个人资料

cppyin

访问： 215989次
积分： 2575
等级： BLOG 5
排名： 第9493名

原创： 38篇 转载： 2篇
译文： 0篇 评论： 283条

文章搜索

文章分类

C/C++ (6)
DirectX (2)
Game (1)
Graphics (17)
OpenGL (5)
OpenKOK (6)
Visual Studio (2)
Windows (1)
心得体会 (1)

文章存档

2011年03月 (10)
2011年02月 (18)
2011年01月 (9)
2010年12月 (3)

阅读排行

从零开始重写KOK1(万王 (20370)
从零实现3D图像引擎：((19288)
当你的程序在朋友的机器 (18139)
VC2010中"Include Direc (14838)
从零实现3D图像引擎：((10604)
从零实现3D图像引擎：((8674)
从零实现3D图像引擎：((8571)

从零实现3D图像引擎：(15)三角形的光栅化

标签： 引擎 网络游戏 工作 object 图形 算法

2011-03-08 20:15 7391人阅读 评论(6) 收藏 举报

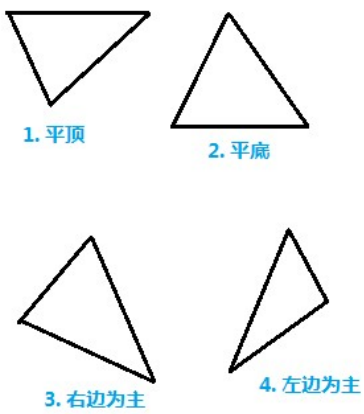
分类： Graphics（16）

1. 为什么要光栅化一个三角形

我们不能总让我们的引擎显示线框，要支持实心颜色、光照还有纹理贴图，这些都需要光栅化一个三角形作为支持。

2. 三角形的类型

为了方便光栅化，一般将三角形分为以下4种：



3. 平底三角形光栅化

先上图：

从零实现3D图像引擎：(' (7380)
从零实现3D图像引擎：(' (7184)
从零实现3D图像引擎：(' (6647)

评论排行

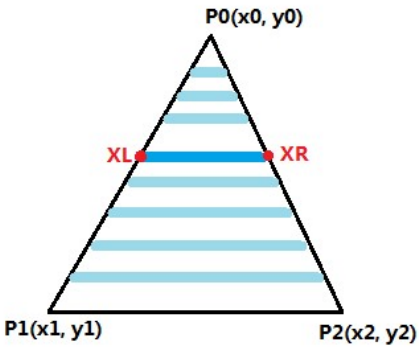
裸辞三月之痒 (55)
从零实现3D图像引擎：((29)
从零实现3D图像引擎：((25)
从零实现3D图像引擎：((24)
从零开始重写KOK1(万王 (22)
从零实现3D图像引擎：((19)
从零开始重写KOK1(万王 (12)
从零开始重写KOK1(万王 (10)
从零实现3D图像引擎：((10)
从零实现3D图像引擎：((8)

推荐文章

*Android官方开发文档Training系列课程中文版：网络操作之XML解析
*Delta - 轻量级JavaWeb框架使用文档
*Nginx正反向代理、负载均衡等功能实现配置
* 浅析ZeroMQ工作原理及其特点
*android源码解析（十九）-->Dialog加载绘制流程
*Spring Boot 实践折腾记（三）：三板斧，Spring Boot下使用Mybatis

最新评论

裸辞三月之痒
ovenxp: 楼主能在乱流中捋出一丝精华，最终摒弃杂念，投身游戏，很值得学习。我们好多人没能珍惜这个机会真是惭愧。...
从零实现3D图像引擎：(12)构建
麻子: @winmain0932:这里原点到平面上某个点的向量 就是这个点本身p - = p
从零实现3D图像引擎：(12)构建
麻子: 设视平面在各个象限的点分别为 其中w为视平面宽 h为视平面高p1 = p2 = p3 = p4 = ...
从零实现3D图像引擎：(12)构建
麻子: 计算一下：，我们可以把各分量缩放1/w，得：----------...
从零实现3D图像引擎：(12)构建
麻子: @tktbxx:是的 这本书这么多年了 也不弄个勘误
从零实现3D图像引擎：(12)构建
麻子: @winmain0932:这里的所有 不包括远近裁剪面
从零实现3D图像引擎：(12)构建
麻子: 书中确实有不少问题
从零实现3D图像引擎：(2)画2D
小抱妖妖: 太厉害了。。
从零实现3D图像引擎：(4)三角函
栗子冰激凌: 支持楼主给我们普及原理性的知识，还附代码。而不用自己去啃书。有的人就是见不得别人好，不用理他们就好了
从零实现3D图像引擎：(10)Hello
丁国华: 谢谢分享 学习了`(*∩_∩*)'



光栅化平底三角形的原理很简单，就是从上往下画横线。在图里我们取任意的一条光栅化直线，这条直线左边的端点x值为XL，右边的为XR。y值就不用考虑了，因为这些线是从上往下画的，所以y就是从y0一直++，直到y1或者y2。

所以**算法**很简单，就是每次增加一个y，就要计算一下XL和XR，然后调用我们很早以前写的那个光栅化直线的函数，来画这条线即可。

怎么求XL和XR呢？

直线有很多种形式可以表示，因为我们现在知道顶点的坐标，所以最直观的表示形式就是两点式：

对于已知直线上的两点(x1,y1)和(x2,y2)有：

(y-y1) / (y2-y1) = (x-x1) / (x2-x1)

因为y已知，我们变换一下公式，表示为x的值为：

x = (y-y1) * (x2-x1) / (y2-y1)

根据上图的，我们只要把P0,P1代入，就可以求得XL，把P0,P2代入，就可以求得XR。不多说了，直接给出实现代码：

```cpp
void _CPPYIN_3DLib::DrawTriangle1
(int x1, int y1, int x2, int y2, int x3, int y3, DWORD color) // 画实心平底三角形
{
    for (int y = y1; y <= y2; ++y)
    {
        int xs, xe;
        xs = (y - y1) * (x2 - x1) / (y2 - y1) + x1 + 0.5;
        xe = (y - y1) * (x3 - x1) / (y3 - y1) + x1 + 0.5;
        DrawLine(xs, y, xe, y, color);
    }
}
```

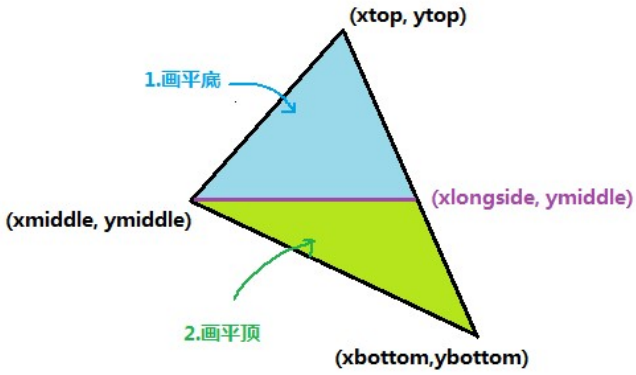
4. 平顶三角形的光栅化

不用多说了，原理同上，直接贴代码了。

```cpp
void _CPPYIN_3DLib::DrawTriangle2
(int x1, int y1, int x2, int y2, int x3, int y3, DWORD color) // 画实心平顶三角形
{
    for (int y = y1; y <= y3; ++y)
    {
        int xs, xe;
```

```
        xs = (y - y1) * (x3 - x1) / (y3 - y1) + x1 + 0.5;
        xe = (y - y2) * (x3 - x2) / (y3 - y2) + x2 + 0.5;
        DrawLine(xs, y, xe, y, color);
    }
}
```

5. 任意三角形的光栅化



其实看了这个图就应该发现特简单，我们求得一个特殊点(xlongside, ymiddle)，之后画一个平底三角形，再画一个平顶三角形就搞定了。

下面的代码实现了这个功能，有几点辅助说明一下：

1. 通过给定三个顶点的y值，可以判断出是否是平顶还是平底，如果满足其一，就直接画。

2. 传入函数的三个顶点是乱序的，所以要根据y值的情况来区分几种情况，在我的实现里，穷举了y值的几种情况，然后给上图中的三个点赋值。

3. 依次画就是了，下面是代码。

```cpp
void _CPPYIN_3DLib::DrawTriangle
(int x1, int y1, int x2, int y2, int x3, int y3, DWORD color) // 画任意实心三角形
{
    if (y1 == y2)
    {
        if (y3 <= y1) // 平底
        {
            DrawTriangle1(x3, y3, x1, y1, x2, y2, color);
        }
        else // 平顶
        {
            DrawTriangle2(x1, y1, x2, y2, x3, y3, color);
        }
    }
    else if (y1 == y3)
    {
        if (y2 <= y1) // 平底
        {
            DrawTriangle1(x2, y2, x1, y1, x3, y3, color);
        }
        else // 平顶
        {
            DrawTriangle2(x1, y1, x3, y3, x2, y2, color);
        }
    }
    else if (y2 == y3)
    {
        if (y1 <= y2) // 平底
        {
```

```
            DrawTriangle1(x1, y1, x2, y2, x3, y3, color);
        }
        else // 平顶
        {
            DrawTriangle2(x2, y2, x3, y3, x1, y1, color);
        }
    }
    else
    {
        double xtop, ytop, xmiddle, ymiddle, xbottom, ybottom;
        if (y1 < y2 && y2 < y3) // y1 y2 y3
        {
            xtop = x1;
            ytop = y1;
            xmiddle = x2;
            ymiddle = y2;
            xbottom = x3;
            ybottom = y3;
        }
        else if (y1 < y3 && y3 < y2) // y1 y3 y2
        {
            xtop = x1;
            ytop = y1;
            xmiddle = x3;
            ymiddle = y3;
            xbottom = x2;
            ybottom = y2;
        }
        else if (y2 < y1 && y1 < y3) // y2 y1 y3
        {
            xtop = x2;
            ytop = y2;
            xmiddle = x1;
            ymiddle = y1;
            xbottom = x3;
            ybottom = y3;
        }
        else if (y2 < y3 && y3 < y1) // y2 y3 y1
        {
            xtop = x2;
            ytop = y2;
            xmiddle = x3;
            ymiddle = y3;
            xbottom = x1;
            ybottom = y1;
        }
        else if (y3 < y1 && y1 < y2) // y3 y1 y2
        {
            xtop = x3;
            ytop = y3;
            xmiddle = x1;
            ymiddle = y1;
            xbottom = x2;
            ybottom = y2;
        }
        else if (y3 < y2 && y2 < y1) // y3 y2 y1
        {
            xtop = x3;
            ytop = y3;
            xmiddle = x2;
            ymiddle = y2;
            xbottom = x1;
            ybottom = y1;
        }
        int xl; // 长边在ymiddle时的x，来决定长边是在左边还是右边
        xl = (ymiddle - ytop) * (xbottom - xtop) / (ybottom - ytop) + xtop + 0.5;

        if (xl <= xmiddle) // 左三角形
        {
            // 画平底
            DrawTriangle1(xtop, ytop, xl, ymiddle, xmiddle, ymiddle, color);

            // 画平顶
            DrawTriangle2(xl, ymiddle, xmiddle, ymiddle, xbottom, ybottom, color);
        }
        else // 右三角形
        {
            // 画平底
            DrawTriangle1(xtop, ytop, xmiddle, ymiddle, xl, ymiddle, color);
```

```
            // 画平顶
            DrawTriangle2(xmiddle, ymiddle, xl, ymiddle, xbottom, ybottom, color);
        }
    }
}
```

6. 实现物体的实心渲染

之前我们只有一种渲染模式就是线框，有了上面的三角形光栅化函数，我们就可以做实心渲染了，下面是实现渲染的代码，非常简单。

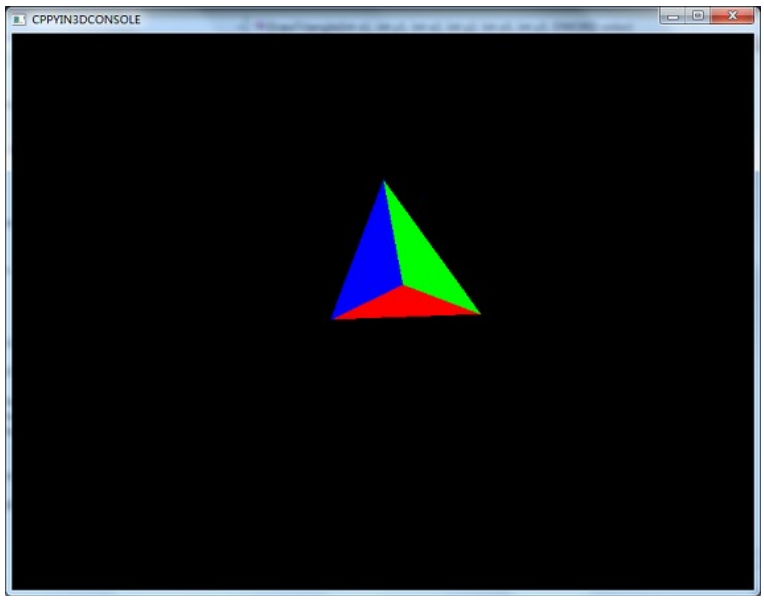
就是遍历物体的每个三角面，然后调用上面的函数而已，嘿嘿。

```cpp
void _CPPYIN_3DLib::ObjectDrawSolid(OBJECT_PTR obj) // 绘制物体实心多边形
{
    for (int i = 0; i < obj->PolyCount; ++i)
    {
        POINT4D_PTR p0 = &(obj->VertexListTrans[obj->PolyList[i].VertexIndexs[0]]);
        POINT4D_PTR p1 = &(obj->VertexListTrans[obj->PolyList[i].VertexIndexs[1]]);
        POINT4D_PTR p2 = &(obj->VertexListTrans[obj->PolyList[i].VertexIndexs[2]]);

        // 只画正面
        if (obj->PolyList[i].State == POLY_STATE_ACTIVE)
        {
            DrawTriangle(p0->x, p0->y, p1->x, p1->y, p2->x, p2->y, obj->PolyList[i].Color);
        }
    }
}
```

7. 截图

这个就是实心渲染的Demo截图了。



8. 代码下载

完整项目源代码：>>点击进入下载页<<

9. 特殊说明

前一阵为了找工作补了补C++，最近开始工作了，做一个网络游戏的客户端，所以没怎么更新这个图形库，最近有空继续更新吧。发现对C++还是不熟练，可能更多的会更新一些C++的东西了。

上一篇 Effective C++ 学习笔记(1) : 语言联邦、弱化预编译器、const、初始化
下一篇 Learning OpenGL : (1) Concept of 3D

我的同类文章

Graphics（16）

- 从零实现3D图像引擎：(14)… 2011-02-25 阅读 5444
- 从零实现3D图像引擎：(13)… 2011-02-23 阅读 4444
- 从零实现3D图像引擎：(12)… 2011-02-21 阅读 7188
- 从零实现3D图像引擎：(11)… 2011-02-18 阅读 10611
- 从零实现3D图像引擎：(10)… 2011-02-17 阅读 4099
- 从零实现3D图像引擎：(9)四… 2011-02-10 阅读 6362
- 从零实现3D图像引擎：(8)参… 2011-02-10 阅读 4051
- 从零实现3D图像引擎：(7)矩… 2011-02-09 阅读 3445
- 从零实现3D图像引擎：(6)向… 2011-02-07 阅读 5562
- 从零实现3D图像引擎：(5)3… 2011-02-07 阅读 5073

更多文章

参考知识库

算法与数据结构知识库

3549 关注 | 4535 收录

猜你在找

使用Cocos2d-x 开发3D游戏
C++语言基础
数据结构和算法
虚幻4开发快速入门
从此不求人:自主研发一套PHP前端开发框架
计算机图形学三角形基元填充算法即三角形光栅化重心
《计算机图形学》实验一利用OpenGL实现直线光栅化的
从零实现3D图像引擎7矩阵函数库
从零实现3D图像引擎9四元数函数库
从零实现3D图像引擎1环境配置与项目框架

查看评论

6楼 dr_c_gao 2013-07-09 02:12发表

将三角形分为4类是很多三角形光栅化算法的普遍处理,其实我觉得分为3类效率更高(比如这里:http://blog.sina.com.cn/s/blog_6f38945b01014fw3.html). 楼主的代码在计算每条扫描线的时候可以考虑不要在循环里用乘除法, 效率太低, 应直接用倒斜率和加法来做线性插值. 期待你的后续博文.

5楼 jyf9822 2013-05-15 11:30发表

坚持更新下去，我第一次这么认真的拜读别人的文章。

4楼 yuzhantao01 2012-08-02 10:13发表

我自己做光栅化算法的时候，发现渲染完的有时三角形会出现色彩不正常，色彩效果类似于凑近显像管电视机时看到的电视机屏幕的三色小矩形的组合。求指点！

3楼 huaiblueboy 2011-12-12 16:37发表

楼主怎么不更新了啊

2楼 starlizhi 2011-05-18 11:28发表

速度不快呀。最多1000个三角形，有更好的方法吗？

1楼 DelphiBoy2003 2011-03-20 12:00发表

这个系列非常精彩，希望楼主坚持更新下去啊

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC
coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap