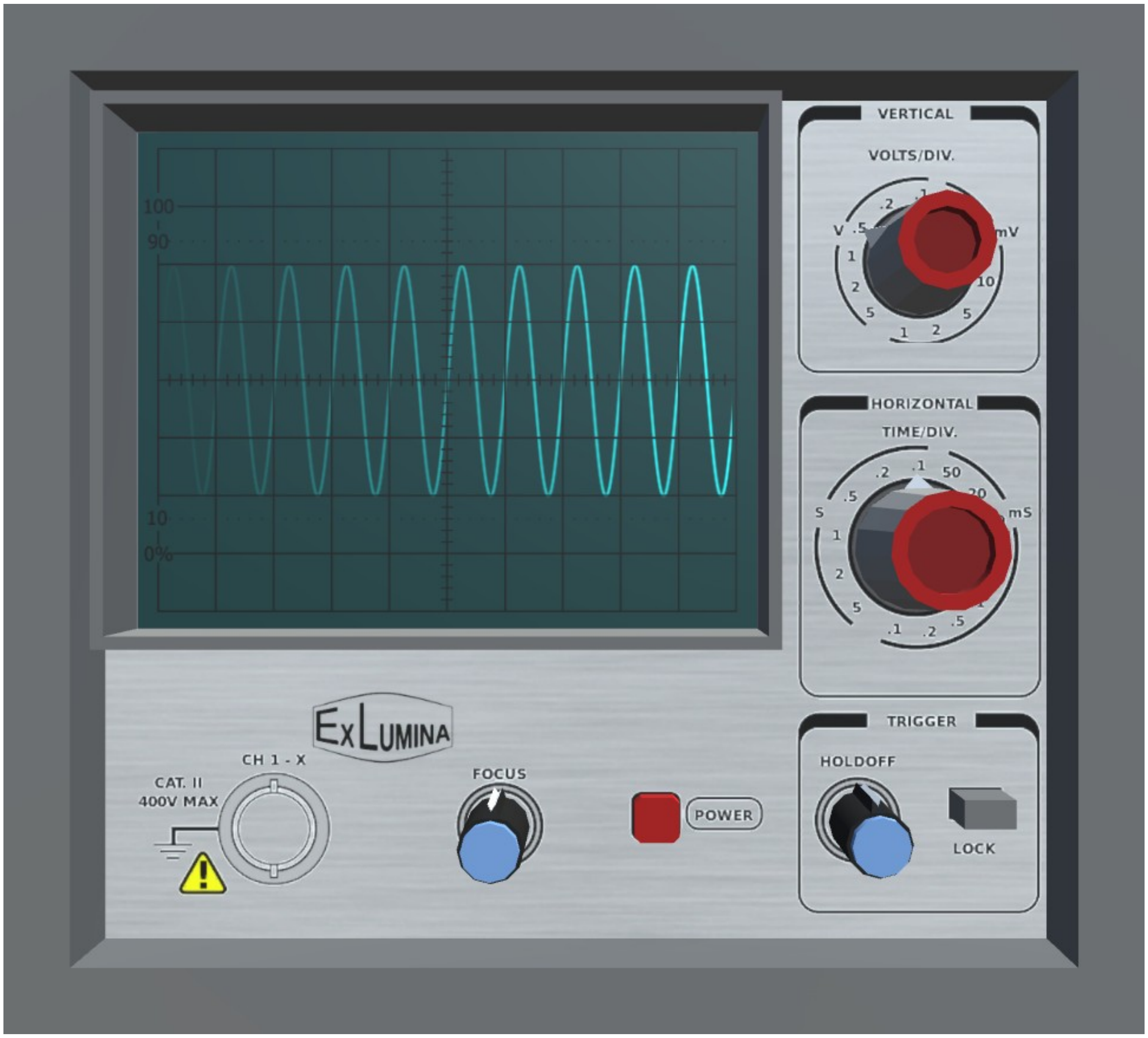
Ex Lumina Oscilloscope 2.0
User's Guide



April 27, 2020
Document revision 2.0.2

Table of Contents

Introduction

Thanks for buying the Ex Lumina Oscilloscope for Unity! You have a low-poly, multi-function simulator that provides a close copy of the cathode ray tube instruments that were common in electronics labs and classrooms throughout the second half of the 20th century. Using the scope in your games is easy. All you have to do is drag and drop either the **ScopeComplete** or the **ScopeScreen** prefabs from the Ex Lumina / Oscillocope / Prefabs folder into your scene. It is fully configured and will begin showing a classic fading sine wave sweeping across the screen when your scene starts.

The rest of this User's Guide will help you make use of the features of the Oscilloscope, including its ability to change the shape, frequency, and amplitude of the signal it displays, in real time, without blocking your game's user interface. Note that this is *not* done by simply scaling a single texture. Instead, the Ex Lumina Oscilloscope recomputes a new trace for every change you make in any of its parameters, resulting in an accurate, sharp, beautifully anti-aliased curve. You can create your own signal function generators to draw any waveform you like, or just use the included **FGCurve** ScriptableObject and Unity's built-in curve editor to create an arbitrary shape. Whatever it is, the Ex Lumina Oscilloscope will draw it for you!

We always want to hear from you, whether for praise or problems. Contact us by e-mail via support@exlumina.com, or visit our Facebook page, "Ex Lumina Corp."

Getting Started

Your Ex Lumina Oscilloscope includes two prefabs that you can use in your game scenes. They are called **ScopeComplete** and **ScopeScreen**, and are in the Ex Lumina / Oscillocope / Prefabs folder. To see them in action, start any of the scenes in the Ex Lumina / Oscillocope Demos / Scenes folder. Briefly, here's what's in each one:

Full Scope – A face-on view of the **ScopeComplete** prefab. Click to the left and right of the centers of the knobs to turn them. Click on the buttons to toggle them on and off.

Screen Only – A face-on view of the **ScopeScreen** prefab. Click on the three colored quads to turn the scope on and off ("Reuse" turns the scope on without computing a new trace). Adjust the parameters in the inspector to see how they affect the trace.

Workbench – An angled view of the **ScopeComplete**. This scene adds a prop to stand in for the FunctionGenerator. In addition to the controls as described in the "Full Scope" scene above you can change frequency and amplitude by pressing and holding the up and down buttons by the numeric displays, and also change the waveform by selecting one of the three "Waveform" buttons. Click on the blue arrows to the left or right side of the view to swing the camera from one device to the other.

ScopeScreen

If you only want a classic oscilloscope screen, with no surrounding enclosure, use the **ScopeScreen** prefab. Simply drag and drop an instance of it into your scene. It is preloaded with all settings and references needed to begin operating right away.

ScopeComplete

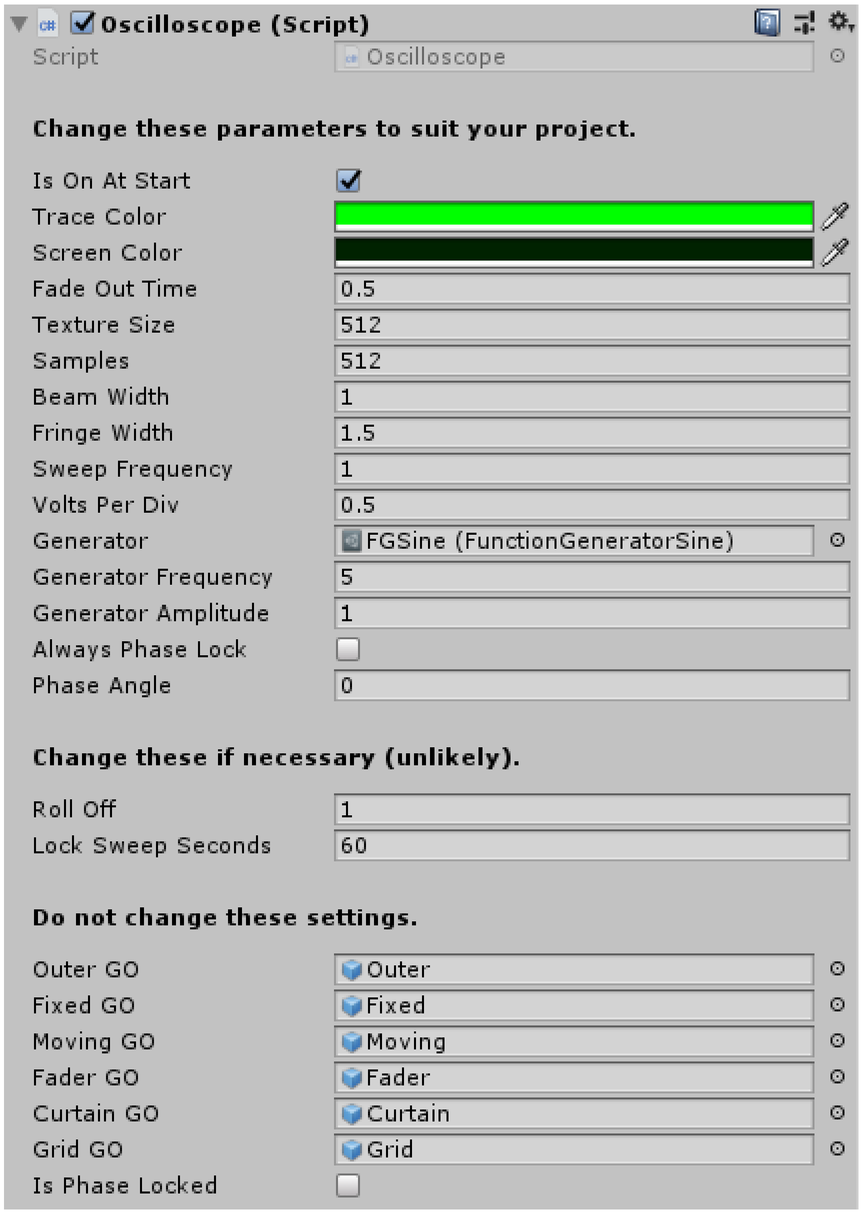
If you want a fully populated three-dimensional prop that provides all the features of **ScopeScreen** but also has its own cabinet, knobs, and buttons, use the **ScopeComplete** prefab. Its operation is identical to that of **ScopeScreen**, but looks more like a classic vintage '70s "sillyscope," which might be just what you want.

Prefab Sizes

The **ScopeComplete** prefab closely models a real oscilloscope, with a screen that is about eight centimeters high and ten centimeters wide. The **ScopeScreen** prefab, on the other hand, is 1.00 meters high, and 1.25 meters wide. This is because most games work best at scales that are in the natural range of things the sizes of cars, barns, trees, starships, and other common everyday objects. In reality, the Ex Lumina Oscilloscope's screen would be about ten centimeters wide, and eight centimeters high (like the **ScopeComplete**'s screen is). To fill the player's screen with the default field of view of 60 degrees, your camera would have to be about seven centimeters away from the screen. Unity's default near clipping plane for its cameras is 30 centimeters, which would clip the oscilloscope screen out of view if you moved it to only seven centimeters away. If you want your screen to be “actual” size, scale your **ScopeScreen** prefab instance in X and Y (but not Z!) by 0.08, and set your camera’s near clipping plane to 0.05. For reasons relating to accuracy in rendering three-dimensional objects, you want the ratio of the camera’s far clipping plane to its near clipping plane to be as low as you can get it. Unity’s default camera far clipping plane is 1000 meters, yielding a ratio of 3,333-1/3 : 1. If you set your near clipping plane to 0.05 (that is, to five centimeters), a far clipping plane of about 167 will keep the same ratio. (Most likely, if you have moved your camera to a point that close to your oscilloscope, or any other 3D prop, you won’t need to see more than 100 meters elsewhere in the same scene.)

Parameters

The Ex Lumina Oscilloscope provides you with a rich set of parameters you can use to customize the appearance and run-time behavior of your scope. When you select the **ScopeScreen** prefab, you will see this list of parameters in the Unity Inspector:



Oscilloscope (Script)

Script	Oscilloscope

Change these parameters to suit your project.

Is On At Start	✓
Trace Color	🟩
Screen Color	■
Fade Out Time	0.5
Texture Size	512
Samples	512
Beam Width	1
Fringe Width	1.5
Sweep Frequency	1
Volts Per Div	0.5
Generator	FGSine (FunctionGeneratorSine)
Generator Frequency	5
Generator Amplitude	1
Always Phase Lock	☐
Phase Angle	0

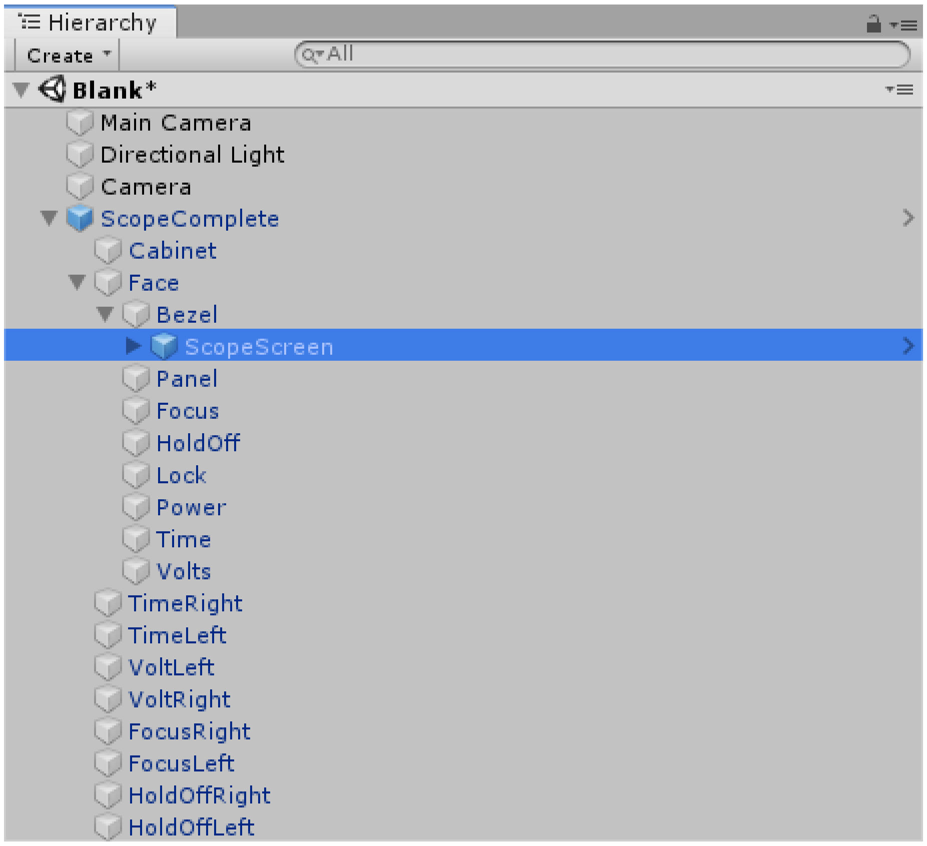
Change these if necessary (unlikely).

Roll Off	1
Lock Sweep Seconds	60

Do not change these settings.

Outer GO	Outer
Fixed GO	Fixed
Moving GO	Moving
Fader GO	Fader
Curtain GO	Curtain
Grid GO	Grid
Is Phase Locked	☐

Note that, if you are using the **ScopeComplete** prefab, the **ScopeScreen** prefab is a child GameObject of its hierarchy. Select it in the Hierarchy window to see its parameters in the Inspector:



```
Blank*
  Main Camera
  Directional Light
  Camera
  ScopeComplete
    Cabinet
    Face
      Bezel
        ScopeScreen
        Panel
        Focus
        HoldOff
        Lock
        Power
        Time
        Volts
    TimeRight
    TimeLeft
    VoltLeft
    VoltRight
    FocusRight
    FocusLeft
    HoldOffRight
    HoldOffLeft
```

Each of the parameters is initially set to a good working value. But you can change them to suit your particular needs. Full information on each is in the following sections.

Is On At Start

A checkbox that sets whether or not your oscilloscope is on and showing a trace when your scene starts. Check to have it on at start, uncheck to leave it off. The default is checked.

Trace Color

Chooses the color of the glowing trace on your oscilloscope screen. The classic choice is green, so that’s the default. However, many oscilloscopes of the twentieth century used other dyes in their phosphor. A popular choice was a light blue. But you can choose any color you like. In general, however, the brighter, the better.

Screen Color

This is the color your phophor screen shows in regions where there is no glowing trace, which is most of the screen. Its default is a dark green. In general, make this about one-eighth of the values of your Trace Color setting (but, again, you can use whatever you like).

Fade Out Time

The old cathode ray tube oscilloscopes did not just “click” off. Their phosphor screens gently faded out when powered down. This setting controls how long, in seconds, it takes for your phosphor to fade out when you turn off your scope. Its default is half a second.

Texture Size

The trace is drawn onto a texture that is subsequently loaded into your scope prefab for display. The bigger your texture is, the sharper your trace will look. You will generally find that a size of 512 is big enough. If your needs don’t call for a close-up view of the scope screen, you can use a smaller size. Remember that Unity advises using powers of two, so pick a value from 64, 128, 256, 512, 1024, etc. The default is 512.

Samples

The signal drawn on your scope screen is “sampled” at equally spaced intervals, then drawn by connecting the samples with straight lines. If your waveform is smooth, you won’t need many samples. At the upper end, you won’t see any improvement in how smooth the trace is if you take more samples the width of your texture, but you can often obtain excellent results with far fewer samples. Try smaller numbers, down to 100 or less, if your waveform is a smooth one. If your waveform is complex, with spikes and notches, keep your sample count at or near the texture size. The default is 512.

Beam Width

The trace simulates a trail of glowing phosphor created by an electron beam. The width of the trace depends on the width of the beam. Use this parameter to set the beam width in pixels. This part of the trace will be entirely in the color you chose for the Trace Color parameter, rather than blending into the unlit screen color. The default is one pixel.

Fringe Width

The region of the phosphor screen near where the electron beam passes glows in a gradient that fades from full color (the Trace Color value) to the fully dark (the Screen Color parameter). This is how your Ex Lumina Oscilloscope achieves the smooth, antialiased traces it draws. You can simulate an unfocused beam by increasing this value. Its default is 1.5 pixels.

Sweep Frequency

A cathode ray tube oscilloscope draws its trace by sweeping the electron beam horizontally, while the signal controls its vertical position. Use the Sweep Frequency parameter to control how often the trace appears. Note that, if you want to see exactly one full cycle of your signal on the scope screen, your Sweep Frequency should equal your Generator Frequency (see below). If you set the Sweep Frequency to half that of your Generator Frequency, you will see two full cycles. At a Sweep Frequency of one-third of your Generator Frequency, you will see three full cycles, and so on. Note that setting the Sweep Frequency higher than the Generator Frequency will not produce a satisfying result. The default is one sweep per second.

Note: Unlike all other parameters, if you change the Sweep Frequency from C# code, do *not* change the value of the Oscilliscope's `sweepFrequency` public member field. Instead, change its `SweepFrequency` public property. See the "Public Properties" section under "Controlling the Oscilloscope from Code," below.

Volts Per Div

The Ex Lumina Oscilloscope screen has a grid drawn on it that is eight squares high and ten squares wide. This parameter controls how many volts are indicated by each square (or "division") in the vertical dimensions. The higher you set this, the smaller the trace will be for a given signal. Conversely, the lower you set this, the bigger the trace will be for a given signal. Note that, as there are eight divisions, if you set the value to 0.25, a signal value of one volt will be four division high, at the top of the screen. The default value is 0.5 volts per division.

Generator

Your scope displays the signal it receives from a virtual signal generator (also called a "function generator"). The Ex Lumina Oscilloscope uses a subclass of Unity's ScriptableObject, called FunctionGenerator, to provide the signal it shows on its screen. You can use any of the FunctionGenerator objects included, or write your own. See the "Function Generators" section for more information. The default choice is the FGSine Function Generator, which produces a sine wave.

Generator Frequency

This controls the number of oscillations per second your virtual signal generator produces. The higher this value is, the more cycles of your signal you will see per trace. The default is 10 cycles per second.

Generator Amplitude

This value is applied to the value from the signal generator as a multiplier. All of the included FunctionGenerator objects produce signals that range in value from one to negative one. So, setting this value higher increases the virtual signal amplitude. Setting it lower decreases it. Think of this as the voltage level your signal will reach at it highest and lowest points. The default is 1.

Always Phase Lock

When your Generator Frequency is an integral multiple of your Sweep Frequency, you will see a fixed, unmoving trace, showing a number of cycles of your signal equal to that multiple. In other cases, the trace will appear to move as, on each new sweep, the signal will somewhere in the range of values other than where it starts at the beginning of a cycles. You can obtain excellent animations that provide a convincing dynamic effect by having the Generator Frequency be slightly above or below an intergral multiple of your Sweep Frequency. In any case, however you can force the scope to draw its trace from the first value in your signal's cycle by setting this parameter to "true" by checking its box. (On a real oscilloscope, this option is called "triggered sweep.") The default is unchecked.

Phase Angle

When Always Phase Lock is checked (or set true), this value controls where in the cycle from your signal generator the trace begins drawing. If you set it to zero, your trace starts from the first point In your signal's cycle. If you set it to .25, it starts one-fourth of the way through your signal's cycle. A value of one-half starts half-way through your signal's cycle. The values map from zero to one, linearly, to the start and end of your cycle. (Note that, electronics engineering parlance, "phase angle" is a value between 0 and 2π. Here, it is from zero to one, to simplify your coding.)

Roll Off

Phosphor fades over time after the electron beam has crossed over it. Use this parameter to control how quickly the fade happens. This is implemented as an exponential value. Higher values cause the phosphor to fade more quickly, while lower values make it fade slowly. The default value is 1.

Lock Sweep Seconds

At some point, if the trace is moving slowly enough (because your Generator Frequency is almost, but not exactly, an integral multiple of your Sweep Frequency), it might as well be stopped entirely, This saves a bit of CPU overhead, so this setting lets you have the trace stop moving entirely if it would take longer than some amount of time to make a complete horizontal traversal of the scope screen. The default value is 60 seconds.

Is Phase Locked

Read-only.

You can read this value to see if the trace is locked or not. That is, if either the Always Phase Lock parameter is true, or the Generator Frequency is an integral multiple of the Sweep Frequency. Don’t set this, however. Treat it as read-only.

Controlling the Oscilloscope from Code

Your Ex Lumina Oscilloscope is fully controllable from C# code you can write yourself. You can change any of its parameters, as well as turn it on and off.

Public Fields (Parameters)

All of the parameters in the “Parameters” section can be accessed from code. Just get a reference to the Oscilloscope component of the GameObject with your scope on it (either with the GetCompoment API call, or by adding a GameObject public field to a MonoBehaviour script of your own and dropping the ScopeScreen instance from the Hierarchy into that field in the Inspector). In every case except Sweep Frequency (see the next section), the name of the field matches the name as shown in the Inspector, but without the spaces and with the first letter always in lower case. So, for example, if you want to set the Generator Frequency, your code would look like this:

```
Oscilloscope scope = scopeScreen.GetComponent<Oscilloscope>();
scope.generatorFrequency = 50;
scope.LoadNew();
```

Note that you can change as many parameters as you like without incurring any overhead. When you have finished setting parameters, call either `LoadNew` to have them take effect immediately if the scope is on (or else the next time `On` is called), or call `On` to have them take effect immediately.

Public Properties

Oscilloscopes usually do not permit a direct setting of their sweep frequencies. Instead, they have controls that allow the operator to set the number of seconds per horizontal division. The Ex Lumina Oscilloscope lets you use whichever you prefer, via two interacting public properties. To set the Sweep Frequency from code, assign your new frequency to the `SweepFrequency` property. Or, if you would rather set the seconds per division, assign that to the `SecondsPerDiv` property. Each updates the other correctly, and each can be written to or read from.

Note: Do not change the `sweepFrequency` public member field from code. Use the `SweepFrequency` public property instead.

Public Methods

Your Ex Lumina Oscilloscope provides three methods, all related to turning it on or off.

```
public void On(bool useLastTrace = false)
```

Turn the scope on if it isn't already on. If useLastTrace is false, compute a new trace on a worker thread and, when it is ready, start displaying it on the scope screen. If useLastTrace is true, do not compute a new trace. Instead, use the last trace computed. Note that, if the worker thread computing a trace from a prior call to the On method is still running when On is called, computation of the newest trace is deferred until after the trace still being computed is completed. In the extreme (and unlikely) case where multiple calls to the On method are made while the worker thread is still computing a prior trace, all intermediate calls are discarded and the trace drawn by the last call is deferred until the worker thread completes the trace it is working on. The ultimate effect is that the last trace requsted is the one you will see, with as little time as possible wasted computing traces you won't see.

```
public void Off()
```

This method simply turns the scope off if it is not already off. Note that, while the call returns immediately, the scope phosphor will fade out gradually as per the value in the Fade Out Time parameter.

```
public void LoadNew()
```

If you have changed any parameters and would like them to take effect without changing the scope's on or off condition, use `LoadNew`. If the scope is on, the new trace will be computed and drawn. If not, the new trace will be computed and drawn the next time `On` is called.

Function Generators

You can create your own ScriptableObjects to provide any signal you want the Ex Lumina Oscilloscope to use. Simply subclass them from FunctionGenerator and override the SampleAt method.

```
public abstract float SampleAt(float t)
```

This method returns a value, nominally on the range [-1, 1], to indicate signal level at time t. The Oscilloscope code assumes that your signal's cycle begins at t=0, and that exactly one full cycle is complete when t=1. However, the `SampleAt` method must be able to return a valid signal value for any value of t passed to it. If your implementation only works on the range [0. 1], you can use the FunctionGenerator's protected method mod1:

```
protected float mod1(float t)
```

This method returns a value in the range [0, 1) (that is, from 0 up to, but not including 1), equal to however much the t is greater than the its floor integer value. Thus, `mod1(5.6)` is 0.6, whereas `mod1(-5.6)` is 0.4.

Note that your class must also include a CreateAssetMenu attribute in order to allow you to create your new ScriptableObject from the Unity editor. See the FunctionGeneratorSine class source code for a good example.

Operation

With the foregoing information, you can create instances of the Oscilloscope prefab, change its settings, control it from code, and alter its behavior in any way you like. To understand what you'll be seeing, here are some points to keep in mind.

Drift and Lock

As described in the "Always Phase Lock" section above, if the Generator Frequency is an integral multiple of the Sweep Frequency, the trace is drawn in exactly the same place on the scope screen for each sweep. If not, the trace is drawn in a different place. When the difference only slightly departs from an integral multiple, the difference from one sweep to the next will be slight, causing the trace to "drift" smoothly to the left or right. When the multiple is an exact integer, the trace will lock into place. The Oscilloscope code detects this and, when the trace is locked, applies no animation. This is the use of the scope that create the least runtime overhead. However, animating the drift creates only slight overhead.

Flicker and Fade

When the sweep frequency is higher than 25 sweeps per second, the Oscilloscope will draw a single trace and, if it is drifting, animate it to show the drift. Below 25 sweeps per second, the Oscilloscope simulates the fading phosphor a real scope would have, by adding a slow fade to the animation. Even a non-drifting (that is, a locked) trace will show this fade. At moderate sweep frequencies, anywhere from about 5 to 25 sweeps per seconds, the flicker will seem to predominate the effect. Below about 5 seconds, the slow fade of the phosphor becomes visible. At these lower sweep frequencies, the visual effect is dramatic and can add a great sense of atmosphere to your scene.