sign up log in tour help stack overflow careers

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

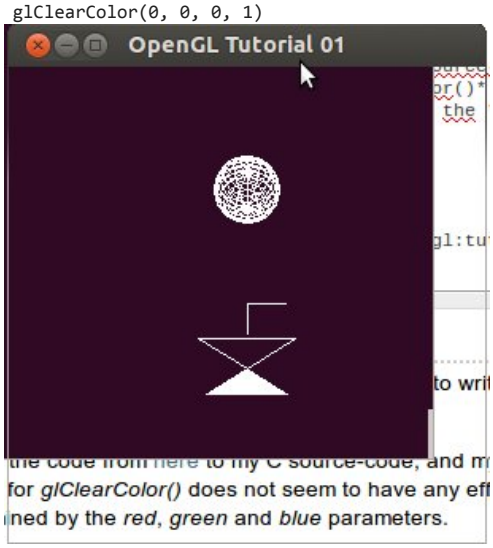
Take the 2-minute tour ×

glClearColor(0, 0, 0, 0) is transparent, not black

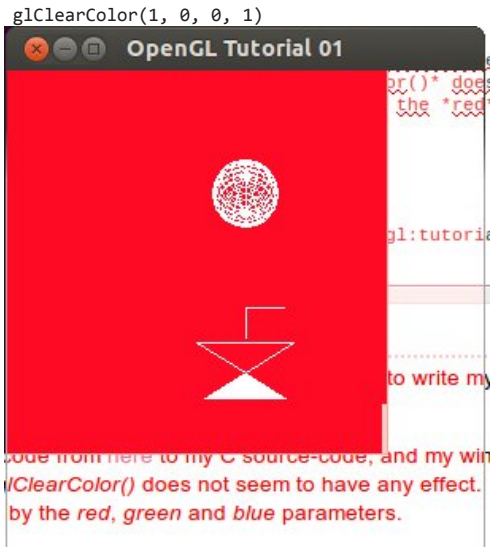
I'm trying to use OpenGL my first time. I was looking at some code online and then tried to write my own, but I always just got an empty (transparent) window. (I used GLUT to open the window).

I thought I did something wrong, so I copyied the code from here to my C source-code, and my window is still transparent. Also, the *alpha* parameter for *glClearColor()* does not seem to have any effect. Instead, the alpha-value **seems** to be determined by the *red*, *green* and *blue* parameters.

```
glClearColor(0, 0, 0, 0)
glClearColor(0, 0, 0, 1)
```

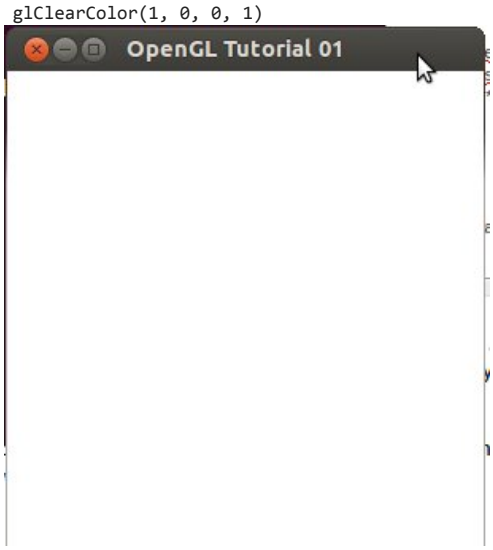



```
glClearColor(1, 0, 0, 0)
glClearColor(1, 0, 0, 1)
```




```
glClearColor(1, 1, 0, 0)
glClearColor(1, 1, 0, 1)
```


```
glClearColor(1, 0, 0, 0)
glClearColor(1, 0, 0, 1)
```



The *alpha* parameter doesn't change the result.

I use Ubuntu 12.04 LTS, *libgl1-mesa-dev*.

Is this a bug or do I do something wrong?

c linux opengl ubuntu freeglut

asked Aug 29 '12 at 10:01

Niklas R
2,720 5 28 70

A note about your question's title: (0, 0, 0, 0) is supposed to be transparent, isn't it? Your problem might rather be "(0, 0, 0, 1) is transparent, not black." – aib Aug 29 '12 at 10:23

@aib Not sure, actually. The fourth parameter is called *alpha*. I guess its `opacity = 1 - alpha` , but I couldn't test it as it disn't have any effect. :) Also, among the tutorial that linked to the code linked above, it

results in a black background. – Niklas R Aug 29 '12 at 12:20

3 @NiklasR: opacity == alpha in the usual case. – datenwolf Aug 29 '12 at 12:39

1 Answer

GLUT does not request an alpha buffer by default, and I suspect what you're seeing might be a fail-safe, ad hoc implementation of window transparency. Try adding:

```
glutInitDisplayMode(GLUT_RGBA | GLUT_ALPHA);
```

near the other GLUT init calls. Keep this function in mind, as you'll need to modify the call again if/when you want depth or stencil buffers, or double buffering.

edited Aug 29 '12 at 10:28

answered Aug 29 '12 at 10:14

aib
20.3k 5 46 65

Thanks, that already had an effect. For example, `glClearColor(1, 1, 0, 1)` is now fully opaque, but `glClearColor(1, 1, 0, 0)` is not, just like in the picture above. Shouldn't it be absolutely transparent when *alpha=0.0*? – Niklas R Aug 29 '12 at 12:44

Yes, any `(r, g, b, 0)` should be fully transparent. Did you try other values such as .8 and .2 for alpha? I suspect they'll round down, but it's still interesting to see what happens. Also, are you absolutely sure your display mode, window manager or whatever backend supports alpha? – aib Aug 29 '12 at 19:15

Well, as `(1, 1, 0, 0)` is a little transparent and `(1, 1, 0, 1)` is fully opaque yellow, setting *alpha* to a value in `[0;1]` seems to interpolate between these. Yes I am, I can make ma Terminal transparent or half-transparent for example. I use the default window-manager that came with LTS 12.04 (uses Unity UI). – Niklas R Aug 29 '12 at 19:20

So `(0, 0, 0, a)` works, and you have a window whose transparency (but not background color) you can adjust? By the way, what are those big rectangles in the background in your pictures? (The ones whose lower-right corner touch the "a" of an "and".) – aib Aug 29 '12 at 22:00

Hm, I admit the pictures are a more confusing than I first thought. It's the terminal-window in the background which originally has a violett background-color. Hm no it doesn't work well, as `a=0` is not fully transparent. :) Well, I'm fine without transparency, but I wonder why `(1, 1, 0, 0)` looks like `(1, 1, 0, 0.66)` . Or is this just by design? – Niklas R Aug 29 '12 at 22:03