Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Моделирование космических объектов»

Студент ИУ7-53Б (Группа) _____ (Подпись, дата) А. В. Криков (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) А. А. Павельев (И. О. Фамилия)

2022 г.

СОДЕРЖАНИЕ

Введение

Физические тела, окружающие нас, обладают различными оптическими свойствами. Они, к примеру, могут отражать или пропускать световые лучи, также они могут отбрасывать тень. Эти и другие свойства нужно уметь наглядно показывать при помощи электронно-вычислительных машин. Этим и занимается компьютерная графика.

Компьютерная графика – представляет собой совокупность методов и способов преобразования информации в графическое представление при помощи ЭВМ. Без компьютерной графики не обходится ни одна современная программа. В течении нескольких десятилетий компьютерная графика прошла долгий путь, начиная с базовых алгоритмов, таких как вычерчивание линий и отрезков, до построения виртуальной реальности.

Целью данного курсового проекта является разработка ПО, моделирующего систему космических объектов.

Для достижения данной цели необходимо решить следующие задачи:

- описать структуру синтезируемой трехмерной сцены;
- описать существующие алгоритмы построения реалистичных изображений;
- выбрать и обосновать выбор реализуемых алгоритмов;
- привести схемы реализуемых алгоритмов;
- определить требования к программному обеспечению;
- описать использующиеся структуры данных;
- описать структуру разрабатываемого ПО;
- определить средства программной реализации;
- реализовать соответствующее ПО;
- провести экспериментальные замеры временных характеристик разработанного ПО.

1 Аналитическая часть

В данном разделе рассматриваются существующие алгоритмы построения реалистичных изображений, обосновывается выбор реализуемого алгоритма и указывается список ограничений, в рамках которых будет работать разрабатываемое ПО.

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов:

- точечный источник света – представляет собой материальную точку, испускающую лучи света во все стороны;
- задний фон;
- космические объекты: ракета и планета – представляют собой набор полигонов (пример таких объектов представлен на рисунке 1.1).

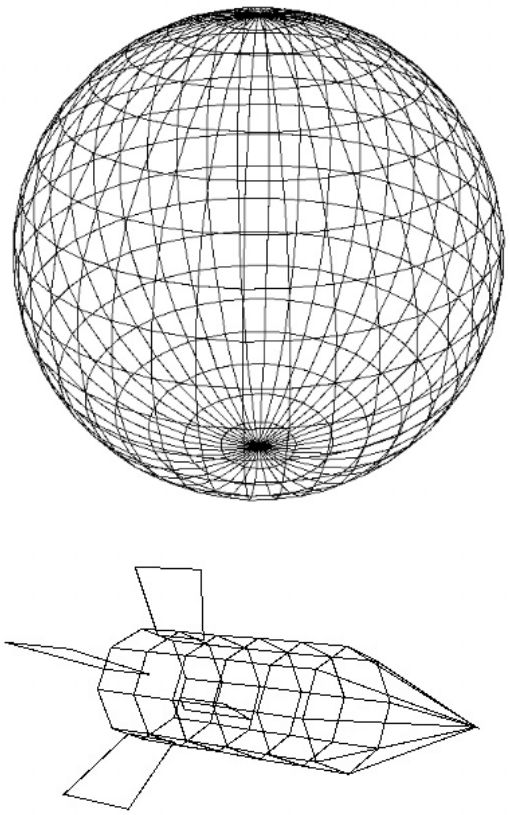

Рисунок 1.1 – Пример ракеты и планеты

В компьютерной графике в основном используются 3 вида моделей трехмерных объектов:

- каркасная (проволочная) модель. Это простейший вид моделей, содержащий минимум информации – о вершинах и рёбрах объектов. Главный недостаток – такая модель не всегда правильно передает представление об объекте (например, если в объекте есть отверстия);

- поверхностная модель. Отдельные участки задаются как участки поверхности того или иного вида. Эта модель решает проблему каркасной, но все еще имеет недостаток – нет информации о том, с какой стороны поверхности находится собственный материал;

- объемная модель. В отличии от поверхностной, содержит указание расположения материала (чаще всего указанием направления внутренней нормали).

Важнейшие требования к модели – правильность отображения информации об объекте и компактность. В рамках поставленной задачи каркасная модель не удовлетворяет первому критерию, а информация о том, где расположен материал, не является необходимой, что делает объемную модель избыточной, поэтому поверхностная модель является наиболее подходящей.

Существует несколько способов задания поверхностной модели:

- аналитический способ. Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра;

- полигональная сетка. Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

Стоит отметить, что одним из решающих факторов в выборе способа задания модели в данном проекте является скорость выполнения преобразований над объектами сцены. Поэтому при реализации программного продукта в данной работе наиболее удобным представлением является модель, заданная полигональной сеткой – это поможет избежать проблем при описании сложных моделей. При этом способ хранения полигональной сетки – это список граней, так как он содержит явное описание граней, что поможет при реализации алгоритма удаления невидимых рёбер и поверхностей.

1.2 Описание алгоритмов удаления невидимых линий и поверхностей

1.2.1 Некоторые теоретические сведения

Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства [1].

Решать задачу можно в:

- объектном пространстве – используется мировая система координат, достигается высокая точность изображения. Обобщенный подход, основанный на анализе пространства объектов, предполагает попарное сравнение положения всех объектов по отношению к наблюдателю.

- пространстве изображений – используется экранная система координат, связанная с устройством в котором отображается результат. (графический дисплей).

Под экранированием подразумевается загораживание одного объекта другим.

1.2.2 Алгоритм Робертса

Алгоритм Робертса решает задачу удаления невидимых линий. Работает в объектном пространстве. Данный алгоритм работает исключительно с выпуклыми телами. Если тело изначально является невыпуклым, то нужно его разбить на выпуклые составляющие. Алгоритм целиком основан на математических предпосылках [1].

Из-за сложности математических вычислений, используемых в данном алгоритме, а так же из-за дополнительных затраты ресурсов на вычисление матриц данный алгоритм является довольно медленным.

1.2.3 Алгоритм Варнака

Алгоритм Варнака [1—3] позволяет определить, какие грани или части граней объектов сцены видимы, а какие заслонены гранями других объектов. Так же как и в алгоритме Робертса анализ видимости происходит в

пространстве изображения. В качестве граней обычно выступают выпуклые многоугольники, алгоритмы работы с ними эффективнее, чем с произвольными многоугольниками. Окно, в котором необходимо отобразить сцену, должно быть прямоугольным. Алгоритм работает рекурсивно, на каждом шаге анализируется видимость граней и, если нельзя легко определить видимость, окно делится на 4 части и анализ повторяется отдельно для каждой из частей (см. рис. 1.2).

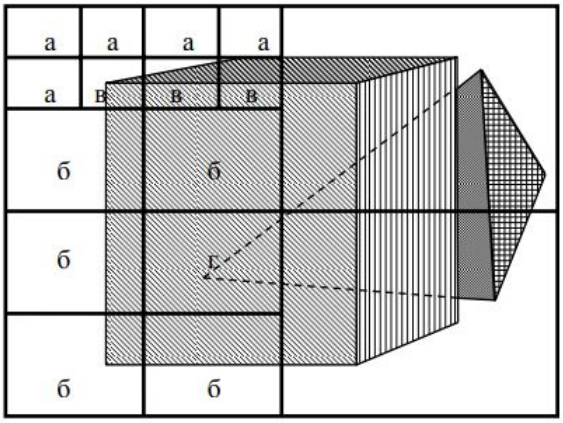

Рисунок 1.2 – Пример разбиения Алгоритмом Варнока

Так как данный алгоритм основывается на рекурсивном разбиении экрана, в зависимости от расположения объектов это может вызвать, как положительной, так и отрицательный эффект. Чем меньше пересечений объектов, тем быстрее алгоритм завершит свою работу.

1.2.4 Алгоритм Z-буфера

Алгоритм Z-буфера [1; 2] позволяет определить, какие пикселы граней сцены видимы, а какие заслонены гранями других объектов. Z-буфер – это двухмерный массив, его размеры равны размерам окна, таким образом, каждому пикселу окна, соответствует ячейка Z-буфера. В этой ячейке хранится значение глубины пиксела (см. рис. 1.3). Перед растеризацией сцены Z-буфер заполняется значением, соответствующим максимальной глубине. В случае, когда глубина характеризуется значением w, максимальной глубине соответствует нулевое значение. Анализ видимости происходит при растеризации граней, для каждого пиксела рассчитывается глубина и сравнивается со значением в Z-буфере, если рисуемый пиксел ближе (его `w` больше значения в Z-буфере), то пиксел рисуется, а значение в Z-буфере заменяется его глубиной.

Если пиксел дальше, то пиксел не рисуется и Z-буфер не изменяется, текущий пиксел дальше того, что нарисован ранее, а значит невидим.

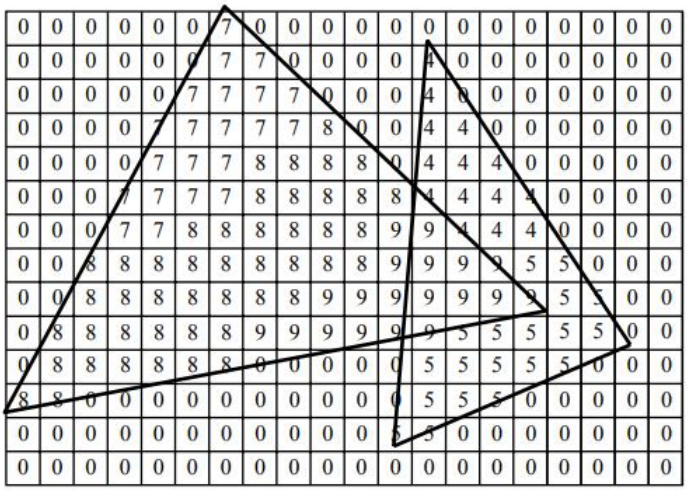

Рисунок 1.3 – Пример работы алгоритма Z-буфера

К недостаткам алгоритма следует отнести довольно большие объемы требуемой памяти, а также имеются другие недостатки, которые состоят в трудоемкости устранения лестничного эффекта и трудности реализации эффектов прозрачности.

1.2.5 Алгоритм прямой трассировки лучей

Основная идея алгоритма прямой трассировки лучей [2] состоит в том, что наблюдатель видит объекты благодаря световым лучам, испускаемым некоторым источником, которые падают на объект, отражаются, преломляются или проходят сквозь него и в результате достигают зрителя.

Основным недостатком алгоритма является излишне большое число рассматриваемых лучей, приводящее к существенным затратам вычислительных мощностей, так как лишь малая часть лучей достигает точки наблюдения. Данный алгоритм подходит для генерации статических сцен и моделирования зеркального отражения, а так же других оптических эффектов [4].

1.2.6 Алгоритм обратной трассировки лучей

Алгоритм обратной трассировки лучей отслеживает лучи в обратном направлении (от наблюдателя к объекту) [2]. Такой подход призван повысить эффективность алгоритма в сравнении с алгоритмом прямой трассировки

лучей. Обратная трассировка позволяет работать с несколькими источниками света, передавать множество разных оптических явлений [5].

Пример работы данного алгоритма приведен на рисунке 1.4.

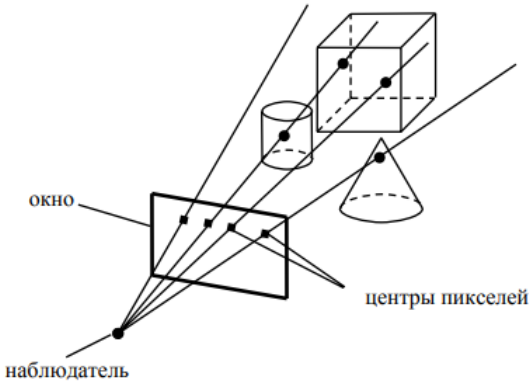

Рисунок 1.4 – Пример работы алгоритма обратной трассировки лучей

Считается, что наблюдатель расположен на положительной полуоси z в бесконечности, поэтому все световые лучи параллельны оси z. В ходе работы испускаются лучи от наблюдателя и ищутся пересечения луча и всех объектов сцены [3]. В результате пересечение с максимальным значением z является видимой частью поверхности и атрибуты данного объекта используются для определения характеристик пиксела, через центр которого проходит данный световой луч.

Для расчета эффектов освещения сцены проводятся вторичные лучи от точек пересечения ко всем источникам света. Если на пути этих лучей встречается непрозрачное тело, значит, данная точка находится в тени.

Несмотря на более высокую эффективность алгоритма в сравнении с прямой трассировкой лучей, данный алгоритм считается достаточно медленным, так как в нем происходит точный расчет сложных аналитических выражений для нахождения пересечения с рассматриваемыми объектами.

1.2.7 Алгоритм художника

Данный алгоритм работает аналогично тому, как художник рисует картину – то есть сначала рисуются дальние объекты, а затем более близкие. Пример работы данного алгоритма приведен на рисунке 1.5. Наиболее распространенная реализация алгоритма – сортировка по глубине, которая заключается в том, что произвольное множество граней сортируется по ближнему расстоянию от наблюдателя, а затем отсортированные грани выводятся на

экран в порядке от самой дальней до самой ближней. Данный метод работает лучше для построения сцен, в которых отсутствуют пересекающиеся грани [1].

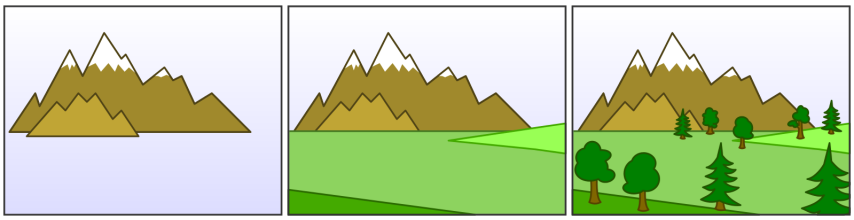

Рисунок 1.5 – Пример работы алгоритма художника

Основным недостатком алгоритма является высокая сложность реализации при пересечении граней на сцене.

1.3 Анализ и выбор модели освещения

Физические модели материалов стараются аппроксимировать свойства некоторого реального материала. Такие модели учитывают особенности поверхности материала или же поведение частиц материала.

Эмпирические модели материалов устроены иначе, чем физически обоснованные. Данные модели подразумевают некий набор параметров, которые не имеют физической интерпретации, но которые позволяют с помощью подбора получить нужный вид модели.

В данной работе следует делать выбор из эмпирических моделей, а конкретно из модели Ламберта и модели Фонга.

1.3.1 Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение, то есть свет при попадании на поверхность рассеивается равномерно во все стороны. При такой модели освещения учитывается только ориентация поверхности (N) и направление источника света (L). Иллюстрация данной модели представлена на рисунке 1.6.

Эта модель является одной из самых простых моделей освещения и очень часто используется в комбинации с другими моделями. Она может быть очень удобна для анализа свойств других моделей, за счет того, что ее легко выделить из любой модели и анализировать оставшиеся составляющие.

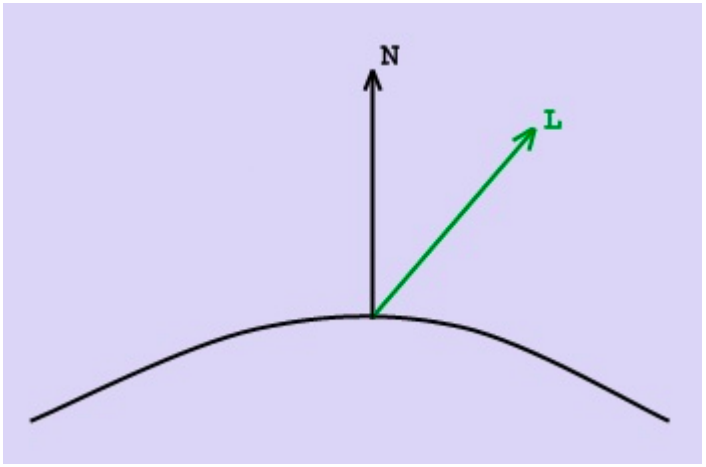

Рисунок 1.6 – Направленность источника света

1.3.2 Модель Фонга

Это классическая модель освещения. Модель представляет собой комбинацию диффузной и зеркальной составляющих. Работает модель таким образом, что кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки.

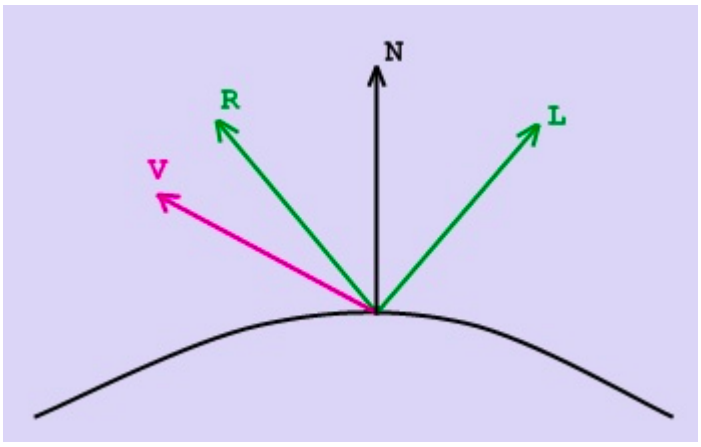

Рисунок 1.7 – Направленность источника света

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения (рисунок 1.7). Нормаль делит угол между лучами на две равные части. L – направление источника света, R – направление отраженного луча, V – направление на наблюдателя.

1.4 Вывод

Поскольку в данной моделируемой системе космические объекты не пересекаются, для удаления невидимых линий был выбран алгоритм художника. Данный алгоритм позволит добиться максимальной производительности, что особенно важно для динамических сцен. Стоит отметить тот факт, что алгоритм художника не требователен к памяти, в отличие, например, от алгоритма Z-буфера.

В качестве модели освещения в данной работе была выбрана модель Ламберта из-за своей простоты по сравнению с моделью Фонга. Для расчета данных модели Ламберта необходимо выполнять меньше вычислений, а значит ее реализация потребует меньшего количества времени.

2 Конструкторская часть

В данном разделе будут рассмотрены требования к программному обеспечению, а также схемы алгоритмов, выбранных для решения поставленной задачи. Так же, будут описаны пользовательские структуры данных и приведена структура реализуемого программного обеспечения.

2.1 Требования к программному обеспечению

ПО должно предоставлять доступ к следующему функционалу:

- визуальное отображение сцены;
- изменение скорости движения системы космических объектов;
- выбор режима отображения моделей;
- перемещение камеры (точки наблюдения);
- изменение положения источника света;
- изменение количества источников света.

К ПО предъявляются следующие требования:

- время отклика программы не должно превышать 5 секунд для корректной работы в интерактивном режиме;
- программа должна корректно реагировать на любые действия пользователя.

2.2 Алгоритм Художника

Этот метод работает аналогично тому, как художник рисует картину – сначала рисуются более далекие объекты, а затем близкие. Метод основывается на том факте, что если самая дальняя точка грани A ближе к наблюдателю, чем самая ближняя точка грани B, то грань B никак не может закрыть грань A. Однако, в некоторых случаях это не всегда так. Можно подобрать такие две грани A и B, что для них не будет выполняться это правило. Тогда стоит применить следующую модификацию данного алгоритма: берутся средние

значения расстояний от вершин граней до наблюдателя и для двух граней A и B выполняется сравнение этих значений. Если среднее значение глубины для грани A меньше соответствующего значения для грани B, то грань A закрывает грань B и сначала рисуется грань B, а затем – грань A. Наиболее распространенная реализация алгоритма сортировки по глубине заключается в том, что произвольное множество граней сортируется по ближайшему расстоянию до наблюдателя, а затем отсортированные грани выводятся на экран в порядке от самой дальней к самой ближней.

2.3 Модель освещения Ламберта

Данная модель вычисляет цвет поверхности в зависимости от того как на нее светит источник света. Согласно данной модели, освещенность точки равна произведению силы источника света и косинуса угла, под которым он светит на точку.

$$I = I_0 * cos(L, N), \tag{2.1}$$

где:

- I – результирующая интенсивность света в точке;
- I_0 – интенсивность источника;
- L – направление из точки на источник;
- N – вектор нормали.

На Рисунке 2.1 представлена схема синтеза изображения с применением алгоритма Художника.

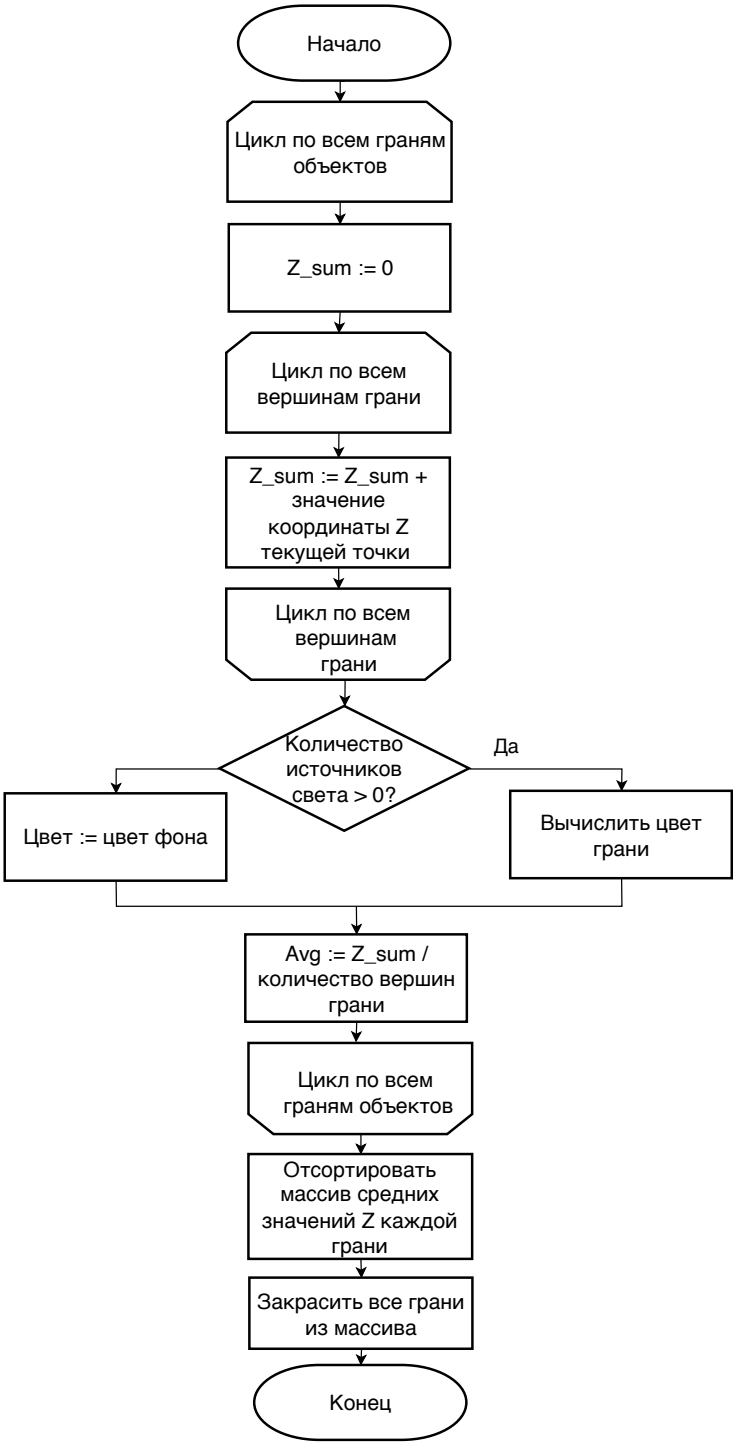

Рисунок 2.1 – Схема алгоритма синтеза изображения с применением алгоритма Художника

2.4 Представление данных в ПО

В данной работе используются следующие типы и структуры данных:

- источник света – задается расположением;
- точка трехмерного пространства – хранит направление по x, y, z;

- цвет – хранит три составляющие RGB модели цвета;
- объект сцены – список полигонов.

2.5 Описание структуры программного обеспечения

На Рисунке 2.2 представлена диаграмма классов реализуемого программного обеспечения.

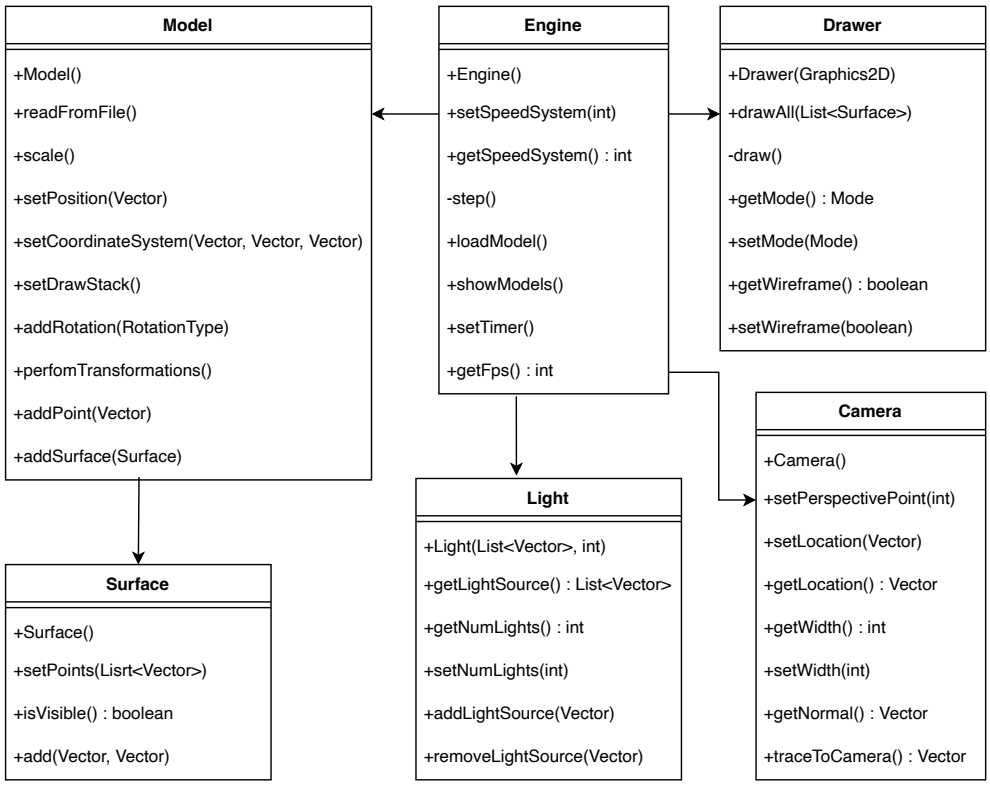

Рисунок 2.2 – Диаграмма классов реализуемого ПО

2.6 Вывод

В данном разделе были представлены требования к разрабатываемому программному обеспечению и разработана схема разрабатываемого алгоритма. Так же, были описаны пользовательские структуры данных и приведена структура реализуемого программного обеспечения.

3 Технологическая часть

В данном разделе будут представлены средства разработки программного обеспечения, интерфейс программного обеспечения и процесс сборки разрабатываемого программного обеспечения.

3.1 Выбор средств реализации

В качестве языка программирования для разработки программного обеспечения был выбран язык `Java` [6]. Данный выбор обусловлен тем, что данный язык предоставляет весь функционал требуемый для решения поставленной задачи.

Для создания пользовательского интерфейса ПО был использован фреймворк `AWT` [7]. Данный фреймворк содержит в себе объекты, позволяющие напрямую работать с пикселами изображения, а так же возможности создания интерактивных пользовательских интерфейсов, что позволит в интерактивном режиме управлять изображением.

Для сборки программного обеспечения использовался инструмент – `Gradle`.

В качестве среды разработки был выбран текстовый редактор `IntelliJ IDEA` [8], поддерживающий возможность установки плагинов, в том числе для работы с `Java` и `Gradle`.

3.2 Описание процесса сборки приложения

Действия, необходимые для сборки проекта приведены приведены в листинге 3.1:

Листинг 3.1 – Сборка реализуемого программного обеспечения

```
1 $ cd SpaceSimulation
2 $ gradle SpaceSimulation
3 $ cd out
```

В результате выполнения данных действий будет скомпилирован jar файл.

3.3 Интерфейс ПО

Интерфейс реализуемого ПО представлен на Рисунках 3.1 – 3.4.

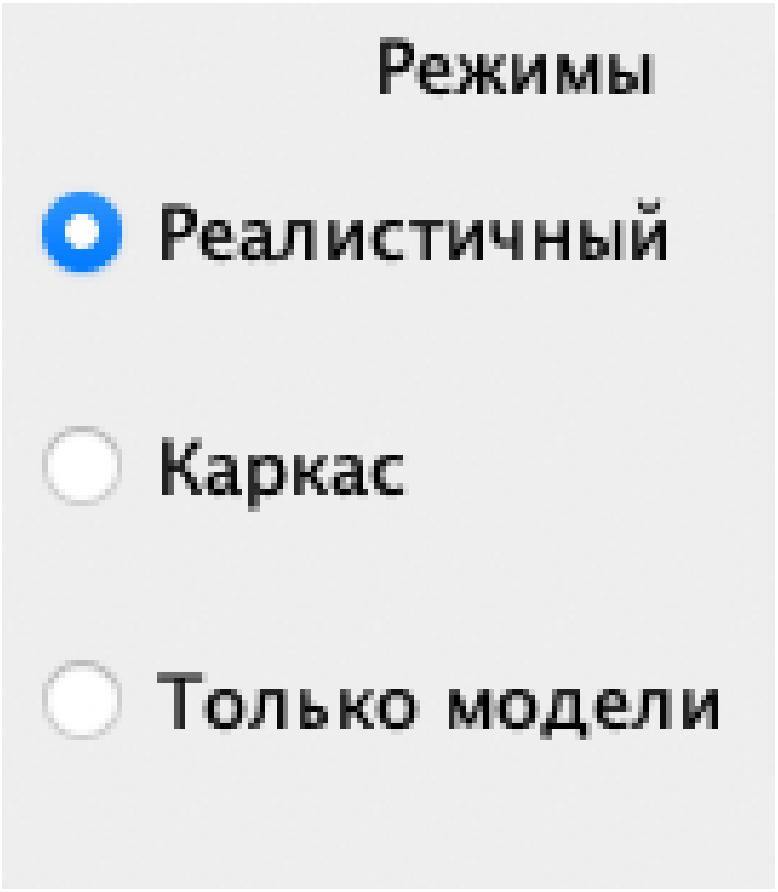

Рисунок 3.1 – Интерфейс программы – группа выбора режима

На рисунке 3.1 представлен интерфейс настройки выбора режима, включающий в себя выбор трех режимов: «реалистичный», «каркас», «только модели». Режим «каркас» представляет собой отображение сцены с помощью каркасной модели. «Реалистичный» режим представляет собой закрашенные модели с задним фоном. Режим «только модели» предоставляет собой режим с закрашенными моделями без заднего фона.

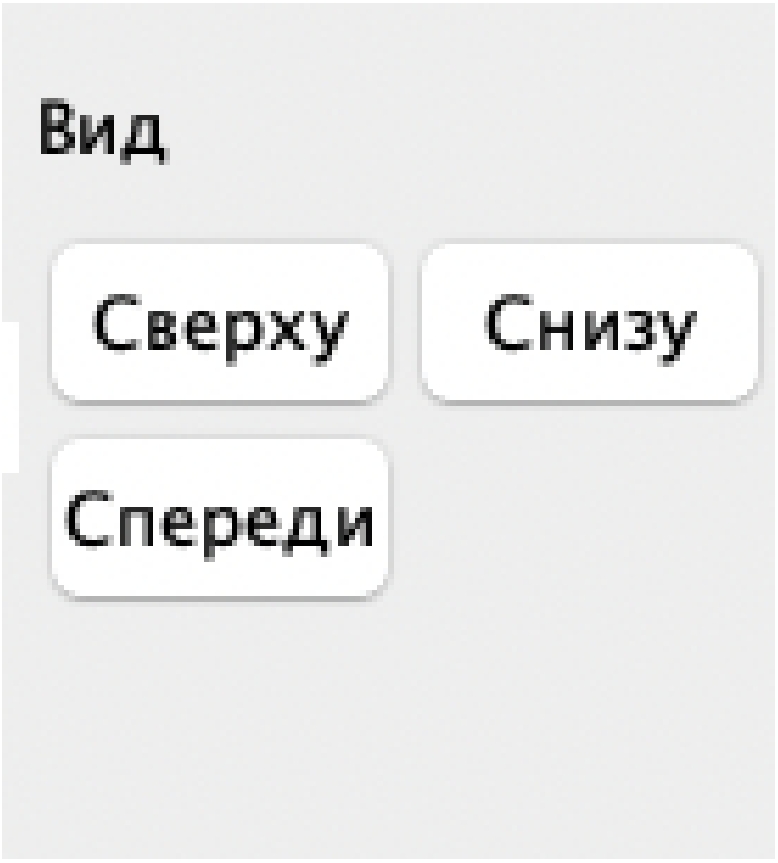

Рисунок 3.2 – Интерфейс программы – группа настроек камеры

На рисунке 3.2 представлен интерфейс настройки параметров камеры (точки наблюдения), включающий в себя выбор положения точки наблюдения: «сверху», «снизу», «спереди»

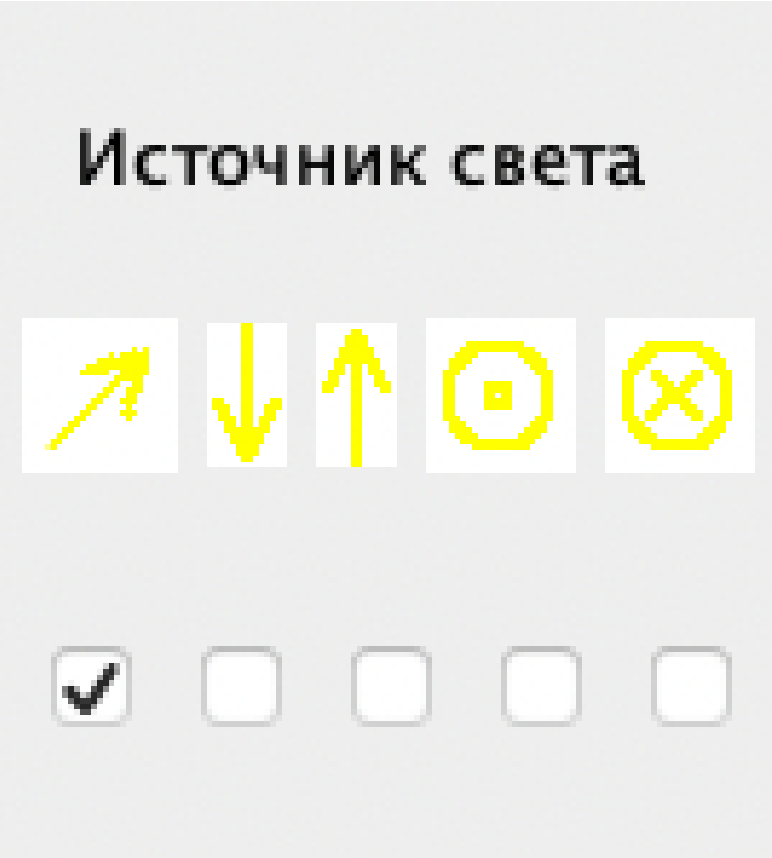

Рисунок 3.3 – Интерфейс программы – группа настроек освещения сцены

На рисунке 3.3 представлен интерфейс настройки параметров освещения синтезируемой сцены, включающий в себя выбор положения источника освещения, а также выбор количества источников освещения.

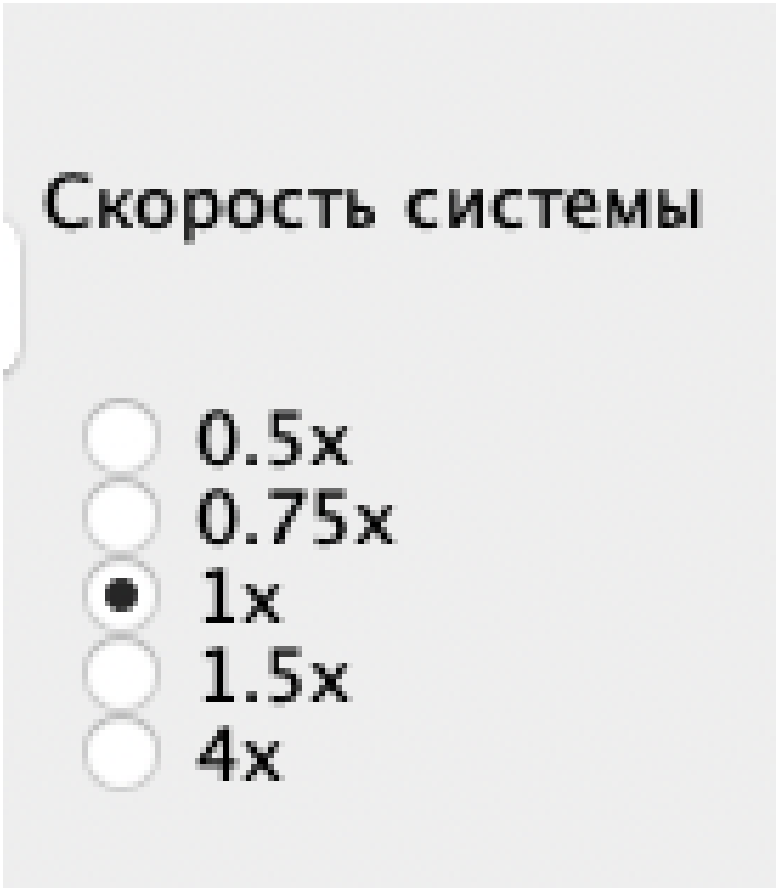

Рисунок 3.4 – Интерфейс программы – группа настроек скорости системы

На рисунке 3.4 представлен интерфейс настройки скорости системы.

3.4 Вывод

В данном разделе были представлены средства разработки программного обеспечения, детали реализации и процесс сборки разрабатываемого программного обеспечения.

4 Экспериментальная часть

В данном разделе будет поставлен эксперимент, в котором будут сравнены временные характеристики работы реализованного программного обеспечения в различных конфигурациях.

4.1 Цель эксперимента

Целью эксперимента является проверка правильности выполнения поставленной задачи, оценка эффективности при увелечении количества отображаемых объектов.

4.2 Апробация

На рисунках 4.1 – 4.3 представлены результаты синтеза изображения в различных режимах. Можно заметить, что сцены работают корректно.

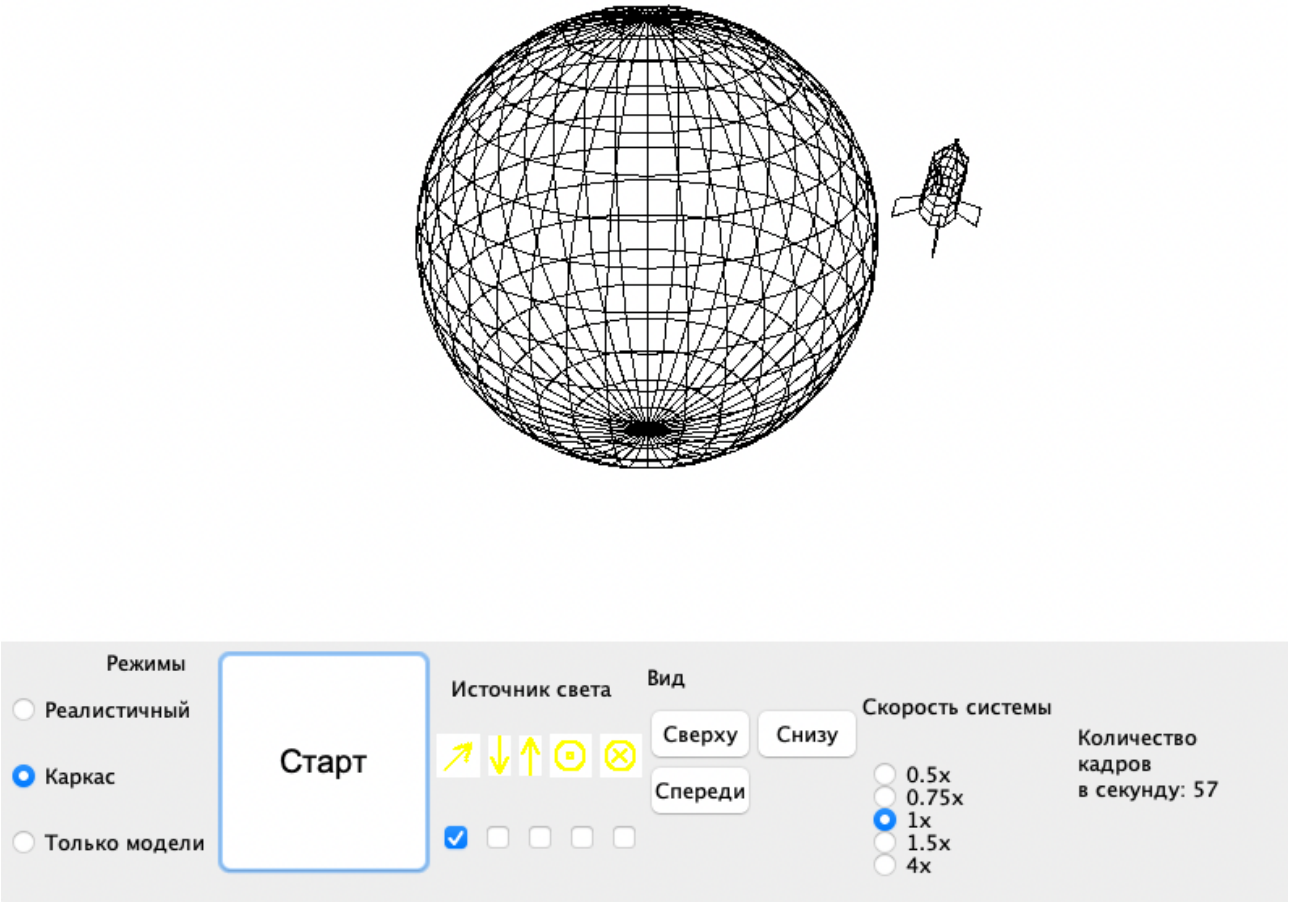

Рисунок 4.1 – Визуализация сцены в режиме «Каркас»

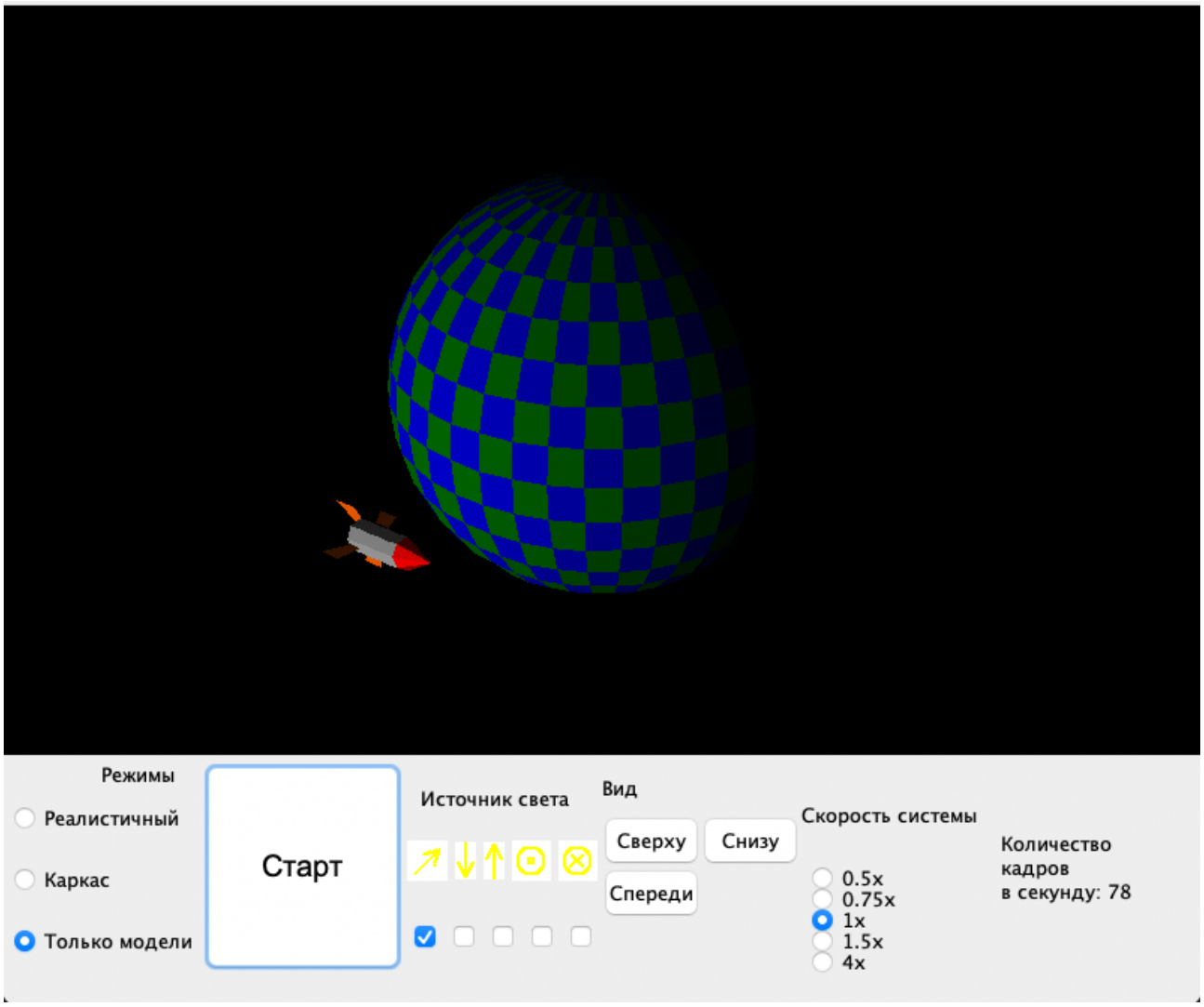

Рисунок 4.2 – Визуализация сцены в режиме «Только модели»

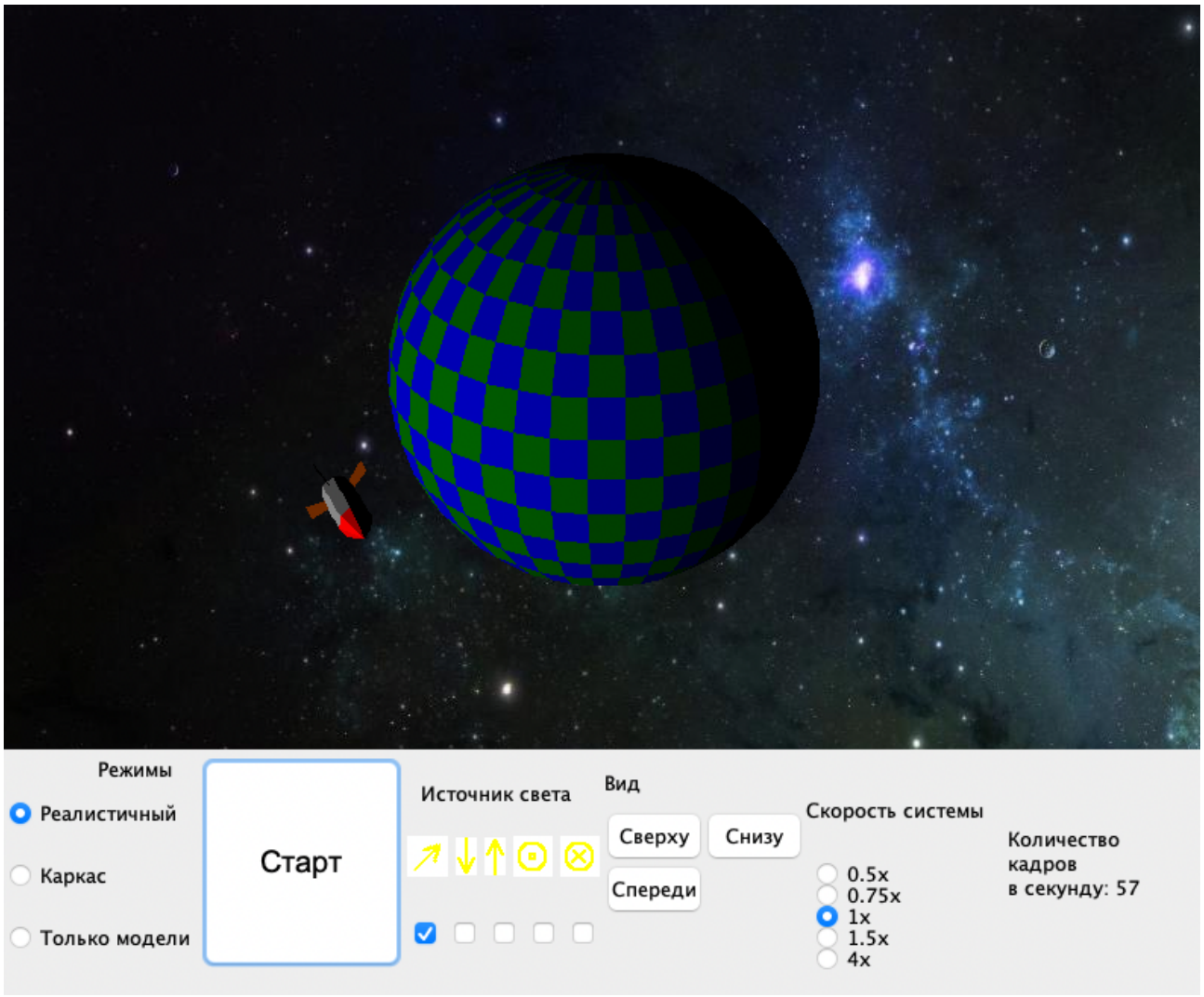

Рисунок 4.3 – Визуализация сцены в режиме «Реалистичный»

На рисунках 4.4 – 4.6 приведен результат работы программы при разном положении наблюдателя. Виды сверху, спереди и снизу работают корректно.

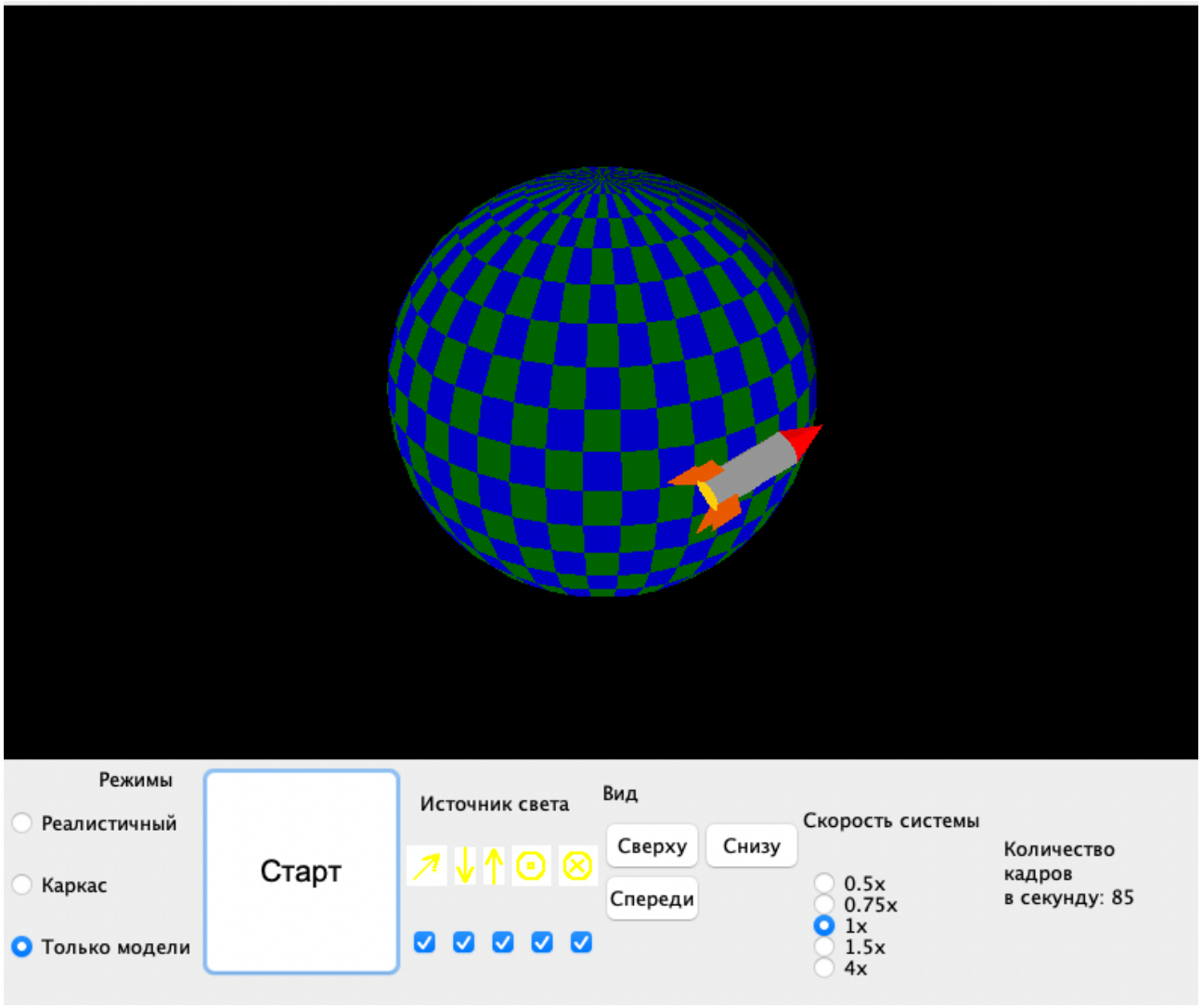

Рисунок 4.4 – Вид спереди, добавлены все источники освещения

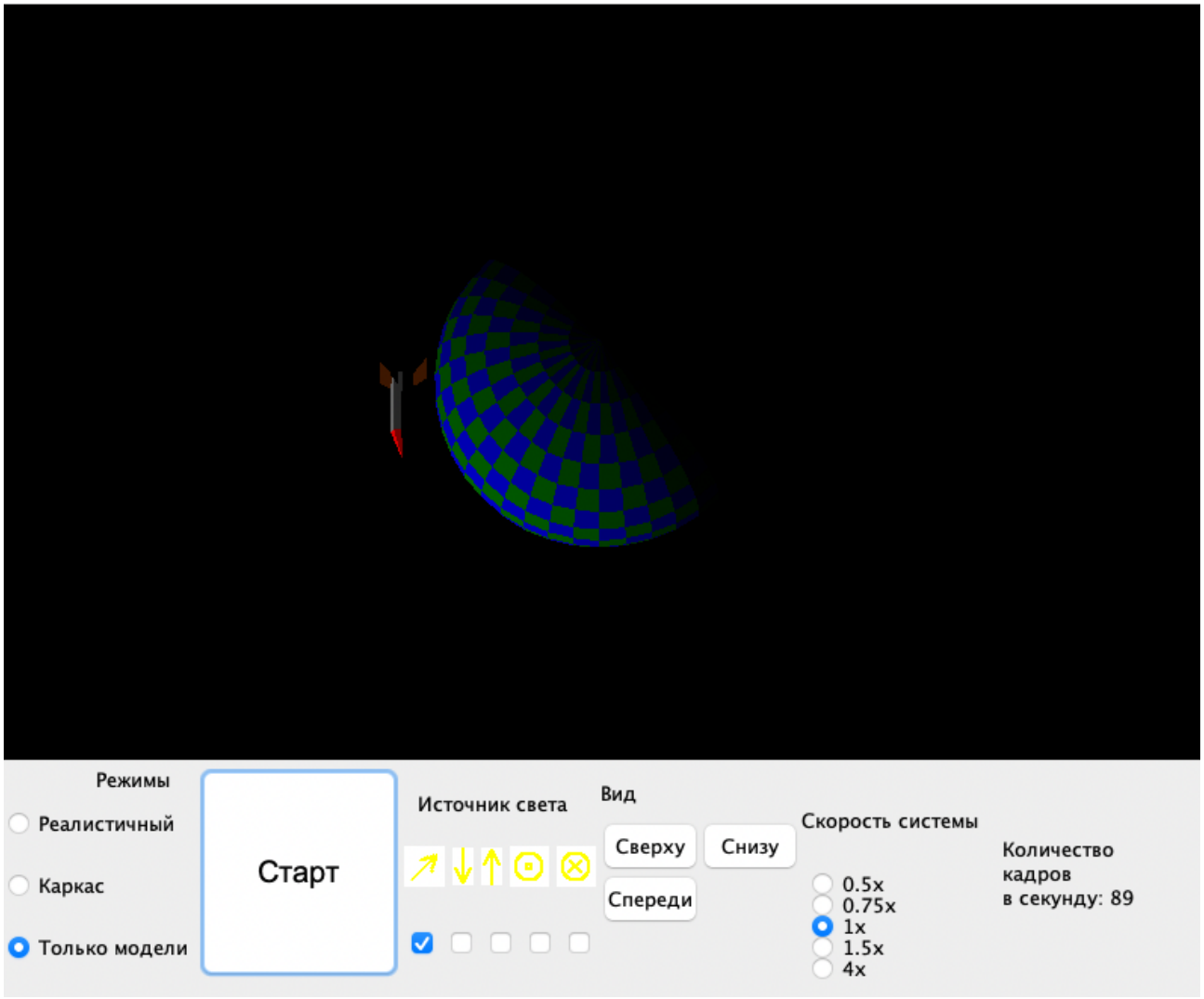

Рисунок 4.5 – Вид сверху

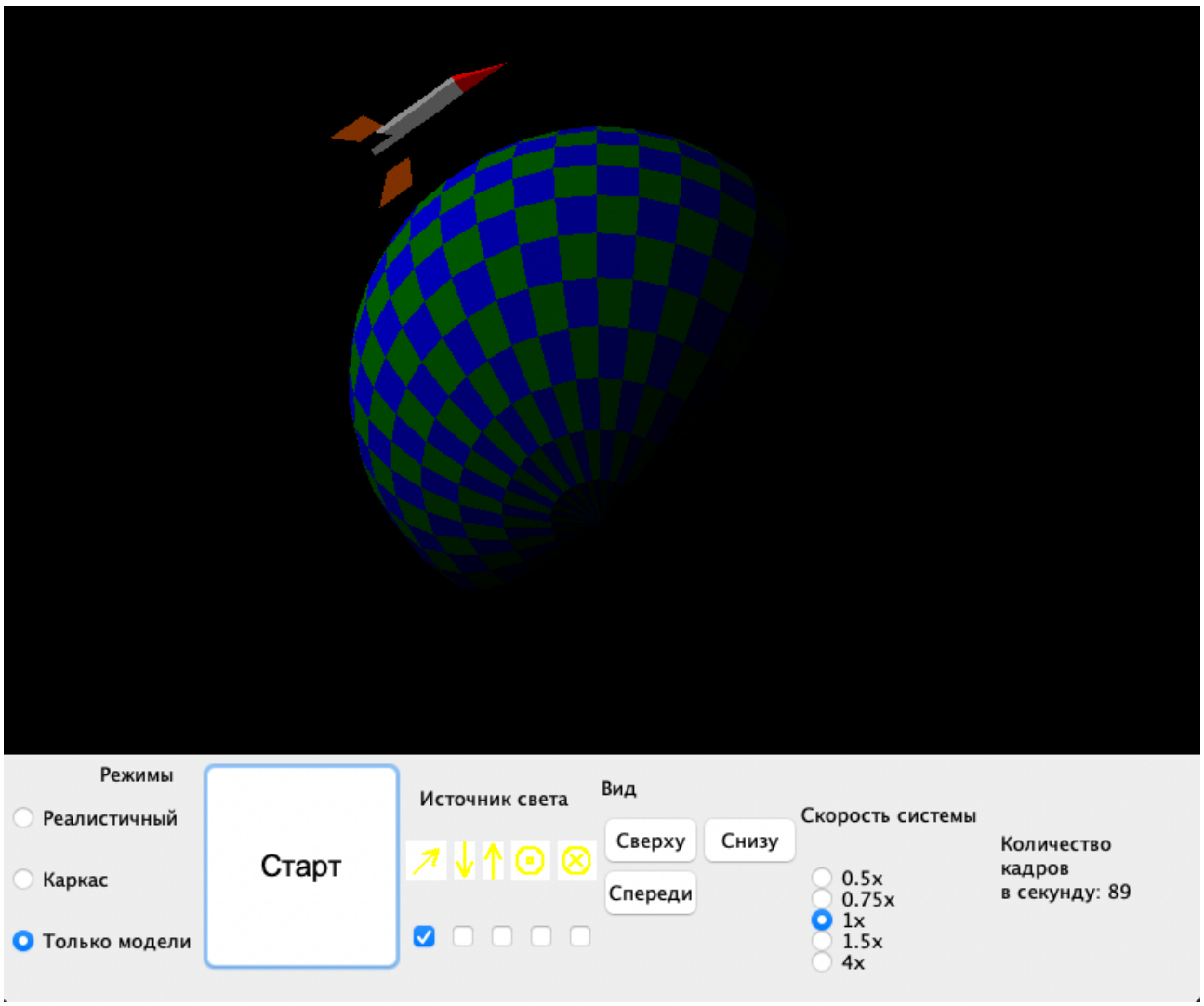

Рисунок 4.6 – Вид снизу

4.3 Технические характеристики

Технические характеристики устройства, на котором выполнялось исследование:

- процессор: Intel Core™ i5-8250U [9] CPU @ 1.60GHz;
- память: 16 GiB;
- операционная система: Manjaro [10] Linux [11] 21.1.4 64-bit.

Исследование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения рабочего стола, окружением рабочего стола, а также непосредственно системой тестирования.

4.4 Описание эксперимента

Предметом серии поставленных экспериментов является реализованный в рамках курсового проекта алгоритм художника.

В рамках данного эксперимента будет производиться оценка влияния количества изображаемых объектов на время работы алгоритма. Для этого будем синтезировать сцены с количествами объектов равными [300, 400, 500, . . . , 1000].

Для снижения погрешности измерений будем усреднять получаемые значения. Для этого каждое из измерений будет проводиться $N = 100$ раз, после чего будет вычисляться среднее арифметическое значение измеряемой величины.

4.5 Результат эксперимента

На рисунке 4.7 приведены графики зависимости времени синтеза изображения от количества изображаемых объектов.

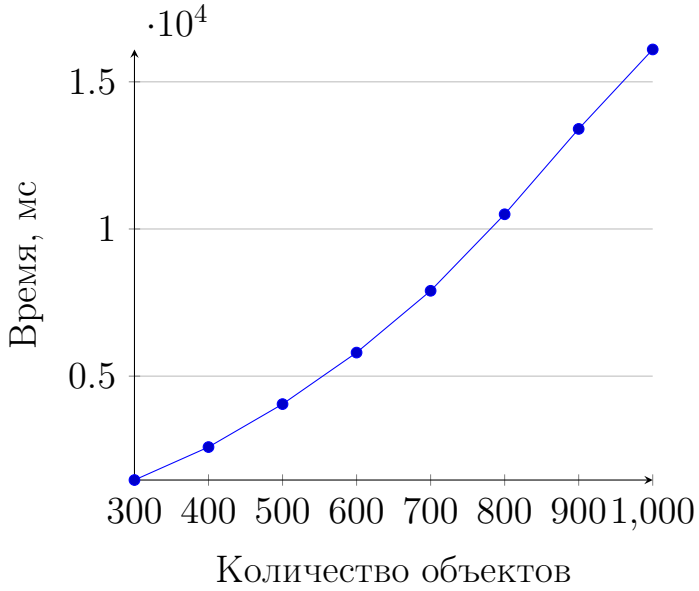

Рисунок 4.7 – График зависимости времени синтеза изображения от количества изображаемых объектов

Из рисунка 4.7 следует, что реализованный алгоритм художника зависит линейно от количества объектов. При этом интерактивность реализуемого ПО теряется при количестве изображаемых объектов равном 500 и более.

4.6 Вывод

В данном разделе было произведено экспериментально сравнение временных характеристик реализованного программного обеспечения.

Время работы алгоритма имеет линейную зависимость от количества отображаемых объектов.

Заключение

В ходе курсового проекта было разработано программное обеспечение, предоставляющее возможность визуализации системы космических объектов. Разработанное программное обеспечение предоставляет функционал для изменения положения наблюдателя и источников освещения, также предусмотрена возможность изменения режимов отображения объектов и изменения скорости движения моделируемой системы в интерактивном режиме. В процессе выполнения данной работы были выполнены следующие задачи:

- описана структура синтезируемой трехмерной сцены;
- описаны существующие алгоритмы построения реалистичных изображений;
- были выбраны реализуемые алгоритмы;
- приведены схемы реализуемых алгоритмов;
- определены требования к программному обеспечению;
- описаны использующиеся структуры данных;
- описана структура разрабатываемого ПО;
- определены средства программной реализации;
- реализовано соответствующее ПО;
- проведены экспериментальные замеры временных характеристик разработанного ПО.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Роджерс Д.* Алгоритмические основы машинной графики. — 1-е изд. — Д. Роджерс. — Москва «Мир», 1989.
2. *Шикин Е. В.* Компьютерная графика. Динамика, реалистические изображения //. — М.: ДИАЛОГ–МИФИ, 1998. — С. 288.
3. Ю.М. Баяковский. Трассировка лучей из книги Джефа Проузиса [Электронный ресурс]. — Режим доступа: `https://www.graphicon.ru/oldgr/courses/cg99/notes/lect12/prouzis/raytrace.htm` дата обращения: 03.11.2021).
4. Проблемы трассировки лучей – из будущего в реальное время [Электронный ресурс]. — Режим доступа: `https://nvworld.ru/articles/ray-tracing/3/` дата обращения: 03.11.2021).
5. *Снижко Е. А.* Компьютерная графика. Динамика, реалистические изображения //. — Балт. гос. техн. ун-т. – СПб., 2005. — С. 132.
6. Standard C++ [Электронный ресурс]. — Режим доступа: `https://isocpp.org/` (дата обращения: 24.10.2021).
7. Package java.awt [Электронный ресурс]. — Режим доступа: `https://docs.oracle.com/javase/9/docs/api/java/awt/package-summary.html`.
8. IntelliJ IDEA - Code Editing. Redefined. — Режим доступа: `https://www.jetbrains.com/ru-ru/idea/`.
9. Процессор Intel® Core™ i5-8250U [Электронный ресурс]. — Режим доступа: `https://ark.intel.com/content/www/ru/ru/ark/products/124967/intel-core-i5-8250u-processor-6m-cache-up-to-3-40-ghz.html` (дата обращения: 24.10.2021).
10. Manjaro - enjoy the simplicity [Электронный ресурс]. — Режим доступа: `https://manjaro.org/` (дата обращения: 24.10.2021).
11. LINUX.ORG.RU – Русская информация об ОС Linux [Электронный ресурс]. — Режим доступа: `https://www.linux.org.ru/` (дата обращения: 24.10.2021).