|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гарасев Никита Алексеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_ИУ7-42Б\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_стационарная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия \_\_\_\_\_\_МГТУ им. Н. Э. Баумана, каф. ИУ7\_\_\_\_\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Гарасев Н.А.\_

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Куров А.В.**\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2020 г.*

**Индивидуальное задание:**

Разработать программу для построения трехмерного сцены, состоящей из примитивов, обладающих различными изменяемыми атрибутами, такие как цвет, прозрачность. Необходимо предоставить возможность добавлять, удалять, а также изменять объекты сцены. Исследовать и выбрать алгоритм для построения трехмерного изображения и моделирования сцены, обеспечивающий работу программы в режиме реального времени. Исследовать зависимость скорости рендеринга одного кадра от разрешения рендера

Оглавление

[1. Аналитическая часть 6](#_Toc52359564)

[1.1. Модель представления объектов сцены 6](#_Toc52359566)

[1.2. Анализ алгоритмов удаления невидимых линий и поверхностей 7](#_Toc52359576)

[1.3. Анализ методов закрашивания 11](#_Toc52359577)

[1.4.Анализ алгоритмов построения теней 13](#_Toc52359578)

[1.5.Анализ моделей освещения 13](#_Toc52359579)

[2. Конструкторская часть 16](#_Toc52359580)

[2.1. Общий алгоритм решения 16](#_Toc52359582)

[2.2.Алгоритм обратной трассировки лучей 16](#_Toc52359584)

[2.3.Метод освещения по Фонгу 16](#_Toc52359585)

[3. Техническая часть 18](#_Toc52359586)

[3.1. Выбор языка программирования и среды разработки 18](#_Toc52359587)

[Заключение 22](#_Toc52359592)

[Список литературы 23](#_Toc52359593)

**Введение**

В современном мире компьютерная графика является неотъемлемой частью человеческой жизни. Она используется повсеместно: для наглядного отображения данных, в компьютерных играх и даже в кино для создания эффектов.

Трёхмерная графика активно применяется для создания изображений на плоскости экрана или листа печатной продукции в науке и промышленности, например, в системах автоматизации проектных работ, архитектурной визуализации, в современных системах медицинской визуализации.

Итак, перед людьми встает задача создания реалистичных изображений. Существует множество алгоритмов компьютерной графики, которые решают эту задачу. Зачастую эти алгоритмы ресурсозатратны: чем более качественное изображение мы получаем в итоге, тем больше времени и памяти мы тратим на его синтез. Это становится проблемой при создании динамической сцены, где на каждом временном интервале необходимо производить расчеты заново.

Для решения поставленного задания необходимо проанализировать задачу, декомпозировать её, исследовать различные методы решения задач, выбрать подходящие методы, и выбрав язык программирования и определив нужные структуры, реализовать эти методы. Затем нужно спроектировать «интуитивно понятный» интерфейс для удобства использования программы.

Таким образом, создание программного комплекса для выполнения вышеописанных операций является актуальным. Для получения адекватного результата необходимо разделить задачи моделирования и визуализации и рассмотреть плюсы и минусы возможных подходов для решения каждой из задач. Получившийся комплекс должен быть способен учитывать такие физические эффекты, как отражение, преломление и рассеивание света, чтобы создавать реалистичное изображение. Более того, возможность наложения на жидкость текстуры и придания ей цвета способна значительно улучшить визуальные качества изображения.

Исходя из вышеописанного, целью практики является выбор или модификация существующих алгоритмов и их реализация для создания трёхмерной сцены.

1. Аналитическая часть

Необходимо декомпозировать задачу, проанализировать и выбрать алгоритмы, удовлетворяющие условие задачи.

# **Модель представления объектов сцены**

Выбор модели представления объектов для сцены. Выбранная модель должна обеспечивать достаточную для задачи точность описания, предоставлять возможность визуализации всех объектов за реальное время.

Необходимо выбрать модель представления объектов. Большая часть трехмерных алгоритмов работает исключительно с многоугольниками. В таком случае выбор полигонального представления объектов обеспечивает независимость представления от конкретного алгоритма, который при необходимости можно изменить, не меняя остальную часть программы.

В качестве примитивов можно взять параллелепипед, сферу и цилиндр. Для параллелепипеда построение полигональной модели является тривиальным. Для сферы и цилиндра разумно использовать общий алгоритм построения полигонального приближения поверхности вращения.

В качестве конкретного типа многоугольника, используемого в программе, был выбран треугольник. У треугольника есть ряд особенностей, которые делают его наиболее подходящим для применения в моделировании объектов в рамках компьютерной графики:

1. Любой многоугольник можно точно разделить на некоторое количество треугольников, треугольник же можно разделить только на меньшие треугольники. Можно утверждать, что треугольник представляет собой атомарный многоугольник.
2. Треугольник, за исключением вырожденных случаев, всегда задает одну плоскость и расположен в одной плоскости, в отличие от многоугольников с большим количеством вершин.
3. Треугольник не может быть невыпуклым, в отличие от других многоугольников. Это свойство упрощает его растеризацию.
4. В треугольнике относительно просто восстановить интерполяцией нормаль к исходной поверхности в любой точке, что актуально для алгоритмов затенения.
5. Треугольники позволяют эффективно использовать память, если хранить их в виде полоски треугольников.

# **Анализ алгоритмов удаления невидимых линий и поверхностей**

При выборе алгоритма удаления невидимых линий нужно учитывать особенности формулировки задачи. Важным вопросом является необходимость реалистичного моделирования преломления и отражения света. Более того, алгоритм должен работать достаточно быстро, чтобы смена кадров не опережала вычисление данных нового кадра.

**Алгоритм обратной трассировки лучей**

Алгоритм является улучшенной модификацией алгоритма прямой трассировки. Из камеры испускаются лучи, проходящие через каждый пиксель вглубь сцены, затем идет поиск пересечений первичного луча с объектами сцены, затем идет поиск пересечений первичного луча с объектами сцены, в случае обнаружения пересечения, рассчитывается интенсивность пикселя, в зависимости от положения источника света, при отсутствии пересечения пиксель закрашивается цветом фона. Вычислительная сложность метода линейно зависит от сложности сцены. Качество изображения получается очень реалистичным, этот метод отлично подходит для создания фотореалистичных картин.

Серьезным недостатком этого алгоритма будет являться большое количество необходимых вычислений. Для получения изображения необходимо создавать огромное число лучей, проходящих через сцену и отражаемых от объекта. С точки зрения реализма изображения этот алгоритм превосходит остальные, позволяя визуализировать тени, эффекты прозрачности, преломления, отражения. Плюсом является и то, что данный метод позволяет изобразить гладкие объекты без аппроксимации их полигональными поверхностями.

**Плюсы:**

* + Хорошо моделирует оптические эффекты
  + Может быть ускорен параллельными вычислениями

**Минусы:**

* + Очень ресурсозатратен
  + Сложен в реализации

**Алгоритм, использующий Z буфер**

Несомненным плюсом данного алгоритма может являться его простота, которая не мешает решению задачи удаления поверхностей и визуализации их пересечения. В этом алгоритме не тратится время на сортировку элементов сцены, что дает преимущество в скорости работы. Особенно полезным это может стать при большом количестве объектов в сцене.

Так как размер синтезируемого изображения сравнительно мал, затраты по памяти, при хранении информации о каждом пикселе, в данном алгоритме незначительны для современных компьютеров.

К недостаткам данного алгоритма можно отнести то, что требуются большие затраты памяти для хранения информации о каждом пикселе, учитывая размер синтезируемого изображения. Также минусом является то, что непосредственно в сам алгоритм z-буфера нельзя внести тени и эффект прозрачности, необходима модернизация, добавляющая дополнительные буферы теней и прозрачности, что еще больше увеличивает расход памяти.

**Плюсы:**

* + Прост в реализации
  + Не требуется сортировка элементов сцены

**Минусы:**

* + Требует хранение информации о каждом пикселе
  + Недостаточно детально моделирует отражение и преломление

**Алгоритм Робертса**

Алгоритм Робертса работает в объектном пространстве и может быть применен для изображения множества выпуклых многогранников. Алгоритм состоит из трех больших этапов: удаления нелицевых граней для каждого тела, удаления видимых ребер данного тела, экранируемых другими телами сцены, удаление видимых ребер при взаимном “протыкании” тел.

Серьезным недостатком является вычислительная трудоемкость алгоритма. В теории она растет как квадрат количества объектов. Поэтому при большом количестве фигур, находящихся в сцене, этот алгоритм будет показывать себя, как недостаточно быстрый. Можно использовать разные оптимизации для повышения эффективности, например сортировку по z. Минусом является и то, что алгоритм не позволяет визуализировать тени, зеркальные эффекты и преломление света. Использование растровых дисплеев, также снижает интерес к данному алгоритму, т.к. он работает в объектном пространстве.

Преимуществом данного алгоритма является точность вычислений. Она достигается за счет работы в объектном пространстве, в отличии от большинства других алгоритмов.

Алгоритм Робертса является очень точным в силу того, что вычисления производятся в объектном пространстве, однако его вычислительная сложность слишком велика. В силу того, что асимптотическая сложность алгоритма равна **O(),** где n – количество объектов, при большом количестве квантов жидкости алгоритм будет слишком сложен для использования.

**Плюсы:**

* + Очень точен

**Минусы:**

* + Сильно теряет в производительности при большом количестве объектов
  + Сложен в реализации
  + Недостаточно детально моделирует отражение и преломление

**Алгоритм Варнока**

Алгоритм Варнока основывается на рекурсивном разбиении экрана. В зависимости от расположения объектов это может стать, как положительной, так и отрицательной стороной алгоритма. Чем меньше пересечений объектов, тем быстрее алгоритм завершит свою работу. Этот алгоритм плох по причине акцента на совсем не нужных задачах построения.

**Плюсы:**

* + Так как не пересечений, весьма эффективен

**Минусы:**

* + Недостаточно детально моделирует отражение и преломление
  + Нет конкретной реализации

**Вывод**: из всех рассмотренных выше алгоритмов наилучшим выбором является **алгоритм обратной трассировки**. Динамическая визуализация в случае, рассматриваемом в данной работе, не обязана быть наиболее эффективна по времени обработки, по причине возможности пререндеринга объектов сцены.

# **Анализ методов закрашивания**

Существует не мало методов закрашивания, каждый из которых имеет как плюсы, так и минусы

**Простая закраска**

В случае использования простой закраски каждая грань закрашивается одним уровнем интенсивности согласно закону Ламберта.

**Плюсы:**

* + Наименее ресурсоёмкий метод закраски

**Минусы:**

* + Плохо учитывает оптические эффекты
  + Резкий переход цветов на гранях
  + Плохо учитывает отражение

**Закраска по Гуро**

Данный метод закраски использует билинейную интерполяцию интенсивностей (иными словами, мы вычисляем интенсивности для вершин и получаем значения для граней интерполируя нормали вершин).

**Плюсы:**

* + Не очень ресурсоёмок
  + Неплохо имитирует криволинейные поверхности

**Минусы:**

* + Не учитывает кривизну поверхности
  + Недостаточно хорошо обрабатывает световые эффекты

**Закраска по Фонгу**

Данный метод закраски использует билинейную интерполяцию нормалей, что позволяет улучшить результирующую картинку за счёт повышения алгоритмической сложности.

Расчёт освещения по Фонгу требует вычисления цветовой интенсивности трёх компонент освещения: фоновой, рассеянной и глянцевых бликов.

**Плюсы:**

* + Создаёт наиболее реалистичные изображения
  + Наилучшим образом имитирует зеркальные блики

**Минусы:**

* + Ресурсоёмок

Вывод: в качестве метода закрашивания был выбран метод **закраски по Фонгу** из-за реалистичного изображения.

## 

## Анализ алгоритмов построения теней

В силу выбора метода **трассировки лучей** вопрос построения теней оказывается решённым: пиксел затенён, если луч попадает на объект, и позже не попадает ни в объект, ни в источник света.

## Анализ моделей освещения

Модели освещения делятся на локальные и глобальные. Локальные модели освещения не учитывают перенос света между поверхностями. Рассматривается свет только от явных точечных источников света, а взаимодействие ограничивается только однократным отражением света от непрозрачной поверхности. Глобальные модели освещения учитывают световое взаимодействие всех объектов сцены. В рамках этих моделей рассматриваются такие вопросы, как многократное отражение и преломление света, рассеянное освещение.

**Фотонная модель освещения**

В фотонной модели освещения источник представляет собой плоскость, излучающую фотоны. По физическим законам определяется интенсивность и направление движения этих фотонов, которые определяют освещенность тех или иных граней.

Преимуществом данной модели освещения является то, что получается реалистичное изображение, соответствующее физическим законам (передает свет и материал объекта)

Недостатком данной модели освещения являются довольно сложные вычисления, вследствие чего ухудшается скорость визуализации.

**Плюсы:**

* + Реалистичное изображение
  + Соответствие физическим законам

**Минусы:**

* + Ресурсоёмок
  + Сложные вычисления

**Модель Ламберта**

Модель Ламберта позволяет симулировать диффузное освещение. Метод основан на том, что свет при попадании на поверхность рассеивается равномерно во все стороны. Таким образом, освещенность в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения. Данная модель освещения является одной из самых простых. Также она очень часто используется в комбинации других моделей, так как практически в любой другой модели освещения можно выделить диффузную составляющую.

Преимуществом, несомненно, является простота данной модели, а также она удобна для анализа свойств других моделей (за счет того, что ее легко выделить из любой модели и анализировать оставшиеся составляющие)

Очевидным недостатком является то, что данная модель моделирует лишь диффузное освещение.

**Плюсы:**

* + Анализ свойств других моделей

**Минусы:**

* + Ограниченность

**Модель Фонга**

Модель Фонга - классическая модель освещения. Модель представляет собой комбинацию диффузной составляющей (модели Ламберта) и зеркальной составляющей и работает таким образом, что кроме равномерного освещения на материале может еще появляться блик, местонахождение которого определяется из закона равенства углов падения и отражения.

Расчёт освещения по Фонгу требует вычисления цветовой интенсивности трёх компонент освещения: фоновой, рассеянной и глянцевых бликов.

(

- вектор нормали к поверхности в точке

- направление проецирования (направление на источник света)

- направление на наблюдателя

- коэффициент фонового освещения

- коэффициент зеркального освещения

- коэффициент диффузного освещения

Преимущество модели Фонга в том, что алгоритм достаточно прост в реализации по сравнению с фотонной моделью освещения, хотя и сложнее модели Ламберта, так как по сути является расширением данной модели. Также, модель Фонга улучшает визуальные качества сцены, по сравнению с моделью Ламберта, добавляя в нее блики.

Главным недостатком данной модели является то, что многие оптические эффекты либо не учитываются, либо рассчитываются с сильным приближением.

**Плюсы:**

* + Достаточно прост
  + Наилучшим образом имитирует зеркальные блики

**Минусы:**

* + Ресурсоёмок

**Вывод:**так как объектами сцены являются различные фигуры, имеющие не только шероховатую поверхность, лучшим выбором будет **модель освещения Фонга**

1. Конструкторская часть

Необходимо детально описать принцип работы программы

* 1. Общий алгоритм решения
* Задать объекты сцены
* Задать источники света и исходное положение наблюдателя
* Вычисление луча «камера - пиксель» (для трассировщика лучей)
* Для каждого луча
  + Нахождение ближайшей к камере грани ближайшей фигуры
  + Вычисление текстурной координаты
  + Вычисление степени освещенности
  + Вычисление затенения

## Алгоритм обратной трассировки лучей

1. Для каждого источника света пустить лучи

2. Для каждого луча

2.1 Найти пересечение луча с объектом

2.2 Ввести дерево луча

2.3 Если есть отражённая или преломлённая компонента

2.3.1 Добавить её (их) в дерево луча

2.4 Иначе, прекратить трассировку луча

2.5 Получить для точки пересечения интенсивность

3. Отобразить результат

## 2.3 Метод освещения по Фонгу

Основная идея модели Фонга заключается в предположении, что освещенность каждой точки тела разлагается на три компоненты:

* фоновое освещение (ambient);
* рассеянный свет (diffuse);
* бликовая составляющая (specular).

Свойства источника определяют мощность излучения каждой из этих компонент, а свойства материала поверхности определяют ее способность воспринимать каждый вид освещения.

Фоновое освещение - постоянная в каждой точке величина надбавки к освещению. Фоновая составляющая освещения вычисляется как:

Ia = kaia , где Ia - свойство материала воспринимать фоновое освещение, ka - фоновая составляющая освещенности в точке, ia - мощность фонового освещения.

Рассеянный свет при попадании на поверхность рассеивается равномерно во все стороны. При расчете рассеянной составляющей освещения учитывается только нормаль к поверхности и направление на источник света. Рассеянная составляющая рассчитывается по закону косинусов:

Id = kd\*)id , где Id - рассеянная составляющая освещенности в точке, kd - свойство материала воспринимать рассеянное освещение, - мощность фонового освещения, - вектор нормали в точке, id - мощность рассеянного освещения.

1. Техническая часть
   1. Выбор языка программирования и среды разработки

В качестве основного языка для реализации программного продукта будет использоваться язык C++. Выбор этого языка программирования продиктован, прежде всего, значительным объёмом ресурсоёмких вычислений, производимых как на этапе моделирования жидкости, так и на этапе визуализации. Значительным плюсом является факт того, что С++ является компилируемым языком со статической типизацией. Это позволяет использовать имеющиеся мощности максимально эффективно.

Более того, самые трудоёмкие и сложные вычисления можно реализовать с использованием многопоточности, что позволит быстрее разобраться с процессом моделирования растекания, упрощая большое количество однотипных действий в том случае, если мы можем правильно их разделить на логические части. Другая часть часто используемых вычислений может быть ускорена с использованием ассемблерных вставок: эта возможность языка C++ будет уместна далеко не везде, но при некоторых вычислениях она позволяет значительно ускорить обработку данных.

Стоит отметить, что всеми вышеописанными преимуществами обладает и язык C, который раскрывает некоторые из этих преимуществ лучше в силу меньшей своей функциональности и более низкоуровневой области применения. Однако язык C++ позволяет разрабатывать программное обеспечение с использованием парадигмы ООП. Парадигма ООП является промышленным стандартом на данный момент в силу простоты поддержки и модернизации программного обеспечения без необходимости полностью перерабатывать исходный программный код. Более того, задачи визуализации требуют использования ряда понятий, которые хорошо выделяются в классы. Более того, многие паттерны проектирования хорошо сочетаются с задачей визуализации в силу возможности построения строгой иерархии.

Ещё одним значимым плюсом языка С++ является наличие большого количества библиотек и фреймворков, облегчающих работу с интерфейсами и многопоточностью. Фреймворк Qt/C++ является очень удобным для реализации приложений с интерфейсом.

* 1. Интерфейс программы

Интерфейс программы состоит из основного окна, которое содержит изображение, строку меню и строку управления новыми объектами, и дополнительного окна управления источниками света.

Меню служит для вызова окна управления источниками света, загрузки/сохранения сцены, отмены добавления объекта и выхода. Масштаб изображения и позиция камеры меняются при помощи мыши.

* 1. Листинг кода

Листинг 1. Реализация рендеринга

**public** void Render**()**

**{**

Vec3d D **=** **null;**

Vec3d color **=** **null;**

int recursion\_depth **=** 3**;**

**for** **(**int x **=** **-**scene**.**size**.**Width **/** 2**;** x **<** scene**.**size**.**Width **/** 2**;** x**++)**

**{**

**for** **(**int y **=** **-**scene**.**size**.**Height **/** 2**;** y **<** scene**.**size**.**Height **/** 2**;** y**++)**

**{**

D **=** CanvasToViewport**(**x**,** y**);**

TransformPrimitive**.**MultiplyMV**(**rotation\_matr**,** **ref** D**);**

color **=** TraceRay**(**O**,** D**,** 1**,** Double**.**PositiveInfinity**,** recursion\_depth**,** x**,** y**);**

PutPixel**(**x**,** y**,** Clamp**(**color**));**

**}**

**}**

**}**

Листинг 2. Вспомогательная функция перевода

**public** Vec3d CanvasToViewport**(**int x**,** int y**)**

**{**

**return** **new** Vec3d**(**x **\*** **(**double**)**Vw **/** scene**.**size**.**Width**,** y **\*** **(**double**)**Vh **/** scene**.**size**.**Height**,** d**);**

**}**

Листинг 3. Вспомогательная функция перевода

**public** void PutPixel**(**int x**,** int y**,** Color color**)**

**{**

int x\_ **=** scene**.**size**.**Width **/** 2 **+** x**;**

int y\_ **=** scene**.**size**.**Height **/** 2 **-** y **-** 1**;**

**if** **(**x\_ **<** 0 **||** x\_ **>=** scene**.**size**.**Width **||** y\_ **<** 0 **||** y\_ **>=** scene**.**size**.**Height**)**

**{**

**return;**

**}**

bmp**.**SetPixel**(**x\_**,** y\_**,** color**);**

**}**

Листинг 4. Определение ближайшего объекта

**public** void ClosestIntersection**(ref** Primitive closest\_object**,** **ref** double closest\_t**,** Vec3d O**,** Vec3d D**,** double t\_min**,** double t\_max**)**

**{**

List**<**Primitive**>** sceneObject **=** scene**.**GetSceneObjects**();**

double t1 **=** 0**;**

double t2 **=** 0**;**

**for** **(**int i **=** 0**;** i **<** sceneObject**.**Count**;** i**++)**

**{**

**if** **(**sceneObject**[**i**]** **is** Sphere**)**

**{**

IntersectRaySphere**(**O**,** D**,** **(**Sphere**)**sceneObject**[**i**],** **ref** t1**,** **ref** t2**);**

**}**

**else**

**{**

**return;**

**}**

**if** **(**t1 **<** closest\_t **&&** t\_min **<** t1 **&&** t1 **<** t\_max**)**

**{**

closest\_t **=** t1**;**

closest\_object **=** sceneObject**[**i**];**

**}**

**if** **(**t2 **<** closest\_t **&&** t\_min **<** t2 **&&** t2 **<** t\_max**)**

**{**

closest\_t **=** t2**;**

closest\_object **=** sceneObject**[**i**];**

**}**

**}**

**}**

Листинг 4. Определение цвета пиксела

**public** Vec3d TraceRay**(**Vec3d O**,** Vec3d D**,** double t\_min**,** double t\_max**,** int depth**,** int x**,** int y**)**

**{**

double closest\_t **=** Double**.**PositiveInfinity**;**

Primitive closest\_object **=** **null;**

ClosestIntersection**(ref** closest\_object**,** **ref** closest\_t**,** O**,** D**,** t\_min**,** t\_max**);**

**if** **(**closest\_object **==** **null)**

**{**

x **+=** 330**;**

y **+=** 330**;**

Color color **=** scene**.**nebo**.**GetPixel**(**x**,** y**);**

**return** **new** Vec3d**(**color**.**R**,** color**.**G**,** color**.**B**);**

**}**

Vec3d P **=** **new** Vec3d**(**O**.**x **+** closest\_t **\*** D**.**x**,** O**.**y **+** closest\_t **\*** D**.**y**,** O**.**z **+** closest\_t **\*** D**.**z**);**

Vec3d N **=** **new** Vec3d**(**P**.**x **-** closest\_object**.**C**.**x**,** P**.**y **-** closest\_object**.**C**.**y**,** P**.**z **-** closest\_object**.**C**.**z**);**

double k **=** 1.0 **/** Vec3d**.**Length**(**N**);**

N **=** **new** Vec3d**(**k**\***N**.**x**,** k **\*** N**.**y**,** k **\*** N**.**z**);**

Vec3d V **=** **new** Vec3d**(-**D**.**x**,** **-**D**.**y**,** **-**D**.**z**);**

double intensity **=** ComputeLighting**(**P**,** N**,** V**,** closest\_object**.**specular**);**

Vec3d localColor **=** **new** Vec3d**(**intensity **\*** closest\_object**.**color**.**x**,** intensity **\*** closest\_object**.**color**.**y**,** intensity **\*** closest\_object**.**color**.**z**);**

**return** localColor

**}**

Приложение

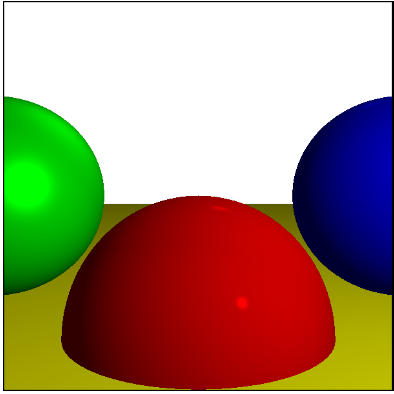


Рисунок 1. *Пример работы ранней версии программы*

# Заключение

Во время прохождения практики были изучены алгоритмы синтеза трёхмерного изображения и отсечения невидимых линий. Был проведён анализ существующих алгоритмов, на основе которого были подобраны алгоритмы, подходящие под требования исходного задания.

Более того, в ходе выполнения задания были рассмотрены методы моделирования света и его свойств, изучены варианты применения данных методов для конструктора сцены объектов.

Список литературы

1. Элементы глобального освещения сцены. Классическая трассировка лучей, В. Е. Турлапов, Д. К. Боголепов [Электронный ресурс]. – Режим доступа: <http://www.graph.unn.ru/rus/materials/CG/CG15_RayTracing.pdf> (дата обращения 15.07.20)
2. Дёмин А. Ю. Основы компьютерной графики: Учебное пособие. – Томск: Изд-во ТПУ, 2011. – 191 с
3. Проблемы трассировки лучей – из будущего в реальное время. [Электронный ресурс]. – Режим доступа: <https://nvworld.ru/articles/ray-tracing> (дата обращения 15.07.20)
4. Raytracing – царь света и теней, Лев Дымченко [Электронный ресурс] Режим доступа: <https://old.computerra.ru/206167/> (дата обращения 13.07.20)
5. Васильев В. Е., Морозов А. В. Компьютерная графика: Учебное пособие. – СПб.: СЗТУ, 2005 – 101 с.