Чухарев Алексей Леонидович

Chukharev Alexey Leonidovich

Разработка программы трассировки лучей с использованием технологии CUDA.

Описание метода.

Метод трассировки лучей применяется для построения изображения трехмерной сцены по ее геометрическому описанию. Данная техника хорошо описана во многих источниках, например [1]. Суть технологии заключается в следующем. Из точки наблюдения через каждый пиксель изображения проводятся лучи (в направлении 3-мерной сцены). Далее, если луч попадает на поверхность какого-то объекта сцены, то пиксель закрашивается соответствующим цветом. В противном случае, пиксель приобретает цвет фона. В чуть более сложном варианте, когда присутствуют один или несколько источников света, необходимо учитывать, является ли область на которую попал луч освещенной. Для этого из точки падения луча по направлению ко всем источникам света проводятся вторичные лучи. Если на пути вторичного луча нет других 3d-объектов, то точку можно считать освещенной. В зависимости от того, сколько источников светят на точку, цвет пикселя будет более или менее насыщенным. Если еще немного усложнить задачу, то нужно учитывать отражение лучей от поверхности объекта. В этом случае алгоритм становится рекурсивным, поскольку из каждой точки падения луча выходит отраженный луч, который рассчитывается точно таким же образом.

Применение GPU

Техника трассировки лучей идеально подходит для реализации на графических процессорах, поскольку при заданной 3d-геометрии сцены каждый пиксель может обрабатываться независимо от других. При использовании видеокарт производства Nvidia и технологии CUDA, трехмерные координаты и другие параметры объектов целесообразно разместить в разделяемой памяти. В качестве основного типа данных лучше выбрать вещественный тип с одинарной точностью (float), так как в данной задаче точность цвета пикселя не играет принципиальной роли. Производительность же операций над вещественными числами с одинарной точностью гораздо выше, чем с двойной. По той же причине для ускорения работы программы можно использовать аппаратные функции работы с вещественными числами (так называемые intrinsics-функции). Для увеличения скорости работы программы можно применить технику выравнивания типов данных. Например, определить структуру описывающую тип вектора следующим образом:

```
#define FORCE_ALIGNING

typedef struct{

        float x;

        float y;

        float z;

#ifdef FORCE_ALIGNING

        float dummy;

#endif

}t_vector;
```

В этом случае добавление элемента `dummy` приведет к тому, что адрес следующего элемента будет иметь адрес, кратный 16 байтам.

Описание кода

В качестве задачи ставилось написание программы генерирующей изображение от 5 до 10 сфер, освещаемых одним или двумя источниками света.

В программе были реализованы следующие __device__ функции для работы с векторными типами данных:

- `void vec_sub (*v1,  *v2,  *v3)`
- `void vec_add (*v1, *v2,  *v3)`
- `void vec_scale (scale, *v1, *v2)`
- `float dotproduct (*v1, *v2)`
- `void normalize_vector (*v)`

Следующие функции предназначены для расчета лучей и точек пересечения луча со сферой

- `void compute_ray(*ray, *view_point,  *pixel)`
- `void compute_reflected_ray(*reflected_ray, *incidence_ray, *intersection)`
- `void compute_ray_to_light(*ray, *intersection, *light)`
- `bool sphere_intersection (*ray, *sphere, *intersection)`

Следующая рекурсивная функция осуществляет непосредственную трассировку луча ray, с глубиной рекурсии depth.

- `t_color TraceRay(ray, depth )`

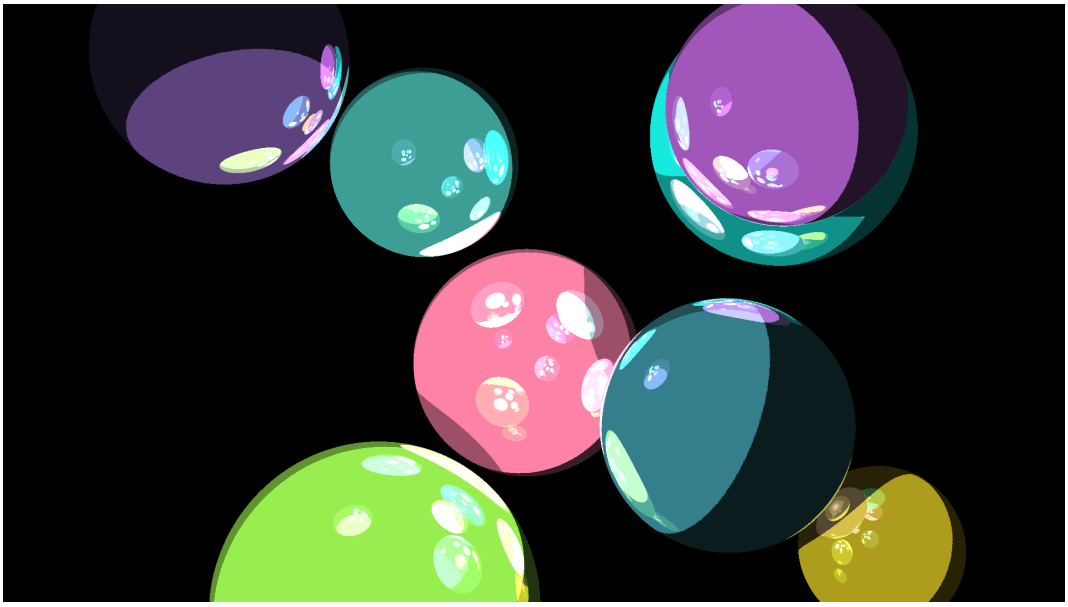
Функция ядро определено следующим образом:

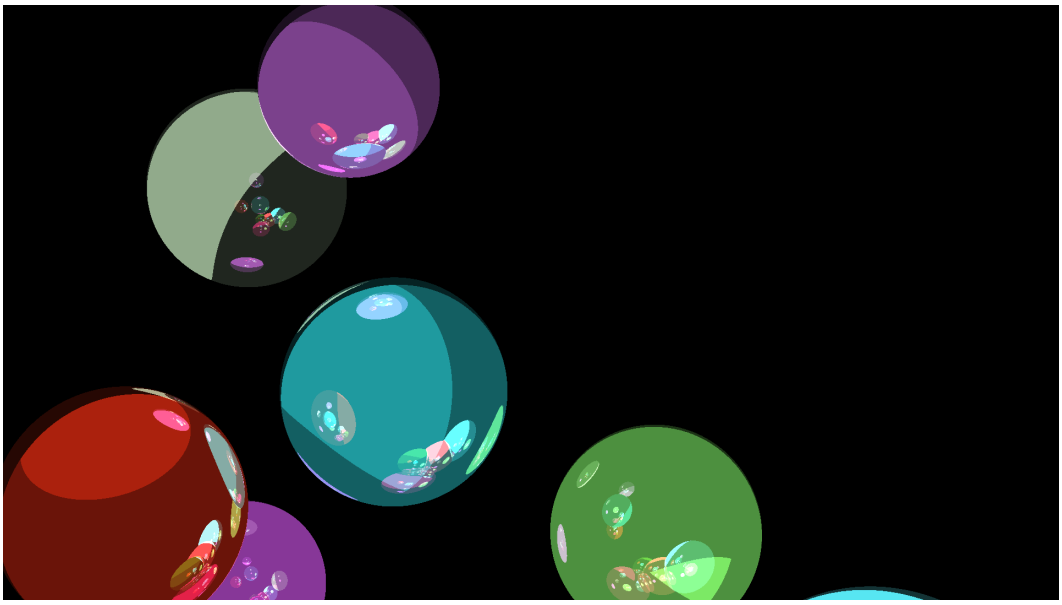
```
__global__ void kernel(unsigned char * dev_image_red,
                       unsigned char * dev_image_blue,
                       unsigned char * dev_image_green,
                       int  height, int width,
                       t_sphere * dev_spheres, int dev_n_spheres,
                       t_light * dev_lights, int dev_n_lights)
```

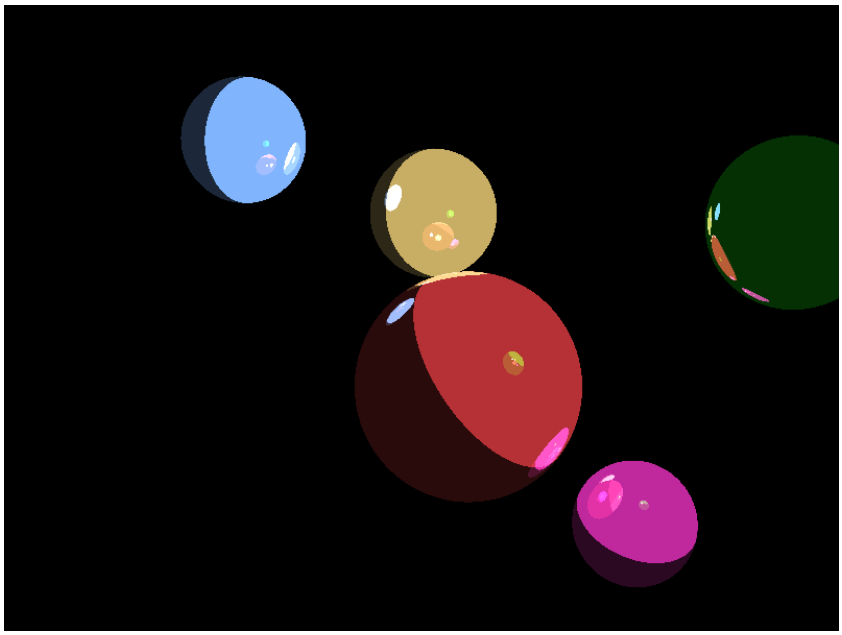
Ядро принимает в качестве параметров указатели на области памяти в которых будут хранится различные компоненты цвета каждого пикселя, разрешение изображения, а также геометрическое описание сцены.

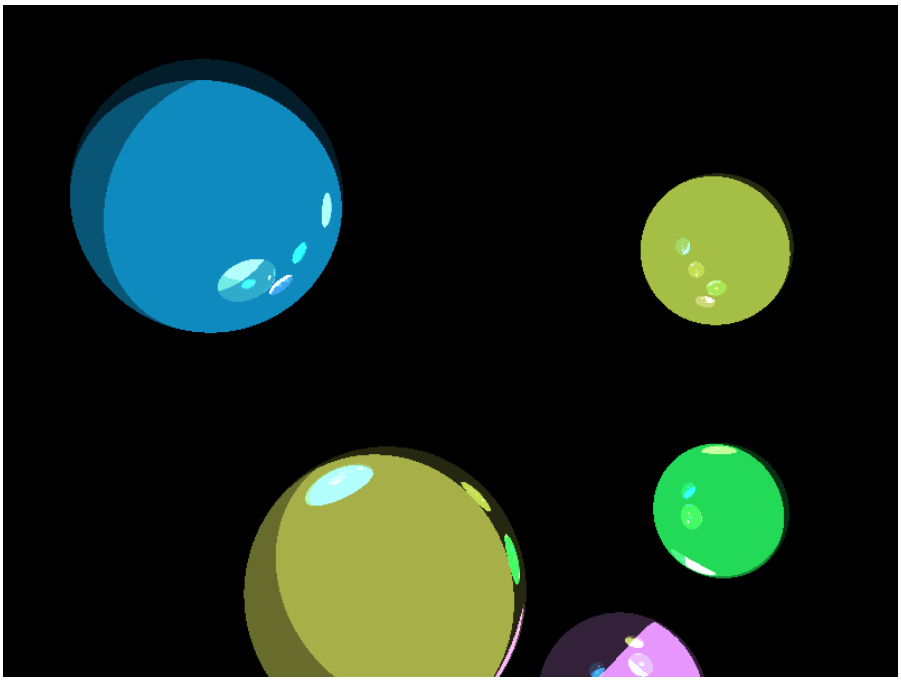
Результаты работы программы

Результатом работы программы должно быть сгенерированное изображение заданного разрешение и случайной геометрией сцены. Ниже приведены несколько иллюстраций работы программы.









Источники

1. https://en.wikipedia.org/wiki/Ray_tracing_graphics