TableLayout

USER GUIDE

V1.00 (June 2016)

To view the latest online version of this document, visit

http://www.digital-legacy.co.za/TableLayout/Documentation

TABLE OF CONTENTS

INTRODUCTION

TableLayout is an advanced Layout Group system inspired by HTML tables.

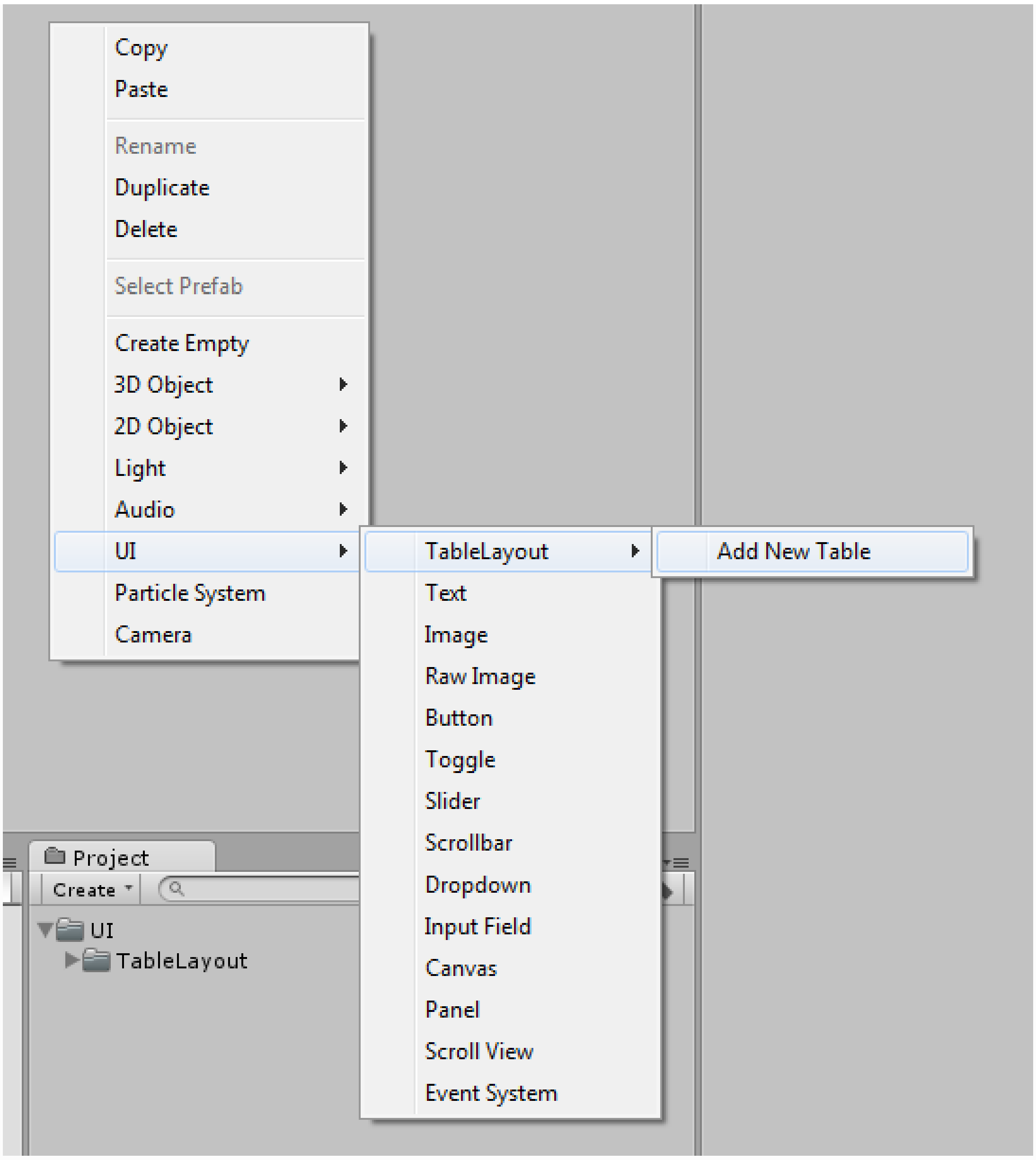
Features:

- Easily line-up elements using rows and columns for professional looking forms and user interfaces
- Show lists of data such as leaderboards/etc. in a clear and easy-to-read fashion
- No programming required (but you can create/modify tables with code if you wish)
- Easily Add and Remove Rows/Cells, even at runtime

WORKING WITH TABLELAYOUT

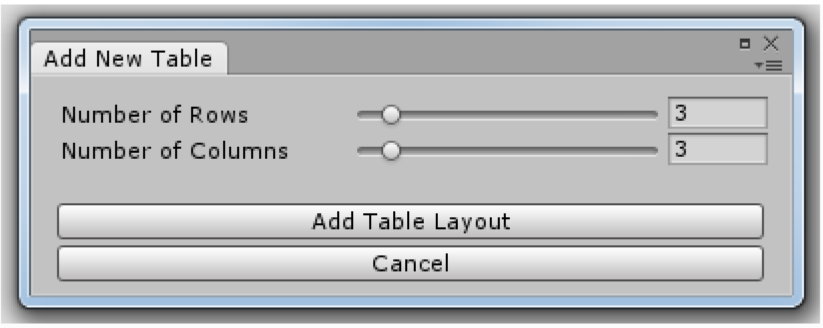
ADDING A NEW TABLELAYOUT

To add a new TableLayout object, right-click the desired location in the hierarchy window, and select
UI -> TableLayout -> Add New Table.



Adding a new TableLayout

You will then be presented with the following wizard:



Add New Table

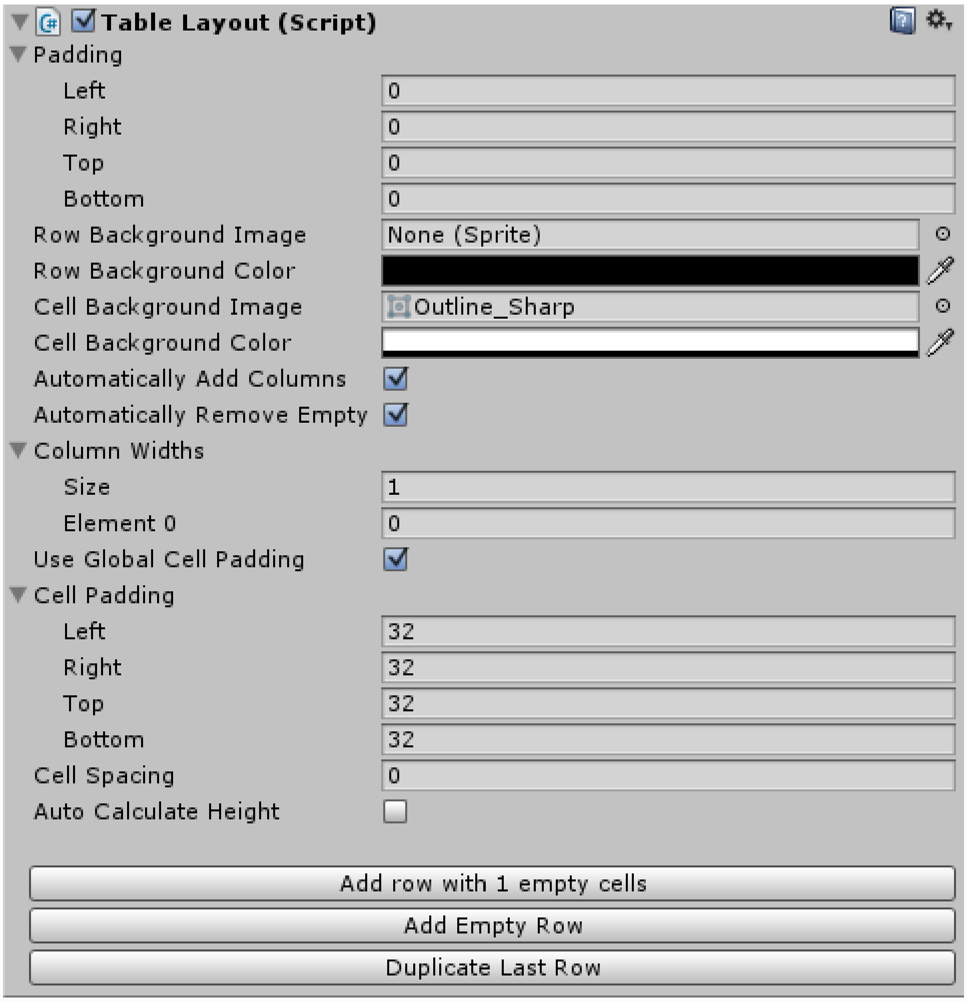
Number of Rows: 3

Number of Columns: 3

Add New Table Wizard

This will allow you to select how many rows and columns the new table should have by default. You can always add/remove rows and columns later if you wish.

THE TABLELAYOUT COMPONENT



☑ Table Layout (Script)	
Padding	
Left	0
Right	0
Top	0
Bottom	0
Row Background Image	None (Sprite)
Row Background Color	⬛
Cell Background Image	Outline_Sharp
Cell Background Color	⬜
Automatically Add Columns	☑
Automatically Remove Empty	☑
Column Widths	
Size	1
Element 0	0
Use Global Cell Padding	☑
Cell Padding	
Left	32
Right	32
Top	32
Bottom	32
Cell Spacing	0
Auto Calculate Height	☐

The TableLayout Component

- **Padding**

  Specifies the padding around the entire Table.

- **Row Background Image**

  Specifies the image to use as the background for each row.

- **Row Background Color**

  Specifies the color to use for each rows background image.

- **Cell Background Image**

  Specifies the image to use as the background for each cell.

- **Cell Background Color**

  Specifies the color to use for each cells background image.

- **Automatically Add Columns**

  If this is set, then the TableLayout will automatically add entries to **Column Widths** if any row has more cells than there are entries in **Column Widths**.

  Essentially, this allows you to have TableLayout automatically handle column widths in the same manner as an HTML table.

- **Automatically Remove Empty Columns**

  If this is set, the TableLayout will automatically remove entries from **Column Widths** if there are columns specified in**Column Widths** which have no cells in them.

- **Column Widths**

  Specifies the width of columns in the table. To have a column automatically sized by TableLayout, specify its width as 0.

- **Use Global Cell Padding**

  If this is set to true, then the **Cell Padding** value specified for this TableLayout will be applied to all of its cells.

- **Cell Padding**

  The padding value to use for each cell in this TableLayout if **Use Global Cell Padding** is set.

- **Cell Spacing**

  Specifies how much space to leave between cells.

- **Auto Calculate height**

  If this is set, the TableLayout will automatically set its own height. This is only useable if each row within the TableLayout has a preferred height set.

- **Buttons**

  - **Add Row with X New Cells**

    This will add a new Row with a full set of empty cells (using the number of entries in Column Widths to determine how many cells to add).

  - **Add Empty Row**

    This will add a new Row with no cells.

  - **Duplicate Last Row**

This will add a new row which is a copy of the last row in the table.

THE TABLEROW COMPONENT

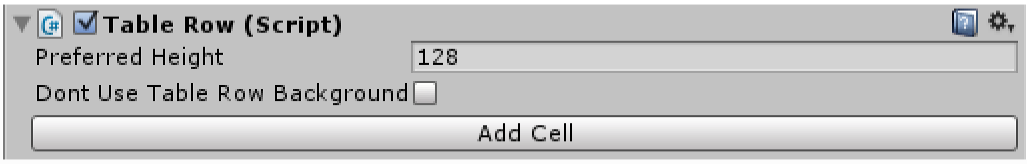


Table Row (Script)

Preferred Height 128

Dont Use Table Row Background □

The TableRow Component

- **Preferred Height**

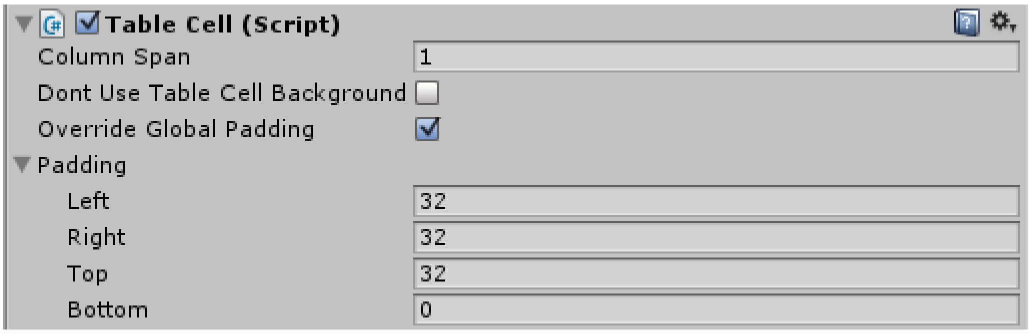
  Sets the height of this row. If the value is 0, then the height of this row will automatically be calculated by the TableLayout.

- **Dont use Row Background**

  If this is set, then you can override the default row background image and color for this row (by modifying the Image component). If it is not set, then TableLayout will override any values set with its own.

THE TABLECELL COMPONENT

Cells store the actual contents of your layout.



☑ Table Cell (Script)

Column Span	1
Dont Use Table Cell Background	□
Override Global Padding	☑
Padding	
Left	32
Right	32
Top	32
Bottom	0

The TableCell Component

- **Column Span**

  How many columns should this Cell span?

- **Dont use Table Cell Background**

  If this is set, then you can override the default cell background image and color for this cell (by modifying the Image component). If it is not set, then TableLayout will override any values set with its own.

- **Override Global Padding**

  If the TableLayout is set to use Global Cell Padding, setting this value will allow you to override that Global Cell Padding for this cell.

- **Padding**

  Sets the padding for this cell.

Working with Code

TableLayout

Adding Rows

You can add more rows using the `AddRow` method. If no argument is specified, then this method will add a new row with a full set of empty cells (as per Column Widths). Alternatively, you can specify a number of cells you wish to add.

Example

```
TableRow newRow = tableLayout.AddRow();
newRow.preferredHeight = 128;
```

Accessing Rows

The rows of a TableLayout object can be accessed through its `Rows` property, e.g.

Example

```
TableRow firstRow = tableLayout.Rows[0];
```

Properties

All TableLayout properties can be accessed and modified through code as per the inspector.

Forcing Updates

If you make adjustments to a TableLayout at any level, it will under most circumstances automatically pick up the changes and update. If however, it does not, you can force an update as follows:

Example

```
tableLayout.CalculateLayoutInputHorizontal();
```

TableRow

Adding Cells

You can add more cells using the `AddCell` method. You can optionally specify content for the cell by providing a RectTransform as an argument.

Example

```
TableCell newCell = tableRow.AddCell();
newCell.columnSpan = 2;
```

Accessing Cells

The rows of a TableLayout object can be accessed through its `Cells` property, e.g.

Example

```
TableCell firstCell = tableRow.Cells[0];
```

Properties

All TableRow properties can be accessed and modified through code as per the inspector.

TABLECELL

All TableCell properties can be accessed and modified through code as per the inspector.