МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Компьютерная графика»

Тема: «Реализация трёхмерного объекта с использованием библиотеки OpenGL»

Студент гр. 7381 _____ Минуллин М.А.

Студентка гр. 7381 _____ Машина Ю.Д.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2019

Цель работы.

Разработать программу, реализующую представление разработанной вами трехмерной сцены с добавлением возможности формирования различного типа проекций теней, используя предложенные функции OpenGL.

Задание.

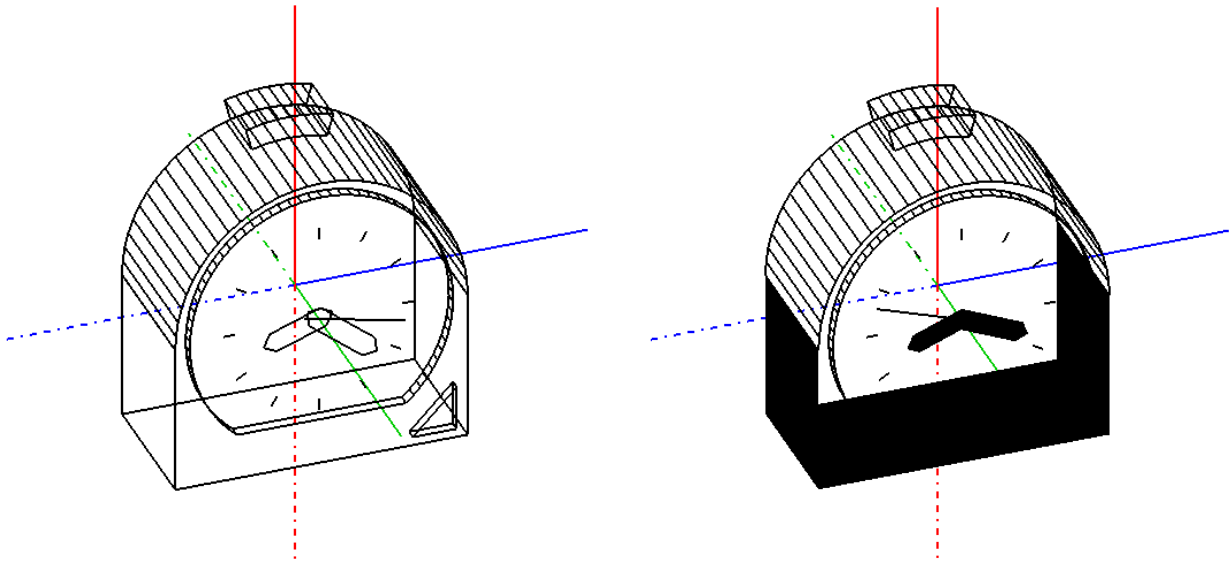
Разработать программу, реализующую представление трехмерной сцены с добавлением возможности формирования различного типа проекций, отражений, используя предложенные функции OpenGL (модель освещения, типы источников света, свойства материалов).

Разработанная программа должна быть пополнена возможностями остановки интерактивно различных атрибутов через вызов соответствующих элементов интерфейса пользователя (замена типа источника света, управление положением камеры, изменение свойств материала модели, как с помощью мыши, так и с помощью диалоговых элементов)

Ход работы.

Насколько мы понимаем, целью для нас в рамках данной лабораторной работы будет создание системы освещения. Очевидно, для демонстрации работы модели света каркасная модель из прошлой работы подходит не очень хорошо. Таким образом, мы принимаем решение заменить каркасную модель на полноценную 3D-модель. Данное решение приводит к необходимости переработать уже созданную модель.

Проблема заключается в том, что для построения каркасной модели помимо полигонов с `gl.glPolygonMode(GL.GL_FRONT_AND_BACK, GL2.GL_LINE);` использовались линии для построения крышки и боковой части циферблата часов. При отключении выше указанного режима, модель приходит в непотребный вид (разница представлена на рис. 1).



а б

Рисунок 1 – Старая модель с включённым режимом отрисовки только границы полигонов (а) и без него (б).

Ожидалось, что вся модель станет чёрного цвета, однако это не так. Крышка строилась линиями, а теперь, чтобы решить эту проблему её также необходимо нарисовать полигонами. Чтобы нарисовать её линиями использовалась функция `glRotated`. Делалось это для того, чтобы не вычислять каждую точку излома крышки в пространстве. Что ж, пришло время произвести эти вычисления самостоятельно.

Пусть даются следующие параметры: центр окружности (x_0, y_0), радиус этой окружности R, две градусные меры α_1 и α_2 и степень разбиения n. Необходимо получить $n+2$ точек $p_i, i = 0; 1; \dots; n; n+1$, принадлежащих заданной окружности таким образом, чтобы они были последовательно равноудалены друг от друга и градусные мера углов xOp_0 и xOp_{n+1} соответствовали углам α_1 и α_2, соответственно.

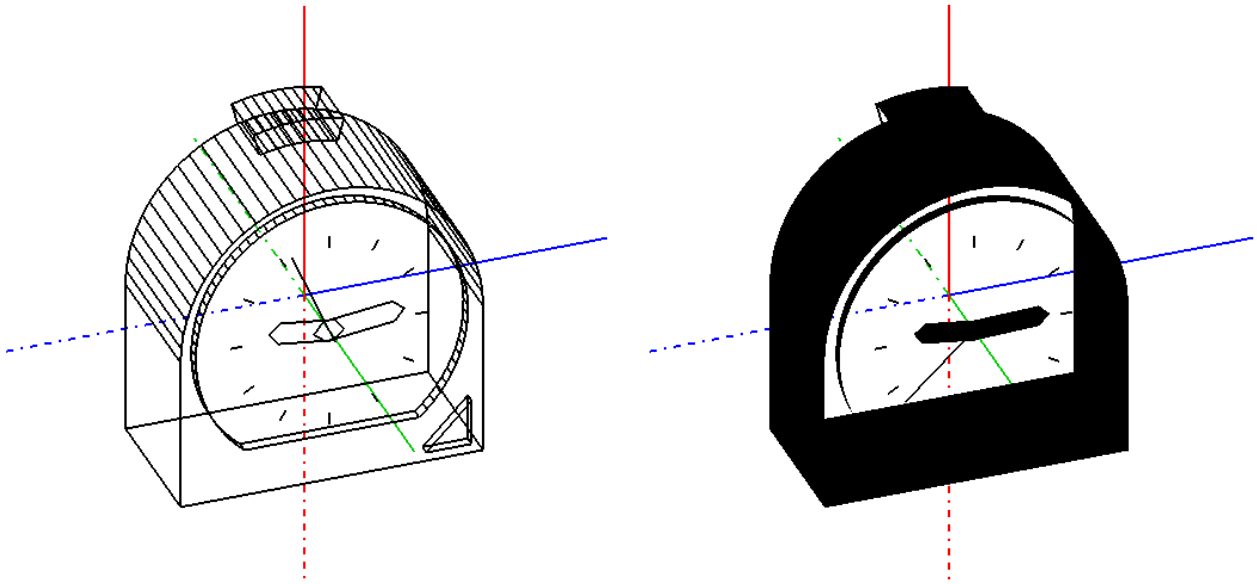
Возвращаемся к школьному курсу геометрии. Определим, насколько осуществляется поворот вектора, координаты конца которого мы будем считать от α_1 до α_2: $\alpha = \alpha_2 - \alpha_1$. Теперь определим градусную меру угла, на который необходимо повернуть угол: $\Delta\alpha = \frac{\alpha}{n+1}$. Теперь остаётся посчитать сами точки:

$$(x_i; y_i) = (x + R\cos\alpha_i\,; y + R\sin\alpha_i), \alpha_i = \alpha_1 + i\Delta\alpha, i = 0; 1; \dots; n; n+1$$

Но этих точек также окажется недостаточно, поскольку нам необходимо сформировать что-то вроде ленты для того, чтобы покрыть часы крышкой. В дано решённой ранее задачи необходимо добавить для окружности вторую идентичную окружность. Однако теперь окружности будут находиться на разной глубине: z_1 и z_2. Точки станут пространственными и считаться парами:

$$\{(x + R\cos\alpha_i\,; y + R\sin\alpha_i\,; z_1); (x + R\cos\alpha_i\,; y + R\sin\alpha_i\,; z_2)\}$$

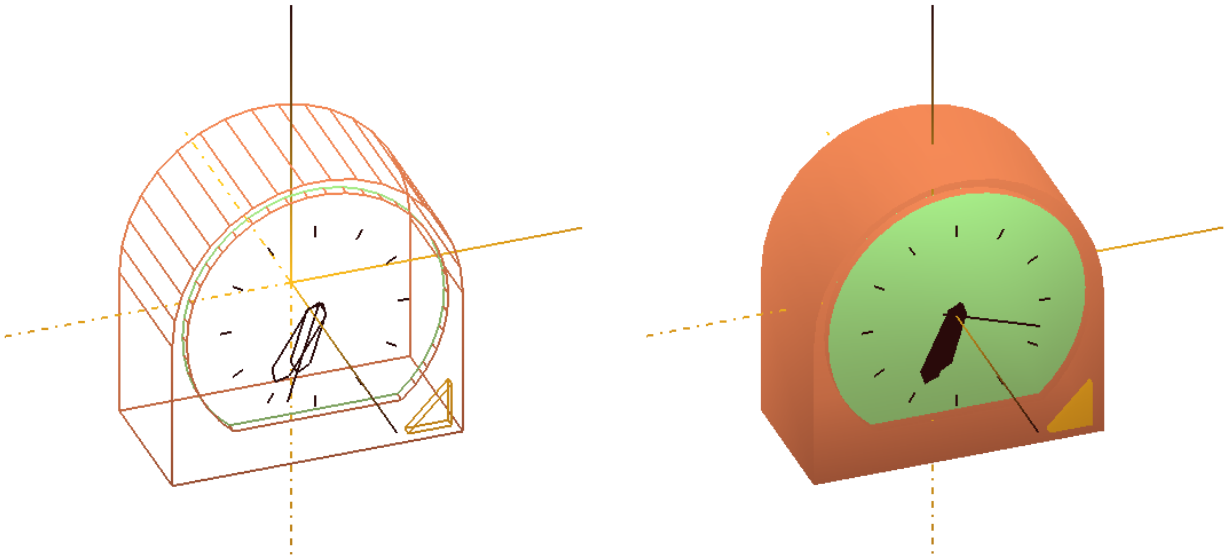
Такое дополнение позволит рисовать крышку и боковую сторону циферблата с помощью примитива `GL_QUAD_STRIP`. Полученный результат представлен на рис. 2.



а б

Рисунок 2 – До и после отключения режима каркасной отрисовки в обновлённой модели будильника.

Мы склонны считать такой вариант ближе к истине, но ещё не окончательным. Осталось решить вопрос с задней стенкой, передней и циферблатом. Однако ответ на этот вопрос в общем остаётся прежним, скроем его в деталях реализации. Окончательная переделанная модель представлена на рис. 3.



а б

Рисунок 3 –До и после отключения режима каркасной отрисовки в модели без примитива «линия» (за исключением секундной стрелки.

Как это получилось?

Включаем режим работы источников света:

```
gl.glEnable(GL2.GL_LIGHTING);
```

Используем всего один источник, его и включаем:

```
gl.glEnable(GL2.GL_LIGHT0);
```

Для света важно иметь значения нормалей поверхности. Поскольку в процессе масштабирования, вектор нормали может получиться не единичной длины, то необходимо включить режим автоматической нормализации. Подробнее на RSDN: http://rsdn.org/?article/?opengl/ogltutor.xml.

```
gl.glEnable(GL2.GL_NORMALIZE);
```

Размещаем источник света, подвешенный со стороны наблюдателя, на уровне верхушки будильника по высоте.

```
float[] positionLight = { 0.0f, 1.0f, 2.0f, 1.0f };
gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_POSITION, positionLight, 0);
```

Задаём направление для источника света в сторону центра мировых координат.

```
float[] directionLight = { 0.0f, -1.0f, -2.0f };
```

```
gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_SPOT_DIRECTION, directionLight, 0);
```

Задаём цвет фонового освещения (розовый оттенок).

```
float[] ambientLight = { 0.4f, 0.f, 0.f, 0.0f };
gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_AMBIENT, ambientLight, 0);
```

Задаём интенсивность света (значение от 0 до 128, 0 – рассеянный свет).

```
gl.glLightf(GL2.GL_LIGHT0, GL2.GL_SPOT_EXPONENT,0.0f);
```

Задаём максимальный угол разброса света (конусовидная область с вершиной в источнике и высотой вдоль направления, параметр от 0 – луч до 180 – максимально рассеянный свет). В нашем случае угол в 45 градусов.

```
gl.glLightf(GL2.GL_LIGHT0, GL2.GL_SPOT_CUTOFF, 45.0f);
```

Чтобы с теневой стороны модель не была просто тёмно-серой, задаём параметр модели освещения GL_LIGHT_MODEL_AMBIENT. По умолчанию он равен (0.2; 0.2; 0.2; 1.0). В нашем случае слегка красноватый.

```
float[] lightModelAmbient = { 0.8f, 0.2f, 0.2f, 0.3f };
gl.glLightModelfv(GL2.GL_LIGHT_MODEL_AMBIENT, lightModelAmbient, 0);
```

На этом этапе картинка получается следующей:

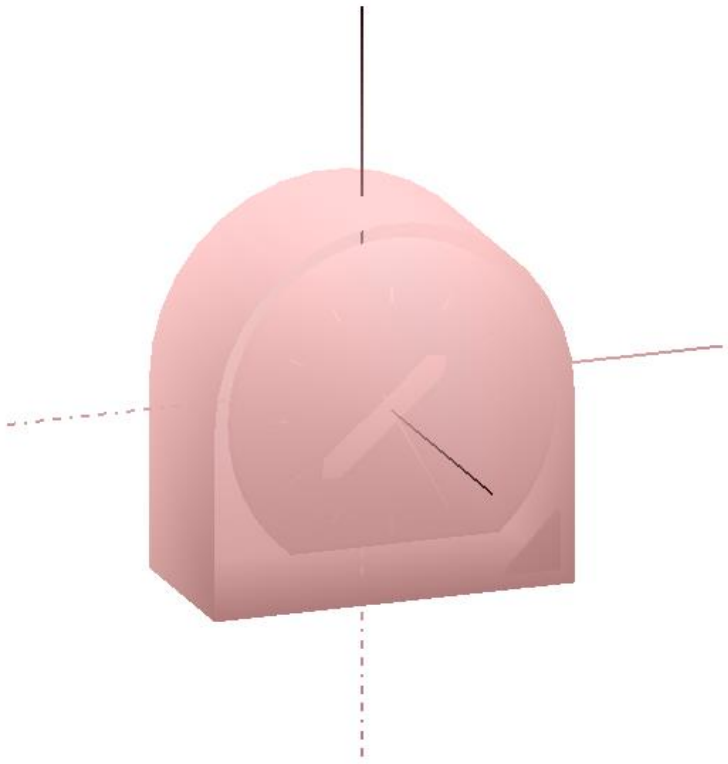


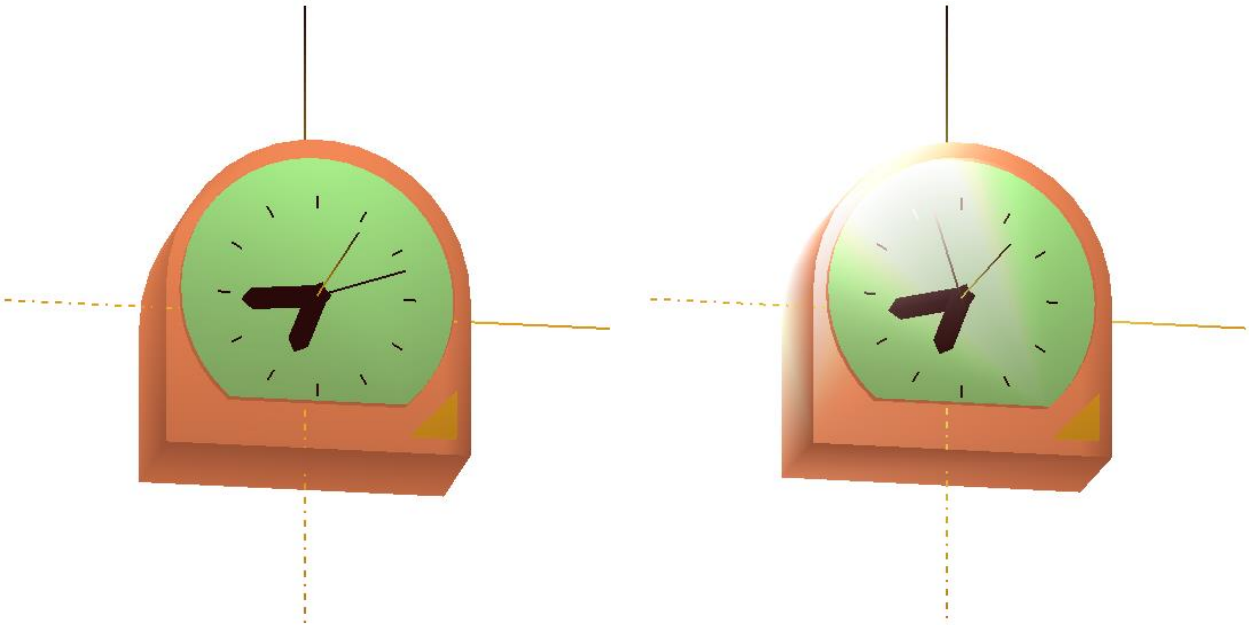
Рисунок 7 – Результат без настроенных материалов.

Система освещения игнорирует все цвета, которые задавались для полигонов. Поэтому вся модель освещена розовым светом. Для того, чтобы сохранить исходные цвета модели, необходимо настроить материалы, из которых она сделана. Поэтому были заданы цвета диффузного отражения материала. Всего 4 материала: для циферблата, кнопки на передней стороне будильника, корпуса и стрелок (указаны в порядке определения).

```
float[] mat_diffuse1 = { 0.5f, 0.9f, 0.5f, 1.0f };
float[] mat_diffuse2 = { 1.0f, 0.8f, 0.1f, 1.0f };
float[] mat_diffuse3 = { 0.8f, 0.5f, 0.3f, 1.0f };
float[] mat_diffuse4 = { 0.0f, 0.0f, 0.0f, 1.0f };
// циферблат
gl.glMaterialfv(GL.GL_FRONT, GL2.GL_DIFFUSE, mat_diffuse1, 0);
drawFace(gl);

// кнопка на панели
gl.glMaterialfv(GL.GL_FRONT, GL2.GL_DIFFUSE, mat_diffuse2, 0);
drawButton(gl);

// корпус
gl.glMaterialfv(GL.GL_FRONT, GL2.GL_DIFFUSE, mat_diffuse3, 0);
drawBox(gl);
drawTape(gl, 0.0, 0.0, -0.5, 0.5, 1.0, 0.0, Math.PI,
triangulationDegree);
drawTape(gl, 0.0, 0.0, 0.42, 0.50, 0.9, Math.toRadians(230.0),
Math.toRadians(-50), 0);
drawTape(gl, 0.0, 0.0, 0.42, 0.50, 0.9, Math.toRadians(-50.0),
Math.toRadians(230.0), 2 * triangulationDegree);
drawBack(gl);

// стрелки и отметки на циферблате
gl.glMaterialfv(GL.GL_FRONT, GL2.GL_DIFFUSE, mat_diffuse4, 0);
drawRisks(gl);
drawClockHands(gl);
```

Мы отказались от реализации колокольчика на будильнике, поскольку считаем, что его наличие в данной работе является непринципиальным, однако значительно увеличивает объём кода, необходимого для построения его 3D-модели. Так же мы опустили некоторое число промежуточных этапов разработки. В данном случае модель уже покрашена, установлен направленный источник освещения и частично настроены параметры материалов для полигона.

В данный момент модель выглядит тускловато, поэтому добавим блики от источника света в свойства материалов. Полученный результат представлен на рис. 4.



а б

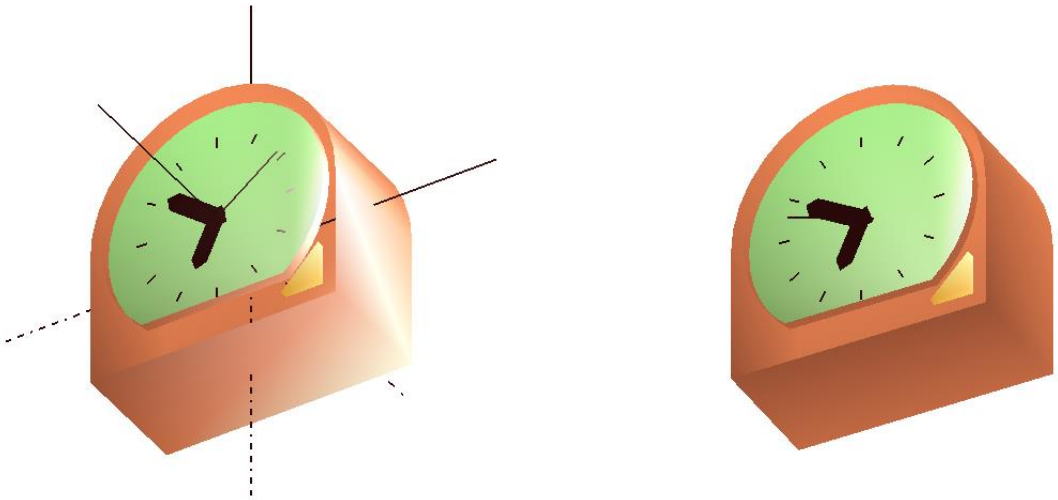
Рисунок 4 – До настройки `GL_SHININESS` материала будильника и после.

Как это получилось?

Для получения эффекта блика на поверхности перед отрисовкой будильника была указана (одна для всех материалов) степень зеркального отражения материала:

```
gl.glMaterialf(GL.GL_FRONT, GL2.GL_SHININESS, 100.0f);
```

Теперь постараемся сделать поверхность будильника матовой, оставив блики только на стекле часов. Полученный результат представлен на рис. 5.



а б

Рисунок 5 – Теперь зеркальное отражение выставлено только на стекле часов и кнопке выключения будильника.

Как это получилось?

Теперь для материалов корпуса и стрелок была уменьшена степень зеркального отражения материала.

```
gl.glMaterialf(GL.GL_FRONT, GL2.GL_SHININESS, 5.0f);
```

Теперь хотелось бы добавить эффект свечения жёлтой кнопке справа и немного для циферблата. Так же добавим источнику света рассеянности, чтобы будильник не выглядел сильно тусклым. Результат представлен на рис. 6.

Как это получилось?

Делается это с помощью задания цвета зеркального отражения. Для кнопки и циферблата был указан тот же цвет, что и для цвета диффузного отражения:

```
float[] no_specular = { 0.0f, 0.0f, 0.0f, 1.0f };
float[] mat_specular = { 1.0f, 1.0f, 1.0f, 1.0f };
// Для кнопки и циферблата
gl.glMaterialfv(GL.GL_FRONT, GL2.GL_SPECULAR, mat_specular, 0);
// Для корпуса и стрелок
gl.glMaterialfv(GL.GL_FRONT, GL2.GL_SPECULAR, no_specular, 0);
```

Рисунок 6 – Итоговый результат работы.

Выводы.

В ходе выполнения лабораторной работы была переработана 3D-модель из предыдущей лабораторной работы. Был добавлен источник освещения. Были настроены материалы модели. Были получены навыки работы с соответствующими функциями из OpenGL. Проект был залит на GiHub – https://github.com/Fikafusik/opengl-3d-clock-lighting.