More Tags

Components

Tags : Add Component ➜ MoreTags ➜ Tags

You can use Tags to view the tags attached in GameObject. Click on the tags to remove it. Enter tag in the text box then click the plus sign to add new tag. Click the plus sign without enter text will show all available tag list.
A Tag Manager is automatically created if it is not present in scene.

Tag Manager : Add Component ➜ MoreTags ➜ Tag Manager

Manage all tags in the scene. You can view all tags, add new tag, remove or rename tags. A tag filter allow you to find GameObject with specific tag pattern, add, remove or change tags to all GameObject listed.

Tag Manager ➞ Presets

Tag Presets are stored in EditorPrefs, you can add, remove or reorder tag in presets. The order and color will be used when showing the tags.

To Manager: Copy all tags from Tag Presets to Tag Manager in the scene.
To Preset: Copy all tags from Tag Manager in the scene to Tag Presets.
Auto Class: Generate a C# class file with all tags. It allow you to use autocomplete in Visual Studio or MonoDevelop.

Tag Manager ➜ All Tags

Show all tags present in the scene. You can add or remove tags like in Tags component. Click rename to rename the selected tag to new tag name in the text box.

Abbreviation: Shorten the display name of the tag. (eg. Group.Tag ➜ G.Tag)

Tag Manager ➜ Tag Pattern and Filter

Tag Pattern: Enter the pattern to show all GameObjects match it.
All: Apply the tag pattern to all GameObjects.
Selected: Apply the tag pattern to selected GameObjects only.
With Children: Include all childrens of selected GameObjects.
Remove All: Remove all selected tag in all listed GameObjects.

Tag Pattern

*: All tags in tag manager, it show all GameObjects with tag.
-*: Show all GameObjects without tag.
Tag1 & Tag2: GameObjects with both Tag1 and Tag2.
Tag1 | Tag2: GameObjects with either Tag1 or Tag2.
-Tag: GameObjects without Tag. (Include GameObject without tag)
Tag1 - Tag2: GameObjects with Tag1 but without Tag2.
Group.*: All children under Group.* , include children of children.
Group.?: All children under Group.* , without children of children.
*.Tag: All tags under any group, with name Tag.
.Tag.: Same as Group.*, but under *.Tag.
*.Tag.?: Same as Group.?, but under *.Tag.
Either(Tag List): GameObjects with either tags in tag list.
Both(Tag List): GameObjects with both tags in tag list.
Tag List:
Tag1 + Tag2: Include Tag1 and Tag2 in tag list.
Tag1 - Tag2: Exclude Tag2 in tag list.

Both(Group.*)
With this pattern, it show GameObjects with all tag under Group.
Either(Group.* - Group.Tag1)
With this pattern, it show any tag under Group but without Group.Tag1.

Runtime

You can add, remove or change tag at runtime:
GameObject.AddTag(Tag1, Tag2...);
GameObject.RemoveTag(Tag1, Tag2...);
GameObject.ChangeTag(OldTag, NewTag);

To check GameObject has the tag attached:
bool GameObject.HasTag(Tag);
bool GameObject.AnyTags(Tag1, Tag2...);
bool GameObject.BothTags(Tag1, Tag2...);

Get the tags attached in GameObject:
string[] GameObject.GetTags();
string[] GameObject.FindTags(TagNames);

GameObject.FindTags("Group.*");
Remove all tags under Group.* that attached in GameObject.
See TagHelper for more information about TagNames.

Add, remove or rename tags in Tag Manager:
TagSystem.AddTag(Tag1, Tag2...);
TagSystem.RemoveTag(Tag1, Tag2...);
TagSystem.RenameTag(OldTag, NewTag);

Clear all tags in Tag Manager:
TagSystemReset();
Remove all null GameObjects in Tag Manager:
TagSystem.RemoveNullGameObject();
Remove all tags no GameObjects attached to.
TagSystem.RemoveUnusedTag();

Get all tags, find GameObjects with the tag:
string[] TagSystem.GetAllTags();
GameObject TagSystem.GetGameObject(string tag);
GameObject[] TagSystem.GetGameObjects(string tag);

To find GameObject with tag pattern:
TagPattern TagSystem.pattern;

You need to use TagSystem.pattern to get new TagPattern:
TagSystem.pattern.With(Tag).GameObjects();
It return all GameObjects with Tag.

TagPattern

Search with single tag, both tags or either tags in list:
TagSystem.pattern.With(Tag);
TagSystem.pattern.Both(Tag1, Tag2...);
TagSystem.pattern.Either(Tag1, Tag2...);

Include all GameObjects: (Both with and without tag)
TagSystem.pattern.All();

You can use method chaining to combine pattern:
TagPattern TagPattern.And();
TagPattern TagPattern.Or();
TagPattern TagPattern.Exclude();

TagSystem.pattern.Both(Tag1, Tag2).Or().Both(Tag3, Tag4);
This equals to (Tag1 & Tag2) | (Tag3 & Tag4).

Combine to other TagPattern:
TagPattern.And(TagPatter);
TagPattern.Or(TagPatter);
TagPattern.Exclude(TagPatter);

var Pattern = TagSystem.pattern.Both(Tag3, Tag4);
TagSystem.pattern.Both(Tag1, Tag2).Or(Pattern);
This also equals to (Tag1 & Tag2) | (Tag3 & Tag4).

Get GameObjects and count from pattern:
GameObject TagPattern.GameObject();
GameObject[] TagPattern.GameObjects();
int TagPattern.Count();

Limit the pattern to search from specific GameObjects:
TagSystem.SearchFrom(GameObjectList);

TagSystem.SearchFrom();
Call SearchFrom() without parameter to reset it.

You can convert pattern string to TagPattern:
TagPattern = “(Tag1 & Tag2) | (Tag3 & Tag4)”;

TagHelper

TagHelper contain some helper class used in auto generated class, those class can combine with operator and convert to TagPattern.

TagName:

Represent the name of a tag. Can cast to / from string.

```
TagName tag = “Group.Sub.Tag”;
string str = tag;
```

Some helper field:

```
string TagName.name; // Tag name
string TagName.last; // Last part of Tag
string[] TagName.parts; // Different part of Tag
```

Convert to TagPattern:

```
TagName tag1 = “Tag1”;
TagName tag2 = “Tag2”;
TagName tag3 = “Tag3”;
TagPattern pattern = tag1; // Pattern with tag1
pattern = tag1 & tag2; // Both tag1 and tag2
pattern = tag1 | tag3; // Either tag1 or tag3
pattern = (tag1 - tag3) | tag2; // (tag1 without tag3) or tag2
```

TagNames:

Represent a list of TagName.

```
TagNames tags = tag1 + tag2; // List include tag1 and tag2
tags -= tag3; // Remove tag3 from the list
tags = “Group.*”; // All Group.* tags
tags = “Group.* - *.Tag”; // All Group.* tags without *.Tag
```

Convert to TagPattern:

```
TagNames.both; // All tags to pattern with TagPattern.Both()
TagNames.either; // All tags to pattern with TagPattern.Either()
```

Get all tags from TagNames:

```
TagNames tags = "Group.*";
foreach (var tag in tags)
    DoSomething(tag);
```

TagGroup:
Extended TagName, represent the group of a tag.
TagGroup.all; // All children under TagGroup, include children of children.
TagGroup.children ; // All children under TagGroup, without children of children.

TagGroup group = “Group”;
var pattern = group.all.either; // Any tags under Group.*
pattern = group.children.both; // All tags directly under Group.?

TagChildren:
Extended TagNames, represent all available children of the name.
TagChildren tags = “Tag”; // Include all tag match *.Tag

tags.all; // All children under *.Tag, include children of children.
tags.children ; // All children under *.Tag, without children of children.

Auto Generated Class

Tag:
Include all tags in Tag Manager

Tag.all; // TagNames include all tags

Tag.all contain list of available TagChildren field.
If Tag Manager contain Group.Tag1 and Group.Tag2
Tag.all.Tag1 and Tag.all.Tag2 will present in Tag class.

Tag class contain list of available TagName and TagGroup field.
Tag.Group, Tag.Group.Tag1 and Tag.Group.Tag2 will present in Tag class.

```
public static class Tag
{
    public static AllTags all = new AllTags();
    public static GroupGroup Group = new GroupGroup("Group");
}

public class GroupGroup : TagGroup
{
    public GroupGroup(string name) : base(name) { }
    public TagName Tag1 = new TagName("Group.Tag1");
    public TagName Tag2 = new TagName("Group.Tag2");
}

public class AllTags : TagNames
{
    public AllTags() : base(TagSystem.AllTags()) { }
    public TagChildren Tag1 = new TagChildren("Tag1");
    public TagChildren Tag2 = new TagChildren("Tag2");
}
```