ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

КОМПЬЮТЕРНАЯ ГРАФИКА

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» и направлениям подготовки 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия» дневной и заочной форм обучения

Могилев 2018

УДК 621.01
ББК 36.4
К 87

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «06» февраля 2018 г., протокол № 10

Составитель канд. техн. наук, доц. А. В. Шилов

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации к курсовому проектированию предназначены для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» и направлениям подготовки 09 03 01 «Информатика и вычислительная техника», 09 03 04 «Программная инженерия».

Учебно-методическое издание

КОМПЬЮТЕРНАЯ ГРАФИКА

Ответственный за выпуск А. И. Якимов

Технический редактор С. Н. Красовская

Компьютерная верстка Н. П. Полевничая

Подписано в печать. Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014
Пр. Мира, 43, 212000, Могилев.

Содержание

Введение

Целью изучения учебной дисциплины «Компьютерная графика» является формирование у обучающихся навыков осваивать новые и применять существующие алгоритмы компьютерной графики, графических приложений, инструментария для написания приложений, стандартов в области разработки графических систем.

Целью данных методических рекомендаций является приобретение обучающимися знаний в областях геометрического моделирования, свойств геометрических моделей, параметризации моделей, геометрических операций над моделями, алгоритмов визуализации: отсечения, развертки, удаления невидимых линий и поверхностей, закраски; способов создания фотореалистических изображений, основных функциональные возможности современных графических систем, организация диалога в графических системах; классификация и обзор современных графических систем. При этом на практике должны быть получены навыки работы с программными средствами, обеспечивающими: аппаратную реализацию графических функций, ввод и вывод графической информации; преобразование: систем координат, форматов хранения графической информации; разработка «открытых» графических систем; 2-D и 3-D моделирование.

Задачей компьютерной графики является визуализация кривых и прямых линий и поверхностей, исходя из их математического описания. Для решения поставленной задачи применяют методы аналитической геометрии и векторной алгебры, начертательной геометрии и черчения, теории графов и математической логики. Компьютерная графика, объединяя эти методы, является самостоятельной научной дисциплиной и реализуется в программном обеспечении современных графических пакетов в различных областях для создания реалистичных изображений различных объектов.

1 Формирование цветов

Цель работы: изучение цветовых характеристик и аддитивной цветовой модели RGB.

Эта модель используется для описания цветов, которые получаются с помощью устройств, основанных на принципе излучения. В качестве основных цветов выбран красный (Red), зеленый (Green) и синий (Blue). Иные цвета и оттенки получаются смешиванием определенного количества указанных основных цветов (рисунок 1.1).

Томас Юнг (1773–1829) взял три фонаря и приспособил к ним красный, зеленый и синий светофильтры. Так были получены источники света соответствующих цветов. Направив на белый экран свет этих трех источников, ученый получил такое изображение.

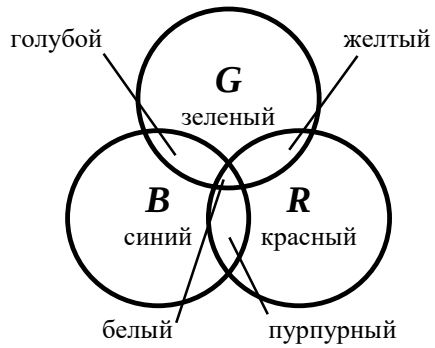


Рисунок 1.1 – Основные цвета RGB и их смешение

Некоторое время спустя, Джемс Максвелл (1831–1879) изготовил первый колориметр, с помощью которого человек мог зрительно сравнивать монохроматический цвет и цвет смешивания в заданной пропорции компонент RGB. Регулируя яркость каждой из смешиваемых компонент, можно добиться уравнивания цветов смеси и монохроматического излучения. Это описывается следующим образом:

$$C = r \cdot R + g \cdot G + b \cdot B,$$

где r, g и b – количество соответствующих основных цветов.

Соотношение коэффициентов r, g и b Максвелл наглядно показал с помощью треугольника, впоследствии названного его именем. Треугольник Максвелла является равносторонним, в его вершинах располагаются основные цвета – R, G и B (рисунок 1.2). Из заданной точки проводятся линии, перпендикулярные сторонам треугольника. Длина каждой линии и показывает соответствующую величину коэффициента r, g или b. Одинаковые значения $r = g = b$ имеют место в центре треугольника и соответствуют белому цвету. Следует также отметить, что некоторый цвет может изображаться как внутренней точкой такого треугольника, так и точкой, лежащей за его пределами. В последнем случае это соответствует отрицательному значению

соответствующего цветового коэффициента. Сумма коэффициентов равна высоте треугольника h, а при высоте равной единице $h = r + g + b = 1$.

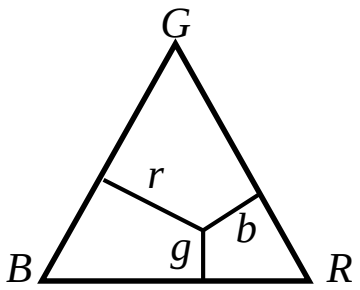


Рисунок 1.2 – Треугольник Максвелла

В качестве основных цветов Максвелл использовал излучения с длинами волн 630, 528 и 457 нм.

К настоящему времени система RGB является официальным стандартом. Решением Международной Комиссии по Освещению (МКО) в 1931 г. были стандартизованы основные цвета, которые было рекомендовано использовать в качестве *R*, *G* и *B*. Это монохроматические цвета светового излучения с длинами волн соответственно: *R* – 700 нм; *G* – 546,1 нм; *B* – 435,8 нм.

Цвет, создаваемый смешиванием трех основных компонент, можно представить вектором в трехмерной системе координат *R*, *G* и *B*, изображенной на рисунке 3 Черному цвету соответствует центр координат – точка *O*(0, 0, 0). Белый цвет выражается максимальным значением компонент. Пусть это максимальное значение вдоль каждой оси равно единице. Тогда белый цвет – это вектор (1, 1, 1). Точки, лежащие на диагонали куба от черного к белому, соответствуют равным значениям: $R = G = B$ (см. рисунок 1.3). Это градации серого – их можно считать белым цветом различной яркости. Вообще говоря, если все компоненты вектора (r, g, b) умножить на одинаковый коэффициент ($k = 0...1$), то цвет (k_r, k_g, k_b) сохраняется, изменяется только яркость. Поэтому, для анализа цвета важно соотношение компонент.

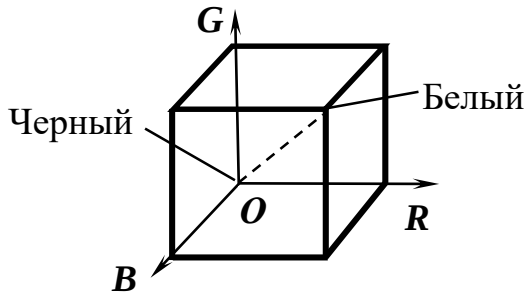


Рисунок 1.3 – Трехмерные координаты *RGB*

Для того чтобы компьютер имел возможность работать с цветными изображениями, необходимо представлять цвета в виде чисел – кодировать цвет. Для модели *RGB* каждая из компонент может представляться числами,

ограниченными некоторым диапазоном – например, дробными числами от 0 до 1 либо целыми числами от 0 до некоторого максимального значения. В настоящее время достаточно распространенным является формат True Color, в котором каждая компонента представлена в виде байта, что дает 256 градаций для каждой компоненты: R = 0...255, G = 0...255, B = 0...255. Количество цветов составляет 256 × 256 × 256 = 16,7 млн.

Такой способ кодирования цветов можно назвать компонентным. В компьютере коды изображений True Color представляются в виде троек байтов, либо упаковываются в длинное целое (четырехбайтное) – 32 бита (так, на пример, сделано в API Windows):

C = 00000000 bbbbbbbb gggggggg rrrrrrrr.

Практическое задание

Разработать программу, которая формирует треугольник Максвелла.

Содержание отчета: титульный лист (пример в приложении Б), цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 1.1 – Вопросы для защиты

Вопрос	Рейтинг
Дать определение термину «Растр»	1
Геометрические характеристики растра	1
Оценка разрешающей способности растра	1
Аддитивная цветовая модель RGB	1
Формирование треугольника Максвелла	2
Трехмерные координаты RGB	1
Соотношение для перекодирования цвета из модели CMY в RGB	1
Компонентный способ кодирования цветов	1
Коды изображений True Color	1
Функция VBA RGB	1

2 Алгоритмы вычерчивания отрезков прямых

Цель работы: изучение алгоритма Брезенхема для вычерчивания отрезков.

Одним из методов разложения отрезка в растр является алгоритм Брезенхема. Этот алгоритм определяет, какие точки двумерного растра нужно закрасить, чтобы получить близкое приближение прямой линии между двумя заданными точками. Большее из приращений, либо Δx, либо Δy, выбирается в качестве единицы растра. В процессе работы одна из координат (в зависимости от углового коэффициента) изменяется на единицу. Изменение другой координаты (на 0 или 1) зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. Такое расстояние мы назовем ошибкой.

Алгоритм построен так, что требуется проверять лишь знак этой ошибки. Из рисунка 2.1 можно заметить, что если угловой коэффициент отрезка из точки с координатами (0, 0) больше чем 1/2, то его пересечение с прямой $x = 1$ будет расположено ближе к прямой $y = 1$, чем к прямой $y = 0$. Следовательно, точка растра (1, 1) лучше аппроксимирует ход отрезка, чем точка (1, 0). Если угловой коэффициент меньше 1/2, то верно обратное. Для углового коэффициента, равного 1/2, нет какого-либо предпочтительного выбора. В данном случае алгоритм выбирает точку (1, 1).

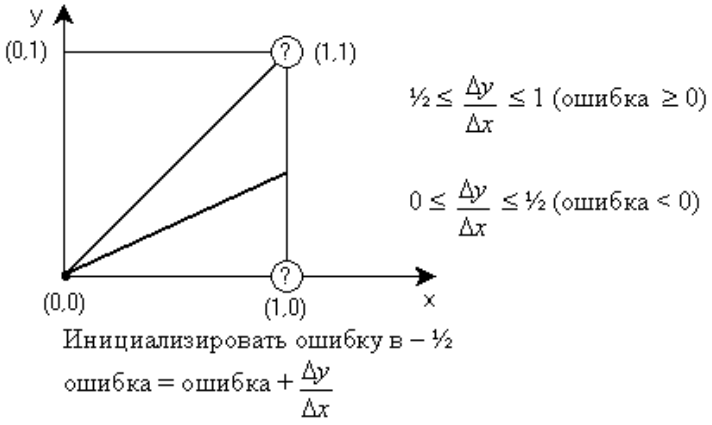


Рисунок 2.1 – Основная идея алгоритма Брезенхема

Так как желательно проверять только знак ошибки, то она первоначально устанавливается равной –1/2. Таким образом, если угловой коэффициент отрезка больше или равен 1/2, то величина ошибки в следующей точке растра может быть вычислена как $e = –1/2 + \Delta y/\Delta x$. Недостаток такого вычисления ошибки в том, что она требует использования арифметики с плавающей точкой и деления для вычисления углового коэффициента. Быстродействие алгоритма можно увеличить, если использовать только целочисленную арифметику и исключить деление.

Не все отрезки проходят через точки растра, поэтому рассмотрим пример, где отрезок с тангенсом угла наклона 3/8 сначала проходит через точку растра (0, 0) и последовательно пересекает три пиксела (рисунок 2.2, а).

Результаты вычисления ошибки при представлении отрезка дискретными пикселами иллюстрируются рисунком 2.2, б. Так как желательно проверять только знак ошибки, то она первоначально устанавливается равной –1/2. Таким образом, если угловой коэффициент отрезка больше или равен 1/2, то величина ошибки в следующей точке растра с координатами (1,0) может быть вычислена как

$$e_1 = e_0 + m,$$

где m – угловой коэффициент.

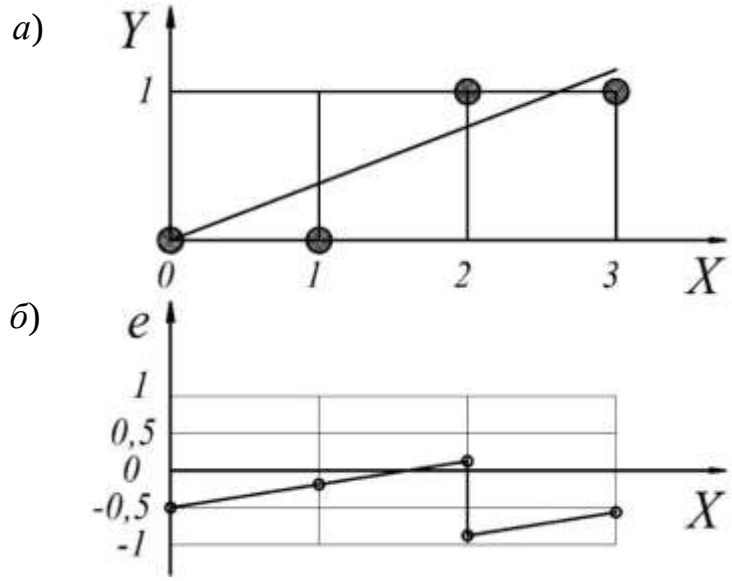


Рисунок 2.2 – Изображение отрезка в растре при его построении методом Брезенхема (*а*) и график ошибки (*б*)

В нашем случае при начальном значении ошибки –1/2

$$e_1 = -1/2 + 3/8 = -1/8.$$

Так как e отрицательно, отрезок пройдет ниже середины пиксела. Следовательно, пиксел на том же самом горизонтальном уровне лучше аппроксимирует положение отрезка, поэтому y не увеличивается. Аналогично вычисляем ошибку в следующей точке растра (2, 0)

$$e_2 = -1/8 + 3/8 = -1/4.$$

Теперь e положительно, а значит, отрезок пройдет выше средней точки. Растровый элемент (2, 1) со следующей по величине координатой y лучше аппроксимирует положение отрезка. Следовательно, y увеличивается на единицу. Прежде чем рассматривать следующий пиксел, необходимо откорректировать ошибку вычитанием из нее единицы

$$e'_2 = 1/4 - 1 = -3/4.$$

Заметим, что пересечение вертикальной прямой $x = 2$ с заданным отрезком лежит на 1/4 ниже прямой $y = 1$. Если же перенести отрезок 1/2 вниз, мы получим как раз величину –3/4. Продолжение вычислений для следующего пиксела дает

$$e_3 = -3/4 + 3/8 = -3/8.$$

Так как e отрицательно, то y не увеличивается. Из всего сказанного следует, что ошибка – это интервал, отсекаемый по оси y рассматриваемым отрезком в каждом растровом элементе (относительно –1/2).

Практическое задание

Разработать программу для отображения объекта № 1.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 2.1 – Вопросы для защиты

Вопрос	Рейтинг
Записать рекуррентное соотношение для последовательных значений по методу ЦДА	1
Алгоритм формирования отрезка методом ЦДА	1
Представить график ошибки в алгоритме Брезенхема	1
Как изменяется величина ошибки при закрашивании пиксела по оси X	1
Как изменяется величина ошибки при закрашивании пиксела по оси Y	1
Пояснить схему алгоритма построения отрезка методом Брезенхема	2

3 Формирование цветов с помощью дизеринга

Цель работы: изучение технологии формирования цветов с помощью дизеринга.

Хорошо, когда растровое устройство отображения может прямо воссоздавать тысячи цветов для каждого пикселя. Не так давно это было проблемой даже для компьютерных дисплеев (а точнее – для видеоадаптеров). Современные растровые дисплеи достаточно качественно отображают миллионы цветов, благодаря чему без проблем можно отображать цветные фотографии. Но для растровых устройств, которые печатают на бумаге, положение совсем другое. Устройства печати обычно имеют высокую разрешающую способность (dpi), часто на порядок большую, чем дисплеи. Однако нельзя непосредственно воссоздать даже сотню градаций серого для пикселей черно-белых фотографий, не говоря уже о миллионах цветов. В большинстве случаев можно увидеть, что оттенки цветов (для цветных изображений) или полутоновые градации (для черно-белых) имитируются комбинированием, смесью точек. Чем качественнее полиграфическое оборудование, тем меньше отдельные точки и расстояние между ними.

Для устройств печати на бумаге проблема количества красок достаточно важна. В полиграфии для цветных изображений обычно используют три цветных краски и одну черную, что в смеси дает восемь цветов (включая черный и белый цвет бумаги). Встречаются образцы печати большим количеством красок – например, карты, напечатанные с использованием восьми красок, однако такая технология печати намного сложнее. Состояние дел с цветной печатью можно оценить на примере относительно простых офисных принтеров, цветные картриджи которых содержат четыре цвета в системе CMYK (**C**yan, **M**agenta, **Y**ellow, Blac**K**). Существуют струйные принтеры с шестью и семью цветными

красками вместо четырех. В семицветных принтерах в палитру обычных красок добавлены бледно-голубая, бледно-пурпурная и бледно-желтая краски. В шестицветных принтерах бледно-желтая краска отсутствует. Увеличение количества красок значительно улучшило качество печати, однако, и этого пока явно недостаточно для полноценной цветопередачи.

Если графическое устройство не способно воссоздавать достаточное количество цветов, тогда используют растрирование – независимо от того, растровое это устройство или не растровое. В полиграфии растрирование известно давно. Оно использовалось несколько столетий тому назад для печати гравюр. В гравюрах изображение создается многими штрихами, причем полутоновые градации реализованы или штрихами различной толщины на одинаковом расстоянии, или штрихами одинаковой толщины с переменной густотой расположения. Такие способы используют особенности человеческого зрения и в первую очередь – пространственную интеграцию. Если достаточно близко расположить маленькие точки различных цветов, то они будут восприниматься как одна точка с некоторым усредненным цветом. Если на плоскости густо расположить много маленьких разноцветных точек, то будет создана визуальная иллюзия закрашивания плоскости некоторым усредненным цветом. Однако если увеличивать размеры точек и (или) расстояние между ними, то иллюзия сплошного закрашивания исчезает – включается другая система человеческого зрения, обеспечивающая нашу способность различать отдельные объекты, подчеркивать контуры.

В компьютерных графических системах часто используют эти методы. Они позволяют увеличить количество оттенков цветов за счет снижения пространственного разрешения растрового изображения. Иначе говоря – это обмен разрешающей способности на количество цветов. В литературе по компьютерной графике такие методы растрирования получили название dithering (дрожание, разрежение).

Простейшим вариантом дизеринга можно считать создание оттенка цвета парами соседних пикселей.

Если рассмотреть ячейки из двух пикселей, то ячейка номер 1 дает оттенок цвета:

$$C = \frac{C_1 + C_2}{2}, \tag{3.1}$$

где C_1 и C_2 – цвета, которые графическое устройство способно непосредственно воспроизвести для каждого пикселя. Числовые значения C, C_1 и C_2 можно рассчитать в полутоновых градациях или в модели *RGB* – отдельно для каждой компоненты. Чаще используют квадратные ячейки больших размеров. Дадим пример ячеек размером 2 × 2. Такие ячейки дают 5 градаций, из них три комбинации (1, 2, 3) образуют новые оттенки.

Расчет цвета, соответствующего одной из комбинаций пикселей в ячейке, можно выполнить следующим образом. Если пиксели ячейки могут быть только

двух цветов (C_1 и C_2), то необходимо подсчитать часть площади ячейки для пикселей каждого цвета. Цвет ячейки C можно оценить соотношением

$$C = \frac{S_1 C_1 + (S - S_1) C_2}{S} = \frac{S_1 C_1 + S_2 C_2}{S}, \qquad (3.2)$$

где S – общая площадь ячейки;

S_1 и S_2 – площади, занятые пикселями цветов C_1 и C_2, причем $S_1 + S_2 = S$.

Проще всего, когда пиксели квадратные, а их размер равен шагу размещения пикселей. Примем площадь одного пикселя за единицу. В этом случае площадь, занимаемая пикселями в ячейке, равна их количеству.

Для ячейки 5 × 5, дадим расчет цвета C для некоторых цветов C_1 и C_2. Пусть C_1 – белый цвет (*RGB*) = (255, 255, 255), а C_2 – черный (*RGB*) = (0, 0, 0), тогда мы получим светло-серый цвет.

Если C_1 – желтый (*RGB*) = (255, 255, 0), а C_2 – красный (*RGB*) = (255, 0, 0), то цветовые координаты итогового цвета в системе *RGB* будут иметь следующие значения C = (255, 204, 0). Это оттенок оранжевого цвета.

Следовательно, если в ячейке размерами $n \times n$ использованы два цвета, то с помощью этой ячейки можно получить $n^2 + 1$ различных цветовых градаций. Две комбинации пикселей – когда все пиксели ячейки имеют цвет C_1 или C_2 – дают цвет ячейки соответственно C_1 или C_2. Все иные комбинации дают оттенки, промежуточные между C_1 и C_2.

Можно считать, что ячейки размером $n \times n$ образуют растр с разрешающей способностью в n раз меньшей, чем у исходного растра, а глубина цвета возрастает пропорционально n^2.

Для характеристики изображений, которые создаются методом дизеринга, используют термин *линиатура растра*. *Линиатура* вычисляется как количество линий (ячеек) растра на единицу длины – сантиметр, миллиметр, дюйм. В последнем случае единицей измерения для линиатуры является *Lpi* (по аналогии с *dpi*).

Практическое задание

Разработать программу закраски объекта № 1 цветом с помощью дизеринга в растре 5 × 5.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 3.1 – Вопросы для защиты

Вопрос	Рейтинг
Дать определение термину «Дизеринг»	1
Сколько градаций серого цвета обеспечивает растр 4 x 4	1
Формула для расчета цвета в системе *RGB*.	1

Дать определение термину «линеатура»	1
Дать определение термину «ЧМ-дизеринг»	1

4 Алгоритмы вычерчивания окружностей

Цель работы: изучение алгоритма Брезенхема для вычерчивания окружностей.

В растр нужно разлагать не только линейные, но и другие, более сложные функции. Один из наиболее эффективных и простых для понимания алгоритмов генерации окружности принадлежит Брезенхему. Для начала заметим, что необходимо сгенерировать только одну четвертую часть окружности. Остальные ее части могут быть получены последовательными отражениями. Первый квадрант отражается относительно прямой $x = 0$ для получения соответствующей части окружности во втором квадранте. Верхняя полуокружность отражается относительно прямой $y = 0$ для завершения построения.

Для вывода алгоритма рассмотрим первую четверть окружности с центром в начале координат. Заметим, что если работа алгоритма начинается в точке $x = 0$, $y = R$, то при генерации окружности по часовой стрелке в первом квадранте y является монотонно убывающей функцией аргументам (рисунок 4.1).

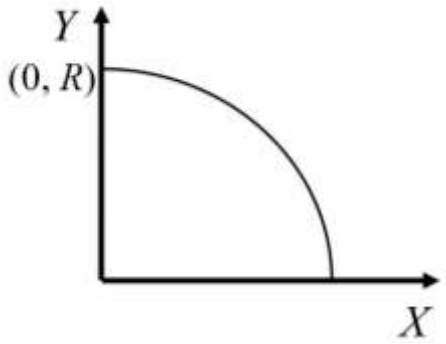


Рисунок 4.1 – Окружность в первом квадранте

Аналогично, если исходной точкой является $y = 0$, $x = R$, то при генерации окружности против часовой стрелки x будет монотонно убывающей функцией аргумента y. В нашем случае выбирается генерация по часовой стрелке с началом в точке $x = 0$, $y = R$. Предполагается, что центр окружности и начальная точка находятся точно в точках растра.

Для любой заданной точки на окружности при генерации по часовой стрелке существует только три возможности выбрать следующий пиксель, наилучшим образом приближающий окружность: горизонтально вправо, по диагонали вниз и вправо, вертикально вниз. На рисунке 4.2 эти направления обозначены m_H, m_D, m_V соответственно. Алгоритм выбирает пиксель, для которого минимален квадрат расстояния между одним из этих пикселей и окружностью, т. е. минимум из

$$m_H = |(x_i + 1)^2 + y_i^2| - R^2,$$
$$m_D = |(x_i + 1)^2 + (y_i - 1)^2| - R^2, \quad (4.1)$$
$$m_V = |(x_i^2 + (y_i - 1)^2| - R^2.$$

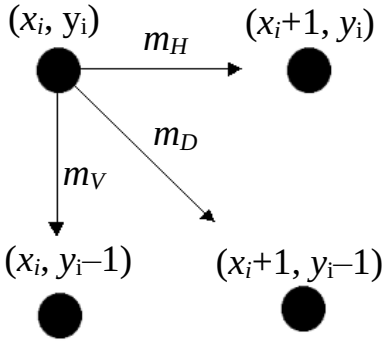


Рисунок 4.2 – Выбор пикселей в первом квадранте

Вычисления можно упростить, если заметить, что в окрестности точки (x_i, y_i) возможны только пять типов пересечений окружности и сетки растра, приведенных на рисунке 4.3.

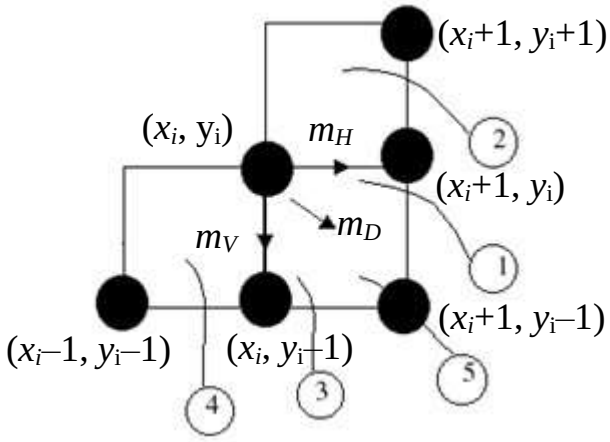


Рисунок 4.3 – Пересечение окружности и сетки растра

Разность между квадратами расстояний от центра окружности до диагонального пикселя $(x_i + 1, y_i - 1)$ и от центра до точки на окружности R^2 равна

$$D_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2. \quad (4.2)$$

В алгоритме Брезенхема желательно использовать только знак ошибки, а не ее величину.

При $D_i < 0$ диагональная точка $(x_i + 1, y_i - 1)$ находится внутри реальной окружности, т. е. это случаи 1 или 2 на рисунке 4.3. Ясно, что в этой ситуации следует выбрать либо пиксель $(x_i + 1, y_i)$, т. е. m_H, либо пиксель $(x_i + 1, y_i - 1)$, т. е. m_D. Для этого сначала рассмотрим случай 1 и проверим разность квадратов расстояний от окружности до пикселей в горизонтальном и диагональном направлениях:

$$d = |(x_i + 1)^2 + y_i^2 - R^2| - |(x_i + 1)^2 + (y_i - 1)^2 - R^2|. \quad (4.3)$$

При $d < 0$ расстояние от окружности до диагонального пикселя больше, чем до горизонтального. Напротив, если $d > 0$, расстояние до горизонтального пикселя больше. Таким образом:

при $d <= 0$ выбираем m_H в $(x_i + 1, y_i - 1)$;

при $d > 0$ выбираем m_D в $(x_i + 1, y_i - 1)$;

при $d = 0$, когда расстояние от окружности до обоих пикселей одинаково, выбираем горизонтальный шаг.

Практическое задание

Составить алгоритм и программу для отображения объекта № 2 методом Брезенхема.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 4.1 – Вопросы для защиты

Вопрос	Рейтинг
Записать формулы, для вычисления расстояния между возможными пикселями и окружностью	1
Привести типы пересечений окружности и сетки растра	1
Как изменяется величина d_i при закрашивании пикселя по m_h	1
Как изменяется величина d_i при закрашивании пикселя по m_V	1
Как изменяется величина d_i при закрашивании пикселя по m_D	1
Пояснить схему алгоритма формирования окружности	3

5 Алгоритмы заполнение многоугольников методом САР

Цель работы: изучение алгоритма с упорядоченным списком ребер.

Простейший способ закраски многоугольника состоит в проверке принадлежности каждой точки этому многоугольнику. Для организации и управления списком активных ребер (САР) можно использовать ряд методов. Сначала отрезки изображения сортируются по наибольшей координате у. В одном из простых методов такой сортировки используются два перемещающихся указателя в отсортированном списке. Указатель начала используется для обозначения начала списка активных ребер, а указатель конца – для обозначения конца этого списка. На рисунке 5.1 представлена сцена из нескольких отрезков с тремя характерными сканирующими строками. Указатель начала в исходном положении устанавливается на начало этого списка, т.е. на отрезок *BC*. Указатель конца установлен на тот последний отрезок в списке,

который начинается выше рассматриваемой сканирующей строки, т. е. на отрезок *BD*. При сканировании изображения необходимо корректировать САР, при этом указатель конца передвигают вниз, чтобы включить в список новые отрезки, начинающиеся на текущей сканирующей строке или выше нее. В то же самое время указатель начала передвигают вниз, чтобы исключить отрезки, кончающиеся выше текущей сканирующей строки. Это изображено на рисунке 5.1 для сканирующих строк, помеченных цифрами 2 и 3.

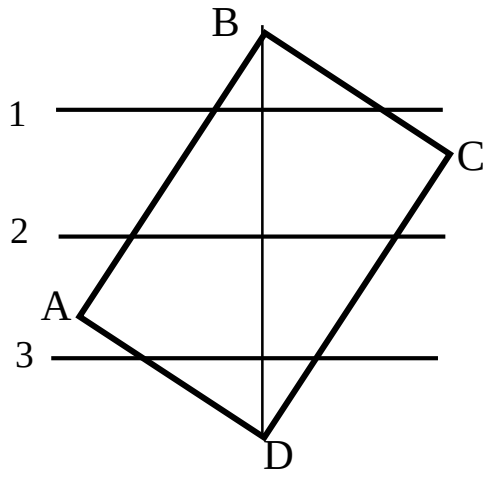


Рисунок 5.1 – Сканирующая строка

Перед сканированием всем ребрам соответствует флаг в значении False. По мере увеличения значения *y* сканирующей строки ведется проверка условия

$$Y_{imin} <= Y_{скан} <= Y_{imax}, \quad (5.1)$$

где Y_{imin}, Y_{imax}, $Y_{скан}$ – минимальная и максимальная координата *i*-го отрезка и координата сканирующей строки.

Если данное условие выполняется, то флагу соответствующего отрезка присваивается значение True и данный отрезок обрабатывается.

Практическое задание

Разработать программу, закрашивающую объект № 1 методом САР.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 5.1 – Вопросы для защиты

Вопрос	Рейтинг
Как определяется наименьшей прямоугольник, содержащий внутри себя многоугольник	1
Дайте определение термину «Активное ребро»	1

Как определить для каждого активного ребра многоугольника точки пересечений со сканирующими строками	1
Пояснить схему алгоритма закраски треугольника методом САР	1
Пояснить схему алгоритма закраски четырехугольника	1
Дать пояснения операторам, используемым в программе	1

6 Алгоритмы заполнение многоугольников методом заливки

Цель работы: изучение алгоритма заполнения с затравкой.

В рассмотренном выше алгоритме заполнение происходит в порядке сканирования. Иной подход используется в алгоритмах заполнения с затравкой. В них предполагается, что известен хотя бы один пиксель из внутренней области многоугольника. Алгоритм пытается найти и закрасить все другие пиксели, принадлежащие внутренней области. Все пиксели на границе области имеют другой цвет. Гранично-определенные области могут быть 4- или 8-связными. Если область 4-связная, то любого пикселя в области можно достичь с помощью комбинации движений только в четырёх направлениях: налево, направо, вверх, вниз. Для 8-связной области пикселя можно достичь с помощью комбинации движений в двух вертикальных, двух горизонтальных и четырёх диагональных направлениях.

Используя стек, можно разработать простой алгоритм заполнения гранично-определенной области. Стек (англ. stack – стопка) – абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in – first out, «последним пришёл – первым вышел»). Когда новые значения добавляются или помещаются в стек, все остальные значения опускаются вниз на один уровень. Когда значения удаляются или извлекаются из стека, остальные значения всплывают или поднимаются вверх на один уровень. Со стеком возможны три операции: добавление элемента (push), удаление элемента (pop) и чтение головного элемента (peek).

Простой алгоритм заполнения с затравкой можно представить в следующем виде:

```
Затравка(x, y) выдает затравочный пиксель
Пиксел(x, y) = Затравка(x, y)          инициализируем стек
Push Пиксел(x, y)
while (стек не пуст)
    Pop Пиксел(x, y) извлекаем пиксел из стека
    if Пиксел(x, y) < > Нов_значение then
        Пиксел(x, y) = Нов_значение
    end if
    проверим, надо ли помещать соседние пикселы в стек
    if (Пиксел(x + 1, y) < > Нов_значение and Пиксел(x + 1, y) < > Гран_значение) then
        Push Пиксел (x + 1, y)
    if (Пиксел(x, y + 1) < > Нов_значение and Пиксел(x, y + 1) < > Гран_значение) then
        Push Пиксел (x, y + 1)
    if (Пиксел(x – 1, y) < > Нов_значение and Пиксел(x – 1, y) < > Гран_значение) then
        Push Пиксел (x - 1, y)
    if (Пиксел(x, y – 1) < > Нов_значение and Пиксел(x, y – 1) < > Гран_значение) then
```

Push Пиксел (x, y – 1) end if
end while.

В алгоритме проверяются и помещаются в стек 4-связные пиксели, начиная с правого от текущего пикселя. Направление обхода пикселей – против часовой стрелки.

Практическое задание

Составить алгоритм и программу для отображения объекта № 2 методом заливки.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 6.1 – Вопросы для защиты

Вопрос	Рейтинг
Какая область называется гранично-определенной	1
В чем различие между 8-связной и 4-связной областью	1
Как определялся затравочный пиксел	1
Для чего используется стек	1
Пояснить схему алгоритма закраски	1

7 Аффинные преобразования

Цель работы: изучение алгоритмов параллельного сдвига, растяжения – сжатия и поворота графического объекта на плоскости и в пространстве.

Любое преобразование объектов представляется как сумма элементарных действий. Рассмотрим частные случаи аффинного преобразования. На рисунках 7.1–7.3 представлены эти действия и соответствующая матрица преобразования.

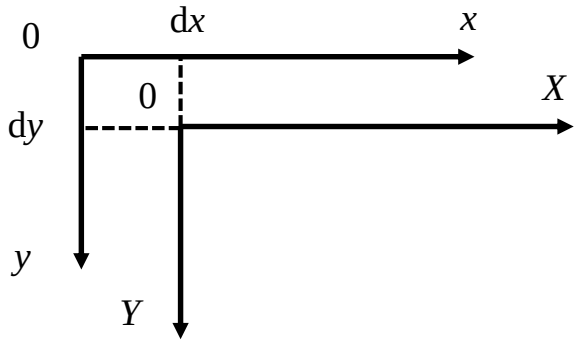


Рисунок 7.1 – Параллельный сдвиг координат

Система уравнений Матрица преобразования

$$\begin{cases} X = x - dx, \\ Y = y - dy. \end{cases}$$

$$\begin{bmatrix} 1 & 0 & -dx \\ 1 & 0 & -dy \\ 0 & 0 & 1 \end{bmatrix}$$

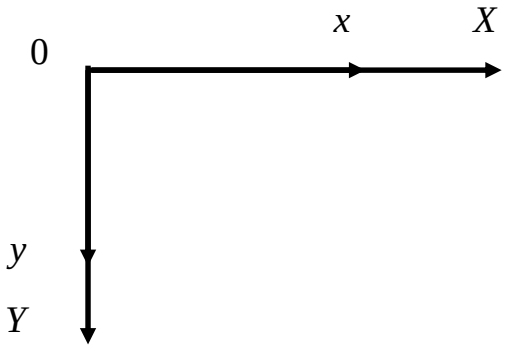


Рисунок 7.2 – Растяжение-сжатие осей координат

Система уравнений

$$\begin{cases} X = x / k_x, \\ Y = y / k_y. \end{cases}$$

Матрица преобразования

$$\begin{bmatrix} 1/k_x & 0 & 0 \\ 1 & 1/k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Коэффициенты k_x и k_y могут быть отрицательными. Например, $k_x = -1$, что соответствует зеркальному отражению относительно оси у.

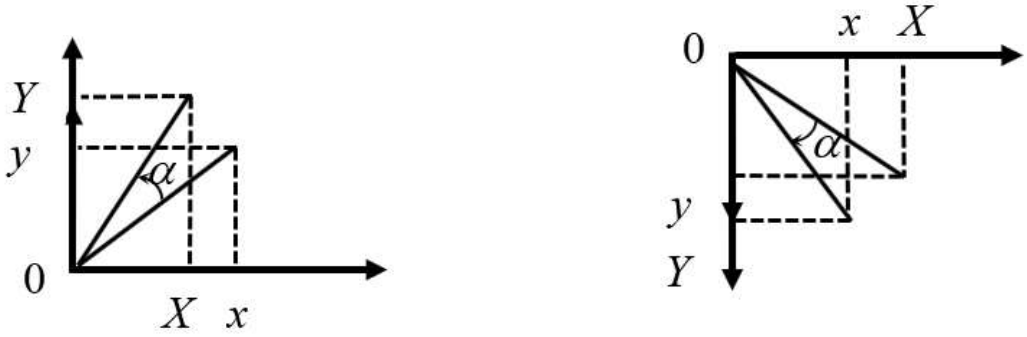


Рисунок 7.3 – Поворот вокруг центра координат

Система уравнений

$$\begin{cases} X = x\cos\alpha + y\sin\alpha, \\ Y = -x\sin\alpha + y\cos\alpha. \end{cases}$$

Матрица преобразования

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Практическое задание.

Составить алгоритм и программу для перемещения, сжатия-растяжения и поворота объекта № 1. При разработке программы умножение матриц производить с помощью соответствующей подпрограммы, без использования

функций листа.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 7 – Вопросы для защиты

Вопросы	Рейтинг
Какие преобразования координат называют «аффинными»	1
Записать линейные преобразования в матричной форме	1
Привести формулу для вычисления элементов матрицы C=A*B	1
Привести формулы параллельного сдвига координат точки	1
Привести формулы растяжение-сжатие координат точки	1
Привести формулы для поворота координат точки	1
Перечислить свойства аффинного преобразования	1
Пояснить схему алгоритма преобразования координат в плоскости	2

8 Метод Робертса

Цель работы: изучение алгоритма удаления невидимых граней методом Робертса.

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Это математически элегантный метод, работающий в объектном пространстве. Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом.

Пусть *F* – некоторая грань многогранника. Плоскость, несущая эту грань, разделяет пространство на два подпространства. Назовем положительным то из них, в которое смотрит внешняя нормаль к грани. Если точка наблюдения – в положительном подпространстве, то грань – лицевая, в противном случае – нелицевая. Если многогранник выпуклый, то удаление всех нелицевых граней полностью решает задачу визуализации с удалением невидимых граней.

Для определения, лежит ли точка в положительном подпространстве, используют проверку знака скалярного произведения $\left(\vec{l},\vec{n}\right)$, где $\vec{l}$ – вектор, направленный к наблюдателю, фактически определяет точку наблюдения; $\vec{n}$ – вектор внешней нормали грани. Если $\left(\vec{l},\vec{n}\right) > 0$, т. е. угол между векторами острый, то грань является лицевой. Если $\left(\vec{l},\vec{n}\right) < 0$, т. е. угол между векторами тупой, то грань является нелицевой.

В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми. Невыпуклые тела должны быть разбиты на выпуклые части. В этом алгоритме выпуклое многогранное тело с плоскими гранями должно представиться набором пересекающихся плоскостей. Уравнение произвольной плоскости в трехмерном пространстве имеет вид

$$a \cdot x + b \cdot y + c \cdot z + d = 0. \qquad (8.1)$$

В матричной форме этот результат выглядит так:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} P \end{bmatrix}^T = 0, \qquad (8.2)$$

где $[P]^T = [a\ b\ c\ d]$ представляет собой плоскость. Поэтому любое выпуклое твердое тело можно выразить матрицей тела, состоящей из коэффициентов уравнений плоскостей, т. е.

$$\begin{bmatrix} V \end{bmatrix} = \begin{bmatrix} a_1 a_2 \ldots a_n \\ b_1 b_2 \ldots b_n \\ c_1 c_2 \ldots c_n \\ d_1 d_2 \ldots d_n \end{bmatrix}, \qquad (8.3)$$

где каждый столбец содержит коэффициенты одной плоскости.

Напомним, что любая точка пространства представима в однородных координатах вектором $[S] = [x\ y\ z\ 1]$. Более того, если точка $[S]$ лежит на плоскости, то $[S] \cdot [P]^T = 0$. Если же $[S]$ не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. В алгоритме Робертса предполагается, что точки, лежащие внутри тела, дают отрицательное скалярное произведение, т. е. нормали направлены наружу. Чтобы проиллюстрировать эти идеи, рассмотрим следующий пример. Рассмотрим единичный куб с центром в начале координат (рисунок 8.1).

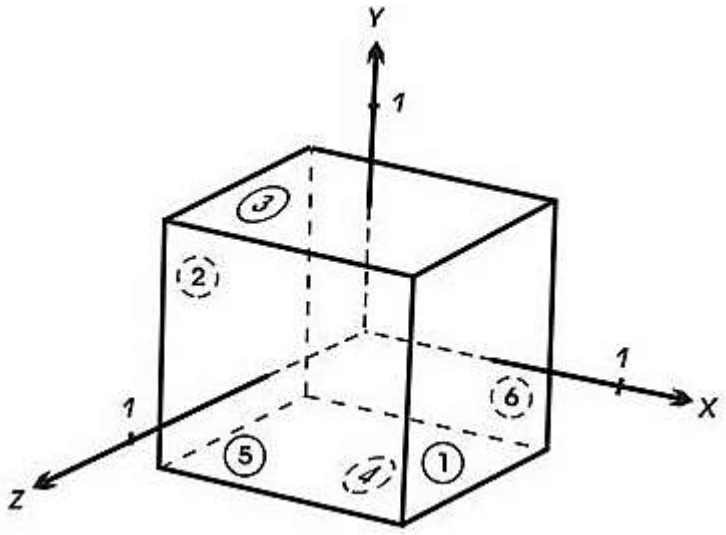


Рисунок 8.1 – Куб с центром в начале координат

Шесть плоскостей, описывающих данный куб, таковы:

$x_1 = 1/2$, $x_2 = -1/2$, $y_3 = 1/2$, $y_4 = -1/2$, $z_5 = 1/2$, $z_6 = -1/2$.

Более подробно уравнение правой плоскости можно записать как

$x_1 + 0 \times y_1 + 0 \times z_1 - (1/2) = 0$ или $2x_1 - 1 = 0$.

Полная матрица тела такова:

$$[V] = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ -1/2 & 1/2 & -1/2 & 1/2 & -1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 \\ -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}. \quad (8.4)$$

Экспериментально проверим матрицу тела с помощью точки, о которой точно известно, что она лежит внутри тела. Если знак скалярного произведения для какой-нибудь плоскости больше нуля, то соответствующее уравнение плоскости следует умножить на –1. Для проверки возьмем точку внутри куба с координатами $x = 1/4$, $y = 1/4$, $z = 1/4$. В однородных координатах эта точка представляется в виде вектора

$$[S] = [1/4\ 1/4\ 1/4\ 1] = [1\ 1\ 1\ 4].$$

Скалярное произведение этого вектора на матрицу объема равно

$$[S][V] = [1 \quad 1 \quad 1 \quad 4] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ -1/2 & 1/2 & -1/2 & 1/2 & -1/2 & 1/2 \end{bmatrix} =$$

$$= [-2 \quad 6 \quad -2 \quad 6 \quad -2 \quad 6].$$

Здесь результаты для второго, четвертого и шестого уравнения плоскостей (столбцов) положительны и, следовательно, составлены некорректно. Умножая эти уравнения (столбцы) на –1, получаем корректную матрицу тела для куба:

$$[V] = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}.$$

В приведенном примере корректность уравнений плоскостей была проверена экспериментально. Разумеется, это не всегда возможно. Существует несколько полезных методов для более общего случая. Хотя уравнение плоскости содержит четыре неизвестных коэффициента, его можно нормировать так, чтобы $d = 1$. Следовательно, трех неколлинеарных точек достаточно для определения этих коэффициентов. Нормированное уравнение в матричной форме при подстановке координат трех неколлинеарных точек (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) имеет следующий вид:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \text{ или } [X] \cdot [C] = [D]. \qquad (8.5)$$

Решение этого уравнения дает значения коэффициентов уравнения плоскости:

$$[C] = [X]^{-1}[D]. \qquad (8.6)$$

Другой способ используется, если известен вектор нормали к плоскости, т. е. $n = a\bar{i} + b \cdot \bar{j} + c \cdot \bar{k}$, где i, j, k – единичные векторы осей x, y, z соответственно. Тогда уравнение плоскости примет вид $ax + by + cz + d = 0$.

Перед началом работы алгоритма удаления невидимых линий или поверхностей для получения желаемого вида сцены часто применяется трехмерное видовое преобразование. Матрицы тел для объектов преобразованной сцены можно получить преобразованием исходных матриц тел.

Если $[T]$ – матрица размером 4 x 4 видового преобразования, то $[VT] = [T]^{-1}[V]$.

Итак, преобразованная матрица тела получается умножением исходной матрицы тела слева на обратную матрицу видового преобразования.

Тот факт, что плоскости имеют бесконечную протяженность и что скалярное произведение точки (вектора к точке) на матрицу тела положительно, если точка лежит вне этого тела, позволяет предложить метод, в котором матрица тела используется для определения граней, которые экранируются самим этим телом.

Положительное скалярное произведение дает только такая плоскость (столбец) в матрице тела, относительно которой точка лежит снаружи, т. е. в положительном подпространстве.

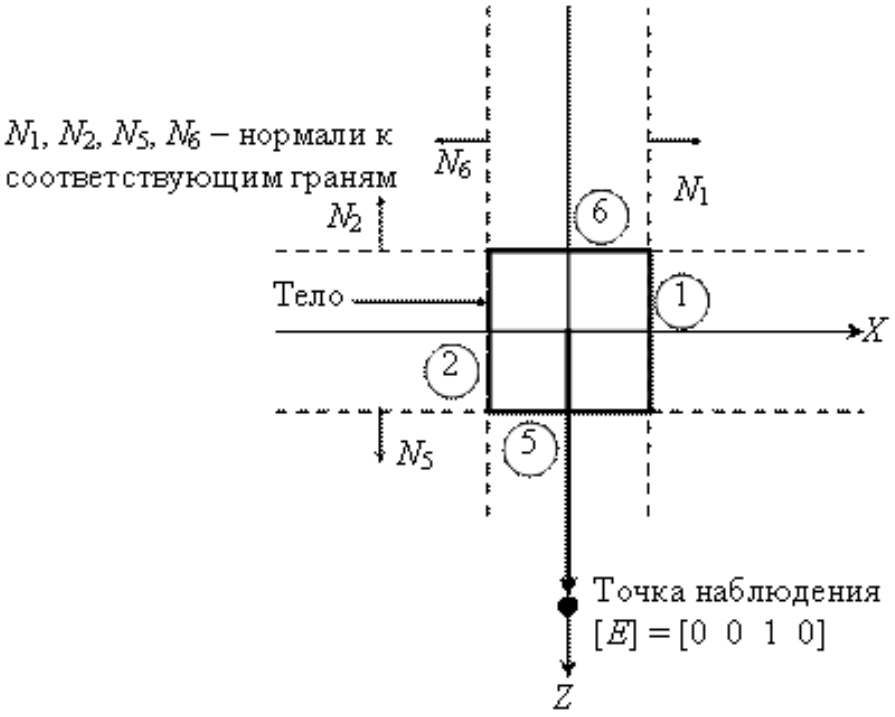


Рисунок 8.2 – Точка наблюдения вне тела

Условие $[E]\cdot[V] < 0$ определяет, что плоскости – нелицевые, а их грани – задние. Заметим, что для аксонометрических проекций (точка наблюдения в бесконечности) это эквивалентно поиску положительных значений в третьей строке матрицы тела. Найдем произведение $[E]\cdot[V]$ для нашего примера (см. рисунок 8.2):

$$[E]\cdot[V] = [0 \quad 0 \quad 1 \quad 0]\cdot\begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} = [0 \quad 0 \quad 0 \quad 0 \quad 2 \quad -2]$$

Отрицательное число в шестом столбце показывает, что грань с этим номером нелицевая. Положительное число в пятом столбце показывает, что грань лицевая. Нулевые результаты соответствуют плоскостям, параллельным направлению взгляда.

Положительный результат вектора E и вектора нормали можно интерпретировать как острый угол между этими векторами, отрицательный результат – как тупой угол. Если угол между векторами острый, то грань является лицевой; если угол тупой, то грань – нелицевая.

Практическое задание

Составить алгоритм и программу для поворота объекта № 3 вокруг

вертикальной оси 0*Z*. При разработке программы умножение матриц производить с помощью соответствующей подпрограммы, без использования функций листа.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 8.1 – Вопросы для защиты

Вопрос	Рейтинг
Привести уравнение плоскости в трехмерном пространстве в алгебраическом и матричном виде	1
Привести пример и пояснить создание матрицы выпуклого тела	1
Как определяется, с какой стороны плоскости находится точка с координатами X_0, Y_0, Z_0	1
Привести формулы видового преобразования тела	1
Пояснить схему алгоритма Робертса	2
Пояснить схему алгоритма поворота объекта № 3 вокруг оси 0*Z*	2

9 Метод *Z*-буфера

Цель работы: изучение алгоритма удаления невидимых граней методом *Z*-буфера.

Алгоритм, использующий *Z*-буфер, это один из простейших алгоритмов удаления невидимых поверхностей. Работает этот алгоритм в пространстве изображения. Идея *Z*-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения, *Z*-буфер – это отдельный буфер глубины, используемый для запоминания координаты *z* или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение *z* каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в *Z*-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка *Z*-буфера новым значением *z*. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по x и y наибольшего значения функции $z(x, y)$.

Формальное описание алгоритма *Z*-буфера таково:

1) заполнить буфер кадра фоновым значением интенсивности или цвета;

2) заполнить *Z*-буфер минимальным значением *z*;

3) преобразовать каждый многоугольник в растровую форму в произвольном порядке;

4) для каждого Пиксел(x, y) в многоугольнике вычислить его глубину $z(x, y)$;

5) сравнить глубину $z(x, y)$ со значением *Z*-буфер(x, y), хранящимся в

Z-буфере в этой же позиции;

6) если $z(x, y) > Z$-буфер(x, y), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить Z-буфер(x, y) на $z(x, y)$. В противном случае никаких действий не производить.

Если известно уравнение плоскости, несущей каждый многоугольник, то вычисление глубины каждого пиксела на сканирующей строке можно проделать пошаговым способом. Грань при этом рисуется последовательно (строка за строкой). Для нахождения необходимых значений используется линейная интерполяция (рисунок 9.1).

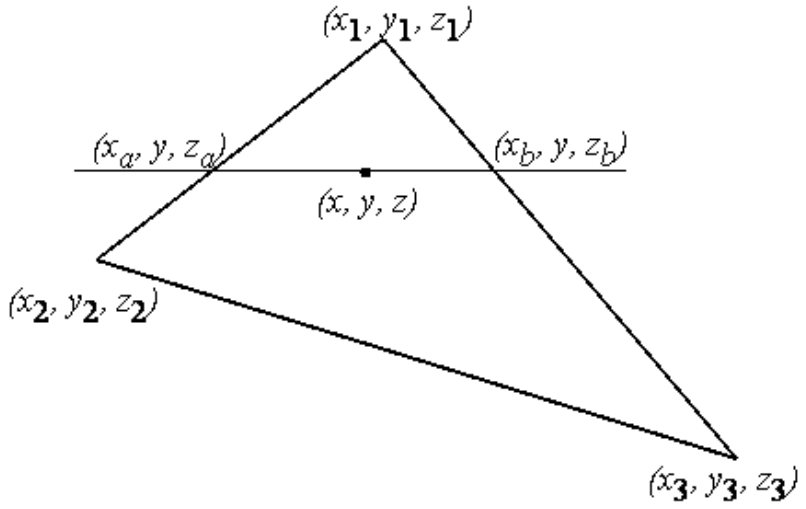


Рисунок 9.1 – Сканирующая строка по грани

Для рисунка y меняется от y_1 до y_2 и далее до y_3, при этом для каждой строки определяется x_a, z_a, x_b, z_b:

$$x_a = x_1 + (x_2 - x_1) \cdot \frac{y - y_1}{y_2 - y_1}, \qquad x_b = x_1 + (x_3 - x_1) \cdot \frac{y - y_1}{y_3 - y_1}, \tag{9.1}$$

$$z_a = z_1 + (z_2 - z_1) \cdot \frac{y - y_1}{y_2 - y_1}, \quad z_b = z_1 + (z_3 - z_1) \cdot \frac{y - y_1}{y_3 - y_1}. \tag{9.2}$$

На сканирующей строке x меняется от x_a до x_b и для каждой точки строки определяется глубина z:

$$za = za + (zb - za) \cdot \frac{x - xa}{xb - xa}. \tag{9.3}$$

Реализация алгоритма вдоль сканирующей строки позволяет совместить алгоритм Z-буфера с алгоритмами растровой развертки ребер и алгоритмами закраски грани.

Практическое задание

Составить алгоритм и программу для визуализации сцены пересечения объекта № 3 и куба.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 9 – Вопросы для защиты

Вопрос	Рейтинг
Какая информация содержится в массиве листа «*Z*-буфер»	1
Привести формулу для определения глубины $z(x, y)$	1
Какая информация содержится в массиве листа «кадр»	1
Пояснить схему алгоритма для визуализации сцены пересечения объекта № 3 и куба	2

10 Модели отражения света

Цель работы: изучение алгоритма изображения поверхности согласно интенсивности отраженного света при учете взаимного расположения поверхности, источника света и наблюдателя.

В компьютерной графике для расчета освещенности граней объектов применяется цветовая модель RGB. Интенсивность отраженного света точек объектов вычисляют отдельно для каждой их трех составляющих цветовых компонент, а затем объединяют в результирующую тройку цветов.

При расчете освещенности граней применяют следующие типы освещения и отражения света от поверхностей:

- рассеянное;
- диффузное;
- зеркальное.

Матовые поверхности обладают свойством диффузного отражения, т. е. равномерного по всем направлениям рассеивания света. Поэтому кажется, что поверхности имеют одинаковую яркость независимо от угла обзора. Для таких поверхностей справедлив закон косинусов Ламберта, устанавливающий соответствие между количеством отраженного света и косинусом угла θ между направлением на точечный источник света интенсивности I_p и нормалью к поверхности (рисунок 10.1). При этом количество отраженного света не зависит от положения наблюдателя.

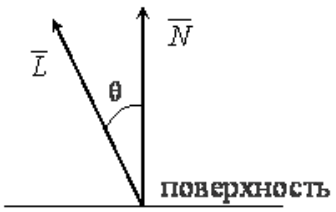


Рисунок 10.1 – Падающий свет и нормаль к поверхности

Освещенность рассеянным светом вычисляется по формуле

$$I_{\mathrm{d}} = I_{\mathrm{p}} \cdot k_{\mathrm{d}} \cdot \cos\theta. \tag{10.1}$$

Значение коэффициента диффузного отражения k_{d} является константой в диапазоне (0, 1) и зависит от материала. Если векторы $\overline{L}$ и $\overline{N}$ нормированы, то, используя скалярное произведение, формулу освещенности можно записать так:

$$I_{\mathrm{d}} = I_{\mathrm{p}} \cdot k_{\mathrm{d}} \cdot (\overline{L} \times \overline{N}). \tag{10.2}$$

Предметы, освещенные одним точечным источником света, выглядят контрастными. Этот эффект аналогичен тому, который можно наблюдать, когда предмет, помещенный в темную комнату, виден при свете направленной на него фотовспышки. В данной ситуации, в отличие от большинства реальных визуальных сцен, отсутствует рассеянный свет, под которым здесь понимается свет постоянной яркости, созданный многочисленными отражениями от различных поверхностей. Такой свет практически всегда присутствует в реальной обстановке. Даже если предмет защищен от прямых лучей, исходящих от точечного источника света, он все равно будет виден из-за наличия рассеянного света. Учитывая это, формулу окраски можно записать так:

$$I_{\mathrm{d}} = I_{\mathrm{a}} \cdot k_{\mathrm{a}} + I_{\mathrm{p}} \cdot k_{\mathrm{d}} \cdot (\overline{L} \times \overline{N}). \tag{10.3}$$

Рассеянный свет представлен членом I_a и k_a определяет количество рассеянного света, которое отражается от поверхности предмета.

Точечный источник света удобнее всего расположить в позиции, совпадающей с глазом наблюдателя. Тени в этом случае отсутствуют, а лучи света, падающие на поверхность, окажутся параллельными. Однако теперь, если две поверхности одного цвета параллельны друг другу и их изображения перекрываются, нормали к поверхностям совпадают и, следовательно, поверхности закрашиваются одинаково, различить их невозможно. Этот эффект можно устранить, если учесть, что энергия падающего света убывает пропорционально квадрату расстояния, которое свет проходит от источника до поверхности и обратно к глазу наблюдателя. Обозначая это расстояние за *R*,

запишем:

$$I_d = I_a \cdot k_a + I_p \cdot k_d \cdot (\overline{L} \times \overline{N}) / R^2. \quad (10.4)$$

Однако данным правилом на практике трудно воспользоваться. Для параллельной проекции, когда источник света находится в бесконечности, расстояние R также становится бесконечным. Даже в случае центральной проекции величина $1/R^2$ может принимать значения в широком диапазоне, поскольку точка зрения часто оказывается достаточно близкой к предмету. В результате закраска поверхностей, которые имеют одинаковые углы θ между $\overline{L}$ и $\overline{N}$, будет существенно различаться. Большей реалистичности можно достичь, если заменить R^2 на $r + k$, где k – некоторая константа, а r – расстояние от центра проекции до поверхности:

$$I_d = I_a \cdot k_a + I_p \cdot k_d \cdot (\overline{L} \times \overline{N}) / (r + k)^2. \quad (10.5)$$

Зеркальное отражение можно получить от любой блестящей поверхности. Угол между нормалью и падающим лучом θ равен углу между нормалью и отраженным лучом. Падающий луч, отраженный, и нормаль располагаются в одной плоскости (рисунок 10.2).

Поверхность считается идеально зеркальной, если на ней отсутствуют какие-либо неровности, шероховатости. Собственный цвет у такой поверхности не наблюдается. Световая энергия падающего луча отражается только по линии отраженного луча. Какое-либо рассеяние в стороны от этой линии отсутствует. В природе, вероятно, нет идеально гладких поверхностей, поэтому полагают, что если глубина шероховатостей существенно меньше длины волны излучения, то рассеивания не наблюдается. Для видимого спектра можно принять, что глубина шероховатостей поверхности зеркала должна быть существенно меньше 0,5 мкм.

Рисунок 10.2 – Зеркальное отражение

Если поверхность зеркала отполирована неидеально, то наблюдается зависимость интенсивности отраженного света от длины волны – чем больше

длина волны, тем лучше отражение. Например, красные лучи отражаются сильнее, чем синие. При наличии шероховатостей имеется зависимость интенсивности отраженного света от угла падения. Отражение света максимально для углов, близких к 90°.

Падающий луч, попадая на слегка шероховатую поверхность реального зеркала, порождает несколько лучей, рассеиваемых по различным направлениям. Зона рассеивания зависит от качества полировки и может быть описана некоторым законом распределения. Как правило, форма зоны рассеивания симметрична относительно линии идеального зеркально отраженного луча. К числу простейших, но достаточно часто используемых, относится эмпирическая модель распределения Фонга, согласно которой интенсивность зеркально отраженного излучения пропорциональна $\cos^p \alpha$, где α – угол отклонения от линии идеально отраженного луча. Показатель p находится в диапазоне от 1 до 200, зависит от качества полировки и вычисляется по формуле

$$I_s = I_p \cdot k_s \cdot \cos^p \alpha, \qquad (10.6)$$

где I_p – интенсивность излучения источника,

k_s – коэффициент пропорциональности, который изменяется от 0 до 1.

Практическое задание

Составить алгоритм и программу для изображения поверхности согласно интенсивности отраженного света при учете взаимного расположения поверхности, источника света и наблюдателя при пересечении объекта № 3 и куба.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 10.1 – Вопросы для защиты

Вопрос	Рейтинг
Перечислите модели освещения	1
Привести формулу эмпирической модели распределения Фонга для зеркального отражения	1
Привести формулу закона Ламберта для диффузного отражения	1
Привести формулу, для учета зеркального и диффузного отражения	1
Привести формулу, для вычисления нормали к грани в пространстве	1
Привести формулу, для определения косинуса угла между вектором нормали направлением на источник света	1
Привести формулу, для определения косинуса угла между отраженным лучом и направлением камеры	1
Пояснить схему алгоритма для визуализации сцены пересечения объекта № 3 и	2

куба	

11 Модель отражения света Гуро

Цель работы: изучение алгоритма изображения поверхности согласно интенсивности отраженного света с помощью модели Гуро.

Метод закраски, который основан на интерполяции интенсивности и известен как метод Гуро (по имени его разработчика), позволяет устранить дискретность изменения интенсивности. Процесс закраски по методу Гуро осуществляется в четыре этапа.

1 Вычисляются нормали ко всем полигонам.

2 Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит вершина (рисунок 11.1).

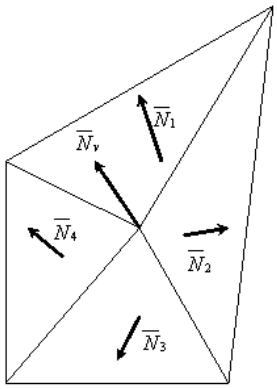


Рисунок 11.1– Нормали к вершинам

3 Используя нормали в вершинах и применяя произвольный метод закраски, вычисляются значения интенсивности в вершинах:

$$\overline{N}_v = (\overline{N}_1 + \overline{N}_2 + \overline{N}_3 + \overline{N}_4) / 4 \qquad (11.1)$$

4 Каждый многоугольник закрашивается путем линейной интерполяции значений интенсивностей в вершинах (11.1) сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки (рисунок 11.2).

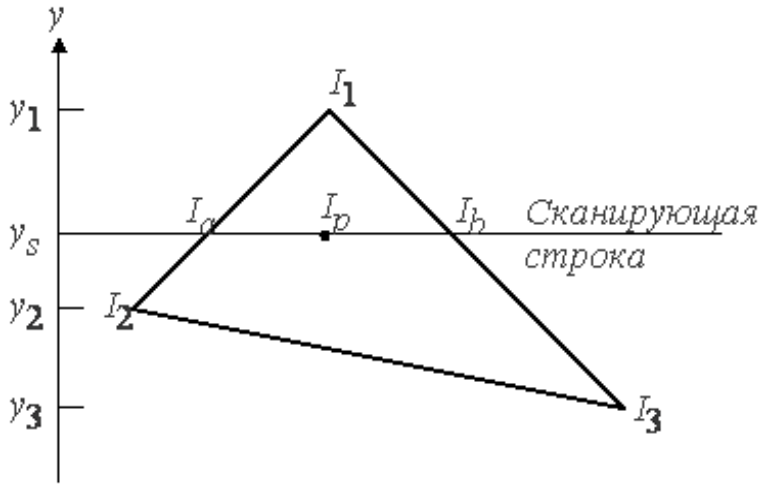


Рисунок 11.2 – Интерполяция интенсивностей

Интерполяция вдоль ребер легко объединяется с алгоритмом удаления скрытых поверхностей, построенным на принципе построчного сканирования. Для всех ребер запоминается начальная интенсивность, а также изменение интенсивности при каждом единичном шаге по координате y. Заполнение видимого интервала на сканирующей строке производится путем интерполяции между значениями интенсивности на двух ребрах, ограничивающих интервал (рисунок 11.2).

$$
\begin{aligned}
I_a &= I_1 \cdot \frac{y_s - y_2}{y_1 - y_2} + I_{2\cdot} \frac{y_1 - y_s}{y_1 - y_2}; \\
I_b &= I_1 \cdot \frac{y_s - y_3}{y_1 - y_3} + I_{2\cdot} \frac{y_1 - y_s}{y_1 - y_3}; \\
I_p &= I_a \cdot \frac{x_a - x_p}{x_b - x_a} + I_{b\cdot} \frac{x_p - x_a}{x_b - x_a}.
\end{aligned}
\qquad (11.1)
$$

Практическое задание

Составить алгоритм и программу для изображения поверхности согласно интенсивности отраженного света при учете взаимного расположения поверхности, источника света и наблюдателя объекта № 3 с использованием алгоритма Гуро.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 11.1 – Вопросы для защиты

Вопрос	Рейтинг
Перечислите этапы закрашивания грани по методу Гуро	3
Привести формулу Вектора нормали в вершине по методу Гуро	1
Привести формулу для интерполирования интенсивности по методу Гуро	2

12 Модель отражения света Фонга

Цель работы: изучение алгоритма изображения поверхности согласно интенсивности отраженного света с помощью модели Фонга.

В методе закраски Фонга используется интерполяция вектора нормали к поверхности вдоль видимого интервала на сканирующей строке внутри многоугольника, а не интерполяция интенсивности. Интерполяция выполняется между начальной и конечной нормалями, которые сами тоже являются результатами интерполяции вдоль ребер многоугольника между нормалями в вершинах. Нормали в вершинах, в свою очередь, вычисляются так же, как в методе закраски, построенном на основе интерполяции интенсивности.

В каждом пикселе вдоль сканирующей строки новое значение интенсивности вычисляется с помощью любой модели закраски. Заметные улучшения по сравнению с интерполяцией интенсивности наблюдаются в случае использования модели с учетом зеркального отражения, т. к. при этом более точно воспроизводятся световые блики. Однако даже если зеркальное отражение не используется, интерполяция векторов нормали приводит к более качественным результатам, чем интерполяция интенсивности, поскольку аппроксимация нормали в этом случае осуществляется в каждой точке.

Практическое задание

Составить алгоритм и программу для изображения поверхности с учетом интенсивности отраженного света, взаимного расположения поверхности, источника света и наблюдателя объекта № 3 с использованием алгоритма Фонга.

Содержание отчета: титульный лист, цель работы, программа, блок-схема в соответствии с ГОСТ 19.701-90, изображение, построенное программой, краткие ответы на вопросы, приведенные в таблице.

Таблица 12.1 – Вопросы для защиты

Вопрос	Рейтинг
Перечислите этапы закрашивания грани по методу Фонга.	3
Привести формулу для интерполирования векторов нормалей по методу Фонга.	2

Список литературы

1 **Пореев, В. Н.** Компьютерная графика / В. Н. Пореев. – Санкт-Петербург: БХВ-Петербург, 2002. – 432 с.: ил.

2 **Сиденко, Л. А.** Компьютерная графика и геометрическое моделирование / Л. А. Сиденко. – Санкт-Петербург: Питер, 2012. – 224 с.

3 **Дегтярев В. М.** Компьютерная геометрия и графика: учебник / В. М. Дегтярев. – 2-е изд., стер. – Москва: Академия, 2013. – 192 с.

4 **Хейфец, А. Л.** Инженерная 3D-компьютерная графика: учеб. пособие для

бакалавров / А. Л. Хейфец, А. Н. Логиновский, И. В. Буторина ; под ред. А. Л. Хейфеца. – 2-е изд., перераб. и доп. – Москва: Юрайт, 2012. – 464 с.

Приложение А (рекомендуемое)

Графические объекты №1, 2, 3 выбираются в соответствии с порядковым номером в списке группы, набранном в алфавитном порядке

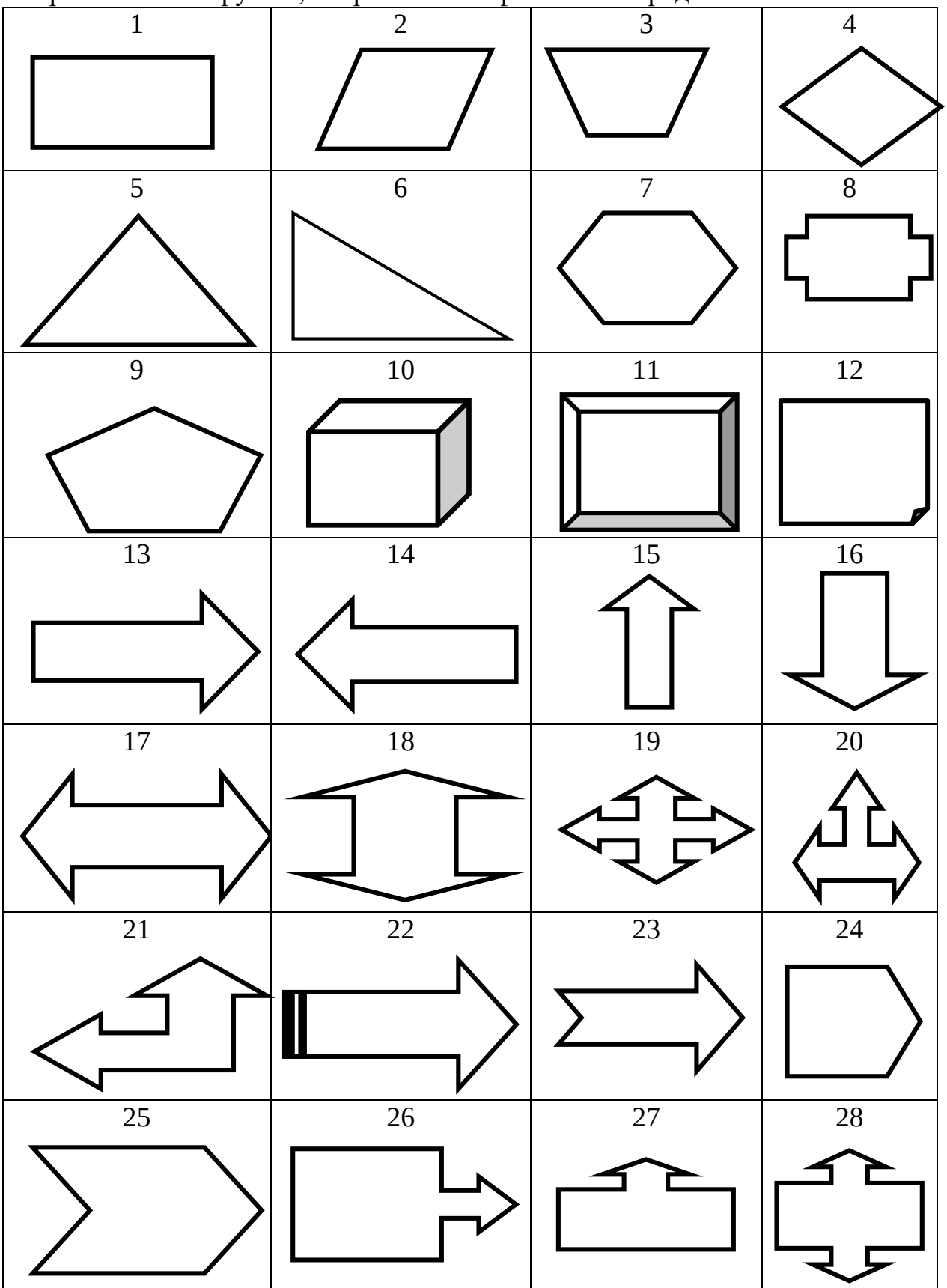


Рисунок А.1 – Графические объекты № 1

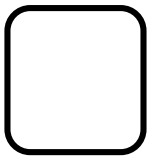
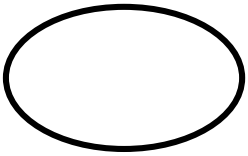
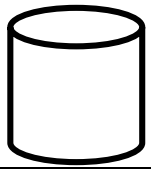
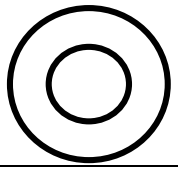
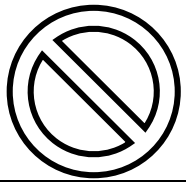
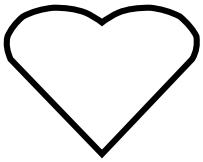
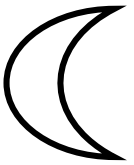
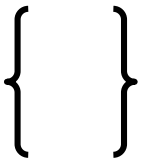
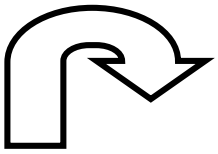
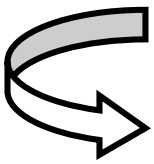
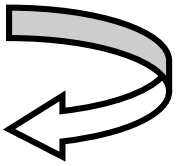
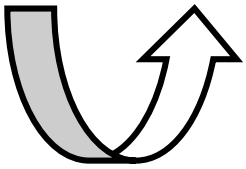
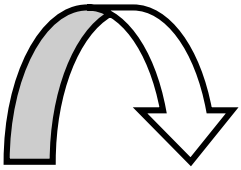
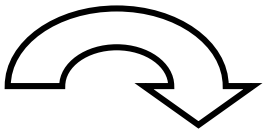
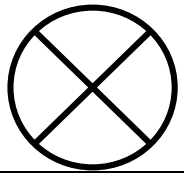
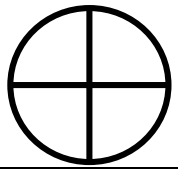
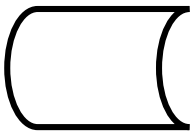
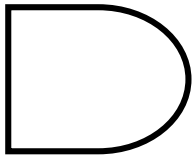
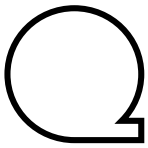
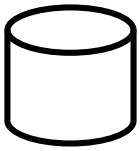
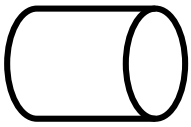
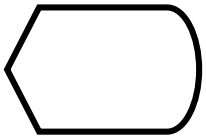
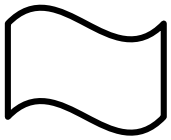
1 	2 	3 	4 
5 	6 	7 	8 
9 	10 	11 	12 
13 	14 	15 	16 
17 	18 	19 	20 
21 	22 	23 	24 
25 	26	27	28

Рисунок А.2 – Графические объекты № 2

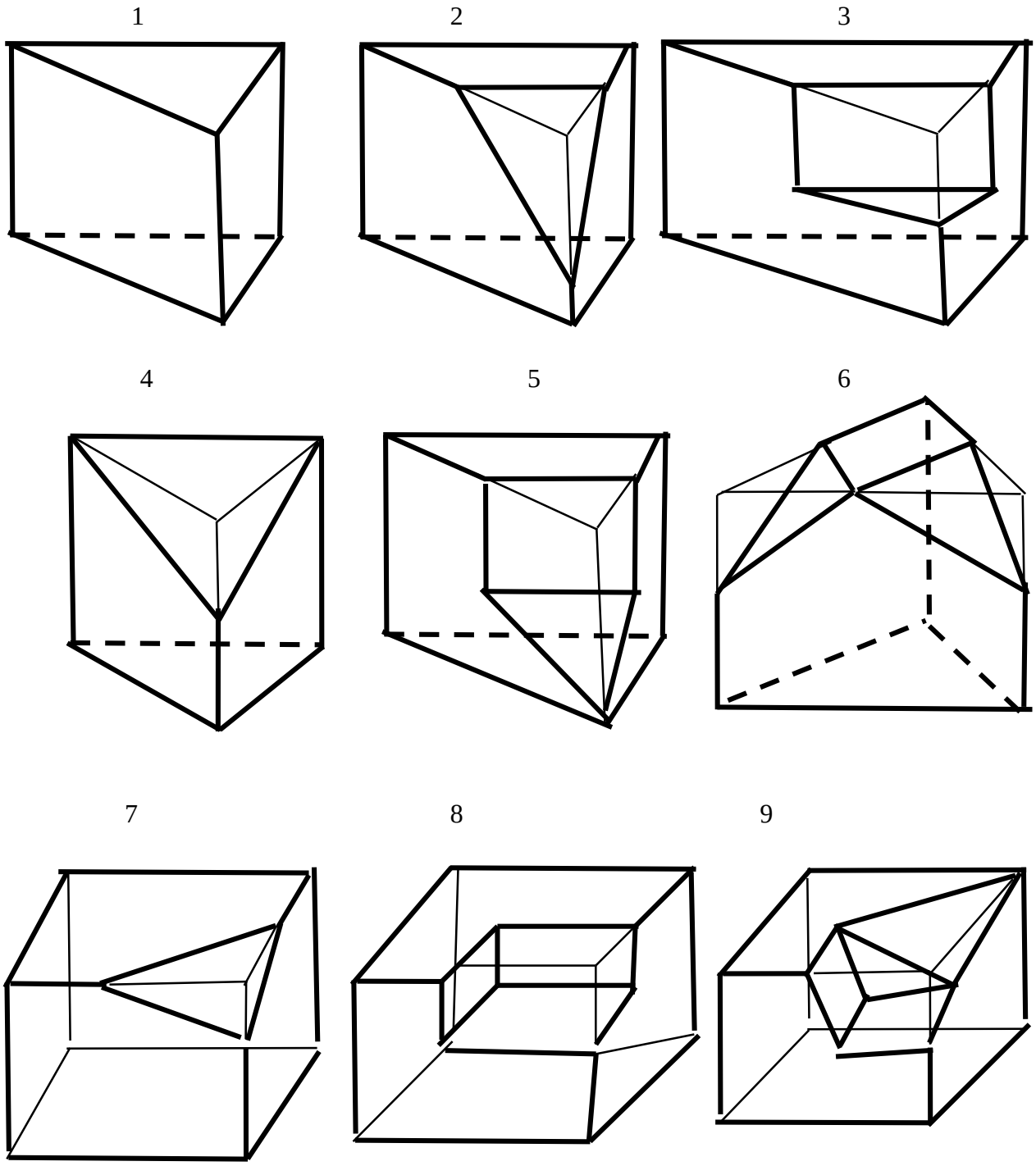
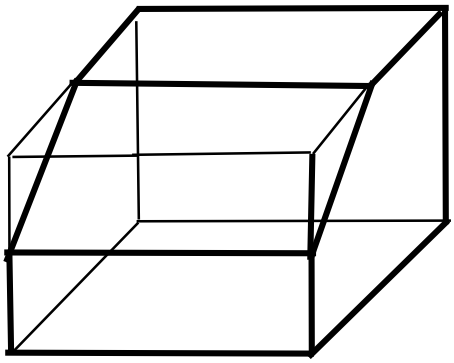
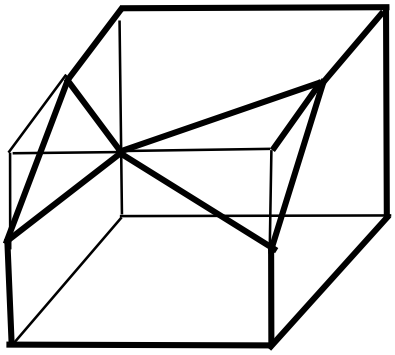


Рисунок А.3 – Графические объекты № 3

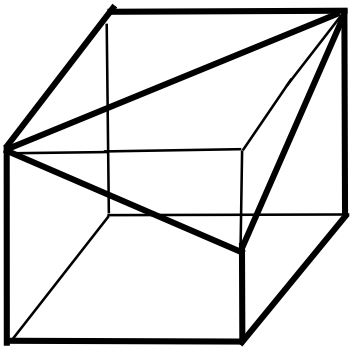
10



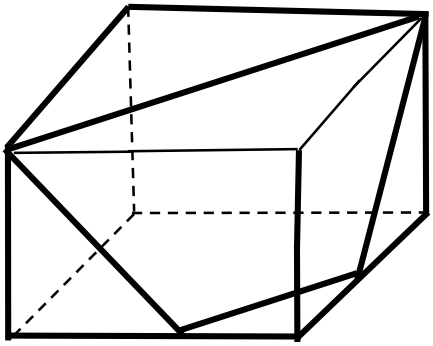
11



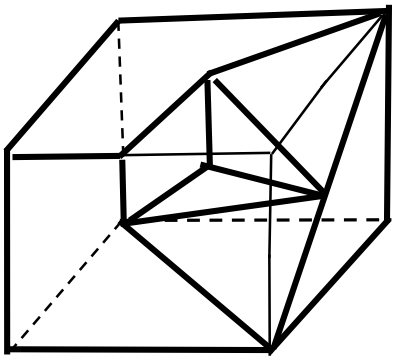
12



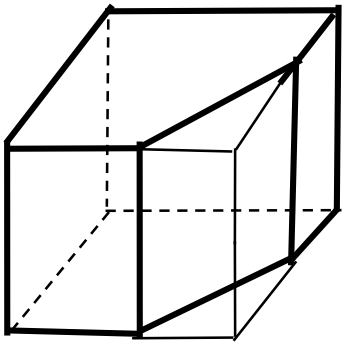
13



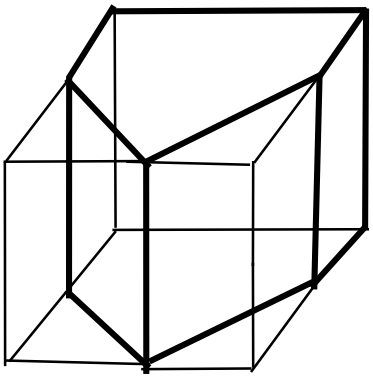
14



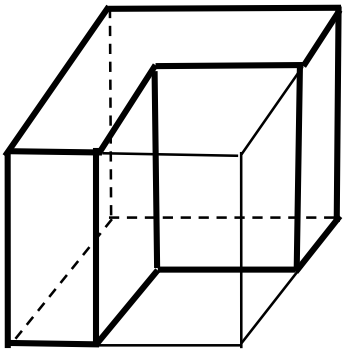
15



16



17



18

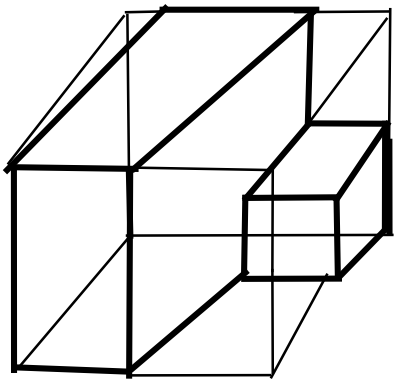
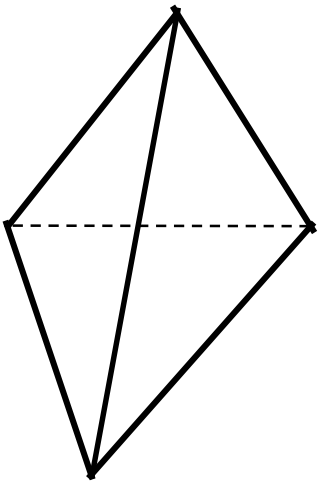
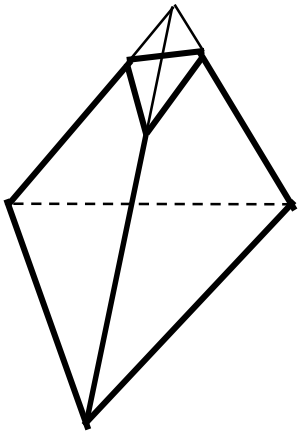


Рисунок А.3 – продолжение

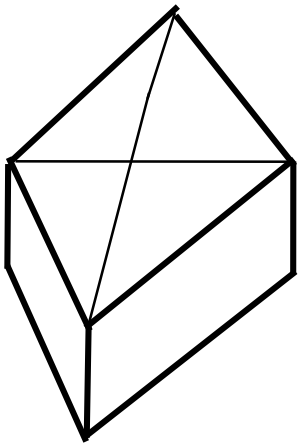
19



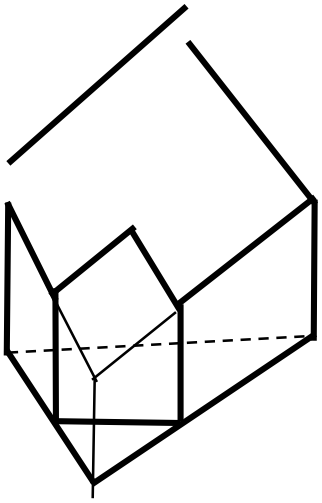
20



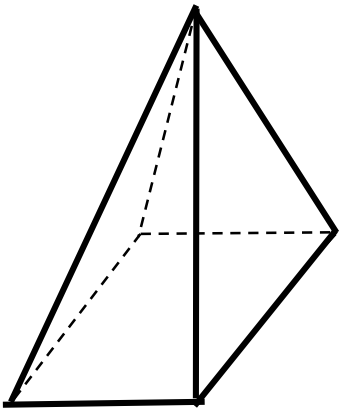
21



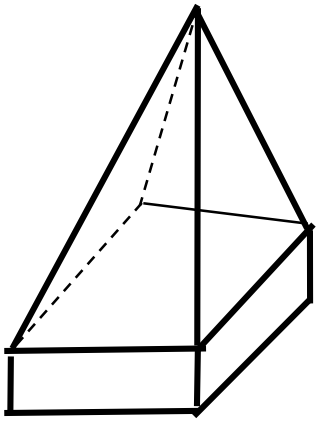
22



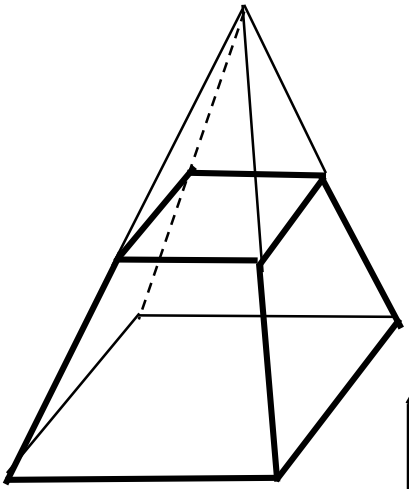
23



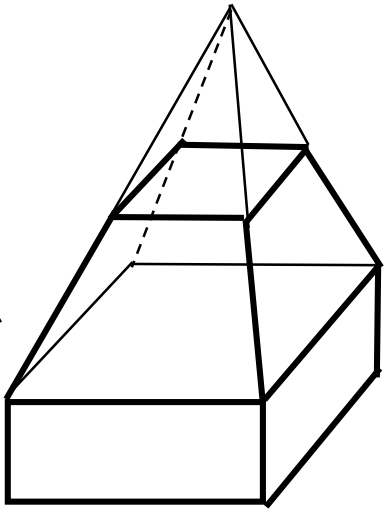
24



25



26



27

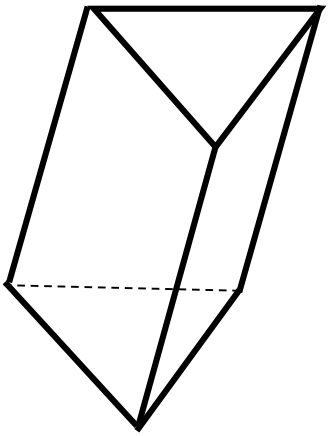


Рисунок А.3 – окончание

Приложение Б

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа № 1 по дисциплине:
«Компьютерная графика»

«Формирование цветов. Изучение цветовых характеристик.
Аддитивная цветовая модель RGB»

Вариант № 7

Выполнил: студент группы №
Проверил: Шилов А.В,

Могилев 2020