STREAM

MANUAL

Table of Contents

Questions or problems?

support@makecodenow.com

Introduction

SECTR STREAM makes it easy to save memory, increase performance, and decrease load times by splitting your scene into multiple chunks and streaming them in and out in realtime. SECTR STREAM is ideal for titles targeting memory limited devices like tablets and smartphones, or games on any platform that want to have large, seamless worlds free of loading screens.

SECTR STREAM includes all of the editor tools and runtime components needed to split your scenes up and stream them at runtime. Games that already have already setup Sectors and Portals with SECTR CORE can be streaming in seconds, and it only takes a few minutes to setup a scene for streaming from scratch. SECTR STREAM even handles streaming "global" objects like lightmaps, nav meshes, and light probes.

Note that async loading is a Pro-only feature, Free users may see "hitching" when loading new sections in SECTR STREAM. These hitches can be avoided with some clever design, and some helpful components are included the package.

All SECTR modules include complete source code, online support, and are fully compatible with Unity, Unity Pro, and Unity Version Control.

Overview

SECTR STREAM works by taking each Sector in the scene and exporting it (along with all of its children) into an individual scene file (called a Chunk). During export, some additional data is added to both the exported scene and the main scene to ensure that the Chunk is loaded correctly. SECTR STREAM provides a convenient GUI for managing this export process, but the code is structured to allow command line exporting as well.

At runtime, each Chunk is managed by a Chunk component. These components are reference counted, which allows Chunk loads to be requested by many different components while guaranteeing that no memory is leaked and nothing is loaded unnecessarily. For Pro users, Chunks are loaded in the background, using Unity's async loading features. For Free users, Chunks are loaded synchronously, which may case a "hitch", depending on the size and complexity of the Chunk being loaded.

While any script object may request that a Chunk be loaded, SECTR STREAM comes with a set of Loader components, designed to make it easy to load Chunks based on common patterns, like entering a trigger volume, opening a door, etc. Users are encouraged to use these components, extend them, or add your own.

One of the tricky aspects of chunking and streaming a Unity scene is handling the "global" level data like lightmaps, light probes, and nav meshes. SECTR STREAM takes a different approach for each of these components, as described in the Global Data section.

SECTR STREAM also includes a solution to another common problem with streaming, namely what to do with global objects that need to be in the scene, but which are in areas that have been unloaded. For more about this, see the Hibernators section.

Setup and Exporting

Preparing a scene for export is easy. It simply requires the creation of one or more Sectors in the scene. For new users, please see the SECTR CORE documentation for

more detail on creating Sectors. Note that for SECTR STREAM, you do not actually have to create Portals unless you want to use some specific Loaders, and even then the Portals do not have to have geometry.

Once you've created some Sectors in your level, you should add some Loaders. Which Loaders are right for you depends entirely on the game you're trying to make, but adding a NeighborLoader to your main player GameObject is a good place to start.

With Sectors and Loaders created, it's time to split your scene into Chunks. The easiest way to do this is through the SECTR STREAM Window, which you can find under Window->SECTR->Stream. Simply press the Export All button, follow the prompts, and wait until it completes. When the export is finished, you'll see that all of your scene geometry is now gone, and each Sector component now has a Chunk component, too. If you look in your Project window, you'll also see that there is a folder with the same name as your scene, and in that folder is a sub-folder called Chunks. That sub-folder contains all of the exported Chunks as individual Unity scene files.

It's important to point out that the export process will only export static Sectors and their children. This behavior is described further in the Global Objects section.

Global Objects

While the bulk of the data in the scene can be exported into Chunks, Unity has some scene data that is considered global to the entire level. Also, as described previously, dynamic Sectors and any GameObjects that are not a child of a Sector will also not be exported. Together, these objects are considered "global" and require some special discussion here.

Global GameObjects

As described above, dynamic (i.e. non-static) Sectors and any GameObjects not parented to a Sector will not be exported into a Chunk. This is necessary because an object can only be exported into a single Chunk, but Members may be in multiple Sectors at once. Non-static sectors are assumed to change at runtime, and are also unsafe to export. Once the game is running, however, both dynamic Sectors and global GameObjects they will still behave with the exported data just as they did before export.

This behavior can actually be used as a design tool. Often, there are some objects in the level that should always be instantiated (i.e. the player, puzzle objects, etc.). As long as these objects do not try to hold direct references to exported objects, then they will function just as they did before the export.

Lightmaps

For games that use them, lightmaps can consume much of the game's total texture memory budget. Unity only has a single, global list of lightmaps, and normally all of them are loaded when the level is loaded. Fortunately, SECTR STREAM includes logic to get around this and actually stream lightmaps.

The lightmap streaming works by computing (at export time) which lightmaps are used by a given Sector. When the Sector is exported, references to those lightmaps are included in the exported data. When the export is complete, any lightmaps that are no longer referenced are removed from the main scene. At runtime, individual lightmap textures are loaded along with the rest of their Chunk, after which SECTR STREAM fixes up the global lightmaps list. The opposite happens when Chunks are

unloaded. The end result is that lightmap textures stream just like everything else.

The only very important limitation to lightmap streaming is that lightmaps must be baked properly. For users of Unity 4.x, you must import all Sectors in the main scene and then bake using Unity's Lightmapping UI. In Unity 5.x, you must have all Sectors exported and then use the Bake Lightmaps button in the SECTR Stream UI. SECTR STREAM tries to ensure that the game does not crash if lightmaps are baked incorrectly, happens, but it may look very strange indeed until the data is re-baked and everything re-exported.

Nav Meshes

In Unity, there is only a single, global nav mesh, which cannot (currently) be split up in any way. SECTR STREAM's solution to this is very simple – generate the Nav Mesh while all Sectors are in the scene, and then leave it alone. It will load along with the main scene and work fine.

Generally, NavMesh data is not a significant amount of total memory, but if this is a concern for you, please email support@makecodenow.com.

Light Probes

Baked light probe data is also stored globally in Unity. While it is possible to split this data up in a manner similar to lightmaps, SECTR STREAM currently does not do so. Like light meshes, simply bake light probe data while everything is in the main scene, and then leave it alone.

Generally, light probes are not a significant amount of total memory, even with a large number of light probes, but if this is a concern for you, please email support@makecodenow.com.

Umbra Occlusion

Umbra's Occlusion data is stored globally, but it is not clear how it should work at all with split up scenes. Because of this Umbra Occlusion is considered **unsupported** by SECTR STREAM.

If you need a dynamic occlusion system that works with SECTR STREAM, please consider SECTR Vis, which is fully compatible with SECTR STREAM.

Loaders

While exporting Chunks is a critical first step towards making a streaming scene, they are not much use until someone actually requests that they be loaded. While any script may request a Chunk load, SECTR STREAM includes a number of different Loader components which implement common loading patterns. This section describes the functioning of each of those components. Note that Loaders do not need to be exclusive. You can have many independent requests to load the same Chunk and SECTR STREAM will do the right thing, loading the Chunk on first request and only unloading when all requests have been released.

Start Loader

This is the simplest of all Loaders. It simply loads whatever Sectors it is in at Start, and then removes itself from the game. Start Loader is meant to be used in combination with Loading Door, to balance out the door's reference counts.

Trigger Loader

This component lets you load a list of Sectors whenever a particular Unity Trigger is activated. These same Sectors will be unloaded whenever the trigger is deactivated (provided no one else is trying to load them).

Neighbor Loader

This component works by loading the current Sector(s) as well as connected (i.e. neighboring) Sectors. The exact definition of what to consider a neighbor Sector is determined by the Max Depth property (for programmers, this is the max depth of a breadth first traversal of the Sector/Portal graph). In practice, a Max Depth of 0 means only load the Sectors that the Neighbor Loader is currently in. A MaxDepth of 1 means also add any Sector connected to a current Sector. A Max Depth of 2 means also load any Sector connected to one of those Sectors. And so on. Note that a MaxDepth greater than 0 requires that your Sectors be connected by Portals, though the exact geometry of the Portals does not matter.

Loading Door

This component is perhaps the most clever of the Loaders. It's designed for games that are laid out like Metroid Prime or Dead Space, where rooms are separated by doors that do not open until the room on the opposite side is loaded, and where that load is triggered by getting close to the door.

Specifically, Loading Door works by combining a Trigger and a reference to a Portal. Whenever the Trigger is activated, the Loading Door loads the Sector on the opposite side of the door's Portal. When the player exits the Trigger, the door unloads a Sector, but which Sector is unloaded depends on whether the player passed through the door or not. The Loading Door also waits for the door to close before unloading anything.

Group Loader

There are occasions where a section of the scene needs to be split into multiple Sectors (perhaps for occlusion culling or game logic) but they need to be loaded as if they were part of a single Sectors. Group Loader takes care of this, by automatically incrementing and decrementing reference counts whenever one of the Sectors in the list is loaded or unloaded.

Region Loader

If you just want to load all of the Sectors within a volume, Region Loader is for you. Region Loder is most commonly used for terrains, where you want to load all terrains grid elements around the player. Region Loader supports a Layer mask which you can use to filter which Sectors are considered by the Region Loader. This is useful if you want to load terrain with the Region Loader, but load other Sectors, like the interior of a building, using other techniques (like a Door Loader).

DIY Loader

While you can use any of the above Loaders, you are not required to use a Loader at all. You are free to directly request Chunks to load and unload in your own script code, PlayMaker routines, or anything else. Just be careful that you always match

calls to SECTR_Chunk.AddReference() and SECTR_Chunk.RemoveReference()).

Hibernators

As described earlier, GameObjects can be left as part of the main scene so that they are never unloaded. However, while these global GameObjects are never unloaded, the Sectors around them may very well be. When this happens, it's often desirable for the global object to "hibernate" – to stop updating – until its part of the scene is loaded again. The Hibernator component is designed to do exactly this.

Specifically, the Hibernator component can disable other components whenever its Sector is unloaded, and re-enable them when the Sector is loaded again. The Hibernator has properties that control which kinds of Components should be affected by hibernation, and whether children of the Hibernator should be affected or not.

For programmers, Hibernator also provides a set of Events that you can subscribe to. These Events will tell you whenever the Hibernator sleeps, wakes up, or is updated while hibernated.

Chunk Proxies

By default, when a Sector is unloaded, it is simply invisible, appearing when loaded and disappearing when unloaded. However, some games need to have a visual representation of the Sector when the Sector itself is unloaded. This "proxy" representation is usually a simplified version of the Sector itself, one that appears when the Sector is unloaded and disappears when the Sector is loaded. SECTR STREAM includes tools and logic within the Chunk component to make creating Proxies simple and easy.

Creating Proxy Meshes

Each Chunk component can only have one Proxy Mesh, though the mesh can have multiple sub-Meshes. This mesh can be any Unity mesh resource, including meshes created by hand. As a convenience, the Chunk component also includes a tool that can create Proxy Meshes for you automatically. Simply expand the Create Proxy Mesh foldout, select the child meshes that you wish to include, and press the Make Proxy Mesh button. This will create a new mesh resource that is a combination of all selected meshes, including combining subsets with identical materials.

Proxy Mesh Memory

It's important to note that while the Proxy Mesh disappears when the Sector is loaded, it is simply hidden, not unloaded. This means that the Proxy Mesh and all textures and materials referenced by it will be in memory as long as the level is loaded. In other words, Proxy Meshes improve visual fidelity but require more memory to do so.

Collaboration

As teams get larger, they often run into an issue where multiple people want to edit the same scene file at the same time, which Unity normally does not allow. With SECTR STREAM, however, it is possible for multiple people to share a (split) scene, provided that they follow a few rules.

Please note that collaboration is considered "beta" and may have issues not discussed here. If you have any great problems or great successes with SECTR STREAM collaboration, please email us at support@makecodenow.com.

Rules

- When editing a Chunk file, care should be taken not to delete or otherwise modify the root node. This node contains important information necessary to stream that Chunk, information which users should not modify.
- Most newly created objects should be parented to the root node or one of its children. This will ensure that they remain part of the Sector during loads and unloads.
- New global objects can be created in the Chunk scenes. However, they will not function correctly until imported into the main scene and re-exported.
- Global Unity data like lightmaps, Nav Meshes, Light Probes, should never be generated within a Chunk scene. Doing so may cause problems at runtime.