-= 02.06.2014 =-

staff:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | ФИ | Класс | телефон | E-mail |
| 01 | Носков Иван | 10-1 |  | Vanyamagov@yandex.ru |
| 02 | Корнеев Олег | 10-2 |  |  |
| 03 | Шульц Святослав | 10-3 | +79218938047 | Svsch9@yandex.ru |
| 04 | Сокол Алексей | 10-4 | +79119747062 | kanootoko@gmail.com |
| 05 | Дунаева Ольга | 10-6 |  |  |
| 06 | Пантелеева Алиса | 10-6 |  |  |
| 07 | Яковлев Михаил | 10-6 | +79215589606 | yakovmikemikle@gmail.com |
| 08 | Гаевой Никита | 10-1 | +79219521012 | nikgaevoy@mail.ru |
| 09 | Абрамова Мария | 10-4 | +79214026088 | maria.abramova1545@gmail.com |
| 10 | Чепцов Василий | 10-4 | +79522401016 | vcheptsov97@gmail.com |

10:00 - start

SPR?? - account

***Far 3 - запуск***

Shift+F4 -> ***start.bat***

>>> net use q: \\server\shspr

F7 -> INFO

***Z:\INFO\***

Shift+F4 -> ***dirinfo***

>>> Familija Imja

>>> 10-?

>>> dd.mm.yyyy

>>> SPR??

>>> E-mail

11:40 start winapi

12:25 WM\_PAINT:

case WM\_PAINT:

- сообщение перерисовки окна

HDC hDC;

PAINTSTRUCT ps;

case WM\_PAINT:

hDC = BeginPaint(hWnd, &ps);

. . .

SetPixel(hDC, 100, 100, RGB(0, 255, 0));

. . .

EndPaint(hWnd, &ps);

получение размера окна:

RECT rc;

GetClientRect(hWnd, &rc);

w = rc.right;

h = rc.bottom;

смена цвета рисования

SelectObject(hDC, GetStockObject(NULL\_PEN));

SelectObject(hDC, GetStockObject(DC\_BRUSH));

SetDCBrushColor(hDC, RGB(255, 0, 0));

или

SelectObject(hDC, GetStockObject(DC\_PEN));

SetDCPenColor(hDC, RGB(255, 0, 0));

перерисовка - отсылка сигнала

InvalidateRect(hWnd, NULL, FALSE/TRUE очистка);

T01FWIN - добавить "следящие глаза"

про "мышь":

1.от сообщений

WM\_MOUSEMOVE:

WM\_LBUTTONDOWN:

WM\_LBUTTONUP:

R

M

wParam - состояние кнопок:

if (wParam & MK\_LBUTTON) . . .

lParam - координаты мыши в окне

x = (SHORT)LOWORD(lParam);

y = (SHORT)HIWORD(lParam);

2.от таймера

- при создании окна - зарегистрировать "таймер"

- приходит WM\_CREATE (надо вернуть 0, -1 - для отмены создания окна)

SetTimer(hWnd, TimerNo, TimeInMs, NULL);

(в WM\_DESTROY -> KillTimer(hWnd, TimerNo);

будет приходить сообщение WM\_TIMER:

- опросить мышу:

POINT pt;

GetCursorPos(&pt); - глобальные координаты

ScreenToClient(hWnd, &pt); - преобразование координат в СК окна

x = pt.x;

DrawEye(hDC, pt.x, pt.y, ... );

void DrawEye( HDC hDC, INT Xc, INT Yc, INT W, INT H )

{

INT x, y;

E(0, 0, W / 2, H);

E(W / 2, 0, W, H);

len = ... ;

x = W / 4;

y = H / 2;

co = (Xc - x) / len;

si = ... ;

xe = x + co \* len;

ye = y + si \* len;

E(xe - r, ye - r, xe + r, ye + r);

}

!!! InvalidateRect(hWnd, NULL, TRUE);

17:00 про GIT

Git - www.git-scm.com

SmartGit - http://www.syntevo.com/smartgithg/

GIT

- repository -

- stage - пометить на изменение

- commit - отослать помеченное

- www.github.com - зарег.

>>> www.gravatar.org

Д.з. - полностью установить Git SmartGit, GitHub,

Gravatar

-= 03.06.2014 =-

повтор:

**WM\_CREATE:**

CREATESTRUCT \*cs;

cs = (CREATESTRUCT \*)lParam;

cs->hInstance ... - все параметры из CreateWindow

**WM\_SIZE:** - при изменении размера окна

INT w = LOWORD(lParam), h = HIWORD(lParam);

**WM\_CLOSE:** - при попытке зактрыть окно

можно отказать, можно вызвать DestroyWindow(hWnd);

**WM\_DESTROY:** - при уничтожении (сообщение о случившемся)

- для главного окна программы PostQuitMessage(0);

**WM\_PAINT:** - для восстановления содержимого

PAINTSTRUCT ps;

HDC hDC;

hDC = BeginPaint(hWnd, &ps);

. . .

EndPaint(hWnd, &ps);

инициировать полную перерисовку -

InvalidateRect(hWnd, NULL, TRUE); TRUE - все закрасить фоном

**WM\_ERASEBKGND:**

- перерисовка фона

HDC hDC = (HDC)wParam;

return 1; - при закраске

**WM\_TIMER:**

- сигнал от таймера

INT TimerNo = wParam;

создать: SetTimer(hWnd, TimerNo, DelayTimeInMs, NULL);

уничтожить: KillTimer(hWnd, TimerNo);

**WM\_MOUSEMOVE:** - перемещение

**WM\_LBUTTONDOWN:** - нажатие

**WM\_LBUTTONUP:** - отпускание MBUTTON RBUTTON

**WM\_LBUTTONDBLCLK:** - двойное нажатие !!! в классе окна

wc.style = ... | CS\_DBLCLK;

**WM\_MOUSEWHEEL:** - кручение колеса

INT z = (SHORT)HIWORD(wParam); - колесико

для всех мышиных сообщений:

INT

x = (SHORT)LOWORD(lParam);

y = (SHORT)HIWORD(lParam);

в wParam - биты кнопок:

MK\_LBUTTON ...

MK\_SHIFT

MK\_CONTROL

!!! SetCapture(hWnd); - заставляет все сообщения от мыши засылать в указанное окно

SetCapture(NULL); - отмена

**WM\_KEYDOWN:** - нажатие

Key = wParam;

VK\_LEFT VK\_UP ...

'A' 'B' ... 'Z'

'1' '2'

(WORD)lParam - счетчик автоповтора

WORD == unsigned short

**WM\_KEYUP:** - отжатие

**WM\_CHAR:** - нажатие символа

CHAR ch = wParam; <-- TranslateMessage(&msg) в цикле сообщений

- 10:45

Двойная буферизация

совместимый контекст - контекст для рисования в памяти:

HDC hMemDC = CreateCompatibleDC(hDC); hDC - контекст окна

удалить - DeleteDC(hMemDC);

создание изображения в памяти для рисования:

HBITMAP hBm = CreateCompatibleBitmap(hDC, W, H);

удалить DeleteObject(hBm);

выбираем его в контекст для рисования:

SelectObject(hMemDC, hBm);

LineTo(hMemDC, 30, 30);

рисование контекста в дрyгой контекст:

bit-block-transfer

BitBlt(hDC\_куда, Xкуда, Yкуда, Wкуда, Hкуда,

hDC\_откуда, Xоткуда, Yоткуда, SRCCOPY);

StretchBlt(hDC\_куда, Xкуда, Yкуда, Wкуда, Hкуда,

hDC\_откуда, Xоткуда, Yоткуда,

Wоткуда, Hоткуда, SRCCOPY);

**SetStretchBltMode(hDC, режим);**

**режимы -** COLORONCOLOR или HALFTONE или BLACKONWHITE или WHITEONBLACK

`

12:00

- про глаза

T02CLOCK

- часы со стрелками - циферблат - faceclock

!!!

POINT pt[5] = {{1, 2}, {11, 30}, ... };

Polygon(hDC, pt, 5);

12:40 - access to Git (register and create repository)

T02CLOCK

HFONT hFnt;

**hFnt = CreateFont(211, 0, 50, 0, FW\_BOLD,**

**TRUE, FALSE, FALSE, RUSSIAN\_CHARSET,**

**OUT\_DEFAULT\_PRECIS, CLIP\_DEFAULT\_PRECIS,**

**PROOF\_QUALITY, VARIABLE\_PITCH | FF\_ROMAN,**

**"Times");**

**-= 04.06.2014 =-**

**Рисование стрелок (hand draw)**

**1.поворот точки вокруг начала координат**

**x=r·cos(β)**

**y=r·sin(β)**

**x'=r·cos(β+α)=r·cos(β)·cos(α)-r·sin(β)·sin(α)=x·cos(α)-y·sin(α)**

**y'=r·sin(β+α)=r·cos(β)·sin(α)+r·sin(β)·cos(α)=x·sin(α)+y·cos(α)**

**VOID DrawArrow( HDC hDC, INT Xc, INT Yc, INT L, INT W, FLOAT Angle )**

**{**

**INT i;**

**POINT pts[] =**

**{**

**{0, -W}, {-W, 0}, {0, L}, {W, 0}**

**}, pts\_draw[sizeof pts / sizeof pts[0]];**

**FLOAT si = sin(Angle), co = cos(Angle);**

**for (i = 0; i < sizeof pts / sizeof pts[0]; i++)**

**{**

**pts\_draw[i].x = Xc + pts[i].x \* co - pts[i].y \* si;**

**pts\_draw[i].y = Yc + pts[i].x \* si + pts[i].y \* co;**

**}**

**Polygon(hDC, pts\_draw, sizeof pts / sizeof pts[0]);**

**}**

**11:00**

**Full Screen - переключение в полноэкранный режим**

**11:30 Линейная алгебра**

**Матрицы**

**1.операции:**

**+**

**\*на число**

**умножение матриц:**

**Anm · Bmk**

**(Anm · Bmk)T= BmkT · AnmT**

**:25**

**2.определитель матрицы (Matrix Determinator)**

**1.перестановки - permutation**

**1..n**

**n!**

**транспозиция - transposition**

**1,2,3 -> 1,3,2**

**инверсия - inversion - больший предшествует меньшему (лексиграфический порядок)**

**1,2,3 -> число инверсий 0**

**1,3,2 -> число инверсий 1**

**2,1,3 -> число инверсий 1**

**2,3,1 -> число инверсий 2**

**3,1,2 -> число инверсий 2**

**3,2,1 -> число инверсий 3**

**четность перестановки == четность числа инферсий**

**при транспозиции четность перестановки меняется на противоположную:**

**1.**

**A1, A2, ... , Ai+1, Ai+2, ... , An**

**A1, A2, ... , Ai+2, Ai+1, ... , An**

**Ai+1>Ai+2  число инверсив при Ai+2, Ai+1 --**

**Ai+1<Ai+2  число инверсив при Ai+2, Ai+1 ++**

**2.**

**A1,A2,...,**Ai**,Ai+1,Ai+2,...,**Aj**,Aj+1,Aj+2,...,An**

**- за (j-i) раз переставим Ai dgthtl**

**A1,A2,...,Ai+1,Ai+2,...,**Ai**,**Aj**,Aj+1,Aj+2,...,An**

**- обмен**

**A1,A2,...,Ai+1,Ai+2,...,**Aj**,**Ai**,Aj+1,Aj+2,...,An**

**- за (j-i) раз переставим Aj назад**

**A1,A2,...,**Aj**,Ai+1,Ai+2,...,**Ai**,Aj+1,Aj+2,...,An**

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

**1,2,3,4, ... ,n**

T03PERM

получить все перестановки из n элементов их четность

*подзадача - перестановки получить в лексикографическом порядке*

рекомендации:

int P[N] = {1, 2, 3, ... , N};

void Go( int Pos )

{

по очереди в позицию Pos переставляем все элементы из массива

+ вызываем эту же функцию для Pos+1

выход из рекурсии

if (Pos == N)

-> выводим массив P

else

{

for (i->Pos..N-1)

Swap() и Go(Pos+1)

}

}

дозапись в файл:

#include <stdio.h>

void Write( void )

{

FILE \*F;

if ((F = fopen("a.log", "a")) != NULL)

{

fprintf(F, "%i\n", 30);

fclose(F);

}

}

Go(Pos + 1);

for (i = Pos + 1; i < N; i++)

{

Swap(&P[Pos], &P[i]);

Parity = !Parity;

Go(Pos + 1);

Swap(&P[Pos], &P[i]);

Parity = !Parity;

}

**INT CountParity**( **VOID** )

{

**INT i**, **j**, **cnt** = **0**;

**for** (**i** = **0**; **i** < **N** - **1**; **i**++)

**for** (**j** = **i** + **1**; **j** < **N**; **j**++)

**cnt** += **P**[**i**]> **P**[**j**];

**return cnt**;

}/\* End of 'CountParity' function \*/

13:25

2.определитель матрицы



**determinant**



16:00

**T04DETERM**

вычислить определитель матрицы по определению

матрица - из файла:

#define MAX 10

double A[MAX][MAX];

int P[MAX], Parity;

int N;

void Load( char \*FileName )

{

FILE \*F;

int i, j;

N = 0;

F = fopen(FileName, "r");

if (F == NULL)

return;

fscanf(F, "%i", &N);

if (N < 0)

N = 0;

else

if (N > MAX)

N = MAX;

for (i = 0; i < N; i++)

for (j = 0; j < N; j++)

fscanf(F, "%lf", &A[i][j]);

}

как считать? - во время получения перестановок:

double GlobalDeterminant = 0; глобальная переменная с результатом - определителем

. . .

if (Pos == N)

{

double prod = 1;

for (i = 0; i < N; i++)

prod \*= A[i][***P[i]***];

если четная перестановка

GlobaDeterminant += prod;

иначе

GlobaDeterminant -= prod;

}

!!! изначально в P[...] - числа от ***0*** до ***N-1***

где-то в main:

LoadMatrix("mmm.txt");

17:30

свойства определителей



-= 05.06.2014 =-

multi-display fullscreen

10:30 метод Гаусса для нахождения определителя матрицы

суть - приведение матрицы к верхне-треугольному виду. Тогда определить равен произведению элементов на главной диагонали.

Алгоритм:

выполняем N итераций для получения ненулевого элемента на позиции Aii­:

1.находим строку с Ai1 и обмениваем эту строку с 1-ой (общий знак - меняем на противоположный: sign=-sign - если строки менялись)

2.из всех строк (j), начиная со 2-й по N вычитаем 1-ю строку, умноженную Aj1/Ai1

3.аналогично с отсальными позициями: 2 ... N-1

замечание: выбираем строку с максимальным элементом в A?i позиции

12:00 - common help

поиск максимального элемента

maxi = i;

maxj = i;

for (ki = i + 1; ki < N; ki++)

for (kj = i + 1 ; kj < N; kj++)

if (A[ki][kj] > A[maxi][maxj])

maxi = ki, maxj = kj;

обмен строки-столбца

if (i != maxi)

SwapRow(i, maxi), sign = -sign;

if (i != maxj)

SwapRow(i, maxj), sign = -sign;

if (A[i][i] == 0) !!! fabs(A[i][i]) < 0.000001 <- threshold | tolerance

return 0;

for (k = i + 1; k < N; k++)

for (j = i; j < N; j++)

A[k][j] -= A[i][j] \* A[k][i] / A[i][i];

12:30

**алгебраическое дополнение элемента**



**минор элемента**



существует связь между алгебраическим дополнение и минором элемента матрицы:

**Aij***=(-1)i+j⋅***Mij**

Обратная матрица

An · Bn = En в англоязычной литературе In identity

Bn · An = En

Введем в рассмотрение присоединенную матрицу

(adjoint matrix - *Adj* **A**)

транспонированная матрица, составленная из алгебраических дополнений соответствующих элементов:

умножим на нее матрицу:

ai1·Aj1+ ai2·Aj2+...+ain·Ajn=?

ответ: |A| если i==j

иначе это

т.е. получили:

применение - СЛАУ - системы алгебраических линейных уравнений:

x1·a11+x2·a12+...+xn·a1n=b1

x1·a21+x2·a22+...+xn·a2n=b2

. . .

x1·an1+x2·an2+...+xn·ann=bn

в матричном виде:

Ann·Xn1=Bn1

Ann-1·Ann·Xn1=Ann-1·Bn1 ==> Xn1=Ann-1·Bn1

x1=(A11·b1+A21·b2+...+An1·bn)/|A|

...

xi=(A1i·b1+A2i·b2+...+Ani·bn)/|A|

x + y = 5

x - 3 \* y = -7

15:15

Вектора

линейная зависимость и независимость векторов

- линейная комбинация:

***λ***1⋅**a**1**+*λ***2⋅**a2+…+*λ***n⋅**an**

тривиальная

Л.З.

не все ***λ***i==0

***λ***1⋅**a**1**+*λ***2⋅**a2+…+*λ***n⋅**an=O**

ThЛюбые три вектора на плоскости линейно зависимы

1.любые два коллинеарны - ...

2.] все неколлинеарны между собой

аналогично - любые 4 вектора в пространстве - Л.З.

базис и координаты

б.на плоскости - упорялоченная двойка неколлинеарных векторов

б.в пространстве - упорялоченная тройка некомпланарных векторов

б. наз. ортогональным, если все б.в. порпарно ортогональны

угол - π/2

плоскость:

**l**1, **l**2 — базис

**x**=*x*1⋅**l**1+*x*2⋅**l**2

пространство:

**l**1, **l**2, **l**3 — базис

**x**=*x*1⋅**l**1+*x*2⋅**l**2+*x*3⋅**l**3

Ось

Скалярное произведение векторов (dot product)

a·b=|a|·|b|·cos∠(a,b)

Рассмотрим ортонормированный базис:

**l**1, **l**2 — базис на плоскости, |**l**1|=|**l**2|=1

**l**1**⊥l**2

**a**=*a*1⋅**l**1+*a*2⋅**l**2

**b**=*b*1⋅**l**1+*b*2⋅**l**2

**a**⋅**b=(***a*1⋅**l**1+*a*2⋅**l**2**)**⋅**(***b*1⋅**l**1+*b*2⋅**l**2**)=**

*a*1⋅**l**1⋅*b*1⋅**l**1+*a*2⋅**l**2⋅*b*1⋅**l**1+ *a*1⋅**l**1⋅*b*2⋅**l**2+*a*2⋅**l**2⋅*b*2⋅**l**2=*a*1⋅*b*1+*a*2⋅*b*2

**l**1⋅**l**1=1

**l**1⋅**l**2=0

**т.е. угол между векторами:**



Векторное произведение векторов (cross product)

**l**1, **l**2, **l**3 — базис на плоскости, |**l**1|=|**l**2|=|**l**3|=1

**l**1**⊥l**2 **l**1**⊥l**3 **l**3**⊥l**2

**a**=*a*1⋅**l**1+*a*2⋅**l**2+*a*3⋅**l**3

**b**=*b*1⋅**l**1+*b*2⋅**l**2+*b*3⋅**l**3

**a×b=**

**(***a*1⋅**l**1+*a*2⋅**l**2+*a*3⋅**l**3**)×(***b*1⋅**l**1+*b*2⋅**l**2+*b*3⋅**l**3**)=**

*a*1⋅**l**1**×***b*1⋅**l**1+*a*1⋅**l**1**×***b*2⋅**l**2+*a*1⋅**l**1**×***b*3⋅**l**3+

*a*2⋅**l**2**×***b*1⋅**l**1+*a*2⋅**l**2**×***b*2⋅**l**2+*a*2⋅**l**2**×***b*3⋅**l**3+

*a*3⋅**l**3**×***b*1⋅**l**1+*a*3⋅**l**3**×***b*2⋅**l**2+*a*3⋅**l**3**×***b*3⋅**l**3=

**l**1**×l**3=**-l**3**×l**1=-**l**2

**l**1**×l**2=**-l**2**×l**1=**l**3

**l**2**×l**3=**-l**3**×l**2=**l**1

**l**1**×l**1=0

**l**2**×l**2=0

**l**3**×l**3=0

*a*1⋅*b*2⋅**l**1**×l**2+*a*1⋅*b*3⋅**l**1**×l**3+

*a*2⋅*b*1⋅**l**2**×l**1+*a*2⋅*b*3⋅**l**2**×l**3+

*a*3⋅*b*1⋅**l**3**×l**1+*a*3⋅*b*2⋅**l**3**×l**2=

*a*1⋅*b*2⋅**l**3-*a*1⋅*b*3⋅**l**2+

*-a*2⋅*b*1⋅**l**3+*a*2⋅*b*3⋅**l**1+

*a*3⋅*b*1⋅**l**2-*a*3⋅*b*2⋅**l**1=

**l**1⋅(*a*2⋅*b*3-*a*3⋅*b*2)+

**l**2⋅(*a*3⋅*b*1-*a*1⋅*b*3)+

**l**3⋅(*a*1⋅*b*2*-a*2⋅*b*1)=

Смешанное произведение векторов

(**a**×**b**)⋅**c**



!!!

Двойное векторное произведение

**a**×(**b**×**c)**=**b**⋅(**a**⋅**c**)-**c**⋅(**a**⋅**b**)

16:50

T05GLOBE

+7 921 752 7150

д.з. http://www.school30.spb.ru/cgsg/cgc/

***про математику***

!!!

про клавиатуру:

BYTE Keys[256];

GetKeyboardState(Keys);

if (Key[VK\_SPACE] & 0x80)

...

if (Key['S'] & 0x80)

...

-= 06.06.2014 =-

12:20 affine coordinate transformation

радиус вектор (x y 1)

свободный вектор (x y 0)

ковариантные и контравариантные вектора

--> vec plan

VEC MATR CAMERA

**Реализация — файлы:** VEC.H VEC.C

#include<math.h>

#define **PI 3.14159265358979323846**

#define **D2R**(**A**)((**A**)\*(**PI** / **180.0**))

/\* базовый вещественный тип \*/

**typedef double DBL**;

/\* тип для вектора в простанстве \*/

**typedef struct tagVEC**

{

**DBL X**, **Y**, **Z**;

} **VEC**;

/\* тип для матрицы - массив в структуре \*/

**typedef struct tagMATR**

{

**DBL A[4][4]**;

} **MATR**;

MATR MatrTranslate( DBL Dx, DBL Dy, DBL Dz )

{

MATR A =

{

{

{ 1, 0, 0, 0},

{ 0, 1, 0, 0},

{ 0, 0, 1, 0},

{Dx, Dy, Dz, 1},

}

};

return A;

}

Векторы:

**VEC VecSet( DBL X, DBL Y, DBL Z )**

**{**

**VEC v;**

**v.X = X;**

**v.Y = Y;**

**v.Z = 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 V, DBL N );**

**VEC VecDivNum( VEC V, DBL N );**

**VEC VecNeg( VEC V )**

**{**

**returb VecSet(-V.X, -V.Y, -V.Z);**

**}**

**DBL VecDotVec( VEC V1, VEC V2 )**

**{**

**return V1.X \* V2.X + ... ;**

**}**

**VEC VecCrossVec( VEC V1, VEC V2 )  
{**

**return VecSet(V1.Y \* V2.Z - V1.Z \* V2.Y,**

**... );**

**}**

**DBL VecLen2( VEC V ); - квадрат длины**

**DBL VecLen( VEC V ); - длина**

**VEC VecNormalize( VEC V );**

**!!!**

**все функции делаем INLINE:**

**в VisualStudio \_\_inline - в файле \*.H можно писать реализацию функции и она доступна только в файле включения:**

**VEC.H:**

**#ifndef ...**

**­\_\_inline VEC VecDotVec( VEC V1, VEC V2 )**

**{**

**return V1.X \* V2.X + ... ;**

**}**

**\_\_inline VEC VecLen( VEC V )**

**{**

**DBL len = VecDotVec(V, V);**

**if (len != 0 && len != 1)**

**return sqrt(len);**

**return len;**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**матрицы:**

**/\* единичная матрица \*/**

**MATR UnitMatrix =**

**{**

**{**

**{1, 0, 0, 0},**

**{0, 1, 0, 0},**

**{0, 0, 1, 0},**

**{0, 0, 0, 1},**

**}**

**};**

**MATR MatrIdenity( VOID )**

**{**

**return UnitMatrix;**

**}**

**MATR MatrTranslate( DBL Dx, DBL Dy, DBL Dz )**

**{**

**MATR m = UnitMatrix;**

**m.A[3][0] = Dx;**

**m.A[3][1] = Dy;**

**m.A[3][2] = Dz;**

**return m;**

**}**

**MATR MatrScale( DBL Sx, DBL Sy, DBL Sz );**

**MATR MatrRotateX( DBL AngleInDegree );**

**MATR MatrRotateY( DBL AngleInDegree );**

**MATR MatrRotateZ( DBL AngleInDegree );**

**MATR MatrRotate( DBL AngleInDegree,**

**DBL X, DBL Y, DBL Z )**

**{**

**DBL a, si, co, len;**

**MATR m;**

**a = D2R(AngleInDegree);**

**si = sin(a);**

**co = cos(a);**

**len = X \* X + Y \* Y + Z \* Z;**

**if (len != 0 && len != 1)**

**len = sqrt(len), X /= len, Y /= len, Z /= len;**

**X \*= si;**

**Y \*= si;**

**Z \*= si;**

**m.A[0][0] = 1 - 2 \* (Y \* Y + Z \* Z);**

**m.A[0][1] = 2 \* X \* Y - 2 \* co \* Z;**

**m.A[0][2] = 2 \* co \* Y + 2 \* X \* Z;**

**m.A[0][3] = 0;**

**m.A[1][0] = 2 \* X \* Y + 2 \* co \* Z;**

**m.A[1][1] = 1 - 2 \* (X \* X + Z \* Z);**

**m.A[1][2] = 2 \* Y \* Z - 2 \* co \* X;**

**m.A[1][3] = 0;**

**m.A[2][0] = 2 \* X \* Z - 2 \* co \* Y;**

**m.A[2][1] = 2 \* co \* X + 2 \* Y \* Z;**

**m.A[2][2] = 1 - 2 \* (X \* X + Y \* Y);**

**m.A[2][3] = 0;**

**m.A[3][0] = 0;**

**m.A[3][1] = 0;**

**m.A[3][2] = 0;**

**m.A[3][3] = 1;**

**return m;**

**}**

**MATR MatrMulMatr( MATR M1, MATR M2 )**

**{**

**MATR m;**

**for (i = 0; i < 4; i++)**

**for (j = 0; j < 4; j++)**

**for (m.A[i][j] = 0, k = 0; k < 4; k++)**

**m.A[i][j] += M1.A[i][k] \* M2.A[k][j];**

**return m;**

**}**

**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 + A12 \* A23 \* A31 + A13 \* A21 \* A32 +**

**-A11 \* A23 \* A32 - A12 \* A21 \* A33 - 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);**

**INT perm[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[perm[i][0]][perm[j][0]],**

**M.A[perm[i][0]][perm[j][1]],**

**M.A[perm[i][0]][perm[j][2]],**

**M.A[perm[i][1]][perm[j][0]],**

**M.A[perm[i][1]][perm[j][1]],**

**M.A[perm[i][1]][perm[j][2]],**

**M.A[perm[i][2]][perm[j][0]],**

**M.A[perm[i][2]][perm[j][1]],**

**M.A[perm[i][2]][perm[j][2]]) / det;**

**return r;**

**}**

**!!!**

**VEC PointTransform( VEC V, MATR M );**

**VEC VectorTransform( VEC V, MATR M );**

**!!! для нормалей:**

**MATR Q = MatrTranspose(MatrInverse(M));**

**N1 = VectorTransform(N, Q);**

**!!! ПРОВЕРКА !!!**

**05 - GDI**

**06 - Linear Algebra**

**07 - Double Buffering**

**08 - WinAPI surway**

**09 - Full Screen**

**10 - SinCos through assembler**

**11 - Vec plan**