Отчет по лабораторной работе №3 по курсу «компьютерная графика»
Студент группы 08-306: Баскаков О.А.
Преподаватель: Измайлов А.А.
Работа выполнена: "25" ноября 2009 г.

Тема: Основы построения фотореалистичных изображений.

Цель работы

Используя результаты лабораторной работы №2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Недопустимы видимые на поверхности швы. Объект должен иметь 3 степени свободы вращения. Обеспечить удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Дать возможность изменить положение источника света и параметров освещения.

Задача

Вариант 1: Прямой усеченный элиптический конус.

Оборудование

Ноутбук с предустновленным QT Creator, видеокарта ATI.

Теоретическая часть

Аппроксимированная выпуклыми многоугольниками (полигонами) фигура, была реализована во 2-ой лабораторной работе. Остается добавить модель освещения. В качестве модели освещения была взята модель Фонга.

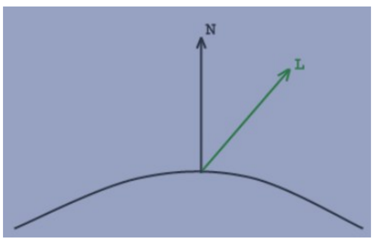
Пусть заданы точечный источник света, расположенный в некоторой точке, поверхность, которая будет освещаться и наблюдатель. Будем считать, что наблюдатель точечный. Каждая точка поверхности имеет свои координаты и в ней определена нормаль к поверхности. Её освещенность складывается из трех компонент: фоновое освещение (ambient), рассеянный свет (diffuse) и бликовая составляющая(specular). Свойства источника определяют мощность излучения для каждой из этих компонент, а свойства материала поверхности определяют её способность воспринимать каждый вид освещения.

1. **Фоновое освещение** это постоянная в каждой точке величина надбавки к освещению. Вычисляется фоновая составляющая освещения как:

   $$I_a = k_a i_a$$

   Из формулы выше видно, что фоновая составляющая освещенности не зависит от пространственных координат освещаемой точки и источника. Поэтому при моделировании освещения, в большинстве случае, не имеет смысла брать более одного фонового источника света. Часто просто задается некое глобальное фоновое освещение всей сцены.

2. **Рассеянный свет** при попадании на поверхность рассеивается равномерно во все стороны. При расчете такого освещения учитывается только ориентация поверхности (нормаль) и направление на источник света. Рассеянная составляющая рассчитывается по закону косинусов (закон Ламберта):
   $$I_d = k_d \cos(\vec{L}, \vec{N}) i_d = k_d(\vec{L} \cdot \vec{N}) i_d$$



3. Модель освещения Кука-Торренса

Так как эта модель используется для расчета отраженного света, то рассеянный свет мы будем вычислять по классической формуле Ламберта, в которой освещенность точки зависит только от угла между нормалью к поверхности в данной точки, и положением источника света. Вычисляется как скалярное произведение нормали и нормализованного положения источника света:

$$K_d = N \bullet L$$

Теперь рассмотрим модель Кука-Торренса. Количество отраженного света зависит от трех факторов:
1. Коэффициент Френеля (F)
2. Геометрическая составляющая, учитывающая самозатенение (G)
3. Компонент, учитывающий шероховатость поверхности (D)

Общая формула для вычисления отраженного света такова:

$$K = \frac{F \cdot G \cdot D}{(\vec{V} \bullet \vec{N}) \cdot (\vec{L} \bullet \vec{N})}$$

Рассмотрим вычисление геометрической составляющей:

$$G = \min\left(1, \frac{2(\vec{H} \bullet \vec{N})(\vec{V} \bullet \vec{N})}{(\vec{V} \bullet \vec{H})}, \frac{2(\vec{H} \bullet \vec{N})(\vec{L} \bullet \vec{N})}{(\vec{V} \bullet \vec{H})}\right)$$

где N – нормаль в точке, V – вектор взгляда, L – положение источника света, H – нормализованная сумма векторов L и V. Все векторы должны быть нормированы.

Компонент, учитывающий шероховатость поверхности – это распределение микрограней поверхности, для более точного учета отраженного от них света. Обычно, для вычисления этого компонента используют распределение Бекмана:

$$D = \frac{1}{4m^2(\vec{H} \bullet \vec{N})^4} \cdot e^{\left(\frac{(\vec{H} \bullet \vec{N})^2 - 1}{m^2(\vec{H} \bullet \vec{N})^2}\right)}$$

где параметр m (от 0 до 1) определяет шероховатость поверхности. Чем он больше, тем поверхность шероховатее, следовательно, отражает свет даже под широкими углами.

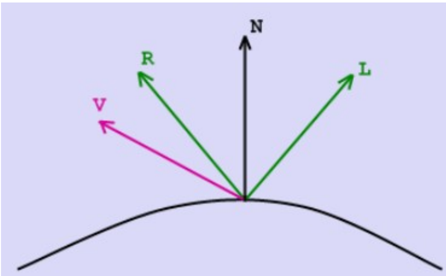
Коэффициент Френеля.
Для вычисления коэффициента Френеля существует много формул, но в данном случае целесообразнее применять аппроксимацию Шлика:

$$F = F_0 + \left(1 - (\vec{V} \bullet \vec{N})\right)^5 \cdot (1 - F_0)$$

где F0 – количество отраженного света при нормальном падении (перпендикулярно поверхности). Для того чтобы не вводить дополнительный параметр, и не усложнять процесс вычислением степени, можно использовать другую формулу:

$$F = \frac{1}{1 + (\vec{V} \bullet \vec{N})}$$

С поправкой на то, что металлы хорошо отражают – она дает так же неплохой результат. Таким образом, мы вычислили все составляющие, и теперь можно подставить их в исходную формулу и получить ожидаемый результат.



Именно зеркальное отражение представляет наибольший интерес, но в то же время его расчет требует больших вычислительных затрат. При фиксированном положении поверхности относительно источников света фоновая и рассеянные составляющие освещения могут быть просчитаны единожды для всей сцены, т.к. их значение не зависит от направления взгляда. С зеркальной составляющей этот фокус не сработает и придется пересчитывать её каждый раз, когда взгляд меняет свое направление.

Для упрощенного вычисления зеркального отражения можно использовать модель Блинна-Фонга. Вычислим в каждой точке вектор полупути H:

$$\vec{H} = \frac{\vec{L} + \vec{V}}{|\vec{L} + \vec{V}|} = (\vec{L} + \vec{V})_{norm}$$

который показывает ориентацию площадки, на которой будет максимальное отражение. Тогда величину (R*V)^a можно заменить величиной (H*N)^b.

Во всех вычислениях выше, для рассеянной и зеркальной компонент, если скалярное произведение в правой части меньше нуля, то соответствующая компонента освещенности полагается равной нулю.

Ход решения

Используя результаты лабораторной работы №2, добавим новое тело. Приходится изменить конвейер рендеринга с дублирующимися вершинах на смежных гранях и реализовать закраску полигонов с учетом освещения.

Для данной лабораторной работы был реализован удобный пользовательский интерфейс, который включает в себя изменение пространственного положения фигуры, её масштабирование левой правой конпкой мыши и колесиком соотв.

Для работы со светом добавлены функции расчета нормали по векторам, косинуса угла и результирующего цвета грани.

Генерация усеченного элипт конуса требует расчета точек пересечения боковой поверхности и усекающей плоскости, заданной коэффициентами парам уравнения.

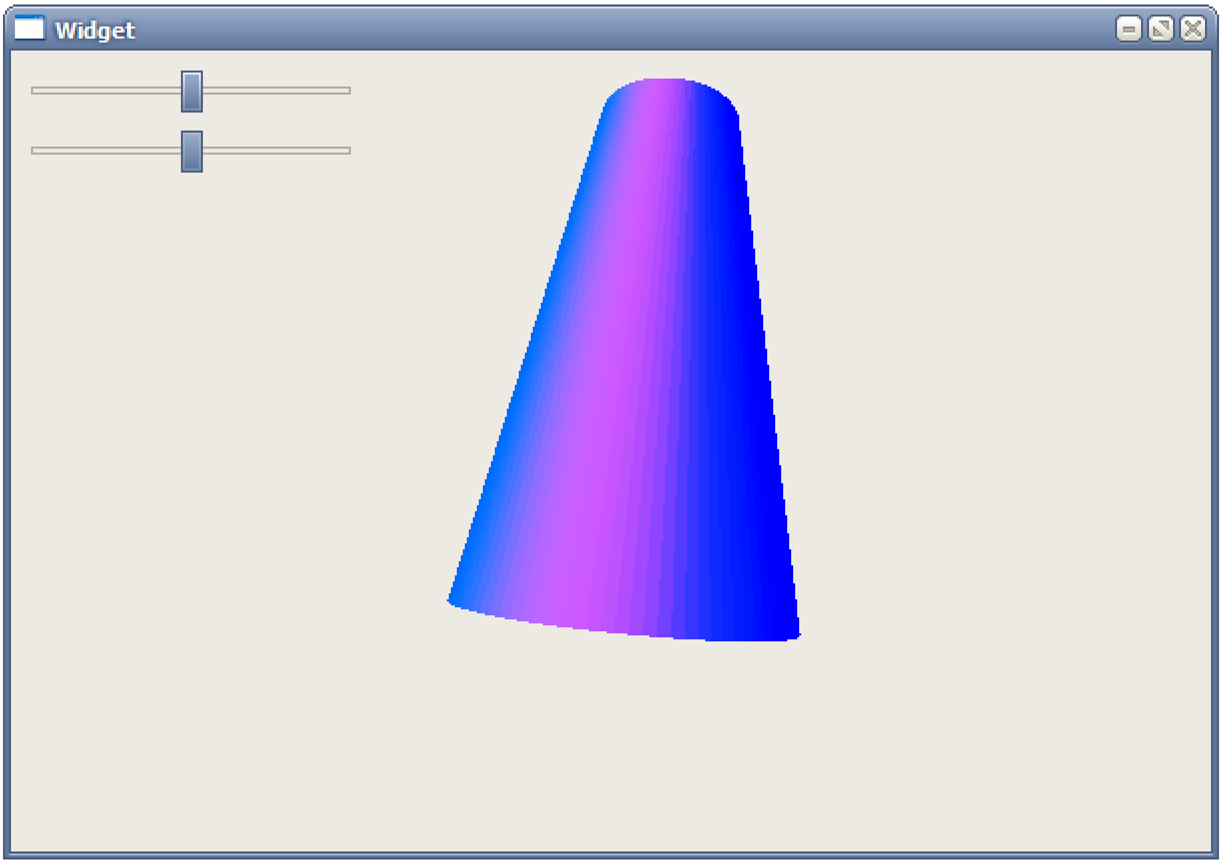
Исходный код

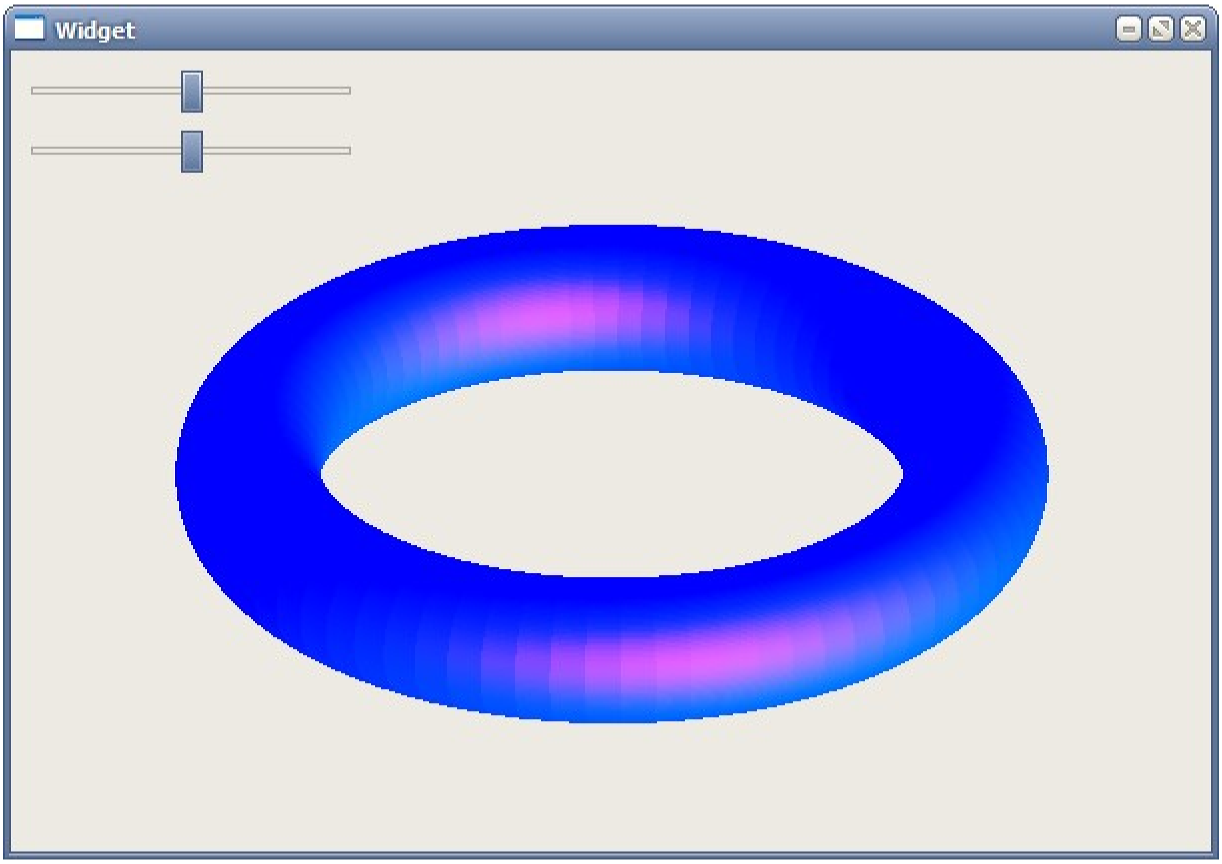
Для подготовки работы были выполнены следующие исходные файлы:

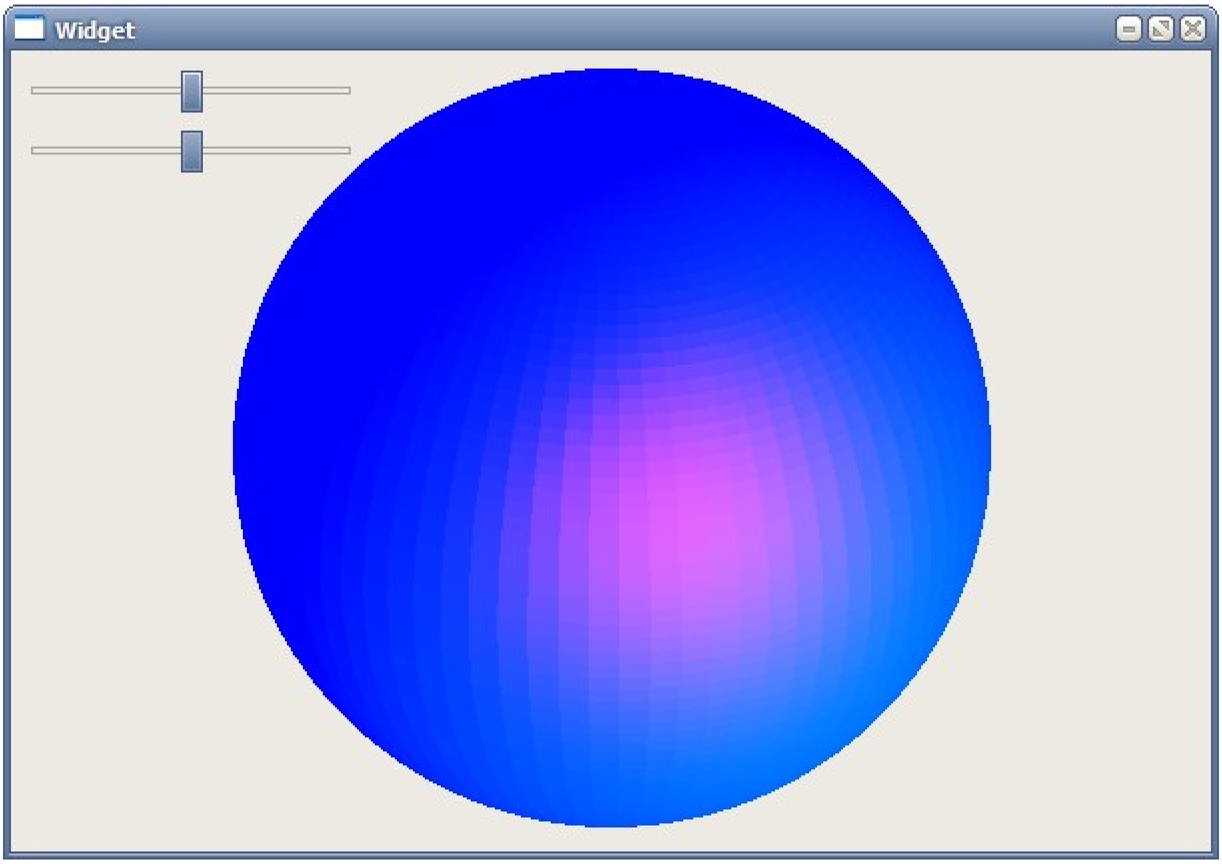
```
void Widget::SetColor(double cosDifuse,double cosSpecular)
{
// qDebug() << cosDifuse;
if (cosDifuse < 0) {cosDifuse = 0;}
if (cosSpecular < 0) {cosSpecular = 0 ;}
double red,green,blue;
red = color_anbient.red()/255.0+
cosDifuse*color_diffuse.red()/255.0+
// pow(cosSpecular*color_specular.red()/255.0,intense);
cosSpecular*color_specular.red()/255.0;
color.setRed(max(0,min(red*255,255)));
}

typedef float Matrix[4][4];
typedef float Point[4];

typedef struct Vector {
            float x;
            float y;
            float z;
            Vector(){};
            Vector (float a, float b, float c)
            {
                x = a;
                y = b;
                z = c;
            }
            Vector(Point p1, Point p2)
            {
                x = p2[0] - p1[0];
                y = p2[1] - p1[1];
                z = p2[2] - p1[2];
            }
        } Vector;

void Point_copy(float dst[4], const float src[4]);
void Point_mul(Point res,const Matrix M, const Point src);
float Point_scalar(Point p1, Point p2);
void Point_minus(Point res, Point p1, Point p2);
void Matrix4f_set_E(float M[4][4]);
void Matrix4f_set_scale(Matrix M, float scale);
void Matrix4f_set_rotX(Matrix M, float rotX);
```

```
void Matrix4f_set_rotY(Matrix M, float rotateY);
void Matrix4f_set_rotZ(Matrix M, float rotZ);
void Matrix4f_new(float M[4][4]);
void Matrix4f_mult(Matrix M1, Matrix M2);

#endif // MATRIX_H

#ifndef MATRIX_H
#define MATRIX_H
#include <math.h>

void Widget::paintEvent(QPaintEvent *event)
{
    QPainter p(this);
int i, j;
float dot[4];
    float ox, oy;

    ox=this->width()/2;
    oy=this->height()/2;

    for(i=0;i<6;i++) // Foreach polygon
    {
      normals = 0;
    for(i=0;i<quads/4;i++) // Foreach polygon
    {
        Vector norm;
        norm = Normalize(Vert[4*i], Vert[4*i+1], Vert[4*i+2]);

        Normals[normals++] = norm;



        // Always A,B = max(0,(A,B))!
        //Set V, H, L, N

        Vector N(-norm.x, -norm.y, -norm.z);
        Vector L;
        Vector V(0, 0, -1);
        N = Normalised(N);
        L = Normalised(light);
        V = Normalised(V);
        Vector H( L.x+V.x, L.y+V.y, L.z+V.z);
        H = Normalised(H);

        double G, D, F, K;
        double HN = CosV(H, N);
        double HN2= HN*HN;
        double VH = CosV(V, H);
        double VN = CosV(V, N);
        double LN = CosV(L, N);
```

```
        double m  = ui->val_m->value() *0.01;
        double f0 = ui->val_f0->value()*0.01;

        G = min(1, min( 2*HN*VN/VH, 2*HN*LN/VH ));

        D = exp( HN2-1/(m*m*HN2) ) / (4*m*m* HN2*HN2) ;

        //F = 1/(1 + CosV(V,N));
        F = f0 + (1-f0)*pow( 1-VN ,5);

        K = G*D*F/( VN*LN);
        K = max(0,K);

        //  qDebug() << normal.x  << normal.x  << normal.x
        SetColor(CosV(norm,light), K);
        //color = color_anbient;
        p.setPen(Qt::NoPen);
        p.setBrush(color);

        QPointF MMM[4];
        for(int k=0; k<4; k++)
        {
            MMM[k].setX( ox+Vert[4*i+k][0]);
            MMM[k].setY( oy-Vert[4*i+k][1]);
        }

        //  MMM[2] = ox+Vert[4*i+(j+1)%4][0];
        //  MMM[3] = oy-Vert[4*i+(j+1)%4][1];

        p.drawPolygon(MMM, 4);
    }
}
```


Widget


Widget



Замечания

Так как вершины представляется однородными координатами, неплохо было бы реализовать перспективную проекцию фигуры, что не было сделано в данной лабораторной работе. Так же параллельный перенос

Выводы

Были изучены различные модели освещения и способы их реализации, а также способы преобразования координат при помощи матриц.