Mihai Tuhari

[mihai-razvan.tuhari@s.unibuc.ro](mailto:mihai-razvan.tuhari@s.unibuc.ro)

<https://github.com/mihaituhari/>

**Grafica pe calculator**

Proiect 1 (2D)

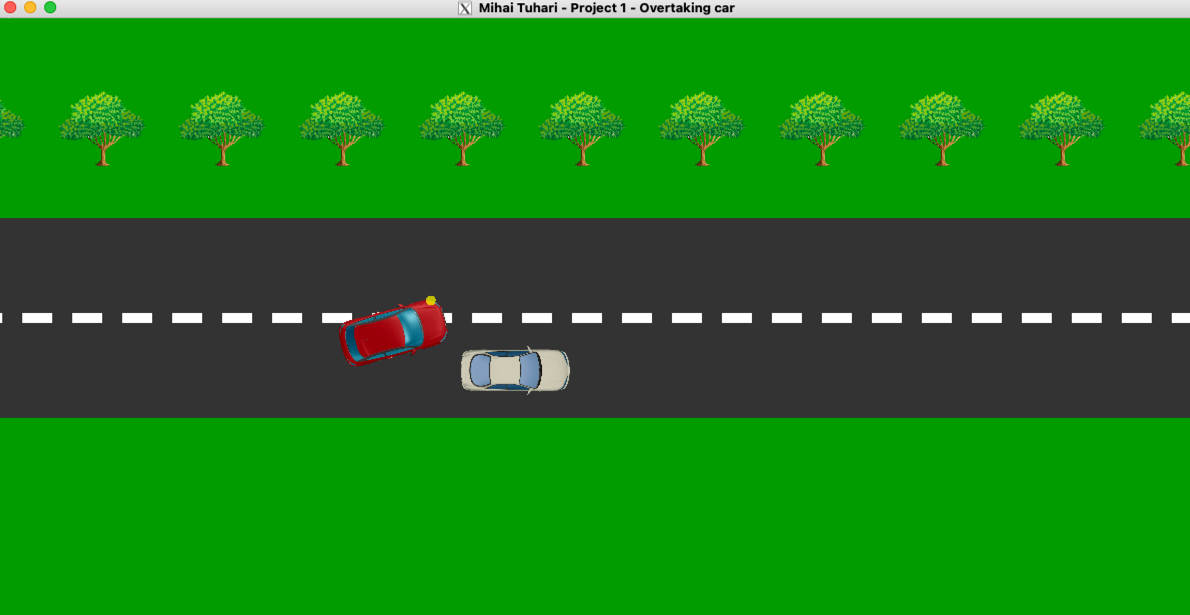
*Dezvoltati un proiect 2D pentru tema aleasa.*

*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.*

🚙 🚗💨 🌲🌲🌲🌲

## Demo

Pentru o lectura mai coerenta si simpla asupra documentatiei de mai jos, incepem cu o captura de ecran a proiectului.



## 

## Introducere

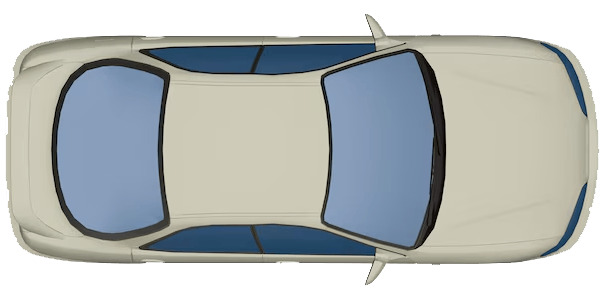
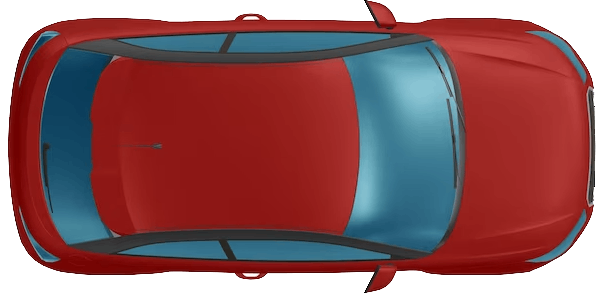
Am realizat o simulare a unei depasiri intre doua masini, folosind OpenGL si C++.

Programul principal este in fisierul [proiect1.cpp](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/proiect1.cpp) si foloseste [libraria STB](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/libs/stb_image.h) pentru incarcarea texturilor.

## **Texturi**

Pentru manipularea texturilor am ales STB in loc de SOIL pentru ca pe MacOS SOIL nu este compatibil.

Texturile se regasesc in folderul [textures](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/textures) si sunt fisiere PNG cu fundal transparent:

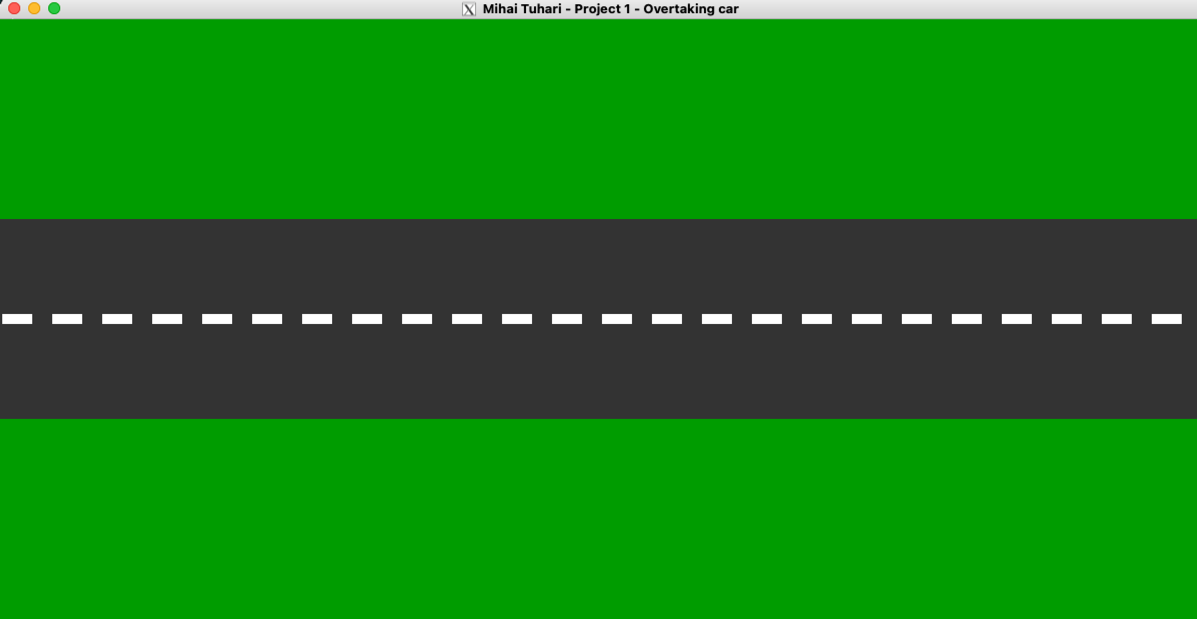
  

## **Conceptul**

Animatia este creata din mai multe componente cu roluri diferite.

### Scena statica

Scena este un dreptunghi care reprezinta drumul pe care se deplaseaza masinile. Este desenat cu culoarea verde iar asfaltul este gri.



### Decor dinamic

🛣️ Pentru a simula miscarea, am adaugat un marcaj discontinuu pe mijlocul drumului prin functia drawRoad(). Aceasta genereaza dreptunghiuri albe cu o anumita distanta intre ele.

🌲 In partea superioara sunt desenati copaci tot pentru a simula miscarea si pentru un plus de complexitate. Acestia sunt adaugati prin drawTrees() si sunt dreptunghiuri ce se repeta pe latimea ecranului si au proportii 39x32 peste care se aplica textura [tree.png](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/textures/tree.png) cu optiunile de blending GL\_SRC\_ALPHA si GL\_ONE\_MINUS\_SRC\_ALPHA pentru a afisa doar textura.



### Masinile

Sunt adaugate 2 masini care se deplaseaza pe scena cu functia drawCar().

* Masina 1 este cea inceata
* Masina 2 este cea rapida

Acestea sunt dreptunghiuri cu proportii 100x55 peste care se aplica texturile [car1.png](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/textures/car1.png) si [car2.png](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/textures/car2.png) cu optiuni de blending similare celor de la copaci.

### Semnalizarea

Masina 2 are atasat si conceptul de "semnalizare" pentru a indica depasirea. 🟡

Semnalizarea este un singur cerc galben (facut cu helperul drawCircle()) cu transparenta 70% ce este atasat de masina. Aceasta este o singura instanta si in functie de valoarea blinkersOn *(0=oprit, 1=stanga, 2=dreapta)* isi schimba pozitia pe axa Y fata de masina.

Programatic, am simulat o semnalizare apropiata de un comportament real:

* Semnalizarea de depasire (stanga) este pornita cand mai sunt 2 lungimi de masina pana la masina din fata
* Pe timpul depasirii, semnalizarea de depasire ramane pornita
* Dupa ce masina lenta a fost intrecuta cu o lungime de masina, este pusa semnalizare de revenire (dreapta)

### Depasirea

In timpul schimbarii de banda, masina 2 schimba banda pe axa Y cu o anumita viteza si unghi de rotatie. La revenire, masina 2 se intoarce la banda initiala.

## **Aspecte tehnice**

### Design modular

Am abordat proiectul cu un design modular si am incercat sa folosesc cat mai multe functii pentru a separa logica.

### Configurabilitate

Animatia este usor configurabila din variabilele globale definite la inceputul fisierului [proiect1.cpp](https://github.com/mihaituhari/fmi/blob/main/gc/proiect1/proiect1.cpp).

Acolo regasiti variabile pentru:

* Globale (dimensiune extra, PI pentru cercuri)
* Texturi (cale fisiere, ID-uri, etc)
* Masini (pozitii initiala, viteze de deplasare, dimensiuni)
* Depasire (rotatie, rotatie maxima, buffer siguranta revenire din depasire)
* Drum (grosime marcaj, spatiere copaci)
* Scena (viteza animatie, interval repetare)

### Transformari

Pentru a realiza animatia, am folosit urmatoarele transformari:

* Translatie
* Rotatie

Cand o masina ajunge la capatul ecranului, aceasta este repusa la inceputul drumului.

### Limba

Intreg codul (cu tot cu comentarii), este scris in limba Engleza din motive de coerenta si simplitate, pentru a evita combinatia intre termeni in limba Romana si Engleza.

### Video

Puteti vedea unvideo cu animatia in actiune la sfarsitul paginii <https://github.com/mihaituhari/fmi/tree/main/gc/proiect1> ori direct [aici](https://github.com/user-attachments/assets/fc02d11f-a344-4455-b8ed-20c73d5e48cb).

### Codul sursa

Este inclus pe paginile urmatoare, dar il puteti regasi si pe contul meu de GitHub la adresa:

<https://github.com/mihaituhari/fmi/tree/main/gc/proiect1>

| */\*\**  *\* Project 1 - Overtaking car*  *\**  *\* @author Mihai Tuhari*  *\* @date November 2024*  *\*/*  #define **STB\_IMAGE\_IMPLEMENTATION**  #include <iostream>  #include <string>  #include <GL/freeglut.h>  #include "libs/stb\_image.h"  *// Globals*  *const int* windowWidth = 1200;  *const int* windowHeight = 600;  *const double* PI = 3.141592653589793; *// For circles*  *// Texture variables*  GLuint textureIDCar1, textureIDCar2, textureTree;  *const* std::string texturePath = "/Volumes/mihai/dev/fmi/gc/proiect1/textures/";  *int* width, height, channels;  *// Car positions and attributes*  GLfloat car1X = 360, car1Y = 220;  GLfloat car2X = 30, car2Y = 220;  GLfloat carSpeed1 = 1, carSpeed2 = 3;  GLfloat carWidth = 110, carHeight = 55;  *// Overtaking variables*  GLfloat car2Rotation = 0;  *const* GLfloat maxRotation = 15;  GLfloat overtakeSafety = 20;  *// Blinker variables*  *bool* blinkerState = *false*; *// On or off for interval*  *const* GLint blinkerInterval = 500;  *const* GLfloat blinkerRadius = 10.0;  GLint blinkersOn = 0; *// 0 = off, 1 = left blinker, 2 = right blinker*  *// Road elements*  GLfloat roadDividerWidth = 5;  GLint treeSpacing = 120;  *// Scene variables*  GLfloat animationInterval = 16; *// ~60 FPS (= 1000 / 60)*  GLfloat sceneSpeed = 3.6;  GLfloat sceneOffset = 0;  *// Flags*  *bool* overtaking = *false*;  *bool* returningToLane = *false*;  *// Circle drawing helper function*  *void* drawCircle(GLfloat radius, GLint segments) {  glBegin(**GL\_TRIANGLE\_FAN**);  glVertex2f(0.0f, 0.0f);  *for* (*int* i = 0; i <= segments; i++) {  GLfloat angle = 2.0f \* PI \* i / segments;  GLfloat x = radius \* cos(angle);  GLfloat y = radius \* sin(angle);  glVertex2f(x, y);  }  glEnd();  }  *// Load a texture from file*  *void* loadTexture(*const* std::string &path, GLuint &textureID) {  stbi\_set\_flip\_vertically\_on\_load(1); *// Flip the image vertically on load*  *unsigned char* \*image = stbi\_load(path.c\_str(), &width, &height, &channels, STBI\_rgb\_alpha);  *if* (!image) {  std::cerr << "Failed to load texture: " << path << std::endl;  exit(1);  }  glGenTextures(1, &textureID);  glBindTexture(**GL\_TEXTURE\_2D**, textureID);  glTexParameteri(**GL\_TEXTURE\_2D**, **GL\_TEXTURE\_MIN\_FILTER**, **GL\_LINEAR**);  glTexParameteri(**GL\_TEXTURE\_2D**, **GL\_TEXTURE\_MAG\_FILTER**, **GL\_LINEAR**);  glTexParameteri(**GL\_TEXTURE\_2D**, **GL\_TEXTURE\_WRAP\_S**, **GL\_CLAMP\_TO\_EDGE**);  glTexParameteri(**GL\_TEXTURE\_2D**, **GL\_TEXTURE\_WRAP\_T**, **GL\_CLAMP\_TO\_EDGE**);  glTexImage2D(**GL\_TEXTURE\_2D**, 0, **GL\_RGBA**, width, height, 0, **GL\_RGBA**, **GL\_UNSIGNED\_BYTE**, image);  stbi\_image\_free(image);  }  *// Draw the moving trees on the side of the road*  *void* drawTrees() {  *int* yOffset = 450;  *int* rectWidth = 39 \* 2.5;  *int* rectHeight = 32 \* 2.5;  *int* numTrees = (windowWidth / treeSpacing) + 2; *// Extra trees for smooth transition*  glEnable(**GL\_TEXTURE\_2D**);  glBindTexture(**GL\_TEXTURE\_2D**, textureTree);  *for* (*int* i = 0; i < numTrees; ++i) {  GLfloat treeX = (i \* treeSpacing) + fmod(sceneOffset, treeSpacing);  glPushMatrix();  glTranslatef(treeX, 0, 0);  glBegin(**GL\_QUADS**);  glTexCoord2f(0.0, 0.0); *// Bottom-left*  glVertex2i(-rectWidth / 2, yOffset);  glTexCoord2f(1.0, 0.0); *// Bottom-right*  glVertex2i(rectWidth / 2, yOffset);  glTexCoord2f(1.0, 1.0); *// Top-right*  glVertex2i(rectWidth / 2, yOffset + rectHeight);  glTexCoord2f(0.0, 1.0); *// Top-left*  glVertex2i(-rectWidth / 2, yOffset + rectHeight);  glEnd();  glPopMatrix();  }  glDisable(**GL\_TEXTURE\_2D**);  }  *// Toggle blinker state*  *void* toggleBlinker(*int* value) {  *if* (blinkersOn) { *// Only toggle if blinking is active*  blinkerState = !blinkerState;  } *else* {  blinkerState = *false*; *// Ensure blinker is off when not needed*  }  glutTimerFunc(blinkerInterval, toggleBlinker, 0);  }  *// Draw asphalt road with moving dividers*  *void* drawRoad() {  *// Asphalt*  glColor3f(0.2, 0.2, 0.2);  glBegin(**GL\_QUADS**);  glVertex2f(0, 200);  glVertex2f(windowWidth, 200);  glVertex2f(windowWidth, 400);  glVertex2f(0, 400);  glEnd();  *// Road divider (moving vertical lines)*  glColor3f(1, 1, 1);  *for* (GLfloat x = sceneOffset; x < windowWidth; x += 50) {  glBegin(**GL\_QUADS**);  glVertex2f(x, 300 - roadDividerWidth); *// Center of the road*  glVertex2f(x + 30, 300 - roadDividerWidth);  glVertex2f(x + 30, 300 + roadDividerWidth);  glVertex2f(x, 300 + roadDividerWidth);  glEnd();  }  }  *// Draw a car*  *void* drawCar(GLfloat x, GLfloat y, GLuint textureID, GLfloat rotationAngle = 0, *bool* fasterCar = *false*) {  glEnable(**GL\_TEXTURE\_2D**);  glBindTexture(**GL\_TEXTURE\_2D**, textureID);  *// Store the original matrix*  glPushMatrix();  *// Move to car position and apply rotation*  glTranslatef(x + carWidth / 2, y + carHeight / 2, 0);  glRotatef(rotationAngle, 0.0f, 0.0f, 1.0f);  *// Draw car body*  glPushMatrix();  glTranslatef(-carWidth / 2, -carHeight / 2, 0);  glBegin(**GL\_QUADS**);  glTexCoord2f(0, 0);  glVertex2f(0, 0);  glTexCoord2f(1, 0);  glVertex2f(carWidth, 0);  glTexCoord2f(1, 1);  glVertex2f(carWidth, carHeight);  glTexCoord2f(0, 1);  glVertex2f(0, carHeight);  glEnd();  glPopMatrix();  glDisable(**GL\_TEXTURE\_2D**);  *// Draw blinker in the same transformed space (of the overtaking car)*  *if* (fasterCar && blinkersOn && blinkerState) {  GLfloat blinkerX, blinkerY;  blinkerX = carWidth / 2 - blinkerRadius;  blinkerY = blinkersOn == 1  ? carHeight \* 0.4 *// Left blinker*  : -carHeight \* 0.4; *// Right blinker*  glTranslatef(blinkerX, blinkerY, 0);  glColor4f(1.0, 1.0, 0.0, 0.7); *// Yellow with transparency*  drawCircle(blinkerRadius / 2, 20);  }  glPopMatrix();  }  *void* init() {  glClearColor(0, 0.6, 0, 1); *// Green background, grass*  glMatrixMode(**GL\_PROJECTION**);  glLoadIdentity();  gluOrtho2D(0.0, windowWidth, 0.0, windowHeight);  glutTimerFunc(blinkerInterval, toggleBlinker, 0);  }  *void* display() {  glClear(**GL\_COLOR\_BUFFER\_BIT**);  drawRoad();  *// Enable blending for trees and cars*  glEnable(**GL\_BLEND**);  glBlendFunc(**GL\_SRC\_ALPHA**, **GL\_ONE\_MINUS\_SRC\_ALPHA**);  drawTrees();  drawCar(car1X, car1Y, textureIDCar1, 0, *false*);  drawCar(car2X, car2Y, textureIDCar2, car2Rotation, *true*);  glDisable(**GL\_BLEND**);  glutSwapBuffers();  }  *void* update(*int* value) {  *// Move the screen*  sceneOffset -= sceneSpeed;  *// Move cars*  car1X += carSpeed1;  car2X += carSpeed2;  *// Update blinker state based on car positions*  GLfloat distance = abs(car1X - car2X);  *if* (car1X > car2X && distance <= carWidth \* 3) {  blinkersOn = 1; *// Left blinker*  } *else if* (car2X > car1X && distance <= carWidth \* 2) {  blinkersOn = 2; *// Right blinker*  } *else* {  blinkersOn = 0; *// No blinker*  }  *// Handle the second car's overtaking behavior*  *if* (!overtaking && !returningToLane) {  *// Start overtaking*  *if* (car1X > car2X && (car1X - car2X < (carWidth \* 1.8))) {  overtaking = *true*;  }  } *else if* (overtaking) {  *// Move up and rotate the car for overtaking*  *if* (car2Y - car1Y < carHeight + overtakeSafety) {  car2Y += 1;  *if* (car2Rotation < maxRotation) {  car2Rotation += 0.8;  }  } *else* {  *// In the overtaking position, parallel to the other car*  car2Rotation = 0;  }  *// Overtaking completed. Will return to initial lane*  *if* (car2X - car1X > (carWidth)) {  overtaking = *false*;  returningToLane = *true*;  }  } *else if* (returningToLane) {  *// Rotate and move down to the initial lane*  car2Y -= 1;  *if* (car2Rotation > -maxRotation) {  car2Rotation -= 0.8;  }  *// Returned to initial lane*  *if* (car2Y <= 220) {  car2Y = 220;  car2Rotation = 0.0;  returningToLane = *false*;  }  }  *// Loop cars when leaving the screen*  *if* (car1X > windowWidth) {  car1X = -carWidth;  }  *if* (car2X > windowWidth) {  car2X = -carWidth;  }  glutPostRedisplay();  glutTimerFunc(animationInterval, update, 0);  }  *int* main(*int* argc, *char* \*\*argv) {  glutInit(&argc, argv);  glutInitDisplayMode(**GLUT\_DOUBLE** | **GLUT\_RGB**);  glutInitWindowSize(windowWidth, windowHeight);  glutCreateWindow("Mihai Tuhari - Project 1 - Overtaking car");  init();  loadTexture(texturePath + "car1.png", textureIDCar1);  loadTexture(texturePath + "car2.png", textureIDCar2);  loadTexture(texturePath + "tree.png", textureTree);  glutDisplayFunc(display);  glutTimerFunc(animationInterval, update, 0);  glutMainLoop();  *return* 0;  } |
| --- |