***1.Основные этапы развития компьютерной графики. Основные направления развития компьютерной графики.***

Основные этапы развития КГ:

• 60-70-е годы научная дисциплина.

Бурное развитие методов, алгоритмов отсечение, генерация примитивных графических элементов, закраска узорами, реалистическое представление сцен (удаление невидимых линий и граней, трассировка лучей, излучающие поверхности);

• 80-е годы прикладная наука.

Отработка методов, средств, аппаратуры в различных сферах приложений;

• 90-е годы основное средство общения человека с ЭВМ.

Направления компьютерной графики:

• изобразительная компьютерная графика,

Объекты: синтезированные изображения.

Задачи:

* построение модели объекта и генерация изображения,
* преобразование модели и изображения,
* идентификация объекта и получение требуемой информации.

• обработка и анализ изображений,

Объекты: дискретное, числовое представление фотографий.

Задачи:

* повышение качества изображения,
* оценка изображения определение формы, местоположения, размеров и других параметров требуемых объектов,
* распознавание образов выделение и классификация свойств объектов (обработка аэрокосмических снимков, ввод чертежей, системы навигации, обнаружения и наведения).

• анализ сцен (перцептивная компьютерная графика),

Предмет: исследование абстрактных моделей графических объектов и взаимосвязей между ними. Объекты могут быть как синтезированными, так и выделенными на фотоснимках. Первый шаг в анализе сцены выделение характерных особенностей, формирующих графический объект(ы).

Примеры: машинное зрение (роботы), анализ рентгеновских снимков с выделением и отслеживанием интересующего объекта, например, сердца.

Итак, в основе анализа сцен (перцептивной компьютерной графики) находятся изобразительная графика + анализ изображений + специализированные средства

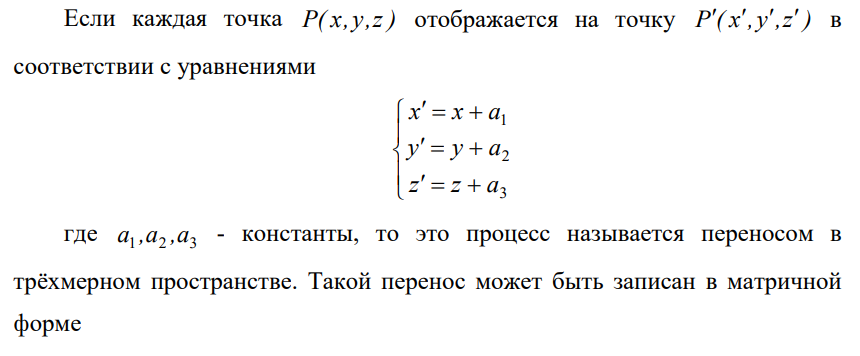
• компьютерная графика для научных абстракций (когнитивная компьютерная графика, способствующая познанию).

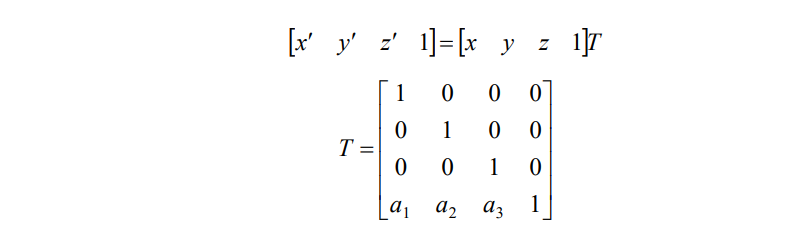
Это компьютерная графика для научных абстракций, способствующая рождению нового научного знания. База мощные ЭВМ и высокопроизводительные средства визуализации.

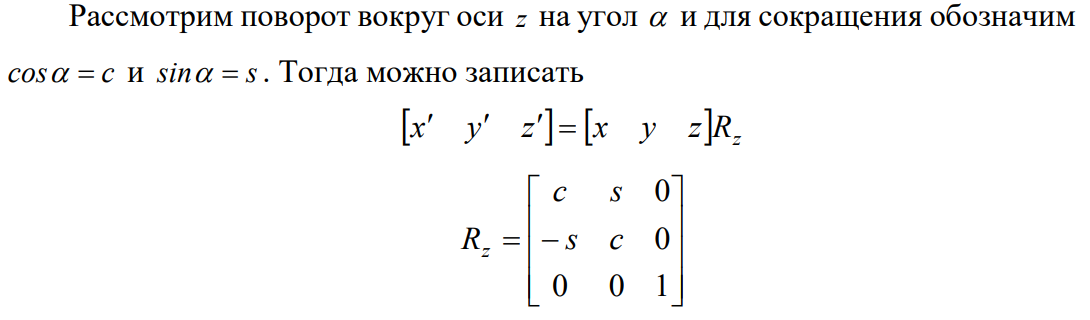
Общая последовательность познания заключается в, возможно циклическом, продвижении от гипотезы к модели (объекта, явления) и решению, результатом которого является знание.

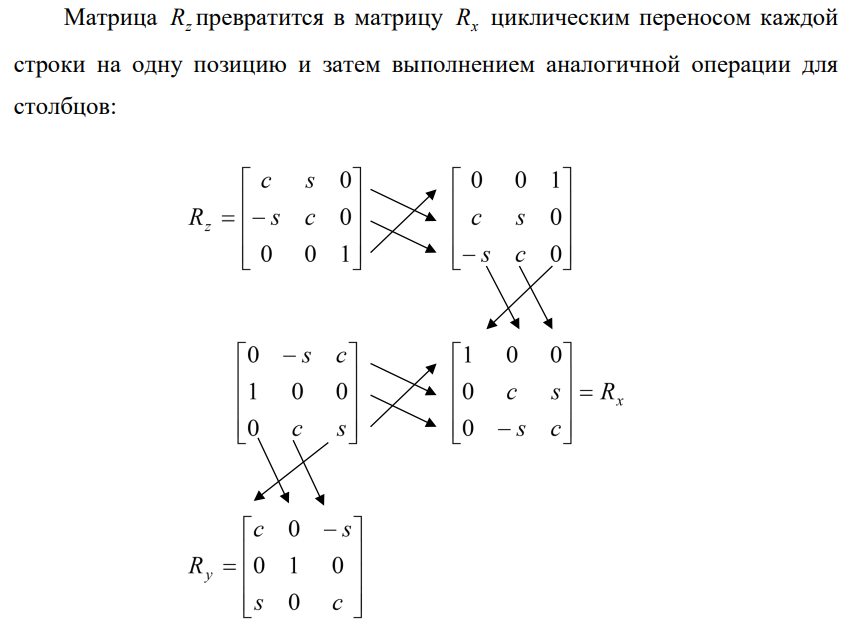
Основная проблема и задача когнитивной кг - создание таких моделей представления знаний, в которых можно было бы однообразно представлять, как объекты, характерные для логического (символического, алгебраического) мышления, так и объекты, характерные для образного мышления.

***2.Перенос и повороты в трёхмерном пространстве.***

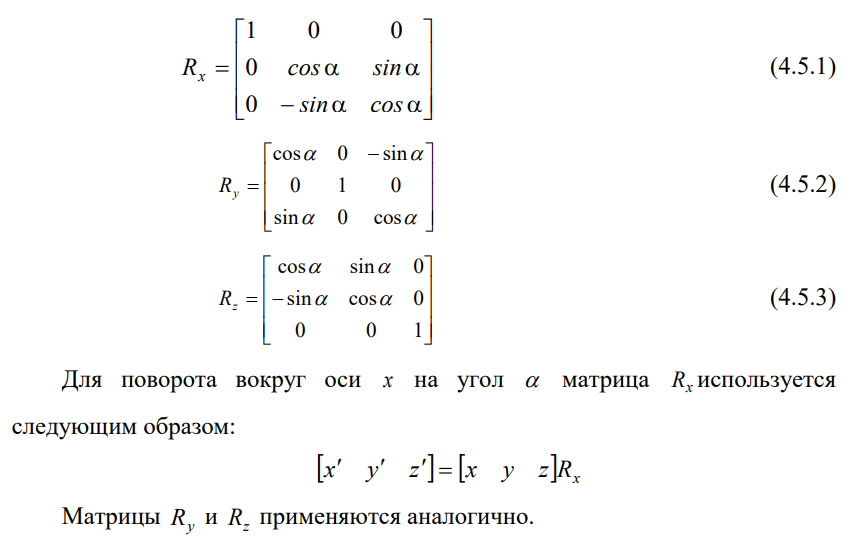


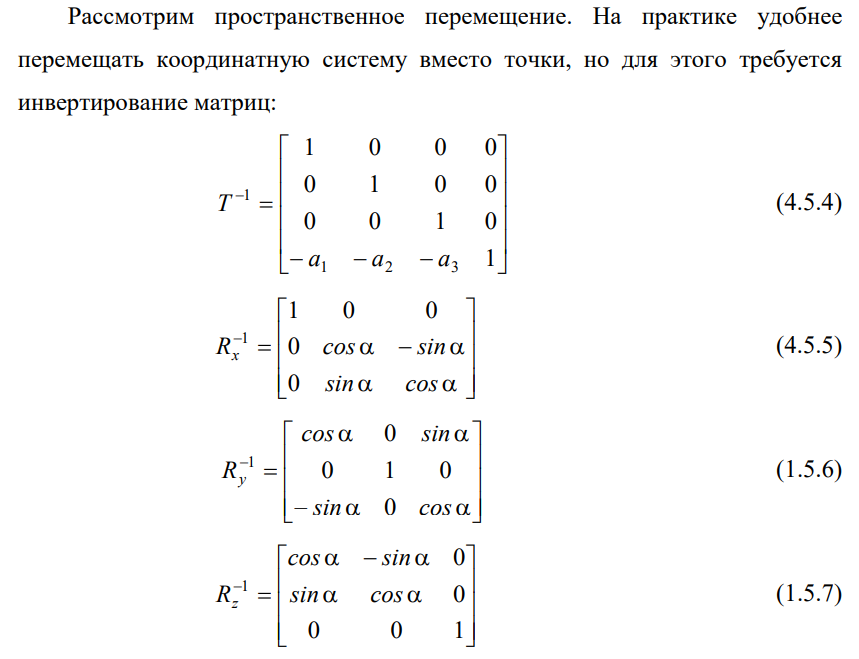


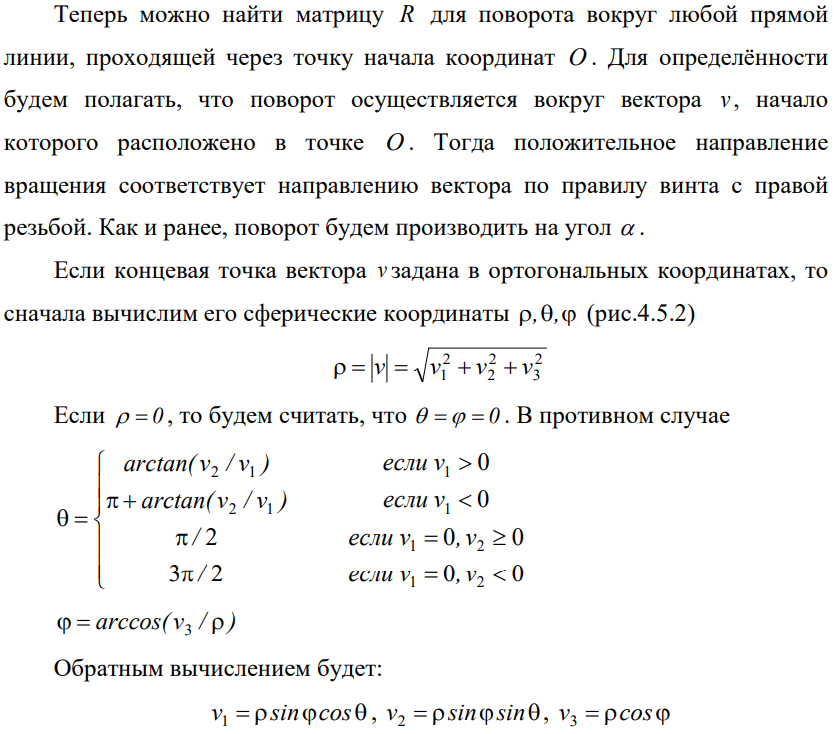


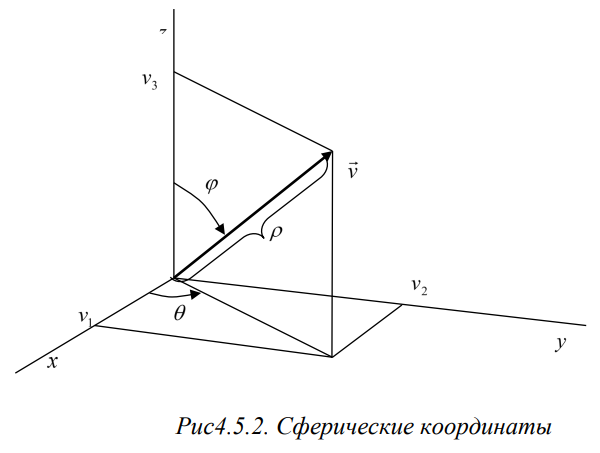


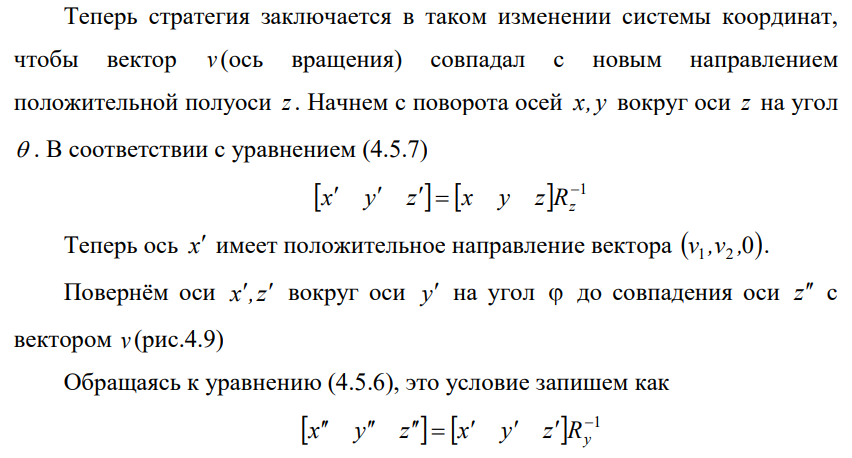
Получим следующие матрицы поворота вокруг осей x, y, z (в положительном направлении):

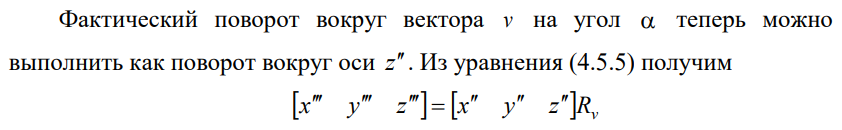


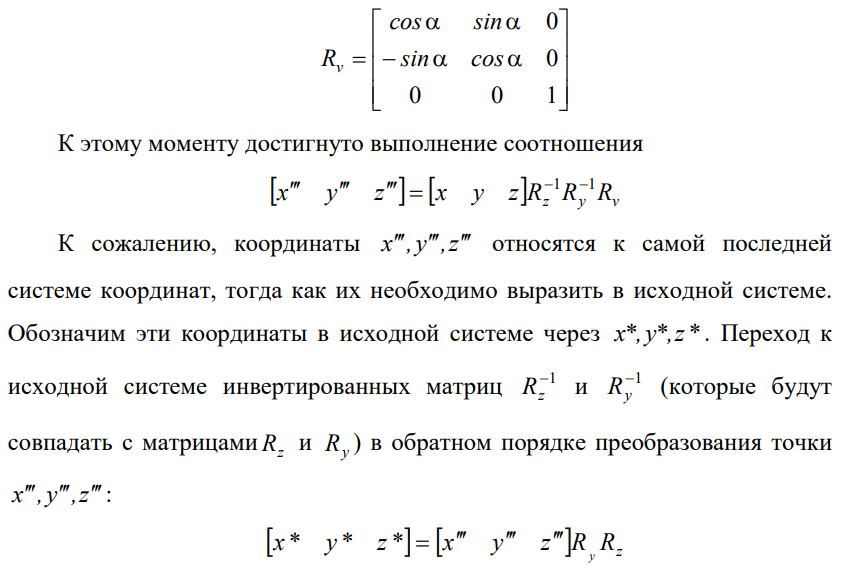


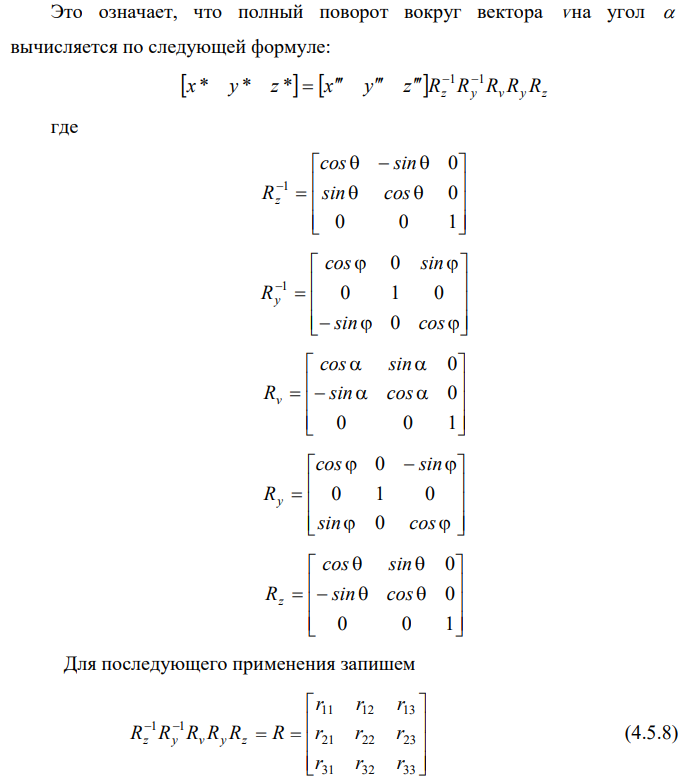


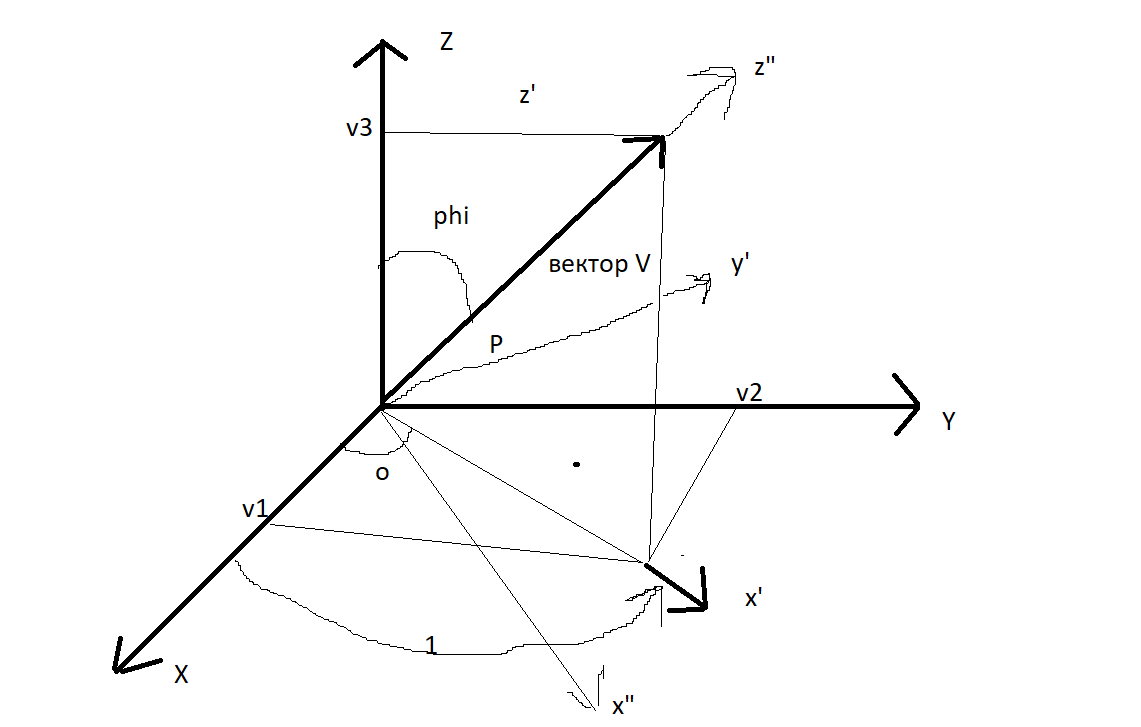




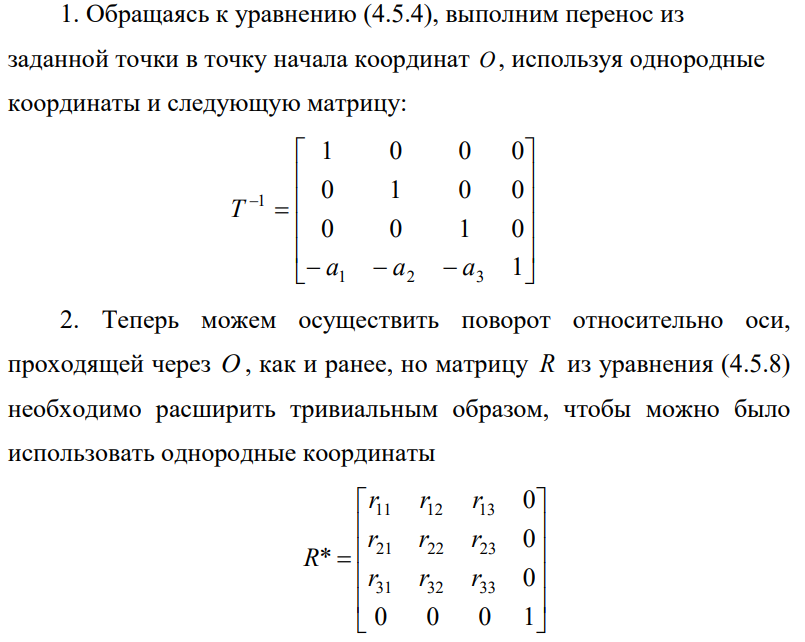


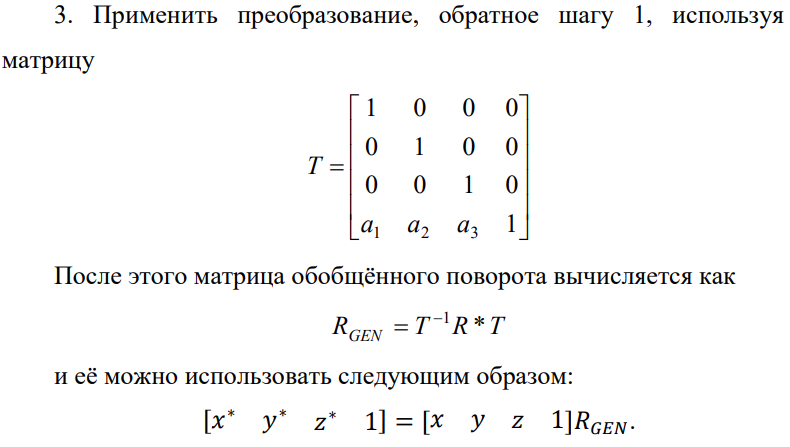






**Поворот относительно вектора, начало координат которого расположено в любой произвольной точке A( a1, a2, a3):**





***3.Приложения компьютерной графики. Виды представления компьютерной графики.***

Основные области:

• Компьютерное моделирование

• Системы автоматизации научных исследований, проектирования, конструирования, производства, управления технологическими процессами

• Бизнес, искусство, СМИ, досуг

• Виртуальная реальность

Виды:

• Растровая - обработка полноценных изображений. Представляют из себя совокупность пикселей, строки и столбцы которых и называются растром.

• Векторная - описывает изображения с использованием математических описаний объектов, окружностей, линий и их параметров. Используется математическое описание объектов (формулы).

• Фрактальная (НЕОБЯЗАТЕЛЬНЫЙ ПУНКТ) - раздел математики, занимающийся визуализацией геометрических фигур, обладающих самоподобием. Получают лишь при помощи программирования.

***4.Виды графического представления геометрических моделей. Цвет. Система RGB, CMYK.***

Цвет в компьютерной графике разбивается на два природных состояния: излучаемый источником света и отражаемый от поверхности. При этом, оба состояния базируются на физическом свойстве светового луча разлагаться на цветовой спектр.

RGB - излучаемые или аддитивные цветовые модели (red, green, blue). Получается при соединении разных излучаемых цветов (0,0,0 - черный, 255,255,255 – белый)

CMY - отражаемые или субтрактивные модели (cyan, magenta, yellow). Отражаемый цвет получается при вычитании других цветов из общего луча света (присутствие всех отражаемых цветов даёт чёрный, отсутствие - белый цвет)

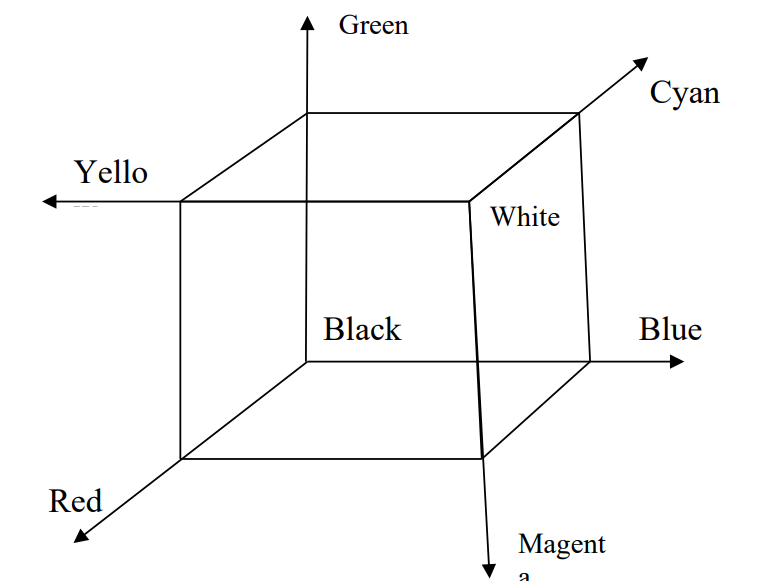
CMYK - наслоение цвета в процессе полиграфической печати. К - последняя буква black, который необходим для текстовой печати

Модели RGB и HSB применяют для создания и обработки аддитивных изображений, а CMYK для печати изображения на полиграфическом оборудовании.

Законы Германа Грассмана:

1. Закон трехмерности

Любой цвет однозначно выражается тремя составляющими, если они линейно независимы. Линейная независимость заключается в невозможности получить любой из этих трех цветов сложением двух остальных.

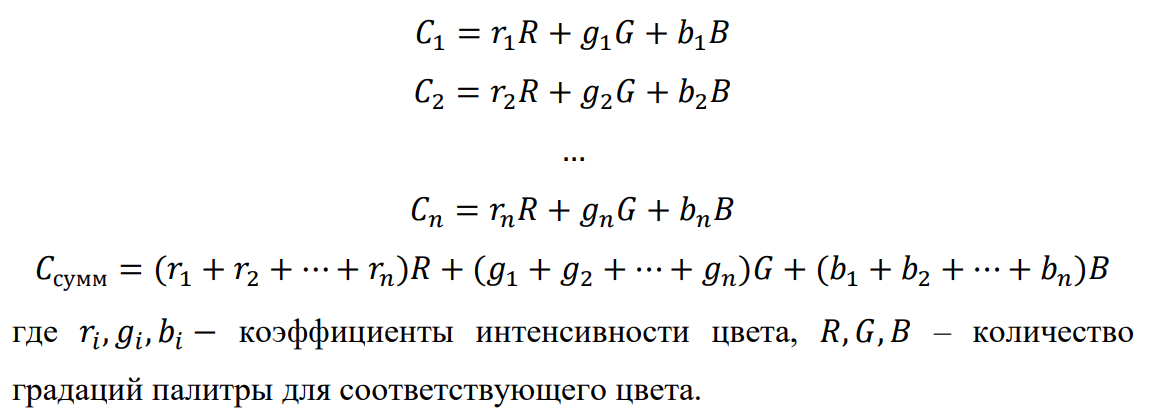


1. Закон непрерывности

При непрерывном изменении излучения цвет смеси также меняется непрерывно. Не существует такого цвета, к которому нельзя было бы подобрать бесконечно близкий.

1. Закон аддитивности

Цвет смеси излучений зависит только от их цвета, но не спектрального состава. То есть цвет (С) смеси выражается суммой цветовых уравнений излучений:



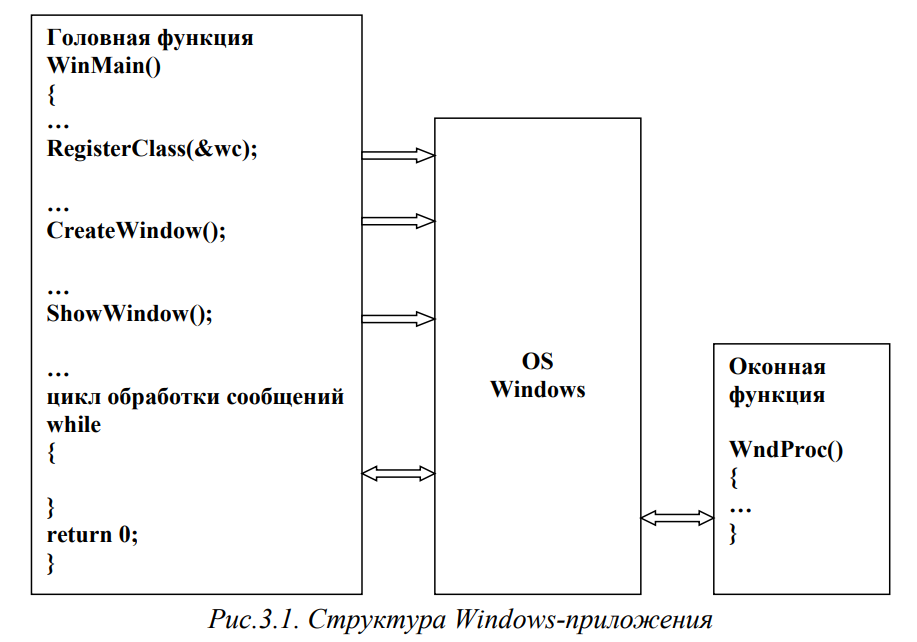
5.Однородные координаты.

6.Известные методы решения задач загораживания.

***7.Основные принципы программирования Windows -приложения.***

В MS-DOS программа монопольно владеет всеми ресурсами системы и является инициатором взаимодействия с операционной системой. Операционная система Windows, задуманная как многозадачная, является инициатором обращения к программе. Все ресурсы Windows являются разделяемыми, и Windows приложение не может владеть ими монопольно. Windows-приложение ожидает посылки сообщения операционной системой, и лишь после его получения, выполняет определённое действие, а затем вновь переходит в режим ожидания очередного сообщения от операционной системы.

***8.Внутреннее компьютерное представления каркаса геометрической модели.***



Основные стандартные действия для работы с Windows-приложением:

1. Определить класс окна

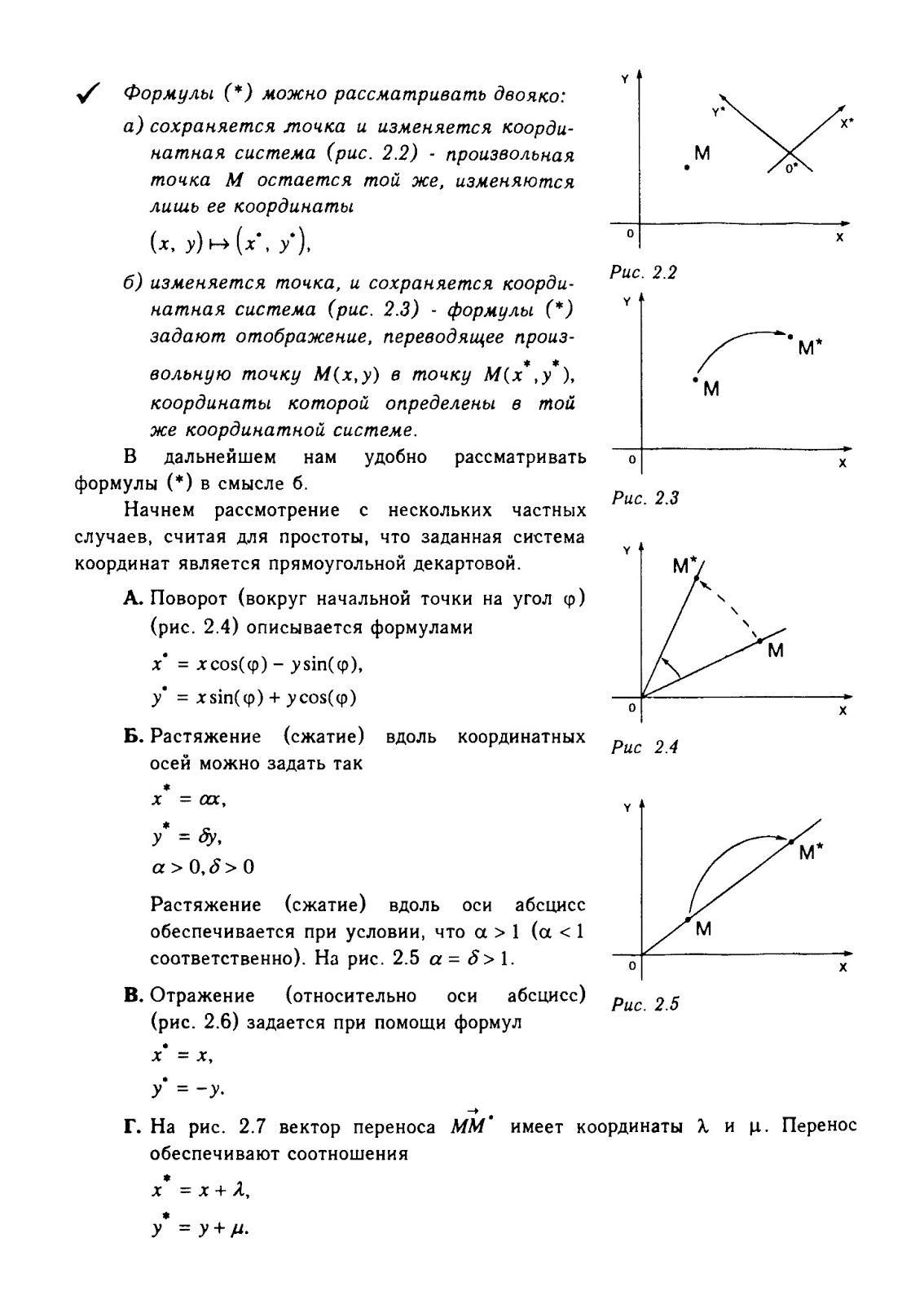
2. Зарегистрировать окно

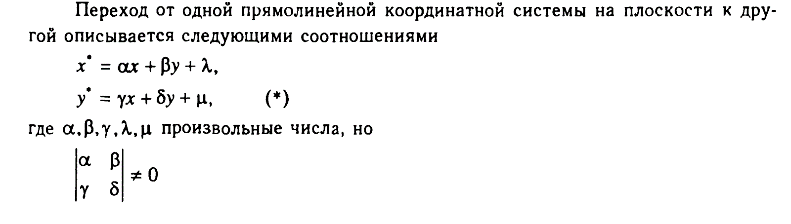
3. Создать окно данного класса

4. Отобразить окно

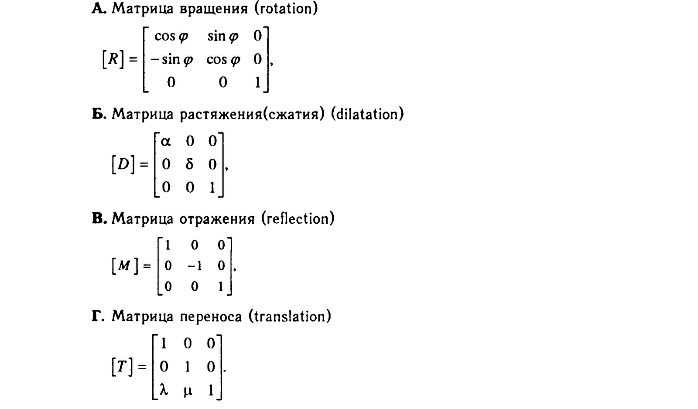
5. Запустить цикл обработки сообщений

***9.Двумерные преобразования и новые координаты.***



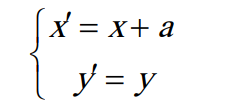


Матричные преобразования:



Дальше идет вывод всех этих уравнений и матриц:

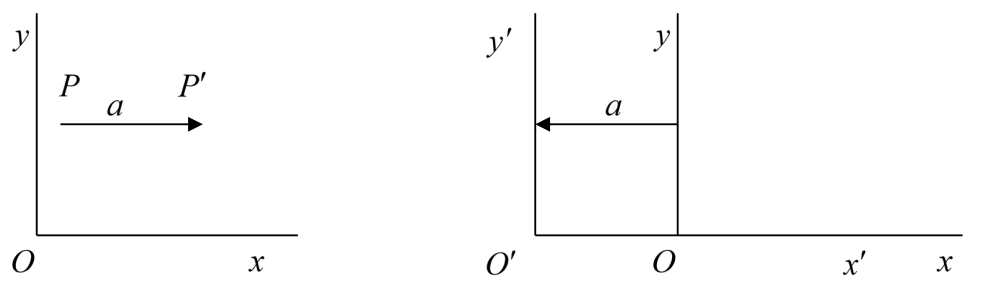
Рассмотрим следующую систему уравнений:



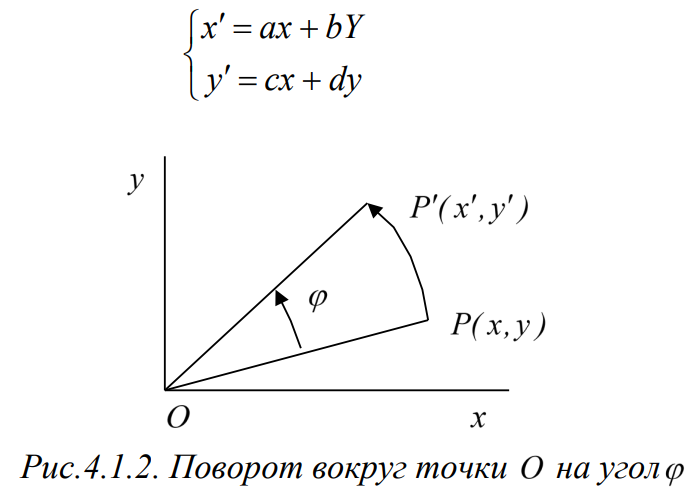
Эти уравнения можно интерпретировать двояким способом:

1. Все точки на плоскости x и y перемещаются вправо на расстояние a

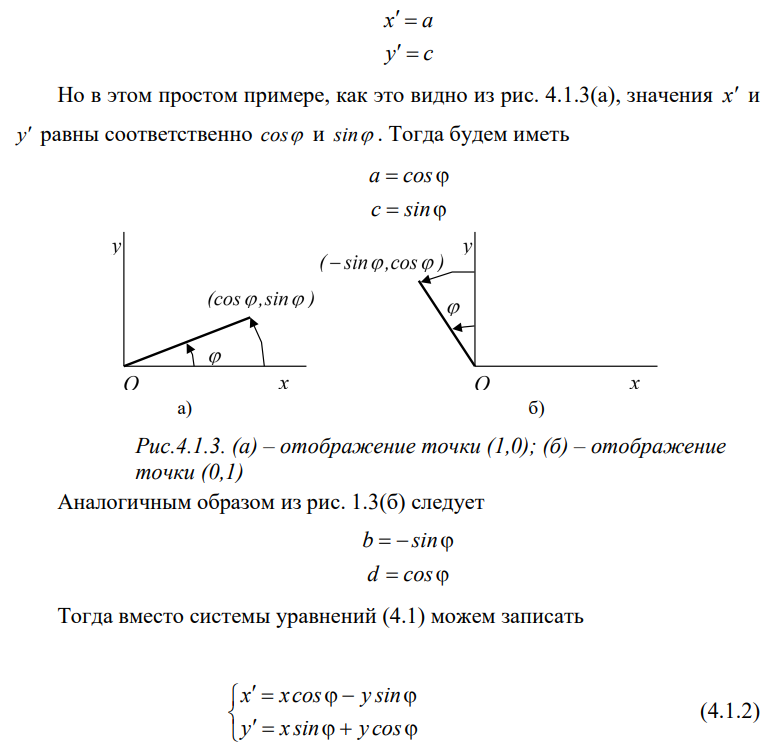
2. Координатные оси x и y перемещаются влево на расстояние a



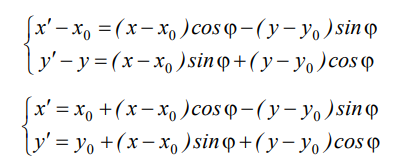
**Поворот точки P( x, y ) вокруг начала координат O на угол ϕ:**



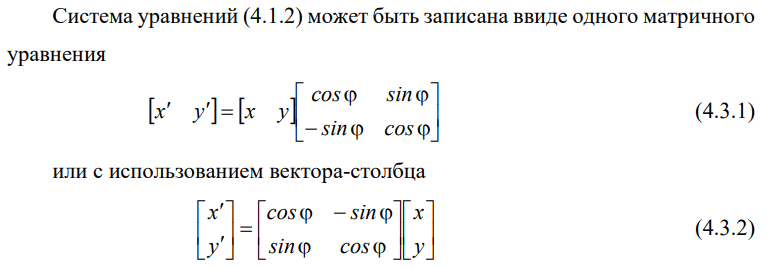
Для получений значений a,b,c,d рассмотрим в начале точку ( x, y ) = (1,0 ), подставив в систему уравнений получим:

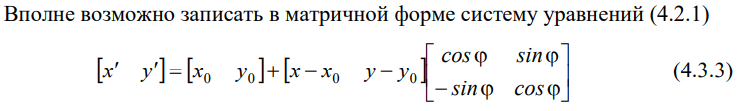


**Поворот относительно заданной точки (x0, y0)** (В системе уравнений заменяем: x на x – x0 , y - на y – y0 , x′ - на x′ − x0 и y′ - на y′ − y0 )

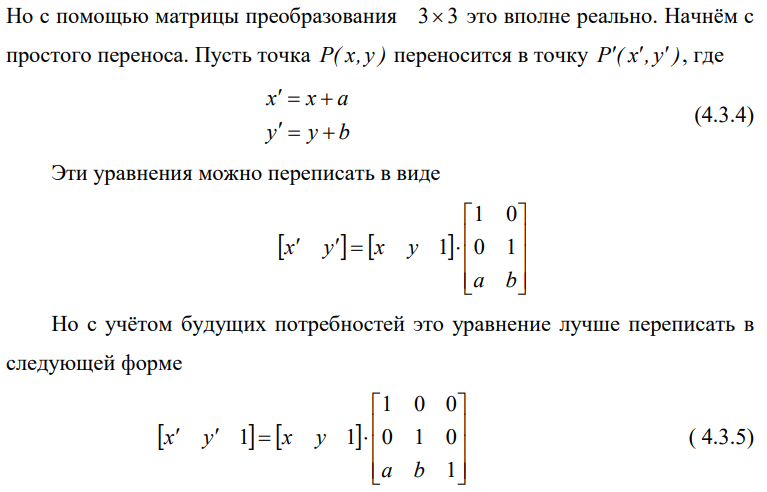


**Матричная запись**



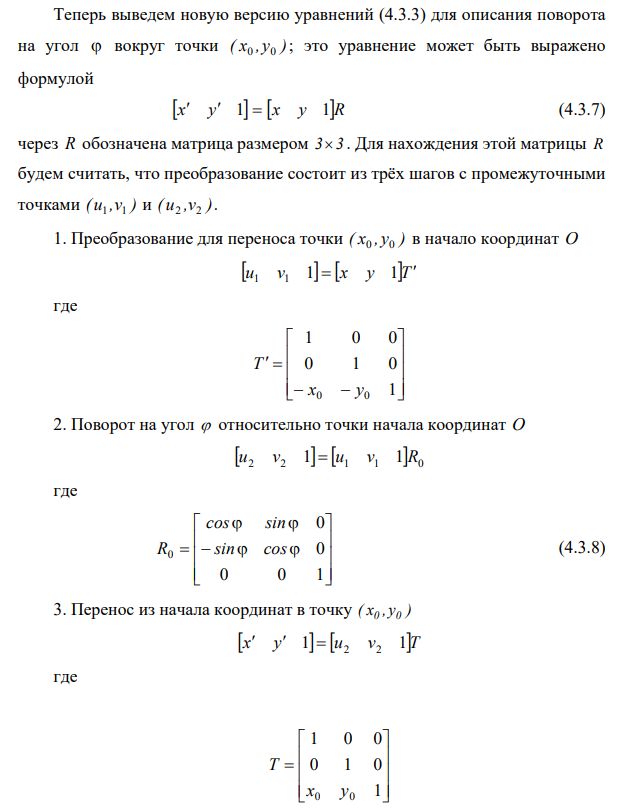
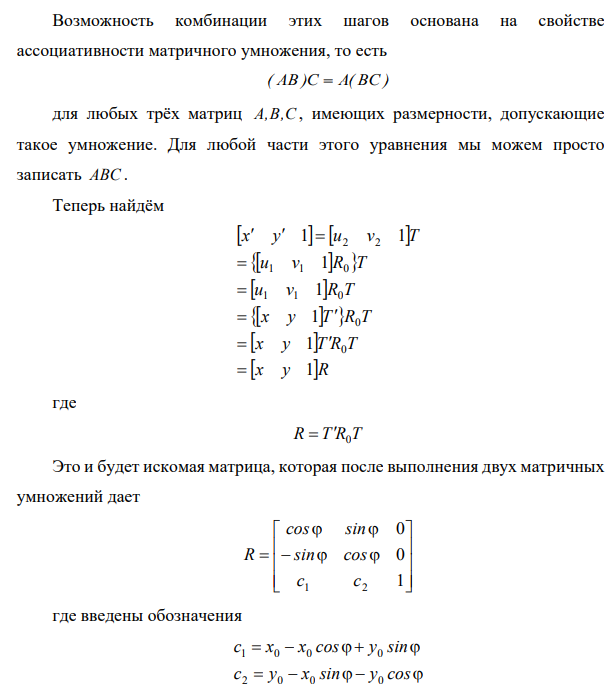


Однако первая часть этого уравнения не является чисто матричным произведением. Исправим это:



Такую запись принято называть записью в системе «однородных координат».

Однородные координаты ― координаты, обладающие тем свойством, что определяемый ими объект не меняется при умножении всех координат на одно и то же ненулевое число. Они возникли из - за необходимости преобразования из n - мерного пространства в n+1 мерное.

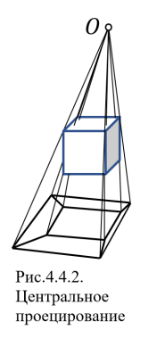
***10.Перспективные изображения как результат центрального проецирования.***

Проецирование — метод получения проекций — изображений пространственных предметов на плоскости проекций при помощи пучка воображаемых проецирующих световых или зрительных лучей [7]. При этом предмет располагается между наблюдателем и плоскостью проекций. Основным законом проецирования, который ложится в основу зрительного восприятия пространства человеком является центральное проецирование.

На рисунке 4.4.1. показан простейший пример центрального проецирования точки 𝑃(𝑥, 𝑦) двухмерного пространства в точку 𝑃′(𝑋) на прямой 𝑙 одномерного пространства. Проецирующий луч 𝑆 проходит через начало системы координат xOy (центр проекции).

В начертательной геометрии рассмотрению подлежат в основном задачи 3D-моделирования, поэтому при центральном (коническом) проецировании из фиксированной точки О (центра проекции) через все точки проецируемого предмета (объекта) мысленно проводят прямолинейные лучи до их пересечения с плоскостью проекций (рис.4.4.2). Точки их пересечения образуют требуемое, но увеличенное изображение предмета.

Существует несколько видов закона центрального проецирования.

Центральная проекция широко используется в системе изображения предметов на плоскости (4.4.2). При удалении центра проекции в бесконечность конические лучи воспринимаются как параллельные.

***11.Разработка каркаса Windows –приложения.***

Чтобы иметь возможность работать с оконным интерфейсом, заготовка или каркас windows приложения должна выполнить которое стандартные действия:

1) определить класс окна;

2)зарегестрировать окно;

3) создать окно данного класса;

4) отобразить окно;

5) запустить цикл обработки сообщения.

**Определитель класса окна.**

WinMain( );

**HWND hWnd;**// дискриптор главного окна (описание);

Дискриптор – описатель, используемый для описания объектов ОС. Дискриптор позволяет отслеживать состояние объекта в памяти, при его перемещении по инициативе ОС.

Тип дискриптора – 32 разрядное целое числовое.

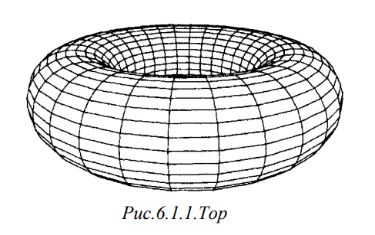
**HWND**– предназначен для хранения дискриптора главного окна программы.

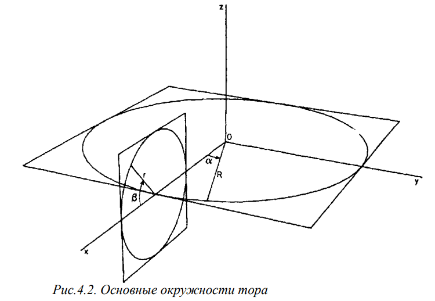
**MSG msg**; (MSG – это структура в которой храниться информация по сообщение передаваемой ОС в окно приложения)

Наша программа начинает выполняться с функции WinMain(). Это и есть главная функция. Функция WinMain() выполняет, обычно, следующие задачи:

* Определяет класс окна.
* Регистрирует данный класс в системе.
* Создает главное окно приложения и другие элементы управления.
* Отображает окно на экране.
* Запускает цикл обработки сообщений.

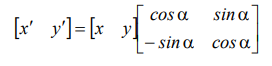
***12.Основные принципы построения алгоритмов генерации простых геометрических объектов. Генерация файла для геометрического объекта «тор».***

Рассмотрим пример, в котором координаты всех вершин вычисляются по ограниченному количеству данных. В качестве такого примера рассмотрим тор, изображенный на рис.6.1.1. Входные данные для этой программы состоят из трех чисел n,R и r( R > r ).

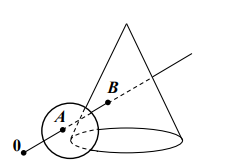
На рис.4.2 большая горизонтальная окружность определяет положение центров окружностей, образующих тор, радиус этой окружности равен R. Выберем n равноудаленных точек на этой окружности в качестве центров малых вертикально расположенных окружностей радиуса r. Параметрическое представление большой окружности описывается формулами x =Rcosα y=Rsinα z=0.

Точка, соответствующая α = 0, является центром малой окружности x = R + rcosB y=0 z=rsinB которая также показана на рис. 4.2. Остальные n −1 малые окружности формируются поворотом этой исходной окружности вокруг оси z на угол α = iδ, где i =1,...,n −1 и δ = 2π / n .

На малой окружности выберем n точек с номерами вершин 0,1,...,n −1. Размещение точек на первой малой окружности определяется параметром β = jδ , им приписываются номера вершин j( j = 0,1,...,n −1). Следующие n вершин, пронумерованные n,n +1,...,2n −1, лежат на соседней малой окружности, соответствующей i =1, и так далее. В общем, мы получим номера вершин i ⋅ n + j(i = 0,1,...,n −1, j = 0,1,...,n −1). Поворот на угол α = iδ относительно оси z записывается как В нашей ситуации исходная малая окружность лежит в плоскости xOz, поэтому y = 0 , что сокращает это матричное произведение до x′ = xcosα y’=ycosα.

Этот же результат может быть получен непосредственно из рис. 4.2.

***13.Основная концепция решения задач загораживания.***

******Рассмотрим некоторые подходы к решению этой задачи, которую часто также называют задачей загораживания.

1.Установка приоритета, в соответствии с которым нужно производить обработку данных для получения целостного изображения. а) Пусть сцена разбита на два фрагмента – **А** и **В**. Если самые дальние точки фрагмента **А** лежат ближе к наблюдателю, чем самые ближние точки фрагмента **В**, то части **В** не могут загородить фрагмента **А**. Поэтому, если мы сначала построим изображение фрагмента **В**, а затем – изображение фрагмента **А**, то в результате правильно получим изображение всей сцены. б). Пусть в объектном пространстве удалось найти плоскость, которая разделяет все пространство на такие два полупространства, в одном из которых находится фрагмент **А** и наблюдатель, а в другом – фрагмент **В**. При этих условиях фрагмент **В** не может загораживать фрагмент **А**. Остается лишь выявить, в каком полупространстве находится наблюдатель.

Если проекции фрагментов **А** и **В** не пересекаются, то обработка каждого из них может производиться независимо.

2.Установка факта независимости фрагментов.

Общая задача о пересечении проекций произвольных объектов весьма сложна, однако использование простых достаточных условий непересечения часто оказывается полезным. Например, описав вокруг проекций прямоугольники со сторонами, параллельными осям координат, и убедившись, что они не пересекаются (это сводится к проверке нескольких неравенств), мы можем быть уверены, что фигуры также не пересекаются.

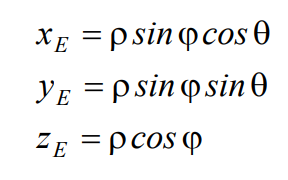
***14.Математические способы приближения функции.***

Задача приближения функций возникает при обработке и отображении экспериментальных данных и в ходе моделирования и отображения геометрии сложных криволинейных объектов.

Простейшая, одномерная задача интерполяции, приводящая к приближению функций, заключается в следующем: в дискретные моменты x1, x2,…,xn наблюдаются значения функции y = f ( x ); требуется восстановить её значение при других x , т.е. определить такую функцию g , чтобы f ( x ) ≈ g( x; a1, a2,...,an ), где a1, a2,..., an определяются из условия совпадения функции g(x) и точках xk, g(x) = f (xk), k = 1, 2,…,n.

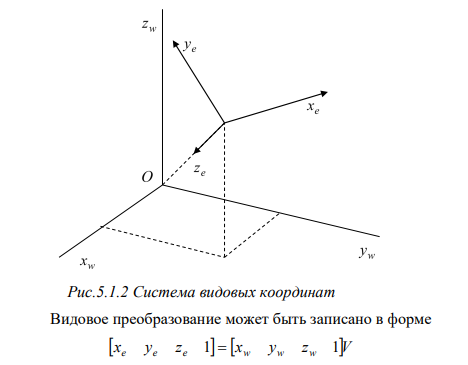
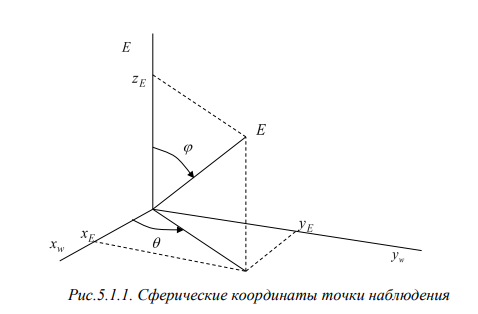
***15.Видовое преобразование как аппарат построения аксонометрических изображений.***

Для выполнения видовых преобразований должны быть заданы точка наблюдения, совпадающая с глазом, и объект. Желательно, чтобы система координат была правой. Будет удобно, если начало её координат располагается где-то вблизи центра объекта, поскольку объект наблюдается в направлении от E к O . Предположим, что это условие выполняется. Пусть точка наблюдения E будет задана в сферических координатах ρ, θ, ϕ по отношению к мировым координатам. То есть мировые координаты могут быть вычислены по формулам:



Обозначение сферических координат схематически изображены на рис. 5.1.1. Говорят, что вектор направления EO (равный − OE ) определяет направление наблюдения. Из точки наблюдения E можно видеть точки объекта только внутри некоторого конуса, ось которого совпадает с линией EO , а вершина – с точкой E . Если заданы ортогональные координаты xE, yE, zE точки E, то можно вычислить её сферические координаты (п.4.5).

Конечной задачей является вычисление экранных координат X ,Y , для которых оси X и Y лежат в плоскости экрана, расположенной между точками E и O и перпендикулярной направлению наблюдения EO . Начало системы видовых координат располагается в точке наблюдения E (рис.5.1.2). При направлении взгляда из E в O положительная полуось xe направлена вправо, а положительная полуось ye - вверх. Такое направление осей позволит впоследствии определить экранные оси в тех же направлениях. Направление оси ze выбирается таким образом, что значения координат увеличиваются по мере удаления от точки наблюдения.

******

где V - матрица видового преобразования размерностью 4×4. Для нахождения матрицы V предположим, что преобразования отображения могут быть составлены из четырёх элементарных преобразований, для которых легко написать свои матрицы преобразований. Матрица V получается путём перемножения этих четырёх матриц. Фактически каждое из четырёх преобразований изменяет координаты и, следовательно, определяется матрицей, обратной матрице, соответствующей преобразованию точки.

***16.Интерполяция точечного каркаса функции полиномами.***

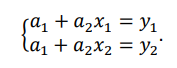
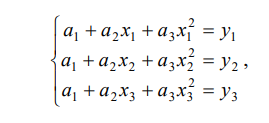
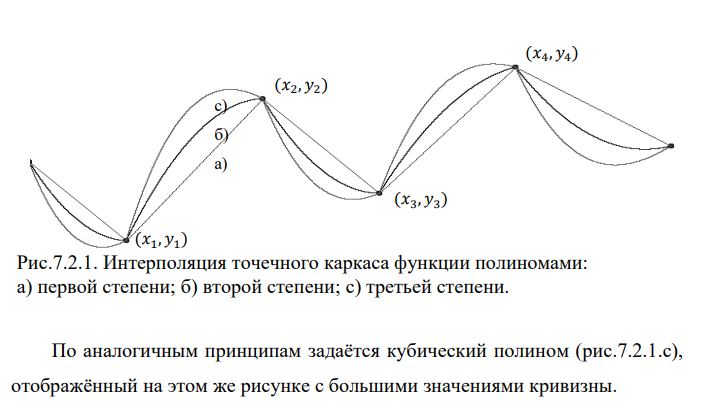
Линейная интерполяция полиномами изображена на рисунке 7.2.1.а и в основе применяет линейный вид многочлена 𝑔(𝑥) = 𝑎1 + 𝑎2𝑥. Чтобы прямая, описанная таким уравнением прошла через точки (𝑥1, 𝑦1) и (𝑥2, 𝑦2) достаточно выразить коэффициенты 𝑎1 и 𝑎2 через соответствующие уравнения системы: 

Рисунок 7.2.1.б отображает квадратичный полином вида g( x ) = a1 + a2x + a3x2 (1), при этом, коэффициенты определяются исходя из условия

******где (𝑥1, 𝑦1), (𝑥2, 𝑦2), (𝑥3, 𝑦3)- узловые точки, определяющие параболу участка кривой (Рис.7.2.1.б).

******

***17.Метод Z-буфера.***

Другим методом «грубой силы» является метод Z-буфера, который весьма удобен для аппаратной реализации ввиду простоты алгоритма и используемого в нем набора операций. Временные характеристики этого метода линейно зависят от количества точек растра и «глубинной сложности сцены», т.е. усредненного числа граней, взаимно закрывающих друг друга.

Для реализации метода используются две области памяти: буфер глубины (Z-буфер) и буфер кадра (хранящий информацию о состоянии пикселов экрана компьютера). Буфер глубины используется для хранения координаты **Z** (глубины) каждого видимого на данной стадии анализа изображения пиксела картинной плоскости. В буфере кадра запоминаются атрибуты (интенсивность и цвет) соответствующего пикселя.

Формально описание метода таково. Предположим, что сцена представлена в виде объединения многоугольников (возможно, пересекающихся). Построим ортогональную проекцию сцены на картинную плоскость z = 0.

Предполагается следующая последовательность шагов:

1. Инициализировать буфер кадра фоновыми значениями интенсивности или цвета.

2. Инициализировать буфер глубины значениями глубины фона.

3. Для каждой грани сцены последовательно:

(4). Преобразовать проекцию границ в растровую форму.

(5). Для каждого пиксела проекции вычислить его глубину z = z( x, y );

(6). Сравнить значение z( x, y ) с соответствующим значением буфера глубины Z( x, y ).

(7). Если z( x, y ) < Z( x, y ), то:

(8). Записать атрибуты этого пиксела в буфер кадра;

(9). Записать значение z( x, y ) в соответствующую позицию буфера глубины Z( x, y ).

(10). Иначе – никаких действий не производить.

***18.Интерполяция сплайнами.***

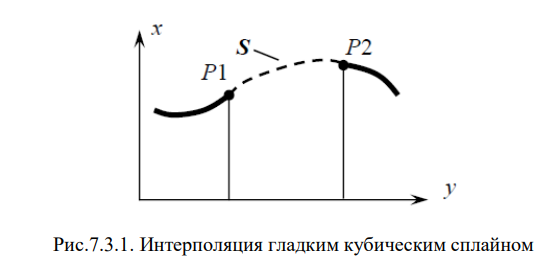
Пусть две точки 𝑃1и 𝑃2, показанные на рисунке 7.3.1, нужно соединить кривой S таким образом, чтобы она проходила через эти точки и гладко сопрягалась с соседними участками кривой. Соседние участки на рисунке показаны толстыми сплошными линиями, а желаемый вид кривой S – штриховой линией. Очевидно, для решения задачи нужно наложить на кривую S четыре ограничения:

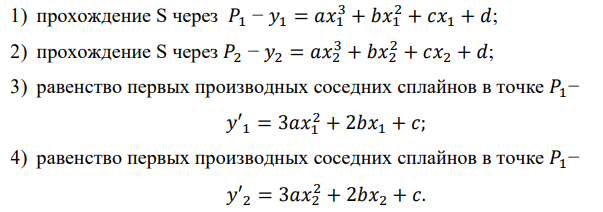
• кривая S проходит через точку 𝑃1;

• кривая S проходит через точку 𝑃2;

• сопряжение кривой S с соседним участком в точке 𝑃1 является гладким;

• сопряжение кривой S с соседним участком в точке 𝑃2 является гладким.

******Для этого требуется иметь математическое описание кривой S в виде многочлена, имеющего не менее четырех коэффициентов. Самым простым из таких многочленов является многочлен третьей степени, который, в общем случае, имеет вид 𝑦 = 𝑎𝑥3 + 𝑏𝑥2 + 𝑐x + 𝑑, включающий как раз четыре коэффициента формы a, b, c и d. Тогда наложение четырех упомянутых ограничений дает систему четырех уравнений с четырьмя неизвестными:

******

***19.Графические средства Windows API.***

**Рисование линии**

Для того, чтобы нарисовать линию, используется функция LineTo():

BOOL WINAPI LineTo(HDC hdc, int x, int y);

Здесь hdc – дескриптор контекста устройства, x, y – координаты конца линии. Начало линии определяется текущей позицией рисования. При создании окна текущая позиция определяется в начале координат.

**Установка текущей позиции**

Для установки точки в текущую позицию с координатами x, y применяется функция:

BOOL WINAPI MoveToEx(HDC hdc, int x, int y, LPPOINT oldPoint);

Если последний параметр NULL, то предыдущие координаты не сохраняются.

**Определение размера клиентской области**

Сообщение WM\_SIZE генерируется системой при создании окна после сообщения WM\_CREATE и при каждом изменении его размеров:

case WM\_SIZE:

sx = LOWORD(lParam); // ширина окна

sy = HIWORD(lParam); // высота окна

break;

**Рисование прямоугольника**

Нарисовать прямоугольник можно при помощи функции:

BOOL WINAPI Rectangle(HDC hdc, int x1, int y1, int x2, int y2);

где (x1, y1) – координаты левого верхнего угла прямоугольника, а (x2, y2) – координаты правого нижнего угла.

**Рисование эллипса**

Функция для изображения эллипса имеет те же параметры, поскольку эллипс определяется ограничивающим его прямоугольником

BOOL WINAPI Ellipse(HDC hdc, int x1, int y1, int x2, int y2);

**Рисование точки**

Функция, выводящая в окно одну точку описана как:

COLORREF WINAPI SetPixel(HDC hdc, int x, int y, COLORREF color);

Зададим функцию рисования красного пикселя:

SetPixel(hdc, x, y, RGB(255, 0, 0));

**Создание пера**

Пользовательское перо создаётся функцией:

HPEN WINAPI CreatePen(int style, int width, COLORREF color)

**Создание кисти**

Кисть используется для заполнения фона окна или замкнутой области внутри окна. Сплошная кисть создаётся при помощи функции:

HBRUSH Create SolidBrush(COLORREF color);

**Создание пути**

Путь – это набор прямых линий и кривых. Его можно применять для графических построений сложных регионов и использовать для отсечения.

HDC WINAPI BeginPath(HDC hdc) // Открывает путь, теперь графические функции в окно ничего не выводят, а строят путь.

HDC WINAPI CloseFigure(HDC hdc) // Закрывает открытую фигуру в пути. Замыкает первую и последнюю точки.

HDC WINAPI EndPath(HDC hdc) // Закрывает путь и помещает его в контекст устройства.

HDC WINAPI FillPath(HDC hdc) // Закрашивает текущей кистью область, ограниченную путём. HDC WINAPI PathToRegion(HDC hdc) // Преобразует путь в область, возвращает его дескриптор. HDC WINAPI StrokePath(HDC hdc) // Обводит путь текущим пером.

20.Перспективное преобразование и её проективная концепция.

21.Кривые Безье.

22.Простая модель освещения.

23.Закраска методом Гуро.

24.Построение В-сплайна.

25.Определение нормали к поверхности.

26.Закраска Фонга.

27.Удаление нелицевых граней многогранника.