

Построение трехмерной сцены объектов

Студент:

- Фролов Евгений ИУ7-55Б

Научный руководитель:

- Романова Т.Н.

Москва 2021

Цель работы - разработать программу для построения реалистического изображения из трехмерных геометрических объектов

Реализованное ПО предоставляет возможность:

а) выбора и добавления в сцену трехмерных объектов;

б) перемещения, поворота и масштабирования объектов;

в) изменение текстуры объекта, его цвет, свойств поверхности.

Формализация сцены

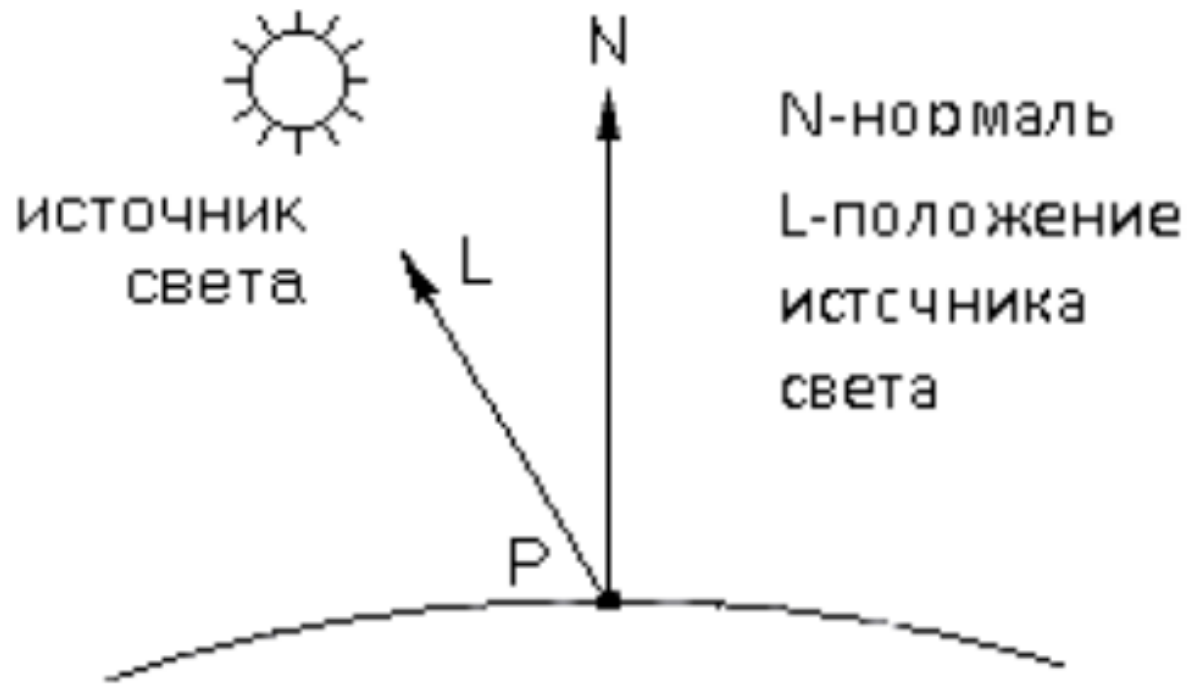
Трёхмерные
геометрические объекты;

источник света;

отражение фигур;

текстуры фигуры.

Простой метод освещения



В простом методе освещения интенсивность рассчитывается по закону Ламберта:

$$I = I_0 * \cos(\alpha), \text{ где}$$

- I – результирующая интенсивность света в точке
- I_0 – интенсивность источника
- α – угол между нормалью к поверхности и вектором направления света

Алгоритм Z-буфера

1. Всем элементам буфера кадра присвоить фоновое значение

2. Инициализировать Z буфер минимальными значениями глубины

3. Выполнить растровую развертку каждого многоугольника сцены:

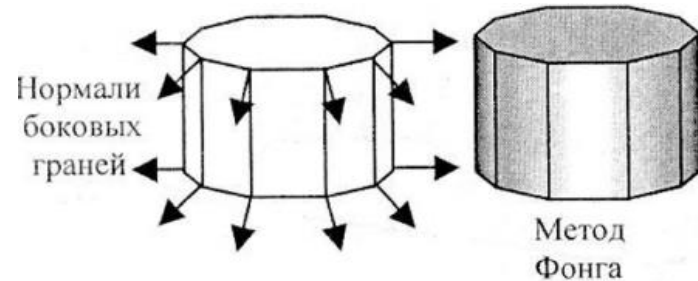
а. Для каждого пикселя, связанного с многоугольником вычислить его глубину $z(x, y)$

б. Сравнить глубину пикселя со значением, хранимым в Z буфере.

Если $z(x, y) > z_{\text{буф}}(x, y)$, то $z_{\text{буф}}(x, y) = z(x, y)$, $\text{цвет}(x, y) = \text{цветПикселя}$.

4. Отобразить результат

Процесс закраски по методу Фонга



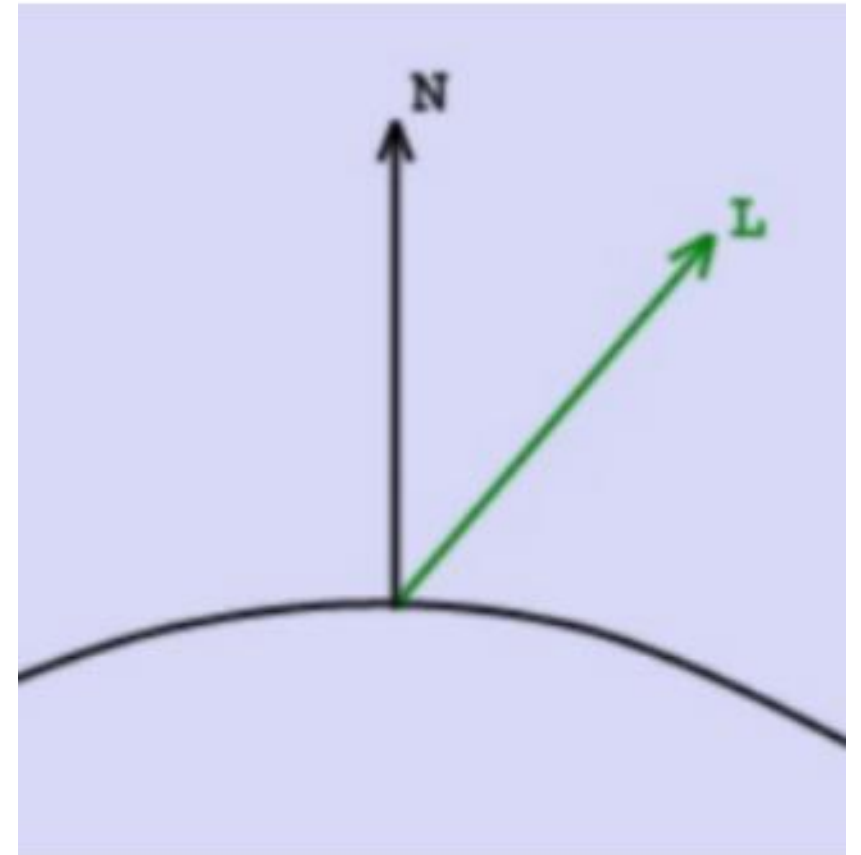
1. Определяются нормали к граням.
2. По нормалям к граням определяются нормали в вершинах.
3. В каждой точке закрашиваемой грани определяется интерполированный вектор нормали.
4. По направлению векторов нормали определяется цвет точек грани.

Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение. Свет при попадании на поверхность рассеивается равномерно во все стороны.

Учитывается:

- ориентация поверхности (нормаль N)
- направление на источник света (вектор L).



Выбор языка программирования и среды разработки

В качестве языка программирования был
выбран язык C++:

- т.к. работал с этим языком во время курса по предмету ООП.

В качестве среды разработки была выбрана
QtCreator:

- бесплатный для студентов;
- удобства отладки и написания кода;
- удобное создание интерфейса.

Структура и состав классов

<p><u>Класс</u></p> <p>Camera</p> <p><u>Поля</u></p> <p>position up direction fov zn zf aspect_ratio projectionMatrix x_rot_angle</p> <p><u>Методы</u></p> <p>Camera viewMatrix shiftX shiftZ shiftY rotateX rotateY rotateQautr</p>	<p><u>Класс</u></p> <p>Model</p> <p><u>Поля</u></p> <p>index_buffer vertex_buffer faces rotation_matrix scale_matrix texture has_texture specular reflective refractive transparency n color box angle_x angle_y angle_z shift_x shift_y shift_z scale_x scale_y scale_z</p> <p><u>Методы</u></p> <p>rotateX rotateY rotateZ shiftX shiftY shiftZ scaleX scaleY scaleZ objToWorld setColor getUId isObject wrap_angle genBox</p>	<p><u>Класс</u></p> <p>SceneManager</p> <p><u>Поля</u></p> <p>camers curr_camera models height width depthBuffer img background_color pixel_shader scene vertex_shader geom_shader models_index current_model vw vh d threads</p> <p><u>Методы</u></p> <p>init shift rotate scale moveCamera shift_Camera rotateCamera uploadModel removeModel setCurrentModel setColor setTexture render rasterize show rasterBarTriangle testAndSet clip setSpecular setReflective setRefraction setAmbIntensity</p>	<p><u>Класс</u></p> <p>Matrix</p> <p><u>Поля</u></p> <p>elements Matrix</p> <p><u>Методы</u></p> <p>Identity Scaling ScaleX ScaleY ScaleZ RotationZ RotationY RotationX Translation ProjectionFOV LookAtLH RotationMatrix Inverse</p>	<p><u>Класс</u></p> <p>Loader</p> <p><u>Поля</u></p> <p>Positions TCords Normals Vertices Indices MeshMatNames listening meshname tempMesh curline LoadedMeshes LoadedVertices LoadedIndices LoadedMaterials</p> <p><u>Методы</u></p> <p>LoadFile GenVerticesFromRawOBJ VertexTriangluation LoadMaterials</p>
--	--	--	---	---

<u>Класс</u>	<u>Класс</u>
Vec3	Vec4
<u>Поля</u>	<u>Поля</u>
x y z	w
<u>Методы</u>	<u>Методы</u>
Vec3 normalize len operator Vec3 operator = operator + operator += operator- operator -= operator != operator / operator * cross dot refract hadamard saturate	Vec4 operator Vec4 operator = operator + operator += operator- operator -= operator != operator / operator * operator *=

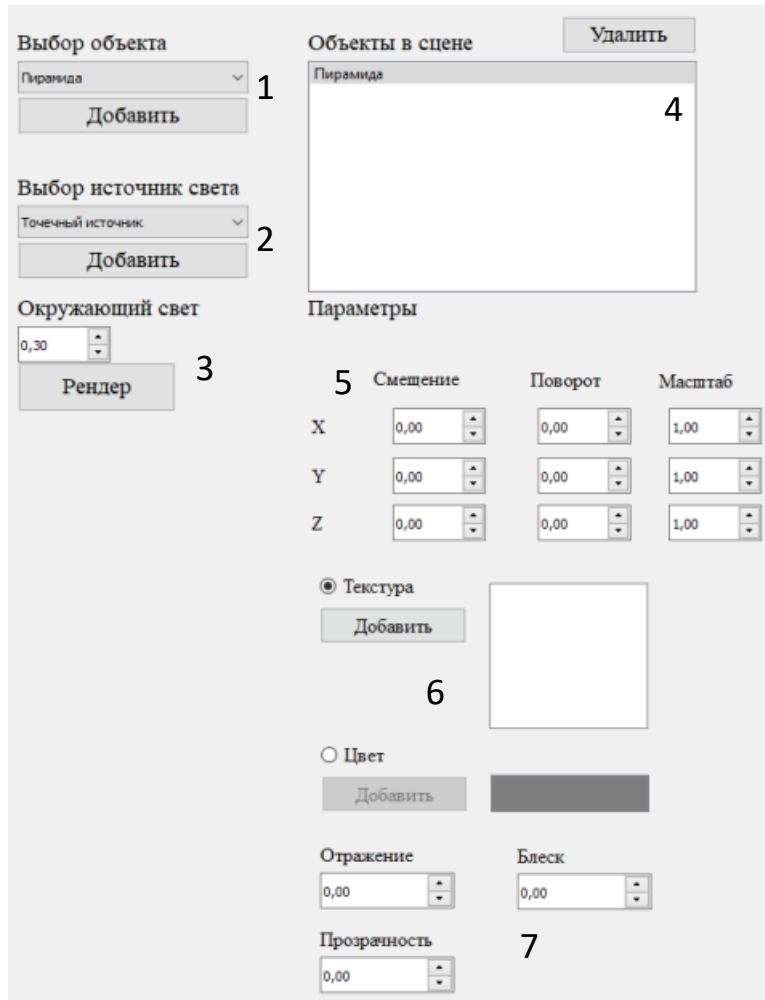
<u>Класс</u>	<u>Класс</u>
VertexShaderInterface	TextureShader
<u>Методы</u>	<u>Поля</u>
shade	texture
<u>Класс</u>	<u>Методы</u>
GeometryShaderInterface	shade
<u>Методы</u>	
shade	
<u>Класс</u>	
PixelShaderInterface	
<u>Методы</u>	
shade	

<u>Класс</u>	<u>Класс</u>	<u>Класс</u>
Ray	Primitive	BoundingBox
<u>Поля</u>	<u>Методы</u>	<u>Поля</u>
invdirection origin sign direction	intersect	min max bounds
<u>Методы</u>		<u>Методы</u>
Ray		BoundingBox

<u>Класс</u>	<u>Класс</u>
Vertex	VertexShader
<u>Поля</u>	<u>Поля</u>
pos normal color u v invW	dir light_color ambient intensity
<u>Методы</u>	<u>Методы</u>
Vertex	VertexShader shade

<u>Класс</u>
Light
<u>Поля</u>
t color_intensity position lightning_power direction
<u>Методы</u>
setType Light shiftX shiftZ shiftY rotateX rotateY rotateZ getDirection isObject
<u>Класс</u>
RayThread
<u>Поля</u>
img models inverse bound height width cam
<u>Методы</u>
run RayThread finished toWorld traceRay cast_ray computeLightning sceneIntersect

Интерфейс программы



1. Выбор объекта - добавление объектов в сцену

2. Выбор источника света

3. Изменение параметра окружающего света

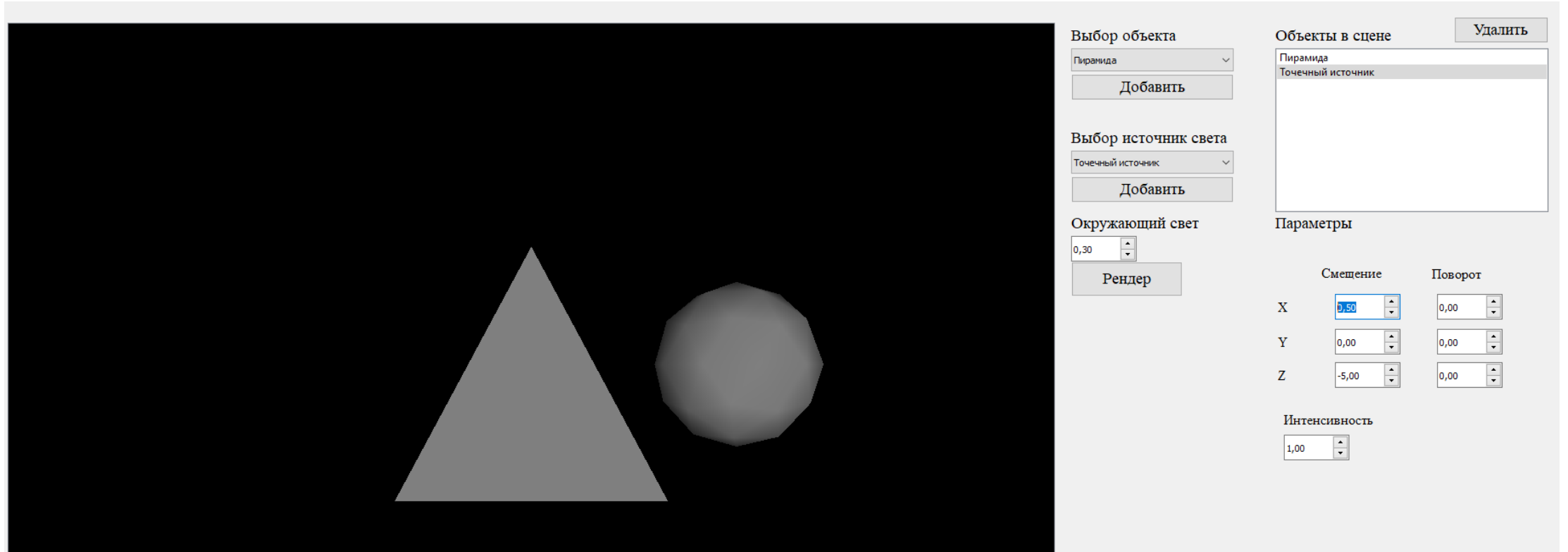
4. Просмотр объектов сцены

5. Изменение положения объекта

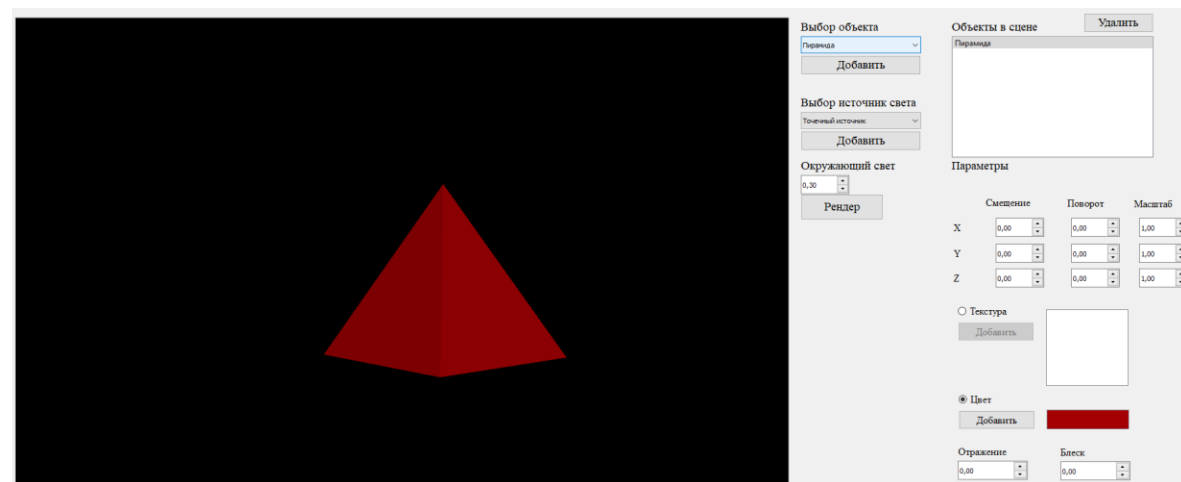
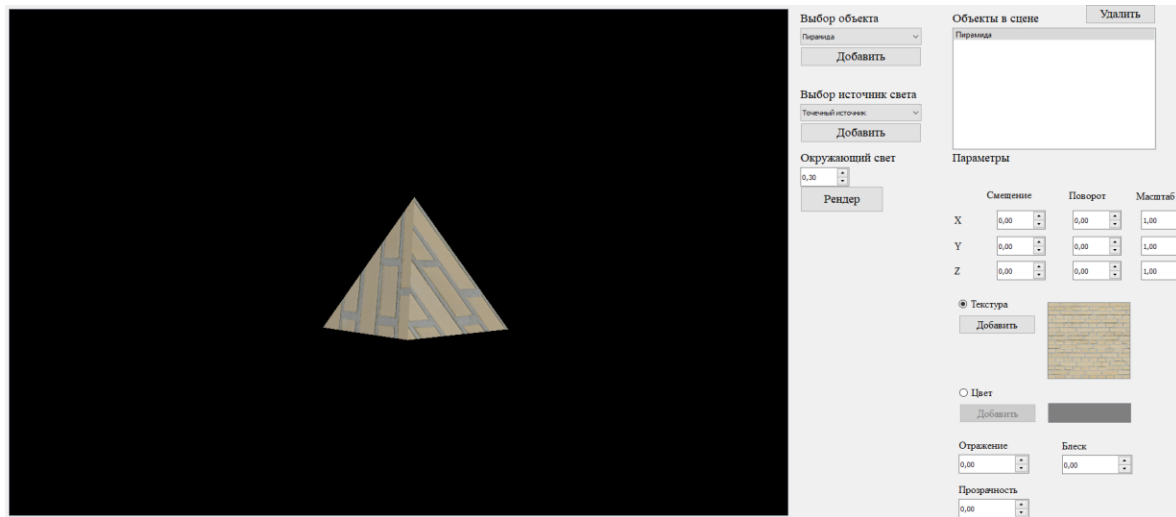
6. Добавление текстуры и изменение цвета объекта

7. Изменение параметров отражения, блеска и прозрачности

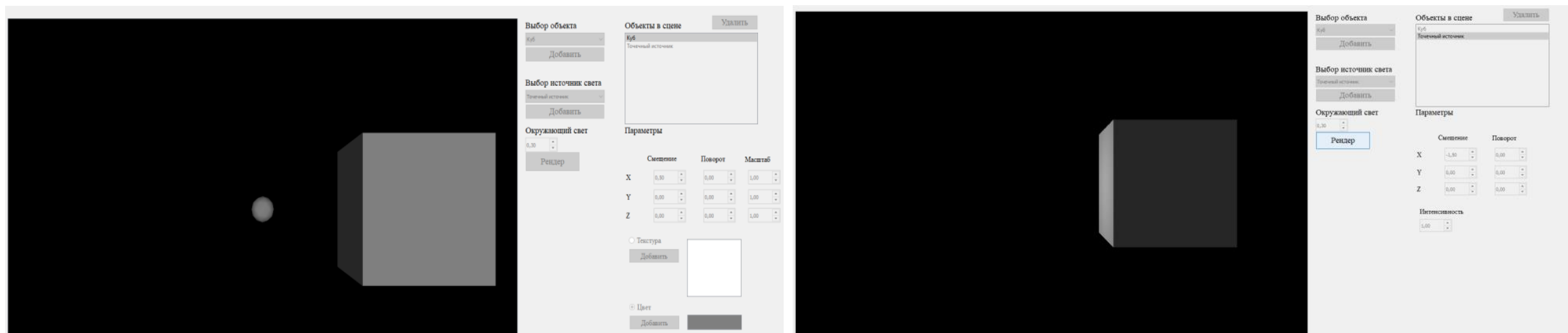
На сцену добавили пирамиду и источник света



Наложение текстуры и цвета



Результат рендеринга



Построение реалистического изображения

