uConstruct Documentation

Content :

About :

uConstruct is an asset that was created to allow people to add building system into their game with any kind of a model, the code is fully generic and you can use any kind of a shape for the buildings (sphere buildings, boxed buildings etc).

uConstruct is an fully generic building system with a fully commented and exposed API to allow people to easily change or integrate 3d party systems into the system, It also comes with full source code.

uConstruct also comes with full source code, and is extremely generic for any modifications. It also comes with an hugely exposed API and gives you the ability to integrate any 3d party systems easily.

Using this document you will learn how to use uConstruct and its API, you will also learn how to integrate 3d party systems and how to get this system to work inside your game, you will see how to use the batching feature, how to design sockets and conditions and how to code your own custom conditions into the system.

One of the most important features in the "uConstruct" is the built-in render batching support, and a multi-threaded Area of interest mechanic, those 2 features allows you to get the max performance you can from unity, the renderer batching reduces the building draw calls into 1, while the AOI keeps only the colliders, sockets and conditions that are around you enabled.

Basics :

uConstruct requires you to setup several things before starting to use it. First of all you will need to create 2 layers. (Building, BuildingSocket), you could generate the layers through the UCManager or manually create them. Please make sure **none** of those are with spaces.

In case you are new with the asset you can check out its functionality by checking the "CubesDemo" that we made for you, in the demo you will see how the system can basically work the same as most other building systems that games has those days and you can learn how to modify the Sockets, Conditions and all the rest.

The controls for the demo :

Keys are shown on the UI.

Left Mouse Click – Place, Right mouse click – Destroy (only works when you have nothing in your hands), Also clicking on the same number of the slot you are currently holding will remove the building in your hands (so for example if you are holding a foundation and you click the foundation slot number again it will remove it from your hands).

Throughout the docs you will learn how to use the API, Customize your own conditions and sockets, generate building types, create buildings and also how to integrate other systems into this asset.

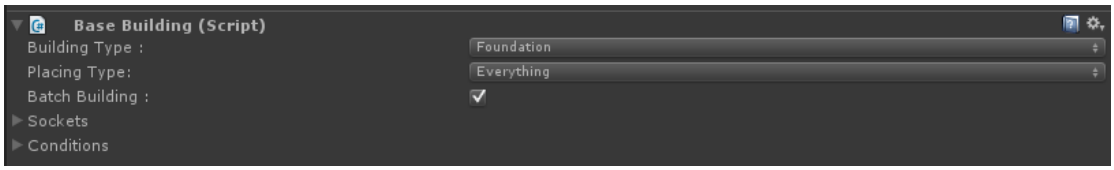
You will also learn how to use some unique features such as building batching, area of interest and much more!, I will also create some videos and put them here and in my Youtube channel so if you prefer videos you will also be able to learn for them.

Buildings :

Any building in the game will need to have a building script. This script will have data about the sockets of the building, the conditions of the building, and much more.

In this section we will learn about the building script and what each variable does, how to customize it and much more.

So lets start, this is how the inspector of a basic building script look like :



We can see several variables there :

Building Type : What building is that ?, this will be used when making sockets so you know what building this socket applies to, you can customize this enum with the system code generator and this is explained down in the docs.

Placing Type : This will be the method of placing the building will use, you can choose "Socket Based" or "Freely Based" or even both, a good example to why you would use both would be when having a foundation, that way you can freely base it in the terrain and it can also be snapped to other foundations.

Batch Building : This variable will choose whether this building will be batched with the system batching mechanic.

Sockets : This section will be explained on the next chapters.

Conditions : This section will be explained on the next chapters.

And that’s it for basic building creation, further down the docs you will also learn on how to make sockets, conditions and more.

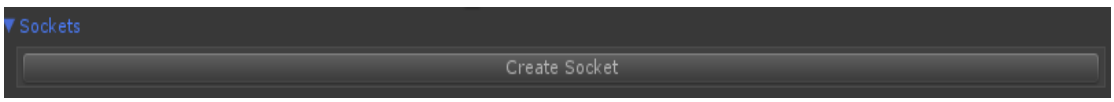
Sockets :

*** What are the sockets :** Sockets are colliders that are placed on the building that will allow you to place other buildings on the building, a socket can be a free placed socket or a snapped point socket (those options will be explained deeply below), Also note that they aren't networked sockets so do not get confused between them.

*** What is "Free Place" socket and "Snap" socket :** Each socket needs to choose whether the building placed on it will be free placed on the collider zone of snapped to the position.
So if for example you place a wall in a rust game, the socket will be a snap socket on the foundation which will make the wall snap to that socket position. Now lets say you are placing a foundation and the terrain is a socket, the terrain will be a free place socket as you can place the foundation freely all over the terrain collider.

Now that we have learned those basic things we can head over to how do we basically create our basic sockets in our building, we will also learn how to create our own socket script that inherits from the BaseSocket script in order to make some custom actions in the more advanced explanations.

So, how do u add a socket into your building?, Firstly open the socket section in the building script like that :



Click on the Create Socket button and a new window should be opened.

Name : What is the name of this socket ? (Can always be changed later, just change the name of the socket game object).

Starting Position Anchor : What local position of the building will the building start on? (wont influence anything in the future, just for easier placing of the socket).

Socket Gizmos Preview GameObject : An prefab (can also be a model, but preferred a prefab) that will be showed as a display, if you put the prefab there when you create the socket it will auto-scale the socket to the prefab size, that way it will help you place and size your socket.

Receive Buildings : What building will be able to be placed on this socket (Note : you can choose multiple types).

Placing Type : What placing method will be used on this socket (again, Free place or Snap Based or Both).

After you are satisfied, click "Create Socket" and now you will see a child was added into your building game object, this game object is your socket, you can also copy paste it for easier socket placing.

Now scale it up as you like and position it correctly on your building, and that is it for creating a basic socket, further down the docs you will also learn how to create a custom socket script and generating your own building types.

Conditions :

*** What are the conditions :** The conditions is a feature that allows you to create fully generic conditions when you place your building, what that mean is, lets take rust for example when you place a wall in rust you first need a pillar at each side of the wall, with the condition you can do that and you can create your own condition.

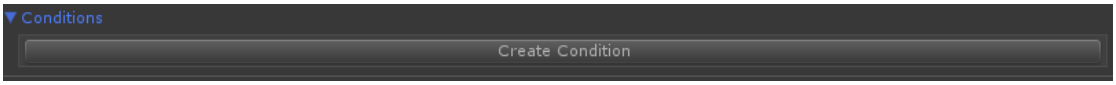
*** What conditions already comes with the system :** As of now, there are 3 built-in conditions that I created for the rust demo but can be used for many scenarios :

CheckForBuildingCondition – This will check for a building infront of the socket. so for example if you have a wall you want to put 2 of them on your building (1 at the right corner and 1 at the left corner).

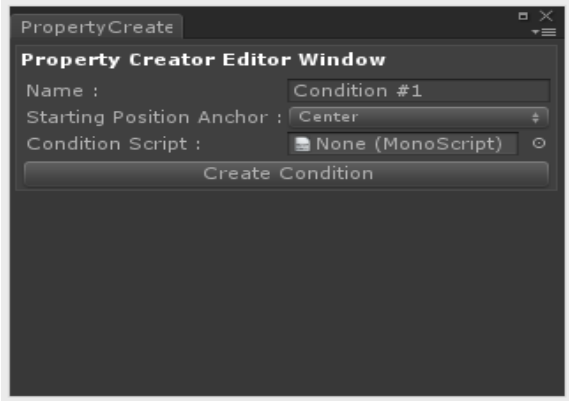
CheckForCollisionCondition – This will check if the building is colliding with any other building so you wont place a building one on another or place a building inside an other building, you will need to have a box collider on this condition in order for this to work.

CheckForGroundCondition – This will check if there is a building under the condition, if there isn't it will add a rigidbody to the building to simulate "collapse".

After we have learned those basics, we can now learn how to add a condition into our building. To start this up open up a building script and open the "Conditions" tab.



Here all of the conditions of the building will be listed. To create a new condition click on the "Create Condition" button and a new window should be opened.



Name : What is the name of the condition (can be edited later by editing the condition game object name).

Starting Position Anchor : local position starting point of the condition (wont effect any future actions, just a method to help you place the condition more easily).

Condition Script : the condition script it will use (can again be changed later by adding a new component to the condition game object, also this script has to inherit from "BaseCondition" and cant be abstract.

Click on "Create Condition" and you will now have a new condition in your building. Place it as you like and that’s how you create a condition.

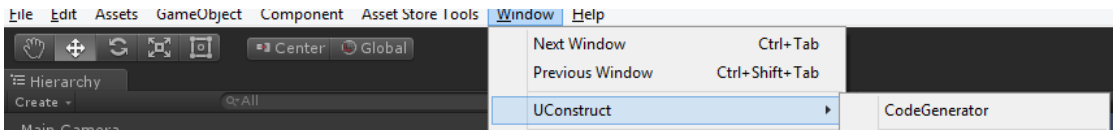
Further down the docs you will learn how to create your own condition and make it do anything you like it to.

Building Types :

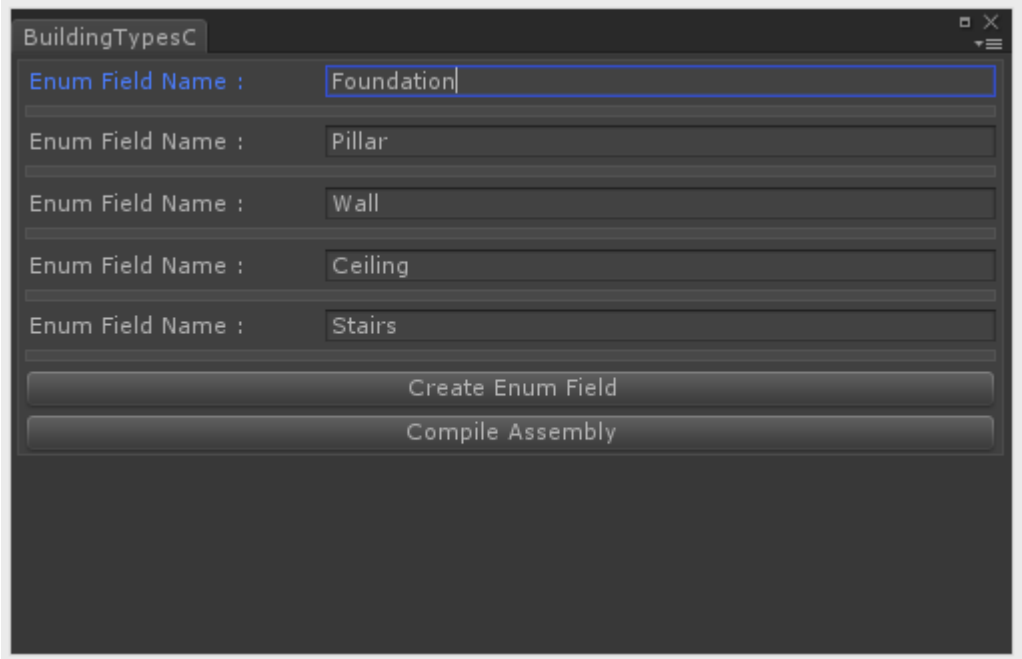
*** What are the building types :** The building types is an enum that helps the system identify building's type. So if for example you have a building Foundation, its type will be foundation so when you create a socket you can say : "This socket will only receive the building with a type of foundation", and that's why building types are required for this system.

*** I want to change the building types names, how do I do that :** One of the features that "uConstruct" offers you is a way to generate building types through editor without any code. The system itself will handle all the code generation for you so you wont need to get your hands dirty.

So lets get started, Open up the "Building System" Tab at the top of Unity and choose "Code Generator"



Now, A window should be opened :



Here, you can add as many as types as you like. If you would like to remove a type hold "Control" and click on the delete button, Please be warned that deleting types might cause your types to get corrupted and you will need to re assign them.

When you are happy click on the "Compile Assembly" button and it will save your changes. (Please Note : if you don’t click on that button, it will not compile your changes).And that’s it for code generation, you can now use the types that you generated.

Building Group :

*** What is a building group :** A building group is a script that basically being created whenever you place the first block on the ground, the system use that script in order to identify a building and also to do several methods such as Batching and Area of interest. The way this system works is by whenever you place the first block on the ground without any snapping (rust example : foundation) it will create a group. Whenever you snap anything to any object inside the group it will add the object into that group. That way the system can have a way to get all buildings parts of a specific building.

*** Why do we need this and how does it help us :** Building groups is one of the most important things inside the building system because without having building groups you wont be able to identify a building and therefor you wont be able to do stuff like batching or area of interest and that's why its important to have it inside the system. By using groups you can access all the parts of the building and with the API of the system you can make custom stuff with it such as ownership or make all buildings protected or any thing really.

Area Of Interest :

*** What is area of interest :** Area of interest or AOI in short for you to save performance and stop yourself from reaching the colliders limit by disabling distanced sockets, conditions and colliders. The system is fully multi-threaded and can be easily used and extended by any one.

*** How do I use it :** It is currently set up inside the cubes demo, it is very easy to set it up to your own needs as well.

https://www.youtube.com/watch?v=PL8HVwGvXPE

Using Batching :

*** What is batching :** Batching is a feature inside the system that allows you to batch all building renderers inside 1(or more, depends on the vertex count) renders. This will decrease your draw calls significantly and improve performance.

*** How to use Batching :** uConstruct automatically does batching for you. Although for each building you want to be batched you need to enable the "Batch Building" option on the BaseBuilding componenet.

uConstructPhysics :

*** Whats uConstructPhysics :** uConstructPhysics is a custom built physics system that has a custom raycast method. The reason we created this is because we don't want each socket to have a collider so we made our own raycast system that lets you have a socket without a collider and it will still detect a hit with the socket.

*** How do I use it :** In order to use uConstructPhysics you call the method : UCPhysics.Raycast/RaycastAll and you will see all the possible options for that method(Please note that this class is in the "uConstruct.Core.Physics" namespace), you can also look in the BuildingPlacer demo script and see how I used the library there.

*** How does this help the asset performance :** This helps the asset performance because as of before, each building that had socket would have a collider for each socket, so buildings like foundation would have 15~ colliders each!, imagine you have 4-6 of them in a scene! That's insane!, that's why I created this raycast system so you do not need colliders on sockets and it is way more performant now, it still has all the raycast hit features such as normal, hit, transform and distance and also if you do not find any collider it will look into the normal physics engine of unity so for example you can still hit the terrain from this method as usual.

CallbackManager :

The system manager (UCCallbackManager) is a script which initialize all needed unity callbacks.

In order to have your system working properly you must call:

```
uConstruct.Core.Manager.UCCallbacksManager.CreateAndInitialize();
```

At the awake method. This will initialize the CallbackManager and create an instance of this on the scene.

The CallbackManager handles loading & saving of the buildings. It also handles all unity callbacks methods.

PrefabDatabase :

The system has a class which handles all the prefabs of the system. This system allows you to have a data of all buildings in the game. By using an generated UID you can grab a GameObject of a building. This helps you when creating a system that requires a prefab and you don’t have access to it.

In order to grab a prefabID of a building all you need to do is :

```
TargetBuilding.prefabID;
```

Where "TargetBuilding" is your target building script.

In order to grab an GameObject from a prefabID, use this method :

```
uConstruct.Core.PrefabDatabase.PrefabDB.instance.GetGO(prefabID);
```

Where "prefabID" is your target building prefab id.

SavingManager :

uConstruct has a built-in saving manager that handles saving of all the buildings in the game. The system is extremely powerful and can be easily extended, uConstruct automatically comes with an enabled saving system so you don't need to edit a thing.

A tutorial about how to extend the saving system and add your own functionality :

https://www.youtube.com/watch?v=7OaRHr515jw&feature=youtu.be

Templates :

Templates are data files that contains data about certain conditions and sockets you choose. By using templates you can have certain amount of buildings have the same certain conditions and sockets and whenever you edit the templates they will be updated on all of the prefabs.

This can help when setting up different tiers of a certain building or having a pre-set condition pack that you want to have on each building and don't want to create them individually on each building.

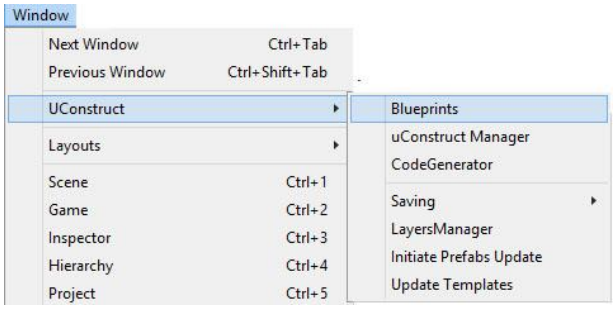
Watch:

https://www.youtube.com/watch?v=UDLDXbDp76M

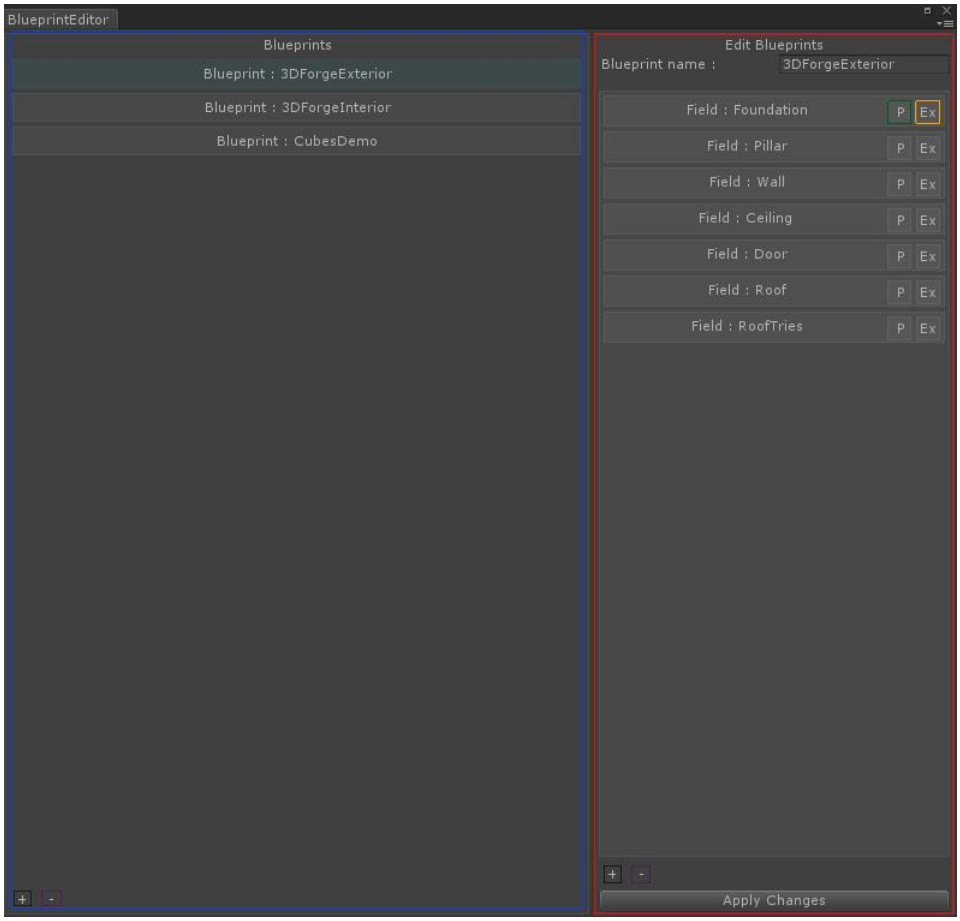
Blueprints :

Blueprints are set of data that are saved on a file in your project, using the data you can get any mesh or a prefab to adjust to that data and magically become a complete building with all of the sockets and conditions, correct size and it will also take into consideration not-centered pivots and fix them automatically.

In order to get started, open up the blueprints editor window :



After you click that button, this window will pop up :



Blueprints zone – This area will contain all of the blueprints you have, you can choose each one and edit in the "Blueprints Editor Zone".

Blueprints Editor Zone – In this zone you will edit the blueprint's name, add, delete or edit its fields, pack or export fields or "Apply Changes" to save all of the changes.

Pack a data from any building into a field, this will allow you to use this data on another mesh/prefab to fully set It up.

Unpack the stored data inside the field into any mesh or a prefab, adjusting it accordingly and setting it up ready to use instantly with no further efforts, it will also fix any pivot issues of the mesh or prefab.

Create a blueprint or a blueprint field.

Delete a blueprint or a blueprint field.

The system will automatically handle scales, pivots and anything else automatically for you so you don't need to do anything to get the building to work!.

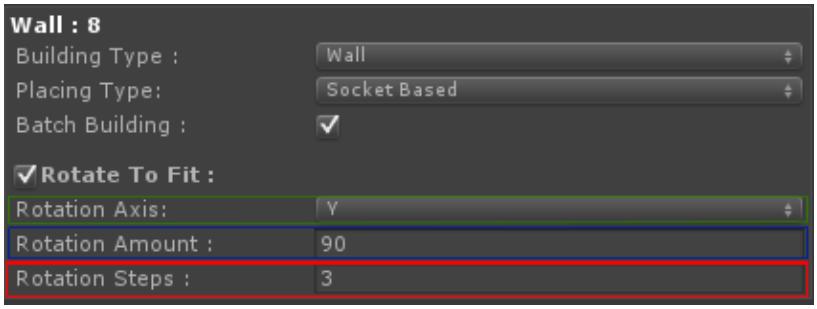
In order to get a better grip at the blueprints feature you could also watch this tutorial :

https://www.youtube.com/watch?v=-er8iz-HECY

Rotate To Fit :

Rotate to fit is a feature that automatically rotates the building into the socket even if the mesh isn't rotated properly, This only works on snap placement so In order to use it your building must be placed by snapping.

Open up one of your buildings and you will see this in the building script :



The axis that the system will try to "Rotate to fit" the building on.

The rotation amount, in this instance its 90 so it will try each side of the wall.

How many steps will it try in order to get it to work ?, so in this instance it will try adjusting 90, 180 and 270, the more steps you add the more accurate it is but the more expensive it is.

Snap Points :

Snap points is a fairly new and unique feature which allows you to basically use any shape with the asset, what does that mean?

When using complex shape with the asset currently will result you in having wrong anchoring and when rotating or snapping, the scale/rotation/position might be wrong.

And that's where the snapping points becomes handy, the snapping points will modify the anchoring of the position, allowing you to have any kind of shape working with the system properly, while allowing snapping to differently sized sockets and the snapping will still work the same!.

How do I use it ?

Using the snapping points is quite an easy process, so let's get started!

First of all, you need to understand how the snap points work:

(Sorry for programmer art)

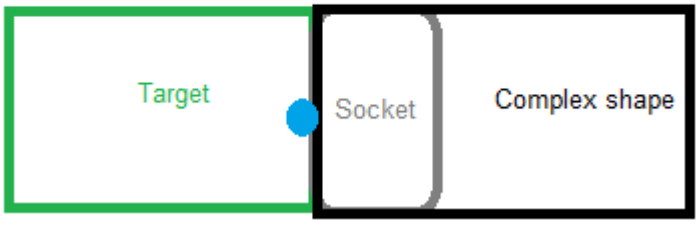
So as you can see, we have the green box (Our target building, the building we are going to snap to), a black box which is the building we are placing and we have the gray box which is the socket.

You can also see we have a blue dot in the Target building, that dot is our snap point. So what will it do in this scenario?

Here's a comprehension :

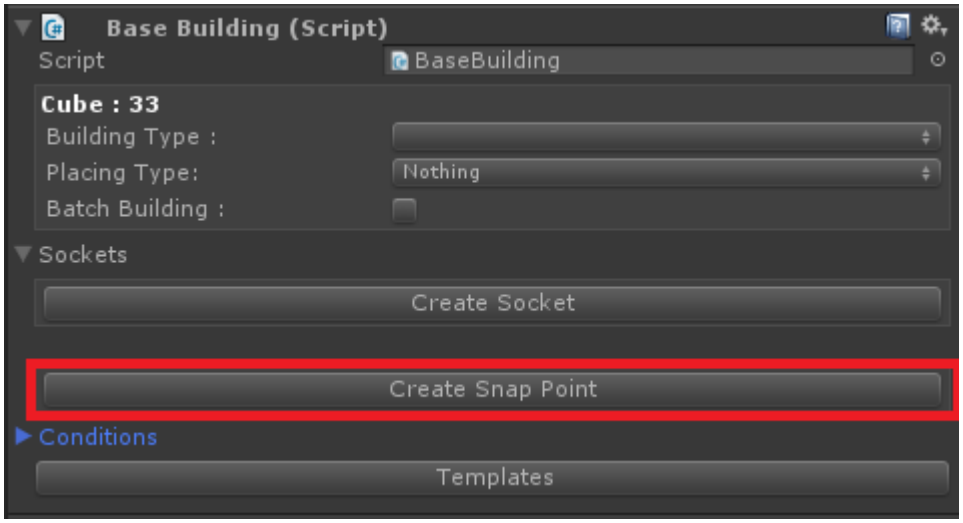
Without snap points

With snap points

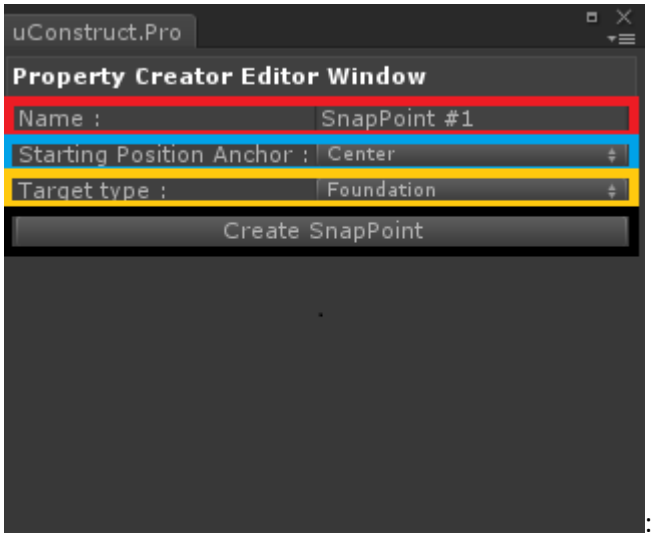


As you can see when using the snap point, it properly handles any kind of shapes, any kind of sizes and allowing you to have more freedom when making structures ☺.

Simply, to use this feature you would want to open up the placed-on building (in our case the Target), and open up the Sockets tab, go down and you will see this:



Click on the button and you will see this window opening up



:

RED – The name of the created snap point (Can be changed).

BLUE – The position on where it will be created (A tool to help you place the snap point, isn't a must).

YELLOW – What buildings will be snapped to this snap point?

BLACK – Create the snap point.

After you have created it, it's just a matter of playing with the position and that's it, the system handles the rest :)

API :

uConstruct has a document with all API methods in the system with summaries.

In order to access it, click on the "API" folder in the doc folder inside the asset and open the "Index" file. This will have all API in it.

----MORE----

Video Tutorials:

[Tutorials Playlist]