#include <windows.h>

#include <cmath>

int xa = 10;

int ya = 10;

int xb = 100;

int yb = 10;

int xc = 10;

int yc = 100;

int tx = 50;

int ty = 100;

float angle = 90;

typedef float Matrix3x3[3][3];

Matrix3x3 theMatrix;

static HWND sHwnd;

static COLORREF redColor=RGB(255,0,0);

static COLORREF blueColor=RGB(0,0,255);

static COLORREF greenColor=RGB(0,255,0);

void SetWindowHandle(HWND hwnd){

sHwnd=hwnd;

}

void setPixel(int x,int y,COLORREF& color=redColor){

if(sHwnd==NULL){

MessageBox(NULL,"sHwnd was not initialized !","Error",MB\_OK|MB\_ICONERROR);

exit(0);

}

HDC hdc=GetDC(sHwnd);

SetPixel(hdc,x,y,color);

ReleaseDC(sHwnd,hdc);

return;

}

void matrix3x3setIdentity(Matrix3x3 m){

int i, j;

for(i = 0; i < 3; i++){

for(j = 0; j < 3; j++){

m[i][j] = (i == j);

}

}

}

void matrixMultiply(Matrix3x3 a, Matrix3x3 b){

Matrix3x3 temp;

for(int i = 0; i < 3; i++)

for(int j = 0; j < 3; j++)

temp[i][j] = a[i][0]\*b[0][j] + a[i][1]\*b[1][j] + a[i][2]\*b[2][j];

for(int i = 0; i < 3; i++)

for(int j = 0; j < 3; j++)

b[i][j] = temp[i][j];

}

void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3){

POINT Pt[3];

HDC hDC=GetDC(sHwnd);

Pt[0].x = x1; Pt[0].y = y1;

Pt[1].x = x2; Pt[1].y = y2;

Pt[2].x = x3; Pt[2].y = y3;

Polygon(hDC, Pt, 3);

}

void initializeTheMatrix(Matrix3x3 m){

m[0][0] = xa; m[1][0] = ya; m[2][0] = 1;

m[0][1] = xb; m[1][1] = yb; m[2][1] = 1;

m[0][2] = xc; m[1][2] = yc; m[2][2] = 1;

}

void assignPoints(int &xa, int &ya, int &xb, int &yb, int &xc, int &yc, Matrix3x3 theMatrix){

xa = theMatrix[0][0]; ya = theMatrix[1][0];

xb = theMatrix[0][1]; yb = theMatrix[1][1];

xc = theMatrix[0][2]; yc = theMatrix[1][2];

}

void translateTriangle(int tx, int ty){

Matrix3x3 tVector;

matrix3x3setIdentity(tVector);

tVector[0][2] = tx;

tVector[1][2] = ty;

initializeTheMatrix(theMatrix);

matrixMultiply(tVector, theMatrix);

assignPoints(xa, ya, xb, yb, xc, yc, theMatrix);

drawTriangle(xa,ya,xb,yb,xc,yc);

}

void rotateTriangle(float angle, int rx, int ry){

Matrix3x3 rVector;

matrix3x3setIdentity(rVector);

angle = angle\*M\_PI/180;

rVector[0][0] = cosf(angle);

rVector[0][1] = -sinf(angle);

rVector[0][2] = rx\*(1-cosf(angle)) + ry\*sinf(angle);

rVector[1][0] = sinf(angle);

rVector[1][1] = cosf(angle);

rVector[0][2] = ry\*(1-cosf(angle)) - rx\*sinf(angle);

initializeTheMatrix(theMatrix);

matrixMultiply(rVector, theMatrix);

assignPoints(xa, ya, xb, yb, xc, yc, theMatrix);

drawTriangle(xa, ya, xb, yb, xc, yc);

}

void scaleTriangle(float xf, float yf, float sx, float sy){

Matrix3x3 sVector;

matrix3x3setIdentity(sVector);

sVector[0][0] = sx;

sVector[0][2] = (1 - sx)\*xf;

sVector[1][1] = sy;

sVector[1][2] = (1 - sy)\*yf;

initializeTheMatrix(theMatrix);

matrixMultiply(sVector, theMatrix);

assignPoints(xa, ya, xb, yb, xc, yc, theMatrix);

drawTriangle(xa, ya, xb, yb, xc, yc);

}

LRESULT CALLBACK WndProc(HWND hwnd,UINT message,WPARAM wParam,LPARAM lParam){

switch(message){

case WM\_PAINT:

SetWindowHandle(hwnd);

drawTriangle(xa, ya, xb, yb, xc, yc);

translateTriangle(tx, ty);

rotateTriangle(angle, 30, 500);

scaleTriangle(600, 200, 0.5, 0.5);

break;

case WM\_CLOSE: // FAIL THROUGH to call DefWindowProc

break;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

default:

break; // FAIL to call DefWindowProc //

}

return DefWindowProc(hwnd,message,wParam,lParam);

}

int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR lpCmdLine,int iCmdShow){

static TCHAR szAppName[] = TEXT("Transformations");

WNDCLASS wndclass;

wndclass.style = CS\_HREDRAW|CS\_VREDRAW ;

wndclass.lpfnWndProc = WndProc ;

wndclass.cbClsExtra = 0 ;

wndclass.cbWndExtra = 0 ;

wndclass.hInstance = hInstance ;

wndclass.hIcon = LoadIcon (NULL, IDI\_APPLICATION) ;

wndclass.hCursor = LoadCursor (NULL, IDC\_ARROW) ;

wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE\_BRUSH) ;

wndclass.lpszMenuName = NULL ;

wndclass.lpszClassName = szAppName ;

// Register the window //

if(!RegisterClass(&wndclass)){

MessageBox(NULL,"Registering the class failled","Error",MB\_OK|MB\_ICONERROR);

exit(0);

}

// CreateWindow //

HWND hwnd=CreateWindow(szAppName,"Translation, Rotation and Scaling - Programming Techniques",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

NULL,

NULL,

hInstance,

NULL);

if(!hwnd){

MessageBox(NULL,"Window Creation Failed!","Error",MB\_OK);

exit(0);

}

ShowWindow(hwnd,iCmdShow);

UpdateWindow(hwnd);

MSG msg;

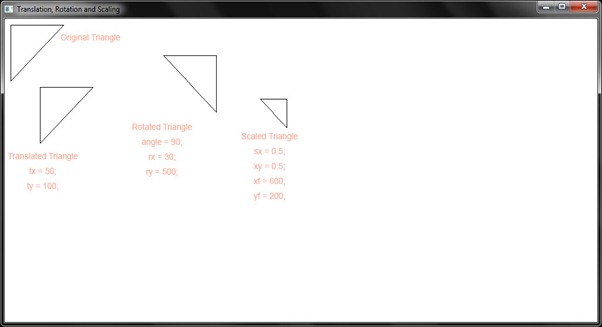
while(GetMessage(&msg,NULL,0,0)){

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

/\*outup\*/