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»

КУРСОВАЯ РАБОТА

НА ТЕМУ:

Программа визуализации гравитационного взаимодействия тел в 3-х мерном пространстве

Студент ИУ7-52Б (группа) _____ (подпись, дата) Шахнович Д.С. (И.О. Фамилия)

Руководитель курсовой работы _____ (подпись, дата) Романова Т.Н. (И.О. Фамилия)

2024 г.

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)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(индекс)

_____ _____
(подпись) (И.О. Фамилия)

_____
(дата)

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Компьютерная графика

Студент группы Шахнович Д. С. гр. ИУ7-52Б
(Фамилия, имя, отчество)

Тема курсовой работы Программа визуализации гравитационного взаимодействия тел в 3-х мерном пространстве

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

Источник тематики (кафедра, предприятие, НИР)

Задание Разработать программу для визуализации гравитационного взаимодействия тел, считающихся материальными точками, в 3-х мерном пространстве. Моделирование взаимодействия объектов задается законом тяготения ньютона с учётом аддитивности гравитационных сил. Систему уравнений движения решать численным методом. Предоставить возможность задавать начальные условия сцены объектами и их векторами скоростей, массами, размерами, цветами. Размеры объекты должны влиять только на визуальное представление тела. Предоставить возможность рассмотреть сцену с разных ракурсов поворотом и переносом камеры. Исследовать возможность учёта освещённости и теней при движении объектов.

Оформление курсовой работы:

Расчетно-пояснительная записка (Отчет по КР) на _____ листах формата А4.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ _____
(подпись, дата) (И.О. Фамилия)

Студент _____ _____

Содержание

ВВЕДЕНИЕ **4**

1 Аналитическая часть **5**

1.1 Задача n тел 5

1.2 Формализованная постановка задачи 8

1.3 Виды 3-х мерных объектов 8

1.4 Способы представления твердотельных моделей 9

1.5 Алгоритмы удаления невидимых поверхностей 12

1.6 Модель освещения 15

1.7 Метод учёта теней 15

2 Конструкторская часть **17**

2.1 Функциональные схемы 17

2.2 Модель камеры 18

2.3 Разработка типов и структур данных 19

2.4 Разработка алгоритма решения задачи n тел 20

2.5 Разработка алгоритма отсечения частей вне видимости камеры 21

2.6 Разработка алгоритма z-буфера 22

2.7 Разработка алгоритма теневого буфера 22

3 Технологическая часть **24**

3.1 Средства реализации 24

3.2 Модульное тестирование 24

3.3 Функциональное тестирование 24

4 Исследовательская часть **28**

4.1 Физическая основа исследования 28

4.2 Конфигурации 28

4.3 Результаты 29

4.4 Вывод 30

ЗАКЛЮЧЕНИЕ **31**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **32**

Приложение А **34**

ВВЕДЕНИЕ

В небесной механике – известна задача n тел, Которая заключается в описании относительного движения n материальных объектов, связанных друг с другом законом всемирного тяготения Ньютона. До сих пор общее решение задачи для трёх тел и более не получено. Проблема не имеет решения в виде однозначных аналитических функций в общем случае, как, например, для двух тел [2]. Объекты в задаче n тел представляются материальными точками, то есть точками в трёхмерном пространстве не имеющими размера, но имеющие массу, а их расстояние их взаимодействия существенно превышает их собственные размеры. Визуализация трёхмерного движения позволяет лучше представлять одно из фундаментальных взаимодействие в физике.

Целью данной работы является разработка программного обеспечения для визуализации задачи n тел в 3-х мерном пространстве в виде динамически-генерируемой анимации.

Для достижения этой цели необходимо выполнить следующие задачи:

1) анализ физической задачи n тел и описание метода её решения;
2) анализ существующих методов и алгоритмов для создания динамического изображения трёхмерной сцены;
3) выбор наиболее подходящих алгоритмов для построения трёхмерной сцены;
4) разработка выбранных алгоритмов;
5) выбор средств реализации ПО;
6) реализация ПО и выбранных алгоритмов;
7) исследование погрешности выбранного численного метода решения задачи.

1 Аналитическая часть

В данной части будет проведён анализ поставленной задачи, объектов трёхмерной сцены, а также существующих методов построения 3-х мерных сцен.

1.1 Задача n тел

Постановка задачи

Пусть существует N частиц, представляемых материальными точками с массами $m_1, ..., m_N$, радиус векторами $\vec{r_1}, ..., \vec{r_N}$ и векторами скоростей $\vec{v_1}, ..., \vec{v_N}$ в момент времени $t = 0$. Попарное взаимодействие точек подчинено закону тяготения ньютона 1.1.

$$\vec{F_{jk}} = G\hat{\vec{r_{jk}}}\frac{m_k m_j}{\vec{r_{jk}}^2}, \tag{1.1}$$

где

— $\vec{F_{jk}}$ – Сила тяготения, действующая j-ой частицей на k-ю;

— $G = 6.67430 * 10^{-11} m^3 s^{-2} kg^{-1}$ – гравитационная постоянная, константа;

— $\vec{r_{jk}} = \vec{r_j} - \vec{r_k}$ – вектор из точки k-ой частицы в точку j-ой;

— $\hat{\vec{r_{jk}}}$ – единичный вектор из точки k-ой частицы в точку j-ой, задающий направление силы;

— m_k, m_j – массы k-ой и j-ой частиц соответственно.

Так как на k-ю частицу действую все остальные частицы системы с силами 1.1, а также с учётом того факта, что силы аддитивны, получается сила, с которой все частицы системы действую на k-ю 1.2.

$$\vec{F_k} = Gm_k \sum_{j=1, j\neq k}^{N} \frac{m_j}{\vec{r_{jk}}^2}\hat{\vec{r_{jk}}}. \tag{1.2}$$

Второй закон Ньютона связывает силу, действующую на частицу, с её ускорением, то есть второй производной радиус-вектора 1.3. По второму закону Ньютона можно связать силу действующую на частицу с её ускорением:

$$m\frac{d^2\vec{r}}{dt^2} = \vec{F}, \tag{1.3}$$

где

— m – масса частицы;

— $\vec{r}$ – радиус-вектор частицы;

— $\vec{F}$ – полная сила, действующая на частицу;

— t – интервал времени.

Тогда объединив уравнения 1.2 и 1.3 получается уравнение движения для k-ой части-

цы 1.4.

$$m_k \frac{d^2\vec{r_k}}{dt^2} = Gm_k \sum_{j=1, j\neq k}^{N} \frac{m_j}{\vec{r_{jk}}^2} \hat{\vec{r}_{jk}}. \tag{1.4}$$

или

$$\frac{d^2\vec{r_k}}{dt^2} = G \sum_{j=1, j\neq k}^{N} \frac{m_j}{\vec{r_{jk}}^2} \hat{\vec{r}_{jk}}. \tag{1.5}$$

Дифференциальные уравнения второго порядка вида 1.5 всех частиц N объединяются в систему уравнений 1.6

$$\begin{cases} \frac{d^2\vec{r_1}}{dt^2} = G \sum_{j=1, j\neq 1}^{N} \frac{m_j}{\vec{r_{j1}}^2} \hat{\vec{r}_{j1}} \\ \dots \\ \frac{d^2\vec{r_N}}{dt^2} = G \sum_{j=1, j\neq N}^{N} \frac{m_j}{\vec{r_{jN}}^2} \hat{\vec{r}_{jN}}. \end{cases} \tag{1.6}$$

Задачей N тел является решение системы уравнений 1.6 движения частиц относительно времени t.

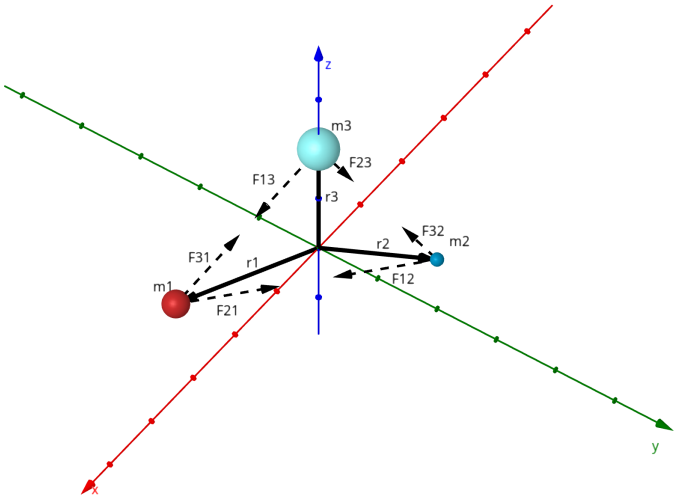


Рисунок 1.1 — «Схема гравитационного взаимодействия при N=3»

Аналитические решения

Для случаев $N = 1$ и $N = 2$ существуют точные аналитические решения, при этом для $N = 1$ по законам инерции частица будет двигаться прямолинейно с постоянной скоростью, а в случае $N = 2$ тела будут двигать по одной из кривых второго порядка: эллипсу, гиперболе или параболе.

Для случая $N = 3$ Г.Э. Брунс и А. Пуанкаре показали, что решение общее решение задачи невозможно выразить через алгебраические или через однозначные трансцендентные функции координат и скоростей тел [1]. Однако в 1912 году Зульдман нашёл точное решение

задачи, но выраженный в виде бесконечных рядов, использовать которые, однако, невозможно в следствие слишком высокой скорости роста кол-ва вычислений [1].

Для случаев $N \geq 3$ на текущий момент времени не было найдено точного аналитического решения ни в какой форме, однако существуют решения для отдельных конфигураций [2].

Метод Эйлера

Поскольку основная задача данной работы визуализация, а не поиск точного решения задачи n тел, то для решения системы уравнений 1.6 будут использованы численные методы интегрирования. Одним из таких является метод эйлера, который будет использоваться в данной работе [3] [4].

Метод эйлера используется для решения задачи Коши вида 1.7.

$$\frac{dy}{dt} = f(t, y(t)), 0 \leq t \leq T, y(0) = y_0 \tag{1.7}$$

Пусть на отрезке интегрирования введена сетку $\omega_\tau = t_i = i\tau, i = 1, 2, \ldots, n$, где $\tau = \frac{T}{n}$ – шаг интегрирования, при этом величина $y_i = y(t_i)$ называется сеточной функцией. Тогда для этого отрезка используется разностная схема Эйлера 1.8.

$$\frac{y_{i+1} - y_i}{\tau} = f(t_i, y_i) \tag{1.8}$$

Все значения y_i, кроме y_0 ищутся итерационно по формуле 1.9, полученной из 1.8.

$$y_{i+1} = y_i + \tau f(t_i, y_i) \tag{1.9}$$

Для применения метода эйлера для решения системы уравнений 1.6 требуется преобразовать её к системе дифференциальных уравнений первого порядка с помощью введения новых переменных - скоростей v_i 1.10.

$$\begin{cases} \frac{d\vec{r_1}}{dt} = \vec{v_1} \\ \ldots \\ \frac{d\vec{r_N}}{dt} = \vec{v_N} \\ \frac{d\vec{v_1}}{dt} = G \sum_{j=2}^{N} \frac{m_j}{\vec{r_{j1}}^2} \hat{\vec{r_{j1}}} \\ \ldots \\ \frac{d\vec{v_N}}{dt} = G \sum_{j=1, j\neq N}^{N} \frac{m_j}{\vec{r_{jN}}^2} \hat{\vec{r_{jN}}} \end{cases} \tag{1.10}$$

Систему 1.10 можно интегрировать методом эйлера используя формулу 1.9. В результате полученная система 1.11 решает задачу n тел для заданного отрезка интегрирования.

$$\begin{cases} \vec{r}_1(i+1) = \vec{r}_1(i) + \tau\vec{v}_1(i) \\ \dots \\ \vec{r}_N(i+1) = \vec{r}_N(i) + \tau\vec{v}_N(i) \\ \vec{v}_1(i+1) = \vec{v}_1(i) + \tau G \sum_{j=2}^{N} \frac{m_j}{\vec{r}^2_{j1}(i)} \hat{\vec{r}}_{j1}(i) \\ \dots \\ \vec{v}_N(i+1) = \vec{v}_N(i) + \tau G \sum_{j=1, j\neq N}^{N} \frac{m_j}{\vec{r}^2_{jN}(i)} \hat{\vec{r}}_{jN}(i) \end{cases}, \tag{1.11}$$

где $\vec{r}_k(i)$ – радиус-вектор k-ой частицы в i-ый шаг сетки, а $\vec{v}_k(i)$ – скорость k-ой частицы в i-ый шаг сетки.

1.2 Формализованная постановка задачи

Входные данные:

— объекты сцены – 3-х мерные тела, передающие относительное расположение материальных точек. Каждый из них представляет собой одно из платоновых тел [5]. Тело обладает визуальными характеристиками: размер и цвет, и физическими характеристиками материальной точки: масса, положение, линейная скорость;

— точечный источник света – точка создающий свет определённого цвета распространяющийся равномерно во все стороны. Задаётся положением и интенсивностью света;

— камера – сущность которая задаёт плоскость проецирования, видимую область сцены. Задаётся положением, направлениями взгляда и вверх, пирамидой видимости, а также расположением и размерами окна.

Выходные данные:

— динамически-генерируемая анимация задачи n тел.

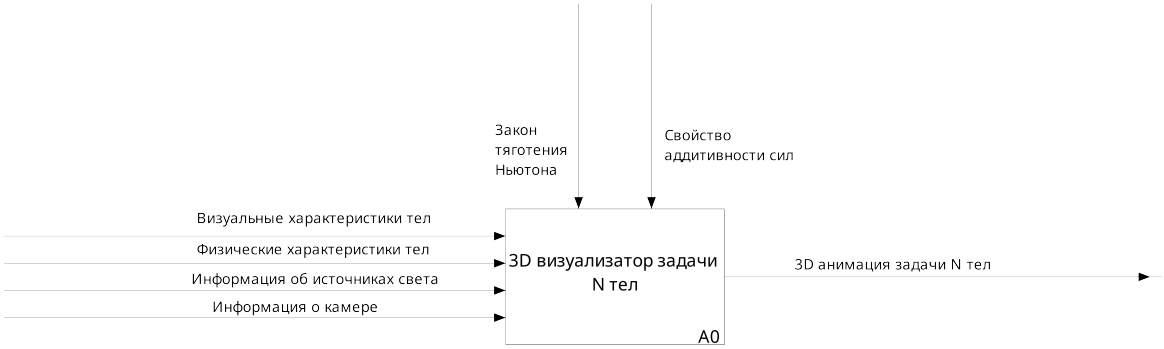


Рисунок 1.2 — «Формализованная постановка задачи в нотации idef0»

1.3 Виды 3-х мерных объектов

В машинной графике существует 3 основные модели, которыми описываются стереометрические тела [10]: каркасная, поверхностная и твердотельная модели.

1) Каркасная модель – простейшая из трёх моделей. Объекты в такой модели задаются с помощью множество точек и рёбер, связывающих эти точки.

2) Поверхностная модель – модель, задаётся информация о поверхности, ограничивающей объект. Сама поверхность может задаваться аналитически или с помощью полигональной аппроксимации.

3) Твердотельная модель – модель, в которой задана поверхность и информация о том, с какой стороны находится материал.

В данной работе не требуется информация о том, с какой стороны расположен материал, однако внутренняя нормаль к поверхности требуется для некоторых алгоритмов, поэтому в данной работе будет использована твердотельная модель.

1.4 Способы представления твердотельных моделей

Существует два подхода к представлению поверхностных моделей в памяти:

— **Аналитический** – задание поверхностей с помощью описания функций и уравнений. Используются для качественного отображения поверхностей вращения и гладких функций;

— **Полигональная сетка** – задание поверхности с помощью набора многоугольников. Если с помощью многоугольников нельзя точно задать поверхность, то с помощью них она аппроксимируются.

В данной работе в качестве объектов выступают платоновы тела, которые являются многогранниками, так что для них лучше подходит представление в виде полигональной сетки.

Представление полигональных сеток

Существуют различные способы хранения полигональных сеток в памяти [9].

Представление точка-точка

Представление «точка-точка» описывает поверхность с помощью списка точек, у каждой из которых хранится список точек, с которыми она соединена. Информация о рёбрах и гранях поверхности не хранятся явно. Представление продемонстрировано на рисунке 1.3.

Список вершин

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,1	v4 v5 v6 v7
v9	.5,.5,0	v0 v1 v2 v3

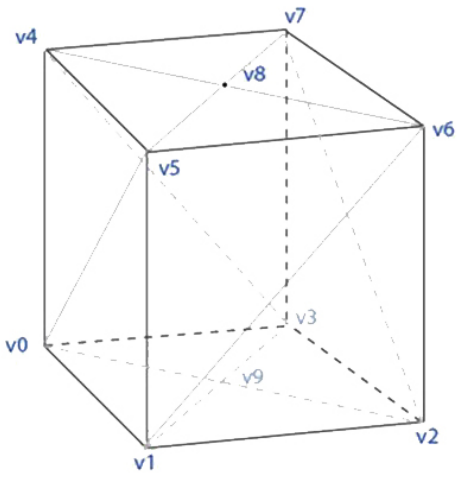


Рисунок 1.3 — «Представление точка-точка»

Представление грань-точка

Представление «грань-точка» описывает поверхность с помощью списка точек и граней. При этом для каждой грани хранятся образующие точки. Представление продемонстрировано на рисунке 1.4.

Список граней

f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

Список вершин

v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f15 f14 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5,.5,0	f8 f9 f10 f11
v9	.5,.5,1	f12 13 14 15

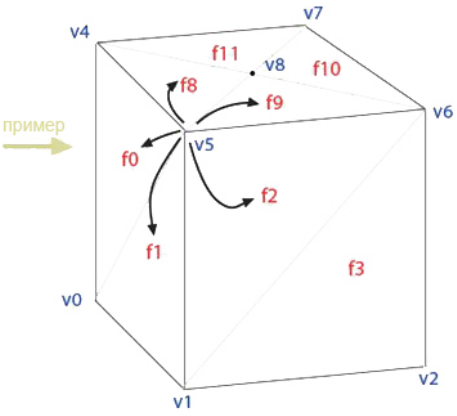


Рисунок 1.4 — «Представление грань-точка»

«Крылатое» представление

При крылатом представлении каждого ребра хранятся две точки, образующие его, две грани, которые он образует и 4 ребра, которые соединены с ним – по 2 с каждой точкой. Для

каждого ребра хранятся ближайшие ребра по часовой и против часовой стрелок. Представление продемонстрировано на рисунке 1.5.

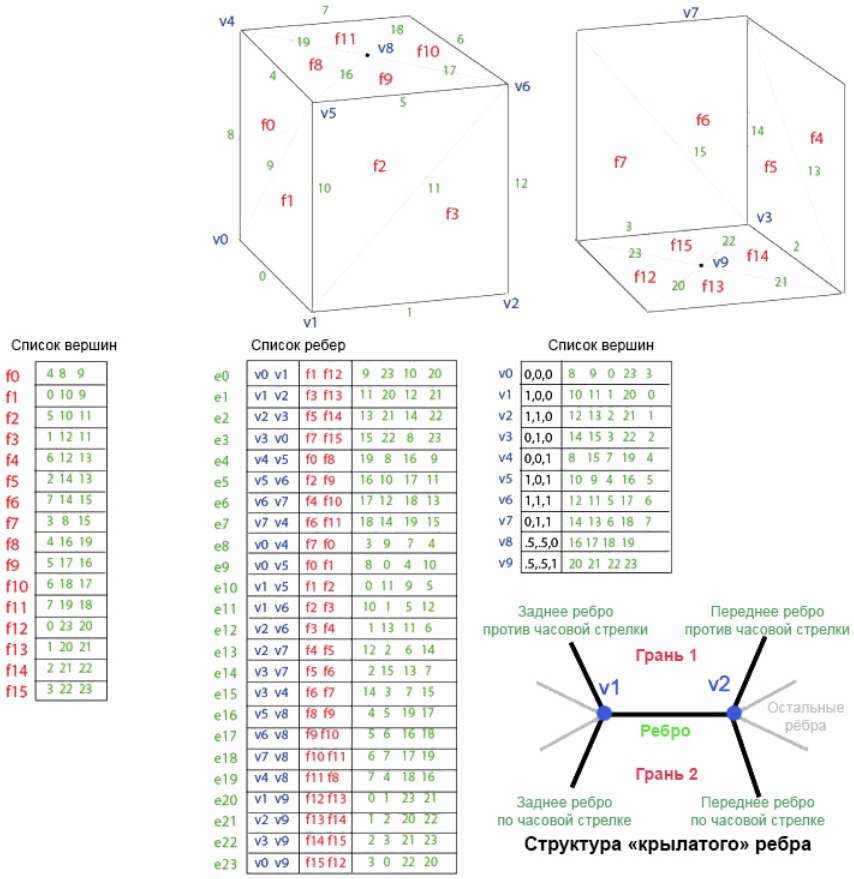


Рисунок 1.5 — «Крылатое представление»

В таблице 1.1 представлено сравнение способов хранения полигональных сеток.

Таблица 1.1 — Сравнение представлений полигональных сеток

Критерий	**«Точка-точка»**	**«Грань-точка»**	**«Крылатое»**
Кол-во указателей для хранения куба [9]	24	**30**	192
Получение всех граней	неявное	**явное**	явное
Получение всех рёбер	неявное	**неявное**	явное

В данной работе было выбрано представление «грань-точка», так как оно позволяет получать список граней без дополнительных операций, а также не требует для хранения дополнительной информации.

1.5 Алгоритмы удаления невидимых поверхностей

Алгоритм удаления невидимых поверхностей является наиболее вычислительно сложной задачей в компьютерной графике [10]. На данном этапе определяются объекты или их части, которые буду отрисованы на экране.

Одна из классификаций алгоритмов – по системе координат, в которой они работают:

— **В пространстве объектов** – имеют дело с физической системой координат, в которой заданы объекты. Точность ограничена точностью вычислений.

— **В пространстве экрана** – имеют дела с системой координат, связанной с экраном. Точность ограничена разрешением экрана.

При выборе алгоритма удаления невидимых поверхностей, важно учесть, что изображение будет строиться динамически в реальном времени и, что объекты будут иметь небольшой размер и отдалены от камеры, так как они созданы для визуализации положения материальной точки, размеры которой малы.

Будут рассмотрены 4 алгоритма удаления невидимых поверхностей [8]:

1) алгоритм Робертса;
2) алгоритм обратной трассировки лучей;
3) алгоритм Варнока;
4) алгоритм использующий z-буфер.

Алгоритмы в объектном пространстве

Алгоритм Робертса

Алгоритм Робертса – первое известное решение задачи удаления невидимых поверхностей [10]. Для работы алгоритма требуются выпуклые тела. Алгоритм состоит из 4-х этапов.

Первый этап – подготовка матрицы тела. Матрица тела – матрица, хранящая информацию об уравнениях, задающих плоскости всех граней. Имеет размерность $4 * n$, где n – число граней тела.

$$V = \begin{pmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{pmatrix}, \tag{1.12}$$

где $a_i x + b_i y + c_i z + d_i = 0$ – уравнение i-ой грани. Матрица должна быть сформирована так, чтобы при умножению на неё точку внутри тела получался вектор положительных чисел.

Второй этап – удаление нелицевых граней. На данном этапе удаляются грани, которые экранируются самими объектами.

Третий этап – удаление рёбер, экранируемых другими телами. Для этого для каждого

оставшегося ребра нужно пустить луч из точки взгляда к точке ребра. Если луч проходит через другие грани, значит эта точка отрезка экранируется.

Четвёртый этап – создание рёбер соединения граней. Для этого запоминаются все точки протыкания в предыдущем пункте и попарно соединяются отрезками. Далее эти отрезки также проверяются на экранирование, как и в 3-ем пункте.

Алгоритм обеспечивает высокую точность вычислений, но сложный вычислительно и налагает условие выпуклости на тела.

Алгоритм обратной трассировки лучей

В данном алгоритме удаление невидимых граней происходит за счёт построения луча из точки обзора через каждый пиксель экрана [7]. Таким образом на экран попадают лишь объекты, пересечение которых с лучом происходит ближе к точке обзора. Трассировку луча можно продолжить отражением и преломления по законам оптики для моделирования зеркального отражения.

Алгоритмы в пространстве экрана

Алгоритм Варнока

Алгоритм Варнока основан на идее, что для обработки областей с малой плотностью информации мозг человека тратит малое количество времени, а с большой плотностью — больше.

Идея алгоритма в том, чтобы разбивать экран на окна и подокна до тех пор, пока его содержимое не будет элементарным для визуализации, или пока не будет достигнут предел разрешения. На рисунке 1.6 представлен пример разбиения экрана алгоритмом Варнока.

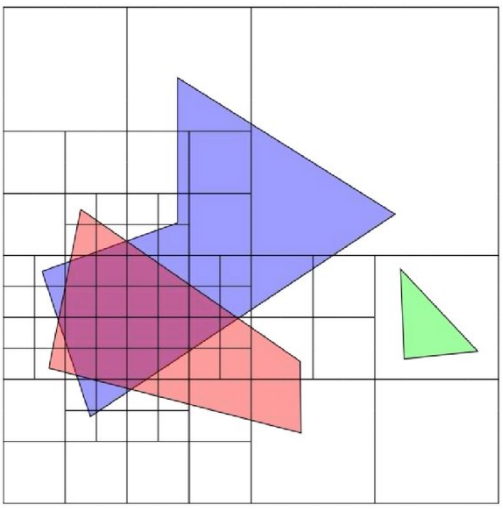


Рисунок 1.6 — «Пример работы алгоритма Варнока»

Алгоритм использующий z-буфер

Алгоритм z-буфера — один из широко используемых алгоритмов отрисовки удаления невидимых поверхностей.

Основой алгоритма является два буфера, которые имеют размеры экрана в пикселях:

— **Буфер кадра** — используется для хранения значения интенсивности(цвета) каждого пикселя, который будет отрисован на экране. По сути хранит в себе то, как экран будет выглядеть в следующем кадре.

— **Буфер глубины(z-буфер)** — Используется для хранения глубины каждого пикселя, описанного в буфере кадра. Нужен для определения видимых граней.

Предварительно алгоритм требует проецирование объекта в пространство экрана. Алгоритм растеризует объект и сохраняет его в буфере кадра и глубины, при этом, если глубина очередного пикселя больше, чем уже сохранённого в буфере, то этот пиксель пропускается.

Но при этом алгоритм требует большего объёма памяти, для хранения буферов [10], чем другие алгоритмы, но он обрабатывает только те пиксели, в которых есть объекты.

Выбор алгоритма удаления невидимых поверхностей

В таблице 1.2 представлено сравнение методов удаления невидимых поверхностей.

Таблица 1.2 — Сравнение алгоритмов удаления невидимых поверхностей

Критерий	**Трассировка лучей**	**Робертс**	**Варнок**	**Z-буфер**
Обрабатываются ли пустые пиксели?	Да	Нет	Да	**Нет**
Дополнительная память	Нет	Нет	Да, 1 буфер размером с количество пикселей	**Да, 2 буфера размером с количество пикселей**
Требует информацию о рёбрах тела	Нет	Да	Нет	**Нет**
Вычислительная сложность	$O(n_{\text{объекты}} * n_{\text{пиксели}} * n_{\text{лучи}})$	$O(n^2_{\text{объекты}})$	$O(n_{\text{объекты}} * n_{\text{подокна}})$	$O(n_{\textbf{объекты}} * n_{\textbf{пиксели проекции}})$

Для данной работы оптимальным является алгоритм z-буфера, так как он является достаточно производительным для динамических сцен, не обрабатывает пустые пиксели, которых будет больше непустых, так как объекты небольшие, а дополнительные затраты памяти не являются критическими.

1.6 Модель освещения

Модели освещения в компьютерной графике делятся на 2 больших класса: глобальные и локальные.

Глобальные модели освещения учитывают отражённый и преломлённый свет от других объектов, однако расчёт такой модели требует трассировку лучей, поэтому не могут быть реализованы в этой работе.

Локальные модели освещения учитывают только прямой свет от источника до рассматриваемого объекта. В данной работе будет использована модель Ламберта [6], для расчёта освещения.

Отражённый свет зависит от характеристик и положения источника, относительно тела. При этом свет может быть отражён диффузно и зеркально. Зеркальное отражение не учитывается в модели ламберта. Диффузное отражённый свет испускается во все стороны равномерно. Интенсивность отражаемого диффузного света зависит закону Ламберта [6]:

$$I = I_l k_d cos(\phi),\ 0 \leq \phi \leq \frac{\pi}{2}, \tag{1.13}$$

где

— I – интенсивность отражённого света;

— I_s – интенсивность источника света

— k_d – коэффициент диффузного отражения тела

— ϕ – угол между нормалью к поверхности в точке падения и направлением света.

Свет, отражённый от других объектов в локальных моделях заменяется на рассеянный свет:

$$I = I_a k_a + I_l k_d cos(\phi),\ 0 \leq \phi \leq \frac{\pi}{2}, \tag{1.14}$$

где I_a – интенсивность рассеянного света, а k_a – коэффициент диффузного отражения рассеянного света.

1.7 Метод учёта теней

Тени делят на проекционные и собственные. Собственные тени образуются телами на себя, проекционные – образованные другими телами.

Для учёта проекционных теней, будет использован метод теневых карт [7]. Данный метод является модификацией z-буфера.

1) Для каждого источника света:
    1.1) преобразовать объекты в пространство источника света;
    1.2) построить буфер глубины методом z-буфера;
2) Для каждой точки из буфера кадра:
    2.1) преобразовать точку в пространство источника света;

2.2) если глубина точки больше глубины соответствующего значения буфера глубины:

2.3) точка в тени;

2.4) иначе:

2.5) точка освещена.

Так как в работе будут использоваться точечные источники света, то для каждого источника необходимо будет построить 6 теневых карт, которые образуют куб, описывающий источник.

Полученные тени будут выглядеть ступенчато (рисунок 1.7), так как их точность ограничена разрешением теневого буфера. Для точного учёта собственных теней будет использован метод, основанный на первых 2-х этапах алгоритма Робертса: если грань относительно источника света является нелицевой, то она находится в тени относительно этого источника света [10].

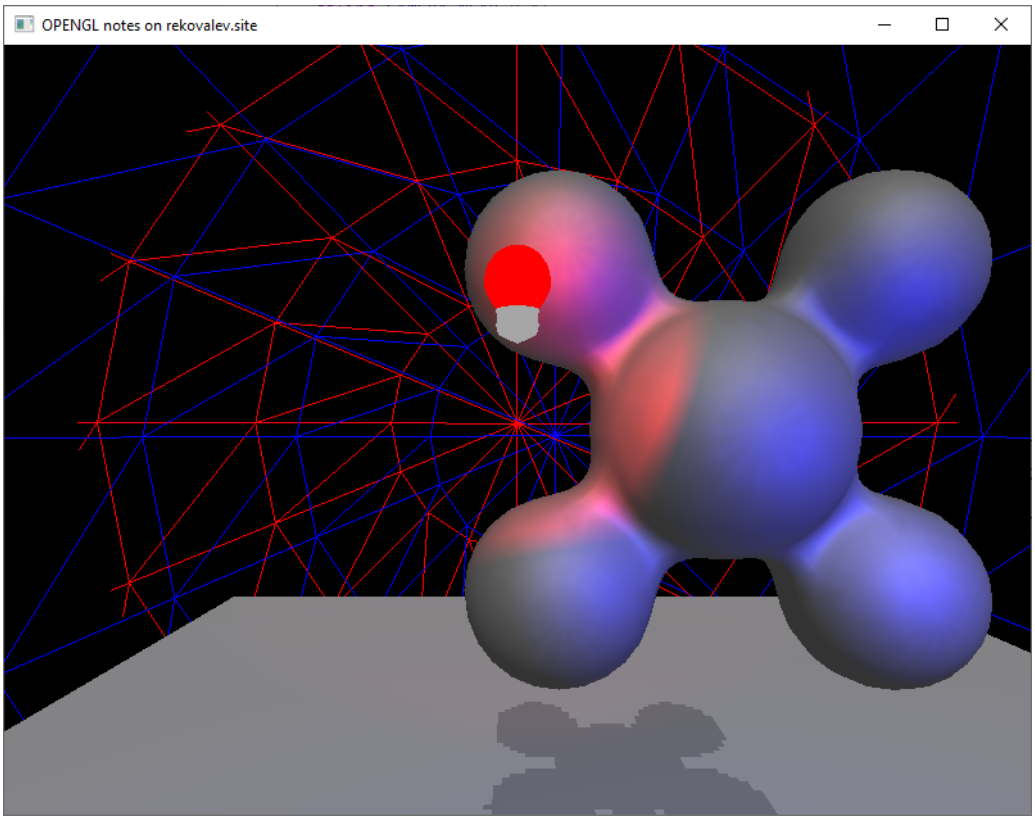


Рисунок 1.7 — «Пример ступенчатости теней, созданных теневыми картами [13]»

Вывод

В результате аналитической части была описана математическая модель рассматриваемой задачи, а также описан метод её решения. Были рассмотрены теоретические сведения о способах представления объектов программно и методы преобразования этих объектов для отображения на экране. Также были выбраны методы и алгоритмы, которые будут использоваться в данной работе.

2 Конструкторская часть

В данной части будут разработаны функциональные схемы разрабатываемого ПО, а также разработаны структуры и алгоритмы, которые будут использованы в ПО.

2.1 Функциональные схемы

На рисунках 2.1 – 2.3 представлено формальное описании разрабатываемого ПО в виде idef0-диаграммы.

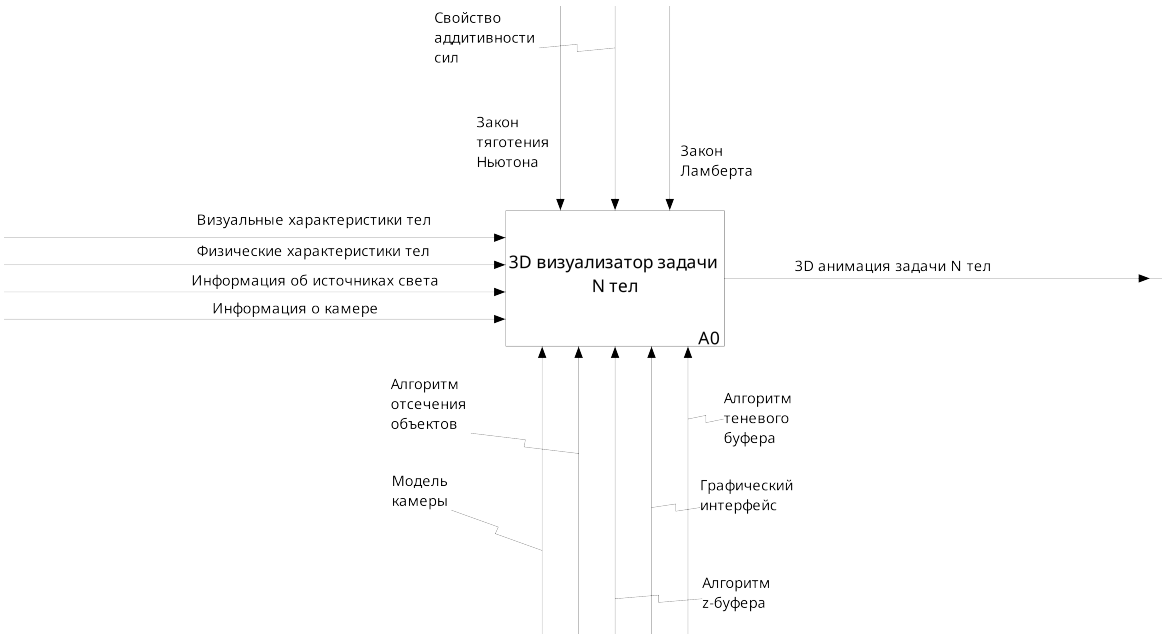


Рисунок 2.1 — «Контекстная диаграмма верхнего уровня в нотации idef0»

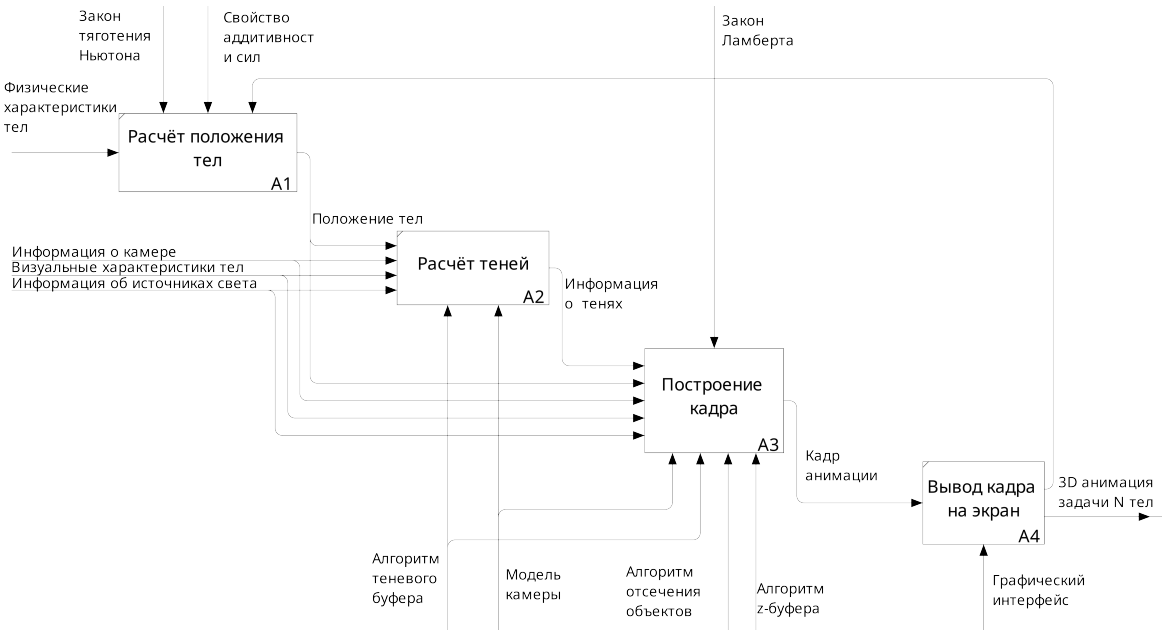


Рисунок 2.2 — «Основной цикл ПО в нотации idef0»

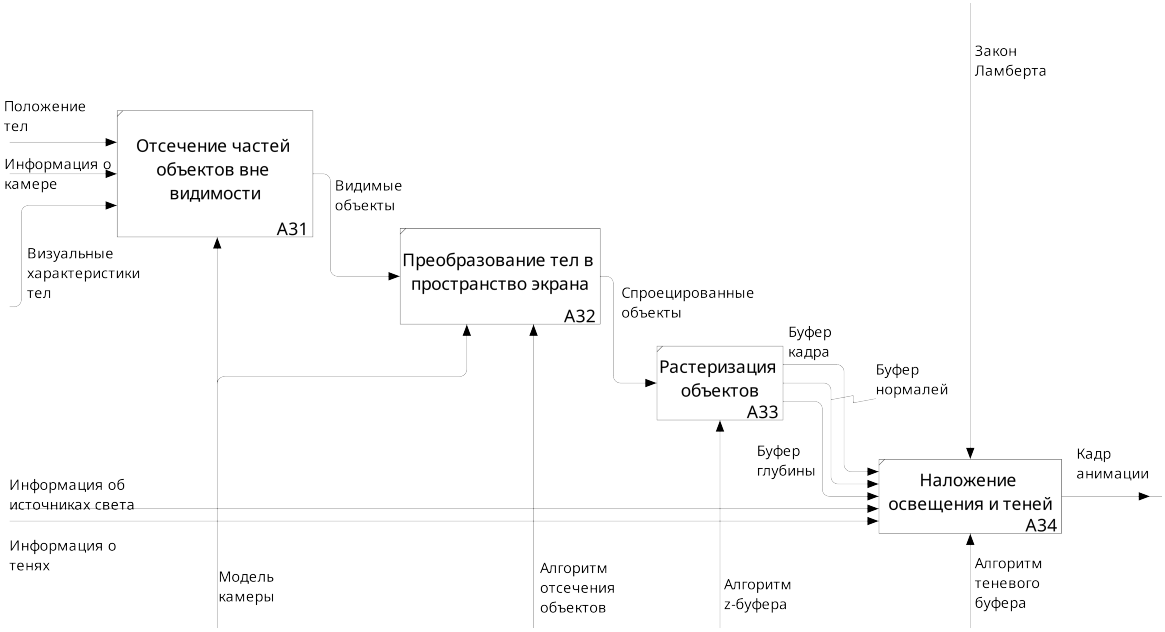


Рисунок 2.3 — «Построение кадра в нотации idef0»

2.2 Модель камеры

Камера задаётся своим положением, направлением взгляда и направлением вверх и пирамидой видимости. Вектор в сторону получается векторным произведением вектора взгляда на вектор вверх. В пространстве камеры направление взгляда совпадает с положительным направлением оси z, а направление вверх – с положительным направлением оси y.

Пирамида видимости – усечённая 4-х гранная пирамида, уходящая в бесконечность, основание которой параллельно XoY. Основание пирамиды задаётся 3-мя параметрами:

— d – расстояние от положения камеры до основания;

— rx – полуширина основания;

— ry – полувысота основания.

Пирамида видимости в пространстве камеры продемонстрирована на рисунке 2.4.

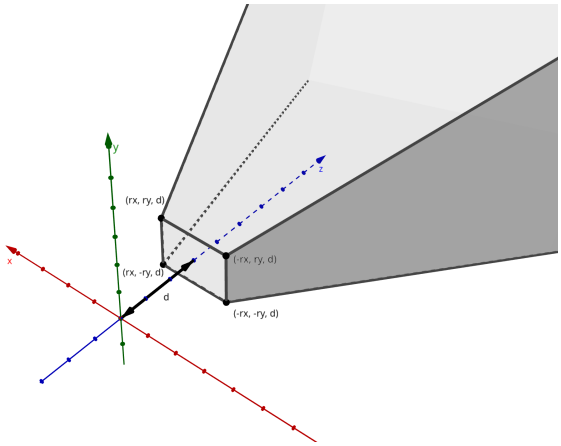


Рисунок 2.4 — «Пирамида видимости камеры»

Перспективная проекция по пирамиде видимости рассчитывается по формуле 2.1 [11].

$$\begin{pmatrix} x_p \\ y_p \\ z_p \\ w_p \end{pmatrix} = \begin{pmatrix} \frac{d}{rx} & 0 & 0 & 0 \\ 0 & \frac{d}{ry} & 0 & 0 \\ 0 & 0 & 1 & 2d \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}. \tag{2.1}$$

Преобразование к пространству камеры выполняется по формуле 2.2 [12].

$$\begin{pmatrix} x_v \\ y_v \\ z_v \\ w_v \end{pmatrix} = \begin{pmatrix} lx & ux & fx & 0 \\ ly & uy & fy & 0 \\ lz & uz & fz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}, \tag{2.2}$$

где

— $\vec{u} = (ux, uy, uz)^T$ – вектор взгляда;
— $\vec{f} = (fx, fy, fz)^T$ – вектор вверх;
— $\vec{l} = (lx, ly, lz)^T = [\vec{u}, \vec{f}]$ – вектор влево;

2.3 Разработка типов и структур данных

В работе и для разработки алгоритмов необходимы следующие типы и структуры данных.

1) Математические абстракции
   1.1) вектор – задаётся 3-мя вещественными числами – координатами вектора;
   1.2) плоскость:
      — $normal$ – вектор внутренней (если задано направление) нормали плоскости;
      — d – свободный член в уравнение плоскости;
2) Физическое тело:
   — pos – положение тела, заданное вектором;
   — vel – вектор скорости тела;
   — $mass$ – масса;
3) Полигон:
   — $v1$, $v2$, $v3$ – вектора, задающие точки образующие полигон;
   — $normal$ – внутренняя (если задано направление) нормаль плоскости, проходящей через полигон;
4) Полигональный объект:
   — $vertices$ – массив векторов, задающих точки образующие объект;
   — $polygons$ – массив полигонов, из которых состоит объект;
   — pos – вектор, задающий положение объекта;
5) Точечный источник света:
   — pos – вектор, задающий положение источника света;

— $intensity$ – интенсивность или цвет света, задаваемый 3-мя беззнаковыми числами: R, G и B, задающие соответственно красный, зелёный и синий цвета;

6) Камера:

— pos – вектор, задающий положение камеры;

— $forward$ – вектор, задающий направление взгляда;

— up – вектор, задающий направление вверх;

— px – полуширина окна камеры;

— py – полувысота окна камеры;

— d – расстояние от камеры, до окна камеры.

2.4 Разработка алгоритма решения задачи n тел

Алгоритм расчёта положения тел

Входные данные:

— массив тел arr;

— промежуток времени dt, заданный вещественным числом.

Выходные данные:

— массив обновлённых тел out_arr.

Алгоритм

1) создать пустой массив тел out_arr размера массива arr;

2) для каждого тела $body$ из массива arr:

2.1) $dp =$ произведение вектора vel на скаляр dt;

2.2) $npos = pos + dp$

2.3) $force =$ нулевой вектор;

2.4) для каждого тела $infl$ из массива arr:

2.4.1) если $infl$ не равен $body$:

2.4.2) прибавить к $force$ силу действующую на $body$ телом $infl$;

2.5) $nvel = vel+$ скалярное произведение $force$ на $\frac{dt}{mass}$;

2.6) $nbody$ – новое тело с параметрами ($npos$, $nvel$, $mass$);

2.7) добавить $nbody$ в out_arr;

3) возврат out_arr.

Алгоритм расчёта силы, действующей одним телом на другое Входные данные:

— $body$ – тело, на которое действует сила;

— $infl$ – тело, которое действует на $body$;

— каждое тело задано:

- pos – точка положения тела;
- vel – вектор скорости тела;
- $mass$ – масса тела;

Выходные данные:

— $force$ – вектор силы, действующий на $body$ со стороны $infl$.

Алгоритм

1) вектор $diff$ – разность векторов $infl.pos$ и $body.pos$;
2) $distance$ – квадрат расстояние между $body$ и $infl$ – $diff^2$;
3) $force$ – нормализованный вектор $diff$;
4) умножить $force$ на скаляр $G = 6.67430e^{-11} * infl.mass * body.mass/distance$;
5) возврат $force$.

2.5 Разработка алгоритма отсечения частей вне видимости камеры

Алгоритм отсечения полигона по плоскости

Входные данные:

— $plane$ – плоскость;

— $poly$ – полигон.

Выходные данные:

— $polys$ – массив отсечённых полигонов;

— $seen$ – флаг, что полигон виден целиком или частично относительно плоскости;

Алгоритм

1) создать пустой массив полигонов $polys$ размером массива 2;
2) создать пустые массивы точек see и $nsee$ размерами 3;
3) если скалярное произведение $plane.normal$ и $poly.v1$ + $plane.d$ >= 0:
    3.1) добавить $v1$ в массив see;
4) иначе:
    4.1) добавить $v1$ в массив $nsee$;
5) аналогично для $v2$ и $v3$;
6) если размер see = 0:
    6.1) вернуть $polys$, $false$;
7) иначе если размер see = 3:
    7.1) добавить $poly$ в $polys$
8) иначе если размер see = 2:
    8.1) $v1$, $v2$, $v3$ = $see[0]$, $see[1]$, $nsee[0]$;
    8.2) добавить в $polys$ полигон, образованный Алгоритм заключается в последовательном отсечении каждого полигона объекта по каждой из плоскостей.$v1$, $v2$ и точкой пересечения ребра $v1v3$ с плоскостью;
    8.3) добавить в $polys$ полигон, образованный $v2$ и точками пересечений рёбер $v1v3$ и $v2v3$ с плоскостью;
9) иначе
    9.1) добавить в $polys$ полигон, образованный видимой точкой и точками пересече-

ния рёбер между видимой точки и невидимыми;

10) вернуть $polys$, $true$

Алгоритм отсечения частей объектов вне поля видимости камеры

При отсечении по зоне видимости используются плоскости, образующие пирамиду видимости. Каждая такая плоскость описывается парой: внутренняя нормаль и скалярное значение. Методом, описанном в [7], были найдены плоскости для используемой модели камеры:

— задняя – $((0, 0, 1), -d)$;

— левая – $((\frac{d}{px}, 0, 1), 0)$;

— правая – $((-\frac{d}{px}, 0, 1), 0)$;

— нижняя – $((0, \frac{d}{py}, 1), 0)$;

— верхняя – $((0, -\frac{d}{py}, 1), 0)$;

Алгоритм отсечения частей объектов вне поля видимости камеры заключается в последовательном отсечении каждого полигона объекта по каждой из плоскостей, по описанному выше алгоритму.

2.6 Разработка алгоритма z-буфера

Алгоритм Z-буфера

Входные данные:

— $objs$ – массив полигональных объектов.

Выходные данные:

— $depthbuf$ – буфер глубины, матрица размерами экрана вещественных чисел;

— $screebbuf$ – буфер экрана, матрица размерами экрана цветов;

— $normalbuf$ – буфер нормалей, матрица размерами экрана векторов нормалей к поверхностям.

1) Инициализировать значение буфера кадра фоновым цветом;
2) Инициализировать буфера глубины максимальным значением вещественных чисел;
3) Для каждого объекта obj из $objs$:
    3.1) для каждого полигона $poly$ из $obj.polygons$:
        3.1.1) растеризовать $poly$;
        3.1.2) для каждого пикселя $pixel$ из растеризованного $poly$, если глубина $pixel$ меньше, чем глубина соответствующего $pixel$ значения из буфера глубины:
        3.1.3) обновить значение цвета в соответствущей $pixel$ ячейки из $screenbuf$;
        3.1.4) обновить нормаль соответствущей $pixel$ ячейки из $screenbuf$ на нормаль $poly$;
4) Вернуть $screenbuf$, $depthbuf$ и $normalbuf$.

2.7 Разработка алгоритма теневого буфера

Алгоритм создания теневого буфера

Входные данные:

— $light$ – точечный источник света;

— $size$ – размер теневых карт;

— $objs$ – массив полигональных объектов.

Выходные данные:

— $shadows$ – массив теневых карт.

Алгоритм

1) Для каждой из 6-ти сторон куба, описывающего источник света:
    1.1) инициализировать теневую карту буфером глубины размером $size$ на $size$;
    1.2) преобразовать объекты $objs$ к пространству экрана стороны куба;
    1.3) построить карту теней алгоритмом z-буфера;
    1.4) добавить карту теней в массив $shadows$;
2) Вернуть $shadows$.

Алгоритм проверки точки в теневом буфере

Входные данные:

— $point$ – точка для проверки;

— $shadows$ – массив теневых карт.

Выходные данные:

— in_shadow – флаг, находится ли $point$ в тени.

Алгоритм

1) $in\_shadows$ = $false$;
2) Для каждой теневой карты из $shadows$:
    2.1) преобразовать $point$ к пространству теневой карты;
    2.2) если глубина $point$ больше, чем глубина соответствующей ячейки из теневой карты:
    2.3) $in\_shadows$ = $true$;
3) Вернуть $in\_shadows$.

Вывод

В результате конструкторской части было создано формальное описание разрабатываемого ПО, определены типы и структуры, необходимые для разработки ПО, а также разработаны основные алгоритмы, которые будут применены в разрабатываемом ПО.

3 Технологическая часть

В этой части будут выбраны средства реализации разрабатываемого ПО, проведено модульное и функциональное тестирование.

3.1 Средства реализации

В качестве основного языка программирования для реализации ПО был выбран Go [14]. Этот язык был выбран так как:

— средствами языка можно реализовать все спроектированные алгоритмы;

— в стандартной библиотеке языка есть средства для реализации всех спроектированных структур.

Для графического интерфейса была выбрана библиотека Fyne [15], так как:

— средств библиотеки достаточно для работы с растровой графикой и вывода её на экран;

— библиотека является кроссплатформенной.

3.2 Модульное тестирование

Модульное тестирование проводилось с помощью стандартного пакета Go – testing.

В качестве меры качества покрытия кода использовалась метрика стандартной утилиты cover входящей в состав Go, которая измеряет покрытие кода как процент от выполненных хотя бы единожды выражений [16].

Тестами были покрыты математические модули, модули матричных преобразований, отсечения и чтения объектов.

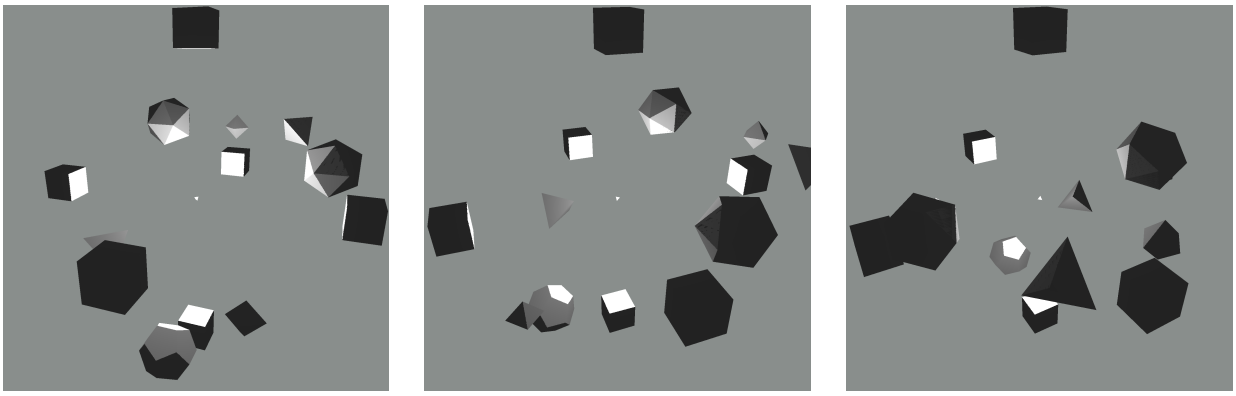
Покрытие кода составило **24.7%**.

Все модульные тесты были пройдены.

3.3 Функциональное тестирование

В качестве функционального тестирования рассмотрены отдельные изображения, полученные разработанным ПО.

На рисунке 3.1 представлены изображения сцены, состоящей из различных объектов с разных ракурсов. В каждом положении направление взгляда было направлен в центр координат.



(a) X=-8.03, Y=8.55, Z=-22.07 (b) X=-20.34, Y=12.49, Z=-7.40 (c) X=-10.82, Y=12.49, Z=18.75

Рисунок 3.1 — «Сцена с разных ракурсов»

На рисунке 3.2 представлена сцена, на которой демонстрируются тени созданные алгоритмом теневых карт при разрешении буферов 128 на 128, На рисунке 3.3 представлена та же сцена, но при увеличенном разрешении – 256 на 256, а на рисунке 3.4 – 512 на 512. Как и ожидалось из-за ограничения в разрешении теневого буфера создаваемые им тени выглядят ступенчато при низком разрешении. При этом чем выше разрешение, тем более гладко выглядят тени

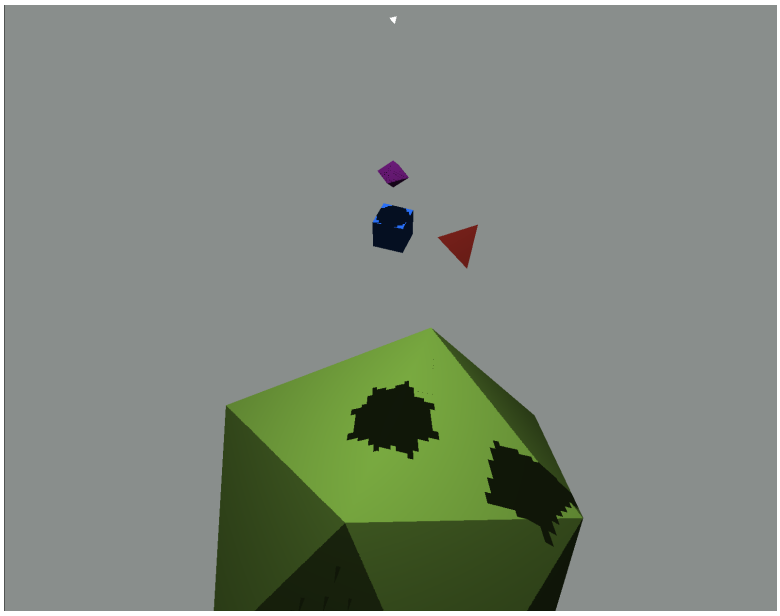


Рисунок 3.2 — «Тени созданные теневыми картами при разрешении 128x128»

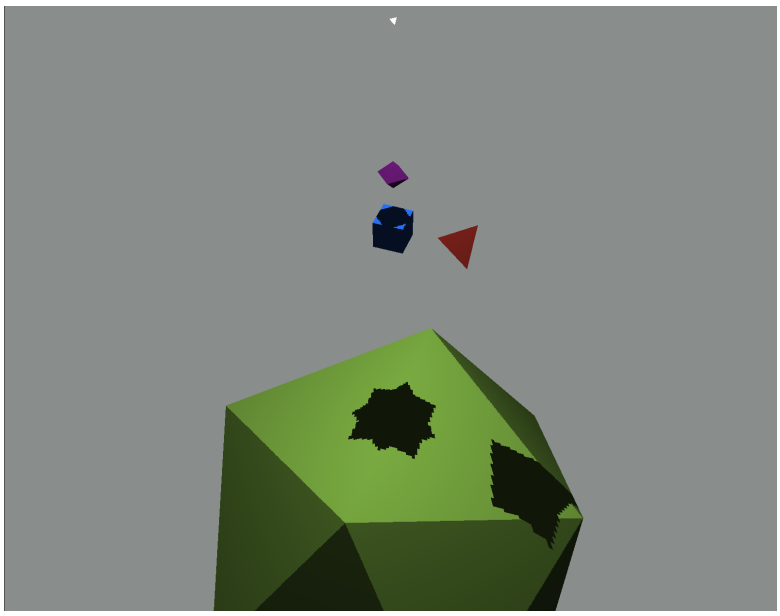


Рисунок 3.3 — «Тени созданные теневыми картами при разрешении 256x256»

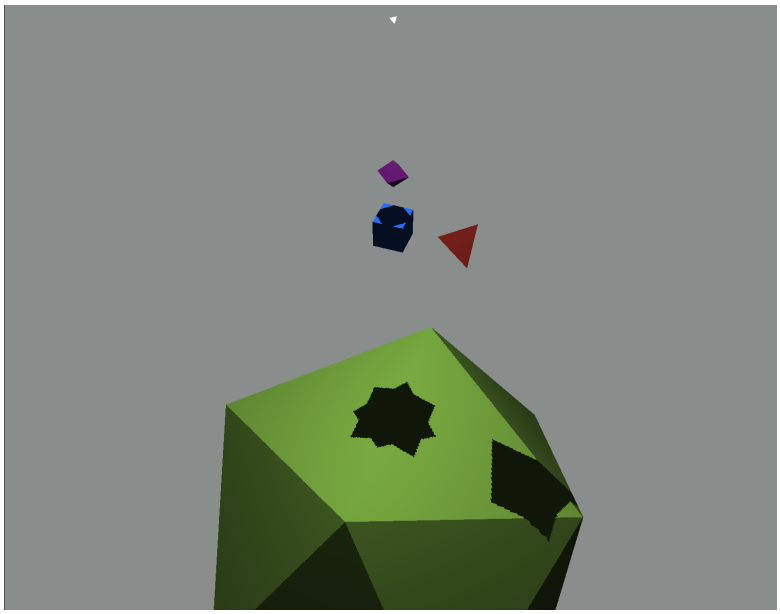
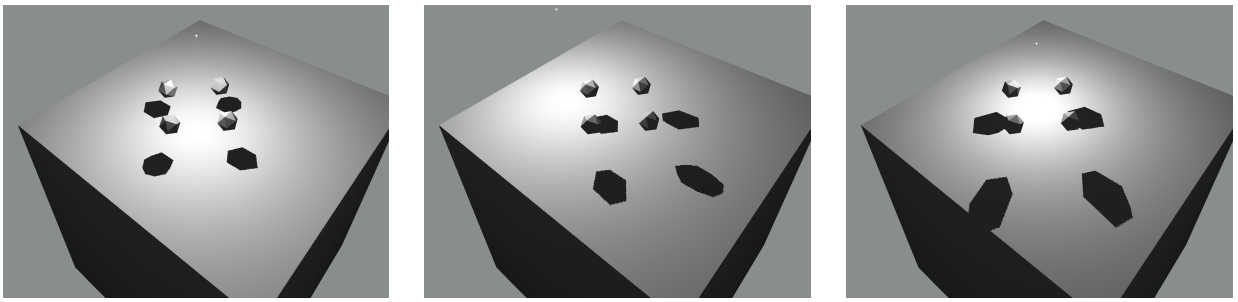


Рисунок 3.4 — «Тени созданные теневыми картами при разрешении 512x512»

На рисунке 3.5 представлена сцена, которая демонстрирует изменения в освещение объектов и отбрасываемых ими тенях при изменении положения точечного источника.



(a) X=0, Y10, Z=0 (b) X=10, Y=10, Z=0 (c) X=3.5, Y=6, Z=3.5

Рисунок 3.5 — «Сцена при изменении положения источника света»

Вывод

В результате технологической частим было реализовано программное обеспечение с графическим интерфейсом для визуализации задачи n тел на языке программирования Go. Разработанное ПО было протестировано. Покрытие кода модульными тестами составило 24.7%, при этом ими были покрыты такие модули как математические модули, модули матричных преобразований, отсечения и чтения объектов.

4 Исследовательская часть

В данной части будет проведён анализ погрешности выбранного численного метода решения задачи n тел.

4.1 Физическая основа исследования

По условию задачи, система состоит из n тел, при этом эти тела взаимодействуют только друг с другом и гравитационно, а сила всемирного тяготения является консервативной, значит система замкнута и имеет место закон сохранения энергии в форме 4.1 [17].

$$E = E_k + E_p = const, \tag{4.1}$$

где E_k – кинетическая энергия системы, а E_p – потенциальная.

Кинетическая энергия системы определяется как сумма кинетических энергий всех входящих её тел 4.2.

$$E_k = \sum_{i=0}^{n} \frac{m_i v_i^2}{2}, \tag{4.2}$$

где m_i – масса i-го тела, а v_i –величина его скорости.

Потенциальная энергия системы определяется как сумма потенциальных энергий всех взаимодействий, имеющих место в системе. В случае гравитационного взаимодействия энергия системы равна сумме энергий всех попарных взаимодействий тел 4.3 [17].

$$E_p = -\sum_{i,j=0;i<j}^{n} G\frac{m_i m_j}{R_{ij}}, \tag{4.3}$$

где $G = 6.67430 * 10^{-11}$ – гравитационная постоянная, константа, R_{ij} – расстояние между i-м и j-м телом.

Таким образом энергия системы в задаче должна сохраняться и отклонение от начальной энергии на каждом шаге численного метода может служить как метрика погрешности метода вычислений.

4.2 Конфигурации

В качестве тестовых конфигураций были выбраны три набора тестовых данных.

Конфигурация 1

Система состоит из 2-х тел находящихся в несовпадающих точках и не имеющих начальную скорость. Таким образом тела сближаются друг с другом до их столкновения.

Конфигурация 2

Система состоит из 4-х тел, каждый из которых лежит в плоскости XoY и все они на разных направлениях данных осей, при этом их скорости направлены перпендикулярно осям, на которых они находятся. Параметры тел подобраны таким образом, что они движутся по

спирали к центру, сближаясь до определённого момента, а затем по спирали отдаляются до позиций близких к исходным, и цикл повторяется.

Конфигурация 3

Система состоит из 1 тела с большой массой и 7 тел с существенно меньшими, при этом начальные скорости заданы так, что они удовлетворяют условию движения по окружности 4.4 и все маломассивные тела вращаются вокруг массивного в разных плоскостях.

$$|\vec{a_n}| = \frac{|\vec{v}|^2}{R} \implies |\vec{v}| = \sqrt{\frac{Gm}{R}} \tag{4.4}$$

4.3 Результаты

В таблице 4.1 представлены результаты исследования погрешностей для каждой из конфигураций.

Из результатов следует, что столкновении тел (конфигурация 1) вызывает большое увеличение погрешности, при этом это характерно для всех исследованных шагов по времени. Такой эффект был ожидаем, так как при сверхблизких дистанциях между объектами делается существенный шаг по времени, при этом на этом шагу учитывается большая сила, соответствующая сверхблизкому расстоянию.

При циклическом сближении и отдалении (конфигурация 2), метод эйлера показывает небольшую погрешность, которая при этом уменьшается вместе с уменьшением шага интегрирования.

При вращении множества маломассивных тел вокруг массивного (конфигурация) погрешность также несущественна, а также никакое из маломассивных тел не слетело с начальной круговой траектории. Однако, в данной конфигурации наблюдается увеличение погрешности при уменьшении шага по времени.

Таблица 4.1 — Результаты оценки погрешности

Время, $0 < t \neq T_{max}$	**Шаг по времени,** τ	**Конфигурация**	**Отклонение энергии**		
			Min, %	**Max, %**	**Avg, %**
11	0.001	1	6.6743e-07	1.01249e+07	1.27481e+06
11	0.0001	1	6.6743e-09	1.10058e+10	1.38693e+09
11	0.00001	1	6.67456e-11	1.88086e+07	2.3705e+06
11	0.000001	1	6.62998e-13	7.7732e+09	9.79688e+08
11	0.0000001	1	0	3.74297e+08	4.71743e+07
300	0.01	2	1.66799e-06	26.642	10.1957
300	0.001	2	1.66799e-08	2.79235	1.21952
300	0.0001	2	1.66797e-10	0.280533	0.123837
10000	0.01	3	3.07938e-11	7.77897e-05	2.30798e-05

Таблица 4.1 — Результаты оценки погрешности

Время, $0 < t \neq T_{max}$	**Шаг по времени,** τ	**Конфигурация**	**Отклонение энергии**		
			Min, %	**Max, %**	**Avg, %**
10000	0.001	3	3.04754e-11	0.000583519	0.000287195
10000	0.0001	3	2.03549e-12	0.00182867	0.00150004

4.4 Вывод

В результате исследовательской части было проведено исследование погрешности механической энергии системы при решении задачи n тел численным методом интегрирования – методом эйлера. В результате исследования погрешность в случаях без столкновения (сближения на сверхблизкие расстояния) тел при шаге интегрирования **0.0001** погрешность механической энергии в среднем составила от **0.0015%** до **0.1238%**, что для задачи визуализации является допустимыми величинами. при шаге интегрирования **0.001** погрешность механической энергии в среднем составила от **0.00028%** до **1.21%**, что также является допустимыми.

В случае столкновения тел наблюдался резкий скачок погрешности($\tilde{1}$0e8%) независимо от шага интегрирования. Такой результат ожидаем для численного метода и является приемлемым для визуализации.

Таким образом в результате исследовательской части было выяснено, что разработанная модель имеет допустимую погрешность.

ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы была описана математическая модель задачи n тел, описан численный метод её решения. Были рассмотрены виды 3-х мерных объектов и способы представления полигональных сеток. Были проанализированы основные методы удаления невидимых поверхностей и выбрана подходящая для разработанного ПО. Была рассмотрена модель освещения на основе закона Ламберта и описан метод учёта теней с помощью теневых карт.

В результате работы было разработано ПО с графическим интерфейсом для визуализации задачи n тел в трёхмерном пространстве. Разработанная ПО позволяет задавать сцены с произвольным числом объектов, задавая их положение, скорость и массу, передвигать источник света и камеру.

В ходе исследования была исследована погрешность используемого численного метода. Исследование показало, что разработанное ПО имеют приемлемую погрешность, кроме случаев столкновения объектов.

В ходе работы поставленные цель и задачи были выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Маркеев, А. П. ЗАДАЧА ТРЕХ ТЕЛ И ЕЕ ТОЧНЫЕ РЕШЕНИЯ [Текст] / А. П. Маркеев // Соровский образовательный журнал. — 1999. — № 9. — С. 112-117.

2. Рубинштейн А.И., Городецкая Т.А., Серебренников П.С., Шипов Н.В., Шмаков А.В. ОБ ОДНОМ ЧАСТНОМ СЛУЧАЕ РЕШЕНИЯ ЗАДАЧИ ТРЕХ ТЕЛ // Проблемы современной науки и образования. - Иваново: Проблемы науки, 2017. - С. 6-9.

3. SOME THEORETICAL AND NUMERICAL ASPECTS OF THE N-BODY PROBLEM // LUND UNIVERSITY LIBRARIES. URL: https://lup.lub.lu.se/student-papers/search/publication/4780668 (дата обращения: 12.09.2024).

4. Самарский А. А. ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ. - СПб: Лань, 2009. - 288 с.

5. Правильные многогранники. // Большая Российская энциклопедия URL: https://bigenc.ru/c/pravil-nye-mnogogranniki-6cab23?ysclid=m3t3ptp0y231309393 (дата обращения: 12.09.2024).

6. Казанцев А.В. Основы компьютерной графики. Часть 1. Математический аппарат компьютерной графики. - Казань: 2001. - 62 с.

7. Гэбриел Гамбетта. Компьютерная графика. Рейтрейсинг и растеризация. — СПб.: Питер, 2022. — 224 с.: ил. — (Серия «Библиотека программиста»).

8. Лекция 9. Удаление невидимых линий. // НГТУ - компьютерные голографические измерительные системы URL: «http://optic.cs.nstu.ru/files/CC/CompGraph/L6_удаление невидимых линий.pdf (дата обращения: 30.11.2024)».

9. On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling : Диссертация на соискание доктора технических наук / Colin Smith ; THE UNIVERSITY OF CALGARY. — Calgary, 2006. — 204 с.

10. Роджерс Д. Алгоритмические основы машинной графики [Текст] / Роджерс Д. — Москва: Мир, 1989 — 512 с.

11. OpenGL Projection Matrix // songho.ca URL: https://songho.ca/opengl/gl_projectionmatrix.html (дата обращения: 30.10.2024).

12. OpenGL Transformation // songho.ca URL: https://songho.ca/opengl/gl_transform.html (дата обращения: 30.10.2024).

13. OpenGL 11: Shadows Part 1 - Directional Shadows // Kovalev's Corner URL: https://rekovalev.site/opengl-11-shadows-p1/ (дата обращения: 11.12.2024).

14. The Go Programming Language // URL: https://go.dev/ (дата обращения: 07.12.2024).

15. Website for the fyne app development framework // URL: https://fyne.io/ (дата обращения: 07.12.2024).

16. The cover story // The Go Programming Lamguage URL: https://go.dev/blog/cover (дата обращения: 07.12.2024).

17. Лекция 4. Закон сохранения энергии в механике. // Научно-учебный комплекс «Фундаментальные науки» МГТУ им. Н.Э. Баумана URL: http://fn.bmstu.ru/files/FN4/lec_2sem/2sem_lec_04.pdf (дата обращения: 7.12.2024).

18. Mario Castro Contreras. Go Design Patterns. - Birmingham: Packt Publishing Ltd., 2017. - 377 с.

Приложение А