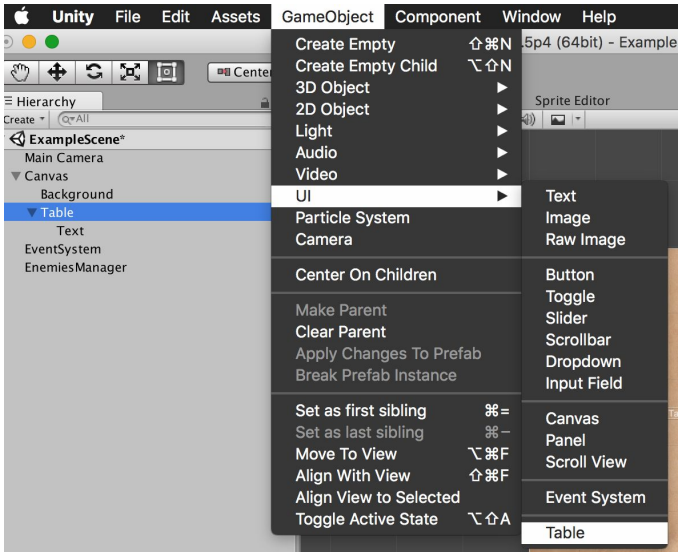
How to use Unity UI Table

Getting Started: Create the Table

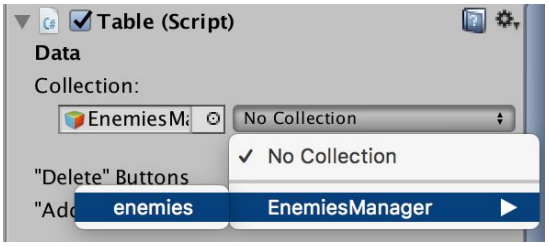
1. In the Game Object menu, select UI / Table.

   Similar to other UI elements, the table will be created in your existing canvas if there is one, or in an automatically created canvas otherwise.



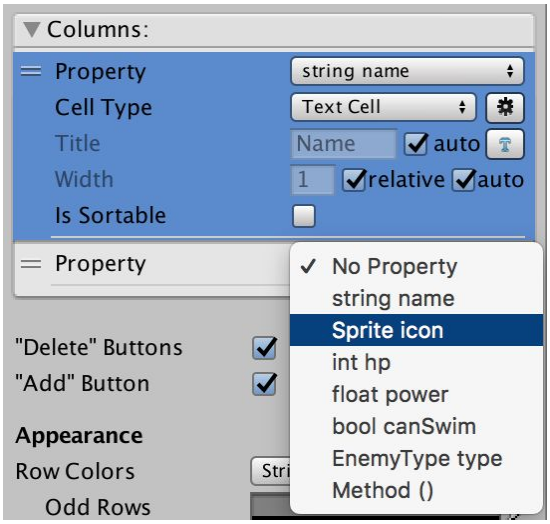
2. In the *Table* component, drag the GameObject containing your collection to the *Collection* field.

3. Select the collection in the popup menu.

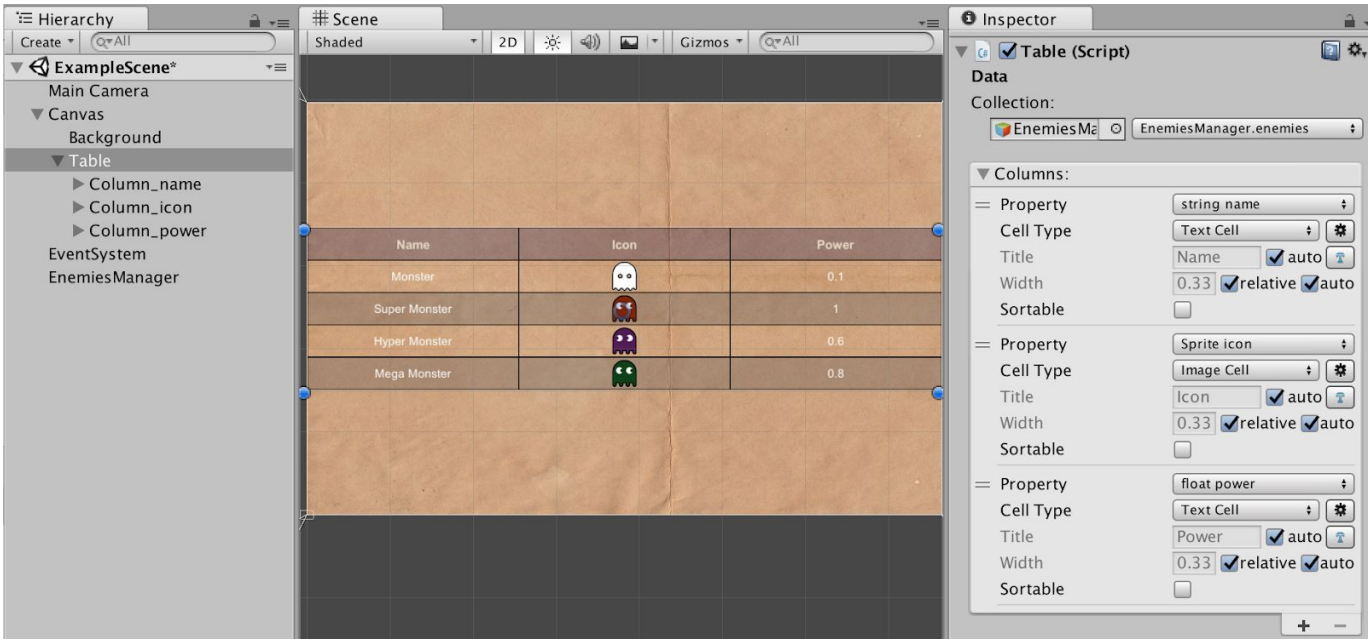


Note: The collection needs to implement the IEnumerable<> interface.

4. The *columns* list appears. Add columns as needed and select the properties, fields or methods of the elements that these columns will represent.



Here you go, the columns are added and filled out automatically with the elements of the collection.



Columns customization

- *Cell Type*: The type of cell to use in this column. The popup menu will automatically find all the cell types that are compatible with the property selected for this column (see Cell Types section)
- *Style*: the gear icon on the right of the cell type opens the cell type style window. There you can set a number of settings depending on the cell type selected. You can create a new style or use an existing style, so you don't have to duplicate the info if multiple columns/tables use the same style.
- *Title*: The title in the column's header. By default, it is set to *auto*, which will use the NicifyVariableName of the property. If auto is disabled, you can choose your own title. You can also set textures as titles by clicking the T button on the right.
- *Width*: The width of the column. It can be set as absolute or relative. If relative, it will be a portion of the total available width in the table (width of the containing rect transform minus the width of the delete column if there is one). It can be set to auto. Columns in auto will equally share the width remaining after all non-auto columns. Usually you will want to leave one column in auto, so it balances the other columns.
- *Sortable*: If checked, at runtime, the user will be able to click the header of this column to sort the table against this column. Click once for ascending sorting, twice for descending, third time to cancel the sorting. (Similar to Windows' File Explorer or Mac's Finder in Details View)

Table options

- Delete Buttons: If checked, buttons will be added at the end of each row. At runtime, these buttons will remove the corresponding element from the collection. It will call the *Remove* method so this method needs to exist for these buttons to work.
- Add Buttons: If checked, a button will be added at the end of the table. At runtime, this button will add an element to the collection. It will call the *Add* method of the collection, and the new element will be created using the parameterless constructor, so both these methods need to exist for this button to work.
- Row Colors: The background color of the rows of the table. If set to *plain*, all rows get the same color. If *Striped*, there are 2 colors, one for odd rows, one for even rows.
- Selectable Lines: If enabled, the user will be able to select rows at runtime. Use Table.SelectedRow to get the currently selected row in your code. You can also select the background color of the selected row.
- Outline: If enabled, a simple stroke outline separates each cell. You can set the color of this outline.
- Headers: Here you can set the appearance of the headers cells.
- Row Height: The row height of the table.
- Scrollable: If enabled, a scroll view will be created and the current table will be set as the content. You can customize the scroll view like any other scroll view.

- Update at Runtime: Depending on your needs, you can decide what the table will refresh at runtime.
    - Cell Style: The columns' cell style will be updated at runtime, so you can modify the style at runtime through code.
    - Cell Content: The content of the cells will be updated continually at runtime, checking the values in the collection. Check this if your data can be modified outside of the table and the changes should reflect in the table. If you want to Update the Content manually, call Table.UpdateContent() in your code when needed, this will be better for performance.
- Limit Rows in Edit Mode: Limits the number of rows to display in edit mode for previewing the table. This is important to avoid slowing down the editor when a large table is displayed.
    - Limit: The maximum number of rows to display in edit mode for previewing the table.
    - Preinstantiate all: If true, all rows will be instantiated in edit mode but inactive. This will make the scene bigger but will be faster to start at runtime. If false, only rows below limit will be pre-instantiated.

Cell Types

There are Cell Types for each basic Unity UI elements available in the Game Object / UI menu:

- Text,
- Image,
- Button,
- Toggle,
- Slider,
- Dropdown and
- Input Field.

These each have settings available in their style object, allowing you to configure them at your needs for each column.

You can also create new cell types. The simplest way to create a new cell type is to duplicate an existing one. Cell Types are prefabs in Assets/RuntimeGUITable/Prefabs. After you duplicated the existing cell type, rename the newly created prefab and move it to your own folders (that will avoid any conflict when updating the plugin). Make the desired changes to your new Cell Type prefab. It will now appear in the Cell Type choice popup of the compatible columns (no matter the containing folder).

If the required cell type is not similar to any of the existing ones, you may have to code your own class deriving CellType, or StyleableCellType if you want some style settings, and add it to the root of your cell type prefab.