# MultipleTags

## (Multiple tags using the existing Unity tag system)

**Made by AiUnity**
http://aiunity.com

# Table of Contents

# Overview

MultipleTags enables you to apply multiple tags to a GameObject, solely using the existing Unity tag system. This is done by stitching tags together to form a tagPath (ie. T1/T2/T3). By delineating tags with backslashes the Unity tag system will present a tagPath in a concise hierarchical menu structure. To augment the Unity tag interface an extremely powerful and elegant tag manager editor GUI is provided. The tag manager makes it easy to add/remove tags, find gameObjects by tag, optimize tags, generate a type safe tag access file, and much more (See screenshots). The tag manager also displays the tags associated with the currently selected gameObject(s). MulitpleTags also provides the ability to use tag groups to facilitate organization and discovery (ie. Color.Red).

At runtime an intuitive set of APIs are available to harness the power of having multiple tags. FindGameObjectsWithTags API exists to find gameObjects with specfied tag(s), while gameObject extension methods allow you to refine your search. The extension methods also allow you to add/remove tags at runtime (Tags must exist in Unity). When searching for gameObjects a boolean enumeration is available or for greater flexibility use a free-form expression (ie. (Tag1 | Tag2) & !Tag3). The search expression also supports regex syntax (ie. Tag.*). To help form and test a search expression use the dropdown menu in the GUI "Find tags" section.

MultipleTags can manually or automatically generate a type safe tag access script (ie. TagAccess.cs). This enables type safety, code completion, and boosts performance of the tag search APIs. Also related is the AiUnity flagship ScriptBuilder product, which uses builders to generate any Unity script. It includes the builder TagBuilder, if you wish to gain direct control of the Tag Access script generation.

# Features

- Completely utilizes Unity tag system.
- The Unity tag manager is augmented by a comprehensive tag GUI control panel.
- Select GameObject(s) in hierarchy to reveal and modify tags in GUI.
- Search/modify Unity tags from GUI.
- Search/modify gameObject tags from GUI.
- Use tag access script to reference Unity tags in a type safe manner.
- APIs available to find gameObjects using tag name, logic expression, or Regex.
- APIs can find gameObjects using group tag syntax (ie. Vehicle.Car).
- APIs can find all gameObjects in a particular group (ie. Vehicle).
- GameObject extension methods enable search refinement and tag modification.
- Automatic or manual creation of a type safe tag access script (TagAccess.cs).

- GUI optimize command allows quick cleanup/optimization of existing tags.
- GUI themed for light and dark skin.
- Works with AiUnity products ScriptBuilder, NLog, and CLog.
- Website, API Reference, Videos, and forums available for support.
- Tested on .NET 3.5 and 4.6.
- Full source code provided.

# Quick starts

## MultipleTags GUI

1. Open up the MultipleTags GUI control panel.
2. Use the "Find Tags" add/delete buttons or the Unity tag manager to add/delete tags.
3. Select a gameObject in the hierarchy and its tag will be reflected in the GUI.
4. Use the GUI tags +/- icons to modify the tags of the selected gameObject(s).
5. Alternatively use the Unity tag system to modify the tag of a gameObject.
6. Be sure to investigate other ways the control panel can help manage tags.
7. Enjoy!

## MultipleTags APIs

1. Load the MultipleTagScene scene and then open the MultipleTagExample example script.
2. Inspect the example comments to learn the intuitive find, check, and modify gameObject(s) APIs.
3. Use TagAccess static members when referencing tags to code in a type safe manner.
4. Set MultipleTags GUI options to auto generate TagAccess script, so that it is always current.
5. Use GameObject extension methods to to refine your gameObject searches.
6. Enjoy!

# GUI

The MultipleTags GUI control panel is a Unity Editor window that can be accessed via Tools-> AiUnity->MultipleTags->ControlPanel. The GUI is designed to make Unity tag management intuitive and powerful. The control panel contains tooltips, selection boxes, and references to make learning simple. Below are the descriptions of the components that make up the GUI.

## Tags

The tags section reveals the tags associated with the currently selected gameObject(s). Notice this will always agree with the Unity Inspector tag entry for the GameObject. When multiple gameObjects are selected the tags common to the gameObjects are shown. From the GUI tags

section you can remove or add tags to the gameObject(s). Multiple tags are delineated by backslashes and form a so called tagPath. The Unity tag system will present a tagPath in a concise hierarchical menu structure. This approach elegantly provides multiple tags without requiring an entirely new cumbersome tag framework. You can see the complete list of tagsPaths in control panel or the Unity Tag manager. In addition to the manual the GUI controls have tooltips and a dedicated MultipleTags Overview video.

# Find tags

Find tags provides a means to discover and modify Unity tags. Choose a tag category to show only tags used or unused by a gameObject. The Tag Expression enables you to filter the full list of tags. The expression can include a tag name, boolean logic, or regex pattern. The plus icon dropdown menu will aid you in composing an expression. The find expression is live and actively returns the results. The Add and Delete button allow you to modify the list of Unity tags.

# Find gameObjects

Find gameObjects allows you to discover gameObjects and modify their tags. The expression allows you to search gameObjects using the same tag syntax as "Find tags". An empty expression signifies all gameObjects having a tag other than "Untagged". To add and remove tags from a particular gameObject use the "+" and "-" icons respectively. Selecting a gameObject in the panel will cause it to be selected in the Unity Hierarchy window, which functions in the other direction as well.

# Options

The options are for configuring the MultipleTags behavior. All options retain their values between editor sessions.

## Internal levels

Specifies which logging levels are enabled for Mulitple Tags internal messages (Debug feature). Adding Levels Info, Debug, or Trace can be immensily helpful if MultipleTags behaves unexpectedly.

## Tag Expand

The "Tag expand" option will facilitate the generation of tag permutations. For example the creation of tagPath T1/T2/T3 would also produce tagPaths TI and T1/T2. Having all tagPath permutations may be useful since a tag must exist for it to be assigned to a gameObject at runtime. On the otherhand fewer tags overall is easier to manage and can have a small performance advantage. Also note with this option enabled the "Tag optimize" command will expand all tagPaths. The default is disabled.

### Group Expand

The "Group expand" option will facilitate the generation of group permutations. For example adding tag T1/Color.Red would automatically create T1/Color.Blue. The knowledge is a result of the optimizer doing a first pass on the Unity tags to learn the tags for each group. Having all tagGroup permutations may be useful since a tag must exist for it to be assigned to a gameObject at runtime. On the otherhand fewer tags overall is easier to manage and can have a small performance advantage. Also note with this option enabled the "Tag optimize" command will expand all tagGroups. The default is enabled.

### TagAccess script

The "TagAccess script" option can manually or automatically generate TagAccess.cs in folder AiUnity/UserData/MultipleTags /Resources. TagAccess provides type safe access to Unity Tags and is utilized by the MultipleTags APIs (See example). With auto selected any change to the Unity tags will cause TagAccess.cs to regenerate. The auto default is enabled.

### Tag optimize

The "Tag optimize" option will analyze and optimize all Unity tags. This operation will irreversibly add and delete Unity tags. Optimization includes consolidating and removing duplicate tagPaths. TagPaths and tagGroups will be conditionally expanded based upon the expand options. With auto selected any tag(s) that are added/removed in the GUI will be optimized.

## API

At runtime an intuitive set of APIs are available to harness the power of having multiple tags. The best way to learn about the APIs is to analyze the well documented and comprehensive included example. The MultipleTags APIs video provides a comprehensive walkthrough of an example. To gain a greater understanding of the API namespace or a particular API use the API reference.

## Tag Access

The TagAccess script provides type safe access to the Unity Tags. It contains a TagPaths property that is used by the internal find APIs to facilitate searches. Within TagAccess the tag groups are represented as nested classes to provide code completion and discovery. For example you could type Color followed by period to discover all the color tags (i.e. Color.Red/Color.Blue). It is also possible to denote any tag in the tag group (i.e. TagAccess.Color.Any()) when searching.

# Find Tags

FindGameObjectsWithTags API exists to find gameObjects with specfied tag(s), while gameObject(s) extension methods allow you to refine your search. The FindGameObjectsWithTags is located in TagService class and contains many overloads to provide power and flexiablity. You can use these API to search for gameObjects by tag string (i.e. "T1"), expression (i.e. "T1 | T2"), or regex pattern (i.e. "T.*"). It is advised to use TagAccess.cs when referencing tags to provide type safety (i.e. FindGameObjectsWithTags(TagAccess.T1)).

# Modify Tags

Unity tags can only be created/deleted from within the editor. This is inherent in the Unity tag mechanism and provides exceptional runtime performance. From the editor you can modify tags with the MultipleTags GUI or using the Unity Tag Manager.

GameObject tags are primarily assigned/removed inside the editor. In addition MultipleTags provides gameObject extension methods that allow you to modify a gameObject tag at runtime. The modified gameObject tag must exist in Unity for the operation to be successful. For example to add tag "T2" to gameObject with tag "T1", it is require that tagPath "T1/T2" already exists.

# API Examples

Below are code snippets from the included AiUnity/MultipleTags/Examples/ examples. Please refer directly to the example(s) to gain a better understanding and analyze the results. The MultipleTags APIs video provides a comprehensive walkthrough of an example.

```
void Start()
{
    // Find gameObjects with tag "T1".
    gameObjects = TagService.FindGameObjectsWithTags(TagAccess.T1);

    // Find gameObjects with tags "T1" or "T2", but not T3.
    gameObjects = TagService.FindGameObjectsWithTags("(T1 | T3) & Color.Blue");

    // Use regex pattern to find gameObjects with tags starting with "T"
    gameObjects = TagService.FindGameObjectsWithTags("T.*");

    // Find gameObjects that use any tag in group "Color".
    gameObjects = TagService.FindGameObjectsWithTags(TagAccess.Color.Any());
```

```
    // Check if gameObject has tag "T1" or "T2"
    hasTags = gameObjectT1.HasTags(TagLogic.Or, TagAccess.T1, TagAccess.T2);

    // Use extension method to refine search
    gameObjects.FindGameObjectsWithTags(TagLogic.And, TagAccess.T3);

    // Add tag to gameObject
    gameObject.AddTags(TagAccess.T2);
}
```