Доработать программу. В программе создаются три трёхмерных куба. Угол обзора можно менять на стрелочки с клавиатуры. Нужно сделать так, чтобы они образовывали угол в 90 градусов.

Листинг программы:

#include "stdafx.h"

#include "kubic.h"

#include <math.h>

#define MAX\_LOADSTRING 100

HINSTANCE hInst;

TCHAR szTitle[MAX\_LOADSTRING];

TCHAR szWindowClass[MAX\_LOADSTRING];

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY \_tWinMain(\_In\_ HINSTANCE hInstance,

 \_In\_opt\_ HINSTANCE hPrevInstance,

 \_In\_ LPTSTR lpCmdLine,

 \_In\_ int nCmdShow)

{

 UNREFERENCED\_PARAMETER(hPrevInstance);

 UNREFERENCED\_PARAMETER(lpCmdLine);

 MSG msg;

 HACCEL hAccelTable;

 LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

 LoadString(hInstance, IDC\_KUBIC, szWindowClass, MAX\_LOADSTRING);

 MyRegisterClass(hInstance);

 if (!InitInstance(hInstance, nCmdShow))

 {

 return FALSE;

 }

 hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_KUBIC));

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

 {

 if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

 {

 TranslateMessage(&msg);

 DispatchMessage(&msg);

 }

 }

 return (int)msg.wParam;

}

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

 WNDCLASSEX wcex;

 wcex.cbSize = sizeof(WNDCLASSEX);

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

 wcex.lpfnWndProc = WndProc;

 wcex.cbClsExtra = 0;

 wcex.cbWndExtra = 0;

 wcex.hInstance = hInstance;

 wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDC\_KUBIC));

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

 wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

 wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_KUBIC);

 wcex.lpszClassName = szWindowClass;

 wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

 return RegisterClassEx(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

 HWND hWnd;

 hInst = hInstance;

 hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

 CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

 if (!hWnd)

 {

 return FALSE;

 }

 ShowWindow(hWnd, nCmdShow);

 UpdateWindow(hWnd);

 return TRUE;

}

void paint(HDC hdc, int startX, int startY, int endX, int endY)

{

 MoveToEx(hdc, startX, startY, NULL);

 LineTo(hdc, endX, endY);

}

float v11, v12, v13, v21, v22, v23, v32, v33, v43,

screen\_dist = 10, cX = 300, cY = 300;

int R = 2000, teta = 30, fi = 55;

int h = 5;

int kub[8][3] = {

 { h, -h, -h },{ h, h, -h },{ -h, h, -h },{ -h, -h, -h },{ h, -h, h },{ h, h, h },{ -h, h, h },{ -h, -h, h },

};

void coef(float R, float teta, float fi)

{

 float T, F, sinT, cosT, sinF, cosF;

 T = teta \* 3.14 / 180;

 F = fi \* 3.14 / 180;

 cosT = cos(T); sinT = sin(T);

 cosF = cos(F); sinF = sin(F);

 v11 = -sinT; v12 = -cosF\*cosT; v13 = -sinF\*cosT;

 v21 = cosT; v22 = -cosF\*sinT; v23 = -sinF\*sinT;

 v32 = sinF; v33 = -cosF;

 v43 = R;

}

void persp(float xw, float yw, float zw, float &X, float &Y)

{

 float xe, ye, ze;

 xe = (float)v11\*xw + v21\*yw;

 ye = (float)v12\*xw + v22\*yw + v32\*zw;

 ze = (float)v13\*xw + v23\*yw + v33\*zw + v43;

 X = (float)screen\_dist \* xe + cX;

 Y = (float)screen\_dist \* ye + cY;

}

void drawLine(HDC hdc, float startX, float startY, float startZ, float endX, float endY, float endZ)

{

 float sX, sY, eX, eY;

 persp(startX, startY, startZ, sX, sY);

 persp(endX, endY, endZ, eX, eY);

 MoveToEx(hdc, sX, sY, NULL);

 LineTo(hdc, eX, eY);

}

void drawKub(HDC hdc, int arr[][3])

{

 coef(R, teta, fi);

 drawLine(hdc, arr[0][0], arr[0][1], arr[0][2], arr[3][0], arr[3][1], arr[3][2]);

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

 {

 drawLine(hdc, arr[i][0], arr[i][1], arr[i][2], arr[i + 1][0], arr[i + 1][1], arr[i + 1][2]);

 }

 drawLine(hdc, arr[4][0], arr[4][1], arr[4][2], arr[7][0], arr[7][1], arr[7][2]);

 for (int i = 4; i < 7; i++)

 {

 drawLine(hdc, arr[i][0], arr[i][1], arr[i][2], arr[i + 1][0], arr[i + 1][1], arr[i + 1][2]);

 }

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

 {

 drawLine(hdc, arr[i][0], arr[i][1], arr[i][2], arr[i + 4][0], arr[i + 4][1], arr[i + 4][2]);

 }

 //-------------------------------------------------------------------------------

 drawLine(hdc, (10 + arr[0][0]), (10 + arr[0][1]), (10 + arr[0][2]), (10 + arr[3][0]), (10 + arr[3][1]), (10 + arr[3][2]));

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

 {

 drawLine(hdc, (10 + arr[i][0]), (10 + arr[i][1]), (10 + arr[i][2]), (10 + arr[i + 1][0]), (10 + arr[i + 1][1]), (10 + arr[i + 1][2]));

 }

 drawLine(hdc, (10 + arr[4][0]), (10 + arr[4][1]), (10 + arr[4][2]), (10 + arr[7][0]), (10 + arr[7][1]), (10 + arr[7][2]));

 for (int i = 4; i < 7; i++)

 {

 drawLine(hdc, (10 + arr[i][0]), (10 + arr[i][1]), (10 + arr[i][2]), (10 + arr[i + 1][0]), (10 + arr[i + 1][1]), (10 + arr[i + 1][2]));

 }

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

 {

 drawLine(hdc, (10 + arr[i][0]), (10 + arr[i][1]), (10 + arr[i][2]), (10 + arr[i + 4][0]), (10 + arr[i + 4][1]), (10 + arr[i + 4][2]));

 }

 //------------------------------------------------------------------

 drawLine(hdc, (-10 + arr[0][0]), (-10 + arr[0][1]), (-10 + arr[0][2]), (-10 + arr[3][0]), (-10 + arr[3][1]), (-10 + arr[3][2]));

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

 {

 drawLine(hdc, (-10 + arr[i][0]), (-10 + arr[i][1]), (-10 + arr[i][2]), (-10 + arr[i + 1][0]), (-10 + arr[i + 1][1]), (-10 + arr[i + 1][2]));

 }

 drawLine(hdc, (-10 + arr[4][0]), (-10 + arr[4][1]), (-10 + arr[4][2]), (-10 + arr[7][0]), (-10 + arr[7][1]), (-10 + arr[7][2]));

 for (int i = 4; i < 7; i++)

 {

 drawLine(hdc, (-10 + arr[i][0]), (-10 + arr[i][1]), (-10 + arr[i][2]), (-10 + arr[i + 1][0]), (-10 + arr[i + 1][1]), (-10 + arr[i + 1][2]));

 }

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

 {

 drawLine(hdc, (-10 + arr[i][0]), (-10 + arr[i][1]), (-10 + arr[i][2]), (-10 + arr[i + 4][0]), (-10 + arr[i + 4][1]), (-10 + arr[i + 4][2]));

 }

}

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

{

 int wmId, wmEvent;

 PAINTSTRUCT ps;

 HDC hdc;

 switch (message)

 {

 case WM\_COMMAND:

 wmId = LOWORD(wParam);

 wmEvent = HIWORD(wParam);

 switch (wmId)

 {

 case IDM\_ABOUT:

 DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

 break;

 case IDM\_EXIT:

 DestroyWindow(hWnd);

 break;

 default:

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

 }

 break;

 case WM\_KEYDOWN:

 {

 switch (wParam)

 {

 case VK\_LEFT:

 {

 teta += 4;

 break;

 }

 case VK\_RIGHT:

 {

 teta -= 4;

 break;

 }

 case VK\_UP:

 {

 fi += 4;

 break;

 }

 case VK\_DOWN:

 {

 fi -= 4;

 break;

 }

 case VK\_ADD:

 {

 R -= 200;

 break;

 }

 case VK\_SUBTRACT:

 {

 R += 200;

 break;

 }

 default:

 break;

 }

 InvalidateRect(hWnd, NULL, true);

 break;

 }

 case WM\_PAINT:

 hdc = BeginPaint(hWnd, &ps);

 drawKub(hdc, kub);

 EndPaint(hWnd, &ps);

 break;

 case WM\_DESTROY:

 PostQuitMessage(0);

 break;

 default:

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

 }

 return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

 UNREFERENCED\_PARAMETER(lParam);

 switch (message)

 {

 case WM\_INITDIALOG:

 return (INT\_PTR)TRUE;

 case WM\_COMMAND:

 if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

 {

 EndDialog(hDlg, LOWORD(wParam));

 return (INT\_PTR)TRUE;

 }

 break;

 }

 return (INT\_PTR)FALSE;

}