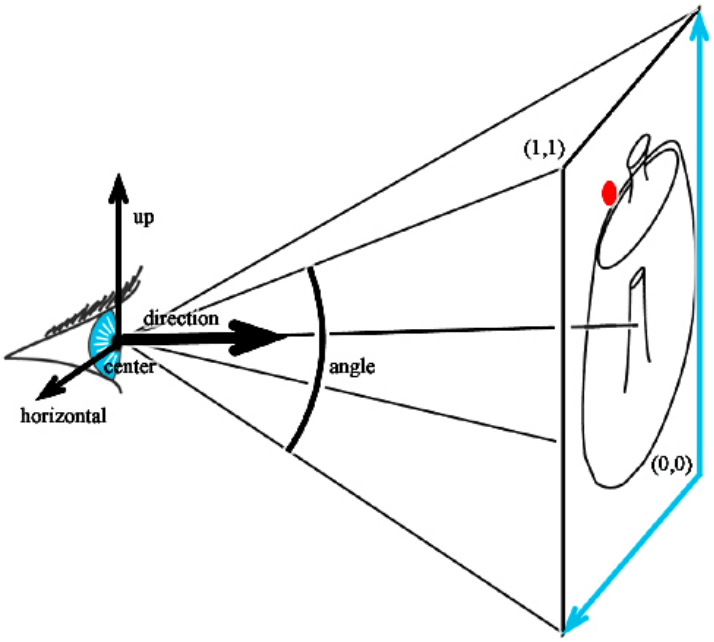
光线跟踪 2

● 简述

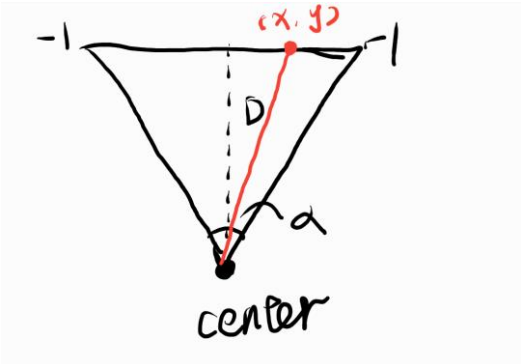
本次作业的主要内容是在前一次作业的基础之上实现透视相机，两种简单的渲染模式: 法向量可视化和散射 diffuse 光照，新的图元（平面和三角形），并且能够正确的渲染经过仿射变换的图元。

● 透视相机的构建

透视相机与前面正交相机不同在于它是从一个光源向周围放射出光线，而不是从一块画布上垂直向前方射出平行光，如下图所示：



从说明文档上我们可以知道透视相机在构建的时候会传入光源点（center），相机方向（direction），正上方向（up）和可视角度（angle），水平方向的向量（horizontal）可以用和正交相机同样的方法生成。由于可视角度已经限定了光线的发射范围，所以我们也需要像前面生成正交相机那样对射出点坐标（的横纵坐标的范围都是从-1 到 1 的）进行变换。首先在视锥上找一个 2*2 大小平面，然后计算从光源到射出点坐标的方向向量。可以看下面这张图：



首先光源点到平面的距离：

$$\tan\frac{\alpha}{2}=\frac{1}{D}$$

$$D=\frac{1}{\tan\frac{\alpha}{2}}$$

然后将输入点（上图中的红点）的横纵坐标转换为世界坐标值，这样根据上面图示的关系就能得到从光源到输入点光线（Ray）的方向向量：

$$\overrightarrow{direction}'=\overrightarrow{center}+\overrightarrow{horizontal}*x+\overrightarrow{up}*y+\overrightarrow{direction}*D$$

● 平面类的构建

平面类在构造的时候会传入平面的法向量和平面到原点的距离（带方向的），在处理前我需要对法向量进行单位化处理。然后我们根据下面方程组来求解光线和平面的交点：

$$\begin{cases}R(t)=\overrightarrow{origin}+t*\overrightarrow{direction}\\ \vec{p}\cdot\vec{n}=d\end{cases}$$

联立方程组，解得：

$$t=\frac{d-\vec{n}\cdot\overrightarrow{origin}}{\vec{n}\cdot\overrightarrow{direction}}$$

解出上述方程得到 t，只要 t 值大于 tmin 并且小于当前 Hit 中的 t 值，即是否是当前距离镜头最近的交点，就更新 Hit 对象。

● 三角形类的构建

三角形类在构造时会传入三角形三个顶点的坐标 a，b，c，在计算交点的过程中我们取三角形的 ca，cb 两条边为基向量 e1，e2：

$$\overrightarrow{e1}=\vec{a}-\vec{c}$$

$$\overrightarrow{e2}=\vec{b}-\vec{c}$$

已知三角形的重心坐标公式：

$$\begin{cases}\alpha\vec{a}+\beta\vec{b}+\gamma\vec{c}=\overrightarrow{point}\\ \alpha+\beta+\gamma=1\end{cases}$$

对上面的公式做变换有：

$$\alpha\vec{a}+\beta\vec{b}+(1-\alpha-\beta)\vec{c}=\overrightarrow{point}$$

$$\alpha(\vec{a}-\vec{c})+\beta(\vec{b}-\vec{c})+\vec{c}=\overrightarrow{point}$$

将光线的方程带入上式，得：

$$\alpha(\vec{a}-\vec{c})+\beta(\vec{b}-\vec{c})+\vec{c}=\overrightarrow{origin}+t*\overrightarrow{direction}$$

化简得：

$$\alpha * \overrightarrow{e1} + \beta * \overrightarrow{e2} + \vec{c} = \overrightarrow{origin} + t * \overrightarrow{direction}$$

利用 Cramer 法则可解上面的三元一次方程组：

$$\begin{cases} t = \dfrac{dett}{det} \\ \alpha = \dfrac{det\alpha}{det} \\ \beta = \dfrac{det\beta}{det} \end{cases}$$

$$det = -\overrightarrow{direction} \cdot (\overrightarrow{e1} \times \overrightarrow{e2})$$

$$dett = (\overrightarrow{origin} - \vec{c}) \cdot (\overrightarrow{e1} \times \overrightarrow{e2})$$

$$det\alpha = -\overrightarrow{direction} \cdot ((\overrightarrow{origin} - \vec{c}) \times \overrightarrow{e2})$$

$$det\beta = -\overrightarrow{direction} \cdot (\overrightarrow{e1} \times (\overrightarrow{origin} - \vec{c}))$$

得到α和β后需要进行验算，看交点是否在三角形的内部，如果在则继续检查 t 值，只要 t 值大于 tmin 并且小于当前 Hit 中的 t 值，即是否是当前距离镜头最近的交点，就更新 Hit 对象。

● 仿射变换类的构建

在进行变换时我们的第一感觉总是对图形本身进行变换，但是这里我们用到的方法是对光线进行图形的逆变换而图形本身不发生任何变换，这么做的原因在于变换后的图形用方程描述往往比较复杂，而基本图形往往比较简单；同时基本图形往往都是相似或者相同的，这样能够节省内存提高速度。设变换矩阵为 M,那么变换后的光线为：

$$R(t)' = M^{-1} * \overrightarrow{origin} + t * M^{-1} * \overrightarrow{direction}$$

变换后我们就会的经过变换后光线与物体的交点和法向量，但是这个法向量并不是变换后图形的法向量，因为我们调用的是基本图元的求交函数，得到的是基本图形交点处的法向量，所以我们同样需要对其进行变换，首先取任意点的切线 v,图元变换后这条切线也会发生相同的变换：

$$\vec{v}' = M\vec{v}$$

对于法向量 n 有：

$$\vec{n}^T\vec{v} = 0$$

对上式做如下变换：

$$\vec{n}^T(M^{-1}M)\vec{v} = 0$$

$$(\vec{n}^T M^{-1})(M\vec{v}) = 0$$

$$(\vec{n}^T M^{-1})\vec{v}' = 0$$

$$\vec{n}^T M^{-1} = \vec{n}'^T$$

$$(M^{-1})^T\vec{n} = \vec{n}'$$

这样只要对得到的法向量按照上式进行变换即可，最后只要将变换后的法向量更新到 Hit 中即可。

● 图像生成

在之前生成图像的时候，我们都是直接填充材质的颜色的，现在为了产生更为真实的效果，我们加入了周围自然光和其它光源的影响，所以在计算物体颜色的时候需要考虑到物体本身材质、周围自然光、其它光源和球体法向量，计算公式如下：

$$\overrightarrow{c_{pixel}} = \overrightarrow{c_{ambient}} * \overrightarrow{c_{object}} + \sum(d * \overrightarrow{c_{light_i}} * \overrightarrow{c_{object}})$$

$$d = \vec{l_i} \cdot \vec{n}$$

其中需要注意的是，这里的向量乘法是对应分量的值相乘，如下式所示：

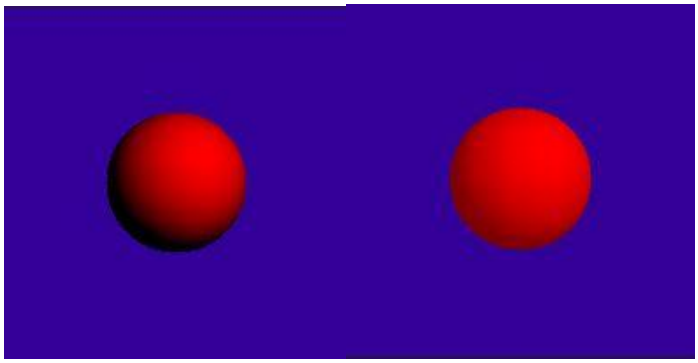
$$\overrightarrow{p_1} * \overrightarrow{p_2} = (x_1 * x_2, y_1 * y_2, z_1 * z_2)$$

这样生成的图像就具有了一定的立体感。

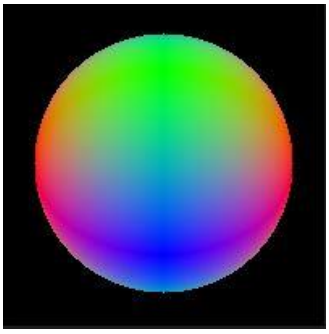
在进行法向量可视化的时候，只需要保证法向量是单位向量并且每一个分量都大于等于 0 相遇等于 1，在处理负值的时候只需要取其绝对值就可以了。

● 实验结果

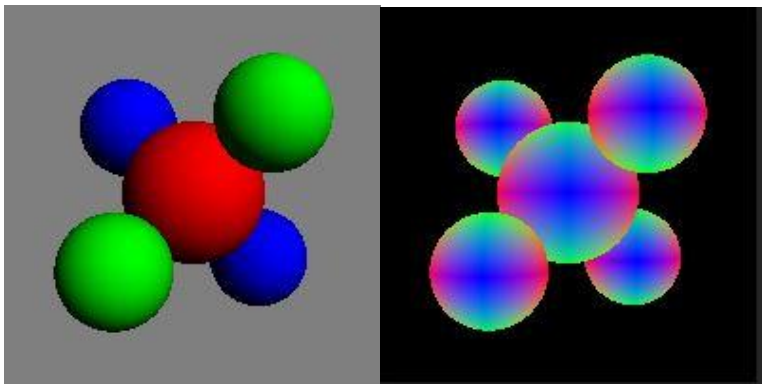
-input scene2_01.txt -size 200 200 -output output2_01.tga
-input scene2_02.txt -size 200 200 -output output2_02.tga



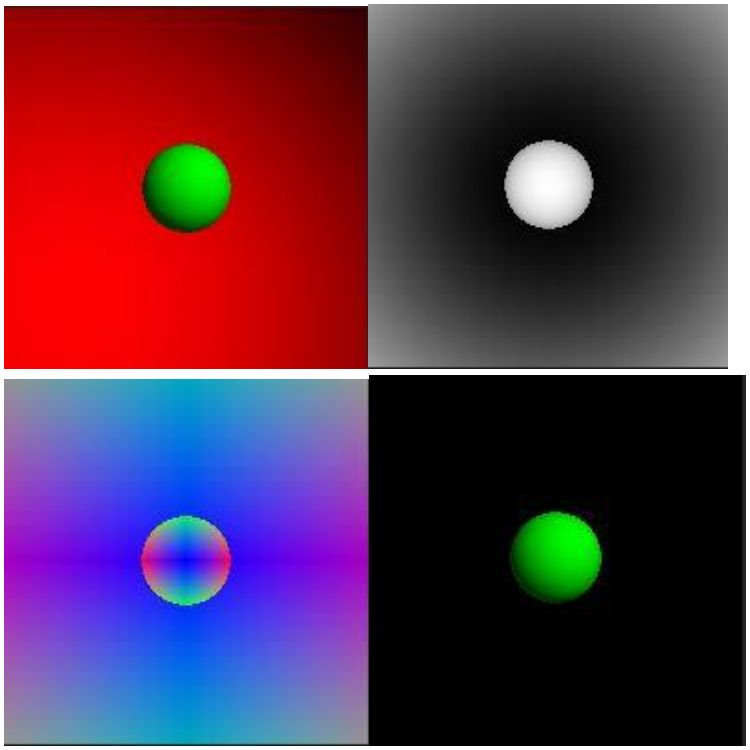
-input scene2_03.txt -size 200 200 -output output2_03.tga -normals normals2_03.tga



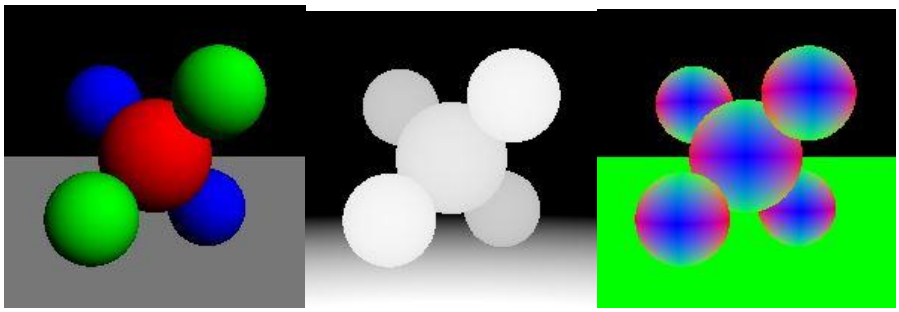
-input scene2_04.txt -size 200 200 -output output2_04.tga -normals normals2_04.tga



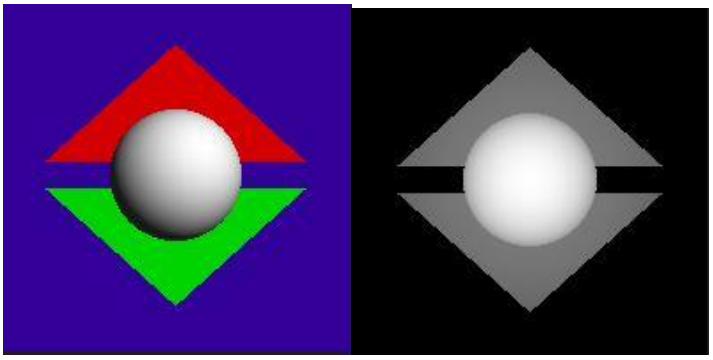
-input scene2_05.txt -size 200 200 -output output2_05.tga -depth 9 11 depth2_05.tga -normals normals2_05.tga -shade_back
-input scene2_05.txt -size 200 200 -output output2_05_no_back.tga

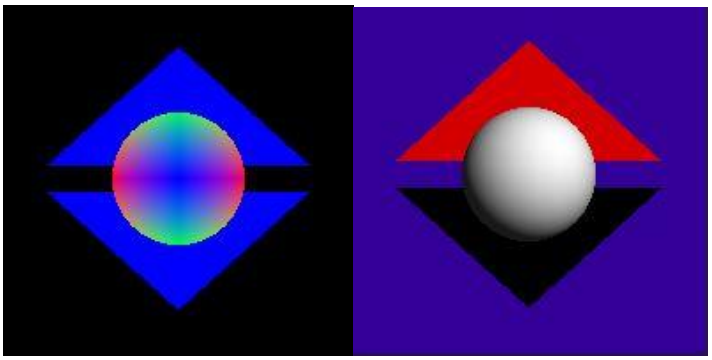


-input scene2_06.txt -size 200 200 -output output2_06.tga -depth 8 20 depth2_06.tga -normals normals2_06.tga

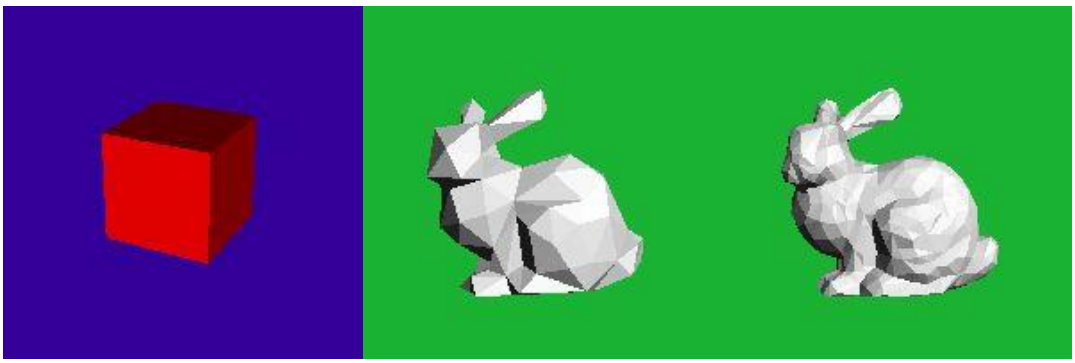


-input scene2_07.txt -size 200 200 -output output2_07.tga -depth 9 11 depth2_07.tga -normals normals2_07.tga -shade_back
-input scene2_07.txt -size 200 200 -output output2_07_no_back.tga

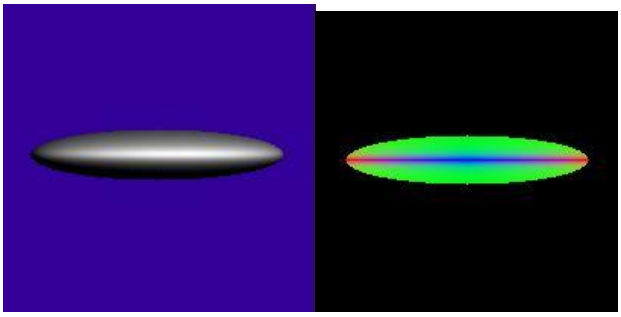




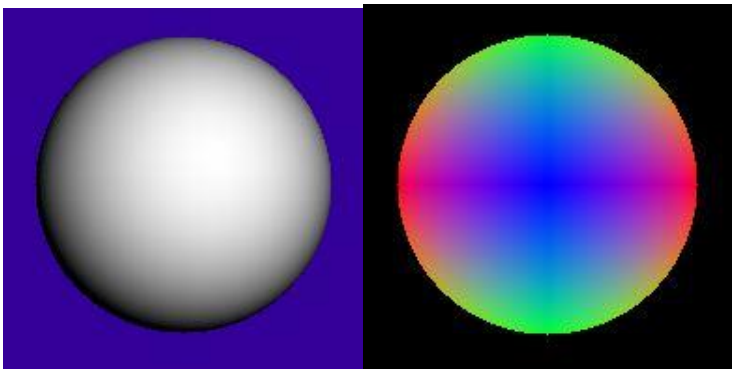
-input scene2_08.txt -size 200 200 -output output2_08.tga
-input scene2_09.txt -size 200 200 -output output2_09.tga
-input scene2_10.txt -size 200 200 -output output2_10.tga



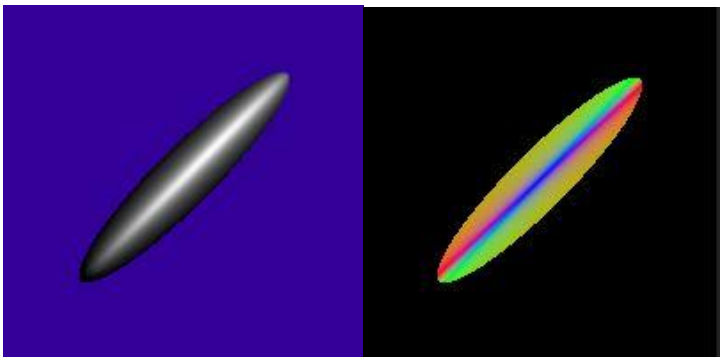
-input scene2_11.txt -size 200 200 -output output2_11.tga -normals normals2_11.tga



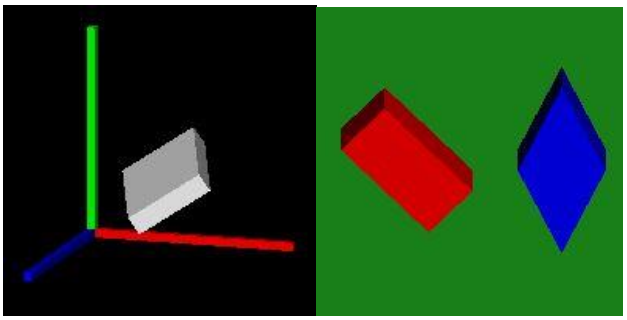
-input scene2_12.txt -size 200 200 -output output2_12.tga -normals normals2_12.tga



-input scene2_13.txt -size 200 200 -output output2_13.tga -normals normals2_13.tga



-input scene2_14.txt -size 200 200 -output output2_14.tga
-input scene2_15.txt -size 200 200 -output output2_15.tga



-input scene2_16.txt -size 200 200 -output output2_16.tga -depth 2 7 depth2_16.tga