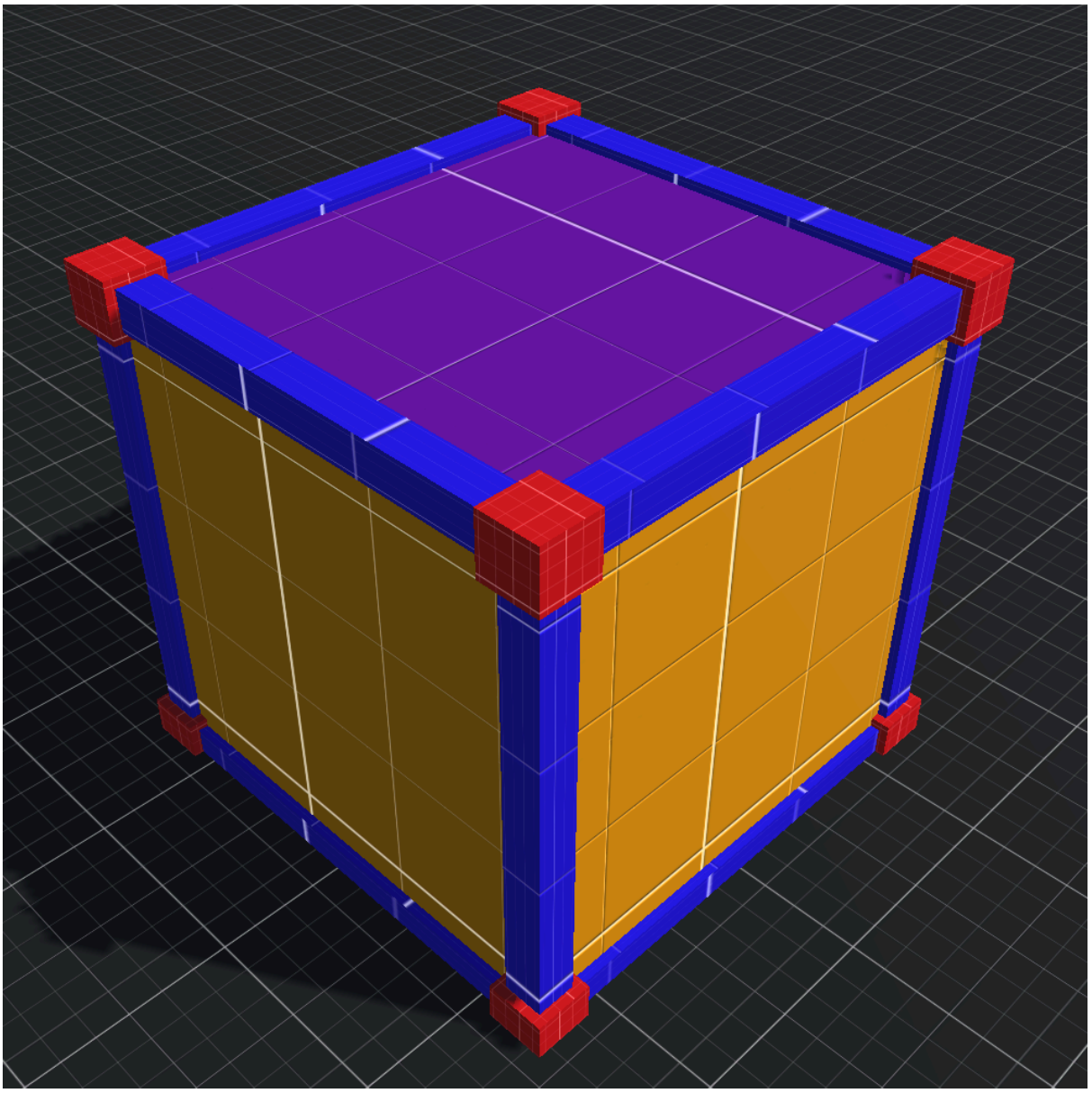
AvancedBuildingSystem

by Szaya



BuildingManager

BuildingManager is the main component of the Advanced Building System. The whole tool is around this script. Support the algorithms and manage the settings.

How does it work?

You as a developer should add a BuildingElement to the BuildingManager through the **IBuildingManagerExternalInterface**. Then the Manager immediately starts to work based on the settings.
Every frame if the BuildingManager has a BuildingElement will make a raycast from the given camera and the manager tries to find the best position to build the BuildingElement.

Features

The BuildingManager has the following features:

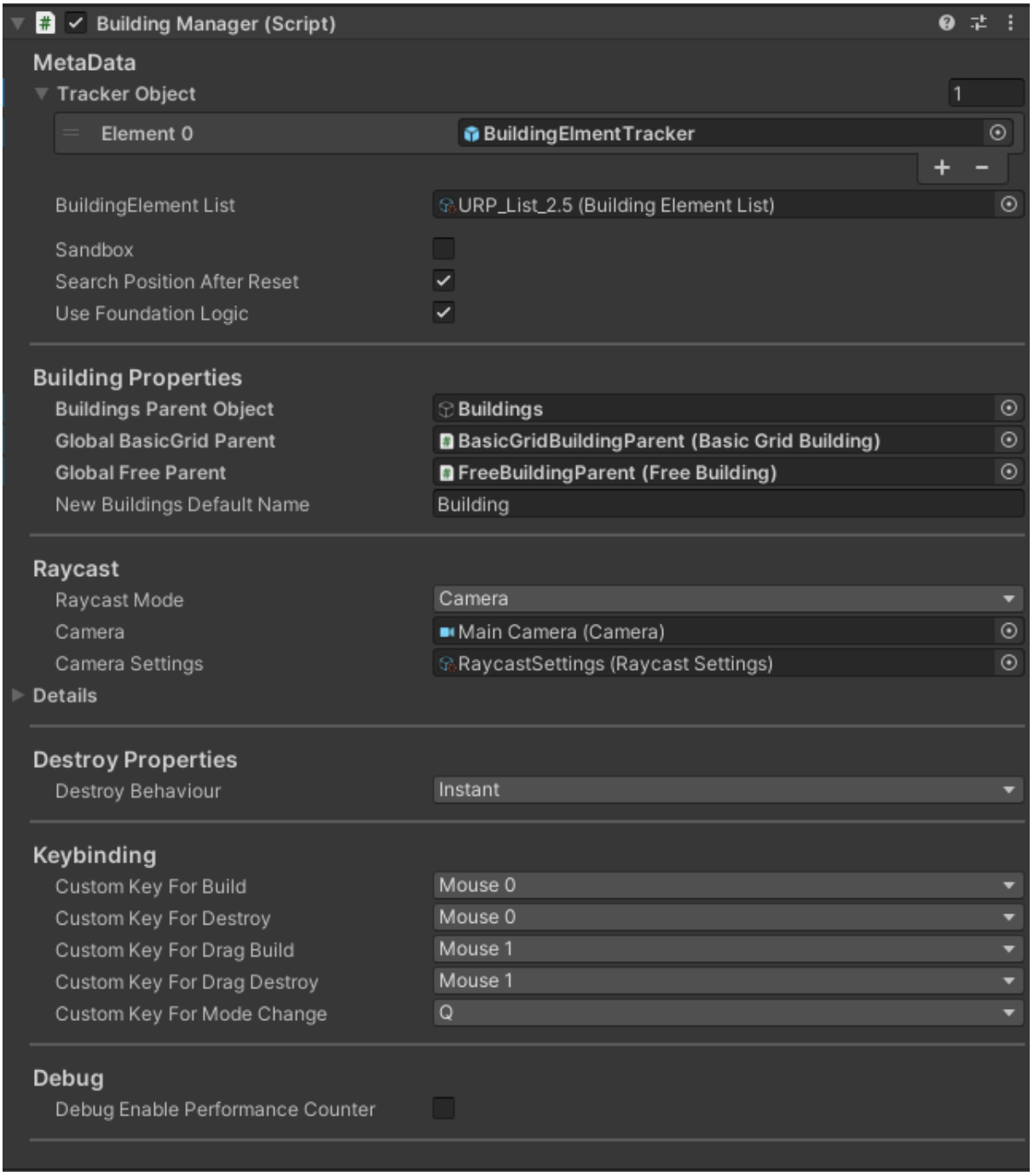
- Main features
  - Simple Building
  - Drag Building
  - Simple Destroy
  - Drag Destroy
- Different building algorithms
  - Free Building
  - Basic Grid Building (2D grid snapping in 3D space)
  - Advanced Grid Building (3D grid snapping)
  - SnapPointBased **(Experimental!)**
- Customizable settings
  - Position, Rotation, Raycast, Layers, Sizes etc
- Custom Keybinding
- Prebuilt Element feature
- Area features
- Foundation logic
- BuildingElement modifications
  - Override already built element
- Blocking elements feature
- Highlight material feature
- Tracking / API (Custom event system)
- Gizmos
- Persistency (Save/Load)
- Debug (Currently only performance measurements)
- Support
  - NavMesh
- Developer Tools

Setup

The setup of the BuildingManager is separated into smaller parts to make the possibility to use different setups for different scenarios and make it as customisable as possible.
The setup are built from the following parts:

- BuildingManager Settings
    - Properties of BuildingManager (BuildingManager's GUI)
    - Raycast Settings (ScriptableObject)
    - BuildingElementList (ScriptableObject)
    - Tracker Object (implementation of BuildingManagerTracker)
- BuildingElement
    - Properties of BuildingElement (BuildingElement's GUI)
    - BuildingElementMetaData (ScriptableObject)
    - BuildingAlgorithms Settings (ScriptableObject)
        - OverrideElement Ruleset (ScriptableObject)
        - SnapPointRelationship (ScriptableObject)
        - SnapPointRelationship List (ScriptableObject)
    - HighlightCollection (ScriptableObject)
- Area
    - Properties of BuildingArea (BuildingArea's GUI)
    - BuildingArea Settings (ScriptableObject)
    - BuildingArea Ruleset (ScriptableObject)

This chapter is about the “Properties of BuildingManager”. The other setup objects are explained in later chapters.
The BuildingManager’s settings are explained later. Now just in short the BuildingManager has the following settings:



MetaData

- Tracker Object
  - A list of the tracker objects that should inherit from the BuildingManagerTracker.
  - With those objects you get the possibility to
    - implement custom functionalities

        - react for custom events
        - make decisions and control the behavior of the BuildingManager.
- BuildingElement List
    - A list of BuildingElements.
    - You can add BuildingElements for the Manager with index if it is contained by this list.
- Sandbox
    - It is a boolean.
    - If it is true the BuildingManager will ignore the Maximum Amount of temporary elements so it can place an infinite amount of elements at the same time with DragBuilding.
- Search Position After Reset
    - It is a boolean.
    - If it is true the BuildingManager will perform a position search if the BuildingElement is changed to ensure that if the BuildingElement's Update function is disabled (for example the time is set to 0) the reset of the BuildingElement is fully finished.
- Use Foundation Logic
    - If it is true the BuildignManager allows to build that element by itself what is signed as Foundation.
    - But snapping to an already built lament can be done with any BuidlingElement.
    - it is only working properly with those algorithms what using snapping logics like AdvancedGrid or SnapPointBased

Building Properties

- Building Parent Object
    - The GameObject in the Hierarchy what will be the parent of the Buildings what created by the BuildingManager
- Global BasicGrid Parent
    - The BasicGrid algorithm has a global logic. It means that the BuildingElements will not belong to different structures. Every BuildingElement belongs to one Building.
    - Every BuildingElement is created under this Object
    - It should have a BasicGridBuilding script as a component.
    - It can be null. If it is null the the BuildingManager will create one under the Building Parent Object
- Global Free Parent
    - It is the same as the Global BasicGrid Parent just for the FreeBuilding algorithm.
- New Building Default Name
    - The default names of the newly created Buildings.

Raycast

- Raycast Mode

    - A Camera or the cursor should be used for raycast.

Destroy Properties

- Destroy Behaviour
    - How should they destroy work? It is explained later.

KeyBindings

Here you can customize the keys for the actions of the BuildingManagers.

Debug

In this chapter of the settings the debug function can be enabled and customized.

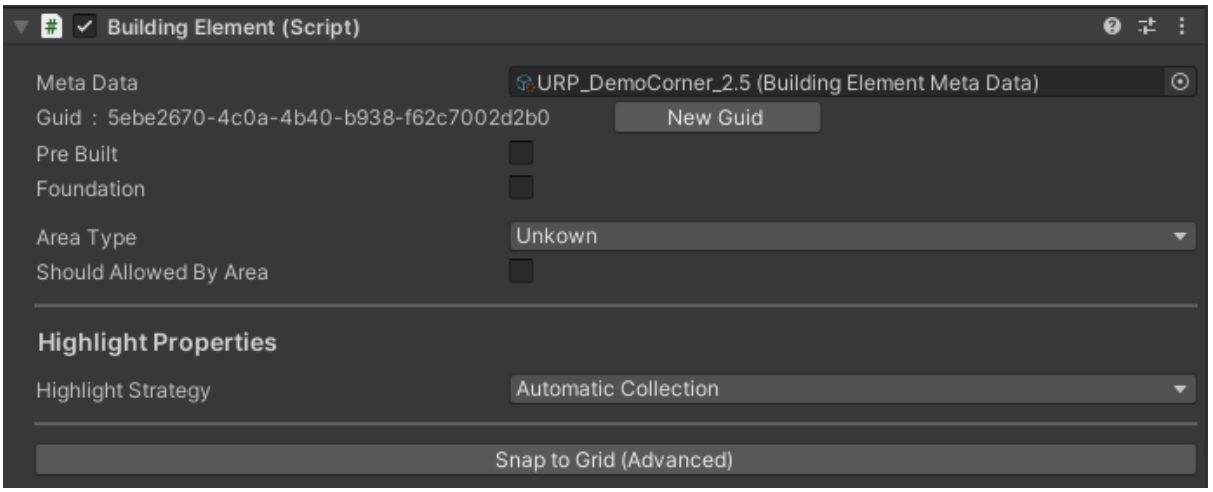
BuildingElement

The BuildingElements are the pieces of the buildings and structures. Also it can be furniture like a table or a chest. It can be a tree or a turret. It's just on you and you project.

You need to make your own BuildingElements for your game. You can do it by yourself or you can use the BuildingManagers custom tool BuildingElementCreator which will be discussed later.

Every BuildingElement should has the following components:

- Collider (it’s depend on your object)
- BuildingElement (Script)



The BuildingElement script has just a few parameters because Most of the properties of a BuildingElement is stored in a BuildingElementMetaData.

MetaData is a collection of properties for BuildingElements. Multiple MetaData can be created and one MetaData can be used for multiple BuildingElements for making it easier to set up similar elements that should have the same properties and the BuildingManager should handle these elements in the same way. Like 2 different walls should be used in a same way so the MetaData can be shared but some specific property can be set directly on the BuildingElement.

The Guid parameter is used for identifying a BuildingElement. The Guid is important to be unique for every different element. The “New Guid” button can be used for generating new unique guid. For example if you copy a BuildingElement the properties of the script will be the same just like the Guid. So the algorithm will think that these are the same BuildingElement based on the Guid. To fix this issue if you copied the BuildingElement regenerate the Guid.

The PreBuild property is an important parameter that can not be set by default so if you are adding a BuildingManager to the scene you should set this property by hand. It is a boolean which means that the BuildingElement should be handled as a PreBuilt element or not. The PreBuilt Element is explained in detail later in the documentation.

Foundation is also a boolean. The BuidlingManager's has the "Use Foundation Logic" property and if that is true that elements what has this Foundation property set to true can be built without snapping to an another BuildingElement.

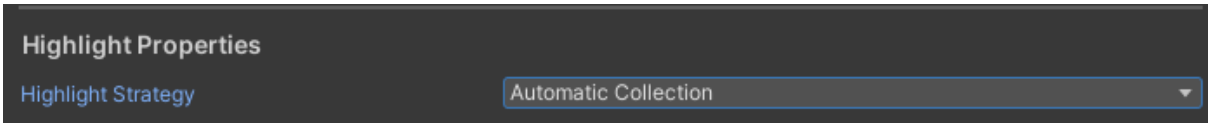
AreaType is used for Area features that will be explained in detail later. With this enum you can set the type of the BuildingElement for the Area feature.

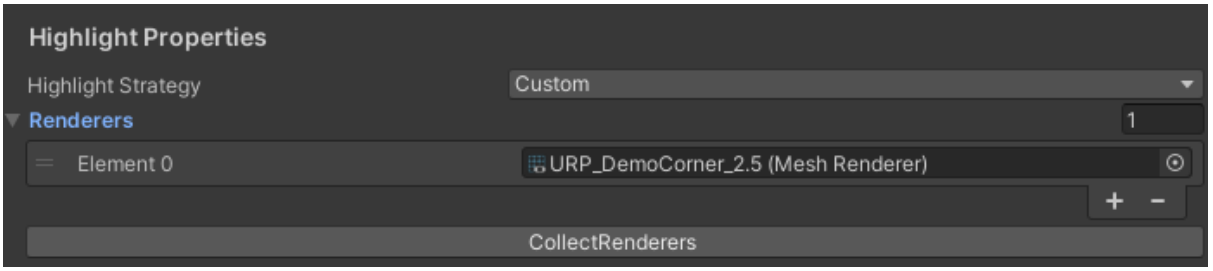
"Should Allowed By Area" means that if it is true the BuildingManager will block the building of that element until any of the BuildingArea doesn't allow it.

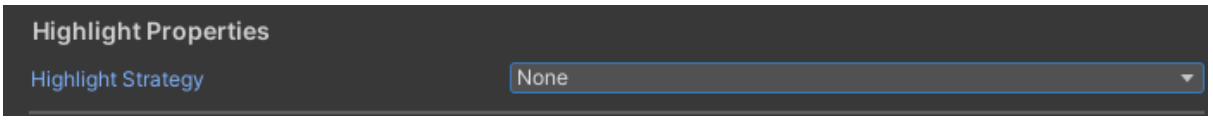
Highlight Properties

The BuidlingManager uses a highlight feature that will change or modify the materials of the renderers of the BuildingElement. It is used to highlight the different states of the BuildingElements like blocked or pre built etc.

You can select front he following 3 behavior:







None: There will be no highlighting. The Building Element will get the event and change its states but doesn't change the materials at all. If you don't want the highlight feature just set this property to None.

Automatic Collection: The BuildingElement will collect the rendered from its game object and its childs and use the highlight logic so it will modify the materials of the renderer. Note that this will collect every renderer and change these materials.

Custom: It is basically the same as the Automatic Collection. The Building Element will use the highlight feature and it will change the material of the rendere just in this time you can set by hand which renderers should be used. To make it easier to collect the renderers of the GameObject the CollectRenderers button can be used for automatically collecting the renderers just like the Automatic Collection. Just in this time you can modify the list of the elements by hand.

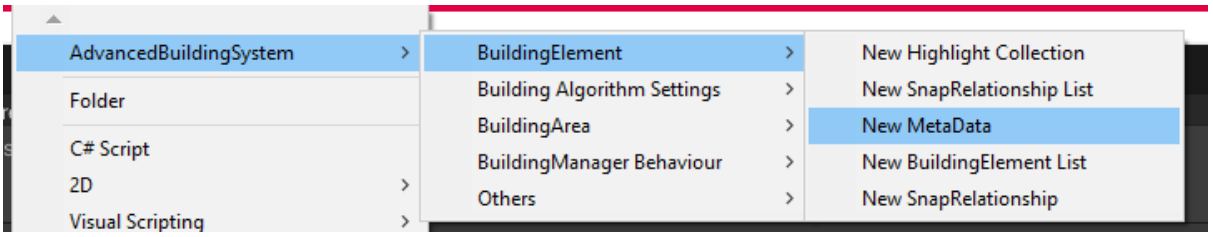
BuildingAlgorithm Specific properties

“Snap to Grid (Advanced)” and “Snap to Grid (Basic)” Buttons are the only available algorithm specific options of the BuildingElements. These 2 buttons can be used during the scene building process when the developer is placing the BuildingElements by hand and he/she wants to ensure that the elements are snapping to the grid. Just place the BuildingElement to the scene and press the button. It will modify its transform to snap into the grind. Note that this feature is only working if the BuildingElement's parent object has the proper type of Building script as component.

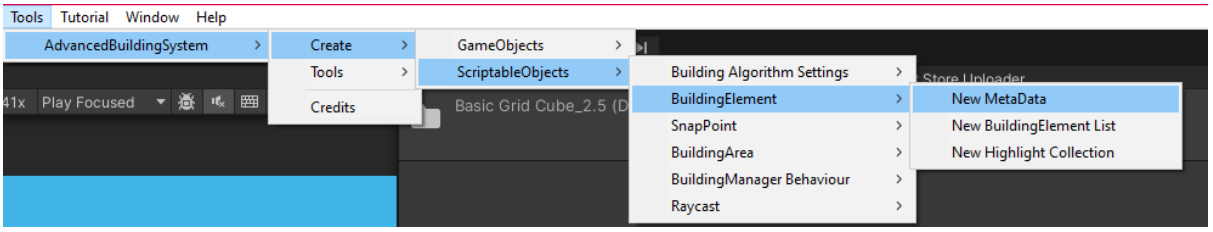
BuildingElementMetaData

For easier setup of the similar BuildingElements that should be handled by The Manager in the same way most of the properties are separated from the element objects into ScriptableObjects.

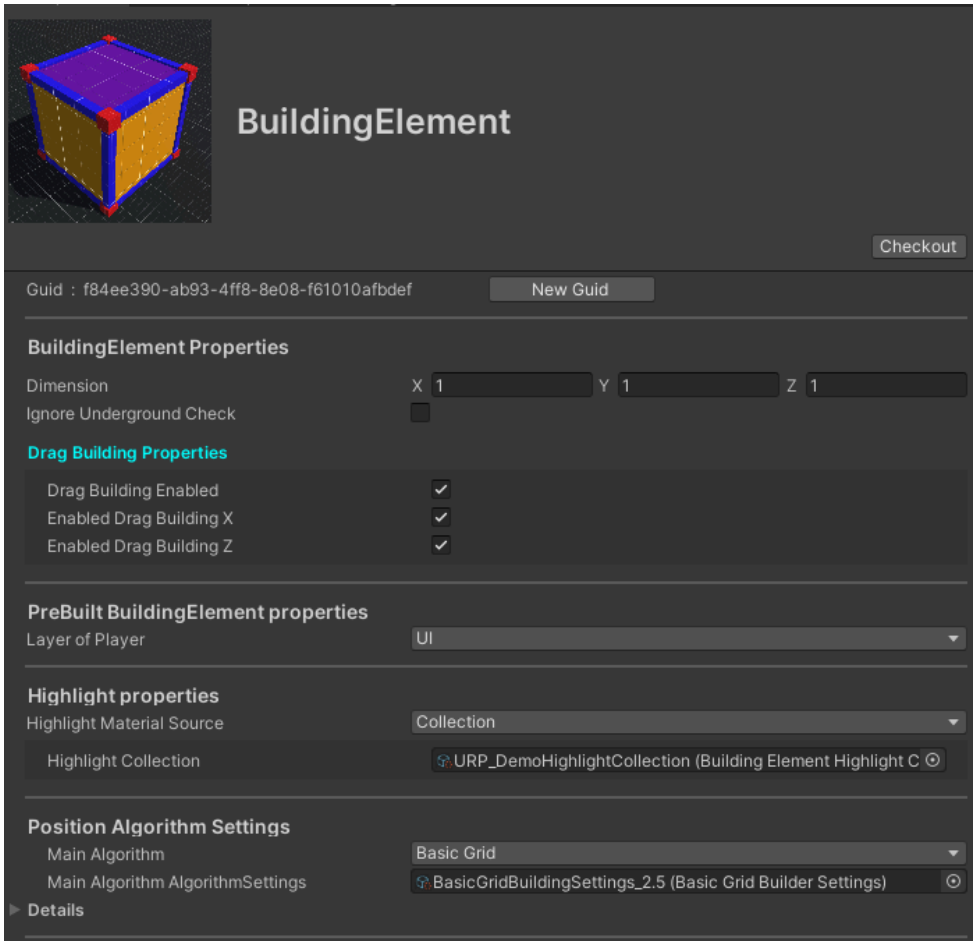
To create a BuildingElementMetaData right click on the Project view:



Or you can create it from the menu bar:

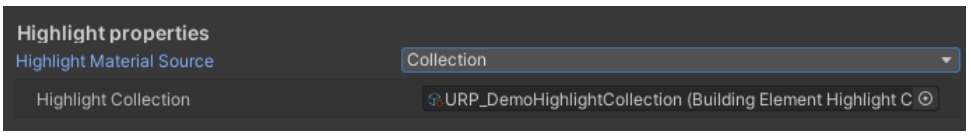


BuildingElement MetaData GUI:

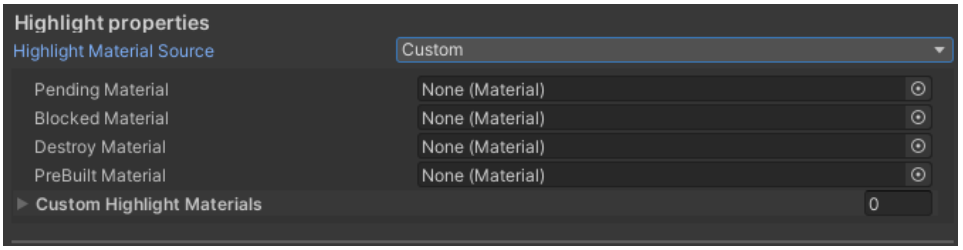


The MetaData has the following properties:

- BuildingElement Properties
    - Guid
        - Can not be set. (It is set automatically when the MetaData has been created)
        - It should be unique.
        - It is used for identifying the BuildingElements with the same MetaData. It is different from the Guid of the BuildingElement's script. That Guid is for identifying the concrete element.
        - If you copy the ScriptableObject ensure that the Guid has been regenerated with the "New Guid" button.
    - Dimension
        - It is used as the size of the BuildingElement. Ensure that it has been set properly because all the calculations are based on this Vector3.
        - For example if you are using BoxCollider for the BuildingElements use the same size for the Dimension as the Vector3 used for the BoxCollider.
    - Ignore UnderGround Check
        - The BuildingManager is trying to avoid placing the elements under the ground where the element can not be seen. If you set this boolean the BuildingManager will allow you to place this element under the ground.
    - Drag Building Properties
        - DragBuilding is one of the main features of the BuildingManager. It is about placing multiple elements at the same time. The DragBuilding is explained in detail later on in the Drag Building chapter.
        - Here you can set up how the DragBuilding algorithm should handle this element during the drag building process or you can just disable the drag building feature on this element so the BuildingManager will not allow the DragBuilding with this element.
- PreBuilt BuildingElement properties
    - Layer of Player
        - The layer what objects can be used for PreBuilt feature which will be explained in the PreBuiltBuildingElement Chapter
- Highlight Properties
    - The Materials used for highlighting the BuildingElements in different scenarios during the BuildingProcess. It is not commonly set on the BuildingManager because you can set it for every MetaData to make it possible to set different highlight material for different elements.
    - The effect on the BuildingElements are partially explained during the BuildingElement properties where you can set which renderers should be affected by the highlight feature. Here you can set the materials for that feature.
    - You have two options:
        - Collection: You should provide a HighlightCollection ScriptableObject which is just a collection of materials for different states for the BuildingElement.



- Custom: You should provide the materials right on the MetaData ScriptbaleObject.

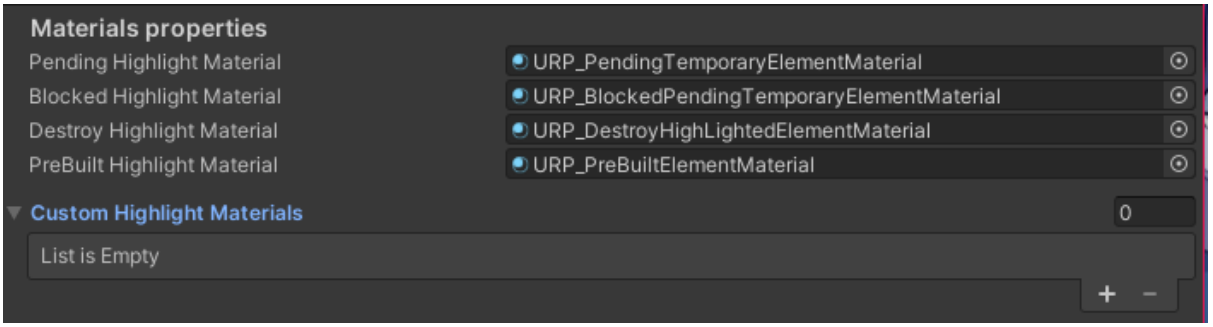


  - What's the purpose of the Custom Highlight Materials?
    - You can set a list of materials that should be used as highlights on the BuildingElement in different scenarios that are independent from the AdvancedBuildingSystem but you want to achieve the same highlight mechanism as the built in scenarios.
    - For example you want to outline the BuildingElement if your cursor is above an element. You should add the material in the list and call the **SetHighlightMaterial** on the BuildingElement with the index of the Highlight material. In this case the index of the outline's material.
- Position Algorithm Settings
  - Every BuildingElement can be built in a different way using different algorithms. For every kind of BuildingElements used in your project you can set different building algorithms.
  - Some algorithms need a fallback algorithm that has been set.
  - Every algorithm needs a setting that can be created as the MetaData (ScriptableObject) Different settings can be created for every MetaData to make it possible to use different build logic for different elements.

Highlight Collection

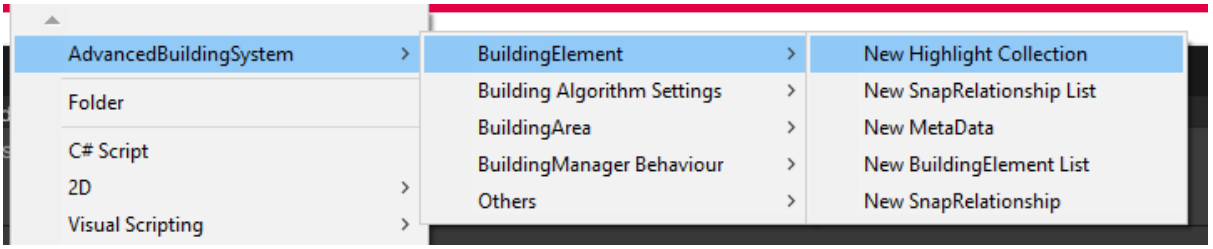
The HighlightCollection is just a list of the materials that should be used for highlighting the BuildingElements in different states. And also a list of different materials for custom highlighting the BuidlingElements.

This custom highlight feature can be done by calling the **SetHighlightMaterial** function on the BuildingElement script and providing the index of the material from the “Custom Highlight Materials” list.

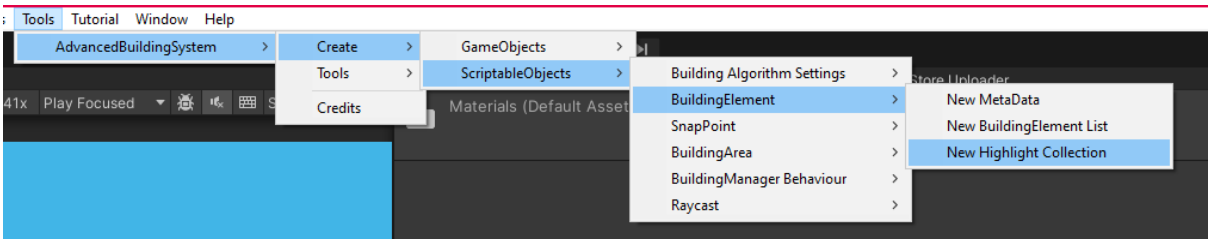


The Highlight Collection can be created just like every other scriptable object used by the BuidlingManager.

From the project:

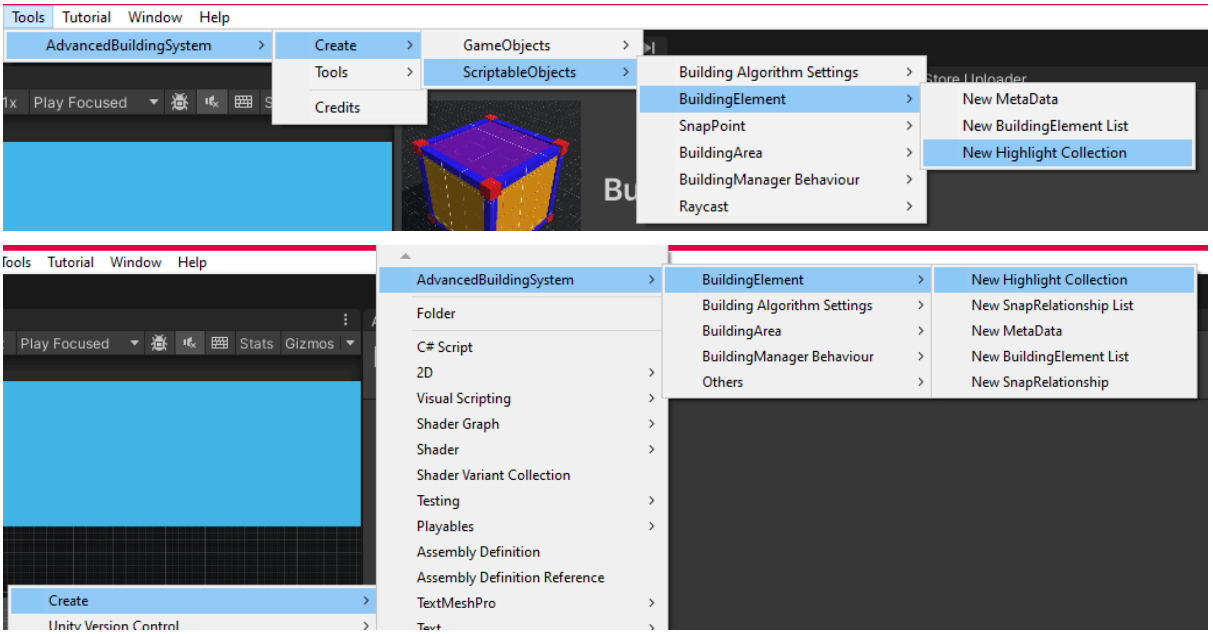


From the Menu:



BuildingElementList

Currently the BuildingElementList is just a list of BuildingElements. It should be provided to the BuidlingManager. The manager will use it to get the BuildingElement GameObjects to instantiate the elements.
When the BuildingManager is activated through the IBuildingManagerExternalInterface interface an index of the BuildingElemenet from the list should be provided to the Manager. The list can be changed during the game using the **SetBuildingElementList** function of the earlier mentioned interface.
Creating the List can be done from the Menu or the project Create option:



Building

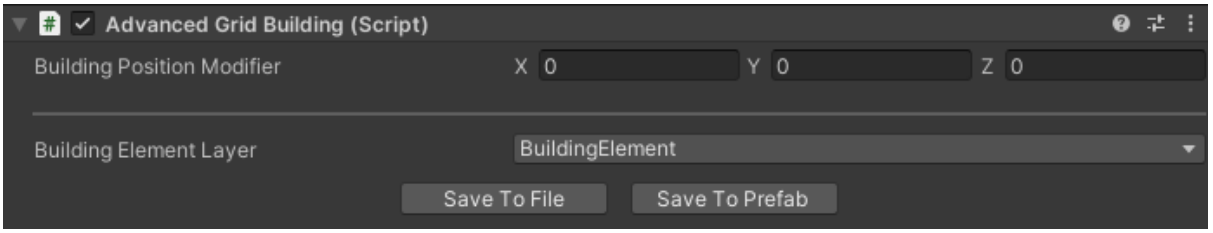
The Buildings are the structures that are built with BuildingElements. It is used for collecting and counting the BuildingElements and it is used for some algorithms' calculation. Because of that every building algorithms' has different needs every building algorithms has its own unique Building

- AdvancedGridBuilding
- BasicGridBuilding
- FreeBuilding
- SnapPointBased (Experimental!)

All of them are inherited from the basic Building class which is inherited from the MonoBehavior.

Every BuildingElement should be under a GameObject with Building script in the Hierarchy and every object under a Building should be a GameObject with BuildingElement script.

The Buildings have basically zero property but it has some just to ensure that if you make a prefab from a Building the required properties are saved into the prefab so these should be serializable.



It is not Recommended to build an AdvancedGridBuilding by hand. Build it in a test scene and use the Save To Prefab button on the Buildings' inspector view. This feature is available on every Building type.

Also you can save the Building into a file with the Save To File. It will save a json file with the save data of the Building that is used during the save/load processes. You can use the BuildingLoader tool for loading a building from json data.

The FreeBuilding and the BasicGridBuildings and global Buildings which means that every time you place a BuildingElement with FreeBuilding or BasicGridBuilding anywhere in the world the element will be created under these objects. It will be explained later. The AdvancedGridBuilding is different because everytime when a new Building is started in the world a new AdvancedGridBuilding is created to ensure that only that elements are under a Building that belongs to that Building.

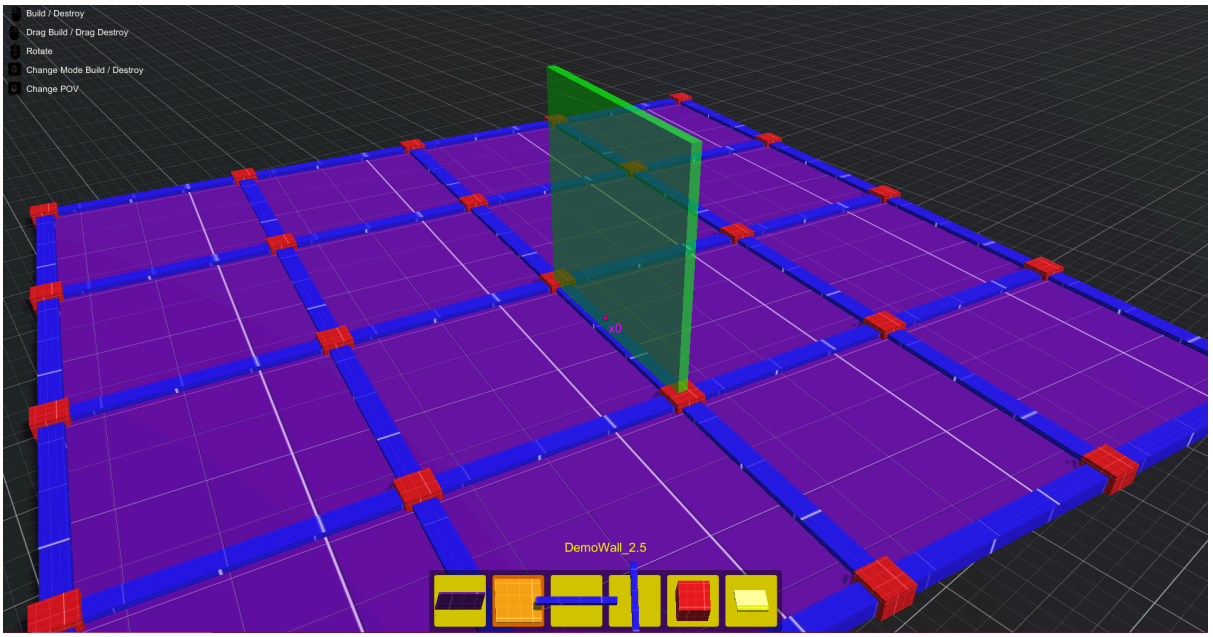
Main Features

SimpleBuild, DragBuild, SimpleDestroy, DragDestroy

These four are the main features of the BuildingManager. These features are required for giving the developers a possibility to place and remove BuildingElements in the scene.

Simple Building

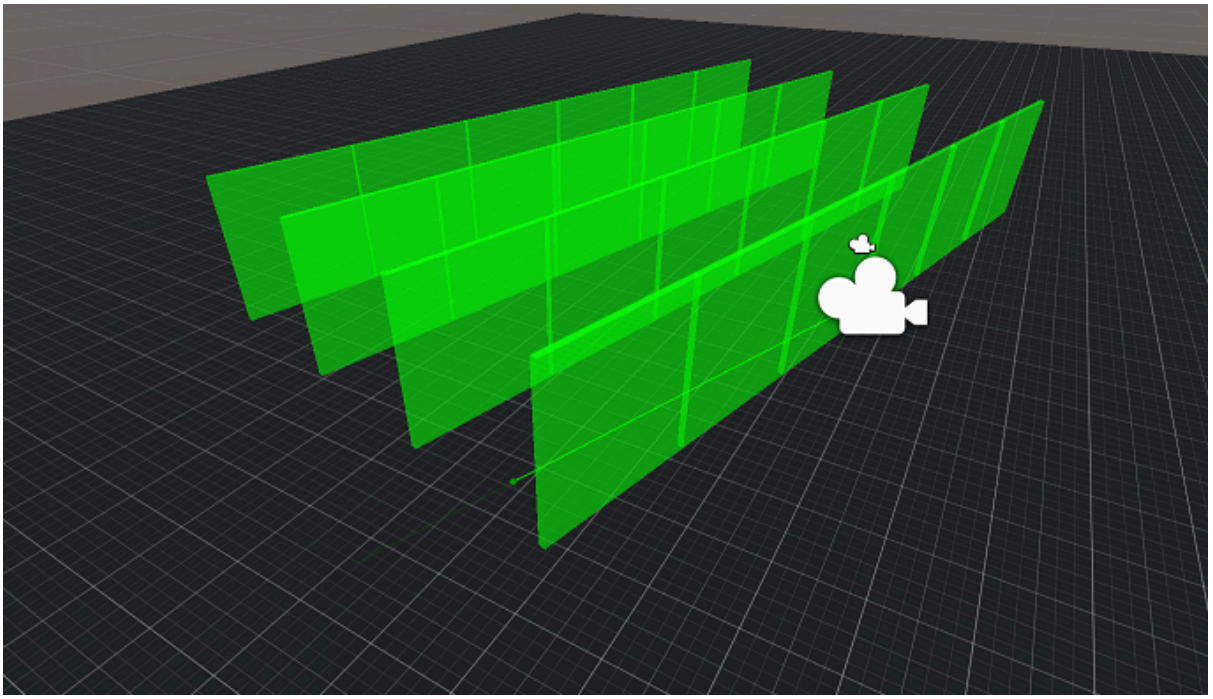
The main goal of the BuildingManger is implementing an in-game modular building tool for developers. What does it mean? The BuildingManager provides an algorithm that the developers can use to make games where the player can build such things in games like buildings, houses and other Structures.
The SimpleBuilding feature is responsible for that the players are able to build a structure or anything as part by part like walls, floors, roofs etc. The SimpleBuilding can only build one element at a time.
During the update after the raycast the BuildingManager tries to find the best position to build the BuildingElement. if it is possible the position search algorithm gives a position based on the specifications where a temporary element will be placed. If it is good to the player the element can be placed permanently there.

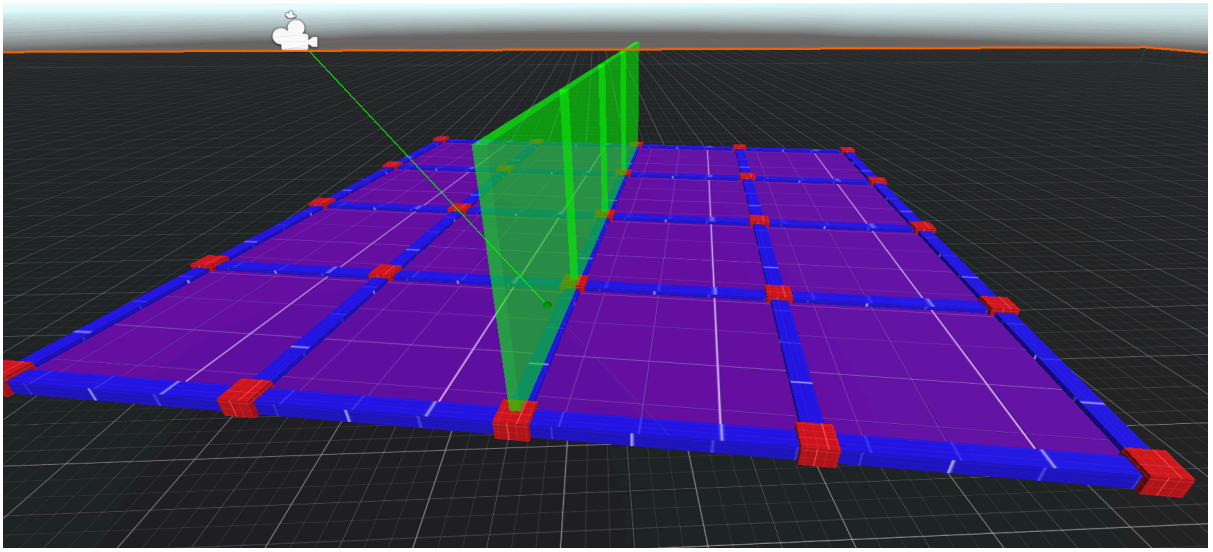


Drag Building

Building a massive structure piece by piece can be very slow. For making the players' life easier the BuildingManager provides a DragBuilding functionality that is responsible for placing multiple elements at once.
During the Update when the position is found by the SimpleBuilding. The player can build the element in a simple way with the left mouse button but if the player holds the right mouse button the BuildingManager changes to DragBuilding mode.
During the DragBuilding mode the manager will use the raycast to know where the TemporaryBuildingElements should be placed. The manager will copy the elements next to each other to where the raycast points to. If the player releases the right mouse button every BuildingElements that have a valid position will be placed. The validity of the positions based on the settings.

Every Building algorithm behaves different ways during the drag building.

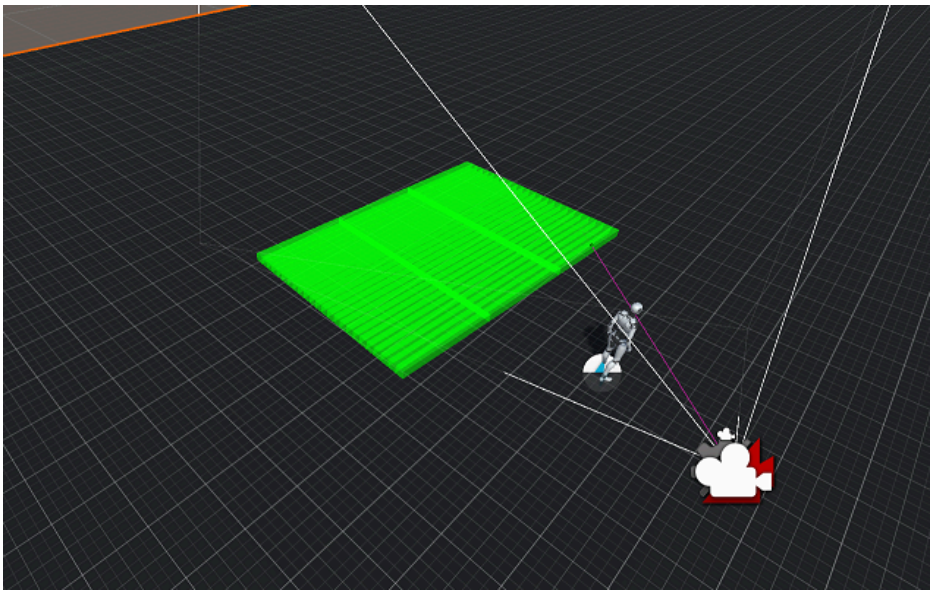




The algorithms will be explained in detail in the **Building Algorithms** chapter. Now let's just see the difference between the algorithms during the drag building.

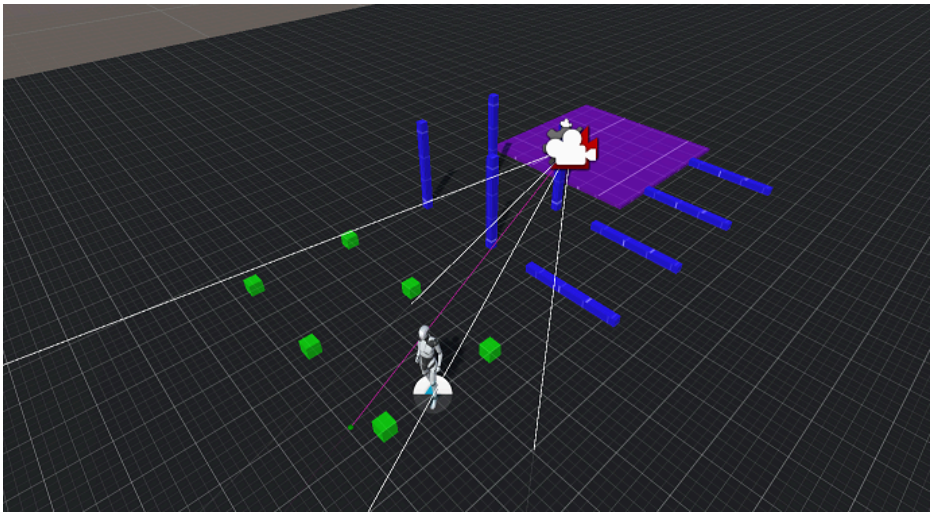
FreeBuilding

The FreeBuilding using the BuildingElment's Dimension for calculating the TemporaryBuildingElements' position.
In this example the BuildingElement's dimension is (0.2 x 0.2 x 2.5) what is placed with DragBuilding + FreeBuilding.

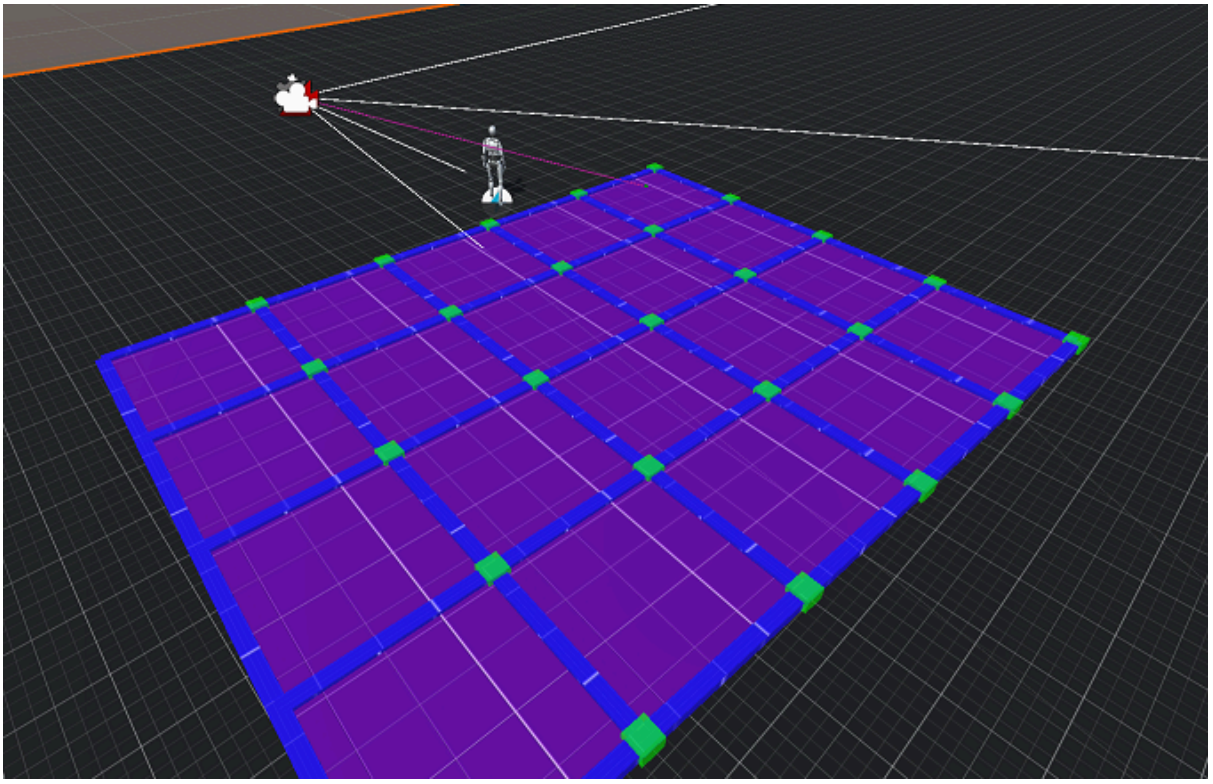


BasicGridBuilding

The GridBuilding is using the size of the Grid to calculate the TemporaryBuildingElements' position. The elements are placed in the middle of the grid's square.



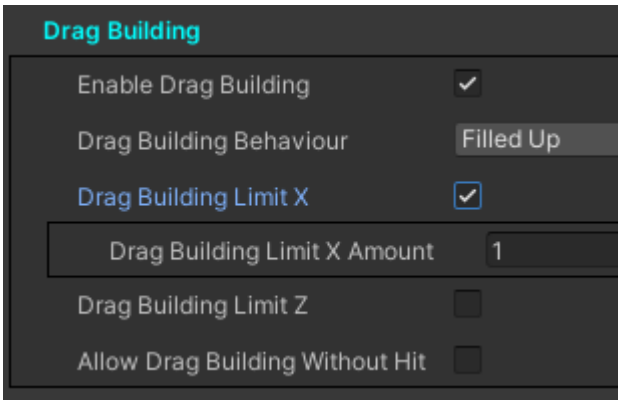
AdvancedGridBuilding
The AdvancedGridBuilding is using the grid's size but it is snapping the elements on different positions based on their types.



(Here the green cubes are placed with dragbuilding. These cubes are signed as corners so these should be snapped on the corner of the grids.)

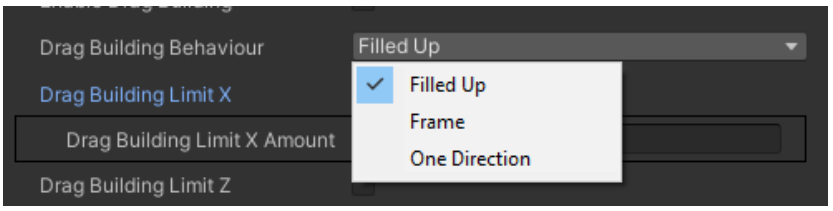
SnapPointBasedBuilding
Currently this feature is not supported by the SnapPointbasedBuilding algorithm.

Settings by the Algorithms



The DragBuilding's settings are simple and limited but do not forget that this feature is heavily affected by other features' settings.
First of all the DragBuilding feature can be deactivated entirely with the "**Enable Drag Building**". In case of that boolean is false if the player presses the set button (Mouse's right button by default) for the DragBuilding the BuildingManager just ignores the button pressed event.
You have the possibility to ban one of the axes for the drag building which means that the player will be able to drag using only the X or only the Z axis.
Or you can just Limit it to a maximum specific number. In the case of limited drag building even if the player has enough material he/she can only build up to the set limit.
Also if the "**Allow Building Without Hit**" feature is enabled then the drag building can be enabled/disabled separately which means that if the raycast does not hit anything and the BuildingElement is on the air can be placed there. But you can decide whether you allow the drag building in the air or not.
"**Drag Building Behaviour**" is an enum and with that you can change the behavior of the DragBuilding algorithm. Basically you can change the shape of the DragBuilding.

DragBuilding Behaviour

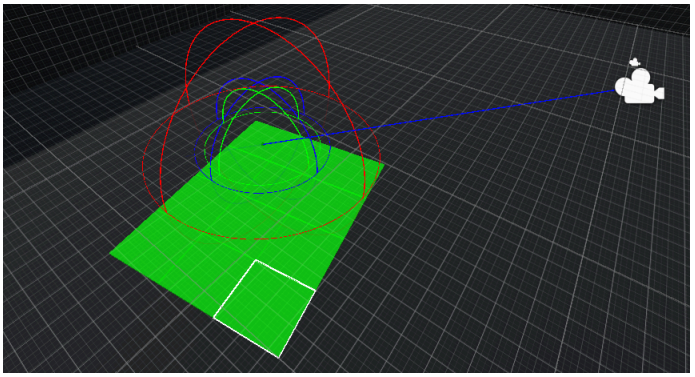


You have the following options:

1. Filled Up
2. Frame
3. One Direction

With these options you can change the shape of the drag building results.

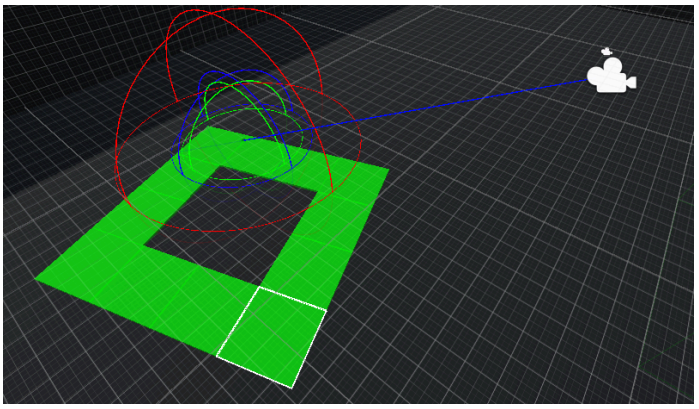
Filled Up:



With the Filled Up setup the algorithm will fill up the whole area. So every line and every row will be filled up with elements.

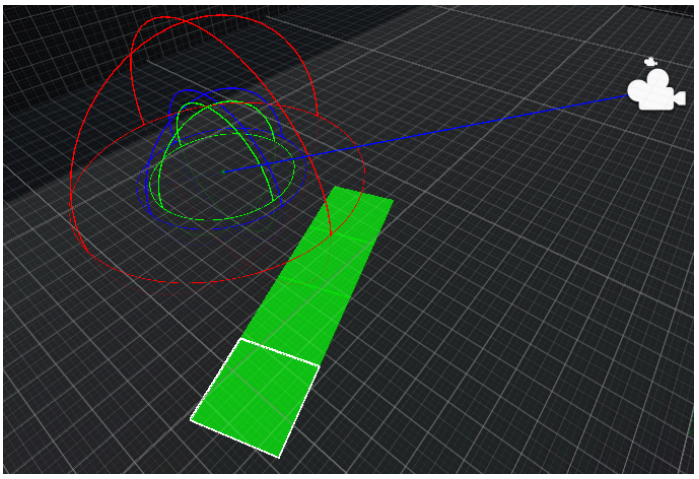
Frame:

With the Frame Setup the algorithm will place the elements only on the edge of the area. The inner elements are not placed.



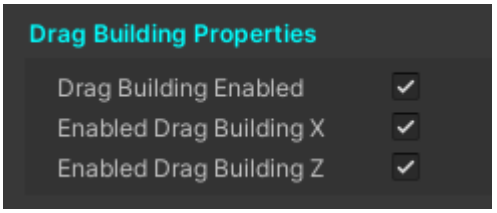
One Direction:

What is the difference from the One Direction setup and the limit of an axis? If you limit one of the axes then the BuildingManager will build only on the other axis. But with this setup you can build a direct line on any axes.
The algorithm will check the raycast's results and it will build the line on the X or Z axes which one is nearest to the raycast's hit point.
So with One Direction you can get a direct line on any of the axes.



Settings by the BuildingElements

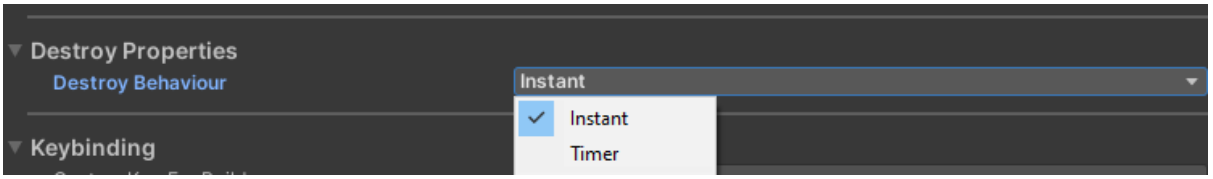
Also the DragBuilding's behavior can be changed by the BuildingElement too.
These Settings are available on the BuildingElement's MetaData. EveryMetaData can decide that its BuildingElements can be used for drag building or not. Or it can limit the axes too.



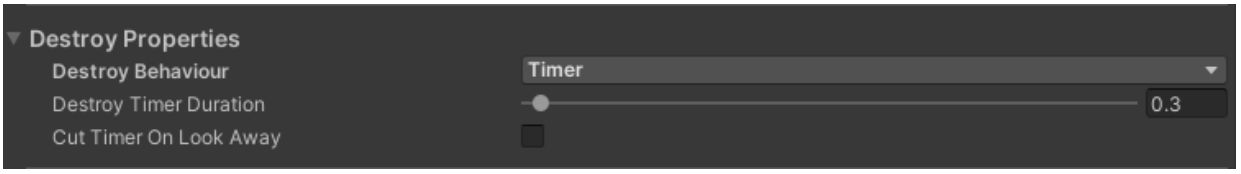
What does it mean? Even if you enable the drag building but the element will disable its possibility for drag building the BuildingElements can not be for drag building. Or if you limit one of the axes on the BuildingManager but the other one is limited by the BuildingElement the whole drag building will not work because all axes are limited.

Simple Destroy

The BuildingManager gives a removing feature too. It works in such ways when you change to Destroy mode with the "**Q"** button then the raycast will be used to track which element would be deleted by the player. The player looking to a BuildingElement will be highlighted and then with the mouse's left button the element can be deleted.
The destruction can happen in an instant way or it can have a little delay in what can be set up on the UI. Here the time should be chosen and then we can set a float as seconds how much time should be held by the player to destroy the element.

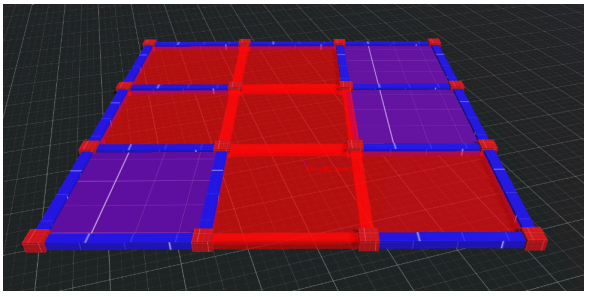


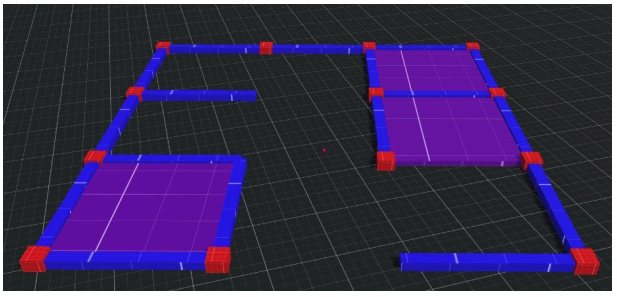
If the time is used then is it questionable the player should be looking at the element for the whole time during the set time? It can be set on the UI too. If the "Cut Timer On Look Away" property is set then if the player looks away from the BuildingElement the destruction is aborted. If it is unset then even if the player is looking anywhere else the destroy will be finished if the button is held for the required time.



Drag Destroy

The BuildingManager makes it possible to destroy more elements at one time. The player should hold the right button of the mouse to achieve this functionality. Every element will be signed for destruction during the time when the player holds the button. Every element will be signed what had been hit by the raycast. If the player wants to finalize the destroy the left button should be pressed too (or hold if the timer is set up).





Building Algorithms

What is a Building Algorithm?

The BuildingManager provides a variety of building algorithms. The main goal of the algorithms is the building. They are trying to find the best position for the new building element. Every algorithm has its rules and works differently. Some are simple and some of them are complex. There is a possibility that a position search algorithm can not find any valid position, then some of them can fallback and use a different type of algorithm that matches with its original logic to find the valid proper position.

Position Algorithms and their fallback algorithms:

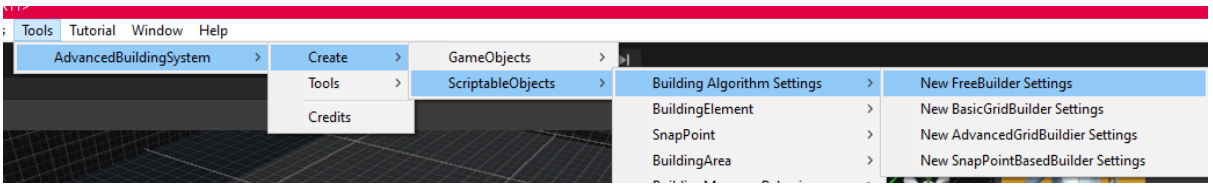
- Free Building
- Basic Grid Building (2D grid snapping)
- Advanced Grid Building (3D grid snapping based on SnapPoint system)
    - Fallback Free Building
    - Fallback Basic Grid Building (2D grid snapping)
- SnapPointBased Building **(Experimental!)**

Note: The default algorithm is the Free Building.

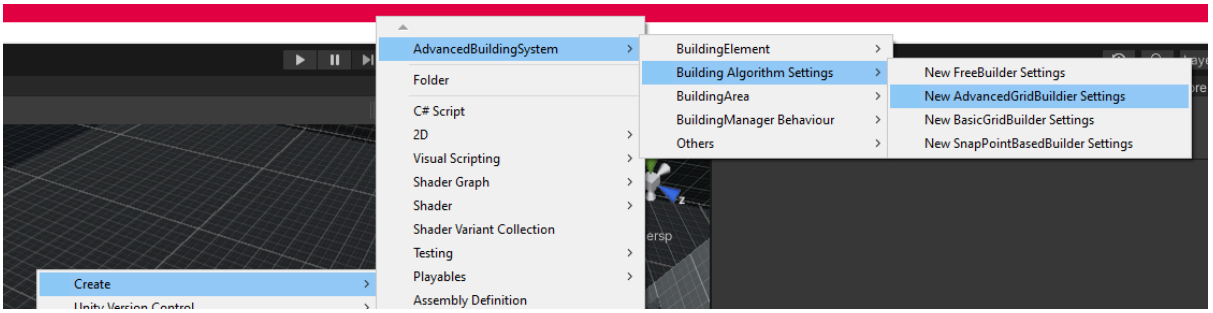
Setup

First you need to make a new Scriptableobject that will hold the properties of the algorithm.

You can create it from the Menu:

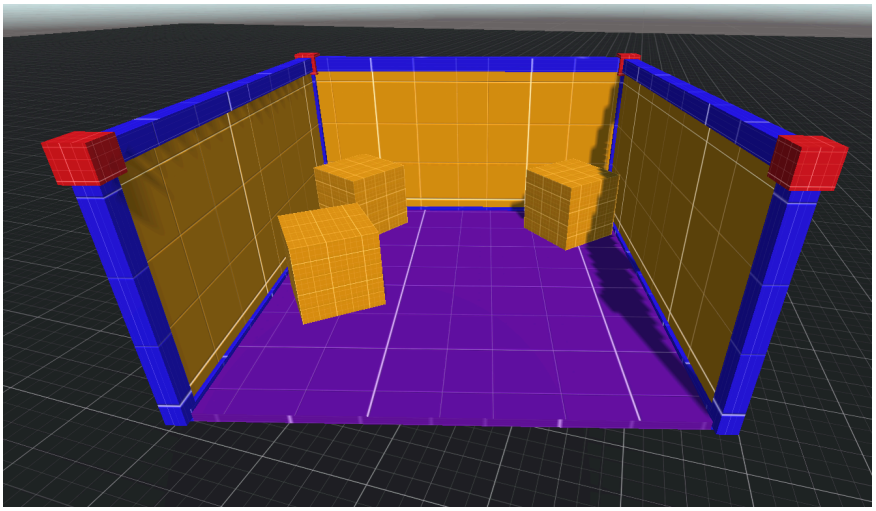


Or you can create it from the project:



You can use different logic for every BuildingElement because in some scenarios you need it. For example:

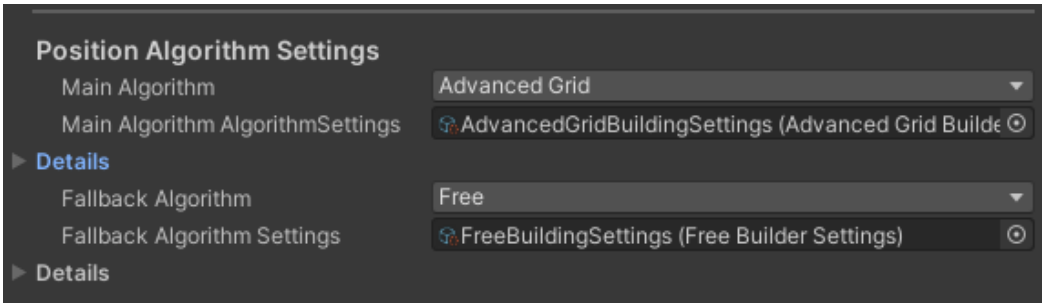
You place the structures' elements with AdvancedGridBuilding but you want to place the furniture with FreeBuilding.



So you need to create your settings but don’t forget that you can use the same settings for multiple BuildingElements.

Tips: Use the same Settings for every BuildingElement that should be placed in the same way and make unique Settings for those, which should be placed in a special way.

This Settings will be stored in the BuildingElements’ MetaData. As mentioned earlier some of the algorithms need a fallback. (AdvancedGridBuilding, SpanPointBasedBuilding)



Every building algorithm has its special properties but there are a lot of basic properties that don't depend on the algorithm. These are the Base Settings.

Base Settings

The basic setting has the following categories:

- Base Settings
  - Basics
  - Layers
  - Rotation
  - Reposition
  - Drag Building
- Element Modification Settings
  - Override Element
- Validation
  - Collision check
  - Ground check

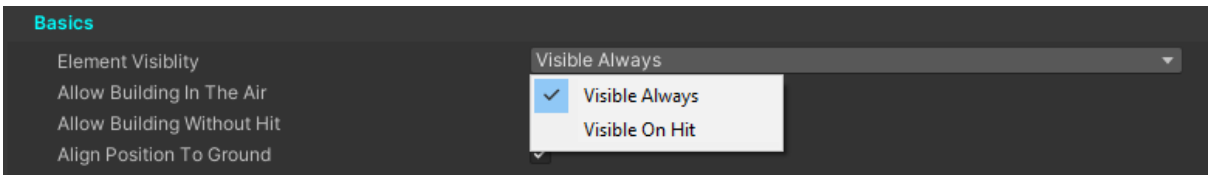
Some of the features are not supported by the algorithms.

	FreeBuilding	BasicGrid Building	AdvancedGrid Building	SnapPoint Based
Basics	**X**	**X**	**X**	**X**
Layers	**X**	**X**	**X**	**X**
Rotation	**X**			
Drag Building	**X**	**X**	**X**	
Reposition		**X**	**X**	

Override Element		X	X	X
Collision check	X	X	X	X
Ground check	X	X	X	X

Basics

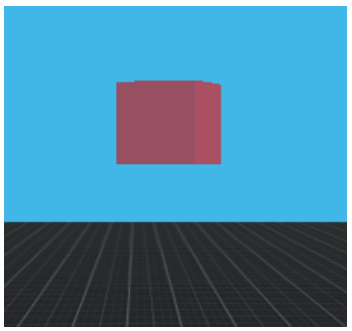
The Element Visibility setting is responsible for when the BuildingElements should be the BuildingElements visible and or blocked based on its position.



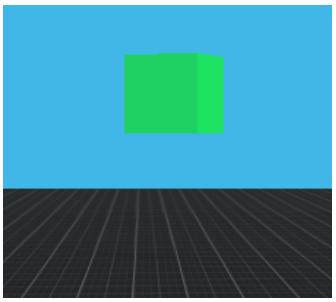
Two main settings are available. The “Visible Always” is the default which means that the BuildingElement is always visible. The “Visible On Hit” means that the BuildingElement is only visible when the raycast hits something.

- Visible On Hit

- Visible Always

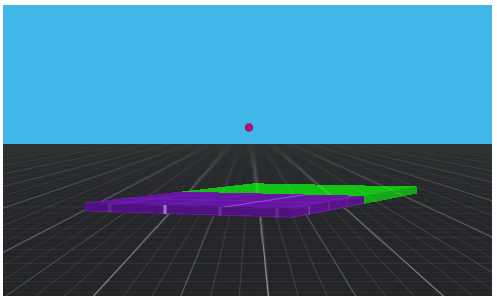


- Visible Always
- Allow Building In The Air

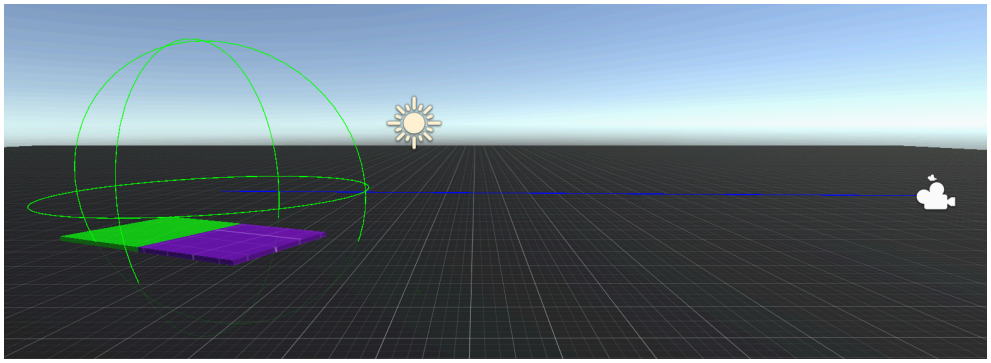


If the “Allow Building In The Air” setting is not set then the BuildingElement is blocked in the air if the raycast is not hit anything and the algorithm can not find any position from the raycast and position. The last mentioned process can be modified by the “Allow Building Without Hit” property.

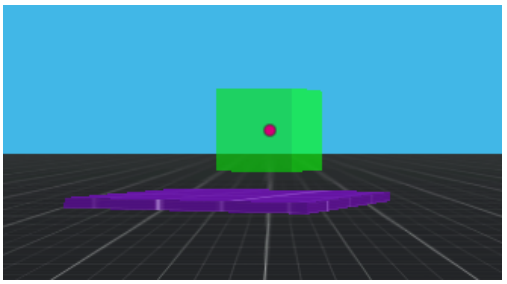
The Difference between the “Allow Building In The Air” and the “Allow Building Without Hit” is that the first one allows the placing of the elements in the air without raycast hit and the second one allows the position search from the air without raycast hit.

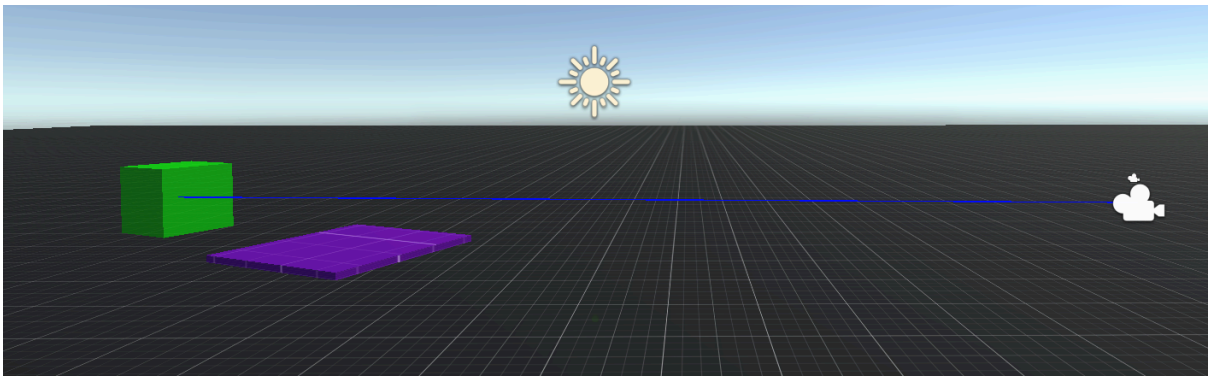


The “Allow Building Without Hit" setting enabled the algorithm's position search from the Raycast’s end position if it didn't hit anything. In this example the AdvancedGridBuilding searches for a position using the end of the raycast. This functionality doesn’t depend on the “Allow Building In The Air”.



The “Allow Building In The Air” is used to enable the placing of the BuildingElements in the air if the raycast doesn't hit anything. In this example the scenario is the same but now a cube is placed with FreeBuilding. Same position, same environment, just the algorithm is different. As you can see the raycast does not hit anything but the BuildingElement isn’t blocked and it is placed at the end of the raycast.





Rotation

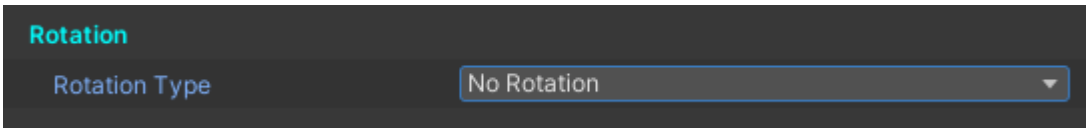
The Rotation settings is applied when you rotate the BuildingElement during the building process. To rotate the element scroll the mouse wheel. Every BuildingElement will be rotated only by the **Y** dimension.
Basically 3 different rotations managed by BuildingManager.

1. Rotation by the Player
2. Rotation by the Snappoint
3. Mixed

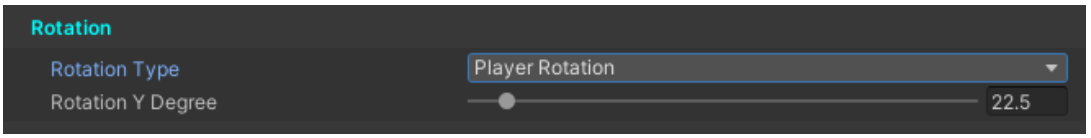
Rotation by the Player is applied when the RotationSettings are used (only supported by the FreeBuilding at this version). Which means that the player can decide the rotation (based on the settings) using the mouse's scroll wheel. The SnappedRotation is used by the BasicGrid and the AdvancedGrid algorithms which means that the snappoint decides the rotation of the BuildingElement. Mixed is applied when the SnappedRotation is used and the player wants to override the snappoint's rotation and scroll the mouse's wheel. In this case the rotation is calculated based on the snappoint's default rotation and modified with the user's given rotation. **This mixed rotation is fixed to 90°.**

Rotation Types (Rotation by the Player)

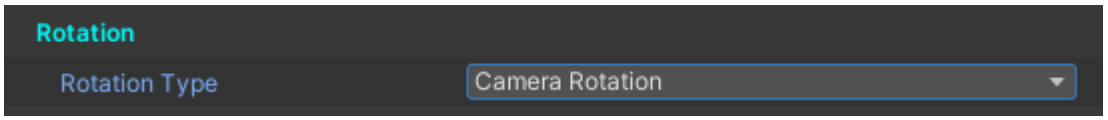
- No Rotation
  Which means that the rotation of the BuildingElements is fixed at 0.



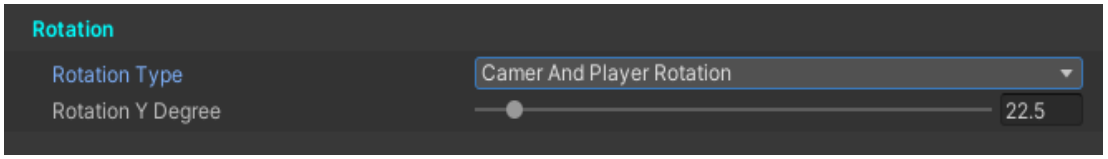
- Player Rotation
  The player can rotate the elements using the mouse's scroll wheel with the set value.



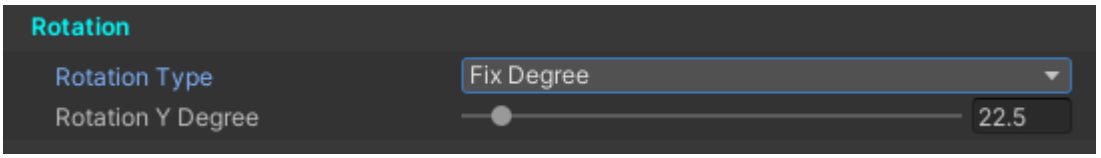
- Camera Rotation
  The rotation of the element is calculated based on the camera to the element is always facing to the camera.



- Camera And Player Rotation **(Recommended)**
  The element is always facing to the camera but its rotation can be overridden by the player. It is the mix of the "User Rotation" and the "Camera Rotation"

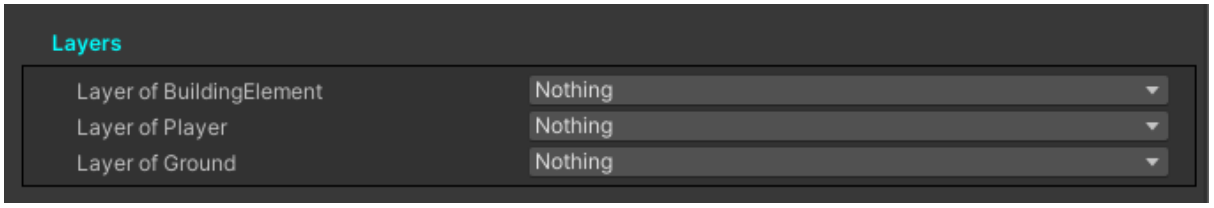


- Fixed Degree
  It is like the “No Rotation” so every BuildingElements’ rotation is fixed with a preset degree. (If you set it to 0 it is equivalent with the “No Rotation”)



Layers

The BuildingManager is using multiple kinds of layers for different purposes. But now just focus on the layer settings on the building algorithms’ basic settings.

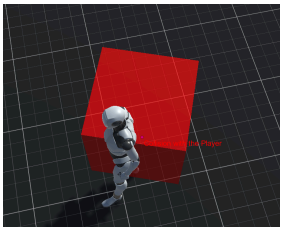


At this moment 3 layers should be set for the algorithm.

- Building Element Layer
- Layers of Player
- Layer of Ground

BuildingElementLayer has multiple uses.

- It is used by the Destroy algorithm to find the BuildingElements.
- It is used by the Buildings to set the layers of its BuildingElements after a save-load process. The Building saves this layer for itself after it was created by the algorithm and uses it after that.
- It is used for searching near BuildingElements in the space by the algorithms like the AdvancedGridBuilding

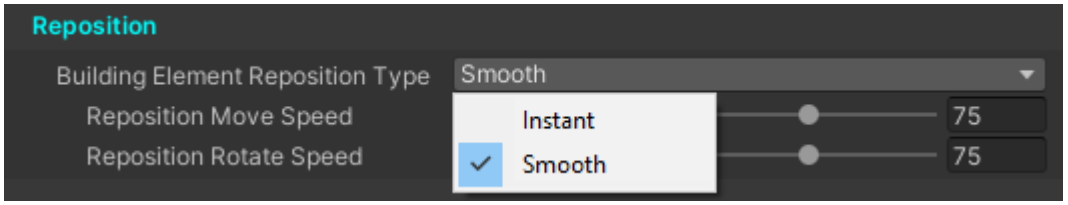


Layers Of Player is important for checking the scenario when the player is colliding with the new BuildingElement. In that case the temporary element should be blocked and can not be placed there. The TemporaryBuildingElements has a trigger collider that blocks them if the player is near.

Layer of Ground is used for the Align to the Ground and the Underground Validation features which are explained in different chapters.

Reposition

The reposition is a setting that allows the developer to choose how the Building elements should change its position and rotation when it is rotated or it has a new better position found by the algorithm.



Two kinds of repositioning are available.

Instant which is just a position + rotation override. The algorithm simply overwrites the rotation and tha position values in 1 frame so the player's point of view the element changes its position instantly.

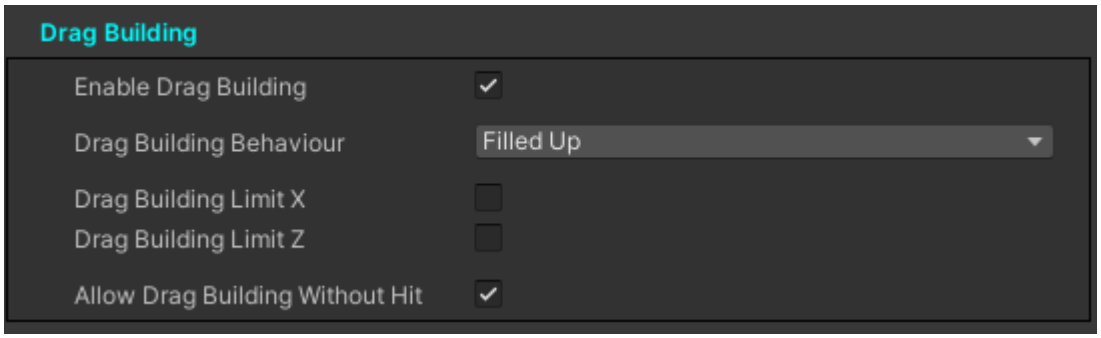
Smooth means that the BuildingElement is set in a state when it should go to the target position and the algorithm moves the BuildingElement just a little bit to the target position until it reaches that. The rotation and the position changes speed can be adjusted independently.

Note that until the BuildingElement isn't reached its final position it can not be placed. The Instant will change it's position immediately so it can be placed any time anywhere while during the Smooth repositioning the element has some delay when it cannot be placed.

This feature is not supported by the FreeBuilding because that algorithm places the BuildingElement where the player looks so it will change it's target position more often almost every frame. Because of that the elements can be placed only when the player looks to the same place for more than 1 frame. It makes the placing of the elements with smooth repositioning more harder. It feels lagging. Because of that the FreeBuilding was forced to use the Instant reposition. If you are using a BuildingAlgorithm that supports the SmoothReposition but it should fallback to FreeBuilding then during the fallback time the instant reposition will be used until it finds a proper position to the main algorithm then the main algorithm's reposition setting will be applied.

Drag Building

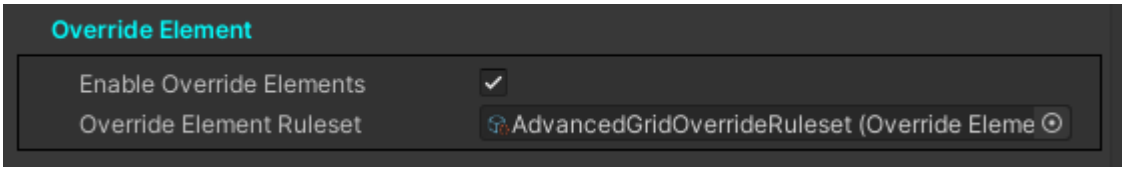
This Setting is already explained in detail in the “Drag Building” chapter.



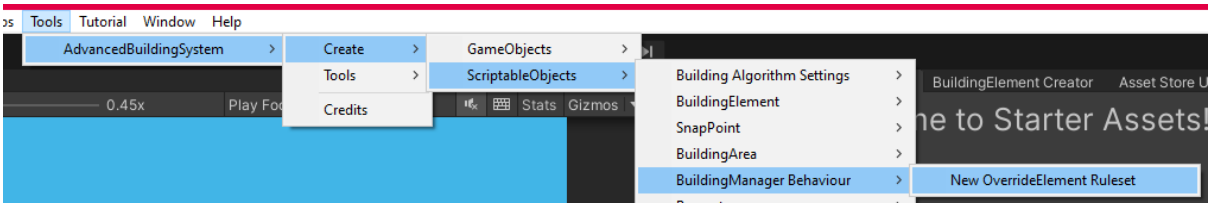
Element Modification Settings

Override Element

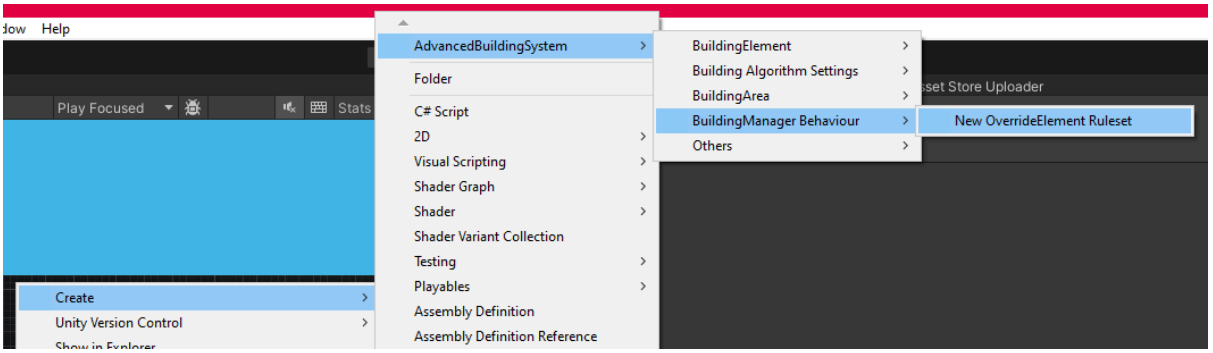
The OverrideElement feature is about swapping an already built element to another one. First of all you can turn off the feature with the “Enable Override Elements” boolean to decrease the unused feature’s performance impact.



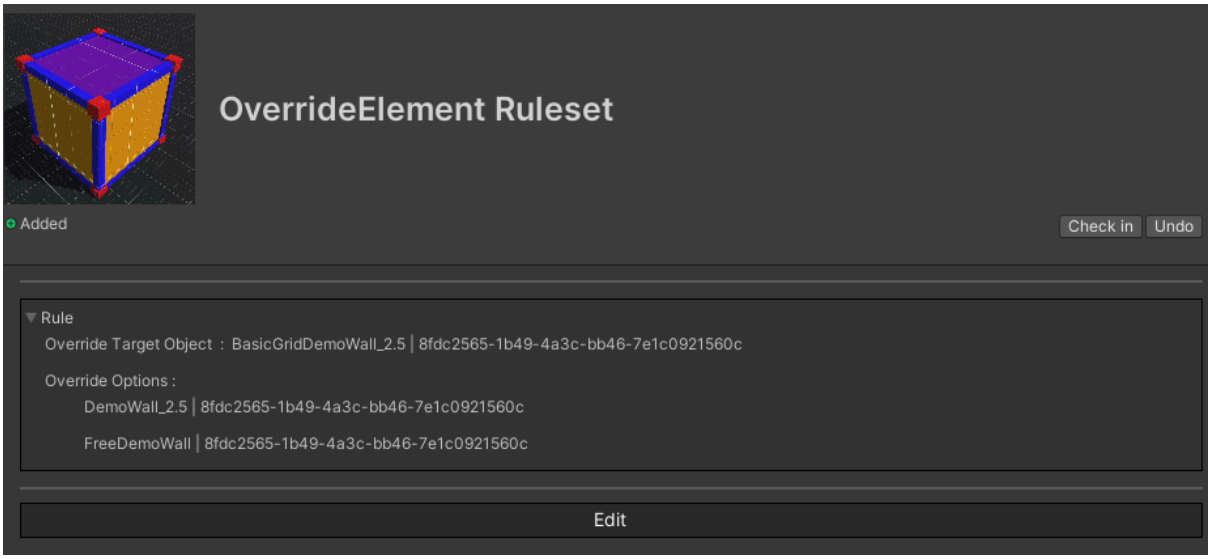
You have to provide an OverrideElementRuleSet which is a scriptable object and it can be created just like the other scriptable objects of the BuildingManager.
From menu:



From project create option:



In the OverrideElementRuleset you can make rules for a given BuildingElemenet and you can create a list of BuildingElements that can override the given first element.
So in this Example the BasicGridDemoWall can be swapped by the algorithm to a DemoWall and a FreeDemoWall.



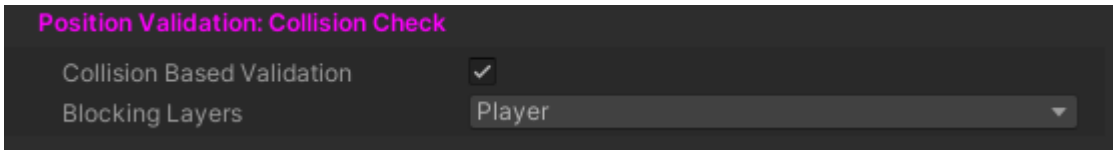
This feature works in such a way when you are building an element in a regular way and the found position is already used the new building element is blocked but if the feature is enabled and one of the rules is allowed to override that element with the new one the temporary element will be enabled. If you build it it will destroy the old element and place the new one.

Validation Settings

The position search algorithm can have a unique special validation mechanism but some mechanism is generic and used by all algorithms and because of that these settings are available in the Validation Settings.

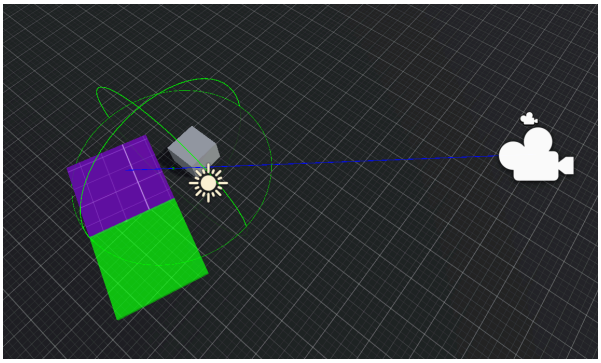
Collision Check

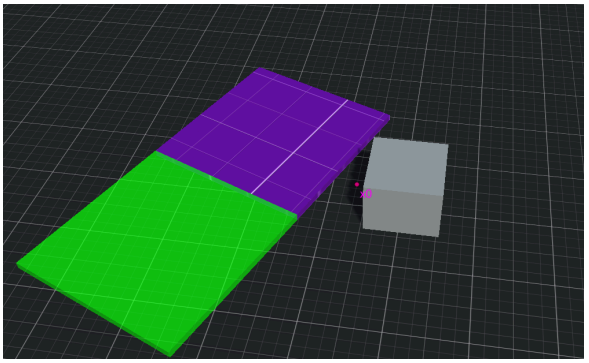
The Collision check is applied when the algorithm tries to validate a possibly good position for a BuildingElement.



First you can Disable this validation for the algorithm that this setting belongs to. This validation works in that way a collision check is performed from the found position. The collision shape is the BuildingElement collider's shape. Also the given layer is used for the collision check. If any GameObject has been found the position is failed by this validation. This is applied during the position search process and it will not allow the BuildingElement to be placed at that position at all.

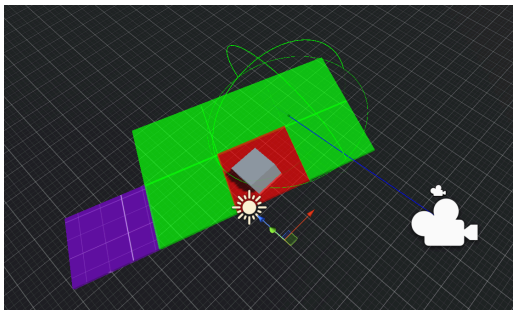
In the following 2 images you can see that scenario when the gray cube has a layer which is not allowed by the collision check. Even if the best position is nearer to the raycast hit point the element can not be placed there because of the validation.

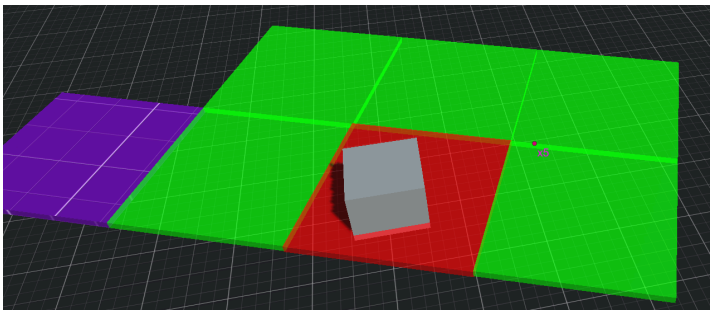




This feature can be useful if you want something to not collide with the BuildingElement. For example a Building is not allowed to collide with a tree.

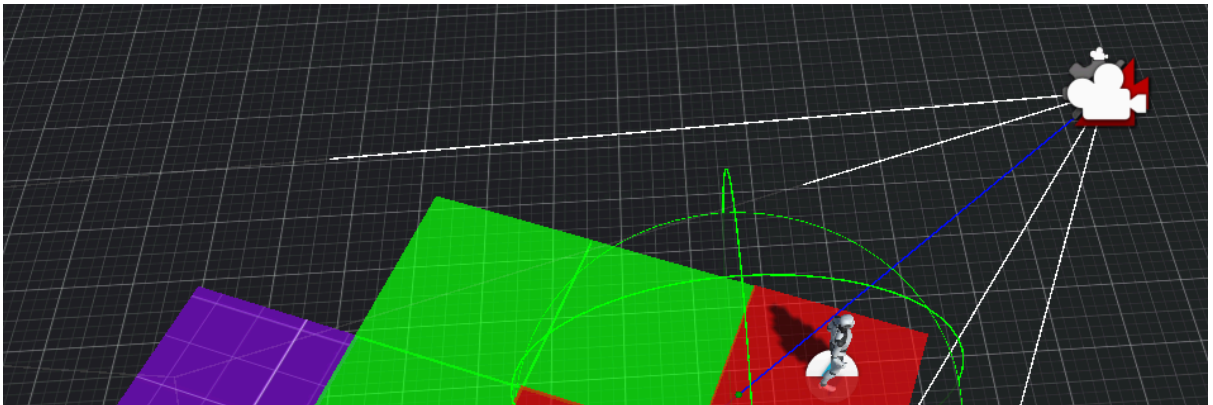
This validation is performed during the drag building too. So every TemporaryBuildingElement should be validated, but in this case the element is blocked because during the drag building process every element will be created but they can be blocked.





Note that this validation is different from the BaseSetting Layer “Layer of Player” settings. This Validation is performed when an element is placed and it is performed only once per element. Meanwhile the other mentioned validation is using a trigger collider because of that if the algorithm placed an temporary element as not blocked because that was successfully validated once then the player can still walk into that element collider and the algorithm should block that element. Because of performance preferences the algorithm will not perform the validation check for every element in every frame. It will be performed only once when the element was placed and after that only the trigger collision can block the elements.

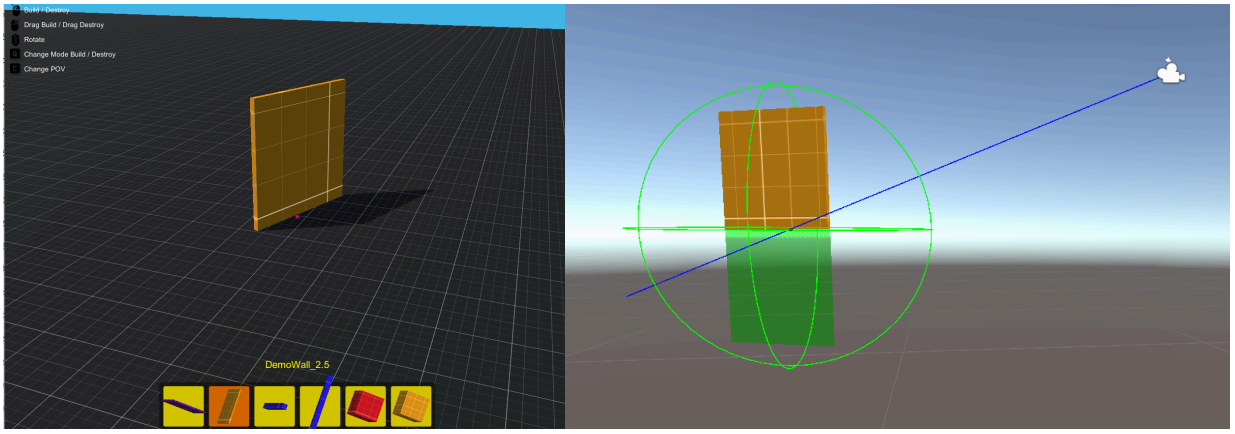
On the following image you can see the two different validations’ same results:



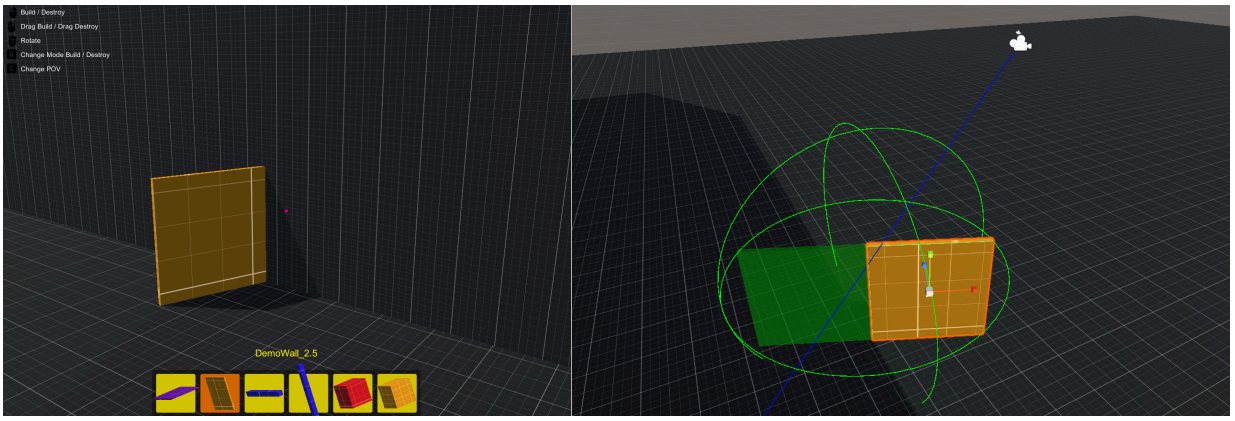
Underground Check

The main goal of this validation is that the BuildingManager tries to place every element above the ground and try to eliminate the following scenarios when the element is magically disappearing meanwhile it was just snapped into the ground.

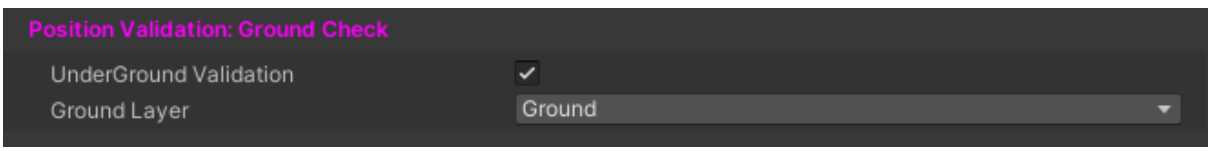
Snapping Under the Ground:



Snapping into the Ground/Wall horizontally:



To avoid these scenarios the BuildingManager has some algorithm that tries to recognise these scenarios and invalidate these positions for snapping. To turn on these algorithms just enable the “UnderGround Validation” in the Ground Check chapter of the Settings.



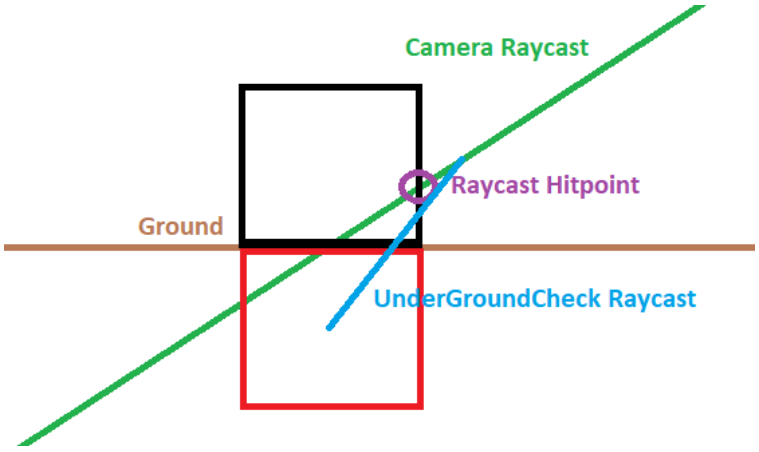
How does the UnderGround Check work?

First of all two totally different algorithms work and they are performed in different scenarios. One is working during the position search process and the other one is working during the drag building process.

First let's talk about the position search process. During the process the algorithms try to find the best position based on the settings. Every BuildingElement will be validated and one of the validation is the UnderGround Check. This validation is performed with raycasts because the collision is not usable here because the elements can be placed in such ways they are inside the ground a little bit but the most important thing is that the BuildingElements can touch and collide with the ground.

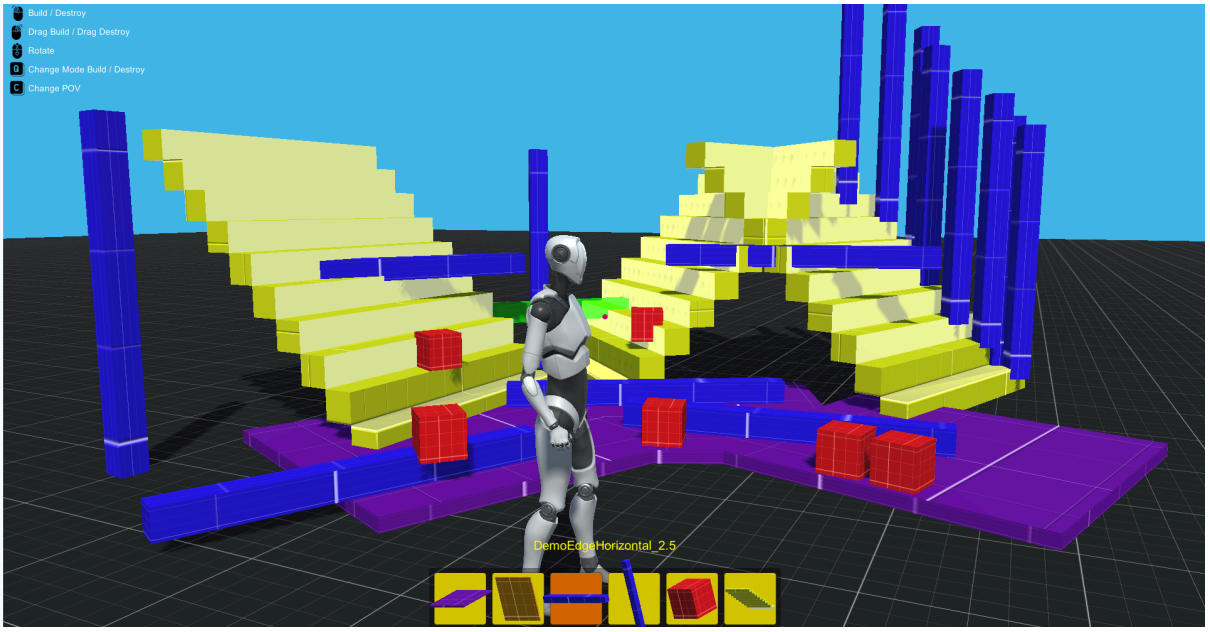
This algorithm will use the raycast's hit point or its endpoint. It gets the raycasts position and the camera position and calculates the position between them which is 0.1 unit far from the raycast position. After that it will perform a raycast to the BuildingElmenet's centerpoint. If it hit anything then the element failed the validation.

The logic behind this algorithm is that the BuildingManager wants to allow that the BuildingElemnets can be inside the ground but it tries to avoid that when the BuildingElements are totally inside the ground. So the Manager tries to find that position when the BuildingElement is more outside the ground than inside and for that it is using the center point so if the centerpoint of the element is outside of the ground the BuildingManager handles it as if it is outside of the ground.



Free Building

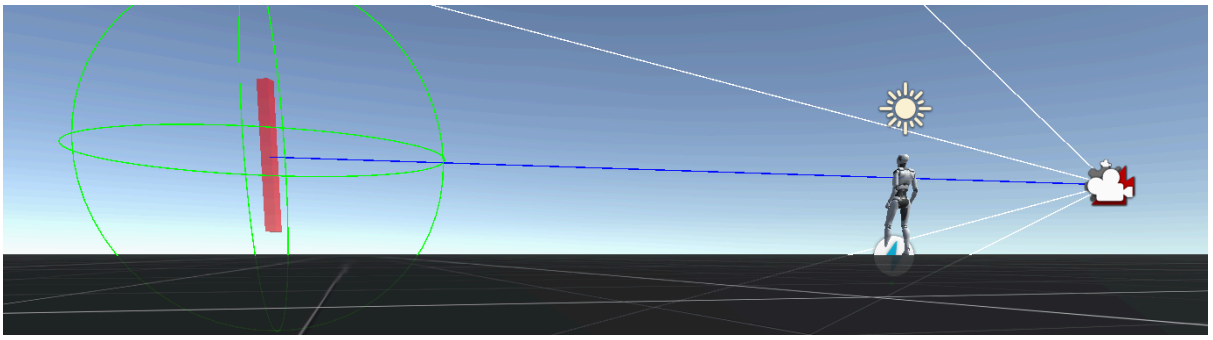
The freebuilding is an algorithm that is used to place the BuildingElements anywhere. It is not using any kind of snapping logic. Because of that this algorithm is not recommended to use for building structures but it can be good for placing furniture into a structure or the best use case is to build the first element for complicated algorithms. Also it is a recommended fallback algorithm for other builder algorithms.



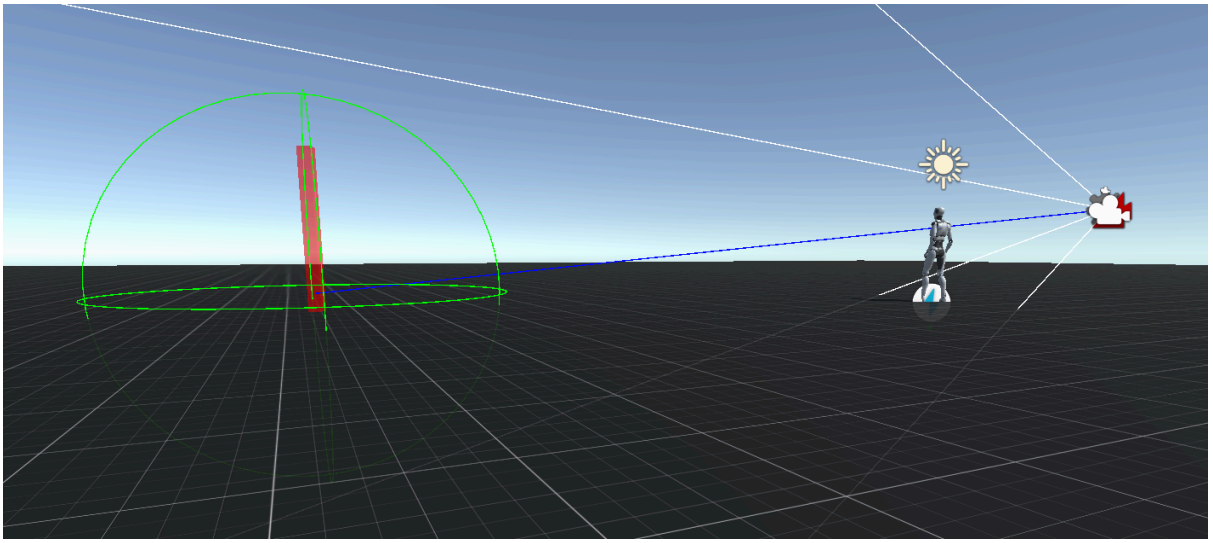
The FreeBuilding is the most simple algorithm and it has no unique settings at this moment. Only the BaseSettings are available and as it said earlier the Reposition settings are not supported but this algorithm is the only one which is supporting the RotationBaseSettings.

How does it work?

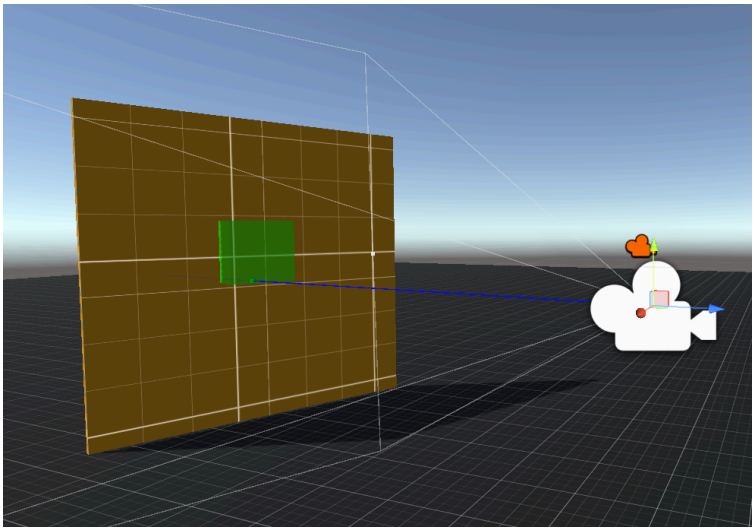
First scenario: If the raycast does not hit anything then the endpoint of the raycast will be used.



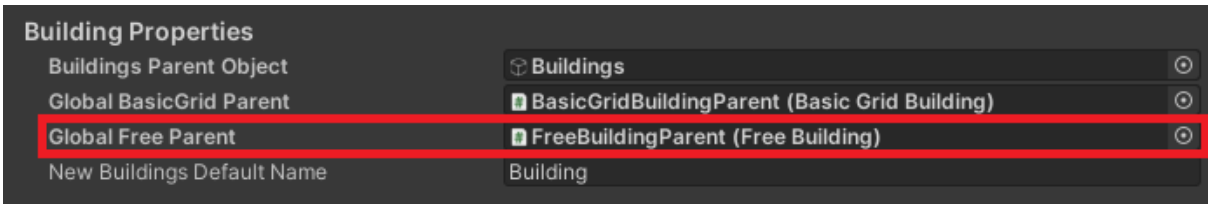
Second scenario: If the raycast does not hit anything then the endpoint of the raycast will be used. Same scenario until this point but the BuildingElement is inside the ground then the BuildingElement will be aligned to the ground. As you can see on the next image the raycast's endpoint is in the air but the building element is fully outside the ground.



Third scenario: If the raycast hit something then the position will be the hitpoint but a position is moved up vertically (Y dimension) with the BuildingElement's dimension.y to achieve the same functionality as the ground alignment.



The GlobalFreeParent keeps counting the positions. what can be set by the BuildingManager's GUI:



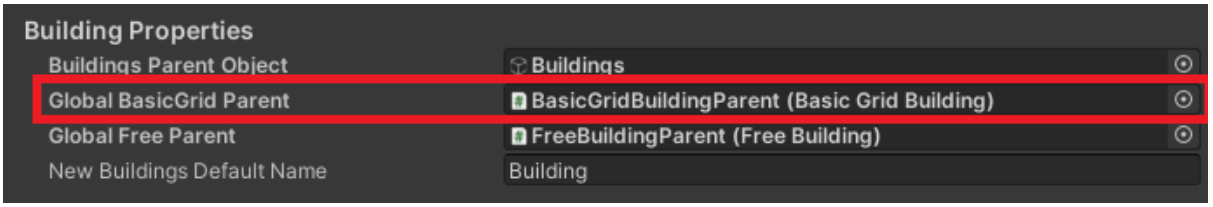
If it is null then the BuildingManager will create a new FreeBuilding as child of the "Building Parent Object"

Basic Grid Building

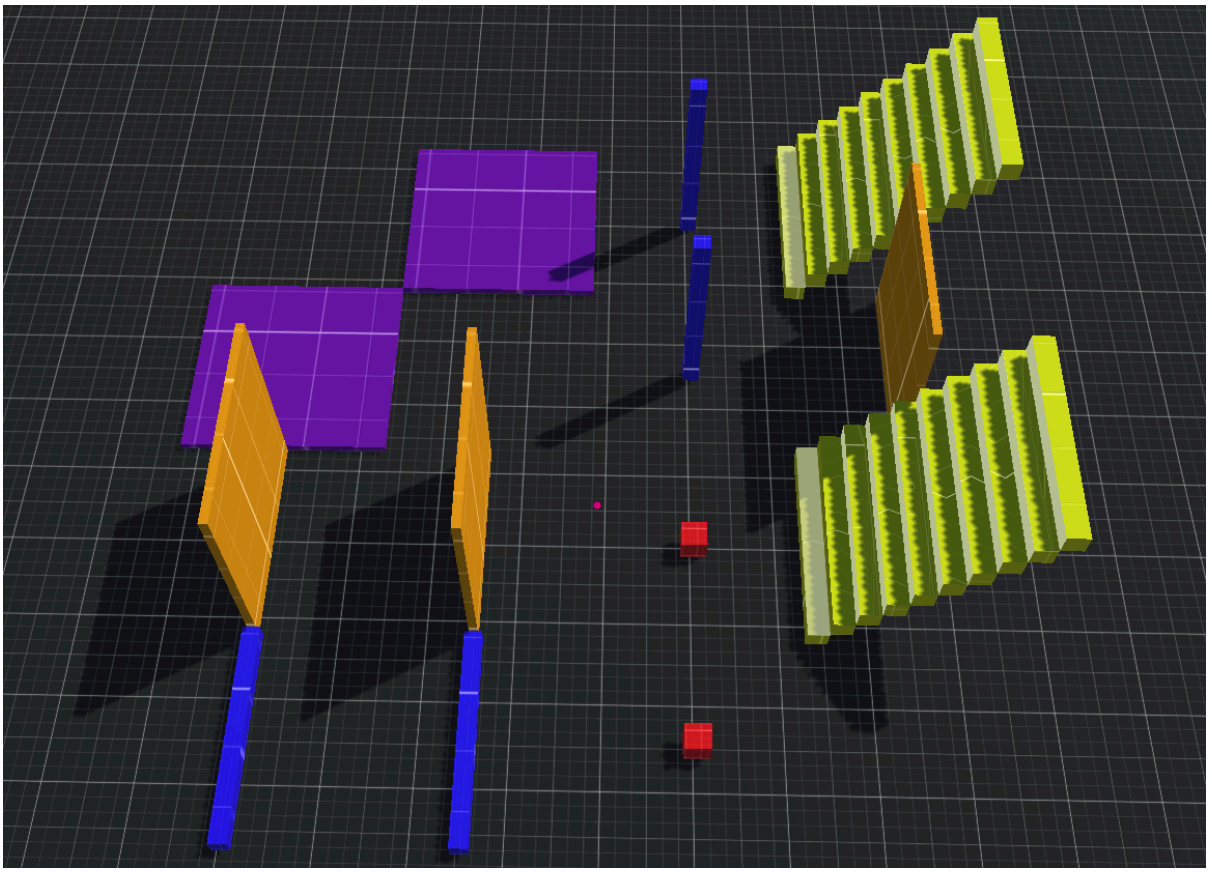
The Basic Grid Building is a simple 2D grid algorithm in 3 dimensions. Every BuildingElement should be snapped on the grid independently of its shape or size. Only the X and Z axes are dependent on the algorithm so independently from the Y axis only 1 BuildingElement can be on the same grid coordinate.

Every frame the Manager regularly searches for a position with the raycast and calculates the nearest grid position from the search result. Independently from the Y axis if the grid position is already used the TemporaryBuildingElement will be blocked.

The GlobalBasicGridParent keeps counting the positions. what can be set by the BuildingManager's GUI:



If it is null then the BuildingManager will create a new BasicGridBuilding as child of the "Building Parent Object"



Settings

The BasicGrid does not support the Rotation settings of the Basic Settings. It is using only the Mixed and the SnappedRotation.

Also it is not supporting the “Align Position To Ground” setting of the Base Settings which is included by the Basics settings. Because of that the BasicGridBuilding has it’s own alignment logic.

The setup is simple because only 2 parameters are available. First the size of the grid should be set. Second the Vertical Grid Placement should be set which is about the positioning on the Y Axis.

3 possible logic is available:

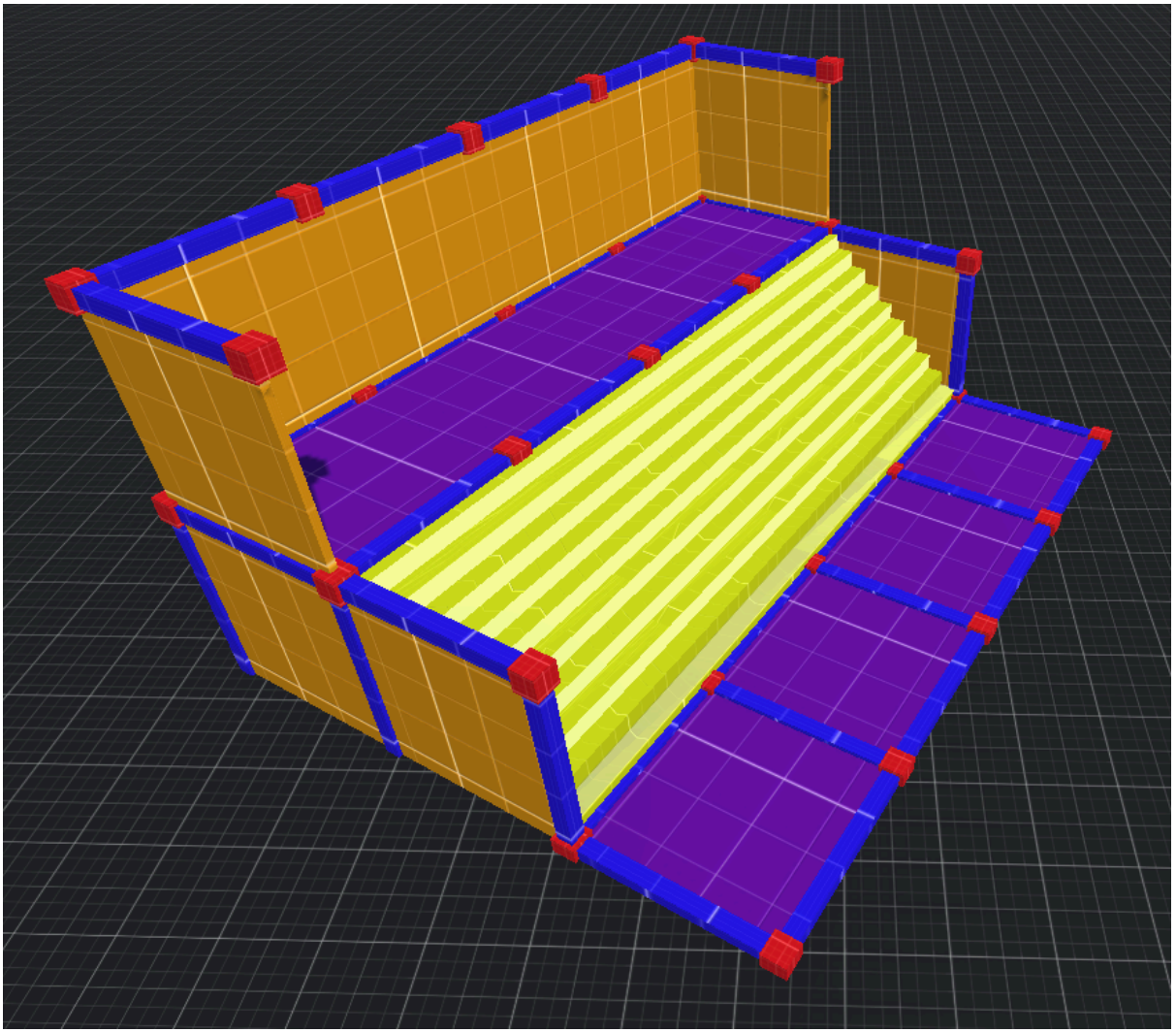
- Fixed Position
  The Y axis value will be a pre-set value. It is a fixed value and doesn’t depend on the raycast, ground, camera etc positions.
- Raycast Position
  The Y axis value is always set to the raycast’s hit point or its endpoint.
- Align To Ground
  It will try to align it to the ground. After the position search the Manager will perform another raycast down from the grid position and if it hits anything then the element will be aligned to the second raycast’s hit point.

AdvancedGridBuilding

The main goal of the algorithm is creating a building by nicely placed and aligned parts. The AdvancedGridBuilding is currently the most complicated building algorithm. It's working in like that it manages 6 different grids at the same time. Every BuildingElement has a type that is used for a sign where that element can snap on the grid. Every type can snap on a different grid with different logic. This ensures that every element will snap onto its required position.

- Floor
- Wall
- Horizontal Edge
- Vertical Edge
- Corner
- Center

This algorithm does not support the rotation settings of the Basic Setting. Only the Mixed and the Snapped rotation is available. (Mixed rotation is fixed to 90°)



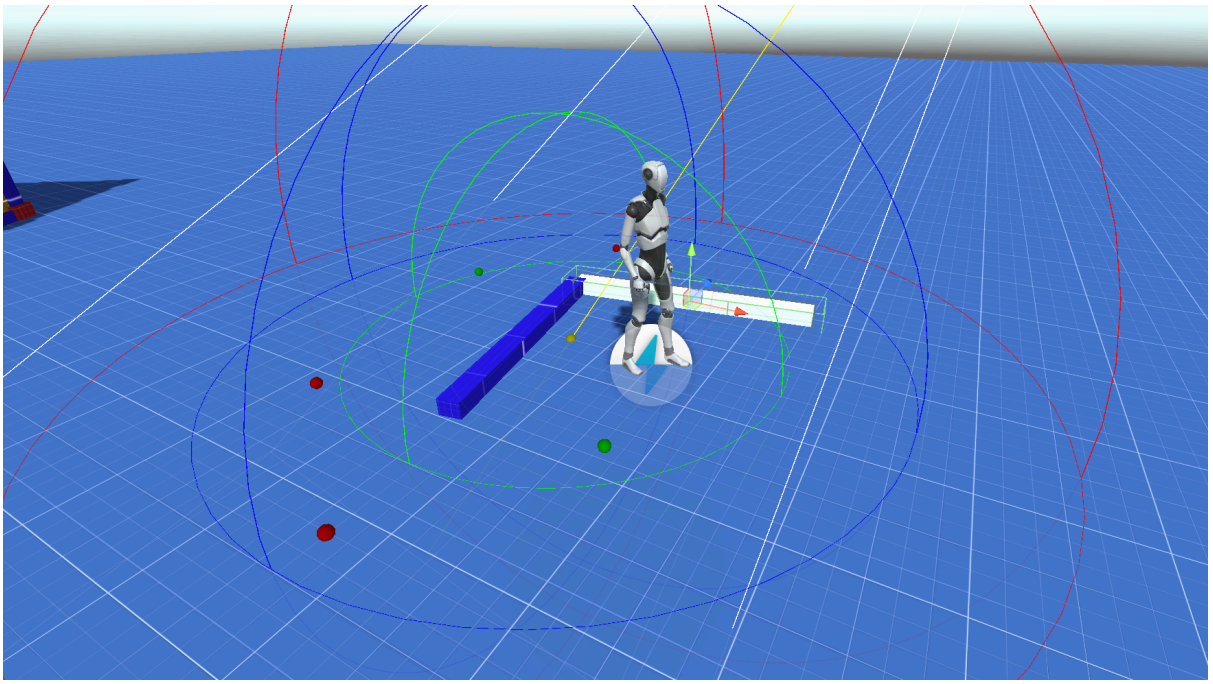
How does it work?

The BuildingManager performs a raycast and the raycast's hit point or the endpoint will be used. From that point the position of the BuildingElement is calculated based on its type. After that the algorithm knows which grid should be used.

A collision check is performed from the calculated position and it searches for BuildingElements. If it finds something, the parent buildings are saved.

After the search the algorithm will check the possible position on every parent and it will try to find the nearest best position.

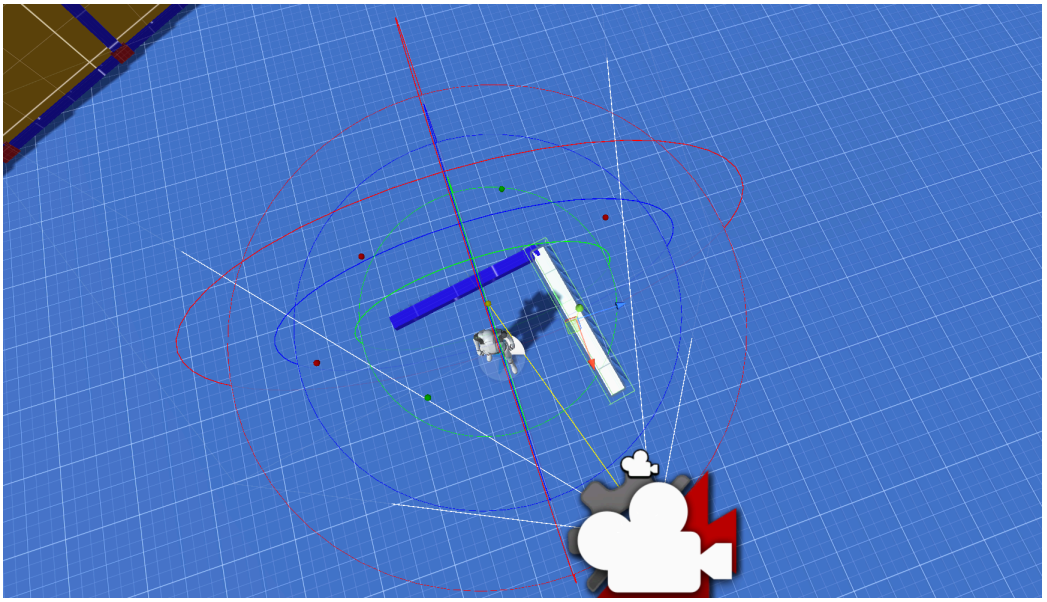
If the algorithm can not find any validated good position the fallback algorithm will be used. It can happen often because the AdvancedGridBuilding is based on snapping and it can not find any position if no BuildingElement is available nearby.

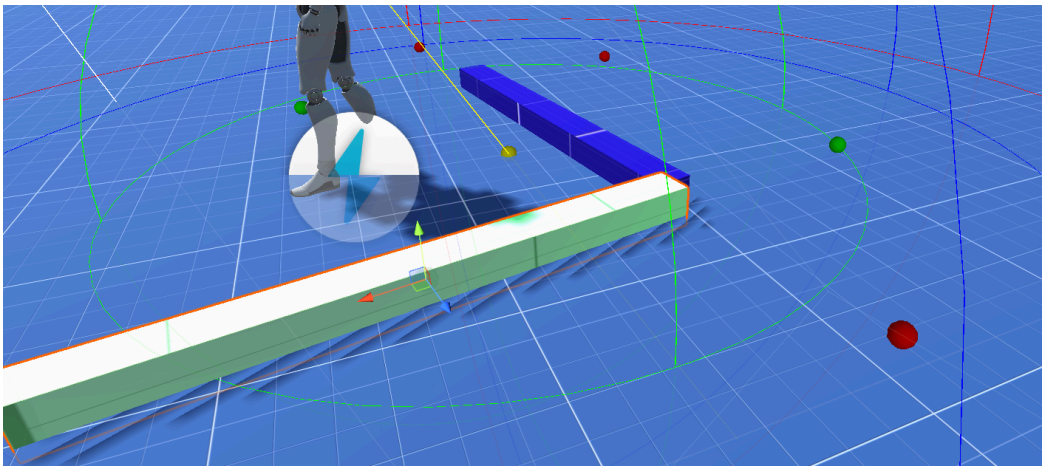


In the image above you can see the algorithm in work. Check the gizmos and the meaning of these:

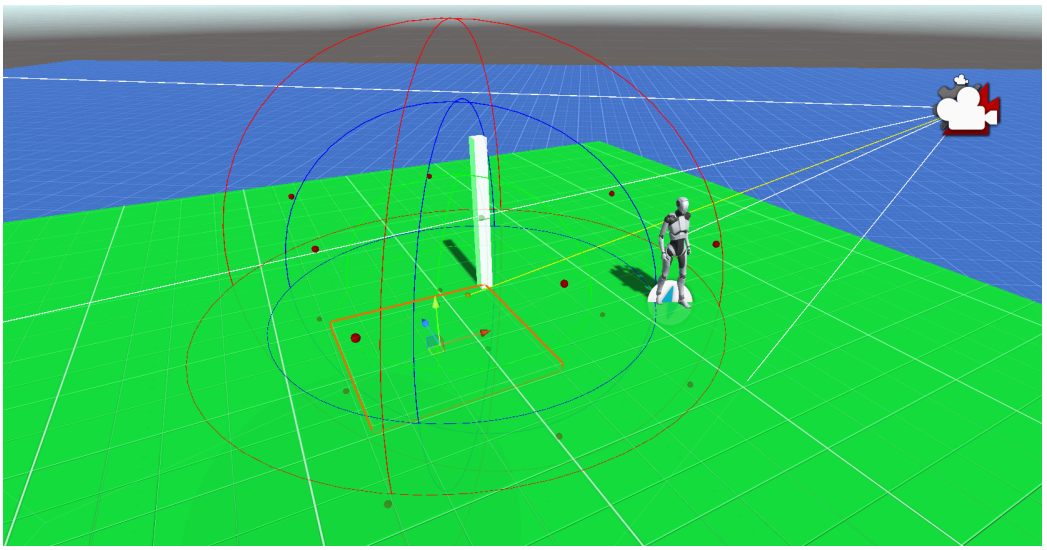
- The yellow line is the raycast.
- The yellow ball is the hitpoint.
- The green sphere is the build range
- The green balls are the positions inside the build range
- The blue sphere is the search range
- The red balls are the snappoints what outside of the build range
- The red sphere is the snappoint draw range (the snappoint gizmos are shown inside that range but is has no effect on the algorithm)

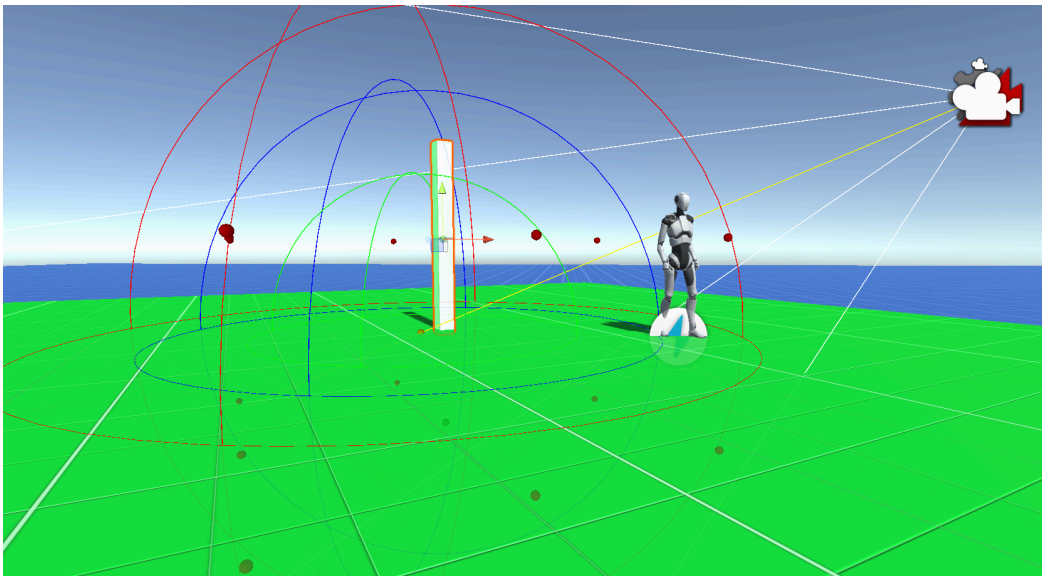
Same scenario different angels:

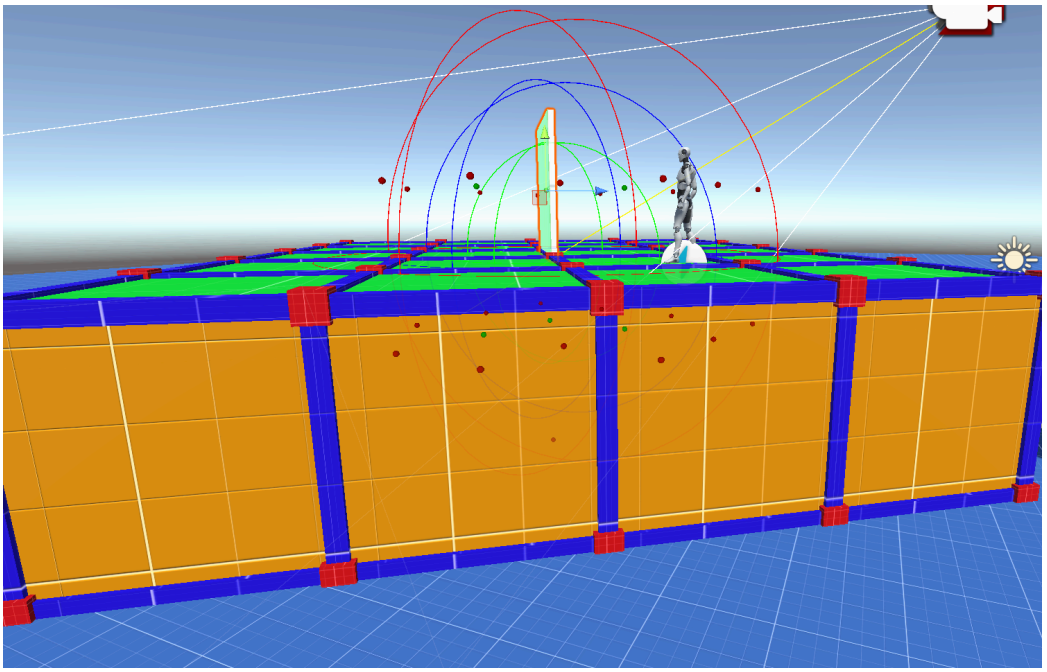


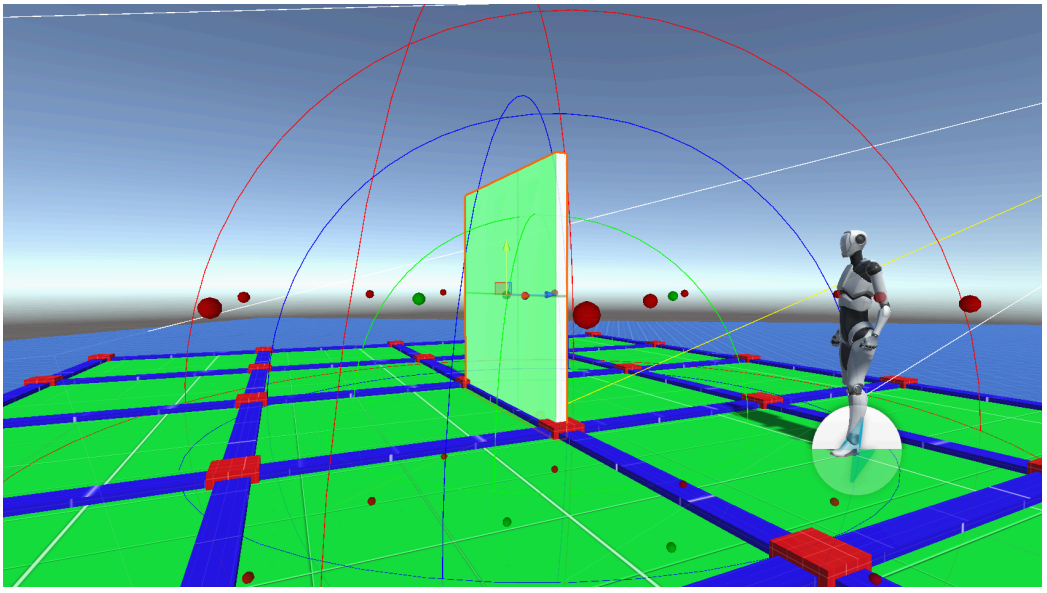


Same more image:



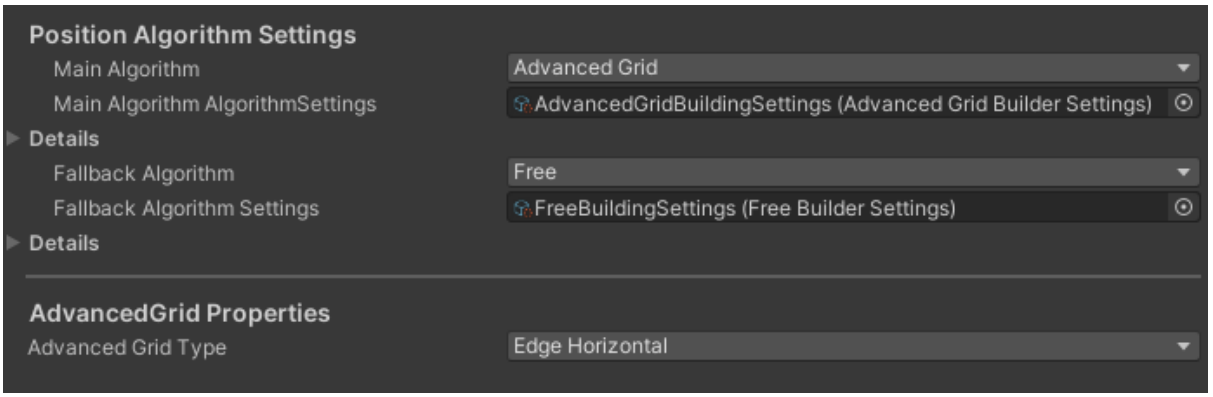






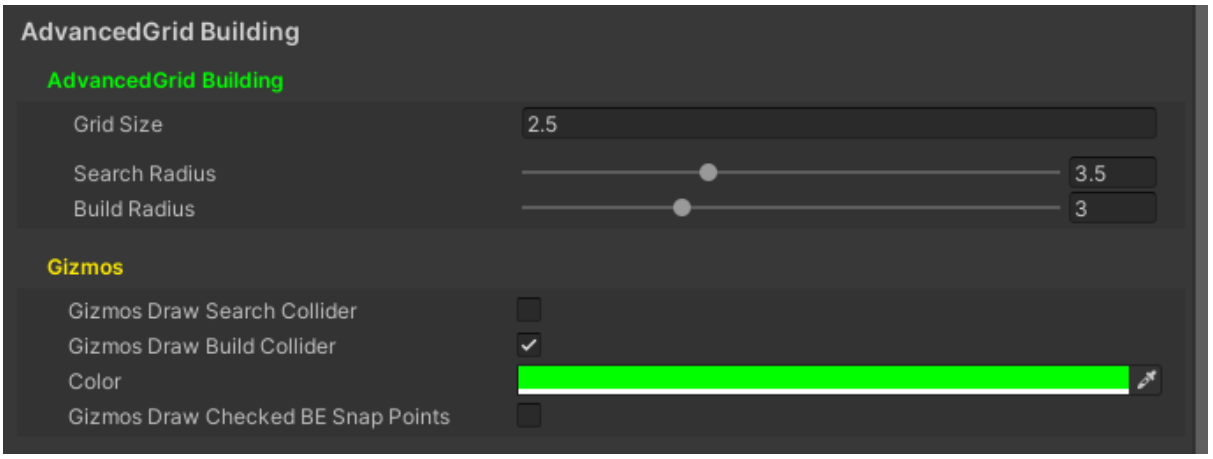
Setup

Every BuildingElement should has an AdvancedGridType:



Also a fallback algorithm should be set. This fallback algorithm is used to find a position if the main algorithm fails.

Note that even if the main algorithm fails and the fallback is used to find a position every other aspect of the algorithm is calculated based on the main algorithm. For example if the AdvancedGridBuilding's main logic fails and the freebuilding is used for searching a position and it finds a proper position the dragbuilding's logic will be calculated based on the main algorithm.



First of all the earlier mentioned gizmos are customizable or turned off.

The most important parameter is the Grid size which is used for the grid logic.

The Search Radius is used for the Collision check when the algorithm is looking for BuildingElements near to the calculated position by the raycast's point.

The Build Radius is customizable which is used for checking the range of the BuildingElement during the validation and if it is further than the set range the position can not be validated.

Drag Building

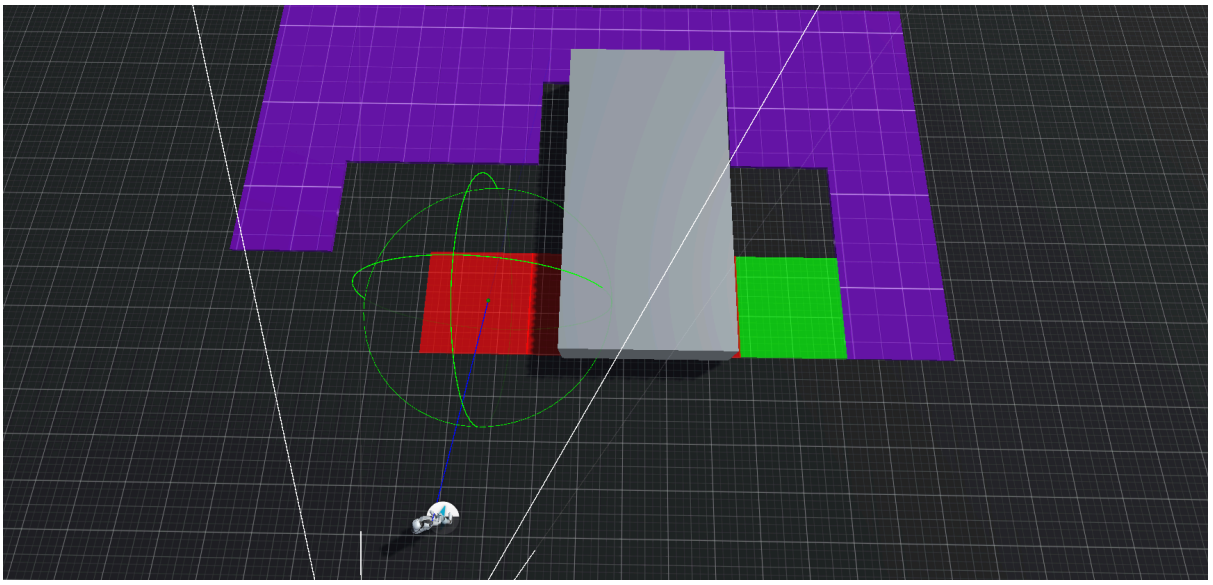
The FreeBuilding and The BasicGridBuilding support the DragBuilding by default. The SnapPointBasedBuilding doesn't support this feature at this moment. The AdvancedGridBuilding has its special rules like:

1. there can not be 2 BuildingElement in the same position
2. every BuildingElement should snap to at least one another valid element

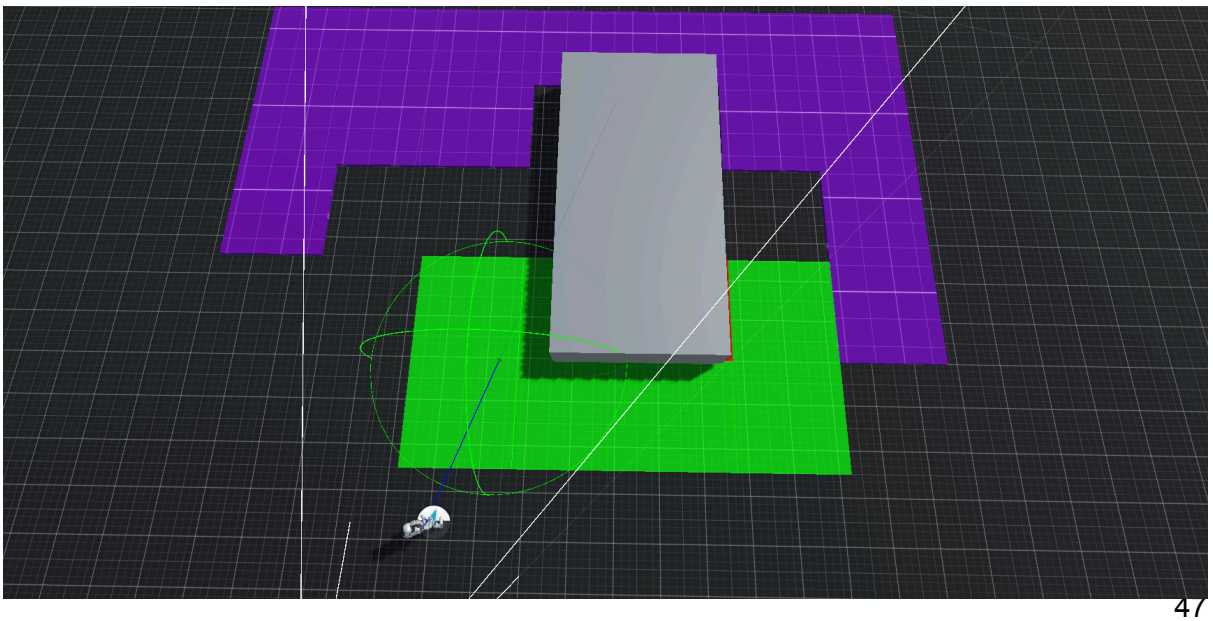
so drag building should follow these rules too.

During the drag building the algorithm checks the neighboring status of the TemporaryBuildingElements and if there isn't any parent then the elements will be blocked.

In the following example a gray object blocks the drag building because the gray object is in the ground layer and the BuildingElement's inside that should be blocked because of the underground check. If an element is not a neighbor of any valid element the element should be blocked too.



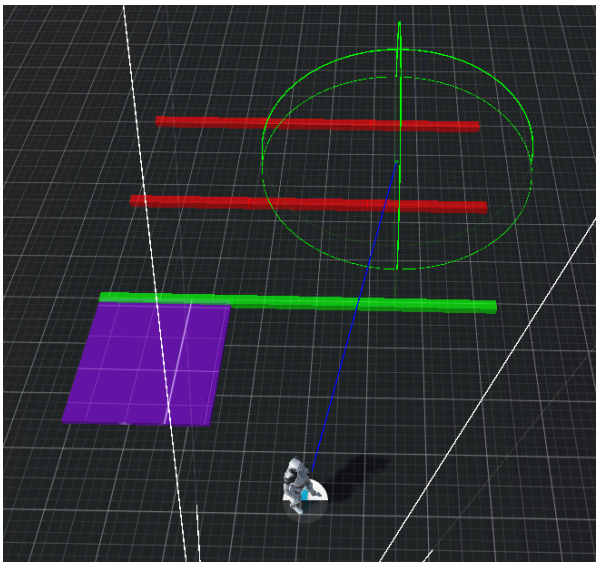
As you can see the elements are available when it has a valid neighbor.

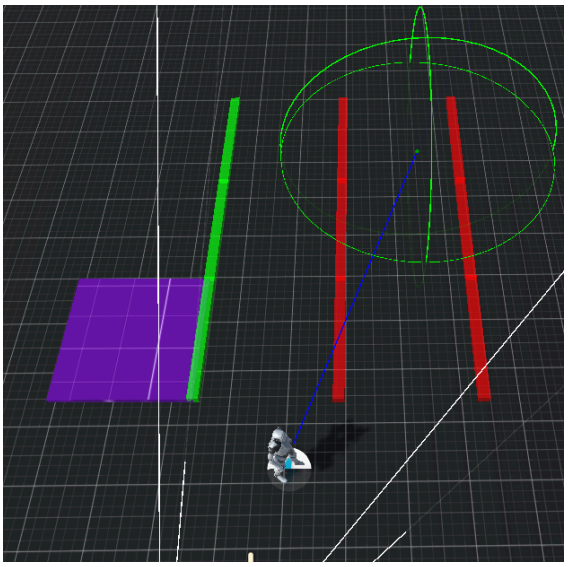


Also it is recursively checked back every time when a new line is added or removed during the drag building.

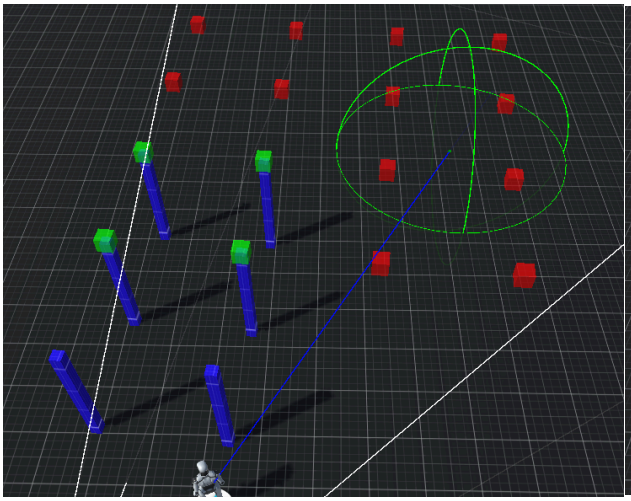
The neighboring validation is different based on the type of the BuildingElement. The following images show the different scenarios.

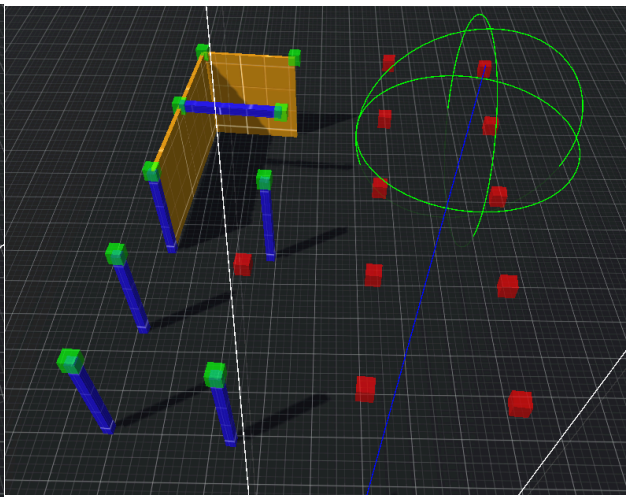
Vertical Edge / Wall: The vertical edge and the wall should snap in a continuous way so the drag can be built by only one axis which is based on the grid's position.





Vertical Edge / Corner: The Corner or the Vertical Edge can never reach another coroner so it can be validated if it has any other neighbor.





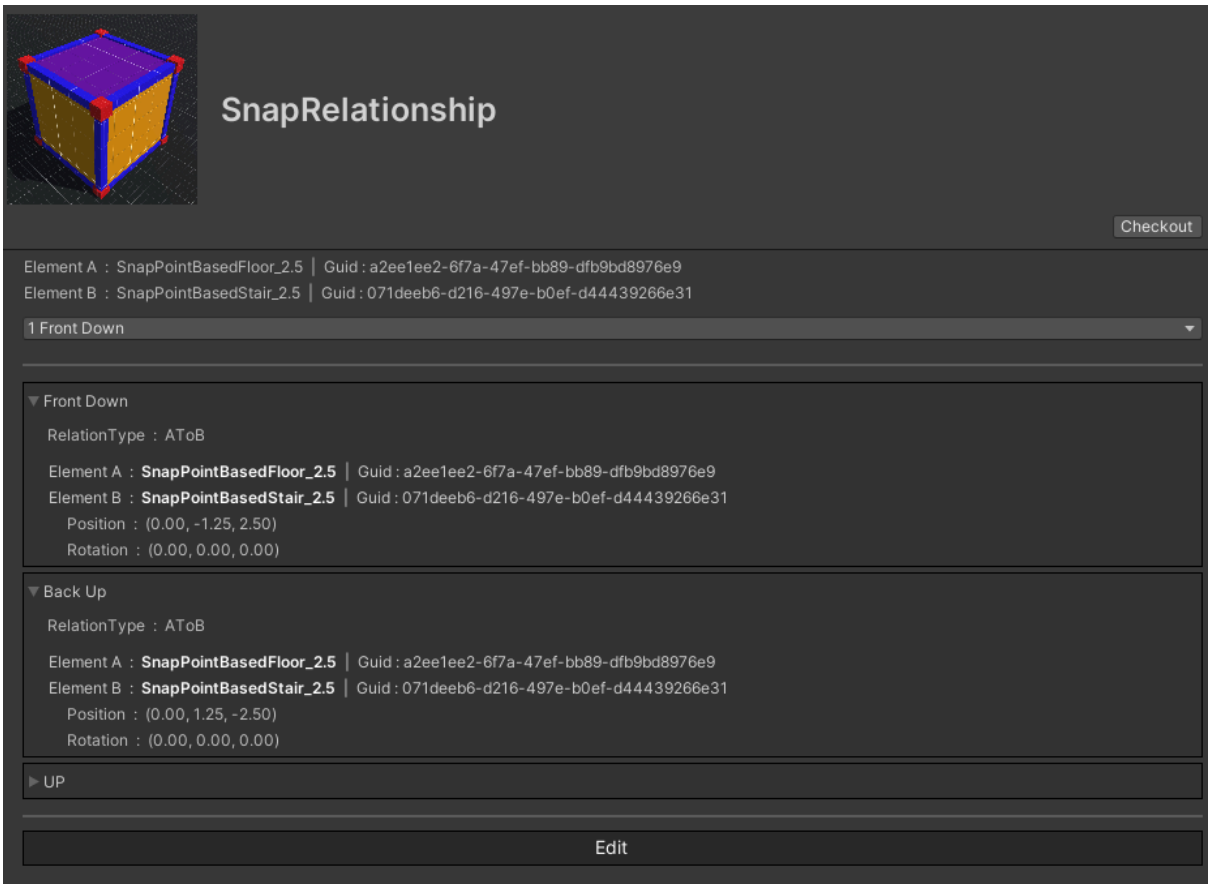
At this version only in one scenario when the BuildingElements can be built in an illegal way when they are built with drag building as first element.

SnapPointBasedBuilding

This algorithm is in the experimental stage at this version. This means that this is not finished. It does not support every feature provided by the BuildingManager. There can be bugs or strange behaviors, nothing is guaranteed.

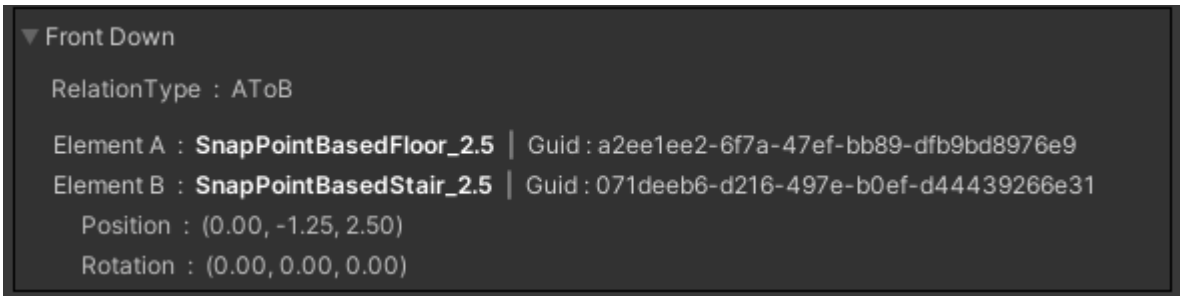
SnapPointBased algorithm is such an algorithm which uses snap points defined by the developers. You as a developer can create SnapPointRelationships and SnapPointLists. The SnapPointRelationship determines the relationship between 2 elements so basically there is a given element that should snap to the other element and you determine how they can snap to each other.(positions, rotations) You determine the SnapPoints between these elements with positions and rotations. After that you can create a SnapPointList with relationships that will be used by the algorithm.

A SnapRelationship looks like this:



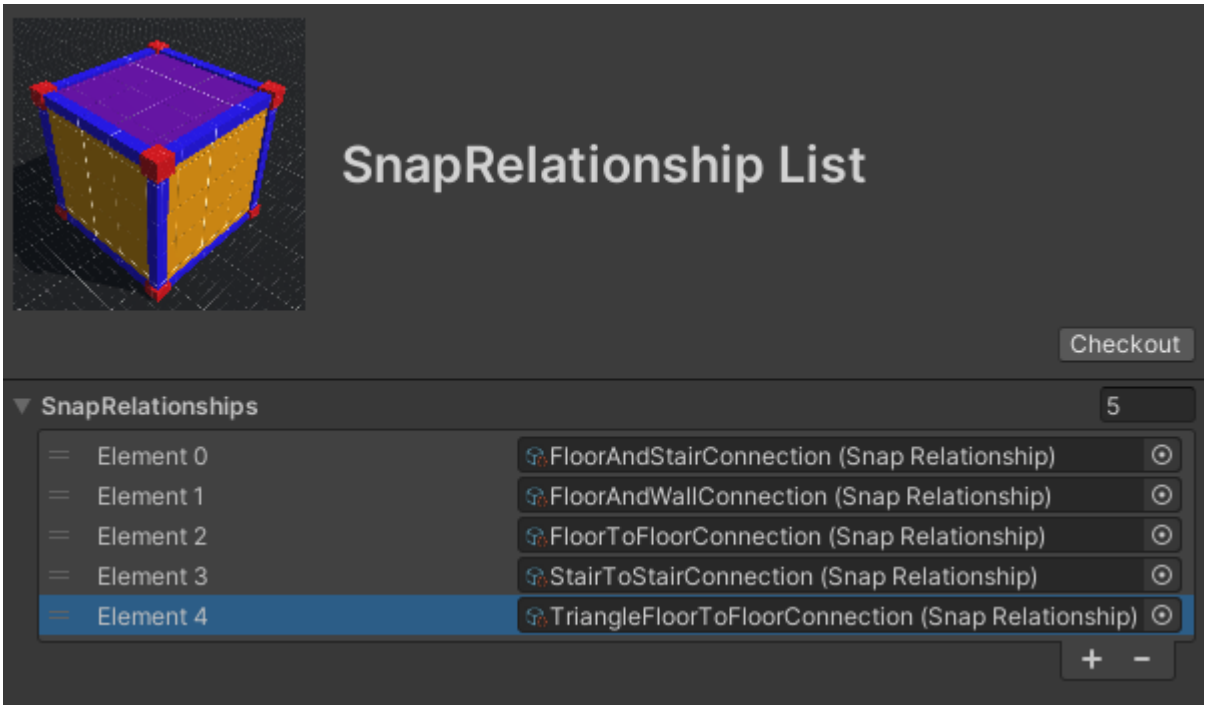
Given “Element A” and “Element B” (a floor element and a stair element) with 3 relations: “Front Down”, “Back Up” and “UP”.

Let’s talk about relationships.



In this example the RelationType is “**AToB**” which means that the ElementA will snap to the ElementB at (0, -1.25, 2.5) position with (0,0,0) rotation. Which means that when the player try to find a position for the ElementA and the algorithm find an ElementB nearby when the SnapPointBased algorithm check the potential positions one of the position will be the **ElementB’s position + (0, -1.25, 2.5)** with **ElementB’s rotation + (0,0,0)**.

The SnapRelationshipList is just a regular list of SnapRelationships.



Tracking

The BuildingManager is a standalone module that doesn't need to be managed by other objects. You as a developer just need to set up the BuildingElement and settings. The Manager will work on by itself and doesn't need to be managed but sometimes it needs some information or some data from the project. To ensure the manager works to its maximum potential an object that implements the **BuildingManagerTracker** which means that you have to inherit from the BuildingManagerTracker and override its function. The Manager will call it's function for:

- get information
- give you information about the state of the building process
- give you the possibility to make decisions to change the behavior of the Manager (allow or deny actions)
- give you the possibility to handle or react to events happened during the build process

BuildingManagerTracker function

- ExternalInterface
    - SetExternalInterface
        - This function is called from the Awake of the BuildingManager. It will give you the Manager as **IBuildingManagerExternalInterface** interface. This interface will be explained later.
- BuildingElment
    - ResetBuildingElement
        - This function is called when the BuildingElement is changed to a different one.
        - It is called too when the BuildingElement is changed to null.
    - BeforePlace
        - This function is called when the player triggers the placing of a BuildingElement. It is called for every single BuildingElement during the DragBuilding too.
        - This function should return a boolean value. If the value is false the BuildingElement will not be placed.
        - It will give you the possibility to create custom logic and decide that a BuildingElement can be created or not so you can override/expand the logic of the BuildingManager
    - BuildingElementPlaced
        - This function is called after a BuildingElement is placed. It will be called once during the DragBuilding too.
- Building
    - BuildingPlaced
        - This function is called when a new Building is created by the Manager
- Building Cycle
    - StartOfBuildingCycle

        - It is called at the start of the building process in every update function. Basically it is called at the start of the Update function.
    - EndOfBuildingProcess
        - It is called at the end of the building process in every update function. Basically it is called at the end of the Update function.
    - RaycastOnHit
        - This function is called when the raycast of the BuildingManager hits something.
- AdvancedGrid and BasicGrid Building
    - PositionCustomValidation
        - It will be called during the AdvancedGridBuilding process. When it validates a snap point position the function has been called.
        - A boolean value should be returned. If the boolean is false the position is dropped by the algorithm.
        - It gives you the possibility to implement a custom snap point validation logic.
- Temporary Building Process
    - DragBuildingStarted
        - It is called when the drag building is started.
    - DragBuildingStoped
        - It is called when the drag building is stopped.
    - FirstElementBlockedStateChanged
        - It is called when the element used for SingleBuilding (the first element of the drag building) changes its state.
    - FirstTMPElementCurrentPosition
        - It is called in every frame to refresh the current position of the first temporary element.
    - CurrentValidTMPElements
        - It is called during the drag building when the number of the temporary elements changes (new line is added or removed).
    - GetMaximumCountOfTMPElements
        - It is called to get information about how many elements can be placed at once during the drag building.
        - It should return a number.
        - It is explained in detail in the Material Functionality chapter.
- Destroy Process
    - BeforeDestroy
        - It is called when the player triggers the destruction of an element.
        - It should Return a boolean. If the boolean is false the element will not be destroyed.
        - It is called for every single element during the DragDestroy feature too.
        - It gives you the possibility to override the logic and create custom logic to avoid the destruction of elements.
    - DestroyTimerStarted
        - This function is called when the destroy timer is started.
        - It is not called if the destroy is set as Instant

- DestroyTimerIsOngoing
  - It is called during the destroy process when the timer is started.
  - It is called in every frame to refresh the leftover time.
- DestroyTimerStoped
  - It is called when the timer has stopped.
- BuildingElementDestroyed
  - It is called right before the BuildingElement is destroyed. Here you have no possibility to cancel the destruction of the element anymore.

IBuildingManagerExternalInterface functions

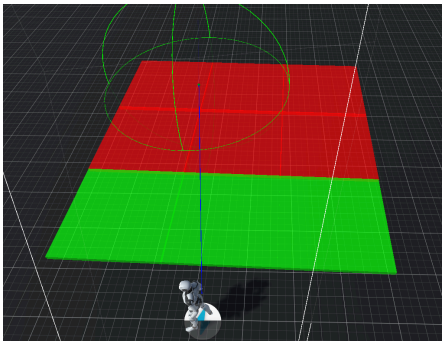
- Activate
    - Call it to activate the BuildingManager
    - It has 2 overload
        - One required a number which is the index of the BuildingElement from the given BuildingElementList
        - The other one needs a BuildingElement what will be used for building
- Deactivate
    - Call is to stop the BuildingManager
- ChangeMode
    - The BuildingManager has 2 modes. Build and Destroy. Usually the player can trigger the change of the modes by pressing the set button but this function gives you the possibility to change it from your algorithm based on your logic.
- GetMode
    - Return the current mode of the BuildingManager.
- SetBuildingElementList
    - Reset the list of the BuildingElements.
    - with 2 overloads
        - One just required the list and only change the list of the manager
        - The other one required an index and rest of the BuildingElement too, not just the list.
- SetRaycastCameraSettings
    - With this function the raycast setting can be changed.
    - It can be useful when the game changes its point of view.
- ResetCamera
    - Change the camera. Also useful during the POV change as SetRaycastCameraSettings.
    - Note that the setting does not contain the camera so if a setting with cursor setup changes to a settings containing a camera setup the camera should give through this function.
- EnableSandbox
    - Change the BuildingManager into sandbox mode.
- DisableSandbox
    - Disable the sandbox mode.
- SetMaximumBuildingElementCount
    - Set a new maximum amount to the Manager. Check the Material Functionality chapterfor more details.

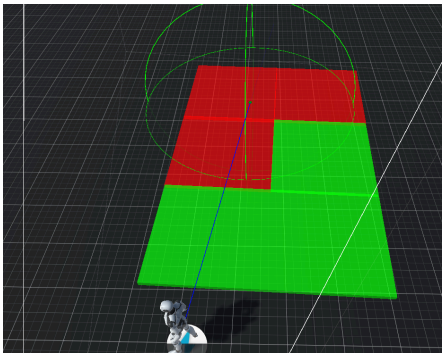
Material Functionality

Base Concept

Every building system uses a material functionality like if you want to build a stone wall you need to have an amount of stone to do that. This BuildingManager can achieve that with its maximum amount of placeable item count functionality. You can set a number which means how many BuildingElements can be placed at the same time with the Drag Building feature.
For example: you set the maximum number to 6 then if the player wants to place more than six elements with the Drag Building feature every element above 6 will be Blocked.
Note that this number can be zero so every element during the drag building will be blocked even if you are using just the simple building that one element will be blocked too. This mimics that when the player has not enough material neither for one element.

Refresh Blocking





The Blocked/Active element's count should be refreshed if you refresh the number with the **IBuildingManagerExternalInterface**'s **SetMaximumBuildingElementCount** function. If this function is called and the given number is different from the previous the block state of the elements will be refreshed.
Also the blocking state can be refreshed every time when the area has changed so new rows/columns are added or removed.
The Algorithm tries to fill up every row with Active elements and Block the others. It can cause different shapes based on the dimension and the amount of the temporary elements dragged area. Check the image above.

Manage MaximumBuildingElementCount

You can manage this number through the **IBuildingManagerTracker** and the **IBuildingManagerExternalInterface** interfaces with the following functions:

- IBuildingManagerExternalInterface
    - SetMaximumBuildingElementCount
- IBuildingManagerTracker
    - GetMaximumCountOfTMPElements

With the **SetMaximumBuildingElementCount** this maximum number can be changed from outside the BuildingManager.

The BuildingManager is using the **GetMaximumCountOfTMPElements** function to get this number if it needs to refresh that number. If the tracker is null or Sandbox boolean (On the BuildingManager script's interface) is set to false the number will be always uint.MaxValue.

Sandbox mode

You can set a boolean called Sandbox. If it is true then the whole material functionality is turned off and the algorithm is ignoring the set number. In the case of Sandbox, true every TemporaryBuildingElement should be Active. (It can be Blocked because of other reasons but not because of the MaximumNumber)

How to / Tips

If you want to achieve the material functionality then you should do the following steps:

1. When a new BuildingElement has been set with the BuildingManager's **Activate** function the manager will call back to the tracker's **GetMaximumCountOfTMPElements** callback.
2. Then you need to give a number like a stone wall was set and one stone wall can be placed for 2 stones. Also the player has 10 stones so the player can build a maximum of 5 walls. Then return 5 and the manager will be able to build up to 5 walls.
3. Everytime when the amount of the stones changed (so the placeable wall amount had changed), refresh the number with the **SetMaximumBuildingElementCount** function.
4. When the player has built some wall the manager will call the **BuildingElementPlaced** function on the tracker with a list of the built elements. Decrease the stone's amount with the material costs of the walls (in this example if the player builds 5 walls then decrease the number of stones with 10). You can use the **SetMaximumBuildingElementCount** function again to refresh this number after the successful building.

PreBuilt BuildingElement

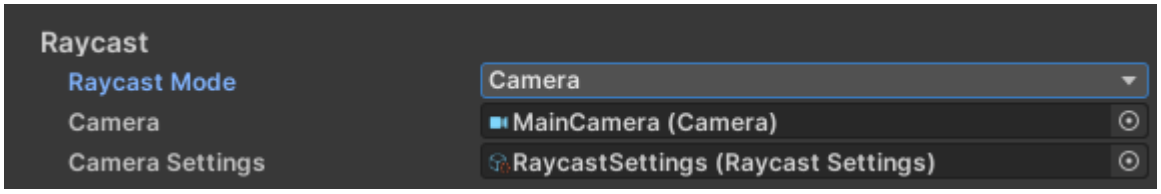
Sometimes that can be useful if some BuildingElement is already in the scene to show to the player where that element should be placed.
So what is a PreBuilt BuildingElement? The PreBuiltBuildingElements are basic BuildingElements but

- it can not collide with the player
- it has a special highlight material
- it handled specially by the AdvancedGridBuilding
    - The BuildingElements can not snap into its position just if it is the same element.
    - The BuildingElements can not snap to a position if every neighboring BuildingElement is PreBuilt.
    - The DragBuilding should handle the PreBuilt elements.
    - If a PreBuilt element from the same BuildingElement is nearby the algorithm will prioritize its position.
    - The BuildingElements can not be validated by neighbor PreBuiltBuildingElements.
- the PreBuiltBuildingElement should be destroyed if a right element is snapped into its position.

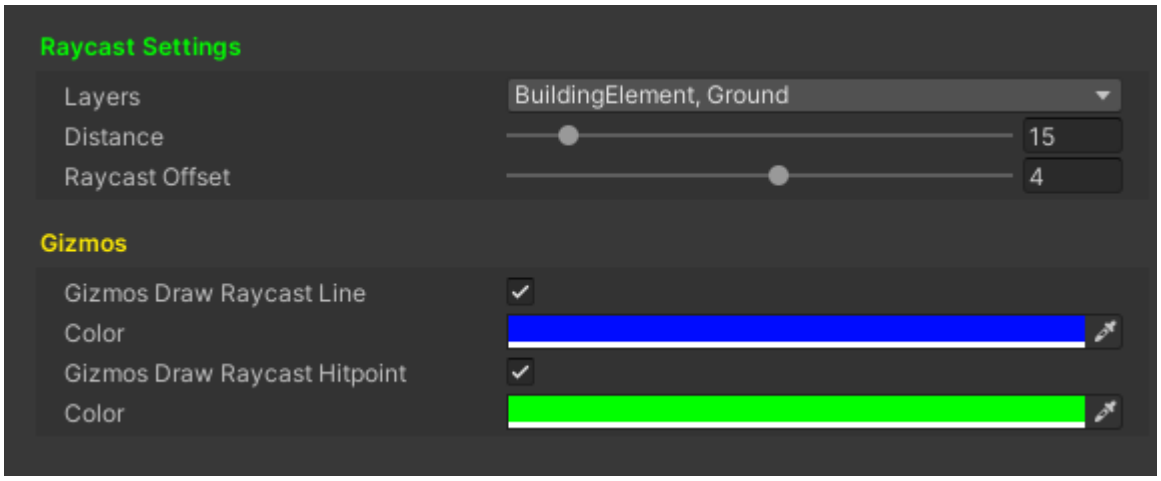
The main goal is that the developers are able to make such BuildingElements that can not be collided by the player because they are “not placed yet”. When the player is building something and a PreBuilt element with the same type is nearby the element should be snapped right into its position.

Raycast

The whole AdvancedBuildingSystem is based on the raycast. The Manager performs a raycast every frame and tries to find the best position based on the setting. But for that the raycast should be set up too.

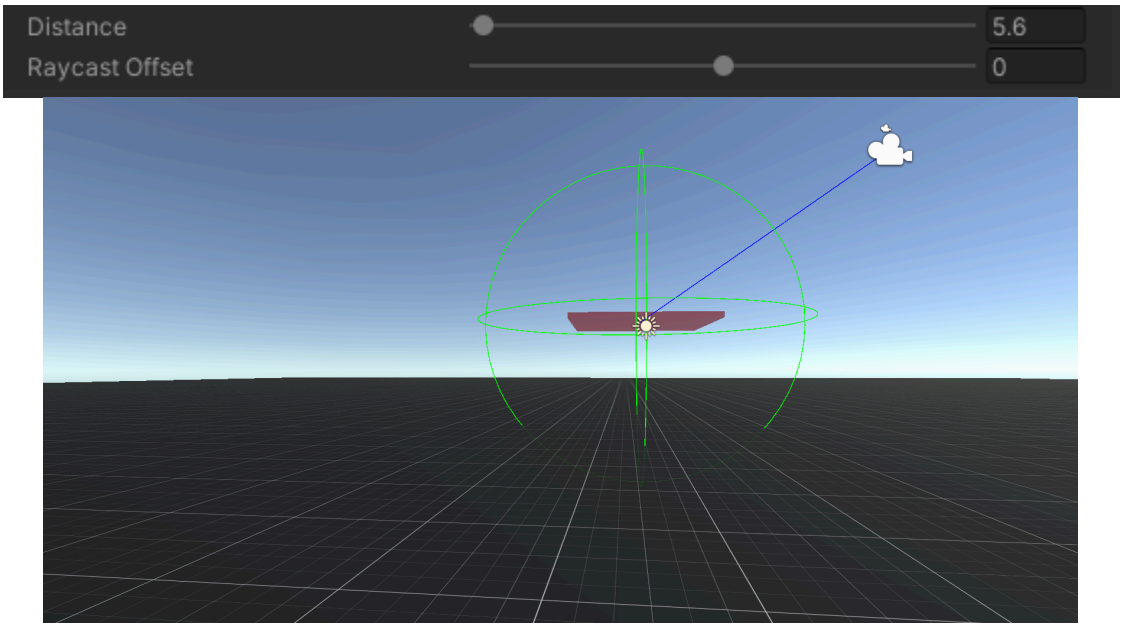


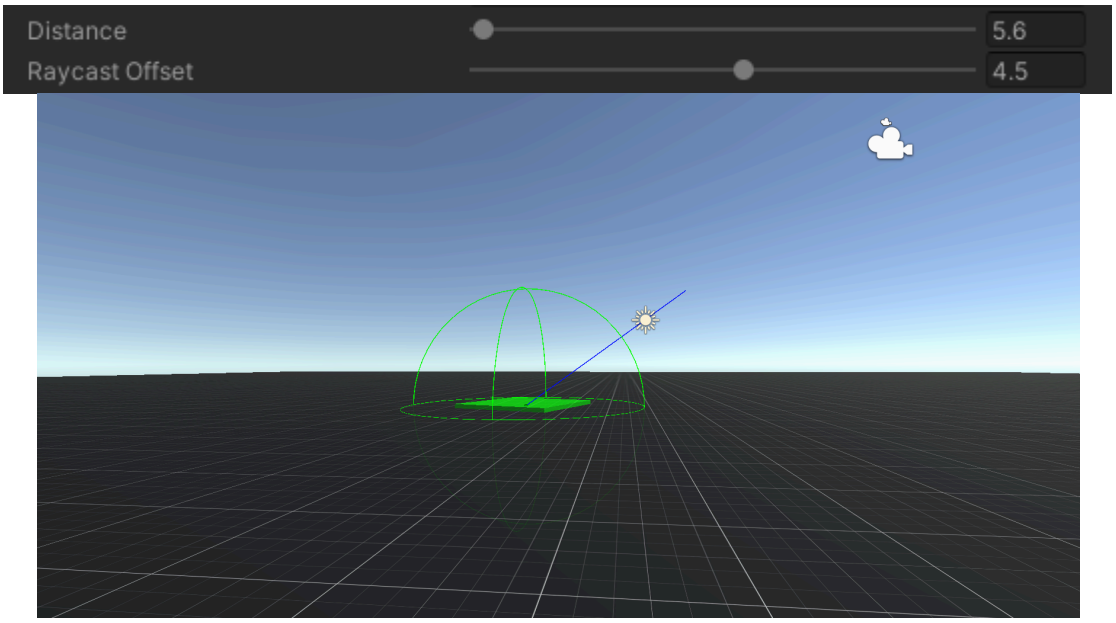
First of all the raycast has 2 different modes. The Cursor or the Camera. The Cursor mode means that the Manager is performing the raycast on the screen from the cursor's current position. It is used for those projects when the cursor should be used for building like drag and dropping a building element into the scene. The Camera is used typically in those projects when the camera position is changing when you move your cursor like FPS or a lot of TPS projects. In Camera mode the center of the screen is used for the raycast.



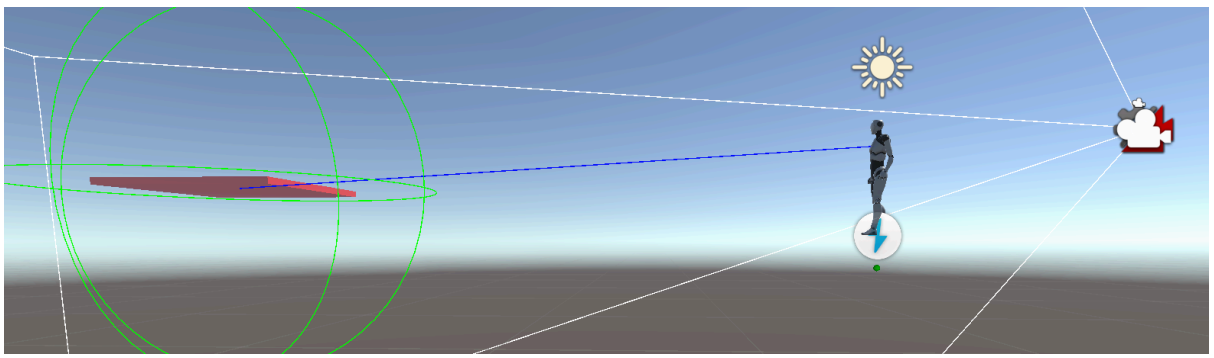
The following properties should be set up:

- Layers
  These layers will be hit by the raycast. Recommended to add the BuildingElemenets' layer too.
- Distance
  How far should the raycast go?
- Raycast offset
  An offset can be set to give the possibility to modify the startpoint of the raycast.
- Gizmos Drag Raycast Line
  Disable the raycast line's Gizmo
- Color (first)
  The color of the raycast line
- Gizmos Draw Raycast Hit Point
  Draw a sphere on the hitpoint of the raycast.
- Color (second)
  The color of the hitpoint Gizmo.





In this example above you can see with the camera offset the the raycast start position is changed. It is typically used to set a minimum distance between the BuildingElements and the camera. Also a good usage in TPS games is to set the raycast start point to the character instead of the camera.

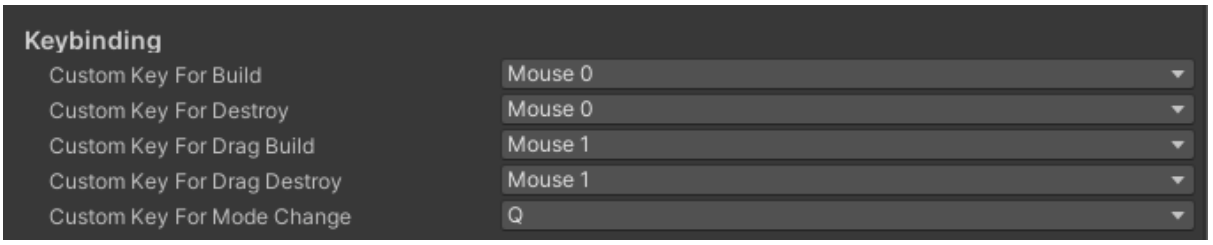


Keybinding

The BuildingManager will handle the inputs of the player. For making it more customizable the keybindings are customizable.

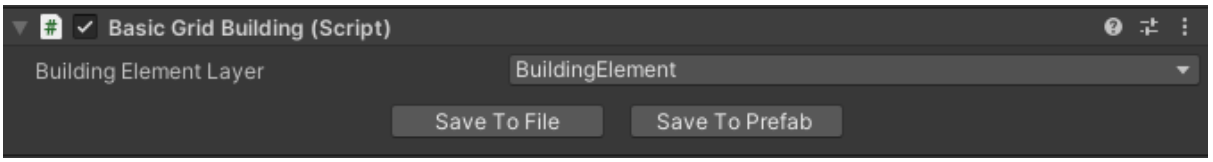
- Simple Build
    - For the Simple Build the default key is the **Mouse 0**.
- Simple Destroy
    - Originally the same button is used for these two features because these 2 are the basic functionality of the BuildingManager just in different modes.
    - **Mouse 0** by default
- Drag Building
    - The default button is **Mouse 1**
- Drag Destroy
    - Similarly to the Simple Destroy this feature's default button is equal with the Drag Building.
    - But the possibility is available to change it and make it different from the other features.
    - **Mouse 1** by default
- Change Mode
    - Changing mode between destroy and build can be done by default with **Q**.
- Rotation
    - The rotation is done with the **scroll wheel** by default and currently it can not be changed.
    - It is used to rotate the BuildingElements.

These keybindings can be done on the BuildingManager's UI.



Persistency (Save/Load)

Of course if anything is created in a game that should be able to persist. The AdvancedBuildingSystem does not provide any persistence functionality, just the possibility to save and load the Buildings that were created by the player. For making the development easier the only persistence feature is available on the UI. The Buildings has this dave functionalities on their GUI:



Also you can load back the saved Buildings from the file with the BuildingLoader tool which will be explained later.
Every Building is inherited from the SaveableMonobehaviour which has this abstract function called GetPersistedData. This function gives back a PersistedDataClass that can be persisted by the developer or you can use the PersistencyManager to persist the Building into JSON string which can be saved easily.
This “Save To File” button is using the same methode.

```
string dataToSave = PersistencyManager.ToJson(target.GetPersistedData());
```

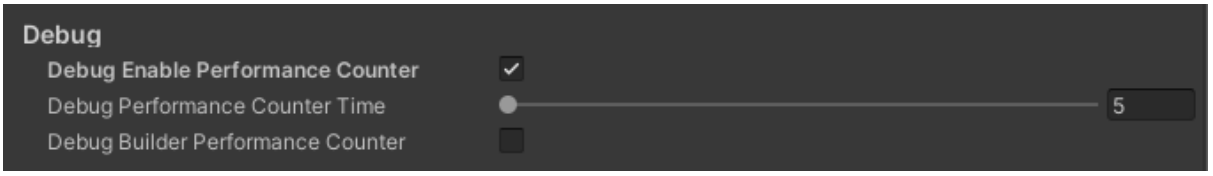
The loading is the same just backward. The PerssitencyManager can be used to create the PersistedData from the JSON text and use the Building’s LoadFromPersistedData function to create the Building.
Or just use the CreateFromJSON function which is a static function of the Building class which can create a Building from a json text. Just like the BuildingLoader tool:

```
private void Load ()
{
    if (Building.CreateFromJSON(m_PersistedData, m_Parent, m_Container))
    {
        EditorGUILayout.HelpBox("Success!", MessageType.Info);
        Logging.Info("BuildingLoader", "Load", "Loading of Building was successful!");
    }
    else
    {
        Logging.Warrning("BuildingLoader", "Load", "Loading of Building has failed!");
    }
}
```

Debug

Currently 1 Debug functionality is available by the BuildingManager which is a performance related logging. If the Performance Counter is enabled the BuildingManager measures its performance and writes out some statistics.

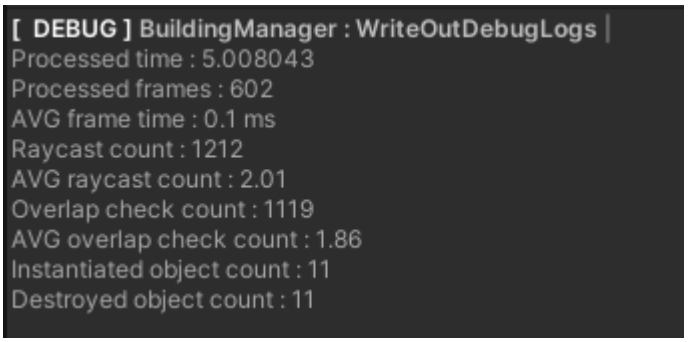
Note that every Debug related part of the code is behind the UNITY_EDITOR macro which means that it should not be compiled into the game. So these functionalities should have 0 impact on the final compiled game!



On the BuildingManager's UI you can enable the "Performance Counter".
Then a number can be seen which is the time of the cycle used by the BuildingManager. The manager will measure its performance for that time and write out to the log some statistics.

Currently the following messages can be written out by the manager:

BuildingManager's Performance statistics.
This log contains basic measurements about the manager's Update function.

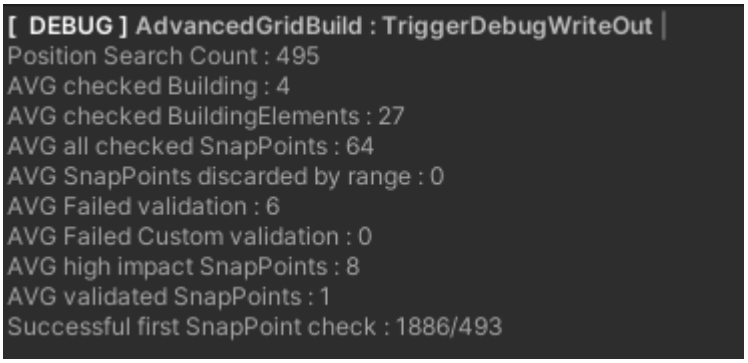


1. Processed time
   How much time is exactly measured. In this example the Performance Counter Time was 5 seconds.
2. Processed frames
   How many frames are processed during the measured time.
3. AVG frame time
   Average process time of the update function.
4. Raycast count
   How any raycast performed during the measured time.
5. AVG raycast count
   Average amount of the raycasts during a single frame.
6. Overlap check count
   How many overlap checks performed during the measured time.
7. AVG overlap check count
   Average amount of the overlap checks during a single frame.

8. Instantiated object count
   How many objects created by the BuildingManager during the measured time.

9. Destroyed object count
   How many objects destroyed by the BuildingManager during the measured time.

AdvanceGrid Search Statistics
This message contains some statistics about the calculations of the AdvanceGrid Search's algorithm. This is only written out if the AdvancedGrid Search has been used.



10. Position Search Count
    How much search is requested by the BuildingManager. One search call should happen in one frame.

11. AVG checked Building
    On average how much Building had been processed per search.

12. AVG checked BuildingElements
    On average how much BuildingElement had been processed per search.

13. AVG all checked SnapPoints
    On average how much SnapPoints had been processed per search.

14. AVG SnapPoints Discarded by range
    On average how much SnapPoints were discarded because of the range. (If the algorithm already found a valid SnapPoint then every snapPoint will be discarded if these are further than the found valid one. The algorithm tries to find the closest SnapPoint.)

15. AVG Failed validation
    On average how many position validation fails in a single frame.

16. AVG Failed Custom validation
    On average how many position custom validation fails in a single frame.

17. AVG high impact SnapPoints
    On average how much SnapPoints had a high performance impact. A SnapPoint is treated as high impact if it had been validated. The validation process required some high calculation based logic that can have an impact on the performance such as raycast, collision check and a lot of iteration on different containers. For better performance the BuildingManager tries to minimize the amount of these SnapPoints. It can be achieved by finding the good (not the best) valid SnapPoint as fast as possible because after that it can discard most of the SnapPoints by range.

18. AVG validated SnapPoints
    On average how much SnapPoint had been successfully validated by the code. The

“High Impact SnapPoint” shows how much SnapPoints were validated and this counter shows how much was successful from these.

19. Successful first SnapPoint check: X/Y"
    On average how many First SnapPoint checks were successful. Before the manager tries to process the SnapPoints it tries to Validate the nearest SnapPoint. If it was valid then the process of the Building stops because this SnapPoint should be the nearest one. Every Other one can be discarded. Only 1 is added to the ”AVG all checked SnapPoints” and 0 is given to the “AVG SnapPoints Discarded by range” because we do not know at this moment how much SnapPoint will be discarded.
    The X is the count of all processed Buildings and the Y is the time when the first SnapPoint check was successful.

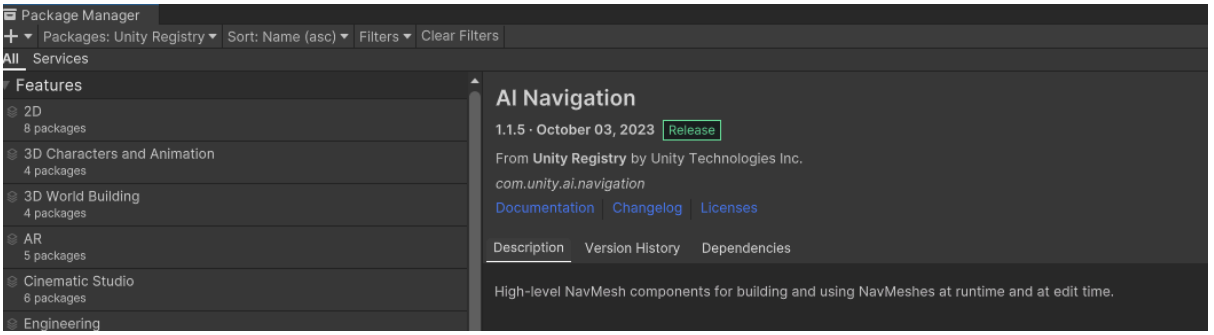
Support

NavMesh

The AdvancedBuildingSystem has just a small support of the NavMesh package.

Setup

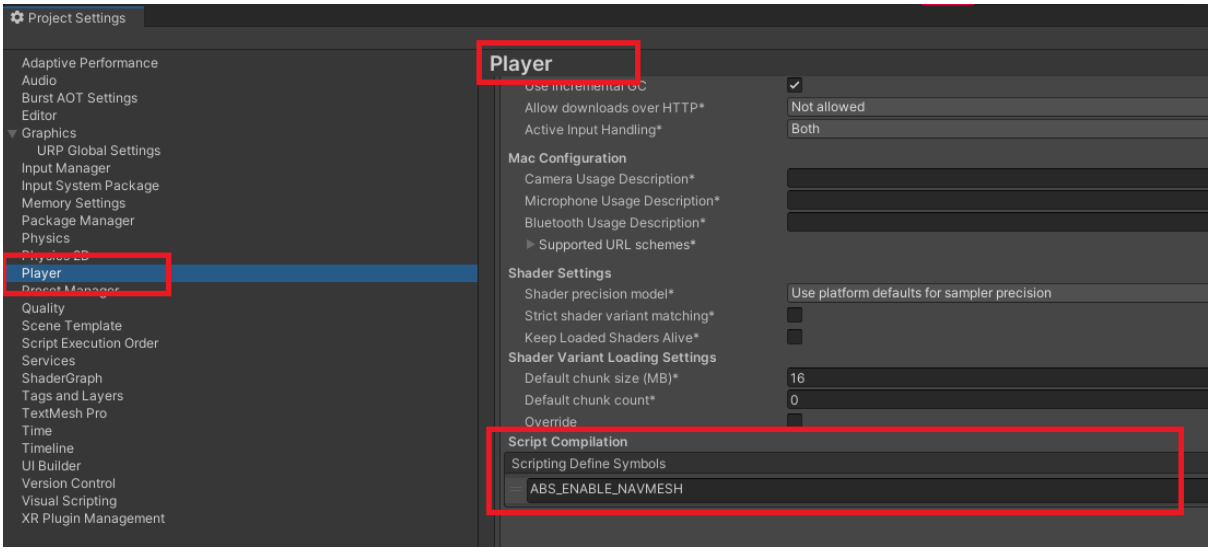
First of all install the “AI Navigation” package from the “Unity Registry”:



All of the features are behind a Macro what should defined to enable the codes like:

```
#if ABS_ENABLE_NAVMESH
        private NavMeshObstacle m_NavMeshObstacle = null;
#endif
```

Edit > Project Settings > Player > Script Compilation > Scripting Define Symbols



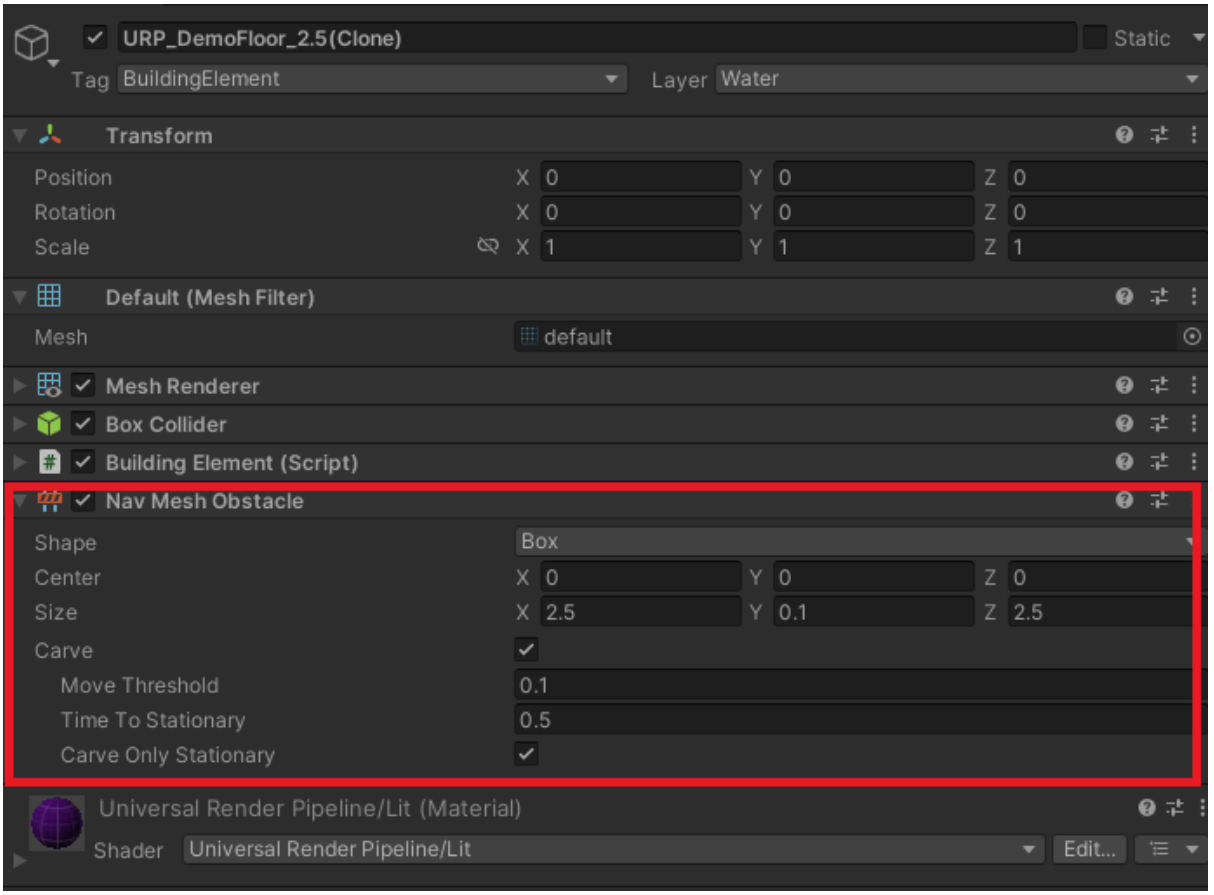
After that the symbol is defined the code parts will be available:

```
#if ABS_ENABLE_NAVMESH
        private UnityEngine.AI.NavMeshObstacle m_NavMeshObstacle = null;
#endif
```

Features

At this moment the only NavMesh supporting feature is that when the element is placed (it is not temporary) the BuidlingElement script will create a Nav Mesh Obstacle component. The Shape will be “Box” and the Size is the same with the BoxCollider’s size.

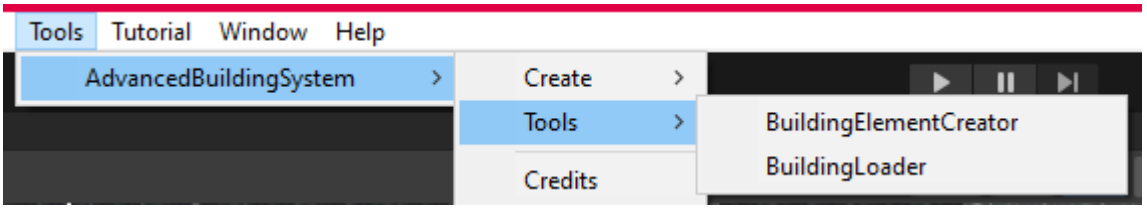
Also this component will be enabled and disabled next to the colliders with the **EnableCollider** function on the BuildingElement.



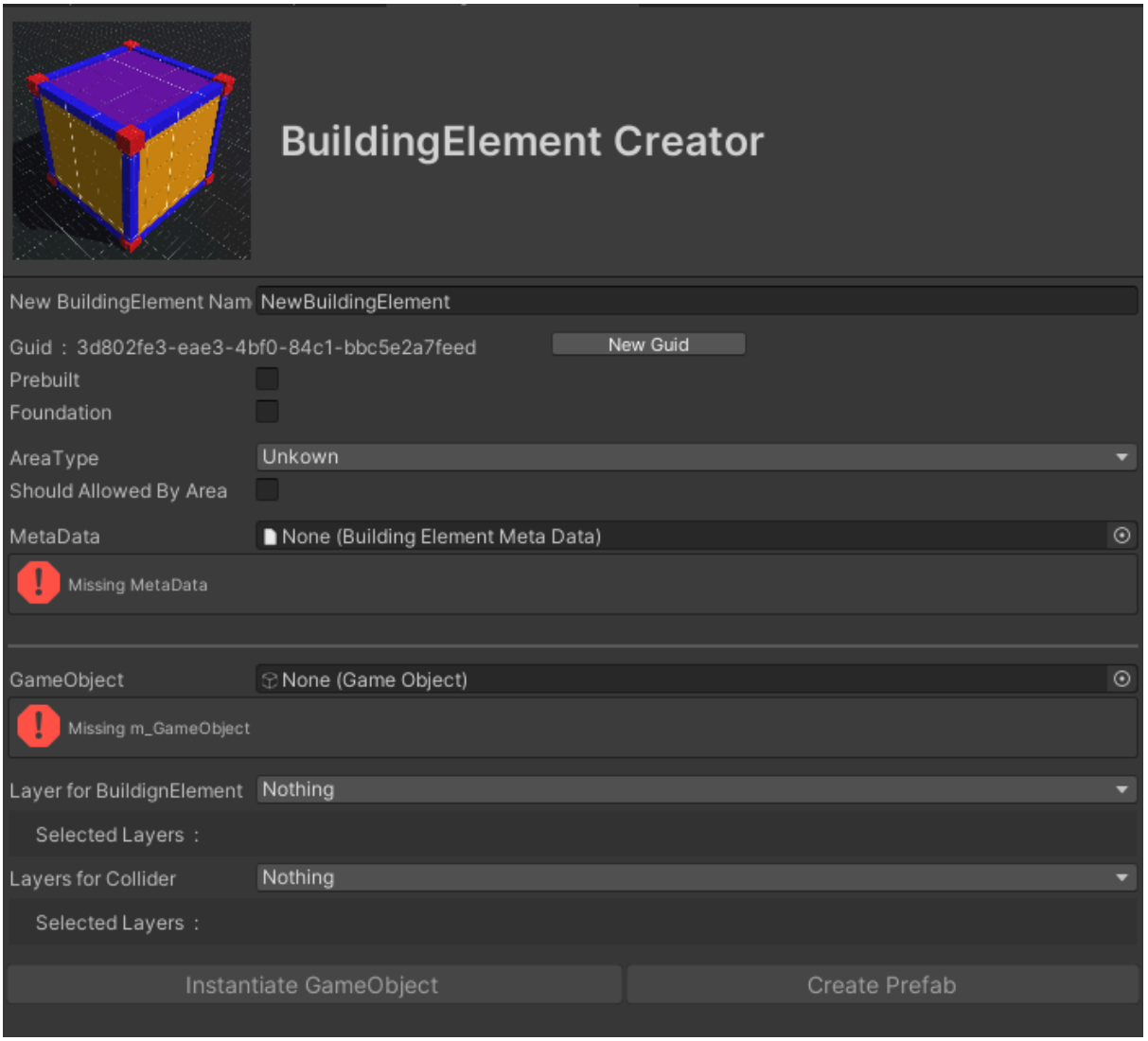
Tools

Building Element Creator

The BuidlingElementCreator is a specific tool created to the AdvancedBuildingSystem. The goal of the creator is to create a tool that can be used to create a BuildingElement. It is made to make the developers workflow with creating the elements faster.
The creator can be found at:



UI:



The setup is easy. You have to provide the required properties to the tool and you can choose from the 2 options.

- Instantiate GameObject
  With this option the BuildingElement will be instantiated in the scene.
- Create Prefab
  With this option a prefab should be created to the selected path about the BuildingElement.
  Note that for this the tool should instantiate the BuidlingElement just like the other option but after the creation of the prefab the GameObject is removed immediately.

When you set the layers keep it in mind when you are using the manager the raycast should hit the BuildingElement.

Building Loader

The BuildingLoader is a complementary tool for the persistence functionality of the BuildingManager. If you have a loading file for a Building with this tool you can load it into your scene. Find it under the Tools > AdvancedBuildingSystem > Tools.
The tool works in that way you have to select a PersistedData file and a BuildingElementList (which contains every BuildingElement from the Building) should be provided and just press the Load button. It will load the persisted Building into your scene. Also you can give a parent object from the Hierarchy to be the parent of the Building.
Note that this tool can be used only when the game is running. If you do not want to run the scene this tool works totally fine with an empty scene.