Get Started With ALINE

This is an excerpt from the full documentation. You can view the full documentation here (http://arongranberg.com/aline/documentation/beta). Most links on this page will just take you to the full documentation.

In this tutorial we will cover how to create a simple script and draw some things from the Update loop as well as in a gizmo callback.

Contents

Downloading

The first thing you need to do, if you haven't done so already, is to download the package.
You can download the package in the Asset Store or from the package website (http://arongranberg.com/aline/download).

Drawing from the Update loop

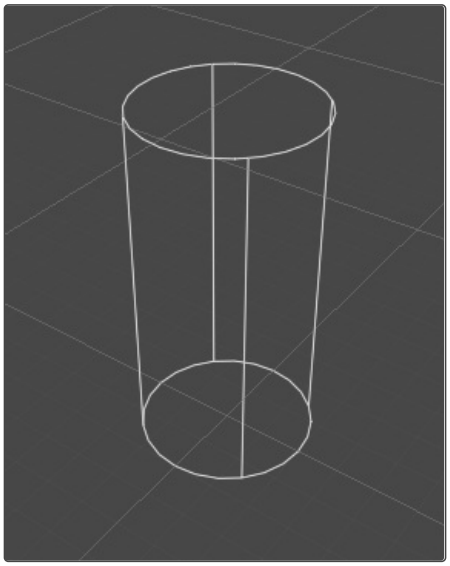
If you want to debug what a script does then drawing things from the code is a great way to do it.

Create a new C# script in Unity and name it GetStartedExample.cs, then open that up in a text editor and write the following code.

```
using UnityEngine;
// Important for the script to be able to find the Draw class
using Drawing;

public class GetStartedExample : MonoBehaviour {
    void Update () {
        // Draw a cylinder at the object's position with a height of 2 and a radius of 0.5
        Draw.WireCylinder(transform.position, Vector3.up, 2f, 0.5f);
    }
}
```

In your Unity scene you can then create a new GameObject and attach the GetStartedExample (http://arongranberg.com/aline/documentation/beta/getstartedexample.html) script to it. If you press play you should see a cylinder rendered at the object's position.



You can draw other things than cylinders of course. Take a look at the Drawing Commands (http://arongranberg.com/aline/documentation/beta/drawingcommands.html) page for a list of them.

If you want to render the cylinder in a different color you can simply add a color parameter at the end. All drawing commands have an optional color parameter at the end.

```
Draw.WireCylinder(transform.position, Vector3.up, 2f, 0.5f, Color.red);
```

Scopes

A powerful abstraction used in this package is the notion of scopes. Scopes can be used to set the color, matrix, line width or duration of multiple drawing commands at once without having to individually specify them or having to do matrix multiplications manually. This is both faster and leads to more readable code.

```
// Draw three red cubes
using (Draw.WithColor(Color.red)) {
    Draw.WireBox(transform.position, Vector3.one);
    Draw.WireBox(transform.position + Vector3.right, Vector3.one);
    Draw.WireBox(transform.position - Vector3.right, Vector3.one);
}
```

Using matrix scopes is very useful if you want to for example draw in local space relative to some object. In the below example a cylinder is drawn in an object's local space. This means it is moved, rotated and scaled with the object automatically.

```
using (Draw.InLocalSpace(transform)) {
    // Draw a box at (0,0,0) relative to the current object
    // This means it will show up at the object's position
    Draw.WireBox(Vector3.zero, Vector3.one);
}

// Equivalent code using the lower level WithMatrix scope
using (Draw.WithMatrix(transform.localToWorldMatrix)) {
    Draw.WireBox(Vector3.zero, Vector3.one);
}
```

Videos cannot be played in pdfs. Take a look at the online documentation for the video which normally goes here.

In the example below the color and duration scopes are shown.

```
// This box will be drawn for 2 seconds
using (Draw.WithDuration(2)) {
    Draw.WireBox(Vector3.zero, Vector3.one);
}

// Scopes can be nested
using (Draw.WithColor(Color.red)) {
    using (Draw.WithDuration(2)) {
        Draw.WireBox(Vector3.zero, Vector3.one);
    }
}
```

> See
> Draw.WithColor (http://arongranberg.com/aline/documentation/beta/withcolor.html#WithColor)
> Draw.WithMatrix (http://arongranberg.com/aline/documentation/beta/withmatrix.html#WithMatrix)
> Draw.WithLineWidth (http://arongranberg.com/aline/documentation/beta/withlinewidth.html#WithLineWidth)
> Draw.WithDuration (http://arongranberg.com/aline/documentation/beta/withduration.html#WithDuration)
> Draw.InLocalSpace (http://arongranberg.com/aline/documentation/beta/inlocalspace.html#InLocalSpace)
> Draw.InScreenSpace (http://arongranberg.com/aline/documentation/beta/inscreenspace.html#InScreenSpace)

Drawing object gizmos

Drawing helper visualizations for objects in the scene view is often helpful when developing a game. With Unity's built-in gizmo system you draw everything in the OnDrawGizmos method and you need to use the Gizmos class instead of the Debug class. In this package you do everything with the Draw (http://arongranberg.com/aline/documentation/beta/draw.html) class.

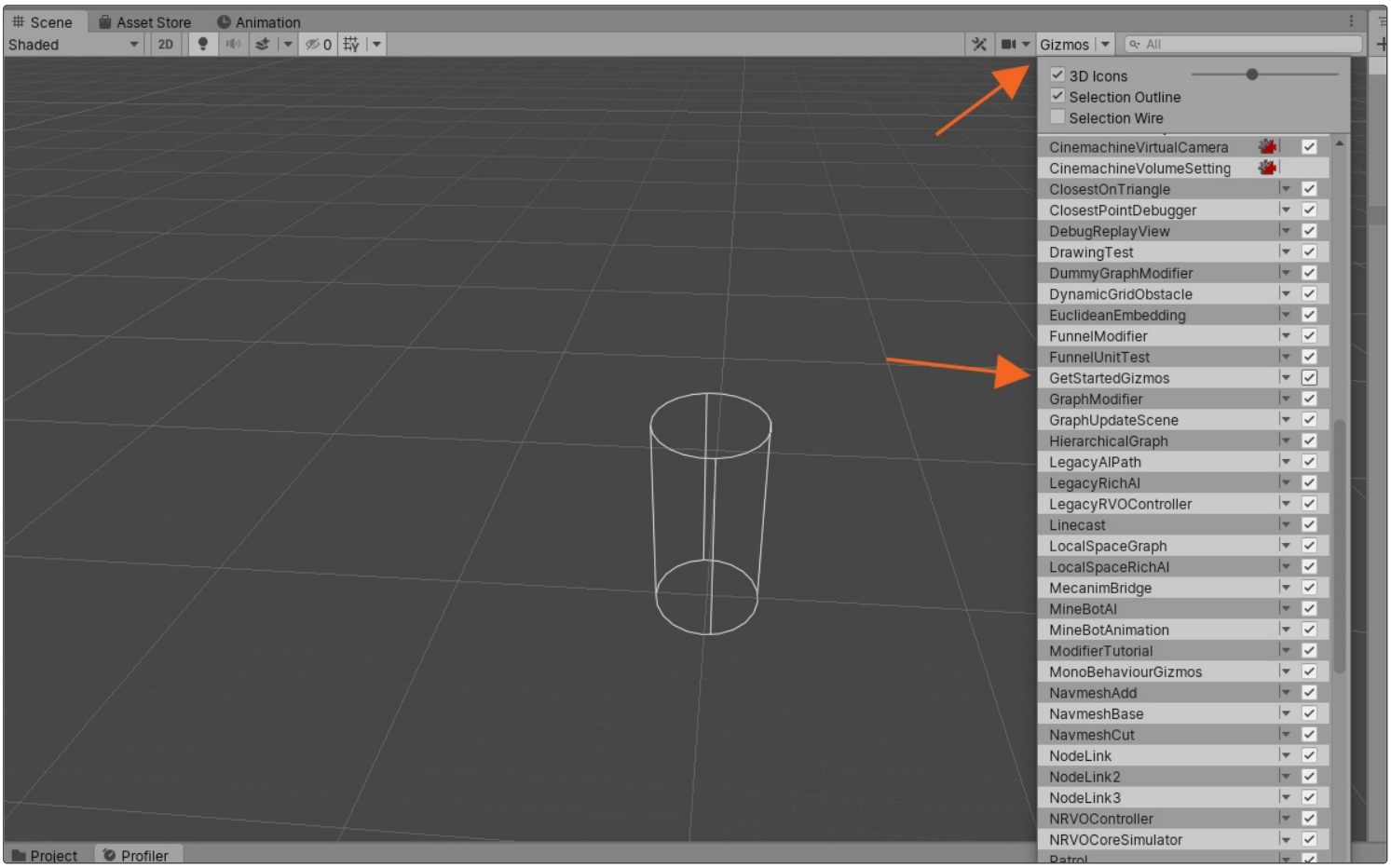
To receive gizmo callbacks you need to make your script inherit from MonoBehaviourGizmos (http://arongranberg.com/aline/documentation/beta/monobehaviourgizmos.html) instead of from MonoBehaviour. You can then override the DrawGizmos method and draw your gizmos there exactly as you would in other parts of the code.

```
using UnityEngine;
using Drawing;

public class GetStartedGizmos : MonoBehaviourGizmos {
    public override void DrawGizmos () {
        using (Draw.InLocalSpace(transform)) {
```

```
        using (Draw.InLocalSpace(transform)) {
            // Draw a cylinder at the object's position with a height of 2 and a radius of 0.5
            Draw.WireCylinder(Vector3.zero, Vector3.up, 2f, 0.5f);
        }
    }
}
```

Gizmos will be drawn all the time, even when the game is not playing or if it is paused. You can toggle gizmo drawing for individual components in the gizmos menu at the top right corner of the scene view and game view.



Selection based drawing

If you have a lot of gizmos in a scene it can sometimes cause quite a bit of clutter. One solution to this to draw simple gizmos most of the time and only draw more elaborate or detailed gizmos when the object is selected.

This can easily be accomplished using the GizmoContext (http://arongranberg.com/aline/documentation/beta/gizmocontext.html) class.

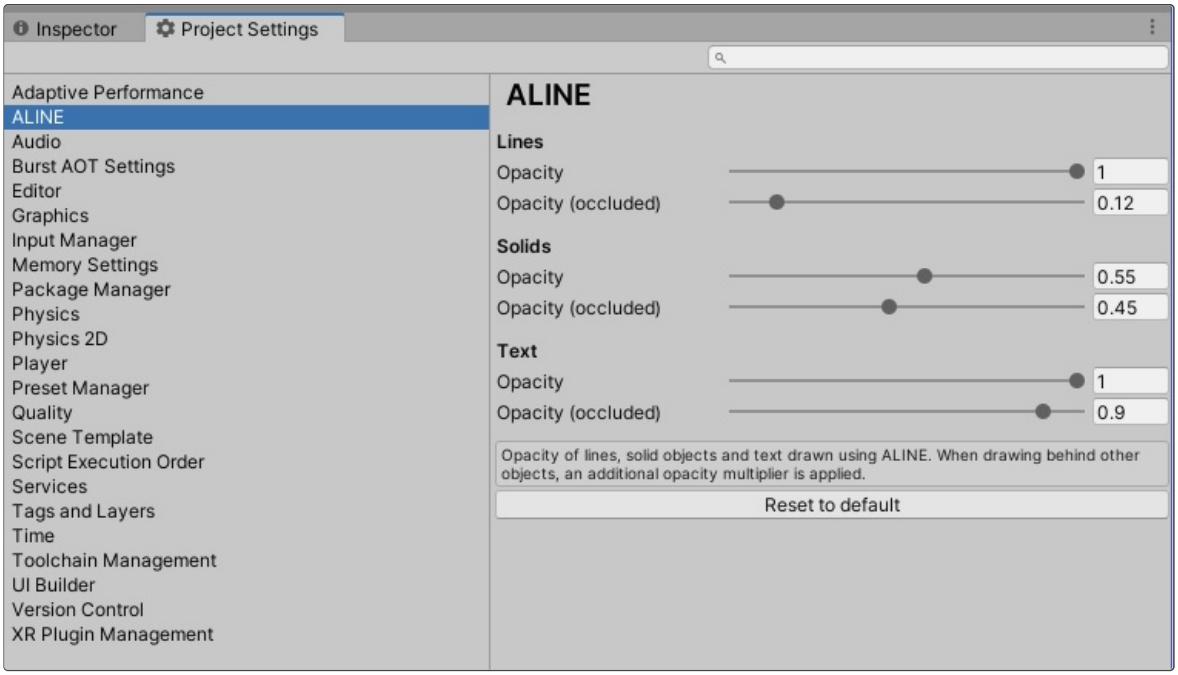
```
public override void DrawGizmos () {
    using (Draw.InLocalSpace(transform)) {
        if (GizmoContext.InSelection(this)) {
            // Draw a yellow cylinder
            Draw.WireCylinder(Vector3.zero, Vector3.up, 2f, 0.5f, Color.yellow);
        } else {
            // Draw a yellow circle with some transparency
            Draw.CircleXZ(Vector3.zero, 0.5f, Color.yellow * new Color(1, 1, 1, 0.5f));
        }
    }
}
```

Videos cannot be played in pdfs. Take a look at the online documentation for the video which normally goes here.

> See
> GizmoContext (http://arongranberg.com/aline/documentation/beta/gizmocontext.html)

Project Settings

Sometimes it can be convenient to adjust the opacity of gizmos globally. This can be done in the project settings. Solid objects (e.g. Draw.SolidBox (http://arongranberg.com/aline/documentation/beta/solidbox.html#SolidBox3)) are by default drawn slightly transparent. You can adjust this in the project settings if you wish.


Inspector
Project Settings
Adaptive Performance
ALINE
Audio
Burst AOT Settings
Editor
Graphics
Input Manager
Memory Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Tags and Layers
Time
Toolchain Management
UI Builder
Version Control
XR Plugin Management
ALINE
Lines
Opacity
1
Opacity (occluded)
0.12
Solids
Opacity
0.55
Opacity (occluded)
0.45
Text
Opacity
1
Opacity (occluded)
0.9
Opacity of lines, solid objects and text drawn using ALINE. When drawing behind other objects, an additional opacity multiplier is applied.
Reset to default