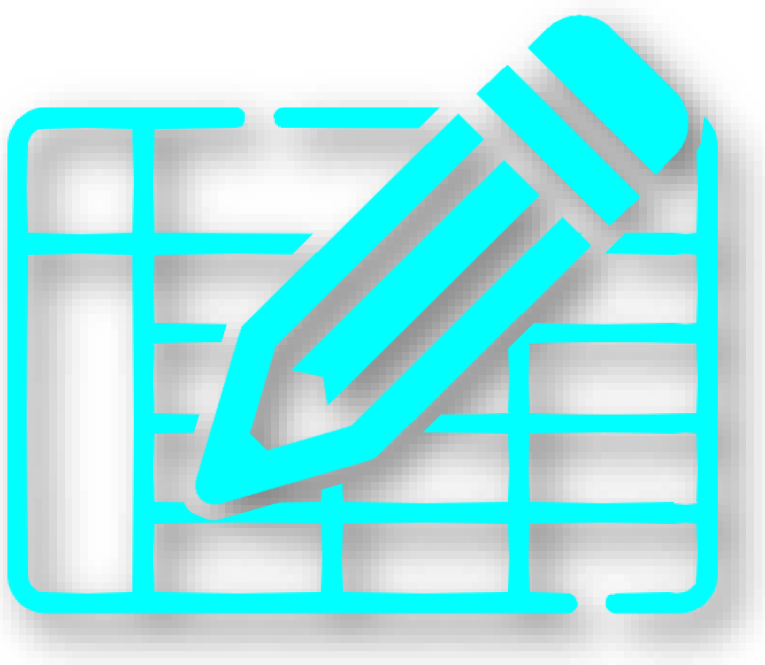

TSV Tools

Documentation

_____

Creator: Lucas Gomes Cecchini

Online Name: AGAMENOM

Overview

TSV Tools is a tool for creating and modifying **.tsv** files for **Unity**.

With it, you can create a text-based data table in the **editor**, and it also allows you to read and write or edit the file in both **Editor** and **Build** mode.

Script Explanations

Tools

Tab Table Utility

Overview

This script is a utility for working with **tab-separated value (TSV)** data, commonly used for simple tables in text files. It's designed for Unity projects and helps **load, modify, save, and access** tabular data easily.
The table is represented as an **array of rows**, where each row is an object containing an **array of cell values** (text strings). These rows are instances of the VerticalTable class.

Structure

Class: VerticalTable
This class represents a single row in the table.

- **horizontalTable**: An array of text strings. Each element in the array is a single cell in the row.

Class: TabTableUtility
This is the main class containing static methods to manage the table data. Here's a breakdown of all variables and methods:

Method: ConvertText(text)

Purpose:
Handles special characters in cells by converting tabs and newlines into placeholder tokens (%tab%, %newline%) and vice versa.
Use case:
When reading from or writing to text files, tab characters or newlines can break the format. This method ensures they are safely encoded and decoded.

Method: LoadTableFile(filePath, tableRef)
Purpose:
Loads a TSV file from disk and populates a table structure with its data.
Parameters:

- filePath: Path to the file to read.
- tableRef: Reference to the array that will be filled with the data.

Steps:

1. Checks if the file exists.
2. Reads the file line by line.
3. Splits each line by tab characters.
4. Converts special characters back to their readable form.
5. Fills the tableRef with rows built from the file.

Use case:
Used when loading external data, such as configuration files, dialogue tables, or game statistics.

Method: SaveTableFile(filePath, table)

Purpose:
Saves the current table into a TSV-formatted file.
Parameters:

- filePath: Destination file path.
- table: The current table data to write.

Steps:

1. Ensures all rows have the same number of columns.
2. Converts all special characters to their safe token format.
3. Writes each row as a tab-separated line to the file.

Use case:
Used to export changes back to disk, for saving edited data in games or tools.

Method: GetText(table, verticalIndex, horizontalIndex)

Purpose:
Gets the text content of a specific cell in the table.
Parameters:

- verticalIndex: Row index.
- horizontalIndex: Column index.

Returns:
The text content in that cell, or null if indexes are invalid.
Use case:
Used when displaying or modifying a specific value in the table.

Method: SetText(tableRef, verticalIndex, horizontalIndex, text)

Purpose:
Sets or updates the content of a specific cell.
Parameters:

- verticalIndex: Row index.
- horizontalIndex: Column index.
- text: New content to assign to the cell.

Use case:
Used when updating data in a cell, such as editing a form field or setting gameplay variables.

Enum: LineDirection

Defines two options:

- **Vertical**: Refers to rows.
- **Horizontal**: Refers to columns.

Used in the next two methods to control which direction to operate on.

Method: AddLine(tableRef, index, lineDirection)

Purpose:
Adds a new row or column to the table at the specified position.
Parameters:

- index: Position to insert the new line.
- lineDirection: Direction of insertion (row or column).

How it works:

- If vertical: inserts a new row with the correct number of empty cells.
- If horizontal: inserts an empty cell into each existing row at the given column index.

Use case:
Used to grow the table dynamically, such as when users click “Add Row” in a UI.

Method: RemoveLine(tableRef, index, lineDirection)

Purpose:
Removes a row or column from the table at the specified position.
Parameters:

- index: Position to remove.
- lineDirection: Direction of removal (row or column).

How it works:

- If vertical: removes the specified row.
- If horizontal: removes the specified cell from each row.

Use case:
Used to delete unwanted rows or columns from the dataset.

Usage Scenario

This utility is particularly useful in tools and editors within Unity that require working with structured data.
Example use cases:

- Editing level data in a grid.
- Managing dialogue or quest tables.
- Importing/exporting data to CSV-like formats.
- Storing configuration data in a human-readable form.

Summary

Feature	Description
Text Conversion	Safely handles special characters for file storage.
Load / Save File	Reads and writes table data from/to a text file.
Cell Access	Retrieves and modifies individual cell values.
Add / Remove Row/Column	Dynamically manages table structure.
Data Format	Uses an object array of rows, each containing a string array of cell data.

TSV Editor Window

This Unity **Editor script** provides a graphical interface to create, open, edit, and save **TSV (Tab-Separated Values)** tables directly within the Unity Editor. It's intended for level designers, data editors, or any other Unity users who want an editable table interface inside Unity.

Variables Explanation

Public Variables

- **tableData**: Holds the actual data of the table as an array of "rows", where each row contains a horizontal array of string values.
- **filePath**: Path of the file currently opened or saved.
- **targetIndex**: Row or column index targeted for deletion.
- **removalDirection**: Defines whether to delete a row (Vertical) or column (Horizontal).
- **rowSizes / columnSizes**: Arrays that hold custom dimensions (height/width) for each row and column.

State Variables

- **isResizingRow / isResizingColumn**: Flags that indicate if the user is resizing a row/column.
- **initialResizePosition**: The mouse position when resizing started.
- **originalDimension**: The original size of the selected row/column before resizing.
- **resizeTargetIndex**: Index of the row or column being resized.
- **mainScrollPosition / tableScrollPosition**: Scroll offsets for the editor window and inner table.
- **isDirty**: True when the table has been modified and not saved yet.

Constants

These help maintain minimum sizes and double-click timing:

- **MinColumnWidth**: Prevents columns from becoming too narrow.
- **MinRowHeight**: Prevents rows from becoming too short.
- **DoubleClickThreshold**: Used to detect double-clicks for adding rows/columns.

UI Settings

- **Color variables**: Used to define how UI elements (lines, labels, background) appear visually.
- **cellTextStyle, headerLabelStyle**: Define how cell contents and headers are drawn.
- **editorBackgroundTexture**: Used for styling purposes.
- **lastInteractionTime**: Stores time of the last interaction to detect double-clicks.

Methods Overview

Window Initialization

- **OpenTableEditorWindow()**: Adds a new menu item in Unity (“Window/TSV Editor Window”) and sets up the window with a custom icon.

Unity Callbacks

- **OnEnable() / OnDestroy()**: Subscribe/unsubscribe to Unity’s undo/redo system. On destroy, the current table is saved temporarily.
- **OnUndoRedo()**: Refreshes the GUI when undo/redo is triggered.
- **OnGUI()**: The main method that draws the entire editor interface and handles all user interactions.

GUI Layout and Interaction (OnGUI)

The editor provides:

- **File Operations**: Open or save .tsv files.
- **Table Creation**: If no file is loaded, you can create a default 2x2 table.
- **Row/Column Removal**: Select an index and direction (horizontal/vertical) to remove a row or column.

- **Editable Grid**: View and modify each cell's contents.
- **Dynamic Resizing**: Click and drag resize handles between cells.
- **Double-click**: Automatically adds a new row or column next to the clicked one.

Table Rendering
- **RenderTable()**:
  - Draws the table headers and all cells.
  - Handles scroll views and background drawing.
  - Detects resizing or double-clicks on columns and rows.
  - Each cell is rendered with editable text fields.
  - New rows or columns are added when a header line is double-clicked.

Mouse Interaction
- **HandleResizeInteraction()**:
  - Handles dragging logic for resizing.
  - Ends the resize when the mouse is released.

File Management
- **LoadTableFromFile(path)**:
  - Parses a .tsv file into the tableData structure.
  - Initializes default row/column dimensions.
- **SaveTableToFile(path)**:
  - Exports the current table to a .tsv file.
  - Resets isDirty and shows a success message.

Utility Method
- **CreateTexture(width, height, color)**:
  - Creates a flat color texture to be used in UI styles.

Usage Example
To use the editor:
1. Go to **Window** → **TSV Editor Window**.
2. Click **Open File** to load a .tsv file.
3. Edit the table directly in the interface.
4. Resize cells by dragging handles.
5. Double-click a row/column header to add a new one.
6. Use **Save Table** to export the result.

Test Table

General Description
This script is meant to be attached to a GameObject in Unity. When selected in the **Inspector**, it exposes tools (via context menus) that allow the user to interact with TSV (Tab-Separated Values) files. The goal is to **test and manipulate TSV table data** using the TabTableUtility system in Unity's Editor.

Variables and Their Roles
- **Table**: This holds the content of the currently loaded table. Each element is a row, and each row contains a series of text cells.
- **File Path**: A string that stores the path of the .tsv file that has been selected or will be saved.
- **Text**: A multi-line input field for the user to enter content that can be placed into a specific cell in the table.
- **Vertical and Horizontal Indexes**: Two integers used to locate a specific cell within the table. These define the row (vertical) and column (horizontal) of a cell.
- **Direction**: A dropdown that lets the user choose whether they are working with rows (vertical) or columns (horizontal). This is used when adding or removing parts of the table.
- **Column Index for Add/Remove**: This determines where in the table a new row or column should be added or removed.

How the Script Works – Method by Method

1. Get File
- This opens a file selection window for the user to pick a .tsv file.
- Once a file is selected, the table is filled with its content using the table utility class.
- If no file is selected, a warning appears in the console.

2. Check Table Validity
- Before performing any action on the table, this helper function checks if the table has actually been loaded and contains data.
- If the table is empty or not set, an error is displayed and the action is cancelled.

3. Save File
- Saves the current table content to a new file path chosen by the user.
- A window opens for selecting the save location and filename.
- The utility class is used to write the data to the chosen location.

4. Get Text
- Looks at the table based on the row and column indexes entered by the user.
- Retrieves and displays the text at that specific cell.
- If the indexes are invalid or out of range, nothing happens.

5. Set Text
- Takes the value from the text field and places it into the specified cell.
- The location is defined by the vertical and horizontal indexes.
- After the change, a message appears confirming the update.

6. Add Column (or Row)
- Adds a new line (row or column) at the specified index.
- Whether a row or column is added depends on the selected direction.
- The new row/column is inserted into the table using the utility method.

7. Remove Column (or Row)
- Removes a row or column at the specified index.
- Again, this depends on the direction chosen.
- The utility method performs the actual deletion from the table.

How to Use This in Unity
1. Attach this script to any GameObject in a scene.
2. In the **Inspector**, you'll see all the fields and variables.
3. Use the **context menu (three dots or right-click on the component header)** to access the available operations:
    - **Get File**: Load a .tsv file.
    - **Save File**: Save the current table.
    - **Get Text**: Read the content of a specific cell.
    - **Set Text**: Change the content of a specific cell.
    - **Add Column**: Insert a row or column at a position.
    - **Remove Column**: Delete a row or column at a position.
4. Modify values like row/column indexes and text before executing actions.
5. Messages about success or errors will appear in the **Console**.

Summary

This script is a **test tool** for working with TSV data inside the Unity Editor. It uses visual inputs and simple menu actions to:
- Load and view table data.
- Modify individual cells.
- Add or remove rows and columns.
- Save modified data back to a file.

It serves as a **bridge between .tsv files and Unity**, enabling designers and developers to manipulate structured table data without needing to open Excel or a text editor.