Задание 3.
Трехмерная карта солнечной системы

Авторы задания: Александра Афанасьева, Владимир Афанасьев, Кристина Зипа

Contents

1 Введение

Цель задания - реализовать симуляцию и визуализацию движения небесных тел солнечной системы в 3D, обеспечить интерактивный режим навигации по ней.

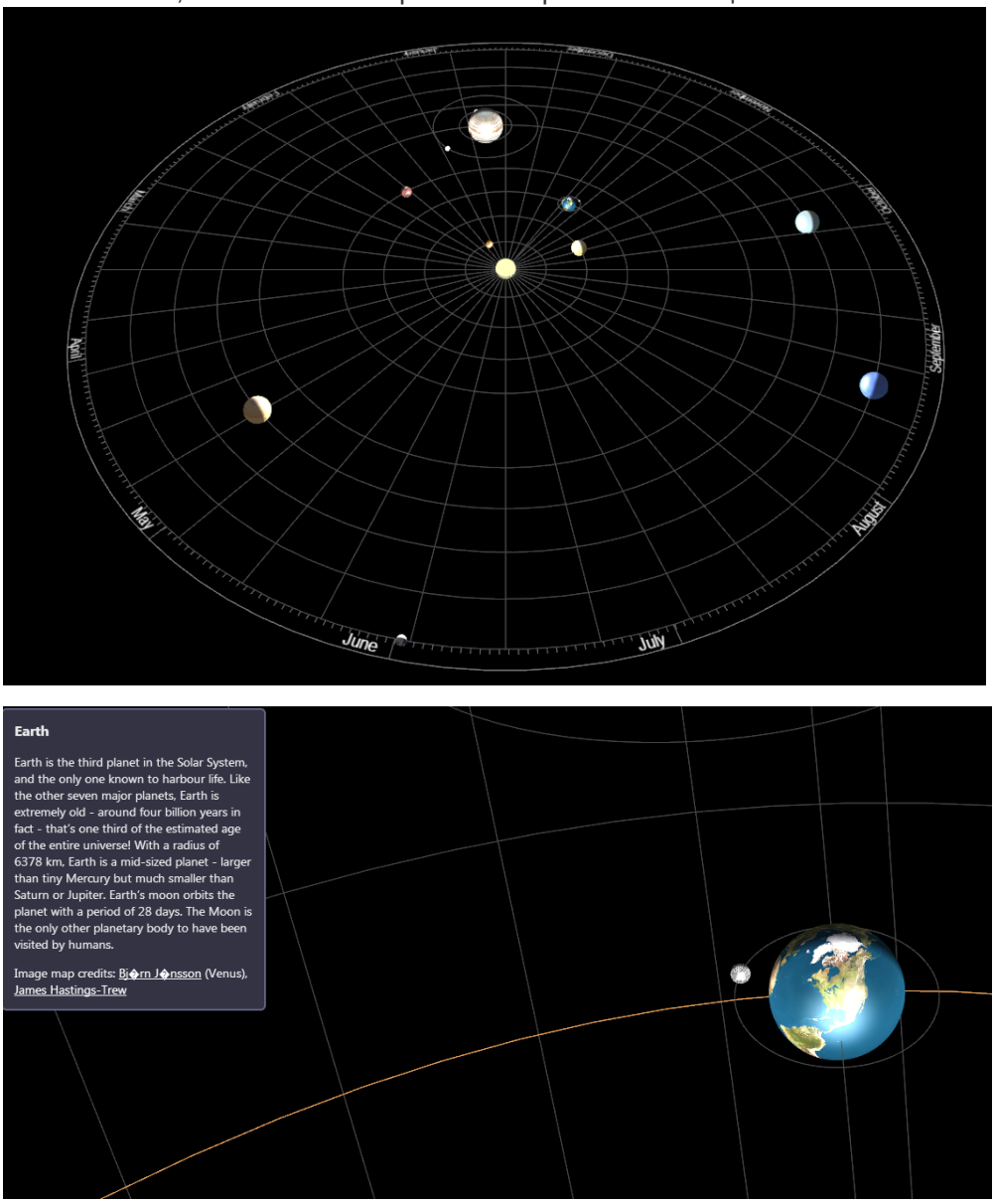


Рисунок 1. Пример выполненного задания (http://www.solar-system-explorer.com/).

Другой пример реализации задания с регулируемым вращением планет.

Задачи:

- реализовать визуализацию физической модели
- создать графическое приложений для GPU с использованием шейдеров OpenGL3/4

- изучить некоторые алгоритмы реалистичной компьютерной графики
- обеспечить интерактивный режим работы, удобный интерфейс взаимодействия с пользователем

2 Требования: обязательная часть (15 баллов)

2.1 Модель движения

1. В солнечной системе должны быть как минимум Солнце, 9 планет (давайте будем считать, что их 9). Спутники, астероиды, кометы изображать необязательно.
2. Движение планет и Солнца должно быть реализовано одним из 2 способов: по модели Кеплера или по гравитационной модели. Каждая планета должна двигаться по своей реальной орбите.
3. Планеты должны вращаться вокруг своей оси. Все скорости движения по орбитам и скорости вращения вокруг своей оси должны быть согласованы между собой.

2.2 Визуализация

1. У программы должно быть 2 режима масштаба: схематичный и реальный.

- В схематичном режиме орбиты должны даваться в логарифмической оси, размеры планет должны быть согласованы между собой (Солнце может быть уменьшено). Цель схематичного режима - удобство использования программы, наглядность.
- В реальном режиме орбиты должны быть соразмерны по расстоянию до Солнца и с размерами планет, т.е. все измерения должны соответствовать физическим величинам с точностью до некоторого коэффициента.

2. Все объекты должны выводиться на экран при помощи шейдерных подпрограмм версии не ниже OpenGl 3.0. Как минимум, должны быть использованы вершинный и фрагментный шейдеры.
3. У всех объектов (кроме Солнца) должны быть корректно подсчитанные гладкие нормали.
4. Солнце должно являться источником освещения (можно считать его точечным). Каждая планета должна освещаться Солнцем с использованием модели Фонга. У планет должно быть минимум 2 различных материала (например, у малых и больших планет).
5. Некоторые планеты (минимум одна - Земля) должны быть текстурированы.

В реальном режиме допускается не просчитывать нормали и освещение в полном объеме при просмотре издалека, однако при приближении к объектам все требования должны быть соблюдены. Главное правило: оптимизация должна быть не заметна пользователю.

2.3 Взаимодействие с пользователем

1. С помощью мыши должно быть организовано вращение камеры. Приближение и удаление можно реализовать либо с помощью колесика мыши, либо с помощью клавиатуры (+\-). Направление взгляда может быть фиксированным в центре сцены. Также у пользователя должна быть возможность перемещения по сцене (т.е. возможность движения в выбранном направлении).
2. Должна быть возможность остановить анимацию движения и запустить ее вновь по клавише “Пробел” с того момента, на котором она остановилась.

3 Требования: дополнительная часть (максимум 5 баллов)

- Качественная неполярная сетка (+1 на CPU, +2 на GPU)

- Сферическая карта звёздного неба (+1)

- Визуализация орбит планет в виде линий (+1)

- Детализация поверхности планет с использованием сдвига вершин модели или бамп-маппинга по карте высот (+2)

- Настройка тесселяции сферы (программное построение сферы из икосаэдра с фиксированным количеством подразбиений) (+2)

- Фильтры постпроцессинга (bloom effect от звезд или комет) (+1)

- Инстанцирование однотипных мешей (полигональных сеток), например, недалеких астероидов, с разными модельно-видовыми матрицами (+1) - функция glDrawElementsInstanced

- Использование сложных мешей с помощью библиотеки l3ds (космическая станция, спутник, чайник Рассела на орбите между Марсом и Юпитером, скафандр, корабль пришельцев и т.д.) (+2)

- Удобная навигация по солнечной системе (+2)

- нелинейная навигация с помощью свободной камеры, скорость камеры в пустых местах системы должна быть на порядки выше, камера не должна резко тормозить
- навигация с использованием перемещения по объектам (клик по объекту - камера перелетает к нему)

- Использование уровней детализации текстур (+1) - функции glTexImage2D и textureLod

- Движущиеся спутники, кольцо астероидов, кометы (+2)

- Управляемая скорость анимации: смещение одной планеты вручную, по ней смещаются все остальные (+1)

- Атмосфера у планет (+2)

+1 балл в случае прозрачных плоскостей

+2 балла за реалистичную атмосферу, если прозрачность зависит от положения и направления камеры

- Имитация столкновения космических объектов (массивные метеоры, астероиды) с изменением траектории (+1) и имитацией разрушения с помощью системы частиц (до +3 с использованием полупрозрачных текстур)

- Тени от спутников на поверхности планет, затмение

- простейшие резкие тени (на Земле от Луны) + 1 балл
- качественные реалистичные размытые тени от двух и более источников, реализация метода shadow volumes +2 балла

- Полупрозрачные кольца Сатурна, соразмерные с масштабом планеты

- кольца как набор вращающихся окружностей различных цветов +1 балл
- кольца как набор полупрозрачных концентрических окружностей с наложенной текстурой +2 балла
- кольца как набор полупрозрачных текстурированных концентрических окружностей с эффектом исчезновения либо с наличием движущихся частиц +3 балла

- Отражения сферического окружения (карты звездного неба):

+1 за простой объект - сферу, куб

+2 за чайник Рассела или любой другой сложный объект

+3 за отражение всего окружения в сцене, а не только сферической карты звездного неба

Дополнительная функциональность в интерфейсе

- Возможность просмотра солнечной системы (галактики) в заданную дату, а также возможность ускорения/замедления времени +1
- Возможность выделения конкретного созвездия/галактики (подсветка, соединение линиями - возможно при покоординатной частичной реализации карты звездного неба) +1
- Настройка количества звезд +1
- Возможность настроить скорость анимации (множитель соотношения реального времени и времени отображения) +1

- Возможность выбрать мышкой один из объектов (кам минимум, Солнце и 9 планет), после чего в углу экрана должно возникнуть информационное окно с краткой информацией об объекте (после клика по окну оно должно исчезнуть) +1

4 Материалы

1) Вам предлагается шаблон программы, рисующей текстурированный квадрат с помощью двух треугольников. По нажатию пробела окраска квадрата меняется градиентную.

Шаблон обеспечивает организацию работы с шейдерами: вершинным и фрагментным. Прилагаются шейдеры версии 3.2.

В шейдеры передаются необходимые матрицы, подключаемые в юниформ-переменные. Организован цикл анимации. Значения четырех пикселей текстуры жестко заданы в программном коде. При реализации задания необходимо организовать подгрузку текстуры из изображения, при этом допускается использование сторонних библиотек.

2) Шаблон проверен в Windows и Linux на сборку. На Mac OS шаблон не работает. При разработке на QT изучите предоставляемые библиотекой примеры.
Шейдеры написаны на основе библиотек freeglut и glew. Библиотеки доступны для операционных систем windows и linux. В линукс нужно установить необходимые пакеты freeglut и glew (dev версию). Бинарники для Windows можно скачать тут и тут. Проект для Visual Studio работает с локальными копиями библиотек, которые предоставляются вместе с шаблоном. В линукс НЕОБХОДИМА установка библиотек из репозитория (apt-get install freeglut, apt-get install glew).

В линукс запускать программу рекомендуется вызовом из терминала ./main в папке bin/bin, построенного make. Запуск из иной директории может не удасться, поскольку используются относительные пути к шейдерам. При разработке своего приложения вы можете исправить это, воспользовавшись платформенно-зависимыми средствами.

Перед запуском программы убедитесь, что:

- вы обновили драйверы видеокарты
- вы установили необходимые библиотеки

Узнать текущую версию OpenGL можно командами:
Linux (необходима установка mesa-utils)
glxinfo | grep -i opengl
Windows
1) запуск glew\bin\Release\Win32\glewinfo.exe
2) открываете файл glewinfo.txt и читаете, что там написано. Вас интересует в первую очередь строка вида

OpenGL version 4.4.0 is supported

Для запуска шаблона там должна быть версия не ниже 3.2.

3) Для работы с векторами и матрицами можно использовать библиотеку glm, или любую другую на ваш вкус.

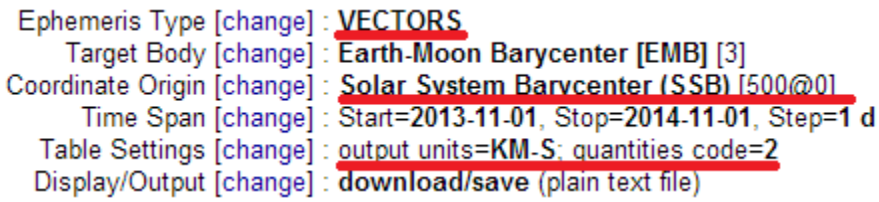
4) Для загрузки .3ds моделей можно использовать библиотеку l3ds.

5) Данные:

- **Параметры основных объектов Солнечной системы в таблице**
- Физические параметры планет, Текстуры планет, База данных малых небесных тел, модели космических кораблей.

6) Координаты известных объектов в заданное время

Настройки для получения значений в нашей системе координат (единицы измерения: километры, секунды):



5 Ресурсы для изучения OpenGL 3 и OpenGL4

1) Самый главный источник информации http://www.opengl.org/documentation/

2) Обучающие примеры:

- https://code.google.com/p/gl33lessons/
- http://nehe.gamedev.net/ (Перевод)
- http://www.opengl-tutorial.org/beginners-tutorials/
- OpenGL 3.3 http://www.gamedev.ru/code/forum/?id=138435
- OpenGL 4 http://www.gamedev.ru/code/articles/tags/OpenGL

3) Теоретические пособия:

- http://duriansoftware.com/joe/An-intro-to-modern-OpenGL.-Table-of-Contents.html
- http://www.arcsynthesis.org/gltut/

4) Полезный ресурс при выполнении бонусных заданий http://steps3d.narod.ru/

6 Правила оформления работы

В присланном архиве должно быть 2 подпапки: src с исходным кодом и bin с исполняемым. Также туда может быть помещен readme с примечаниями, куда вы должны записать пункты из

бонусной части, которые вы реализовали. В противном случае проверяющий может пропустить наличие реализации бонуса. Также в файле нужно отобразить информацию по управлению камерой и настройками программы, если оно нетривиально.

При разработке для Windows в папке bin должны быть exe файл, шейдеры и дополнительные библиотеки (см. как, например, сделано в шаблоне).

При разработке для Linux в каталоге src или в корне архива должен быть Makefile, который сформирует аналогичный вывод в папку bin (примерно как в шаблоне).

7 Подсказки к решению: база

7.1 Расчет движения планет

При расчете движения планет можно использовать один из 2 вариантов:

- Модель Кеплера движения каждой планеты по своей орбите в отдельности
- Гравитационную модель взаимодействия всех планет со всеми, т.е. приближённое решение задачи N тел

7.1.1 Глобальные координаты

O - неподвижная точка и центр глобальных координат. Плоскость орбиты Земли будем считать плоскостью эклиптики, она также неподвижна. На плоскости эклиптики выбрано некоторое направление, соответствующее т.н. точке весеннего равноденствия. Зададим оси так:

X - направление из центра Солнца в точку весеннего равноденствия

Z - перепендикуляр к плоскости эклиптики, направленный в сторону земного севера

Y задана так, что XYZ - правый ортонормированный базис.

7.1.2 О дискретизации

Рекомендуется записывать все величины в системе СИ (т.е. расстояние в метрах, массу в килограммах, время в секундах) и использовать для хранения их переменные double.

При расчётах координат планет использовать шаг дискретизации Δt по времени в 10-100 раз меньше, чем время отрисовки кадра, но не более 1 часа при быстрой анимации.

7.1.3 Расчет движения планеты по модели Кеплера

Считаем, что центр Солнца лежит в центре глобальных координат, Солнце неподвижно.

1) Орбита в глобальных координатах.

Орбита планеты является эллипсом, один из фокусов которого - центр Солнца. Параметры орбиты (форма и положение) однозначно определяются Кеплеровыми элементами орбиты. Нужно вычислить по ним локальный базис орбиты для удобства преобразования координат:

O - геометрический центр эллипса орбиты (не Солнце)

X сонаправлена с большей осью орбиты

Z направлена в сторону севера

Y сонаправлена с меньшей осью орбиты, XYZ - ортонормированный базис

2) Координаты центра планеты в плоскости ее орбиты

- Задание начальных координат и начальной скорости планеты в нулевой момент времени
- Вычисление координат планеты в следующий момент времени (текущий момент времени + Δt) из закона площадей (2 закон Кеплера). Его можно использовать приближённо, т.е. считать площади треугольников, а не эллиптических секторов.

Таким образом, сначала вычисляются относительные двумерные координаты планеты на орбите, затем они преобразуются в глобальные с помощью перехода между базисами.

7.1.4 Гравитационная модель взаимодействия

Приближённо решается задача N тел. Солнце движется, как и планеты.

Исходные данные для всех тел (Солнца,планет, спутников, ...)

- Координаты (3d точка)
- Скорость (3d вектор)
- Масса (скаляр)

Текущие координаты всех тел вычисляются итерационно. Простейший способ вычисления координат тела в следующий момент времени:

1. По закону всемирного тяготения вычислить равнодействующую всех сил, действующих на тело, на основе его положения в текущий момент времени и получить текущее ускорение
2. Считается, что весь промежуток времени Δt на тело действует постоянное ускорение, вычисленное на предыдущем шаге.
3. Вычисляем новую скорость и новое положение путём интегрирования постоянного ускорения по времени:
   $Velocity \mathrel{+}= Acceleration * \Delta t$
   $Position \mathrel{+}= Velocity_0 * \Delta t + Acceleration * \Delta t^2 / 2$

Важно использовать на текущей итерации расчёта координаты всех тел с этого же шага. Новые координаты должны где-то сохраняться. Координаты можно изменять только после завершения текущей итерации для всех тел в системе!

Вместо этого простейшего способа можно использовать многошаговую экстраполяцию ускорения, или записать систему дифференциальных уравнений движения системы тел и решать её одним из более сложных численных методов (например, Рунге-Кутты 4 порядка и выше).

В случае наличия большого количества тел (спутников, астероидов) нужно оптимизировать скорость вычислений. Например, игнорировать влияние далёких лёгких объектов, или ещё как-то.

7.2 Генерация сферы

Для генерации сферы вам необходимо создать полигональную сетку, содержащую координаты каждой точки, нормаль в этой точке и текстурную координату. При использовании индексного буфера (как в шаблоне) дополнительно необходимо заполнить его информацией о гранях (треугольниках). Плюс индексного подхода в отсутствии дублирования вершин.

Существует несколько способов генерации сферы.

Первый способ - самый очевидный. Нужно рассмотреть сферические координаты сферы. Тогда радиус будет фиксирован (например, 1). Точки на полярной сетке можно выбрать равномерным перебором зенита и азимута с фиксированным шагом. Текстурная координата будет совпадать со значениями углов, нормаль - нормированный единичный вектор, направленный по зениту и азимуту, если центр сферы находится в начале координат.

Проблема первого подхода - в неравномерности расположения точек. Чтобы избежать этой проблемы, используют итерационный процесс подразбиение некоторой аппроксимации сферы. В качестве нее обычно выбирают правильный многогранник: икосаэдр или додэкаэдр, можно куб.

Исходная фигура генерируется тривиальным фиксированным начальным набором координат. Затем каждый полигон (треугольник у икосаэдра) подразбивается. Например, можно рекурсивно разбивать каждое ребро посередине и получать 4 треугольника из одного на каждом шаге. Каждая точка проецируется на сферу (для этого ее нужно поделить на радиус в сферической системе и умножить на радиус итоговой сферы), и шаг повторяется. Нормали и текстурные координаты можно рассчитать по полярным углам.

При втором подходе нужно следить за перестроением массива индексов (треугольников) итоговой фигуры.

Можно произвести аналогичное преобразование с помощью генерации новой геометрии в тесселяционных и геометрических шейдерах (за дополнительные баллы).

О работе с тесселяционными шейдерами подробно написано здесь. Тесселяция реализуется с помощью двух шейдеров (1 и 2). После генерации всех подразбиений в геометрическом шейдере можно пересчитать нормали и текстурные координаты.

7.3 Камера

На самом деле можно себе представить, что камера помещена в центре сцены, а все остальные объекты движутся и вращаются вокруг нее.

С камерой связана модельно-видовая матрица. Модельно-видовую матрицу нужно передать в вершинный шейдер (это уже сделано в шаблоне).

Для обработки событий с клавиатуры нужно объявить функцию keyPressed, а затем указать opengl, что эту функцию следует использовать как соответствующий обработчик.

```
glutKeyboardFunc(keyPressed);
```

Для обработки событий с мыши - аналогично, нужно объявить функцию mouseFunc, а затем указать ее opengl посредством фунции glutMouseFunc.

```
void glutMouseFunc(void (*func)(int button, int state, int x, int y));
```

Примеры кода можно найти в методичке.

При создании камеры вам пригодится аппарат линейной алгебры. Камера по сути - это точка в пространстве с заданной ориентацией. Ориентацию можно задавать двумя углами в сферической системе координат. Кроме классических путей решения задачи, можно попробовать подход с кватернионами.

7.4 Текстура

Пример использования текстуры дан в шаблоне.

Для использования полноценных текстур их необходимо предварительно загрузить из файла. Размер текстуры должен быть кратен степеням двойки, также должно использоваться выравнивание рядов по 4 байта в памяти. Чтобы обойти это ограничение используйте glPixelStorei.

В схематичном режиме при приближении к планетам может быть реализована подгрузка текстур для заданных полигональных сеток (или, в более простом варианте, для всей планеты целиком).

Текстуры всех планет можно скачать здесь.

7.5 Освещение

Формулы для расчета классической модели Фонга можно посмотреть здесь.

В фрагментном шейдере необходимо рассчитать направление векторов от точки планеты или звезды на источник и на камеру. Положение источника задается встроенной переменной gl_LightSource (в случае, если в коде на cpu источник присутствовал в виде glLight(...)). В противном случае вы можете передать в шейдер нужное число источников с параметрами, используя uniform.

Для вычисления отраженного вектора в коде шейдера рекомендуется использовать команду reflect.

Планетам соответствуют материалы и текстуры, с которыми рекомендуется связать uniform переменные. Uniform-ы можно заполнять при отрисовке каждой конкретной модели.

Схематичный режим работы

Для реализации схематичного режима работы программы возможно:

1) масштабирование координат планет по некоторой функции

2) масштабирование размеров планет по некоторой функции

В качестве масштабирующей функции можно использовать логарифм.

Важно: в схематичном режиме нужно менять только масштаб при рендеринге, никаких реальных координат, использующихся в моделировании движения масштабировать не нужно, иначе можно запутаться.

7.6 Расчет нормалей

Для базы это можно сделать на CPU, при расчете на GPU в геометрическом шейдере вы получаете дополнительные баллы.

При простейшем подходе к расчету нормалей можно посчитать нормаль для каждого треугольника, а потом присвоить ее всем вершинам треугольника. Нормаль вычисляется как векторное произведение векторов-образующих сторон треугольника.

Для подсчета гладких нормалей некоторых объектов можно использовать заранее известные формулы (например, для сферы нормаль может быть вычислена аналитически в любой ее точке).

Нормаль для произвольного меша можно вычислить в геометрическом шейдере следующим образом:
1) сначала берем треугольники, прилежащие к данному (3 штуки),

2) для каждого треугольника вычисляем нормаль (вышеуказанным образом,

3) для каждой их трех вершин 3 полученные нормали усредняются (возможно взвешенное усреднение: например, учет раствора угла каждого треугольника или длины противолежащей стороны).

При таком подходе, вам нужно будет перестроить вашу модель, и передавать на отрисовку не GL_TRIANGLES, а GL_TRIANGLES_ADJACENCY (примитив из шести вершин).

Можно также реализовать сглаживание нормалей через интерполяцию гладкими кривыми.

8 Подсказки к решению: бонусы

8.1 Подгрузка текста в окно opengl

Как подгружать текст в окно opengl, написано здесь. Придется использовать команды вида

```
glTranslatef(100,100,0);
glutStrokeString(GLUT_STROKE_ROMAN,"Hello!");
```

Они установят курсор в заданную позицию окна и выведут на нам текст Hello в заданном стиле.

Для красоты текст удобно расположить на полупрозрачной подложке: прямоугольнике с прямыми или сглаженными углами светлого цвета (при темном тексте) или темного с яркой рамкой (при светлом тексте).

Атмосфера

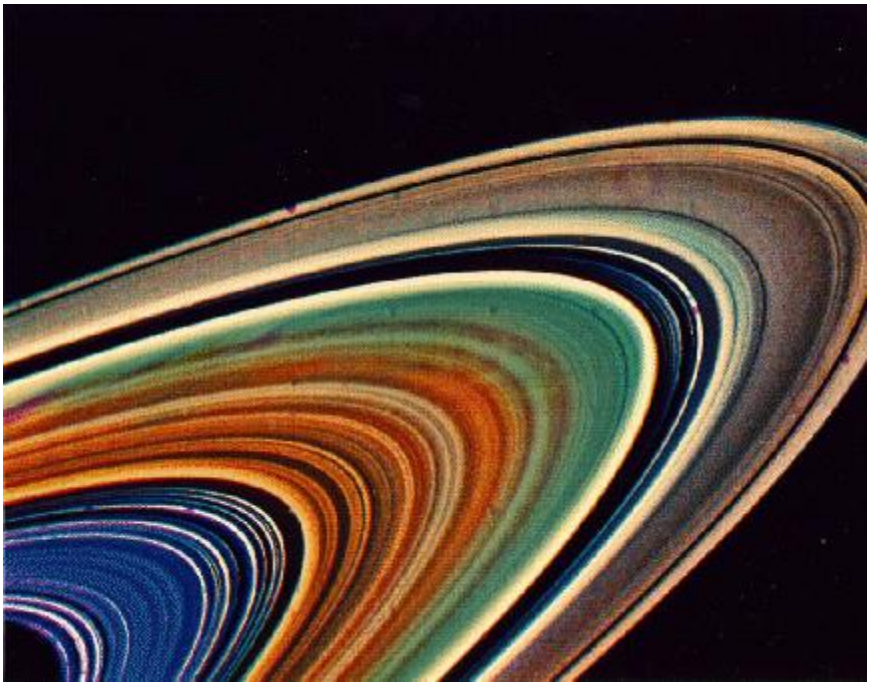
В простейшем случае атмосфера представляет собой одну либо несколько прозрачных сфер, окружающих планету. Если сфер несколько, прозрачность должна увеличиваться с увеличением радиуса. Атмосфера должна освещаться от Солнца по диффузной модели освещения. Атмосфера будет похожа на атмосферу, если во фрагментном шейдере будет установлена зависимость прозрачности от положения камеры (чем больше расстояние меша от камеры, тем больше прозрачность мешей атмосферы) - при таком подходе достаточно будет одной сферы.

Хорошо, если эффект атмосферы будет реализован в отдельном шейдере.

8.2 Плавная камера

Как в схематичном, так и в реальном режимах работы программы движение камеры должно быть согласовано с ее положением. Камера должна знать о положениях небесных тел и увеличивать шаг перемещения в зависимости от удаленности от планет. В случае, если камера находится близко к одной из планет (ближе заданного порога), она должна плавно (линейно, либо логарифмически) уменьшать шаг.

8.3 Прозрачные кольца Сатурна



Кольца Сатурна рекомендуется реализовать в схематичном режиме работы. В простейшем варианте кольца могут представлять собой набор вращающихся окружностей различных цветов. При более сложном подходе - полупрозрачные вращающиеся концентрические круги с вырезанным центрами. Хорошо, если на кольца будет наложена текстура.

При приближении к планете возможна визуализация колец в виде набора вращающихся частиц различных цветов. Следует учесть, что кольца разного диаметра вращаются с разной скоростью, чем меньше диаметр кольца (т.е. чем ближе оно к планете), тем больше скорость вращения.

Также возможна реализация эффекта “исчезающих колец Сатурна”. Кольца могут плавно исчезать и появляться снова с некоторой заданной периодичностью. Данный эффект можно реализовать, установив зависимость прозрачности мешей колец (либо частиц, при хорошем приближении) от времени.

Рекомендуется реализовать кольца Сатурна в отдельном шейдере.

При отрисовке колец важно правильно воспользоваться буфером глубины и рисовать объекты в правильном порядке: сначала выводятся все непрозрачные объекты, затем прозрачные в порядке удаленности от камеры (сначала - наиболее удаленные).

8.4 Тени от спутников на планетах

Возможно несколько различных вариантов реализации теней от спутников на поверхности планет. Можно реализовать как простейшие резкие тени от Солнца (или от Луны, если рассматриваем спутники Земли) либо более реалистичные размытые тени от двух и более источников. Поощряются все известные методы реализации теней (shadow volumes и др.).

8.5 Созвездия

Возможна реализация звездного неба в качестве сферического окружения. Приветствуется реализация нескольких отдельных созвездий, как минимум 3-х (для этого рекомендуется завести несколько дополнительных объектов созвездий и рисовать их при помощи GL_LINE_STRIP поверх сферического окружения).

8.6 Отражения сферического окружения

Отражения сферического окружения на поверхности объекта можно получить при помощи отражения вектора камеры, попавшего на модель объекта, на сферическое окружение и дальнейшей выборки текстуры в получившемся направлении. При отражении вектора, идущего от камеры к объекту, рекомендуется использовать гладкую нормаль.