Proiect 1 - Grafică pe Calculator

- Depășire între 2 dreptunghiuri -

Linte Robert Ovidiu
Popescu Paullo Robertto Karloss
Grupa 331

Conf. Dr. Stupariu Mihai-Sorin

- 2 Noiembrie 2022 -

Cuprins

Introducere

1.1 Modul de organizare al echipei

Echipa noastră:

- Linte Robert Ovidiu
- Popescu Paullo Robertto Karloss

Se va specifica **numele** membrului care a **contribuit** la realizarea fiecărei *etape*.

1.2 Obiectivele proiectului

Simularea unei "depășiri":

- O mașină (un dreptunghi) se deplasează pe o șosea uniform (print translație)
- O altă mașină (alt dreptunghi) vine din spate (tot prin translații)
- La un moment dat a doua mașină intră în depășire
- A doua mașină trece în fața primei mașini
- Se afișează la final câștigătorul "cursei"

Aprofundarea cunoștințelor în OpenGL prin:

- Folosirea translațiilor
- Desenarea obiectelor
- Folosirea culorilor

1.3 Vizionarea proiectului

Puteți viziona demo-ul proiectului aici.

Desenarea obiectelor

Această etapă a fost realizată de ***Popescu Paullo Robertto Karloss***.

2.1 Prezentarea tablei de joc

Tabla de joc conține:

- Șoseua propiu-zisă
- O linie punctată pe post de marcaj rutier
- Două mașini de culori diferite (prima roșie, a doua albastră) care au forma unor dreptunghiuri
- Iarbă pe marginea șoselei
- Un text la finalul șoselei cu mesajul "FINISH", pentru scoate în evidență câștigătorul "cursei"

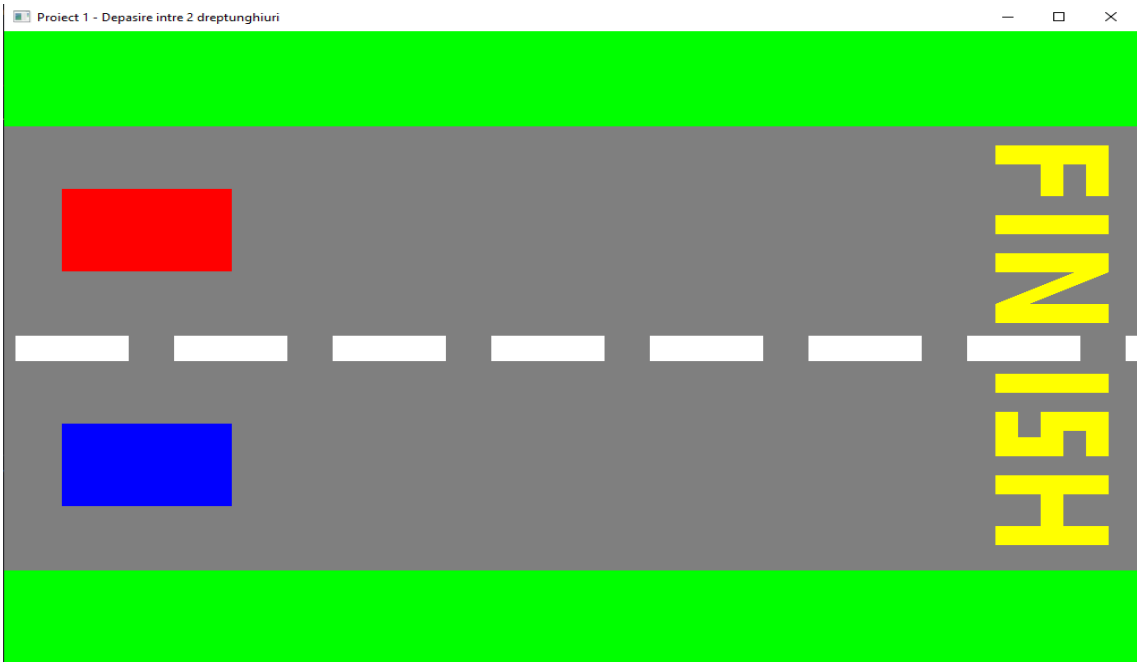

Figura 2.1: Tabla de joc

2.2 Cum a fost construita tabla de joc

Pentru a construi tabla de joc am creat un background (un dreptunghi) de culoare verde. Peste acesta am adăugat șoseaua (un dreptunghi de culoare gri). După care, in interiorul șoselei am creat două dreptunghiuri, unul roșu si unul albastru, pe post de mașini, drepunghiuri albe pe post de linie punctată și alte dreptunghirui de culoare galbenă pentru a scrie mesajul "FINISH".

2.3 Cod sursă

```
606     glDrawArrays(GL_POLYGON, 0, 4);
607     glDrawArrays(GL_POLYGON, 4, 4);
608     glDrawArrays(GL_POLYGON, 8, 4);
609     glDrawArrays(GL_POLYGON, 12, 4);
610     glDrawArrays(GL_POLYGON, 16, 4);
611     glDrawArrays(GL_POLYGON, 20, 4);
612     glDrawArrays(GL_POLYGON, 24, 4);
613     glDrawArrays(GL_POLYGON, 28, 4);
614     glDrawArrays(GL_POLYGON, 32, 4);
615     glDrawArrays(GL_POLYGON, 36, 4);
616     glDrawArrays(GL_POLYGON, 48, 4);
617     glDrawArrays(GL_POLYGON, 52, 4);
618     glDrawArrays(GL_POLYGON, 56, 4);
619     glDrawArrays(GL_POLYGON, 60, 4);
620     glDrawArrays(GL_POLYGON, 64, 4);
621     glDrawArrays(GL_POLYGON, 68, 4);
622     glDrawArrays(GL_POLYGON, 72, 4);
623     glDrawArrays(GL_POLYGON, 76, 4);
624     glDrawArrays(GL_POLYGON, 80, 4);
625     glDrawArrays(GL_POLYGON, 84, 4);
626     glDrawArrays(GL_POLYGON, 88, 4);
627     glDrawArrays(GL_POLYGON, 92, 4);
628     glDrawArrays(GL_POLYGON, 96, 4);
629     glDrawArrays(GL_POLYGON, 100, 4);
630     glDrawArrays(GL_POLYGON, 104, 4);
631     glDrawArrays(GL_POLYGON, 108, 4);
```

Figura 2.2: Cod OpenGL pentru Tabla de Joc

Adăugarea translațiilor

Această etapă a fost realizată de ***Linte Robert Ovidiu***.

3.1 Prezentarea Translației 1

Dreptunghiul roșu pleacă cu o viteză inițială mai mică decât a dreptunghiului albastru.

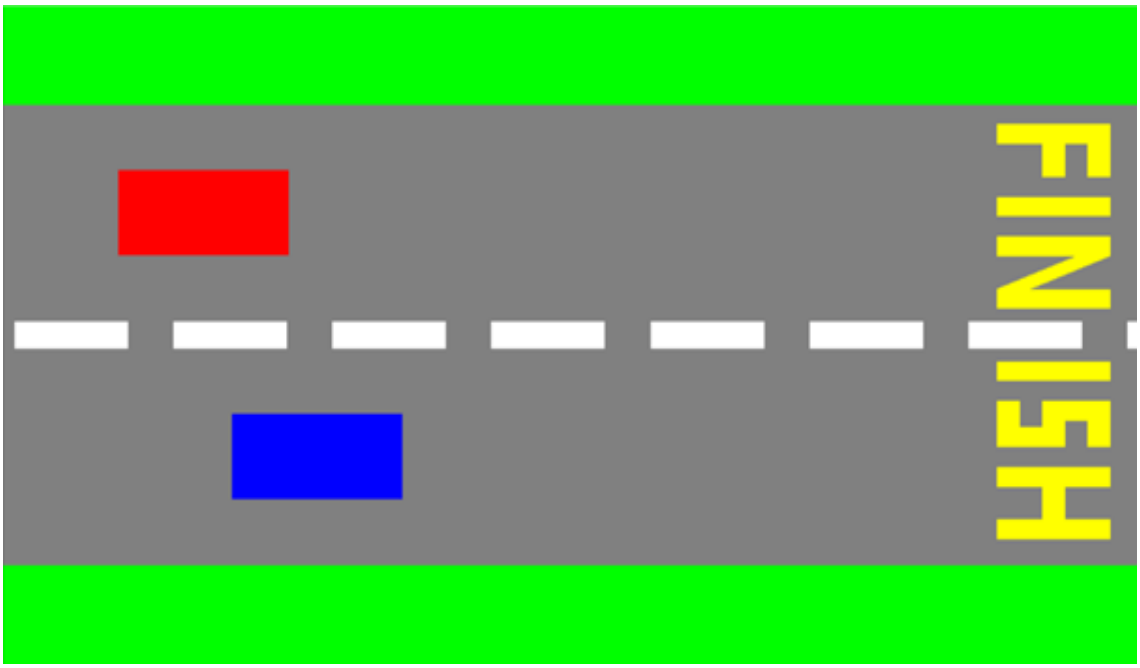

Figura 3.1.1: Deplasarea inițială a dreptunghiurilor

3.1.1 Cod sursă

```
635     // Matricea pentru dreptunghiul rosu
636     myMatrix = resizeMatrix * matrTransl * matrDepl * matrScale2;
637     // Culoarea
638     codCol = 2;
639     // Transmitere variabile uniforme
640     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
641     glUniform1i(codColLocation, codCol);
642     // Apelare DrawArrays
643     glDrawArrays(GL_POLYGON, 40, 4);
644
645     // Matricea pentru dreptunghiul rosu
646     myMatrix = resizeMatrix * matrTransl2 * matrDepl * matrScale2 * matrRot;
647     // Culoarea
648     codCol = 1;
649     // Transmitere variabile uniforme
650     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
651     glUniform1i(codColLocation, codCol);
652     // Apelare DrawArrays
653     glDrawArrays(GL_POLYGON, 44, 4);
654
```

Figura 3.1.2: Cod OpenGL pentru prima translație

```
590     resizeMatrix = glm::ortho(-width, width, -height, height); // scalam, "aducem" scena la "patratul standard" [-1,1]x[-1,1]
591     matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(i, k, 0.0)); // controleaza translatia de-a lungul lui Ox
592     matrTransl3 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, h, 0.0));
593     matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(1.0, 1.0, 0.0)); // plaseaza patratul rosu
594     matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 1.0, 0.0)); // folosita la desenarea patratului rosu
595     matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(j, l, 0.0));
596     matrRot = glm::rotate(glm::mat4(1.0f), angle, glm::vec3(0.0, 0.0, 1.0)); // rotatie folosita la deplasarea patratului rosu
```

Figura 3.1.3: Cod OpenGL pentru prima translație

```
53  void miscad(void)
54  {
55      if (i > -1 && i <= 400 && j <= 750)
56      {
57          i = i + alpha;
58          alpha = +step;
59      }
60      if (j > -1.0 && j <= 150) {
61          j = j + alpha2;
62          alpha2 = +step2;
63      }
```

Figura 3.1.4: Cod OpenGL pentru prima translație

3.2 Prezentarea Translației 2

În momentul în care dreptunghiul albastru, îl depășeste total pe cel roșu, începe procesul de depășire *(se aplică o rotație de 0.5 pe dreptunghiul albastru cât și o translație pe diagonală, iar în final se aplică o rotație inversă pentru a-l aduce pe poziția inițială, i.e. paralel cu axa Ox).*

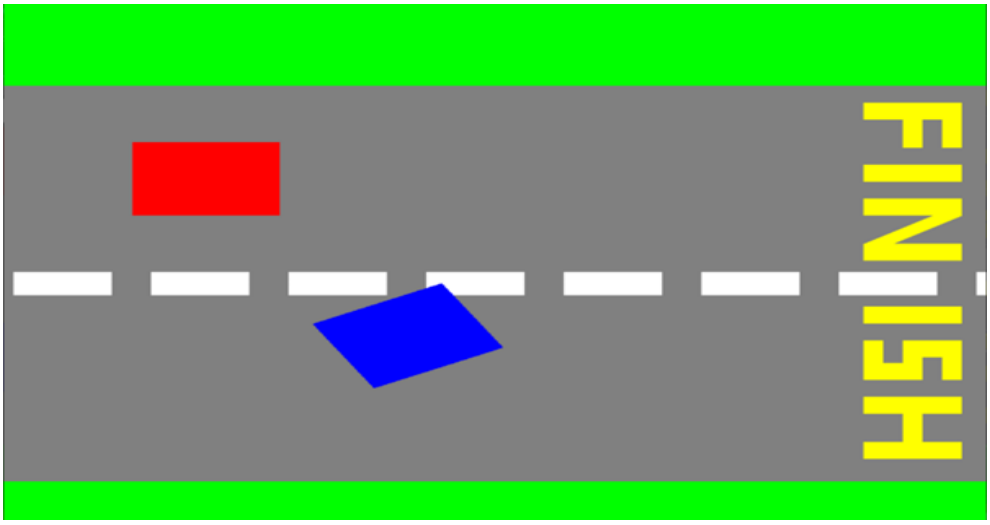

Figura 3.2.1: Depășire dreptunghi roșu

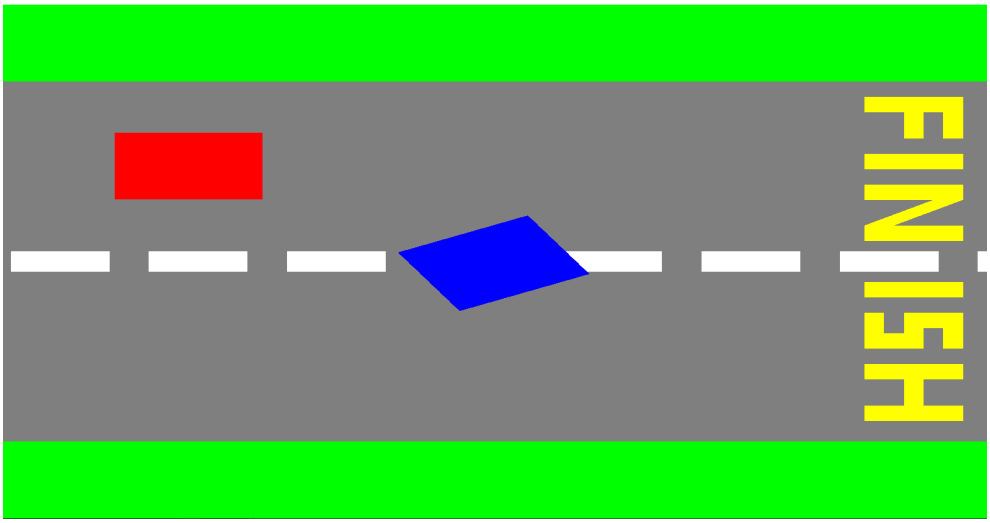

Figura 3.2.2: Depășire dreptunghi roșu

3.2.1 Cod sursă

```
53  void miscad(void)
54  {
55      if (i > -1 && i <= 400 && j <= 750)
56      {
57          i = i + alpha;
58          alpha = +step;
59      }
60      if (j > -1.0 && j <= 150) {
61          j = j + alpha2;
62          alpha2 = +step2;
63      }
```

Figura 3.2.3: Cod OpenGL pentru a doua translație

3.3 Prezentarea Translației 3

După translația 2, dreptunghiul albastru accelerează până trece linia de *"FINISH"*.

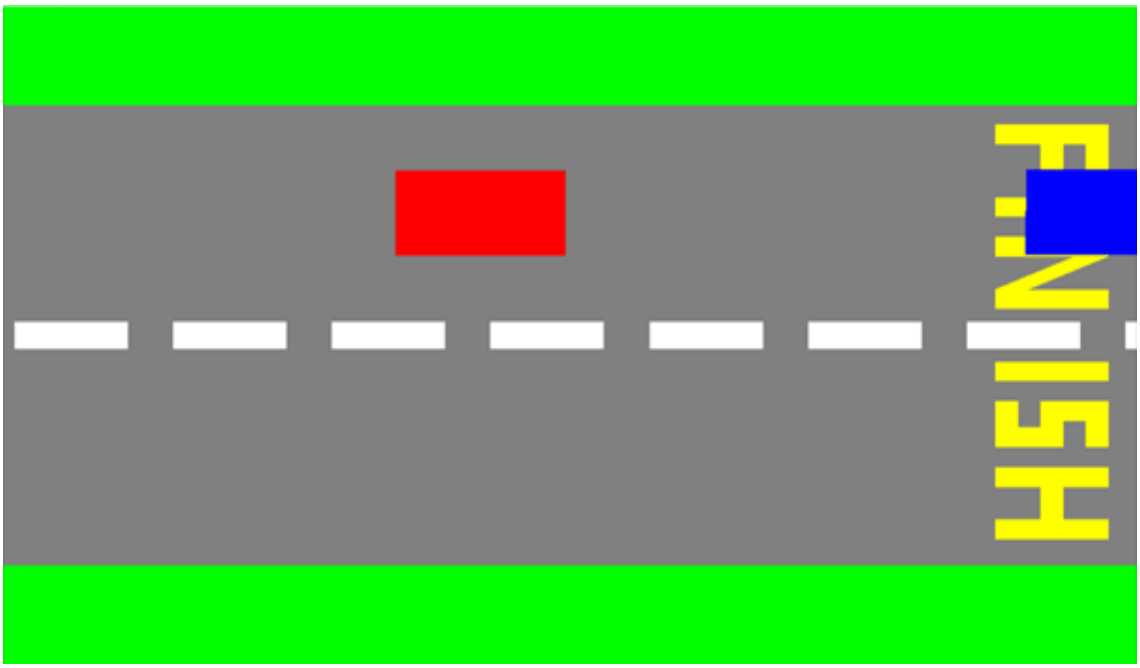

Figura 3.2.1: Accelerare dreptunghi albastru

3.3.1 Cod sursă

```
82    if (l >= 370 && j <= 850 && angle <= 0) {
83        j = j + alpha2;
84        alpha2 = +step3;
85    }
```

Figura 3.3.1: Cod OpenGL pentru a treia translație

Afișarea câștigătorului

4.1 Prezentarea înainte de translația finală

Literele sunt inițial puse în afara ecranului *(nu sunt vizibile în tabla de joc)*.

4.1.1 Cod sursă

```
82    if (l >= 370 && j <= 850 && angle <= 0) {
83        j = j + alpha2;
84        alpha2 = +step3;
85    }
```

Figura 4.1.1: Cod OpenGL pentru literele ascunse

Această etapă a fost realizată de ***Popescu Paullo Robertto Karloss***.

4.2 Prezentare după translație

După ce dreptunghiul albastru reușește să treacă linia de *"FINISH"*, sunt translatate literele în zona de sus a tablei de joc.

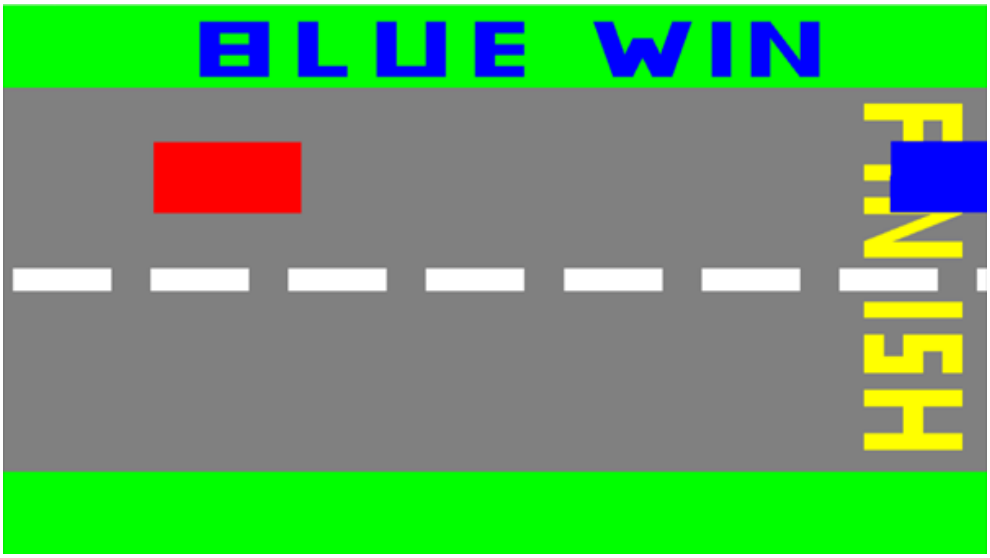

Figura 4.1.1: Cod OpenGL pentru a prima translație

4.2.1 Cod sursă

```
86      if (j >= 750 && h >= -230) {
87          h = h - alpha4;
88          alpha4 = +step3;
89      }
```

Figura 4.1.1: Cod OpenGL pentru translația finală

Această etapă a fost realizată de ***Linte Robert Ovidiu***.

Codul Sursă Complet

Codul îl puteți găsii în fișierul ***proiectGrafica.cpp*** sau atașat mai jos.

```
/* DESCRIERE: DEPASIRE INTRE DREPTUNGHIURI -- varianta cu OpenGL "nou"
  - utilizeaza diverse transformari si compunerea acestora folosind biblioteca glm
  - functii pentru utilizarea mouse-ului
*/

#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"

// Din biblioteca glm
#include "glm/glm.hpp"
#include "glm/gtc/matrix_transform.hpp"
#include "glm/gtx/transform.hpp"
#include "glm/gtc/type_ptr.hpp"

using namespace std;

GLuint
VaoId,
VboId,
ColorBufferId,
ProgramId,
myMatrixLocation,
matrScaleLocation,
matrTranslLocation,
matrRotlLocation,
codColLocation;
```

```
33
34 int codCol;
35 float PI = 3.141592, angle = 0;
36 float tx = 0; float ty = 0;
37 float width = 500, height = 500;
38 float i = 0.0, j = 0.0, h = 0.0, k = 0.0, l = 0.0,alpha4 = 0.0,
      alpha2 = 0.0, alpha3 = 0.0, step2 = 0.5, step3 = 1.5, alpha =
      0.0, step = 0.1, beta = 0.003, ok = 0;
39 glm::mat4
40 myMatrix, resizeMatrix, matrTransl, matrTransl2, matrTransl3,
      matrScale1, matrScale2, matrRot, matrDepl;
41
42 void displayMatrix()
43 {
44   for (int ii = 0; ii < 4; ii++)
45   {
46     for (int jj = 0; jj < 4; jj++)
47       cout << myMatrix[ii][jj] << "  ";
48     cout << endl;
49   };
50   cout << "\n";
51 };
52
53 void miscad(void)
54 {
55   if (i > -1 && i <= 400 && j <= 750)
56   {
57     i = i + alpha;
58     alpha = +step;
59   }
60   if (j > -1.0 && j <= 150) {
61     j = j + alpha2;
62     alpha2 = +step2;
63   }
64
65   if (j >= 150 && j <= 250 && angle <= 0.5) {
66     angle += beta;
67     l = l + alpha3;
68     alpha3 = +step3;
69   }
70
```

```
  if (l >= 200 && l <= 370 && angle >= 0.5) {
    j = j + alpha2;
    alpha2 = +step2;
    l = l + alpha3;
    alpha3 = +step2;
  }
  if (l >= 370 && angle <= 0.6 && angle >= 0) {
    angle -= beta;
    j = j + alpha2;
    alpha2 = +step3;
  }
  if (l >= 370 && j <= 850 && angle <= 0) {
    j = j + alpha2;
    alpha2 = +step3;
  }
  if (j >= 750 && h >= -230) {
    h = h - alpha4;
    alpha4 = +step3;
  }

  glutPostRedisplay();
}




void mouse(int button, int state, int x, int y)
{
  switch (button) {
  case GLUT_LEFT_BUTTON:
    if (state == GLUT_DOWN) {
      alpha = -step;
      alpha2 = -step2;
    }
    glutIdleFunc(miscad);
    break;
  case GLUT_RIGHT_BUTTON:
    if (state == GLUT_DOWN) {
      alpha = step;
      alpha2 = step2;
    }
```

```
    glutIdleFunc(miscad);
    break;
  default:
    break;
  }
}

void CreateVBO(void)
{
  // varfurile
  GLfloat Vertices[] = {
    // varfuri pentru axe
    -500.0f, 500.0f, 0.0f, 1.0f,
    500.0f, 500.0f, 0.0f, 1.0f,
    500.0f, -500.0f, 0.0f, 1.0f,
    -500.0f, -500.0f, 0.0f, 1.0f,

    -500.0f, 350.0f, 0.0f, 1.0f,
    500.0f, 350.0f, 0.0f, 1.0f,
    500.0f, -350.0f, 0.0f, 1.0f,
    -500.0f, -350.0f, 0.0f, 1.0f,

    -490.0f, 20.0f, 0.0f, 1.0f,
    -390.0f, 20.0f, 0.0f, 1.0f,
    -390.0f, -20.0f, 0.0f, 1.0f,
    -490.0f, -20.0f, 0.0f, 1.0f,

    -350.0f, 20.0f, 0.0f, 1.0f,
    -250.0f, 20.0f, 0.0f, 1.0f,
    -250.0f, -20.0f, 0.0f, 1.0f,
    -350.0f, -20.0f, 0.0f, 1.0f,

    -210.0f, 20.0f, 0.0f, 1.0f,
    -110.0f, 20.0f, 0.0f, 1.0f,
    -110.0f, -20.0f, 0.0f, 1.0f,
    -210.0f, -20.0f, 0.0f, 1.0f,

    -70.0f, 20.0f, 0.0f, 1.0f,
    30.0f, 20.0f, 0.0f, 1.0f,
    30.0f, -20.0f, 0.0f, 1.0f,
    -70.0f, -20.0f, 0.0f, 1.0f,
```

```
153
154     70.0f, 20.0f, 0.0f, 1.0f,
155     170.0f, 20.0f, 0.0f, 1.0f,
156     170.0f, -20.0f, 0.0f, 1.0f,
157     70.0f, -20.0f, 0.0f, 1.0f,
158
159     210.0f, 20.0f, 0.0f, 1.0f,
160     310.0f, 20.0f, 0.0f, 1.0f,
161     310.0f, -20.0f, 0.0f, 1.0f,
162     210.0f, -20.0f, 0.0f, 1.0f,
163
164     350.0f, 20.0f, 0.0f, 1.0f,
165     450.0f, 20.0f, 0.0f, 1.0f,
166     450.0f, -20.0f, 0.0f, 1.0f,
167     350.0f, -20.0f, 0.0f, 1.0f,
168
169     490.0f, 20.0f, 0.0f, 1.0f,
170     500.0f, 20.0f, 0.0f, 1.0f,
171     500.0f, -20.0f, 0.0f, 1.0f,
172     490.0f, -20.0f, 0.0f, 1.0f,
173
174     -450.0f, 250.0f, 0.0f, 1.0f,
175     -300.0f, 250.0f, 0.0f, 1.0f,
176     -300.0f, 120.0f, 0.0f, 1.0f,
177     -450.0f, 120.0f, 0.0f, 1.0f,
178
179     -450.0f, -250.0f, 0.0f, 1.0f,
180     -300.0f, -250.0f, 0.0f, 1.0f,
181     -300.0f, -120.0f, 0.0f, 1.0f,
182     -450.0f, -120.0f, 0.0f, 1.0f,
183
184     //FINISH
185     //F
186     475.0f, 320.0f, 0.0f, 1.0f,
187     375.0f, 320.0f, 0.0f, 1.0f,
188     375.0f, 290.0f, 0.0f, 1.0f,
189     475.0f, 290.0f, 0.0f, 1.0f,
190
191     475.0f, 290.0f, 0.0f, 1.0f,
192     475.0f, 240.0f, 0.0f, 1.0f,
193     455.0f, 240.0f, 0.0f, 1.0f,
```

```
    455.0f, 290.0f, 0.0f, 1.0f,

    435.0f, 290.0f, 0.0f, 1.0f,
    435.0f, 240.0f, 0.0f, 1.0f,
    415.0f, 240.0f, 0.0f, 1.0f,
    415.0f, 290.0f, 0.0f, 1.0f,
    //I
    475.0f, 210.0f, 0.0f, 1.0f,
    375.0f, 210.0f, 0.0f, 1.0f,
    375.0f, 180.0f, 0.0f, 1.0f,
    475.0f, 180.0f, 0.0f, 1.0f,
    //N
    475.0f, 150.0f, 0.0f, 1.0f,
    375.0f, 150.0f, 0.0f, 1.0f,
    375.0f, 120.0f, 0.0f, 1.0f,
    475.0f, 120.0f, 0.0f, 1.0f,

    475.0f, 70.0f, 0.0f, 1.0f,
    375.0f, 70.0f, 0.0f, 1.0f,
    375.0f, 40.0f, 0.0f, 1.0f,
    475.0f, 40.0f, 0.0f, 1.0f,

    475.0f, 120.0f, 0.0f, 1.0f,
    445.0f, 120.0f, 0.0f, 1.0f,
    375.0f, 70.0f, 0.0f, 1.0f,
    405.0f, 70.0f, 0.0f, 1.0f,

    //I
    475.0f, -40.0f, 0.0f, 1.0f,
    375.0f, -40.0f, 0.0f, 1.0f,
    375.0f, -70.0f, 0.0f, 1.0f,
    475.0f, -70.0f, 0.0f, 1.0f,

    //S
    475.0f, -100.0f, 0.0f, 1.0f,
    455.0f, -100.0f, 0.0f, 1.0f,
    455.0f, -170.0f, 0.0f, 1.0f,
    475.0f, -170.0f, 0.0f, 1.0f,

    475.0f, -100.0f, 0.0f, 1.0f,
    415.0f, -100.0f, 0.0f, 1.0f,
```

```
    415.0f, -130.0f, 0.0f, 1.0f,
    475.0f, -130.0f, 0.0f, 1.0f,

    415.0f, -100.0f, 0.0f, 1.0f,
    435.0f, -100.0f, 0.0f, 1.0f,
    435.0f, -170.0f, 0.0f, 1.0f,
    415.0f, -170.0f, 0.0f, 1.0f,

    415.0f, -170.0f, 0.0f, 1.0f,
    415.0f, -140.0f, 0.0f, 1.0f,
    375.0f, -140.0f, 0.0f, 1.0f,
    375.0f, -170.0f, 0.0f, 1.0f,

    375.0f, -100.0f, 0.0f, 1.0f,
    395.0f, -100.0f, 0.0f, 1.0f,
    395.0f, -170.0f, 0.0f, 1.0f,
    375.0f, -170.0f, 0.0f, 1.0f,

    //H
    475.0f, -200.0f, 0.0f, 1.0f,
    375.0f, -200.0f, 0.0f, 1.0f,
    375.0f, -230.0f, 0.0f, 1.0f,
    475.0f, -230.0f, 0.0f, 1.0f,

    475.0f, -280.0f, 0.0f, 1.0f,
    375.0f, -280.0f, 0.0f, 1.0f,
    375.0f, -310.0f, 0.0f, 1.0f,
    475.0f, -310.0f, 0.0f, 1.0f,

    435.0f, -310.0f, 0.0f, 1.0f,
    415.0f, -310.0f, 0.0f, 1.0f,
    415.0f, -200.0f, 0.0f, 1.0f,
    435.0f, -200.0f, 0.0f, 1.0f,

    //Blue win

    //B
    -300.0f, 700.0f, 0.0f, 1.0f,
    -300.0f, 600.0f, 0.0f, 1.0f,
    -280.0f, 600.f, 0.0f, 1.0f,
    -280.0f, 700.0f, 0.0f, 1.0f,
```

```
276
277     -250.0f, 700.0f, 0.0f, 1.0f,
278     -250.0f, 600.0f, 0.0f, 1.0f,
279     -230.0f, 600.f, 0.0f, 1.0f,
280     -230.0f, 700.0f, 0.0f, 1.0f,
281
282     -300.0f, 700.0f, 0.0f, 1.0f,
283     -300.0f, 680.0f, 0.0f, 1.0f,
284     -230.0f, 680.f, 0.0f, 1.0f,
285     -230.0f, 700.0f, 0.0f, 1.0f,
286
287     -300.0f, 660.0f, 0.0f, 1.0f,
288     -300.0f, 640.0f, 0.0f, 1.0f,
289     -230.0f, 640.f, 0.0f, 1.0f,
290     -230.0f, 660.0f, 0.0f, 1.0f,
291
292     -300.0f, 620.0f, 0.0f, 1.0f,
293     -300.0f, 600.0f, 0.0f, 1.0f,
294     -230.0f, 600.f, 0.0f, 1.0f,
295     -230.0f, 620.0f, 0.0f, 1.0f,
296
297     //L
298     -200.0f, 700.0f, 0.0f, 1.0f,
299     -200.0f, 600.0f, 0.0f, 1.0f,
300     -180.0f, 600.f, 0.0f, 1.0f,
301     -180.0f, 700.0f, 0.0f, 1.0f,
302
303     -200.0f, 620.0f, 0.0f, 1.0f,
304     -200.0f, 600.0f, 0.0f, 1.0f,
305     -150.0f, 600.f, 0.0f, 1.0f,
306     -150.0f, 620.0f, 0.0f, 1.0f,
307
308     //U
309     -120.0f, 700.0f, 0.0f, 1.0f,
310     -120.0f, 600.0f, 0.0f, 1.0f,
311     -100.0f, 600.f, 0.0f, 1.0f,
312     -100.0f, 700.0f, 0.0f, 1.0f,
313
314     -70.0f, 700.0f, 0.0f, 1.0f,
315     -70.0f, 600.0f, 0.0f, 1.0f,
316     -50.0f, 600.f, 0.0f, 1.0f,
```

```
317     -50.0f, 700.0f, 0.0f, 1.0f,
318
319     -120.0f, 620.0f, 0.0f, 1.0f,
320     -50.0f, 620.0f, 0.0f, 1.0f,
321     -50.0f, 600.0f, 0.0f, 1.0f,
322     -120.0f, 600.0f, 0.0f, 1.0f,
323
324     //E
325     -20.0f, 700.0f, 0.0f, 1.0f,
326     -20.0f, 600.0f, 0.0f, 1.0f,
327     0.0f, 600.f, 0.0f, 1.0f,
328     0.0f, 700.0f, 0.0f, 1.0f,
329
330     -20.0f, 620.0f, 0.0f, 1.0f,
331     30.0f, 620.0f, 0.0f, 1.0f,
332     30.0f, 600.0f, 0.0f, 1.0f,
333     -20.0f, 600.0f, 0.0f, 1.0f,
334
335     -20.0f, 660.0f, 0.0f, 1.0f,
336     30.0f, 660.0f, 0.0f, 1.0f,
337     30.0f, 640.0f, 0.0f, 1.0f,
338     -20.0f, 640.0f, 0.0f, 1.0f,
339
340     -20.0f, 700.0f, 0.0f, 1.0f,
341     30.0f, 700.0f, 0.0f, 1.0f,
342     30.0f, 680.0f, 0.0f, 1.0f,
343     -20.0f, 680.0f, 0.0f, 1.0f,
344
345     //W
346     100.0f, 700.0f, 0.0f, 1.0f,
347     120.0f, 700.0f, 0.0f, 1.0f,
348     140.0f, 600.0f, 0.0f, 1.0f,
349     120.0f, 600.0f, 0.0f, 1.0f,
350
351     140.0f, 600.0f, 0.0f, 1.0f,
352     120.0f, 600.0f, 0.0f, 1.0f,
353     140.0f, 670.0f, 0.0f, 1.0f,
354     160.0f, 670.0f, 0.0f, 1.0f,
355
356     140.0f, 670.0f, 0.0f, 1.0f,
357     160.0f, 670.0f, 0.0f, 1.0f,
```

```
    180.0f, 600.0f, 0.0f, 1.0f,
    160.0f, 600.0f, 0.0f, 1.0f,

    180.0f, 600.0f, 0.0f, 1.0f,
    160.0f, 600.0f, 0.0f, 1.0f,
    180.0f, 700.0f, 0.0f, 1.0f,
    200.0f, 700.0f, 0.0f, 1.0f,

    //I

    220.0f, 700.0f, 0.0f, 1.0f,
    240.0f, 700.0f, 0.0f, 1.0f,
    240.0f, 600.0f, 0.0f, 1.0f,
    220.0f, 600.0f, 0.0f, 1.0f,

    //N

    260.0f, 700.0f, 0.0f, 1.0f,
    280.0f, 700.0f, 0.0f, 1.0f,
    280.0f, 600.0f, 0.0f, 1.0f,
    260.0f, 600.0f, 0.0f, 1.0f,

    310.0f, 700.0f, 0.0f, 1.0f,
    330.0f, 700.0f, 0.0f, 1.0f,
    330.0f, 600.0f, 0.0f, 1.0f,
    310.0f, 600.0f, 0.0f, 1.0f,

    280.0f, 700.0f, 0.0f, 1.0f,
    280.0f, 670.0f, 0.0f, 1.0f,
    310.0f, 600.0f, 0.0f, 1.0f,
    310.0f, 630.0f, 0.0f, 1.0f,
  };

  // culorile varfurilor din colturi
  GLfloat Colors[] = {
    0.0f, 1.0f, 0.0f, 1.0f,
    0.0f, 1.0f, 0.0f, 1.0f,
    0.0f, 1.0f, 0.0f, 1.0f,
    0.0f, 1.0f, 0.0f, 1.0f,

    0.5f, 0.5f, 0.5f, 1.0f,
```

```
399     0.5f, 0.5f, 0.5f, 1.0f,
400     0.5f, 0.5f, 0.5f, 1.0f,
401     0.5f, 0.5f, 0.5f, 1.0f,
402
403     1.0f, 1.0f, 1.0f, 1.0f,
404     1.0f, 1.0f, 1.0f, 1.0f,
405     1.0f, 1.0f, 1.0f, 1.0f,
406     1.0f, 1.0f, 1.0f, 1.0f,
407
408       1.0f, 1.0f, 1.0f, 1.0f,
409     1.0f, 1.0f, 1.0f, 1.0f,
410     1.0f, 1.0f, 1.0f, 1.0f,
411     1.0f, 1.0f, 1.0f, 1.0f,
412
413       1.0f, 1.0f, 1.0f, 1.0f,
414     1.0f, 1.0f, 1.0f, 1.0f,
415     1.0f, 1.0f, 1.0f, 1.0f,
416     1.0f, 1.0f, 1.0f, 1.0f,
417
418       1.0f, 1.0f, 1.0f, 1.0f,
419     1.0f, 1.0f, 1.0f, 1.0f,
420     1.0f, 1.0f, 1.0f, 1.0f,
421     1.0f, 1.0f, 1.0f, 1.0f,
422
423       1.0f, 1.0f, 1.0f, 1.0f,
424     1.0f, 1.0f, 1.0f, 1.0f,
425     1.0f, 1.0f, 1.0f, 1.0f,
426     1.0f, 1.0f, 1.0f, 1.0f,
427
428       1.0f, 1.0f, 1.0f, 1.0f,
429     1.0f, 1.0f, 1.0f, 1.0f,
430     1.0f, 1.0f, 1.0f, 1.0f,
431     1.0f, 1.0f, 1.0f, 1.0f,
432
433       1.0f, 1.0f, 1.0f, 1.0f,
434     1.0f, 1.0f, 1.0f, 1.0f,
435     1.0f, 1.0f, 1.0f, 1.0f,
436     1.0f, 1.0f, 1.0f, 1.0f,
437
438       1.0f, 1.0f, 1.0f, 1.0f,
439     1.0f, 1.0f, 1.0f, 1.0f,
```

```
    1.0f, 1.0f, 1.0f, 1.0f,
    1.0f, 1.0f, 1.0f, 1.0f,

    1.0f, 0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f, 1.0f,

    0.0f, 0.0f, 1.0f, 1.0f,
    0.0f, 0.0f, 1.0f, 1.0f,
    0.0f, 0.0f, 1.0f, 1.0f,
    0.0f, 0.0f, 1.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
```

```
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,

    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
    1.0f, 1.0f, 0.0f, 1.0f,
```

```
522
523       1.0f, 1.0f, 0.0f, 1.0f,
524       1.0f, 1.0f, 0.0f, 1.0f,
525       1.0f, 1.0f, 0.0f, 1.0f,
526       1.0f, 1.0f, 0.0f, 1.0f,
527
528       1.0f, 1.0f, 0.0f, 1.0f,
529       1.0f, 1.0f, 0.0f, 1.0f,
530       1.0f, 1.0f, 0.0f, 1.0f,
531       1.0f, 1.0f, 0.0f, 1.0f,
532   };
533
534   // se creeaza un buffer nou
535   glGenBuffers(1, &VboId);
536   // este setat ca buffer curent
537   glBindBuffer(GL_ARRAY_BUFFER, VboId);
538   // punctele sunt "copiate" in bufferul curent
539   glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices,
          GL_STATIC_DRAW);
540
541   // se creeaza / se leaga un VAO (Vertex Array Object) - util cand
          se utilizeaza mai multe VBO
542   glGenVertexArrays(1, &VaoId);
543   glBindVertexArray(VaoId);
544   // se activeaza lucrul cu atribute; atributul 0 = pozitie
545   glEnableVertexAttribArray(0);
546   glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 0, 0);
547
548   // un nou buffer, pentru culoare
549   glGenBuffers(1, &ColorBufferId);
550   glBindBuffer(GL_ARRAY_BUFFER, ColorBufferId);
551   glBufferData(GL_ARRAY_BUFFER, sizeof(Colors), Colors,
          GL_STATIC_DRAW);
552   // atributul 1 =  culoare
553   glEnableVertexAttribArray(1);
554   glVertexAttribPointer(1, 4, GL_FLOAT, GL_FALSE, 0, 0);
555 }
556 void DestroyVBO(void)
557 {
558   glDisableVertexAttribArray(1);
559   glDisableVertexAttribArray(0);
```

```
560   glBindBuffer(GL_ARRAY_BUFFER, 0);
561   glDeleteBuffers(1, &ColorBufferId);
562   glDeleteBuffers(1, &VboId);
563   glBindVertexArray(0);
564   glDeleteVertexArrays(1, &VaoId);
565 }
566
567 void CreateShaders(void)
568 {
569   ProgramId = LoadShaders("03_02_Shader.vert", "03_02_Shader.frag");
570   glUseProgram(ProgramId);
571 }
572 void DestroyShaders(void)
573 {
574   glDeleteProgram(ProgramId);
575 }
576
577 void Initialize(void)
578 {
579   glClearColor(1.0f, 1.0f, 1.0f, 0.0f); // culoarea de fond a
        ecranului
580   CreateVBO();
581   CreateShaders();
582   codColLocation = glGetUniformLocation(ProgramId, "codCuloare");
583   myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
584 }
585 void RenderFunction(void)
586 {
587   glClear(GL_COLOR_BUFFER_BIT);
588
589   // TO DO: schimbati transformarile (de exemplu deplasarea are loc
        pe axa Oy sau pe o alta dreapta)
590   resizeMatrix = glm::ortho(-width, width, -height, height); //
        scalam, "aducem" scena la "patratul standard" [-1,1]x[-1,1]
591   matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(i, k, 0.0))
        ; // controleaza translatia de-a lungul lui Ox
592   matrTransl3 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, h,
        0.0));
593   matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(1.0, 1.0,
        0.0)); // plaseaza patratul rosu
```

```
594   matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 1.0, 0.0))
          ; // folosita la desenarea patratului rosu
595   matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(j, l, 0.0)
          );
596   matrRot = glm::rotate(glm::mat4(1.0f), angle, glm::vec3(0.0, 0.0,
          1.0)); // rotatie folosita la deplasarea patratului rosu
597
598   // Matricea de redimensionare (pentru elementele "fixe")
599   myMatrix = resizeMatrix;
600   // Culoarea
601   codCol = 0;
602   // Transmitere variabile uniforme
603   glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
          ;
604   glUniform1i(codColLocation, codCol);
605
606   glDrawArrays(GL_POLYGON, 0, 4);
607   glDrawArrays(GL_POLYGON, 4, 4);
608   glDrawArrays(GL_POLYGON, 8, 4);
609   glDrawArrays(GL_POLYGON, 12, 4);
610   glDrawArrays(GL_POLYGON, 16, 4);
611   glDrawArrays(GL_POLYGON, 20, 4);
612   glDrawArrays(GL_POLYGON, 24, 4);
613   glDrawArrays(GL_POLYGON, 28, 4);
614   glDrawArrays(GL_POLYGON, 32, 4);
615   glDrawArrays(GL_POLYGON, 36, 4);
616   glDrawArrays(GL_POLYGON, 48, 4);
617   glDrawArrays(GL_POLYGON, 52, 4);
618   glDrawArrays(GL_POLYGON, 56, 4);
619   glDrawArrays(GL_POLYGON, 60, 4);
620   glDrawArrays(GL_POLYGON, 64, 4);
621   glDrawArrays(GL_POLYGON, 68, 4);
622   glDrawArrays(GL_POLYGON, 72, 4);
623   glDrawArrays(GL_POLYGON, 76, 4);
624   glDrawArrays(GL_POLYGON, 80, 4);
625   glDrawArrays(GL_POLYGON, 84, 4);
626   glDrawArrays(GL_POLYGON, 88, 4);
627   glDrawArrays(GL_POLYGON, 92, 4);
628   glDrawArrays(GL_POLYGON, 96, 4);
629   glDrawArrays(GL_POLYGON, 100, 4);
630   glDrawArrays(GL_POLYGON, 104, 4);
```

```
631     glDrawArrays(GL_POLYGON, 108, 4);
632
633
634
635     // Matricea pentru dreptunghiul rosu
636     myMatrix = resizeMatrix * matrTransl * matrDepl * matrScale2;
637     // Culoarea
638     codCol = 2;
639     // Transmitere variabile uniforme
640     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
            ;
641     glUniform1i(codColLocation, codCol);
642     // Apelare DrawArrays
643     glDrawArrays(GL_POLYGON, 40, 4);
644
645     // Matricea pentru dreptunghiul rosu
646     myMatrix = resizeMatrix * matrTransl2 * matrDepl * matrScale2 *
            matrRot;
647     // Culoarea
648     codCol = 1;
649     // Transmitere variabile uniforme
650     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
            ;
651     glUniform1i(codColLocation, codCol);
652     // Apelare DrawArrays
653     glDrawArrays(GL_POLYGON, 44, 4);
654
655
656     myMatrix = resizeMatrix * matrTransl3;
657     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
            ;
658     glUniform1i(codColLocation, codCol);
659     glDrawArrays(GL_POLYGON, 112, 4);
660     glDrawArrays(GL_POLYGON, 116, 4);
661     glDrawArrays(GL_POLYGON, 120, 4);
662     glDrawArrays(GL_POLYGON, 124, 4);
663     glDrawArrays(GL_POLYGON, 128, 4);
664     glDrawArrays(GL_POLYGON, 132, 4);
665     glDrawArrays(GL_POLYGON, 136, 4);
666     glDrawArrays(GL_POLYGON, 140, 4);
667     glDrawArrays(GL_POLYGON, 144, 4);
```

```
668     glDrawArrays(GL_POLYGON, 148, 4);
669     glDrawArrays(GL_POLYGON, 152, 4);
670     glDrawArrays(GL_POLYGON, 156, 4);
671     glDrawArrays(GL_POLYGON, 160, 4);
672     glDrawArrays(GL_POLYGON, 164, 4);
673     glDrawArrays(GL_POLYGON, 168, 4);
674     glDrawArrays(GL_POLYGON, 172, 4);
675     glDrawArrays(GL_POLYGON, 176, 4);
676     glDrawArrays(GL_POLYGON, 180, 4);
677     glDrawArrays(GL_POLYGON, 184, 4);
678     glDrawArrays(GL_POLYGON, 188, 4);
679     glDrawArrays(GL_POLYGON, 192, 4);
680     glDrawArrays(GL_POLYGON, 196, 4);
681     glDrawArrays(GL_POLYGON, 200, 4);
682     glutSwapBuffers();
683     glFlush();
684 }
685 void Cleanup(void)
686 {
687     DestroyShaders();
688     DestroyVBO();
689 }
690
691 int main(int argc, char* argv[])
692 {
693     glutInit(&argc, argv);
694     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
695     glutInitWindowPosition(100, 100);
696     glutInitWindowSize(1000, 700);
697     glutCreateWindow("Proiect 1 - Depasire intre 2 dreptunghiuri");
698     glewInit();
699     Initialize();
700     glutDisplayFunc(RenderFunction);
701     glutMouseFunc(mouse);
702     glutCloseFunc(Cleanup);
703     glutMainLoop();
704 }
```

Referințe

- Fișierul *03_02_animatie_new.cpp* din *Laborator 3*.
- *Materialele din Curs*.