相关主题 RECOMMEND ARTICLE
- HDR渲染器的实现(基于OpenGL)
- Vertex Cache (OpenGL实现)
- Slope(斜坡) 法线生成算法，在地形渲染中的应用
- 简单的运动模糊效果实现
- 具有真实感的3D图形(下)
- 具有真实感的3D图形(上)
- 多动画集在D3D下的渲染
- Portal系统

MORE

推荐文章 RECOMMEND ARTICLE
- 数据广播方案的优化
- 网络游戏的位置同步
- 游戏音乐制作案例之《战火 红色警戒》音效制作揭秘
- 英雄连Online 原画
- 游戏音乐制作案例之《乱武天下》
- 游戏音乐制作案例之《诛仙》
- 《鹿鼎记》最新原画
- MIDP2.1规范的新特性

MORE

热门文章 HOT ARTICLE
- [电子书下载]游戏设计 — 原理与实践
- [电子书下载]网络游戏开发
- 游戏设计全过程
- [电子书下载]游戏设计技术
- [电子书下载]游戏设计理论
- CS游戏人物模型制作教程
- CG人物插画基本流程
- [转贴]MAX高级人头教程

MORE

您的位置: 图形技术

文章标题	地形教程 - 法线的计算		
来源:	[Azure]	浏览:	[776]

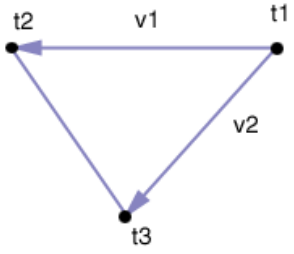
Terrain Tutorial
地形教程

Computing Normals
法线的计算

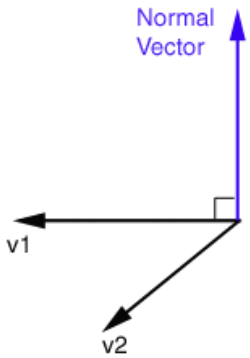
To apply lighting to a terrain, either using OpenGL lights, or simulating them, it is necessary to first compute normals. A normal is a vector that defines how a surface responds to lighting, i.e. how it is lit. The amount of light reflected by a surface is proportional to the angle between the lights direction and the normal. The smaller the angle the brighter the surface will look.
为了给地形进行光照,任意一个OpenGL的光照,或者模拟它们，那么我们必须要首先计算法线．法线是一个向量定义了表面对光照的响应.例如,如何去照亮它.表面的反射光量是与光线方向与法线方向的夹角成正比．夹角越小表面就会看起来越亮．

Normals in OpenGL can be defined per face or per vertex. If defining a normal per face then the normal is commonly defined as a vector which is perpendicular to the surface. In order to find a perpendicular vector to a face, two vectors coplanar with the face are needed. Afterwards the cross product will provide the normal vector, i.e. a perpendicular vector to the face.
OpenGL里面可以定义面法线和顶点法线．如果定义了面法线,那么这个法线向量一般都要正交于这个表面．为了去找到面的正交向量，需要两个共面向量．然后它们叉乘就会产生法向量，一个正交于面的向量．

So the first step is to compute two vectors coplanar to a face. Assuming the the faces are triangles defined by points t1,t2,t3, then two possible vectors are:
所以首先来计算面的两个共面的向量.假设面是三角形用t1,t2,t3定义，然后两个可能的向量是:
v1 = t2 - t1
v2 = t3 - t1



With the two vectors, v1 and v2, it is now possible to compute the cross product between them to find a perpendicular vector to the face.
有了两个向量v1和v2，现在我们就可以计算叉乘,来找到正交于面的一个向量.



The equations bellow show the necessary steps to compute a normal vector v. The required opeartion is called cross product, and it is represented by "x".
下面的等式展示了计算法线v的必要的一些步骤. 这个需要的运算叫做叉乘, 我们把它表示为"X"
v = v1 x v2

v = [vx,vy,vz] where,

vx = v1y * v2z - v1z * v2y

vy = v1z * v2x - v1x * v2z

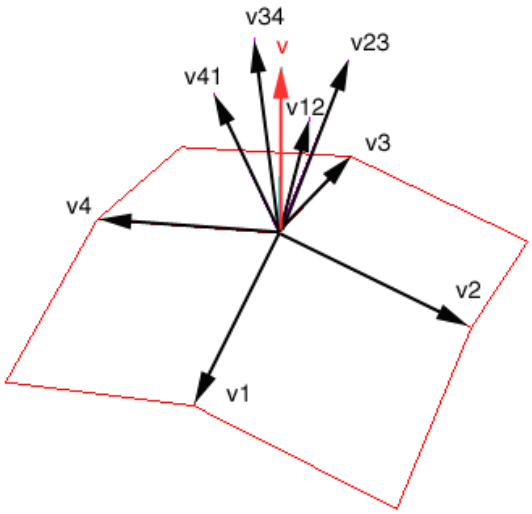
vz = v1x * v2y - v1y * v2x
Another necessary step to obtain proper lighting is to normalise the vector, i.e. make it unit length. OpenGL takes into consideration the length of the normal vector when computing lighting. Normalisation implies first computing the lenght of the vector, and then dividing each component by the vectors length.
另一个必要的步骤获得正确的光照是归一化这个向量, 就是让它是单位长度. OpenGL在计算光照的时候需要考虑归一化的法向量.归一化的过程意味着第一步首先计算向量的长度,然后向量的每个部分除以此长度.

The length of a vector is computed as:
向量的长度是以下公式计算来:

$$l = \sqrt{vx * vx + vy * vy + vz * vz}$$

Therefore the normalized vector nv is computed as:
因此,归一化的向量nv计算过程如下:
nv = [nvx,nvy,nvz] where,

nvx = vx / l
nvy = vy / l
nvz = vz / l
The main problem with assigning a normal per face is that the terrain looks faceted, i.e. the brightness of each face is constant, and there is a clear difference between faces with differen orientations. In order to get a smoother look normals should be computed per vertex, and not per face. When computing normals per vertex it is necessary to take into account the faces that share the vertex. So for instance if using quads, each vertex (excluding the corner and border vertices), is shared by four polygons. The normal at a vertex should be computed as the normalised sum of all the unit length normals for each face the vertex shares. Consider the following image:
为地形使用面法线最大的问题就是它看起来一块一块的， 因为每个面的亮度都是恒定的，并且每个方向的面上的亮度都有明显的区别。为了看起来更加的光滑，我们必须为每个顶点计算法线，而不是每个面计算法线。当计算顶点法线的时候，我们有必要考虑到此顶点共享的所有面，所以我们用方形举例，每个顶点（排除掉角落和边缘的顶点）被四个多边形共享。这个顶点的法线就应该是所有共享面法线的和在归一化的结果。看看下面的图：



In the above image, v represents the normal at the center vertex. Each vij represents a normal for each face that shares the center vertex. So for instance v12 is the unit lenght normal for the bottom right face.
在上面的图里，v代表了中心顶点的法线。每个vij表示每个共享面的法线。所以例如v12是右下方面的单位长度的法线。

The vertex normal v is computed as the normalised sum of all vij vectors:
顶点法线v被计算作为所有vij向量归一化后的和。
v = normalised(sum(v12, v23, v34, v41))

where

vij = normalised(vi x vj) // normalised cross product
It is also posible to consider the eight neighbour vertices, instead of only four. This latter option will probably look smoother in the general case.
也可以考虑8个相临的顶点。这个是后来的选项比一般的做法看起来更加光滑。

Note that when computing the normals a scale is assumed. If the application has performed non-uniform scaling the normals will no longer be correct. If scaling the heights is required use the function terrainScale provided in the terrain library. This function recomputes the normals. If the grid needs scaling then use the function terrainDim to enlarge the terrain
注意当计算法线时假设了一个缩放。如果程序使用了一个非归整的缩放，那么法线将不再正确。如果缩放了高度，那么就需要使用函数terrainScale来调整。这个函数重新计算了法线。如果网格需要缩放，那么就使用terrainDim函数来扩大地形。

无任何评论！

请您注意：

- 尊重网上道德，遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规
- 尊重网上道德，遵守中华人民共和国的各项有关法律法规
- 承担一切因您的行为而直接或间接导致的民事或刑事法律责任
- 中国网游研发中心新闻留言板管理人员有权保留或删除其管辖留言中的任意内容
- 您在中国网游研发中心留言板发表的作品，中国网游研发中心有权在网站内转载或引用
- 参与本留言即表明您已经阅读并接受上述条款

发表评论：

昵　　称：

GO

联系EMAIL：