



# miniRT

Мой первый RayTracer с miniLibX

*Описание: Этот проект представляет собой введение в прекрасный мир Raytracing. После завершения вы сможете визуализировать простые изображения, созданные компьютером, и вы больше никогда не будет бояться применять математические формулы.*

## **СОДЕРЖАНИЕ**

I	<b>Вступление</b>	2
II	<b>Общие инструкции</b>	3
III	<b>Обязательная часть - бонусная</b>	4
IV	<b>часть miniRT</b>	9
V	<b>Примеры</b>	11

# Глава I

## Вступление

Когда дело доходит до рендеринга трехмерных компьютерных изображений, есть два возможных подхода: « Растеризация », Который используется почти всеми графическими движками из-за его эффективности и« Трассировка лучей. »

Значок « Трассировка лучей », Впервые разработанный в 1968 году (но с тех пор усовершенствованный), даже сегодня является более дорогостоящим в вычислениях, чем метод « Растеризация ». Метод. В результате он плохо адаптирован к сценариям использования в реальном времени, но обеспечивает гораздо более высокую степень визуального реализма.



Рисунок I.1: Изображения выше созданы с использованием техники трассировки лучей. Впечатляет, не правда ли?

Прежде чем вы сможете даже начать создавать такую высококачественную графику, вы должны освоить основы: miniRT ваш первый трассировщик лучей закодирован в C, нормальный и скромный, но функциональный.

Основная цель miniRT заключается в том, чтобы доказать себе, что вы можете реализовать любые математические или физические формулы, не будучи математиком, мы реализуем здесь только самые основные функции трассировки лучей, поэтому просто сохраняйте спокойствие, сделайте глубокий вдох и не паникуйте! После этого проекта вы сможете показывать ооо красивых картинок, чтобы оправдать количество часов, которые вы проводите в школе.

# Глава II.

## Общие инструкции

- Ваш проект должен быть написан в соответствии с Нормой. Если у вас есть бонусные файлы / функции, они будут включены в проверку норм, и вы получите 0 если внутри есть ошибка нормы.
  - Ваши функции не должны завершаться неожиданно (ошибка сегментации, ошибка шины, двойное освобождение и т. д.), За исключением неопределенного поведения. Если это произойдет, ваш проект будет считаться нефункциональным и получит 0 во время оценки.
  - При необходимости все пространство памяти, выделенное кучей, должно быть должным образом освобождено. Утечки недопустимы.
  - Если предмет требует этого, вы должны отправить Makefile который скомпилирует ваши исходные файлы в требуемый вывод с помощью флагов - Стена, -Wextra а также - Веррор, и ваш Make-файл не должен повторно связываться.
  - Ваш Makefile должен как минимум содержать правила \$( ИМЯ), all, clean, fclean а также ре.
  - Чтобы превратить бонусы в свой проект, вы должны включить правило бонус в ваш Make-файл, который добавит все различные заголовки, библиотеки или функции, запрещенные в основной части проекта. Бонусы должны быть в другом файле \_ бонус. {с / h}.
- Оценка обязательной и бонусной части проводится отдельно.
- Если ваш проект позволяет вам использовать свой libft, вы должны скопировать его источники и связанные с ним Makefile в libft папка с соответствующим Make-файлом. Ваш проект Makefile должен скомпилировать библиотеку, используя ее Makefile, затем скомпилируйте проект.
  - Мы рекомендуем вам создавать тестовые программы для вашего проекта, даже если эта работа **не должны быть отправлены и не будут оцениваться**. Это даст вам возможность легко проверить свою работу и работу ваших коллег. Вы найдете эти тесты особенно полезными во время защиты. Действительно, во время защиты вы можете использовать свои тесты и / или тесты партнера, которого вы оцениваете.
  - Отправьте свою работу в назначенный репозиторий git. Оцениваться будет только работа в репозитории git. Если DeepThreadt назначен для оценки вашей работы, это будет сделано после ваших оценок коллег. Если во время выставления оценок Deepoughtt в каком-либо разделе вашей работы произойдет ошибка, оценка будет остановлена.

## Глава III.

### Обязательная часть - miniRT

<b>Название программы</b>	miniRT
<b>Сдать файлы</b>	Все ваши файлы
<b>Сделать файл</b>	all, clean, fclean, re, бонус сцена в
<b>Аргументы</b>	формате *.rt
<b>Внешние функции.</b>	<ul style="list-style-type: none"><li>• открыть, закрыть, прочитать, написать, printf, malloc, бесплатно, perror, strerror, exit</li><li>• Все функции математической библиотеки (-lm man man 3 math)</li><li>• Все функции MinilibX</li></ul>
<b>Libft авторизован</b>	да
<b>Описание</b>	Цель вашей программы - генерировать изображения с использованием протокола трассировки лучей. Каждое из этих компьютерных изображений будет представлять сцену, видимую с определенного угла и положения, определяемого простыми геометрическими объектами, и каждое со своей собственной системой освещения.

Ограничения следующие:

- Ты должен использовать miniLibX. Либо версия, доступная в операционной системе, либо из ее исходников. Если вы решите работать с источниками, вам нужно будет применить те же правила для своих libft как написано выше в Общий инструкции часть.
- Управление вашим окном должно оставаться плавным: переход на другое окно, сворачивание и т. д.
- Вам понадобятся как минимум эти 5 простых геометрических объектов: плоскость, сфера, цилиндр, квадрат и треугольник.

- Если возможно, все возможные пересечения и внутренняя часть объекта должны быть обработаны правильно.
- Ваша программа должна иметь возможность изменять размер уникальных свойств объекта: диаметр для сферы, размер стороны для квадрата и ширину и высоту для цилиндра.
- Ваша программа должна уметь применять преобразование перемещения и вращения к объектам, источникам света и камерам (за исключением сфер, треугольников и источников света, которые нельзя повернуть).
- Управление освещением: точечная яркость, жесткие тени, окружающее освещение (объекты никогда не находятся полностью в темноте). С цветными и многоточечными светильниками необходимо обращаться правильно.
- В случае если Глубокая мысль однажды у него есть глаза, чтобы оценить ваш проект, и если вы хотите иметь возможность визуализировать красивые обои для рабочего стола ...  
Вместо того, чтобы открывать окно, ваша программа должна сохранять визуализированное изображение в BMP формат, когда его второй аргумент - " --спаси".
- Если второй аргумент не указан, программа отображает изображение в окне и соблюдает следующие правила:
  - Нажатие ESC должен закрыть окно и полностью выйти из программы.
  - Щелчок по красному крестику на рамке окна должен закрыть окно и полностью выйти из программы.
  - Если заявленный размер сцены больше разрешения дисплея, размер окна будет установлен в зависимости от текущего разрешения дисплея.
  - Если имеется более одной камеры, вы должны иметь возможность переключаться между ними, нажимая клавиши клавиатуры по вашему выбору.
  - Использование изображений принадлежащий minilibX настоятельно рекомендуется.
- Ваша программа должна принимать в качестве первого аргумента файл описания сцены с расширением. rt расширение.
  - Он будет содержать окно / визуализированное изображение размер, что подразумевает ваш miniRT должен иметь возможность отображать в любом положительном размере.
  - Каждый тип элемента может быть разделен одним или несколькими разрывами строки.
  - Каждый тип информации из элемента может быть разделен одним или несколькими пробелами.
  - Каждый тип элемента может быть установлен в файле в любом порядке.
  - Элементы, обозначенные заглавной буквой, могут быть объявлены в сцене только один раз.

◦ Информация о каждом элементе - это, прежде всего, идентификатор типа (состоящий из одного или двух символов), за которым следует вся конкретная информация для каждого объекта в строгом порядке, например:

\* Разрешение:

1920 1080 рэнд

- Идентификатор: **p**
- X размер рендеринга
- Y размер рендеринга

\* Окружающая молния:

A 0,2 255,255,255

- Идентификатор: **A**
- Коэффициент внешней освещенности в диапазоне [0,0,1,0]: **0,2**
- Цвета R, G, B в диапазоне [0-255]: **255, 255, 255**

\* Камера:

c -50,0,0,20 0,0,1 70

- Идентификатор: **c**
- Координаты x, y, z точки обзора: **0,0,0,0,20,6**
- 3D нормализованный вектор ориентации. В диапазоне [-1,1] для каждой оси x, y, z: **0,0,0,0,1,0**
- FOV: горизонтальное поле зрения в градусах в диапазоне [0,180] \*

Light:

л - 40,0,50,0,0,0 0,6 10,0,255

- Идентификатор: **л**
- X, y, z координаты световой точки: **0,0,0,0,20,6**
- Коэффициент яркости света в диапазоне [0,0,1,0]: **0,6**
- Цвета R, G, B в диапазоне [0-255]: **10, 0, 255**

\* Сфера:

зр 0,0,0,0,20,6 12,6 10,0,255

- Идентификатор: **зр**
- X, y, z координаты центра сферы: **0,0,0,0,20,6**
- Диаметр сферы: **12,6**
- Цвета R, G, B в диапазоне [0-255]: **10, 0, 255**

\* Самолет:

```
pl 0,0,0, -10,0 0,0,1,0,0,0 0,0,225
```

- Идентификатор: **pl**

- Координаты x, y, z: **0,0,0, -10,0**

- 3D нормализованный вектор ориентации. В диапазоне [-1,1] для каждой оси x, y, z:  
**0,0,0,0,1,0**

- Цвета R, G, B в диапазоне [0-255]: **0, 0, 255**

\* Квадрат:

```
кв 0,0,0,20,6 1,0,0,0,0 12,6 255,0,255
```

- Идентификатор: **кв**

- Координаты x, y, z центра квадрата: **0,0,0,20,6**

- 3D нормализованный вектор ориентации. В диапазоне [-1,1] для каждой оси x, y, z:  
**1.0,0.0,0.0**

- Размер стороны: **12,6**

- Цвета R, G, B в диапазоне [0-255]: **255, 0, 255**

\* Цилиндр:

```
Сай 50,0,0,0,20,6 0,0,0,1,0 14,2 21,42 10,0,255
```

- Идентификатор: **Сай**

- Координаты x, y, z: **50,0,0,0,20,6**

- 3D нормализованный вектор ориентации. В диапазоне [-1,1] для каждой оси x, y, z:  
**0,0,0,0,1,0**

- Диаметр цилиндра: **14,2**

- Высота цилиндра: **21,42**

- Цвета R, G, B в диапазоне [0,255]: **10, 0, 255**

\* Треугольник:

```
тр 10,0,20,0,10,0 10,0,10,0,20,0 20,0,10,0,10,0 0,0,255
```

- Идентификатор: **тр**

- Координаты x, y, z первой точки: **10,0,20,0,10,0**

- Координаты x, y, z второй точки: **10,0,10,0,20,0**

- Координаты x, y, z третьей точки: **20,0,10,0,10,0**

- Цвета R, G, B в диапазоне [0,255]: **0, 255, 255**

- Пример обязательной части с минималистом. **rt** сцена:

```
p 1920 1080
A 0,2
      255 255 255

с - 50,0,20    0,0,0    70
л - 40,0,30    0,7
      255 255 255

pl 0,0,0
зр 0,0,20
кв 0,100,40   0,0,1,0   30
        255,0,225
Сай 50,0,0,0,20,6 0,0,1,0   14,2 21,42
        42,42,0
        10,0,255
tr 10,20,10   10,10,20   20,10,10
        0,0,255
```

- Если в файле обнаружена какая-либо неправильная конфигурация, программа должна завершиться должным образом и вернуть «Ошибка \ n» с последующим явным сообщением об ошибке по вашему выбору.
- Для защиты было бы идеально иметь целый набор сцен с упором на то, что является функциональным, чтобы облегчить контроль над создаваемыми элементами.

# Глава IV.

## Бонусная часть

Очевидно Трассировка лучей техника может справиться со многими другими вещами, такими как отражение, прозрачность, преломление, более сложные объекты, мягкие тени, каустика, глобальное освещение, отображение рельефа, рендеринг файлов .obj и т. д.

Но для miniRT project, мы хотим упростить ваш первый трассировщик лучей и ваши первые шаги в CGI.

Итак, вот список нескольких простых бонусов, которые вы могли бы реализовать. Если вы хотите получить более крупные бонусы, мы настоятельно рекомендуем вам перекодировать новый трассировщик лучей позже в вашей жизни разработчика, после того, как этот маленький будет готов и полностью функционален.



Рисунок IV.1: Пятно, космический скайбокс и сияющая сфера с текстурой земли и рельефным картированием.



Бонусы будут оцениваться только в том случае, если ваша обязательная часть ИДЕАЛЬНА. Под СОВЕРШЕННЫМ мы, естественно, подразумеваем, что он должен быть полным, что он не может выйти из строя, даже в случае серьезных ошибок, таких как неправильное использование и т. д. По сути, это означает, что если ваша обязательная часть не наберет ВСЕ баллы во время оценивания, ваши бонусы будут быть полностью ИГНОРИРОВАННЫМ.

Список бонусов:

- Нормальное нарушение, например, с использованием синуса, дающего волновой эффект.
- Нарушение цвета: шахматная доска.
- Нарушение цвета: эффект радуги с использованием нормалей объекта.
- Параллельный свет, следующий в точном направлении.
- Составной элемент: Куб (6 квадратов).
- Составной элемент: Пирамида (4 треугольника, 1 квадрат).
- Установка колпачков на баллоны ограниченного размера.
- Еще один объект 2-й степени: Конус, Гиперболоид, Параболоид ...
- Одноцветный фильтр: сепия, фильтры R / G / B ..
- Сглаживание.
- Простая стереоскопия (например, красные / зеленые очки).
- Многопоточный рендеринг.
- Текстурирование сферы: уф-картографирование.
- Обработка текстур карты рельефа.
- Красивый скайбокс.
- Интерактивность клавиатуры (перемещение / вращение) с камерой.
- Интерактивность клавиатуры (перемещение / вращение) с объектами.
- Изменение поворота камеры с помощью мыши.



Вам разрешается использовать другие функции для завершения бонусной части, если их использование оправдано во время вашей оценки. Вы также можете изменить ожидаемый формат файла сцены в соответствии с вашими потребностями. Быть умным!



Чтобы заработать все бонусные баллы, вам необходимо подтвердить как минимум 14 из них, поэтому выбирайте с умом, но будьте осторожны, чтобы не тратить зря времени!

# **Глава V**

## **Примеры**

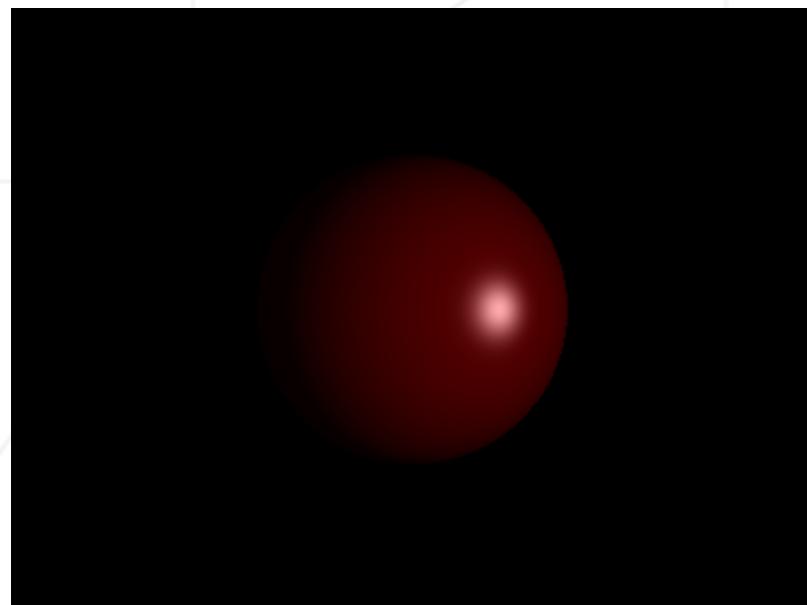


Рисунок V.1: Сфера, одно пятно, немного блеска (необязательно)



Рисунок V.2: Цилиндр, одно пятно

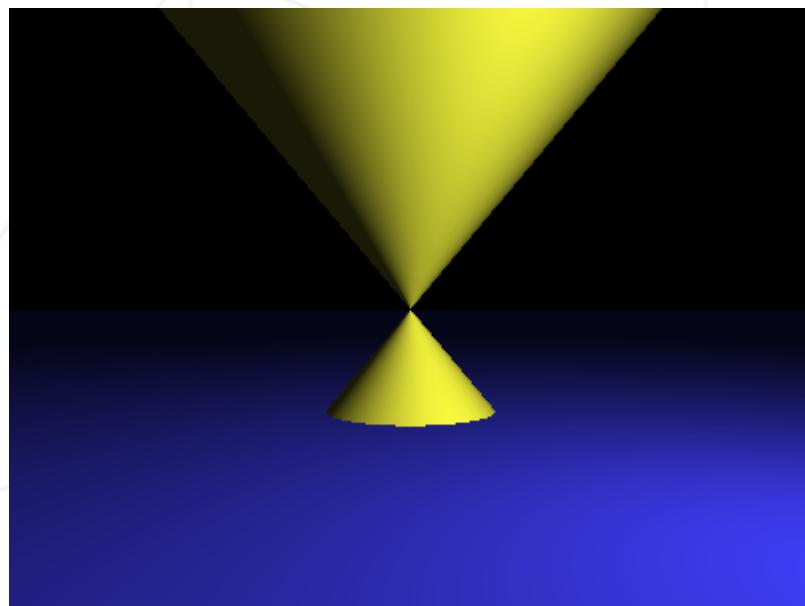


Рисунок V.3: Конус (необязательно), плоскость, одно пятно

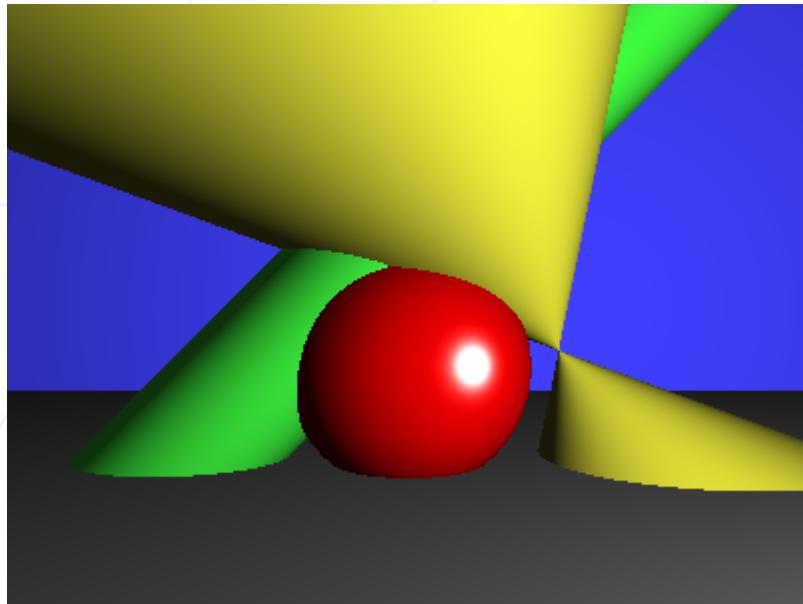


Рисунок V.4: Всего понемногу, включая 2 самолета

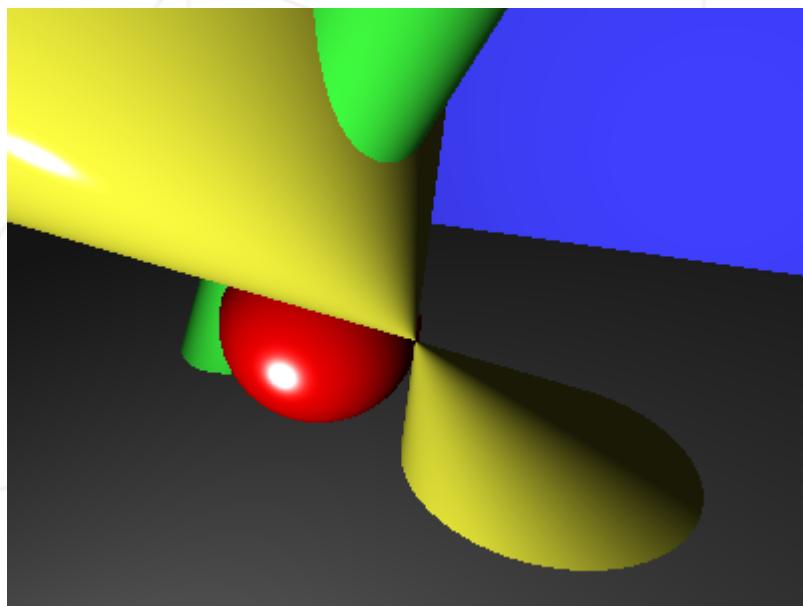


Рисунок V.5: Одна и та же сцена, разные камеры

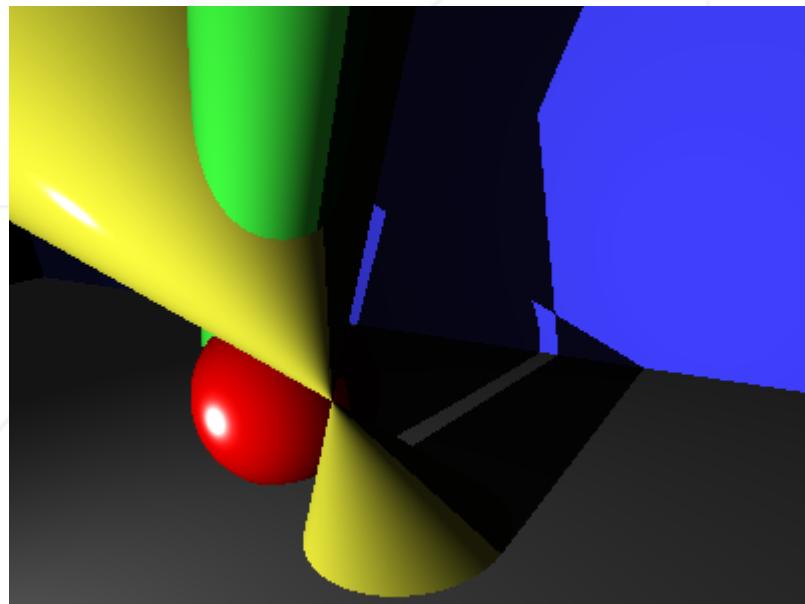


Рисунок V.6: На этот раз с тенями

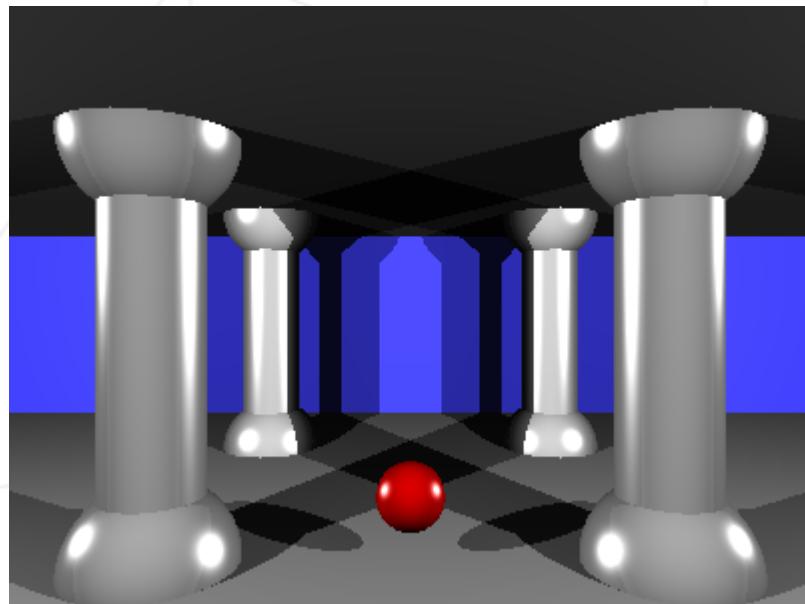


Рисунок V.7: С несколькими точками

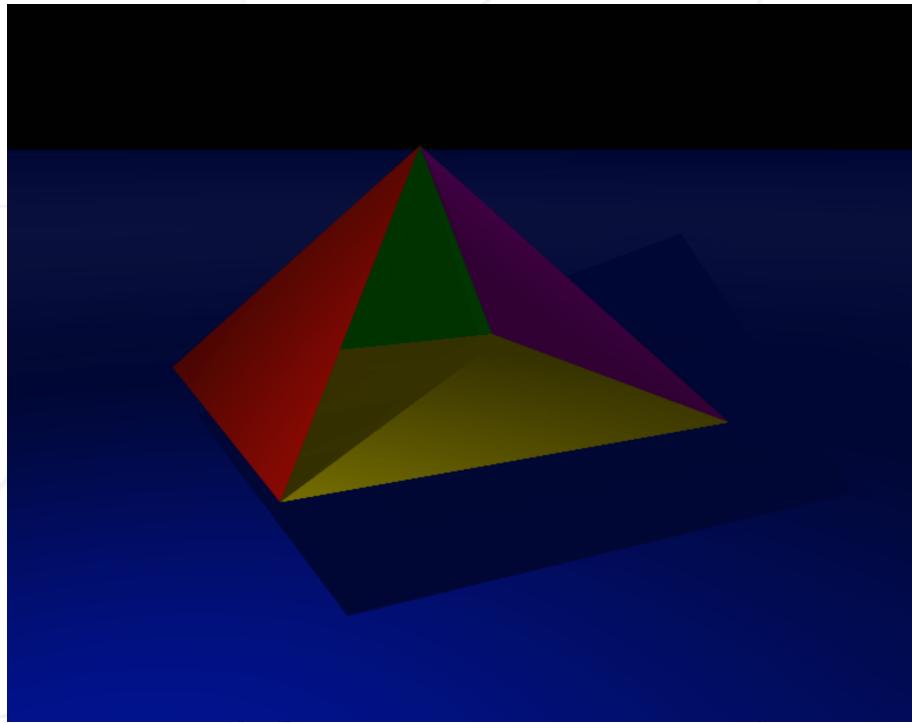


Рисунок V.8: Плоскость, 3 треугольника, 1 квадрат и одно пятно низкой яркости слева

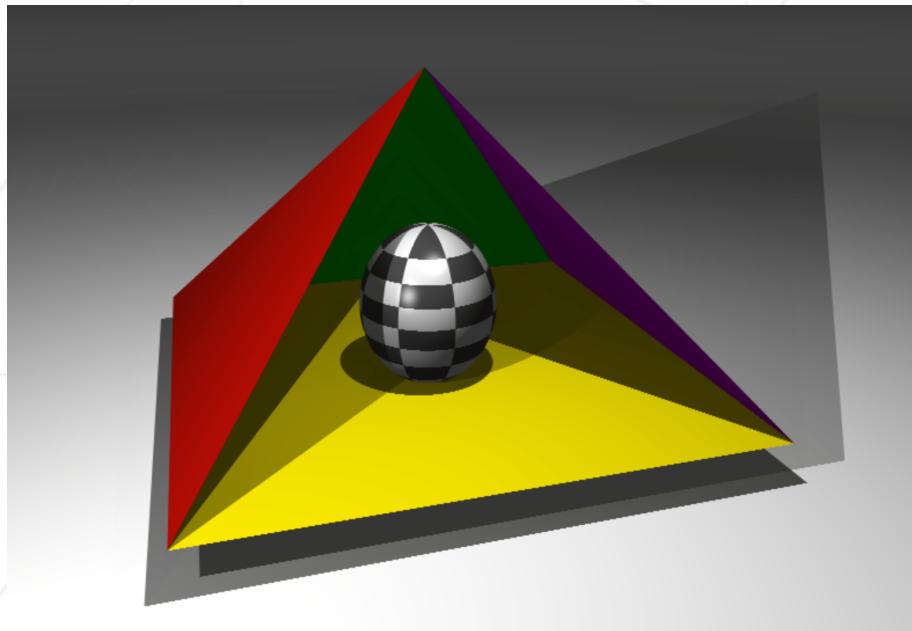


Рисунок V.9: И, наконец, с несколькими пятнами и блестящей клетчатой (необязательной) сферой посередине.