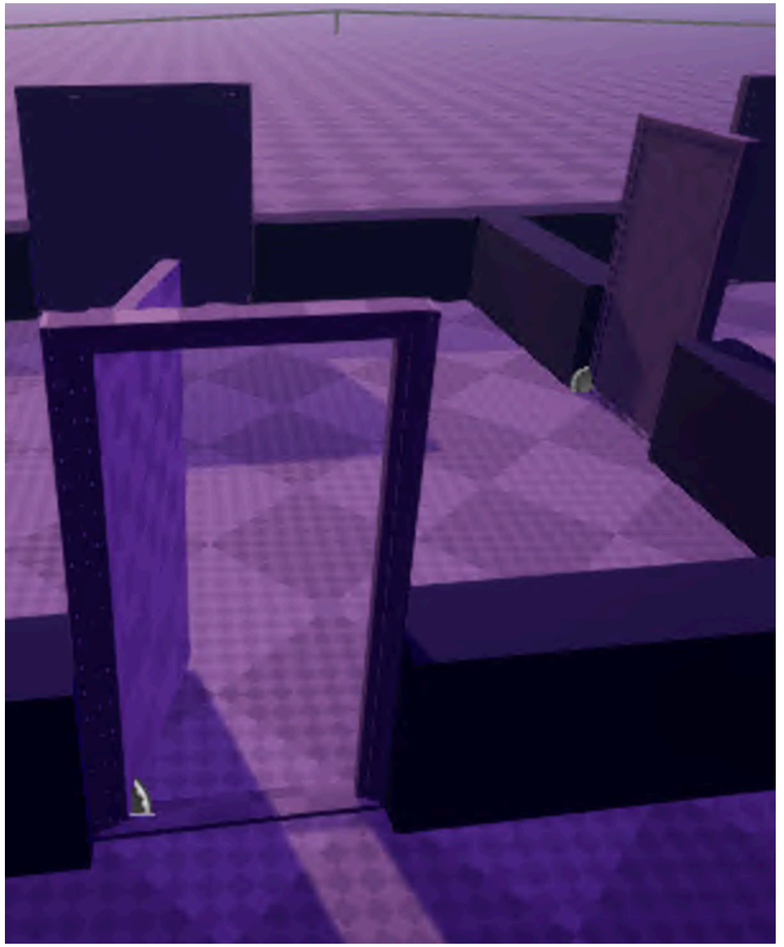
Condensing World & Streaming State Representation

- Author systems to operate on discreet states
  - Design actors to support reload scenarios
- Reduce states into tractable, configurable IDs
  - Avoid tying state data directly to assets
- Some state will be specific to actors in the world
  - Tag to save space and facilitate reset capabilities
- Allow save game data to be manually modified
  - Optional text-based serialisation can be useful for debugging and desk-checking game states
  - This functionality can work very well as a debug tool when allowing for state capture



Capturing Live State & Save Game Concerns

- Actors configure themselves on inference from world state
  - Avoids abusing Data Layers for minor details
- Taking in game state as a snapshot at any point is a relatively safe path assuming all critical triggers and state are serialised
  - Ensure save points and state change process is clear and controlled
- Ensure live capture cannot generate invalid game state
  - Reduce valid states where possible
- Managing world state conflicts is hard to do in an automated way, so good documentation on the effects of all quests over time is valuable

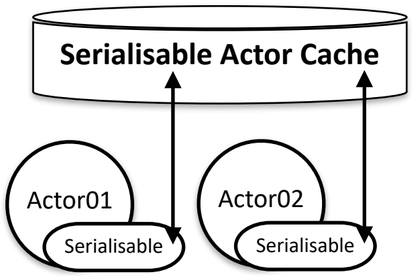


Intermediate State Caching

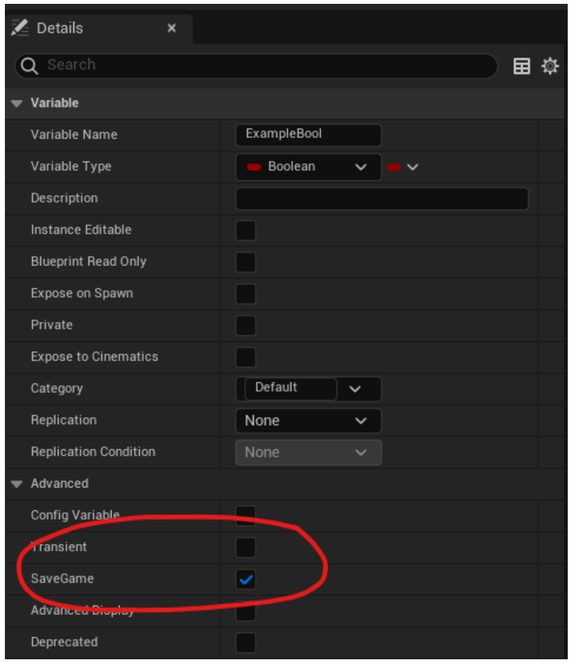
- Actors being streamed in and out need a way to restore critical state
  - It may also be important, depending on your game to be able to modify object states while unloaded

- The **USerialisableActorStateComponent** provides a simplified way defining these states
- Leverages **SerializeScriptProperties** with **SaveGame** - can be serialised out to an archive or variant

- Actors should be written to configure on changes coming from cache
- Unloaded lookup can be done based on the actor's SoftObjectPointer
- Special handling is needed for WorldPartitionSpawners
- Can be handled in batch




```
166
167  UCLASS(Blueprintable, meta = (BlueprintSpawnableComponent))
     1 derived blueprint class
168  class [redacted]UActorStateExampleComponent : public UActorComponent
169  {
170      GENERATED_BODY()
171
172  protected:
173
174      UPROPERTY(BlueprintReadWrite, SaveGame)
175      FVector ExampleVector = FVector::ZeroVector;  Unchanged in assets
176
177  };
178
```



Level Streaming Persistence Plugin

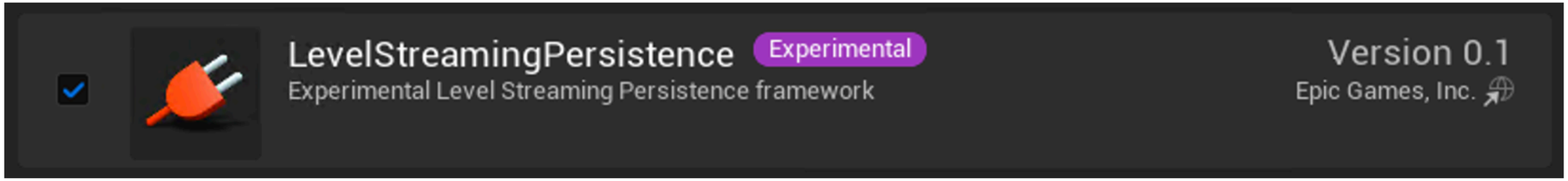
- Yup, Epic has a plugin for that…
- “Level Streaming Persistence” plugin is still experimental as of Unreal 5.6
- All persistent actor properties must be marked up in a config file with their full path
  - Can be brittle to path changes and renames
- Does not handle replacement of actor destroyed state
- Provides ability to remotely modify properties marked “public”

Level Streaming Persistence (Experimental)

JEAN-SEBASTIEN GUAY | Posted on April 2023

Level Streaming Persistence is a plugin that manages saving and restoration of specific marked properties on objects contained within a streamed-out/in Level. These properties are defined within the engine configuration (ini) file.

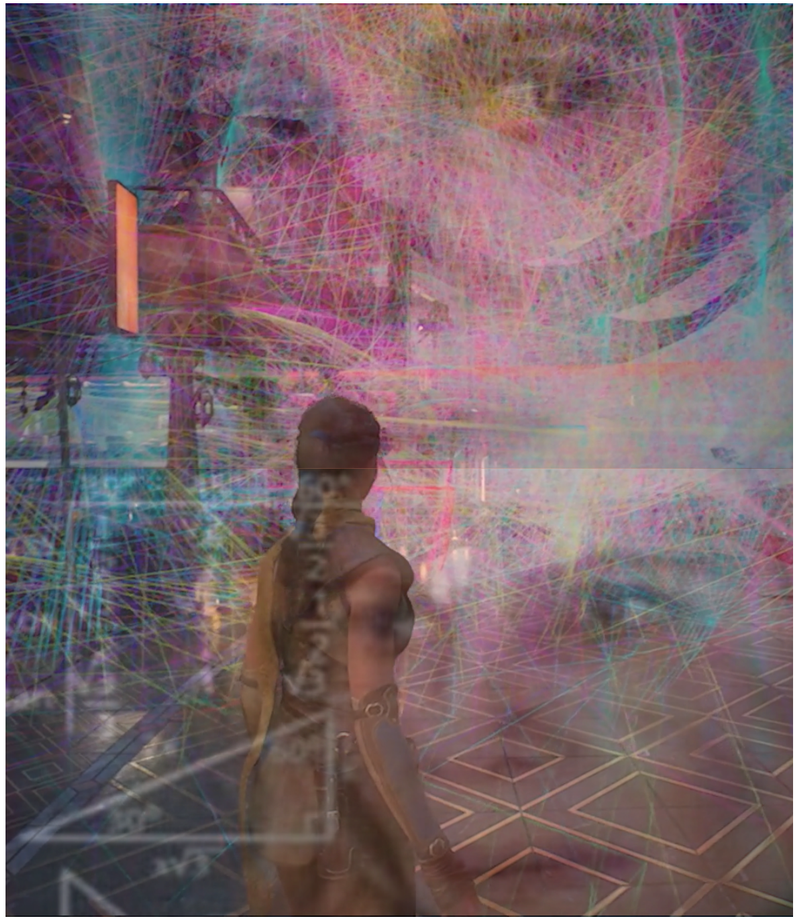
This provides persistence of properties during gameplay from the in-memory snapshot, as well as across sessions, if implemented through the provided API.



Wrap-up, Summary And Q&A

Summing it all up

- World Partition is a very deep bag of tools enabling parallel workflow, level streaming, and actor/geometry optimisation
- Be aware of the tooling and dimensionality you have available, and intentionally plan out its usage ahead of time
- Managing open world state can become quite complex - it's valuable to build in assumptions, and limit possibility space
- Approach all systems and features with a plan for persistence early and evaluate impact of streaming on all functionality



Q&A

Thanks for watching!

Slides Available Here For Download

JonJonDev.com/documents/GCAP25StreamingAndSerialisation.pdf

JONJONDEV.COM

JONJONDEV

@JONJONDEV.COM