

Маятник Ньютона

СТУДЕНТ: СУКОЧЕВА А.

РУКОВОДИТЕЛЬ: ВОЛКОВА Л. Л.

Цель работы:

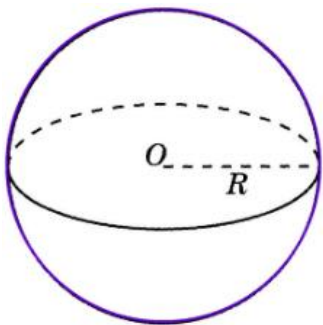
Разработать программу, использующую обратную трассировку лучей для визуализации маятника Ньютона.

Ускорить работу алгоритма путем параллельных вычислений.

Визуализируемые объекты

Сфера

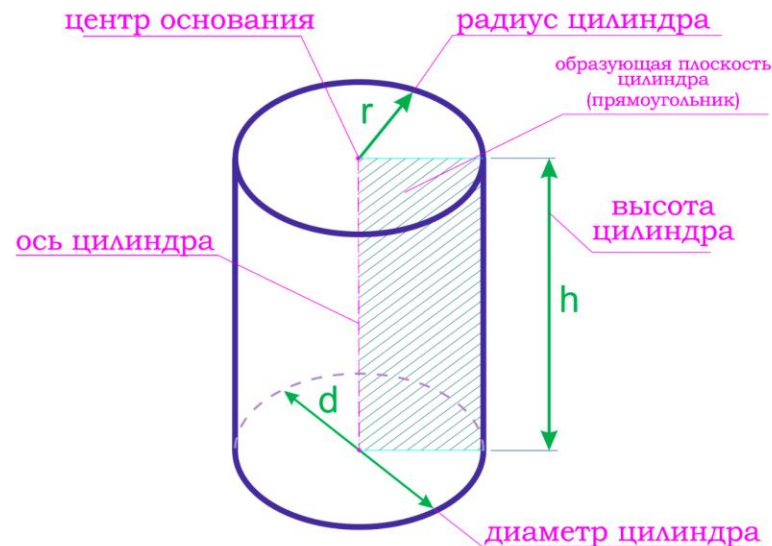
Задается центром, цветом и радиусом. Нормаль в найденной точке сферы является вектор, испущенный из центра сферы к найденной точке.



Цилиндр

Задается центром, цветом, радиусом, минимальным и максимальным значением высоты и осью, образующие цилиндра параллельны ей.

Нормаль в найденной точке цилиндра вычисляется аналогично сфере, за исключением того, что одну компонента центра цилиндра нужно поднять до точки пересечения.



Выбор алгоритма

Алгоритм Робертса

Алгоритм Варнока

Алгоритм Вейлера-Азертона

Алгоритм художника

Алгоритм Z-буфера

Алгоритм прямой
трассировки лучей

Алгоритм обратной
трассировки лучей

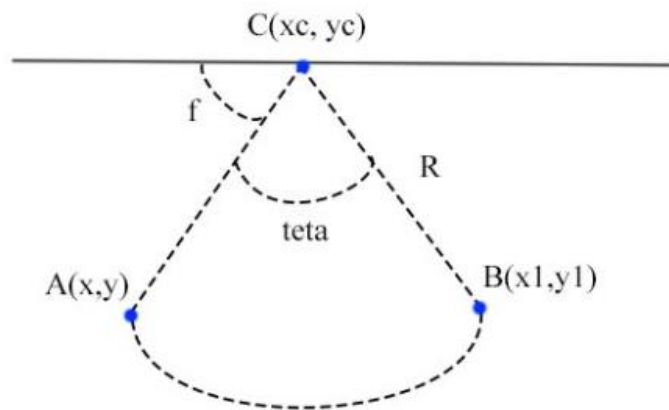


Обратная трассировка лучей

Из некоторой точки пространства, называемой виртуальным глазом или камерой, через каждый пиксель изображения испускается луч и находится точка пересечения с объектом сцены. Далее из найденной точки пересечения испускаются лучи до каждого источника освещения. Если данные лучи пересекают другие объекты сцены, значит точка пересечения находится в тени относительно рассматриваемого источника освещения и освещать ее не нужно. Освещение со всех видимых источников света складываются. Далее, если материал рассматриваемого объекта имеет отражающие или преломляющие свойства, из точки испускается отраженный луч и для него вся процедура трассировки рекурсивно повторяется.

Движение механической системы

Для движения механической системы были произведены преобразования над центром сферы, а именно поворот. Для поворота использовалась некоторая опорная точка и угол.



Повернув одну сферу на некоторый угол все изображение нужно было повторно визуализировать.

Увеличение скорости работы трассировки лучей

Поскольку алгоритм обратной трассировки лучей обрабатывает каждый пиксель экрана независимо, можно распараллелить обработку всего экрана, разбив его на некоторые части, что позволит увеличить скорость работы алгоритма.

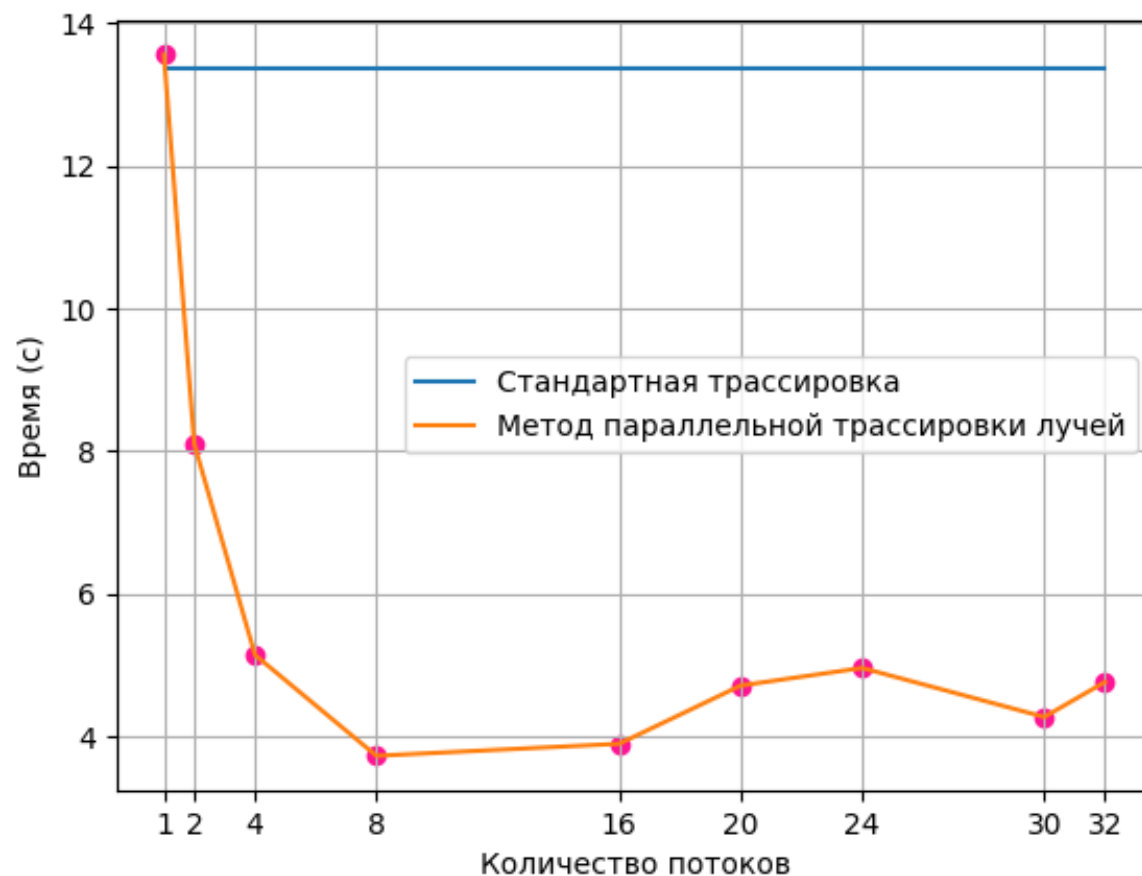



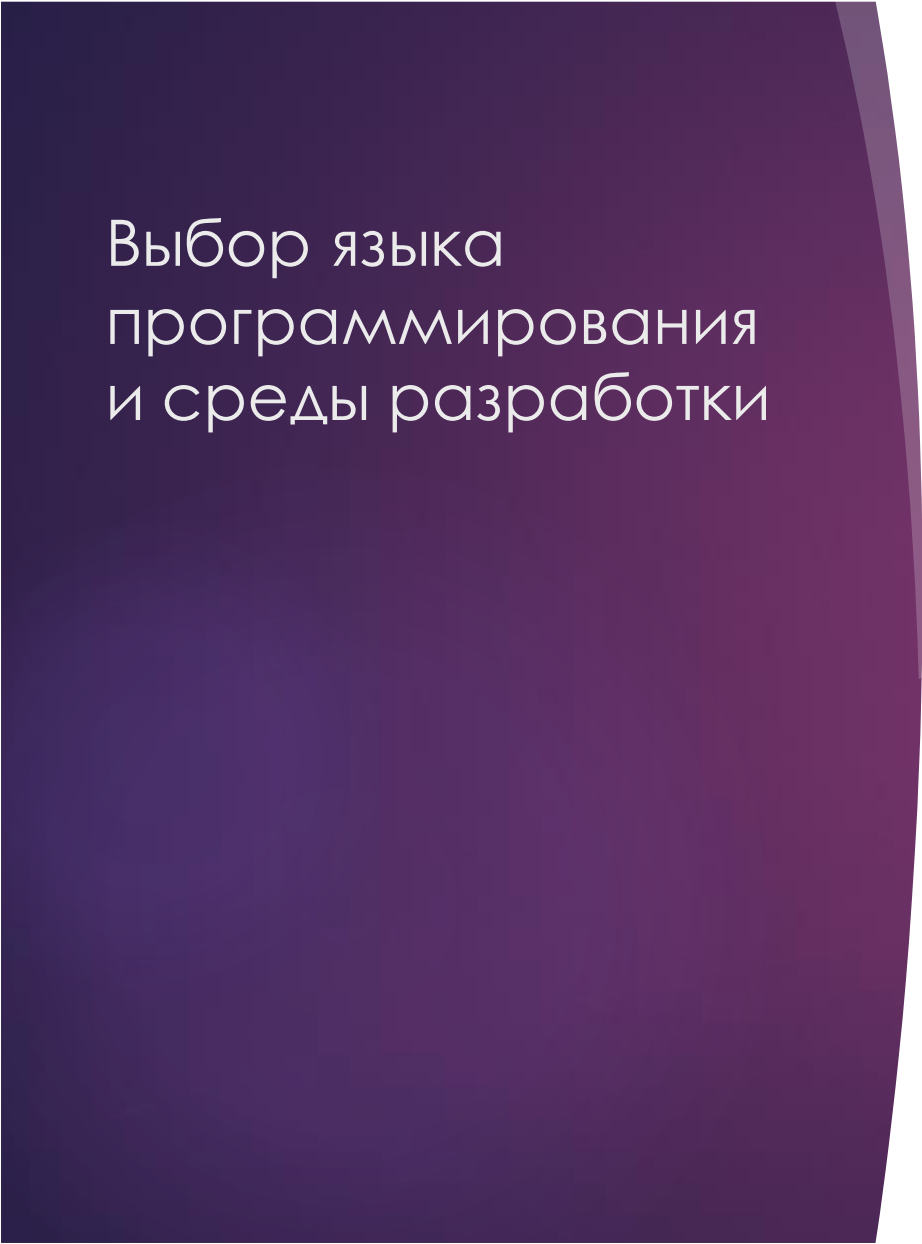
Общий алгоритм визуализации сцены

1. Получение информации о сцене.
2. Отрисовка последовательных изображений с использованием параллельной трассировки лучей и сохранение их в памяти.
3. Запуск механической системы и поочередная визуализация отрендеренных изображений.

Замеры времени

Замеры времени показывают выигрыш параллельной реализации алгоритма трассировки лучей.





Выбор языка
программирования
и среды разработки

В качестве языка
программирования был выбран
язык программирования - C#

В качестве среды разработки был
использован Visual Studio Code

<i>Vector</i>
- _x: double
- _y: double
- _z: double
+ DotProduct(Vector vector)
+ CrossProduct(Vector v)
+ CoDirectional(Vector v)
+ RotatePositive(xCenter, yCenter, angle)
+ RotateNegative(xCenter, yCenter, angle)
+ ReflectRay(Vector n)
+ Normalize()
+ ToString()
+ Sign()
+ ToString()

<i>Colors</i>
- _r: byte
- _g: byte
- _b: byte
+ _CheckLimits(double num)
+ _CheckMax(double num)
+ _CheckMin(double num)
+ RotateNegative(xCenter, yCenter, angle)
+ ToString()

<i>Light</i>
- _position: Vector
- _intensity: double
- _type: LightType
+ ToString()

<i>Shape</i>
- _center: Vector
- _clr: Colors
- _reflective: double
- _type: TypeShape
+ ToString()

<i>Cylinder</i>
- _center: Vector
- _clr: Colors
- _reflective: double
- _type: TypeShape
- _radius: double
- _max: double
- _min: double
- _axis: Axis
+ GetAxesValue()
+ ToString()

<i>Sphere</i>
- _center: Vector
- _clr: Colors
- _reflective: double
- _type: TypeShape
- _radius: double
+ ToString()

<i>MainForm</i>
- _scene: List<Shape>
- _imgBox: PictureBox
- _currImgIndex: int
- _arrayBitmap: List<Bitmap>
- _mode: Mode
- _cameraPosition: Vector
- _lights: List<Light>
+ MainForm(List<Shape> scene)
- SettingsWindows()
- InitializeComponent()
- createArrayImgBox()
- ButtonOnClick(object obj, EventArgs e)
- OnTimedEvent(Object o, ElapsedEventArgs e)
- PutPixel(float i, float j, Color color, Bitmap img)
- TraceRay(Vector origin, Vector direction)
- FuncVertically(object obj)
- DrawScene()
- FuncVertically(object obj)
- ComputeLighting(Vector p, Vector n, Vector v)
- FindNearestObject(Vector origin, Vector direction)
- IsVisible(Vector origin, Vector direction, Shape obj)

Структура и состав классов

Заключение

В данном проекте была разработана программа, которая успешно выполняет визуализацию маятника Ньютона с использованием трассировки лучей. Также скорость работы алгоритма увеличена благодаря использованию параллельных вычислений.



Спасибо за внимание!