**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ**

[DEPARTAMENTUL INFORMATICĂ ŞI INGINERIA SISTEMELOR](https://utm.md/subdiviziuni-universitare/facultati/facultatea-calculatoare-informatica-si-microelectronica/catedra-calculatoare/)

**Raport**

**LUCRARE DE LABORATOR NR.2**

**La Grafica pe calculator**

**Varianta 8**

**A realizat: Nicolenco Eugeniu gr. C-162**

**A verificat: Lector sup. Plotnic Constantin**

**Chișinău 2018**

**2. TRANSFORMĂRI GEOMETRICE A IMAGINILOR**

***Scopul lucrării:*** Obţinerea cunoştinţelor practice în realizarea transformărilor geometrice 2D şi 3D a imaginilor

***Partea teoretică:*** temele: 8, 9

***Sarcina lucrării:***

1. Elaborarea programului de rotire, scalare şi deplasare a imaginilor în plan şi în spaţiu
2. Extrageţi la ecran un triunghi de culoarea R
3. Extrageţi la ecran un pătrat de culoarea S
4. Îndepliniţi transformarea tuturor punctelor de culoarea S în conformitate cu variantele (tabela 2). Punctul ce se deplasează se recolorează în culoarea R. Pentru determinarea culorii punctului folosiţi procedura *GetPixel.*

Rotirea să fie efectuată în jurul unui punct oarecare din spaţiul coordonatelor ecran

Tabela 2. Variantele de îndeplinire a lucrării

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | Rotirea cu  unghiul | Coeficientul de scalare | Deplasarea  X şi Y |
| **1** | 25 | 0.5 | 10, 20 |
| **2** | 60 | 1.2 | 10, 30 |
| **3** | 45 | 1.6 | 20, 40 |
| **4** | 100 | 2.2 | 30, 40 |
| **5** | 120 | 0.8 | 100,100 |
| **6** | 180 | 2.0 | 10, 20 |
| **7** | 35 | 1.45 | 15, 25 |
| **8** | 200 | 1.8 | 25, 30 |
| **9** | 300 | 1.3 | 200, 20 |
| **10** | 310 | 2.1 | 100, 30 |
| **11** | 260 | 2.7 | 10,100 |
| **12** | 100 | 1.85 | 100, 10 |
| **13** | 245 | 1.25 | 240, 32 |
| **14** | 80 | 1.4 | 25, 45 |
| **15** | 130 | 0.6 | 28, 41 |
| **16** | 240 | 1.7 | 5, 20 |
| **17** | 320 | 0.8 | 32, 15 |
| **18** | 280 | 1.9 | 80, 64 |
| **19** | 70 | 0.4 | 30, 40 |
| **20** | 150 | 1.65 | 230, 50 |

**Listingul programului :**

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

#include <time.h>

#include <conio.h>

#include <graphics.h>

#include <iostream>

using namespace std;

void scale(int \*point, float scaleFactor) {

point[0] = floor(point[0]\*scaleFactor);

point[1] = floor(point[1]\*scaleFactor);

}

void rotatet (int \*point,float rotationAngle) {

float deg2rad=rotationAngle\*M\_PI/180;

point[0]=floor(point[0]\*cos(deg2rad)+point[0]\*sin(deg2rad));

point[1]= floor(point[1]\*cos(deg2rad)-point[1]\*sin(deg2rad));

}

void translate(int \*point,int translateX, int translateY){

point[0] = point[0] + translateX;

point[1] = point[1] + translateY;

}

void swapPoints(int \*pointList,int pos1, int pos2){

int temp1,temp2;

temp1=pointList[2\*pos1];

temp2=pointList[2\*pos1+1];

pointList[2\*pos1]=pointList[2\*pos2];

pointList[2\*pos1+1]=pointList[2\*pos2+1];

pointList[2\*pos2]=temp1;

pointList[2\*pos2+1]=temp2;

}

int main() {

int gd=DETECT, gm, pivotX=100,

pivotY=200,colorR,colorS,tempX,tempY,tempPoint[2];

int

resultRotation[10],rectangleX[]={36,56},rectangleY[]={240,270},errorcode,counter ;

int resultScaling[10],resultTranslation[10];

int triangle[]={10,20,30,180,190,40,10,20};

time\_t t;

initgraph(&gd, &gm, " ");

floodfill(1,1,WHITE);

/\* read result of initialization \*/

errorcode = graphresult();

if (errorcode != grOk) /\* an error occurred \*/{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1); /\* terminate with an error code \*/

}

srand ((unsigned) time(&t));

colorR =(rand()%15)+1;

do {colorS =(rand()%15)+1;}

while(colorR == colorS);

setcolor(colorR);

drawpoly(4,triangle);

setcolor(colorS);

rectangle(rectangleX[0],rectangleY[0],rectangleX[1],rectangleY[1]);

setcolor(RED);

line(pivotX-5,pivotY-5,pivotX+5,pivotY+5);

line(pivotX+5,pivotY-5,pivotX-5,pivotY+5);

//getch();

counter=0;

for (tempX=0;tempX<2;tempX++){

for(tempY=0;tempY<2;tempY++){

tempPoint[0]=rectangleX[tempX];

tempPoint[1]=rectangleY[tempY];

//

translate(tempPoint,-pivotX,-pivotY);

rotatet(tempPoint,200);

translate(tempPoint,pivotX,pivotY);

resultRotation[counter]=tempPoint[0];

resultRotation[counter+1]=tempPoint[1];

//

scale(tempPoint,1.8);

resultScaling[counter]=tempPoint[0];

resultScaling[counter+1]=tempPoint[1];

//

translate(tempPoint,25,30);

resultTranslation[counter]=tempPoint[0];

resultTranslation[counter+1]=tempPoint[1];

counter+=2; } }

resultRotation[8]=resultRotation[0];

resultRotation[9]=resultRotation[1];

resultScaling[8]=resultScaling[0];

resultScaling[9]=resultScaling[1];

resultTranslation[8]=resultTranslation[0];

resultTranslation[9]=resultTranslation[1];

swapPoints(resultRotation,1,2);

swapPoints(resultScaling,1,2);

swapPoints(resultTranslation,1,2);

swapPoints(resultRotation,2,3);

swapPoints(resultScaling,2,3);

swapPoints(resultTranslation,2,3);

setcolor(colorR);

drawpoly(5,resultRotation);

outtextxy(resultRotation[0],resultRotation[1]+20,"Rotatia");

drawpoly(5,resultScaling);

outtextxy(resultScaling[0],resultScaling[1]+20,"Rotatia + Scalarea");

drawpoly(5,resultTranslation);

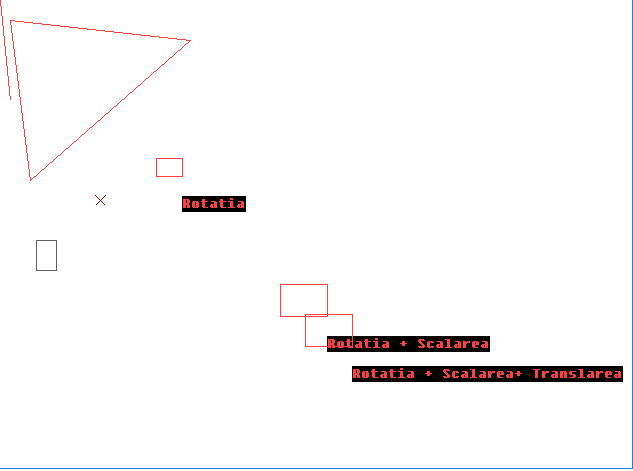
outtextxy(resultTranslation[0],resultTranslation[1]+20,"Rotatia + Scalarea+ Translarea");

getch();

closegraph();

restorecrtmode();

return 0; }



1. Rotirea unui obiect în jurul unei axea) Desenarea unui cub specificat prin coordonatele vârfurilor sale

b) Rotirea cubului în jurul axelor orizontală şi verticală care trec prin centrul sau, precum şi în jurul axei OZ

**Listingul programului:**

#include <conio.h>

#include <stdio.h>

#include <graphics.h>

#include <math.h>

const int size = 8;

const int pix[size][3] ={{-25,-25,-25},{25,-25,-25},{25,25,-25},{-25,25,-25},

{-25,-25,25}, {25,-25,25}, {25,25,25}, {-25,25,25}};

int newpix[size][3];

int X,Y,Z,sX,sY;

int RotX = 0,RotY = 0,RotZ = 0;

int ShiftX = 325,ShiftY = 225;

long double Scale = 2.5;

void Perspect()

{

sX = X + Z /2;

sY = Y - Z /2;

}

void Compute()

{

double sinx = sin((RotX \*M\_PI) /180.0);

double siny = sin((RotY \*M\_PI) /180.0);

double sinz = sin((RotZ \*M\_PI) /180.0);

double cosx = cos((RotX \*M\_PI) /180.0);

double cosy = cos((RotY \*M\_PI) /180.0);

double cosz = cos((RotZ \*M\_PI) /180.0);

double x,y,z,x1,y1,z1;

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

{

x = pix[i][0];

y = pix[i][1];

z = pix[i][2];

x1 = x\*cosz + y\*sinz;

y1 = -x\*sinz + y\*cosz;

z1 = z;

x = x1;

y = y1\*cosx + z1\*sinx;

z = -y1\*sinx + z1\*cosx;

x1 = x\*cosy - z\*siny;

y1 = y;

z1 = x\*siny + z\*cosy;

newpix[i][0] = x1\*Scale;

newpix[i][1] = y1\*Scale;

newpix[i][2] = z1\*Scale;

newpix[i][0] += ShiftX;

newpix[i][1] += ShiftY;

newpix[i][2] += 25;

}

}

void DrawPix()

{

int i,j;

int x,y;

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

{

X = newpix[i][0];

Y = newpix[i][1];

Z = newpix[i][2];

Perspect();

x = sX;

y = sY;

j = (i < 3) ? (i+1) : 0;

X = newpix[j][0];

Y = newpix[j][1];

Z = newpix[j][2];

Perspect();

line(x,y,sX,sY);

}

for (i = 4;i < 8;i++)

{

X = newpix[i][0];

Y = newpix[i][1];

Z = newpix[i][2];

Perspect();

x = sX;

y = sY;

j = (i < 7) ? (i+1) : 4;

X = newpix[j][0];

Y = newpix[j][1];

Z = newpix[j][2];

Perspect();

line(x,y,sX,sY);

}

for (i = 0;i < 4;i++) // conectarile a 2 patrate

{

X = newpix[i][0];

Y = newpix[i][1];

Z = newpix[i][2];

Perspect();

x = sX;

y = sY;

j = i+4;

X = newpix[j][0];

Y = newpix[j][1];

Z = newpix[j][2];

Perspect();

line(x,y,sX,sY);

}

}

int main(){

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

return 0;

}

setbkcolor(0);

setcolor(15);

setwritemode(XOR\_PUT); // regim de afisare cu stergere

Compute();

DrawPix();

int ch = 0;

int flg = 0;

do

{

switch (ch)

{

case 75:

ShiftX -= 3;

flg = 1;

break;

case 77:

ShiftX += 3;

flg = 1;

break;

case 72:

ShiftY -= 3;

flg = 1;

break;

case 80:

ShiftY += 3;

flg = 1;

break;

case '+':

Scale += 0.1;

flg = 1;

break;

case '-':

if (Scale > 0.0)

Scale -= 0.1;

flg = 1;

break;

case 'x': case 'X':

if (RotX < 357)

RotX += 3;

else

RotX = 0;

flg = 1;

break;

case 'y': case 'Y':

if (RotY < 357)

RotY += 3;

else

RotY = 0;

flg = 1;

break;

case 'z': case 'Z':

if (RotZ < 357)

RotZ += 3;

else

RotZ = 0;

flg = 1;

break;

}

if (flg == 1)

{

DrawPix();

Compute();

DrawPix();

flg = 0;

}

ch = getch();

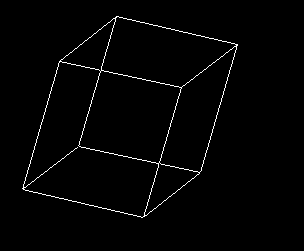
} while (ch != 1);

closegraph();

return 0;

}

**Rezultatul programului:**



Butoane de comanda:

Left Arrow – translarea spre stinga

Right Arrow – translarea spre dreapta

Up Arrow – translarea in sus

Down Arrow – translarea in jos

„+” – Zoom in

„-„ – Zoom out

z\Z – Rotirea in jurul axei z

y\Y – Rotirea in jurul axei y

x\X – Rotirea in jurul axei x

(butoanele pot fi un pic incurcate, dar functionalul este corect)