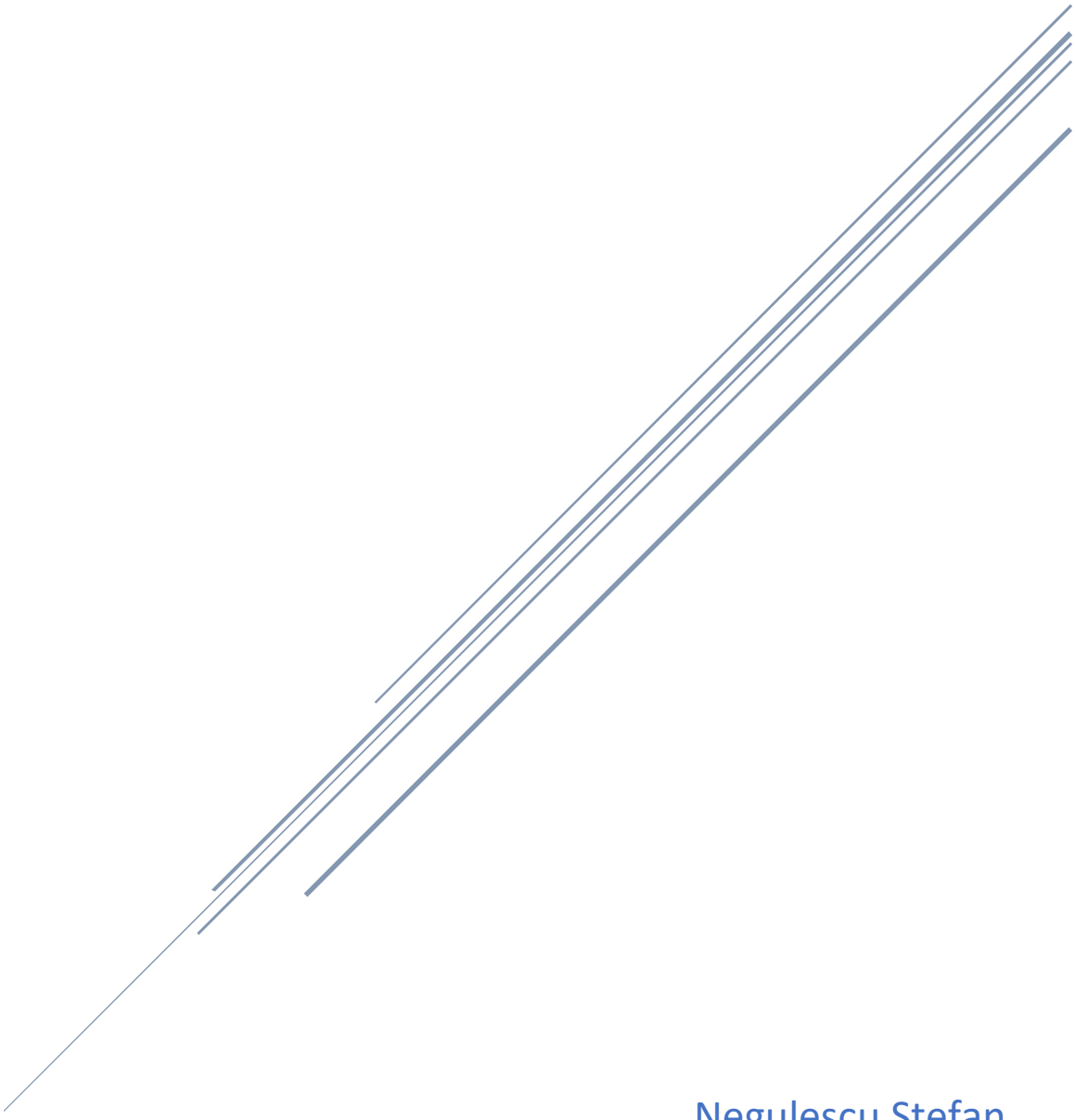
DOCUMENTATIE

Proiect I



Negulescu Stefan
Grupa 332

Contents

Conceptul proiectului

Proiectul simuleaza o cursa cu 2 masini. Pe parcursul cursei au loc accelerari, viraje si depasiri. Castiga masina care trece prima linia de finish.

Transformari

- **Inaintarea masinilor (spre stanga):** Aceasta miscare este realizata prin translatie, coordonata x fiind cea incrementata. In acest caz unghiul car.ang = 0.
- **Viraj (schimbarea benzii):** Aceasta miscare este realizata printr-o translatie si o rotatie. Pentru rotatie sunt utilizate si alte 2 translatii cu ajutorul carora aceasta are loc cu centrul in origine. Unghiul car.ang variaza.
- **Depasire:** Aceasta este realizata prin 2 viraje.
- **Miscarea camerei:** Pozitia observatorului este tratata in aceasta cursa ca o camera care se muta in functie de pozitia masinilor. Astfel Obsx este incrementat odata cu coordonata x a masinii albastre.
- **Drift:** Asemanator cu virajul, diferenta fiind unghiul de rotatie mai mare, cat si translatia masinii in sus.

Compunerea transformarilor in cazul masinilor arata astfel:

```
// se schimba pozitia observatorului
   glm::vec3 Obs = glm::vec3(Obsx, Obsy, Obsz);

// pozitia punctului de referinta
   Refx = Obsx; Refy = Obsy;
   glm::vec3 PctRef = glm::vec3(Refx, Refy, -1.0f);

// verticala din planul de vizualizare
   glm::vec3 Vert = glm::vec3(Vx, 1.0f, 0.0f);
   view = glm::lookAt(Obs, PctRef, Vert);

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

   matrTrans1 = glm::translate(glm::mat4(1.0f), glm::vec3(car.i, car.j, 0.0));
   matrTrans_11 = glm::translate(glm::mat4(1.0f), glm::vec3(-50.0, -15.0, 0.0));
   matrTrans_12 = glm::translate(glm::mat4(1.0f), glm::vec3(50.0, 15.0, 0.0));
   matrRot1 = glm::rotate(glm::mat4(1.0f), car.ang, glm::vec3(0.0, 0.0, 1.0));
          myMatrix = glm::mat4(1.0f);


   myMatrix = projection * view * myMatrix * matrTrans1 * (matrTrans_12 * matrRot1 *
   matrTrans_11);
```

De ce este original?

In realizarea proiectului nu a fost folosita nicio alta sursa de referinta.

Detalii

1. A fost create o clasa pentru masini cu atributele i, j, si ang, acestea fiind variabilele de matricele de translatie si rotatie. Aceasta clasa contine si metodele de deplasare ale masinilor. De asemenea, sunt initializate si pozitiile initiale ale masinilor.

```
class Car {
public:
	float i, j, ang;
	void forward() {
		//i += 5 ;
		if (ang >= PI / 30)
			ang -= PI / 100;
		else if (ang <= -PI / 30)
			ang += PI / 100;
		else
			ang = 0;
	}
	void go_up() {
		if (ang < 0) {
			j += 0.4;
			ang += PI / 100;
		}
		else {
			j += 1;
			if (ang < PI / 25)
				ang += PI / 100;
		}
	}
	void go_down() {
		if (ang > 0) {
			j -= 0.4;
			ang += -PI / 100;
		}
		else {
			j -= 1;
			if (ang > -PI / 25)
				ang += -PI / 100;
		}
	}
	void move_right() {
```

```
		if (j >= -70)
			go_down();
		else
			forward();
		glutPostRedisplay();
	}

	void move_left() {

		if (j <= 0)
			go_up();
		else
			forward();
		glutPostRedisplay();
	}

	void drift() {

		if (ang <= PI / 3.5)
			ang += PI / 400;
		i -= 1;
		j += 0.1;

		glutPostRedisplay();
	}
}blue_car, green_car;

void SetCars() {
	blue_car.i = 0.0;
	blue_car.j = 0.0;
	blue_car.ang = 0.0;
	green_car.i = 0.0;
	green_car.j = -70.0;
	green_car.ang = 0.0;
	}
```

2. Programul prezinta un posibil scenariu al cursei, descris in urmatoare functie.

```
void action(void) {

	// same speed
	if (blue_car.i < 500) {
		Obsx += 3;
		green_car.i += 3;
		blue_car.i += 3;
	}
	else
	// blue accelereaza
	if (blue_car.i >= 500 && blue_car.i <= 1000) {
		Obsx += 5;
```

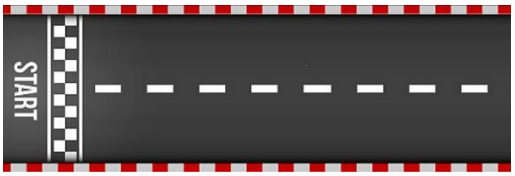
```
            green_car.i += 4;
            blue_car.i += 5;
        }
        else
        // blue depaseste
        if (blue_car.i >= 1000 && blue_car.i <= 2000) {
            Obsx += 5;
            green_car.i += 4;
            blue_car.i += 5;
            blue_car.move_right();
        }
        else
        // green intra in depasire pe banda stanga
        if (blue_car.i > 2000 && blue_car.i < 3000) {
            Obsx += 6;
            green_car.i += 6;
            blue_car.i += 6;
            green_car.move_left();
        }
        else
        // green trece de blue
        if (blue_car.i >= 3000 && blue_car.i < 5500) {
            Obsx += 6;
            green_car.i += 7;
            blue_car.i += 6;
        }
        else
        // green depaseste blue, trecand in fata sa, pe aceeasi banda
        if (blue_car.i >= 5500 && blue_car.i < 7500) {
            Obsx += 6;
            green_car.i += 6;
            blue_car.i += 6;
            green_car.move_right();
        }


        glutPostRedisplay();

    }
```

3. Coordonatele varfurilor

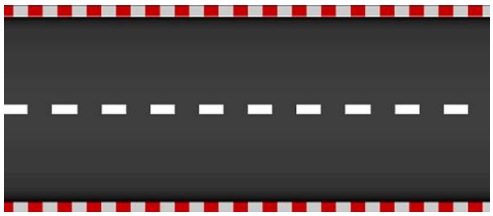
```
    // varfurile
    GLfloat Vertices[] = {

        // coordonate                  // culori                       // coordonate de
texturare
        -50.0f,  -100.0f, 0.0f, 1.0f,     1.0f, 0.0f, 0.0f,          0.0f, 0.0f, //
stanga jos
        500.0f,  -100.0f, 0.0f, 1.0f,     0.0f, 1.0f, 0.0f,          1.0f, 0.0f,  //
dreapta jos
        500.0f,  50.0f, 0.0f, 1.0f,       1.0f, 1.0f, 0.0f,          1.0f, 1.0f, //
dreapta sus
```

```
       -50.0f,  50.0f, 0.0f, 1.0f,       0.0f, 1.0f, 1.0f,          0.0f, 1.0f,  //
stanga sus
              0.0f, -20.0f, 0.0f, 1.0f,         1.0f, 0.0f, 0.0f,          0.0f, 0.0f,
              100.0f, -20.0f, 0.0f, 1.0f,       1.0f, 0.0f, 0.0f,          1.0f, 0.0f,
           100.0f, 50.0f, 0.0f, 1.0f,           1.0f, 0.0f, 0.0f,          1.0f, 1.0f,
              0.0f, 50.0f, 0.0f, 1.0f,          1.0f, 0.0f, 0.0f,          0.0f, 1.0f

    };

    // indicii pentru varfuri
    GLuint Indices[] = {
      0, 1, 2, // Primul triunghi
      0, 2, 3, //  Al doilea triunghi
         4, 5, 6, 7
           };
```

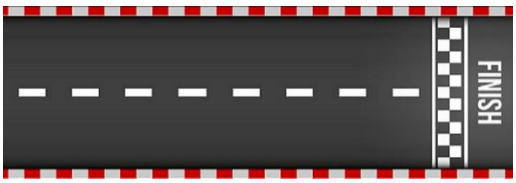
4. Pentru textura drumului sunt folosite 3 imagini. Prima este imaginea liniei de start, a doua este drumul normal (aceasta fiind folosita de 10 ori), si imaginea cu linia de finish. Pentru desenarea acestora se folosesc aceleasi 4 varfuri, fiecare imagine fiind translatata la dreapta precedentei.




```
       // Start texture
       glActiveTexture(GL_TEXTURE0);
       glBindTexture(GL_TEXTURE_2D, texture);
       CreateShaders();
       matrTrans_5 = glm::translate(glm::mat4(1.0f),
glm::vec3(0.0, 0.0, 0.0));
       myMatrix = projection * view * myMatrix * matrTrans_5;
       myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
       glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
       glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
       glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
```

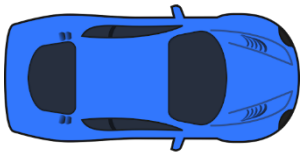


```
       // Road texture
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, texture_1);
       for (int j = 0; j <= 10; j++) {
              matrTrans_5 =
glm::translate(glm::mat4(1.0f), glm::vec3((j + 1) * 545,
0.0, 0.0));
              myMatrix = glm::mat4(1.0f);
              myMatrix = projection * view * myMatrix * matrTrans_5;
              myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
              glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
              glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
              glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
       }
```

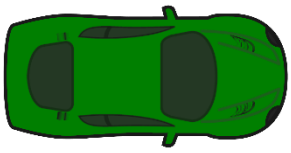



```
        // Finish texture
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_2D, texture_2);
        matrTrans_5 = glm::translate(glm::mat4(1.0f),
glm::vec3(12*545, 0.0, 0.0));
        myMatrix = glm::mat4(1.0f);
        myMatrix = projection * view * myMatrix * matrTrans_5;
        myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
        glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
        glUniform1i(glGetUniformLocation(ProgramId, "myTexture"), 0);
        glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
```

5. Pentru masini au fost folosite 2 imagini diferite pentru texturare.



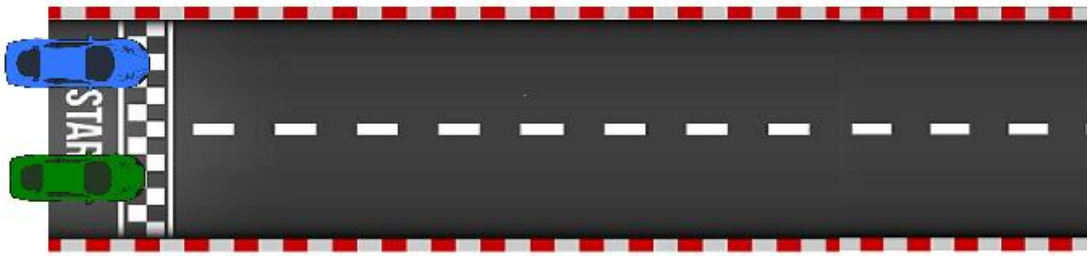
```
        // Blue car
        matrTrans1 = glm::translate(glm::mat4(1.0f),
glm::vec3(blue_car.i, blue_car.j, 0.0));
        matrTrans_11 = glm::translate(glm::mat4(1.0f), glm::vec3(-
50.0, -15.0, 0.0));
        matrTrans_12 = glm::translate(glm::mat4(1.0f),
glm::vec3(50.0, 15.0, 0.0));
        matrRot1 = glm::rotate(glm::mat4(1.0f), blue_car.ang,
glm::vec3(0.0, 0.0, 1.0));
        myMatrix = glm::mat4(1.0f);
        myMatrix = projection * view * myMatrix * matrTrans1 * (matrTrans_12 * matrRot1 *
matrTrans_11);
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_2D, texture_3);
        myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
        glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
        glPointSize(20.0);
        glDrawElements(GL_POLYGON, 4, GL_UNSIGNED_INT, (void*)(24));
```



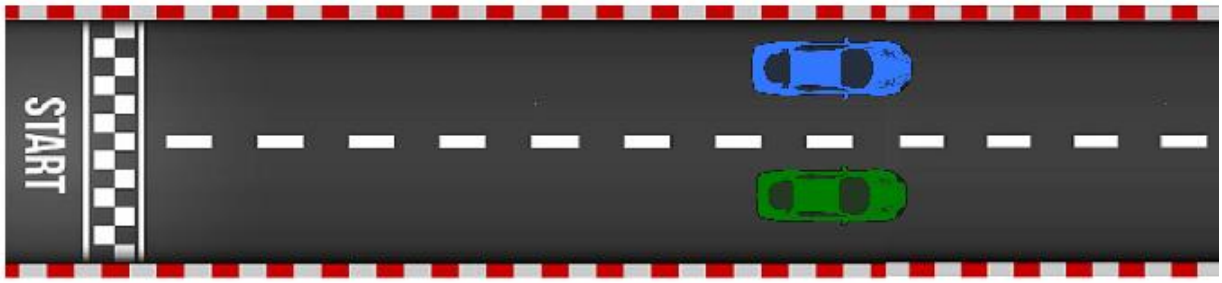
```
        // Green car
        //matrTrans2 = glm::translate(glm::mat4(1.0f),
glm::vec3(xx, -70.0, 0.0));
        matrTrans2 = glm::translate(glm::mat4(1.0f),
glm::vec3(green_car.i,green_car.j, 0.0));
        matrTrans_21 = glm::translate(glm::mat4(1.0f), glm::vec3(-
50.0, -15.0, 0.0));
        matrTrans_22 = glm::translate(glm::mat4(1.0f),
glm::vec3(50.0, 15.0, 0.0));
        matrRot2 = glm::rotate(glm::mat4(1.0f), green_car.ang,
glm::vec3(0.0, 0.0, 1.0));
        myMatrix = glm::mat4(1.0f);
        myMatrix = projection * view * myMatrix * matrTrans2 * (matrTrans_22 * matrRot2 *
matrTrans_21);
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_2D, texture_4);
        myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
        glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
        glPointSize(20.0);
        glDrawElements(GL_POLYGON, 4, GL_UNSIGNED_INT, (void*)(24));
```

6. Capturi de ecran

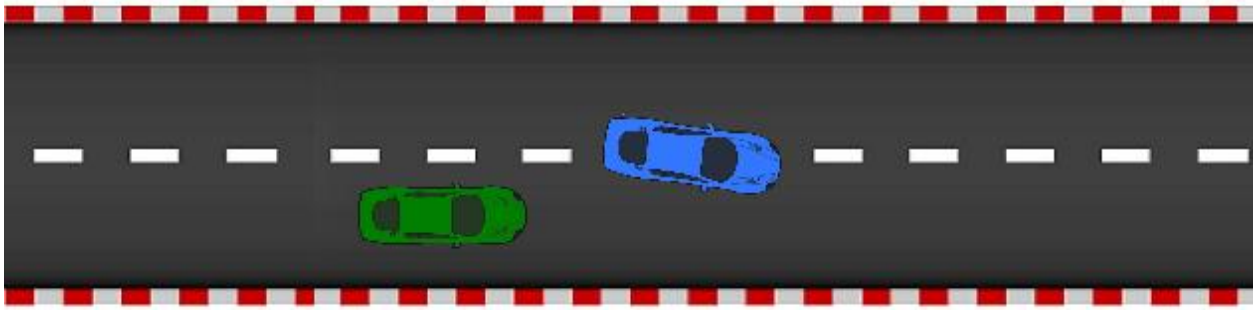
- Initial masinile stationeaza inaintea liniei de start



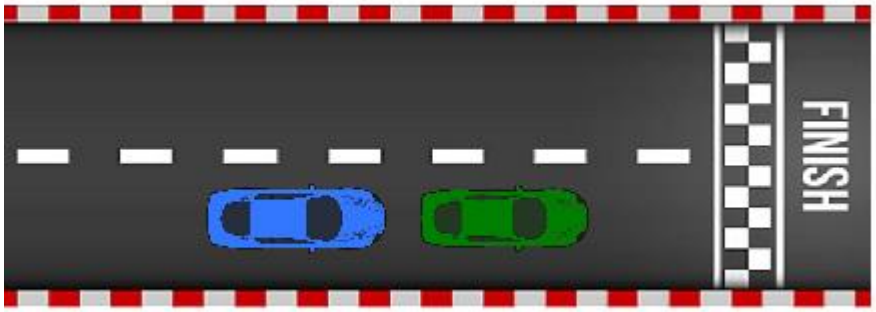
- Dupa start, acestea se deplaseaza cu aceeasi viteza.



- Masina albastra vireaza dreapta pentru a ajunge in fata celei verzi.



- Aproape de final, masina verde preia conducerea.



- Ambele masini executa un drift inainte de a frana dupa trecerea liniei de finish.