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

КУРСОВАЯ РАБОТА

по дисциплине «Компьютерная графика»

Тема: Визуализация 3D объекта

Студент гр. 7383 _____ Власов Р.А.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Власов Р.А.

Группа 7383

Тема работы: Визуализация 3D объекта

Исходные данные:

Необходимо реализовать 3D визуализацию каркаса бутылки, показанной на рис. 1.

Содержание пояснительной записки:

- Введение
- Общие теоретические сведения
- Ход работы
- Примеры работы программы
- Заключение
- Список использованных источников

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 9.05.2020

Дата сдачи реферата: 31.05.2020

Дата защиты реферата: 31.05.2020

Студент _____ Власов Р.А.

Преподаватель _____ Герасимова Т.В.

АННОТАЦИЯ

В ходе данно работы разработана программа на языке программирования C++ с использованием библиотеки OpenGL и фреймворка Qt, которая позволяет визуализировать в 3D каркас бутылки.

SUMMARY

In this work, C++ program was developed using OpenGL library and Qt framework. The program visualizes a bottle frame in 3D.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В ходе данной работы необходимо написать программу для визуализации 3D сцены, которая показана на рис. 1, с использованием библиотеки OpenGL. При этом генерация сцены должна осуществляться в программе, нельзя пользоваться сторонними средствами.

Рисунок 1 — 3D сцена

Процесс генерации объекта должен быть описан в работе.

1. ОБЩИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Компьютерная графика — область деятельности, в которой компьютерные технологии используются для создания и обработки изображений и другой визуальной информации.

Сцена — визуальное представление пространства, на котором изображаются все объекты и фон.

Рендеринг — процесс подготовки и прорисовки изображения.

Основной задачей, которая решается при рендеринге изображения, является отображение точек в мировых координатах на двумерную плоскость экрана. Для этого выполняются видовое и перспективное приобразования.

Преобразование объектов осуществляется при помощи операций над матрицами. Такие преобразования, как параллельный перенос, масштабирование и вращение, осуществляются при помощи видовой матрицы. Матрица проекций отвечает за способ проецирования изображения на плоскость экрана. Матрица текстуры определяет наложение текстуры на объект. Все матрицы имеют размерность 4x4.

OpenGL — платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трехмерную компьютерную графику. В спецификации OpenGL содержится описание всех необходимых инструментов для задания видовой матрицы, матриц проекции и текстуры, а также создания примитивов. Удобство спецификации OpenGL заключается в том, что она имеет единый набор функций с одинаковым поведениям на всех платформах, что позволяет использовать ее в программах на любом языке программирования.

2. ХОД РАБОТЫ

Работа выполнения на языке программирования C++ с использованием библиотеки OpenGL. Для реализации графического интерфейса был применен фреймворк Qt.

Для отображения сцены был создан класс GLWidget, в качестве родительского класса был взят QOpenGLWidget, позволяющий удобно отобразить сцену в окне Qt. Окно программы открывается в полноэкранном режиме.

Для создания сцены был переопределен метод paintGL, в котором вызывается функция генерации объекта, а также выполняются все вспомогательные операции для освещения сцены и отображения вспомогательных объектов.

Управление осуществляется интерактивно с использованием мыши или клавиатуры. Для управления положением камеры с помощью мыши были реализованы методы mousePressEvent, mouseMoveEvent, mouseReleaseEvent и wheelEvent. Для управления положением камеры с помощью клавиатуры (клавиш WASD) были реализованы методы keyPressEvent и keyReleaseEvent. Клавиша Escape закрывает программу.

Генерация 3D объекта осуществляется по частям. Объект можно разбить на 3 основные части:

- 6 цилиндров, составляющих основу объекта;
- 3 тора и ряд цилиндров, которые их удерживают;
- 4 изогнутые рамки вокруг объекта.

Для создания рамок, в свою очередь, также применялось разбиение на прямые части и части с изгибом, которые вычислялись с применением разных матемематических формул для достижения различной формы. Их удобно генерировавать с помощью цилиндров. Торы также удобно генерируются с помощью цилиндров.

Данное разбиение объекта говорит о том, что его удобно генерировать с помощью большого количества цилиндров. Такие функции библиотеки OpenGL, как glPushMatrix, glPopMatrinx, glTranslatef, glRotatef,позволяют располагать каждый объект независимо от остальных, что сильно упрощает процесс генерации объекта.

Код функции, генерирующей объект, приведен в приложении А.

Для создания освещения к объекту был применен GL_COLOR_MATERIAL. Для указания нормалей к поверхностям была использована функция glNormal3f. Также в углу экрана размещена кнопка, открывающая окно с выбором цвета освещения.

При создании сцены было реализовано отражение. Для этого была построена перевернутая копия предмета. Для создания эффекта отражения, так как библиотека OpenGL не поддерживает прозрачность, к платформе между объектами был применен GL_BLEND с функцией glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA). Использование данного приема работает только в случае, если объекты с эффектом прозрачности рендерятся только после всех объектов, расположенных за ними. Такой порядок позволяет обеспечить GL_DEPTH_TEST.

3. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Тестирование программы проводилось в операционной системе Ubuntu 19.04 с использованием компилятра g++ (Ubuntu 8.3.0-6ubuntu1) 8.3.0. В других системах тестирование не проводилось.

На рис. 2-9 приведены скриншоты работы программы.

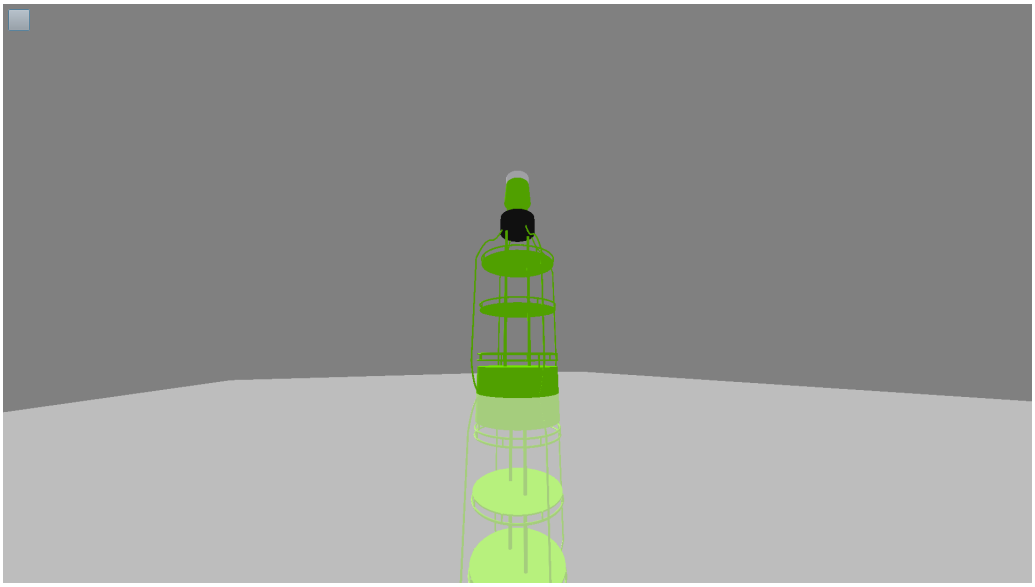

Рисунок 2 — Объект снизу

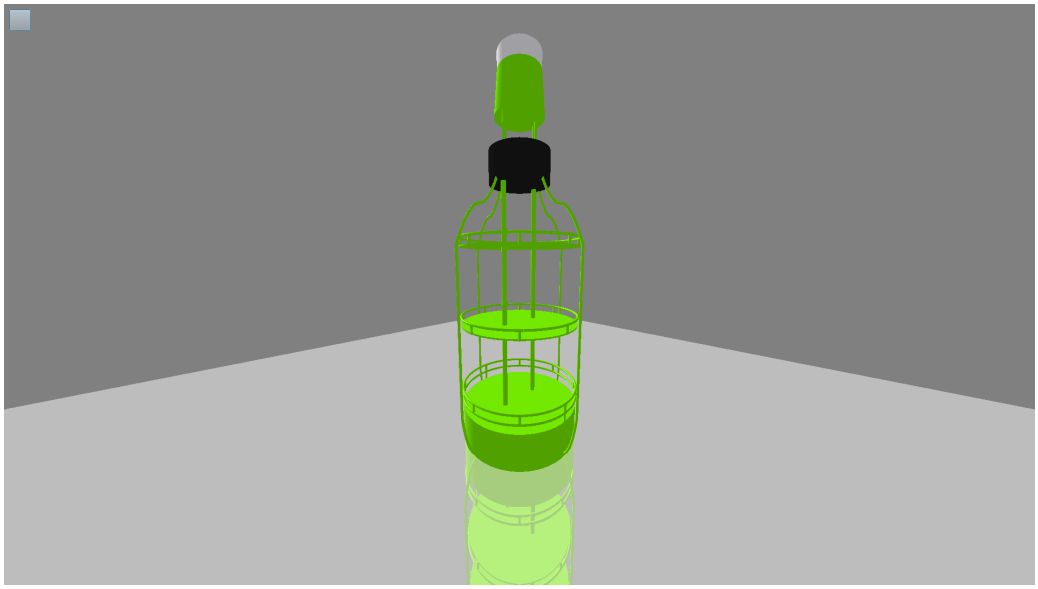

Рисунок 3 — Объект в профиль

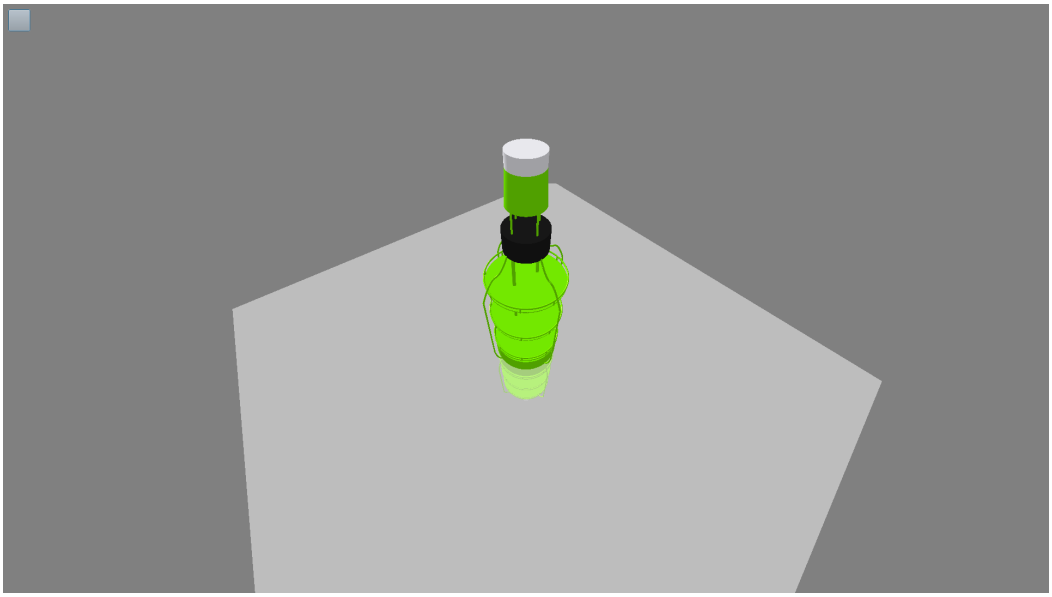

Рисунок 4 — Объект сверху

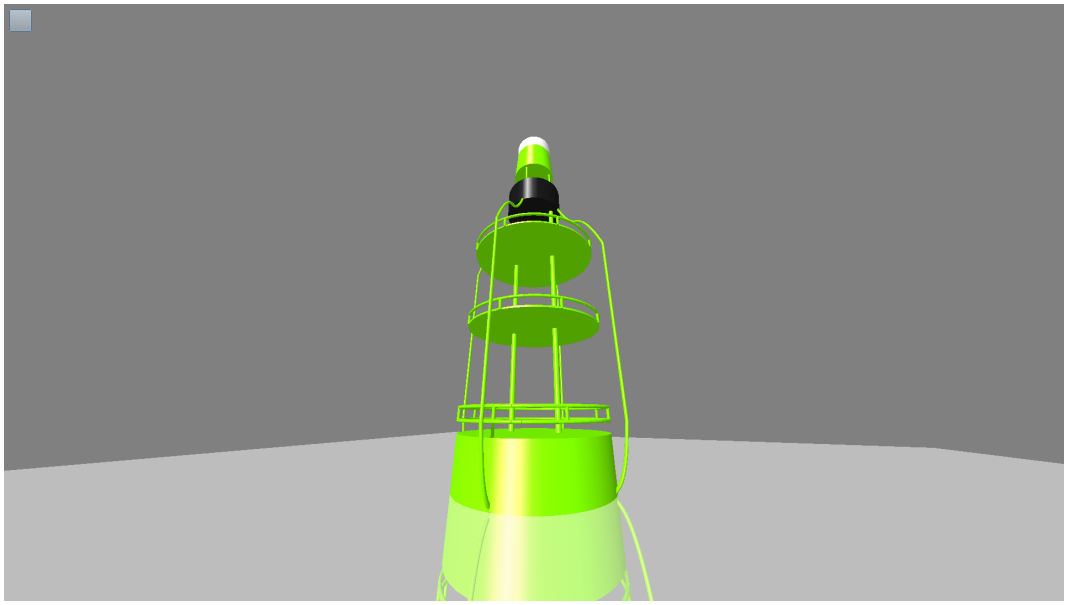

Рисунок 5 — Объект от источника освещения

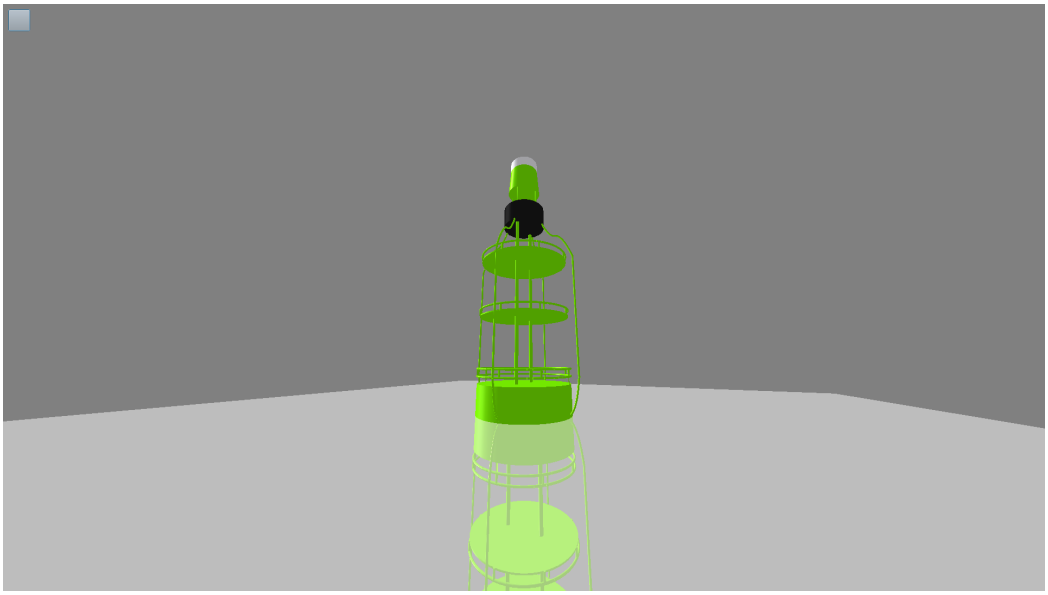

Рисунок 6 — Объекта перпендикулярно источнику освещения

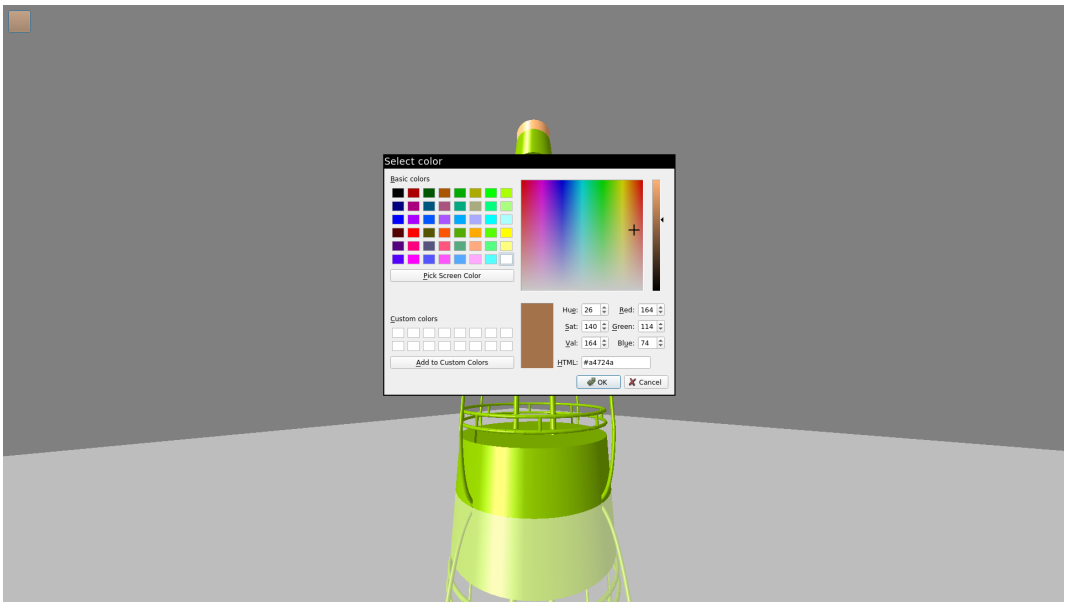

Рисунок 7 — Окно выбора цвета освещения

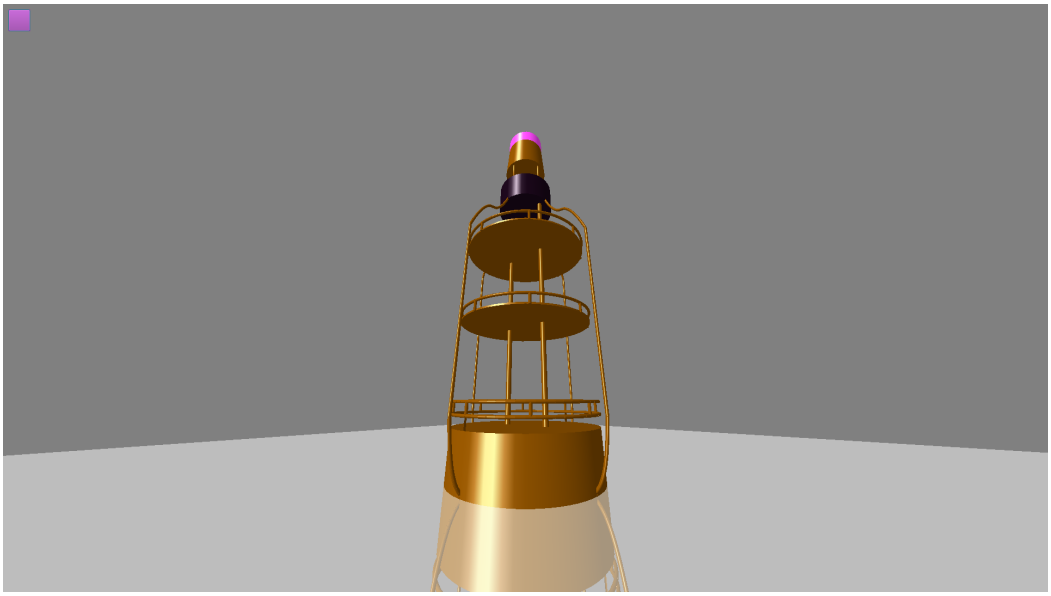

Рисунок 8 — Объект в фиолетовом свете

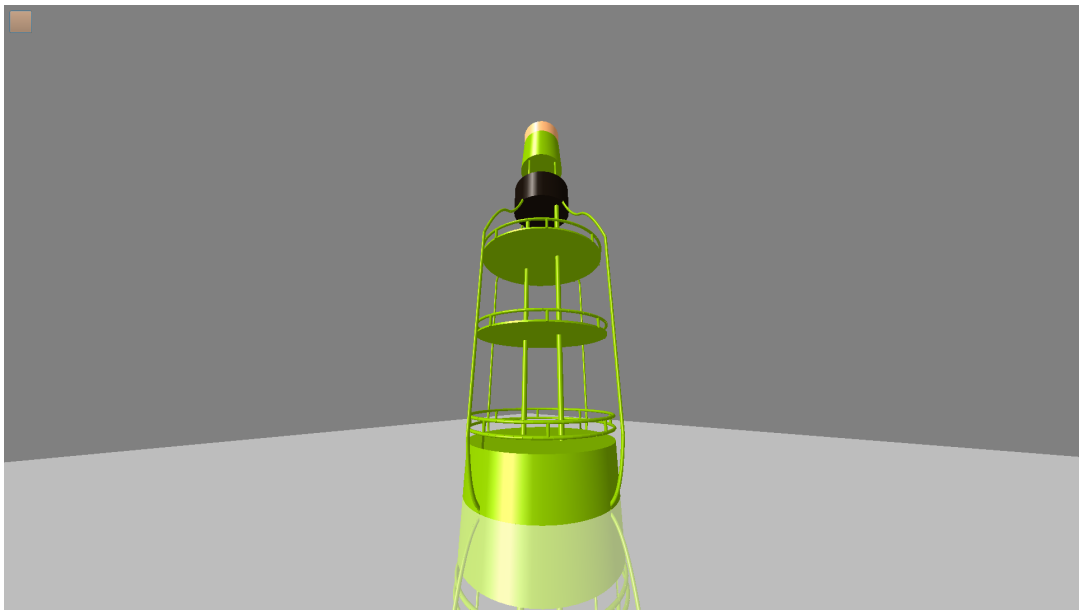

Рисунок 9 — Объект в коричневом свете

В ходе тестирования ошибок выявлено не было, все функции работают корректно.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы была реализована программа на языке программирования C++ с использованием библиотеки OpenGL для визуализации 3D сцены, в качестве которой выступала модель каркаса бутылки. Была реализована возможность вращения камеры вокруг объекта, а также выбора цвета освещения. Также на сцене был реализован эффект отражения объекта от платформы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация OpenGL // OpenGL API Documentation URL:https://www.opengl.org/documentation/ (Дата обращения: 26.05.2020)

2. Источники света // Компьютерная графика URL:https://compgraphics.info/OpenGL/lighting/light_sources.php (Дата обращения: 26.05.2020)

3. OpenGL ES 3.0. Руководство разработчика / Гинсбург Д., Пурномо Б., Шрейнер Д., Мунши А. М.: ДМК, 2015. 448 с.

ПРИЛОЖЕНИЕ А
КОД ГЕНЕРАЦИИ ОБЪЕКТА

```
void drawBottle(float pos[3], int num_segments) {
    glPushMatrix();
    glTranslatef(pos[0], pos[1], pos[2]);

    glColor3ub(127,255,0);
    // bottom
    float coords[3] = {0,0,0};
    drawVerticalCylinder(coords, 10, 10, 6, num_segments);
    // bottom torus 1
    drawHorizontalTorus(0, 0, 7.5f, 10, 0.2f, num_segments);
    // bottom torus 2
    drawHorizontalTorus(0, 0, 9, 10, 0.2f, num_segments);
    // bottom columns 1
    drawColumns(0, 0, 6, 10, 0.2f, 1.5, 2);
    // bottom columns 2
    drawColumns(0, 0, 7.5f, 10, 0.2f, 1.5, 8);

    // first platform
    coords[2] = 20;
    drawVerticalCylinder(coords, 10, 10, 0.25, num_segments);
    // first platform torus
    drawHorizontalTorus(0, 0, 21.5f, 10, 0.2f, num_segments);
    // first platform columns
    drawColumns(0, 0, 20, 10, 0.2f, 1.5, 8);

    // second platform
    coords[2] = 33;
    drawVerticalCylinder(coords, 10, 10, 0.25, num_segments);
    // second platform torus
    drawHorizontalTorus(0, 0, 34.5f, 10, 0.2f, num_segments);
    // second platform columns
```

```
    drawColumns(0, 0, 33, 10, 0.2f, 1.5, 8);

    // bottle cap bottom
    coords[2] = 53;
    drawVerticalCylinder(coords, 4, 3.5, 6.5, num_segments);

    // center cylinders
        // first
    float coords1[3] = {0, 3.5, 6};
    drawVerticalCylinder(coords1, 0.4f, 0.4f, 38, num_segments);
        // second
    coords1[1] *= -1;
    drawVerticalCylinder(coords1, 0.4f, 0.4f, 38, num_segments);

    for (int fr = 0; fr < 4; fr++) {
        glPushMatrix();
        glRotatef(90 * fr, 0, 0, 1);
        drawFrame(10, 0.1f, 0.2f, num_segments);
        glPopMatrix();
    }

    glColor3ub(25,25,25);
    // bottle label
    coords[2] = 43;
    drawVerticalCylinder(coords, 5, 5, 5, 32);

    glColor3ub(255,255,255);
    // bottle cap top
    coords[2] = 59.5;
    drawVerticalCylinder(coords, 3.5, 3.5, 2.5, 32);

    glPopMatrix();
}
```