Proiect 2D: Depasire

Studenti: grupa 341

Ionescu Radu
Ion Melania
Flutur Angelica
Constantin Ioana

1. Enunt:

   Simulati o "depasire": o masina / un dreptunghi se deplaseaza uniform (prin translatie), un alt dreptunghi vine din spate (tot prin translatii/rotatii), la un moment dat intra in depasire, apoi trece in fata primului.

2. Conceptul proiectului:

   Proiectul curent presupune reprezentarea unei scene bidimensionale ce reface momentul unei depasiri dintre 2 masini pe autostrada. Primul obiect se misca uniform de-a lungul unei traiectorii, in timp ce al doilea obiect se misca din spate catre fata, urmand ca intre 2 puncte fixe setate pe axa Oy acesta sa realizeze o depasire, prin translatie.

3. Pentru a realiza sceneta am utilizat exclusive translatii.

4. Originalitatea proiectului consta atat in vizualizarea diferita a scenei, in utilizarea diverselor texturi, cat si in redarea unei situatii desprinse din viata reala: masina care realizeaza depasirea este chiar ambulanta, surprinsa intr-o misiune de salvare.

5. Portiuni de interes din codul sursa:

   1) Alegerea coordonatelor in plan:

```
//masini
-70.0f, -581.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   0.0f, 0.0f,  // Stanga jos;
-20.0f, -581.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   1.0f, 0.0f,  // Dreapta jos;
-20.0f, -480.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   1.0f, 1.0f,  // Dreapta sus;
-70.0f, -480.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   0.0f, 1.0f,  // Stanga sus;

-70.0f, -811.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   0.0f, 0.0f,  // Stanga jos;
-20.0f, -811.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   1.0f, 0.0f,  // Dreapta jos;
-20.0f, -710.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   1.0f, 1.0f,  // Dreapta sus;
-70.0f, -710.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   0.0f, 1.0f,  // Stanga sus;
```

```
//varfuri pentru banci
-230.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f, // Stanga jos;
-190.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f, // Dreapta jos;
-190.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f, // Dreapta sus;
-230.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

-230.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,  // Stanga jos;
-190.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,  // Dreapta jos;
-190.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,  // Dreapta sus;
-230.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

230.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,  // Stanga jos;
190.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,  // Dreapta jos;
190.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,  // Dreapta sus;
230.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

230.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,   // Stanga jos;
190.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,   // Dreapta jos;
190.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,   // Dreapta sus;
230.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;
```

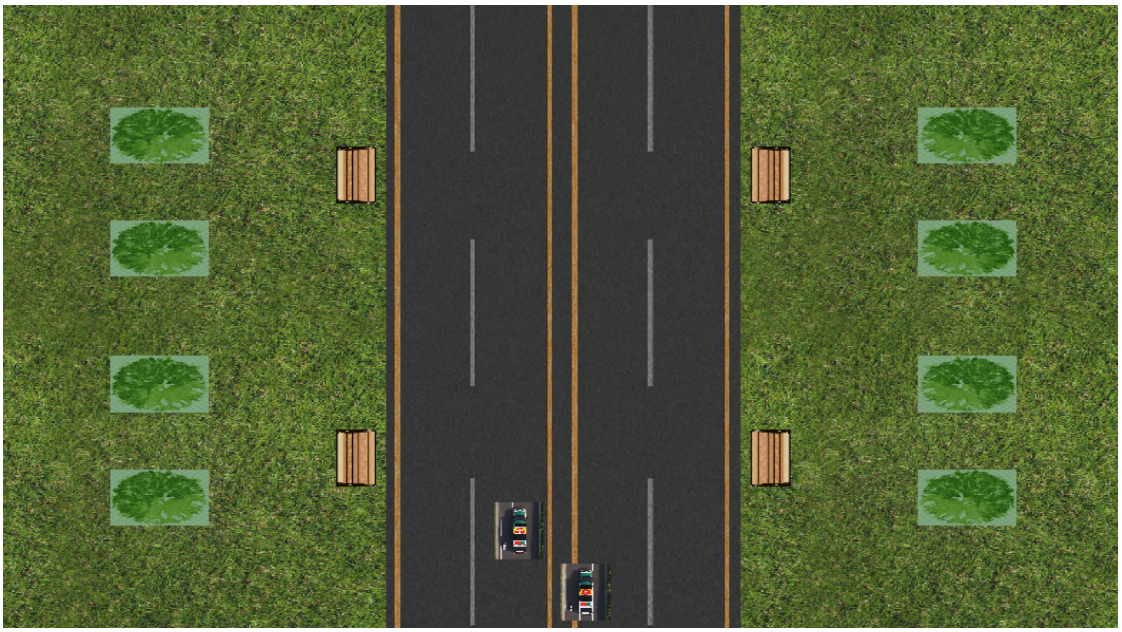
2) Realizarea animatiei: deplasarea masinilor pentru a reface momentul depasirii

```
void MoveUp(void)
{
    j = j + step;

    if (j > 1500)
        j = 0;

    j2 = j2 + step2;

    if (j2 > 1500)
        j2 = 0;

    if (j2 > 50 && j2 < 220)
    {
        i = i + step3;

    }
    if (j2 >= 260 && j2 < 430)
    {
        i = i - step3;

    }
    glutPostRedisplay();    //  Forteza redesenarea scenei;
}
```

3) Aplicarea unor texturi, pentru un plus de originalitate

```
//desenare masini
//masina care se deplaseaza in linie dreapta
LoadTexture("ambulanta_buna.png");
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
myMatrix = resizeMatrix * matrTransl3;
codCol = 1;
glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
glUniform1i(codColLocation, codCol);
glDrawArrays(GL_POLYGON, 68, 4);
```

4) Rulare



6. Contributii individuale:

Miscarea tuturor obiectelor:

- Intreaga echipa

Design drum:

- Ion Melania

Design Masina:

- Intreaga echipa

Plante + banci

- Flutur Angelica + Ionescu Radu

Documentatia:

- Intreaga echipa

7. Coduri sursa:

```
#include <windows.h>        //      Utilizarea functiilor de sistem Windows (crearea
de ferestre, manipularea fisierelor si directoarelor);
#include <stdlib.h>         //  Biblioteci necesare pentru citirea shaderelor;
#include <stdio.h>
#include <GL/glew.h>        //  Defineşte prototipurile functiilor OpenGL si
constantele necesare pentru programarea OpenGL moderna;
#include <GL/freeglut.h>    //      Include functii pentru:
                                                                //          - gestionarea
ferestrelor si evenimentelor de tastatura si mouse,
                                                                //  - desenarea de
primitive grafice precum dreptunghiuri, cercuri sau linii,
                                                                //  - crearea de meniuri si
submeniuri;
#include "loadShaders.h"            //          Fisierul care face legatura intre program si
shadere;
#include "glm/glm.hpp"              //          Bibloteci utilizate pentru transformari
grafice;
#include "glm/gtc/matrix_transform.hpp"
#include "glm/gtx/transform.hpp"
#include "glm/gtc/type_ptr.hpp"
#include "SOIL.h"                               //          Biblioteca pentru texturare;
//  Identificatorii obiectelor de tip OpenGL;
```

```
GLuint
VaoId,
VboId,
EboId,
ProgramId,
viewLocation,
projLocation,
myMatrixLocation,
codColLocation,
texture;
//		Dimensiunile ferestrei de afisare;
GLfloat
winWidth = 800, winHeight = 600;
//		Variabile catre matricile de transformare;
glm::mat4
myMatrix, resizeMatrix, matrTransl, matrTransl2, matrTranslDiag, matrTransl3,
matrScale, matrScale1, matrScale2, matrRot, matrDepl;

//		Variabile pentru proiectia ortogonala;
float xMin = -580, xMax = 580.f, yMin = -560.f, yMax = 560.f;

// ADAUGATE DIN TEXTURARE
float width = 800, height = 600, zNear = 0, zFar = 1000, fov = 45;
int codCol;
float PI = 3.141592;
float obsX = 0.0, obsY = 0.0, obsZ = 800.f;
float refX = 0.0f, refY = 0.0f;
float vX = 0.0;
glm::mat4
projection, view;

//		Variabile pentru deplasarea pe axa Ox si pentru rotatie;
float i = 0.0, j = 0, j2 = 0, step = 30.4, step2 = 15.2, beta = 0.064, step3 = 9.6;

//  Crearea si compilarea obiectelor de tip shader;
//		Trebuie sa fie in acelasi director cu proiectul actual;
//  Shaderul de varfuri / Vertex shader - afecteaza geometria scenei;
//  Shaderul de fragment / Fragment shader - afecteaza culoarea pixelilor;
void CreateShaders(void)
{
		ProgramId = LoadShaders("Shader.vert", "Shader.frag");
		glUseProgram(ProgramId);
}

void MoveUp(void)
{
		j = j + step;
```

```
		if (j > 1500)
			j = 0;

		j2 = j2 + step2;

		if (j2 > 1500)
			j2 = 0;

		if (j2 > 50 && j2 < 220)
		{
			i = i + step3;

		}
		if (j2 >= 260 && j2 < 430)
		{
			i = i - step3;

		}
		glutPostRedisplay();	//	Forteza redesenarea scenei;
}


void LoadTexture(const char* photoPath)
{
		glGenTextures(1, &texture);
		glBindTexture(GL_TEXTURE_2D, texture);
		// Desfasurarea imaginii pe orizonatala/verticala in functie de
parametrii de texturare;
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_CLAMP);
		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(photoPath, &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);
}
```

```
//  Se initializeaza un Vertex Buffer Object (VBO) pentru tranferul datelor spre memoria placii grafice (spre shadere);
//  In acesta se stocheaza date despre varfuri (coordonate, culori, indici, texturare etc.);
void CreateVBO(void)
{
		//  Coordonatele varfurilor;
		static const GLfloat Vertices[] =
		{  // COORDONATE                                                    # CULORI
		     # COORDONATE TEXTURARE
		   -450.0f, 0.0f,  0.0f,  1.0f,   1.0f, 0.0f, 0.0f,  0.0f, 0.0f,          // Stanga jos;
		    450.0f,  0.0f,  0.0f,  1.0f,   0.0f, 1.0f, 0.0f,  1.0f, 0.0f,          // Dreapta jos;
		    0.0f, -300.0f,  0.0f,  1.0f,   1.0f, 1.0f, 0.0f, 1.0f, 1.0f,          // Dreapta sus;
		    0.0f, 300.0f,  0.0f,  1.0f,   0.0f, 1.0f, 1.0f,  0.0f, 1.0f,  // Stanga sus;

		   // //model masina 1
		   // -90.0f,  -590.0f, 0.0f, 1.0f,   1.0f, 1.0f, 0.0f,        1.0f, 1.0f,  // Dreapta sus;
		   // -40.0f,  -590.0f, 0.0f, 1.0f,   0.0f, 1.0f, 0.0f,        1.0f, 0.0f,  // Dreapta jos;
		   //  -20.0f,   -590.0f, 0.0f, 1.0f,     1.0f, 0.0f, 0.0f,     0.0f, 0.0f,  // Stanga jos;
		   //  -20.0f,   -550.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,      0.0f, 1.0f,  // Stanga sus;
		   // -40.0f,  -530.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  // Stanga sus;
		   // -90.0f,  -530.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  // Stanga sus;
		   // //roata 1
		   // -85.0f, -597.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  // Stanga sus;
		   // -85.0f, -582.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  // Stanga sus;
		   //-70.0f, -582.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  // Stanga sus;
		   //-70.0f, -597.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  // Stanga sus;
		   // //roata2
		   //-35.0f, -597.0f, 0.0f, 1.0f,     0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  // Stanga sus;
		   //-35.0f, -582.0f, 0.0f, 1.0f,     0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  // Stanga sus;
		   //-50.0f, -582.0f, 0.0f, 1.0f,    1.0f, 0.0f, 1.0f, 1.0f,  1.0f,  // Stanga sus;
```

```
                    //   -50.0f, -597.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,          0.0f, 1.0f,  //
Stanga sus;



                    // //model masina 2
                    //-20.0f,  -360.0f, 0.0f, 1.0f,   0.0f, 1.0f, 0.0f,          1.0f, 0.0f,
// Dreapta jos;
                    //-70.0f,  -360.0f, 0.0f, 1.0f,   1.0f, 1.0f, 0.0f,          1.0f, 1.0f,
// Dreapta sus;
                    //-90.0f,   -360.0f, 0.0f, 1.0f,     1.0f, 0.0f, 0.0f,       0.0f, 0.0f,
// Stanga jos;
                    //-90.0f,   -320.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  //
Stanga sus;
                    //-70.0f,  -300.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,          0.0f, 1.0f,  //
Stanga sus;
                    //-20.0f,  -300.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,          0.0f, 1.0f,  //
Stanga sus;
                    // //roata 1
                    // -85.0f, -367.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,          0.0f, 1.0f,  //
Stanga sus;
                    // -85.0f, -352.0f, 0.0f, 1.0f,   0.0f, 1.0f, 1.0f,          0.0f, 1.0f,  //
Stanga sus;
                    //-70.0f, -352.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  //
Stanga sus;
                    //-70.0f, -367.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  //
Stanga sus;
                    ////roata 2
                    //-35.0f, -367.0f, 0.0f, 1.0f,     0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  //
Stanga sus;
                    //-35.0f, -352.0f, 0.0f, 1.0f,     0.0f, 1.0f, 1.0f,        0.0f, 1.0f,  //
Stanga sus;
                    //-50.0f, -352.0f, 0.0f, 1.0f,    1.0f, 0.0f, 1.0f, 1.0f,  1.0f,  // Stanga
sus;
                    //-50.0f, -367.0f, 0.0f, 1.0f,    0.0f, 1.0f, 1.0f,         0.0f, 1.0f,  //
Stanga sus;



                     // copaci
                     -460.0f,  -270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,       0.0f, 0.0f,
// Stanga jos;
                     -360.0f, -270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 0.0f,
// Dreapta jos;
                     -360.0f,  -370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,       1.0f, 1.0f,
// Dreapta sus;
```

```
                -460.0f,  -370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,       0.0f, 1.0f,  //
Stanga sus;

                -460.0f,  -70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 0.0f,
// Stanga jos;
                -360.0f, -70.0f,  0.0f,  1.0f,    0.0f, 0.3f, 0.0f,          1.0f, 0.0f,
// Dreapta jos;
                -360.0f,  -170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,       1.0f, 1.0f,
// Dreapta sus;
                -460.0f,  -170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,       0.0f, 1.0f,  //
Stanga sus;

                -460.0f,  170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 0.0f,
// Stanga jos;
                -360.0f, 170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,         1.0f, 0.0f,
// Dreapta jos;
                -360.0f,  70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,         1.0f, 1.0f,
// Dreapta sus;
                -460.0f,  70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,         0.0f, 1.0f,  //
Stanga sus;

                -460.0f,  370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 0.0f,
// Stanga jos;
                -360.0f, 370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,         1.0f, 0.0f,
// Dreapta jos;
                -360.0f,  270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 1.0f,
// Dreapta sus;
                -460.0f,  270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 1.0f,  //
Stanga sus;



                //copaci textura 2
                460.0f,  -270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 0.0f,
// Stanga jos;
                360.0f, -270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,          1.0f, 0.0f,
// Dreapta jos;
                360.0f,  -370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 1.0f,
// Dreapta sus;
                460.0f,  -370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 1.0f,  //
Stanga sus;

                460.0f,  -70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,          0.0f, 0.0f,
// Stanga jos;
                360.0f, -70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,           1.0f, 0.0f,
// Dreapta jos;
                360.0f,  -170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 1.0f,
// Dreapta sus;
```

```
                460.0f,  -170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 1.0f,  //
Stanga sus;

                460.0f,  170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 0.0f,
// Stanga jos;
                360.0f, 170.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 0.0f,
// Dreapta jos;
                360.0f,  70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 1.0f,
// Dreapta sus;
                460.0f,  70.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 1.0f,  //
Stanga sus;

                460.0f,  370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 0.0f,
// Stanga jos;
                360.0f, 370.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        1.0f, 0.0f,
// Dreapta jos;
                360.0f,  270.0f,  0.0f,  1.0f,  0.0f, 0.3f, 0.0f,        1.0f, 1.0f,
// Dreapta sus;
                460.0f,  270.0f,  0.0f,  1.0f,   0.0f, 0.3f, 0.0f,        0.0f, 1.0f,  //
Stanga sus;



                //Varfuri pentru cele 2 benzi
                -180.0f, -561.0f, 0.0f, 1.0f,     0.5f,0.5f,0.5f,     0.0f, 0.0f,    // Stanga
jos;
                0.0f, -561.0f, 0.0f, 1.0f,                    0.5f,0.5f,0.5f,
1.0f, 0.0f, // Dreapta jos;
                0.0f, 561.0f, 0.0f, 1.0f,                     0.5f,0.5f,0.5f,
1.0f, 1.0f,       // Dreapta sus;
                -180.0f, 561.0f, 0.0f, 1.0f,                  0.5f,0.5f,0.5f,
0.0f, 1.0f,  // Stanga sus;

                0.0f, -561.0f, 0.0f, 1.0f,                    0.5f,0.5f,0.5f,
0.0f, 0.0f,       // Stanga jos;
                180.0f, -561.0f, 0.0f, 1.0f,                  0.5f,0.5f,0.5f,
1.0f, 0.0f, // Dreapta jos;
                180.0f, 561.0f, 0.0f, 1.0f,                   0.5f,0.5f,0.5f,
1.0f, 1.0f,       // Dreapta sus
                0.0f, 561.0f, 0.0f, 1.0f,                     0.5f,0.5f,0.5f,
0.0f, 1.0f,  // Stanga sus;

                //varfuri pentru iarba
                        //xMin = -580, yMin = -560.f, xMax = 580.f, , yMax = 560.f;
                -580.0f, -560.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 0.0f, 0.0f,          // Stanga
jos;
                -179.0f, -560.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // Dreapta jos;
                -179.0f, 600.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f, 1.0f,          //
Dreapta sus;
```

```
                    -580.0f, 600.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 0.0f, 1.0f,  // Stanga sus;

                    180.0f, -560.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 0.0f, 0.0f,          // Stanga
jos;
                    580.0f, -560.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f, 0.0f, // Dreapta jos;
                    580.0f, 600.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 1.0f, 1.0f,           //
Dreapta sus;
                    180.0f, 600.0f, 0.0f, 1.0f, 0.5f, 0.5f, 0.5f, 0.0f, 1.0f,  // Stanga sus;

                    //varfuri pentru banci
                    -230.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,         // Stanga
jos;
                    -190.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,         //
Dreapta jos;
                    -190.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,         //
Dreapta sus;
                    -230.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

                    -230.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,          // Stanga
jos;
                    -190.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,          //
Dreapta jos;
                    -190.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,          //
Dreapta sus;
                    -230.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

                    230.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,          // Stanga
jos;
                    190.0f, -200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,          //
Dreapta jos;
                    190.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,          //
Dreapta sus;
                    230.0f, -300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

                    230.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 0.0f,           // Stanga
jos;
                    190.0f, 300.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 0.0f,           //
Dreapta jos;
                    190.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 1.0f, 1.0f,           //
Dreapta sus;
                    230.0f, 200.0f, 0.0f, 1.0f, 0.4f, 0.3f, 0.0f, 0.0f, 1.0f,  // Stanga sus;

                    //masini
                    -70.0f, -581.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   0.0f, 0.0f,          // Stanga
jos;
                    -20.0f, -581.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   1.0f, 0.0f,          //
Dreapta jos;
```

```
                    -20.0f, -480.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   1.0f, 1.0f,        //
Dreapta sus;
                    -70.0f, -480.0f, 0.0f, 1.0f,   1.0f, 0.0f, 0.0f,   0.0f, 1.0f,  // Stanga sus;

                    -70.0f, -811.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   0.0f, 0.0f,        // Stanga
jos;
                    -20.0f, -811.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   1.0f, 0.0f,        //
Dreapta jos;
                    -20.0f, -710.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   1.0f, 1.0f,        //
Dreapta sus;
                    -70.0f, -710.0f, 0.0f, 1.0f,   0.0f, 0.0f, 1.0f,   0.0f, 1.0f,  // Stanga sus;

                };

                // Indicii care determina ordinea de parcurgere a varfurilor;
                static const GLuint Indices[] =
                {
                    0, 1, 2, 3, 0, 2
                };

                //  Transmiterea datelor prin buffere;

                //  Se creeaza / se leaga un VAO (Vertex Array Object) - util cand se
utilizeaza mai multe VBO;
                glGenVertexArrays(1, &VaoId);                                   //
Generarea VAO si indexarea acestuia catre variabila VaoId;
                glBindVertexArray(VaoId);

                //  Se creeaza un buffer pentru VARFURI - COORDONATE, CULORI si
TEXTURARE;
                glGenBuffers(1, &VboId);
                                                                //  Generarea bufferului si
indexarea acestuia catre variabila VboId;
                glBindBuffer(GL_ARRAY_BUFFER, VboId);
                                                                //  Setarea tipului de buffer -
atributele varfurilor;
                glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices,
GL_STATIC_DRAW);

                // Se creeaza un buffer pentru INDICI;
                glGenBuffers(1, &EboId);
                                                                //  Generarea
bufferului si indexarea acestuia catre variabila EboId;
                glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EboId);
                glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(Indices), Indices,
GL_STATIC_DRAW);

                // Se activeaza lucrul cu atribute;
```

```
		//  Se asociaza atributul (0 = coordonate) pentru shader;
		glEnableVertexAttribArray(0);
		glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 9 * sizeof(GLfloat),
(GLvoid*)0);
		//  Se asociaza atributul (1 =  culoare) pentru shader;
		glEnableVertexAttribArray(1);
		glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 9 * sizeof(GLfloat),
(GLvoid*)(4 * sizeof(GLfloat)));
		//  Se asociaza atributul (2 =  texturare) pentru shader;
		glEnableVertexAttribArray(2);
		glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, 9 * sizeof(GLfloat),
(GLvoid*)(7 * sizeof(GLfloat)));
}

//  Elimina obiectele de tip shader dupa rulare;
void DestroyShaders(void)
{
		glDeleteProgram(ProgramId);
}

//  Eliminarea obiectelor de tip VBO dupa rulare;
void DestroyVBO(void)
{
		//  Eliberarea atributelor din shadere (pozitie, culoare, texturare
etc.);
		glDisableVertexAttribArray(2);
		glDisableVertexAttribArray(1);
		glDisableVertexAttribArray(0);

		//  Stergerea bufferelor pentru VARFURI(Coordonate + Culori),
INDICI;
		glBindBuffer(GL_ARRAY_BUFFER, 0);
		glDeleteBuffers(1, &VboId);
		glDeleteBuffers(1, &EboId);

		//  Eliberaea obiectelor de tip VAO;
		glBindVertexArray(0);
		glDeleteVertexArrays(1, &VaoId);
}

//  Functia de eliberare a resurselor alocate de program;
void Cleanup(void)
{
		DestroyShaders();
		DestroyVBO();
}

//  Setarea parametrilor necesari pentru fereastra de vizualizare;
```

```
void Initialize(void)
{
	glClearColor(1.0f, 1.0f, 1.0f, 1.0f);		//  Culoarea de fond a ecranului;
	//CreateVBO();
	//  Trecerea datelor de randare spre bufferul folosit de shadere;
	CreateShaders();
//  Initilizarea shaderelor;
	// Instantierea variabilelor uniforme pentru a "comunica" cu shaderele;
	myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
	codColLocation = glGetUniformLocation(ProgramId, "codCol");
	matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(100.f, 100.f, 0.0));
	matrScale = glm::scale(glm::mat4(1.0f), glm::vec3(2.0f, 0.5f, 0.0));


	// Dreptunghiul "decupat";
	resizeMatrix = glm::ortho(xMin, xMax, yMin, yMax);
	viewLocation = glGetUniformLocation(ProgramId, "view");
	projLocation = glGetUniformLocation(ProgramId, "projection");
	glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
}

//  Functia de desenarea a graficii pe ecran;
void RenderFunction(void)
{
	glClear(GL_COLOR_BUFFER_BIT);			//  Se curata ecranul OpenGL pentru a fi desenat noul continut;
	// Transmiterea variabilei uniforme pentru MATRICEA DE TRANSFORMARE spre shadere;


	//tipurile de miscari
	matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(i, 0.0, 0.0));
	// Se translateaza de-a lungul axei Ox;
	matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, j, 0.0));
	   //	Se translateaza de-a lungul axei Ox;
	matrTranslDiag = glm::translate(glm::mat4(1.0f), glm::vec3(i, j, 0.0));

	matrTransl3 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, j2, 0.0));
	matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(90.0, 200.0, 0.0));	   //	Se translateaza patratul ROSU fata de patratul ALBASTRU;
	matrScale1 = glm::scale(glm::mat4(1.0f), glm::vec3(1.1, 0.3, 0.0));
	   //	Se scaleaza coordonatele initiale si se obtine dreptunghiul ALABSTRU;
```

```
                matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(0.25, 0.25, 0.0));
                   //        Se scaleaza coordonatele initiale si se obtine patratul
ROSU;

                CreateVBO();
                // Incarcarea texturii si legarea acesteia cu shaderul;
                //desenare iarba+textura
                codCol = 5;
                LoadTexture("iarba.png");
                glActiveTexture(GL_TEXTURE0);
                glBindTexture(GL_TEXTURE_2D, texture);
                glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);

                myMatrix = resizeMatrix;

                glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);
                glUniform1i(codColLocation, codCol);
                glDrawArrays(GL_POLYGON, 44, 4);
                glDrawArrays(GL_POLYGON, 48, 4);

                codCol = 6;
                LoadTexture("banca.png");
                glActiveTexture(GL_TEXTURE0);
                glBindTexture(GL_TEXTURE_2D, texture);
                glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);

                myMatrix = resizeMatrix;

                glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);
                glUniform1i(codColLocation, codCol);
                glDrawArrays(GL_POLYGON, 52, 4);
                glDrawArrays(GL_POLYGON, 56, 4);
                glDrawArrays(GL_POLYGON, 60, 4);
                glDrawArrays(GL_POLYGON, 64, 4);

                LoadTexture("tree.png");
                glActiveTexture(GL_TEXTURE0);
                glBindTexture(GL_TEXTURE_2D, texture);
                glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
                ////triunghi3
                myMatrix = resizeMatrix;
                codCol = 3;
                glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);
                glUniform1i(codColLocation, codCol);
                //glDrawArrays(GL_POLYGON, 28, 4);
```

```
                    glDrawArrays(GL_POLYGON, 4, 4);
                    glDrawArrays(GL_POLYGON, 8, 4);
                    glDrawArrays(GL_POLYGON, 12, 4);
                    glDrawArrays(GL_POLYGON, 16, 4);
                    glDrawArrays(GL_POLYGON, 20, 4);
                    glDrawArrays(GL_POLYGON, 24, 4);
                    glDrawArrays(GL_POLYGON, 28, 4);
                    glDrawArrays(GL_POLYGON, 32, 4);




                    //desenarea benzilor + textura
                    codCol = 4;
                    glUniform1i(codColLocation, codCol);
                    LoadTexture("road-texture.png");
                    glActiveTexture(GL_TEXTURE0);
                    glBindTexture(GL_TEXTURE_2D, texture);
                    glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
                    glDrawArrays(GL_POLYGON, 36, 4);
                    glDrawArrays(GL_POLYGON, 40, 4);



                    //desenare masini
                    //masina care se deplaseaza in linie dreapta
                    LoadTexture("ambulanta_buna.png");
                    glActiveTexture(GL_TEXTURE0);
                    glBindTexture(GL_TEXTURE_2D, texture);
                    glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
                    myMatrix = resizeMatrix * matrTransl3;
                    codCol = 1;
                    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);
                    glUniform1i(codColLocation, codCol);
                    glDrawArrays(GL_POLYGON, 68, 4);



                    //masina care depaseste
                    LoadTexture("ambulanta_buna.png");
                    glActiveTexture(GL_TEXTURE0);
                    glBindTexture(GL_TEXTURE_2D, texture);
                    glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
                    myMatrix = resizeMatrix * matrTranslDiag;
                    codCol = 2;
                    glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE,
&myMatrix[0][0]);
                    glUniform1i(codColLocation, codCol);
```

```
		CreateVBO();
		glDrawArrays(GL_POLYGON, 72, 4);

		MoveUp();

		glutSwapBuffers();//	Inlocuieste imaginea deseneata in fereastra cu cea randata;
		glFlush();
//  Asigura rularea tuturor comenzilor OpenGL apelate anterior;
}

//		Punctul de intrare in program, se ruleaza rutina OpenGL;
int main(int argc, char* argv[])
{
		//  Se initializeaza GLUT si contextul OpenGL si se configureaza fereastra si modul de afisare;

		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
		   //	Se folosesc 2 buffere (unul pentru afisare si unul pentru randare => animatii cursive) si culori RGB;

		glutInitWindowSize(winWidth, winHeight);
		   //  Dimensiunea ferestrei;
		glutInitWindowPosition(100, 100);
			//  Pozitia initiala a ferestrei;
		glutCreateWindow("Scena principala depasire- OpenGL <<nou>>");
		// Creeaza fereastra de vizualizare, indicand numele acesteia;

		// Se initializeaza GLEW si se verifica suportul de extensii OpenGL modern disponibile pe sistemul gazda;
		//  Trebuie initializat inainte de desenare;

		glewInit();

		Initialize();		//  Setarea parametrilor necesari pentru fereastra de vizualizare;
		glutDisplayFunc(RenderFunction);  //  Desenarea scenei in fereastra;
		glutIdleFunc(RenderFunction);		//	Asigura rularea continua a randarii;
		glutCloseFunc(Cleanup);		//  Eliberarea resurselor alocate de program;

		//  Bucla principala de procesare a evenimentelor GLUT (functiile care incep cu glut: glutInit etc.) este pornita;
		//  Prelucreaza evenimentele si deseneaza fereastra OpenGL pana cand utilizatorul o inchide;
```

```
                        glutMainLoop();

                        return 0;
        }
```

```
//
//

#version 330 core

//  Variabile de intrare (dinspre programul principal);
layout (location = 0) in vec4 in_Position;     //  Se preia din buffer de pe prima pozitie (0) atributul
care contine coordonatele;
layout (location = 1) in vec4 in_Color;        //  Se preia din buffer de pe a doua pozitie (1)
atributul care contine culoarea;
layout (location=2) in vec2 texCoord;          //  Se preia din buffer de pe a treia pozitie (2)
atributul care contine textura;

//  Variabile de iesire;
//out vec4 gl_Position;   //  Transmite pozitia actualizata spre programul principal;
out vec4 ex_Color;      //  Transmite culoarea (de modificat in Shader.frag);
out vec2 tex_Coord;     //  Transmite textura (de modificat in Shader.frag);

//  Variabile uniforme;
uniform mat4 myMatrix;
uniform mat4 view;
uniform mat4 projection;

void main(void)
{
   gl_Position = myMatrix*in_Position;
          ex_Color=in_Color;
   tex_Coord = vec2(texCoord.x, 1-texCoord.y);
}
```

```
#version 330 core
//          Variabile de intrare (dinspre Shader.vert);
in vec4 ex_Color;
in vec2 tex_Coord;                  //          Coordonata de texturare;

//          Variabile de iesire          (spre programul principal);
out vec4 out_Color;                 //          Culoarea actualizata;

//  Variabile uniforme;
uniform sampler2D myTexture;
uniform int codCol;
```

```
void main(void)
{
  switch (codCol)
  {
            case 0:
              out_Color = ex_Color;
              break;
            case 1:
                        //out_Color=vec4(1.0,0.0,0.0,1.0); //red
                        out_Color = texture(myTexture, tex_Coord);
                        break;
            case 2:
                        out_Color = texture(myTexture, tex_Coord);
                        break;
            case 3:
                        out_Color = mix(texture(myTexture, tex_Coord), ex_Color, 0.5);        //
Amestecarea texturii si a culorii;
                        break;
            case 4:
                         out_Color = mix(texture(myTexture, tex_Coord), ex_Color, 0.2);       // textura
benzi
                         break;
            case 5:
                         out_Color = mix(texture(myTexture, tex_Coord), ex_Color, 0.2);       // textura
iarba
                         break;
            case 6:
                        out_Color = mix(texture(myTexture, tex_Coord), ex_Color, 0.2);        // textura
banca
                        break;
            default:
                        out_Color = ex_Color;
                        break;
  };
}
```