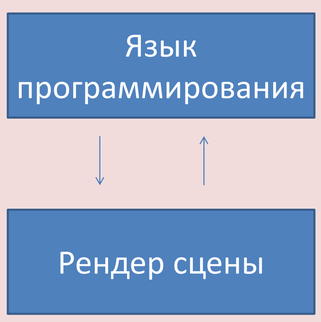
Разработка системы визуализации фотореалистичных трехмерных сцен в реальном времени с использованием GPU

Иванов Тимофей Андреевич 9-1, Уросова Софья Александровна 9-1, Амбросовская Дарья Викторовна 9-2, Григорович Вячеслав Дмитриевич 10-5, Писарев Евгений Александрович 10-5, Синяков Степан Евгеньевич 10-5, Мосягин Олег Сергеевич 11-5

Введение

Проблема рендеринга (отрисовки) фотореалистичных сцен в реальном времени является одной из самых актуальных проблем компьютерной графики на сегодняшний день. Целью проекта является разработка программного обеспечения, позволяющего создавать трехмерные сцены и выводить фотореалистичные изображения с высокой частотой кадров.

Структура проекта



Язык

Язык динамически типизирован. Единственный способ хранения – переменные. В них хранятся выражения и функции. У переменных существует неограниченное количество полей, которые обладают тем же функционалом. Возможна перегрузка операторов путем присвоения в определённое поле новой функции. Реализованы ссылки(значение выражения зависит от текущего значения переменной, на которую ссылаются), что позволяет создавать постоянную зависимость от переменных. Точкой входа является главный файл.

Пример

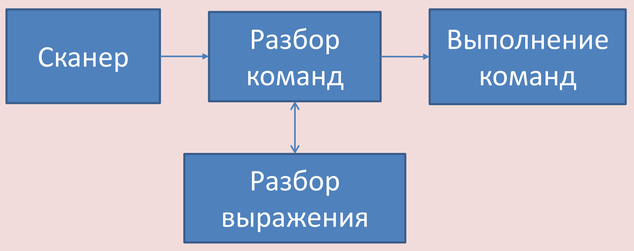
```
include("std.trg");

MyFunc = function()
{
  light = #point_light(#vec3(30));
  #Scene.Add(light);
};

Scene = scene();
SpherePos = vec3(24, 8, 3);

Scene.Add(model("pml30.g3dm"));
Scene.Add(sphere(3.0, SpherePos));
MyFunc();
```

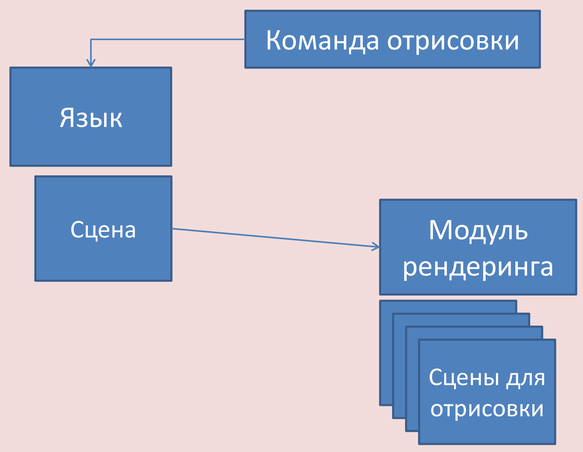
Принцип работы



Работу языка можно разбить на 3 стадии - сканер, синтаксический анализатор и виртуальная машина. Первым работает сканер. Он разбивает исходный код программы на лексемы. Их получает синтаксический анализатор и создаёт команды для виртуальной машины. Это может быть служебное слово(if, while, break…) или выражение(a.b = 2 + 3^9). Выражение хранится как дерево операций. Виртуальная машина выполняет команды и при необходимости считает выражения.

Генерация сцен

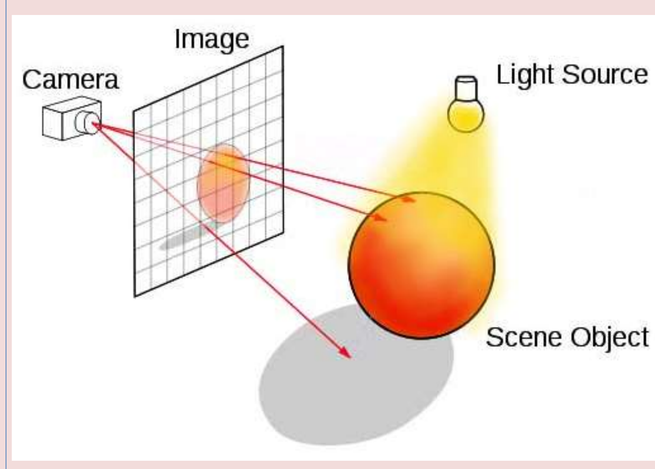
В ходе выполнения виртуальной машиной команд, модуль языка получает необходимые данные для модуля рендеринга. В языке присутствует объект scene, имеющий поля для работы со сценой. С помощью команды Scene.Add в очередь рисования добавляются объекты (источники света и модели). После команды Scene.Render (построение кадра), модуль языка посылает полученные данные модулю рендеринга и продолжает свою работу.



Построение кадра

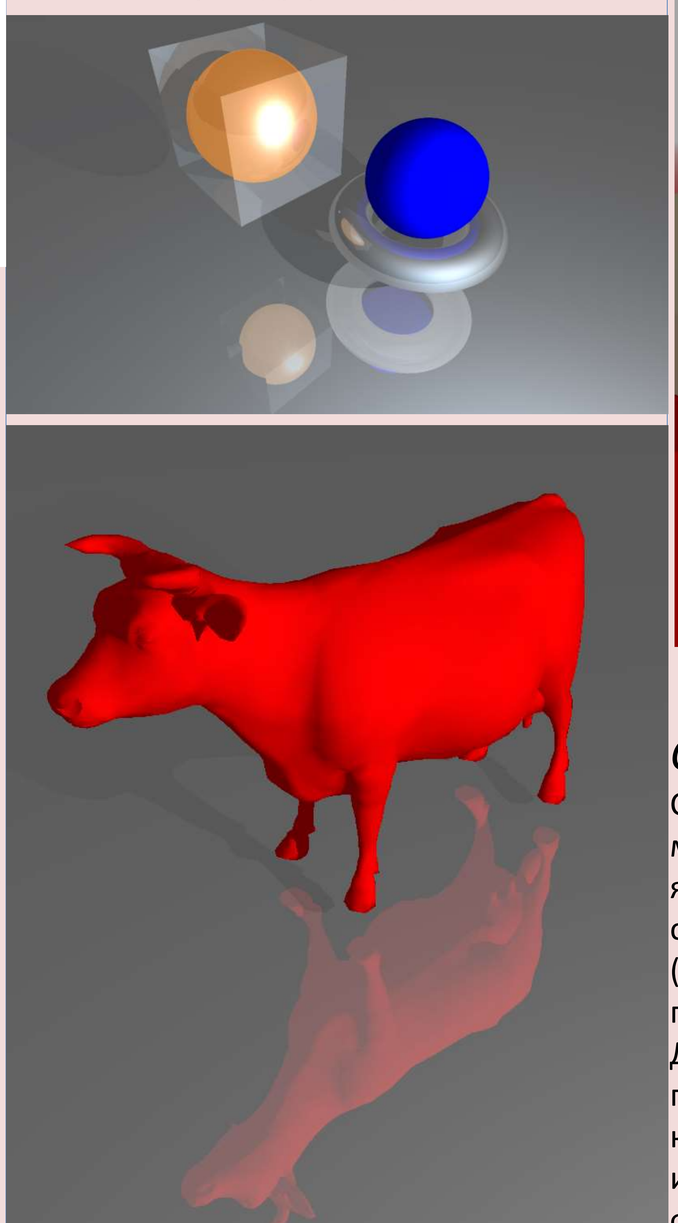
Трассировка лучей

Построение кадра выполняется методом обратной трассировки лучей.



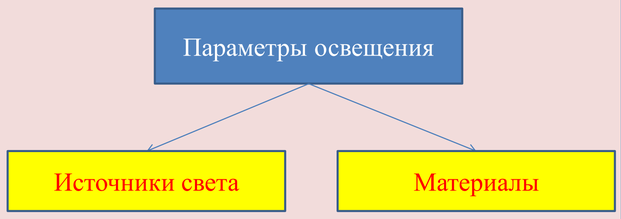
Фигуры

В качестве основной фигуры, из которой состоят все объекты, используется треугольник. Выбор обусловлен тем, что с помощью треугольников можно задать любые геометрические фигуры (сфера, плоскость, куб и др.) и модели.



Освещение

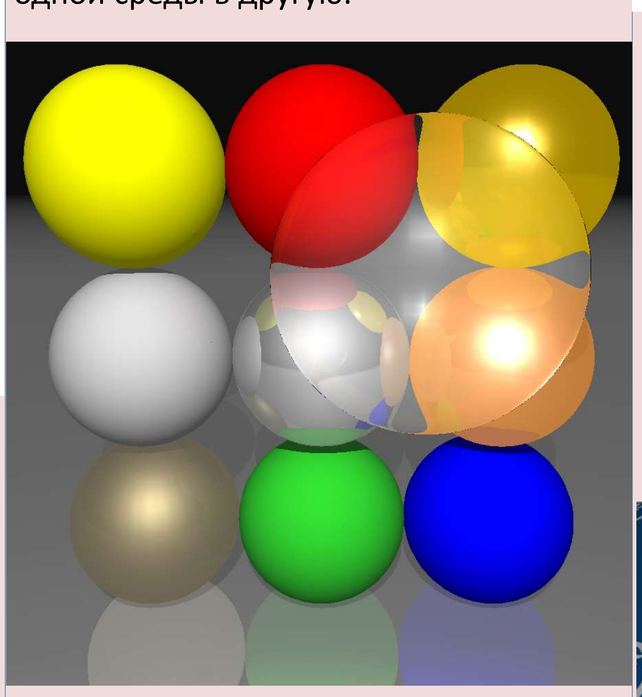
В проекте реализована глобальная система освещения, которая учитывает отражения и преломления. Для расчета локального освещения от каждой лампы используется система BRDF (bidirectional reflectance distribution function).

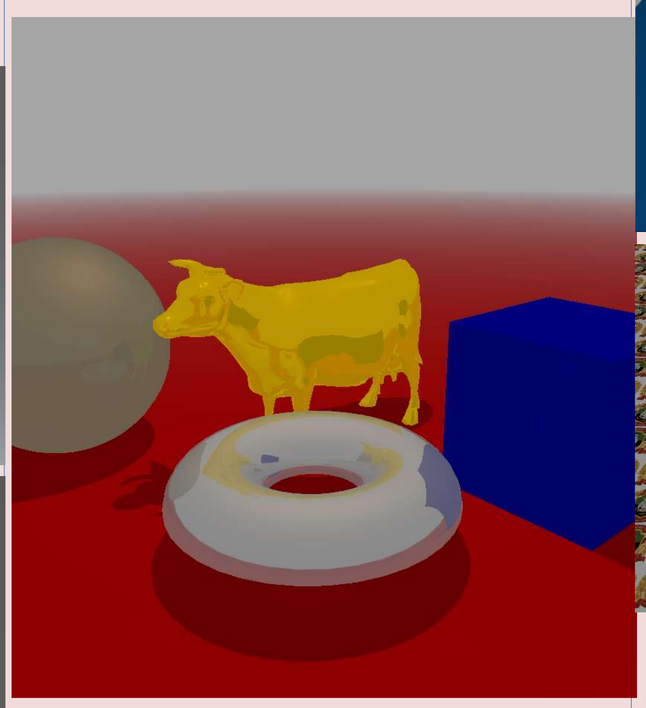


- Цвет
- Точечный
  - -Положение
  - -Коэффициенты затухания
- Направленный
  - -Направление

- Фоновое освещение
- Цвет объекта
- Коэффициент отражения
- Прозрачность
- Наличие металлических свойств
- Гладкость
- Коэффициент преломления

Ещё одним параметром освещения является среда, которая влияет на плотность тумана и коэффициент преломления. Последний влияет на степень преломление при переходе из одной среды в другую.





Сцена

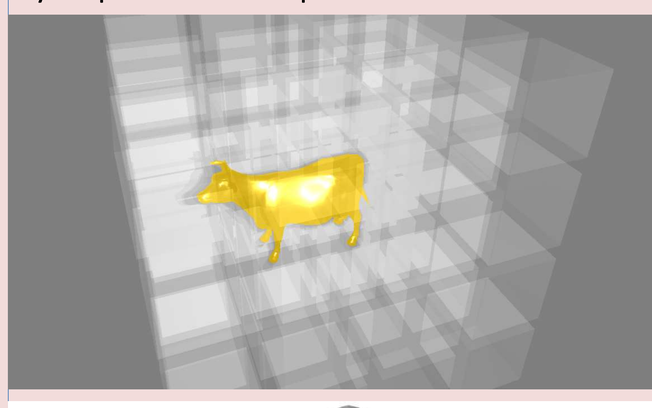
Сцена является связующим звеном между языком и конвейером вывода. В языке она представлена как тип данных, собирающий информацию об объектах (фигурах, источниках света) для построения изображения.

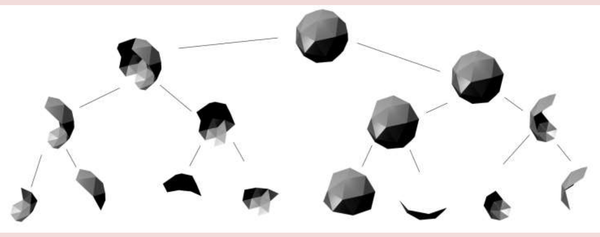
Для модуля рендеринга сцена представляет собой хранилище данных, которые используются при отрисовки изображения. Также из сцен формируется очередь отрисовки.

Регулярная сетка и бинарное дерево

Для оптимизации поиск пересечений пространство сцены разбивается на ограничивающие прямоугольные параллелепипеды (далее воксели). При пересечении с вокселем ищутся пересечения с его внутренними треугольниками.

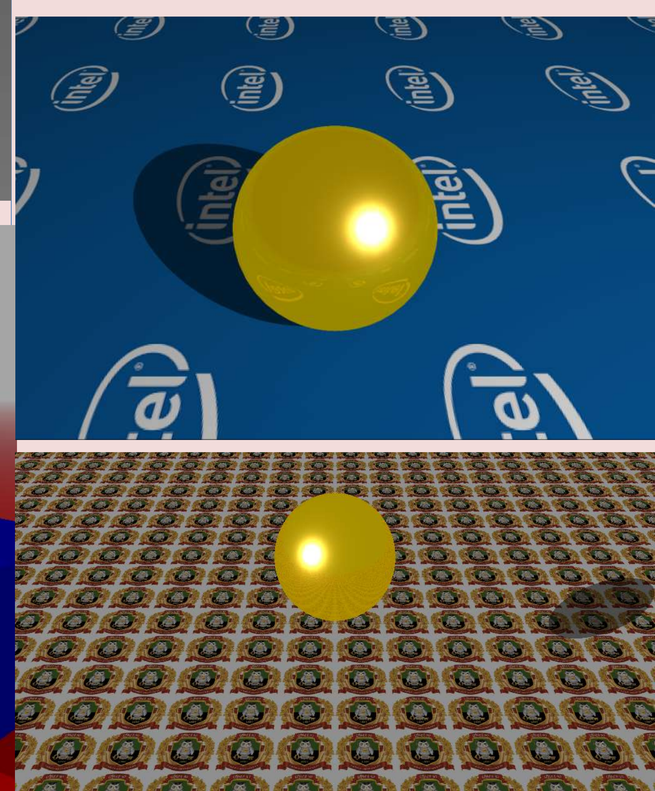
Также для ускорения вычислений объекты представляются в бинарного дерева параллелепипедов («боксов»), вложенных друг в друга. В результате ищутся пересечения с треугольниками одного листа дерева, что значительно ускоряет поиск пересечений.





CUDA

Для уменьшения времени расчёта кадра было решено перенести вычисления на GPU и использовать CUDA API. Алгоритм обратной трассировки лучей легко распараллеливается на видеокарте за счёт использования её ядер, т. к. цвет каждого пикселя считается независимо от всех остальных.



Заключение

Авторам удалось написать программу, позволяющую создавать фотореалистичные изображения на собственном языке программирования. В ходе исследования были изучены алгоритмы, связанные с обратной трассировкой лучей, и методы их реализации. В будущем авторы планируют добавить новые способы оптимизации вычислений, распределенные вычисления по локальной сети, а также улучшить физическое взаимодействие объектов.