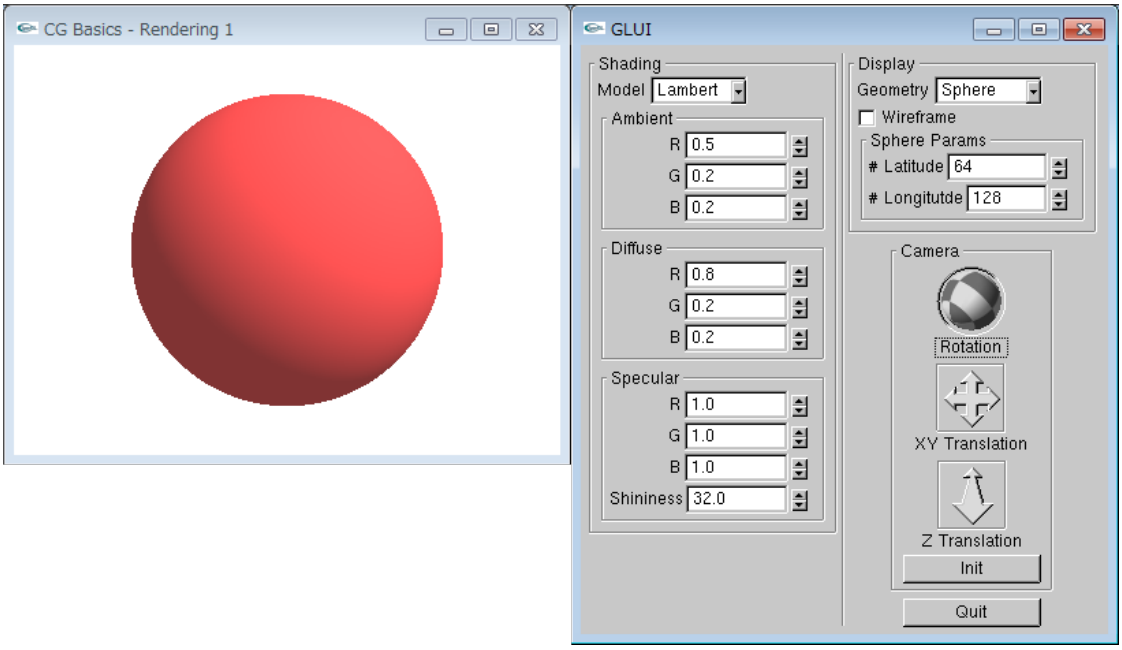
Vec3Algrebra.h に以下のインライン関数を追加した。

```
inline void Vec3commul(float outVec[3], const float v1[3], const float v2[3])
{
    outVec[0] = v1[0] * v2[0];
    outVec[1] = v1[1] * v2[1];
    outVec[2] = v1[2] * v2[2];
}
```

Lambertモデル

```
void CalcLambertModel(float outColor[3], const float ViewDir[3], const float Normal[3])
{
    // 第2項目
    float iinkd[3];
    Vec3commul(iinkd, g_LightColor, g_DiffuseColor);
    float L[3];
    Vec3Copy(L, g_LightDir);
    Vec3Negate(L);
    //float dotnl = Vec3Dot(Normal, L);
    float cosa = Vec3Dot(Normal, L) / (Vec3Norm(Normal) * Vec3Norm(L));
    if(cosa < 0) { cosa = 0; }
    float iinkdcosa[3];
    Vec3Mul(iinkdcosa, cosa, iinkd);
    // 第1項目と足す
    Vec3Add(outColor, g_AmbientColor, iinkdcosa);
}
```

実行結果



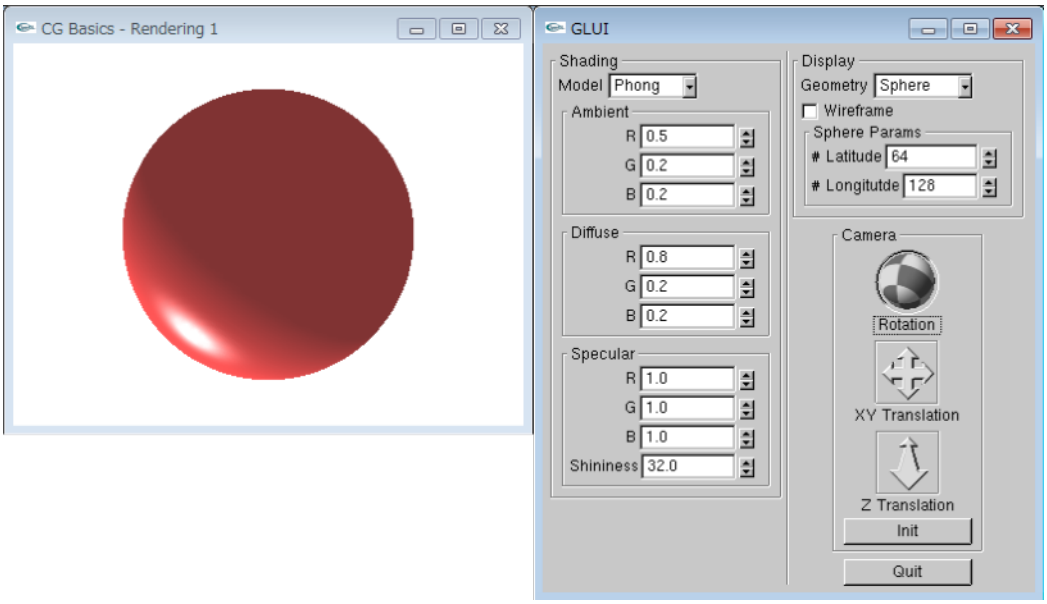
Phongモデル

```
void CalcPhongModel(float outColor[3], const float ViewDir[3], const float Normal[3])
{
    // 第２項目
    float IinKd[3];
    Vec3commul(IinKd, g_LightColor, g_DiffuseColor);
    float L[3];
    Vec3Copy(L, g_LightDir);
    Vec3Negate(L);
    float cosa = Vec3Dot(Normal, L) / (Vec3Norm(Normal) * Vec3Norm(L));
    if(cosa < 0) { cosa = 0; }
    float IinKdcosa[3];
    Vec3Mul(IinKdcosa, cosa, IinKd);

    // 第３項目
    // Rを求める！！！！！
    float IinKs[3];
    Vec3commul(IinKs, g_LightColor, g_SpecularColor);
    Vec3Copy(L, g_LightDir);
    float N_Lcosa[3];
    Vec3Mul(N_Lcosa, (Vec3Norm(L) * cosa / Vec3Norm(Normal)), Normal);
    float LverticalN[3];
    Vec3Add(LverticalN, L, N_Lcosa);
    float LtoR[3];
    Vec3Mul(LtoR, 2, LverticalN);
    float R[3];
    Vec3Sub(R, LtoR, L);
    // Rを使って第３項目を求める
    float RV = Vec3Dot(R, ViewDir);
    float cosc = RV / (Vec3SqrNorm(R) * Vec3SqrNorm(ViewDir));
    if(cosc < 0) { cosc = 0; }
    float cosnc = pow(cosc, g_SpecularShininess);
    float IinKscosnc[3];
    Vec3Mul(IinKscosnc, cosnc, IinKs);

    // 第１〜３項目を足す
    float except_Ia[3];
    Vec3Add(except_Ia, IinKdcosa, IinKscosnc);
    Vec3Add(outColor, g_AmbientColor, except_Ia);
}
```

実行結果



コードがごちゃごちゃして数式が直感的に分からないし、簡単な計算のに二度と読み返したくないコードになった。
関数が配列を返せば簡単になると思ったのだが、配列を返すことは普通にはできないらしい。
三谷先生のときのように3次元ベクトルを構造体で作れば構造体を返す以下のようなインライン関数が作れる。

```
inline Vector3d operator*( const double& k, const Vector3d& v ) { return Vector3d( k*v.x, k*v.y, k*v.z ); }
inline Vector3d operator*( const Vector3d& v, const double& k ) { return Vector3d( v.x*k, v.y*k, v.z*k ); }
inline Vector3d operator/( const Vector3d& v, const double& k ) { return Vector3d( v.x/k, v.y/k, v.z/k ); }
```

数式が手書きの数式に近い形で書けるようになるだろう。
GUI操作ツールが入ってきてコードを理解する余裕がなくなったのですが、本当はコードをちゃんと理解できた方が良いんでしょうね。厳しいです。