Sasu Alexandru-Cristian
Grupa 342

Depășire cu accident

I. Conceptul proiectului

O mașină de culoare bej circulă pe o bandă a unei șosele. La un moment dat, o mașină albastră care circulă pe aceeași bandă ca și mașina bej, o ajunge din urmă pe aceasta, iar după puțin timp de mers în spatele mașinii bej, cea din urmă este depășită de mașina albastră. În timp ce mașina albastră realizează depășirea, pe cealaltă bandă sosește un camion, care lovește mașina albastra, producându-se un accident.

II. Transformările incluse

Înmulțirea matricelor a fost realizată în programul principal, transformările aferente înmulțirilor fiind următoarele:

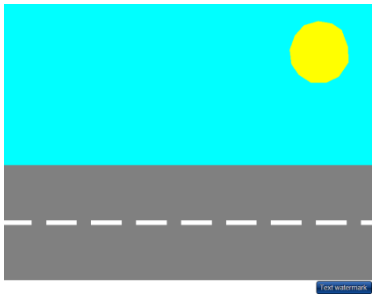
a. **glm::ortho** – pentru: desenarea scenei aplicației, cu o anumită lățime și înălțime.
b. **glm::translate** – pentru: poziționarea separatoarelor de benzi pe centrul șoselei, și deplasarea acestora în partea stângă a ecranului, elementele care aparțin soarelui (soarele, ochii, gura), poziționarea și deplasarea tuturor vehiculelor, poziționarea și deplasarea roților vehiculelor.
c. **glm::scale** – pentru: redimensionarea separatoarelor de benzi, redimensionarea soarelui, redimensionarea mașinii albastre, redimensionarea camionului, redimensionarea roților vehiculelor.
d. **glm::rotate** – pentru: rotația mașinii albastre atunci când realizează depășirea.

III. Originalitatea proiectului

Proiectul este original datorită faptului că prezintă scenariul unei depășiri atipice: în loc ca depășirea să se realizeze cu succes, aceasta este încheiată printr-un accident. Un alt element de originalitate îl reprezintă apariția feței soarelui odată ce are loc accidentul.

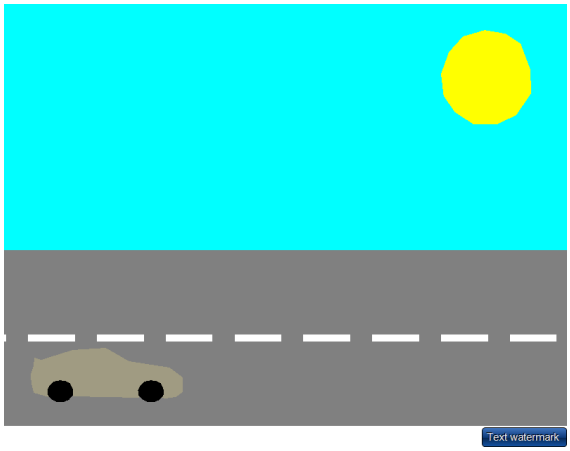
IV. Prezentarea proiectului

Starea inițială (șoseaua este goală):

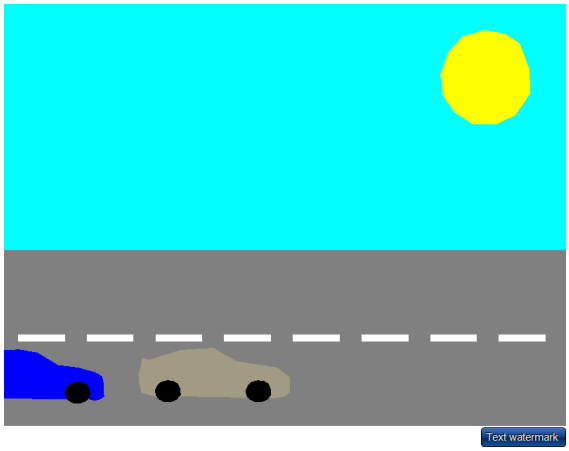


Sasu Alexandru-Cristian
Grupa 342

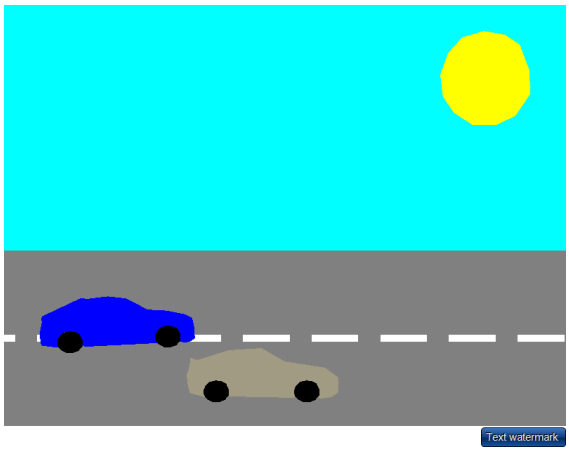
Apare mașina bej, care se deplasează cu o viteză constantă pe banda de jos:



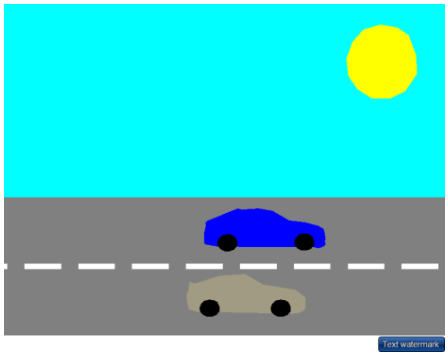
Apare mașina albastră, care se deplasează în spatele mașinii bej, cu o viteză constantă:



Mașina albastră încearcă să depășească mașina bej:

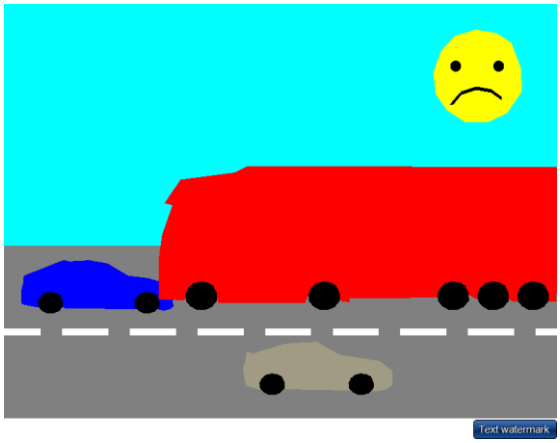


Mașina albastră este mai în față decât mașina bej:

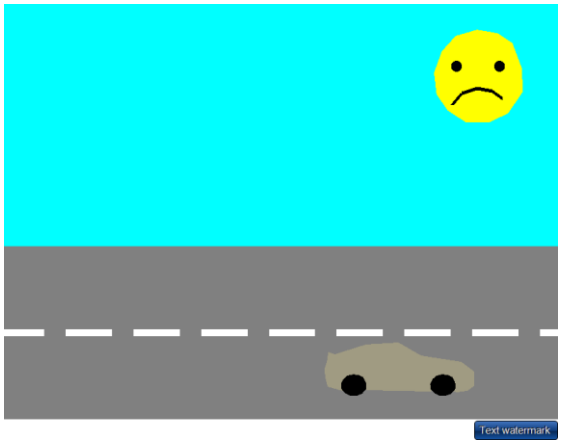


Sasu Alexandru-Cristian
Grupa 342

Mașina albastră este lovită de camion și soarele se întristează:



Mașina bej își continuă drumul, în mod normal:



Porțiunea de cod care controlează separatoarele de benzi:

```
laneDiv1X += laneDivSpeedX;
laneDiv2X += laneDivSpeedX;
laneDiv3X += laneDivSpeedX;
laneDiv4X += laneDivSpeedX;
laneDiv5X += laneDivSpeedX;
laneDiv6X += laneDivSpeedX;
laneDiv7X += laneDivSpeedX;
laneDiv8X += laneDivSpeedX;
laneDiv9X += laneDivSpeedX;
laneDiv10X += laneDivSpeedX;

if (laneDiv1X <= -(width + 100))
    laneDiv1X = width + 100;
if (laneDiv2X <= -(width + 100))
    laneDiv2X = width + 100;
if (laneDiv3X <= -(width + 100))
    laneDiv3X = width + 100;
if (laneDiv4X <= -(width + 100))
    laneDiv4X = width + 100;
if (laneDiv5X <= -(width + 100))
    laneDiv5X = width + 100;
if (laneDiv6X <= -(width + 100))
    laneDiv6X = width + 100;
if (laneDiv7X <= -(width + 100))
    laneDiv7X = width + 100;
if (laneDiv8X <= -(width + 100))
    laneDiv8X = width + 100;
if (laneDiv9X <= -(width + 100))
    laneDiv9X = width + 100;
if (laneDiv10X <= -(width + 100))
    laneDiv10X = width + 100;
```

Sasu Alexandru-Cristian
Grupa 342

Porțiunea de cod care controleaza mașina bej:

```
beigeCarX += beigeCarSpeedX;
```


Porțiunea de cod care controleaza mașina albastră:

```
if (beigeCarX >= -385.0 && blueCarX <= beigeCarX - 280.0) // The blue car is right behind the beige car, and it is following it
    blueCarX += blueCarSpeedX;

if (-275.0 + moveBlueCarDiagonally < -155.0) // The blue car is driving on the right lane (below the lane dividers)
{
    if (blueCarX >= -495.0)
        if (blueCarAngle < PI / 10)
        {
            blueCarAngle += alpha;
            moveBlueCarDiagonally += moveBlueCarDiagonallySpeed;
        }
}
else // The blue car is driving on the left lane (above the lane dividers)
{
    if (blueCarAngle >= 0)
        blueCarAngle -= alpha;

    if (truckX - blueCarX <= 300) // When the blue car collides with the truck
        blueCarX += (truckSpeedX - 0.1);
    else // The blue car has not yet collided with the truck
        blueCarX += blueCarSpeedX;
}
```


Porțiunea de cod care controlează camionul:

```
truckX += truckSpeedX;
```


Porțiunea de cod aferentă transformării matricelor:

```
void generateTransformationMatrices(void)
{
    resizeMatrix = glm::ortho(-width, width, -height, height);

    matrScaleSun = glm::scale(glm::mat4(1.0f), glm::vec3(1.4, 1.3, 0.0));
    matrTranslSun = glm::translate(glm::mat4(1.0f), glm::vec3(180.0, 90.0, 0.0));
    matrTranslSunLeftEye = glm::translate(glm::mat4(1.0f), glm::vec3(285.0, 210.0, 0.0));
    matrTranslSunRightEye = glm::translate(glm::mat4(1.0f), glm::vec3(355.0, 210.0, 0.0));
    matrTranslSunMouth = glm::translate(glm::mat4(1.0f), glm::vec3(220.0, 45.0, 0.0));

    matrScaleLaneDiv = glm::scale(glm::mat4(1.0f), glm::vec3(0.75, 0.1, 0.0));
    matrTranslLaneDiv1 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv1X, -175.0, 0.0));
    matrTranslLaneDiv2 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv2X, -175.0, 0.0));
    matrTranslLaneDiv3 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv3X, -175.0, 0.0));
    matrTranslLaneDiv4 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv4X, -175.0, 0.0));
    matrTranslLaneDiv5 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv5X, -175.0, 0.0));
    matrTranslLaneDiv6 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv6X, -175.0, 0.0));
    matrTranslLaneDiv7 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv7X, -175.0, 0.0));
    matrTranslLaneDiv8 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv8X, -175.0, 0.0));
    matrTranslLaneDiv9 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv9X, -175.0, 0.0));
    matrTranslLaneDiv10 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv10X, -175.0, 0.0));

    matrScaleCarWheel = glm::scale(glm::mat4(1.0f), glm::vec3(0.4, 0.3, 0.0));

    matrTranslBeigeCar = glm::translate(glm::mat4(1.0f), glm::vec3(beigeCarX, -315.0, 0.0));
    matrTranslBeigeCarBackWheel = glm::translate(glm::mat4(1.0f), glm::vec3(beigeCarX + 30.0, -275.0, 0.0));
    matrTranslBeigeCarFrontWheel = glm::translate(glm::mat4(1.0f), glm::vec3(beigeCarX + 175.0, -275.0, 0.0));

    matrScaleBlueCar = glm::scale(glm::mat4(1.0f), glm::vec3(0.17, 0.17, 0.0));
    matrTranslBlueCar = glm::translate(glm::mat4(1.0f), glm::vec3(blueCarX + moveBlueCarDiagonally, -275.0 + moveBlueCarDiagonally, 0.0));
    matrTranslBlueCarBackWheel = glm::translate(glm::mat4(1.0f), glm::vec3(blueCarX + 8.0 - blueCarAngle * 30 + moveBlueCarDiagonally, -277.0 + blueCarAngle * 45 + moveBlueCarDiagonally, 0.0));
    matrTranslBlueCarFrontWheel = glm::translate(glm::mat4(1.0f), glm::vec3(blueCarX + 165.0 - blueCarAngle * 35 + moveBlueCarDiagonally, -277.0 + blueCarAngle * 185 + moveBlueCarDiagonally, 0.0));
    matrRotBlueCar = glm::rotate(glm::mat4(1.0f), blueCarAngle, glm::vec3(0.0, 0.0, 1.0));

    matrScaleTruckWheel = glm::scale(glm::mat4(1.0f), glm::vec3(0.5, 0.4, 0.0));

    matrScaleTruck = glm::scale(glm::mat4(1.0f), glm::vec3(0.5, 0.5, 0.0));
    matrTranslTruck = glm::translate(glm::mat4(1.0f), glm::vec3(truckX, truckY, 0.0));
    matrTranslTruckFrontWheel1 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 75.0, -155.0, 0.0));
    matrTranslTruckFrontWheel2 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 275.0, -155.0, 0.0));
    matrTranslTruckBackWheel1 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 485.0, -155.0, 0.0));
    matrTranslTruckBackWheel2 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 550.0, -155.0, 0.0));
    matrTranslTruckBackWheel3 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 615.0, -155.0, 0.0));
}
```

Sasu Alexandru-Cristian
Grupa 342

Resursele utilizate:

Site pentru modelarea mașinilor:
https://www.mobilefish.com/services/record_mouse_coordinates/record_mouse_coordinates.php?fbclid=IwAR0bL0CBIOxTMKrkuomuOqofe8PTPpUMLC17LUZU9KJTrvptmN5JgKBVXUQ

Cod din laboratoarele trecute

Sasu Alexandru-Cristian
Grupa 342

Anexe

Codul sursă:

```
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <GL/glew.h>
#include <GL/freeglut.h>
#include "loadShaders.h"

#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,
colCodLocation;
```

Sasu Alexandru-Cristian
Grupa 342

```
int codCol;

float PI = 3.141592, blueCarAngle = 0, alpha = 0.00005;

float width = 450, height = 300;

float laneDiv1X = -440.0, laneDiv2X = -330.0, laneDiv3X = -220.0, laneDiv4X = -110.0, laneDiv5X =
0.0, laneDiv6X = 110.0, laneDiv7X = 220.0, laneDiv8X = 330.0, laneDiv9X = 440.0, laneDiv10X =
550.0,

	beigeCarX = -750.0,

	blueCarX = -750.0,

	truckX = 50000.0, truckY = -235.0;

float laneDivSpeedX = -0.5,

	beigeCarSpeedX = 0.03,

	blueCarSpeedX = 0.15, moveBlueCarDiagonally = 0.0, moveBlueCarDiagonallySpeed = 0.06,

	truckSpeedX = -2.2;



glm::mat4

myMatrix, resizeMatrix,

matrScaleSun, matrTranslSun, matrTranslSunLeftEye, matrTranslSunRightEye, matrTranslSunMouth,

matrScaleLaneDiv, matrTranslLaneDiv1, matrTranslLaneDiv2, matrTranslLaneDiv3, matrTranslLaneDiv4,
matrTranslLaneDiv5, matrTranslLaneDiv6, matrTranslLaneDiv7, matrTranslLaneDiv8,
matrTranslLaneDiv9, matrTranslLaneDiv10,

matrScaleCarWheel,

matrTranslBeigeCar, matrTranslBeigeCarBackWheel, matrTranslBeigeCarFrontWheel,

matrScaleBlueCar, matrTranslBlueCar, matrTranslBlueCarBackWheel, matrTranslBlueCarFrontWheel,
matrRotBlueCar,

matrScaleTruckWheel,

matrScaleTruck, matrTranslTruck, matrTranslTruckFrontWheel1, matrTranslTruckFrontWheel2,
matrTranslTruckBackWheel1, matrTranslTruckBackWheel2, matrTranslTruckBackWheel3;



void generateTransformationMatrices(void)

{

	resizeMatrix = glm::ortho(-width, width, -height, height);



	matrScaleSun = glm::scale(glm::mat4(1.0f), glm::vec3(1.4, 1.3, 0.0));

	matrTranslSun = glm::translate(glm::mat4(1.0f), glm::vec3(180.0, 90.0, 0.0));

	matrTranslSunLeftEye = glm::translate(glm::mat4(1.0f), glm::vec3(285.0, 210.0, 0.0));

	matrTranslSunRightEye = glm::translate(glm::mat4(1.0f), glm::vec3(355.0, 210.0, 0.0));
```

```
	matrTranslSunMouth = glm::translate(glm::mat4(1.0f), glm::vec3(220.0, 45.0, 0.0));


	matrScaleLaneDiv = glm::scale(glm::mat4(1.0f), glm::vec3(0.75, 0.1, 0.0));

	matrTranslLaneDiv1 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv1X, -175.0, 0.0));

	matrTranslLaneDiv2 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv2X, -175.0, 0.0));

	matrTranslLaneDiv3 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv3X, -175.0, 0.0));

	matrTranslLaneDiv4 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv4X, -175.0, 0.0));

	matrTranslLaneDiv5 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv5X, -175.0, 0.0));

	matrTranslLaneDiv6 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv6X, -175.0, 0.0));

	matrTranslLaneDiv7 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv7X, -175.0, 0.0));

	matrTranslLaneDiv8 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv8X, -175.0, 0.0));

	matrTranslLaneDiv9 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv9X, -175.0, 0.0));

	matrTranslLaneDiv10 = glm::translate(glm::mat4(1.0f), glm::vec3(laneDiv10X, -175.0, 0.0));


	matrScaleCarWheel = glm::scale(glm::mat4(1.0f), glm::vec3(0.4, 0.3, 0.0));


	matrTranslBeigeCar = glm::translate(glm::mat4(1.0f), glm::vec3(beigeCarX, -315.0, 0.0));

	matrTranslBeigeCarBackWheel = glm::translate(glm::mat4(1.0f), glm::vec3(beigeCarX + 30.0,
-275.0, 0.0));

	matrTranslBeigeCarFrontWheel = glm::translate(glm::mat4(1.0f), glm::vec3(beigeCarX +
175.0, -275.0, 0.0));


	matrScaleBlueCar = glm::scale(glm::mat4(1.0f), glm::vec3(0.17, 0.17, 0.0));

	matrTranslBlueCar = glm::translate(glm::mat4(1.0f), glm::vec3(blueCarX +
moveBlueCarDiagonally, -275.0 + moveBlueCarDiagonally, 0.0));

	matrTranslBlueCarBackWheel = glm::translate(glm::mat4(1.0f), glm::vec3(blueCarX + 8.0 -
blueCarAngle * 30 + moveBlueCarDiagonally, -277.0 + blueCarAngle * 45 + moveBlueCarDiagonally,
0.0));

	matrTranslBlueCarFrontWheel = glm::translate(glm::mat4(1.0f), glm::vec3(blueCarX + 165.0 -
blueCarAngle * 35 + moveBlueCarDiagonally, -277.0 + blueCarAngle * 185 + moveBlueCarDiagonally,
0.0));

	matrRotBlueCar = glm::rotate(glm::mat4(1.0f), blueCarAngle, glm::vec3(0.0, 0.0, 1.0));


	matrScaleTruckWheel = glm::scale(glm::mat4(1.0f), glm::vec3(0.5, 0.4, 0.0));


	matrScaleTruck = glm::scale(glm::mat4(1.0f), glm::vec3(0.5, 0.5, 0.0));
```

```
	matrTranslTruck = glm::translate(glm::mat4(1.0f), glm::vec3(truckX, truckY, 0.0));

	matrTranslTruckFrontWheel1 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 75.0, -
155.0, 0.0));

	matrTranslTruckFrontWheel2 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 275.0, -
155.0, 0.0));

	matrTranslTruckBackWheel1 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 485.0, -
155.0, 0.0));

	matrTranslTruckBackWheel2 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 550.0, -
155.0, 0.0));

	matrTranslTruckBackWheel3 = glm::translate(glm::mat4(1.0f), glm::vec3(truckX + 615.0, -
155.0, 0.0));

}



void move(void)

{

	laneDiv1X += laneDivSpeedX;

	laneDiv2X += laneDivSpeedX;

	laneDiv3X += laneDivSpeedX;

	laneDiv4X += laneDivSpeedX;

	laneDiv5X += laneDivSpeedX;

	laneDiv6X += laneDivSpeedX;

	laneDiv7X += laneDivSpeedX;

	laneDiv8X += laneDivSpeedX;

	laneDiv9X += laneDivSpeedX;

	laneDiv10X += laneDivSpeedX;


	if (laneDiv1X <= -(width + 100))

		laneDiv1X = width + 100;

	if (laneDiv2X <= -(width + 100))

		laneDiv2X = width + 100;

	if (laneDiv3X <= -(width + 100))

		laneDiv3X = width + 100;

	if (laneDiv4X <= -(width + 100))

		laneDiv4X = width + 100;

	if (laneDiv5X <= -(width + 100))
```

```
            laneDiv5X = width + 100;
      if (laneDiv6X <= -(width + 100))
            laneDiv6X = width + 100;
      if (laneDiv7X <= -(width + 100))
            laneDiv7X = width + 100;
      if (laneDiv8X <= -(width + 100))
            laneDiv8X = width + 100;
      if (laneDiv9X <= -(width + 100))
            laneDiv9X = width + 100;
      if (laneDiv10X <= -(width + 100))
            laneDiv10X = width + 100;


      beigeCarX += beigeCarSpeedX;


      if (beigeCarX >= -385.0 && blueCarX <= beigeCarX - 280.0) // The blue car is right behind
the beige car, and it is following it
            blueCarX += blueCarSpeedX;


      if (-275.0 + moveBlueCarDiagonally < -155.0) // The blue car is driving on the right lane
(below the lane dividers)
      {
            if (blueCarX >= -495.0)
                  if (blueCarAngle < PI / 10)
                  {
                        blueCarAngle += alpha;
                        moveBlueCarDiagonally += moveBlueCarDiagonallySpeed;
                  }
      }
      else // The blue car is driving on the left lane (above the lane dividers)
      {
            if (blueCarAngle >= 0)
                  blueCarAngle -= alpha;


            if (truckX - blueCarX <= 300) // When the blue car collides with the truck
```

```
                    blueCarX += (truckSpeedX - 0.1);
             else // The blue car has not yet collided with the truck
                    blueCarX += blueCarSpeedX;
      }

      truckX += truckSpeedX;

      glutPostRedisplay();
}

void mouse(int button, int state, int x, int y)
{
      if (state == GLUT_DOWN)
             glutIdleFunc(move);
}

void CreateVBO(void)
{
      GLfloat Vertices[] = {
             // Vertices for the lane dividers
             -50.0f,  -50.0f, 0.0f, 1.0f, // Vertex 0
             50.0f, -50.0f, 0.0f, 1.0f,
             50.0f,  50.0f, 0.0f, 1.0f,
             -50.0f,  50.0f, 0.0f, 1.0f,

             // Vertices for the road
             -450.0f, -50.0f, 0.0f, 1.0f, // Vertex 4
             450.0f, -50.0f, 0.0f, 1.0f,
             450.0f, -300.0f, 0.0f, 1.0f,
             -450.0f, -300.0f, 0.0f, 1.0f,

             // Vertices for the beige car
             40.0f, 109.0f, 0.0f, 1.0f, // Vertex 8
```

```
            29.0f, 112.0f, 0.0f, 1.0f,
            27.0f, 99.0f, 0.0f, 1.0f,
            23.0f, 88.0f, 0.0f, 1.0f,
            24.0f, 76.0f, 0.0f, 1.0f,
            28.0f, 62.0f, 0.0f, 1.0f,
            52.0f, 56.0f, 0.0f, 1.0f,
            52.0f, 72.0f, 0.0f, 1.0f,
            60.0f, 70.0f, 0.0f, 1.0f,
            76.0f, 89.0f, 0.0f, 1.0f,
            88.0f, 79.0f, 0.0f, 1.0f,
            93.0f, 67.0f, 0.0f, 1.0f,
            91.0f, 57.0f, 0.0f, 1.0f,
            196.0f, 55.0f, 0.0f, 1.0f,
            196.0f, 71.0f, 0.0f, 1.0f,
            205.0f, 85.0f, 0.0f, 1.0f,
            222.0f, 89.0f, 0.0f, 1.0f,
            237.0f, 81.0f, 0.0f, 1.0f,
            240.0f, 66.0f, 0.0f, 1.0f,
            238.0f, 54.0f, 0.0f, 1.0f,
            255.0f, 56.0f, 0.0f, 1.0f,
            266.0f, 63.0f, 0.0f, 1.0f,
            266.0f, 84.0f, 0.0f, 1.0f,
            245.0f, 98.0f, 0.0f, 1.0f,
            180.0f, 107.0f, 0.0f, 1.0f,
            143.0f, 126.0f, 0.0f, 1.0f,
            90.0f, 123.0f, 0.0f, 1.0f,

            // Vertices for the cars' wheels and the sun
            140.0f, 118.0f, 0.0f, 1.0f, // Vertex 35
            123.0f, 129.0f, 0.0f, 1.0f,
            99.0f, 133.0f, 0.0f, 1.0f,
            74.0f, 126.0f, 0.0f, 1.0f,
            58.0f, 109.0f, 0.0f, 1.0f,
```

```
            49.0f, 85.0f, 0.0f, 1.0f,
            52.0f, 61.0f, 0.0f, 1.0f,
            65.0f, 43.0f, 0.0f, 1.0f,
            86.0f, 30.0f, 0.0f, 1.0f,
            113.0f, 30.0f, 0.0f, 1.0f,
            135.0f, 40.0f, 0.0f, 1.0f,
            152.0f, 64.0f, 0.0f, 1.0f,
            151.0f, 90.0f, 0.0f, 1.0f,

            // Vertex for the sun's eyes
            0.0f, 0.0f, 0.0f, 1.0f, // Vertex 48

            // Vertices for the blue car
            40.0f, 365.0f, 0.0f, 1.0f, // Vertex 49
            21.0f, 346.0f, 0.0f, 1.0f,
            28.0f, 313.0f, 0.0f, 1.0f,
            26.0f, 272.0f, 0.0f, 1.0f,
            2.0f, 224.0f, 0.0f, 1.0f,
            2.0f, 178.0f, 0.0f, 1.0f,
            9.0f, 118.0f, 0.0f, 1.0f,
            165.0f, 91.0f, 0.0f, 1.0f,
            180.0f, 140.0f, 0.0f, 1.0f,
            204.0f, 197.0f, 0.0f, 1.0f,
            257.0f, 228.0f, 0.0f, 1.0f,
            326.0f, 228.0f, 0.0f, 1.0f,
            379.0f, 190.0f, 0.0f, 1.0f,
            415.0f, 135.0f, 0.0f, 1.0f,
            420.0f, 84.0f, 0.0f, 1.0f,
            1122.0f, 75.0f, 0.0f, 1.0f,
            1134.0f, 137.0f, 0.0f, 1.0f,
            1155.0f, 185.0f, 0.0f, 1.0f,
            1213.0f, 219.0f, 0.0f, 1.0f,
            1300.0f, 219.0f, 0.0f, 1.0f,
```

```
            1360.0f, 183.0f, 0.0f, 1.0f,
            1374.0f, 125.0f, 0.0f, 1.0f,
            1369.0f, 63.0f, 0.0f, 1.0f,
            1425.0f, 75.0f, 0.0f, 1.0f,
            1461.0f, 99.0f, 0.0f, 1.0f,
            1456.0f, 130.0f, 0.0f, 1.0f,
            1458.0f, 183.0f, 0.0f, 1.0f,
            1451.0f, 228.0f, 0.0f, 1.0f,
            1441.0f, 267.0f, 0.0f, 1.0f,
            1377.0f, 301.0f, 0.0f, 1.0f,
            1213.0f, 341.0f, 0.0f, 1.0f,
            1028.0f, 361.0f, 0.0f, 1.0f,
            829.0f, 466.0f, 0.0f, 1.0f,
            658.0f, 493.0f, 0.0f, 1.0f,
            461.0f, 483.0f, 0.0f, 1.0f,
            410.0f, 495.0f, 0.0f, 1.0f,
            384.0f, 464.0f, 0.0f, 1.0f,
            271.0f, 430.0f, 0.0f, 1.0f,
            177.0f, 382.0f, 0.0f, 1.0f,

            // Vertices for the truck
            361.0f, 582.0f, 0.0f, 1.0f, // Vertex 88
            322.0f, 571.0f, 0.0f, 1.0f,
            183.0f, 561.0f, 0.0f, 1.0f,
            131.0f, 493.0f, 0.0f, 1.0f,
            157.0f, 487.0f, 0.0f, 1.0f,
            123.0f, 401.0f, 0.0f, 1.0f,
            112.0f, 330.0f, 0.0f, 1.0f,
            112.0f, 215.0f, 0.0f, 1.0f,

            194.0f, 212.0f, 0.0f, 1.0f,
            224.0f, 257.0f, 0.0f, 1.0f,
            230.0f, 263.0f, 0.0f, 1.0f,
```

```
            283.0f, 260.0f, 0.0f, 1.0f,
            294.0f, 246.0f, 0.0f, 1.0f,
            306.0f, 204.0f, 0.0f, 1.0f,

            588.0f, 204.0f, 0.0f, 1.0f,
            610.0f, 249.0f, 0.0f, 1.0f,
            629.0f, 255.0f, 0.0f, 1.0f,
            674.0f, 270.0f, 0.0f, 1.0f,
            700.0f, 236.0f, 0.0f, 1.0f,
            708.0f, 209.0f, 0.0f, 1.0f,

            734.0f, 206.0f, 0.0f, 1.0f,
            734.0f, 217.0f, 0.0f, 1.0f,
            1030.0f, 220.0f, 0.0f, 1.0f,

            1051.0f, 220.0f, 0.0f, 1.0f,
            1054.0f, 249.0f, 0.0f, 1.0f,
            1078.0f, 272.0f, 0.0f, 1.0f,
            1114.0f, 272.0f, 0.0f, 1.0f,
            1148.0f, 249.0f, 0.0f, 1.0f,
            1148.0f, 215.0f, 0.0f, 1.0f,

            1169.0f, 215.0f, 0.0f, 1.0f,
            1177.0f, 249.0f, 0.0f, 1.0f,
            1206.0f, 272.0f, 0.0f, 1.0f,
            1248.0f, 264.0f, 0.0f, 1.0f,
            1269.0f, 228.0f, 0.0f, 1.0f,
            1269.0f, 207.0f, 0.0f, 1.0f,

            1285.0f, 207.0f, 0.0f, 1.0f,
            1295.0f, 249.0f, 0.0f, 1.0f,
            1327.0f, 278.0f, 0.0f, 1.0f,
            1366.0f, 270.0f, 0.0f, 1.0f,
```

```
        1384.0f, 238.0f, 0.0f, 1.0f,
        1384.0f, 207.0f, 0.0f, 1.0f,

        1405.0f, 207.0f, 0.0f, 1.0f,
        1408.0f, 238.0f, 0.0f, 1.0f,
        1463.0f, 236.0f, 0.0f, 1.0f,
        1471.0f, 598.0f, 0.0f, 1.0f,
        401.0f, 600.0f, 0.0f, 1.0f,
        403.0f, 288.0f, 0.0f, 1.0f,
        592.0f, 288.0f, 0.0f, 1.0f,
        592.0f, 275.0f, 0.0f, 1.0f,
        359.0f, 272.0f, 0.0f, 1.0f,
        388.0f, 304.0f, 0.0f, 1.0f,
        377.0f, 584.0f, 0.0f, 1.0f
    };

    GLfloat Colors[] = {
      1.0f, 0.0f, 0.0f, 1.0f,
      0.0f, 1.0f, 0.0f, 1.0f,
      0.0f, 0.0f, 1.0f, 1.0f,
      1.0f, 0.0f, 0.0f, 1.0f,
    };

    glGenBuffers(1, &VboId);
    glBindBuffer(GL_ARRAY_BUFFER, VboId);
    glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices, GL_STATIC_DRAW);

    glGenVertexArrays(1, &VaoId);
    glBindVertexArray(VaoId);
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 0, 0);

    glGenBuffers(1, &ColorBufferId);
```

```
	glBindBuffer(GL_ARRAY_BUFFER, ColorBufferId);
	glBufferData(GL_ARRAY_BUFFER, sizeof(Colors), Colors, GL_STATIC_DRAW);
	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("proiect_1_Shader.vert", "proiect_1_Shader.frag");
	glUseProgram(ProgramId);
}
void DestroyShaders(void)
{
	glDeleteProgram(ProgramId);
}

void Initialize(void)
{
	glClearColor(0.0f, 1.0f, 1.0f, 0.0f);
	CreateVBO();
	CreateShaders();
	colCodLocation = glGetUniformLocation(ProgramId, "colorCode");
	myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
```

Sasu Alexandru-Cristian
Grupa 342
}

```
void RenderFunction(void)
{
	glClear(GL_COLOR_BUFFER_BIT);

	generateTransformationMatrices();

	// Setting up the sun
	myMatrix = resizeMatrix * matrTranslSun * matrScaleSun;
	codCol = 1;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glUniform1i(colCodLocation, codCol);
	glDrawArrays(GL_POLYGON, 35, 13);

	// Setting up the road
	myMatrix = resizeMatrix;
	codCol = 2;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glUniform1i(colCodLocation, codCol);
	glDrawArrays(GL_QUADS, 4, 4);

	// Setting up the lane dividers
	codCol = 3;
	glUniform1i(colCodLocation, codCol);
	myMatrix = resizeMatrix * matrTranslLaneDiv1 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv2 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv3 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
```

```
	myMatrix = resizeMatrix * matrTranslLaneDiv4 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv5 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv6 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv7 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv8 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv9 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);
	myMatrix = resizeMatrix * matrTranslLaneDiv10 * matrScaleLaneDiv;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 0, 4);

	// Setting up the beige car
	codCol = 5;
	glUniform1i(colCodLocation, codCol);
	myMatrix = resizeMatrix * matrTranslBeigeCar;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 8, 27);

	// Setting up the beige car's wheels
	codCol = 4;
	glUniform1i(colCodLocation, codCol);
	myMatrix = resizeMatrix * matrTranslBeigeCarBackWheel * matrScaleCarWheel;
```

```
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);
	myMatrix = resizeMatrix * matrTranslBeigeCarFrontWheel * matrScaleCarWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);

	// Setting up the blue car
	codCol = 6;
	glUniform1i(colCodLocation, codCol);
	myMatrix = resizeMatrix * matrTranslBlueCar * matrRotBlueCar * matrScaleBlueCar;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 49, 39);

	// Setting up the blue car's wheels
	codCol = 4;
	glUniform1i(colCodLocation, codCol);
	myMatrix = resizeMatrix * matrTranslBlueCarBackWheel * matrScaleCarWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);
	myMatrix = resizeMatrix * matrTranslBlueCarFrontWheel * matrScaleCarWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);

	// Setting up the truck
	codCol = 7;
	glUniform1i(colCodLocation, codCol);
	myMatrix = resizeMatrix * matrTranslTruck * matrScaleTruck;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 88, 52);

	// Setting up the truck's wheels
	codCol = 4;
	glUniform1i(colCodLocation, codCol);
```

```
	myMatrix = resizeMatrix * matrTranslTruckFrontWheel1 * matrScaleTruckWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);
	myMatrix = resizeMatrix * matrTranslTruckFrontWheel2 * matrScaleTruckWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);
	myMatrix = resizeMatrix * matrTranslTruckBackWheel1 * matrScaleTruckWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);
	myMatrix = resizeMatrix * matrTranslTruckBackWheel2 * matrScaleTruckWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);
	myMatrix = resizeMatrix * matrTranslTruckBackWheel3 * matrScaleTruckWheel;
	glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
	glDrawArrays(GL_POLYGON, 35, 13);

	if (truckX - blueCarX <= 310) // When the blue car collides with the truck
	{
		glPointSize(15.0);
		glEnable(GL_POINT_SMOOTH);
		glLineWidth(5);
		codCol = 4;
		glUniform1i(colCodLocation, codCol);
		myMatrix = resizeMatrix * matrTranslSunLeftEye;
		glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
		glDrawArrays(GL_POINTS, 48, 1);
		myMatrix = resizeMatrix * matrTranslSunRightEye;
		glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
		glDrawArrays(GL_POINTS, 48, 1);
		myMatrix = resizeMatrix * matrTranslSunMouth;
		glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
		glDrawArrays(GL_LINE_STRIP, 35, 5);
	}
```

```
	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("Overtaking");
	glewInit();
	Initialize();
	glutDisplayFunc(RenderFunction);
	glutMouseFunc(mouse);
	glutCloseFunc(Cleanup);
	glutMainLoop();
}
```

Sasu Alexandru-Cristian
Grupa 342

Codul din shader-ul de fragment:

```
#version 330

in vec4 ex_Color;
uniform int colorCode;

out vec4 out_Color;

void main(void)
{
  switch (colorCode)
  {
      case 0:
        out_Color = ex_Color;
        break;
      case 1: // yellow - color for the sun
            out_Color=vec4 (1.0, 1.0, 0.0, 0.0);
            break;
      case 2: // gray - color for road
            out_Color=vec4 (0.5, 0.5, 0.5, 0.0);
            break;
      case 3: // white - color for the lane dividers
            out_Color=vec4 (1.0, 1.0, 1.0, 0.0);
            break;
      case 4: // black - color for the vehicles' wheels
            out_Color=vec4 (0.0, 0.0, 0.0, 0.0);
            break;
      case 5: // beige - color for the car that is overtaken
            out_Color=vec4 (0.627, 0.608, 0.511, 0.0);
            break;
      case 6: // blue - color for the car that performs the overtaking
            out_Color=vec4 (0.0, 0.0, 1.0, 0.0);
            break;
      case 7: // red - color for the truck
            out_Color=vec4 (1.0, 0.0, 0.0, 0.0);
            break;
      default:
            break;
  };
}
```

Sasu Alexandru-Cristian
Grupa 342

Codul din shader-ul de vârf:

```
#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;
}
```