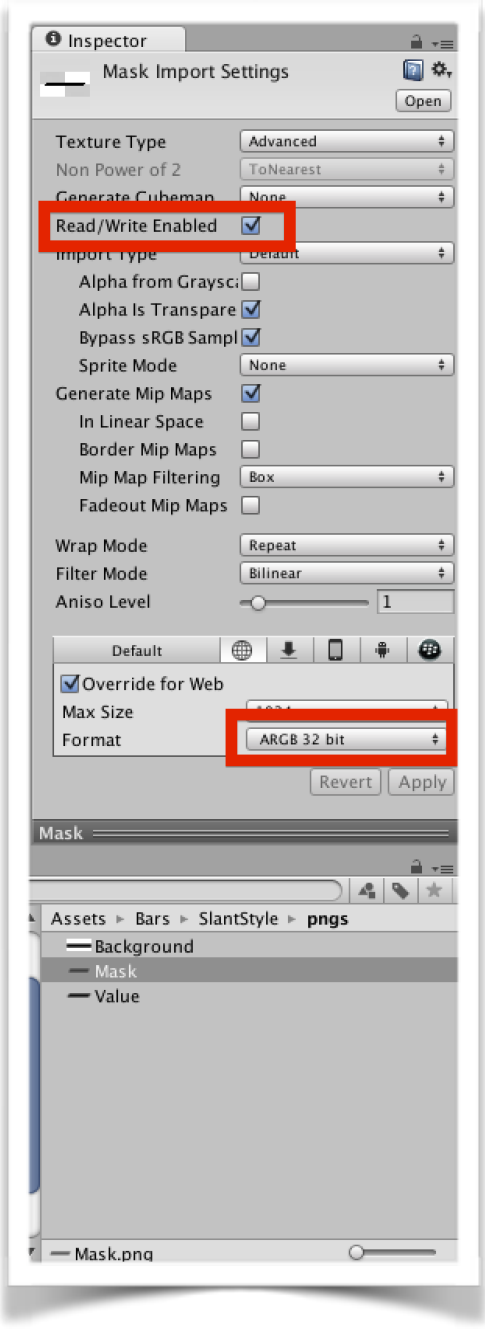
Gradient GUI Bars

Hello Thanks for purchasing this product.
If you need to contact me for any reason you can find my contact information here.
Please, consider donating to me (link)

This product uses one script to display a GUIbar, and the text in percent on the screen.
These GUIbar can display anything from Health, Mana, Special, Energy, and many other things.

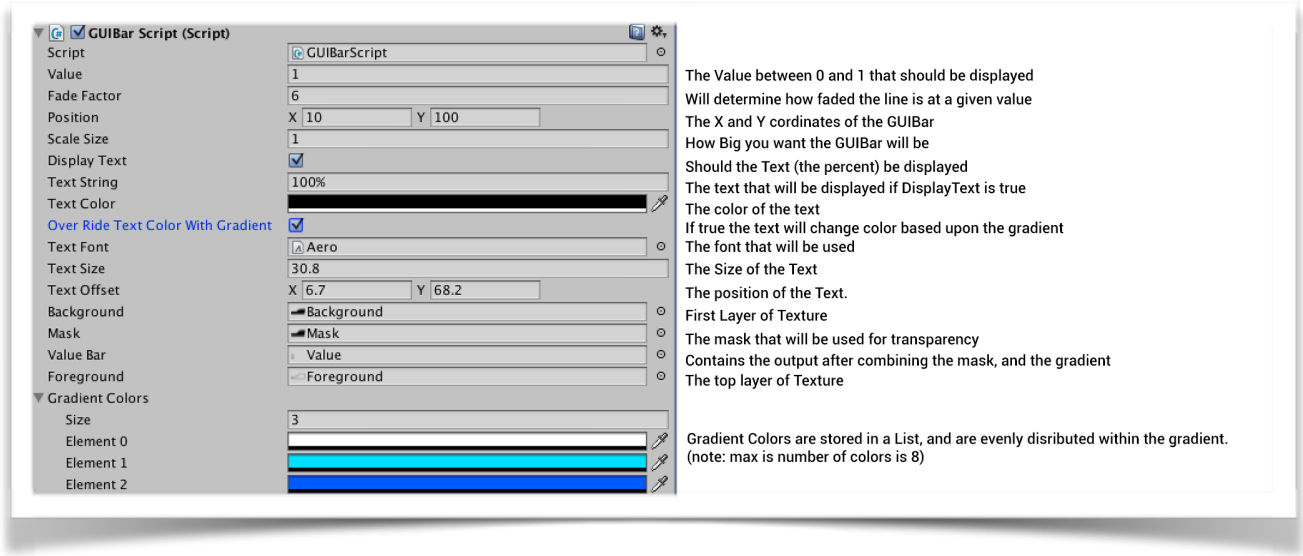
Note: if you decide to make your own custom theme you need to adjust the Import settings for the Mask.png file you use, and the Value.png file you use. (see image below)

the only two that have to change is the Read/Write Enabled, and the Format



Gradient GUI Bars

Below is a little overview of the variables you can see in the Unity UI.



The rest of this document will be referring to the lines of code in GUIBarScript.cs.

Line 7 to 44
Declaring Variables

```
	//Declare variables

	//for the CurrentValue, and the Value it will be after the
update
	private float CurrentValue;
	public float Value;

	//FadeValue is current amount the bar is faded
	private float FadeValue;
	//FadeFactor is a little complex, Open the ReadMe file to
know more
	public float FadeFactor = 6f;

	//position and scale of the GUIBar on the screen
	public Vector2 Position;
	public float ScaleSize;

	//Font Variables
	public bool DisplayText = true;
	public string TextString;
	public Color TextColor;
```

```
public bool OverRideTextColorWithGradient = false;
public Font TextFont;
public float TextSize;
public Vector2 TextOffset;

//Carries the .png images to display the GUIBar
public Texture2D Background;
public Texture2D Mask;
public Texture2D ValueBar;
public Texture2D Foreground;

//Carries the colors that the GUIbar will be
public List<Color> GradientColors = new List<Color>();

//These are used for redrawing the GUIBar
private Gradient g = new Gradient();
private GradientColorKey[] gck;
private GradientAlphaKey[] gak;
private Color[] MaskPixels;
```

Lines 46 to 95
Drawing OnGUI Method

Within these Lines draw the textures that are needed, as well as the precent. However the most important thing it's doing is limiting the number of times the UpdateBar Method is run.

Lines 50 to 56
Is just an if statement that decides weather an update is needed based on the Value and CurrentValue variables.

WARNING!

running the UpdateBar Method can reduce FPS (frames per second)

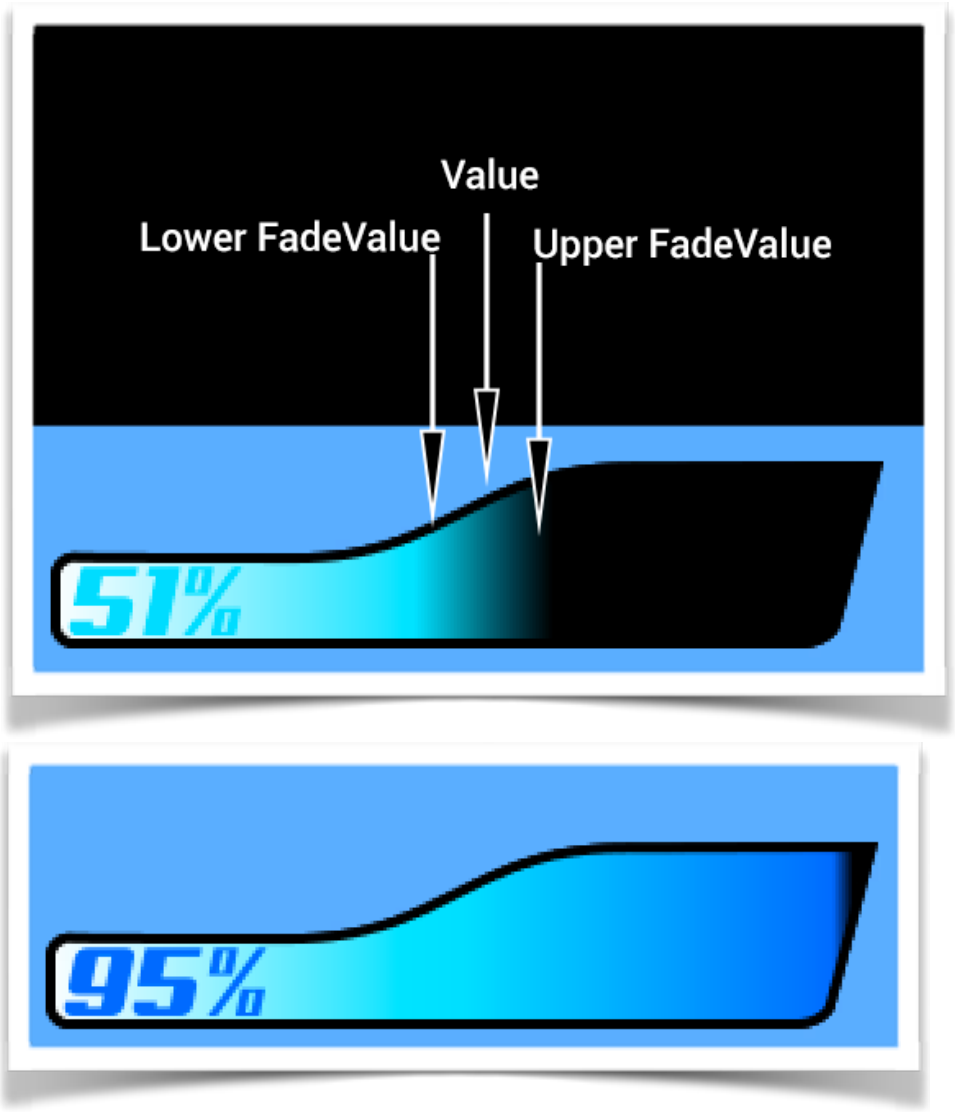
This is because updating the bar requires that we loop through each pixel in the Mask texture.

Lines 97 to 131
This is the UpdateBar Method that updates the gradient and maps it to the ValueBar texture using the Mask texture.

Lines 133 to 201
This is the UpdateGradient Method that changes the gradient based on the Value.

Line 146
this line uses a little bit of trig so that the FadeValue can be set.

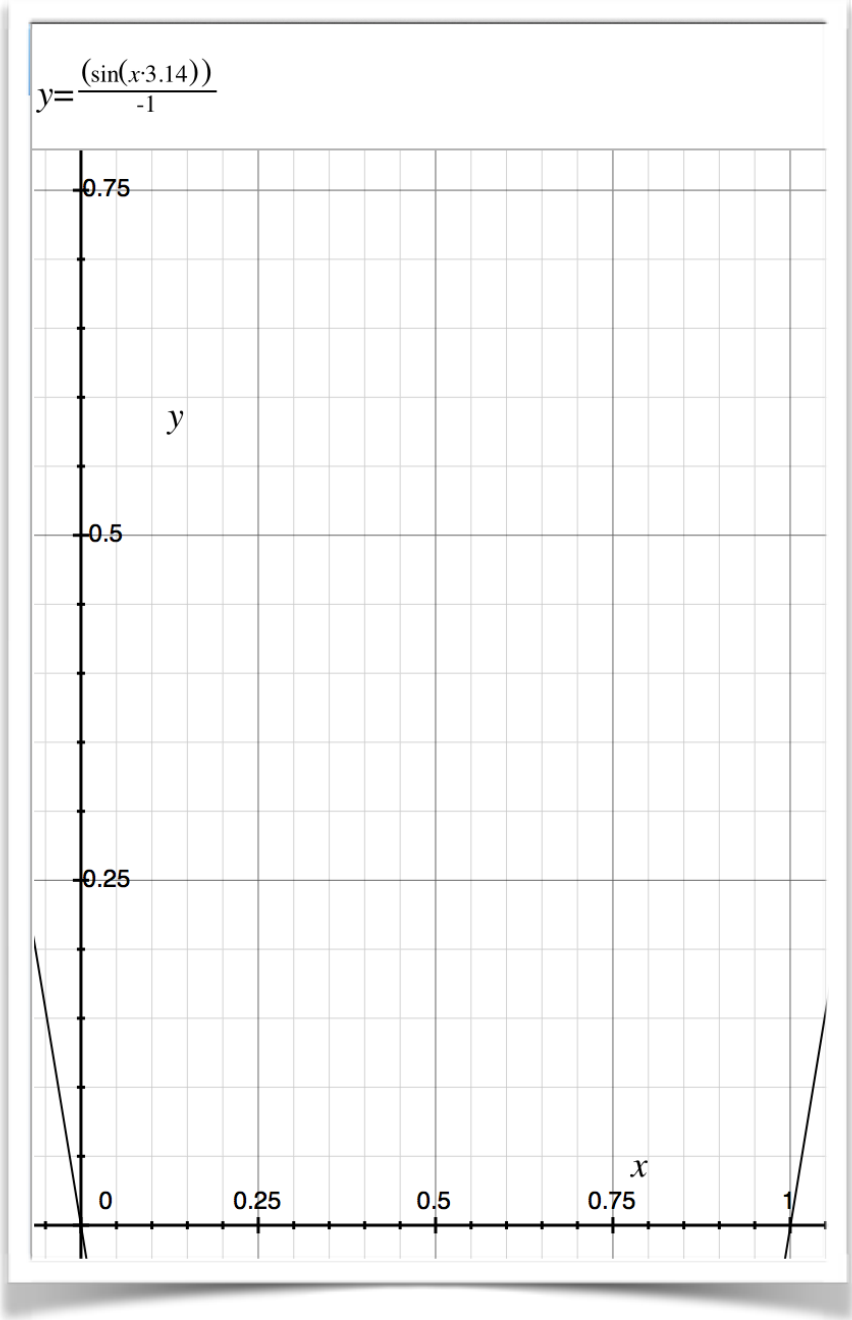
```
FadeValue = ((Mathf.Sin ((Value) * 3.14f))/FadeFactor) ;
```



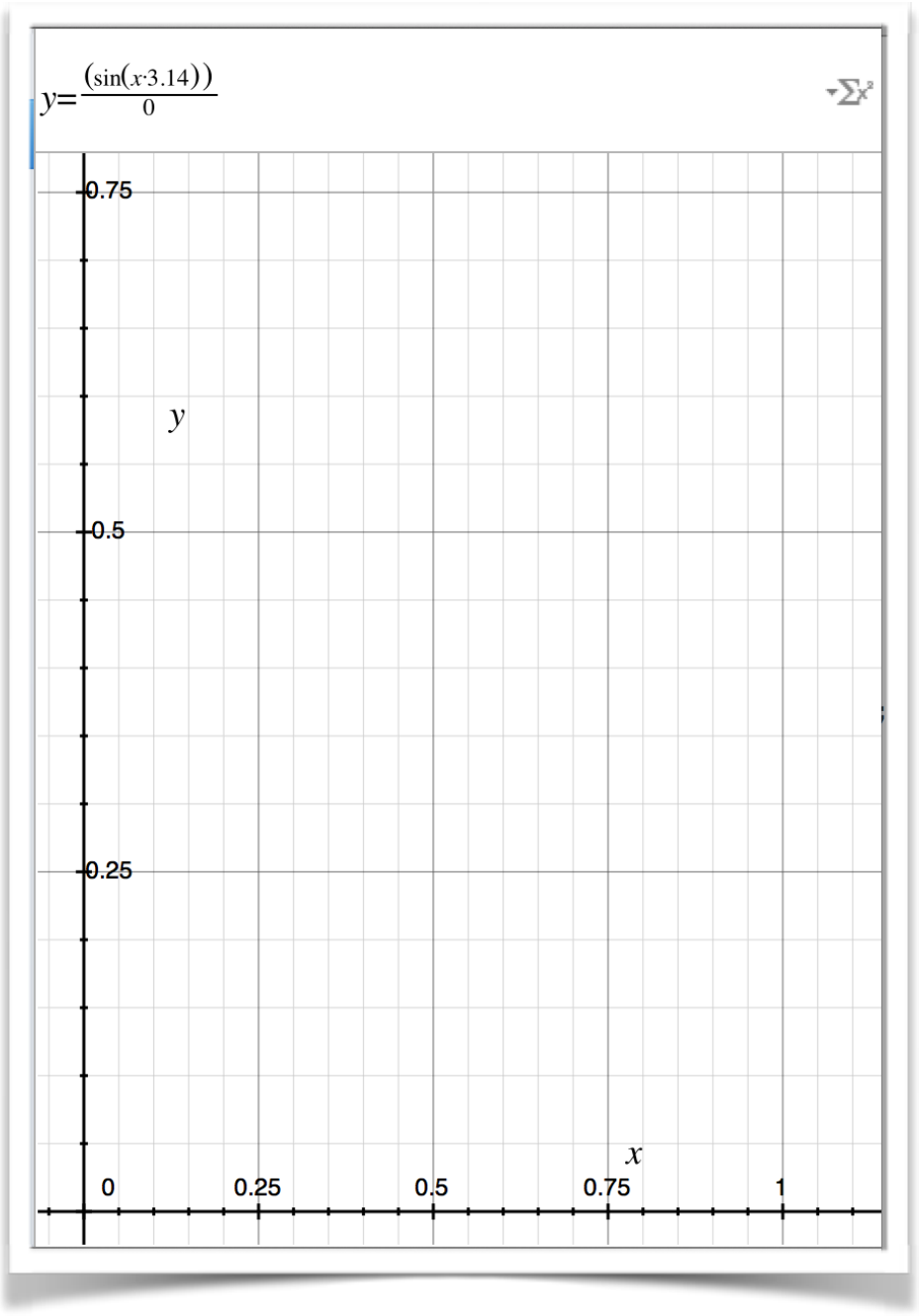
So the Largest Fade will always be at 50% (0.05), and the fade will be at 0 when the value 100 or 0.

so lets view a few graphs.

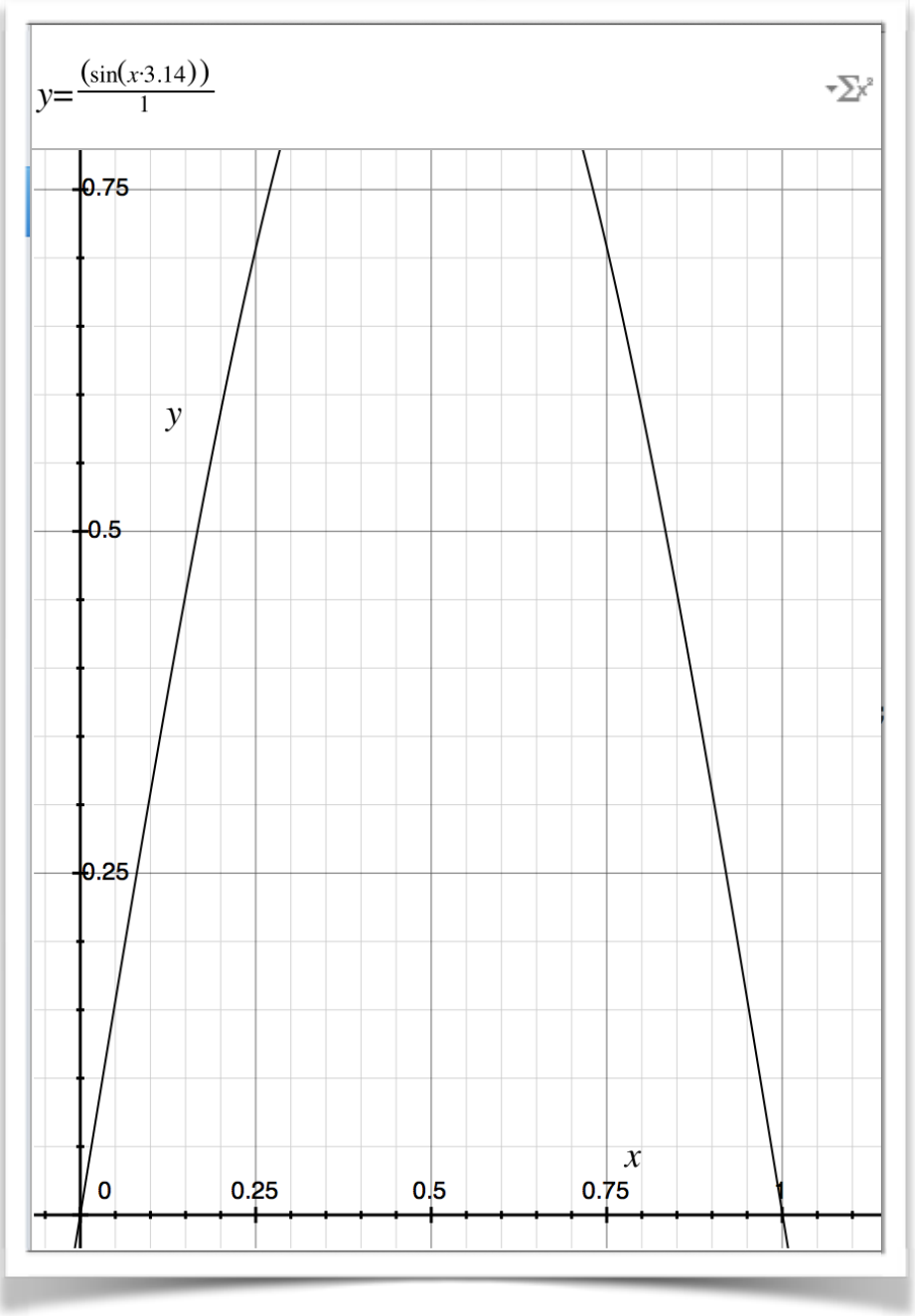
if the FadeFactor is -1, the FadeValue should be Negative as well but i am overriding it to 0.



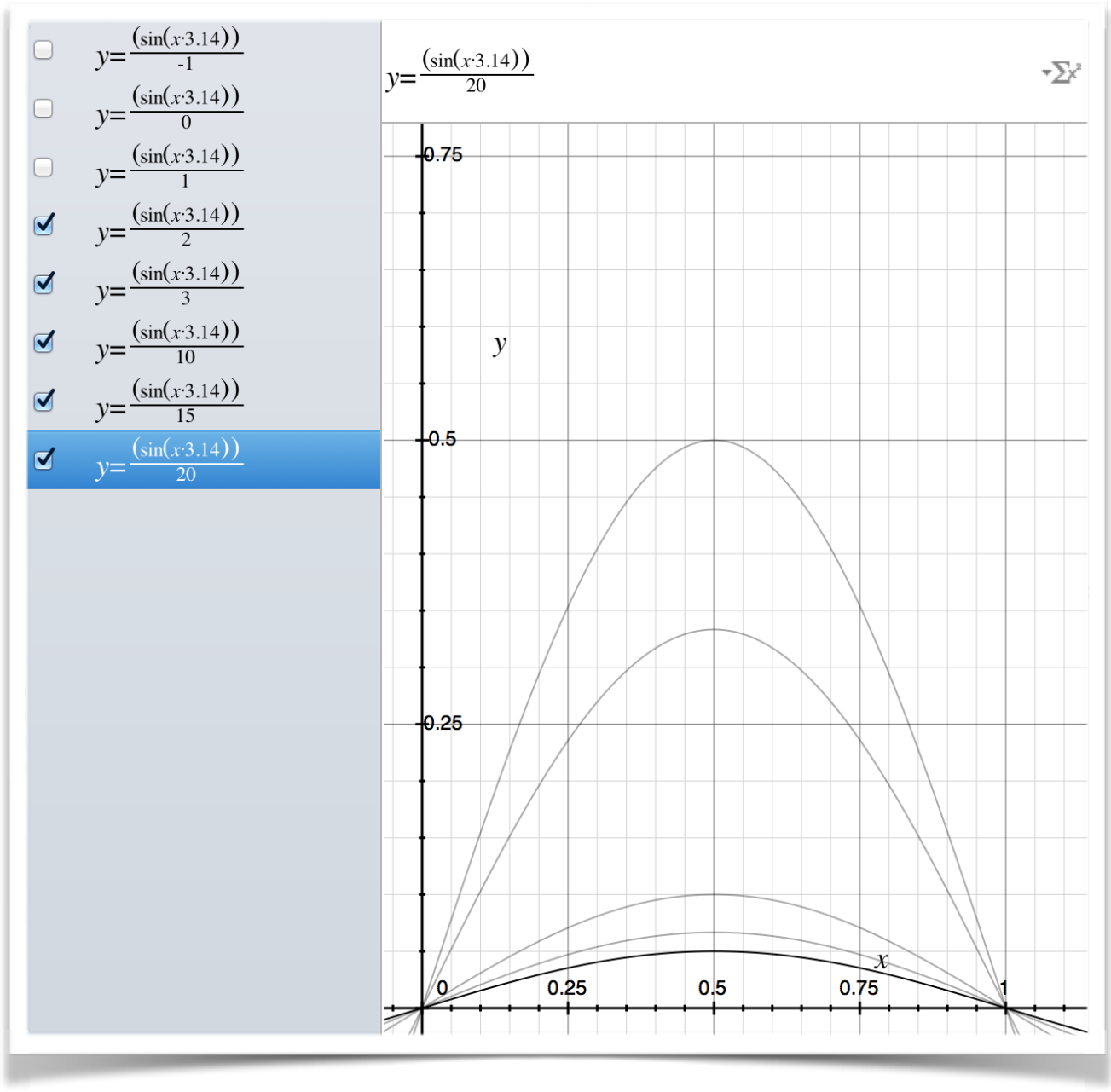
```
if the FadeFactor is 0, the FadeValue will be
undefined, therefore it does not render well.
```



This is the graph when the FadeFactor is 1.



This is the graph when the FadeFactor is 2,3,10,15,20.



Lines 206 to 246
These lines of code contain public Methods that can be called if needed.

AddNewColor
Adds a new color to the gradient.

ChangeColor
Changes a color in the gradient.

RemoveColor
Removes a color from the gradient

SetNewValue –Override
Sets a new Value and (if needed will update the bar)

ForceUpdate
This will force the update of the bar.