PROIECT

SCENA 3D

GRAFICA PE CALCULATOR

Conceptul proiectului

Proiectul consta intr-o scena 3D reprezentand o sala de clasa, folosind tehnici de iluminare, texturare si amestecare.

Elemente incluse

- Obiectele 3D
  - Peretii, tavanul si podeaua salii de clasa
  - Tabla
  - Fereastra
  - Bancile
  - Cartile

Obiectele 3D sunt reprezentate folosind translatii, scalari si rotatii.

- Lumina – lumina provine dintr-o singura sursa punctuala aflata in stanga scenei pentru a simula lumina naturala care intra in clasa pe fereastra.
- Texturare – am folosit texturarea pentru a adauga o textura de parchet podelei
- Amestecare – pentru realizarea ferestrei si a paharului de pe catedra am folosit tehnica de amestecare pentru a crea transparenta specifica acestor obiecte

Obiectele sunt desenate folosind indexare. Vectorul Vertices contine coordonatele varfurilor, culoarea acestora, normalele si coordonatele de texturare.

```
// coordonate                    // culori                // normale            //texturare

// varfuri masa
-500.0f,  -500.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, -1.0f, -1.0f,    0.0f, 0.0f,
 500.0f,  -500.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -1.0f, -1.0f,     0.0f, 0.0f,
 500.0f, 500.0f,  -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,   1.0f, 1.0f, -1.0f,     0.0f, 0.0f,
-500.0f,  500.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, 1.0f, -1.0f,     0.0f, 0.0f,
-500.0f,  -500.0f, -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, -1.0f, 1.0f,    0.0f, 0.0f,
 500.0f,  -500.0f, -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -1.0f, 1.0f,     0.0f, 0.0f,
 500.0f, 500.0f,  -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,   1.0f, 1.0f, 1.0f,     0.0f, 0.0f,
-500.0f,  500.0f, -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, 1.0f, 1.0f,     0.0f, 0.0f,
```

Lumina este simulate folosind modelul Phong, cu ajutoul formulelor prezentate in cadrul laboratorului.

```
{
  // Ambient
  float ambientStrength = 0.2f;
  vec3 ambient = ambientStrength * lightColor;

  // Diffuse
  vec3 normala = normalize(Normal);
   vec3 lightDir = normalize(inLightPos - FragPos);
  // vec3 dir=vec3(0.0,-150.0,200.0); // sursa directionala
  // vec3 lightDir=normalize(dir);
  float diff = max(dot(normala, lightDir), 0.0);
  vec3 diffuse = diff * lightColor;

  // Specular
  float specularStrength = 0.2f;
  vec3 viewDir = normalize(inViewPos - FragPos);//vector catre observator normalizat (V)
  vec3 reflectDir = reflect(-lightDir, normala); // reflexia razei de lumina (R)
  float spec = pow(max(dot(viewDir, reflectDir), 0.0), 1);
  vec3 specular = specularStrength * spec * lightColor;
  vec3 emission=vec3(0.0, 0.0, 0.0);
  vec3 result = emission+(ambient + diffuse + specular) * ex_Color;
  //out_Color = vec4(result, 1.0f);
```

Texturarea este realizata cu ajutorul coordonatelor specificate in vectorul Vertices.

```
ex_Color=in_Color;
tex_Coord = vec2(texCoord.x, 1-texCoord.y);
```

```
if ( codCol==3)
out_Color = mix(texture(myTexture, tex_Coord), out_Color, 0.2);
```

Amestecarea este realizata folosind a patra coordonata a culorii, alpha si functiile specifice amestecarii.

```
  glEnable(GL_BLEND);
  glDepthMask(GL_FALSE);
  glBlendFunc(GL_SRC_ALPHA, GL_SRC_ALPHA);
  codCol = 0;
  glUniform1i(codColLocation, codCol);
  glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(238));
  glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(388));
  glDepthMask(GL_TRUE);
  glDisable(GL_BLEND);
```

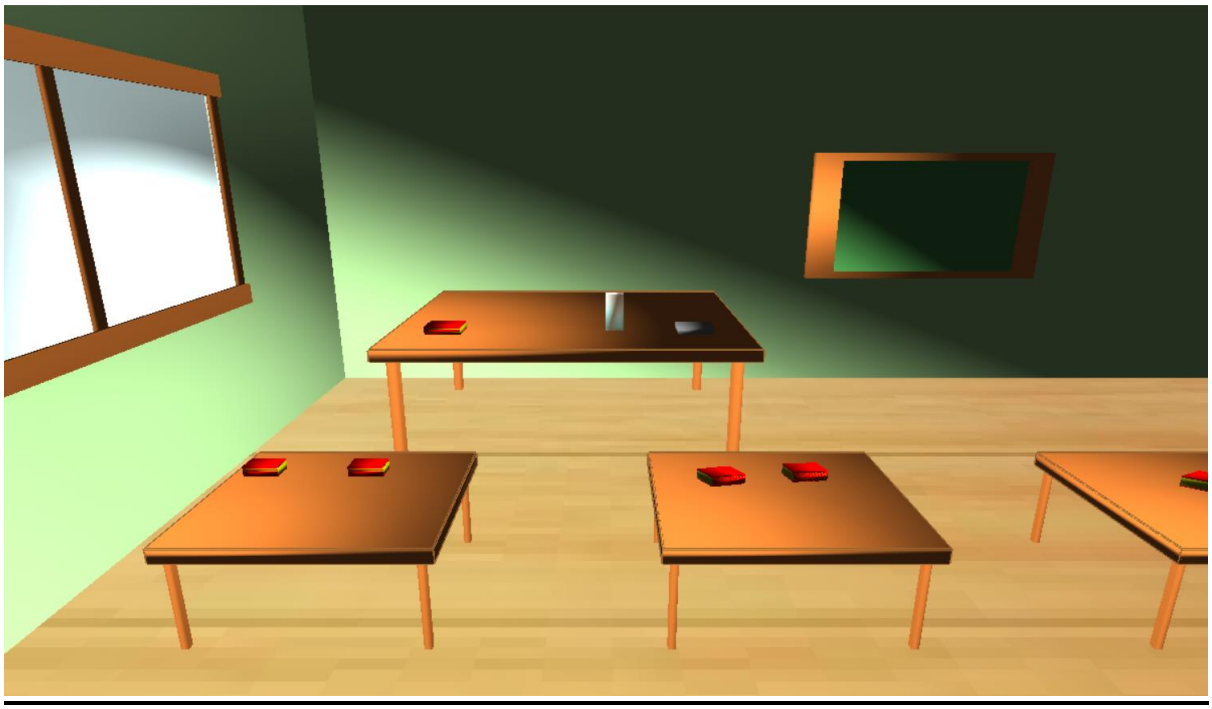
Folosind tastatura, privitorul poate survola scena.

```
void processNormalKeys(unsigned char key, int x, int y)
{

    switch (key) {
    case 'l':
        Vx -= 0.1;
        break;
    case 'r':
        Vx += 0.1;
        break;
    case '+':
        dist += 5;
        break;
    case '-':
        dist -= 5;
        break;

    }
    if (key == 27)
        exit(0);
}
```

Rezultat final



Resurse utilizate

Am pornit de la codurile din laboratoare, folosind exemplele de transformari, de iluminare, texturare si amestecare si de utilizare a tastaturii.

Cod

```
#include <windows.h>  // biblioteci care urmeaza sa fie incluse
#include <stdlib.h> // necesare pentru citirea shader-elor
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <GL/glew.h> // glew apare inainte de freeglut
#include <GL/freeglut.h> // nu trebuie uitat freeglut.h

#include "loadShaders.h"

#include "glm/glm/glm.hpp"
#include "glm/glm/gtc/matrix_transform.hpp"
#include "glm/glm/gtx/transform.hpp"
#include "glm/glm/gtc/type_ptr.hpp"
#include "SOIL.h"


using namespace std;

////////////////////////////////////
// identificatori
GLuint
VaoId,
VboId,
EboId,
```

```
ColorBufferId,
ProgramId,
myMatrixLocation,
matrUmbraLocation,
viewLocation,
projLocation,
matrRotlLocation,
codColLocation,
depthLocation;

GLuint texture;

int codCol;
float PI = 3.141592;

// matrice utilizate
glm::mat4 myMatrix, matrRot;

// elemente pentru matricea de vizualizare
float Refx = 1500.0f, Refy = 100.0f, Refz = 0.0f;
float alpha = PI / 8, beta = 0.0f, dist = 500.0f;
float Obsx, Obsy, Obsz;

float Vx = 0.0, Vy = 0.0, Vz = 1.0;
glm::mat4 view;

// elemente pentru matricea de proiectie
float width = 800, height = 600, xwmin = -800.f, xwmax = 800, ywmin = -600,
ywmax = 600, znear = 0.1, zfar = 10, fov = 45;
glm::mat4 projection;

// sursa de lumina
float xL = 500.f, yL = -2000.f, zL = 700.f;
```

```
// matricea umbrei
float matrUmbra[4][4];


void displayMatrix()
{
    for (int ii = 0; ii < 4; ii++)
    {
        for (int jj = 0; jj < 4; jj++)
            cout << myMatrix[ii][jj] << "  ";
        cout << endl;
    };

};

void processNormalKeys(unsigned char key, int x, int y)
{

    switch (key) {
    case 'l':
        Vx -= 0.1;
        break;
    case 'r':
        Vx += 0.1;
        break;
    case '+':
        dist += 5;
        break;
    case '-':
        dist -= 5;
        break;
```

```
    }
    if (key == 27)
        exit(0);
}
void processSpecialKeys(int key, int xx, int yy) {

    switch (key) {
    case GLUT_KEY_LEFT:
        beta -= 0.01;
        break;
    case GLUT_KEY_RIGHT:
        beta += 0.01;
        break;
    case GLUT_KEY_UP:
        alpha += 0.01;
        break;
    case GLUT_KEY_DOWN:
        alpha -= 0.01;
        break;
    }
}


void LoadTexture()
{
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
    glGenTextures(1, &texture);
    glBindTexture(GL_TEXTURE_2D, texture);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

```
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

    int width, height;

    unsigned char* image = SOIL_load_image("parquet2.jpg", &width, &height,
0, SOIL_LOAD_RGB);

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
GL_UNSIGNED_BYTE, image);

    glGenerateMipmap(GL_TEXTURE_2D);



    SOIL_free_image_data(image);

    glBindTexture(GL_TEXTURE_2D, 0);

}



void CreateVBO(void)

{

    // varfurile

    GLfloat Vertices[] = {



        // coordonate                    // culori
// normale             //texturare



        // varfuri masa

       -500.0f,  -500.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f,
-1.0f, -1.0f,    0.0f, 0.0f,

        500.0f,  -500.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -
1.0f, -1.0f,     0.0f, 0.0f,

        500.0f, 500.0f,  -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,   1.0f,
1.0f, -1.0f,      0.0f, 0.0f,

       -500.0f,  500.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f,
1.0f, -1.0f,      0.0f, 0.0f,

       -500.0f,  -500.0f, -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f,
-1.0f, 1.0f,     0.0f, 0.0f,

        500.0f,  -500.0f, -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -
1.0f, 1.0f,      0.0f, 0.0f,

        500.0f, 500.0f,  -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,   1.0f,
1.0f, 1.0f,       0.0f, 0.0f,

       -500.0f,  500.0f, -120.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f,
1.0f, 1.0f,       0.0f, 0.0f,
```

```
       // varfuri tabla

       -800.0f, 1800.0f, -100.0f, 1.0f, 0.9f, 0.5f, 0.2f, 1.0f,  -1.0f,
1.0f, -1.0f, 0.0f, 0.0f,

       -800.0f, 900.0f, -100.0f, 1.0f, 0.9f, 0.5f, 0.2f, 1.0f,  -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

       -750.0f, 900.0f, -100.f, 1.0f, 0.9f, 0.5f, 0.2f, 1.0f,  1.0f, -1.0f,
-1.0f, 0.0f, 0.0f,

       -750.0f, 1800.0f, -100.f, 1.0f, 0.9f, 0.5f, 0.2f,  1.0f, 1.0f, 1.0f,
-1.0f,  0.0f, 0.0f,

       -800.0f, 1800.0f, 400.0f, 1.0f, 0.9f, 0.5f, 0.2f, 1.0f,  -1.0f,
1.0f, 1.0f,  0.0f, 0.0f,

       -800.0f, 900.0f, 400.0f, 1.0f, 0.9f, 0.5f, 0.2f, 1.0f,  -1.0f, -
1.0f, 1.0f,  0.0f, 0.0f,

       -750.0f, 900.0f, 400.f, 1.0f, 0.9f, 0.5f, 0.2f,  1.0f, 1.0f, -1.0f,
1.0f,  0.0f, 0.0f,

       -750.0f, 1800.0f, 400.f, 1.0f, 0.9f, 0.5f, 0.2f,  1.0f, 1.0f, 1.0f,
1.0f,  0.0f, 0.0f,




       -740.0f, 1700.0f, -70.0f, 1.0f, 0.3f, 0.6f, 0.3f,  1.0f, 1.0f, 1.0f,
1.0f,  0.0f, 0.0f,

       -740.0f, 1000.0f, -70.0f, 1.0f, 0.3f, 0.6f, 0.3f, 1.0f,  1.0f, -
1.0f, 1.0f,  0.0f, 0.0f,

       -740.0f, 1000.0f, 370.f, 1.0f, 0.3f, 0.6f, 0.3f,  1.0f,  1.0f, 1.0f,
1.0f,  0.0f, 0.0f,

       -740.0f, 1700.0f, 370.f, 1.0f, 0.3f, 0.6f, 0.3f,  1.0f, 1.0f, 1.0f,
1.0f,  0.0f, 0.0f,




      //picioare masa

      430.0f,  430.0f, -500.0f, 1.0f,  0.9f, 0.5f, 0.2f,    1.0f,  -1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,

        460.0f,  430.0f, -500.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,

        460.0f, 460.0f,  -500.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,

       430.0f,  460.0f, -500.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,

       430.0f,  430.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,
```

```
        460.0f,  430.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  -1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,

        460.0f, 460.0f,  -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -
1.0f, 0.0f, 0.0f, 0.0f,

       430.0f,  460.0f, -150.0f, 1.0f,  0.9f, 0.5f, 0.2f,  1.0f,  1.0f, -
1.0f, 0.0f,  0.0f, 0.0f,




        //varfuri ground

        -810.0f,  -1000.0f, -550.0f, 1.0f,  0.5f, 0.8f, 0.5f,  1.0f,  -
1.0f, -1.0f, 1.0f, 0.0f,  1.0f,

        6000.0f,  -1000.0f, -550.0f, 1.0f, 0.5f, 0.8f, 0.5f,  1.0f,  1.0f,
-1.0f, 1.0f, 0.0f,  0.0f,

        6000.0f, 3000.0f,  -550.0f, 1.0f,  0.5f, 0.8f, 0.5f,  1.0f,  1.0f,
1.0f, 1.0f, 1.0f, 0.0f,

       -810.0f,  3000.0f, -550.0f, 1.0f,  0.5f, 0.8f, 0.5f,  1.0f,   1.0f,
1.0f, 1.0f, 1.0f,  1.0f,



       //varfuri perete stanga

        6000.0f, -1000.0f,  1000.0f, 1.0f,   0.7f, 0.9f, 0.6f,  1.0f,
1.0f, 1.0f, 1.0f,  0.0f, 0.0f,

        6000.0f, -1000.0f,  -550.0f, 1.0f,   0.7f, 0.9f, 0.6f,  1.0f,
1.0f, 1.0f, 1.0f,  0.0f, 0.0f,

        -810.0f,  -1000.0f, -550.0f, 1.0f,  0.7f, 0.9f, 0.6f,  1.0f,  -
1.0f, -1.0f, 1.0f,  0.0f, 0.0f,

        -810.0f,  -1000.0f, 1000.0f, 1.0f,  0.7f, 0.9f, 0.6f,  1.0f,  -
1.0f, 1.0f, 1.0f,  0.0f, 0.0f,



        //varfuri perete dreapta

        -810.0f,  3000.0f, 1000.0f, 1.0f,  0.7f, 0.9f, 0.6f,  1.0f,  1.0f,
1.0f, 1.0f,  0.0f, 0.0f,

        -810.0f,  3000.0f, -550.0f, 1.0f,   0.7f, 0.9f, 0.6f,  1.0f,  1.0f,
1.0f, 1.0f,  0.0f, 0.0f,

         6000.0f, 3000.0f,  -550.0f, 1.0f,   0.7f, 0.9f, 0.6f,  1.0f,
1.0f, 1.0f, 1.0f,  0.0f, 0.0f,

        6000.0f, 3000.0f,  1000.0f, 1.0f,   0.7f, 0.9f, 0.6f,  1.0f,  1.0f,
1.0f, 1.0f,  0.0f, 0.0f,



        //perete spate
```

```
        -810.0f,  -1000.0f, 1000.0f, 1.0f, 0.7f, 0.9f, 0.6f,  1.0f,  1.0f,
1.0f, 1.0f,  0.0f, 0.0f,

        -810.0f,  -1000.0f, -550.0f, 1.0f,   0.7f, 0.9f, 0.6f, 1.0f,  1.0f,
-1.0f, 1.0f,  0.0f, 0.0f,

        -810.0f,  3000.0f, -550.0f, 1.0f,  0.7f, 0.9f, 0.6f,  1.0f, 1.0f,
1.0f, 1.0f,  0.0f, 0.0f,

        -810.0f,  3000.0f, 1000.0f, 1.0f,  0.7f, 0.9f, 0.6f, 1.0f,   1.0f,
1.0f, 1.0f,  0.0f, 0.0f,



        //varfuri tavan

        -810.0f,  -1000.0f, 1000.0f, 1.0f,  0.5f, 0.8f, 0.5f,  1.0f,  -
1.0f, 1.0f, -1.0f, 0.0f,  1.0f,

        6000.0f,  -1000.0f, 1000.0f, 1.0f, 0.5f, 0.8f, 0.5f,  1.0f,  1.0f,
1.0f, -1.0f, 0.0f,  0.0f,

        6000.0f, 3000.0f,  1000.0f, 1.0f,  0.5f, 0.8f, 0.5f,  1.0f, 1.0f,
1.0f, -1.0f, 1.0f, 0.0f,

       -810.0f,  3000.0f, 1000.0f, 1.0f,  0.5f, 0.8f, 0.5f,  1.0f,  1.0f,
1.0f, -1.0f, 1.0f,  1.0f,




       // varfuri usa

       500.0f,  2980.0f, -550.0f, 1.0f,  0.6f, 0.3f, 0.1f,   1.0f, -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        -200.0f,  2980.0f, -550.0f, 1.0f, 0.6f, 0.3f, 0.1f,  1.0f,  1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        -200.0f, 2995.0f,  -550.0f, 1.0f,  0.6f, 0.3f, 0.1f,  1.0f,   1.0f,
1.0f, -1.0f, 0.0f, 0.0f,

       500.0f,  2995.0f, -550.0f, 1.0f,  0.6f, 0.3f, 0.1f,  1.0f,  -1.0f,
1.0f, -1.0f, 0.0f, 0.0f,

       500.0f,  2980.0f, 500.0f, 1.0f,  0.6f, 0.3f, 0.1f,  1.0f,  -1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        -200.0f,  2980.0f, 500.0f, 1.0f, 0.6f, 0.3f, 0.1f,  1.0f,  1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        -200.0f, 2995.0f,  500.0f, 1.0f,  0.6f, 0.3f, 0.1f,  1.0f,  1.0f,
1.0f, 1.0f, 0.0f, 0.0f,

       500.0f,  2995.0f, 500.0f, 1.0f,  0.6f, 0.3f, 0.1f,   1.0f, -1.0f,
1.0f, 1.0f, 0.0f, 0.0f,



        // varfuri suport glob
```

```
        200.0f, 300.0f, -118.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f,  -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        300.0f, 300.0f, -118.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f,  1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        300.0f, 400.0f, -118.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f,  1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, 400.0f, -118.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f,  -1.0f,
1.0f, -1.0f, 0.0f, 0.0f,

        200.0f, 300.0f, -100.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f,  -1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, 300.0f, -100.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f,  1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, 400.0f, -100.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        200.0f, 400.0f, -100.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f, -1.0f, 1.0f,
1.0f, 0.0f, 0.0f,



        //geam

        1500.0f, -995.0f, 600.0f, 1.0f, 0.7f, 0.9f, 0.9f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        1500.0f, -995.0f, -70.0f, 1.0f, 0.7f, 0.9f, 0.9f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        100.0f, -995.0f, -70.0f, 1.0f, 0.7f, 0.9f, 0.9f, 1.0f, 1.0f, -1.0f,
1.0f, 0.0f, 0.0f,

        100.0f, -995.0f, 600.0f, 1.0f, 0.7f, 0.9f, 0.9f, 1.0f, -1.0f, 1.0f,
1.0f, 0.0f, 0.0f,



        1500.0f, -997.0f, 600.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        1500.0f, -997.0f, -70.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        100.0f, -997.0f, -70.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, 1.0f, -1.0f,
1.0f, 0.0f, 0.0f,

        100.0f, -997.0f, 600.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, -1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        1500.0f, -990.0f, 600.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        1500.0f, -990.0f, -70.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        100.0f, -990.0f, -70.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, 1.0f, -1.0f,
1.0f, 0.0f, 0.0f,
```

```
        100.0f, -990.0f, 600.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.2f, -1.0f, 1.0f,
1.0f, 0.0f, 0.0f,



        //carte

        200.0f, -300.0f, -118.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        300.0f, -300.0f, -118.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, 1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        300.0f, -400.0f, -118.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, 1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, -400.0f, -118.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, -1.0f,
1.0f, -1.0f, 0.0f, 0.0f,

        200.0f, -300.0f, -100.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, -1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, -300.0f, -100.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, 1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, -400.0f, -100.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        200.0f, -400.0f, -100.0f, 1.0f, 0.7f, 0.7f, 0.0f, 1.0f, -1.0f,
1.0f, 1.0f, 0.0f, 0.0f,



        200.0f, -300.0f, -122.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        300.0f, -300.0f, -122.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        300.0f, -400.0f, -122.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, -400.0f, -122.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f,
1.0f, -1.0f, 0.0f, 0.0f,

        200.0f, -300.0f, -119.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, -300.0f, -119.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, -400.0f, -119.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        200.0f, -400.0f, -119.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f,
1.0f, 1.0f, 0.0f, 0.0f,



        200.0f, -300.0f, -99.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,
```

```
        300.0f, -300.0f, -99.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, -1.0f,
-1.0f, 0.0f, 0.0f,

        300.0f, -400.0f, -99.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, -400.0f, -99.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, -300.0f, -96.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, -
1.0f, 1.0f, 0.0f, 0.0f,

        300.0f, -300.0f, -96.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, -1.0f,
1.0f, 0.0f, 0.0f,

        300.0f, -400.0f, -96.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        200.0f, -400.0f, -96.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, 1.0f,
1.0f, 0.0f, 0.0f,




        302.0f, -300.0f, -98.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, -1.0f,
1.0f, 0.0f, 0.0f,

        302.0f, -400.0f, -98.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,

        302.0f, -400.0f, -122.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        302.0f, -300.0f, -122.0f, 1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,




        //pahar


        200.0f, 100.0f, -118.0f, 1.0f, 0.0f, 0.8f, 1.0f, 0.7f, -1.0f, -
1.0f, -1.0f, 0.0f, 0.0f,

        250.0f, 100.0f, -118.0f, 1.0f, 0.0f, 0.8f, 1.0f, 0.7f, 1.0f, -1.0f,
-1.0f, 0.0f, 0.0f,

        250.0f, 150.0f, -118.0f, 1.0f, 0.0f, 0.8f, 1.0f, 0.7f, 1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, 150.0f, -118.0f, 1.0f, 0.0f, 0.8f, 1.0f, 0.7f, -1.0f, 1.0f,
-1.0f, 0.0f, 0.0f,

        200.0f, 100.0f, -10.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.4f, -1.0f, -1.0f,
1.0f, 0.0f, 0.0f,

        250.0f, 100.0f, -10.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.4f, 1.0f, -1.0f,
1.0f, 0.0f, 0.0f,
```

```
        250.0f, 150.0f, -10.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.4f, 1.0f, 1.0f,
1.0f, 0.0f, 0.0f,
        200.0f, 150.0f, -10.0f, 1.0f, 1.0f, 1.0f, 1.0f, 0.4f, -1.0f, 1.0f,
1.0f, 0.0f, 0.0f,


    };


    // indicii pentru varfuri
    GLubyte Indices[] = {


        // fetele "mesei"
         1, 2, 0,   2, 0, 3,
         2, 3, 6,   6, 3, 7,
         7, 3, 4,   4, 3, 0,
         4, 0, 5,   5, 0, 1,
         1, 2, 5,   5, 2, 6,
         5, 6, 4,   4, 6, 7,


         // fete tabla
        9, 10, 8,   10, 8, 11,
        10, 11, 14,   14, 11, 15,
        15, 11, 12,   12, 11, 8,
        12, 8, 13,   13, 8, 9,
        9, 10, 13,   13, 10, 14,
        13, 14, 12,   12, 14, 15,


         ///* fetele cubului
         //5, 6, 4,   6, 4, 7,
         //6, 7, 10, 10, 7, 11,
         //11, 7, 8,   8, 7, 4,
         //8, 4, 9,   9, 4, 5,
         //5, 6, 9,   9, 6, 10,
         //9, 10, 8,  8, 10, 11,
         // fetele conului
```

```
 //12, 13, 18,
 //13, 14, 18,
 //14, 15, 18,
 //15, 16, 18,
 //16, 17, 18,
 //17, 12, 18,*/

 //conturul
0,1,2,3,
0,4, 1,5, 2,6,  3,7,
4,5,6,7,

//tabla
17,18,16, 18,16,19,

//picioare masa
21, 22, 20,   22, 20, 23,
 22, 23, 26,   26, 23, 27,
 27, 23, 24,   24, 23, 20,
 24, 20, 25,   25, 20, 21,
 21, 22, 25,   25, 22, 26,
 25, 26, 24,   24, 26, 27,

 //podea
 29, 30, 28,  30, 28, 31,

 //perete stanga
 33, 34, 32,  34, 32, 35,

 //perete dreapta
 37, 38, 36,  38, 36, 39,

 //perete spate
```

```
 41, 42, 40,  42, 40, 43,

 //tavan
 45, 46, 44,  46, 44, 47,

 //usa
 49, 50, 48,   50, 48, 51,
 50, 51, 54,   54, 51, 55,
 55, 51, 52,   52, 51, 48,
 52, 48, 53,   53, 48, 49,
 49, 50, 53,   53, 50, 54,
 53, 54, 52,   52, 54, 55,

 //suport
57, 58, 56,   58, 56, 59,
58, 59, 62,   62, 59, 63,
63, 59, 60,   60, 59, 56,
60, 56, 61,   61, 56, 57,
57, 58, 61,   61, 58, 62,
61, 62, 60,   60, 62, 63,


// geam
 65, 66, 64,  66, 64, 67,

 // geam
 69, 70, 68,   70, 68, 71,
 70, 71, 74,   74, 71, 75,
 75, 71, 72,   72, 71, 68,
 72, 68, 73,   73, 68, 69,
 69, 70, 73,   73, 70, 74,
 73, 74, 72,   72, 74, 75,
```

```
    //carte
    77, 78, 76,   78, 76, 79,
    78, 79, 82,   82, 79, 83,
    83, 79, 80,   80, 79, 76,
    80, 76, 81,   81, 76, 77,
    77, 78, 81,   81, 78, 82,
    81, 82, 80,   80, 82, 83,

    85, 86, 84,   86, 84, 87,
    86, 87, 90,   90, 87, 91,
    91, 87, 88,   88, 87, 84,
    88, 84, 89,   89, 84, 85,
    85, 86, 89,   89, 86, 90,
    89, 90, 88,   88, 90, 91,


   93, 94, 92, 94, 92, 95,
   94, 95, 98, 98, 95, 99,
   99, 95, 96, 96, 95, 92,
   96, 92, 97, 97, 92, 93,
   93, 94, 97, 97, 94, 98,
   97, 98, 96, 96, 98, 99,

   101, 102, 100, 102, 100, 103,

   //pahar
   105, 106, 104, 106, 104, 107,
   106, 107, 110, 110, 107, 111,
   111, 107, 108, 108, 107, 104,
   108, 104, 109, 109, 104, 105,
   105, 108, 109, 109, 106, 120,
   109, 110, 108, 108, 110, 111,
};
```

```
    // se creeaza un VAO (Vertex Array Object) - util cand se utilizeaza
mai multe VBO

    glGenVertexArrays(1, &VaoId);

    // se creeaza un buffer nou (atribute)

    glGenBuffers(1, &VboId);

    // se creeaza un buffer nou (indici)

    glGenBuffers(1, &EboId);


    // legarea VAO

    glBindVertexArray(VaoId);


    // legarea buffer-ului "Array"

    glBindBuffer(GL_ARRAY_BUFFER, VboId);

    // punctele sunt "copiate" in bufferul curent

    glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices,
GL_STATIC_DRAW);


    // legarea buffer-ului "Element" (indicii)

    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EboId);

    // indicii sunt "copiati" in bufferul curent

    glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(Indices), Indices,
GL_STATIC_DRAW);


    // se activeaza lucrul cu atribute; atributul 0 = pozitie

    glEnableVertexAttribArray(0);

    glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 13 * sizeof(GLfloat),
(GLvoid*)0);


    // se activeaza lucrul cu atribute; atributul 1 = culoare

    glEnableVertexAttribArray(1);
```

```
    glVertexAttribPointer(1, 4, GL_FLOAT, GL_FALSE, 13 * sizeof(GLfloat),
(GLvoid*)(4 * sizeof(GLfloat)));


    // se activeaza lucrul cu atribute; atributul 2 = normale

    glEnableVertexAttribArray(2);

    glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, 13 * sizeof(GLfloat),
(GLvoid*)(8 * sizeof(GLfloat)));


    // se activeaza lucrul cu atribute; atributul 3 = texturare

    glEnableVertexAttribArray(3);

    glVertexAttribPointer(3, 2, GL_FLOAT, GL_FALSE, 13 * sizeof(GLfloat),
(GLvoid*)(11 * sizeof(GLfloat)));


}

void DestroyVBO(void)

{

    glDisableVertexAttribArray(3);

    glDisableVertexAttribArray(2);

    glDisableVertexAttribArray(1);

    glDisableVertexAttribArray(0);


    glBindBuffer(GL_ARRAY_BUFFER, 0);


    glDeleteBuffers(1, &VboId);

    glDeleteBuffers(1, &EboId);


    glBindVertexArray(0);

    glDeleteVertexArrays(1, &VaoId);

}




void CreateShaders(void)

{

    ProgramId = LoadShaders("Shader.vert", "Shader.frag");
```

```
    glUseProgram(ProgramId);
}

void DestroyShaders(void)
{
    glDeleteProgram(ProgramId);
}

void Initialize(void)
{

    myMatrix = glm::mat4(1.0f);
    matrRot = glm::rotate(glm::mat4(1.0f), PI / 8, glm::vec3(0.0, 0.0,
1.0));
    glClearColor(0.7f, 0.9f, 0.9f, 0.0f); // culoarea de fond a ecranului
    CreateShaders();

}
void RenderFunction(void)
{

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);

    //pozitia observatorului
    Obsx = Refx + dist * cos(alpha) * cos(beta);
    Obsy = Refy + dist * cos(alpha) * sin(beta);
    Obsz = Refz + dist * sin(alpha);

    // reperul de vizualizare
    glm::vec3 Obs = glm::vec3(Obsx, Obsy, Obsz);   // se schimba pozitia
observatorului
    glm::vec3 PctRef = glm::vec3(Refx, Refy, Refz); // pozitia punctului de
referinta
```

```
    glm::vec3 Vert = glm::vec3(Vx, Vy, Vz); // verticala din planul de
vizualizare

    view = glm::lookAt(Obs, PctRef, Vert);


    projection = glm::infinitePerspective(fov, GLfloat(width) /
GLfloat(height), znear);

    myMatrix = glm::mat4(1.0f);

    // matricea pentru umbra

    float D = -0.5f;

    matrUmbra[0][0] = zL + D; matrUmbra[0][1] = 0; matrUmbra[0][2] = 0;
matrUmbra[0][3] = 0;

    matrUmbra[1][0] = 0; matrUmbra[1][1] = zL + D; matrUmbra[1][2] = 0;
matrUmbra[1][3] = 0;

    matrUmbra[2][0] = -xL; matrUmbra[2][1] = -yL; matrUmbra[2][2] = D;
matrUmbra[2][3] = -1;

    matrUmbra[3][0] = -D * xL; matrUmbra[3][1] = -D * yL; matrUmbra[3][2] =
-D * zL; matrUmbra[3][3] = zL;


    CreateVBO();


    // variabile uniforme pentru shaderul de varfuri

    myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);

    matrUmbraLocation = glGetUniformLocation(ProgramId, "matrUmbra");

    glUniformMatrix4fv(matrUmbraLocation, 1, GL_FALSE, &matrUmbra[0][0]);

    viewLocation = glGetUniformLocation(ProgramId, "view");

    glUniformMatrix4fv(viewLocation, 1, GL_FALSE, &view[0][0]);

    projLocation = glGetUniformLocation(ProgramId, "projection");

    glUniformMatrix4fv(projLocation, 1, GL_FALSE, &projection[0][0]);


    // Variabile uniforme pentru iluminare

    GLint lightColorLoc = glGetUniformLocation(ProgramId, "lightColor");

    GLint lightPosLoc = glGetUniformLocation(ProgramId, "lightPos");

    GLint viewPosLoc = glGetUniformLocation(ProgramId, "viewPos");

    GLint codColLocation = glGetUniformLocation(ProgramId, "codCol");

    glUniform3f(lightColorLoc, 1.0f, 1.0f, 1.0f);
```

```
glUniform3f(lightPosLoc, xL, yL, zL);
glUniform3f(viewPosLoc, Obsx, Obsy, Obsz);




// desenare cub
codCol = 0;
glUniform1i(codColLocation, codCol);
myMatrix = glm::mat4(1.0f);
myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);


glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(0));
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(36));
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(88));
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(94));

glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(136));
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(142));
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(148));
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(154));
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(160));
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(196));
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(232));

glEnable(GL_BLEND);
glDepthMask(GL_FALSE);
glBlendFunc(GL_SRC_ALPHA, GL_SRC_ALPHA);
codCol = 0;
glUniform1i(codColLocation, codCol);
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(238));
```

```
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(388));
    glDepthMask(GL_TRUE);
    glDisable(GL_BLEND);

    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(274));
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(310));
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(346));
    glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(382));



    myMatrix = glm::translate(glm::mat4(1.0f), glm::vec3(-900.f, 0.f,
0.0));
    myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(94));

    myMatrix = glm::translate(glm::mat4(1.0f), glm::vec3(-900.f, -900.f,
0.0));
    myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(94));

    myMatrix = glm::translate(glm::mat4(1.0f), glm::vec3(0.f, -900.f,
0.0));
    myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(94));



    //desenare banci
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 4; j++) {
```

```
            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.5f,
1.0)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 2000.f, j *
1800 - 900.f, -200.0));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(0));

            codCol = 2;

            glUniform1i(codColLocation, codCol);

            glLineWidth(3);

            glDrawElements(GL_LINE_LOOP, 4, GL_UNSIGNED_BYTE, (void*)(72));

            glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(76));

            glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(78));

            glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(80));

            glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(82));

            glDrawElements(GL_LINE_LOOP, 4, GL_UNSIGNED_BYTE, (void*)(84));



            codCol = 0;

            glUniform1i(codColLocation, codCol);

            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.5f,
0.7f)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 2000.f, j *
1800 - 900.f, -350.0));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(94));

            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.5f,
0.7f)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 2000.f, j *
1800 - 1800.f, -350.0));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(94));

            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.5f,
0.7f)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 1100.f, j *
1800 - 1800.f, -350.0));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
```

```
            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(94));

            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.5f,
0.7f)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 1100.f, j *
1800 - 900.f, -350.0));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(94));



            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.75f,
1.0)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 1500.f, j *
1200 - 200.f, -200.0)) * glm::rotate(glm::mat4(1.0f),
glm::radians((i+j)*10.0f), glm::vec3(0.0f, 0.0f, 1.0f));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(274));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(310));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(346));

            glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE,
(void*)(382));



            myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f, 0.75f,
1.0)) * glm::translate(glm::mat4(1.0f), glm::vec3(i * 2000 + 1500.f, j *
1200 - 500.f, -200.0)) * glm::rotate(glm::mat4(1.0f), glm::radians((i +
2*j) * 10.0f), glm::vec3(0.0f, 0.0f, 1.0f));

            myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

            glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(274));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(310));

            glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE,
(void*)(346));
```

```
            glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE,
(void*)(382));

        }


    }




    //desenare fereastra

    codCol = 4;

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

        myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(1.0f, 0.5f, 1.8f))
* glm::translate(glm::mat4(1.0f), glm::vec3(i * 700.0f - 350.0f, -2400.0f,
465.0));

        myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

        glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);

        glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(94));

    }



    for (int i = 0; i < 2; i++) {

        myMatrix = glm::scale(glm::mat4(1.0f), glm::vec3(4.1f, 0.5f, 2.0f))
* glm::translate(glm::mat4(1.0f), glm::vec3(520.0f, -2370.0f, i*300 +
430.0)) * glm::rotate(glm::mat4(1.0f), glm::radians(90.0f), glm::vec3(0.0f,
1.0f, 0.0f));

        myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

        glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);

        glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_BYTE, (void*)(94));

    }




    codCol = 2;

    glUniform1i(codColLocation, codCol);

    myMatrix = glm::mat4(1.0f);

    myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");

    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);

    glLineWidth(3);

    glDrawElements(GL_LINE_LOOP, 4, GL_UNSIGNED_BYTE, (void*)(72));
```

```
    glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(76));
    glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(78));
    glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(80));
    glDrawElements(GL_LINES, 2, GL_UNSIGNED_BYTE, (void*)(82));
    glDrawElements(GL_LINE_LOOP, 4, GL_UNSIGNED_BYTE, (void*)(84));


    codCol = 3;
    glUniform1i(codColLocation, codCol);
    myMatrix = glm::mat4(1.0f);
    myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);

    LoadTexture();
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, texture);
    glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
    glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, (void*)(130));

    glutSwapBuffers();
    glFlush();
}
void Cleanup(void)
{
    DestroyShaders();
    DestroyVBO();
}

int main(int argc, char* argv[])
{

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
```

```
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(1200, 900);
    glutCreateWindow("Proiect 3D");
    glewInit();
    Initialize();
    glutIdleFunc(RenderFunction);
    glutDisplayFunc(RenderFunction);
    glutKeyboardFunc(processNormalKeys);
    glutSpecialFunc(processSpecialKeys);



    glutCloseFunc(Cleanup);
    glutMainLoop();

}
```

Shadere

Shader frag

```
// Shader-ul de fragment / Fragment shader

 #version 400

in vec3 FragPos;
in vec3 Normal;
in vec3 inLightPos;
in vec3 inViewPos;
in vec3 ex_Color;
in vec2 tex_Coord;

out vec4 out_Color;

uniform vec3 lightColor;
uniform int codCol;
uniform sampler2D myTexture;

void main(void)
  {
      // Ambient
    float ambientStrength = 0.2f;
    vec3 ambient = ambientStrength * lightColor;

    // Diffuse
    vec3 normala = normalize(Normal);
```

```
     vec3 lightDir = normalize(inLightPos - FragPos);
    // vec3 dir=vec3(0.0,-150.0,200.0); // sursa directionala
    // vec3 lightDir=normalize(dir);
    float diff = max(dot(normala, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    // Specular
    float specularStrength = 0.2f;
    vec3 viewDir = normalize(inViewPos - FragPos);//vector catre observator normalizat
(V)
    vec3 reflectDir = reflect(-lightDir, normala); // reflexia razei de lumina (R)
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 1);
    vec3 specular = specularStrength * spec * lightColor;
    vec3 emission=vec3(0.0, 0.0, 0.0);
    vec3 result = emission+(ambient + diffuse + specular) * ex_Color;
       //out_Color = vec4(result, 1.0f);

    // Efect de ceata
    vec3 fogColor = vec3(0.5, 0.5, 0.5);
    float dist=length(inViewPos - FragPos);
    // float fogFactor=exp(-0.002*dist); // intre 0 si 1; 1 corespunde aproape de
obiect
    float fogFactor=1.0;
    out_Color = vec4(mix(fogColor,result,fogFactor), 1.0f);

    if ( codCol==1)
              out_Color=vec4 (0.0, 0.0, 0.0, 0.0);
    if ( codCol==2)
       out_Color=vec4 (0.6, 0.4, 0.2, 0.0);
    if ( codCol==3)
       out_Color = mix(texture(myTexture, tex_Coord), out_Color, 0.2);

  }
```

Shader vert

```
// Shader-ul de varfuri

 #version 400


layout(location=0) in vec4 in_Position;
layout(location=1) in vec3 in_Color;
layout(location=2) in vec3 in_Normal;
layout(location=3) in vec2 texCoord;


out vec3 FragPos;
out vec3 Normal;
out vec3 inLightPos;
out vec3 inViewPos;
out vec3 ex_Color;
out vec2 tex_Coord;

uniform mat4 matrUmbra;
uniform mat4 myMatrix;
uniform mat4 view;
uniform mat4 projection;
```

```
uniform vec3 lightPos;
uniform vec3 viewPos;
uniform vec3 lightColor;
uniform int codCol;


void main(void)
  {
    ex_Color=in_Color;
    tex_Coord = vec2(texCoord.x, 1-texCoord.y);
       if ( codCol==0 || codCol==2 || codCol == 3 )
    {
             gl_Position = projection*view*myMatrix*in_Position;
        Normal=mat3(projection*view*myMatrix)*in_Normal;
        inLightPos= vec3(projection*view*myMatrix* vec4(lightPos, 1.0f));
        inViewPos=vec3(projection*view*myMatrix*vec4(viewPos, 1.0f));
        FragPos = vec3(gl_Position);
    }
       if ( codCol==1 )
             gl_Position = projection*view*matrUmbra*myMatrix*in_Position;
        FragPos = vec3(gl_Position);
   }
```