Smart Grids Toolkit

Convenient solution for grids in the Inspector.

DOCUMENTATION

Version: 1.3
Author: Nearmint Studios
Last Updated: 1 June 2025

TABLE OF CONTENTS

INTRODUCTION

The **Smart Grids Toolkit** asset is a versatile tool for managing 2D grids of different data types (string, bool, int, float) in Unity. It allows you to declare grids easily, interact with them in your scripts, and easily modify grid properties within the Unity inspector. The Smart Grids will be referred to as *'Grid 2D'* in this document.

Key Features

- **Support for multiple data types:** Easily declare grids of type string, bool, int, float, or Enum.
- **Inspector-friendly grid editor:** Modify grid cell values directly from the Unity inspector.
- **Resizable grids:** Resize grids programmatically or directly in the inspector without losing data.
- **Flexible cell size and padding customization:** Customize the grid's appearance based on your needs.

HOW TO USE

A. Declaration

First things first, add the SmartGridsToolkit namespace in your script.

```
using SmartGridsToolkit;
```

You can declare a Grid2D of any supported type in your script:

a. Declaring without Initialization:

```
// Declaration without initialization (default size is 3x3)
public Grid2DString stringGrid2D;
public Grid2DBool boolGrid2D;
public Grid2DInt intGrid2D;
public Grid2DFloat floatGrid2D;
public Grid2DEnum<ExampleEnum> enumGrid2D; //ExampleEnum replaces your Enum.
```

b. Declaring with Initialization:

```
// Declaration with initialization
public Grid2DString stringGrid2D = new Grid2DString(5, 5);
public Grid2DBool boolGrid2D = new Grid2DBool(5, 5);
public Grid2DInt intGrid2D = new Grid2DInt(5, 5);
public Grid2DFloat floatGrid2D = new Grid2DFloat(5, 5);
public Grid2DEnum<ExampleEnum> enumGrid2D = new Grid2DEnum<TestEnum>(6,6);
//ExampleEnum replaces your Enum.
```

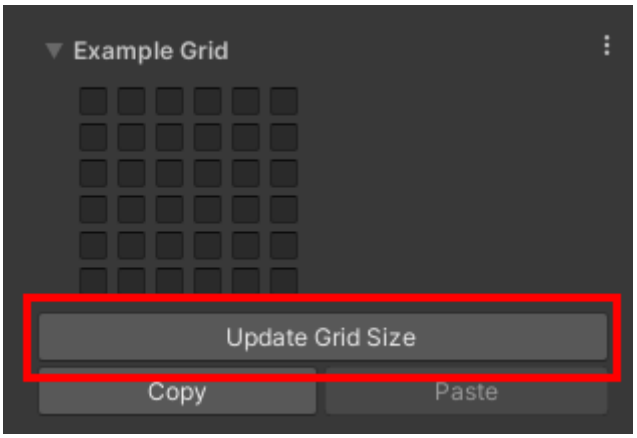
B. Resizing the Grid2D

You can resize any grid dynamically through code while preserving the current values:

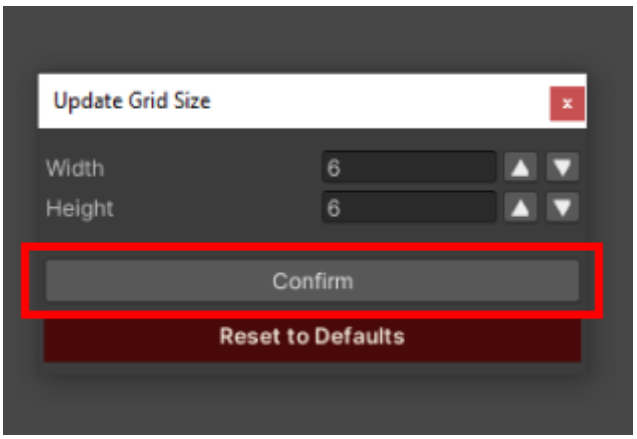
a. Resizing from the script:

```
// Example: Resizing the string grid to (width = 5, height = 7)
stringGrid2D.ResizeGrid(5, 7);
```

b. Resizing from the Inspector:



Click the *Update Grid Size* button.

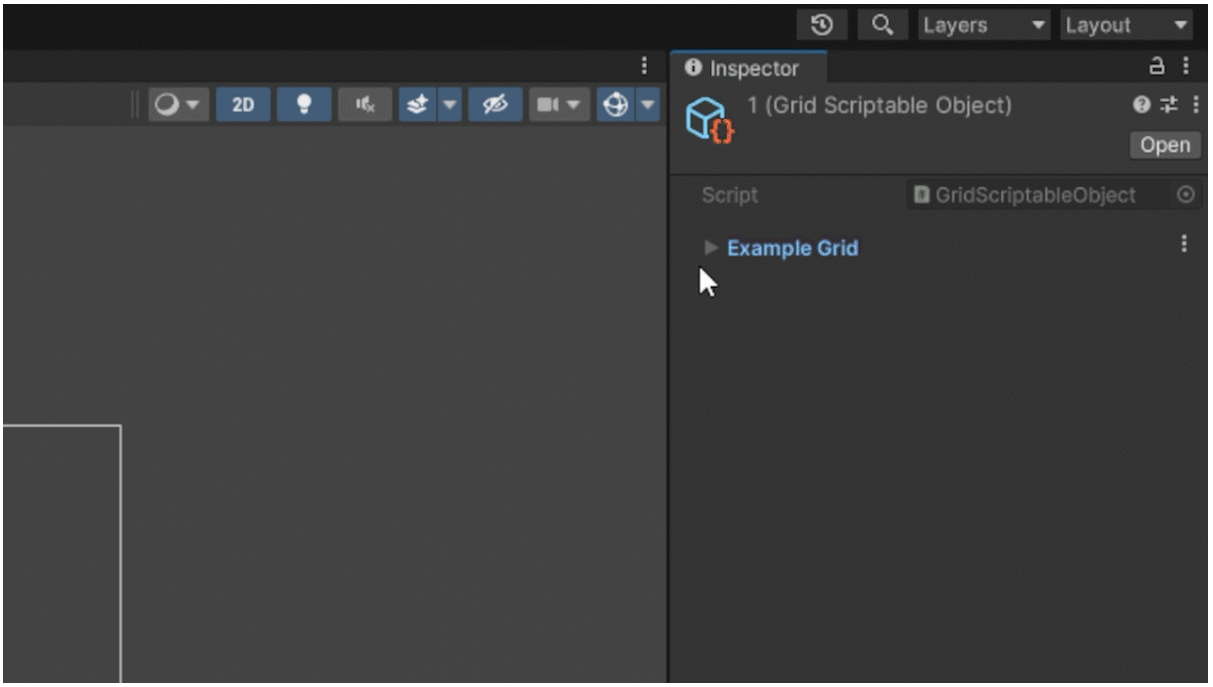


Enter your *desired values* and click *Confirm*.

C. Modifying Cell Values

a. Once you've declared your grid, it will be visible in the inspector. You can click on the foldout to see your grid.

b. Each cell can be directly modified by clicking on it and entering a value appropriate for the grid's type.



D. Accessing Cell Values

You can access specific cells using the '**GetCellValue(int x, int y)**' method:

```
// Example 1: Accessing a cell value at x = 2, y = 3 in a string type Grid2D
string value = stringGrid2D.GetCellValue(2, 3);
```

```
// Example 2: Accessing a cell value at x = 1, y = 0 in a bool type Grid2D
bool isActive = boolGrid2D.GetCellValue(1, 0);
```

```
// Example 3: Spawning a GameObject depending on a Grid2DBool defined as boolGrid.

gridWidth = boolGrid.WidthCount;
gridHeight = boolGrid.HeightCount;

for (int i = 0; i < gridWidth; i++)
{
   for(int j = 0; j < gridHeight; j++)
   {
      bool cellValue = boolGrid.GetCellValue(i, j);

      GameObject squareGO = Instantiate(squarePrefab);
      squareGO.transform.position = new Vector3(i, j, 0);
   }
}
```

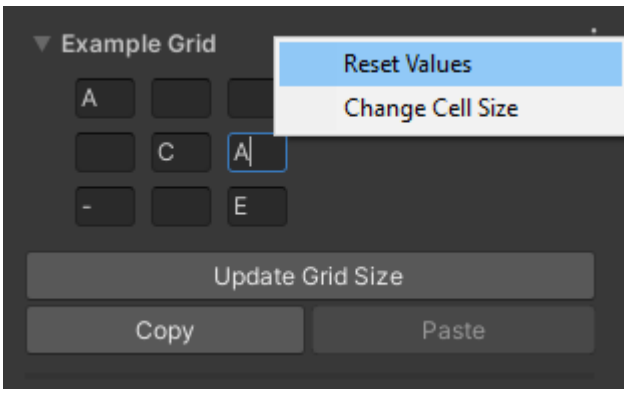
E. Setting Cell Values

You can set values for specific cells using the '**SetCellValue(int x, int y, var value)**' method:

```
// Example: Setting Cell Values for a StringGrid.
    void StringGridSetting()
    {
      for (int i = 0; i < stringGrid.WidthCount; i++)
      {
        for (int j = 0; j < stringGrid.HeightCount; j++)
        {
          stringGrid.SetCellValue(i, j, "A");
        }
      }
    }
```

F. Resetting all values in a grid

The grid can be reset using the Reset option.



Alternatively, the grid can be initialized again in the script to reset it.

```
    public Grid2DString stringGrid2D = new Grid2DString(5, 5);
```

G. Using the Copy Constructor

You can create a duplicate of an existing Grid2D using the copy constructor. This is useful when you want to make changes without affecting the original grid.

```
// Example: Creating a copy of a bool grid.
    Grid2DBool originalGrid = exampleBoolGrid;
    Grid2DBool copiedGrid = new Grid2DBool(originalGrid);

// Changes to copiedGrid will not affect originalGrid.
    copiedGrid.SetCellValue(0, 0, false)
```

EXAMPLE USAGE WORKFLOW

a. Declare and initialize your grid in your script.
b. Access and modify grid cells in the Unity inspector.
c. Use the GetCellValue(x, y) method in your script to access cell values.
d. Resize the grid from the inspector if your level or logic requires it.
e. Customize the grid's appearance by adjusting cell size and padding.

TROUBLESHOOTING

a. If grid data **isn't** preserved across game objects, make sure your grid data is copied correctly using the provided copy-paste functionality.
b. Copied Grid2D data can only be pasted to a grid of the same type.

SUPPORT & FEEDBACK

If you encounter any issues or have suggestions for improvements, feel free to reach out:

Email: admin@nearmintstudios.com

Please do leave us a review on the Unity Asset store if we can help improve your workflow. 😊

FEATURED

The **Smart Grids Toolkit** asset was used in several of our games including: ‘Parasocial’ & ‘Cozy Words: Word Trivia’.

Check out **‘Parasocial’** to see ‘Smart Grids Toolkit’ in action and get inspiration for using it in your projects!

Link to the game: https://veekshith-k.itch.io/parasocial

‘Cozy Words: Word Trivia’ is available on the Google Play Store:

Link to the game: https://play.google.com/store/apps/details?id=com.NearMint.CozyWords&hl=en_IN&gl=US

CHANGELOGS

- Version 1.0: Initial release of the Grid2D asset.
- Version 1.1:
  - Bug Fixes
  - Optimizations
  - A scroll bar was added for bigger grid widths.
  - Added a new Enum Grid type.
- Version 1.2:
  - Bug Fixes
  - Added options to change the width and height padding
- Version 1.3:
  - Added copy constructors for all Grid2D types.

LICENSE

You are free to use this asset for both personal and commercial projects. You are allowed to:

- Modify the asset to suit your needs.
- Use it in any type of Unity project, including commercial products.
- Distribute and share the asset as part of a larger project.

You are not allowed to:

- Resell or redistribute the asset as-is or as part of a collection where the asset is the primary value.

No warranty is provided with this asset. Use it at your own risk. The creator is not liable for any damages or issues that arise from using this asset.

Attribution is appreciated but not required.