Дисципліна "Комп'ютерний синтез та обробка зображень"

тема "Синтез зображення монохромного крапкового джерела світла "

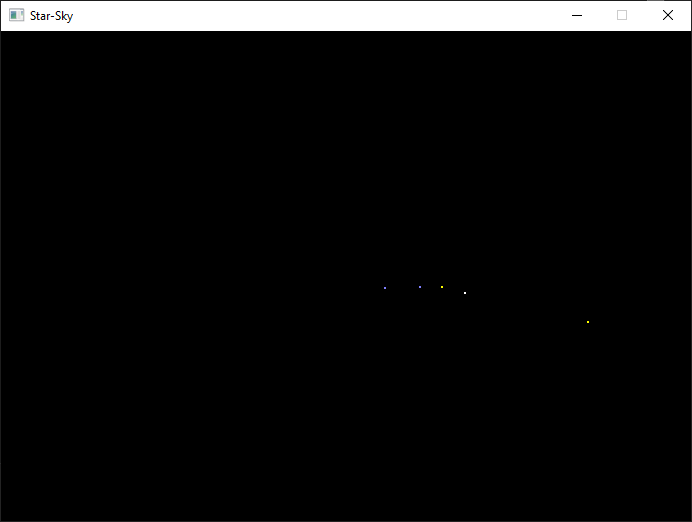
**Мета роботи:** вивчення алгоритмів синтезу зображення статичної сцени класу "зоряне небо" – сукупності крапкових монохромних джерел світла.

**Постановка задачі лабораторної роботи**

Розробити програму, що виконує синтез сцени, яка містить крапкові джерела світла. Параметри сцени наведені у таблиці:

|  |  |  |  |
| --- | --- | --- | --- |
| *Nl* | *d* | *2Ex* | *2Ey* |
| 5 | 0.5 | 1 | 2 |

**Інтерфейс розробленої програми**



**Вихідний код розробленої програми**

#include "graphics.h"

#include <math.h>

#define PI 3.141592653589793238462643383279502884197169399375105820974

float l[5][3] = { { 20,10,10 },{ 70,60,10 },{ 10,100,40 },{ 50,90,70 },{ 120,95,100 } }; //lights

float lskn[5][3]; // lights in skn

int skn[3] = { 10,20,30 }; // X0Y0Z0

float d = 0.5;

float ex2 = 1;

float ey2 = 2; // k = 1

double ptg[3] = { -PI / 6, -PI / 3, PI / 9 }; // rad = gr.\* pi / 180

int col = 9;

int cols[3]; // 14 - very close, 15 - further, 9 - the furthest

int dispX = 700;

int dispY = 500;

char c;

int main()

{

initwindow(dispX, dispY, "lab4");

do { // matrix a = Rx\*Rz\*Ry -> matrix of rotation round Oy, Oz and Ox

float a[3][3] =

{ { cos(ptg[0])\*cos(ptg[1]), -sin(ptg[1]), sin(ptg[0])\*cos(ptg[1]) },

{ cos(ptg[0])\*sin(ptg[1])\*cos(ptg[2]) + sin(ptg[0])\*sin(ptg[2]), cos(ptg[1])\*cos(ptg[2]), sin(ptg[0])\*sin(ptg[1])\*cos(ptg[2]) - cos(ptg[0])\*sin(ptg[2]) },

{ cos(ptg[0])\*sin(ptg[1])\*sin(ptg[2]) - sin(ptg[0])\*cos(ptg[2]), cos(ptg[1])\*sin(ptg[2]), sin(ptg[0])\*sin(ptg[1])\*sin(ptg[2]) + cos(ptg[0])\*cos(ptg[2]) } };

// lights in MSK to SKN

for (int i = 0; i < 5; i++)

{

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

{

lskn[i][j] = l[i][j] + skn[j];

}

}

// povorot y, z, x: new lights[3] = float a[3][3] \* lights[3]

for (int i = 0; i < 5; i++)

{

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

{

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

}

}

// otsechenie

bool pr[5] = { 0,0,0,0,0 };

for (int i = 0; i < 5; i++)

{

if ((abs(lskn[i][1]) <= lskn[i][0]) && (abs(lskn[i][2]) <= lskn[i][0]) && (lskn[i][0] >= 0.5) && (lskn[i][0] <= 1000))

pr[i] = true;

else pr[i] = false;

}

// proecirovanie

float p[3][2];

for (int i = 0; i < 5; i++)

{

if (pr[i] = 1)

{

p[i][0] = lskn[i][2] / (lskn[i][0] / d + 1);

p[i][1] = lskn[i][1] / (lskn[i][0] / d + 1);

if (lskn[i][0]<40) cols[i] = 14;

else if (lskn[i][0]<80) cols[i] = 15;

else cols[i] = 9;

}

}

// output in DisplayFormat

for (int i = 0; i < 5; i++)

{

if (pr[i] = 1)

{

putpixel(dispX / 2 + p[i][0] \* 100, dispY / 2 - p[i][1] \* 100, cols[i]);

}

}

// video -> next step

delay(100);

cleardevice();

ptg[1] = ptg[1] + PI / 100;

if (ptg[1] >= PI / 3) ptg[1] = -PI / 3;

} while (1);

closegraph();

return 0;

}