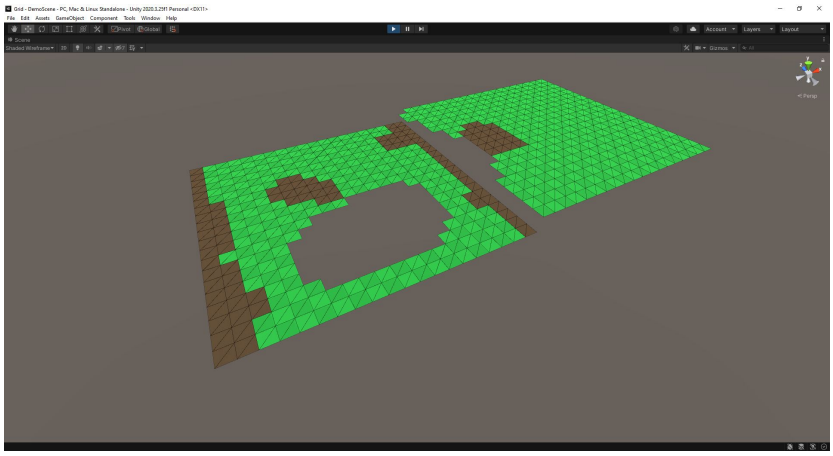
Grid Builder

Index:
Grid Square
Grid Selector
Object Placer
Select Object
Grid Object
Remove Mode
Object Remover
Scripting
Limitations
contact



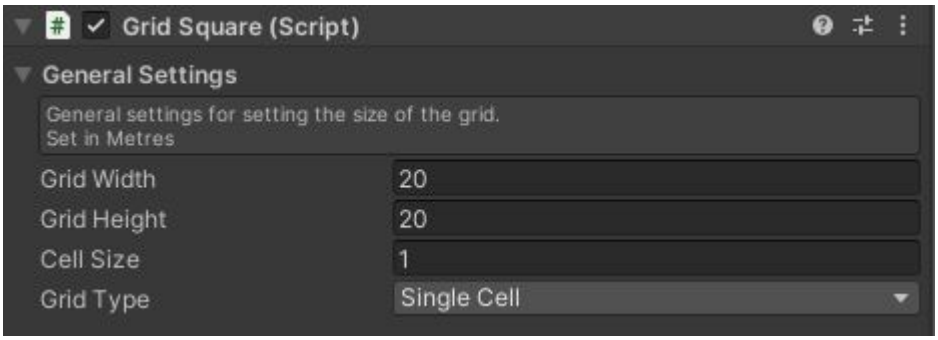
Overview

Intro:
Grid Builder is a quick and convenient way of creating many multi-purpose grid systems with built in placement and removal logic. Tower defence, strategy, real-time strategy, simulation, construction, table top games plus many more genres would find this system useful. With simple mode, you could even develop a painting or creation app using the cells as pixels. See **Limitations**.

Features

Grid Square

General Settings:
Here you can adjust your basic settings of the grid itself.



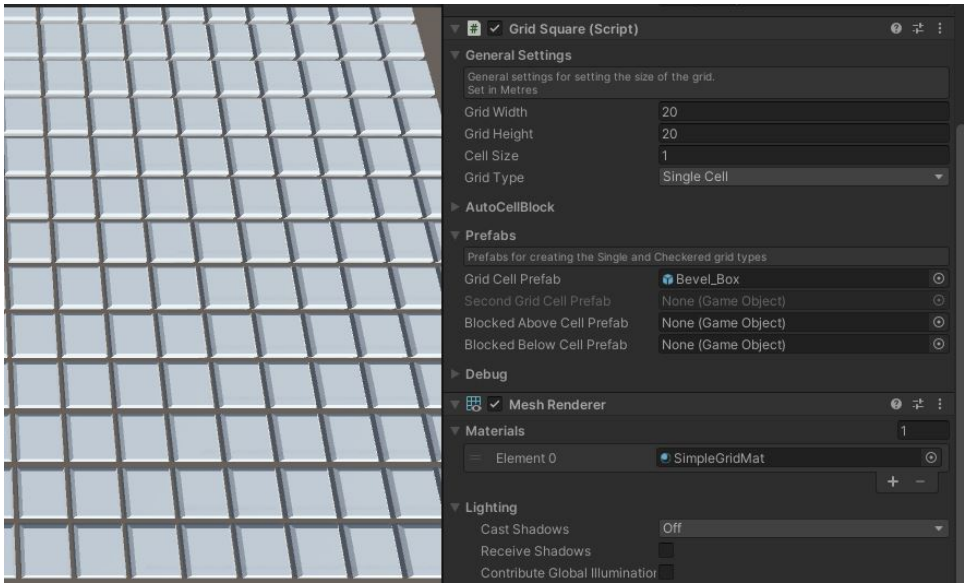
Grid Width
How many grid cells wide should the grid be on the X axis.
Please see **Limitations** for advice on maximum sizes.

Grid Height
How many grid cells high should the grid be on the Z axis.
Please see **Limitations** for advice on maximum sizes.

Cell Size
The size of each individual cell in metres. Generally a good idea to keep this to even whole numbers as this not only makes the calculations simpler, but also provides a good size structure for the game.

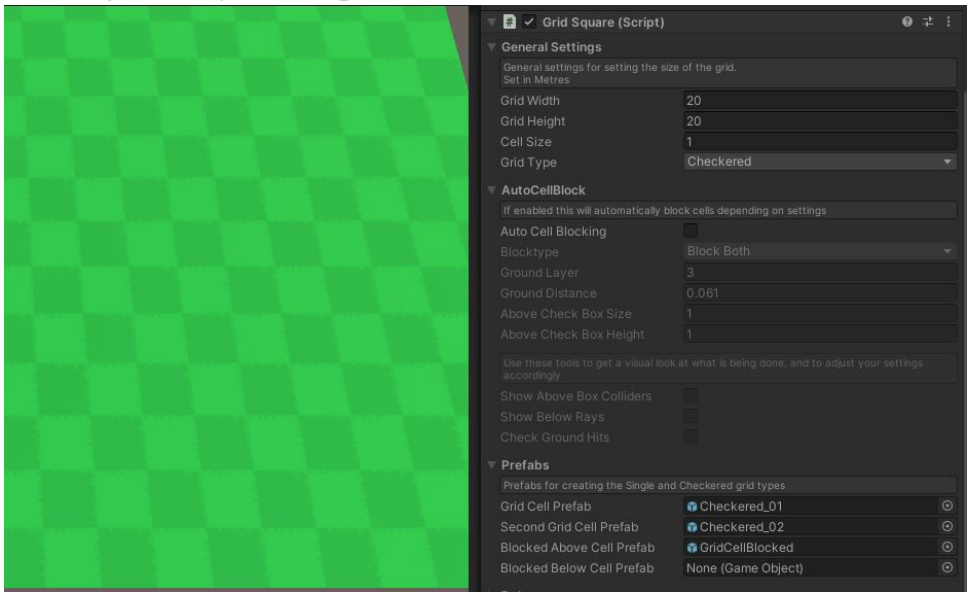
Grid Type
You have 3 options to choose from, Single cell, Checkered and Simple.

Single Cell
Creates the grid out of the prefab assigned in the gridCell prefab slot.



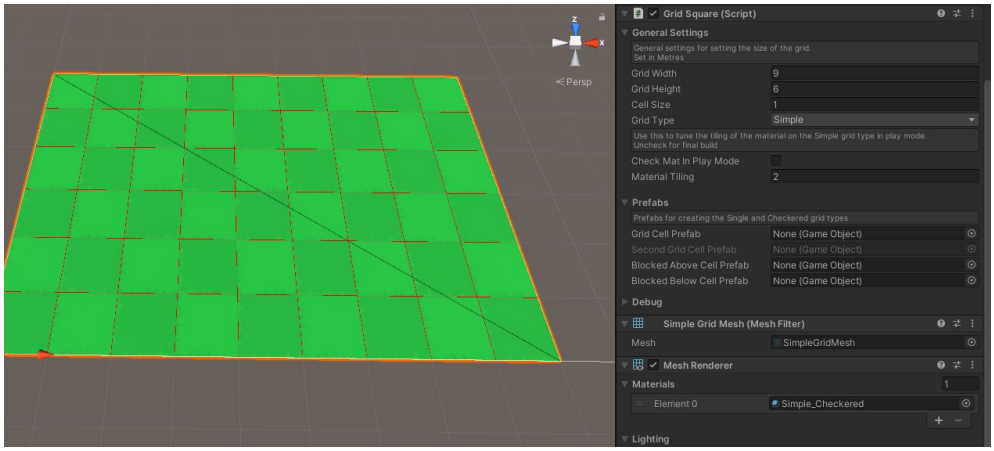
This mode supports Auto Cell Blocking and also use of 2 blocking prefabs.

Chequered
Creates a check pattern out of two assigned prefabs. It is identical to the Single Cell type except for the second prefab variation. This mode can allow for interesting patterns if using 3D objects as prefabs.



This mode supports Auto Cell Blocking and also use of 2 blocking prefabs.

Simple
Looks similar to chequered in the picture, with one major difference, the grid density. The grid is created at runtime to make up a single plane, attached to the meshfilter with a tiled material for the design.



This mode does **not** support Auto Cell Blocking so should be used on simple grids or grids that only require a flat 2d look with no cells that need to be removed.

If in **Simple** mode you will get an extra 2 options to adjust the material.

Draw Simple
This will only draw the boundary lines for the grid instead of every row and column. When working with extreme size grids this can be useful for keeping track of the object while still keeping a decent frame rate to work with.

Check Mat Runtime
Will simply allow to view the material as you edit **Tile X** and **Tile Y** variable during gameplay. This will then allow you to achieve some variation even with a simple material.

Tile X
Tiles the current material on the X axis. A value of 1 would tile the current material once for each Grid Width value. If using a chequered material, as provided with the project, a value of 2 is used.

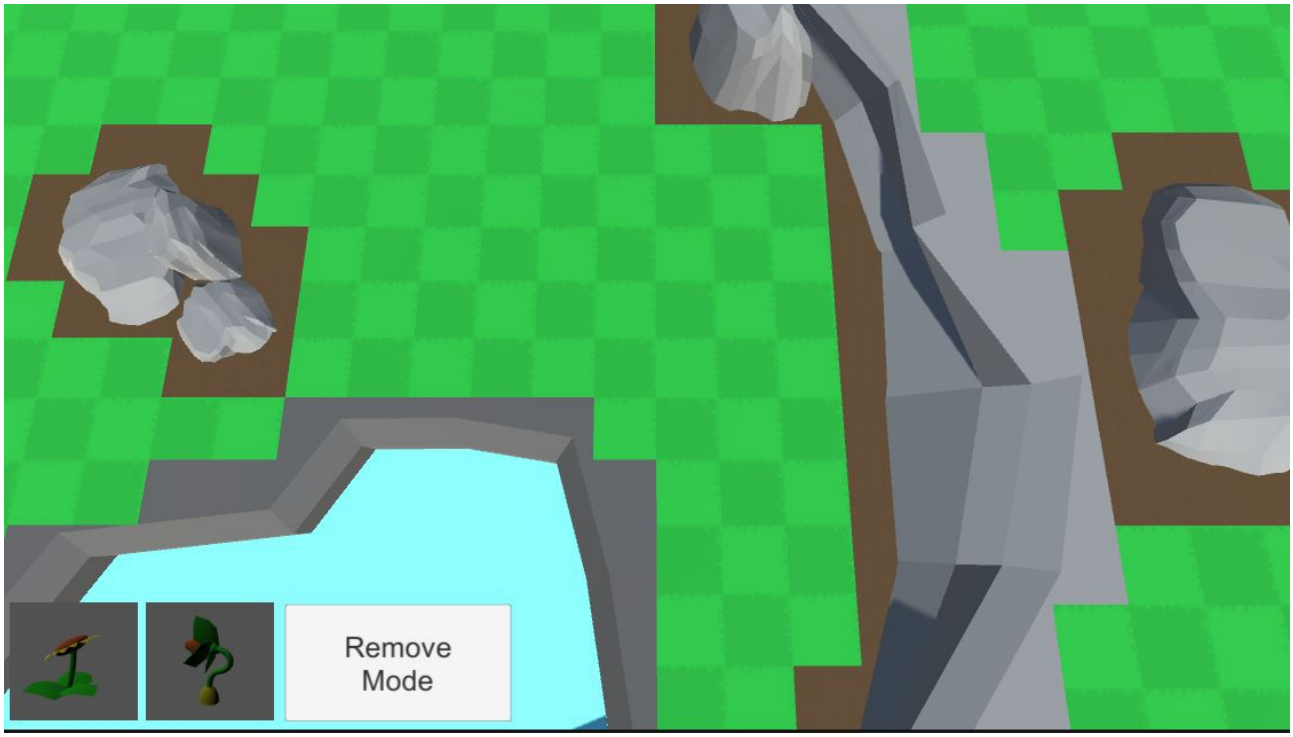
TileY
Tiles the current material on the Y axis. A value of 1 would tile the current material once for each Grid Height value. If using a chequered material, as provided with the project, a value of 2 is used.

Auto Cell Block:

This feature blocks cells by marking the cell as used. This in turn gives some freedom when building grids around other objects like rocks or cliffs. It also has the innate ability to create the cells in any shape you like, such as a layout of a house, town, ship, or any other design you choose.

It can block cells checking for objects on or above the grid.
It also can check to see if there is anything underneath the grid.

You can of course have the choice of prefabs to place or not place in both instances.



A very simplistic design using a chequered grid with Auto Cell Block on with two prefabs for placeable objects and one prefab for the blocked cells(above). Note the pool and cliff face have no generated cells even though the grid covers them.

Auto Cell Blocking
Turns the auto cell blocking feature on or off.

Block Type
This allows you to choose what to block, objects above the grid, void beneath the grid, or both in combination.

Ground Layer
The ground layer is the layer your actual terrain or ground objects will have to be on for this feature to work. Objects that are being blocked above the grid can be on any layer. For removing cells if there is a void, the terrain will need to be on this assigned layer. This is an integer that represents the layer. See Unity documentation about creating a new layer and adding your objects.

Ground Distance
The ground distance is how far the below rays check to find the ground. The rays should always be going just past the ground to work properly. See **Show Below Rays** to make this easier.

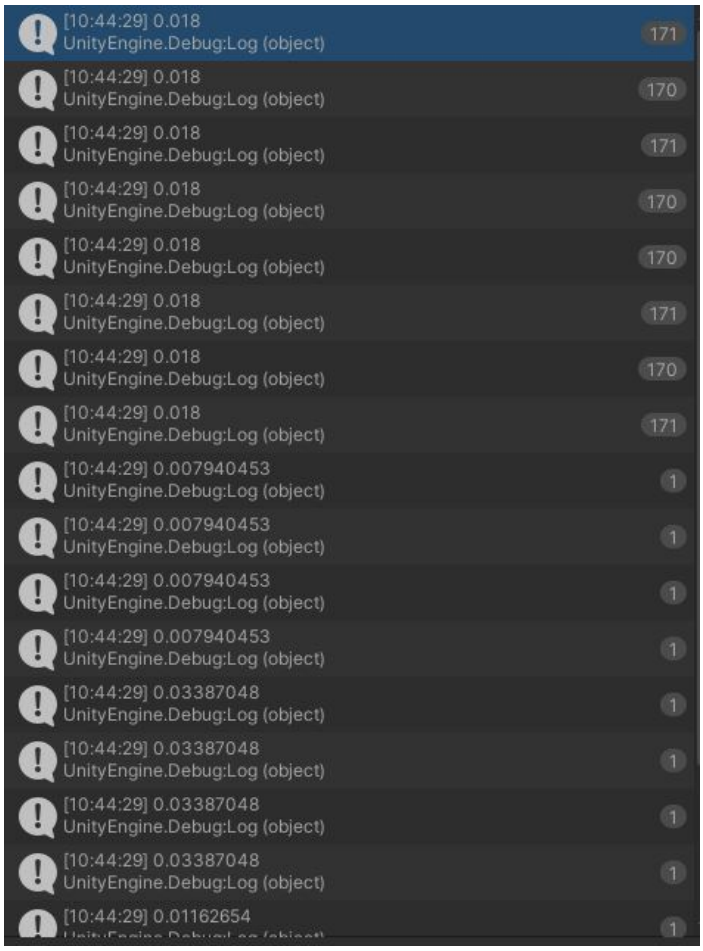
Above Check Box Size
This increases the size of the box created from the centre of each cell. The larger you create this box, the more removed cells or space around the collision objects you will have. See **Show Above Box Colliders** to get a better understanding of how this is working.

Above Check Box Height
This will do alter the Y value of the checkBox and find objects higher and lower. If you have bridges and would like to remove the cells underneath them, increasing this value may help. Again checking **Show Above Box Colliders** would help see how this is being applied.

Show Above Box Colliders
This just gives a visual representation of the box colliders created from the centre of each cell.

Show Below Rays
Shows the rays created from each cell point. They should intersect the ground layer geometry slightly.

Check Ground Hits
Checking this will give you a debug.log of all the distances of all the points in the grid. Most commonly the log with the most same hits will determine the distance between the grid, and the largest flattest part of your terrain. You can increase this value slightly and then use it for your **Ground Distance.**



Quite clearly here the grid is set 0.018 metres above the object in the **Ground Layer**. So perhaps a value of 0.02 could be used as the **Ground Distance**.

Prefabs:

The **Simple** grid type does not use this method of creating the grid so prefabs are ignored.
The prefabs are the cells to be used if they are filled. If they are not, the grid will still run, but just leave the cell without a physical object in it. If **Auto Cell Blocking** is on, this again will still run without prefabs, and cells will still be marked as taken, but without anything physical to represent it.

Grid Cell Prefab
Places a selected prefab in any cell that is not marked full at runtime.

Second Grid Cell Prefab
If the **Grid Type** is set to **Chequered** a second prefab is used to create the pattern.

Blocked Above Cell Prefab
If **Auto Cell Blocking** is on, places this prefab on any cell that is not placeable above the grid. If the grid is blocked by rocks or mountains, this would place this prefab around them to create a border.

Blocked Below Cell Prefab
If **Auto Cell Blocking** is on, places this prefab on any cell that is not placeable below the grid. If there is a cliff face or a drop to a body of water, this would place the prefab there. In most cases this would be left clear.

Debug:

Two simple tools to use for use in debugging.
Remember these will impact performance if used on large grids as they draw text in the scene view.

Draw Cell Positions
Checking this will enable you to view the centre position of each cell, and ultimately the position that is being checked to see see if the cell is empty or not.

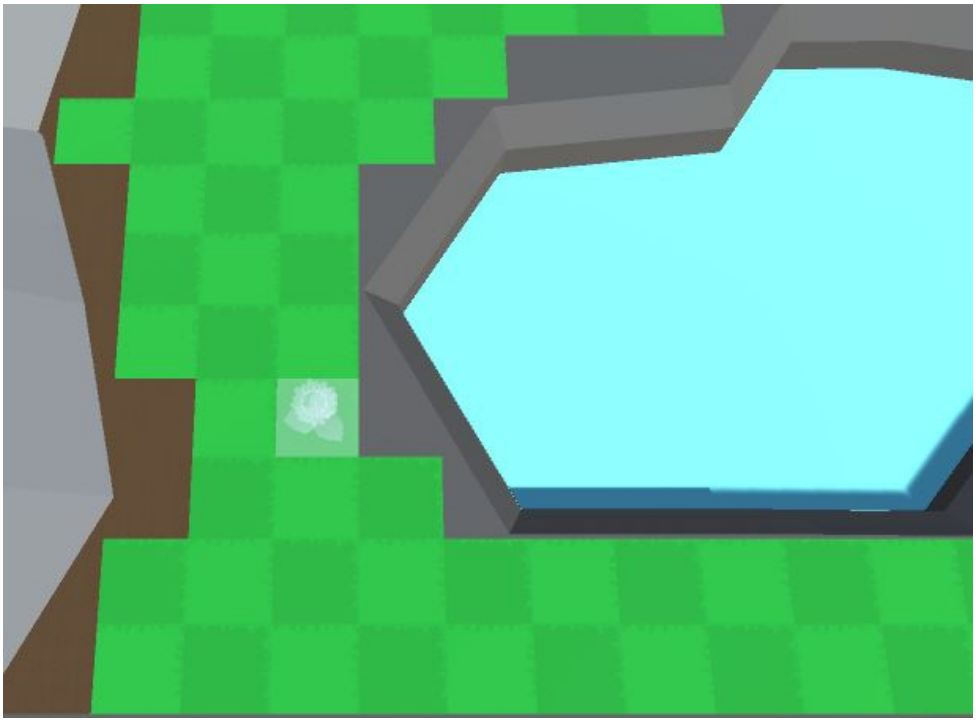
Draw Grid Positions
Checking this will enable you to view each point created on the grid.

Grid Selector

The grid selector object needs a mesh filter component and a mesh renderer. Both will be added when adding the Grid Selector script to an object. You can find meshes in /Meshes/GridSelectors and the material in /Materials/GridSelector.

This allows you to see a visual representation of moving on the grid. It also allows you if an Object Placer is assigned to place objects on the grid.

The Grid Selector if paired with an Object Placer creates a preview Object to create a 'where to place' visual. This Grid Selector needs to be passed an object for this.



General Settings:

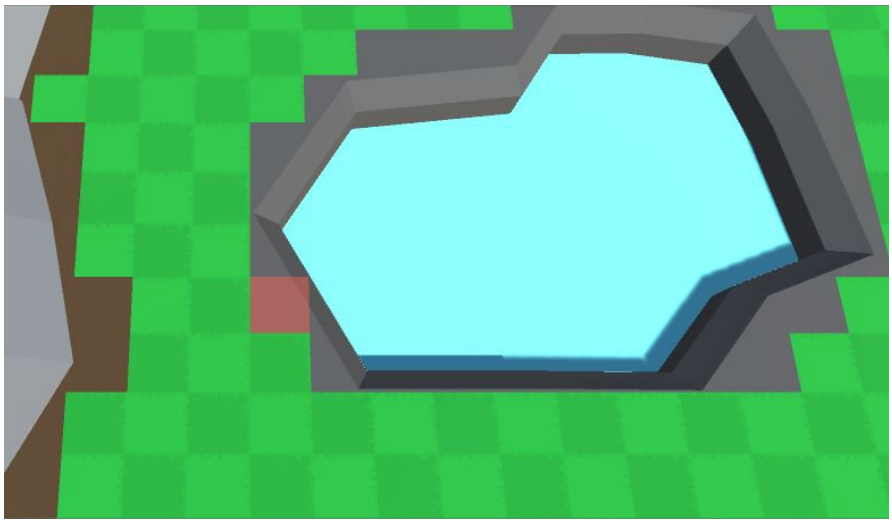
Here you can adjust your basic settings of the Grid Selector itself.

Smooth Move
This enables the selector to transition from one point to the next rather than teleporting to its position.

Move Speed
Sets the speed of the **Smooth Move**. A lower value slows down the transition.

Hover Distance
How much on the Y axis to place the Grid Selector object.
It will need to be slightly above the grid in order to see it.

Invalid Placement Feedback
This will turn the selector material to **Invalid Placement Mat** when hovering over cells that are taken and then back to the Selectors own material when on a free cell.

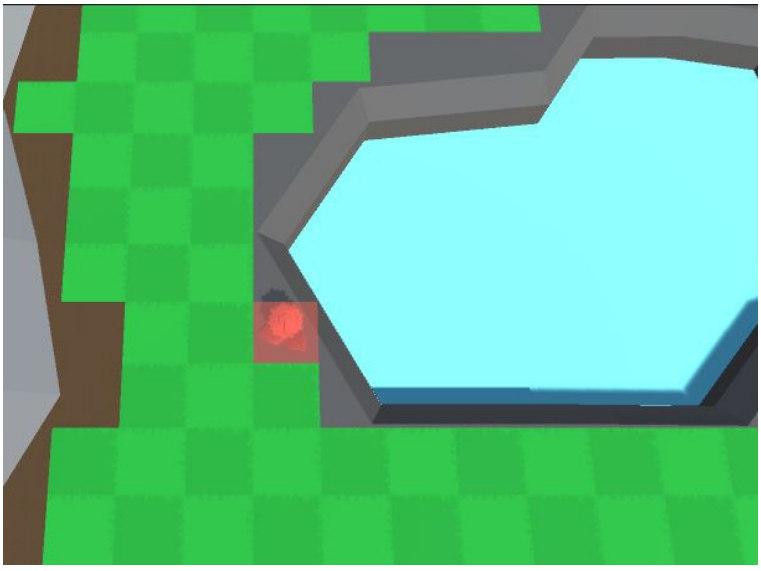


Invalid Placement Mat
This material replaces the selector material if **Invalid Placement Feedback** is on. If an **Object Placer** is assigned, and **Show Invalid Preview Obj** is checked, it will also turn the object to the same material if the cell is invalid.

Object Placer
Assign the Object Placer object here to place objects on the grid.

Obj Preview Mat
This will change to the preview objects material to the one assigned here. If nothing is assigned, it will just use the object material.

Show Invalid Preview Obj
If checked, will show the preview object with the **Invalid Placement Mat** on it on cells that are unavailable.



Drag Build
If the left mouse button is help down, will continuously build on available cells as you move the mouse over them. If this is unchecked, only builds on clicking the left mouse button.

Object Placer

The Object Placer script is a very simple one function script that actually handles the placing of the object down on the Grid. It marks the cell as now full and creates an instance of the preview object from the Grid Selector.

It also has the function of attaching the Grid Object script to any objects being placed. This simple script just stores a reference to the grid it is placed on. This is used by the removal script to clear that cell on that particular grid.

Select Object

The Select Object script should be placed on a button object. When clicking the button this will pass the assigned object to the Grid Selector (if in scene) for use as a preview object and placing it.

Prefab
The object to be passed for placing and preview on the grid. Needs to be a prefab.

Grid Object

This script is applied to any objects placed down on the grid. Its one and only purpose is to store a reference to the grid is was placed on.

Remove Mode

Remove mode enables the Object Remover script and should be placed onto a button. It also disables any previews from the Grid Selector.

Object Remover

The Object Remover when enabled, will highlight any object with the Grid Object class attached to it presuming the **Remove Preview Mat** is assigned. The Object Placer adds this class automatically on object creation. When hovered over one of these objects, and left clicking will delete the objects themselves, and remove them from the Grid Square they are assigned to.

Remove Preview Mat
The material used to replace the hovered objects material.

Camera Movement and Get Prefab Image

There are two small extra scripts attached with the package for demonstration purposes only. A very simple camera movement script and a script for grabbing the asset thumbnail from the unity editor. Feel free to discard or use these as you see fit.

Scripting

There are some useful public functions included in the scripts that come with this package which could come in useful in whatever application this is being used for.

This system uses Clamped Vector3's as keys in a dictionary with the corresponding game object if there is one as the value. If there is no game object, the value will be null.

For the purposes of rounding and other issues, the positions created in the dictionary as well as the positions for the Grid Selector are clamped to 3 decimal places. So a position that would have been 1.567943 will now become 1.567000.

Grid Square

public bool CheckCellStatus(Vector3 cellPos)
This function returns true if the given key (cellPos) is found in the dictionary and is that positions value is null. It will return false if that keys value is any object except for null.

public void ChangeCellStatus(Vector3 cellPos, GameObject obj)
This will assign a value(obj) to the key(cellPos) provided. Thus marking that cell as full if the object is not null. Send null to 'empty' the cell.

public GameObject GetGridObjContainer()
Will return this grids container of objects that have been placed. The container is created at runtime.

public float GetCellSize()
Returns this grids cell size.

public float GetGridHeight()
Returns this grids grid height.

public float GetGridWidth()
Returns this grids grid width.

public Dictionary<Vector3, GameObject> GetCellsDictionary()
Returns this grids Dictionary. There is a debug.log for each entry in the dictionary.

Grid Selector

public void ChangeObjMat(GameObject obj, Material newMat)
This will loop through all children of the obj including itself and through all assigned materials and assign the newMat to each of them.

public void DeselectPreview()
This function destroys the preview object from the hierarchy and sets the object to place as null.

public void SetGameObjectToPlace(GameObject obj)
This simply takes in the obj, and sets it for the Grid Selector to preview or place the object.

public Vector3 GetSelectorPosition()
This returns the current clamped position of the grid selector. This value can then be used to find out if the cell is free using gridSquare.CheckCellStatus(Vector3 cellPos)

Object Placer

public void PlaceObject(
GameObject obj,
Vector3 placePos,
GridSquare grid,
int layer)
This function places the object on the grid, and changes the layer back to the original layer the object was on.
It also adds the GridObject script to it, to keep a reference to the grid it is on.

Select Object

No public methods.

Grid Object

public void SetObjectGridSquare(GridSquare grid)
Sets this objects grid to the provided one. Is called when the object is placed.

public GridSquare GetObjectGridSquare()
Returns this objects grid.

Remove Mode

public void DisableRemoveMode()
Turns the Object Remover script off.

public void EnableRemoveMode()
Turns the Object Remover script on.

public void ChangeText()
Updates the text on the attached text component depending on if the Object Remover is enabled or not.

Object Remover

No public methods.

Limitations

Grid Size
Although this system is set up to run smoothly on most devices, increasing cell counts to high numbers especially on single cell or chequered mode will certainly impact performance depending on hardware.

On single cell and chequered I have tested up to 90,000 cells before a slowdown on my midrange machine.

On simple mode I have tested up 250,000 cells, with absolutely no slow down at runtime. For this kind of application such a painting or anything using high cell counts, I would turn off gizmos so working in the editor is useable.

Cell Size
Currently the Grid Selector only checks once for the cell size of the first grid it finds. So it sets its own size based on that. If you decide to have different cell sizes for different grids, then it would not match up. You could however do this check for cell size when entering each different grid. In the Grid Selector class this should be fairly easy, I have left it to check once purely for performance as most games the size of the grid/grids will be the same.

Grid Amount
There is no limit to how many different grids you can place, and where. The grid selector will work on all of them. Some below on the ground, some on mountains, and some in between, all good.

Axis
The grid is limited to running on the XZ axis with Y being up. You could if your comfortable with coding dive into the classes to change the XYZ values around with whatever you choose.

Simple Mode
Simple mode is designed for exactly that, simple grids. High cell counts are fine, however this does not have the auto cell block feature.

Design
The design aspect of this system is totally in your hands. I have supplied a few prefabs and Materials/Textures to get started but the look of it is totally up to you.

Renderer
This package can be used with any renderer. It is created in the legacy renderer so all materials are standard. However a simple material conversion to URP or HDRP is all it takes. See the Unity Documentation for how to do that.

Expanding
This system could certainly be expanded on if desired, creating perhaps a check for cells next to one another so you could place objects larger than one cell large.

Creating the option to rotate objects could easily be implemented.

Allowing for currently placed objects to be moved.

And many more features that current games of many genres already have.

If you have any questions or queries please do not hesitate to contact me at support@golemitegames.com