Grafica pe calculator

Proiect 1: Depasire

Catarama Antonia, Stancu Denisa

Grupa 342

1. Conceptul proiectului

Tema acestui proiect este depasirea unei masini. Asadar, am desenat 3 masini, doua dintre ele participa la depasire si cea de-a treia este o masina pe contrasens. Am adaugat ca elemente de design faruri si roti pentru cele 3 masini, iar cele doua benzi sunt despartite printr-o linie intrerupta pentru ca depasirea sa fie una legala. Imaginea este privita de sus.

2. Transformari

2.1. Translatia

Am aplicat functia de translatie pentru deplasarea celor 3 masini, o data in matricea care controleaza translatia de-a lungul axei Ox si o data in matricea care se ocupa de deplasare.

```
matrTrans1 = glm::translate(glm::mat4(1.0f), glm::vec3(i, 0.0, 0.0)); // controleaza translatia de-a lungul lui Ox pt masina albastra
matrTrans2 = glm::translate(glm::mat4(1.0f), glm::vec3(a, b, 0.0)); // controleaza translatia de-a lungul lui Ox pt masina rosie
matrTrans3 = glm::translate(glm::mat4(1.0f), glm::vec3(-i, 300, 0.0)) ; // controleaza translatia de-a lungul lui Ox pt masina de pe contrasens
matrTrans4 = glm::translate(glm::mat4(1.0f), glm::vec3(a, b, 0.0)); //roti
matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); // plaseaza dreptunghiul rosu
matrDepl2 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); // plaseaza dreptunghiul albastru
matrDepl4 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); // plaseaza masina de pe contrasens
matrDepl5 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); //roti
```

2.2. Scalarea

Am aplicat functia de scalare atat pentru desenarea celor 3 masini, cat si pentru desenarea liniei intrerupte, pentru care am desenat mai multe dreptunghiuri.

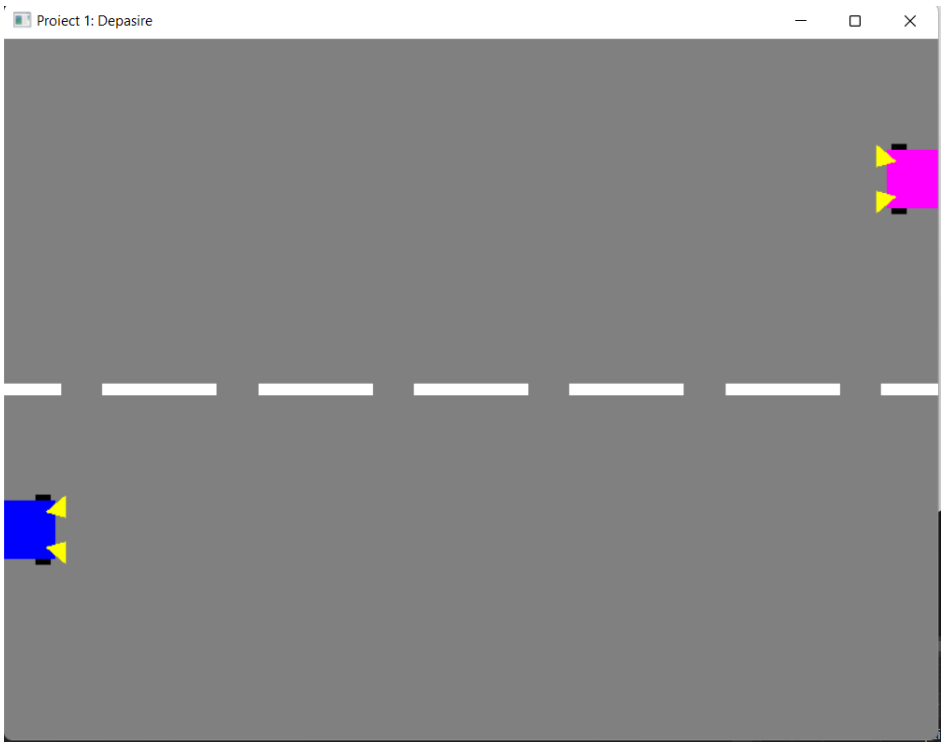
```
matrScale1 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 0.5, 0.0)); // folosita la desenarea dreptunghiului albastru
matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 0.5, 0.0)); // folosita la desenarea dreptunghiului rosu
matrScale3 = glm::scale(glm::mat4(1.0f), glm::vec3(1.1, 0.1, 0.0)); // folosita la desenarea linie
matrScale4 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 0.5, 0.0)); // folosita la desenarea dreptunghiului mov (de pe contrasens)
```

3. Originalitate

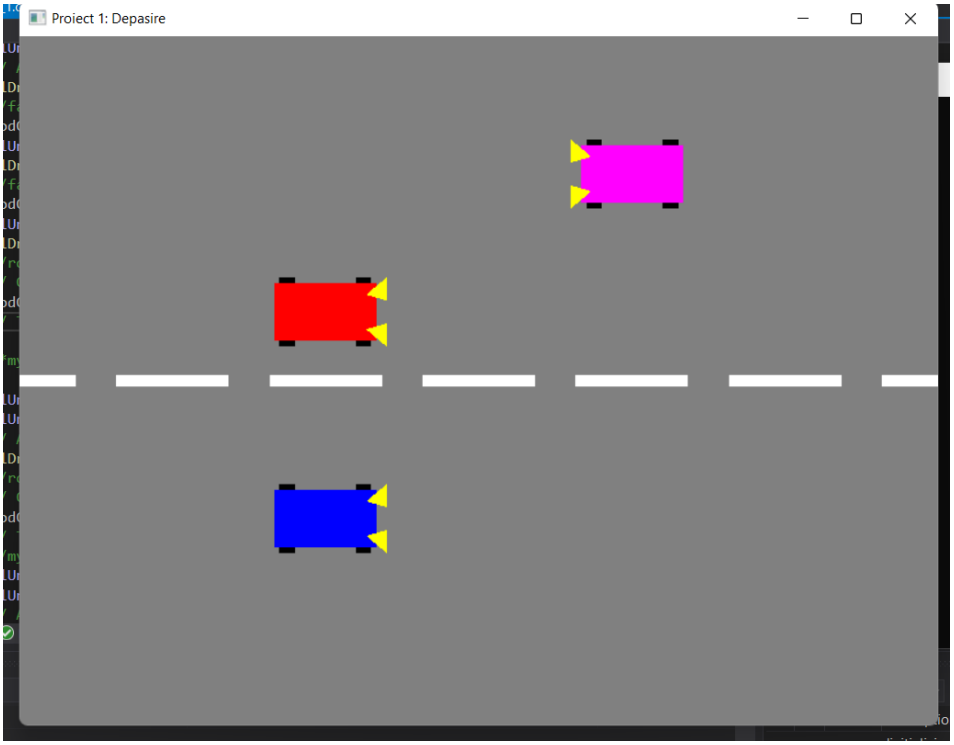
Originalitatea acestui proiect se regaseste in elemetele de design si in abordarea temei.

4. Capturi de ecran relevante

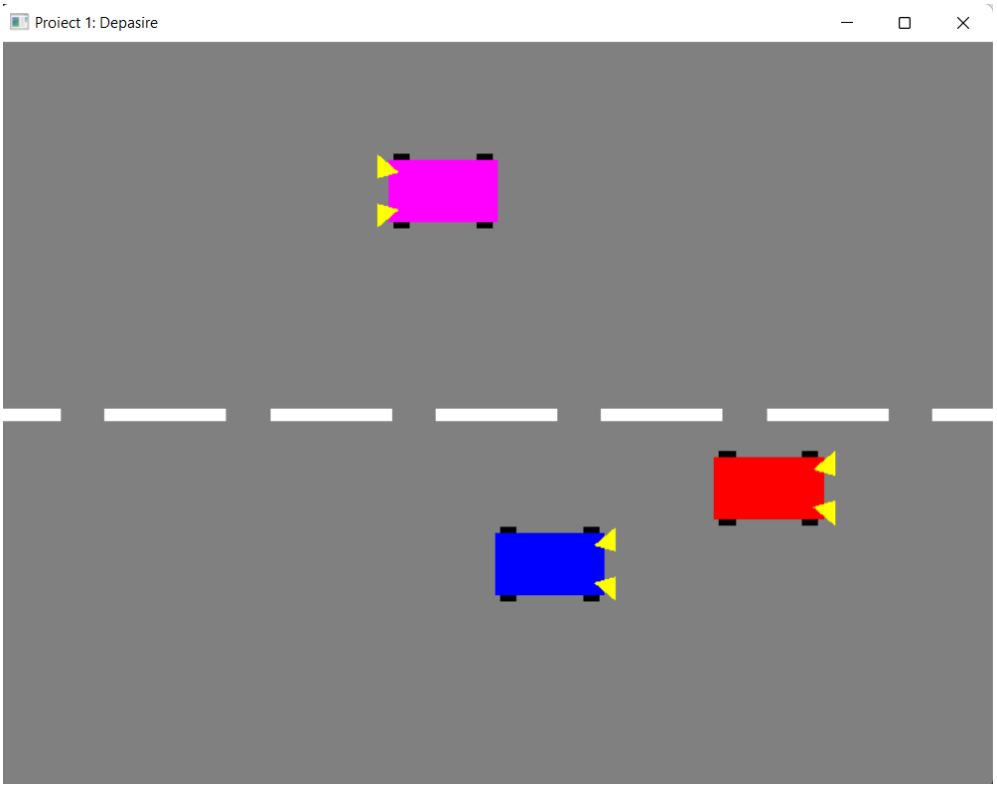
4.1. Rezultat



1.1. Masina albastra si masina mov deplasandu-se pe banda lor.



1.2. Masina rosie depaseste masina albastra.



1.3. Masina rosie revine pe banda sa.

4.2. Cod

```
for (int k = 0;k <= 10; k++)
{
    matrDepl3 = glm::translate(glm::mat4(1.0f), glm::vec3(k * 150.0 - 450, 0.0, 0.0));
    myMatrix = resizeMatrix * matrDepl3 * matrScale3;
    // Culoarea
    codCol = 3;
    // Transmitere variabile uniforme
    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
    glUniform1i(codColLocation, codCol);
    // Apelare DrawArrays
    glDrawArrays(GL_POLYGON, 4, 4);
}
```

2.1. Desenarea linie ce desparte benzile.

```
// Matricea pentru dreptunghiul albastru
myMatrix = resizeMatrix * matrTransl * matrDepl2 * matrScale1;
// Culoarea
codCol = 1;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 4, 4);
//far1
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 8, 3);
//far2
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 11, 3);
```

2.2. Desenarea masinii albastre.

```
// Matricea pentru dreptunghiul rosu
myMatrix = resizeMatrix * matrTrans2 * matrDepl * matrScale2;
// Culoarea
codCol = 2;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 4, 4);
//far1
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 8, 3);
//far2
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 11, 3);
```

2.3. Desenarea masinii rosii.

```
//masina contrasens
myMatrix = resizeMatrix * matrTrans3 * matrDepl4 * matrScale4;
// Culoarea
codCol = 4;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 4, 4);
//far1
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 14, 3);
//far2
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 17, 3);
```

2.3. Desenarea masinii mov.

```
//roti1
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 20, 4);
//roti2
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 24, 4);
//roti3
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 28, 4);
//roti4
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 32, 4);
```

2.4. Desenarea rotilor.

```
void miscas(void)
{
    i = i + step;
    a = a + 0.1;
    switch (s) {
    case 0:
    {
        if (a >= i - 750)
            b = b + 0.03;
        if (a >= i)
            s = 1;
        break;
    }
    case 1:
    {
        b = b - 0.03;
        if (b <= 0.0)
            s = 2;
        break;
    }
    case 2:
    {
        break;
    }
    }
    angle = angle + beta;
    glutPostRedisplay();
}
```

2.5. Sensul in care merg masinile, depasirea si revenirea pe banda.

5. Contributii individuale

Antonia Catarama:

- desenarea patratelor ce reprezinta masinile si a liniei dintre benzi
- deplasarea masinilor pe sensul benzii

Denisa Stancu:

- desenarea elementelor de design pentru fiecare masina
- depasirea si revenirea in banda a masinii rosii

Cod sursă

1.1 Main

```
#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;


int codCol;
float PI = 3.141592, angle = 0;
float tx = 0; float ty = 0;
float width = 450, height = 300;
float i = -450.0, j = 0.0, alpha = 0.0, step=0.05, beta = 0.0002; // i = masina albastra, a = masina rosie
float a = -750.0, b = 0.0;
int s = 0.0;
glm::mat4
	myMatrix, resizeMatrix, matrTransl, matrTrans2, matrScale1, matrScale2, matrRot, matrDepl, matrDepl2, matrScale3, matrDepl3, matrDepl4, matrTrans3, matrScale4;


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


void miscas(void)
{
	i = i + step;
	a = a + 0.1;
	switch (s) {
	case 0:
```

```
	{
		if (a >= i - 750)
			b = b + 0.03;
		if (a >= i)
			s = 1;
		break;
	}
	case 1:
	{
		b = b - 0.03;
		if (b <= 0.0)
			s = 2;
		break;
	}
	case 2:
	{
		break;
	}
	}
	angle = angle + beta;
	glutPostRedisplay();
}

void mouse(int button, int state, int x, int y)
{
	switch (button) {
	case GLUT_LEFT_BUTTON:
		if (state == GLUT_DOWN)
			alpha = -step;
		glutIdleFunc(miscas);
```

```
		break;
	default:
		break;
	}
}

void CreateVBO(void)
{
	// varfurile
	GLfloat Vertices[] = {
	// varfuri pentru axe
	-450.0f, 0.0f, 0.0f, 1.0f,
	450.0f,  0.0f, 0.0f, 1.0f,
	0.0f, -300.0f, 0.0f, 1.0f,
	0.0f, 300.0f, 0.0f, 1.0f,
	// varfuri pentru dreptunghi
	-50.0f,  -50.0f, 0.0f, 1.0f,
	50.0f, -50.0f, 0.0f, 1.0f,
	50.0f,  50.0f, 0.0f, 1.0f,
	-50.0f,  50.0f, 0.0f, 1.0f,
	//faruri masini jos
	 60.0f, 20.0f, 0.0f, 1.0f,
	 60.0f, 60.0f, 0.0f, 1.0f,
	 40.0f, 30.0f, 0.0f, 1.0f,

	 60.0f, -20.0f, 0.0f, 1.0f,
	 60.0f, -60.0f, 0.0f, 1.0f,
	 40.0f, -30.0f, 0.0f, 1.0f,
	 //faruri masina sus
	  -60.0f, -20.0f, 0.0f, 1.0f,
```

```
  -60.0f, -60.0f, 0.0f, 1.0f,
  -40.0f, -30.0f, 0.0f, 1.0f,

  -60.0f, 20.0f, 0.0f, 1.0f,
  -60.0f, 60.0f, 0.0f, 1.0f,
  -40.0f, 30.0f, 0.0f, 1.0f,
//roti1
-45.0f, 50.0f, 0.0f, 1.0f,
-30.0f, 50.0f, 0.0f, 1.0f,
-30.0f,  60.0f, 0.0f, 1.0f,
-45.0f,  60.0f, 0.0f, 1.0f,
//roti2
45.0f, -50.0f, 0.0f, 1.0f,
30.0f, -50.0f, 0.0f, 1.0f,
30.0f,  -60.0f, 0.0f, 1.0f,
45.0f,  -60.0f, 0.0f, 1.0f,
//roti3
-45.0f, -50.0f, 0.0f, 1.0f,
-30.0f, -50.0f, 0.0f, 1.0f,
-30.0f,  -60.0f, 0.0f, 1.0f,
-45.0f,  -60.0f, 0.0f, 1.0f,
//roti4
45.0f, 50.0f, 0.0f, 1.0f,
30.0f, 50.0f, 0.0f, 1.0f,
30.0f,  60.0f, 0.0f, 1.0f,
45.0f,  60.0f, 0.0f, 1.0f
};
// se creeaza un buffer nou
glGenBuffers(1, &VboId);
// este setat ca buffer curent
```

```
	glBindBuffer(GL_ARRAY_BUFFER, VboId);
	// punctele sunt "copiate" in bufferul curent
	glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices, GL_STATIC_DRAW);

	// se creeaza / se leaga un VAO (Vertex Array Object) - util cand se utilizeaza mai multe VBO
	glGenVertexArrays(1, &VaoId);
	glBindVertexArray(VaoId);
	// se activeaza lucrul cu atribute; atributul 0 = pozitie
	glEnableVertexAttribArray(0);
	glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 0, 0);

	// un nou buffer, pentru culoare
	glGenBuffers(1, &ColorBufferId);
	glBindBuffer(GL_ARRAY_BUFFER, ColorBufferId);
	// atributul 1 =  culoare
	glEnableVertexAttribArray(1);
	glVertexAttribPointer(1, 4, GL_FLOAT, GL_FALSE, 0, 0);
}

void DestroyVBO(void)
{
	glDisableVertexAttribArray(1);
	glDisableVertexAttribArray(0);
	glBindBuffer(GL_ARRAY_BUFFER, 0);
	glDeleteBuffers(1, &ColorBufferId);
	glDeleteBuffers(1, &VboId);
	glBindVertexArray(0);
	glDeleteVertexArrays(1, &VaoId);
}
```

```
void CreateShaders(void)
{
	ProgramId = LoadShaders("proiect1_Shader.vert", "proiect1_Shader.frag");
	glUseProgram(ProgramId);
}

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

void Initialize(void)
{
	glClearColor(0.5f, 0.5f, 0.5f, 0.5f); // culoarea de fond a ecranului
	CreateVBO();
	CreateShaders();
	codColLocation = glGetUniformLocation(ProgramId, "codCuloare");
	myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
}

void RenderFunction(void)
{
	glClear(GL_COLOR_BUFFER_BIT);

	resizeMatrix = glm::ortho(-width, width, -height, height); // scalam, "aducem" scena la "patratul standard" [-1,1]x[-1,1]
	matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(i, 0.0, 0.0)); // controleaza translatia de-a lungul lui Ox pt masina albastra
	matrTrans2 = glm::translate(glm::mat4(1.0f), glm::vec3(a, b, 0.0)); // controleaza translatia de-a lungul lui Ox pt masina rosie
```

```
matrTrans3 = glm::translate(glm::mat4(1.0f), glm::vec3(-i, 300, 0.0)) ; // controleaza translatia de-a lungul lui Ox pt masina de pe contrasens

matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); // plaseaza dreptunghiul rosu

matrDepl2 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); // plaseaza dreptunghiul albastru

matrDepl4 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, -120.0, 0.0)); // plaseaza masina de pe contrasens

matrScale1 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 0.5, 0.0)); // folosita la desenarea dreptunghiului albastru

matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 0.5, 0.0)); // folosita la desenarea dreptunghiului rosu

matrScale3 = glm::scale(glm::mat4(1.0f), glm::vec3(1.1, 0.1, 0.0)); // folosita la desenarea linie

matrScale4 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 0.5, 0.0)); // folosita la desenarea dreptunghiului mov (de pe contrasens)


// Matricea de redimensionare (pentru elementele "fixe")

myMatrix = resizeMatrix;


for (int k = 0;k <= 10; k++)

{

	matrDepl3 = glm::translate(glm::mat4(1.0f), glm::vec3(k * 150.0 - 450, 0.0, 0.0));

	myMatrix = resizeMatrix * matrDepl3 * matrScale3;

	// Culoarea

	codCol = 3;

	// Transmitere variabile uniforme

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

	glUniform1i(codColLocation, codCol);

	// Apelare DrawArrays

	glDrawArrays(GL_POLYGON, 4, 4);

}
```

```
// Matricea pentru dreptunghiul albastru
myMatrix = resizeMatrix * matrTransl * matrDepl2 * matrScale1;
// Culoarea
codCol = 1;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 4, 4);
//far1
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 8, 3);
//far2
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 11, 3);
//roti1
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 20, 4);
//roti2
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
```

```
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 24, 4);
//roti3
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 28, 4);
//roti4
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 32, 4);

// Matricea pentru dreptunghiul rosu
myMatrix = resizeMatrix * matrTrans2 * matrDepl * matrScale2;
// Culoarea
codCol = 2;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 4, 4);
//far1
```

```
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 8, 3);
//far2
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 11, 3);
//roti1
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 20, 4);
//roti2
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 24, 4);
//roti3
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
```

```
glDrawArrays(GL_POLYGON, 28, 4);
//roti4
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 32, 4);

//masina contrasens
myMatrix = resizeMatrix * matrTrans3 * matrDepl4 * matrScale4;
// Culoarea
codCol = 4;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 4, 4);
//far1
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 14, 3);
//far2
codCol = 5;
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_TRIANGLES, 17, 3);
//roti1
// Culoarea
codCol = 7;
```

```
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 20, 4);
//roti2
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 24, 4);
//roti3
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 28, 4);
//roti4
// Culoarea
codCol = 7;
// Transmitere variabile uniforme
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
// Apelare DrawArrays
glDrawArrays(GL_POLYGON, 32, 4);
```

```
	glutSwapBuffers();
	glFlush();
}

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

int main(int argc, char* argv[])
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
	glutInitWindowPosition(100, 100);
	glutInitWindowSize(800, 600);
	glutCreateWindow("Proiect 1: Depasire");
	glewInit();
	Initialize();
	glutDisplayFunc(RenderFunction);
	glutMouseFunc(mouse);
	glutCloseFunc(Cleanup);
	glutMainLoop();
}
```

1.2. Shader.frag

```
// Shader-ul de fragment / Fragment shader
 #version 330
```

```
in vec4 ex_Color;
uniform int codCuloare;
out vec4 out_Color;

void main(void)
{
  switch (codCuloare)
  {
        case 0:
          out_Color = ex_Color;
          break;
        case 1:
              out_Color=vec4 (0.0, 0.0, 1.0, 0.0);
              break;
        case 2:
              out_Color=vec4 (1.0, 0.0, 0.0, 0.0);
              break;
        case 3:
              out_Color=vec4 (1.0, 1.0, 1.0, 0.0);
              break;
        case 4:
              out_Color=vec4 (1.0, 0.0, 1.0, 0.0);
              break;
        case 5:
              out_Color=vec4 (1.0, 1.0, 0.0, 0.0);
              break;
        case 6:
              out_Color=vec4 (0.0, 1.0, 0.0, 0.0);
              break;
        case 7:
```

```
                out_Color=vec4 (0.0, 0.0, 0.0, 0.0);
                break;
        default:
                break;
  };
}
```

1.3. Shader.vert

```
// Shader-ul de varfuri
#version 330

layout (location = 0) in vec4 in_Position;
layout (location = 1) in vec4 in_Color;

out vec4 gl_Position;
out vec4 ex_Color;
uniform mat4 myMatrix;

void main(void)
{
   gl_Position = myMatrix*in_Position;
   ex_Color = in_Color;
}
```