Modular Gate System

Hello and welcome, here you can read about the Modular Gate System, what it is, what it can do, and how to use it. As for what it is, it is a pair of actors. The Gate_Section_BP is an actor that allows you to create an individual section of a gate. It has options to set the meshes it will use to create the various edges, corners, doors, railings, and other parts. It is not generally intended to be used on its own though. The second actor is the Gate_Master_BP, this actor can be placed into the world and will handle the spawning of the Gate_Section_BP actors and their options to create a large gate that can be controlled as if it were a single actor. Just set the length, width, starting open/close position, and some other options and it'll build the desired type of gate.

As for what it can do, well the gates that are made can open and close. Each door moving over the other one in sequence until the central area is clear, or covered. This works on small simple setups with just one or two door sections, or on large massive multi segmented gateways that can let in large ships or such. While definitely not the type of door you'll want in all situations, for those that want horizontally sliding doors of a most likely mechanical nature then this system may be of use as it can greatly speed up the creation and implementation of such doors.

For details on how to use this system and all of its options, read on.

Among other things, this pack will also provide you with…

- A gate system that you can plug in your own art assets into too create a gateway that matches your desired look, and with your own float curves adjust the movement.
- Allows for dynamically sized gates to be built, with the movement of door sections adjusted to match the size of the gateway created allowing for easily usable gates of almost any size.
- A large number of options allowing for customization of how the gateway can be built, it's size, or what parts to build (such as only the doors, or to exclude the doors, or to include rails or not, etc...).
- A series of meshes included for demonstration purposes, with textures and materials that can allow for customized coloring and other options.

NOTICE: If you spawn into the demo map with a non floating default character, swap the level game mode to "Game_Mode" default type to use the default player pawn that can freely fly around. Depending on the default spawn character in your project, this may be needed. Update incoming in the meantime. Sorry for the minor inconvenience to some people.

Additional Tutorials and Demo vids for other Packages can be found here:
https://www.youtube.com/playlist?list=PLAgvMMfMQH2O1-YfsKXSwcFK2bvVg7UrP

How To Use - Gate_Master_BP

The main actor you will be interacting with is the Gate_Master_BP from this package. The Gate_Master_BP works by spawning a series of the Gate_Section_BPs and giving them the appropriate values. You can use the Gate_Section_BP on it's own as well if you only need one section, however as most of its options are the same so we will focus on the Gate_Master_BP actor.

At its core this system is built to take the art assets that you have (static meshes) and automatically position them to make the desired type of gate, shaft, hatch, or door as you may want to call it. The most important factor is the size of the door, horizontally, or how wide it is. One door sections width is the effective "unit" size that other art assets should be made to match. This doesn't mean everything needs to be rigidly confined to a blocky form or into those exact conditions though, just that they will be placed using that rough grid. If you want parts to poke out from one section or to overlap with another, you can. As long as it in the end looks like you want that's all that matters, as an example you could reference the rounded edge type of gate meshes included in this package. The corners in that case poke outside of their own "unit" in order to provide a smooth rounded corner on the inside gap where the door sections go. The rear parts also don't go all the way back, so that if you want to place these in a smaller level in a floor or such they won't get in the way as much. Now, let's move on and discus how to actually use the actor.

First, in order to use the Gate_Master_BP you'll want to drag it into your levels viewport, and place it so it's pivot point is at the desired location for a corner of the actor. Now you can look at its details panel with it selected, and customize its options. The most common ones to touch are Length and Width. Length determines how "long" the gate is and width how "wide" it will be. This counts the inside sections of the gate, or how many "door" segment would be needed to fill it, so 1x1 will still be 3x3 units in size to allow for its corners and side sections to surround the door. The door sections will slide out along the X, or forward axis of the actor. So upping length will have a long series of doors sliding over each other, while upping width might have a single door section sliding in/out that is just really wide.

There are a variety of options, like setting the initial state of the door and rails if present, limiting the actor to only building certain parts of the gate like the sides, or only the doors themselves, what meshes to pick from for different parts, and what type of movement to have.

In order to choose the meshes, you'll want to go to the Construction category and look at the mesh arrays. Here you can add or change meshes that are being used for specific parts of the actor. If you have multiple entries in an array, it will grab one at random, useful for making a more varied appearance if you have the art assets for it. I'll discuss the specific meshes in more detail later.

Also in the Construction category is the curves. You can assign a curve to the doors horizontal movement, the doors vertical movement, and the rails vertical movement. These curves should have values of -1 to 0 in most cases, and you should be sure to add a point on the curve at the desired end time. Door movements will use the horizontal curves total length to determine how long each door sections movement takes. To create your own curves, right click

in the content browser and under the miscellaneous section, you'll find the curve option (second from the top currently). After this a window will pop up, select the “float curve” option here. Open a the curve, and then create points along it with Shift+Clicks. Time is left to right, value is top to bottom. At zero time, you’ll in most cases want to use a value of 0 to reference the default position, and then adjust it to -1 with a point after it. Then, if necessary add another point at the desired end duration of the movement. You can also make the curves smoother by selecting them and pressing 1 to get smoothed tangents. If you create a curve that starts at 0 time and 0 value, and then moves to 1 time and -1 value, then use that as a horizontal door motion then the door would as soon as it is told to start opening immediately start moving towards its edge section, and after 1 second it would have moved back 1 unit, and the next door section if present would begin moving. In my case I set the first 0 value point a bit after 0 seconds so that the vertical movement curve will have had time to first slide the door section down, though this is not necessary depending on your goal.

With these and some other fairly self explanatory options you’ll have effectively made your gate, or door, as you want it. So now how do you actually use it? In order to use it, you’ll want to reference the Gate_Master_BP and (casting to it if necessary) use one of a variety of events. There are Open_Gate, Stop_Gate, and Close_Gate events for the gate. These will open, close, or stop motion with all the doors in that actor.

There are also similar options for rails, Raise_Rails, Stop_Rails, and Lower_Rails. In the demo map I use the level blueprint to open, stop, and close all doors with 1, 2, and 3 keys.

For certain types of custom behavior, it also might be best to build it into your gate actor itself, in this case it’s best to create a child actor by right clicking on the Gate_Master_BP and selecting Create Child Blueprint Class. This child can then have some code build into it that can perform the desired actions, or simply be given some different default values for meshes or curves so you can when building in the editor more easily just grab a different actor from the content browser that uses the desired meshes or curves without needing to manually change them if you are regularly using a few different varieties.

In the project you’ll find two childs of the master gate class, one that uses rounded edge section meshes, making a different looking type of gate quickly and easily, or another that had some nodes built in to change the emissive values of its rails so they would glow whenever they are raised.

Now let’s do a quick overview of the different types of meshes you’ll be able to use, split up by category.

Upper Edge meshes make up the border of your gateway. You’ll want side and corner types, though in some cases you might just use the same mesh for both.

Gap meshes are meshes that fill the vertical gap under the upper edge meshes that is created to allow for the doors to have space to slide into. You’ll have options for side gaps (where the doors slide back into), corner gaps, and width side gaps, (for the latter two the doors won’t necessarily be sliding back into them, but alongside them).A gap mesh should be as tall as a door is (or at least the door height value you enter into the options). You also may want to

make X3, X5, and X10 tall meshes, that is to say a single mesh that would be able to take the place of 3, 5, or 10 single meshes of that gap type. Gaps can on larger gateways become the most numerous mesh, adding the most to the total number of components and border checks for visibility. To help with this you can make larger single meshes to that where you can drastically cut down on the number of these mesh components that must be created to fill the "gaps" needed for the door sections to return to/move along.

Lower Wall meshes, these are like a filler that can pop up below the the door and gap sections. There are options for side and corner types.

Rail meshes. While called rails it's really just any mesh that you might want to be raised and lowered vertically. In my case this was a safety railing to raise around the outside edge of gateways going into the ground or such, but use it as you want. There are options for side and corner types.

To make change the look of the meshes materials, go to the Materials folder, and then Material Instances. In here you'll find the materials that the various meshes reference, you can open these up and then look through their options. You'll find options for roughness, color, emissives, masks, normal multiplier, and many other things, number 1, 2, and 3. This is in reference to the color channel from the used mask. It's a somewhat simple approach but it works to a decent extent and gives all the various parts a common series of options to poke at or reference.

If you want to create new material instances then you can either copy and alter an existing one, or right click on the Master Material and select Create Material Instance. Then to use new materials you'll want to copy the relevant meshes and apply the new material to them, and then in the Gate Systems options pick that new mesh that has the new material applied to it.

Also, while very useful for many situations, there are some limits to keep in mind with this tool. One of those is, for example, matters of size. If you want to make a massive gateway that is built from hundreds or more Gate_Sections, and those are all built with dozens of individual meshes and components… well, as you can imagine that can result in a lot of individual meshes or components to keep track of. If you want to make a larger system like this, it may be a good idea to make some larger meshes and use larger unit sizes for these gates, or to disable the edge sections if you can just drop in a single larger mesh with tile textures to fill the role. And if you do just want lots of small individual door sections (because it looks cool, I know that's why I like building large gates with this system), then it might be worth making the borders with larger purpose build static meshes in your level, and adjusting the options so this system just handles the doors and their movement. If your gateways are on the large side but not using a hundred plus individual Gate_Sections, then you can usually get by just by having the various X sizes for the gap meshes. For example, say you have a gateway that's 20 units long and 1 wide, that means there are 10 doors per side, so 10 levels or gaps per actor, and 46 Gate_Section actors in total. Using just 1x meshes, that means 460 meshes and components are added in total just for the gaps, when if just using upper side, and one lower wall otherwise

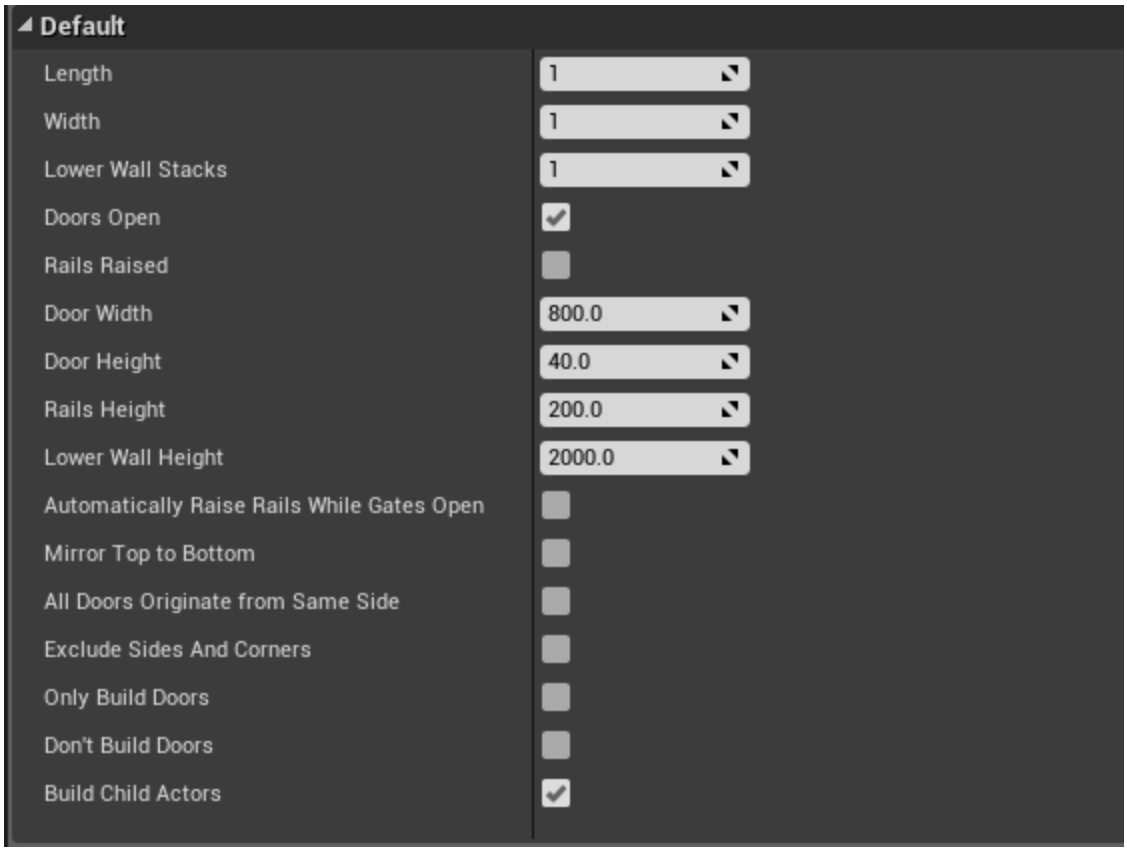
that's just 2 components for all but two of the actors that also had the 10 doors each. So that 460 is a pretty big number. For this reason, if you had the 10x sized mesht that 460 would be cut down to just 46, lowering the total number of meshes/components in the actors around 70%. In most cases, making sure you have the various sized gap meshes is one of the first things to do for optimizing larger gateways, though if you only user smaller gateways then you may not even need to bother making them, as quick and simple as they are to make (just being vertically stacked copies of the original in most cases).

Of course everyone's specific needs do vary, but if you are making a game and trying to make it performant, then while this system works fine on the smaller end for fast and modular gateways, in the larger end it might do to its building block nature have its limits, and as a result save less time by not being able to be used to build as much of the gateway. As much as I love a massive gateway, that's fifty units wide, if when mixed in with the rest of the levels assets it slows down gameplay then it will warrant the extra time. You can also cut down on the number of door sections for really wide gateways by just using a single width unit, but using a door mesh that is a much wider than the unit size, at least enough to cover the desired area.

Hopefully these hassles or special case considerations won't be needed often, but it's still worth covering it here I figure in case you end up in those situations, so you know how to handle them better if you do deal with them, and if you expect to be in those situations before buying this then to know what extra work may be required for performant gameplay in the end. But again, for most cases this isn't of much concern with smaller gateways.

Gate_Master_BP Options

Default



Length: How many sections long the doors hole will be. You will have half of this number (Rounded up) in layers of doors to cover the hole.

Width: How many sections wide the doors hole will be.

Lower wall stacks: Controls the number of lower wall meshes spawned, in vertical stacks.

Doors Open: Will be true if the doors are completely open (can be set before play to have the gates start open or closed).

Rails Raised: True if the rails are currently raised (before play, this can control the initial rails position).

Door Width: The width of a door section, as well as the width fo the whole actor as it will use to determine how to space them. In most cases the edge, gap, lower wall, and other meshes will also share this width.

Door Height: How tall or thick a door mesh is, this height should be shared by the height of the Gap meshes as well.

Rails Height: The height of the rails mesh, or how much it should move up when raised.

Lower Wall Height: The height of the Lower Wall meshes.

Automatically Raise Rails While Gates Open: If true, then the rails will be raised before the doors can open, and the rails will be lowered after the doors are closed automatically.

Mirror Top To Bottom: If true then the upper edge will be mirrored down below the gap areas. In most cases set Lower Wall Stacks to 0 when using this.

All Doors Originate From Same Side: If true then all doors will originate from the same side of the gateway, and move towards its frontal axis, rather than coming together in the middle.

First Door Ignores Vertical Movement: If true then the first door section won't move vertically, and will maintain a -1 height unit value, while those below/after it can still move vertically.

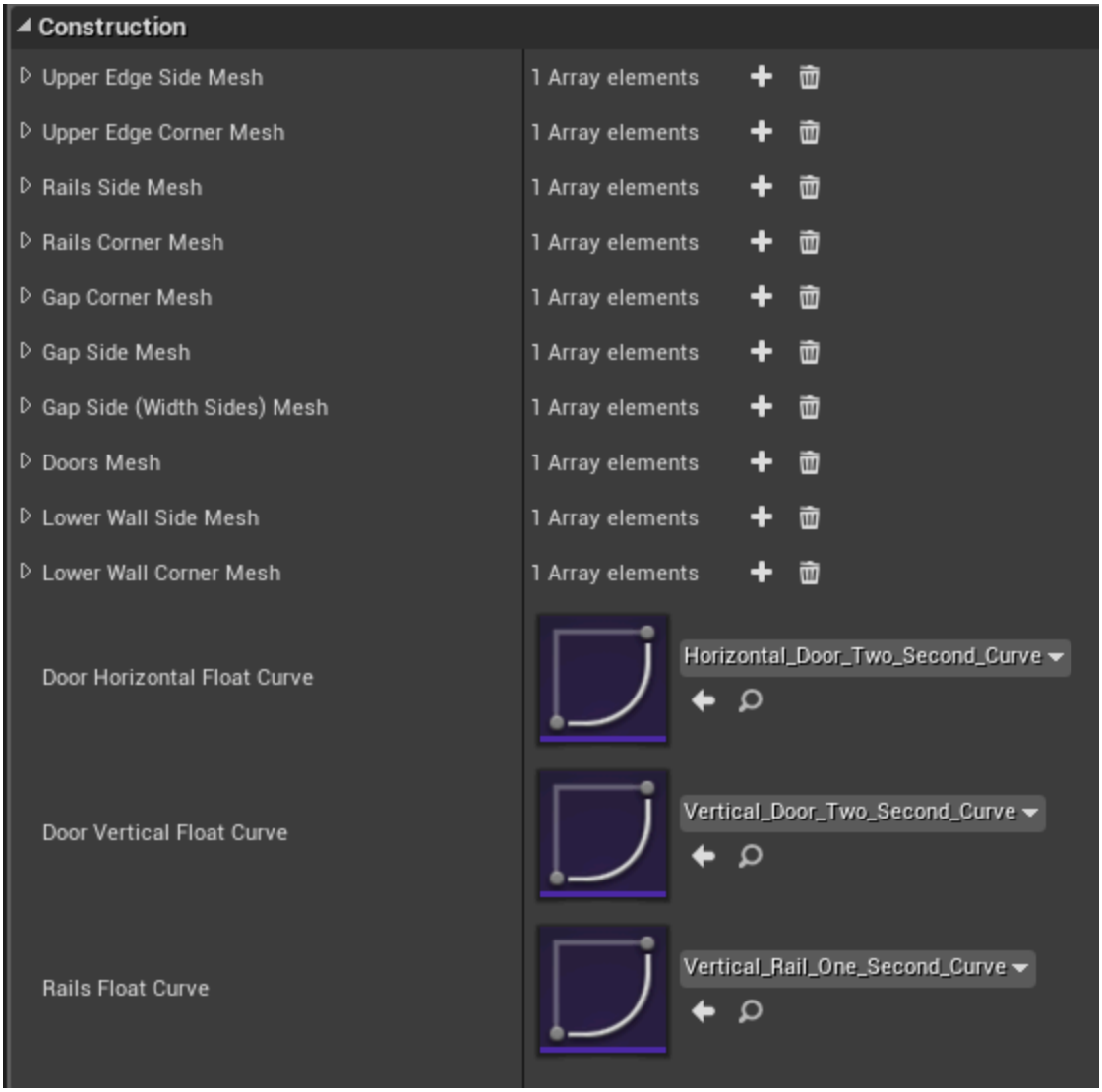
Exclude Sides and Corners: If true then no Gate_Sections_BPs will be created for the length sides or the corners.

Only Build Doors: If true then the child Gate Sections will only build doors, no edges, walls, rails, or anything else. It is highly advised that you also enable "excludes sides and corners" while using this option (seriously, they will be just empty actors those sides and corners if you don’t also tick that one).

Don’t Build Doors: If true this Gate Section will not build any doors, only the edge parts.

Build Child Actors: If set to false then child actors wont be built, while you'll want to make sure this is true before starting the game, if you find it slow to move this actor around, then you can disable this variable to allow for quicker movement in the viewport, and then re enable it afterwards.

Construction



Upper Edge Side Mesh: The possible meshes used for the upper edge sections surrounding the gate on the sides.

Upper Edge Corner Mesh: The possible meshes used for the upper edge sections surrounding the gate on the corners.

Rails Side Mesh: Possible meshes used on the rails on the sides of the gate.

Rails Corner Mesh: Possible meshes used on the rails on the corners of the gate.

Gap Corner Mesh: The possible meshes for the gaps areas needed where doors might slide into (for corners).

Gap Side Mesh: The possible meshes for the gaps areas needed where doors might slide into (for sides with doors)).

Gap Side (Width Sides) Mesh: The possible meshes for the gaps areas needed where doors might slide into (for sides without doors).

Doors Mesh: Possible meshes for door sections.

Lower Wall Side Mesh: Possible meshes for the lower side wall sections.

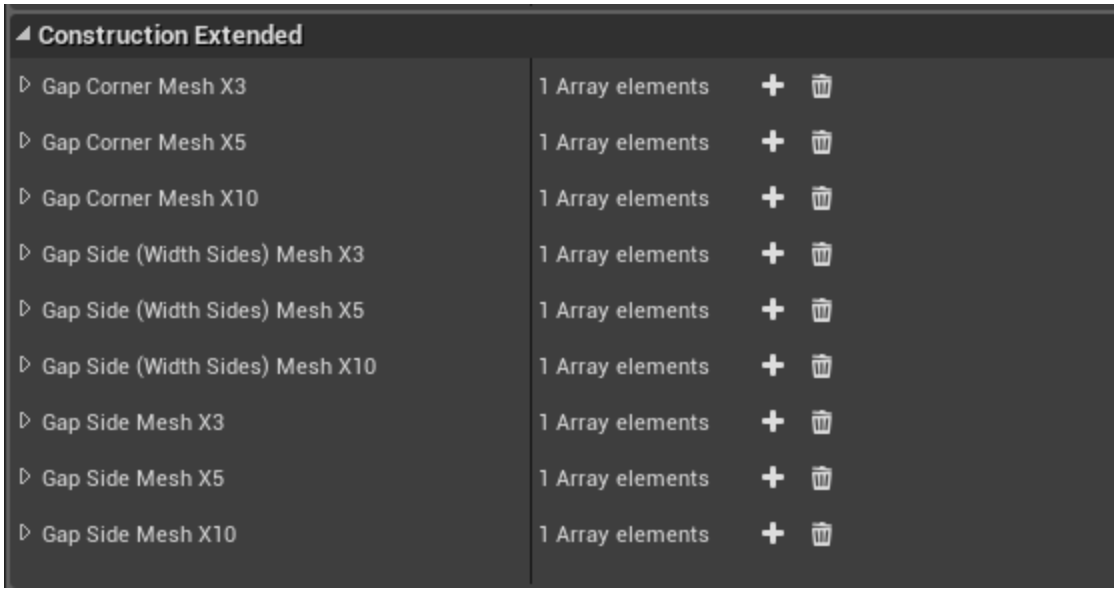
Lower Wall Corner Mesh: Possible meshes for the lower corner wall sections.

Door Horizontal Float Curve: This curve controls how the doors will move horizontally, and over how much time.

Door Vertical Float Curve: This curve controls how the doors will move vertically.

Rails Float Curve: This curve controls how the rails will move vertically, and over how much time.

Construction Extended



*These arrays should contain taller versions of the otherwise used gap meshes, taller by the given multiple value. These meshes will, if present, be used in place of the smaller meshes if there is space for them, lowering total numbers of components/meshes created with larger gateways.

Gap Corner Mesh X3: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Corner Mesh X5: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Corner Mesh X10: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Side (Width Sides) Mesh X3: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Side (Width Sides) Mesh X5: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Side (Width Sides) Mesh X10: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Side Mesh X3: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Side Mesh X5:A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Gap Side Mesh X10: A version of the original mesh that should be X times taller, allowing a single mesh to fill in for multiple smaller ones.

Information

*These are mostly meant for reference by the player, not to be modified or changed.

Levels: How many vertical gap levels there should be, this is the same number as you'll have doors.

Corners: An array containing the Gate_Section_BPs that make up the corners of this group.

Length Edges: An array containing the Gate_Section_BPs that make up the length edges of this group.

Width Edges: An array containing the Gate_Section_BPs that make up the width edges of this group.

All Edge Sections: An array of all the child actor Gate_Section_BPs that make up this Gate_Master actor.

Auto Lower Rails Timer: When Auto Raise Rails is enabled, this timer will count down till the gates fully close.

Things to Keep In Mind

If you ever have any issues, bugs, questions about how something works, or even possibly requests for additional features then feel free to contact me at Pineconedemon@gmail.com and I will try to get back to you as quickly as I can. You can also find more of my stuff on the marketplace through my username, Sean Dachtler and display name Terricon4. And I am for the record always interested in seeing what people might be making with my stuff, so feel free to toss me an email showing whatever cool things you've been doing with it.

I now have a discord channel here as well where you can chat with others or ask questions. https://discord.gg/d7paEng

Updates

4/11/18:

Adjusted the demo maps "game mode" setting so that you should spawn in with the default player pawn that can fly around even if your project has a default game mode that would otherwise spawn 3rd person or such. A minor issue since the demo map was made wit a flying camera type of player experience in mind.