Iova Nicoleta Carmen
Tismănaru Artemis Constantina
Grafică pe Calculator
Grupa 464

Proiect 2D

-Depășirea unei mașini-

Descrierea proiectului

Proiectul simulează o scenă de trafic în care o mașină încearcă să depășească în siguranță pe un drum, folosind grafică OpenGL. Programul prezintă vizual imaginea unei șosele animate, cu mai multe mașini aflate în mișcare. Fiecare mașină are propriile caracteristici de poziționare, viteză și comportament, iar acțiunea centrală a simulării este încercarea de depășire a unei mașini aflate în față.

Programul gestionează animațiile, coliziunile, și sincronizarea mișcărilor mașinilor, oferind o atmosferă realistă de trafic. În plus, drumul și fundalul adaugă context vizual, iar texturile și rotațiile sunt utilizate pentru a îmbunătăți realismul scenei.

Pentru complexitatea proiectului am ales ca depășirea să aibă loc la dorința utilizatorului, atunci când acesta acționează cu un click mouse-ul sau touchpad-ul. De asemenea, drept elemente adiționale am integrat și o mașină care circulă regulamentar pe sens opus.

Dacă în urma acțiunii utilizatorului de a depăși mașina în cauză intră în coliziune cu mașina de pe sensul opus se va produce un accident în urma căruia mașina care circulă regulamentar este aruncată în afara carosabilului iar o mașină de poliție își va face apariția și va opri în dreptul accidentului. În schimb, în cazul în care depășirea se petrece în parametrii normali, fiecare mașină își continuă traiectoria.

Funcții importante din cod

- **Configurarea obiectelor grafice – VAO**

Funcția pentru fundalul ferestrei – iarba:

```
void initializeBackgroundVAO() {
// Vertices for a full-screen quad
float backgroundVertices[] = {
// Position       // Texture Coords
-1.0f, -1.0f,     0.0f, 0.0f,
1.0f, -1.0f,     1.0f, 0.0f,
1.0f,  1.0f,     1.0f, 1.0f,
-1.0f,  1.0f,     0.0f, 1.0f
};

glGenVertexArrays(1, &backgroundVAO);
glGenBuffers(1, &backgroundVBO);
```

```
    glBindVertexArray(backgroundVAO);

    glBindBuffer(GL_ARRAY_BUFFER, backgroundVBO);
    glBufferData(GL_ARRAY_BUFFER,                    sizeof(backgroundVertices),
    backgroundVertices, GL_STATIC_DRAW);

    glVertexAttribPointer(0,  2,  GL_FLOAT,  GL_FALSE,  4  *  sizeof(float),
    (void*)0);
    glEnableVertexAttribArray(0);

    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 4 * sizeof(float), (void*)(2
    * sizeof(float)));
    glEnableVertexAttribArray(1);

    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
    }
```

Funcția pentru mașini:

```
    void initializeVAO() {
        float carVertices[] = {
            -0.1f, -0.05f, 0.0f, 0.0f,
             0.0f, -0.05f, 1.0f, 0.0f,
             0.0f,  0.05f, 1.0f, 1.0f,
            -0.1f,  0.05f, 0.0f, 1.0f
        };

        glGenVertexArrays(1, &VAO);
        glGenBuffers(1, &VBO);

        glBindVertexArray(VAO);

        glBindBuffer(GL_ARRAY_BUFFER, VBO);
        glBufferData(GL_ARRAY_BUFFER,     sizeof(carVertices),     carVertices,
    GL_STATIC_DRAW);

        glVertexAttribPointer(0,  2,  GL_FLOAT,  GL_FALSE,  4  *  sizeof(float),
    (void*)0);
        glEnableVertexAttribArray(0);

        glVertexAttribPointer(1,  2,  GL_FLOAT,  GL_FALSE,  4  *  sizeof(float),
    (void*)(2 * sizeof(float)));
        glEnableVertexAttribArray(1);

        glBindBuffer(GL_ARRAY_BUFFER, 0);
        glBindVertexArray(0);
    }
```

Iova Nicoleta Carmen
Tismănaru Artemis Constantina
Grafică pe Calculator
Grupa 464
Funcția pentru drum:

```
void initializeRoadVAO() {
    float roadVertices[] = {
        // PosX, PosY, TexCoordX, TexCoordY
       -1.0f, -0.4f, 0.0f, 0.0f,
        1.0f, -0.4f, 1.0f, 0.0f,
        1.0f,  0.4f, 1.0f, 1.0f,
       -1.0f,  0.4f, 0.0f, 1.0f
    };

    glGenVertexArrays(1, &roadVAO);
    glGenBuffers(1, &roadVBO);

    glBindVertexArray(roadVAO);

    glBindBuffer(GL_ARRAY_BUFFER, roadVBO);
    glBufferData(GL_ARRAY_BUFFER,    sizeof(roadVertices),    roadVertices,
GL_STATIC_DRAW);

    glVertexAttribPointer(0,  2,  GL_FLOAT,  GL_FALSE,  4  *  sizeof(float),
(void*)0);
    glEnableVertexAttribArray(0);

    glVertexAttribPointer(1,  2,  GL_FLOAT,  GL_FALSE,  4  *  sizeof(float),
(void*)(2 * sizeof(float)));
    glEnableVertexAttribArray(1);

    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);
}
```

- **Funcții de desenare a obiectelor:**

Funcția de desenare a fundalului:

```
void drawBackground() {
    glUseProgram(shaderProgram);

    glm::mat4 transform = glm::mat4(1.0f);
    glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "transform"), 1,
GL_FALSE, glm::value_ptr(transform));

    glBindTexture(GL_TEXTURE_2D, backgroundTexture);
    glBindVertexArray(backgroundVAO);
    glDrawArrays(GL_QUADS, 0, 4);
    glBindVertexArray(0);
}
```

Iova Nicoleta Carmen
Tismănaru Artemis Constantina
Grafică pe Calculator
Grupa 464
Funcția de desenare a mașinilor:

Această funcție desenează o mașină într-o poziție specificată, cu o textură și o rotație opțională. Aplică transformările pentru poziția și rotația mașinii folosind matrice, astfel încât mașina să se poată mișca și întoarce în cadrul simulării.

```
void drawCar(float posX, float posY, GLuint textureID, float rotation = 0.0f)
{
    glUseProgram(shaderProgram);

    // Matrice de transformare pentru pozitionarea si rotirea masinii
    glm::mat4 transform = glm::mat4(1.0f);

    // Translatare la pozitia initiala a masinii
    transform = glm::translate(transform, glm::vec3(posX, posY, 0.0f));

    // Roteste in jurul centrului masinii
    transform = glm::translate(transform, glm::vec3(0.025f, 0.025f, 0.0f));
// Mutam in centru
    transform     =     glm::rotate(transform,     glm::radians(rotation),
glm::vec3(0.0f, 1.0f, 0.0f)); // Rotim in jurul centrului
    transform   =   glm::translate(transform,   glm::vec3(-0.025f,   -0.025f,
0.0f)); // Mutam inapoi

    // Aplicam transformarea
    glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "transform"), 1,
GL_FALSE, glm::value_ptr(transform));

    // Desenam masina cu textura
    glBindTexture(GL_TEXTURE_2D, textureID); // Aplicam textura masinii
    glBindVertexArray(VAO);
    glDrawArrays(GL_QUADS, 0, 4);
    glBindVertexArray(0);
}
```

- **Funcția de display:**

Aici sunt apelate funcțiile de desenare pentru drum și mașini. Această funcție este chemată automat de OpenGL la fiecare refresh al ecranului pentru a afișa poziția actualizată a obiectelor.

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    // fundalul
    drawBackground();

    // drumul
    glUseProgram(shaderProgram);
    glm::mat4 roadTransform = glm::mat4(1.0f);
    roadTransform    =    glm::rotate(roadTransform,    glm::radians(180.0f),
glm::vec3(0.0f, 0.0f, 1.0f));
    glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "transform"), 1,
GL_FALSE, glm::value_ptr(roadTransform));
    glBindTexture(GL_TEXTURE_2D, roadTexture);
    glBindVertexArray(roadVAO);
    glDrawArrays(GL_QUADS, 0, 4);
    glBindVertexArray(0);

    // masinile
    drawCar(car1PosX, -0.2f, carTexture);
    drawCar(car2PosX, car2PosY, carTexture, car2Rotation);
```

```
        drawCar(car3PosX, car3PosY, carTexture, car3Rotation);

        // masina de politie
        if (car4Visible) {
            drawCar(car4PosX, car4PosY, policeCarTexture);
        }
        glutSwapBuffers();
    }
```

- **Funcția de detectare a coliziunii:**

```
void checkCollisionAndAvoidance() {
    // Distanta minima intre masini
    float minSafeDistance = 0.10f;

    // Daca masina de pe contrasens (car3) este prea aproape de masina care
depaseste (car2)
    if (fabs(car2PosX - car3PosX) < minSafeDistance && car2PosY >= 0.08f) {
        car3PosY += 0.05f;
        isColliding = true;
        hit = true;
        car4Visible = true;  // Activeaza vizibilitatea masinii de politie
        car4StopPosX = car3PosX;
    }
    else {
        isColliding = false;
    }
}
```

- **Funcția de mișcare a mașinilor:**

Actualizează poziția mașinilor în funcție de viteza și condițiile de depășire. Dacă mașina din spate accelerează și începe depășirea, funcția ajustează poziția și unghiul pentru a simula ieșirea pe contrasens și revenirea pe banda inițială.

```
void update(int value) {
    car1PosX += initialSpeed;
    car2PosX += car2Speed;

    if (!hit) {
        car3PosX += car3Speed;
    }

    checkCollisionAndAvoidance();

    if (car4Visible) {
        if (car4PosX > car4StopPosX) {
            car4PosX += car4Speed;
        }
        else {
            car4PosX = car4StopPosX;  // Opreste matina 4 la pozitia dorita
        }
    }

    if (isColliding) {
        car3Rotation += 10.0f;
    }

    if (isAccelerating && car2Speed < maxSpeed) {
        car2Speed += accelerationRate;
    }
```

```
        if (isAccelerating && !overtakingInProgress) {
            overtakingInProgress = true;
        }

        if (overtakingInProgress) {
            if (!overtook) {
                car2Rotation = 8.0f;
                if (car2PosY <= 0.1f) car2PosY += 0.002f;
                if (car2PosX >= car1PosX - 0.05f) {
                    overtook = true;
                }
            }
            else {
                car2Rotation = 0.0f;
                if (car2PosY > -0.2f) car2PosY -= 0.001f;
                else {
                    overtakingInProgress = false;
                    isAccelerating = false;
                    car2Speed = initialSpeed;
                }
            }
        }

        glutPostRedisplay();
        glutTimerFunc(16, update, 0);
    }
```

- **Funcția de mouse:**

Detectează click-ul stânga și activează accelerația mașinii din spate, declanșând manevra de depășire.

```
    void mouse(int button, int state, int x, int y) {
        if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
            isAccelerating = true;
            car2Speed = initialSpeed;
        }
    }
```

Shader-ul de vertex

```
#version 330 core
layout (location = 0) in vec2 aPos;
layout (location = 1) in vec2 aTexCoord;
uniform mat4 transform;
out vec2 TexCoord;
void main() {
    gl_Position = transform * vec4(aPos, 0.0, 1.0);
    TexCoord = aTexCoord;
}
```

Iova Nicoleta Carmen
Tismănaru Artemis Constantina
Grafică pe Calculator
Grupa 464

Shader-ul de fragment

```
#version 330 core
out vec4 FragColor;
uniform sampler2D texture1;
in vec2 TexCoord;
void main() {
    FragColor = texture(texture1, TexCoord);
}
```

Transformările obiectelor:

1. Translația – deplasarea mașinilor de-a lungul axei X, simulând mișcarea lor pe drum
2. Rotația – asupra mașinii care depășește cât și a mașinii care circulă pe contrasens au fost aplicate rotații care să simuleze depășirea (ieșirea de pe bandă) și urmarea unui accident (aruncarea mașinii accidentate în afara carosabilului)

Puteți vizualiza aici:

https://youtu.be/XOTl5I4RG-k (Coliziunea)

https://youtu.be/t0Cl7Pth_zw (Depășire în siguranță)

Imaginea finală: