MAD PRODUCTION PIPELINE

CLEARANCE LEVEL: ALL

DEPARTMENTS: MAD

AUTHORS: SPOOK_EXE – L3 MAD SCRIPTER

OVERVIEW: This document will be your official MAD guide and production information source for Snoop's SCPF. This document defines the production pipeline of how things should be done in the foundation, more specifically the manufacturing department. This document contains the variety of many development aspects of the game, so please read **all** that **could** apply to your area.

DATE: 12th May 2022

TABLE OF CONTENTS

Section-1 | A Note From SPOOK

Sup fellow developer of Snoop SCPF. This document does not necessarily apply to areas such as GFX and SFX artists, but are for primarily Scripters, Builders and Modelers.

This document I have written is designed to help you understand and produce excellent consistency and organization when importing your assets such as scripts, models and builds into the site.

It is key that you do follow these through your development process as it will make everyone's lives much easier when editors of the game place, and/or if you are providing assets to be added into the game.

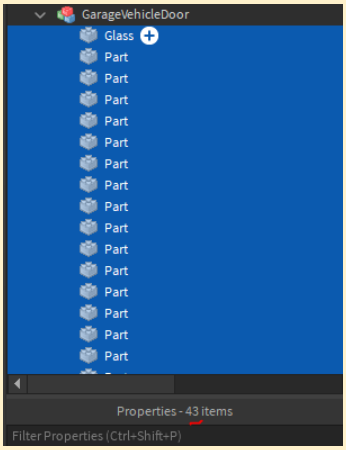
Without further ado, find your primary role section (Scripting, Building, Modeling) and read my tips and tricks for organizing and editing your assets!

Section-2 | ALL-ROUND GENERAL KNOWLEDGE

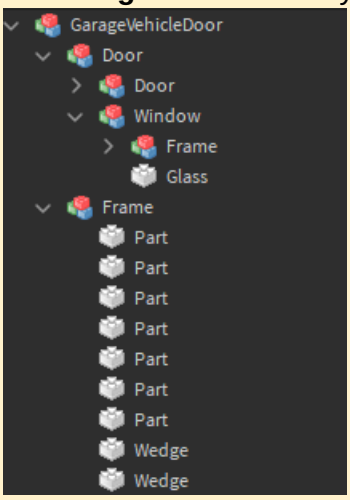
When working with anything that is being added into the game, in any regard, make sure to organize them. When organizing, make sure to name the assets to what they are, and make sure the children of the models (stuff inside the model itself) are all separated into their different items.

Here are some examples of good and bad organization below.

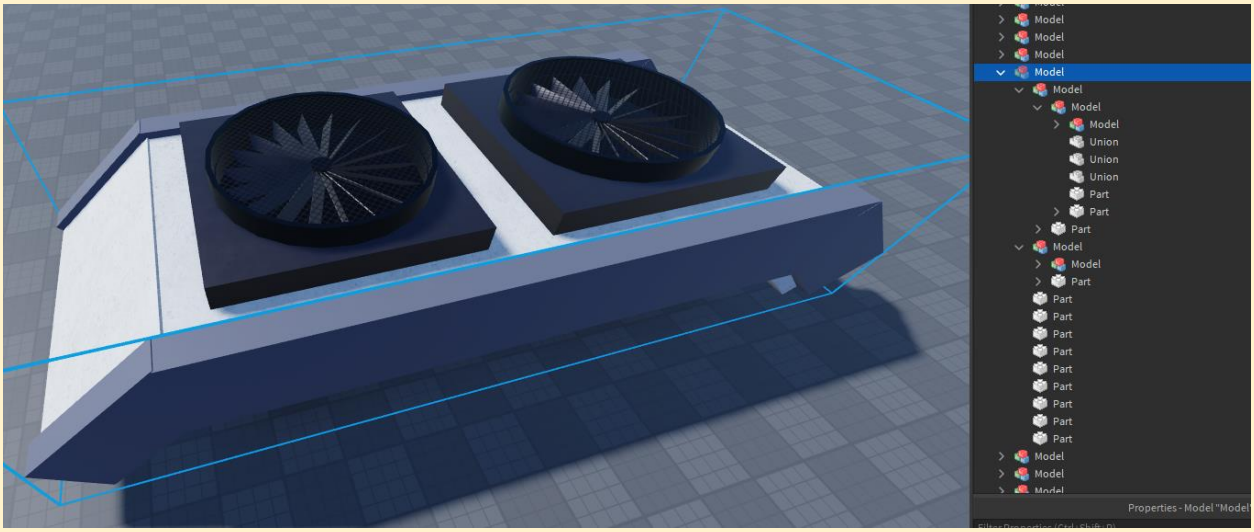
Bad Organization: no groups to separate and identify the parts.



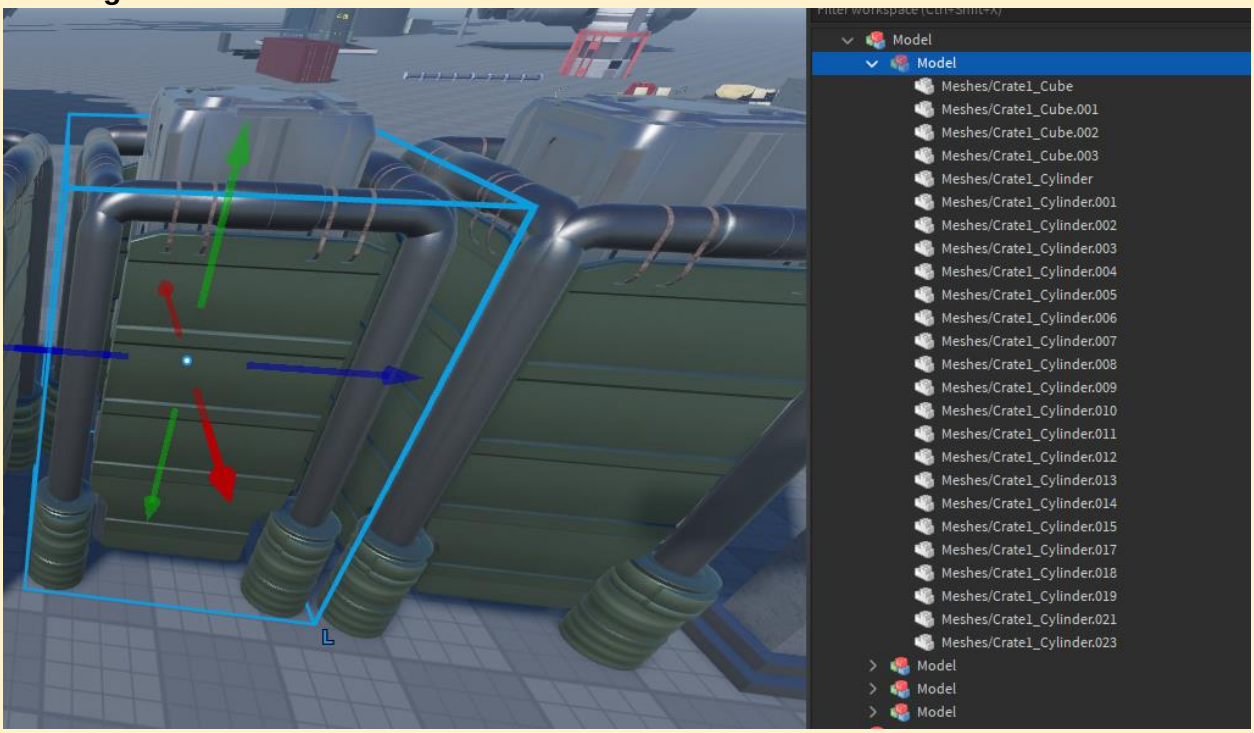
Good organization: everything is grouped and named, easy to identify what is what.



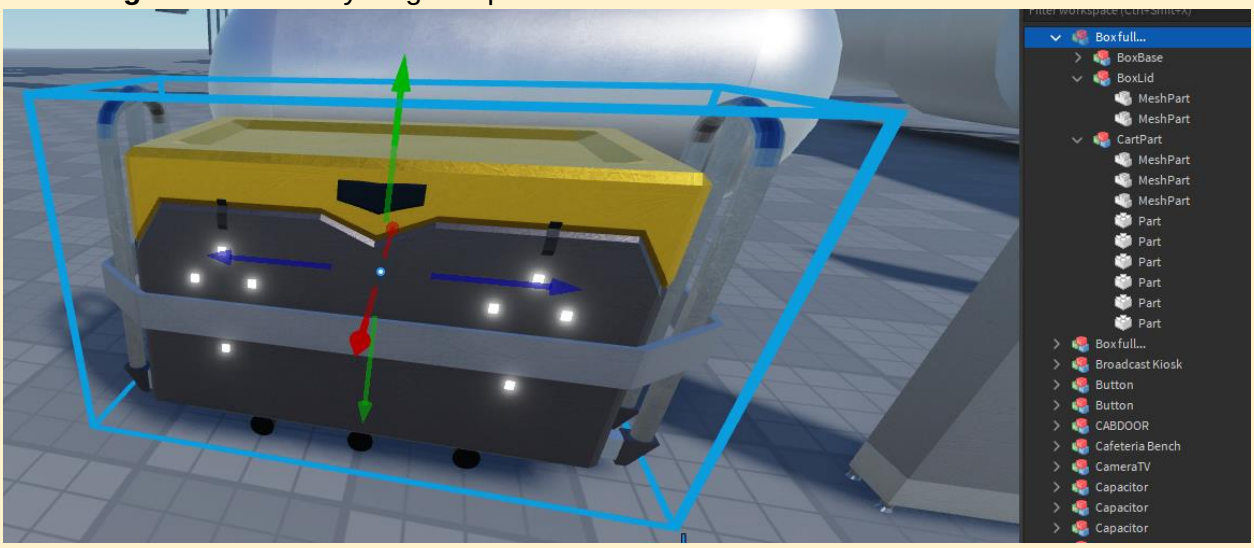
Bad Organization: even though it is grouped, nothing is named. Hard to distinguish what is what.



Bad Organization: same as above.

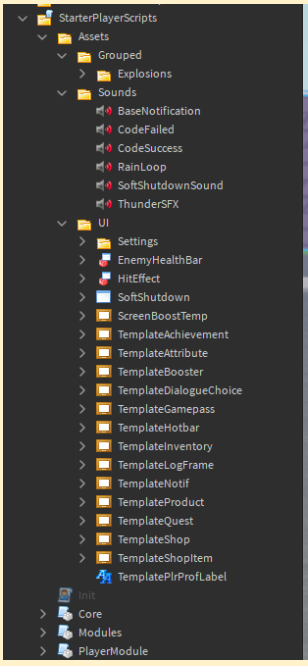


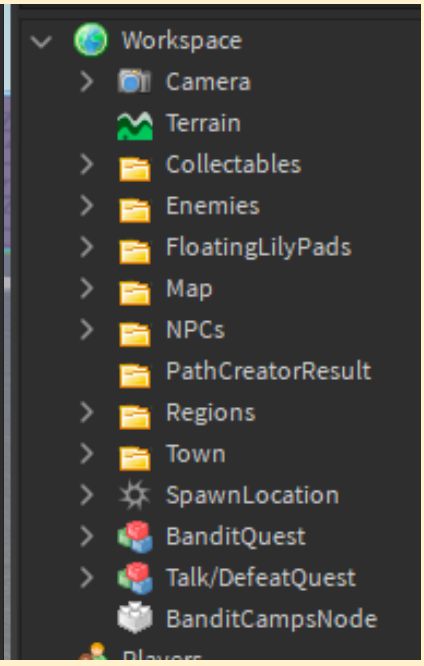
Good Organization: everything is separated and named.

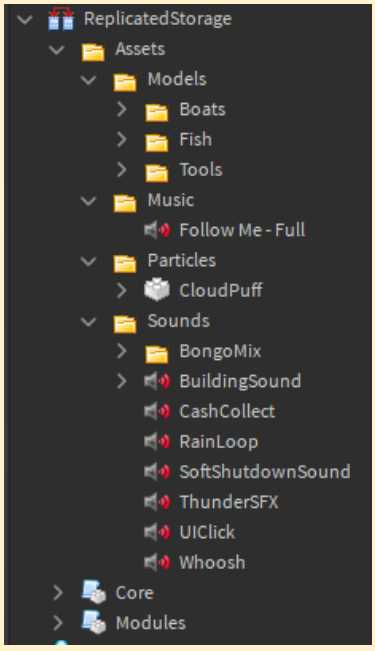


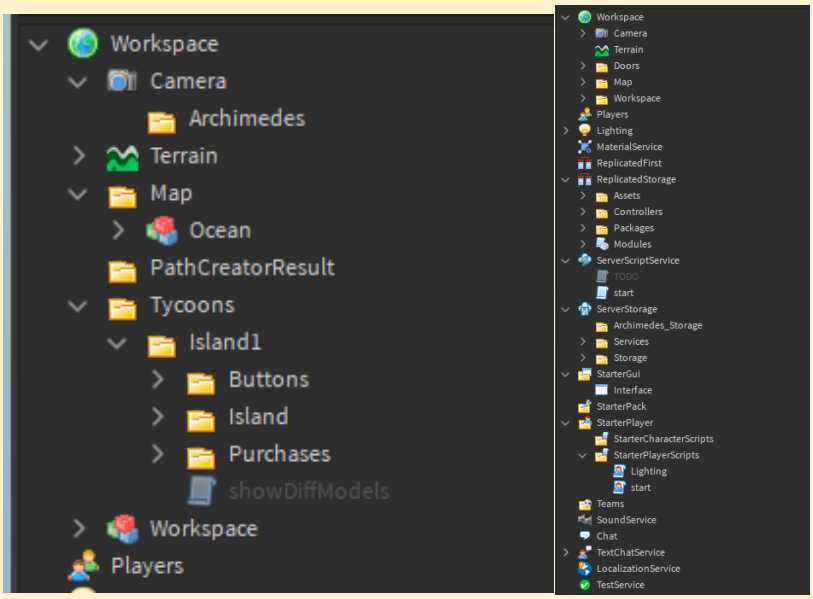
Improving on organization can dramatically improve the speed and efficiency of other departments and overall reduces headaches when editing the game. Imagine being able to easily identify objects in the game without going through much trouble, everybody would love that.

Here are some examples from many of my own projects where I am addicted to organization in them, keeping everything separate and organized.

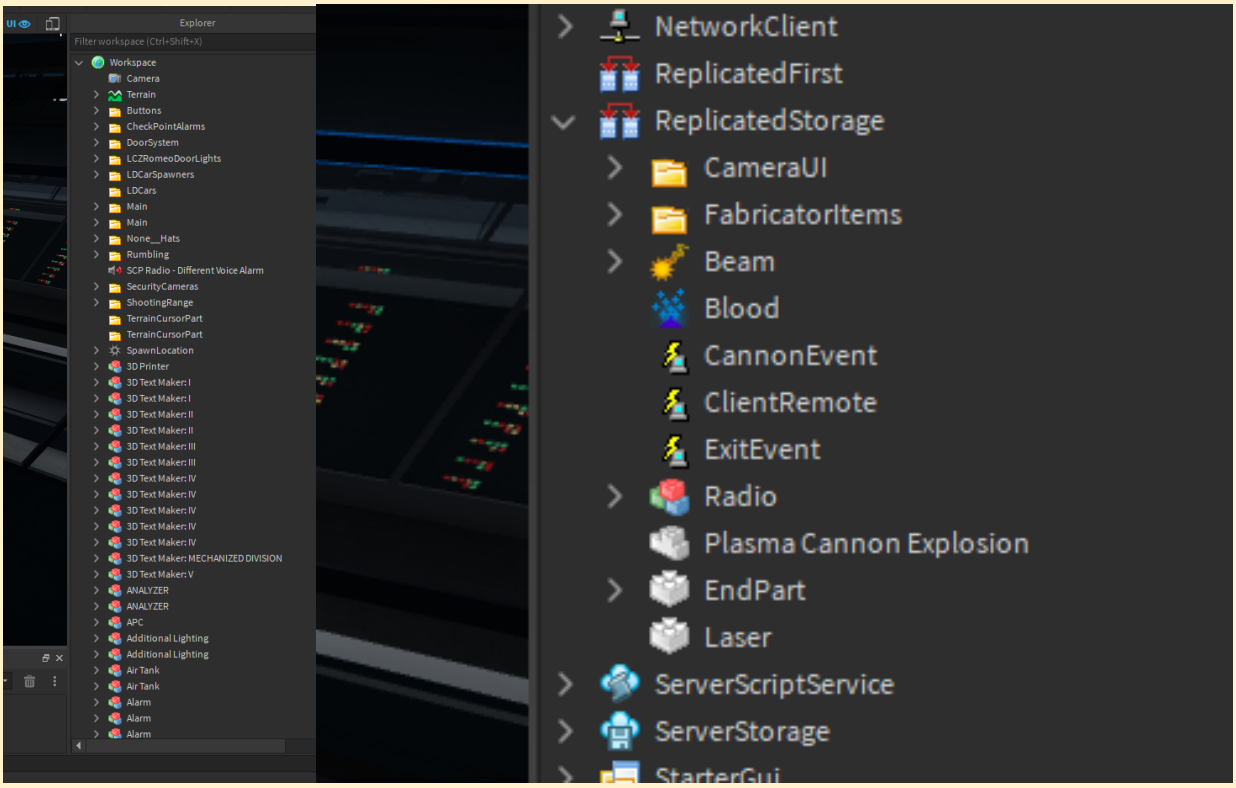






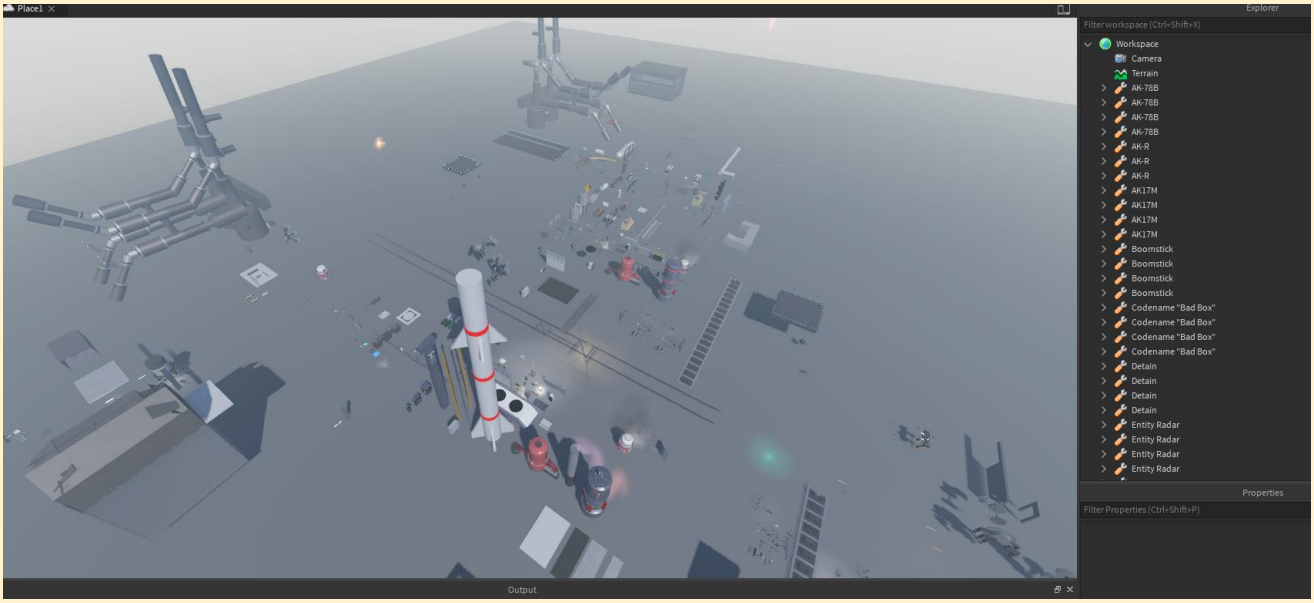


From what I've seen currently (12th May 2022), not much is organized at all in all aspects (general assets, scripting, building and modelling) which is a huge problem for such a large project.

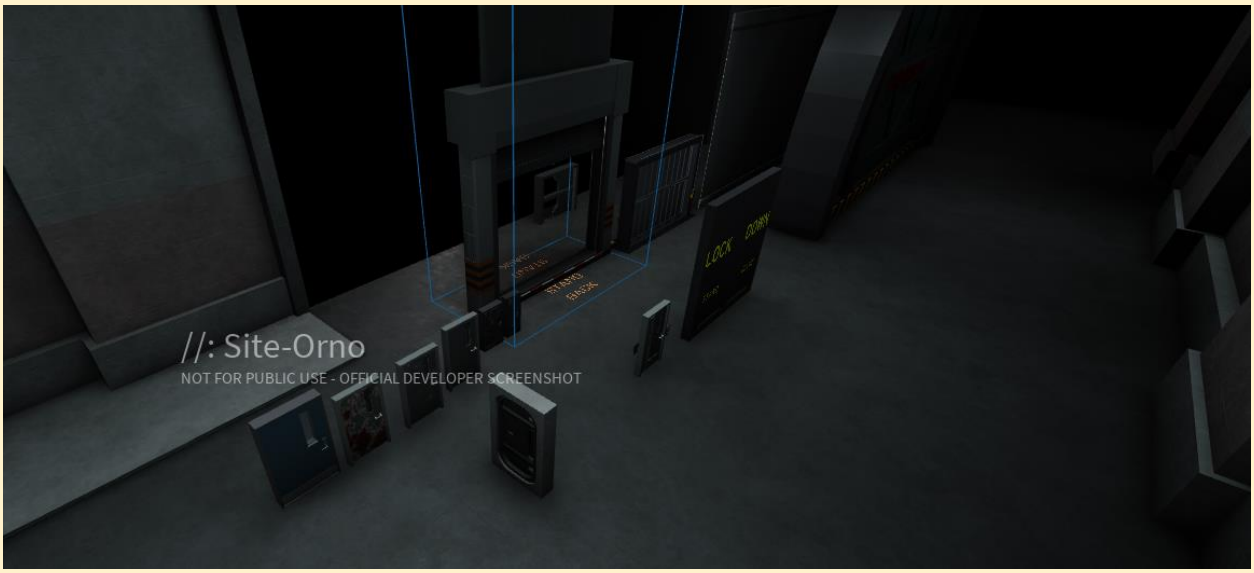


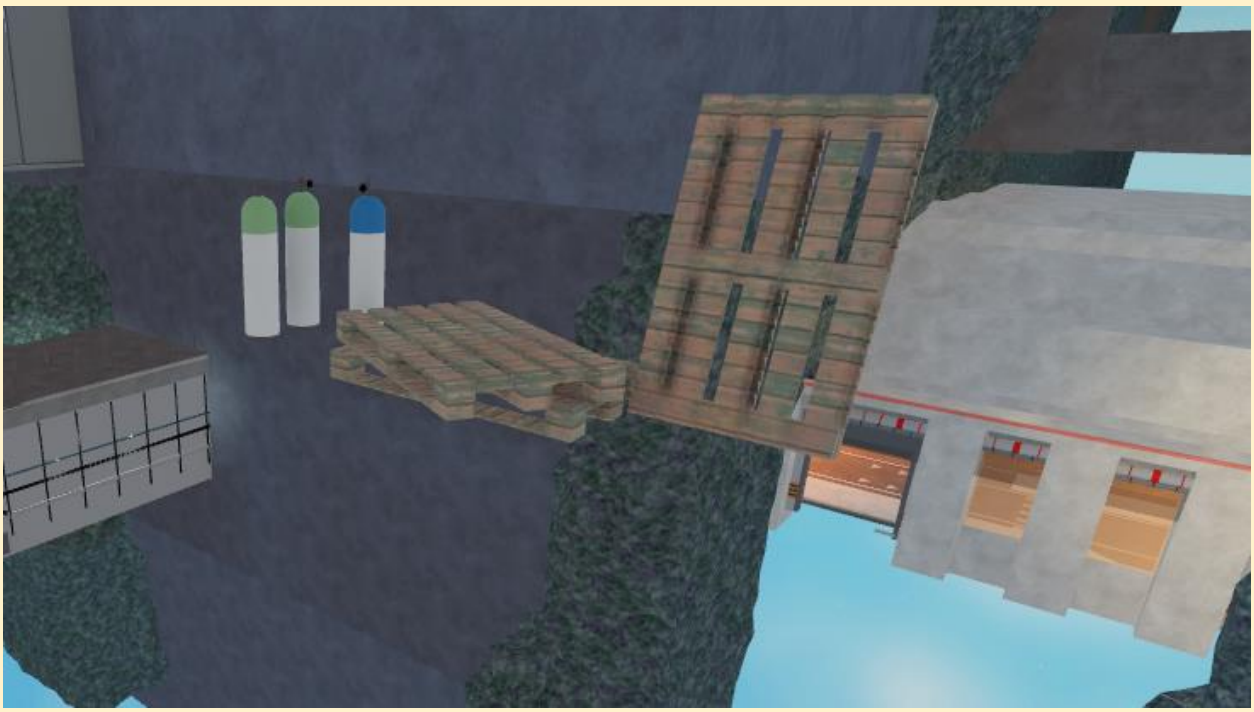
Onto another issue with general assets. **Keep them out of the main game.** Put them in separate asset places or a singular place. Currently, there is one main asset place in the SCP group which contains lots of models (keep that place organized unlike it is right now!) and should be utilized to its fullest. When we keep the extra models out and organized in a separate place, it can dramatically help with load times as only essential assets are kept in the main place. This is also another major issue with the current game as of 12th May 2022.

Current asset place: unorganized but contains heaps of assets that could slow download times in the main place which is great to have here.



Current state of the SCP site; random assets floating everywhere.





CONFIDENTIAL!

This document may not be shared or used by personnel below the designated clearance level.

From what I've shown here, you can see there is a lot of assets that are just floating around taking up precious computer memory for some (not everyone has a beefy computer) and so its important to keep things that are unnecessary out of the place.

You can exclude things like copy/paste spamming temporarily models around as it would be faster doing it that way, or essential models that you are **actively building** or **using whilst actively building an area**. Anything unused however should be moved out of the place, and this counts for anything underneath ServerStorage / ReplicatedStorage and such services.

Section-3 | SCRIPTING GENERAL KNOWLEDGE

In the Site Orno scripting team, the scripting pipeline is setup using Git and Rojo. We use advanced version control from git to monitor the amount of work someone does and allow others to review peers' works' before or after they've been contributed to the project.

When doing an assignment, scripters need to create a new branch and a pull request to the main dev (development) branch, as the dev branch itself disallows direct commits because of peer review. Keeping track of a scripter's progress is important, so make sure to regularly commit to the repository or keep an update in the development log. Prolonged absence of work in the repository without an inactivity request may result in consequences, from a warn to termination of the scripter role.

Rojo is a system that allows someone to synchronize a directory in their filesystem (on their device) to Roblox Studio. This itself follows a certain format, so its important that you know how to structure your files so that it syncs to the correct places and being the correct type of script.

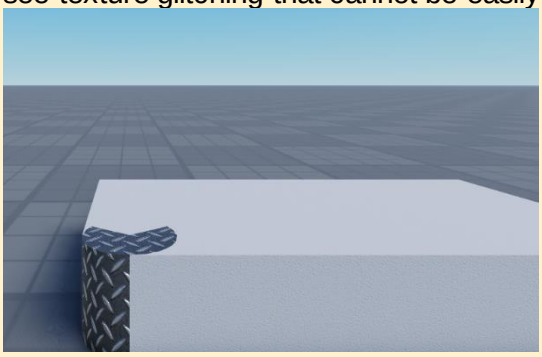
Overall, the scripting development pipeline is designed to allow the highest insight into the work being done and to make it easy to see scripts. It is also designed this way so that we can limit the number of people with studio access to the game, as any changes to the dev branch can simply be added to the game with the synchronization using Rojo.

Section-4 | BUILDING/MODELING

This section is split up into 2 parts: the building section and the modeling section.

BUILDING – Unions

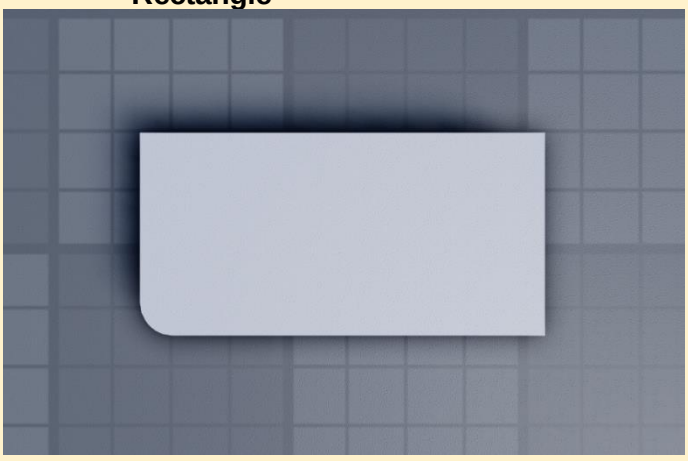
When building within site Orno please try to minimize the number of unions you use. If you must use unions here are some tips and tricks on how to make sure there are no problems with said unions. Before I talk about how to avoid union problems, please remember to only union if you see texture glitching that cannot be easily fixed. (Example image below)



Example of texture glitching (Z-fighting)

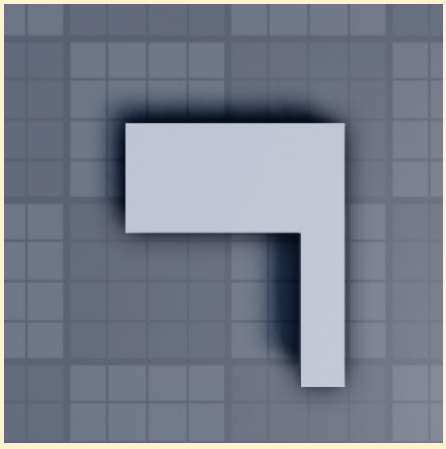
To start off, I'm going to be crystal clear here, **the collision model for unions suck ass.** I mean this wholly, and fully. There are ways to avoid major issues. I will now present a few examples of how you can correctly deal with union collisions.
When using unions only union "cube-like" shapes. An example of what I mean is shown below. The problem with union shapes like the ones shown to the right is that they create terrible collision models, and this can lead to many problems.

Rectangle



Compatible with union

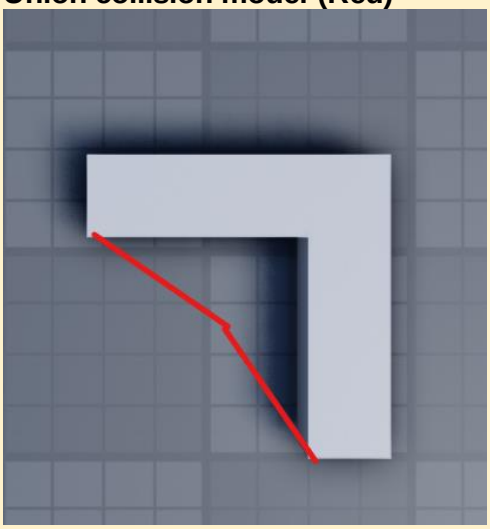
Polygon



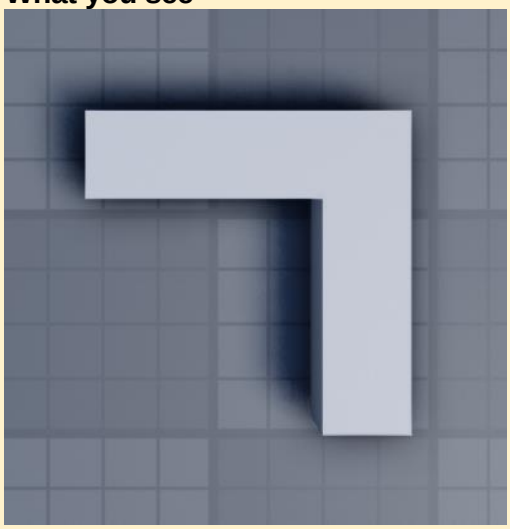
Incompatible

Collision model demonstration (exaggerated, but you get the point)

Union collision model (Red)

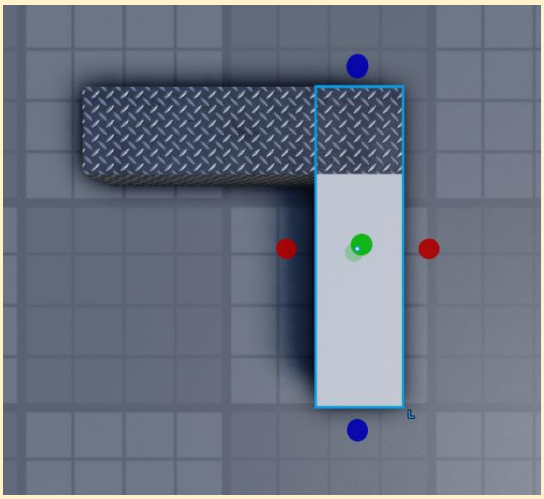


What you see

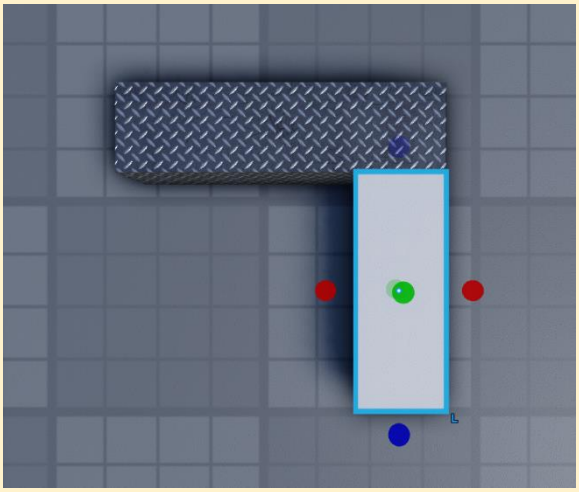


With all said there are some ways to fix z-fighting (texture issues), here are some examples of how to do that below.

Avoid collisions between parts



Collisions

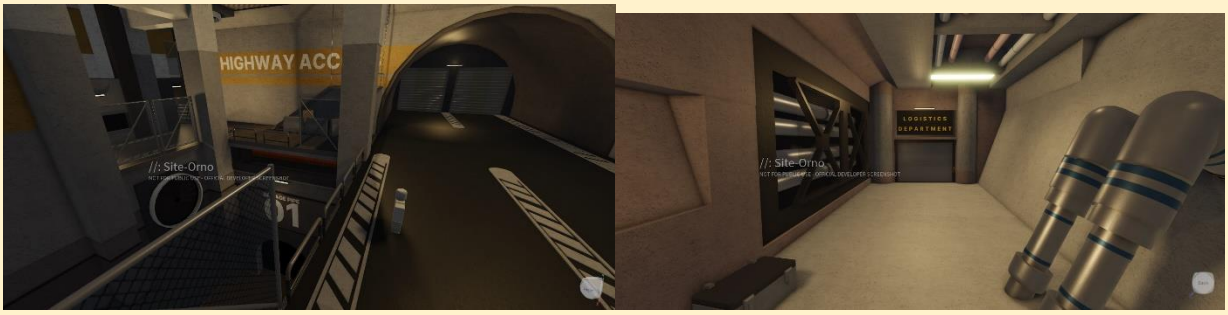


No collisions

Textures, Lighting and General Notes
IMPORTANT: PLEASE REMEMBER THIS IS A HORROR GAME.

With that being said, I can't stress this enough; atmosphere and lighting really do matter. Here are some examples of good lighting that fit the setting versus bad lighting.
(This goes without saying but not everything will be super dark and not everything will be super bright)

Good lighting **Bad lighting**



Explanation (Left Picture): In this room, you can see various variations in lighting. For example, the hallways are lit up a bit more and you have lights highlighting points of interest, (garage doors, label sign, etc). The lighting matches the different sections of the room.

Explanation (Right Picture): This room is generally bright, which is okay BUT as you can see it's literally just same amount of brightness throughout, there is no variation or highlighting of certain areas.

When you are building, please keep in mind the player. I know it's very easy to get lost in your work but please don't make a room with ceilings that are 4 stories tall. A very **easy** way of making sure this doesn't happen is to simply have a dummy with you while building. This makes sure that you don't make any of these mistakes. (In the second image below you will notice that a basic door is the height of 2+ dummies, that is ridiculous).

Example of good scaling **Example of bad scaling**

When dealing with textures here are some things to keep in mind. If you can, please only apply textures to **visible areas**. If you need to make a texture more noticeable you can do so by either **making it darker or making it less transparent.** There should **never be a reason to double-stack textures.** Double stacking is a complete waste and can cause lag issues.

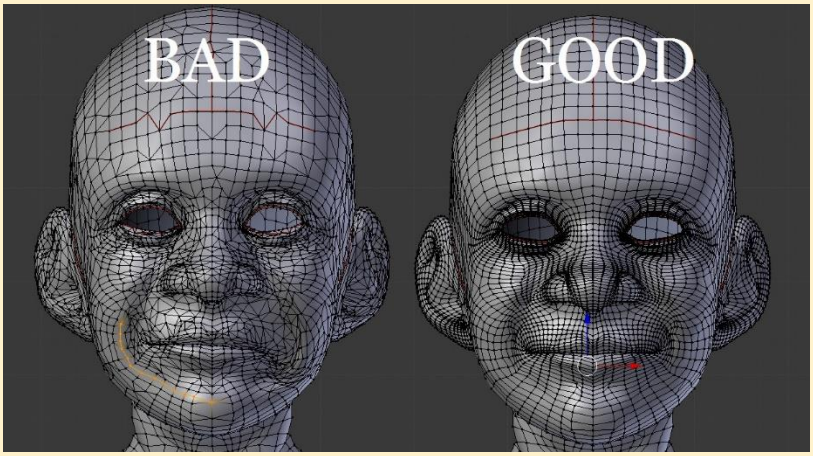
Now that I'm essentially done with everything, here are some **general notes** to keep in mind.

- Don't be scared to **ask questions** about things to your superiors.

- **Log your time.** Please for the love of God, log when you work in #development-log, it's inspirational to see you folks working on things and is a general morale booster to the team.
- **Show off your work.** We **want** to see what you are doing; communication is a value. Don't think you're a dick or a jackass, show off that thing you spent 4 hours making we want to see. Most importantly, be **proud** of your work.
- You are always welcome to **take a break**. If you are feeling burnt out, tell us and make sure you log it in #inactivity-notice. We care about you and if you ever need something please contact a member of HR or just anybody.
- Please for the love of God **don't build useless things**. I can understand if it was a specific request but if you make for say, a hyper detailed wall for absolutely no reason with 4 thousand parts in it, that's getting deleted.

Modeling - Topology

Before I start this section, please know that the person who is writing this isn't the world's greatest modeler. To start off when modeling please keep in mind the topology of what you're making. Whenever you make a model please keep the topology in mind, topology as I'm guessing most of you know is **super important** for any model. Good topology basically makes a 1-hour job of texturing into a 10–20-minute job as many of you know. Below are some good examples of what I mean. (Yes, I know it's a creepy example)

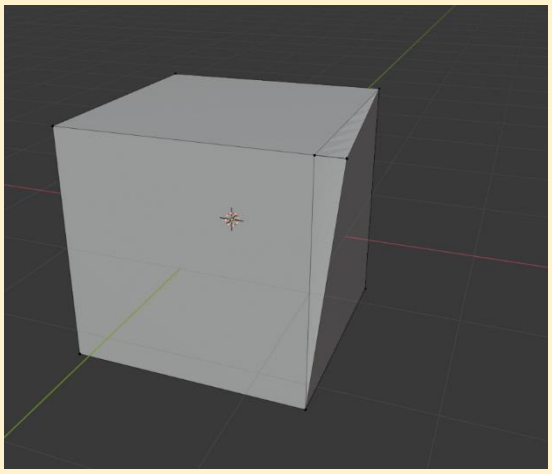


This goes without saying but for the love of God **keep triangles in mind when modeling.** When you make something don't just subdivide for no reason. This kind of goes hand in hand with topology as good topology means that you don't have random triangles for no reason. Please also remember to demolish your models, when possible, to rid them of any unneeded triangles. If you do demolish your mesh's, please keep in mind that demolishing changes the topology of your model. This means that you may have to fix the topology.

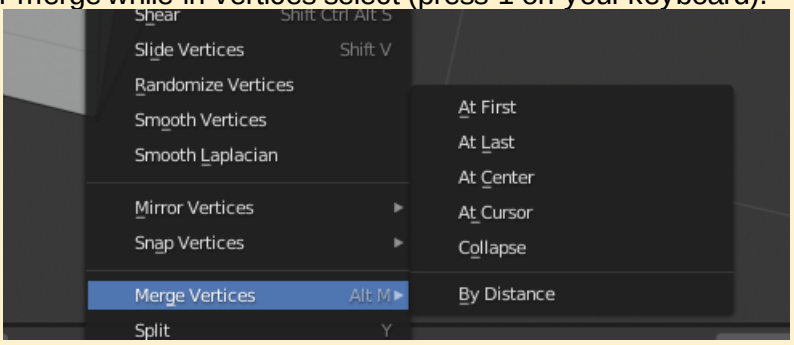
Overlapping Vertices

This is a common mistake that happens a lot, I'm sure most of you know how to fix this easily but I will still go over it as it's very important to know especially on mirrored surfaces. When a vertex is overlapping with another, this tends to cause a large amount of shading glitches and just makes it a pain to model in general.

Example of overlapping vertices



Overlapping vertices can be easily fixed by selecting 2 overlapping vertices then **right clicking** and hovering over merge while in vertices select (press 1 on your keyboard).



After you do this, you can easily choose where you want the vertices to be merged, either in **the center of the 2 vertices, the first of the 2 selected vertices or last of 2 selected vertices. I do not recommend using at cursor.** To speed this process up, you can simply press **Shift+R** to repeat the process **although you do need to have 2 selected vertices first.**
Now that I'm essentially done with everything, here are some **general notes** to keep in mind.

- Don't be scared to **ask questions** about things to your superiors.
- **Log your time.** Please for the love of God, log when you work in #development-log, it's inspirational to see you folks working on things and is a general morale booster to the team.
- **Show off your work.** We **want** to see what you are doing; communication is a value. Don't think you're a dick or a jackass, show off that thing you spent 4 hours making we want to see. Most importantly, be **proud** of your work.
- You are always welcome to **take a break**. If you are feeling burnt out, tell us and make sure you log it in #inactivity-notice. We care about you and if you ever need something please contact a member of HR or just anybody.
- Please for the love of God **don't model useless things**. I can understand if it was a specific request but if you make for say, a hyper detailed wall for absolutely no reason with 20 thousand triangles, that's getting deleted, and you've just wasted your time.

Section-5 | MESH OPTIMIZATION

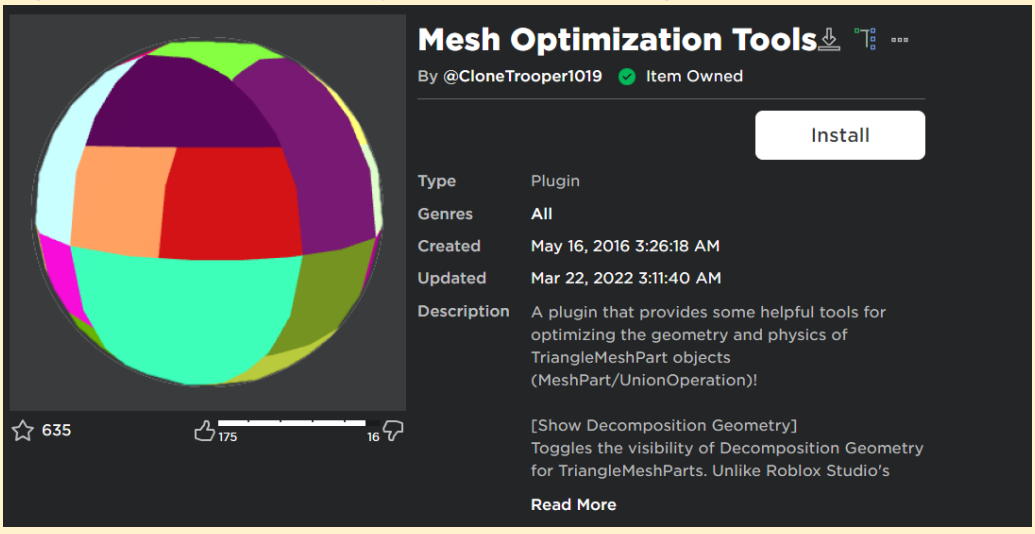
When working with MeshParts and Unions (preferably MeshParts), there are additional properties within the MeshPart, when changed, that can potentially introduce major performance boosts when that specific MeshPart, or collection of MeshParts is used in bulk.

Examples of this can be found below, however, for this quick text section I will get slightly technical about why having specific properties can introduce more memory or performance hit; primarily the “CollisionFidelity” property.

Every object within the Roblox environment has a physical bound and outline created of **Quads and Triangles (Precise Decomposition)** for the physics collision system. This represents the physical boundaries of that object and is used for any collisions or raycasti ng with the object if the CanCollide **OR** CanTouch is enabled, and the CanQuery option which affects Raycasting into the object, but that should not be worried about.

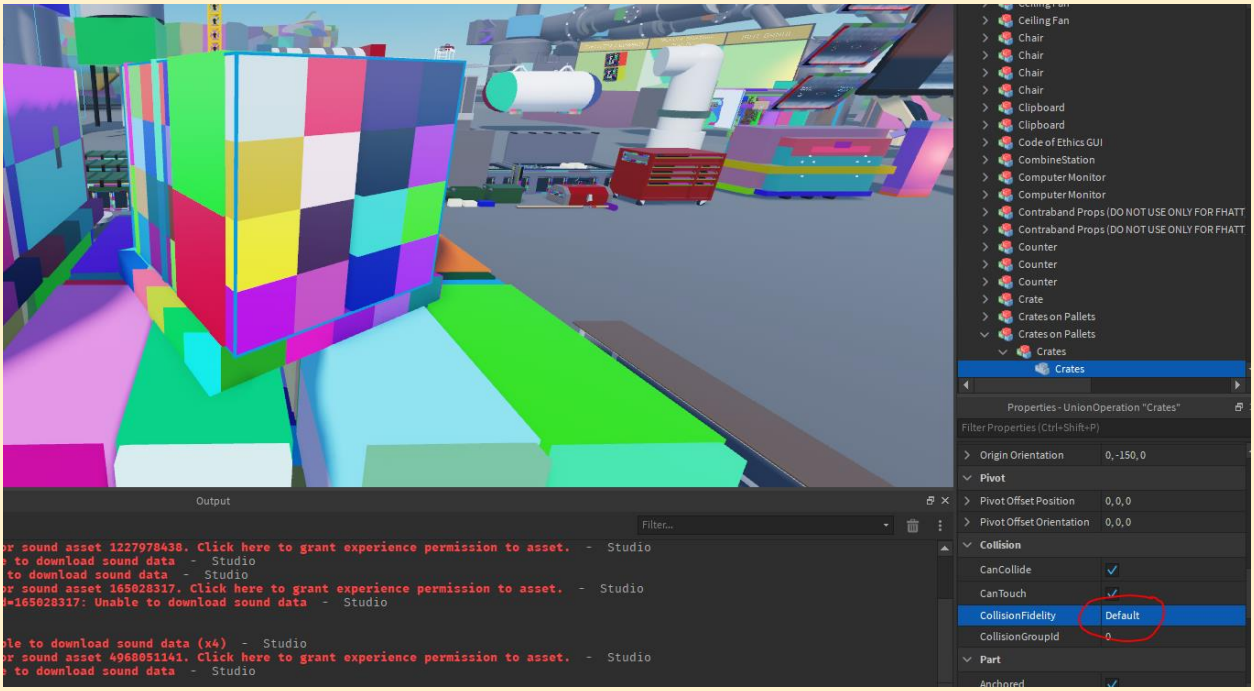
By optimizing the number of quads and triangles that are found in the meshes, the total number of quads and meshes which are fed into the physics system can be dramatically reduced by changing the CollisionFidelity of the mesh. By normal means, changing it from default should be your number one priority, however, some cases may be presented where you cannot change it from default.

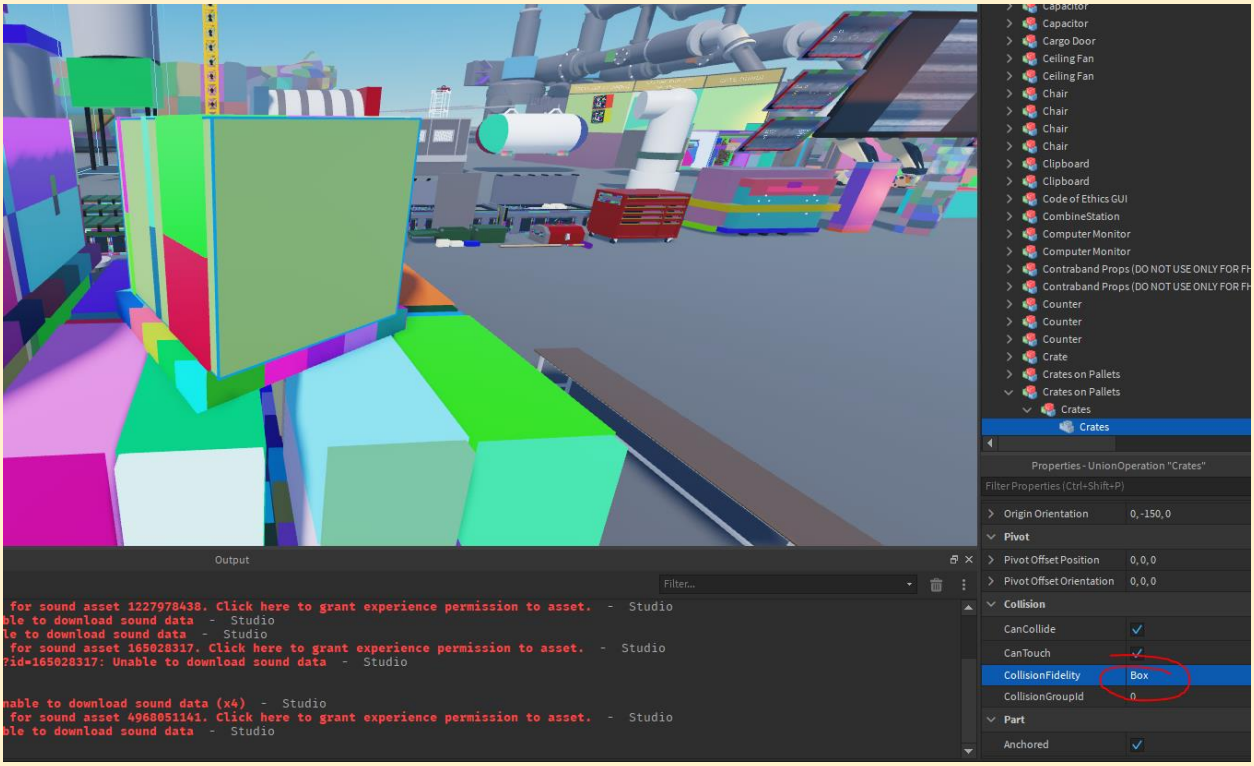
An extremely helpful plugin which shows the quads and triangles which was used in the examples below, is here:
https://www.roblox.com/library/414923656/Mesh-Optimization-Tools



Now to have a look at the examples, these demonstrate the number of quads that are being reduced by using the different options.

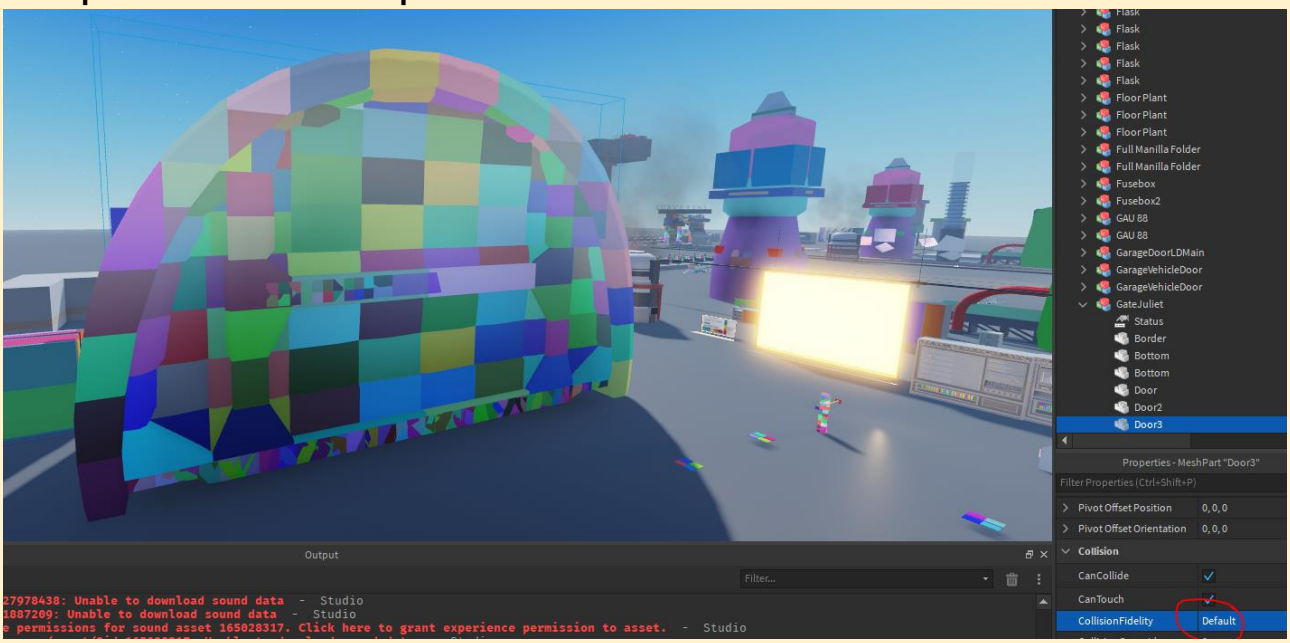
Example 1: Crates exa mple

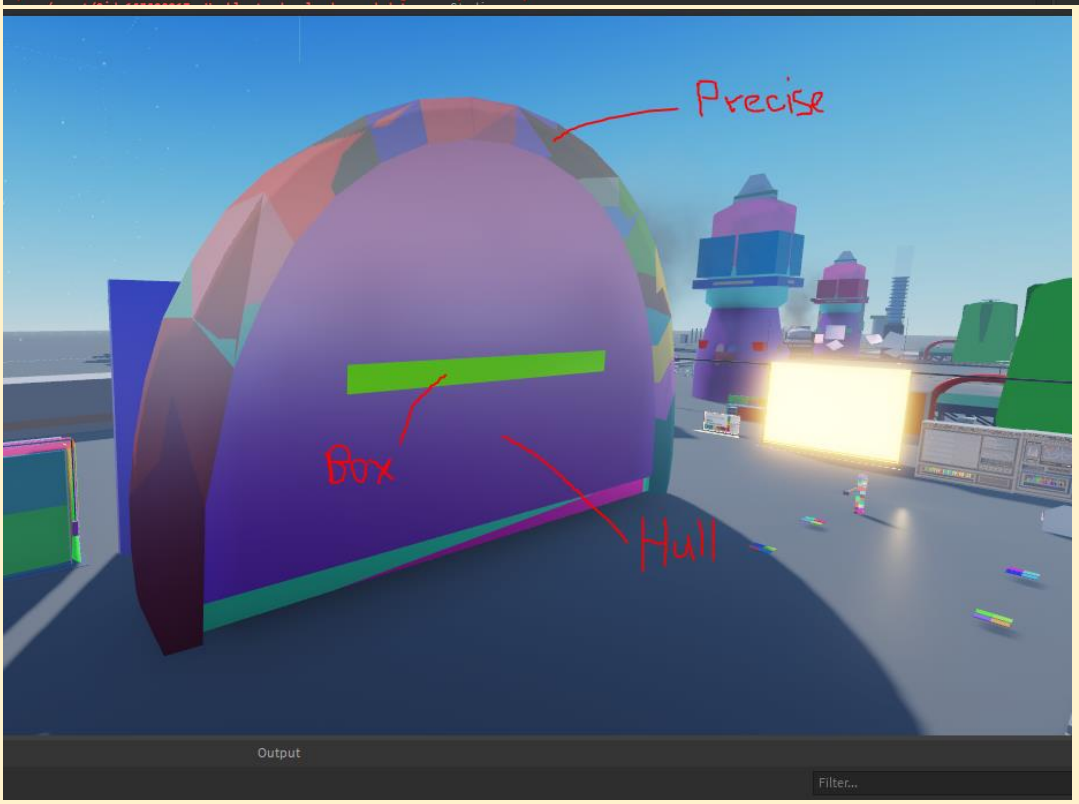




Dramatically reduced the number of quads for that one mesh, also there is no need for the precise collision of that crate mesh.

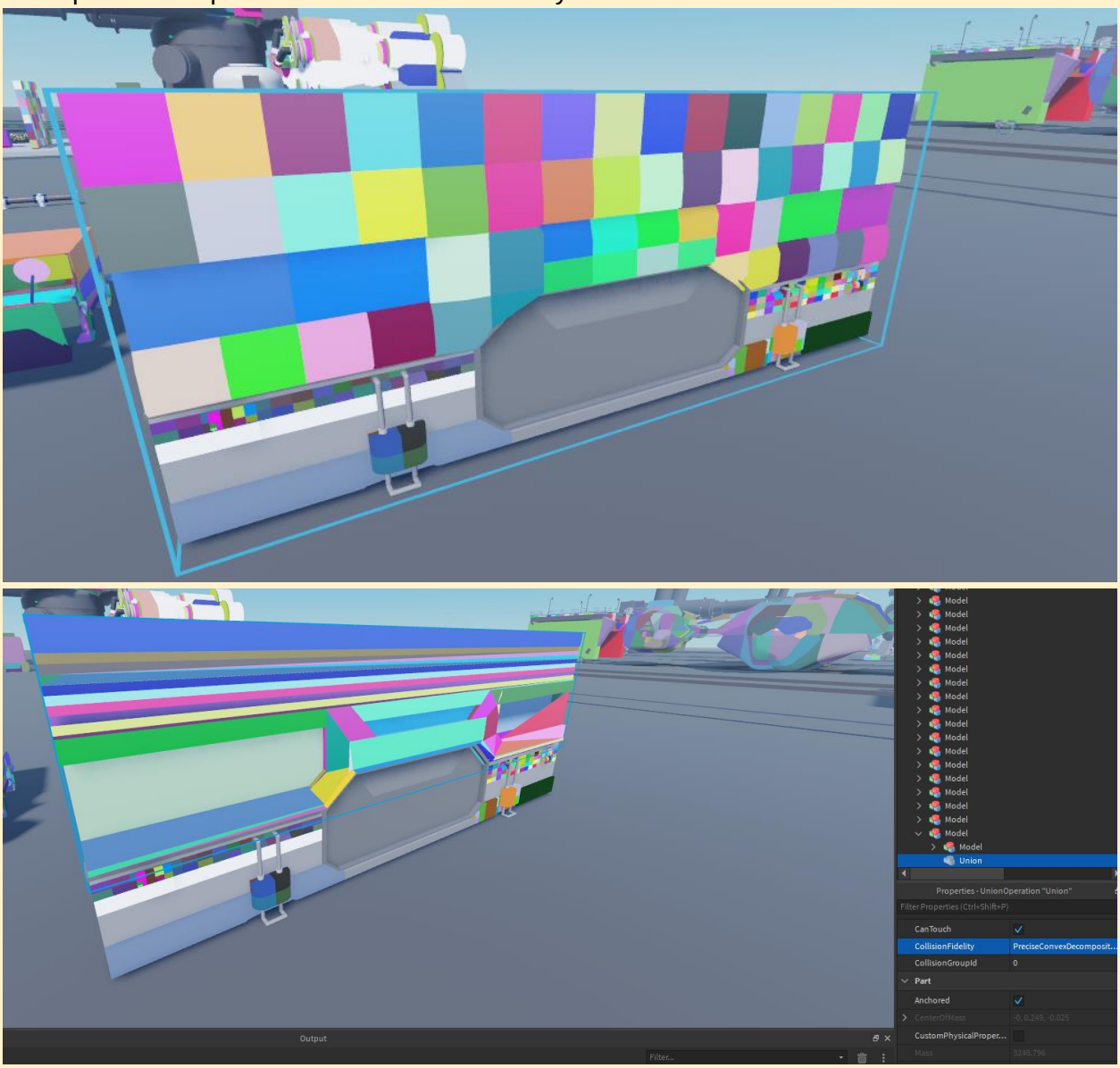
Example 2: Gate Juliet example



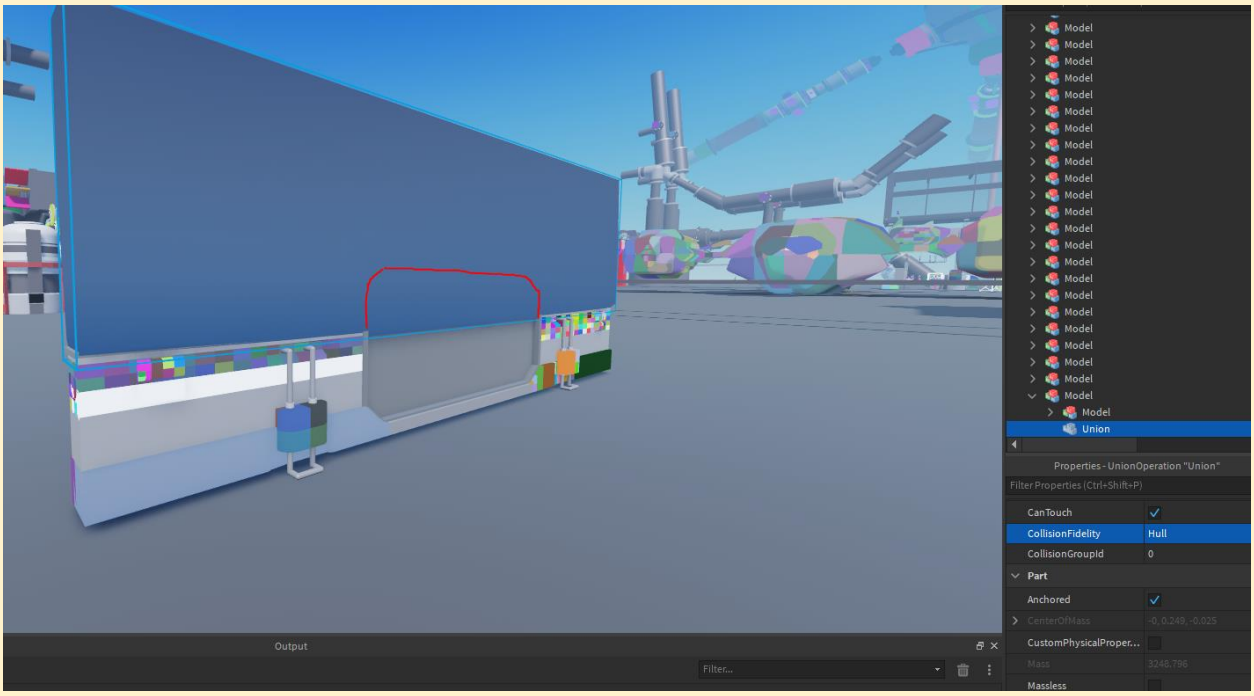


Again, no need for precise collision so we can reduce it overall. The precise composition may make the mesh look finicky at times also, however, that cannot be fixed unless its taken back into blender and corrected by mesh editing means.

Example 3: Example scenario where default only works.



This is an example case where the union cannot support the different options due to the variety of shapes which Roblox poorly put together through the union system. In cases like this, you should report the shape in the discord to the builders/modelers so it can be looked at and hopefully fixed. Just skip over this one and leave it on default.



Using a mixture of precise decomposition, hull and box, you can optimize the mesh in a way which keeps its collision box practically the same, but overall reduces the number of quads within the model and each MeshPart itself.

Obviously, such as in example 3, some cases you cannot use anything but default and that is completely fine, just make sure to note it somewhere for a builder as we may need to potentially fix the issue if that specific meshpart / model is used in lots of places. That is the primary lag producer for meshes so keep this in mind when working with meshes.

Another important note, for anything that is **non-collidable and non-interactable** (eg: bushes), set their collision fidelity to **box** as that minimizes the total number of quads/triangles that it is taking up in memory.

That is all for this section, hopefully you grew a braincell or two!

Section-# | TITLE

…
…
…