



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»

---

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОЙ РАБОТЕ*

### *НА ТЕМУ:*

*«Моделирование изображения объекта в  
неотполированном цветном зеркале»*

Студент ИУ7-53Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

Авдейкина В. П.  
(И. О. Фамилия)

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата)

Новик Н. В.  
(И. О. Фамилия)

*2024 г.*

## РЕФЕРАТ

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>4</b>
<b>1 Аналитическая часть</b>	<b>5</b>
1.1 Формализация задачи синтеза изображения . . . . .	5
1.2 Физическая природа зеркальных поверхностей . . . . .	6
1.3 Существующие алгоритмы решения задачи . . . . .	8
1.3.1 Описание алгоритмов . . . . .	8
1.3.2 Сравнение алгоритмов . . . . .	10
<b>2 Конструкторская часть</b>	<b>12</b>
2.1 Описание функциональной декомпозиции и основные структуры данных . . . . .	12
2.2 Схемы используемых алгоритмов . . . . .	16
2.3 Перенос и поворот . . . . .	16
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>19</b>

# ВВЕДЕНИЕ

Компьютерная графика в 21 веке не перестала быть развивающейся наукой и упоминается в [1], [2], [3], [4], [5]. Одной из стандартных задач компьютерной графики является синтез изображения [2], [6].

За последние 20 лет визуализация зеркальных поверхностей (зеркал) остается актуальной проблемой, о чем свидетельствует ее обсуждение в [7], [8], [9], [10].

Цель работы — разработка программного обеспечения для моделирования статической сцены с изображением предмета в неотполированном цветном зеркале.

Для достижения поставленной цели требуется решить следующие задачи:

- 1) проанализировать предметную область зеркальных поверхностей, выполнить формализацию задачи синтеза изображения в контексте моделирования статической сцены, рассмотреть известные методы и алгоритмы ее решения;
- 2) спроектировать программное обеспечение;
- 3) выбрать средства реализации и разработать программное обеспечение;
- 4) исследовать характеристики разработанного программного обеспечения.

# 1 Аналитическая часть

В данном разделе формализована задача синтеза изображения предмета в неотполированном цветном зеркале в контексте моделирования статической сцены, проанализированы известные алгоритмы решения этой задачи.

## 1.1 Формализация задачи синтеза изображения

На рисунке 1.1 представлена диаграмма IDEF0 формализуемой задачи.

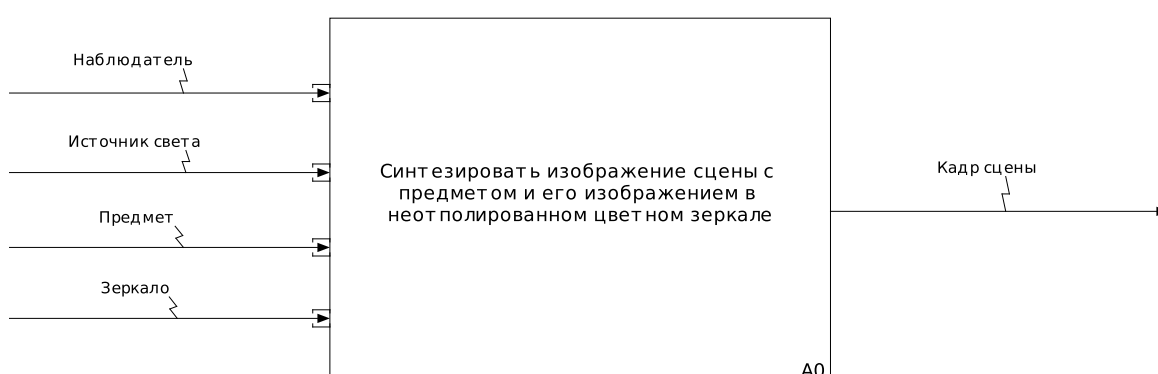


Рисунок 1.1 — Диаграмма IDEF0 формализуемой задачи

*Модель* — отображение формы и размеров какой-либо сущности [6].

В данной работе используется поверхностное представление моделей, при котором они представляются аналитически (полностью или частично) [6].

В качестве поверхностей выбраны треугольные поверхности, формат файлов с моделями — STL.

Процесс получения модели называется *моделированием*.

Сцена состоит из набора сущностей:

- 1) *наблюдатель*, характеризующийся положением точки наблюдения в пространстве, направлениями взгляда и вертикальной оси [6];
- 2) *точечный источник света*, характеризующийся положением в пространстве, интенсивностью и цветом света [6];

- 3) *предмет* — модель геометрического тела, свойства (радиус, высота и так далее) которого определяются заданными пользователем параметрами;
- 4) *зеркало* — модель ограниченной поверхности, обладающая степенью полировки, цветом, радиусом и описанием границ.

Зеркало и предмет называются *объектами (сцены)*.

Модели, используемые в работе, требуют соблюдения физических законов.

## 1.2 Физическая природа зеркальных поверхностей

*Луч (световой)* — узкий пучок света, представленный геометрической линией, вдоль которой в определенном направлении распространяется свет [11], [12].

В геометрической оптике полагается, что луч распространяется прямолинейно до тех пор, пока не встретится отражающая поверхность или граница среды преломления [2].

Если луч падает на поверхность в точку  $A$  и отражается от нее, а через эту точку к поверхности проведена нормаль  $n$ , то углы, заключенные между нормалью и направлениями падающего, отраженного лучей ( $a$ ,  $b$ ), называются углами *падения*, *отражения* соответственно [11], [12], [13]. Описанные элементы представлены на рисунке 1.2.

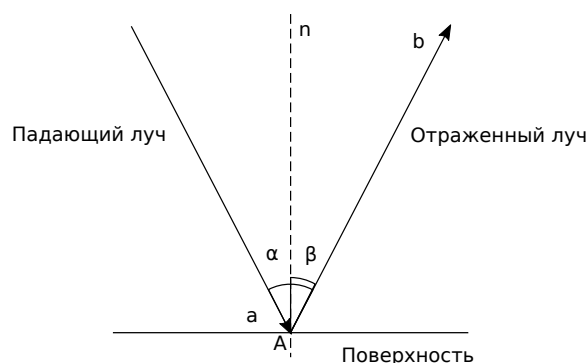


Рисунок 1.2 — Отражение луча от поверхности

*Зеркало* — поверхность, от которой отражаются лучи.

*Зеркальным отражением* называется такое отражение, для которого справедлива формула 1.1:

$$\alpha = \beta, \quad (1.1)$$

где  $\alpha$  — угол падения,  $\beta$  — угол отражения, лучи и нормаль находятся в одной плоскости [2], [12], [13].

Отражение луча света от некоторого объекта сцены представлено на рисунке 1.3. Рисунок 1.3 содержит следующие обозначения:  $\vec{N}$  — нормаль к поверхности в точке падения луча,  $\vec{V}$ ,  $\vec{L}$ ,  $\vec{R}$  — векторы, обозначающие направления от точки объекта к наблюдателю, обратное вектору наблюдения и отраженного луча соответственно,  $\alpha = \angle(\vec{L}, \vec{N})$ ,  $\beta = \angle(\vec{N}, \vec{R})$ ,  $\omega = \angle(\vec{R}, \vec{V})$ .

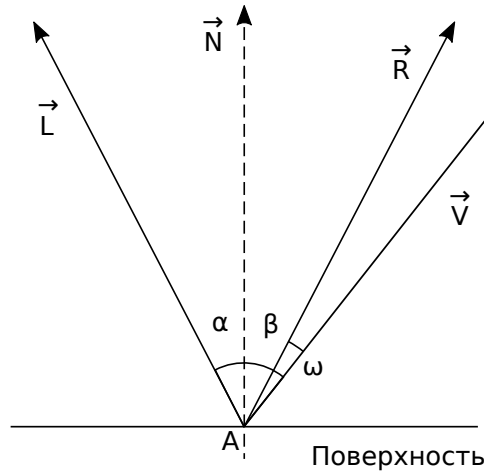


Рисунок 1.3 — Отражение луча света от некоторого объекта сцены

*Идеальным зеркалом* называется такое зеркало, отраженный от которого свет наблюдатель с направлением взгляда  $-\vec{V}$  сможет увидеть только в том случае, если угол  $\omega$  равен нулю, а для  $\alpha$  и  $\beta$  выполняется формула 1.1 [14].

Для зеркал, которые не являются идеальными, отражение описывается моделью Фонга, согласно которой интенсивность  $I_s$  отраженного света представлена в виде формулы 1.2 [2], [14]:

$$I_s = I_p K_s \cos^n \omega, \quad (1.2)$$

где  $K_s$  — коэффициент, учитывающий свойства объекта параметром зеркального отражения,  $n$  — числовой коэффициент, связанный со скоростью убывания интенсивности отраженного света,  $I_p$  — интенсивность точечного источника света.

*Диффузным отражением* называется такое отражение, при котором равенство из формулы 1.1 не выполняется и происходит равномерное по всем направлениям рассеивание отраженного света [2], [14]. Его интенсивность  $I_d$  можно найти по закону Ламберта, выраженному формулой 1.3 [14]:

$$I_d = I_p K_d \cos \alpha, \quad (1.3)$$

где  $K_d$  — коэффициент, учитывающий свойства объекта параметром диффузного отражения,  $\cos \alpha = (\vec{L}, \vec{N})$ .

Согласно модели Уиттеда интенсивность  $I$  некоторой точки с некоторыми оптическими свойствами определяется суммарной интенсивностью по формуле 1.4 с использованием формул 1.2, 1.3 [2], [14]:

$$I = I_a K_a C + I_t K_t + I_p K_d \cos(\vec{L}, \vec{N}) + I_p K_s \cos^n \omega, \quad (1.4)$$

где  $I_t$ ,  $I_a$  — интенсивности преломленного луча и фонового света,  $K_t$ ,  $K_a$  — коэффициенты, учитывающие свойства объекта параметрами прозрачности и фоновой подсветки,  $C$  — заданный цвет точки.

### 1.3 Существующие алгоритмы решения задачи

Среди известных алгоритмов визуализации поверхностей существуют такие алгоритмы, как выбрасывание лучей (raycasting), пошаговое распространение лучей (ray marching), обратная трассировка лучей.

#### 1.3.1 Описание алгоритмов

*Обратная трассировка лучей* представлена авторами [2], [14] с помощью этапов:

- 1) от наблюдателя в каждую точку (пиксель) экрана испускаются первичные лучи;
- 2) один из лучей достигает объекта;
- 3) луч преломляется или отражается, то есть порождает вторичные лучи;
- 4) обработка луча прекращается, когда он перестает пересекать объекты сцены.

Интенсивность некоторой точки объекта в описанном алгоритме вычисляется по формуле 1.4.

Пример визуализации обработки луча в алгоритме обратной трассировки представлен на рисунке 1.4.



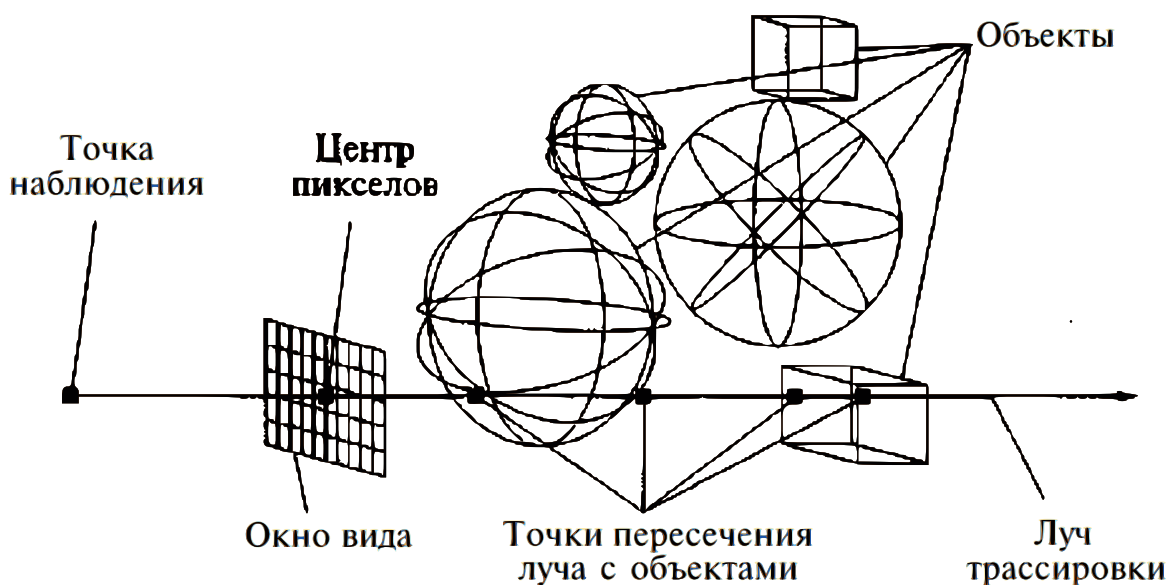


Рисунок 1.4 — Визуализация обработки луча в алгоритме обратной трассировки [15]

В соответствии с [16] *алгоритм выбрасывания лучей (raycasting)* позволяет обрабатывать поверхностные модели и представляется в виде последовательности шагов (выполняемыми для каждого шага угла обзора):

- 1) испустить луч из точки наблюдения;
- 2) рассчитать расстояние от места испускания луча до каждого из объектов сцены;
- 3) определить, для какого из них оно является наименьшим;
- 4) выполнить визуализацию вертикальной «полосы» этого объекта.

Пример визуализации процесса обработки модели алгоритмом выбрасывания лучей представлен на рисунке 1.5.

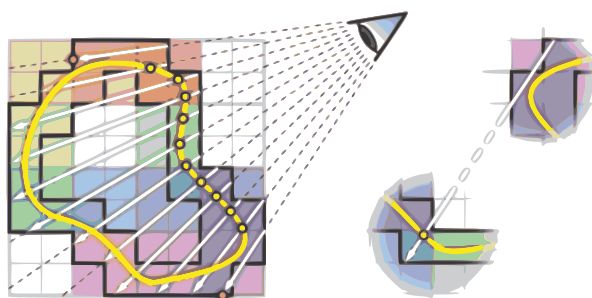


Рисунок 1.5 — Визуализация процесса обработки 2D карты алгоритмом выбрасывания лучей (raycasting) [17]

Алгоритм пошагового распространения лучей (*ray marching*) описывается в [18] для каждого луча, испускаемого из точки наблюдения через пиксели экрана, следующими действиями:

- 1) рассчитать значения функции расстояния со знаком до каждого объекта сцены относительно текущей точки начала распространения луча;
- 2) выбрать наименьшее из этих значений;
- 3) если выбранное расстояние меньше некоторой заданной малой величины, запустить рекурсивное распространение лучей из точки объекта в соответствии с оптическими свойствами его поверхности;
- 4) если выбранное расстояние больше некоторой заданной большой величины, визуализировать фон;
- 5) принять за следующую точку начала ту, которая находится на выбранном расстоянии по направлению распространения исходного луча, перейти на шаг 1.

Пример визуализации обработки одного луча в алгоритме пошагового распространения представлен на рисунке 1.6.

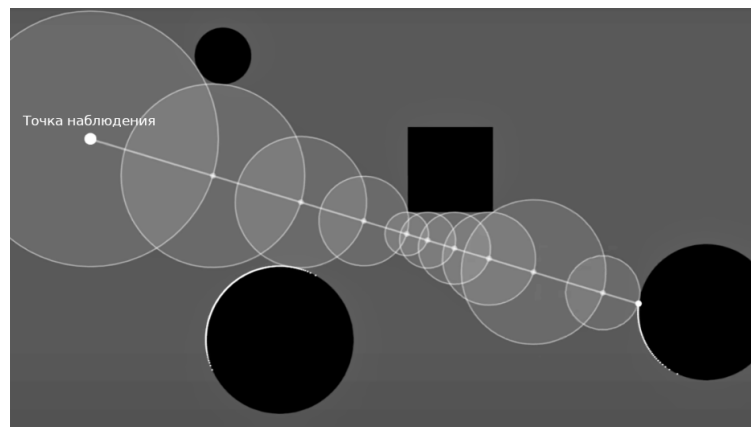


Рисунок 1.6 — Визуализация обработки одного луча в алгоритме пошагового распространения [18]

### 1.3.2 Сравнение алгоритмов

На основе источников, которые были использованы для приведенного выше описания, осуществлено сравнение анализируемых алгоритмов.

В таблице 1.1 приведены результаты сравнения, критерии которого расположены по горизонтали и включают в себя:

- 1) сложность алгоритма в зависимости от чисел пикселей  $C$  и объектов  $N$ ;
- 2) рабочее пространство алгоритма (сцены — «С», экранное — «Э»);
- 3) форма моделей, которая может быть использована при использовании алгоритма (каркасная — «К», поверхностная — «П», объемная — «О»);
- 4) возможность совмещения алгоритма и модели освещения Уиттеда;
- 5) принадлежность обрабатываемых точек объекту.

Таблица 1.1 — Сравнение существующих алгоритмов решения задачи синтеза изображения в контексте моделирования статической сцены

	1	2	3	4	5
Выбрасывание лучей	$O(C \cdot N)$	Э	П	Нет	Да
Пошаговое распространение лучей	$O(C \cdot N)$	Э	О	Да	Нет
Обратная трассировка лучей	$O(C \cdot N)$	Э	О	Да	Да

## Вывод

В данном разделе была выполнена формализация объектов сцены и задачи синтеза изображения предмета в неотполированном цветном зеркале в контексте моделирования статической сцены, проанализированы известные алгоритмы решения этой задачи, предметная область зеркал.

В качестве основного алгоритма выбрана обратная трассировка лучей с использованием модели освещения Уиттеда.

## 2 Конструкторская часть

В данном разделе выполняется и формально описывается функциональная декомпозиция разрабатываемого ПО, даются описания используемых структур данных, описываются применяемые алгоритмы.

### 2.1 Описание функциональной декомпозиции и основные структуры данных

На рисунке 2.1 представлена диаграмма IDEF0 результата функциональной декомпозиции разрабатываемого программного обеспечения.

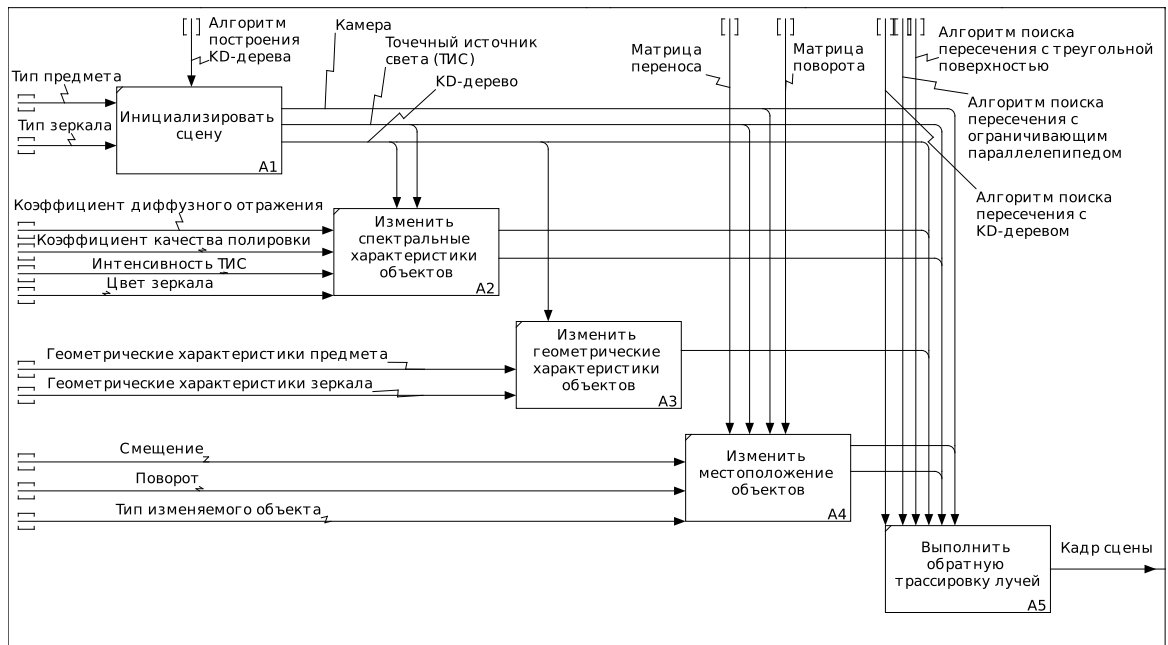


Рисунок 2.1 — Диаграмма IDEF0 результата функциональной декомпозиции ПО

С учетом параметрического уравнения прямой и условия коллинеарности векторов выводится формула 2.1 — уравнение для нахождения вектора с началом в  $(0, 0, 0)$  и концом в конце луча, выпущенного из точки  $O$  (origin) по направлению от точки  $O$  до  $F$  —  $\vec{D}$  (destination) [15], [19], [20]:

$$\vec{P} = \vec{O} + t \cdot \vec{D} \quad (2.1)$$

Если поверхность, пересечение луча с которой требуется найти, однозначно задана и известна, то коэффициент  $t$  находится с помощью формулы 2.2,

то есть для полной информации о луче в рассматриваемых далее алгоритмах достаточно двух величин —  $\vec{O}, \vec{D}$ .

$$t = \frac{\vec{P} - \vec{O}}{\vec{D}} \quad (2.2)$$

В алгоритме обратной трассировки лучей необходимо, чтобы была известна информация о точке пересечения луча с некоторой поверхностью (для обработки вторичных лучей) и ее спектральных характеристиках (для выполнения вычислений по формуле 1.4).

Вычислительная стоимость определения пересечений произвольного луча с одним выделенным объектом, как указывают авторы [15], [19], [20], может оказаться высокой, поэтому для избавления от ненужной части поиска производится проверка пересечения луча с ограничивающим телом объекта. *Ограничивающее тело* — некоторое простое геометрическое тело (например, параллелепипед, сфера), описанное около одного или нескольких объектов сцены [20]. Для организации ограничивающих тел используются иерархические структуры, одной из которых является kD-дерево [20]. *kD-деревом* называется бинарное дерево ограничивающих параллелепипедов, вложенных друг в друга [20].

Таким образом, возникает необходимость наличия следующих структур (совокупностей) данных для упрощения формального описания используемых алгоритмов:

- Point — информация о координатах в декартовой системе координат (значения  $x, y, z$ );
- Ray — информация о луче: начало (*origin*), направление (*destination*);
- Material — информация о спектральных характеристиках поверхности: интенсивность фонового излучения (*ambient*), диффузная составляющая интенсивности поверхности (*diffuse*), коэффициент отражения (*reflection*), коэффициент качества полировки (*polish*);
- Hitinfo — информация о пересечении луча с поверхностью: значение параметра в соответствии с формулой 2.2 ( $t$ ), информация о материале поверхности (*material*), точка пересечения (*hitPoint*);

- BoundingBox — информация об ограничивающем параллелепипеде: минимальные и максимальные координаты тела ( $min\_p$ ,  $max\_p$  соответственно);
- Object — информация об объекте сцены: данные поверхности ( $data$ ), центр ( $center$ ), ограничивающий параллелепипед ( $bbox$ );
- kDNode — информация об узле kD-дерева: дочерние узлы ( $left$ ,  $right$ ), признак листа ( $isLeaf$ ), ограничивающий параллелепипед ( $bbox$ ), соответствующая направлению декомпозиции ограничивающего параллелепипеда ось ( $axis$ ), массив объектов ( $objects$ ), количество объектов ( $size$ ).

Камера (Camera) — структура данных, содержащая информацию о наблюдателе: местоположение, направления взгляда и вертикальной оси. Дополнительной информацией являются отношение сторон используемого экрана и область видимости (угол  $fov$ ), необходимые для обеспечения возможности применения алгоритма обратной трассировки лучей. Местоположение экрана зависит от камеры.

Используемая модель камеры представлена на рисунке 2.2.

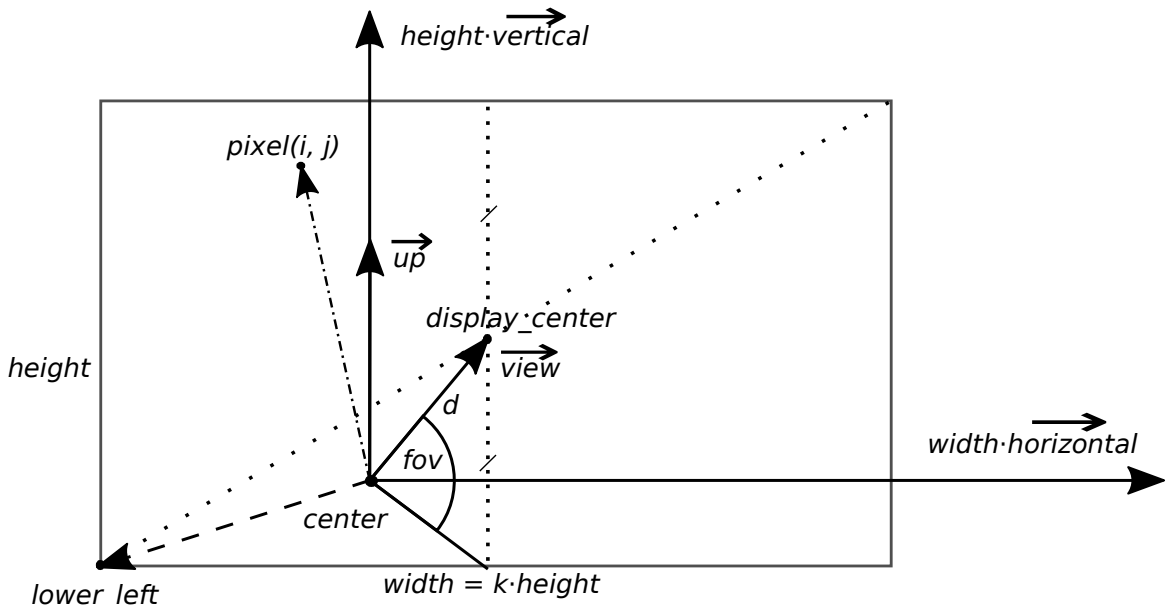


Рисунок 2.2 — Используемая модель камеры

Для выполнения трассировки лучей необходимо иметь возможность вычислить луч, выпущенный из камеры, расположенной в точке  $center$ , через некоторый пиксель экрана  $(i, j)$ .

Пусть:

- 1) центр экрана — точка  $display\_center$ , его ширина и высота —  $width$  и  $height$ , а их отношение —  $k = \frac{width}{height}$ ;
- 2) камера расположена в точке  $center$ , направление взгляда —  $\overrightarrow{view}$ , вертикальная ось —  $\overrightarrow{up}$ , область видимости задана углом  $fov$ ;
- 3) расстояние между камерой и экраном —  $d$ .

Тогда из определения тангенса угла выводится формула 2.3:

$$height = 2 \cdot d \cdot tg(fov), \quad (2.3)$$

где  $d$  принимается равным 1.

Согласно определению  $k$  получается формула 2.4:

$$width = k \cdot height. \quad (2.4)$$

Для вычисления горизонтального и вертикального направлений используются формулы 2.5, 2.6 соответственно (после применения каждой из формул результат необходимо нормализовать).

$$\overrightarrow{horizontal} = \overrightarrow{view} \times \overrightarrow{up} \quad (2.5)$$

$$\overrightarrow{vertical} = \overrightarrow{horizontal} \times \overrightarrow{view} \quad (2.6)$$

Искомый луч высчитывается относительно начала отсчета, в качестве которого выбрана нижняя левая точка экрана —  $lower\_left$  (то есть вектор  $\overrightarrow{lower\_left}$ ). Для нахождения начала отсчета используется формула 2.7:

$$\overrightarrow{lower\_left} = \overrightarrow{view} - \frac{\overrightarrow{display\_h} + \overrightarrow{display\_v}}{2}, \quad (2.7)$$

где  $\overrightarrow{display\_h} = width \cdot \overrightarrow{horizontal}$ ,  $\overrightarrow{display\_v} = height \cdot \overrightarrow{vertical}$ .

С использованием формулы 2.7 получаем конечную формулу 2.8 для вычисления луча:

$$\overrightarrow{ray} = \overrightarrow{lower\_left} + i \cdot \overrightarrow{display\_h} + j \cdot \overrightarrow{display\_v} \quad (2.8)$$

Доступ к данным структур будет условно обозначаться с использованием оператора «.».

## 2.2 Схемы используемых алгоритмов

Схема алгоритма построения kD-дерева представлена на рисунке ??.

Схема алгоритма поиска пересечения луча с kD-деревом представлена на рисунке ??.

Схема алгоритма добавления объекта в kD-дерево представлена на рисунке ??.

На рисунке ?? приведены схемы алгоритмов обработки одного луча и перебора всех лучей, выпускаемых из камеры.

## 2.3 Перенос и поворот

Также согласно техническому заданию необходимо изменять положение источника света, то есть дать пользователю возможность выполнить перенос в отношении источника света, который реализуется с помощью матрицы переноса в трехмерном пространстве, представленной в формуле 2.9 [6], [20].

Аналогично перенос камеры осуществляется с помощью матрицы, представленной в формуле 2.9, а поворот в отношении камеры реализуется с использованием матриц поворота вокруг осей  $x$ ,  $y$ ,  $z$  на угол  $\alpha$ , которые представлены в формулах 2.10, 2.11, 2.12 соответственно [6], [20].

$$M(dx, dy, dz) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{pmatrix} \quad (2.9)$$

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \quad (2.10)$$

$$R_y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \quad (2.11)$$



$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.12)$$

## Вывод

В данном разделе были формально описаны функциональная декомпозиция разрабатываемого ПО и применяемые алгоритмы, даны описания используемых структур данных.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Залогова Л. А.* Компьютерная графика. Элективный курс : практикум. — М. : БИНОМ. Лаборатория знаний, 2005. — С. 245.
2. *Порев В.* Компьютерная графика // СПб.: БХВ-Петербург. — 2002. — Т. 432. — С. 3.
3. *Большаков В. П., Тозик В. Т., В Ч. А.* Инженерная и компьютерная графика. — БХВ-Петербург, 2013. — С. 288.
4. *Митин А. И., Свертилова Н. В.* Компьютерная графика : справочно-методич. пособие. — М.–Берлин : Директ-медиа, 2016. — С. 251.
5. *Турлюн Л. Н.* Компьютерная графика как особый вид современного искусства. — Издательство АлтГУ, 2014.
6. *Куров А. В.* Компьютерная графика : неопубликованный конспект лекций. — 2023.
7. Realistic materials in computer graphics / Н. Р. Lensch [и др.] // ACM SIGGRAPH 2005 Courses. — 2005.
8. *Reshetouski I., Ihrke I.* Mirrors in computer graphics, computer vision and time-of-flight imaging // Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications: Dagstuhl 2012 Seminar on Time-of-Flight Imaging and GCPR 2013 Workshop on Imaging New Modalities. — Springer. 2013. — С. 77—104.
9. *Miguel A. L., Nogueira A. C., Goncalves N.* Real-time 3D visualization of accurate specular reflections in curved mirrors a GPU implementation // 2014 International Conference on Computer Graphics Theory and Applications (GRAPP). — IEEE. 2014. — С. 1—8.
10. Demonstration of light reflection concepts for rendering realistic 3D tree images / Т. Hiranyachattada [и др.] // Journal of Physics: Conference Series. Т. 2145. — IOP Publishing. 2021. — С. 12—74.
11. *Тюрин Ю. И., Чернов И. П., Крючков Ю. Ю.* Физика. Оптика: учебник. — Томск : Издательство Томского политехнического университета, 2009. — С. 240.

12. *Ландсберг Г. С.* Оптика. Учебное пособие: Для вузов. — М : ФИЗМАТ-ЛИТ, 2003. — С. 848.
13. *Родионов С. А.* Основы оптики : конспект лекций. — СПб. : СПб ГИТМО (ТУ), 2000. — С. 167.
14. *Демин А. Ю.* Основы компьютерной графики: учебное пособие. — Томск : Издательство Томского политехнического университета, 2011. — С. 191.
15. *Божско А. Н., Жук Д. М., Маничев В. Б.* Компьютерная графика : учебное пособие для вузов. — М. : изд-во МГТУ им. Н.Э. Баумана, 2007. — С. 392.
16. *Евстратов В. В.* Создание программы визуализации псевдотрехмерного изображения с помощью рейкастинга // Молодой ученый. — 2020. — № 50. — С. 12—15.
17. The ray casting engine and ray representatives / J. L. Ellis [и др.] // Proceedings of the first ACM symposium on solid modeling foundations and CAD/CAM applications. — 1991. — С. 255—267.
18. *Bredenbals A.* Visualising Ray Marching in 3D / Bredenbals Anton. — Groningen : University of Groninge, 2022. — Режим доступа: <https://fse.studenttheses.ub.rug.nl/id/eprint/27977> (дата обращения: 05.12.2023).
19. *Роджерс Д.* Алгоритмические основы машинной графики: Пер. с англ. — М. : Мир, 1989. — С. 512.
20. *Боресков А. В.* Программирование компьютерной графики. — М. : ДМК Пресс, 2019. — С. 372.