lingedeng的专栏

聚沙成塔，集腋成裘

个人资料

lingedeng

访问： 65526次

积分： 904

等级： BLOG 3

排名： 千里之外

原创： 15篇 转载： 4篇

译文： 12篇 评论： 18条

openGL——视图

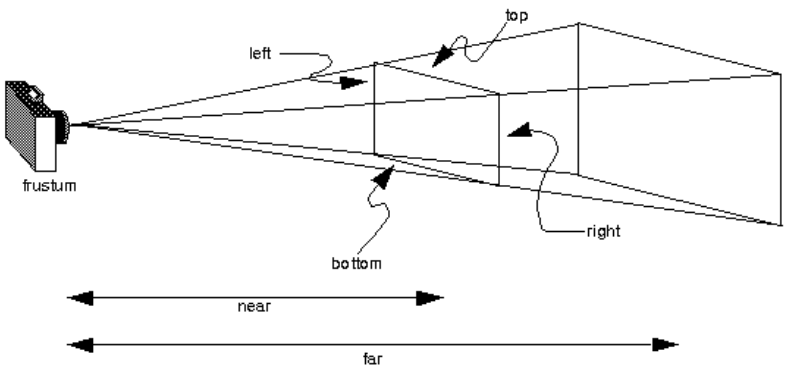
分类： openGL 2012-02-28 15:03 2027人阅读 评论(0) 收藏 举报

import float library buffer 图形 语言

glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
在窗口中定义一个像素矩形，最终的图像会映射到这个矩形中。（x，y）指定了的视口的左下角，width和height指定了这个视口矩形的宽度和高度。在默认情况下，视口的初始值为（0, 0, winWidth, winHeight)，其中winWidth和winHeight指定了窗口大小。你可以将其想像为数码相机的显示屏，是最终成像的地方。

gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);
定义一个视图矩阵，并将其与当前矩阵相乘。目标观察点由eyex，eyey，eyez指定，centerx, centery, centerz指定了视线上的任意一点。upx,upy和upz指定表示哪个方向是向上的（也就是说，在视景体中至下向上的方向）。你可以将其想像为一个数码相机，放置在一定的位置并使镜头指向一个方向。默认值为gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, -100.0, 0.0, 1.0, 0.0)，表示数码相机位于原点并指向z轴负方向，以y轴的正方向为向上方向。

glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar);
创建一个表示透视视图平截头体的矩阵，并把它与当前矩阵相乘。如果观察点为默认情况时，平截头体的视景体是由这个函数的参数定义:(left, bottom, -zNear)和(right, top, -zNear)分别指定了近侧裁剪平面左下角和右上角的(x,y,z)坐标。zNear和zFar分别表示从观察点到近侧和远侧裁剪平面的距离，它们都应该是正值。你可以将其想像为数码相机的视野范围（视景体）。



gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);
创建一个表示透视视图平截头体的矩阵，并把它与当前矩阵相乘。fovy是yz平面上视野的角度，它的值必须在[0.0, 180.0]之间。aspect是这个平截头体的横纵比，也就是宽度除以高度。如果观察点为默认情况时，zNear和zFar分别表示从观察点到近侧和远侧裁剪平面的距离，它们都应该是正值。该函数是glFrustum(...)函数的替代函数。

iphone——使用run loop (4072)
GDataXMLNode——XM (3059)
iphone——Thread,RunL (2952)
iPad页面管理——Split V (2889)
iPad页面管理——Popov (2862)
iphone——Three20库代 (2346)

评论排行

乐视(letv)网tkey破解 (18)
iphone——视图layout (0)
iphone——iOS Human I (0)
Python——文件处理 (0)
openGL——视图 (0)
RFC2616 HTTP/1.1 标志 (0)
A Road Map Through Na (0)
A Road Map Through Na (0)
A Road Map Through Na (0)
Google C++ Style Guide (0)

推荐文章

最新评论

乐视(letv)网tkey破解
xiongzhend: @nihaommaa:我传给你大家一起研究

乐视(letv)网tkey破解
nihaommaa: 小程序工程及运行的Flash player已上传(LEVT_Decode.zip)在哪里下载啊

乐视(letv)网tkey破解
sinat_24431293: @asdsjw:你是解密哪个文件出错了？KLetvPlayer.swf 还是 楼主所说的二进制转成s...

乐视(letv)网tkey破解
asdsjw: 我也是 Error #1010: 术语尚未定义，并且无任何属性。不知道为什么我是用反编译出来，然后放...

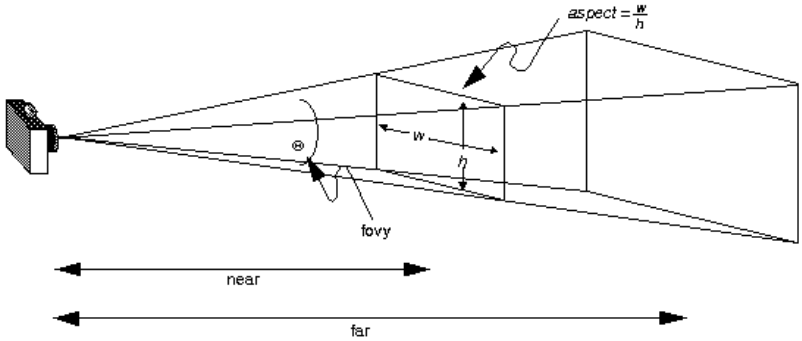
乐视(letv)网tkey破解
sinat_24431293: 楼主还关注乐视的算法吗 现在用你这套程序将二进制转成swf后还是不行，用你的代码解密过程报错： E...

乐视(letv)网tkey破解
rain0605: 楼主的LEVT_Decode.zip文件在哪？怎么没有找到啊？

乐视(letv)网tkey破解
sora0009: LEVT_Decode.zip 没有发现呢通过别人提供的算法已经可以获取到xml 然后再怎么进行组装...

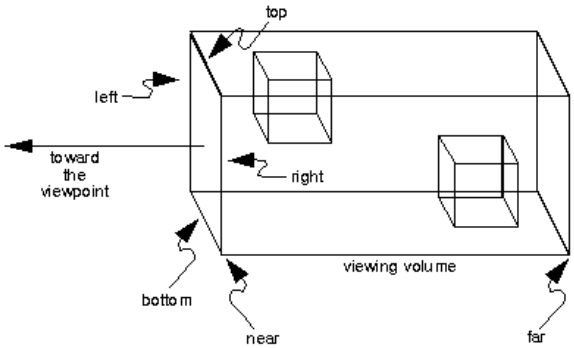
乐视(letv)网tkey破解
cmos1: 乐视单页 代理演示 http://www.austgl.com/b/thread-4098-1-1.h...

乐视(letv)网tkey破解
cmos1: 乐视单页 代理 http://www.austgl.com/b/forum.ph... &tid=4...



视景体有两个用途。首先，视景体定义了一个场景如何映射到屏幕上（即通过透视投影还是正投影）；其次，视景体定义了哪此场景（或场景的一部分）被剪裁到最终的图像之外。
透视投影：与我们平时眼睛所见到的物体一样，满足近大远小的原则；正投影：不论物体距离观察点的远近，我们所看到物体的大小保持不变。

glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble nFar);
创建一个表示正交平行视景体的矩阵，并把它与当前矩阵相乘。如果观察点为默认情况时，正交平行视景体是由这个函数的参数定义:(left, bottom, -zNear)和(right, top, -zNear)分别指定了近侧裁剪平面左下角和右上角的(x,y,z)坐标。zNear和zFar分别表示从观察点到近侧和远侧裁剪平面的距离，它们可以为正值或负值，甚至可以设置为0，但不能取相同的值。你可以将其想像为数码相机的视野范围（视景体）。



glOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);
创建一个表示将二维坐标投影到屏幕上的矩阵，并把它与当前矩阵相乘。剪裁区域为矩形，(left, bottom)和(right, top)分别表示剪裁区域的左下角和右上角的坐标。该函数为glOrtho的二维版本。

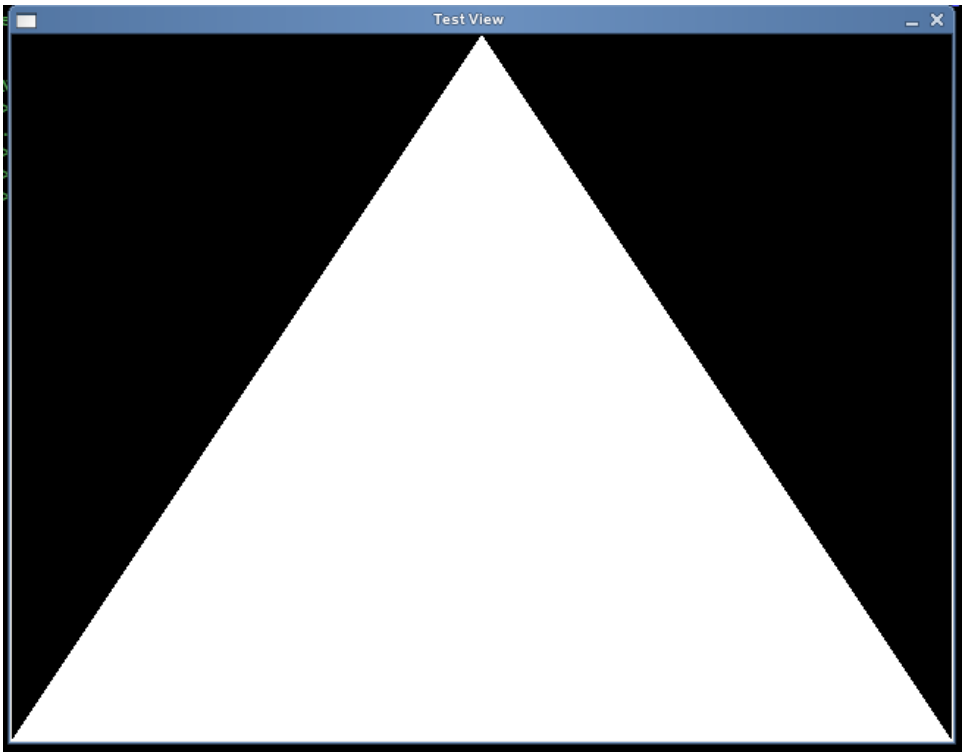
下面来看具体的opengl例子，该例子使用pyglet（a cross-platform windowing and multimedia library for Python. 官网:www.pyglet.org）编写。您也可以选择自己熟悉的语言进行改写，关键还是原理。

```python
from pyglet.gl import *
import pyglet

class MyWindow(pyglet.window.Window):
    def __init__(self, w, h, caption):
        super(MyWindow, self).__init__(w, h, caption)

    def on_resize(self, width, height):
        glViewport(0, 0, width, height)
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(45.0, float(width)/height, 0.1, 100.0)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()

    def on_draw(self):
        glClearColor(0.0, 0.0, 0.0, 0.0)
        glClear(GL_COLOR_BUFFER_BIT)

        glBegin(GL_TRIANGLES)
        glVertex3f(0.0, 0.041421356237309503, -0.1)
        glVertex3f(-0.05522847498307934, -0.041421356237309505, -0.1)
        glVertex3f(0.05522847498307934, -0.041421356237309505, -0.1)
        glEnd()
```

乐视(letv)网tkey破解
u012876441: @lingedeng:已经可用，thanks

```
25.
26. if __name__ == '__main__':
27.     myWin = MyWindow(640, 480, 'View Test')
28.
29.     pyglet.app.run()
```

示例程序中定义的窗口大小为640 * 480，使用透视投影绘制了一个底边位于窗口底部，与底边对应的顶点位于窗口顶部中央的三角形。如下图。



通过glVertex3f(...)函数我们指定要绘制三角形的顶点坐标，其z坐标指定为-0.1。根据gluPerspective(45.0, float(width)/height, 0.1, 100.0)投影函数的定义，z坐标的近剪裁面位于视点的0.1个单位距离处，所以我们将在近剪裁平面上绘制该三角形。那么x,y坐标又是如何确定的呢?

同样根据gluPerspective(...)函数，根据其对应的图示，想像一条由视点中心出发的直线向透视平截头体延伸，穿过近剪裁面的中心并到达远剪裁面的中心。该直线与近剪裁面和远剪裁面的距离分别为0.1和100。视野在yz面上角度为45度，则该直线与上剪裁面的夹角则为45.0/2。近剪裁面中心到近剪裁面顶部中心的直线，视点与近剪裁面的直线以及视点中心与近剪裁面顶部中心点的直线构成了一个直角三角形，与近剪裁面中心到近剪裁面顶部中心的直线对角的角度为45.0/2度。根据三角形正切公式:tan(r)=y/x,直角三角形某角的正切值等于其对边与邻边的比例。我们已知r = 45.0/2, x=0.1，则y = tan(r)*x可以得到近剪裁面的一半高度。

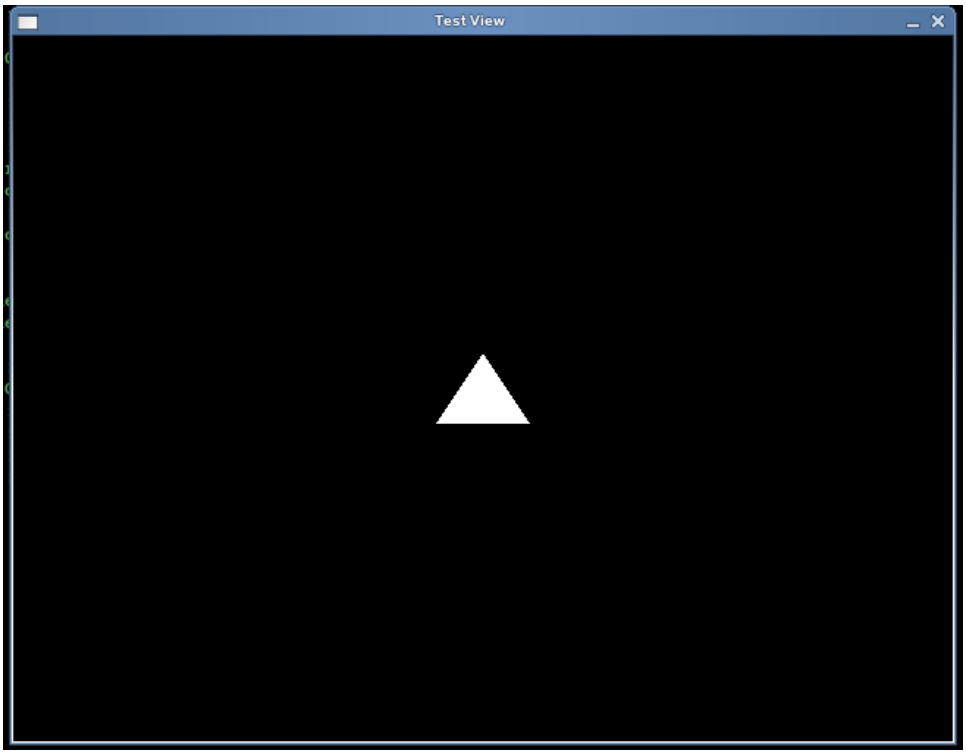
tan(...)函数以弧度作为参数，弧度／角度的转换公式：弧度 = 角度/180*PI；角度=弧度*180*PI。将已知值代入y=tan(r)*x公式，则y=tan(45.0/2/(180*PI))*0.1,得到y = 0.041421356237309503，近剪裁面y轴的坐标范围为［-0.041421356237309503， 0.041421356237309503］。

再根据gluPerspective(...)函数的第二个参数，我们窗口的大小为680 * 480,即比例为4.0/3,可以得到近剪裁面x轴的范围为[-0.05522847498307934， 0.05522847498307934]。

接下来，我们将三角形的顶点坐标的z轴设置为-1.0

```
[python]

01. glVertex3f(0.0, 0.041421356237309503, -1.0)
02. glVertex3f(-0.05522847498307934, -0.041421356237309505, -1.0)
03. glVertex3f(0.05522847498307934, -0.041421356237309505, -1.0)
```

通过设置，我们将绘制的物体向z轴负方向移动了10个单位距离（即远离了视点10个单位距离），根据透视投影的原理（近大远小），则绘制的三角形应比原来的小。再次运行该程序，结果如下图。



我们已经看到了运行结果，再来问两个问题：1.为什么改变了z轴位置，所绘制的三角形比原来小了？2.如何更改x，y坐标，使z=-1.0时绘制出与原程序相同的结果？
对于问题1，也许你做如下回答：因为设置了透视投影并且我们将要绘制的场景远离了视点。根据透视投影的原理（近大远小），所以现在绘制的三角形比原来的小。这只是字面上的解释。
其实内部的原理还是使用了直角三角形的正切公式以及窗口的横纵比。将x的值变为了1.0，代入y=tan(r)*x公式，即y=tan(45.0/2/(180*PI))*1.0，得到z＝-1.0时绘制面y坐标的范围：[－0.41421356237309503，0.41421356237309503]。再根据窗口横纵比，得到z＝-1.0时绘制面x坐标的范围：[-0.5522847498307934，0.5522847498307934]。因为在投影映射矩形大小未变的情况下，绘制图形的坐标系变为了原来的十倍，而我们所指定三角形x，y的顶点坐标并没有变化，所以绘制出的三角形只有原来的十分之一大小。对于远剪裁面的坐标我们也可以使用相同的方法进行计算，从而得到x=[－41.421356237309503, 41.421356237309503]，y=[-55.22847498307934, 55.22847498307934]。

解决了问题1，问题2也就迎刃而解。只要将x，y的顶点坐标值扩大为原来的十倍，即能得到与原程序相同的绘制结果。

```python
glVertex3f(0.0, 0.41421356237309503, -1.0)
glVertex3f(-0.5522847498307934, -0.41421356237309505, -1.0)
glVertex3f(0.5522847498307934, -0.41421356237309505, -1.0)
```

以上只是个人对于openGL视图的认识和简单理解，如有不正确的地方，望指正。

上一篇 Python——文件处理
下一篇 RFC2616 HTTP/1.1 标志转换及通用语法