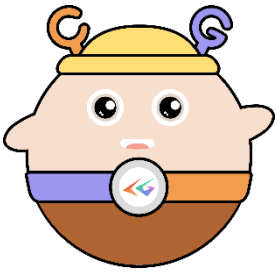
華中科技大學



计算机图形学课程

实验：几何纹理（法线贴图）

目录

切线空间和法线贴图

在正式搭建环境之前，我们先来介绍一下读完下面的部分你会了解些什么。

- 了解什么是法线贴图
- 为什么需要切线空间
- 如何加载和使用法线贴图
- 如何引入切线空间并在着色器中使用

接下来，我们来介绍一下绘制具有法线贴图的立方体的效果。绘制效果如下（左图为无切线空间的法线贴图，右图为有切线空间的法线贴图）：

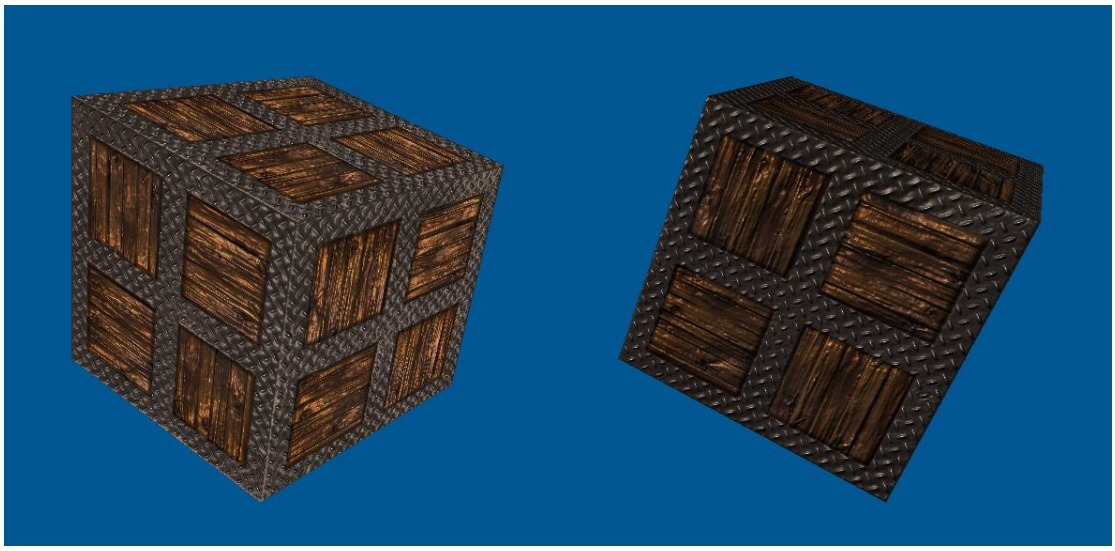


图 1 绘制具有法线贴图的立方体的效果

1 什么是法线贴图

在我们进行各种图形的绘制时，为了尽量提高真实感，会采用纹理贴图的方法，但是例如我们在一个正方形上贴砖墙的纹理，现实中的砖墙是凹凸不平的，而且我们绘制的正方形却是一个平面，在加入光照模型时，也不能通过光照，反应砖墙的细节，这时我们引入法线贴图的办法去赋予纹理每个点相应的法线信息，去形成不同的光照效果，如图，左图为无法线贴图的情况，右图为引入法线贴图的情况。

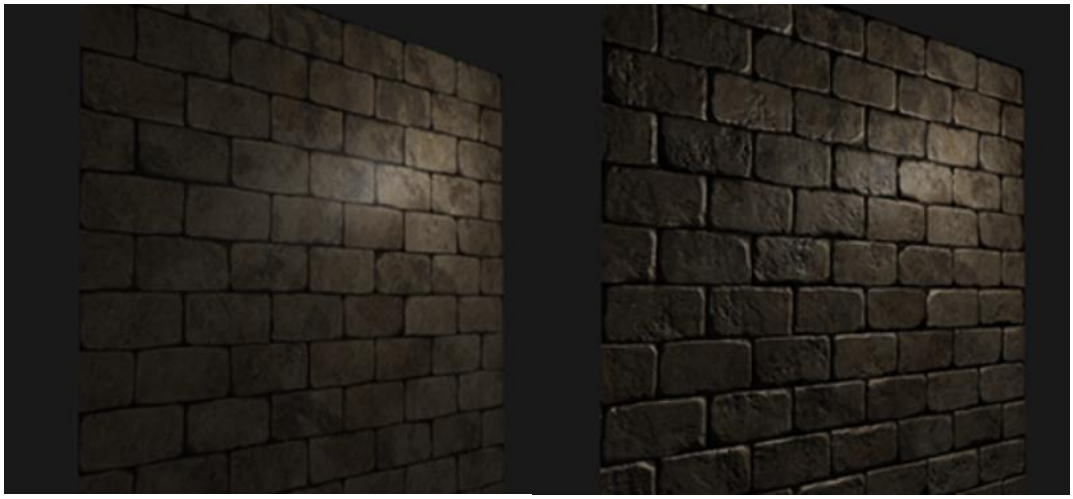


图 2 是否具有法线贴图的效果

法线贴图相当于针对原纹理贴图的每个像素点添加的其独特的法线细节。所有的法线贴图都是对应局部坐标的法线，他们是一种偏蓝色调的纹理（你在网上找到的几乎所有法线贴图都是这样的）。这是因为所有法线的指向都偏向 z 轴（0, 0, 1）这是一种偏蓝的颜色。法线向量从 z 轴方向也向其他方向轻微偏移，颜色也就发生了轻微变化，这样看起来便有了一种深度。例如，你可以看到在每个砖块的顶部，颜色倾向于偏绿，这是因为砖块的顶部的法线偏向于指向正 y 轴方向（0, 1, 0），这样它就是绿色的了。

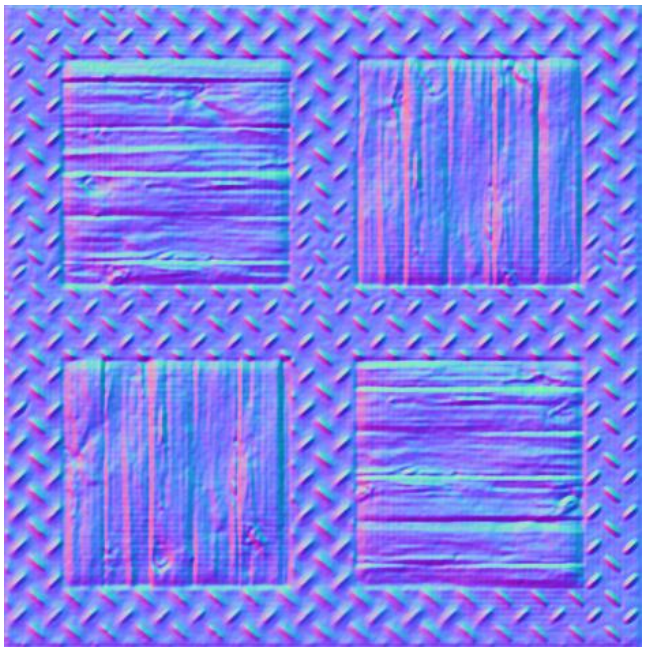


图 3 法线贴图

2 为什么需要切线空间

使用 Heightmap 可以为物体表面增加法线细节，但是从 heightmap 提取的法线是局部坐标系内的法线，正常的使用需要我们建立一个切线空间去将 heightmap 中提取出的法线转换到世界坐标系中，之前的效果图我们可以看到，左侧为不引入切线空间的法线贴图，它的每个面的法线都是一样的，所以显得亮度相同，而右侧是引入了切线空间的法线贴图，它每个面的法线朝向都不相同。

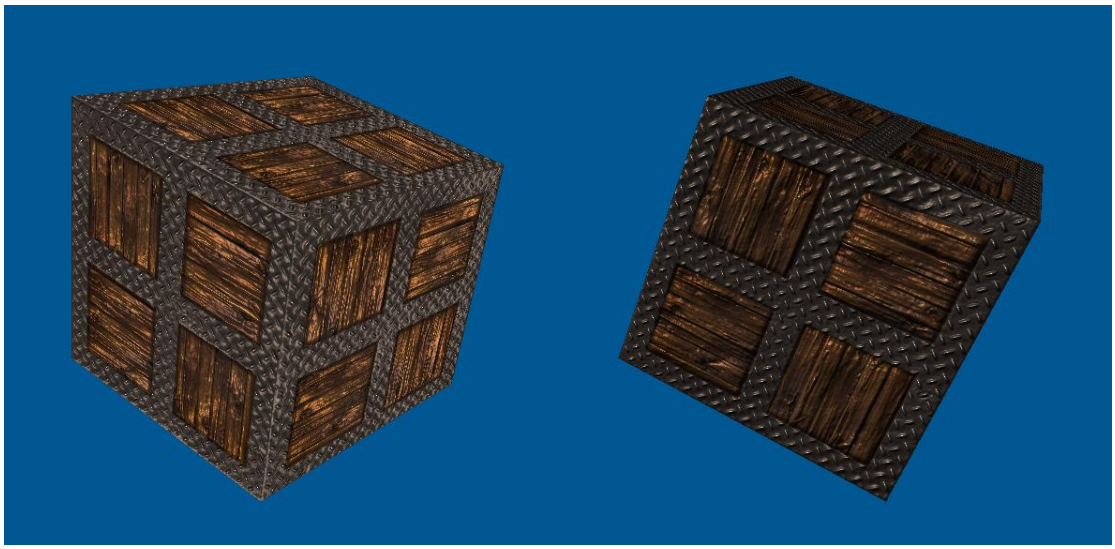


图 4 是否具有切线空间的效果

3 加载法线贴图

加载法线贴图和加载纹理的方式可以说是一模一样的，在此不再赘述。

```
Texture cube_normal;
unsigned int cube_normal_texture=
cube_normal.LoadTextureFromFile("res/texture/cube_normal.jpg");//加载法线贴图
```

法线贴图是把法线数据储存在纹理图片中，所以我们需要根据法线贴图和纹理坐标去采样获得一点的法线贴图的颜色值，将其 rgb 颜色值，作为法线的 xyz 轴，读取成法线向量，但是颜色的范围是在 0~1 之间，而法线参数是在-1~1

之间，所以我们需要进行范围的转化，先乘以 2 扩大范围到 0~2，再减去 1，使法线范围变化到-1~1，，最后一般我们使用的法线都应该是单位向量，所以我们需要进行标准化操作。

```
vec3 normal = texture(texture_normal, fs_in.TexCoords).rgb;
normal = normalize(normal * 2.0f - 1.0f);
```

然后我们使用从法线贴图中取样获得的法线来计算光照即可。

4 引入切线空间

法线贴图中的法线向量在切线空间中，法线永远指着正 z 方向。如果模型上有无数的朝向不同方向的表面，这就不可行了。所以一种解决问题的方式是计算出一种矩阵，把法线从局部空间变换到一个世界空间，这样它们就能和表面法线方向对齐了。

这种矩阵叫做 TBN 矩阵这三个字母分别代表 tangent、bitangent 和 normal 向量。这是建构这个矩阵所需的向量。要建构这样一个把切线空间转变为不同空间的变异矩阵，我们需要三个相互垂直的向量，它们沿一个表面的法线贴图对齐于：上、右、前。

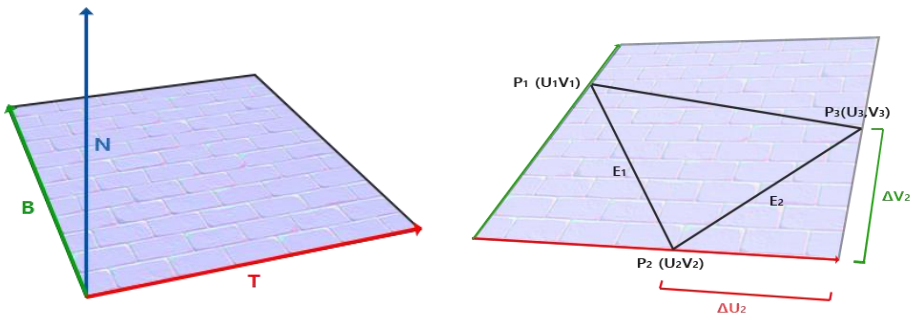


图 4 切线空间原理图

为了获得 TBN 矩阵，需要进行以下步骤

首先，我们可以把边 E1 和 E2 用切线向量 T 和副切线向量 B 的线性组合表示出来

E 是两个向量位置的差，Δ U 和Δ V 是纹理坐标的差。

图中我们可以看到边 E2 纹理坐标的不同，E2 是一个三角形的边，这个三角形的另外两条边是Δ U2 和Δ V2，它们与切线向量 T 和副切线向量 B 方向相同。这样我们可以把边 E1 和 E2 用切线向量 T 和副切线向量 B 的线性组合表示出来（注意 T 和 B 都是单位长度，在 TB 平面中所有点的 T、B 坐标都在 0 到 1 之间，因此可以进行这样的组合）：

$$E_1 = \Delta U_1 T + \Delta V_1 B$$
$$E_2 = \Delta U_2 T + \Delta V_2 B$$

然后也可以写成这样：

$$(E_{1x}, E_{1y}, E_{1z}) = \Delta U_1 (T_x, T_y, T_z) + \Delta V_1 (B_x, B_y, B_z)$$
$$(E_{2x}, E_{2y}, E_{2z}) = \Delta U_2 (T_x, T_y, T_z) + \Delta V_2 (B_x, B_y, B_z)$$

上面的方程允许我们把它们写成另一种格式：矩阵乘法

$$\begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix} = \begin{bmatrix} \Delta U_1 & \Delta V_1 \\ \Delta U_2 & \Delta V_2 \end{bmatrix} \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

两边都乘以Δ UΔ V 的逆矩阵等于：

$$\begin{bmatrix} \Delta U_1 & \Delta V_1 \\ \Delta U_2 & \Delta V_2 \end{bmatrix}^{-1} \begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

最后，用 1 除以逆矩阵的行列式，再乘以它的共轭矩阵

$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \frac{1}{\Delta U_1 \Delta V_2 - \Delta U_2 \Delta V_1} \begin{bmatrix} \Delta V_2 & -\Delta V_1 \\ -\Delta U_2 & \Delta U_1 \end{bmatrix} \begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix}$$

由此，我们可以手工算出 TBN 矩阵，当然，也可以通过编写算法去计算相应的 TBN 矩阵

以下为 TBN 矩阵的计算算法（其中 pos1，pos2，pos3 为世界空间内坐标，uv1，uv2，uv3 为纹理坐标），由于三个向量两两正交，则可以计算出第三个向量：

```
//我们先计算三角形的边和deltaUV 坐标:
glm::vec3 edge1 = pos2 - pos1;  //即计算公式中的E1
glm::vec3 edge2 = pos3 - pos1;  //即计算公式中的E2
glm::vec2 deltaUV1 = uv2 - uv1; //即计算公式中的U1 V1
glm::vec2 deltaUV2 = uv3 - uv1; //即计算公式中的U2 V2
//有了计算切线和副切线的必备数据，计算:
GLfloat f = 1.0f / (deltaUV1.x * deltaUV2.y - deltaUV2.x * deltaUV1.y);
tangent1.x = f * (deltaUV2.y * edge1.x - deltaUV1.y * edge2.x);
tangent1.y = f * (deltaUV2.y * edge1.y - deltaUV1.y * edge2.y);
tangent1.z = f * (deltaUV2.y * edge1.z - deltaUV1.y * edge2.z);
tangent1 = glm::normalize(tangent1);  //把结果转换成单位矩阵
bitangent1.x = f * (-deltaUV2.x * edge1.x + deltaUV1.x * edge2.x);
bitangent1.y = f * (-deltaUV2.x * edge1.y + deltaUV1.x * edge2.y);
bitangent1.z = f * (-deltaUV2.x * edge1.z + deltaUV1.x * edge2.z);
bitangent1 = glm::normalize(bitangent1); //把结果转换成单位矩阵
```

切线空间的 TN 向量我们从顶点数组中传入，但是，需要注意的是传入的 T N 向量是局部坐标系的 TN 向量，我们需要乘以 Model 将其转换至世界坐标系，model 矩阵是 mat4 所以我们需要先将 T 向量变化为 4 维向量，乘完后再变回三维向量，最后依旧不要忘记标准化操作，最后得到了我们需要的 TN 向量：

```
vec3 T = normalize(vec3(model * vec4(aTangent, 0.0f)));
vec3 N = normalize(vec3(model * vec4(aNormal, 0.0f)));
vec3 B = normalize(cross(T, N));
```

接下来，只需要在从法线贴图读取了法线数据并进行范围转化之后，将 TBN 矩阵与法线相乘，即可引入切线空间，将法线变换到世界空间中。

```
// 从法线贴图范围[0,1]获取法线
vec3 normal = texture(texture_normal, fs_in.TexCoords).rgb;
// 将法线向量转换为范围[-1,1]
normal = normalize(normal * 2.0f - 1.0f);
//引入切线到世界空间变换
```

```
normal = normalize(fs_in.TBN * normal);
```