



staff: +7(921)7527150 cgsg@yandex.ru

№	ИФК	ФИ	Класс	телефон	E-mail
01	AG4	Голод Александра	10-4	+7(921)3250586	alex.evans1999@mail.ru
02	DG5	Григорьев Денис	10-5	+7(981)9517561	denis.ru200@yandex.ru
03	AK5	Кириллова Арина	10-5	+7(951)6808836	horsegrand@yandex.ru
04	NK5	Кондрашов Никита	10-5	+7(921)8995596	nik2000408k@mail.ru
05	AH5	Худяков Андрей	10-5	+7(904)5576575	anhud0000@gmail.com

10:00 - start

0. About

WinAPI - MS-Window Application Program Interface

Windows SDK - software development kit *.h + *.lib

основной заголовочный файл - <windows.h>

1. Parameters

```
typedef struct tagNAME
{
    int x;
    float y;
    ...
} NAME;
```

2. Hungarian notation

Чарльз Симони

```
POINT ptCorner;
RECT rcWindow;
INT cxWindow, cyWindow;
BYTE bCount;
DWORD dwSize;
WORD wSize;
BOOL bFlag;
```

LP - long pointer - far pointer

NP - near pointer

3. Handles

HWND hWnd;

```
hWnd = CreateWindow(...);
if (hWnd == NULL)
    error
```

```
...
DestroyWindow(hWnd);
```

```
HBRUSH hbr;
HDC hDC;

hbr = CreateSolidBrush(RGB(255, 255, 0));
hDC = GetDC(NULL);

SelectObject(hDC, hbr);
SelectObject(hDC, GetStockObject(NULL_PEN));

Rectangle(hDC, 0, 0, 300, 159);

DeleteObject(hbr);
ReleaseDC(NULL, hDC);
```

Z:\INFO\dirinfo
Z:\SUM2017\T##*****

-= Base WinAPI pattern =-

<windows.h>

переопределены стандартные типы:

CHAR VOID INT LONG UINT ULONG BYTE WORD DWORD

функции именуются: CreateWindow CreateSolidBrush

макро-имена (константы):

- WS_*** - window style - стили окна (WS_BORDER)
- CS_*** - class style - стили класса окна (CS_DBLCLK)
- WM_*** - window message - сообщения для окон (WM_LBUTTONDOWN, WM_CREATE)
- SW_*** - show window - параметры показа окна (SW_SHOWMINIMIZED)
- PS_*** - pen style (PS_SOLID)
- SM_*** - system metric - параметры данных системы (SM_CXSCREEN)
- CW_*** - create window - параметры создания окна (CW_USEDEFAULT)

основные дескрипторы:

HINSTANCE - дескриптор приложения - "номер" программы

HWND - дескриптор окна

HDC - дескриптор контекста устройства

структуры:

WNDCLASS - описание общих параметров окон

POINT - координаты точки (целые)

MSG - структура сообщений

точка входа не 'main', а 'WinMain'

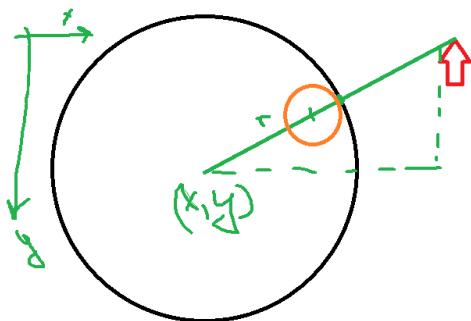
Определить позицию курсора мыши на экране и преобразовать ее в окно:

POINT pt;

```
GetCursorPos(&pt);
ScreenToClient(hWnd, &pt);
pt.x и pt.y - позиция курсора в окне
```

T01EYES

Написать программу слежения "глазами" за мышой.



Advice:

hDC

...

```
SelectObject(hDC, объект рисования);
```

```
GetStockObject(объект)
```

NULL_PEN

NULL_BRUSH

BLACK_PEN

BLACK_BRUSH

WHITE_PEN

WHITE_BRUSH

LTGRAY_BRUSH

свою кисть:

```
HBRUSH hbr, hbrOld;
```

```
hbr = CreateSolidBrush(RGB(r, g, b));
```

```
hbrOld = SelectObject(hDC, hbr);
```

рисуем ...

```
SelectObject(hDC, hbrOld);
```

```
DeleteObject(hbr);
```

```
Rectangle(hDC, x1, y1, x2, y2);
```

```
Ellipse(hDC, x1, y1, x2, y2);
```

```
MoveToEx(hDC, x1, y1, NULL);
```

```
LineTo(hDC, x2, y2);
```

```
VOID DrawEye( HDC hDC, INT Xc, INT Yc, INT R, INT X, INT Y )
```

{

```
    INT dx = X - Xc, dy = Y - Yc;
```

float

```
    len = sqrt(dx * dx + dy * dy),
```

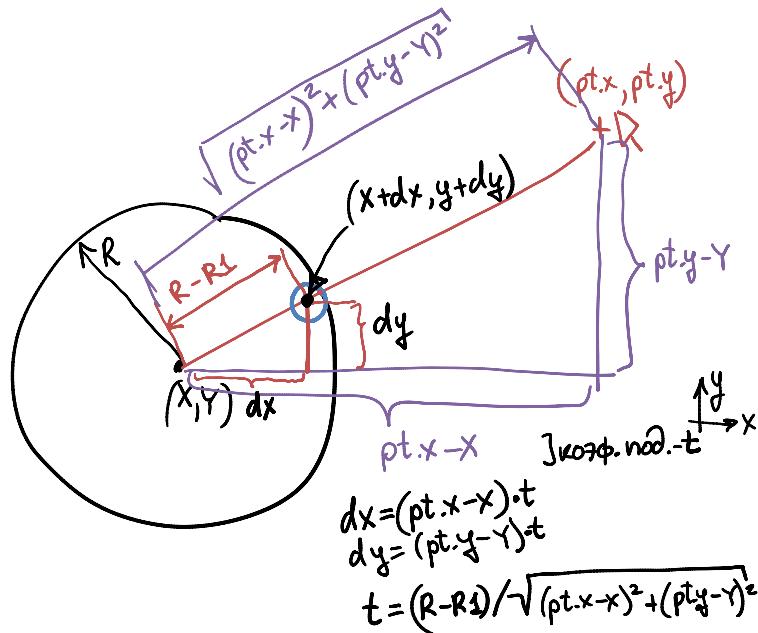
```
    si = dy / len, co = dx / len;
```

```

    . . .
}

. . .
WM_TIMER:
. . .
DrawEye(hDC, 100, 100, 30, pt.x, pt.y);
. . .

```



Д.3.:

- для системы контроля версий (привязать к E-mail)
www.github.com - завести account
www.gravatar.com - завести account
- прочитать: п.5, п.6, посмотреть функции из п.10
 (указатели, массивы, строки, структуры)

-= 02.06.2017 =-

.: 10:00 :.

подключение сетевых дисков

Х:

\server\tools

Q:

\server\shspr

копируем

Q:\0602*.vssettings

--> d:\Documents and Settings\spr**\

Мои документы\Visual Studio 2008\

Settings\...

.: 10:30 :.

Двойная буферизация

-- создать буфер кадра - картинка в памяти, совпадающая с размером окна

HBITMAP hBm = CreateCompatibleBitmap(hDC, w, h);

-- научить систему рисовать в картинку-буфер кадра

>>> создаем контекст в памяти для рисования в изображения

```
HDC hMemDC = CreateCompatibleDC(hDC); /* hDC - контекст окна */
/* задаем для контекста в памяти изображение */
SelectObject(hMemDC, hBm);

-- скопировать буфер кадра в окно
/* Bit Block Transfer */
BitBlt(hDC, 0, 0, w, h, hMemDC, 0, 0, SRCCOPY);
```

уничтожение:

```
DeleteObject(hBm);
DeleteDC(hMemDC);
```

>>> github
 kirillova.arina.ak5@gmail.com
 github/kirillova.arina.ak5
 github/ak5_2017

.::: 14:30 :::

загрузка картинок:

```
BITMAP bm;
static HDC hMemDCLogo;
static HBITMAP hBmLogo;

. . .
hDC = GetDC(hWnd);
hMemDCLogo = CreateCompatibleDC(hDC);
ReleaseDC(hWnd, hDC);

hBmLogo = LoadImage(NULL, "A.BMP", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
SelectObject(hMemDCLogo, hBmLogo);

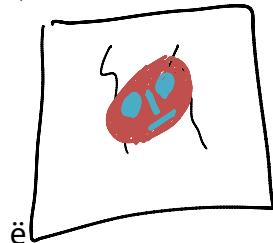
. . .
GetObject(hBmLogo, sizeof(BITMAP), &bm);
BitBlt(hDC, x, y, bm.bmWidth, bm.bmHeight,
       hMemDCLogo, 0, 0, SRCCOPY);
```

Microsoft Windows Device Independed Bitmap

*.DIB *.BMP

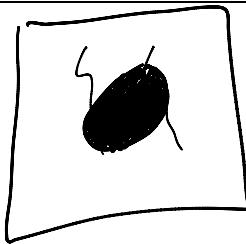
Sprites

(независимые изображения, возможно с прозрачными точками)



- изображение кодируется двумя картинками:

- AND маска - все белое (111111), зона изображения - черное (000000)



- XOR маска - само изображение (в черной зоне от AND маски - обычное, в белой - имнверсное)



T02CLOCK

-= 03.06.2017 =-

.:: 14:00 ::.

Практика T02CLOCK

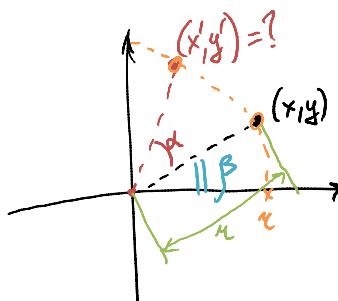
.:: 15:00 ::.

Функция Polygon

Поворот точки вокруг координат

Рисование стрелок (hand draw)

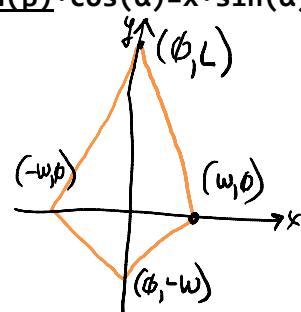
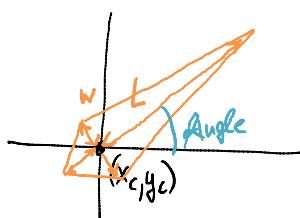
1. поворот точки вокруг начала координат



$$x = r \cdot \cos(\beta)$$

$$y = r \cdot \sin(\beta)$$

$$\begin{aligned} x' &= r \cdot \cos(\beta + \alpha) = r \cdot \cos(\beta) \cdot \cos(\alpha) - r \cdot \sin(\beta) \cdot \sin(\alpha) = x \cdot \cos(\alpha) - y \cdot \sin(\alpha) \\ y' &= r \cdot \sin(\beta + \alpha) = r \cdot \cos(\beta) \cdot \sin(\alpha) + r \cdot \sin(\beta) \cdot \cos(\alpha) = x \cdot \sin(\alpha) + y \cdot \cos(\alpha) \end{aligned}$$



2. Polygon:

```
POINT pts[] =
{
    {0, 0}, {30, 300}, {200, 20}
```

};

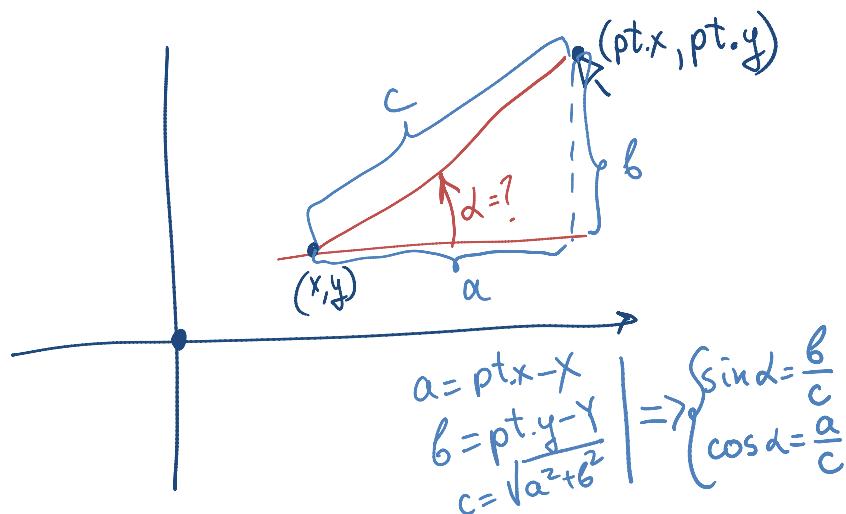
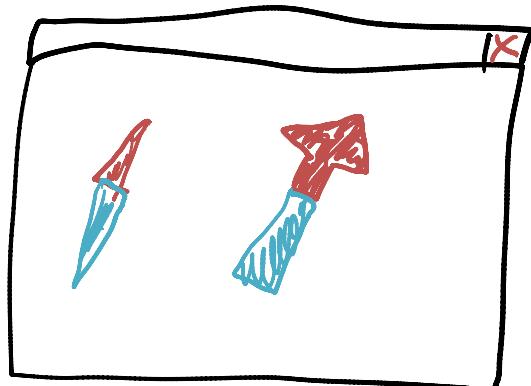
Polygon(hdc, pts, 3);

-= 05.06.2017 =-

.:: 10:55 ::.

T03POLE

Отобразить магнитное поле. Поляс притяжения - позиция курсора мыши



```
for (y = 0; y < H; y += 30)
    for (x = 0; x < W; x += 30)
        DrawArrow(hMemDC, x, y, pt.x, pt.y);
```

!!!

T03POLE.C - WinMain MyWinFunc

POLE.C - DrawArrow

POLE.H - прототипы

.:: 11:00 ::.

Практика T03POLE

```
#define ARROW_RL 50
#define ARROW_BL 13
#define ARROW_WD 18
```

```

static POINT pts[] =
{
    {-ARROW_WD, 0},
    {0, ARROW_RL},
    {ARROW_WD, 0},
    {0, -ARROW_BL},
    {-ARROW_WD, 0},
};

POINT pts_res[sizeof(pts) / sizeof(pts[0])];

/* Transform points */
for (i = 0; i < sizeof(pts) / sizeof(pts[0]); i++)
{
    pts_res[i].x = pts[i].x * si + pts[i].y * co + Xc;
    pts_res[i].y = -(pts[i].x * co - pts[i].y * si) + Yc;
}

SetDCBrushColor(hDC, RGB(255, 0, 0));
Polygon(hDC, pts_res, 3);

SetDCBrushColor(hDC, RGB(23, 30, 255));
Polygon(hDC, pts_res + 2, 3);

```

.:: 12:45 ::.

Полноэкранный режим

.:: 14:40 ::.

Linear Algebra

1. Common definitions

Пример умножения матриц:

$$\begin{pmatrix} 1 & 2 & 0 & 1 \\ 1 & 4 & 5 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 2 \\ 2 & 0 & 1 \\ 2 & 3 & 1 \\ 3 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 8 & 2 & 4 \\ 19 & 16 & 11 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

Матрица

Квадратная Матрица

Нулевая Матрица

Единичная Матрица

Диагональная Матрица

сложение (addition/add/sum)

умножение на число (numeric product)

умножение матриц (matrix multiplication)

транспонирование (transpose)

2. определитель матрицы (Matrix Determinator)**2.1. перестановки - permutation**

1..n

их количество – n!

(1, 2, 3, 4, ..., n) – основная перестановка

(2, 1, 3, 4, ..., n)

(2, 3, 1, 4, ..., n)

...

(n, n-1, n-2, ..., 1)

Все перестановка из 3 чисел:

1,2,3

3,1,2

2,3,1

1,3,2

2,1,3

3,2,1

Лексикографический порядок:

1,2,3

1,3,2

2,1,3

2,3,1

3,1,2

3,2,1

транспозиция – transposition1,2,3 -> 1,3,21,2,3 -> 3,2,1**инверсия – inversion** - больший предшествует меньшему (лексиграфический порядок)

выпишем количество инверсия для всех перестановок из 3 чисел

1,2,3 -> число инверсий 0

1,3,2 -> число инверсий 1 (3,2)

2,1,3 -> число инверсий 1 (2,1)

2,3,1 -> число инверсий 2 (2,1 3,1)

3,1,2 -> число инверсий 2 (3,1 3,2)

3,2,1 -> число инверсий 3 (3,2 3,1 2,1)

четность перестановки == четность числа инверсий

Th.

при транспозиции четность перестановки меняется на противоположную:

1. транспозиция соседних элементов

 $A_1, A_2, \dots, A_{i+1}, A_{i+2}, \dots, A_n$ $A_1, A_2, \dots, A_{i+2}, A_{i+1}, \dots, A_n$ $A_{i+1} > A_{i+2}$ число инверсий при A_{i+2}, A_{i+1} -- $A_{i+1} < A_{i+2}$ число инверсий при A_{i+2}, A_{i+1} ++

2. транспозиция произвольных элементов:

 $A_1, A_2, \dots, A_i, A_{i+1}, A_{i+2}, \dots, A_j, A_{j+1}, A_{j+2}, \dots, A_n$

- за ($j-i$) раз переставим A_i в позицию j :

$A_1, A_2, \dots, A_{i+1}, A_{i+2}, \dots, A_i, A_j, A_{j+1}, A_{j+2}, \dots, A_n$

- обмен соседей:

$A_1, A_2, \dots, A_{i+1}, A_{i+2}, \dots, A_j, A_i, A_{j+1}, A_{j+2}, \dots, A_n$

- за ($j-i$) раз переставим A_j назад:

$A_1, A_2, \dots, A_j, A_{i+1}, A_{i+2}, \dots, A_i, A_{j+1}, A_{j+2}, \dots, A_n$

итого - $2 \cdot (j - i) + 1$ - транспозиций соседей, нечетное число

T04PERM

получить все перестановки из n элементов и их четность

>>> подзадача - перестановки получить в лексикографическом порядке

Рекомендации:

1.

12345

12354

12435

12453

12534

12543

13245

...

2.

обмен значениями двух переменных:

```
void Swap( int *A, int *B )
{
    int tmp = *A;

    *A = *B;
    *B = tmp;
}

void main( void )
{
    int p = ..., q = ...;

    Swap(&p, &q);
    ...
}
```

3.

макет:

изначально (заполняем массив P основной перестановкой):

```
for (i = 0; i < N; i++)
    P[i] = i + 1;
```

поочередно меняем элементы в массиве P :

```

for (i = 0; i < N; i++)
{
    Swap(&P[0], &P[i]);
    for (j = 1; j < N; j++)
    {
        Swap(&P[1], &P[j]);
        for (k = 2; k < N; k++)
        {
            Swap(&P[2], &P[k]);
            . . .
        }
    }
}

```

!!! переменное количество вложенных конструкций реализуется с помощью рекурсии – вызова из функции самой себя.

Рекурсия состоит из 2 атрибутов:

- выход
- рекурсивный вызов

у нас:

```

void Go( int Pos )
{
    if (Pos >= N)
        return;
    for (i = Pos; i < N; i++)
    {
        ставим на позицию Pos по очереди каждый i-й элемент
        Swap(&P[Pos], &P[i]); <-- четность меняется на противопол. (кроме Pos == i)
        Go(Pos + 1);
        возвращаем назад:
        Swap(&P[Pos], &P[i]); <-- четность меняется на противопол. (кроме Pos == i)
    }
}

```

В первый раз функцию вызываем:

`Go(0);`

четность – parity

четный – even

нечетный – odd

пример функции вывода (`<stdio.h>`)

+

```

void PrintPerm( void )
{
    int i;
    FILE *F;

    F = fopen("PERM.TXT", "a"); /* append */
    if (F == NULL)
        return;
    for (i = 0; i < N; i++)
        fprintf(F, "%i", P[i]);
    fprintf(F, " - %s\n", parity ? "odd" : "even");
    fclose(F);
}

```

}

Lexicographics order:**1 2 3 4 5****2 1 3 4 5****3 1 2 4 5****4 1 2 3 5****5 1 2 3 4**

и все возвращаю назад:

<1<2<3<45**+ восстановить четность**

pos

```
* * * a b c d e f
* * * b a c d e f
* * * c a b d e f
* * * d a b c e f
```

. . .

... + возврат!!!

-->

* * * a b c d e f

-= 06.06.2017 =-

.: 10:00 :.

Практика: Т05DET

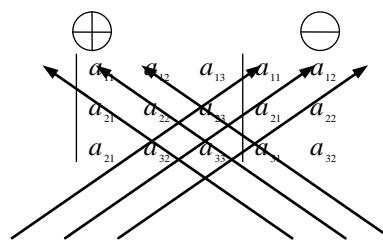
.: 11:05 :.

Свойства определителя

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = (-1)^{J(1,2)} a_{11} \cdot a_{22} + (-1)^{J(2,1)} \cdot a_{12} \cdot a_{21} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \cdot a_{22} \cdot a_{33} - a_{11} \cdot a_{23} \cdot a_{32} + a_{12} \cdot a_{23} \cdot a_{31} - a_{12} \cdot a_{21} \cdot a_{33} + a_{13} \cdot a_{21} \cdot a_{32} - a_{13} \cdot a_{22} \cdot a_{31}$$

Саррюс:



$$\begin{vmatrix} 1 & 2 & 5 \\ 0 & 30 & 9 \\ 0 & 0 & 5 \end{vmatrix} = 150$$

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 5 \end{vmatrix} = 150$$

11 23 32
11 32 23 -->

$$\begin{aligned} & \left| \begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{i1} + b_{i1} & a_{i2} + b_{i2} & \dots & a_{in} + b_{in} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{array} \right| = \sum_{k=1}^{n!} (-1)^{J(\alpha_1, \alpha_2, \dots, \alpha_n)} \cdot a_{1\alpha_1} \cdot a_{2\alpha_2} \cdot \dots \cdot (a_{i\alpha_i} + b_{i\alpha_i}) \cdot \dots \cdot a_{n\alpha_n} = \\ & = \sum_{i=1}^n (-1)^{J(\alpha_1, \alpha_2, \dots, \alpha_n)} \cdot a_{1\alpha_1} \cdot a_{2\alpha_2} \cdot \dots \cdot a_{i\alpha_i} \cdot \dots \cdot a_{n\alpha_n} + \sum_{i=1}^n (-1)^{J(\alpha_1, \alpha_2, \dots, \alpha_n)} \cdot a_{1\alpha_1} \cdot a_{2\alpha_2} \cdot \dots \cdot b_{i\alpha_i} \cdot \dots \cdot a_{n\alpha_n} = \\ & = \left| \begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{in} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{array} \right| + \left| \begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ b_{i1} & b_{i2} & \dots & b_{in} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{array} \right| \end{aligned}$$

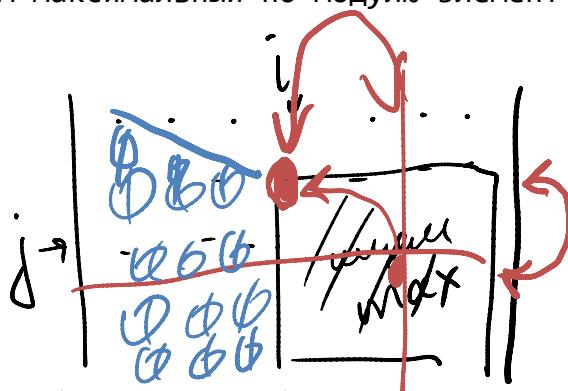
$$\begin{vmatrix} 4 & 5 \\ 2 & 1 \end{vmatrix} = \begin{vmatrix} 2+2 & 1+4 \\ 2 & 1 \end{vmatrix} = 4 \cdot 1 - 5 \cdot 2 = (2+2) \cdot 1 - (1+4) \cdot 2 = \begin{vmatrix} 2 & 1 \\ 2 & 1 \end{vmatrix} + \begin{vmatrix} 2 & 4 \\ 2 & 1 \end{vmatrix} = 2 \cdot 1 - 1 \cdot 2 + 2 \cdot 1 - 4 \cdot 2 = -6$$

$$\begin{vmatrix} 1 & 2 & 4 \\ 2 & 2 & 1 \\ 1 & 2 & 3 \end{vmatrix} = [2] - 2 \cdot [1], [3] - [1] = \begin{vmatrix} 1 & 2 & 4 \\ 0 & -2 & -7 \\ 0 & 0 & -1 \end{vmatrix} = 2$$

T05DET + вычисление определителя методом Гаусса

для каждого j-го столбца

находим максимальный по модулю элемент в матрице от (j,j) до (n,n)



нашли $\rightarrow (\max_i, \max_j)$

меняем \max_i строку с j-й строкой и \max_j столбец с j-м столбцом

избавляемся от элементов в j-й столбце в строках от j+1 до n:
вычитаем из каждой k-й строки:

```
for (i = j; i < n; i++)
    A[k][i] -= A[j][i] * (A[k][j] / A[j][j]);
        === coef
```

!!! не забываем учитывать знак при обмене строк и столбцов после поиска минимума

max:

```
max_i = j;
max_j = j;
for (ii = j; ii < n; ii++)
    for (jj = j; jj < n; jj++)
        if (fabs(A[max_i][max_j]) < fabs(A[ii][jj]))
            max_i = ii, max_j = jj;
```

Swap:

```
if (max_i != j)
{
    sign = -sign;
    for (i = j; i < n; i++)
        Swap(&A[max_i][i], &A[j][i]);
}
if (max_j != j)
{
    sign = -sign;
    for (i = j; i < n; i++)
        Swap(&A[i][max_j], &A[i][j]);
}
```

```
det = 1;
for (i = 0; i < N; i++)
{
    /* look for maximum matrix element -> j row k column */
    j = k = i;
    for (y = i; y < N; y++)
        for (x = i; x < N; x++)
            if (fabs(A[y][x]) > fabs(A[j][k]))
                k = x, j = y;

    if (A[j][k] == 0)
        return 0;

    if (j != i)
    {
        /* Swap j row and i row (elements [i..N-1]) */
        for (x = i; x < N; x++)
            Swap(&A[i][x], &A[j][x]);
        det = -det;
    }
    if (k != i)
    {
        /* Swap k column and i column (elements [0..N-1]) */
        for (y = 0; y < N; y++)
            Swap(&A[y][i], &A[y][k]);
        det = -det;
    }

    /* Subtract from every row [i+1..N-1] i row multiplied by A[j][i] / A[i][i] */
    for (j = i + 1; j < N; j++)
    {
        coef = A[j][i] / A[i][i];
        for (k = i; k < N; k++)
```

```

    A[j][k] -= A[i][k] * coef;
}
}
for (i = 0; i < N; i++)
det *= A[i][i];

```

.: 16:00 ::.

Element Adjoyment (+ Element Minor)

$$\mathbf{A}_{ij} = \begin{vmatrix} a_{11} & \dots & a_{1,j-1} & 0 & a_{1,j+1} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,1} & \dots & a_{i-1,j-1} & 0 & a_{i-1,j+1} & \dots & a_{i-1,n} \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ a_{i+1,1} & \dots & a_{i+1,j-1} & 0 & a_{i+1,j+1} & \dots & a_{i+1,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n,1} & \dots & a_{n,j-1} & 0 & a_{n,j+1} & \dots & a_{n,n} \end{vmatrix}$$

$$\begin{aligned}
& \begin{vmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & a_{i3} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix} = \\
& = \begin{vmatrix} a_{11} & & a_{12} & & a_{13} & & \dots & & a_{1n} \\ \dots & & \dots & & \dots & & \dots & & \dots \\ a_{i1} + 0 + 0 \dots + 0 & 0 + a_{i2} + 0 + \dots + 0 & 0 + 0 + a_{i3} + \dots + 0 & \dots & 0 + 0 + 0 + \dots + a_{in} \\ \dots & \dots & \dots & \dots & \dots \\ a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{i1} & 0 & 0 & \dots & 0 & 0 & a_{i2} & 0 & \dots & 0 \\ \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix} = \\
& = \begin{vmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{11} & a_{12} & a_{13} & \dots & a_{1n} & a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ \dots & \dots \\ a_{i1} & 0 & 0 & \dots & 0 & 0 & a_{i2} & 0 & \dots & 0 & 0 & a_{i3} & 0 & \dots & 0 \\ \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix} + \dots \\
& + \begin{vmatrix} a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix} = \\
& = a_{i1} \cdot \begin{vmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & 0 \end{vmatrix} + a_{i2} \cdot \begin{vmatrix} a_{11} & 0 & a_{13} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & \dots & 0 \end{vmatrix} + a_{i3} \cdot \begin{vmatrix} a_{11} & a_{12} & 0 & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 1 & \dots & 0 \end{vmatrix} + \dots + \\
& a_{in} \cdot \begin{vmatrix} a_{11} & a_{12} & a_{13} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & 0 \end{vmatrix} = a_{i1} \cdot A_{i1} + a_{i2} \cdot A_{i2} + a_{i3} \cdot A_{i3} + \dots + a_{in} \cdot A_{in}
\end{aligned}$$

$$\mathbf{M}_{ij} = \begin{vmatrix} a_{11} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i-1,1} & \cdots & a_{i-1,j-1} & a_{i-1,j+1} & \cdots & a_{i-1,n} \\ a_{i+1,1} & \cdots & a_{i+1,j-1} & a_{i+1,j+1} & \cdots & a_{i+1,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & \cdots & a_{n,j-1} & a_{n,j+1} & \cdots & a_{n,n} \end{vmatrix}$$

$$A_{ij} = \begin{vmatrix} a_{11} & \cdots & 0 & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & \cdots & 0 & \cdots & a_{nn} \end{vmatrix} =$$

меняя столбцы: j с $j+1$, $j+1$ с $j+2$, ..., $n-1$ с n "загоним" 1 в правый столбец
(знак определителя менялся $n-j-1$ раз)

меняя строки: i с $i+1$, $i+1$ с $i+2$, ..., $n-1$ с n "загоним" 1 в нижнюю строки
(знак определителя менялся $n-i-1$ раз)

получаем определитель матрицы:

$$= (-1)^{n-j-1+n-i-1} \cdot \begin{vmatrix} a_{11} & \cdots & a_{1,j-1} & a_{1,j+1} & \cdots & a_{1n} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i-1,1} & \cdots & a_{i-1,j-1} & a_{i-1,j+1} & \cdots & a_{i-1,n} & 0 \\ a_{i+1,1} & \cdots & a_{i+1,j-1} & a_{i+1,j+1} & \cdots & a_{i+1,n} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{n,j-1} & a_{n,j+1} & \cdots & a_{nn} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix} = (-1)^{j+i} \cdot M_{ij}$$

пример:

$$\begin{vmatrix} 1 & 2 & 4 \\ 2 & 2 & 1 \\ 1 & 2 & 3 \end{vmatrix} = \text{по первой строке} = \\ = (-1)^{1+1} \cdot 1 \cdot \begin{vmatrix} 2 & 1 \\ 2 & 3 \end{vmatrix} + (-1)^{1+2} \cdot 2 \cdot \begin{vmatrix} 2 & 1 \\ 1 & 3 \end{vmatrix} + (-1)^{1+3} \cdot 4 \cdot \begin{vmatrix} 2 & 2 \\ 1 & 2 \end{vmatrix} = 4 - 10 + 8 = 2$$

Обратная матрица (Inverse Matrix).

$$A_n \cdot B_n = E_n$$

$$B_n \cdot A_n = E_n$$

Введем в рассмотрение присоединенную матрицу
(adjoint matrix - Adj A)

$$\text{adj } A = \begin{pmatrix} A_{11} & A_{21} & A_{31} & \dots & A_{n1} \\ A_{12} & A_{22} & A_{32} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & A_{3n} & \dots & A_{nn} \end{pmatrix}$$

$$A \cdot \text{adj } A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} A_{11} & A_{21} & A_{31} & \dots & A_{n1} \\ A_{12} & A_{22} & A_{32} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & A_{3n} & \dots & A_{nn} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11} \cdot A_{11} + a_{12} \cdot A_{12} + \dots + a_{1n} \cdot A_{1n} & a_{11} \cdot A_{21} + a_{12} \cdot A_{22} + \dots + a_{1n} \cdot A_{2n} & \dots & a_{11} \cdot A_{n1} + a_{12} \cdot A_{n2} + \dots + a_{1n} \cdot A_{nn} \\ a_{21} \cdot A_{11} + a_{22} \cdot A_{12} + \dots + a_{2n} \cdot A_{1n} & a_{21} \cdot A_{21} + a_{22} \cdot A_{22} + \dots + a_{2n} \cdot A_{2n} & \dots & a_{21} \cdot A_{n1} + a_{22} \cdot A_{n2} + \dots + a_{2n} \cdot A_{nn} \\ \dots & \dots & \dots & \dots \\ a_{n1} \cdot A_{11} + a_{n2} \cdot A_{12} + \dots + a_{nn} \cdot A_{1n} & a_{n1} \cdot A_{21} + a_{n2} \cdot A_{22} + \dots + a_{nn} \cdot A_{2n} & \dots & a_{n1} \cdot A_{n1} + a_{n2} \cdot A_{n2} + \dots + a_{nn} \cdot A_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} |A| & 0 & \dots & 0 \\ 0 & |A| & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & |A| \end{pmatrix} = |A| \cdot E$$

$$A \cdot \text{adj } A = |A| \cdot E \text{ аналогично } \text{adj } A \cdot A = |A| \cdot E$$

$$A^{-1} = \frac{1}{|A|} \cdot \text{adj } A$$

$$\begin{pmatrix} 1 & 2 & 4 \\ 2 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}^{-1} = \frac{1}{2} \cdot \begin{pmatrix} \begin{vmatrix} 2 & 1 \\ 2 & 3 \end{vmatrix} & -\begin{vmatrix} 2 & 4 \\ 2 & 3 \end{vmatrix} & \begin{vmatrix} 2 & 4 \\ 2 & 1 \end{vmatrix} \\ -\begin{vmatrix} 2 & 1 \\ 1 & 3 \end{vmatrix} & \begin{vmatrix} 1 & 4 \\ 1 & 3 \end{vmatrix} & -\begin{vmatrix} 1 & 4 \\ 2 & 1 \end{vmatrix} \\ \begin{vmatrix} 2 & 2 \\ 1 & 2 \end{vmatrix} & -\begin{vmatrix} 1 & 2 \\ 1 & 2 \end{vmatrix} & \begin{vmatrix} 1 & 2 \\ 2 & 2 \end{vmatrix} \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} 4 & 2 & -6 \\ -5 & -1 & 7 \\ 2 & 0 & -2 \end{pmatrix}$$

проверка:

$$\frac{1}{2} \cdot \begin{pmatrix} 4 & 2 & -6 \\ -5 & -1 & 7 \\ 2 & 0 & -2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 4 \\ 2 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{Ответ: } A^{-1} = \begin{pmatrix} 2 & 1 & -3 \\ -2.5 & -0.5 & 3.5 \\ 1 & 0 & -1 \end{pmatrix}$$

.::: 16:35 :::

Задача самостоятельно: $A = \begin{pmatrix} 1 & 2 & 4 \\ 4 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$, найти $A^{-1} = ?$

$$\text{Ответ} = \frac{1}{5} \cdot \begin{pmatrix} -1 & 2 & -2 \\ -1 & -3 & 13 \\ 2 & 1 & -6 \end{pmatrix}$$

2.5. Применение – системы линейных алгебраических уравнений – СЛАУ

$$\begin{cases} x_1 \cdot a_{11} + x_2 \cdot a_{12} + \cdots + x_n \cdot a_{1n} = b_1 \\ x_1 \cdot a_{21} + x_2 \cdot a_{22} + \cdots + x_n \cdot a_{2n} = b_2 \\ \dots \\ x_1 \cdot a_{n1} + x_2 \cdot a_{n2} + \cdots + x_n \cdot a_{nn} = b_n \end{cases}$$

перепишем в матричном виде: $A_{nn} \cdot X_{n1} = B_{n1}$

слева и справа домножим на A_{nn}^{-1}

$$A_{nn}^{-1} \cdot A_{nn} \cdot X_{n1} = A_{nn}^{-1} \cdot B_{n1}$$

$$X_{n1} = A_{nn}^{-1} \cdot B_{n1}$$

т.е.

$$X_{n1} = \frac{1}{|A|} \cdot \text{adj } A \cdot B_{n1}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{pmatrix} = \frac{1}{|A|} \cdot \begin{pmatrix} A_{11} & A_{21} & A_{31} & \dots & A_{n1} \\ A_{12} & A_{22} & A_{32} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & A_{3n} & \dots & A_{nn} \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{pmatrix}$$

$$= \frac{1}{|A|} \cdot \begin{pmatrix} b_1 \cdot A_{11} + b_2 \cdot A_{21} + b_3 \cdot A_{31} + \dots + b_n \cdot A_{n1} \\ b_1 \cdot A_{12} + b_2 \cdot A_{22} + b_3 \cdot A_{32} + \dots + b_n \cdot A_{n2} \\ b_1 \cdot A_{13} + b_2 \cdot A_{23} + b_3 \cdot A_{33} + \dots + b_n \cdot A_{n3} \\ \dots \\ b_1 \cdot A_{1n} + b_2 \cdot A_{2n} + b_3 \cdot A_{3n} + \dots + b_n \cdot A_{nn} \end{pmatrix}$$

каждая строка – разложение по столбцу:

т.е., рассмотрим i -ю строку (значение x_i):

$$x_i = \frac{1}{|A|} \cdot \begin{vmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,i-1} & b_2 & a_{2,i+1} & \dots & a_{2,n} \\ a_{3,1} & a_{3,2} & \dots & a_{3,i-1} & b_3 & a_{3,i+1} & \dots & a_{3,n} \\ \dots & \dots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,i-1} & b_{n-1} & a_{n-1,i+1} & \dots & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{n,n} \end{vmatrix}$$

пример – система:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 = b_2 \end{cases}$$

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}$$

$$x_2 = \frac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}$$

самостоятельно:

$$x + y = 5$$

$$x - 3 * y = -7$$

$$A = \begin{pmatrix} 1 & 1 \\ 1 & -3 \end{pmatrix}, B = \begin{pmatrix} 5 \\ -7 \end{pmatrix}$$

решение:

$$|A| = -4, x = \frac{\begin{vmatrix} 5 & 1 \\ -7 & -3 \end{vmatrix}}{|A|} = \frac{-8}{-4} = 2, y = \frac{\begin{vmatrix} 1 & 5 \\ 1 & -7 \end{vmatrix}}{|A|} = \frac{-12}{-4} = 3$$

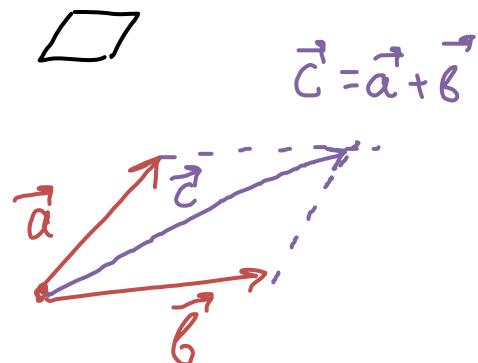
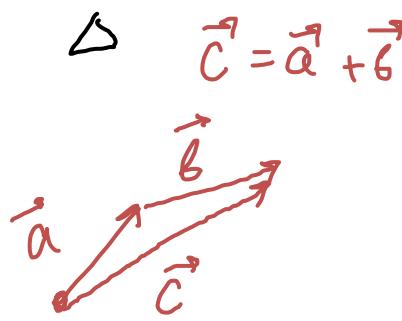
.: 17:00 ::.

Практика T06GAUSS – решение СЛАУ

-= 07.06.2017 =-

Analitical Geometry**Векторы**

сложение | умножение на число



коллинеарность | компланарность | ортогональность | линейная зависимость

линейная зависимость и независимость векторов

- линейная комбинация:

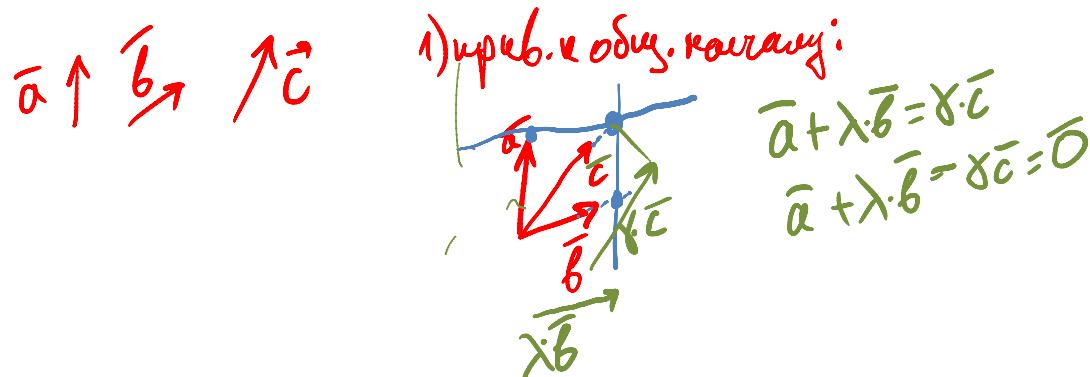
$$\lambda_1 \cdot \underline{\vec{a}_1} + \lambda_2 \cdot \underline{\vec{a}_2} + \dots + \lambda_n \cdot \underline{\vec{a}_n}$$

$$\lambda_1 \cdot \vec{a}_1 + \dots + \lambda_n \cdot \vec{a}_n$$

тривиальная

не все $\lambda_i = 0$ - нетривиальная ($\sum_{i=1}^n \lambda_i^2 \neq 0$)

$$\lambda_1 \cdot \underline{\vec{a}_1} + \lambda_2 \cdot \underline{\vec{a}_2} + \dots + \lambda_n \cdot \underline{\vec{a}_n} = \underline{0}$$

Th. Любые три вектора на плоскости линейно зависимы

Скалярное произведение векторов (dot product)

обозначение – $\vec{a} \cdot \vec{b}$ или (\vec{a}, \vec{b})

Это $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos \vec{a}, \vec{b}$

Рассмотрим ортонормированный базис:

$$\vec{e}_1, \vec{e}_2$$

ортонормированный базис: $|\vec{e}_1| = |\vec{e}_2| = 1, \vec{e}_1 \cdot \vec{e}_2 = 0$

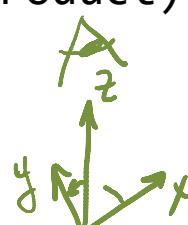
$$\begin{aligned}\vec{a} &= a_1 \cdot \vec{e}_1 + a_2 \cdot \vec{e}_2 \\ \vec{b} &= b_1 \cdot \vec{e}_1 + b_2 \cdot \vec{e}_2\end{aligned}$$

$$\begin{aligned}\vec{a} \cdot \vec{b} &= (a_1 \cdot \vec{e}_1 + a_2 \cdot \vec{e}_2) \cdot (b_1 \cdot \vec{e}_1 + b_2 \cdot \vec{e}_2) = \\ &= a_1 \cdot \vec{e}_1 \cdot b_1 \cdot \vec{e}_1 + a_2 \cdot \vec{e}_2 \cdot b_1 \cdot \vec{e}_1 + a_1 \cdot \vec{e}_1 \cdot b_2 \cdot \vec{e}_2 + a_2 \cdot \vec{e}_2 \cdot b_2 \cdot \vec{e}_2 = \\ &= a_1 \cdot b_1 + a_2 \cdot b_2\end{aligned}$$

Векторное произведение векторов (cross product)

1. ориентация троек - handhandled

- правая тройка (right hand vectors)
- левая тройка (left hand vectors)



counter-clock-wise - против часовой стрелки (CCW)

clock-wise - по часовой стрелки (CW)

2. само векторное произведение:

$$\vec{a} \times \vec{b} = \vec{c}$$

при этом

$$\vec{c} \perp \vec{a}$$

$$\vec{c} \perp \vec{b}$$

$$|\vec{c}| = |\vec{a}| \cdot |\vec{b}| \cdot \sin \widehat{\vec{a}, \vec{b}}$$

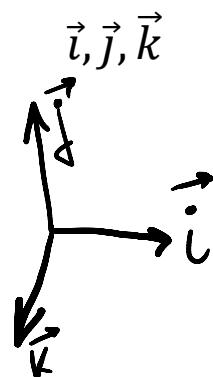
$(\vec{a}, \vec{b}, \vec{c})$ – правая тройка

$$!!! \quad \vec{a} \times \vec{a} = \vec{0}$$

$$!!! \quad \vec{a} \cdot \vec{a} = |\vec{a}|^2$$

Рассмотрим ортонормированный базис:

(правая тройка)



$$|\vec{i}| = |\vec{j}| = |\vec{k}| = 1,$$

$$\vec{i} \cdot \vec{j} = 0, \vec{i} \cdot \vec{k} = 0, \vec{j} \cdot \vec{k} = 0, \vec{i} \cdot \vec{i} = 1, \vec{j} \cdot \vec{j} = 1, \vec{k} \cdot \vec{k} = 1$$

$$\vec{i} \times \vec{j} = \vec{k}$$

$$\vec{j} \times \vec{i} = -\vec{k}$$

$$\vec{j} \times \vec{k} = \vec{i}$$

$$\vec{k} \times \vec{j} = -\vec{i}$$

$$\vec{k} \times \vec{i} = \vec{j}$$

$$\vec{i} \times \vec{k} = -\vec{j}$$

$$\vec{a} = a_1 \cdot \vec{i} + a_2 \cdot \vec{j} + a_3 \cdot \vec{k}$$

$$\vec{b} = b_1 \cdot \vec{i} + b_2 \cdot \vec{j} + b_3 \cdot \vec{k}$$

$$\vec{a} \times \vec{b} = (a_1 \cdot \vec{i} + a_2 \cdot \vec{j} + a_3 \cdot \vec{k}) \times (b_1 \cdot \vec{i} + b_2 \cdot \vec{j} + b_3 \cdot \vec{k}) =$$

$$(a_1 \cdot \vec{i} \times b_1 \cdot \vec{i} + a_2 \cdot \vec{j} \times b_1 \cdot \vec{i} + a_3 \cdot \vec{k} \times b_1 \cdot \vec{i}) +$$

$$(a_1 \cdot \vec{i} \times b_2 \cdot \vec{j} + a_2 \cdot \vec{j} \times b_2 \cdot \vec{j} + a_3 \cdot \vec{k} \times b_2 \cdot \vec{j}) +$$

$$(a_1 \cdot \vec{i} \times b_3 \cdot \vec{k} + a_2 \cdot \vec{j} \times b_3 \cdot \vec{k} + a_3 \cdot \vec{k} \times b_3 \cdot \vec{k}) =$$

$$a_2 \cdot \vec{j} \times b_1 \cdot \vec{i} + a_3 \cdot \vec{k} \times b_1 \cdot \vec{i} + a_1 \cdot \vec{i} \times b_2 \cdot \vec{j} + a_3 \cdot \vec{k} \times b_2 \cdot \vec{j} + a_1 \cdot \vec{i}$$

$$\times b_3 \cdot \vec{k} + a_2 \cdot \vec{j} \times b_3 \cdot \vec{k} =$$

$$a_2 \cdot b_1 \cdot (-\vec{k}) + a_3 \cdot b_1 \cdot (\vec{j}) + a_1 \cdot b_2 \cdot (\vec{k}) + a_3 \cdot b_2 \cdot (-\vec{i}) + a_1 \cdot b_3$$

$$\cdot (-\vec{j}) + a_2 \cdot b_3 \cdot (\vec{i}) =$$

$$\vec{i} \cdot (a_2 \cdot b_3 - a_3 \cdot b_2) + \vec{j} \cdot (a_3 \cdot b_1 - a_1 \cdot b_3) + \vec{k} \cdot (a_1 \cdot b_2 - a_2 \cdot b_1) =$$

$$\vec{i} \cdot \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} - \vec{j} \cdot \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} + \vec{k} \cdot \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

Смешанное произведение (в ортонормированном базисе):

$$(\vec{a} \times \vec{b}) \cdot \vec{c} = \left(\begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}, - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix}, \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \right) \cdot (c_1, c_2, c_3) =$$

$$c_1 \cdot \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} - c_2 \cdot \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} + c_3 \cdot \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}$$

знак определителя > 0 , если $(\vec{a}, \vec{b}, \vec{c})$ - правая тройка

Двойное векторное произведение:

$$\vec{a} \times (\vec{b} \times \vec{c}) = \vec{b} \cdot (\vec{a} \cdot \vec{c}) - \vec{c} \cdot (\vec{a} \cdot \vec{b})$$

Примеры вычислений для ортонормированных базисов

Смешанное произведение:

$$(\vec{P} \times \vec{Q}) \cdot \vec{R} = \begin{vmatrix} p_x & p_y & p_z \\ q_x & q_y & q_z \\ r_x & r_y & r_z \end{vmatrix} \quad (>0 \text{ если тройка векторов } \vec{P}, \vec{Q}, \vec{R} \text{ - правая, } <0 \text{ - левая})$$

Двойное векторное произведение:

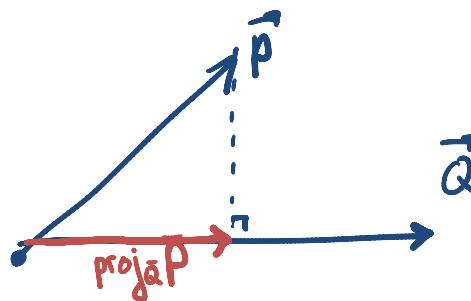
$$\vec{a} \times (\vec{b} \times \vec{c}) = \vec{b} \cdot (\vec{a} \cdot \vec{c}) - \vec{c} \cdot (\vec{a} \cdot \vec{b})$$

частный случай:

$$\vec{P} \times (\vec{Q} \times \vec{P}) = \vec{P} \times (-\vec{P} \times \vec{Q}) = -(-\vec{P} \times \vec{Q}) \times \vec{P} = (\vec{P} \times \vec{Q}) \times \vec{P}$$

дополнительно

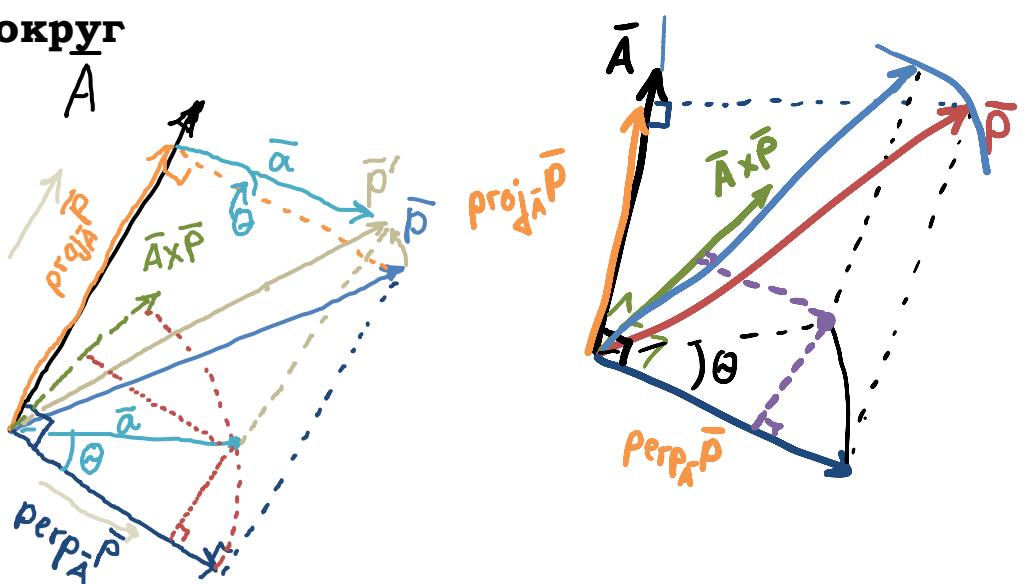
$$\vec{P} \times (\vec{Q} \times \vec{P}) = \vec{P}^2 \cdot \vec{Q} - (\vec{P} \cdot \vec{Q}) \cdot \vec{P}$$

Проектирование:

$$\begin{aligned} \text{proj}_{\vec{Q}} \vec{P} &= \frac{(\vec{P} \cdot \vec{Q})}{\|\vec{Q}\|} \vec{Q} = \frac{1}{\|\vec{Q}\|} \cdot (p_x q_x + p_y q_y + p_z q_z) \cdot \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix} = \\ &= \frac{1}{\|\vec{Q}\|} \cdot \begin{pmatrix} p_x q_x^2 + p_y q_x q_y + p_z q_z q_x \\ p_x q_x q_y + p_y q_y^2 + p_z q_z q_y \\ p_x q_x q_z + p_y q_y q_z + p_z q_z^2 \end{pmatrix} = \\ &= \frac{1}{\|\vec{Q}\|} \cdot \begin{pmatrix} p_x q_x q_x + p_y q_y q_x + p_z q_z q_x \\ p_x q_x q_y + p_y q_y q_y + p_z q_z q_y \\ p_x q_x q_z + p_y q_y q_z + p_z q_z q_z \end{pmatrix} = \\ &= \frac{1}{\|\vec{Q}\|} \cdot \begin{pmatrix} q_x^2 & q_y q_x & q_z q_x \\ q_x q_y & q_y^2 & q_z q_y \\ q_x q_z & q_y q_z & q_z^2 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \end{aligned}$$

Векторное произведение:

$$\vec{P} \times \vec{Q} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ p_x & p_y & p_z \\ q_x & q_y & q_z \end{vmatrix} = \begin{pmatrix} p_y q_z - p_z q_y \\ p_z q_x - p_x q_z \\ p_x q_y - p_y q_x \end{pmatrix} = \begin{pmatrix} 0 & q_z & -q_y \\ -q_z & 0 & q_x \\ q_y & -q_x & 0 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$$

Поворот вектора вокруг произвольной оси.

] \vec{A} — единичный вектор, вокруг которого вращаем вектор \vec{P} на угол θ .

Запишем \vec{P} как сумму двух перпендикулярных векторов:

$$\vec{P} = \text{proj}_{\vec{A}}\vec{P} + \text{perp}_{\vec{A}}\vec{P}$$

После поворота $\text{proj}_{\vec{A}}\vec{P}$ не изменится, а вектор $\text{perp}_{\vec{A}}\vec{P}$ повернется в плоскости вектора $\text{perp}_{\vec{A}}\vec{P}$ и перпендикуляра к векторам $\text{proj}_{\vec{A}}\vec{P}$ и $\text{perp}_{\vec{A}}\vec{P}$ (он равен $\vec{A} \times \vec{P}$) на угол θ .

Итоговый вектор \vec{P}' равен:

$$\begin{aligned}\vec{P}' &= \text{proj}_{\vec{A}}\vec{P} + (\vec{A} \times \vec{P}) \cdot \sin \theta + \text{perp}_{\vec{A}}\vec{P} \cdot \cos \theta = \text{proj}_{\vec{A}}\vec{P} + (\vec{A} \times \vec{P}) \cdot \sin \theta + (\vec{P} - \text{proj}_{\vec{A}}\vec{P}) \cdot \cos \theta \\ &= \vec{P} \cdot \cos \theta + \text{proj}_{\vec{A}}\vec{P} \cdot (1 - \cos \theta) + (\vec{A} \times \vec{P}) \cdot \sin \theta\end{aligned}$$

перепишем последнюю формулу в матричном виде для вынесения \vec{P} за скобки ($\vec{P} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$):

$$\begin{aligned}&\left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \cos \theta \right) \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \left(\frac{1}{\|\vec{A}\|} \cdot \begin{pmatrix} a_x^2 & a_y a_x & a_z a_x \\ a_x a_y & a_y^2 & a_z a_y \\ a_x a_z & a_y a_z & a_z^2 \end{pmatrix} \cdot (1 - \cos \theta) \right) \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \\ &+ \left(\begin{pmatrix} 0 & a_z & -a_y \\ -a_z & 0 & a_x \\ a_y & -a_x & 0 \end{pmatrix} \cdot \sin \theta \right) \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}\end{aligned}$$

внесем множители в матрицы и помним, что $\|\vec{A}\| = 1$, вычислим матрицу перед \vec{P} :

$$\begin{aligned}&\begin{pmatrix} \cos \theta & 0 & 0 \\ 0 & \cos \theta & 0 \\ 0 & 0 & \cos \theta \end{pmatrix} + \begin{pmatrix} a_x^2 \cdot (1 - \cos \theta) & a_y a_x \cdot (1 - \cos \theta) & a_z a_x \cdot (1 - \cos \theta) \\ a_x a_y \cdot (1 - \cos \theta) & a_y^2 \cdot (1 - \cos \theta) & a_z a_y \cdot (1 - \cos \theta) \\ a_x a_z \cdot (1 - \cos \theta) & a_y a_z \cdot (1 - \cos \theta) & a_z^2 \cdot (1 - \cos \theta) \end{pmatrix} \\ &+ \begin{pmatrix} 0 & a_z \cdot \sin \theta & -a_y \cdot \sin \theta \\ -a_z \cdot \sin \theta & 0 & a_x \cdot \sin \theta \\ a_y \cdot \sin \theta & -a_x \cdot \sin \theta & 0 \end{pmatrix}\end{aligned}$$

ИТОГ:

$$\begin{pmatrix} \cos \theta + a_x^2 \cdot (1 - \cos \theta) & a_y a_x \cdot (1 - \cos \theta) + a_z \cdot \sin \theta & a_z a_x \cdot (1 - \cos \theta) - a_y \cdot \sin \theta \\ a_x a_y \cdot (1 - \cos \theta) - a_z \cdot \sin \theta & \cos \theta + a_y^2 \cdot (1 - \cos \theta) & a_z a_y \cdot (1 - \cos \theta) + a_x \cdot \sin \theta \\ a_x a_z \cdot (1 - \cos \theta) + a_y \cdot \sin \theta & a_y a_z \cdot (1 - \cos \theta) - a_x \cdot \sin \theta & \cos \theta + a_z^2 \cdot (1 - \cos \theta) \end{pmatrix}$$

Итоговая матрица поворота:

$$\text{Rotate}_{\vec{A}(a_x, a_y, a_z)}(\theta) = \begin{pmatrix} c + a_x^2(1 - c) & a_y a_x(1 - c) + a_z s & a_z a_x(1 - c) - a_y s \\ a_x a_y(1 - c) - a_z s & c + a_y^2(1 - c) & a_z a_y(1 - c) + a_x s \\ a_x a_z(1 - c) + a_y s & a_y a_z(1 - c) - a_x s & c + a_z^2(1 - c) \end{pmatrix}$$

здесь $c = \cos \theta$ и $s = \sin \theta$.

Это для нотации вектор-столбцов — $\vec{P} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \Rightarrow \vec{P}' = \text{Rotate}_{\vec{A}(a_x, a_y, a_z)}(\theta) \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$.

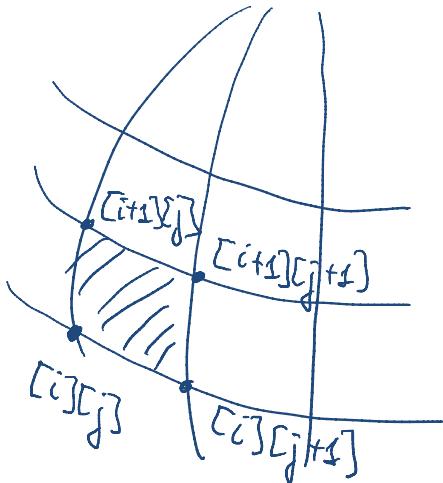
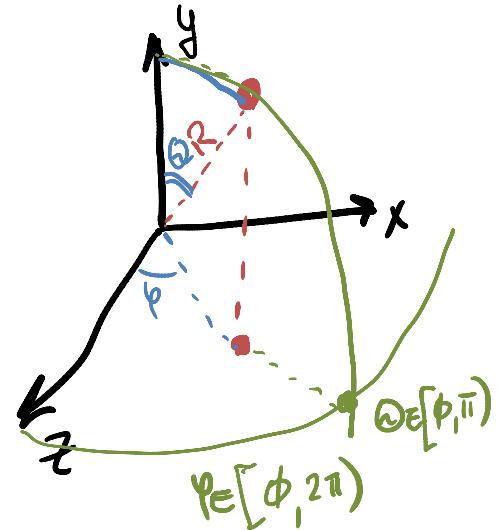
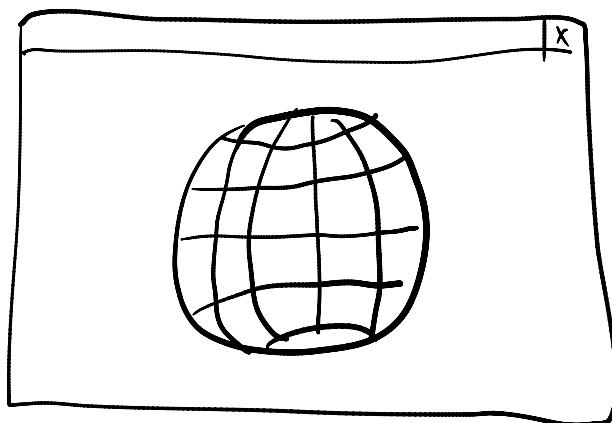
Для нотации вектор-строки — $\vec{P} = (p_x \ p_y \ p_z)$ — матрица транспонируется:

$$\text{Rotate}_{\vec{A}(a_x, a_y, a_z)}(\theta) = \begin{pmatrix} c + a_x^2(1 - c) & a_x a_y (1 - c) - a_z s & a_x a_z (1 - c) + a_y s \\ a_y a_x (1 - c) + a_z s & c + a_y^2(1 - c) & a_y a_z (1 - c) - a_x s \\ a_z a_x (1 - c) - a_y s & a_z a_y (1 - c) + a_x s & c + a_z^2(1 - c) \end{pmatrix}$$

$$\vec{P}' = (p_x \ p_y \ p_z) \cdot \text{Rotate}_{\vec{A}(a_x, a_y, a_z)}(\theta)$$

.::: 12:55 ::.

Практика: T07GLOBE



```
typedef double DBL;
typedef struct tagVEC
{
    DBL X, Y, Z;
} VEC;
```

Сетку сферы храним в двумерном массиве:

```
VEC G[N][M];

for (theta = 0, i = 0; i < N; i++, theta += PI / N)
    for (phi = 0, j = 0; j < M; j++, phi += 2 * PI / M)
    {
        x = R * sin(theta) * sin(phi);
```

```

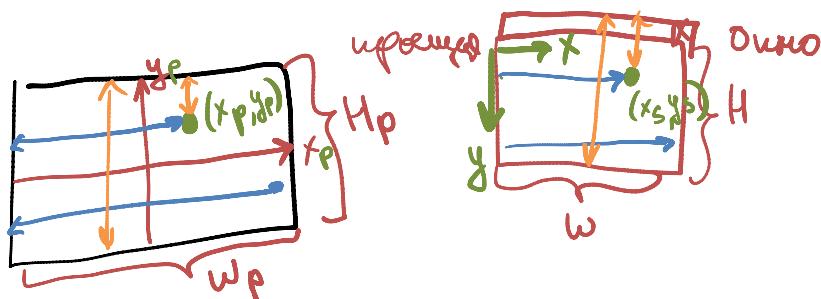
y = R * cos(theta);
z = R * sin(theta) * cos(phi);

G[i][j].X = x;
. . .
}

```

```
VOID SphereBuild( DBL R );
```

Отрисовка:



```
DBL size = 1, rx = size, ry = size;
```

```

if (W > H)
    rx *= (DBL)W / H;
else
    ry *= (DBL)H / W;

```

```
Wp = rx;
Hp = ry;
```

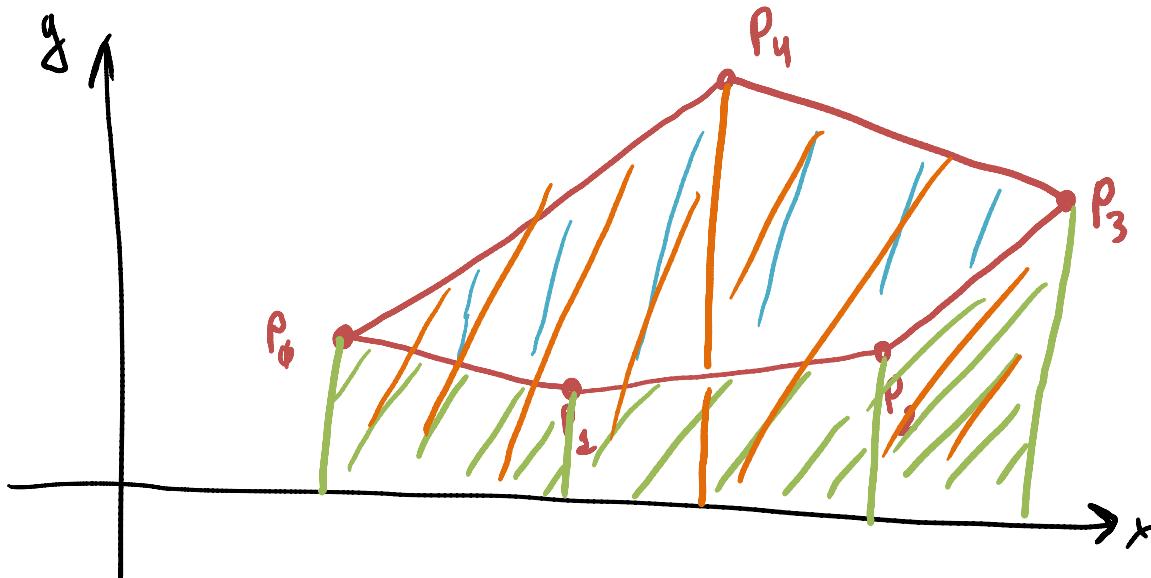
$$\frac{X_p - \left(-\frac{W_p}{2}\right)}{W_p} = \frac{x_s}{W}$$

```
xs = xp * W / Wp + W / 2; (xp == G[i][j].X)
```

$$\frac{\frac{H_p}{2} - y_p}{H_p} = \frac{y_s}{H}$$

```
ys = H / 2 - yp * H / Hp; (yp == G[i][j].Y)
```

.: 16:35 :.

Как определить обход точек в многоугольнике?

Вычислим площадь многоугольника как разность площадей "верхних" трапеций и "нижних":

$$S_{\text{top}} = (P_4.X - P_0.X)(P_0.Y + P_4.Y) / 2 + \\ (P_3.X - P_4.X)(P_3.Y + P_4.Y) / 2$$

$$S_{\text{bottom}} = (P_1.X - P_0.X)(P_0.Y + P_1.Y) / 2 + \\ (P_2.X - P_1.X)(P_1.Y + P_2.Y) / 2 + \\ (P_3.X - P_2.X)(P_2.Y + P_3.Y) / 2$$

Общая площадь:

$$S = S_{\text{top}} - S_{\text{bottom}} = \\ (P_4.X - P_0.X)(P_0.Y + P_4.Y) / 2 + \\ (P_3.X - P_4.X)(P_3.Y + P_4.Y) / 2 - \\ (P_1.X - P_0.X)(P_0.Y + P_1.Y) / 2 - \\ (P_2.X - P_1.X)(P_1.Y + P_2.Y) / 2 - \\ (P_3.X - P_2.X)(P_2.Y + P_3.Y) / 2$$

переупорядочим:

$$S = \\ (P_0.X - P_1.X)(P_0.Y + P_1.Y) / 2 + \\ (P_1.X - P_2.X)(P_1.Y + P_2.Y) / 2 + \\ (P_2.X - P_3.X)(P_2.Y + P_3.Y) / 2 + \\ (P_3.X - P_4.X)(P_3.Y + P_4.Y) / 2 + \\ (P_4.X - P_0.X)(P_4.Y + P_0.Y) / 2$$

$$S = \frac{1}{2} \sum_{i=0}^4 (P_i.X - P_{i+1}.X) \cdot (P_i.Y + P_{i+1}.Y)$$

при этом если $i+1 == 4 \rightarrow$ полагаем значение индекса 0

$$S = \frac{1}{2} \sum_{i=0}^{N-1} (P_i.X - P_{i+1}.X) \cdot (P_i.Y + P_{i+1}.Y)$$

для количества точек N (0..N-1)

!!! если точки указаны против часовой стрелки - площадь получится положительная, а если для нашего случая:

проверяемое выражение:

$$(p[0].x - p[1].x) * (p[0].y + p[1].y) + \\ (p[1].x - p[2].x) * (p[1].y + p[2].y) + \\ (p[2].x - p[3].x) * (p[2].y + p[3].y) + \\ (p[3].x - p[0].x) * (p[3].y + p[0].y)$$

если < 0 – многоугольник пропускаем!!!

-= 08.06.2017 =-

.:: 16:45 ::.

Преобразования на плоскости и в пространстве.

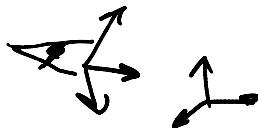
- plane transformation



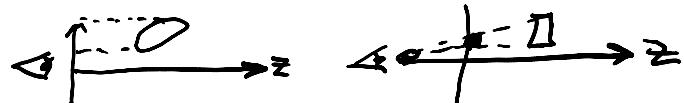
- space transformation



- camera transformation



- projection (parallel & central/perspective)



Translation

Scaling

Shearing (Skew)

Rotation

Homogeneous coordinates $(x \ y \ 1)$ – radius vector $(x \ y \ 0)$ – arbitrary vector

$\bar{N} = (A, B)$
 $\bar{N}' = (A', B')$
 $\bar{P} = (x, y)$
 $\bar{P}' = (x', y')$
 $\bar{N}' \cdot \bar{P}' = \phi$
 $(\bar{N} \cdot Q) \cdot (\bar{P} \cdot M) = 0$
 в матр. виде:
 $((A \ B \ \phi) \cdot Q_{3 \times 3}) \cdot ((x \ y \ \phi) \cdot M^T)$

$$(A \cdot B)^T = B^T \cdot A^T$$

$$(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$$

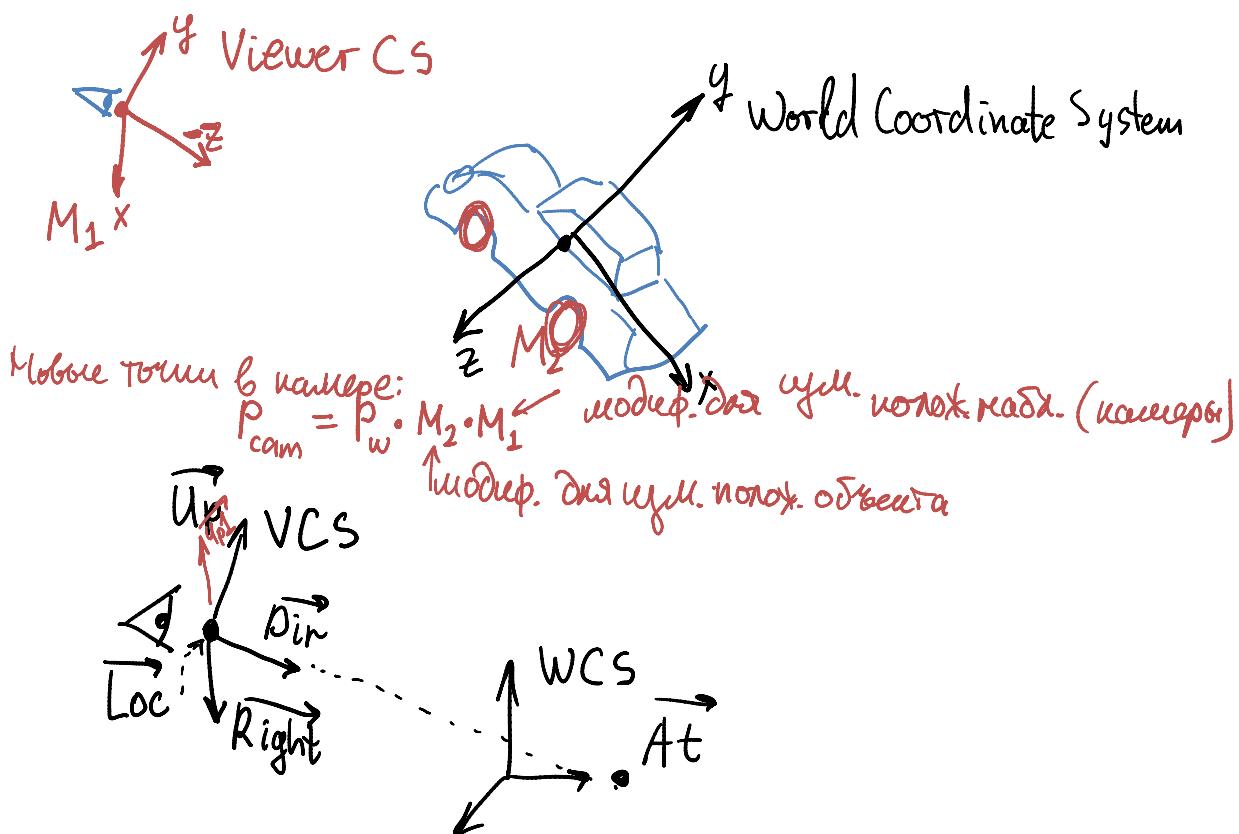
$$\sqrt{(A \cdot \phi)} \cdot \underbrace{(Q \cdot M^T) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}}_E = \phi$$

$$Q \cdot M^T = E$$

$$Q = (M^T)^{-1} = (M^{-1})^T$$

```
P = (X, Y, Z)
M = A[4][4];
{
    . . . 0
    . . . 0
    . . . 0
    . . . 1
}
```

Viewing in 3D



Камера задается:

VEC

- Loc, - позиция в пространстве (location)
- Dir, - направление "вперед" (direction)
- Up, - направление "вверх"
- Right; - направление "влево"

для удобства обычно эти параметры пересчитывают из: Loc, At, Up;

здесь - Loc - позиция, At - точка, куда смотри камера (по ней вычисляется вектор Dir), Up1 - приблизительное направление вверх:

$$\overrightarrow{Dir} = (\overrightarrow{At} - \overrightarrow{Loc}).normalize, \overrightarrow{Right} = (\overrightarrow{Dir} \times \overrightarrow{Up1}).normalize, \overrightarrow{Up} = (\overrightarrow{Right} \times \overrightarrow{Dir})$$

определение матрицы видового преобразования:

$$\begin{aligned} |Dir| &= |Right| = |Up| = 1 \\ Dir \cdot Up &= Dir \cdot Right = Right \cdot Up = 0 \\ Right \times Dir &= Up \\ Dir \times Up &= Right \\ Up \times Right &= Dir \end{aligned}$$

Будем искать преобразования как комбинацию параллельного переноса и поворота:

$$M = M_{\text{translation}} \cdot M_{\text{rotation}}$$

параллельный перенос – матрица $M_{\text{translation}}$:

$$M_{\text{translation}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -L_x & -L_y & -L_z & 1 \end{pmatrix}$$

(здесь L – Loc)

поворот – матрица M_{rotation} :

$$\begin{aligned} (Rx \ Ry \ Rz \ 0) \cdot M_{\text{rotation}} &= (1 \ 0 \ 0 \ 0) \\ (Ux \ Uy \ Uz \ 0) \cdot M_{\text{rotation}} &= (0 \ 1 \ 0 \ 0) \\ (Dx \ Dy \ Dz \ 0) \cdot M_{\text{rotation}} &= (0 \ 0 \ -1 \ 0) \end{aligned}$$

$$\begin{pmatrix} Rx & Ry & Rz & 0 \\ Ux & Uy & Uz & 0 \\ Dx & Dy & Dz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} ? & ? & ? & 0 \\ ? & ? & ? & 0 \\ ? & ? & ? & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Rx \cdot ?_{11} + Ry \cdot ?_{21} + Rz \cdot ?_{31} = 1$$

$$Rx \cdot ?_{12} + Ry \cdot ?_{22} + Rz \cdot ?_{32} = 0$$

$$Rx \cdot ?_{13} + Ry \cdot ?_{23} + Rz \cdot ?_{33} = 0$$

$$Ux \cdot ?_{11} + Uy \cdot ?_{21} + Uz \cdot ?_{31} = 0$$

$$Ux \cdot ?_{12} + Uy \cdot ?_{22} + Uz \cdot ?_{32} = 1$$

$$Ux \cdot ?_{13} + Uy \cdot ?_{23} + Uz \cdot ?_{33} = 0$$

$$Dx \cdot ?_{11} + Dy \cdot ?_{21} + Dz \cdot ?_{31} = 0$$

$$Dx \cdot ?_{12} + Dy \cdot ?_{22} + Dz \cdot ?_{32} = 0$$

$$Dx \cdot ?_{13} + Dy \cdot ?_{23} + Dz \cdot ?_{33} = -1$$

решение – скалярное произведение:

$$Rx \cdot Rx + Ry \cdot Ry + Rz \cdot Rz = 1$$

$$Rx \cdot Ux + Ry \cdot Uy + Rz \cdot Uz = 0$$

$$Rx \cdot -Dx + Ry \cdot -Dy + Rz \cdot -Dz = 0$$

$$U_x \cdot R_x + U_y \cdot R_y + U_z \cdot R_z = 0$$

$$U_x \cdot U_x + U_y \cdot U_y + U_z \cdot U_z = 1$$

$$U_x \cdot -D_x + U_y \cdot -D_y + U_z \cdot -D_z = 0$$

$$D_x \cdot R_x + D_y \cdot R_y + D_z \cdot R_z = 0$$

$$U_x \cdot U_x + U_y \cdot U_y + U_z \cdot U_z = 1$$

$$D_x \cdot -D_x + D_y \cdot -D_y + D_z \cdot -D_z = -1$$

$$\begin{pmatrix} R_x & R_y & R_z & 0 \\ U_x & U_y & U_z & 0 \\ D_x & D_y & D_z & 0 \end{pmatrix} \begin{pmatrix} Rx & U_x & -D_x & 0 \\ Ry & U_y & -D_y & 0 \\ Rz & U_z & -D_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$M_{translation} \cdot M_{rotation} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -L_x & -L_y & -L_z & 1 \end{pmatrix} \cdot \begin{pmatrix} R_x & U_x & -D_x & 0 \\ R_y & U_y & -D_y & 0 \\ R_z & U_z & -D_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} R_x & U_x & -D_x & 0 \\ R_y & U_y & -D_y & 0 \\ R_z & U_z & -D_z & 0 \\ -L_x \cdot R_x - L_y \cdot R_y - L_z \cdot R_z & -L_x \cdot U_x - L_y \cdot U_y - L_z \cdot U_z & L_x \cdot D_x + L_y \cdot D_y + L_z \cdot D_z & 1 \end{pmatrix}$$

План проектирования библиотеки поддержки пространственной геометрии.

Реализация — файлы: VEC.H VEC.C

```
#include <math.h>

#define PI 3.14159265358979323846
#define D2R(A) ((A) * (PI / 180.0))
#define Degree2Radian(a) D2R(a)

/* базовый вещественный тип */
typedef double DBL;

/* тип для вектора в пространстве */
typedef struct tagVEC
{
    DBL X, Y, Z;
} VEC;

/* тип для матрицы — массив в структуре */
typedef struct tagMATR
{
    DBL M[4][4];
} MATR;

/* функции реализации */
VEC VecSet( DBL X, DBL Y, DBL Z )
{
    VEC v = {X, Y, Z};

    return v;
}

VEC VecAddVec( VEC V1, VEC V2 )
{
    return VecSet(V1.X + V2.X, V1.Y + V2.Y, V1.Z + V2.Z);
}

VEC VecSubVec( VEC V1, VEC V2 );

```

```

VEC VecMulNum( VEC V1, DBL N );
VEC VecDivNum( VEC V1, DBL N );
VEC VecNeg( VEC V )
{
    return VecSet(-V.X, -V.Y, -V.Z);
}

/* скалярное произведение векторов */
DBL VecDotVec( VEC V1, VEC V2 );
/* векторное произведение векторов */
VEC VecCrossVec( VEC V1, VEC V2 );
DBL VecLen2( VEC V );
DBL VecLen( VEC V );
VEC VecNormalize( VEC V );
VEC VecMulMatr( VEC V, MATR M );
/* еще можно с присваиванием: */
VEC VecAddVecEq( VEC *VRes, VEC V2 )
{
    VRes->X += V2.X;
    VRes->Y += V2.Y;
    VRes->Z += V2.Z;
    return *VRes;
}

/* преобразования: */
VEC PointTransform( VEC V, MATR M );
VEC VectorTransform( VEC V, MATR M );
??? VecMulMatr VecMulMatr43 VecMulMatr33
/* пример для нормалей: */
MATR Q = MatrTranspose(MatrInverse(M));
N1 = VectorTransform(N, Q);
Реализация матриц:
/* единичная матрица */
static MATR UnitMatrix =
{
    {
        {1, 0, 0, 0},
        {0, 1, 0, 0},
        {0, 0, 1, 0},
        {0, 0, 0, 1}
    }
};
MATR MatrIdentity( VOID )
{
    return UnitMatrix;
}
MATR MatrTranslate( VEC T )
{
    MATR m = UnitMatrix;

    m.A[3][0] = T.X;
    m.A[3][1] = T.Y;
    m.A[3][2] = T.Z;
    return m;
}
MATR MatrScale( VEC S );
MATR MatrRotateX( DBL AngleInDegree );
MATR MatrRotateY( DBL AngleInDegree );
MATR MatrRotateZ( DBL AngleInDegree );
MATR MatrRotate( DBL AngleInDegree,
                  VEC R )
{
    DBL A = D2R(AngleToDegree), si = sin(A), co = cos(A);
    VEC V = VecNormalize(R);
    MATR M =
    {
        {
            {co + V.X * V.X * (1 - co),
             V.X * V.Y * (1 - co) - V.Z * si,
             V.X * V.Z * (1 - co) + V.Y * si, 0},

```

```

{V.Y * V.X * (1 - co) + V.Z * si,
 co + V.Y * V.Y * (1 - co),
 V.Y * V.Z * (1 - co) - V.X * si, 0},
 {V.Z * V.X * (1 - co) - V.Y * si,
 V.Z * V.Y * (1 - co) + V.X * si,
 co + V.Z * V.Z * (1 - co), 0},
 {0, 0, 0, 1}
}
};

return M;
}

MATR MatrMulMatr( MATR M1, MATR M2 )
{
    MATR r;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            for (r.A[i][j] = 0, k = 0; k < 4; k++)
                r.A[i][j] += M1.A[i][k] * M2.A[k][j]
}
MATR MatrTranspose( MATR M );

DBL MatrDeterm3x3( DBL A11, DBL A12, DBL A13,
                     DBL A21, DBL A22, DBL A23,
                     DBL A31, DBL A32, DBL A33 )
{
    return A11 * A22 * A33 - A11 * A23 * A32 - A12 * A21 * A33 +
           A12 * A23 * A31 + A13 * A21 * A32 - A13 * A22 * A31;
}

DBL MatrDeterm( MATR M )
{
    return
        M.A[0][0] * MatrDeterm3x3(M.A[1][1], M.A[1][2], M.A[1][3],
                                    M.A[2][1], M.A[2][2], M.A[2][3],
                                    M.A[3][1], M.A[3][2], M.A[3][3]) -
        M.A[0][1] * MatrDeterm3x3(M.A[1][0], M.A[1][2], M.A[1][3],
                                    M.A[2][0], M.A[2][2], M.A[2][3],
                                    M.A[3][0], M.A[3][2], M.A[3][3]) +
        M.A[0][2] * MatrDeterm3x3(M.A[1][0], M.A[1][1], M.A[1][3],
                                    M.A[2][0], M.A[2][1], M.A[2][3],
                                    M.A[3][0], M.A[3][1], M.A[3][3]) -
        M.A[0][3] * MatrDeterm3x3(M.A[1][0], M.A[1][1], M.A[1][2],
                                    M.A[2][0], M.A[2][1], M.A[2][2],
                                    M.A[3][0], M.A[3][1], M.A[3][2]);
}

MATR MatrInverse( MATR M )
{
    MATR r;
    DBL det = MatrDeterm(M);
    /* перестановки для индексов алгебраических дополнений/миноров */
    INT p[4][3] = {{1, 2, 3}, {0, 2, 3}, {0, 1, 3}, {0, 1, 2}};

    if (det == 0)
        return UnitMatrix;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            r.A[j][i] =
                MatrDeterm3x3(
                    M.A[p[i][0]][p[j][0]], M.A[p[i][0]][p[j][1]], M.A[p[i][0]][p[j][2]],
                    M.A[p[i][1]][p[j][0]], M.A[p[i][1]][p[j][1]], M.A[p[i][1]][p[j][2]],
                    M.A[p[i][2]][p[j][0]], M.A[p[i][2]][p[j][1]], M.A[p[i][2]][p[j][2]]) /

```

```

    det;
    return r;
}

```

!!!

В Visual Studio существует расширение языка Си:

если перед функцией поставить служебное слово `__inline` функция становится статической и разворачивается прямо в текст

программы (как `define`), такие функции помещают в файлы заголовков

```

__inline void Swap1( double *a, double *b )
{
    double tmp = *a;

    *a = *b;
    *b = tmp;
}

```

тогда вызов

`Swap1(&a, &b)` превратится в текст `__t__ = a, a = b, b = __t__;` где `__t__` какая-то переменная

!!!!

Функция:

```

MATR MatrView( VEC Loc, VEC At, VEC Up1 )
{
    VEC
    Dir = VecNormalize(VecSubVec(At, Loc)),
    Right = VecNormalize(VecCrossVec(Dir, Up1)),
    Up = VecNormalize(VecCrossVec(Right, Dir));
    MATR m =
    {
        {
            {Right.X, Up.X, -Dir.X, 0},
            {Right.Y, Up.Y, -Dir.Y, 0},
            {Right.Z, Up.Z, -Dir.Z, 0},
            {-VecDotVec(Loc, Right), -VecDotVec(Loc, Up), VecDotVec(Loc, Dir), 1}
        }
    };
}

return m;
}

```

-= 09.06.2017 =-

::: 12:15 :::

про преобразования:

```

/* Vector by matrix multiplication (without projection) function.
 * ARGUMENTS:
 *   - source vector:
 *     VEC V;
 *   - multiplied matrix:
 *     MATR M;
 * RETURNS:
 *   (VEC) result vector.
 */
__inline VEC VecMulMatr43( VEC V, MATR M )

```

```

{
    return VecSet(V.X * M.A[0][0] + V.Y * M.A[1][0] + V.Z * M.A[2][0] + M.A[3][0],
                  V.X * M.A[0][1] + V.Y * M.A[1][1] + V.Z * M.A[2][1] + M.A[3][1],
                  V.X * M.A[0][2] + V.Y * M.A[1][2] + V.Z * M.A[2][2] + M.A[3][2]);
} /* End of 'VecMulMatr43' function */

/* Vector by matrix multiplication (only orientation) function.
 * ARGUMENTS:
 *   - source vector:
 *     VEC V;
 *   - multiplied matrix:
 *     MATR M;
 * RETURNS:
 *   (VEC) result vector.
 */
__inline VEC VecMulMatr3( VEC V, MATR M )
{
    return VecSet(V.X * M.A[0][0] + V.Y * M.A[1][0] + V.Z * M.A[2][0],
                  V.X * M.A[0][1] + V.Y * M.A[1][1] + V.Z * M.A[2][1],
                  V.X * M.A[0][2] + V.Y * M.A[1][2] + V.Z * M.A[2][2]);
} /* End of 'VecMulMatr3' function */

/* Vector by matrix multiplication (with homogenous devide) function.
 * ARGUMENTS:
 *   - source vector:
 *     VEC V;
 *   - multiplied matrix:
 *     MATR M;
 * RETURNS:
 *   (VEC) result vector.
 */
__inline VEC PointTransform( VEC V, MATR M )
{
    FLT w = V.X * M.A[0][3] + V.Y * M.A[1][3] + V.Z * M.A[2][3] + M.A[3][3];

    return VecSet((V.X * M.A[0][0] + V.Y * M.A[1][0] + V.Z * M.A[2][0] + M.A[3][0]) / w,
                  (V.X * M.A[0][1] + V.Y * M.A[1][1] + V.Z * M.A[2][1] + M.A[3][1]) / w,
                  (V.X * M.A[0][2] + V.Y * M.A[1][2] + V.Z * M.A[2][2] + M.A[3][2]) / w);
} /* End of 'PointTransform' function */
}

```

::= 12:50 ::.

оптимизация вычисления sin и cos:

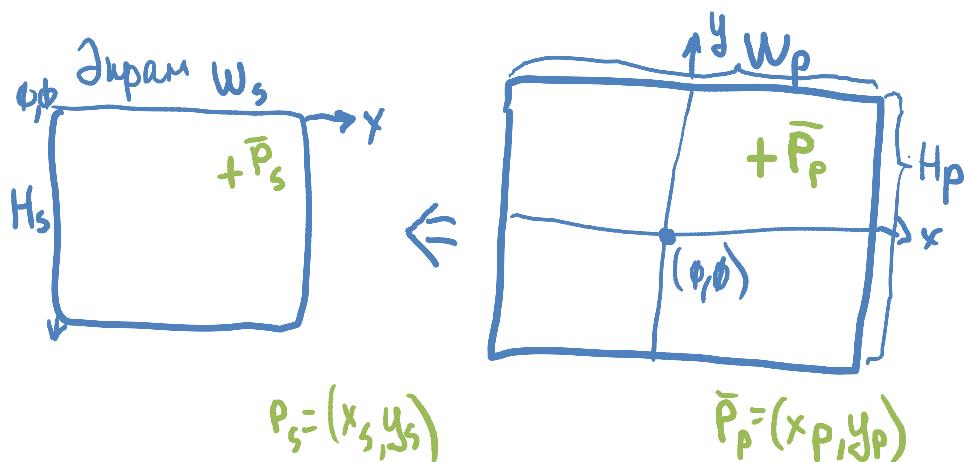
CPU (+NPU/Math CoProcessor)
8 registers (st0-st7)

```

__asm {
    fsincos
}

 ::= 14:50 ::.
Проектирование
Projection & NDC (normalized device coordinate)

```

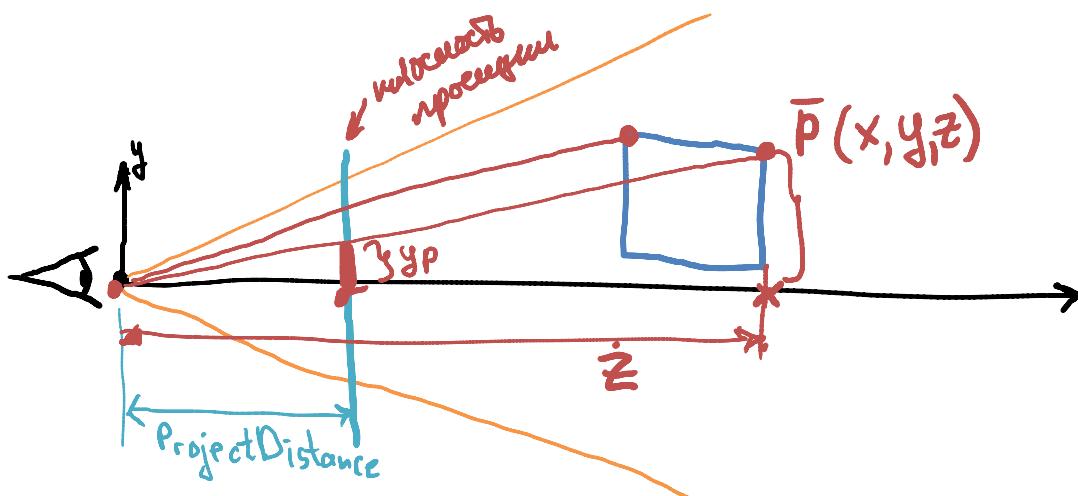


$$x_s = \frac{\left(x_p + \frac{W_p}{2}\right)}{W_p} \cdot W_s = \frac{W_s}{2} + x_p \cdot \frac{W_s}{W_p}$$

$$y_s = \frac{\left(-y_p + \frac{H_p}{2}\right)}{H_p} \cdot H_s = \frac{H_s}{2} - y_p \cdot \frac{H_s}{H_p}$$

это ортографическая (orthographics, параллельная - ортогональная) проекция

Как учесть перспективу:



$$\frac{z}{ProjectDistance} = \frac{y}{y_p} \Rightarrow y_p = \frac{(y \cdot ProjectDistance)}{z}$$

$$x_p = \frac{x \cdot ProjectDistance}{z}$$

после чего – как в параллельной проекции

!!!

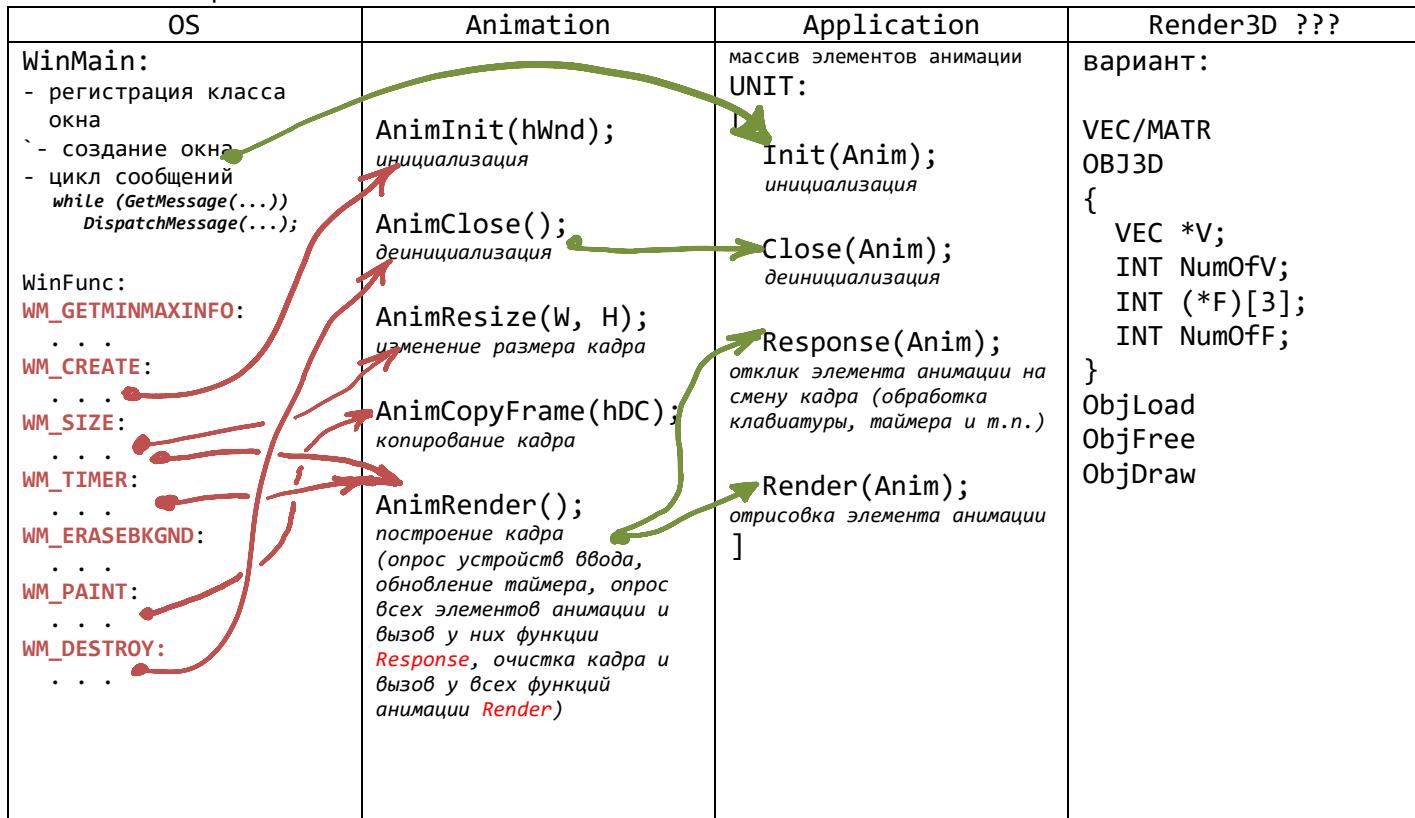
`malloc - <stdlib.h>`

```
DBL *a;  
  
a = malloc(sizeof(DBL) * n);  
if (a == NULL)  
    return;  
for (i = 0; i < n; i++)  
    a[i] = ... ;  
...  
free(a);
```

Загрузка модели из формата
Alias Wavefront - Power Animator - MAYA
geometry exchange file format: *.OBJ

```
v X Y Z  
...  
f #1 #2 #3      f #1/? #2/? #3/?      f #1/?/? #2/?/? #3/?/?      f #1//? #2//? #3//?  
...
```

Макет анимации:



Вопросы:

- таймер
- чтение клавиатура, мышь, джойстик
- как создать массив объектов анимации с разным поведением и размером?

Начнем с объектов анимации.

у каждого объекта анимации – свои функции **Init**, **Close**, **Response**, **Render**
храним их как указатели на функции!!!

Что они получают как параметры?

- "указатель на себя" – self pointer – (C++/C# **this**) – указатель на те данные, к которым функция применяет свои действия (**UNIT *Uni**)
- "указатель на анимацию" – указатель на структуру со всеми параметрами анимации (буфер кадра, состояние мыши, клавы, джойстика, время и т.п.) – контекст анимации (**ANIM *Ani**)

```
typedef struct tagUNIT
{
    VOID (*Init)( UNIT *Uni, ANIM *Ani );
    VOID (*Close)( UNIT *Uni, ANIM *Ani );
    VOID (*Response)( UNIT *Uni, ANIM *Ani );
    VOID (*Render)( UNIT *Uni, ANIM *Ani );
} UNIT;
```

```
UNIT *Units[MAX_UNITS];
```

!!! ООП: данные управляют собственным поведением !!!

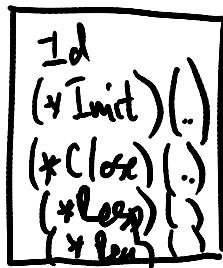
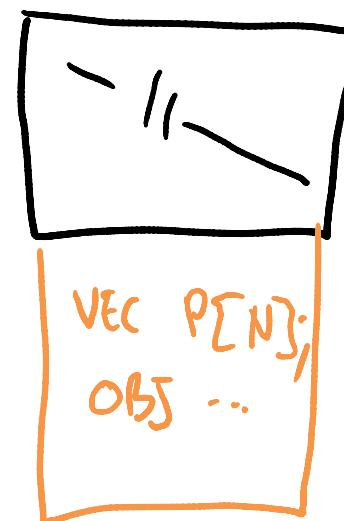
AnimRender:

```
...
for (i = 0; i < NumOfUnits; i++)
    Units[i]->Response(Units[i], &Anim);
...
for (i = 0; i < NumOfUnits; i++)
```

```
Units[i]->Render(Units[i], &Anim);
...
```

ООП:

- ИНКАРСУЛЯЦИЯ (incapsulation) - скрытие данных
- НАСЛЕДОВАНИЕ (inheritance) - включение полей одних структур - в другие
- ПОЛИМОРФИЗМ (polymorph, override) - различное поведение наследуемых функций

UNIT**UNIT_COW****UNIT_TREES**

...

базовый тип

закрепляем общие поля:

```
#define UNIT_BASE_FIELDS \
    VOID (*Init)( UNIT *Uni, ANIM *Ani );      \
    VOID (*Close)( UNIT *Uni, ANIM *Ani );       \
    VOID (*Response)( UNIT *Uni, ANIM *Ani );    \
    VOID (*Render)( UNIT *Uni, ANIM *Ani )
```

```
typedef struct tagUNIT
{
    UNIT_BASE_FIELDS;
} UNIT;
```

производный тип – примеры:

```
typedef struct tagUNIT_COW
{
    UNIT_BASE_FIELDS;
    VEC Pos;
    OBJ3D Model;
};
```

!!! Система анимации хранит массив УКАЗАТЕЛЕЙ на наши объекты, а так как в начале каждого - одинаковый тип UNIT - то она может пользоваться Init, Close, Response и Render

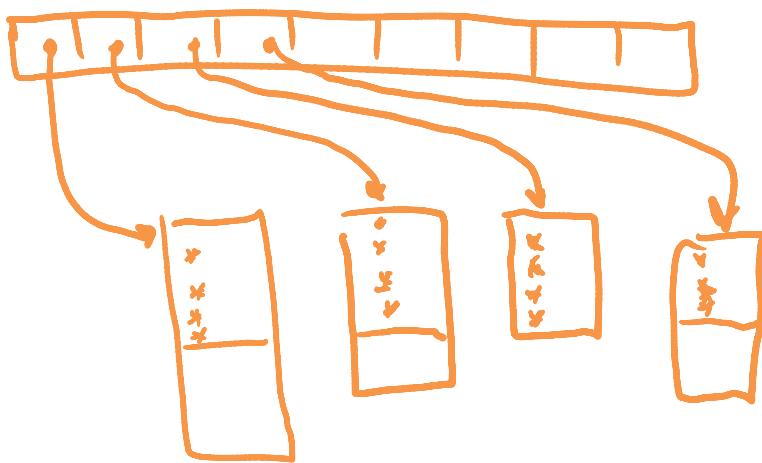
```
UNIT *Units[MAX];
```

+ функция:

```
VOID AnimAddUnit( UNIT *Uni )
{
    if (NumOfUnits < MAX_UNITS)
    {
        Units[NumOfUnits++] = Uni;
        Uni->Init(Uni, &Anim);
    }
}
```

}

Unit 5:



НАЧАЛО РЕАЛИЗАЦИИ:

```
DEF.H
MemHandle
UINT64 INT64 (DBL FLT <-- VEC.H)
```

MAIN.C - WinMain + WinFunc (работа с WinAPI)

!!! все реализуется с префиксом ИФК (VG4)

AG4 DG5 AK5 AH5

ANIM.C

```
VG4_Anim***  
VG4_AnimInit( HWND hWnd );  
VG4_AnimClose( VOID );  
VG4_AnimResize( INT W, INT H );  
VG4_AnimCopyFrame( HDC hDC );  
VG4_AnimRender( VOID );  
VG4_AnimAddUnit( vg4UNIT *Uni );  
VG4_AnimFlipFullScreen( VOID );
```

ANIM.H:

+ VG4_Anim - глобальная переменная - структура контекста анимации типа vg4ANIM:

typedef struct tagvg4ANIM

{

...

} vg4ANIM;

правильно:

typedef struct tagvg4ANIM vg4ANIM; -> в начале файла

typedef struct tagvg4UNIT vg4UNIT; -> в начале файла

```
struct tagvg4ANIM  
{
```

HWND hWnd;	- окно
HDC hDC;	- контекст в памяти
INT W, H;	- размер окна
HBITMAP hFrame;	- картинка кадра
vg4UNIT *Units[VG4_MAX_UNITS];	- массив объектов анимации
INT NumOfUnits;	- текущее количество объектов анимации

};

```
struct tagvg4UNIT  
{
```

```

/* Unit initialization function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
VOID (*Init)( vg4UNIT *Uni, vg4ANIM *Ani );

/* Unit deinitialization function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
VOID (*Close)( vg4UNIT *Uni, vg4ANIM *Ani );

/* Unit inter frame events handle function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
VOID (*Response)( vg4UNIT *Uni, vg4ANIM *Ani );

/* Unit render function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
VOID (*Render)( vg4UNIT *Uni, vg4ANIM *Ani );
};


```

UNITS.C - вся базовая работа с объектами анимации

--- функции - "заглушки" - нужны для начальной инициализации любого объекта анимации по умолчанию

```

/* Unit initialization function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
static VOID VG4_UnitInit( vg4UNIT *Uni, vg4ANIM *Ani )
{
} /* End of 'VG4_UnitInit' function */
/* Unit deinitialization function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */


```

```

static VOID VG4_UnitClose( vg4UNIT *Uni, vg4ANIM *Ani )
{
} /* End of 'VG4_UnitClose' function */

/* Unit inter frame events handle function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
static VOID VG4_UnitResponse( vg4UNIT *Uni, vg4ANIM *Ani );
{
} /* End of 'VG4_UnitResponse' function */

/* Unit render function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
static VOID VG4_UnitRender( vg4UNIT *Uni, vg4ANIM *Ani )
{
} /* End of 'VG4_UnitRender' function */

```

- основная функция создания объекта анимации по умолчанию:

```

/* Unit creation function.
 * ARGUMENTS:
 *   - unit structure size in bytes:
 *     INT Size;
 * RETURNS:
 *   (vg4UNIT *) pointer to created unit.
 */
vg4UNIT * VG4_AnimUnitCreate( INT Size )
{
    vg4UNIT *Uni;

    /* Memory allocation */
    if (Size < sizeof(vg4UNIT) || (Uni = malloc(Size)) == NULL)
        return NULL;
    memset(Uni, 0, Size);
    /* Setup unit methods */
    Uni->Init = VG4_UnitInit;
    Uni->Close = VG4_UnitClose;
    Uni->Response = VG4_UnitResponse;
    Uni->Render = VG4_UnitRender;
    return Uni;
} /* End of 'VG4_AnimUnitCreate' function */

```

пример своего объекта анимации:

U_BBALL.C

```

#include "anim.h"

typedef struct
{

```

```

VG4_UNIT_BASE_FIELDS;
VEC Pos;
} vg4UNIT_BALL;

/* Unit ball initialization function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT_BALL *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
static VOID VG4_UnitInit( vg4UNIT_BALL *Uni, vg4ANIM *Ani )
{
    Uni->Pos = VecSet(0, 1, 0);
} /* End of 'VG4_UnitInit' function */

/* Unit render function.
 * ARGUMENTS:
 *   - self-pointer to unit object:
 *     vg4UNIT_BALL *Uni;
 *   - animation context:
 *     vg4ANIM *Ani;
 * RETURNS: None.
 */
static VOID VG4_UnitRender( vg4UNIT_BALL *Uni, vg4ANIM *Ani )
{
    DrawSphere(Uni->Pos, 111);
} /* End of 'VG4_UnitRender' function */

/* Unit ball creation function.
 * ARGUMENTS: None.
 * RETURNS:
 *   (vg4UNIT *) pointer to created unit.
 */
vg4UNIT * VG4_UnitCreateBall( VOID )
{
    vg4UNIT_BALL *Uni;

    if ((Uni = (vg4UNIT_BALL *)VG4_AnimUnitCreate(sizeof(vg4UNIT_BALL))) == NULL)
        return NULL;
    /* Setup unit methods */
    Uni->Init = VG4_UnitInit;
    Uni->Render = VG4_UnitRender;
    return (vg4UNIT *)Uni;
} /* End of 'VG4_UnitCreateBall' function */

```

UNITS.H

```

-- функции-конструкторы примеров:
/* Unit ball creation function.
 * ARGUMENTS: None.
 * RETURNS:
 *   (vg4UNIT *) pointer to created unit.
 */
vg4UNIT * VG4_UnitCreateBall( VOID );

#include "units.h"
WinMain:
. . .
VG4_AnimAddUnit(VG4_UnitCreateBall());

```

VG4_AnimRender:

```

    . . . timer
    . . . опросили все (kbd, mouse, joystick)
    for (i = 0; i < VG4_Anim.NumOfUnits; i++)
        VG4_Anim.Units[i]->Response(VG4_Anim.Units[i], &VG4_Anim);

    . . . очищаем кадр
    for (i = 0; i < VG4_Anim.NumOfUnits; i++)
    {
        . . . можно сбросить все кисти и перья
        VG4_Anim.Units[i]->Render(VG4_Anim.Units[i], &VG4_Anim);
    }
    . . .
}

```

Source Files**Math Support**

VEC.H

VEC.C

Animation System

ANIM.H

ANIM.C

UNITS.C

Unit Samples

UNITS.H

U_BBALL.C

U_CLOCK.C

U_POLE.C

MAIN.C

DEF.H

---> Cow/Frustum/Kbd/Mouse/Joystick

-= 10.06.2017 =-

.: 10:00 :.

Практика

.: 14:00 :.

Timer

Time	- общее время анимационной системы
DeltaTime	- время с прошлого кадра
IsPause	- флаг паузы
GlobalTime	- время без учета паузы
GlobalDeltaTime	- время с прошлого кадра без учета паузы
FPS	- frames per second

???, StartTime, OldTime, OldFPSTime, FrameCounter

Old school:

<time.h>

```

long t = clock();
double sec = (double)t / CLOCKS_PER_SEC;

```

на старте программы AnimInit:

StartTime = clock();

В Response:

```
long t = clock();
TimeInSec = (t - StartTime) / (DBL)CLOCKS_PER_SEC;
```

High Resolution Timer

специальный тип LARGE_INTEGER:

```
LARGE_INTEGER t;
```

Опрос частоты обновления таймера:

```
QueryPerformanceFrequency(&t); -- сколько замеров в секунду
TimePerSec = t.QuadPart;
```

Опрос таймера:

```
QueryPerformanceCounter(&t);
Time = t.QuadPart;
```

В нашем проекте:

-- дополнения в структуре ANIM:

```
/* Timer data */
DBL
    GlobalTime, GlobalDeltaTime, /* Global time and interframe interval */
    Time, DeltaTime,           /* Time with pause and interframe interval */
    FPS;                      /* Frames per seond value */
BOOL
    IsPause;                  /* Pause flag */
```

-- статические переменные в ANIM.C:

```
/* Timer local data */
static UINT64
    VG4_StartTime, /* Start program time */
    VG4_OldTime,   /* Time from program start to previous frame */
    VG4_OldTimeFPS, /* Old time FPS measurement */
    VG4_PauseTime, /* Time during pause period */
    VG4_TimePerSec, /* Timer resolution */
    VG4_FrameCounter; /* Frames counter */
```

-- инициализация таймера AnimInit:

```
LARGE_INTEGER t;
. . .
/* Timer initialization */
QueryPerformanceFrequency(&t);
VG4_TimePerSec = t.QuadPart;
QueryPerformanceCounter(&t);
VG4_StartTime = VG4_OldTime = VG4_OldTimeFPS = t.QuadPart;
VG4_PauseTime = 0;
VG4_FrameCounter = 0;
```

```
VG4_Anim.IsPause = FALSE;
VG4_Anim.FPS = 50;
. . .
```

-- обработка таймера AnimRender:

```
LARGE_INTEGER t;
. . .
/** Handle timer **/
```

```

VG4_FrameCounter++;
-- increment frame counter (for FPS)
QueryPerformanceCounter(&t);
-- obtain current timer value
/* Global time */
VG4_Anim.GlobalTime = (DBL)(t.QuadPart - VG4_StartTime) / VG4_TimePerSec;
VG4_Anim.GlobalDeltaTime = (DBL)(t.QuadPart - VG4_OldTime) / VG4_TimePerSec;
/* Time with pause */
if (VG4_Anim.IsPause)
{
    VG4_PauseTime += t.QuadPart - VG4_OldTime;
    VG4_Anim.DeltaTime = 0;
}
else
{
    VG4_Anim.Time = (DBL)(t.QuadPart - VG4_PauseTime - VG4_StartTime) / VG4_TimePerSec;
    VG4_Anim.DeltaTime = VG4_Anim.GlobalDeltaTime;
}
/* FPS */
if (t.QuadPart - VG4_OldTimeFPS > VG4_TimePerSec)
{
    VG4_Anim.FPS = (DBL)VG4_FrameCounter * VG4_TimePerSec / (t.QuadPart - VG4_OldTimeFPS);
    VG4_OldTimeFPS = t.QuadPart;
    VG4_FrameCounter = 0;
}
VG4_OldTime = t.QuadPart;

```

.:: 15:25 ::.

Альтернативный цикл сообщений (посылка перерисовки в режиме простого процессора – Idle):

было

```

INT WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                     CHAR *CmdLine, INT CmdShow )
{
    . . .
    MSG msg;
    . . .
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
    . . .
} /* End of 'WinMain' function */

```

стало

```

INT WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                     CHAR *CmdLine, INT CmdShow )
{
    . . .
    MSG msg;
    . . .
    while (TRUE)
        if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
        {
            if (msg.message == WM_QUIT)
                break;
            TranslateMessage(&msg);
            DispatchMessage(&msg);

```

```

    }
    else
        SendMessage(hWnd, WM_TIMER, 47, 0);
    return msg.wParam;
    . . .
} /* End of 'WinMain' function */

```

.:: 15:35 ::.



Ввод (input)

1. Keyboard

Существует функция: VOID GetKeyboardState(BYTE Keys[256]);
Возвращает состояние каждой клавиши – старший бит 1 – клавиша нажата:
BYTE Keys[256];

```

GetKeyboardState(Keys);
if (Keys[VK_ESCAPE] & 0x80)
    . . .
VK_LEFT ... '0' '1' ... '9' 'A' ... 'Z'

```

У нас:

-- дополнения в структуре ANIM:

```

BYTE Keys[256];
BYTE KeysOld[256];
BYTE KeysClick[256];

```

-- обработка клавиатуры AnimRender:

```

GetKeyboardState(VG4_Anim.Keys);
for (i = 0; i < 256; i++)
{
    VG4_Anim.Keys[i] >>= 7;
    VG4_Anim.KeysClick[i] = VG4_Anim.Keys[i] && !VG4_Anim.KeysOld[i]; ← Отслеживание однократного нажатия
}
memcpy(VG4_Anim.KeysOld, VG4_Anim.Keys, 256);

```

2. Mouse

!!! Машины кнопки хранятся в массиве Keys:
VK_LBUTTON, VK_RBUTTON, VK_MBUTTON

!!! Колесо: mouse wheel

```

WM_MOUSEWHEEL:
delta_z = (SHORT)HIWORD(wParam);

```

-- дополнения в структуре ANIM:

```

INT
Mx, My, Mz,      /* Absolute coordinates */

```

```
Mdx, Mdy, Mdz; /* Relative coordinates shift */
```

-- обработка мыши AnimRender:

```
POINT pt;
```

```
GetCursorPos(&pt);
```

```
ScreenToClient(VG4_Anim.hWnd, &pt);
```

```
VG4_Anim.Mdx = pt.x - VG4_Anim.Mx;
```

```
VG4_Anim.Mdy = pt.y - VG4_Anim.My;
```

```
VG4_Anim.Mx = pt.x;
```

```
VG4_Anim.My = pt.y;
```

глобальная переменная:

```
INT VG4_MouseWheel;
```

в MyWinFunc:

```
case WM_MOUSEWHEEL:
```

```
    VG4_MouseWheel += (SHORT)HIWORD(wParam);
```

```
    return 0;
```

в AnimRender:

```
VG4_Anim.Mdz = VG4_MouseWheel;
```

```
VG4_Anim.Mz += VG4_MouseWheel;
```

```
VG4_MouseWheel = 0;
```

3. Joystick



доп библиотеки:

```
#include <mmsystem.h> (после <windows.h>)
```

+ подключаем библиотеку: /* winmm.lib */

допустимо в файле записать:

```
#pragma comment(lib, "winmm")
```

-- дополнения в структуре ANIM:

```
BYTE
```

```
JBut[32], /* Joystick button state */
```

```
JButOld[32], /* Joystick button old state */
```

```
JButtonClick[32]; /* Joystick button click info */
```

```
INT JPov; /* Joystick point-of-view control [0,1..8] */
```

```
DBL
```

```
Jx, Jy, Jz, Jr; /* Joystick axes */
```

-- обработка джойстика AnimRender:

```
/*** Joystick handle ***/
```

```
if (joyGetNumDevs() > 0)
```

```
{
```

```
JOYCAPS jc;
```

```
/* Get joystick info */
```

```
if (joyGetDevCaps(JOYSTICKID1, &jc, sizeof(JOYCAPS)) == JOYERR_NOERROR)
```

```
{
```

```
JOYINFOEX ji;
```

```

ji.dwSize = sizeof(JOYINFOEX);
ji.dwFlags = JOY_RETURNALL;
if (joyGetPosEx(JOYSTICKID1, &ji) == JOYERR_NOERROR)
{
    /* Axes */
    VG4_Anim.Jx = VG4_GET_JOYSTICK_AXIS(X);
    VG4_Anim.Jy = VG4_GET_JOYSTICK_AXIS(Y);
    VG4_Anim.Jz = VG4_GET_JOYSTICK_AXIS(Z);
    VG4_Anim.Jr = VG4_GET_JOYSTICK_AXIS(R);

    /* Buttons */
    for (i = 0; i < 32; i++)
    {
        VG4_Anim.JBut[i] = (ji.dwButtons >> i) & 1;
        VG4_Anim.JButClick[i] = VG4_Anim.JBut[i] && !VG4_Anim.JButOld[i];
        memcpy(VG4_Anim.JButOld, VG4_Anim.JBut, 32);
    }

    /* Point of view */
    VG4_Anim.JPov = ji.dwPOV == 0xFFFF ? 0 : ji.dwPOV / 4500 + 1;
}
}
}
}

::: 17:30 :::

```

Загрузка модели из формата
 Alias Wavefront - Power Animator - MAYA
 geometry exchange file format: *.OBJ

```

v_X Y Z
. . .
f_#1 #2 #3      f #1/? #2/? #3/?      f #1/?/? #2/?/? #3/?/?      f #1//? #2//? #3//?
. . .

```

```

OBJ3D
{
    VEC *V;
    INT NumOfV;
    INT (*F)[3];
    INT NumOff;
}
ObjLoad
ObjFree

```

ObjDraw

```

-----/* Project parameters */
DBL
    VG4_RndWp,      /* Project plane width */
    VG4_RndHp,      /* Project plane height */
    VG4_RndProjDist, /* Distance from viewer to project plane */
    VG4_RndProjSize; /* Project plane inner size */

```

```

MATR
    VG4_RndMatrView; /* Viewer matrix */

```

```

/* Rendering system initialization function.
* ARGUMENTS: None.

```

```

* RETURNS: None.
*/
VOID VG4_RndInit( VOID )
{
    VG4_RndWp = 1;
    VG4_RndHp = 1;
    VG4_RndProjDist = 1;
    VG4_RndProjSize = 1;

    VG4_RndMatrView = MatrView(VecSet1(5), VecSet1(0), VecSet(0, 1, 0));
} /* End of 'VG4_RndInit' function */

/* Project parameters adjust function.
* ARGUMENTS: None.
* RETURNS: None.
*/
VOID VG4_RndSetProj( VOID )
{
    VG4_RndWp = VG4_RndProjSize;
    VG4_RndHp = VG4_RndProjSize;
    if (VG4_Anim.W > VG4_Anim.H)
        VG4_RndWp *= (DBL)VG4_Anim.W / VG4_Anim.H;
    else
        VG4_RndHp *= (DBL)VG4_Anim.H / VG4_Anim.W;
} /* End of 'VG4_RndSetProj' function */

/* Object drawing function.
* ARGUMENTS:
*   - object pointer:
*     vg4OBJ3D *Obj;
*   - world coordinate system transform matrix:
*     MATR M;
* RETURNS: None.
*/
VOID VG4_RndObjDraw( vg4OBJ3D *Obj, MATR M )
{
    POINT *pts;

    if ((pts = malloc(sizeof(POINT) * Obj->NumOfV)) == NULL)
        return;

    M = MatrMulMatr(M, VG4_RndMatrView);

    /* Project all points */
    for (i = 0; i < Obj->NumOfV; i++)
    {
        VEC p = VecMulMatr(Obj->V[i], M);
        DBL
            xp = p.X * VG4_RndProjDist / p.Z,
            yp = p.Y * VG4_RndProjDist / p.Z;

        pts[i].x = VG4_Anim.W / 2 + xp * VG4_Anim.W / VG4_RndWp;
        pts[i].y = VG4_Anim.H / 2 - yp * VG4_Anim.H / VG4_RndHp;
    }

    /* Draw all facets */
    for (i = 0; i < Obj->NumOffF; i++)
    {
        POINT *p = &pts[Obj->F[i][0]];
        MoveToEx(VG4_Anim.hDC, p->x, p->y, NULL);
    }
}

```

```

p = &pts[Obj->F[i][1]];
LineTo(VG4_Anim.hDC, p->x, p->y);

p = &pts[Obj->F[i][2]];
LineTo(VG4_Anim.hDC, p->x, p->y);

p = &pts[Obj->F[i][0]];
LineTo(VG4_Anim.hDC, p->x, p->y);
}
free(pts);
} /* End of 'VG4_RndObjDraw' function */

```

ObjFree

```

-----  

/* Object free memory function.  

 * ARGUMENTS:  

 *   - object pointer:  

 *     vg4OBJ3D *Obj;  

 * RETURNS: None.  

 */  

VOID VG4_RndObjFree( vg4OBJ3D *Obj )  

{  

    if (Obj->V != NULL)  

        free(Obj->V);  

    memset(Obj, 0, sizeof(vg4OBJ3D));  

} /* End of 'VG4_RndObjFree function */

```

ObjLoad

```

-----  

/* Object free memory function.  

 * ARGUMENTS:  

 *   - object pointer:  

 *     vg4OBJ3D *Obj;  

 *   - model *.OBJ file name:  

 *     CHAR *FileName;  

 * RETURNS:  

 *   (BOOL) TRUE if success, FALSE otherwise.  

 */  

BOOL VG4_RndObjFree( vg4OBJ3D *Obj, CHAR *FileName )  

{  

    if (Obj->V != NULL)  

        free(Obj->V);  

    memset(Obj, 0, sizeof(vg4OBJ3D));  

} /* End of 'VG4_RndObjFree function */

```

-= 13.06.2017 -=

.:: 10:00 ::.

Практика

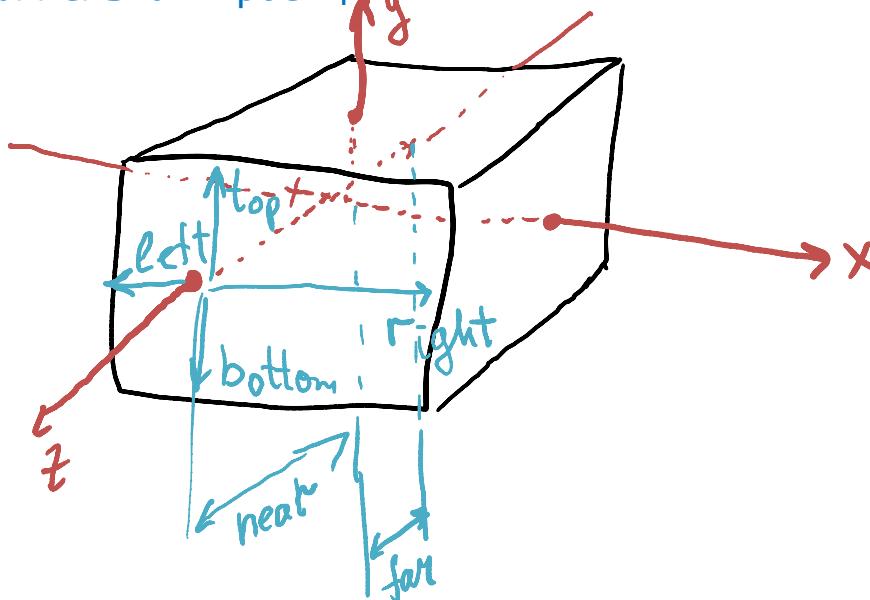
.:: 14:25 ::.

Normalized Device Coordinates (NDC)

нормализованные координаты устройства

X,Y,Z : -1..1

Параллельная проекция

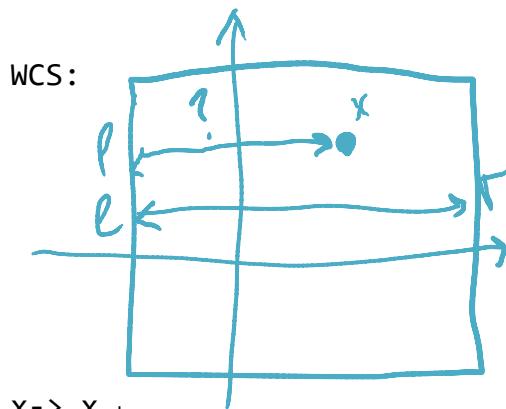


$(left, 0, 0) \rightarrow (-1, 0, 0)$
 $(right, 0, 0) \rightarrow (1, 0, 0)$
 $(0, bottom, 0) \rightarrow (0, -1, 0)$
 $(0, top, 0) \rightarrow (0, 1, 0)$
 $(0, 0, near) \rightarrow (0, 0, -1) \quad // near == ProjDist$
 $(0, 0, far) \rightarrow (0, 0, 1)$

NDC:



WCS:

 $x \rightarrow x_{ndc}$

$$\frac{x_{ndc} - (-1)}{1 - (-1)} = \frac{x - left}{right - left}$$

$$x_{ndc} = 2 \cdot \frac{x - left}{right - left} - 1$$

аналогично:

$$y_{ndc} = 2 \cdot \frac{y - bottom}{top - bottom} - 1$$

разворачиваем z:

$$z_{ndc} = 2 \cdot \frac{-z - near}{far - near} - 1$$

$$x_{ndc} = 2 \cdot \frac{x - left}{right - left} - 1 = x \cdot \left(\frac{2}{right - left} \right) + \left(\frac{(-2 \cdot left)}{right - left} - 1 \right) = \\ = x \cdot \left(\frac{2}{right - left} \right) + \left(-\frac{right + left}{right - left} \right)$$

$$P_{ndc} = P \cdot \begin{pmatrix} \frac{2}{right - left} & 0 & 0 & 0 \\ 0 & \frac{2}{top - bottom} & 0 & 0 \\ 0 & 0 & -\frac{2}{far - near} & 0 \\ -\frac{right + left}{right - left} & -\frac{top + bottom}{top - bottom} & -\frac{far + near}{far - near} & 1 \end{pmatrix}$$

это матрица ортографической (ортогональной) проекции (M_{ortho})

```
MATR MatrOrtho( DBL Left, DBL Right,
                  DBL Bottom, DBL Top,
                  DBL Near, DBL Far );
```

Частный случай – Right = -Left = Wp / 2 и Top = -Bottom = Hp / 2

$$P_{ndc} = P \cdot \begin{pmatrix} \frac{2}{Wp} & 0 & 0 & 0 \\ 0 & \frac{2}{Hp} & 0 & 0 \\ 0 & 0 & -\frac{2}{far - near} & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & 1 \end{pmatrix}$$

Теперь последовательность преобразований:

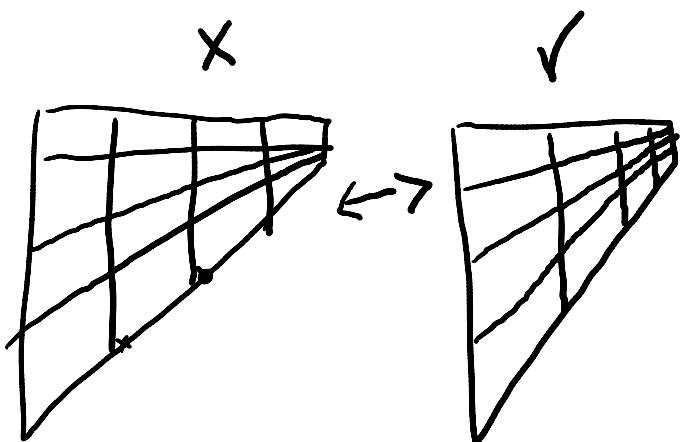
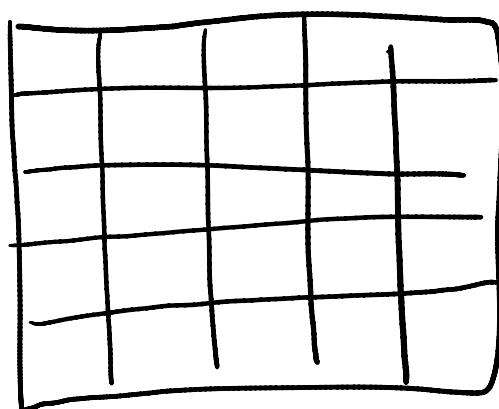
```
P1 = P * (M_world * M_view * M_project(ortho))
```

рисуем на экран:

```
POINT((P1.X + 1) * Ws / 2, (-P1.Y + 1) * Hs / 2);
```

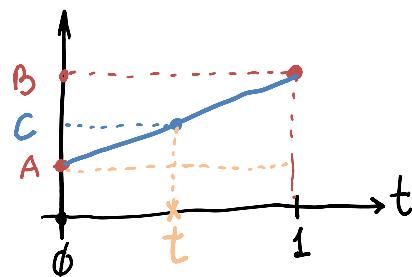
Центральная проекция

(1).интерполяция глубины



!!! О ЛИНЕЙНОЙ ИНТЕПОЛЯЦИИ

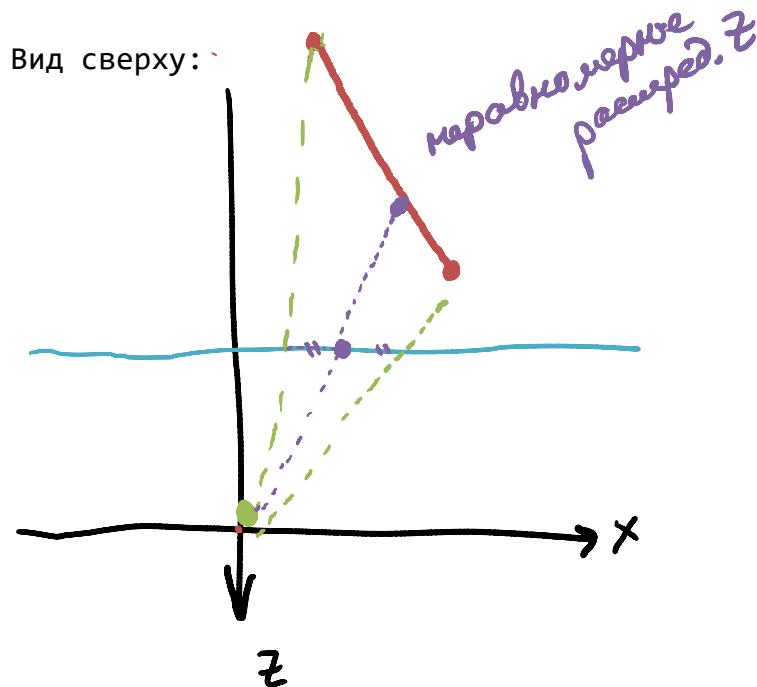
Пусть интерполирующий коэффициент t меняется от 0 до 1, а интеполируемые значения – А (для $t = 0$) и В (для $t = 1$).



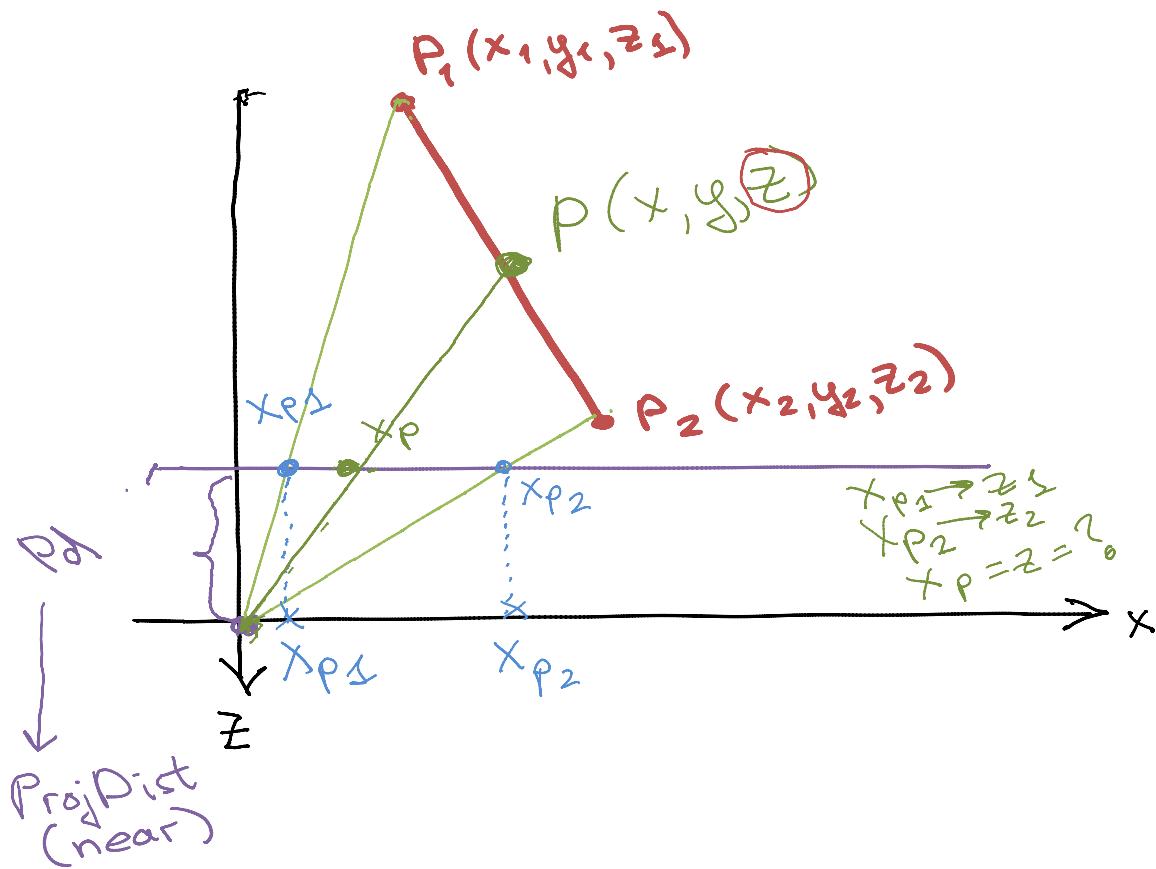
Из подобия треугольников получаем: $\frac{C-A}{B-A} = \frac{t-0}{1-0}$, откуда:

$$C = A + t \cdot (B - A) \text{ или } C = A \cdot (1 - t) + B \cdot t$$

Как изменяется координата Z при интерполяции точки на экране при отрисовке треугольников:



Детальный чертеж:



Здесь точки $\vec{P}_1 = (x_1, y_1, z_1)$ и $\vec{P}_2 = (x_2, y_2, z_2)$ задают концы отрезка в пространстве. На плоскость проекции они проецируются точками (по оси X) x_{p1} и x_{p2} (координаты). При построении отрезка на плоскости проекции (т.е., в частности, и на экране) происходит линейная интерполяция вдоль оси X от точки x_{p1} до точки x_{p2} . Координата по оси X меняется по закону линейной интерполяции. У нас на отрезке от x_{p1} до x_{p2} координата меняется по закону:

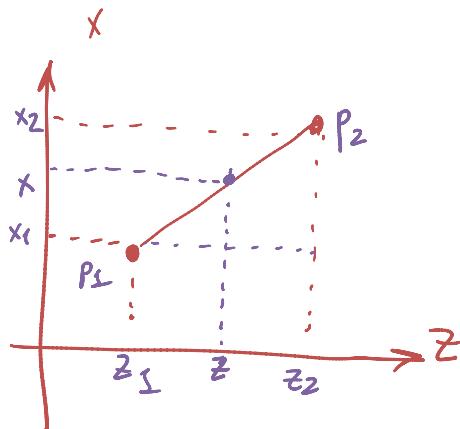
$$x_p = x_{p1} \cdot (1 - t) + x_{p2} \cdot t$$

На концах отрезка глубина (координата по оси Z) у крайних точек равна z_1 и z_2 соответственно.

Найдем формулу, по которой изменяется координата по оси Z при интерполяции по плоскости проекции (экрану) от точки x_{p1} до точки x_{p2} .

Для этого построим уравнение прямой, проходящей через точки p_1 и p_2 и уравнение прямой, проходящей через начало координат и точку x_p и найдем общую точку – ее координата по оси Z и является искомой.

Для начала выведем уравнение прямой в плоскости ZX, задаваемой двумя точками:



Из подобия треугольников получаем:

$$\frac{x_2 - x_1}{x - x_1} = \frac{z_2 - z_1}{z - z_1}$$

домножим слева и справа на $(x - x_1) \cdot (z - z_1)$ получим:

$$(z - z_1) \cdot (x_2 - x_1) = (x - x_1) \cdot (z_2 - z_1)$$

Для прямой, проходящей через точки p_1 и p_2 уравнение так и выглядит.

Для прямой, проходящей через начало координат (координаты (0,0,0)) и точку x_p (координаты (x_p, y_p, P_d) , здесь $P_d = ProjectDistance$) уравнение выглядит так:

$$(z - 0) \cdot (x_p - 0) = (x - 0) \cdot (P_d - 0)$$

или

$$z \cdot x_p = x \cdot P_d$$

Выражаем x :

$$x = z \cdot \frac{x_p}{P_d}$$

и подставляем в первое уравнение:

$$(z - z_1) \cdot (x_2 - x_1) = \left(z \cdot \frac{x_p}{P_d} - x_1 \right) \cdot (z_2 - z_1)$$

Выделим z в левой и в правой части части:

$$z \cdot (x_2 - x_1) - z_1 \cdot (x_2 - x_1) = z \cdot \frac{x_p}{P_d} \cdot (z_2 - z_1) - x_1 \cdot (z_2 - z_1)$$

Переносим налево:

$$z \cdot \left((x_2 - x_1) - \frac{x_p}{P_d} \cdot (z_2 - z_1) \right) = (z_1 \cdot (x_2 - x_1) - x_1 \cdot (z_2 - z_1))$$

И выражим z :

$$z = \frac{(z_1 \cdot (x_2 - x_1) - x_1 \cdot (z_2 - z_1))}{(x_2 - x_1) - \frac{x_p}{P_d} \cdot (z_2 - z_1)} = \frac{z_1 \cdot x_2 - x_1 \cdot z_2}{(x_2 - x_1) - \frac{x_p}{P_d} \cdot (z_2 - z_1)}$$

Помним, что координаты по оси X точек p_1 и p_2 с координатами x_{p1} и x_{p2} связаны соотношением центральной проекции:

$$x_{p1} = x_1 \cdot \frac{P_d}{z_1}$$

$$x_{p2} = x_2 \cdot \frac{P_d}{z_2}$$

откуда выражим x_1 и x_2 :

$$x_1 = x_{p1} \cdot \frac{z_1}{P_d}$$

$$x_2 = x_{p2} \cdot \frac{z_2}{P_d}$$

и подставим в формулу для вычисления z :

$$z = \frac{z_1 \cdot x_{p2} \cdot \frac{z_2}{P_d} - x_{p1} \cdot \frac{z_1}{P_d} \cdot z_2}{\left(x_{p2} \cdot \frac{z_2}{P_d} - x_{p1} \cdot \frac{z_1}{P_d} \right) - \frac{x_p}{P_d} \cdot (z_2 - z_1)}$$

Сократим числитель и знаменатель на $\frac{1}{P_d}$:

$$z = \frac{z_1 \cdot x_{p2} \cdot z_2 - x_{p1} \cdot z_1 \cdot z_2}{(x_{p2} \cdot z_2 - x_{p1} \cdot z_1) - x_p \cdot (z_2 - z_1)}$$

Подставим значение x_p , полученное в результате интерполяции $x_p = x_{p1} \cdot (1 - t) + x_{p2} \cdot t$ в эту формулу:

$$z = \frac{z_1 \cdot x_{p2} \cdot z_2 - x_{p1} \cdot z_1 \cdot z_2}{(x_{p2} \cdot z_2 - x_{p1} \cdot z_1) - (x_{p1} \cdot (1 - t) + x_{p2} \cdot t) \cdot (z_2 - z_1)}$$

$$z = \frac{z_1 \cdot z_2 \cdot (x_{p2} - x_{p1})}{x_{p2} \cdot z_2 - x_{p1} \cdot z_1 - x_{p1} \cdot (z_2 - z_1) - t \cdot (x_{p2} - x_{p1}) \cdot (z_2 - z_1)}$$

$$z = \frac{z_1 \cdot z_2 \cdot (x_{p2} - x_{p1})}{x_{p2} \cdot z_2 - x_{p1} \cdot z_2 - t \cdot (x_{p2} - x_{p1}) \cdot (z_2 - z_1)}$$

$$z = \frac{z_1 \cdot z_2 \cdot (x_{p2} - x_{p1})}{(x_{p2} - x_{p1}) \cdot z_2 - t \cdot (x_{p2} - x_{p1}) \cdot (z_2 - z_1)}$$

$$z = \frac{z_1 \cdot z_2 \cdot (x_{p2} - x_{p1})}{(x_{p2} - x_{p1})(z_2 - t \cdot (z_2 - z_1))}$$

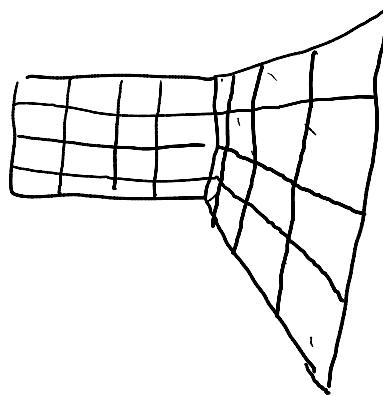
числитель и знаменатель сокращаем на $(x_{p2} - x_{p1})$:

$$z = \frac{z_1 \cdot z_2}{z_2 - t \cdot (z_2 - z_1)} = \frac{z_1 \cdot z_2}{z_2 \cdot (1 - t) + z_1 \cdot t}$$

Вычислим $\frac{1}{z}$:

$$\frac{1}{z} = \frac{z_2 \cdot (1 - t) + z_1 \cdot t}{z_1 \cdot z_2} = \frac{z_2 \cdot (1 - t)}{z_1 \cdot z_2} + \frac{z_1 \cdot t}{z_1 \cdot z_2} = \frac{1}{z_1} \cdot (1 - t) + \frac{1}{z_2} \cdot t$$

(2).интерполяция атрибутов



] в точках 1 и 2 есть атрибуты - b_1 и b_2 . Как влияет интерполяция на них:

$$\frac{b - b_1}{b_2 - b_1} = \frac{z - z_1}{z_2 - z_1}$$

перепишем относительно b :

$$b = b_1 + (b_2 - b_1) \cdot \frac{z - z_1}{z_2 - z_1}$$

подставляем:

$$z = \frac{z_1 \cdot z_2}{z_2 \cdot (1 - t) + z_1 \cdot t}$$

итог

$$b = \frac{b_1 \cdot z_2 + b_2 \cdot \frac{z_1 \cdot z_2}{z_2 \cdot (1 - t) + z_1 \cdot t} - b_2 \cdot z_1 - b_1 \cdot \frac{z_1 \cdot z_2}{z_2 \cdot (1 - t) + z_1 \cdot t}}{z_2 - z_1}$$

приводим к общему знаменателю

$$b = \frac{b_1 \cdot z_2 \cdot (z_2 \cdot (1-t) + z_1 \cdot t) + b_2 \cdot z_1 \cdot z_2 - b_2 \cdot z_1 \cdot (z_2 \cdot (1-t) + z_1 \cdot t) - b_1 \cdot z_1 \cdot z_2}{(z_2 \cdot (1-t) + z_1 \cdot t) \cdot (z_2 - z_1)}$$

рассмотрим числитель:

$$\begin{aligned} b_1 \cdot z_2 \cdot (z_2 \cdot (1-t) + z_1 \cdot t) + b_2 \cdot z_1 \cdot z_2 - b_2 \cdot z_1 \cdot (z_2 \cdot (1-t) + z_1 \cdot t) - b_1 \cdot z_1 \cdot z_2 = \\ = b_1 \cdot z_2 \cdot (z_2 \cdot (1-t) + z_1 \cdot (t-1)) - b_2 \cdot z_1 \cdot (-z_2 \cdot t + z_1 \cdot t) = \\ = (z_2 - z_1) \cdot (b_1 \cdot z_2 \cdot (1-t) + b_2 \cdot z_1 \cdot t) \end{aligned}$$

подставляем:

$$b = \frac{(z_2 - z_1) \cdot (b_1 \cdot z_2 \cdot (1-t) + b_2 \cdot z_1 \cdot t)}{(z_2 \cdot (1-t) + z_1 \cdot t) \cdot (z_2 - z_1)} = \frac{b_1 \cdot z_2 \cdot (1-t) + b_2 \cdot z_1 \cdot t}{z_2 \cdot (1-t) + z_1 \cdot t}$$

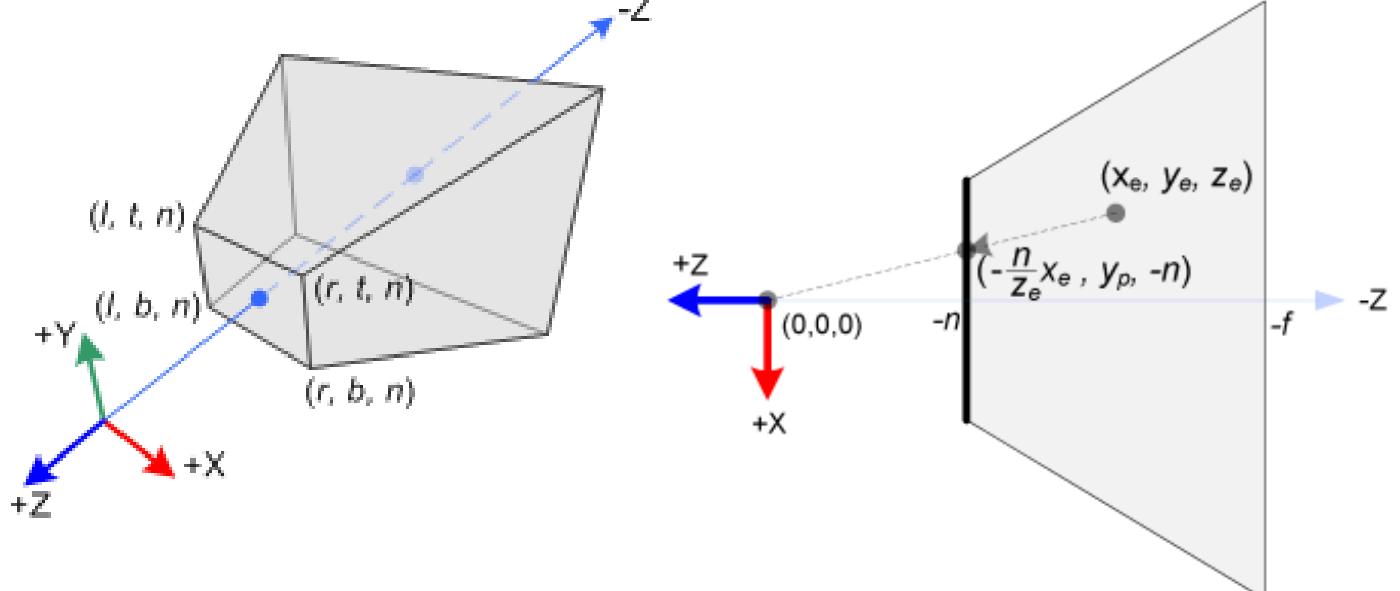
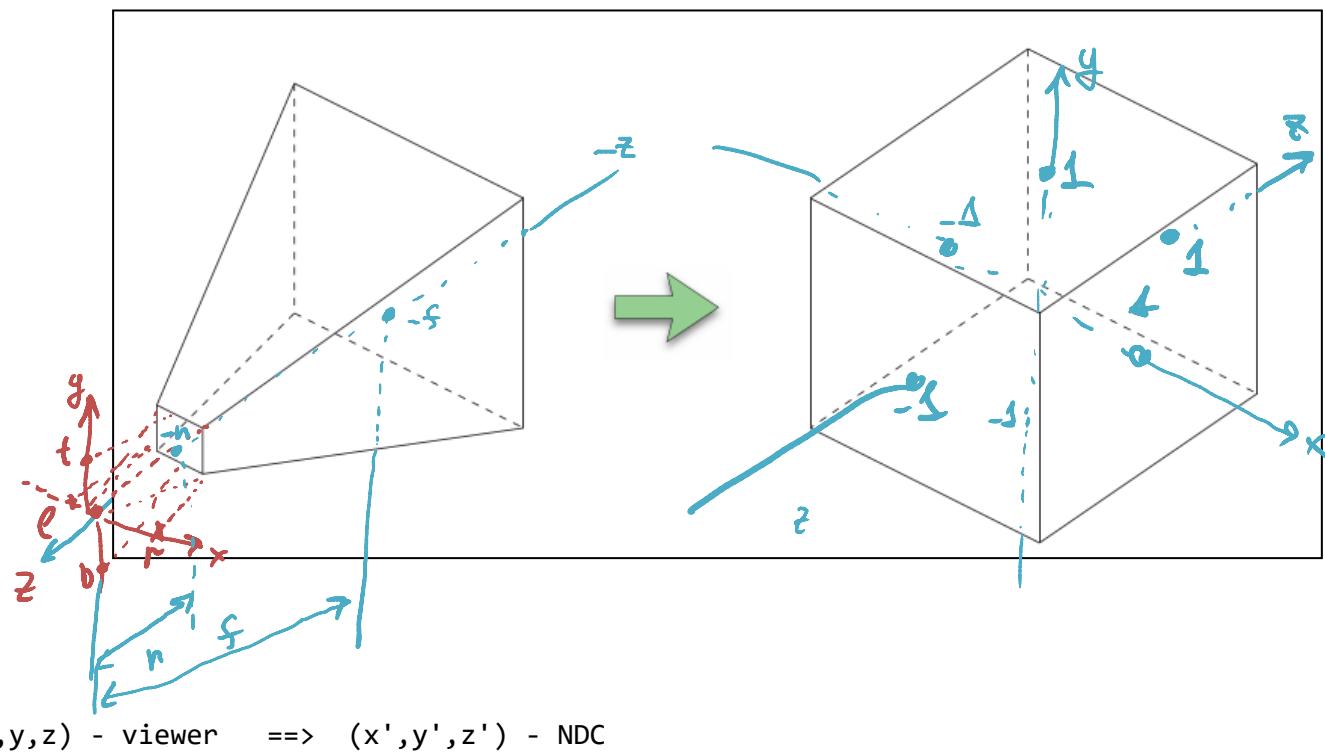
домножим и поделим $\frac{z_1 \cdot z_2}{z_1 \cdot z_2}$

$$b = \frac{b_1 \cdot z_2}{z_2 \cdot (1-t) + z_1 \cdot t} \cdot (1-t) + \frac{b_2 \cdot z_1}{z_2 \cdot (1-t) + z_1 \cdot t} \cdot t$$

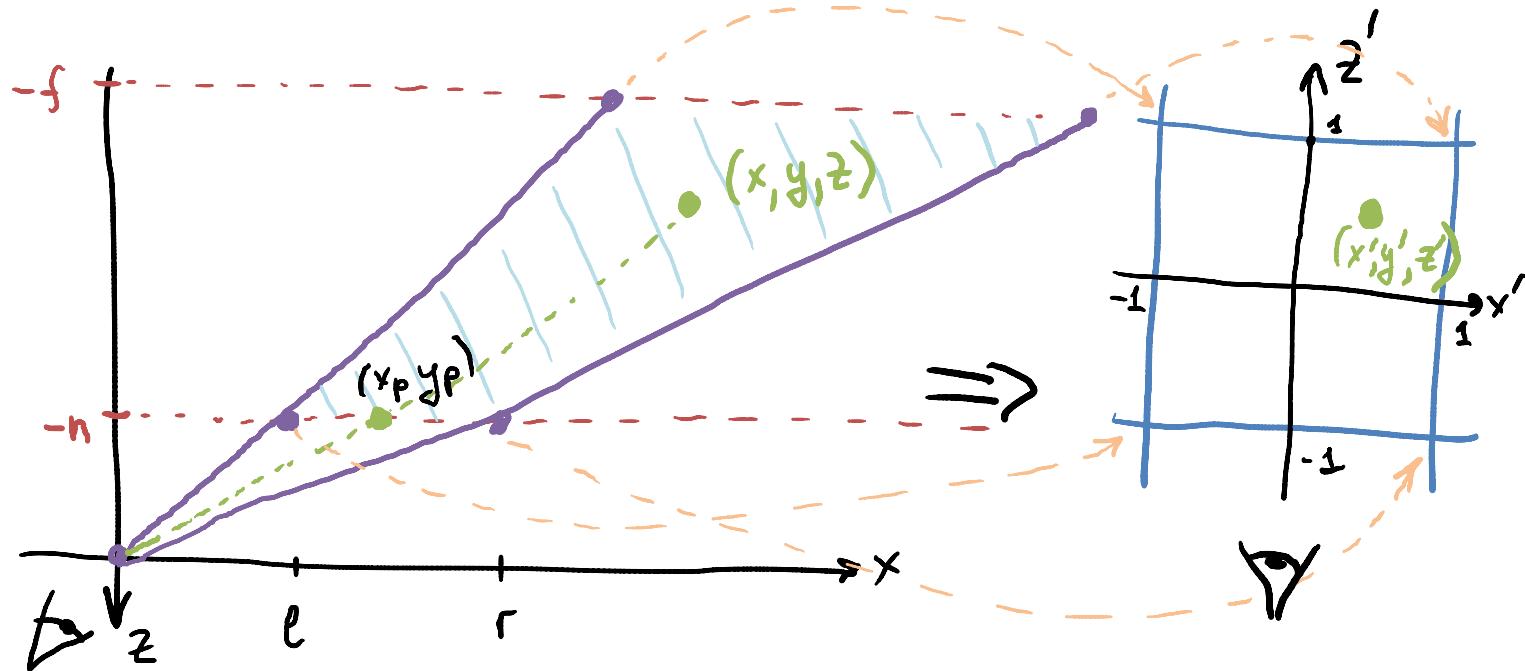
$$b = \frac{\frac{b_1}{z_1}}{\frac{1}{z_1} \cdot (1-t) + \frac{1}{z_2} \cdot t} \cdot (1-t) + \frac{\frac{b_2}{z_2}}{\frac{1}{z_1} \cdot (1-t) + \frac{1}{z_2} \cdot t} \cdot t$$

$$b = \frac{\frac{b_1}{z_1}}{\frac{1}{z}} \cdot (1-t) + \frac{\frac{b_2}{z_2}}{\frac{1}{z}} \cdot t$$

$$\frac{\mathbf{b}}{\mathbf{z}} = \frac{\mathbf{b}_1}{\mathbf{z}_1} \cdot (1-t) + \frac{\mathbf{b}_2}{\mathbf{z}_2} \cdot t$$



Вид сверху:



$$x_p = -\frac{n}{z} \cdot x, x_p \in [l..r] \rightarrow x' \in [-1..1], x' = 2 \cdot \frac{x_p - l}{r - l} - 1$$

$$y_p = -\frac{n}{z} \cdot y, y_p \in [b..t] \rightarrow y' \in [-1..1], y' = 2 \cdot \frac{y_p - b}{t - b} - 1$$

подставляем x_p и y_p :

$$x' = 2 \cdot \frac{-\frac{n}{z} \cdot x - l}{r - l} - 1 = \frac{2 \cdot n}{r - l} \cdot \left(-\frac{x}{z}\right) - \frac{r + l}{r - l}$$

$$y' = 2 \cdot \frac{-\frac{n}{z} \cdot y - b}{t - b} - 1 = \frac{2 \cdot n}{t - b} \cdot \left(-\frac{y}{z}\right) - \frac{t + b}{t - b}$$

 z' будем искать от $\frac{1}{z}$

$$z' = \frac{A}{z} + B$$

известно, что:

$$\begin{aligned} z = -n &\rightarrow z' = -1 \\ z = -f &\rightarrow z' = 1 \end{aligned}$$

получаем:

$$-1 = \frac{A}{-n} + B$$

$$1 = \frac{A}{-f} + B$$

вычтем из 2-го первое:

$$\begin{aligned} 2 &= \frac{A}{-f} - \frac{A}{-n} \\ 2 &= A \cdot \frac{f - n}{f \cdot n} \end{aligned}$$

Получаем:

$$A = \frac{2 \cdot n \cdot f}{f - n}$$

подставляем A для B:

$$1 = \frac{\frac{2 \cdot n \cdot f}{f - n}}{-f} + B$$

$$1 + \frac{\frac{2 \cdot n \cdot f}{f - n}}{f} = B$$

$$B = \frac{f + n}{f - n}$$

подставляем для z':

$$z' = \frac{2 \cdot n \cdot f}{f - n} \cdot \frac{1}{z} + \frac{f + n}{f - n}$$

мы работаем с однородными координатами:

$$\begin{aligned}x' &= \frac{x''}{w''} \\y' &= \frac{y''}{w''} \\z' &= \frac{z''}{w''} \\1 &= \frac{w''}{w''}\end{aligned}$$

формула преобразования:

$$(x'', y'', z'', w'') = (x, y, z, 1) \cdot \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix} = (w'' \cdot x', w'' \cdot y', w'' \cdot z', w'')$$

будем искать при $w'' = -z$

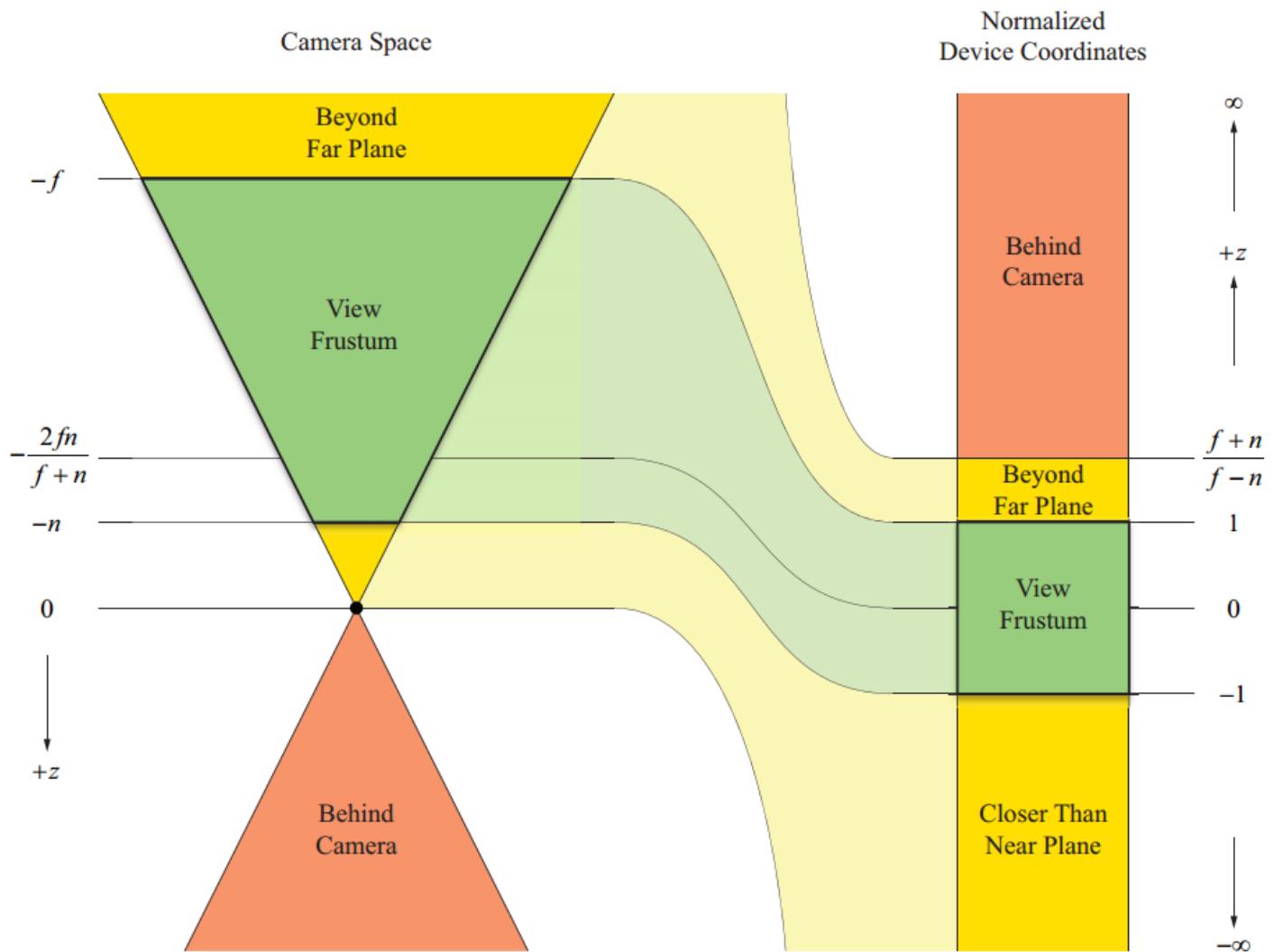
$$\begin{aligned}(-z) \cdot x' &= \frac{2 \cdot n}{r - l} \cdot x + \frac{r + l}{r - l} \cdot z \\(-z) \cdot y' &= \frac{2 \cdot n}{t - b} \cdot y + \frac{t + b}{t - b} \cdot z \\(-z) \cdot z' &= -\frac{2 \cdot n \cdot f}{f - n} - \frac{f + n}{f - n} \cdot z \\(-z) \cdot 1 &= -z\end{aligned}$$

Итоговая матрица:

$$\begin{pmatrix} \frac{2 \cdot n}{r - l} & 0 & 0 & 0 \\ 0 & \frac{2 \cdot n}{t - b} & 0 & 0 \\ \frac{r + l}{r - l} & \frac{t + b}{t - b} & -\frac{f + n}{f - n} & -1 \\ 0 & 0 & -\frac{2 \cdot n \cdot f}{f - n} & 0 \end{pmatrix}$$

Она называется View Frustum Matrix

MATR MatrFrustum(DBL l, DBL r, DBL b, DBL t, DBL n, DBL f);



Render in 3D

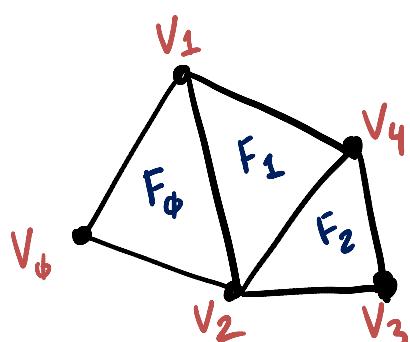
Rendering pipeline (конвейер вывода)

```
primitive -> {triangles ( $P_{012}$ )} -> { $P_i$ } * WorldCSmatrix * ViewCSmatrix * Projectionmatrix  

-> screen map (viewport transform) -> rasterize triangles
```

по реализации:

- представление данных



F -facet V - vertex
 $V = \{V_0, V_1, V_2, V_3, V_4\};$
 $I = \{\phi, 2, 1, 24, 1, 2, 3, 4\};$

будем использовать топологию TriMesh – сетка треугольников:

- * массив вершин:

VEC *V;

INT NumOfV;

- * массив троек индексов вершин для каждого треугольника:

INT *I; - будем группировать по 3

INT NumOfI;

тип:

```
typedef struct tagvg4PRIM
{
    VEC *V; /* Vertex array */
    INT NumOfV; /* Vertex array size */
    INT *I; /* Index array */
    INT NumOfI; /* Index array size */
} vg4PRIM;

!!! + VG4_RndMatrProj = MatrFrustum(left, right,
                                         bottom, top,
                                         ProjectDist, FarClip);
```

при построении:

```
MATR WVP = MatrMulMatr(M, MatrMulMatr(VG4_RndViewMatr, VG4_RndProjMatr));

/* Project all primitive vertices */
for (i = 0; i < Pr->NumOfV; i++)
{
    VEC P = PointTransform(Pr->V[i], WVP);

    pnts[i].x = (P.X + 1) * VG4_Anim.W / 2;
    pnts[i].y = (-P.Y + 1) * VG4_Anim.H / 2;
}
/* Draw triangles */
for (i = 0; i < Pr->NumOfI; i += 3)
{
    POINT p[3];

    p[0] = pnts[Pr->I[i]];
    p[1] = pnts[Pr->I[i + 1]];
    p[2] = pnts[Pr->I[i + 2]];
    Polygon(VG4_Anim.hDC, p, 3);
}
```

+ глобальные переменные:

```
MATR VG4_RndViewMatr, VG4_RndProjMatr;
DBL
VG4_RndProjSize = 1, /* Project plane unit square size */
VG4_RndProjDist = 0.1, /* Distance to project plane */
VG4_RndProjFarClip = 1000; /* Distance to project far clip plane */
```

функция установки проекции: RndSetProj

-= 14.06.2017 =-

.: 10:25 :.

OpenGL



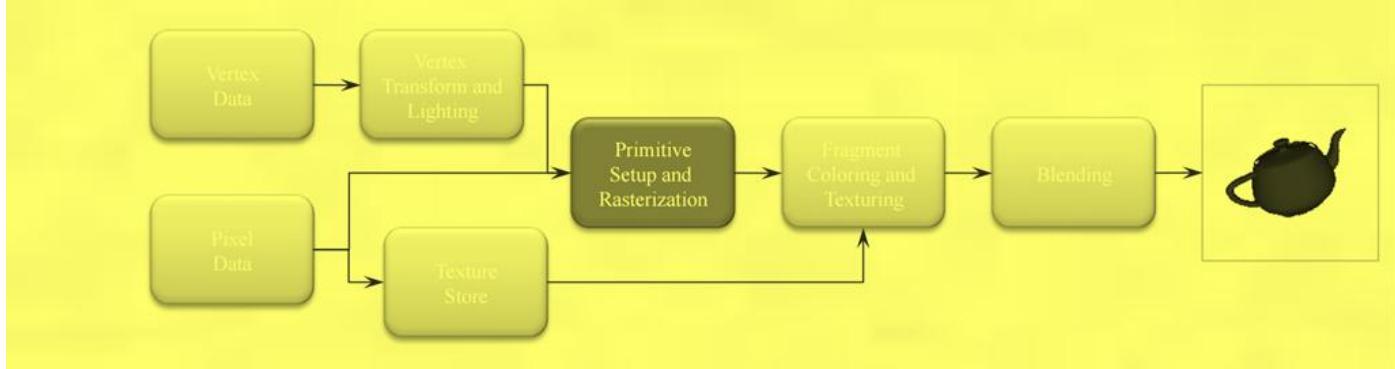
OpenGL (Open Graphics Library)

<http://www.opengl.org>

1. ВВЕДЕНИЕ

1.0 (1/07/1994)

Rendering Pipeline



Fixed Rendering Pipeline

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
glLoadIdentity();
glFrustum(-1, 1, -1, 1, 1, 100);
gluLookAt(5, 5, 5, 0, 0, 0, 0, 1, 0);
glRotated(90 * Time, 0, 1, 0);
```

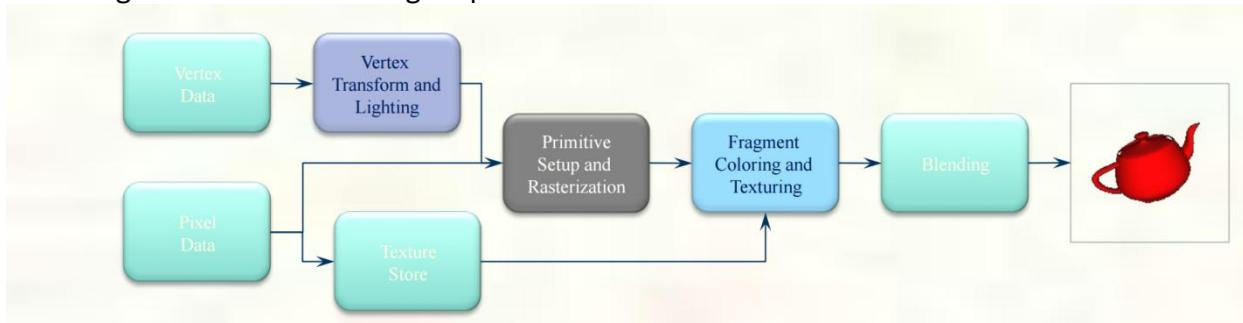
```
	glColor3d(1, 0, 0);
```

```
 glBegin(GL_TRIANGLES);
    glVertex3d(0, 0, 0);           3 - quantity, d - double
    glVertex3d(0, 1, 0);
    glVertex3d(1, 0, 0);
 glEnd();
```

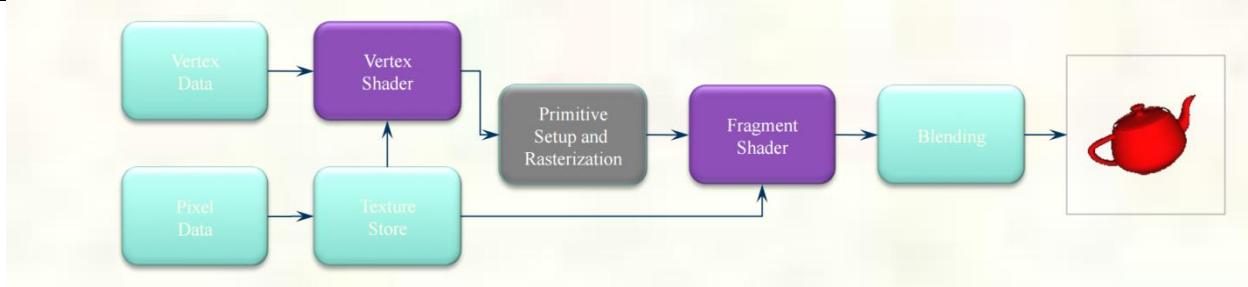
```
 glBegin(GL_TRIANGLES);
    glColor3d(1, 0, 0);
    glVertex3d(0, 0, 0);
    glColor3d(1, 0, 1);
    glVertex3d(1, 0, 0);
    glColor3d(1, 1, 0);
    glVertex3d(0, 1, 0);
 glEnd();
```

```
glFinish();
```

2.0 Programmable Rendering Pipeline



3.1 (24/3/2009)

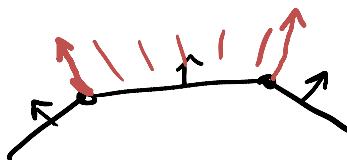


Shader - программа "тонирования"

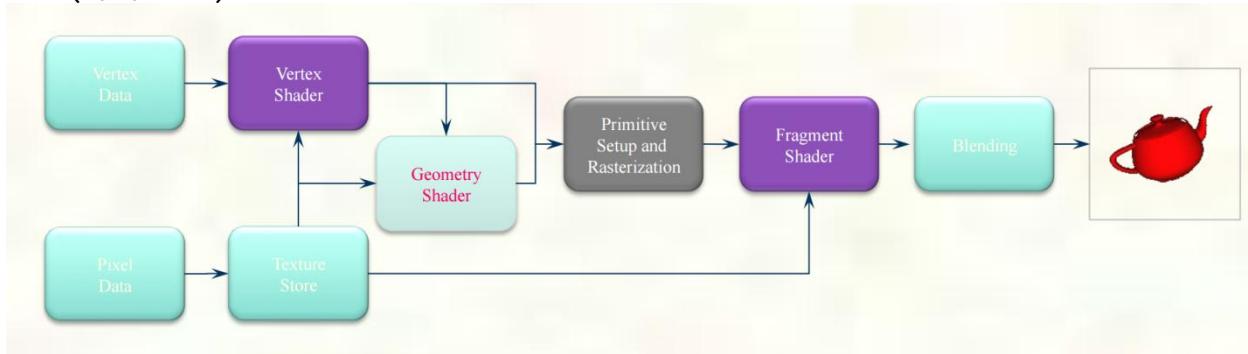
Shade - тонирование

Shading algorithms:

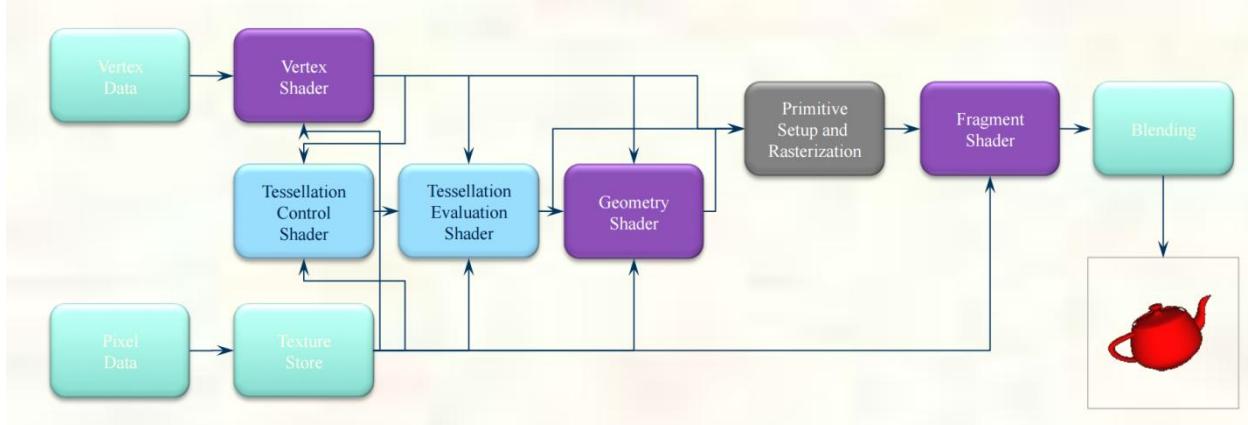
- Wireframe - каркас
- Flat - плоская закраска - грань рисуется одним цветом
- Gouraud - интерполяция цвета
- Phong - интерполяция нормалей



3.2 (3/8/2009)



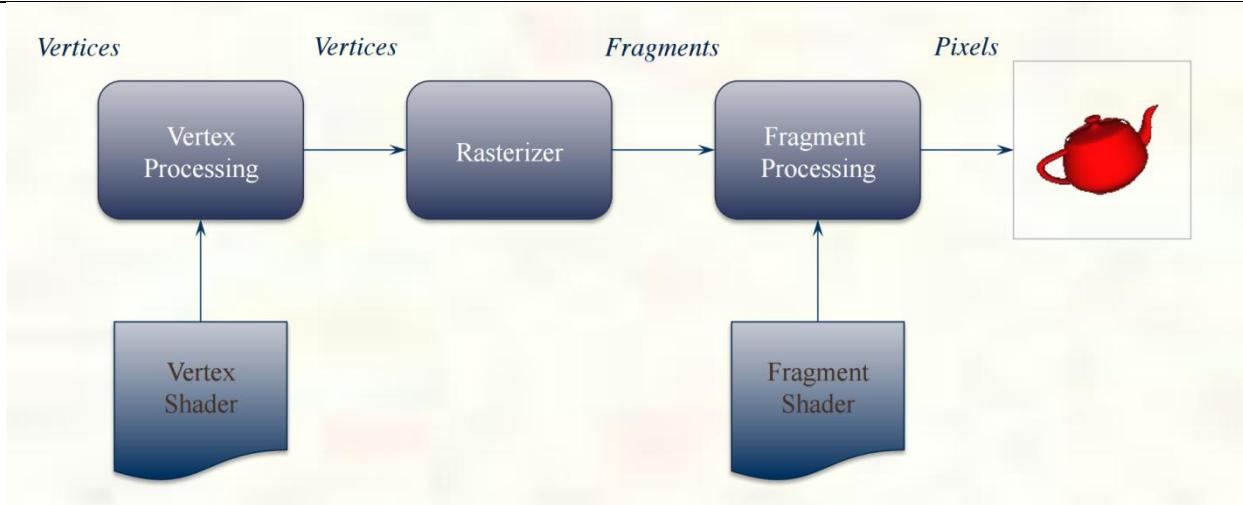
4.1 (25/7/2010)



4.5 (2014)

---> Vulkan (Direct3D)

Общий макет:



GLSL – OpenGL Shader Language

2. ЧТО НАДО

```

#include <gl/gl.h>
#include <gl/glu.h>

<+> библиотеки: opengl32.lib + glu32.lib

(/WINDOWS/System32/opengl32.dll glu32.dll)
(/WINDOWS/SysWOW64/opengl64.dll glu64.dll)
  
```

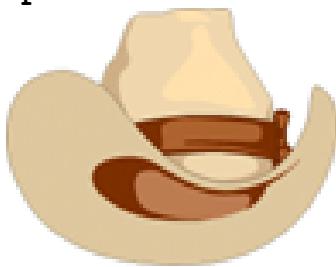
Extensions

```

void (*f)( void );
f = glGetFunction("Clear");
...
f();
  
```

!!! дополнительно - библиотека GLEW

OpenGL Extension Wrangler Library



<http://glew.sourceforge.net>

```

#define GLEW_STATIC  <-- отказ от glew32.dll
#include <glew.h>
glew32.lib
+
!!! glew32.dll -> WINDOWS/System32
  
```

внутри программы после инициализации OpenGL - выполняем инициализацию GLEW

```

if (glewInit() != GLEW_OK)
    return FALSE;
  
```

нам важно!!!

```
if (!(GLEW_ARB_vertex_shader && GLEW_ARB_fragment_shader))
    return FALSE;
```

3. Соглашения об именах:

GLname - типы - GLint GLfloat GLdouble GLuint GLbyte

glNameName - функции - glClearColor glShadeModel

GL_NAME_NAME - константы-параметры -

GL_COLOR_MATERIAL GL_COLOR_BUFFER_BIT

для функций из glu.h -> **GLUname gluNameName GLU_NAME_NAME**

для функций из glew.h -> **GLEWname glewNameName GLEW_NAME_NAME**

еще есть набор функций и т.п. для OpenGL под Windows (wgl.h):

WGLname wglNameName WGL_NAME_NAME

+ в концовках некоторых функций добавляется суффикс о количестве параметров и их типе:

```
glColor:
    glColor3d( GLdouble R, GLdouble G, GLdouble B );
        glColor3d(1.0, 1.0, 1.0);
    glColor4f( GLfloat R, GLfloat G, GLfloat B, GLfloat A );
        glColor4f(1.0F, 1.0F, 1.0F, 1.0F);
    glColor3ub( GLubyte R, GLubyte G, GLubyte B );
        glColor3ub(255, 255, 255);

    glColor3fv( GLfloat color[3] );
        GLfloat c[3] = {1.0f, 1.0f, 1.0f};
        glColor3fv(c);

    typedef struct
    {
        FLT R, G, B;
    } COLOR3;
    COLOR3 Col = {1, 0, 1};
    glColor3fv(&Col.R);

typedef struct
{
    DBL X, Y, Z;
} VEC;

VEC p = {1, 0, 30};
. . .
glVertex3d(p.X, p.Y, p.Z);
<=>
glVertex3dv(&p.X);
```

4. ИНИЦИАЛИЗАЦИЯ

ИЗМЕНЕНИЯ:

RENDER.H

```
/* OpenGL specific includes */
#define GLEW_STATIC
#include <glew.h>
#include <gl/gl.h>
#include <gl/glu.h>
```

ANIM.H

```
vg4ANIM:
    HDC hDC;      /* Drawing window context */
    HGLRC hGLRC; /* Rendering context */
```

RENDER.C

```
#pragma comment(lib, "opengl32")
#pragma comment(lib, "glu32")
#pragma comment(lib, "glew32s")
```

RndInit:

```
    . . .
    glEnable(GL_DEPTH_TEST);
    . . .
```

RndDraw:

```
    glLoadMatrixf(&WVP.A[0][0]);

    /* Draw all facets */
    glBegin(GL_TRIANGLES);
    for (i = 0; i < Obj->NumOff; i++)
        for (j = 0; j < 3; j++)
    {
        VEC p;

        p = Obj->V[Obj->F[i][j]];
        glVertex3fv(&p.X);
    }
    glEnd();
```

ANIM.C

AnimInit:

```
    INT i;
    . . .
    PIXELFORMATDESCRIPTOR pfd = {0};

    memset(&VG4_Anim, 0, sizeof(vg4ANIM));
```

```
    /* Store window and create memory device context */
    VG4_Anim.hWnd = hWnd;
    VG4_Anim.hDC = GetDC(hWnd);
```

```
. . .
```

```

/* OpenGL initialization: setup pixel format */
pfd.nSize = sizeof(pfd);
pfd.nVersion = 1;
pfd.dwFlags = PFD_DOUBLEBUFFER | PFD_SUPPORT_OPENGL;
pfd.cColorBits = 32;
pfd.cDepthBits = 32;
i = ChoosePixelFormat(VG4_Anim.hDC, &pfd);
DescribePixelFormat(VG4_Anim.hDC, i, sizeof(pfd), &pfd);
SetPixelFormat(VG4_Anim.hDC, i, &pfd);

/* OpenGL initialization: setup rendering context */
VG4_Anim.hGLRC = wglCreateContext(VG4_Anim.hDC);
wglMakeCurrent(VG4_Anim.hDC, VG4_Anim.hGLRC);
if (glewInit() != GLEW_OK ||
    !(GLEW_ARB_vertex_shader && GLEW_ARB_fragment_shader))
{
    wglGetCurrent(NULL, NULL);
    wglDeleteContext(VG4_Anim.hGLRC);
    ReleaseDC(VG4_Anim.hWnd, VG4_Anim.hDC);
    return FALSE;
}
. . .

```

AnimClose:

```

. . .
/* Delete OpenGL data */
wglGetCurrent(NULL, NULL);
wglDeleteContext(VG4_Anim.hGLRC);
/* Delete GDI data */
ReleaseDC(VG4_Anim.hWnd, VG4_Anim.hDC);

```

AnimResize:

```

. . .
glViewport(0, 0, W, H);
. . .

```

AnimCopyFrame:

```

SwapBuffers(VG4_Anim.hDC);

```

AnimRender:

```

. . .
/** Clear frame ***/
/* Clear background */
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

. . .

```

.:: 17:30 ::.

VBO - vertex buffer object

Представление геометрических данных в видеокарте.

для работы необходимы - массив вершин и буфер вершин (OpenGL):

vertex array	vertex buffer
(описание данных)	(сами данные)

массив вершин - **VertexArray** - связка посыпаемых данных сверху вниз (*Layout*).
буфер вершин - **VertexBuffer** - массив данных, отсыпаемых в видеокарту.

инициализационный этап

```
INT VA, VBuf;

glGenVertexArrays(1, &VA);
glGenBuffers(1, &VBuf);

/* делаем активным массив вершин */
glBindVertexArray(VA);
```

заносим данные в буфер вершин:

```
/* делаем активным буфер */
glBindBuffer(GL_ARRAY_BUFFER, VBuf);
/* сливаем данные (NumOfV - количество вершин, V - массив вершин) */
glBufferData(GL_ARRAY_BUFFER, sizeof(vg4VERTEX) * NumOfV, V, GL_STATIC_DRAW);
```

указываем в массиве вершин буфер и какие данные содержит:
все данные – FLT (float):

```
typedef struct tagvg4VERTEX
{
    VEC P; /* Vertex position */
    VEC2 T; /* Vertex texture coordinates */
    VEC N; /* Vertex normal */
    VEC4 C; /* Vertex color (rgba) */
} vg4VERTEX;
```

```
typedef struct tagVEC2
{
    FLT X, Y; /* X,Y or U,V (common case) or S,T (OpenGL style) */
} VEC2;
typedef struct tagVEC4
{
    FLT X, Y, Z, W; /* X,Y,Z,W or R,G,B,A */
} VEC2;
```

```
/* делаем активным буфер */
glBindBuffer(GL_ARRAY_BUFFER, VBuf);

/* задаем порядок данных */
/* Layout (номер атрибута),
   количество компонент,
   mun,
   надо ли нормировать,
   размер в байтах одного элемента буфера (stride),
   смещение в байтах до начала данных */
glVertexAttribPointer(0, 3, GL_FLOAT, FALSE, sizeof(vg4VERTEX),
                      (VOID *)0); /* позиция */
glVertexAttribPointer(1, 2, GL_FLOAT, FALSE, sizeof(vg4VERTEX),
                      (VOID *)sizeof(VEC)); /* текстурные координаты*/
glVertexAttribPointer(2, 3, GL_FLOAT, FALSE, sizeof(vg4VERTEX),
                      (VOID *)(sizeof(VEC) + sizeof(VEC2))); /* нормаль */
glVertexAttribPointer(3, 4, GL_FLOAT, FALSE, sizeof(vg4VERTEX),
                      (VOID *)(sizeof(VEC) * 2 + sizeof(VEC2))); /* цвет */

/* включаем нужные атрибуты (Layout) */
 glEnableVertexAttribArray(0);
```

```
glEnableVertexAttribArray(1);
glEnableVertexAttribArray(2);
glEnableVertexAttribArray(3);

/* выключили массив вершин */
glBindVertexArray(0);
```

удаление

```
/* делаем активным массив вершин */
glBindVertexArray(VA);
/* "отцепляем" буфер */
glBindBuffer(GL_ARRAY_BUFFER, 0);
glDeleteBuffers(1, &VBuf);
/* делаем неактивным массив вершин */
glBindVertexArray(0);
glDeleteVertexArrays(1, &VA);
```

Индексы:

инициализационный этап

```
INT IBuf;

. . .

glGenBuffers(1, &IBuf);
```

заносим данные в буфер индексов:

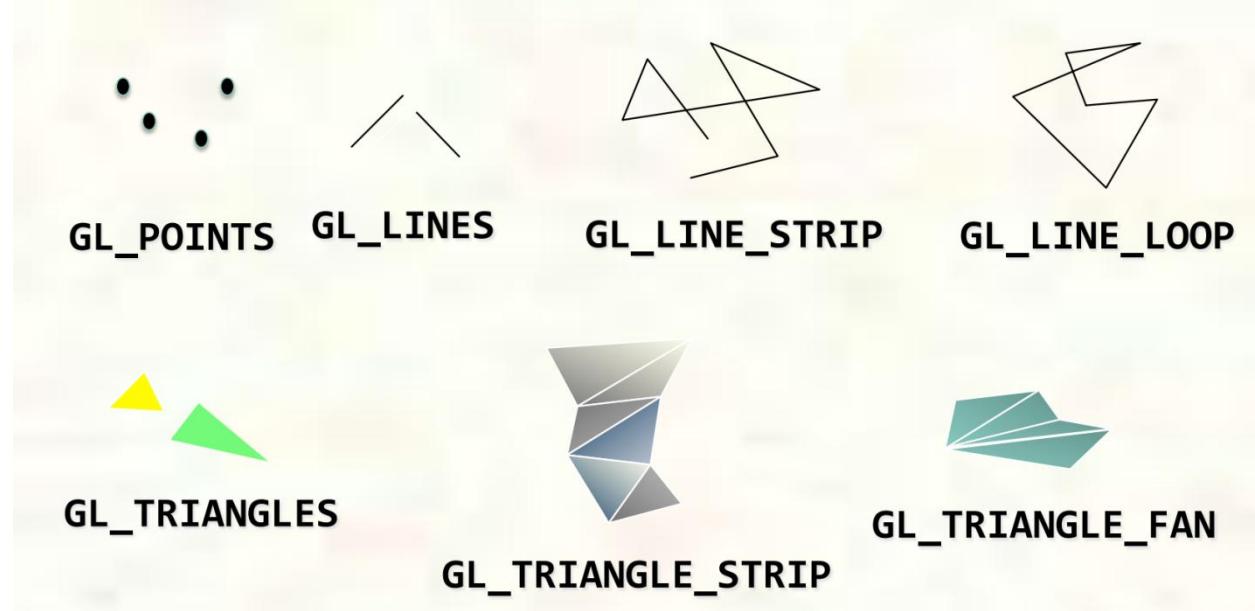
```
/* делаем активным буфер */
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, IBuf);
/* сливаем данные (NumOfI - количество индексов, I - массив индексов) */
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(INT) * NumOfI, I, GL_STATIC_DRAW);
```

удаление

```
glDeleteBuffers(1, &IBuf);
```

Рисование:

Примитивы OpenGL



```
/* делаем активным массив вершин */
glBindVertexArray(VA);
/* делаем активным массив индексов */
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, IBuf);
/* отрисовка */
glDrawElements(GL_TRIANGLES, NumOfI, GL_UNSIGNED_INT, NULL);
```

У нас изменения:

```
-- добавляем типы VEC2, VEC4, vg4VERTEX;
??? Vec2Set(X, Y), Vec2Set1(V)
??? Vec4Set(X, Y, Z, W), Vec4Set1(V)
```

-- добавляем тип хранения примитива:

```
typedef struct tagvg4PRIM
{
    BOOL IsTrimesh; /* TRUE - trimesh, FALSE - triangle strip */
    INT NumOfI;      /* Number of indices */
    MATR M;          /* Primitive transformation matrix */

    /* VBO data */
    INT VA, VBuf;
    /* Index buffer */
    INT IBuf;
} vg4PRIM;
```

-- план по функциям:

```
/* Create primitive function.
 * ARGUMENTS:
 * - created primitive:
 *   vg4PRIM *Pr;
 * - trimesh flag:
 *   BOOL IsTrimesh;
 * - vertex array:
 *   vg4VERTEX *V;
 * - vertex array size:
 *   INT NumOfV
 * - index array:
 *   INT *I;
 * - index array size:
 *   INT NumOfI;
 * RETURNS: None.
 */
```

```
VOID VG4_RndPrimCreate( vg4PRIM *Pr, BOOL IsTrimesh,
                        vg4VERTEX *V, INT NumOfV,
                        INT *I, INT NumOfI )
```

```
{ . . .
    Pr->NumOfI = NumOfI;
} /* Ens of 'VG4_RndPrimCreate' function */
```

```
/* Free primitive function.
 * ARGUMENTS:
 * - deleted primitive:
 *   vg4PRIM *Pr;
 * RETURNS: None.
 */
```

```

VOID VG4_RndPrimFree( vg4PRIM *Pr )
{
} /* Ens of 'VG4_RndPrimFree' function */

#define MatrMulMatr3(A, B, C) \
    (MatrMulMatr(A, MatrMulMatr(B, C)))
#define MatrMulMatr4(A, B, C, D) \
    (MatrMulMatr(A, MatrMulMatr3(B, C, D)))

/* Draw primitive function.
* ARGUMENTS:
*   - drawing primitive:
*     vg4PRIM *Pr;
*   - transformation matrix:
*     MATR M;
* RETURNS: None.
*/
VOID VG4_RndPrimDraw( vg4PRIM *Pr, MATR M )
{
    M = MatrMulMatr4(Pr->M, M, VG4_RndMatrView, VG4_RndMatrProj);
    glLoadMatrixf(M.A[0]);
    . . .
} /* Ens of 'VG4_RndPrimDraw' function */

```

-- объект – массив примитивов:

типа:

```
typedef struct tagvg4OBJ
```

```
{
    vg4PRIM *P; /* Primitive array */
    INT NumOfP; /* Primitive array size */
} vg4OBJ;
```

```
/* Create object function.
```

```
* ARGUMENTS:
*   - created object:
*     vg4OBJ *Obj;
*   - number of primitives:
*     INT NumOfP;
* RETURNS: None.
*/
```

```
VOID VG4_RndObjCreate( vg4OBJ *Obj, INT NumOfP )
```

```
{
    memset(Obj, 0, sizeof(vg4PRIM));
    if ((Obj->P = malloc(sizeof(vg4PRIM) * NumOfP)) == NULL)
        return;
    memset(Obj->P, 0, sizeof(vg4PRIM) * NumOfP);
    Obj->NumOfP = NumOfP;
} /* Ens of 'VG4_RndObjCreate' function */
```

```
/* Free object function.
```

```
* ARGUMENTS:
*   - deleted object:
*     vg4OBJ *Obj;
* RETURNS: None.
```

```

/*/
VOID VG4_RndObjFree( vg4OBJ *Obj )
{
    INT i;

    for (i = 0; i < Obj->NumOfP; i++)
        VG4_RndPrimDelete(&Obj->P[i]);
    free(Obj->P);
    memset(Obj, 0, sizeof(vg4PRIM));
} /* Ens of 'VG4_RndObjFree' function */

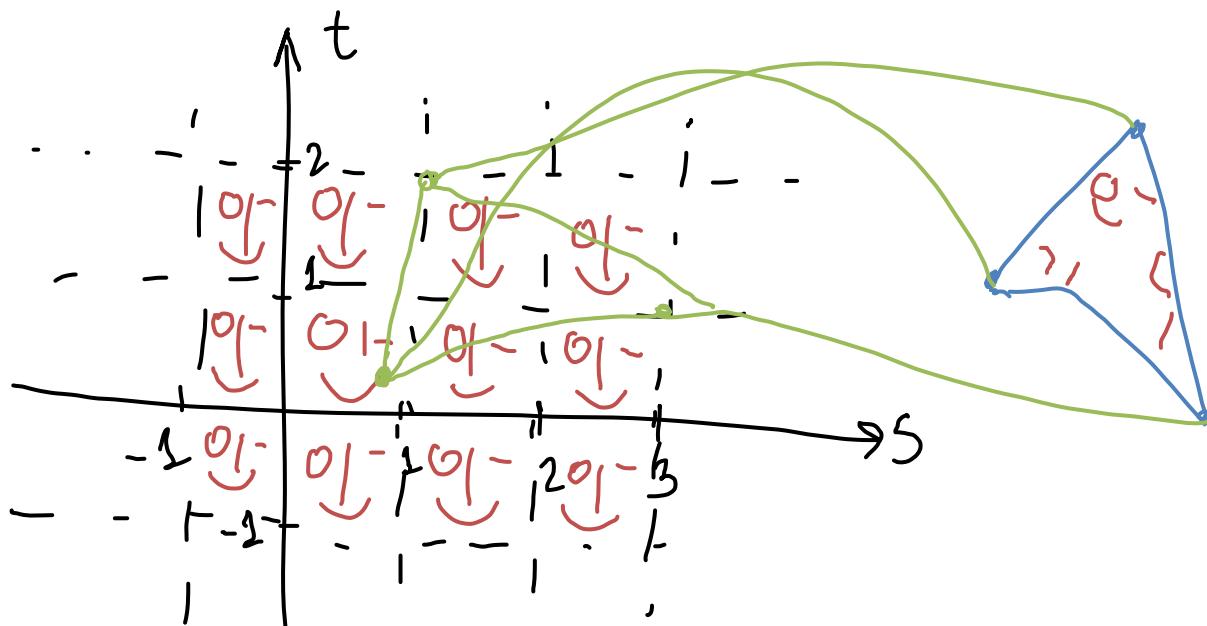
/* Draw primitive function.
 * ARGUMENTS:
 * - drawing object:
 *     vg4OBJ *Obj;
 * - transformation matrix:
 *     MATR M;
 * RETURNS: None.
 */
VOID VG4_RndObjDraw( vg4OBJ *Obj, MATR M )
{
    INT i;

    for (i = 0; i < Obj->NumOfP; i++)
        VG4_RndPrimDraw(&Obj->P[i], M);
} /* Ens of 'VG4_RndObjDraw' function */

```

-= 17.08.2017 =-

Текстурирование



OpenGL:



