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 лабораторной работе №4 по курсу "Анализ алгоритмов"

Тема Параллельное программирование

Студент Зайцева А.А.

Группа ИУ7-52Б

Преподаватели Волкова Л.Л.

Москва — 2021 г.

Оглавление

Введение

Существуют различные способы написания программ, одним из которых является использование параллельных вычислений. Параллельные вычисления – способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно) [1].

Основная цель параллельных вычислений – уменьшение времени решения задачи. Многие необходимые для нужд практики задачи требуется решать в реальном времени или для их решения требуется очень большой объем вычислений. Таким трудоемким алгоритмом является, например, алгоритм трассировки лучей - один из методов построения реалистичных сцен с учетом теней, эффектов отражения, преломленияи т.д..

Целью данной работы является изучение организации параллельных вычислений на базе алгоритма трассировки лучей.

В рамках выполнения работы необходимо решить следующие задачи:

1) изучить основы параллельных вычислений;

2) изучить алгоритм трассировки лучей;

3) разработать параллельную версию алгоритма трассировки лучей;

4) реализовать последовательную и параллельную трассировку лучей;

5) провести сравнительный анализ времени работы реализаций.

1 Аналитическая часть

В данном разделе будет приведена теория, необходимая для разработки и реализации последовательного и параллельного алгоритма трассировки лучей.

1.1 Алгоритм трассировки лучей

Алгоритм трассировки лучей работает в пространстве изображений и имеет 2 подхода: прямой и обратный.

Основная идея алгоритма прямой трассировки лучей состоит в том, что наблюдатель видит объекты благодаря световым лучам, которые испускает некоторый источник и которые падают на объект, отражаются, преломляются или проходят через него и в результате достигают зрителя. Если проследить за лучами, то становится понятно, что среди них лишь малая часть дойдет до наблюдателя, что показано на рисунке 1.1 слева, а значит большая часть вычислений произведена напрасно.

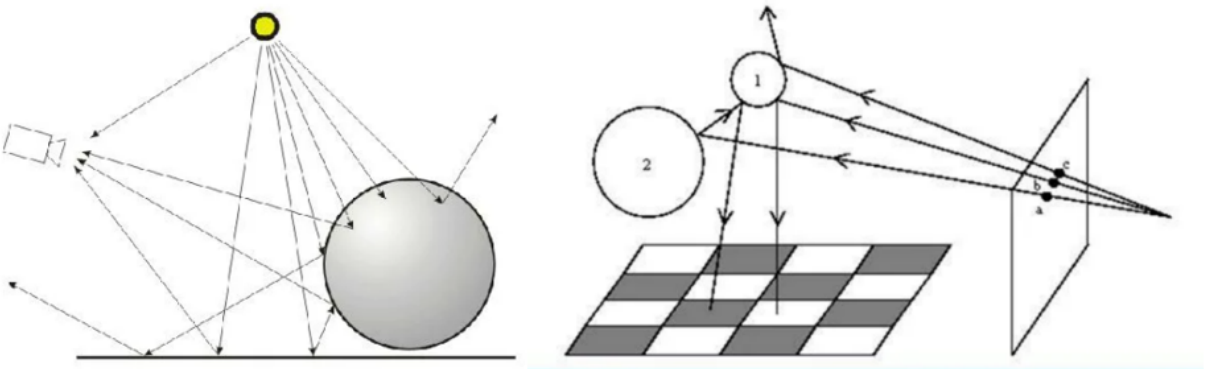

Рисунок 1.1 – Прямая и обратная трассировки лучей

Заменой данному алгоритму служит метод обратной трассировки лучей, который отслеживает лучи в обратном направлении. В ходе работы испускаются лучи от наблюдателя, как показано на рисунке 1.1 справа, и ищутся пересечения луча и всех объектов сцены.

Каждый раз, когда луч пересекает некоторую поверхность, из точки пересечения испускаются новые лучи - отраженный и преломленный. Пути

этих лучей отслеживаются по всей модели, и если лучи пересекают другие поверхности, то снова испускаются лучи.

В каждой точке, где луч пересекает поверхность, рисуется луч тени из точки пересечения к каждому источнику света. Если этот луч пересекает другую поверхность перед тем, как достигнуть источника света, то на ту поверхность, с которой был послан луч, падает тень с поверхности, блокирующей свет [2].

1.2 Параллелизация алгоритма трассировки лучей

Поскольку алгоритм обратной трассировки лучей обрабатывает каждый пиксель экрана независимо, можно использовать параллельные вычисления для уменьшения времени его работы, разбив экран на некоторые части. Наиболее часто используется горизонтальное или вертикальное разбиение (в данной работе используется второе) [3].

1.3 Вывод из аналитической части

В данном разделе были рассмотрены идеи и материалы, необходимые для разработки и реализации последовательного и параллельного алгоритмов трассировки лучей.

2 Конструкторская часть

В данном разделе будет приведеная схема последовательного алгоритма обратной трассировки лучей, а также схемы главного и рабочего потока для параллельной реализации этого алгоритма.

2.1 Схема последовательного алгоритма трассировки лучей

На рисунке 2.1 приведена схема последовательного алгоритма трассировки лучей.

Рисунок 2.1 – Схема последовательного алгоритма трассировки лучей

2.2 Схема параллельного алгоритма трассировки лучей

На рисунке 2.2 приведена схема главного потока для параллельной реализации алгоритма трассировки лучей, который разбивает экран на вертикальные участки, запускает и контролирует рабочие потоки.

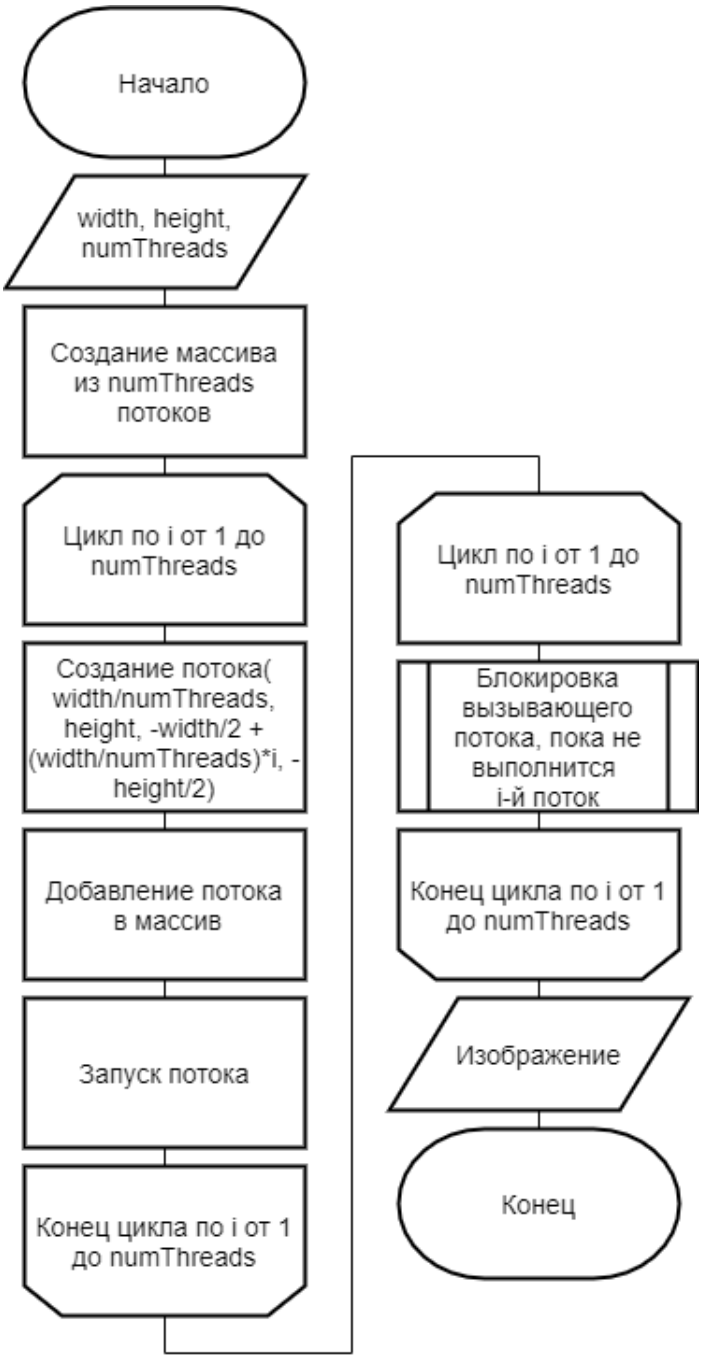

Рисунок 2.2 – Схема главного потока параллельного алгоритма трассировки лучей

На рисунке 2.3 приведена схема рабочего потока для параллельной реализации алгоритма трассировки лучей.

Рисунок 2.3 – Схема рабочего потока параллельного алгоритма трассировки лучей

2.3 Вывод из конструкторской части

Были разработаны схемы базового и параллельного алгоритмов обратной трассировки лучей.

3 Технологическая часть

В данном разделе производится выбор средств реализации, а также приводятся листинги реализованных алгоритмов и результаты тестирования программы.

3.1 Выбор средств реализации

Основное требование к языку программирования в данной лабораторной работе - наличие в нем нативных потоков. Язык C# обладает этим свойством и уже использовался мною ранее, поэтому и был выбран [4].

Для замеров времени используется предоставляемый класс Stopwatch, а для организации распараллеливания - пространство имен System.Threading, которое содержит в себе классы, поддерживающие многопоточное программирование [4], и самостоятельно реализованный вспомогательный класс Limits.

В качестве среды разработки выбрана “Visual Studio 2019” так как она предоставляет автоматическую проверку кода и его автоматический рефакторинг, что делает возможным быстрое выявление и исправление ошибок.

3.2 Реализация алгоритмов

В листингах 3.1 - 3.3 представлены реализации рассматриваемых алгоритмов.

Листинг 3.1 – Реализация последовательного алгоритма трассировки лучей

```
1 public Bitmap render_simple()
2 {
3   Camera camera = scene.camera;
4   Vec3 view_vector = null;
5   Vec3 color = null;
6   for (int x = -scene.canvasWidth / 2; x < scene.canvasWidth / 2;
      x++)
```

```
7   {
8     for (int y = -scene.canvasHeight / 2; y < scene.canvasHeight /
          2; y++)
9     {
10      view_vector = ProjectPixel(x, y) * camera.rotation_mtrx;
11      color = TraceRay(camera.position, view_vector,
            projection_plane_d, Double.PositiveInfinity,
            recursion_depth, x, y);
12      SetPixel(x, y, CountColor(color));
13
14    }
15  }
16
17  return this.bmp;
18 }
```

Листинг 3.2 – Реализация главного потока параллельного алгоритма трассировки лучей

```
1 public Bitmap render_parallelized()
2   {
3     Thread[] threads = new Thread[numThreads];
4     for (int i = 0; i < numThreads; i++)
5     {
6       Limits p = new Limits(scene.canvasWidth / numThreads, scene.
            canvasHeight, -scene.canvasWidth / 2 + scene.canvasWidth
            / numThreads * i, -scene.canvasHeight / 2);
7       threads[i] = new Thread(render_par_working);
8       threads[i].Start(p);
9     }
10    foreach (Thread thread in threads)
11    {
12      thread.Join();
13    }
14
15    return this.bmp;
16  }
```

Листинг 3.3 – Реализация рабочего потока параллельного алгоритма трассировки лучей

```
1
2   public void render_par_working(object obj)
```

```
3   {
4     Limits p = (Limits)obj;
5     Camera camera = scene.camera;
6     Vec3 view_vector = null;
7     Vec3 color = null;
8     for (int x = p.start_x; x < p.start_x + p.width; x++)
9     {
10      for (int y = p.start_y; y < p.start_y + p.height; y++)
11      {
12        view_vector = ProjectPixel(x, y) * camera.rotation_mtrx;
13        color = TraceRay(camera.position, view_vector,
              projection_plane_d, Double.PositiveInfinity,
              recursion_depth, x, y);
14        SetPixel(x, y, CountColor(color));
15
16      }
17    }
```

3.3 Тестирование

На рисунке 3.1 приведен результат тестирования программы по методу черного ящика: проведен запуск программного обеспечения для рендера сцены, состоящей из поверхности земли, повехности неба (стена на некотором расстоянии с текстурой) и флюгера (сложносоставной модели), добавлены два источника света – один точечный и один направленный. Программа успешно выполнила рендер сцены на восьми потоках, изображение совпало с ожидаемым – с тем, которое было получено при запуске последовательной версии алгоритма. На рис. 3.1 представлено полученное изображение.

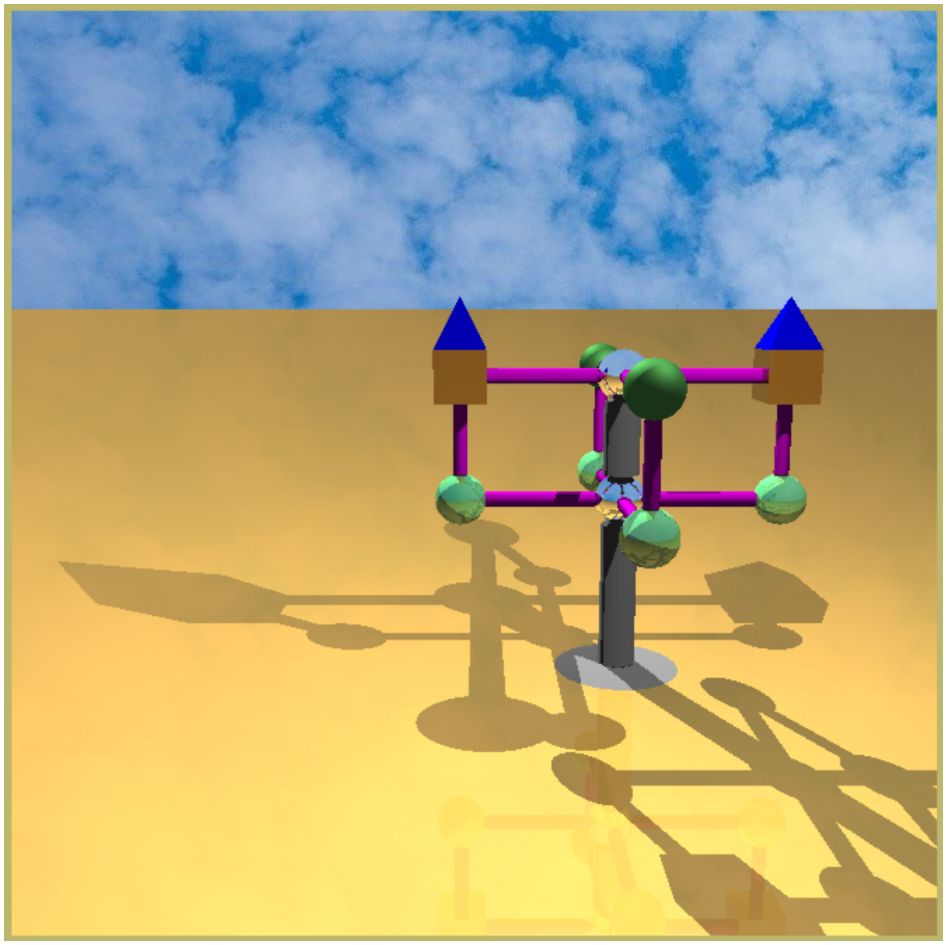

Рисунок 3.1 – Результат тестирования программы

3.4 Вывод из технологической части

Был производен выбор средств реализации, реализованы и протестированы последовательный и параллельный алгоритмы обратной трассировки лучей.

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Windows 10;
- оперативная память: 16 Гб;
- процессор: Intel® Core™ i5-8259U;
- количество ядер: 4;
- количество логических процессоров: 8.

Во время тестирования ноутбук был включен в сеть питания и нагружен только встроенными приложениями окружения и системой тестирования.

4.2 Сравнение времени выполнения реализаций алгоритмов

Сравнивалось время работы (обычное, по таймеру) последовательной и параллельной реализаций алгоритма обратной трассировки лучей, причем во втором случае сравнивалось также время работы реализации в зависимости от количества потоков (1, 2, 4, ..., 4*количесттво логических ядер=32).

Перечисленные реализации сравнивались по времени обработки сцены в зависимости от количества объектов (прямоугольных параллелепипедов) в ней: от 5 до 35 с шагом 5.

Так как некоторые реализации выполняются достаточно быстро, а замеры времени имеют некоторую погрешность, они для каждой реализации

и каждого количества элементов на сцене выполнялись 10 раз, а затем вычислялось среднее время работы.

На рисунке 4.1 приведены результаты сравнения времени работы всех реализаций на всех данных (в легенде количество потоков указано как n_threads).

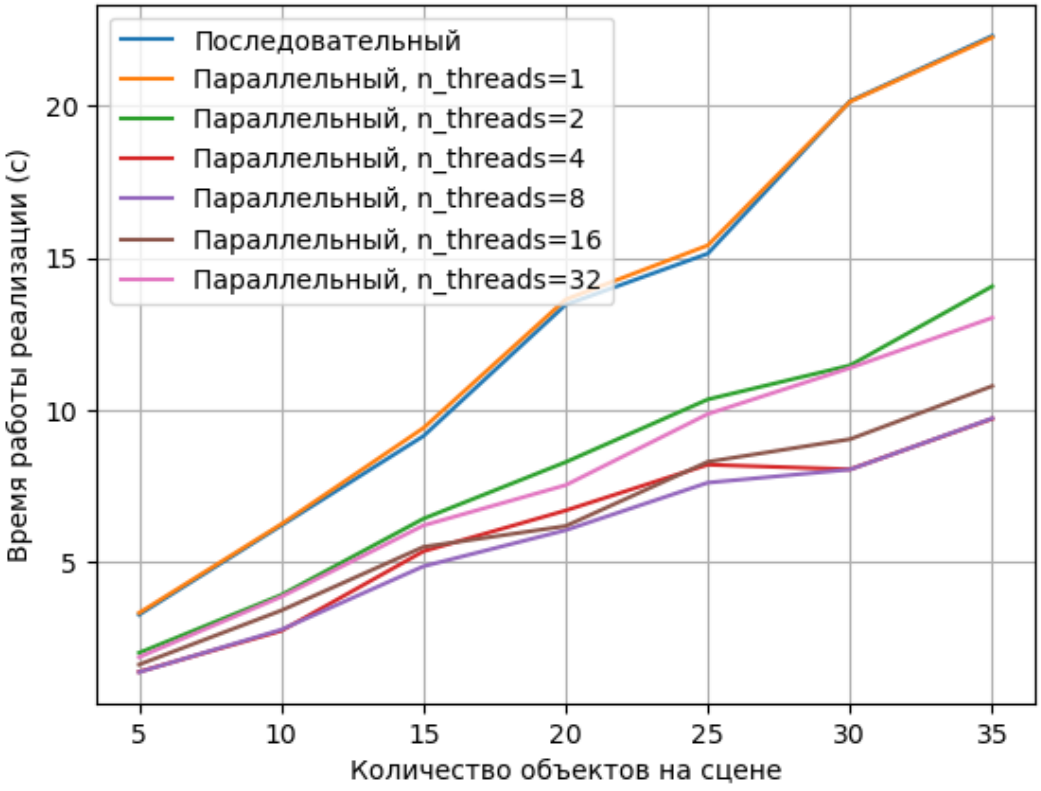

Рисунок 4.1 – Сравнение времени работы реализаций в зависимости от количества элементов на сцене

Последовательная реализация и параллельная реализация с одним потоком, как и ожидалось, работают примерно одинаковое количество времени, хотя при этом вторая немного дольше в связи с накладными расходами на создание потока. Эти две реализации затратили наибольшее количество времени из всех сравниваемых.

Далее с ростом числа потоков время работы соответствующей параллельной реализации уменьшается, так как независимые вычисления производятся одновременно на разных ядрах. Это происходит вплоть до момента, когда используются 8 потоков, то есть их количество равно количеству логических ядер в компьютере.

При дальнейшем увеличении числа потоков время работы параллельной реализации больше, чем в описанной выше наилучшей точке, так как количество потоков становится больше количества логических ядер в компьютере, и, соответственно, некоторые из них вынуждены ожидать освобождения занятого другим потоком процессора, который смог бы провести необходимые вычисления. В результате теряется смысл в выделении этих потоков, так как одновременной обработки каждого из них не происходит, а дополнительное время на их организацию затрачивается.

4.3 Вывод из исследовательской часть

Таким образом, далеко не всегда двукратное увеличение числа потоков улучшает результат по времени. Это происходит, пока количество потоков меньше или равно количеству логических ядер в ЭВМ. Поэтому рекомендуемым числом потоков можно назвать число, равное количеству логических процессоров.

Заключение

В результате выполнения лабораторной работы была достигнута поставленная цель: были изучены основы организации параллельных вычислений на базе алгоритма трассировки лучей.

В ходе выполнения лабораторной работы были выполнены следующие задачи:

1) изучены основы параллельных вычислений;
2) изучен алгоритм трассировки лучей;
3) разработан параллельный алгоритм трассировки лучей;
4) реализованы последовательная и параллельная трассировки лучей;
5) проведен сравнительный анализ времени работы реализаций.

Литература

[1] Параллельные вычисления [Электронный ресурс]. Режим доступа: https://ru.bmstu.wiki/\T2A\CYRP\T2A\cyra\T2A\cyrr\T2A\cyra\T2A\cyrl\T2A\cyrl\T2A\cyre\T2A\cyrl\T2A\cyrsftsn\T2A\cyrn\T2A\cyrery\T2A\cyre_\T2A\cyrv\T2A\cyrery\T2A\cyrch\T2A\cyri\T2A\cyrs\T2A\cyrl\T2A\cyre\T2A\cyrn\T2A\cyri\T2A\cyrya (дата обращения: 05.10.2021).

[2] Роджерс Д. Математические основы машинной графики. М.: Мир, 1989. Т. 512.

[3] Параллельная обработка в алгоритме визуализации с трассировкой лучей. [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/parallelnaya-obrabotka-v-algoritmah-vizualizatsii-s-trassirovkoy-lu (дата обращения: 05.10.2021).

[4] Краткий обзор языка C#. [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 06.10.2021).