|  |  |
| --- | --- |
|  | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение высшего профессионального образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова» (БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-02 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | О |  | Естественнонаучный |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | | | Компьютерная геометрия и графика |

КУРСОВАЯ РАБОТА

на тему

|  |
| --- |
| Разработка трёхмерной сцены «Часы» |
| с использованием библиотеки OpenGL |
|  |

|  |  |
| --- | --- |
| Выполнил студент группы | И506Б |
| Петров А.Е. |  |
| Фамилия И.О. |  |
| **РУКОВОДИТЕЛЬ** | |
| Снижко Е.А. | |
| Фамилия И.О. Подпись | |
| Оценка | |
| « » | 20 23 г. |

**СОДЕРЖАНИЕ**

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 3](#_Toc124133427)

[ВВЕДЕНИЕ 4](#_Toc124133428)

[1 Описание разработанной программы 5](#_Toc124133429)

[1.1 Постановка задачи 5](#_Toc124133430)

[1.2 Реализация программы 5](#_Toc124133431)

[1.2.1 Моделирование сцены 5](#_Toc124133432)

[1.2.2 Внешние ресурсы 6](#_Toc124133433)

[1.2.3 Анимирование 9](#_Toc124133434)

[1.2.4 Освещение 10](#_Toc124133435)

[1.2.5 Камера 10](#_Toc124133436)

[1.2.6 Обработка пользовательского ввода 11](#_Toc124133437)

[2 Тестирование 13](#_Toc124133438)

[ЗАКЛЮЧЕНИЕ 16](#_Toc124133439)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc124133440)

[ПРИЛОЖЕНИЕ А Исходный текст программы 18](#_Toc124133441)

**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

В пояснительной записке применяются следующие термины с соответствующими определениями:

GUI (graphical user interface) — система средств для взаимодействия пользователя с электронными устройствами, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана

Сериализация — процесс перевода структуры данных в битовую последовательность. Обратной к операции сериализации является операция десериализации (структуризации) — создание структуры данных из битовой последовательности

**ВВЕДЕНИЕ**

Целью курсовой работы является разработка трехмерной сцены «Часы».

Для достижения поставленной цели необходимо было решить следующие задачи:

* создать составной объект часов с использованием графических возможностей библиотеки OpenGL в соответствии с реальной моделью;
* реализовать наложение текстур на составляющие части часов;
* разработать алгоритмы анимации перемещения стрелок часов;
* реализовать систему освещения с помощью средств библиотеки OpenGL;
* разработать модуль камеры для визуального взаимодействия с трехмерной моделью часов;
* разработать алгоритмы управления и взаимодействия со сценой, ее объектами и камерой.

**1 Описание разработанной программы**

**1.1 Постановка задачи**

Сцена должна состоять из часов (составной объект, к циферблату должны быть присоединены секундная, минутная и часовая стрелки) и нескольких источников света. Стрелки должны корректно отображать время, положение часовой стрелки должно зависеть от положения минутной стрелки, полный оборот секундной стрелки приводит в движение минутную. У пользователя должна быть возможность взаимодействовать с каждым объектом сцены посредством GUI, в том числе вращение, перемещение, масштабирование, изменение цвета световых источников, изменение скорости симуляции.

**1.2 Реализация программы**

1.2.1 Моделирование сцены

Все объекты сцены имеют схожую структуру. Объекты описываются набором компонентов и набором дочерних объектов. Выбранная сцена состоит из 6 объектов: камера, источник света, циферблат, три стрелки. Для стрелок родительским объектом будет циферблат, следовательно их модельная матрица будет получена путём перемножения на модельную матрицу циферблата.

Для отрисовки всех объектов используется один и тот же шейдер, который проверят наличие материалов и текстур, привязанных к объекту, и корректно обрабатывает свет.

Загрузка объектов сцены осуществляется при помощи разбора JSON файла, в котором хранится описание сцены, UML модель класса объекта сцены отображена на рисунке 2. Для работы с JSON форматом была подключена библиотека nlohman-json, при помощи которой сериализуются и десериализуются объекты.

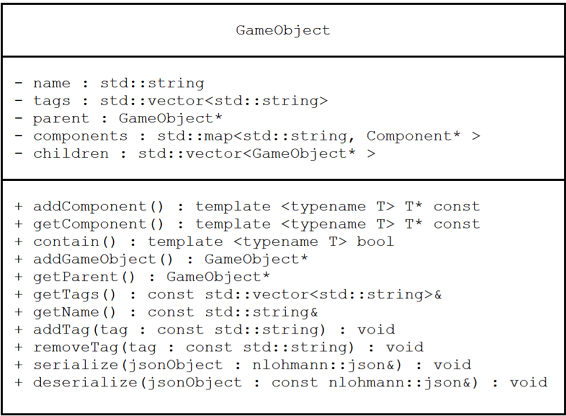


Рисунок 2 – Класс GameObject

1.2.2 Внешние ресурсы

Для работы со внешними ресурсами, такими как шейдеры, модели и текстуры, был разработан статический класс-фабрика ResourceManager. Класс предоставляет методы для создания и загрузки из вне объектов, описанных ранее. Во избежание утечек памяти взаимодействие с объектами ресурсов осуществляется через методы класса ResourceManager. UML схема разработанного класса продемонстрирована на рисунке 3.

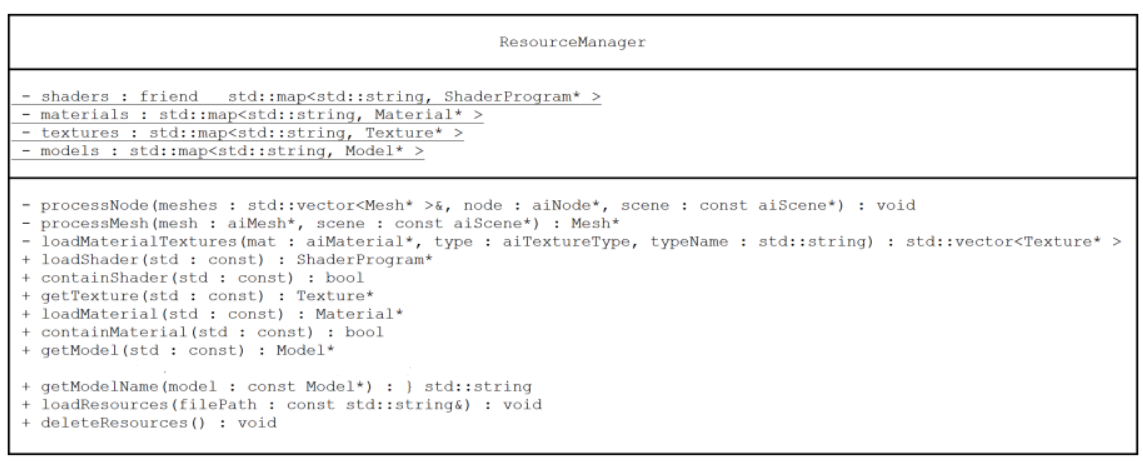


Рисунок 3 – Менеджер ресурсов

1.2.2.1 Шейдеры

Разработанным приложением предусмотрена загрузка шейдера во время выполнения программы. Для создания объекта класса ShaderProgram, необходимо указать пути до файлов, содержащих текст вершинного и фрагментного шейдеров. Для считывания файла шейдера используются стандартные потоки C++, помещая результат в строки [1]. Класс ShaderProgram предоставляет возможность посредством внутренних методов передавать в шейдер параметры различных типов.

1.2.2.2 Текстуры

Для загрузки текстур была подключена библиотека STBI\_Image. Программой предусмотрена настройка загружаемой текстуры (выбор фильтрации и способа усекания).

В качестве фильтрации по умолчанию был выбран метод фильтрования GL\_NEAREST. Пока он установлен, OpenGL будет выбирать пиксель, который находится ближе всего к текстурной координате.

В случаях, когда текстурные координаты выходят из промежутка (0,0) и (1,1), поведение OpenGL зависит от параметра усекания текстуры (texture wrapping). По умолчанию параметр имеет значение GL\_REPEAT, пока он установлен OpenGL будет повторять текстуру [2].

Программой предусмотрено разделение текстур на типы (диффузный и бликовый). Это даёт возможность влиять на диффузный (и, косвенным образом, на фоновый, так как это почти всегда одно и то же) и бликовый компоненты объекта с большей точностью.

Текстуры передаются в шейдер при помощи методов второго.

1.2.2.3 Модели

Для загрузки трёхмерных моделей используется библиотека ASSIMP, при помощи которой данные о модели преобразуются в разработанный класс Model. Класс модели хранит список полигональных сеток (мешей). Каждая сетка хранит в себе списки вершин и индексов, благодаря этим данным генерируются вершинные и индексный буферы [3]. Вершина хранит в себе нормаль, модельные и текстурные координаты. Схема взаимодействия описанных сущностей отображена на рисунке 4.

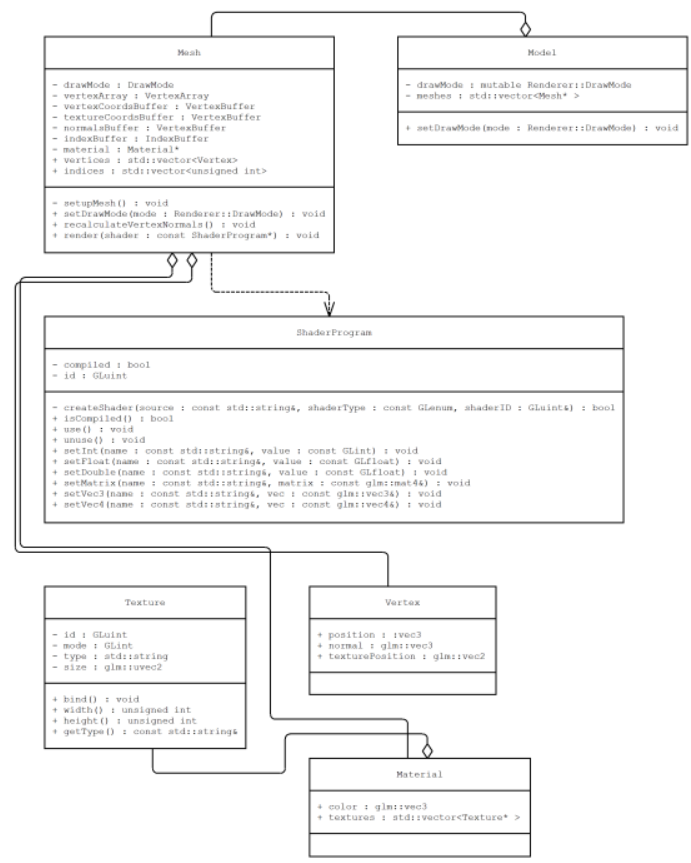


Рисунок 4 – Схема отражающая взаимосвязь классов ресурсов

1.2.3 Анимация

Анимации реализованы через изменение параметров позиции и поворота относительно родительского объекта, и привязаны ко времени, для создания эффекта плавности. Для этого был разработан статический класс Time, который вычисляет разницу между текущим временем и временем выполнения предыдущего программного цикла и сохраняет его в переменную deltaTime, благодаря которой достигается плавность анимации.

1.2.4 Освещение

Логика источника света заключена в компоненте освещения, который обладает характеристиками интенсивности и цвета, которые пользователь может менять при помощи графического интерфейса. Программа поддерживает одновременное использование 16 источников позиционного и направленного света. Текст фрагментного шейдера, отвечающего за вычисление воздействия источника света на объект, продемонстрирован на рисунке 5 [4].

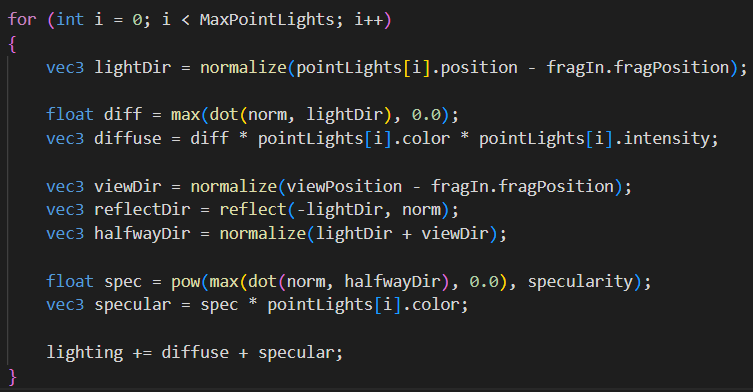


Рисунок 5 – Вычисление света

1.2.5 Камера

Так как все объекты, находящиеся на сцене, имеют схожую структуру, а поведение конкретного объекта определяется набором компонентов, логика камеры была вынесена в отдельный компонент. Программой предусмотрено переключение между проекционной и ортогональной проекциями. Построение проекционной и видовой матриц реализовано через функции библиотеки GLM (OpenGL Mathematics) [5]. Схема архитектурного устройства компонента камеры представлена на рисунке 6. Пользователь имеет возможность перемещать камеру нажатием на клавиши (W/A/S/D), и вращать (←↑→↓).

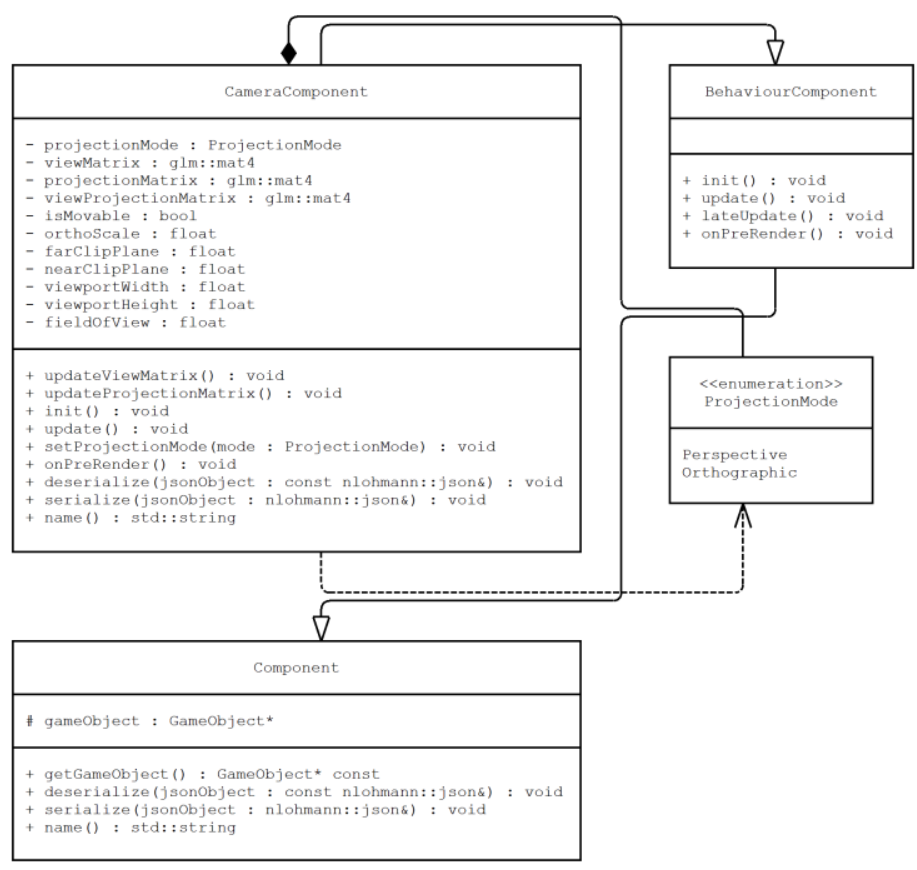


Рисунок 6 – Компонент, описывающий камеру

1.2.6 Обработка пользовательского ввода

Для обработки пользовательского ввода был разработан статический класс-одиночка InputHandler, благодаря которому из любой другой точки программы можно получить информацию о состоянии клавиш или мыши. Методы данного класса связываются с функциями обратного вызова, предоставленными библиотекой GLFW. UML диаграмма класса показана на рисунке 7.

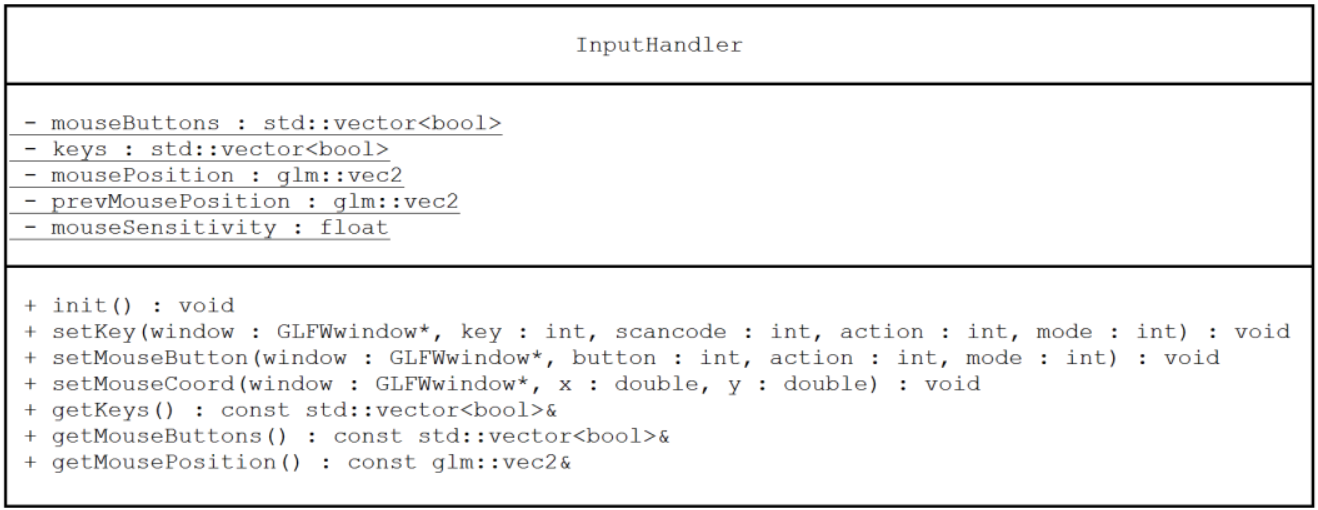


Рисунок 7 – Класс InputHandler

**2 Тестирование**

На рисунке 8 показано окно выбора файла с проектом. Файл должен иметь расширение «json».

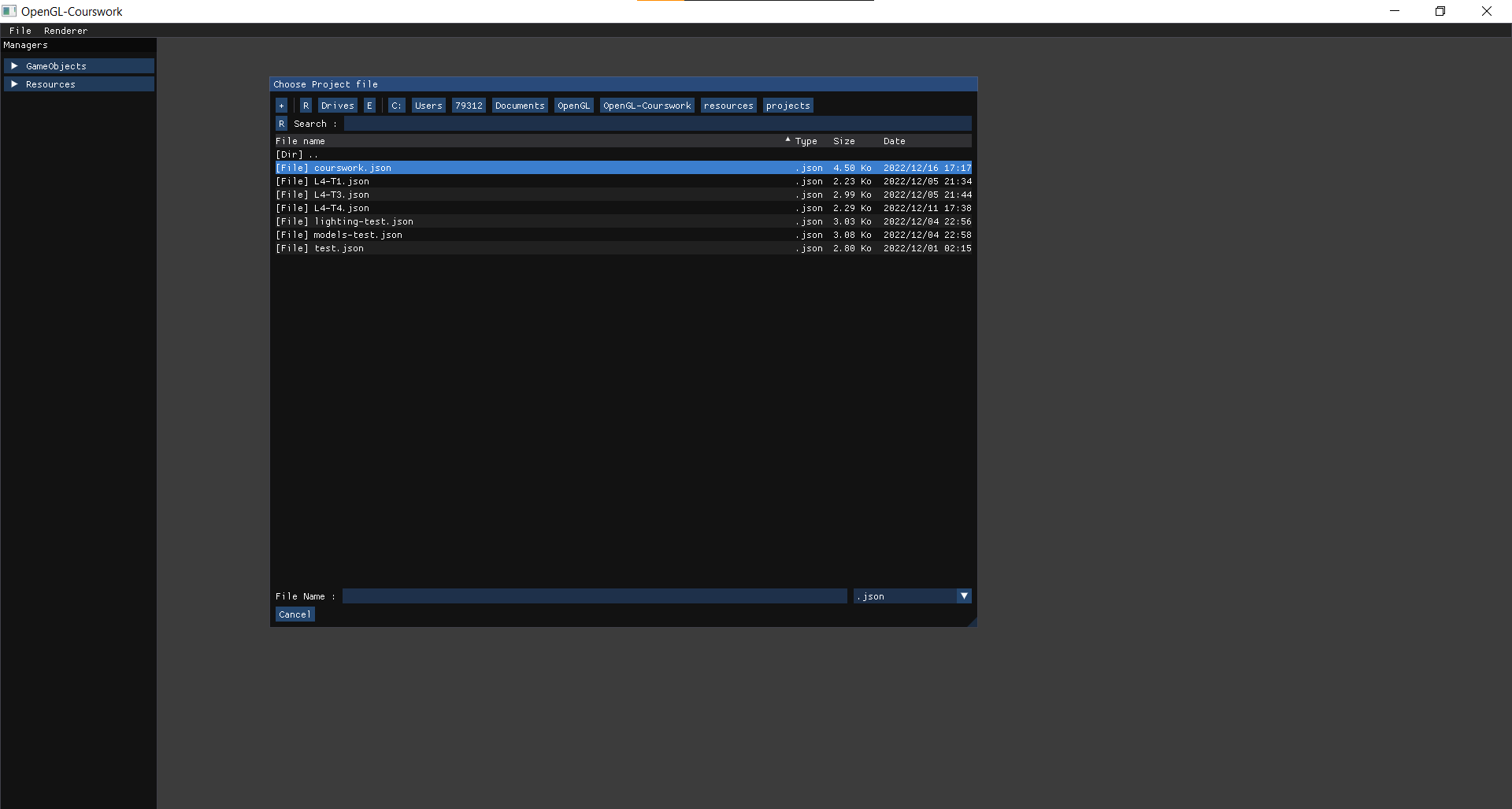


Рисунок 8 – Выбор файла проекта

На рисунке 9 показана загруженная сцена. На рисунке также видно, что слева отображена панель загруженных ресурсов (шейдеры, текстуры, материалы, модели) и игровых объектов, а справа панель выбранного игрового объекта, где можно просмотреть информацию об его полях/компонентах.

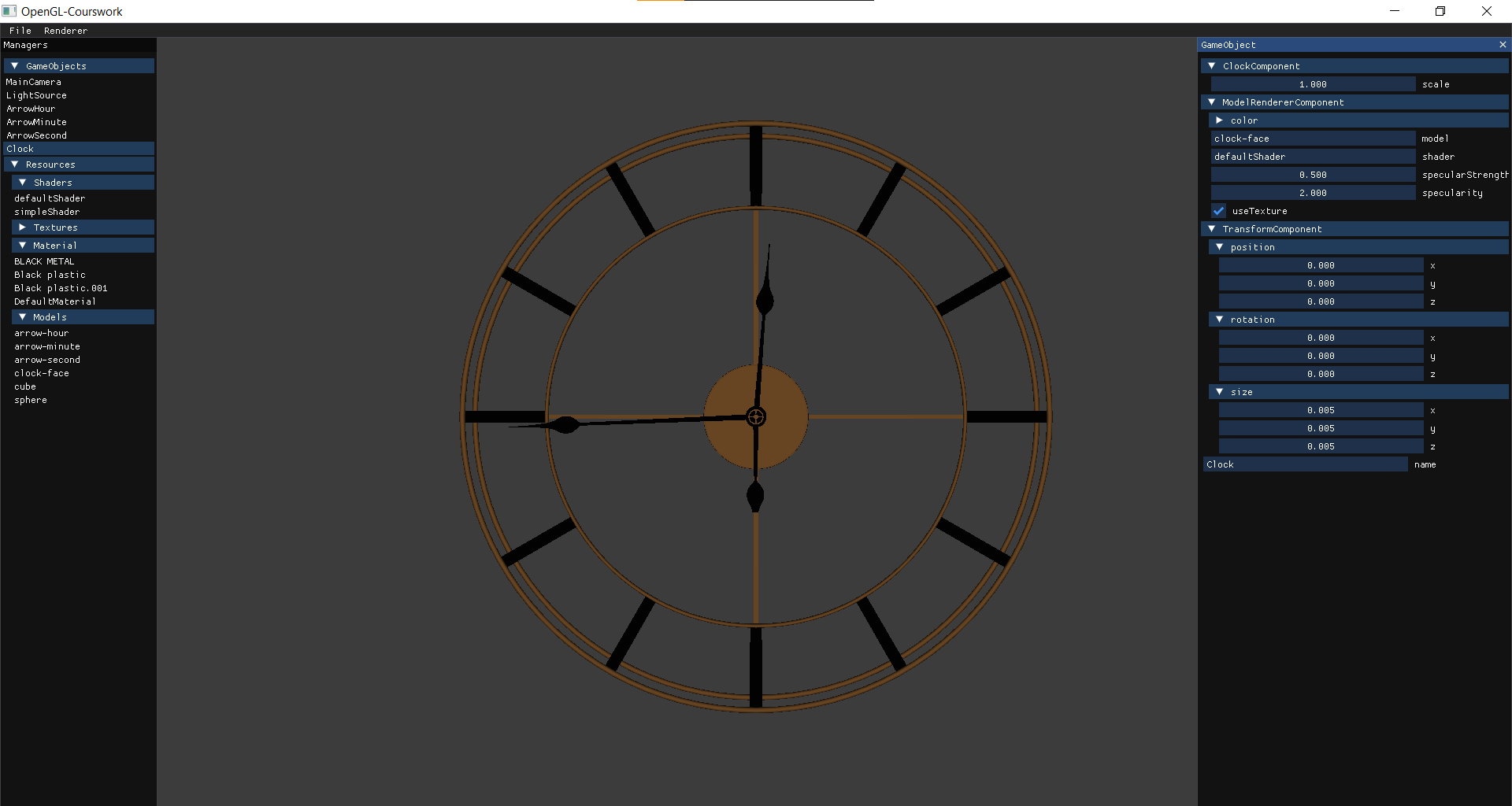


Рисунок 9 – Загруженная сцена

На рисунке 10 показана сцена после переключения камеры на ортогональную проекцию, на панели справа также видны поля объекта камеры

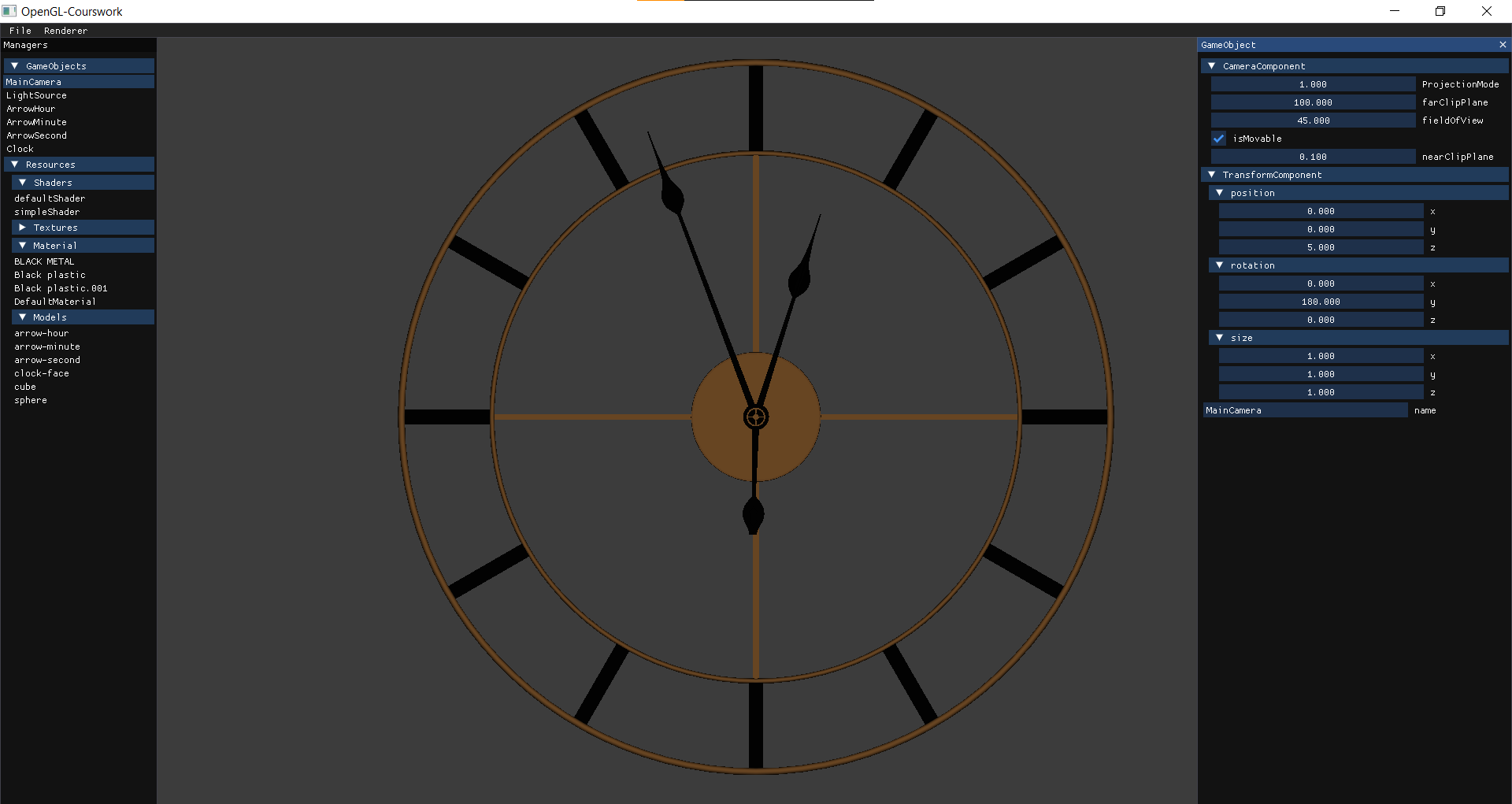


Рисунок 10 – Сцена после изменения параметров камеры

На рисунке 11 показана сцена после перемещения, вращения и масштабирования объекта часов.

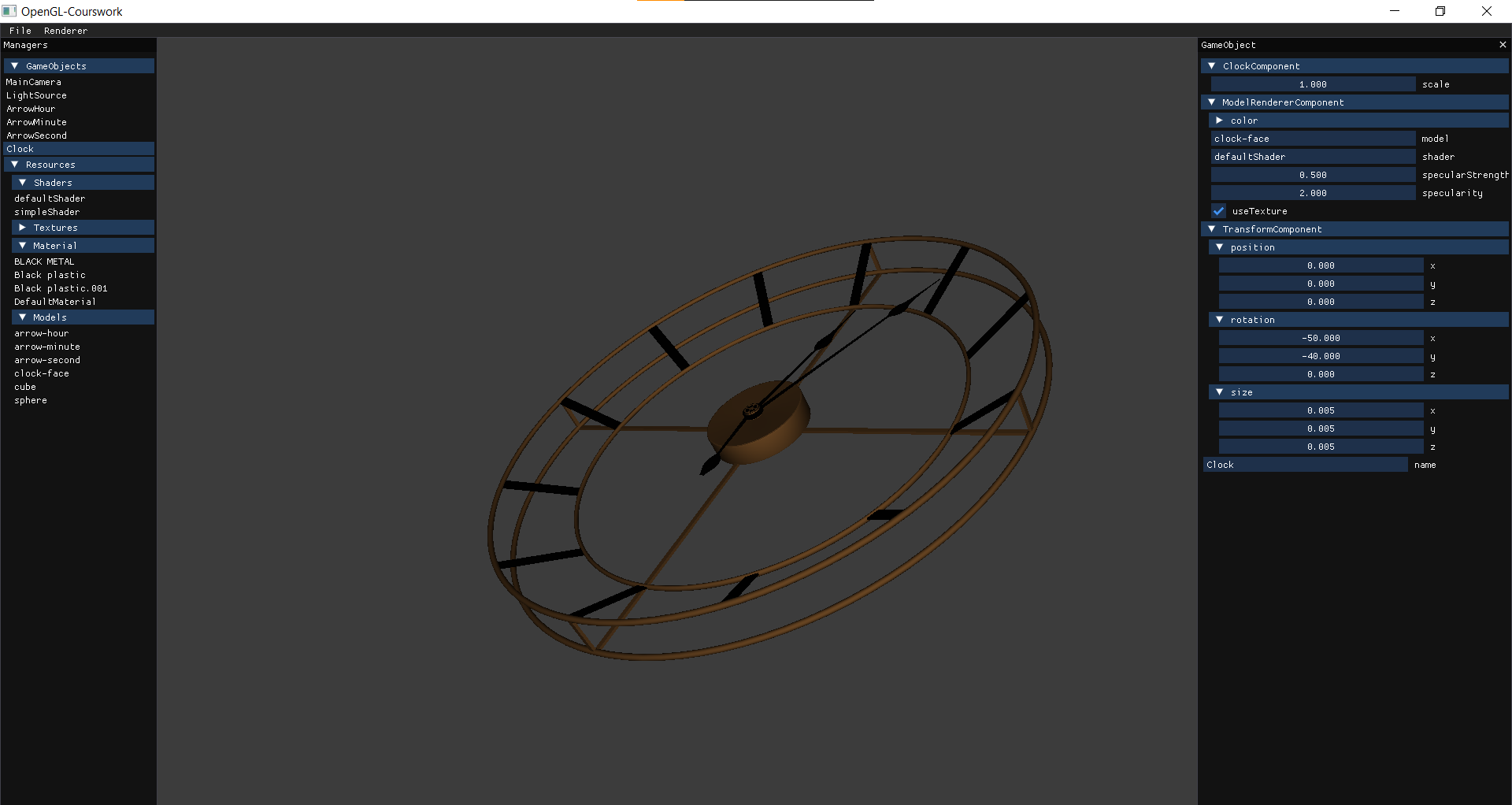


Рисунок 11 – Сцена после изменения полей объекта часов

На рисунке 12 показана сцена после изменения позиции и цвета источника освещения.

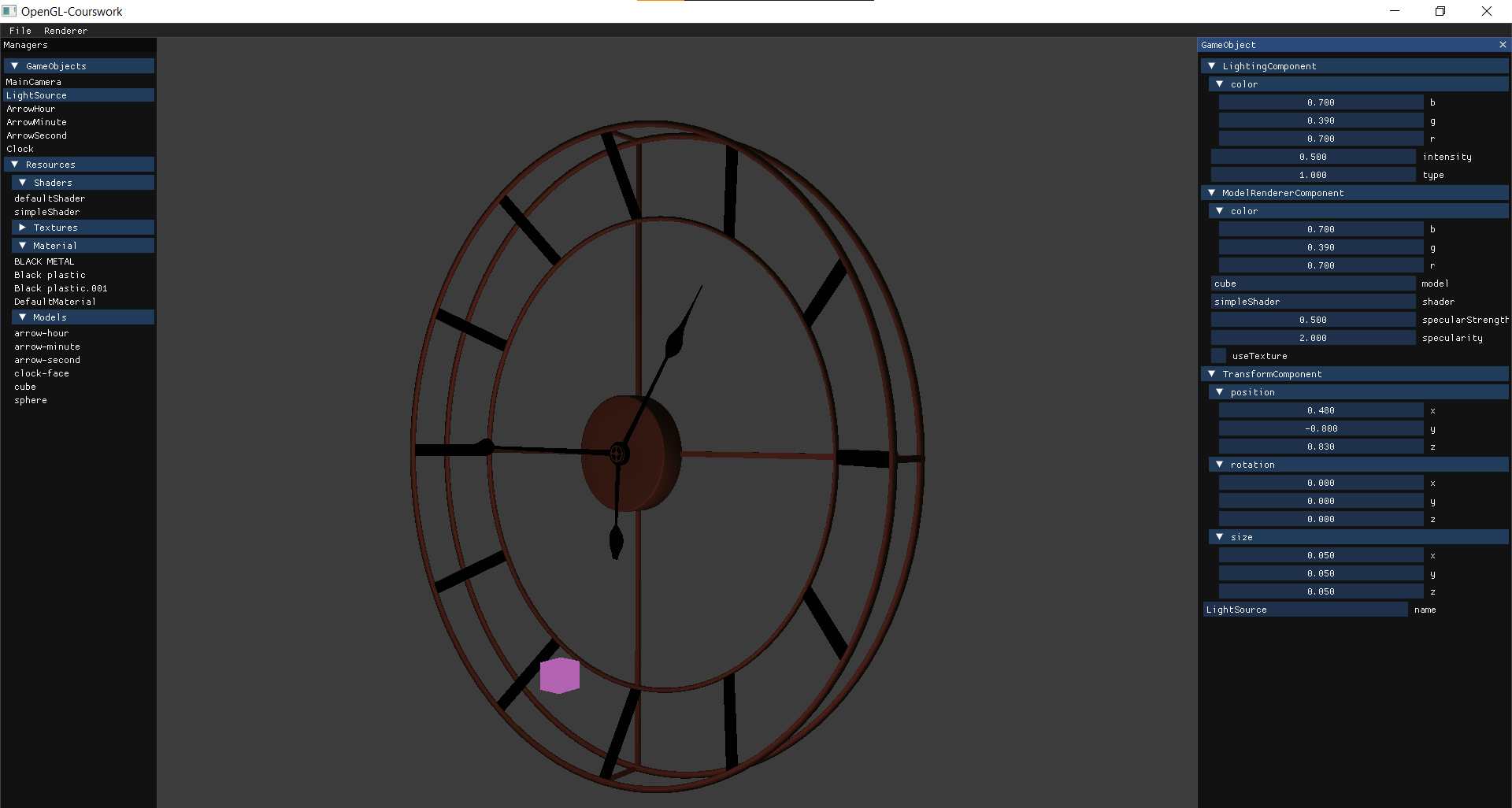


Рисунок 13 – Изменение параметров источника света

**ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы была смоделирована трёхмерная сцена «часы», включающая составные объекты, такие как циферблат и стрелки. Для придания идентичности с реальной моделью часов была разработана система материалов, которая позволяет взаимодействовать с текстурами. Была реализована система множественного освещения и написан соответствующий шейдер. Благодаря компонентной системе в приложение были легко интегрированы анимации и компонент камеры. Для обработки пользовательского ввода была реализована система, использующая функции обратного вызова.

Курсовая работа выполнена в IDE Visual Studio 2019 на языке С++ с использованием графической библиотеки OpenGL 4.6 (GLFW, GLAD, GLM) и сторонних библиотек (ASSIMP, STB\_Image, Dear ImGui, nlohman json).

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. learnopengl. Урок 1.5 — Shaders / Хабр – URL: https://habr.com/ru/post/313380 (дата обращения 29.09.2022).
2. learnopengl. Урок 1.6 — Текстуры / Хабр – URL: https://habr.com/ru/post/315294 (дата обращения 13.10.2021).
3. learnopengl. Урок 3.3 — Класс 3D-модели / Хабр – URL: https://habr.com/ru/post/338998 (дата обращения 16.10.2021).
4. learnopengl. Урок 2.6 — Несколько источников освещения / Хабр – URL: https://habr.com/ru/post/338254 (дата обращения 18.10.2021).
5. learnopengl. Урок 1.9 — Камера / Хабр – URL: https://habr.com/ru/post/327604 (дата обращения 19.10.2021).

**ПРИЛОЖЕНИЕ А  
Исходный текст программы**

Исходные тексты программы располагаются в прилагаемом архиве OpenGL-Courswork.zip.