

Nume:

Bragă Alecsandru-Nicolae

Gava Bogdan

Valcu Mihai-Constantin

Grupa: 351

Proiect 3D Grafica pe Calculator

**Crash Mission Game**

Descriere

Pentru acest proiect am realizat un joc in care controlam o masina de politie aflata in misiune si trebuie sa fie evitate masinile care vin pe contrasens. Scopul este de a obtine un scor cat mai mare.

Convertirea proiectului 2D in 3D

* Masina care vine de pe contrasens era formata din mai multe poligoane de cate 4 laturi (capota, parbriz, faruri, etc) o fost convertita intr-un paralelipiped care are 4 sfere pentru roti.

 

* Am desenat niste copaci 3D, acestia fiind formatii dintr-un cilindru cu o sfera in extremitatea superioara.



Aspecte punctuale

1. Ca si obiectele 3D avem masinile si copacii.
2. Am incercat sa adaugam o lumina si materiale cu culori cat mai naturale.
3. Ca si texturi avem imaginea de la inceputul jocului.
4. Scorul, textul de pe ecranul de inceput si sfarsit este realizat cu glutBitmapCharacter.
5. Tasta “X” porneste jocul cand suntem in ecranul de start.
6. Tasta “C” inchide jocul dupa ce masina se buseste si intram in ecranul de final.
7. Masina pe care o controlam se deplaseaza folosind sagetile.
8. La fel ca la jocul 2D, jocul 3D se deplaseaza tot pe 3 benzi de circulatie incercand sa evite masinile de pe contrasens.

Originalitate

La fel ca la proiectul 2D, am retinut highscore-ul, scorul actual este afisat pe ecran, dificultatea creste odata cu scorul la intervale fixe. Am adaugat si un meniu de inceput cu o imagine de background reprezentativa. Ca jocul sa para realist am facut ca toate obiectele sa vina spre noi, acest lucru dand impresia ca ne aflam in miscare. A trebuit adaptat pe 3D algoritmul de miscare a masinilor/ copacilor, cel de coliziune si random respawn a masinilor.

Contributii individuale

* Alecsandru: Adaptare algoritmi pe 3D
* Bogdan: Design (creare obiecte 3D, strada, etc)
* Mihai: Adapatare meniu, Calculare Coordonate

Resurse utilizate

* Materialele si rezolvarile de laborator

Filmare

<https://drive.google.com/drive/folders/1SFHLh7kyyCiC87o-FC1TKzVpbxOJjZaL>

**Cod sursa**

[**https://github.com/alecsbraga/OpenGL-Crash\_Mission\_3D**](https://github.com/alecsbraga/OpenGL-Crash_Mission_3D)

#include <iostream>

#include <fstream>

#include<gl/freeglut.h>

#include<math.h>

#include "SOIL.h"

using namespace std;

// actual vector representing the camera's direction

float lx = 0.0000001f;

// XZ position of the camera

float x = 0.0f, zAdv = -50.0f, xAdv;

int parte;

bool over = false;

int score = 0;

bool changeScore = false;

int highscore;

int pct = 1000;

bool gameover = false;

bool inMenu = true;

bool inEndScreen = false;

float xs = -20.0, ys = 30.0, zs = 5.0, ts = 0.0;

void delay(float secs)

{

float end = clock() / CLOCKS\_PER\_SEC + secs;

while ((clock() / CLOCKS\_PER\_SEC) < end);

}

void Initialize()

{

glClearColor(0.67, 0.84, 0.89, 0.0);

}

void changeSize(int w, int h)

{

// Prevent a divide by zero, when window is too short

// (you cant make a window of zero width).

if (h == 0)

h = 1;

float ratio = w \* 1.0 / h;

// Use the Projection Matrix

glMatrixMode(GL\_PROJECTION);

// Reset Matrix

glLoadIdentity();

// Set the viewport to be the entire window

glViewport(0, 0, w, h);

// Set the correct perspective.

gluPerspective(45.0f, ratio, 0.1f, 100.0f);

// Get Back to the Modelview

glMatrixMode(GL\_MODELVIEW);

}

void lumina()

{

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glShadeModel(GL\_SMOOTH);

// initializari si proprietatile universale

GLfloat alphaAmb = 1.0;

GLfloat alphaDifuz = 1.0;

GLfloat difuzLumina[4];

GLfloat unghiCutOff = 180.0;

ts = 0.0;

alphaAmb = 1;

alphaDifuz = 0.3;

difuzLumina[0] = alphaDifuz; difuzLumina[1] = alphaDifuz; difuzLumina[2] = alphaDifuz; difuzLumina[3] = 0;

unghiCutOff = 180;

// Sursa de lumina

GLfloat ambientLumina[] = { alphaAmb, alphaAmb, alphaAmb, alphaAmb };

GLfloat pozitial0[] = { xs, ys, zs, ts };

GLfloat directieSpot[] = { 0, 0, -1, 0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, pozitial0);

//glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambientLumina);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, difuzLumina);

//glLightfv(GL\_LIGHT0, GL\_SPOT\_DIRECTION, directieSpot);

//glLightf(GL\_LIGHT0, GL\_SPOT\_CUTOFF, unghiCutOff);

//glLightf(GL\_LIGHT0, GL\_SPOT\_EXPONENT, 3.0);

//glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.1);

//glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.1);

//glLightf(GL\_LIGHT0, GL\_QUADRATIC\_ATTENUATION, 0.2);

};

void Copaci() {

//Trunchi

GLfloat maro[] = { 0.5f, 0.3f, 0.0f };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, maro);

GLUquadricObj\* cylinder;

glPushMatrix();

glTranslatef(0, 0.5, 0.0);

glRotated(270, 0.5, 0, 0);

cylinder = gluNewQuadric();

gluQuadricDrawStyle(cylinder, GLU\_FILL);

gluCylinder(cylinder, 0.25, 0.25, 2.0, 15, 7.5);

glPopMatrix();

GLfloat verde[] = { 0, 0.5, 0.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, verde);

// Draw Body

glTranslatef(0.0f, 2.27f, 0.0f);

glutSolidSphere(0.75f, 25, 25);

}

void gameplay() {

//cout << over;

if ((xAdv + 0.5f >= x && xAdv - 0.5 <= x) && zAdv > 4.0f)

{

over = true;

}

else {

switch (parte)

{

case 0:

xAdv = 0.0f;

break;

case 1:

xAdv = -2.0f;

break;

case 2:

xAdv = 2.0f;

break;

}

float dificultate = 0.1f;

if (zAdv >= -90.0f)

zAdv = zAdv + dificultate;

if (zAdv > 5.0f) {

parte = rand() % 3;

zAdv = -90.0f;

score += 100;

cout << "Score: " << score << endl;

changeScore = true;

}

if (score >= pct && pct <= 15000)

{

dificultate += 0.1;

pct += 1000;

}

if (highscore < score) {

highscore = score;

}

}

}

GLfloat coloareText[] = { 0.0, 0.0, 0.0 };

void RenderString(float x, float y,float z, void\* font, const unsigned char\* string)

{

glPushAttrib(GL\_CURRENT\_BIT);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, coloareText);

glRasterPos3f(x, y, z);

glutBitmapString(font, string);

glPopAttrib();

}

void LoadTexture(void)

{

GLuint texture;

glGenTextures(1, &texture);

glBindTexture(GL\_TEXTURE\_2D, texture);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP); // Set texture wrapping to GL\_REPEAT

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("1.jpg", &width, &height, 0, SOIL\_LOAD\_RGB);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image);

// SOIL\_free\_image\_data(image);

// glBindTexture(GL\_TEXTURE\_2D, 0);

}

void startMenu(void) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

gluLookAt(x, 1.0f, 5.0f,

x + lx, 1.0f, 4.0f,

0.0f, 1.0f, 0.0f);

glEnable(GL\_TEXTURE\_2D);

LoadTexture();

glBegin(GL\_QUADS);

glTexCoord2f(1.0, 1.0);glVertex3f(-2.5, -3.0, -0.1);

glTexCoord2f(1.0, 0.0);glVertex3f(-2.5, 4.0, -0.1);

glTexCoord2f(0.0, 0.0);glVertex3f(2.5, 4.0, -0.1);

glTexCoord2f(0.0, 1.0);glVertex3f(2.5, -3.0, -0.1);

glEnd();

glDisable(GL\_TEXTURE\_2D);

glColor3f(0.0, 0.0, 0.0);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, coloareText);

glRasterPos3f(-0.5, 1.5, 0);

char message[] = "Crashing mission!";

for (int i = 0; i < strlen(message); i++)

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message[i]);

glRasterPos3f(-1, 1.3, 0);

char message2[] = "A game about cars, crashes and you!";

for (int i = 0; i < strlen(message2); i++)

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message2[i]);

glRasterPos3f(-0.75, 1.1, 0);

char message3[] = "Press X to start the game";

for (int i = 0; i < strlen(message3); i++) {

glColor3f(0, 0, 0);

if (message3[i] == 'X') {

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message3[i]);

}

else glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message3[i]);

}

glutSwapBuffers();

}

void endScreen(void) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

gluLookAt(x, 1.0f, 5.0f,

x + lx, 1.0f, 4.0f,

0.0f, 1.0f, 0.0f);

glColor3f(1.0, 0.0, 0.0);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, coloareText);

glRasterPos3f(-1.55 + x, 1.9, 0);

char message4[] = "You crashed, too bad";

for (int i = 0; i < strlen(message4); i++)

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message4[i]);

glRasterPos3f(-1.75 + x, 1.6, 0);

char message3[] = "Press C to close the game";

for (int i = 0; i < strlen(message3); i++) {

if (message3[i] == 'X') {

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message3[i]);

}

else glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, message3[i]);

}

glutSwapBuffers();

}

void renderScene(void) {

// Clear Color and Depth Buffers

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// Reset transformations

glLoadIdentity();

// Set the camera

gluLookAt(x, 1.0f, 5.0f,

x + lx, 1.0f, 4.0f,

0.0f, 1.0f, 0.0f);

char\* text = new char[200];

text = (char\*)"Fereste-te de masinile!";

char scorText[10 + sizeof(char)];

char highscoreText[10 + sizeof(char)];

sprintf\_s(scorText, "%d", score);

sprintf\_s(highscoreText, "%d", highscore);

//cout << scorText;

//strcat(text, scorText);

RenderString(-0.5f + x, 2.9f, 0.0f, GLUT\_BITMAP\_9\_BY\_15, (const unsigned char\*)text);

RenderString(-2.0f + x, 2.90f, 0.0f, GLUT\_BITMAP\_9\_BY\_15, (const unsigned char\*)"Scor: ");

RenderString(-1.5f + x, 2.90f, 0.0f, GLUT\_BITMAP\_9\_BY\_15, (const unsigned char\*)scorText);

RenderString(-2.0f + x, 2.70f, 0.0f, GLUT\_BITMAP\_9\_BY\_15, (const unsigned char\*)"Highscore: ");

RenderString(-1.1f + x, 2.70f, 0.0f, GLUT\_BITMAP\_9\_BY\_15, (const unsigned char\*)highscoreText);

// Draw iarba

GLfloat coloareIarba[] = { 0.0, 0.3, 0.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, coloareIarba);

glBegin(GL\_QUADS);

glVertex3f(-100.0f, 0.0f, -100.0f);

glVertex3f(-100.0f, 0.0f, 100.0f);

glVertex3f(100.0f, 0.0f, 100.0f);

glVertex3f(100.0f, 0.0f, -100.0f);

glEnd();

//Draw sosea

GLfloat coloareSoasea[] = { 0.01, 0.01, 0.01 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, coloareSoasea);

glBegin(GL\_QUADS);

glVertex3f(-3.0f, 0.2f, -100.0f);

glVertex3f(-3.0f, 0.2f, 100.0f);

glVertex3f(3.0f, 0.2f, 100.0f);

glVertex3f(3.0f, 0.2f, -100.0f);

glEnd();

//Draw linii-sosea

GLfloat liniieSoasea[] = { 1.0, 1.0, 1.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, liniieSoasea);

glBegin(GL\_QUADS);

glVertex3f(-3.0f, 0.21f, -100.0f);

glVertex3f(-3.0f, 0.21f, 100.0f);

glVertex3f(-2.9f, 0.21f, 100.0f);

glVertex3f(-2.9f, 0.21f, -100.0f);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(3.0f, 0.21f, -100.0f);

glVertex3f(3.0f, 0.21f, 100.0f);

glVertex3f(2.9f, 0.21f, 100.0f);

glVertex3f(2.9f, 0.21f, -100.0f);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(-1.0f, 0.21f, -100.0f);

glVertex3f(-1.0f, 0.21f, 100.0f);

glVertex3f(-0.9f, 0.21f, 100.0f);

glVertex3f(-0.9f, 0.21f, -100.0f);

glEnd();

glBegin(GL\_QUADS);

glVertex3f(1.0f, 0.21f, -100.0f);

glVertex3f(1.0f, 0.21f, 100.0f);

glVertex3f(0.9f, 0.21f, 100.0f);

glVertex3f(0.9f, 0.21f, -100.0f);

glEnd();

// Draw copaci

for (int j = -50; j < 1; j++) {

glPushMatrix();

glTranslatef(-3.3, -0.3, 100 + zAdv + j \* 5.0);

Copaci();

glPopMatrix();

}

for (int j = -50; j < 1; j++) {

glPushMatrix();

glTranslatef(3.3, -0.3, 100 + zAdv + j \* 5.0);

Copaci();

glPopMatrix();

}

//Draw adversar

glPushMatrix();

glTranslatef(xAdv, 0.95f, zAdv);

//Draw corpul masinii

glBegin(GL\_POLYGON);

GLfloat culoareMasina[] = { 0.0, 0.0, 1.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, culoareMasina);

//FRONT

glVertex3f(0.5, -0.25, 1.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, -0.25, 1.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.5, -0.25, 2.5);

glVertex3f(0.5, -0.25, 1.5);

glVertex3f(-0.5, -0.25, 1.5);

glVertex3f(-0.5, -0.25, 2.5);

glEnd();

glBegin(GL\_POLYGON);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(0.5, -0.25, 2.5);

glVertex3f(-0.5, -0.25, 2.5);

glVertex3f(-0.5, -0.5, 2.5);

glEnd();

//BACK

glBegin(GL\_POLYGON);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, -0.5, -0.5);

// RIGHT

glBegin(GL\_POLYGON);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(0.5, -0.25, 1.5);

glVertex3f(0.5, -0.25, 2.5);

glVertex3f(0.5, -0.5, 2.5);

glEnd();

//LEFT

glBegin(GL\_POLYGON);

glVertex3f(-0.5, -0.5, 2.5);

glVertex3f(-0.5, -0.25, 2.5);

glVertex3f(-0.5, -0.25, 1.5);

glVertex3f(-0.5, 0.5, 0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, -0.5, -0.5);

glEnd();

//TOP

glBegin(GL\_POLYGON);

glVertex3f(0.5, 0.5, 0.5);

glVertex3f(0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, -0.5);

glVertex3f(-0.5, 0.5, 0.5);

glEnd();

//BOTTOM

glBegin(GL\_POLYGON);

glVertex3f(0.5, -0.5, -0.5);

glVertex3f(0.5, -0.5, 2.5);

glVertex3f(-0.5, -0.5, 2.5);

glVertex3f(-0.5, -0.5, -0.5);

glEnd();

glFlush();

//Faruri

GLfloat culoareFaruri[] = { 1.0, 1.0, 0.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, culoareFaruri);

glPushMatrix();

glTranslatef(0.25, -0.3f, 2.5);

glutSolidSphere(0.1f, 50, 50);

glPopMatrix();

glPushMatrix();

glTranslatef(-0.25, -0.3f, 2.5);

glutSolidSphere(0.1f, 50, 50);

glPopMatrix();

//Roti

GLfloat culoareRoti[] = { 0.0, 0.0, 0.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, culoareRoti);

glPushMatrix();

glTranslatef(-0.5, -0.5f, 2.0);

glutSolidSphere(0.2f, 50, 50);

glPopMatrix();

glPushMatrix();

glTranslatef(-0.5, -0.5f, -0.25);

glutSolidSphere(0.2f, 50, 50);

glPopMatrix();

glPushMatrix();

glTranslatef(0.5, -0.5f, 2.0);

glutSolidSphere(0.2f, 50, 50);

glPopMatrix();

glPushMatrix();

glTranslatef(0.5, -0.5f, -0.25);

glutSolidSphere(0.2f, 50, 50);

glPopMatrix();

//parbriz

glBegin(GL\_POLYGON);

GLfloat culoareParbriz[] = { 0.0, 0.76, 1.0 };

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, culoareParbriz);

glVertex3f(0.4, -0.15, 1.5);

glVertex3f(0.4, 0.47, 0.5);

glVertex3f(-0.4, 0.47, 0.5);

glVertex3f(-0.4, -0.15, 1.5);

glEnd();

glPopMatrix();

if (over == true) {

delay(1);

inEndScreen = true;

glutDisplayFunc(endScreen);

if (gameover == false) {

ofstream g("Highscore.txt");

g << highscore;

g.close();

gameover = true;

//exit(0);

}

}

lumina();

gameplay();

glutPostRedisplay();

glutSwapBuffers();

glFlush();

}

void keys(unsigned char key, int x, int y) {

switch (key) {

case 'x':

if (inMenu) {

inMenu = false;

glutDisplayFunc(renderScene);

}

break;

case 'c':

if (inEndScreen)

exit(0);

break;

}

glutPostRedisplay();

}

void processSpecialKeys(int key, int xx, int yy) {

float fraction = 1000000.0f;

switch (key) {

case GLUT\_KEY\_RIGHT:

if (!inMenu){

x += lx \* fraction;

if (x > 2)

x = 2;

}

break;

case GLUT\_KEY\_LEFT:

if (!inMenu) {

x -= lx \* fraction;

if (x < -2)

x = -2;

}

}

}

int main(int argc, char\*\* argv) {

// init GLUT and create window

ifstream f("Highscore.txt");

f >> highscore;

f.close();

cout << highscore;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(100, 100);

glutInitWindowSize(600, 600);

glutCreateWindow("Crash Mission");

Initialize();

// register callbacks

glutDisplayFunc(startMenu);

//glutDisplayFunc(renderScene);

glutReshapeFunc(changeSize);

glutSpecialFunc(processSpecialKeys);

glutKeyboardFunc(keys);

// OpenGL init

glEnable(GL\_DEPTH\_TEST);

// enter GLUT event processing cycle

glutMainLoop();

return 1;

}