Задачи:

Написать программу для моделирования освещения объектов по Фонгу.

Написать программу для моделирования закраски объектов по Гуро (не выполнено).

Сложный объект в динамике.

Код:

**Main**:

Создаем рабочее окно.

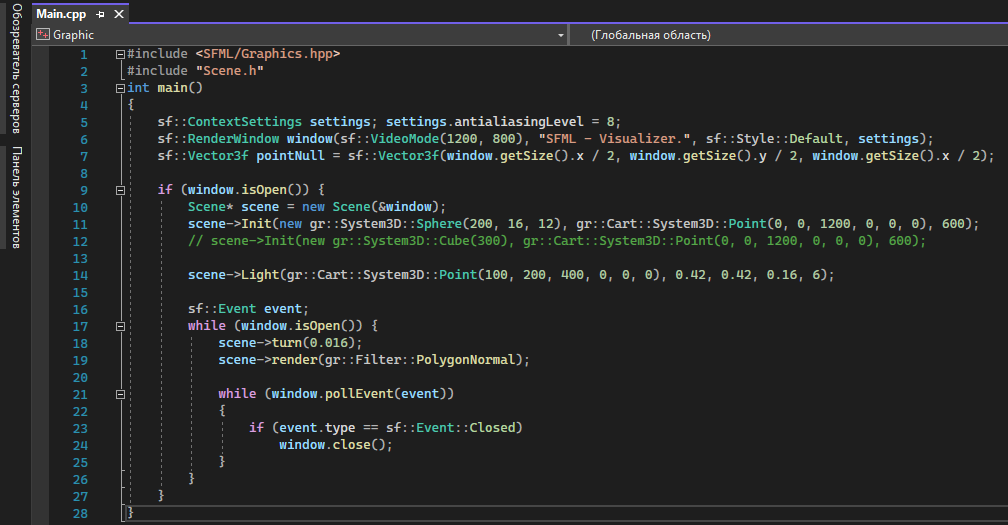
Создаем сцену (Scene\* scene = new Scene(&window)), где будем все обрабатывать.

Инициализируем сцену (scene->Init(new gr::System3D::Sphere(200, 16, 12), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)), где new gr::System3D::Sphere(200, 16, 12) – фигура, gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0) – камера, 600 – d.

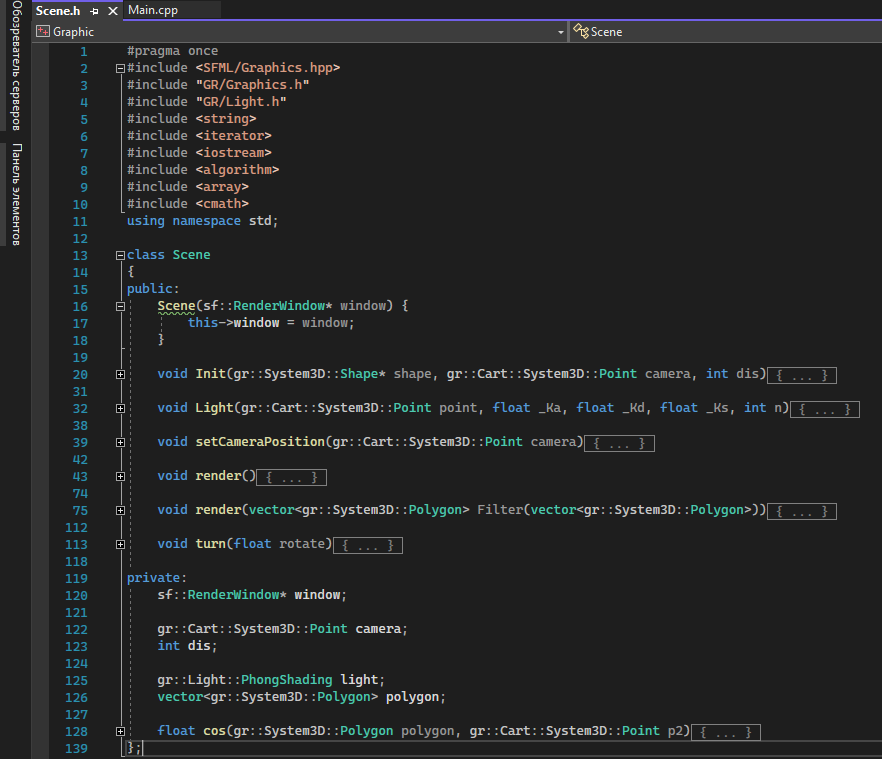
Создание источника света (scene->Light(gr::Cart::System3D::Point(100, 200, 400, 0, 0, 0), 0.42, 0.42, 0.16, 6)).

Поворот сцены по оси Y (scene->turn(0,016)).

Рендер сцены (scene->render(gr::Filter::PolygonNormal)).



**Scene**:



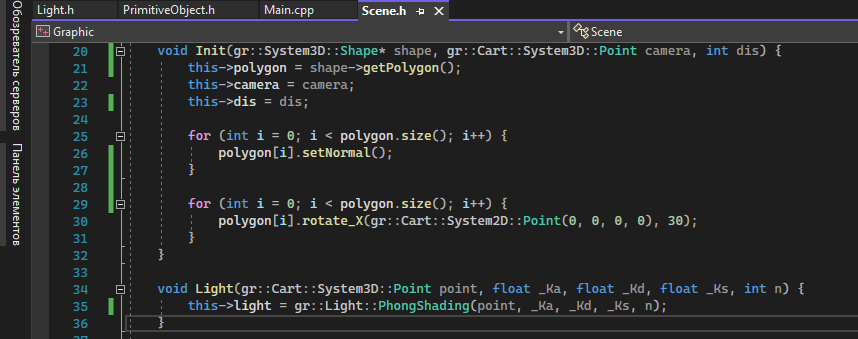
**Scene (Инициализация и свет) (Init / Light)**:

Init:

Заполняем данные.

Для каждого полигона высчитываем нормаль (берем нормали всех точек, принадлежащих полигону, суммируем их и нормируем).

Поворачиваем все полигоны относительно центра по оси X на 30 градусов.

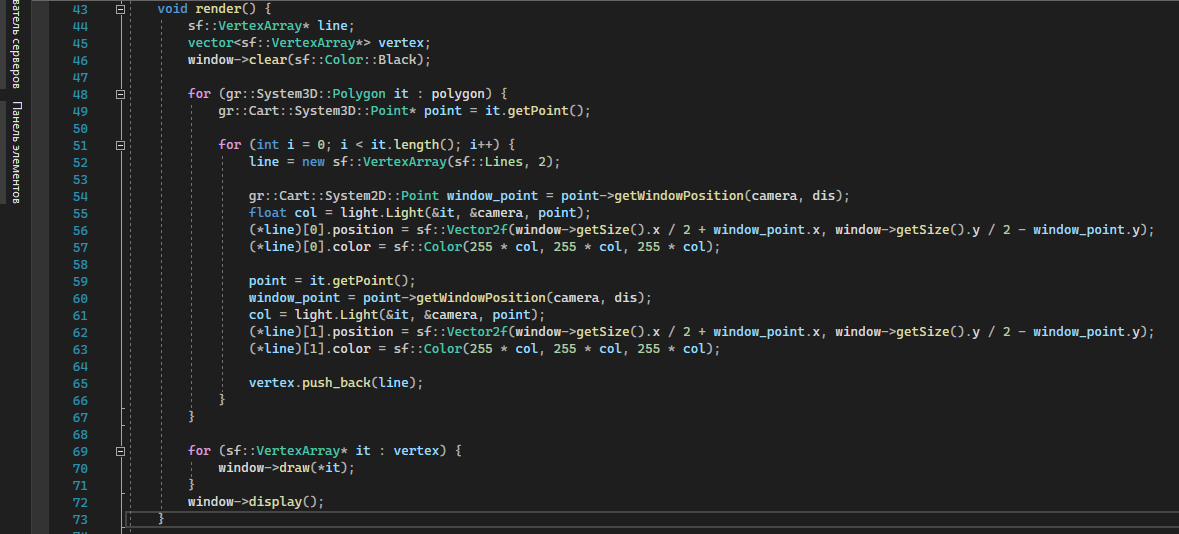


**Scene (Рендер) (render)**:

Очищаем экран.

Проходим по всем полигонам сцены и для всех них проектируем их на экран камеры (В результате получаем решетчатую фигуру).

Выводим результат.



**Scene (Рендер) (render(vector<gr::System3D::Polygon> Filter(vector<gr::System3D::Polygon>))):**

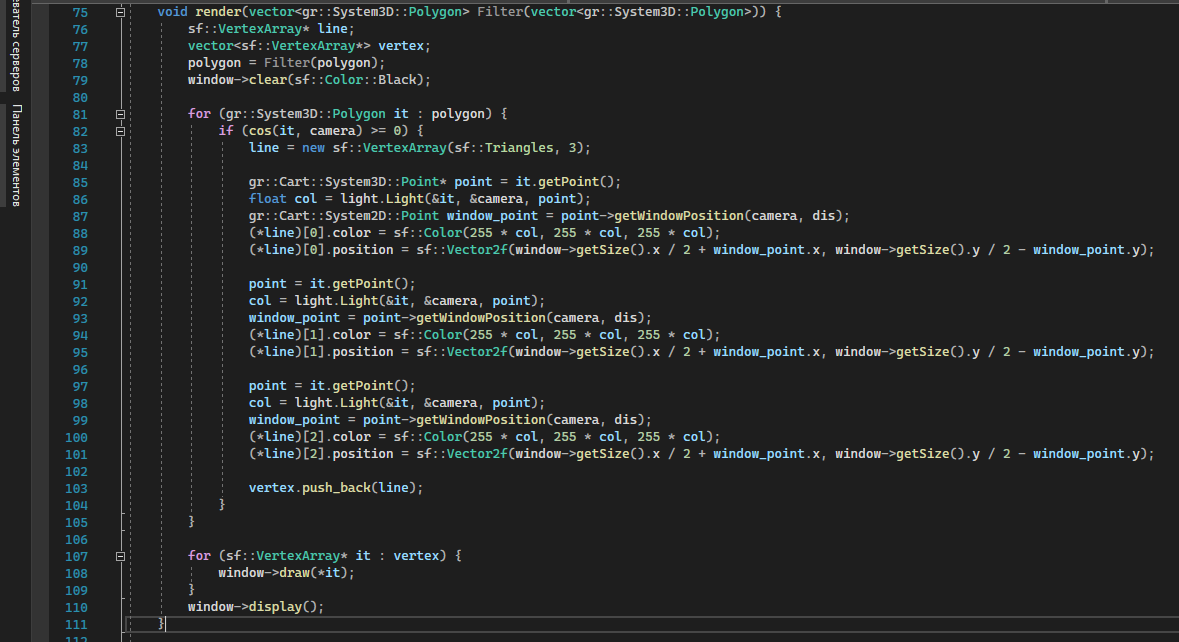
Очищаем экран.

Преобразуем нормали точек полигонов согласно фильтру.

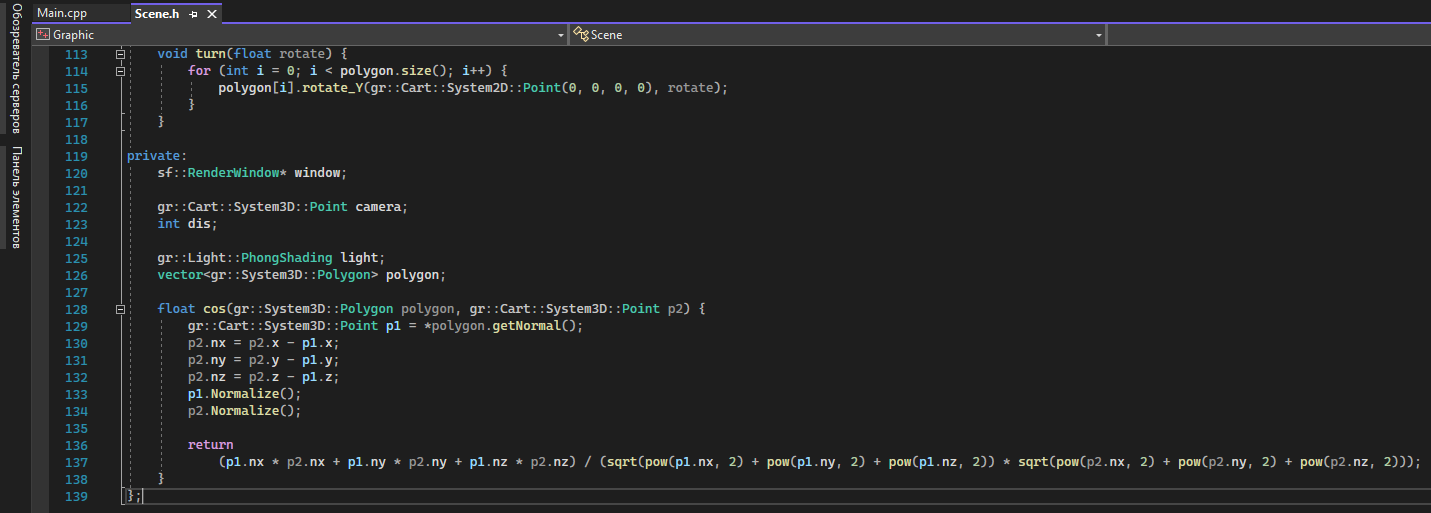
Делаем проверку видимости (простейший буфер глубины) (косинус угла между нормалью полигона и вектором к камере).

Проходим по всем полигонам сцены и для всех них проектируем их на экран камеры (В результате получаем фигуру).

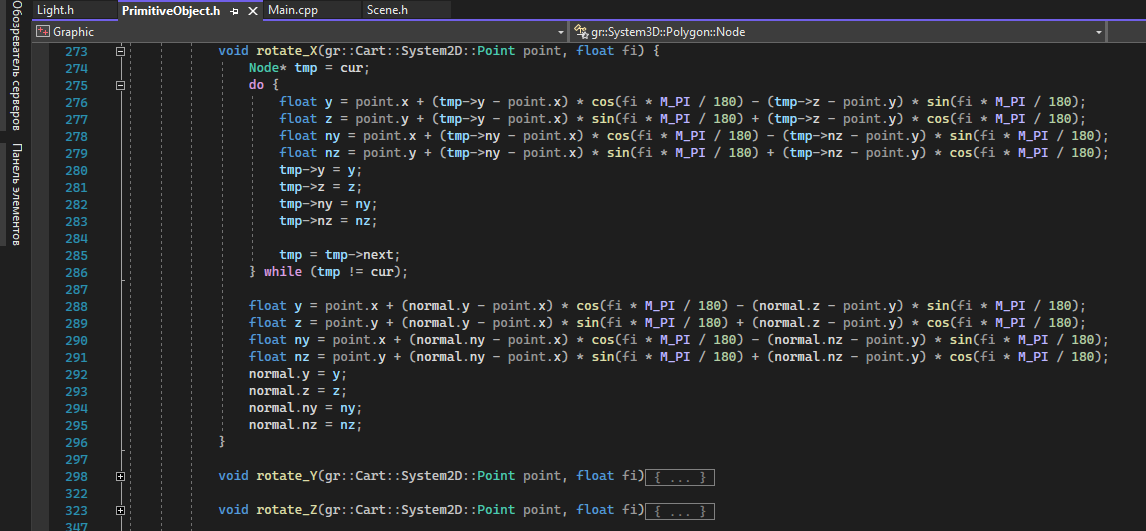
Выводим результат.



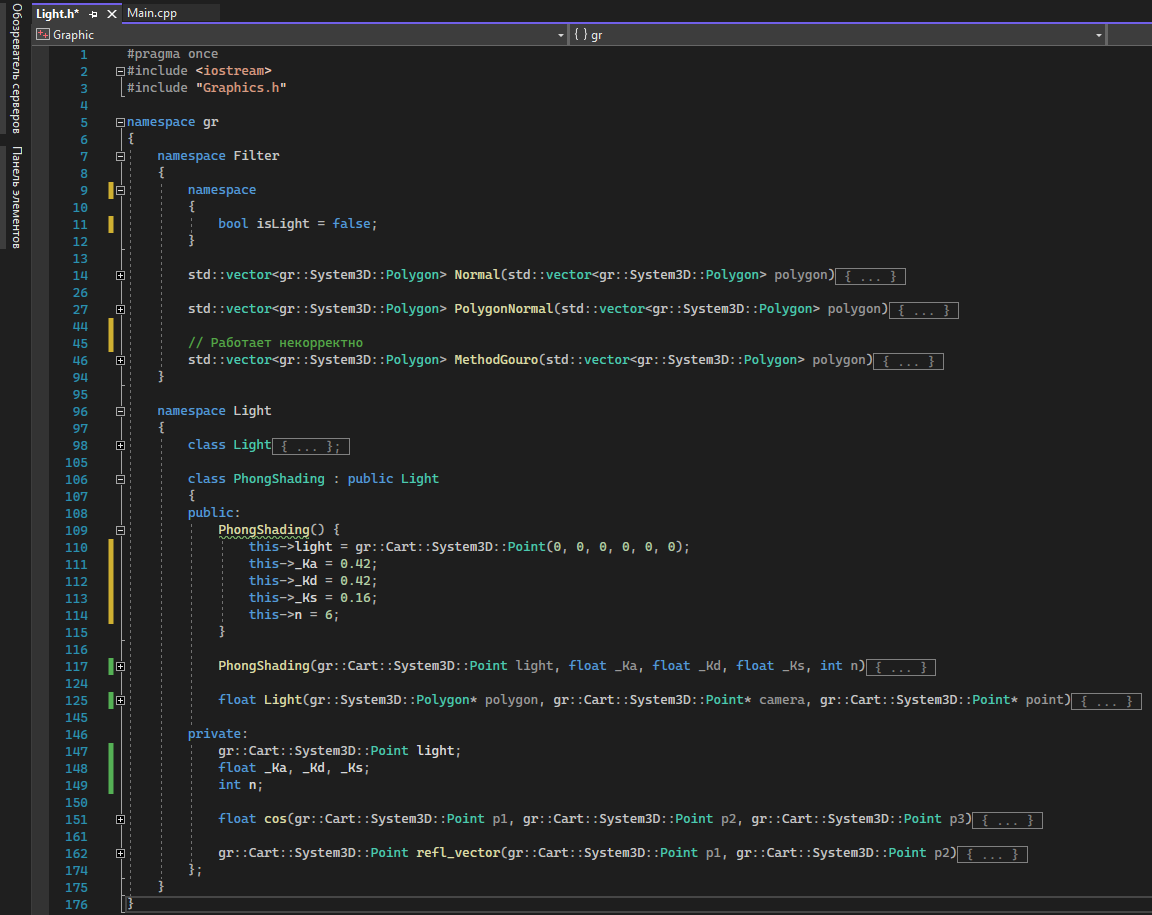
**Scene (Поворот и косинус):**



**Поворот полигона по оси X**:



**Light**:

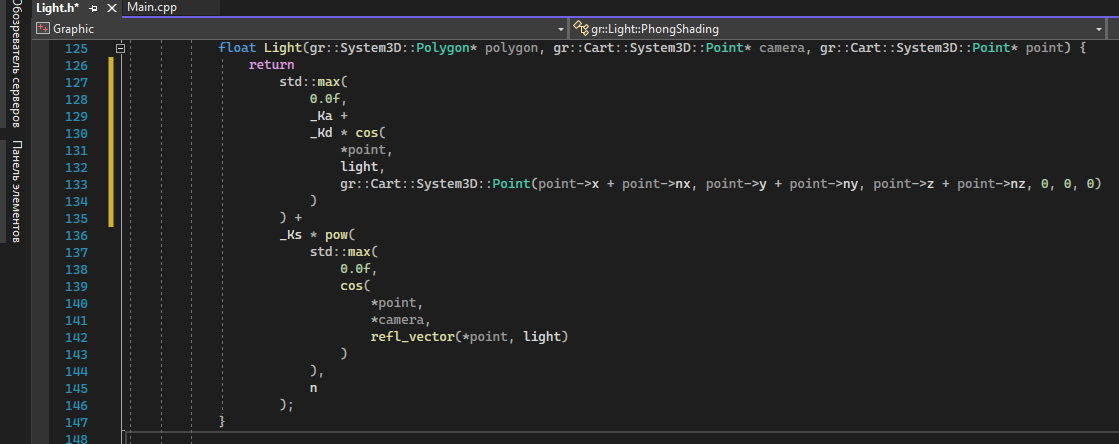


**Light (Вычисляем освещенность по Фонгу)**:

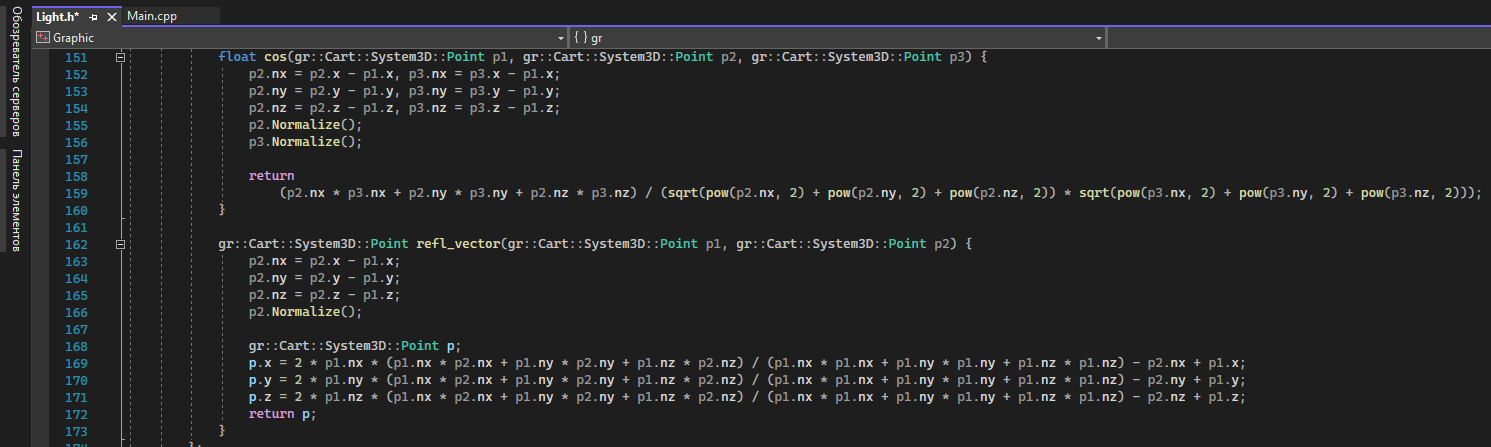
\_Ka – фоновое освещение.

\_Kd – рассеянная составляющая освещенности в точки (освещенность).

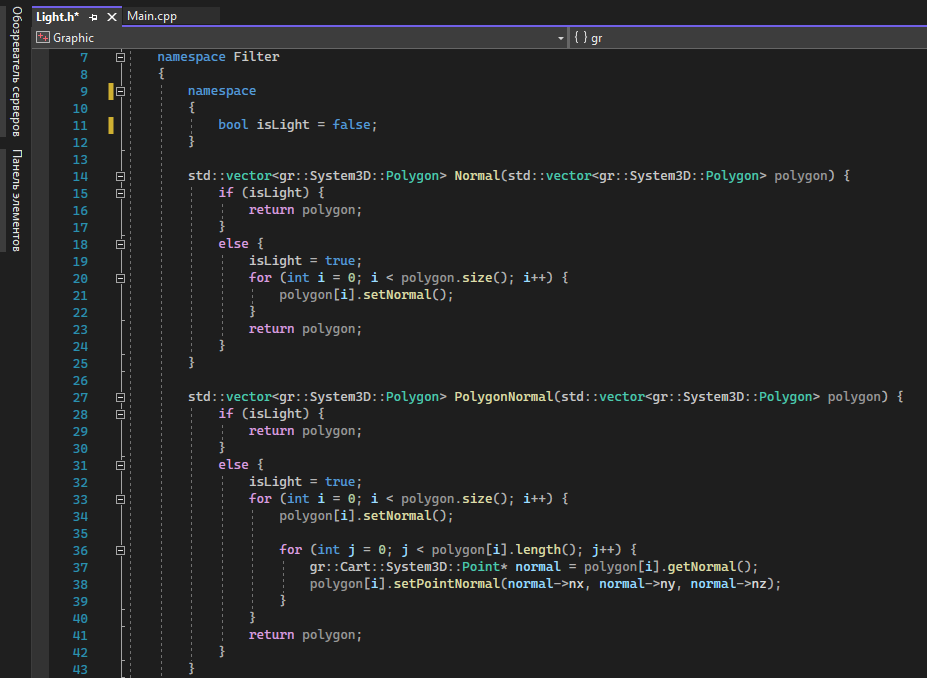
–Ks – зеркальная составляющая освещенности в точки (блики).



**Light (Косинус между точками и вычисление отражённого луча)**:



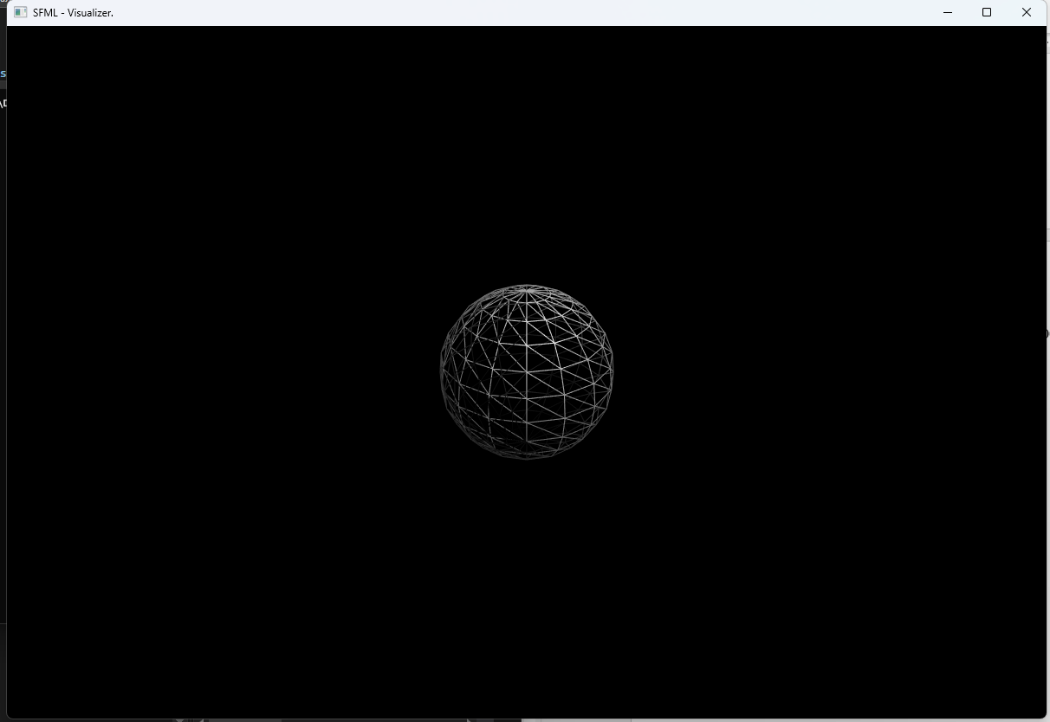
**Вычисляем нормали:**



Работа программы:

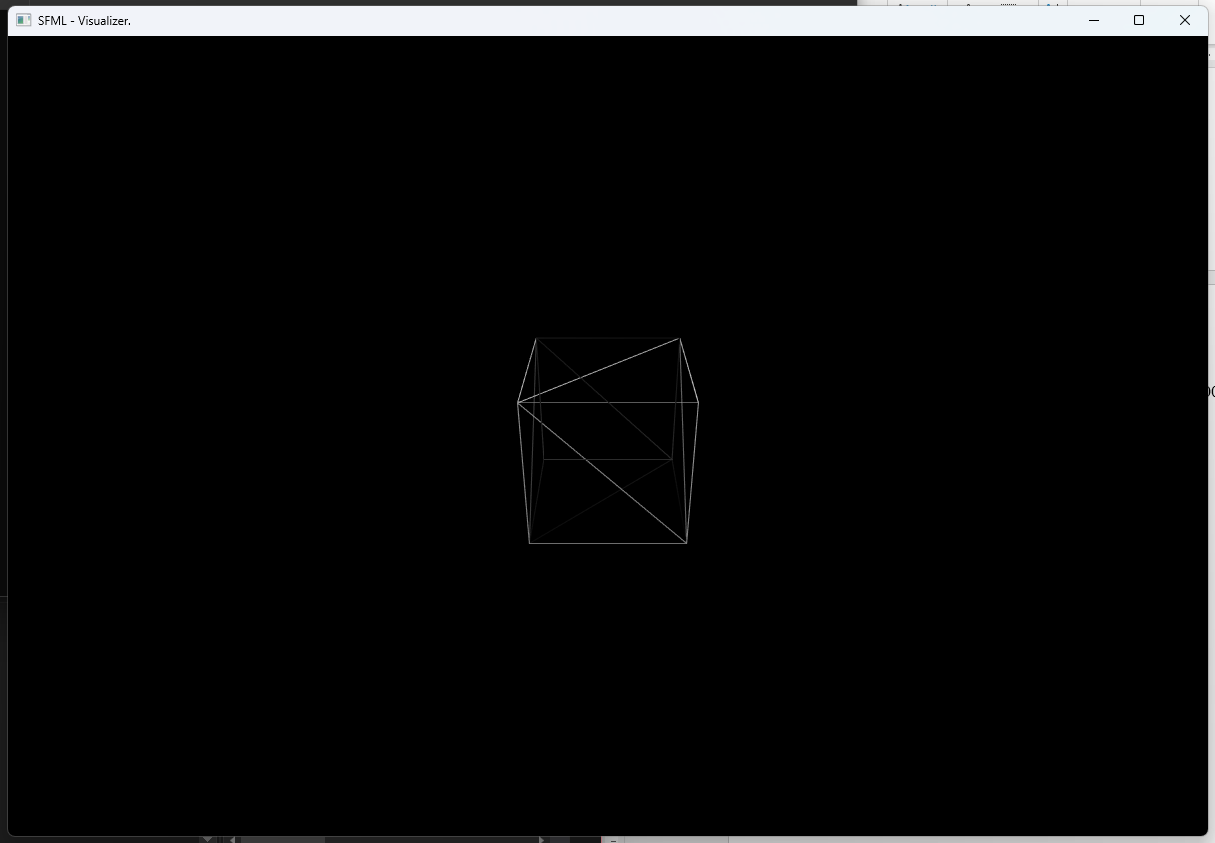
scene->Init(new gr::System3D::Sphere(200, 16, 12), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 200, 400, 0, 0, 0), 0.42, 0.42, 0.16, 6); scene->render()



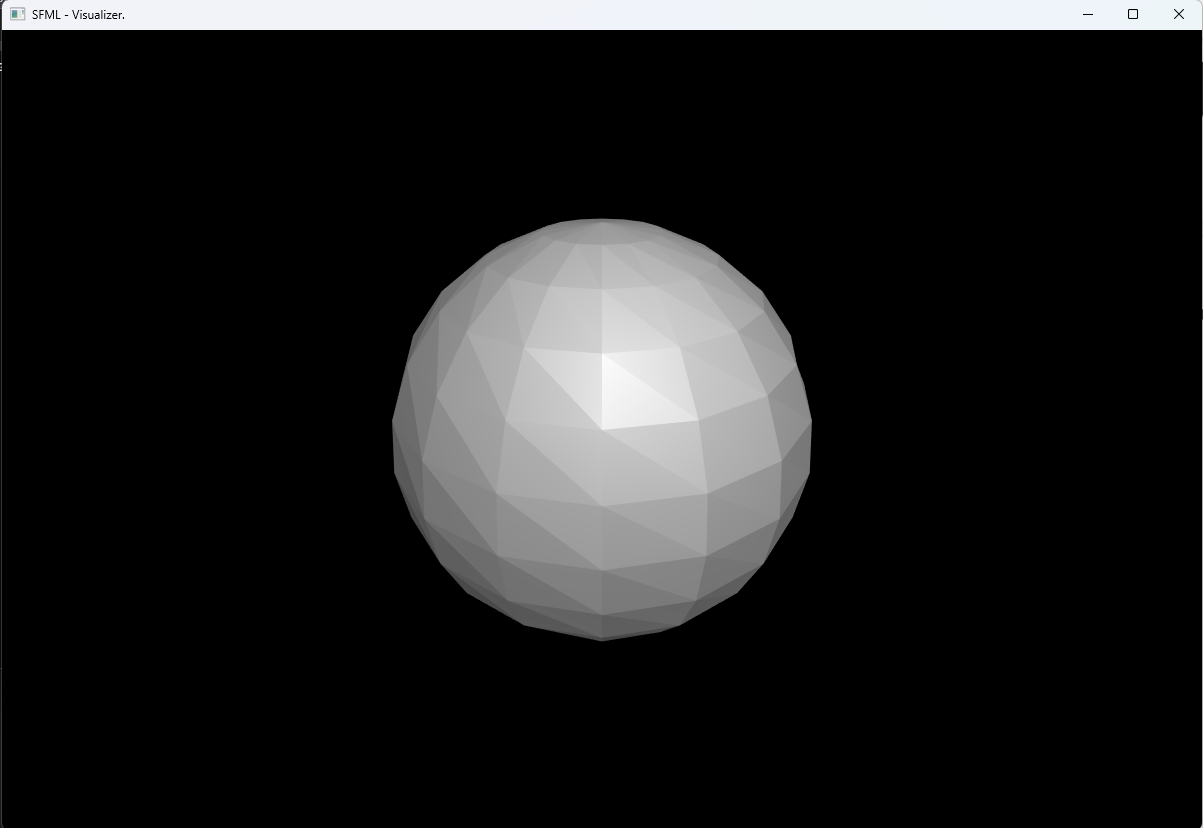
scene->Init(**new gr::System3D::Cube(300)**, gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 200, 400, 0, 0, 0), 0.42, 0.42, 0.16, 6); scene->render()



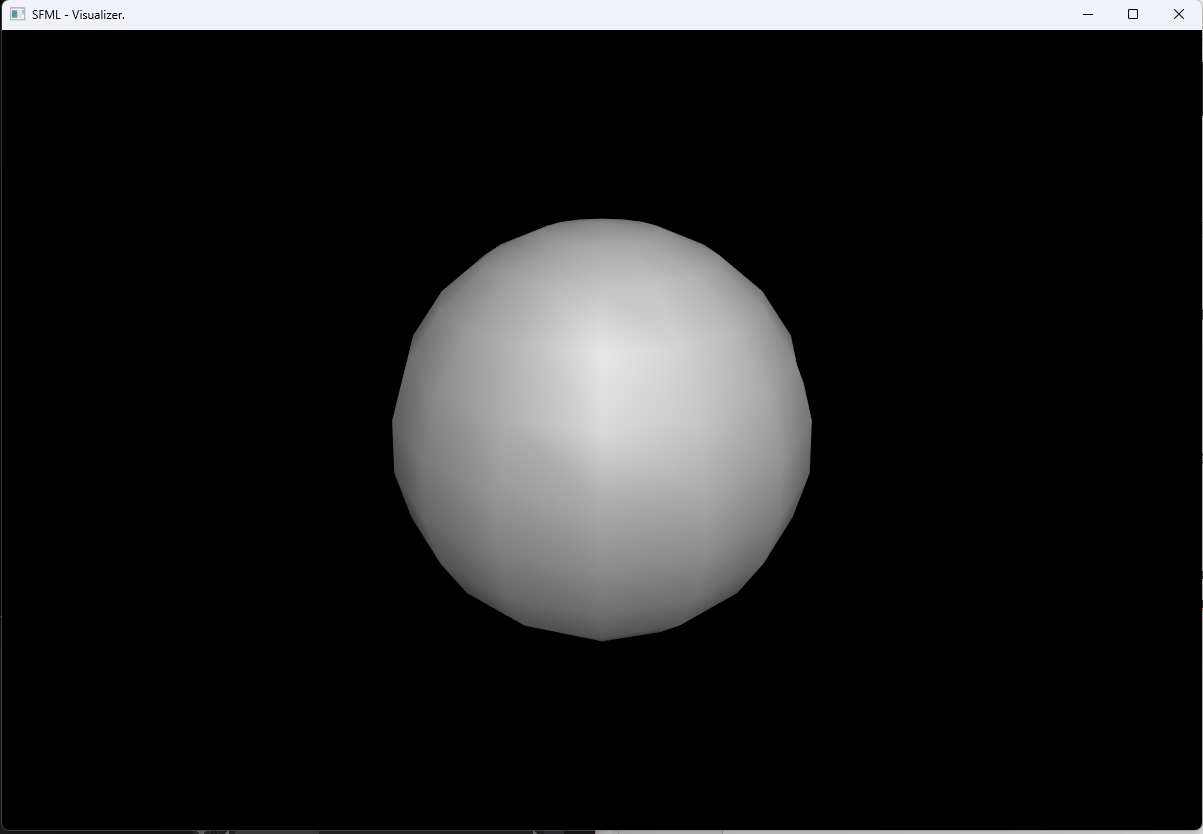
scene->Init(new gr::System3D::Sphere(**400**, 16, 12), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 200, **800**, 0, 0, 0), 0.42, 0.42, 0.16, 6); scene->render(**gr::Filter::PolygonNormal**)



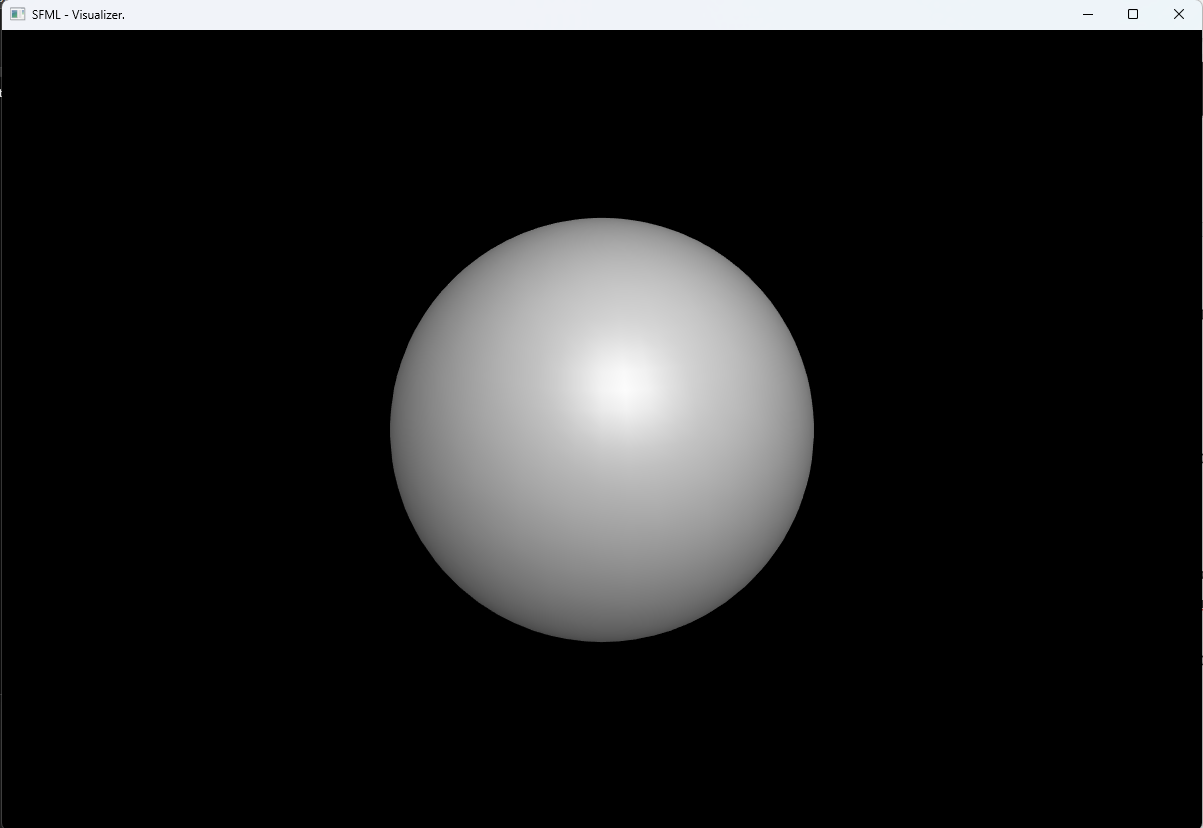
scene->Init(new gr::System3D::Sphere(400, 16, 12), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600); scene->Light(gr::Cart::System3D::Point(100, 200, 800, 0, 0, 0), 0.42, 0.42, 0.16, 6); scene->render(**gr::Filter::Normal**)

**// gr::Filter::Normal – сглаживаться будет только круг, с квадратом будет работать аналогично gr::Filter::PolygonNormal**



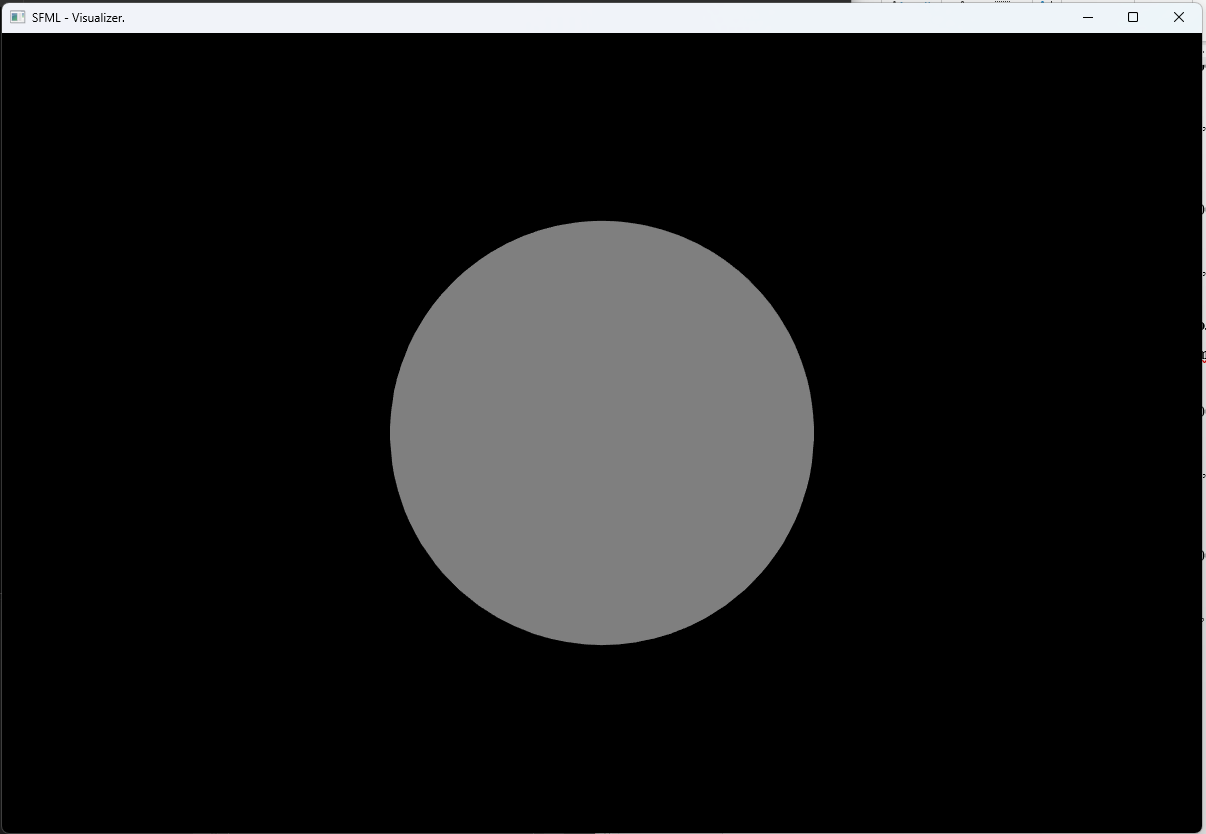
scene->Init(new gr::System3D::Sphere(400, **64**, **48**), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 200, 800, 0, 0, 0), 0.42, 0.42, 0.16, 6); scene->render(gr::Filter::Normal)



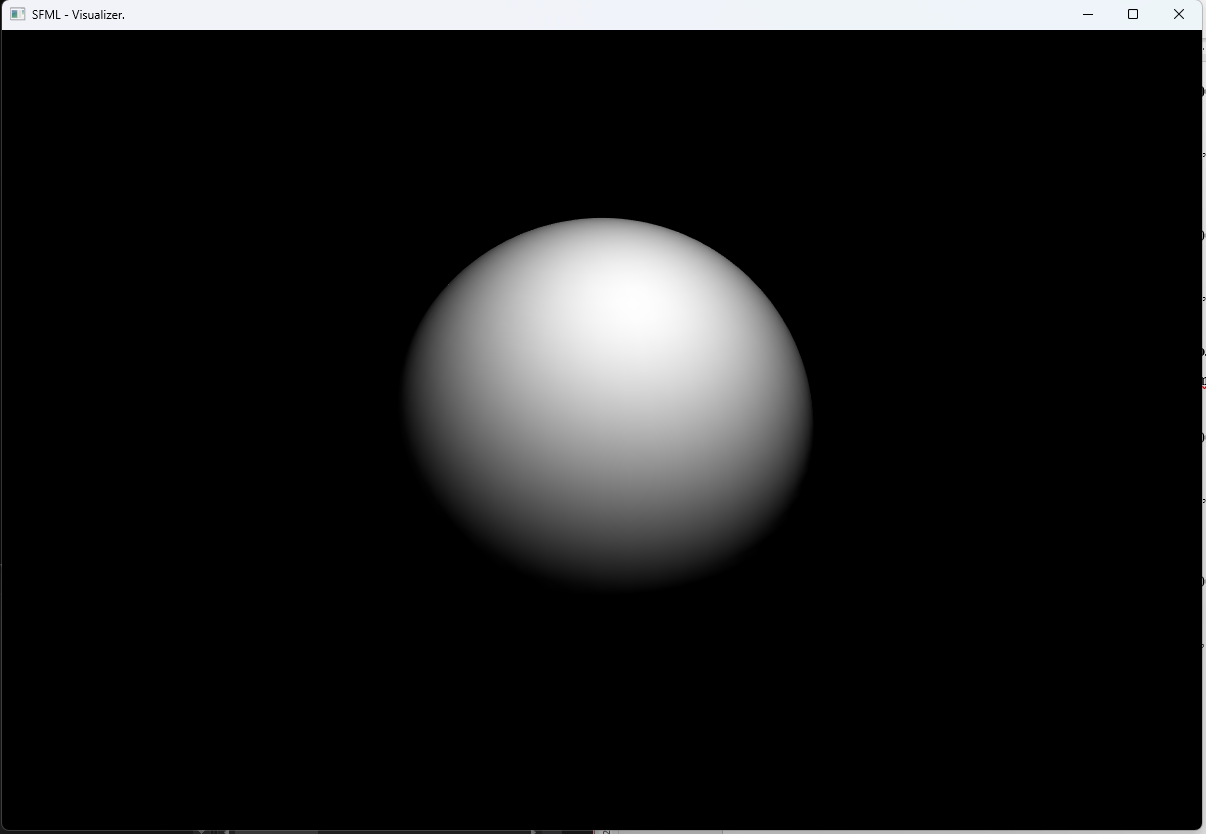
scene->Init(new gr::System3D::Sphere(400, 64, 48), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, **400**, 800, 0, 0, 0), **0.5, 0, 0, 6**); scene->render(gr::Filter::Normal)



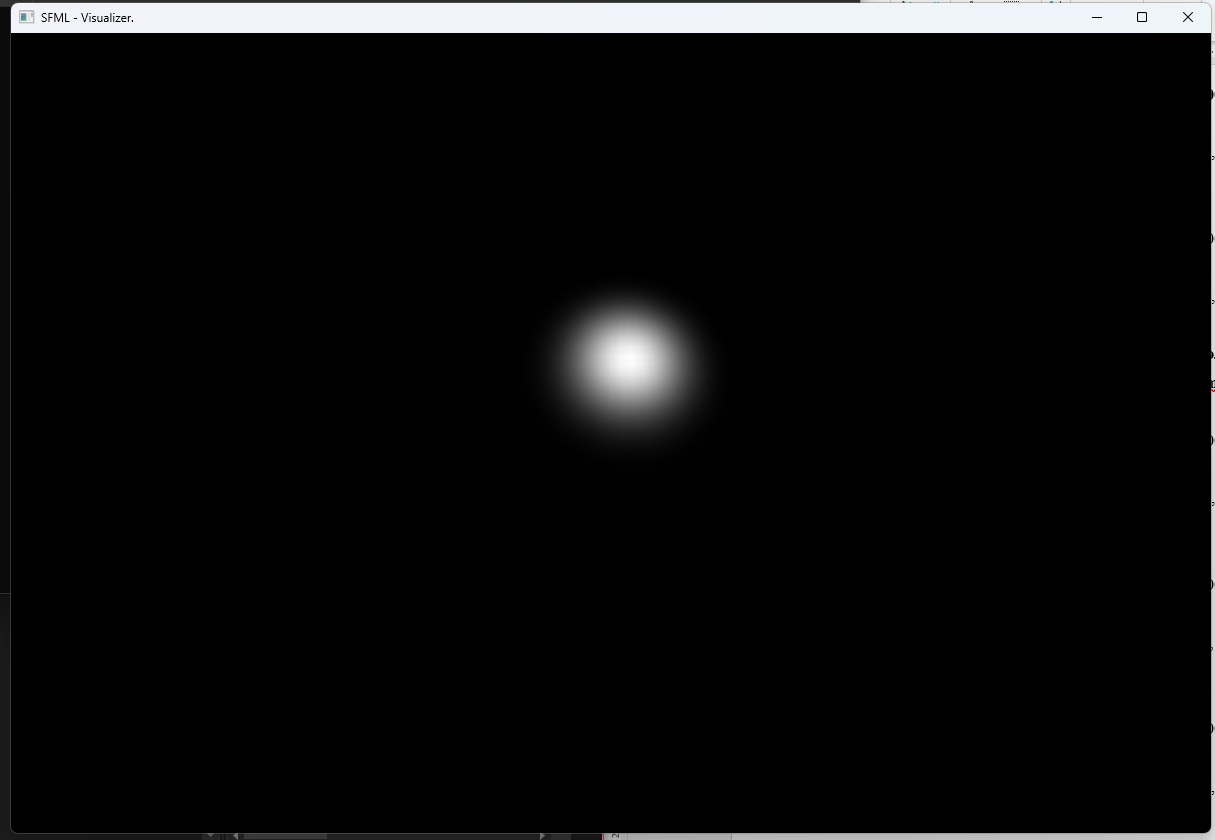
scene->Init(new gr::System3D::Sphere(400, 64, 48), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 400, 800, 0, 0, 0), **0, 1, 0, 6**); scene->render(gr::Filter::Normal)



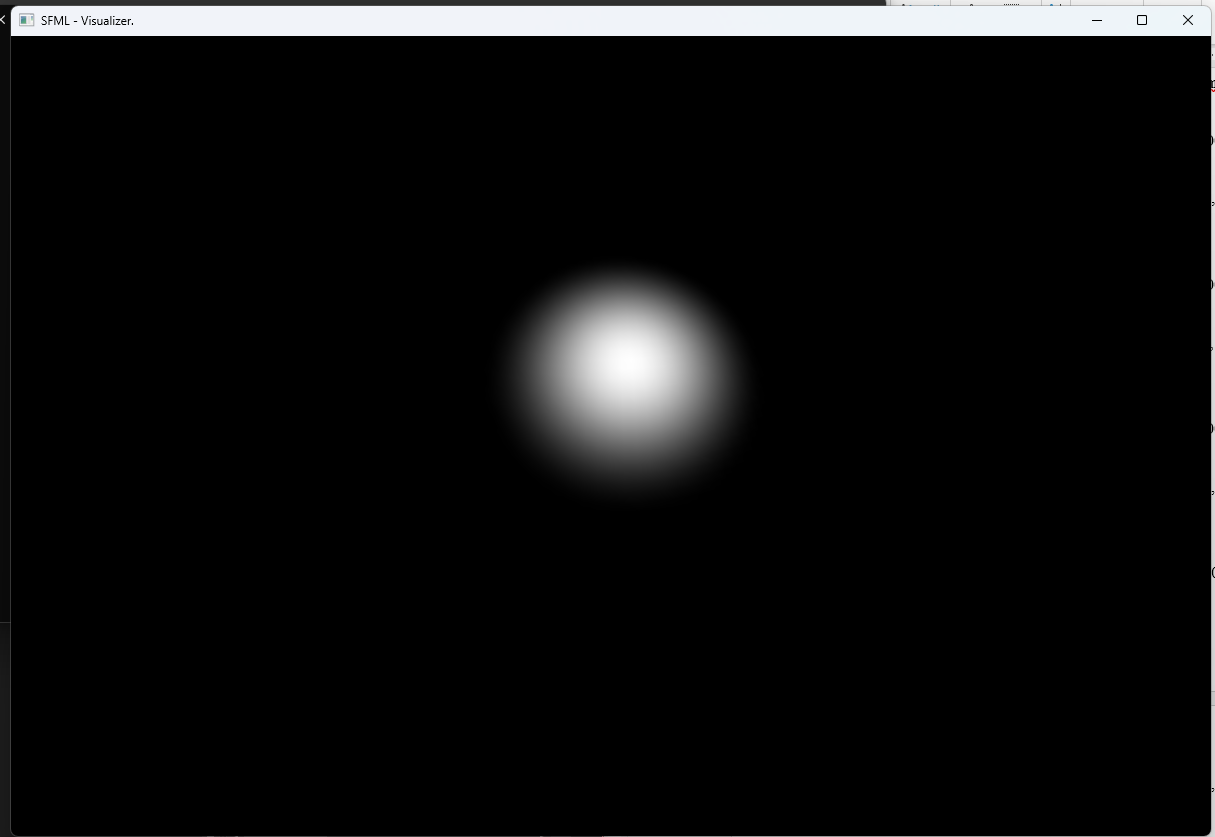
scene->Init(new gr::System3D::Sphere(400, **128, 96**), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 400, 800, 0, 0, 0), **0, 0, 1, 6**); scene->render(gr::Filter::Normal)



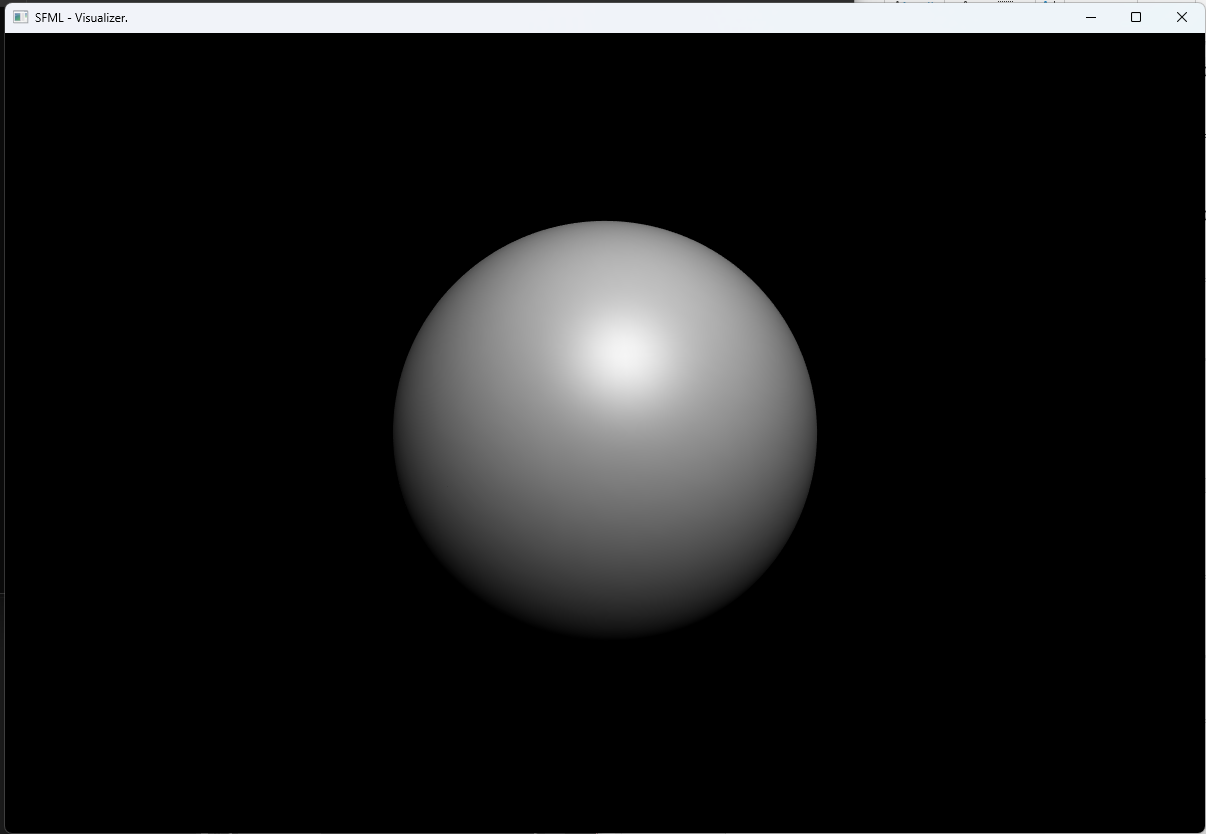
scene->Init(new gr::System3D::Sphere(400, **128, 96**), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 400, 800, 0, 0, 0), **0, 0, 1, 2**); scene->render(gr::Filter::Normal)



scene->Init(new gr::System3D::Sphere(400, 128, 96), gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 400, 800, 0, 0, 0), **0.2, 0.56, 0.24, 6**); scene->render(gr::Filter::Normal)



scene->Init(**new gr::System3D::Cube(300),** gr::Cart::System3D::Point(0, 0, 1200, 0, 0, 0), 600)

scene->Light(gr::Cart::System3D::Point(100, 400, 800, 0, 0, 0), 0.2, 0.56, 0.24, 6); scene->render(gr::Filter::Normal); **scene->turn(0.016);**

