Autori: Popescu Florin-Daniel, Tudor Nicu Cornel
Grupa: 363

Forza Horizon 6

1. Descriere

În proiectul nostru am creat un mini-game cu mașini 2D care se bazează pe evitarea obstacolelor întâlnite pe drum. Am folosit tranziții pentru a oferi o imagine dinamică jucătorului pentru a crea o experiență de joc fluidă si realistă, iar folosind diverse texturi și culori am reușit să creăm un mediu vizual atractiv și plin de culoare.

2. Aspecte punctuale

- ✓ Utilizarea primitivelor
  - Am folosit 2 poligoane pentru a desena iarba de sus și de jos.
  - Am utilizat un poligon pentru a desena cerul.
  - Pentru soare am folosit o funcție care desenează un poligon cu 50 de laturi pentru a-i oferi formă de cerc.
  - Pentru lună este același principiu ca la soare, iar pentru a desena craterele din lună am folosit mai multe cercuri puse unul peste altul, de culori diferite.
  - Modelele de nori (1, 2, 3) sunt formațiuni de cercuri aliniate oferind forma unor nori.
  - Pentru liniile discontinue am folosit o linie de grosime 5 aplicându-i funcția GL_LINE_STIPPLE.
  - Pentru liniile continue de sus și de jos am folosit 2 poligoane de 4 laturi.
  - Pentru modelul de copac 1 care este un brad, am folosit un poligon de 4 laturi pentru a desena trunchiul, apoi 3 triunghiuri puse unul peste altul pentru a-i oferi forma specifică unui brad.
  - Pentru modelul de copac 2 care este un fag, am folosit un poligon de 4 laturi pentru a desena trunchiul, apoi 5 cercuri aliniate pentru a desena frunzele.
  - Pentru lampă am folosit un poligon cu 4 laturi pentru a desena baza acestuia, iar apoi un cerc pentru a desena becul.
  - Pentru modelul de mașină am folosit o combinație de poligoane pentru a desena caroseria, stopurile și farurile iar pentru roți o combinație de cercuri.
- ✓ Utilizarea transformărilor
  - Am folosit transformări (glTranslate3f), pentru a poziționa elementele așa cum ne-am dorit.

- ✓ Utilizarea stivelor de matrice
  - Pentru stivele de matrice am aplicat Push și Pop pe toate elementele dinamice pentru a nu interfera cu desenele statice. Un exemplu ar fi atunci când am desenat liniile discontinue, delimitările laterale aveau o mișcare sacadată dacă nu foloseam stivele de matrice. Acest lucru se aplică atât brazilor cât și fagilor deoarece dacă nu aplicam stive de matrice pe aceștia ,erau pomi care rămâneau statici și pomi care se mișcau.
  - Pentru nori am folosit stive de matrice pentru a crea un aspect de sine stătător, iar pentru lămpi se aplică același principiu.
- ✓ Input interactiv
  - Pentru tasta 'n' se activează un set de culori și de funcții de desenare pentru a crea un scenariu de noapte.
  - Pentru tasta 'z' (valori predefinite) se activează un set de culori și de funcții de desenare pentru a crea un scenariu de zi.
  - Am folosit citirea și scrierea din fișiere pentru a afișa în consolă textul „New Highscore!".
  - Am modificat textul inițial „ Depășește mașinile!" în textul „Watch Out!" și l-am făcut să dispară după 3 secunde de când jocul a pornit.

3. Originalitate

Consider ca proiectul este original și elaborat deoarece chiar dacă este un proiect 2D el este gândit în 3D, acest lucru reiese din distanța observatorului față de elementele din jur, de exemplu, nori, copaci, felinare. Am creat o mașină cu un design unic și am reușit să introducem schimbarea zi/noapte. Dinamica peisajului realist.

4. Contribuții individuale

Contribuții comune: idee proiect(perspectiva 3D), elemente interactive(zi/noapte), Florin a desenat copaci iar Nicu i-a grupat pentru a crea un aspect ambiental, îngustarea benzilor și crearea liniilor discontinue, am schimbat fundalul în culoarea gri.
Contribuții Popescu Florin: M-am ocupat în special de partea de design unde am implementat următoarele: modelul de brad, fag, cele 3 modele de nori, mașină, soarele, cerul, lampă.
Contribuții Tudor Nicu: M-am ocupat de partea de dinamică a proiectului ceea ce înseamnă transformări și stive de matrice: mișcarea formațiunilor de nori, mișcarea benzilor, mișcarea lămpilor, mișcarea pomilor.

Autori: Popescu Florin-Daniel, Tudor Nicu Cornel
Grupa: 363

Am creat de asemenea și câteva părți de design precum: luna și adăugarea unor noi formațiuni de copaci.

5. **Resurse utilizate**

Ne-am inspirat pentru a crea modelul de nori din acest videoclip:
https://www.youtube.com/watch?v=Z3IdSTlzPMc&list=PLXY4_qxp8fUeIMnZIisboZGifEQt0rCAk&ab_channel=AstrasoftAcademy
si am contribuit in plus prin adăugarea mișcării acestora si schimbarea culorii.

6. **Link Google Drive**

https://drive.google.com/file/d/1xHEU3XTjA-tzzbsSY-SblsQ8n7D3wjpX/view?usp=share_link

7. **Cod Sursa**

```
#include <iostream>
#include<windows.h>
#include <GL/freeglut.h>
#include <fstream>

using namespace std;


#define PI 3.1416

GLdouble left_m = -100.0;
GLdouble right_m = 700.0;
GLdouble bottom_m = -140.0;
GLdouble top_m = 460.0;

GLfloat posNor1 = 520;
GLfloat posNor2 = 700;
GLfloat posNor3 = 550;
GLfloat posNor4 = 1000;
GLfloat posNor5 = 1500;

int startTime = 0;
int color = 0;
```

```
int highScore;


float miscareLinie1 = 0;
float miscareLinie2 = 0;

float x1Jos = -1500;
float x2Jos = 1500;
float yJos = 20;

float x1Sus = -1500;
float x2Sus = 1500;
float ySus = 120;

float posXCopaci[15] = { 10, -200, 210, 260, 60, -40, 160, -150, -250, -300,
-350, 310, 360, 410 };
float posYCopaci = 20;
float posFormC1 = 0.0;
float posFormC2 = 0.0;

float xL = 70;
float yL = 240;

float yL2 = -130;

float becX = 75;
float becY = 275;

float becY2 = -100;

float posFormLampa1 = 0.0;
float posFormLampa2 = 0.0;
float posFormLampa3 = 0.0;
float posFormLampa4 = 0.0;

double ok = 1;
double j = 0.0;
double i = 0.0;
double contor = 0;
```

```
double loc_vert = 800;
int vector[3] = { -30, 70, 170 };
double height = vector[rand() % 3];
int score = 0;
double timp = 0.5;
int pct = 1000;
double rsj, rdj, rss, rds = 0;

void init(void)
{
  glClearColor(0.5, 0.5, 0.5, 0.0);
  glMatrixMode(GL_PROJECTION);
  glOrtho(left_m, right_m, bottom_m, top_m, -1.0, 1.0);
}

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

  glColor3f(0.0f, 0.0f, 0.0f);
  glRasterPos2f(x, y);
  glutBitmapString(font, string);
}

void startgame(void)
{

  if (height != j || (loc_vert > 90 || loc_vert < -90))
  {

        if (i < -380)
        {
              i = 0;
        }
        i = i - 2 * timp;

        loc_vert -= timp;

        if (loc_vert < -150)
        {
              score += 100;
```

```
                height = vector[rand() % 3];
                cout << "Score:  " << score << endl;
                loc_vert = 800;
        }

        if (score >= pct && pct <= 15000)
        {
                timp += 0.4;
                pct += 1000;
        }

        glutPostRedisplay();
    }
    else {
        ok = 0;
    }
}

void high(void)
{
    ifstream f("highScore.out");
    if (f.is_open())
    {
        f >> highScore;
        f.close();
    }
    ofstream g("highScore.out");
    if (score > highScore) {
        cout << "New High Score!" << endl;

        if (g.is_open()) {

            g << score << endl;
            g.close();
        }

    }
    else
    {
        if (g.is_open()) {
```

```
                g << highScore << std::endl;
                g.close();
            }
        }
    }

    void circle(GLdouble rad)
    {
        GLint points = 50;
        GLdouble delTheta = (2.0 * PI) / (GLdouble)points;
        GLdouble theta = 0.0;

        glBegin(GL_POLYGON);
        {
            for (i = 0; i <= 50; i++, theta += delTheta)
            {
                glVertex2f(rad * cos(theta), rad * sin(theta));
            }
        }
        glEnd();
    }

    void linieDiscontinua(float x1, float x2, float y) {
        glLineWidth(3);
        if (color == 0)
            glColor3f(1.0, 1.0, 0.0);
        else
            glColor3f(1.0, 1.0, 0.7);
        glEnable(GL_LINE_STIPPLE);
        glLineStipple(5, 0x00FF);

        glPushMatrix();

        glBegin(GL_LINES);
        glVertex2i(x1, y);
        glVertex2i(x2, y);
        glEnd();

        glDisable(GL_LINE_STIPPLE);
    }
```

Autori: Popescu Florin-Daniel, Tudor Nicu Cornel
Grupa: 363

```
void linieJ(float x1, float x2, float y)
{
  glPushMatrix();
  glTranslatef(miscareLinie1, 0, 0);
  linieDiscontinua(x1, x2, y);
  glPopMatrix();
}

void linieS(float x1, float x2, float y)
{
  glPushMatrix();
  glTranslatef(miscareLinie2, 0, 0);
  linieDiscontinua(x1, x2, y);
  glPopMatrix();
}

void miscare_linii(int value) {

  miscareLinie1 -= 1;

  if (miscareLinie1 < -200)
  {

        linieJ(x1Jos + miscareLinie1, x2Jos - miscareLinie1, yJos);
        miscareLinie1 = 1000;
  }

  miscareLinie2 -= 1;

  if (miscareLinie2 < -200)
  {
        linieS(x1Sus + miscareLinie2, x2Sus - miscareLinie2, ySus);
        miscareLinie2 = 1000;
  }

  glutPostRedisplay();
  glutTimerFunc(2, miscare_linii, 0);
}
```

```
void modelCopac1()
{
  if (color == 0)
        glColor3f(0.38f, 0.19f, 0.0f);
  else
        glColor3f(0.38f, 0.19f, 0.2f);

  glBegin(GL_POLYGON);
  glVertex2f(200, 260);
  glVertex2f(200, 245);
  glVertex2f(210, 245);
  glVertex2f(210, 260);
  glEnd();

  if (color == 0)
        glColor3f(0.0, 0.43, 0.0);
  else
        glColor3f(0.0, 0.2, 0.0);

  glBegin(GL_POLYGON);
  glVertex2f(180, 260);
  glVertex2f(205, 280);
  glVertex2f(230, 260);
  glEnd();

  glBegin(GL_POLYGON);
  glVertex2f(180, 270);
  glVertex2f(205, 290);
  glVertex2f(230, 270);
  glEnd();

  glBegin(GL_POLYGON);
  glVertex2f(180, 280);
  glVertex2f(205, 310);
  glVertex2f(230, 280);
  glEnd();

}

void modelCopac2()
```

```
{
  if (color == 0)
        glColor3f(0.38f, 0.19f, 0.0f);
  else
        glColor3f(0.38f, 0.19f, 0.2f);
  glBegin(GL_POLYGON);
  glVertex2f(200, 260);
  glVertex2f(200, 245);
  glVertex2f(210, 245);
  glVertex2f(210, 260);
  glEnd();

  if (color == 0)
        glColor3f(0.0, 0.5, 0.0);
  else
        glColor3f(0.15, 0.3, 0.2);

  glPushMatrix();
  glTranslatef(210, 268, 0);
  circle(10);
  glPopMatrix();

  glPushMatrix();
  glTranslatef(200, 268, 0);
  circle(10);
  glPopMatrix();

  glPushMatrix();
  glTranslatef(195, 280, 0);
  circle(10);
  glPopMatrix();

  glPushMatrix();
  glTranslatef(205, 290, 0);
  circle(10);
  glPopMatrix();

  glPushMatrix();
  glTranslatef(213, 283, 0);
  circle(10);
```

```
    glPopMatrix();

}

void setCopaci(float x, float y)
{
  glPushMatrix();
  glTranslatef(x, y, 0.0);
  modelCopac1();
  glPopMatrix();

  glPushMatrix();
  glTranslatef(x + 30, y + 10, 0.0);
  modelCopac2();
  glPopMatrix();

  glPushMatrix();
  glTranslatef(x + 50, y + 15, 0.0);
  modelCopac2();
  glPopMatrix();

  glPushMatrix();
  glTranslatef(x + 90, y + 9, 0.0);
  modelCopac1();
  glPopMatrix();
}

void copaci(float posX, float posY, float& value)
{
  glPushMatrix();
  glTranslatef(value, 0.0, 0.0);
  setCopaci(posX, posY);
  glPopMatrix();
}

void miscare_copaci(int value)
{
  posFormC1 -= 2;
  if (posFormC1 < -1200)
  {
```

```
        glPushMatrix();
        copaci(posXCopaci[0] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[1] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[2] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[3] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[4] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[5] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[6] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[7] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[8] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[9] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[10] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[11] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[12] + posFormC1, posYCopaci, posFormC1);
        copaci(posXCopaci[13] + posFormC1, posYCopaci, posFormC1);
        glPopMatrix();
        posFormC1 = 1000;
}

posFormC2 -= 2;
if (posFormC2 < -1000)
{
        glPushMatrix();
        copaci(posXCopaci[0] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[1] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[2] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[3] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[4] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[5] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[6] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[7] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[8] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[9] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[10] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[11] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[12] + posFormC2, posYCopaci, posFormC2);
        copaci(posXCopaci[13] + posFormC2, posYCopaci, posFormC2);
        glPopMatrix();
        posFormC2 = 850;
}
```

```
    glutPostRedisplay();
    glutTimerFunc(10, miscare_copaci, 0);
}


void model_nor1() {

    if (color == 0)
            glColor3f(1.0, 1.0, 1.0);
    else
            glColor3f(0.8, 0.9, 1.0);


    glPushMatrix();
    glTranslatef(200, 380, 0);
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(220, 390, 0);
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(240, 400, 0);
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(260, 390, 0);
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(280, 380, 0);
```

```
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(260, 373, 0);
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(220, 373, 0);
    circle(15);
    glPopMatrix();


    glPushMatrix();
    glTranslatef(240, 373, 0);
    circle(15);
    glPopMatrix();
  }

  void model_nor2() {
    if (color == 0)
        glColor3f(1.0, 1.0, 1.0);
    else
        glColor3f(0.8, 0.9, 1.0);

    glPushMatrix();
    glTranslatef(305, 205, 0);
    circle(10);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(320, 210, 0);
    circle(15);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(334, 207, 0);
```

```
    circle(10);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(320, 207, 0);
    circle(10);
    glPopMatrix();
  }

  void model_nor3() {
    if (color == 0)
        glColor3f(1.0, 1.0, 1.0);
    else
        glColor3f(0.8, 0.9, 1.0);

    glPushMatrix();
    glTranslatef(300, 200, 0);
    circle(15);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(320, 210, 0);
    circle(15);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(340, 220, 0);
    circle(16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(360, 210, 0);
    circle(15);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(380, 200, 0);
    circle(15);
    glPopMatrix();
```

```
    glPushMatrix();
    glTranslatef(360, 190, 0);
    circle(20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(320, 190, 0);
    circle(20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(340, 190, 0);
    circle(20);
    glPopMatrix();
}

void nor1() {
    glPushMatrix();
    glTranslatef(posNor1, -10, 0);
    model_nor1();
    glPopMatrix();
}

void nor2() {
    glPushMatrix();
    glTranslatef(posNor2, 45, 0);
    model_nor1();
    glPopMatrix();
}

void nor3()
{
    glPushMatrix();
    glTranslatef(posNor3, 120, 0);
    model_nor2();
    glPopMatrix();
}

void nor4()
{
```

```
    glPushMatrix();
    glTranslatef(posNor4, 120, 0);
    model_nor3();
    glPopMatrix();
}

void nor5()
{
    glPushMatrix();
    glTranslatef(posNor5, 120, 0);
    model_nor2();
    glPopMatrix();
}

void sun() {
    if (color == 0)
            glColor3f(1.0, 1.0, 0.0);


    glPushMatrix();
    glTranslatef(420, 390, 0);
    circle(35);
    glPopMatrix();
}

void moon() {
    glColor3f(0.9, 0.9, 0.9);
    glPushMatrix();
    glTranslatef(420, 390, 0);
    circle(35);
    glPopMatrix();

    glColor3f(0.7, 0.7, 0.7);
    glPushMatrix();
    glTranslatef(420, 410, 0);
    circle(10);
    glPopMatrix();

    glColor3f(0.65, 0.65, 0.65);
    glPushMatrix();
```

```
glTranslatef(420, 410, 0);
circle(7);
glPopMatrix();

glColor3f(0.7, 0.7, 0.7);
glPushMatrix();
glTranslatef(400, 370, 0);
circle(6);
glPopMatrix();

glColor3f(0.65, 0.65, 0.65);
glPushMatrix();
glTranslatef(400, 370, 0);
circle(4);
glPopMatrix();

glColor3f(0.7, 0.7, 0.7);
glPushMatrix();
glTranslatef(430, 370, 0);
circle(8);
glPopMatrix();

glColor3f(0.65, 0.65, 0.65);
glPushMatrix();
glTranslatef(430, 370, 0);
circle(5);
glPopMatrix();

glColor3f(0.7, 0.7, 0.7);
glPushMatrix();
glTranslatef(430, 380, 0);
circle(5);
glPopMatrix();

glColor3f(0.65, 0.65, 0.65);
glPushMatrix();
glTranslatef(430, 380, 0);
circle(3);
glPopMatrix();
```

```
    glColor3f(0.7, 0.7, 0.7);
    glPushMatrix();
    glTranslatef(400, 390, 0);
    circle(11);
    glPopMatrix();

    glColor3f(0.65, 0.65, 0.65);
    glPushMatrix();
    glTranslatef(400, 390, 0);
    circle(8);
    glPopMatrix();
  }

  void iarba() {
    if (color == 0)
        glColor3f(0.55, 0.788, 0.451);
    else
        glColor3f(0.0, 0.2, 0.4);

    // Iarba de jos
    glBegin(GL_POLYGON);
    glVertex2i(-100, -140);// Stanga jos
    glVertex2i(700, -140); // Dreapta jos
    glVertex2i(700, -80);  // Dreapta sus
    glVertex2i(-100, -80); // Stanga sus
    glEnd();

    // Iarba de sus
    glBegin(GL_POLYGON);
    glVertex2i(-100, 240);// Stanga jos
    glVertex2i(700, 240); // Dreapta jos
    glVertex2i(700, 280);  // Dreapta sus
    glVertex2i(-100, 280); // Stanga sus
    glEnd();
  }

  void lampa(float x1, float y1, float xC, float yC)
  {
    glColor3f(0.5, 0.5, 0.5);
    glBegin(GL_POLYGON);
```

```
    glVertex2f(x1, y1 + 30);
    glVertex2f(x1, y1);
    glVertex2f(x1 + 10, y1);
    glVertex2f(x1 + 10, y1 + 30);
    glEnd();

    if (color == 1) {
        glColor3f(1.0, 1.0, 0.0);
        glPushMatrix();
        glTranslatef(xC, yC, 0.0);
        circle(10);
        glPopMatrix();
    }
    else {
        glColor3f(0.9, 0.9, 0.9);
        glPushMatrix();
        glTranslatef(xC, yC, 0.0);
        circle(8);
        glPopMatrix();
    }


}

void lampi(float x1, float y1, float xC, float yC, float& value)
{
  glPushMatrix();
  glTranslatef(value, 0.0, 0.0);
  lampa(x1, y1, xC, yC);
  glPopMatrix();
}

void miscare_lampi(int value)
{
  posFormLampa1 -= 1;

  if (posFormLampa1 < -550)
  {
        lampi(xL, yL, becX, becY, posFormLampa1);
        posFormLampa1 = 600;
```

```
    }

    posFormLampa2 -= 1;

    if (posFormLampa2 < -550)
    {
        lampi(xL + 380, yL, becX + 380, becY, posFormLampa2);
        posFormLampa2 = 600;
    }

    posFormLampa3 -= 1;

    if (posFormLampa3 < -550)
    {
        lampi(xL + 380, yL2, becX + 380, becY2, posFormLampa3);
        posFormLampa3 = 600;
    }

    posFormLampa4 -= 1;

    if (posFormLampa4 < -550)
    {
        lampi(xL, yL2, becX, becY2, posFormLampa4);
        posFormLampa4 = 600;
    }

    glutPostRedisplay();
    glutTimerFunc(2, miscare_lampi, 0);
}

void cer() {
    if (color == 0)
        glColor3f(0.196078, 0.6, 0.8);
    else
        glColor3f(0.0, 0.0, 0.2);
    //Cod adaugat de noi schimbam culoarea cerului
    // Cerul
    glBegin(GL_POLYGON);
    glVertex2i(-100, 280);// Stanga jos
    glVertex2i(700, 280); // Dreapta jos
```

```
    glVertex2i(700, 460); // Dreapta sus
    glVertex2i(-100, 460);// Stanga sus
    glEnd();
}

void masina_principala()
{
    //desenam masina
    glPushMatrix();
    glTranslated(0.0, j, 0.0);

    if (color == 0)
        glColor3ub(255, 0, 0);
    else
        glColor3ub(139, 0, 0);

    glBegin(GL_POLYGON);
    glVertex2d(-45, 0);
    glVertex2d(50, 0);
    glVertex2d(40, -20);
    glVertex2d(-45, -20);
    glEnd();

    if (color == 0)
        glColor3ub(255, 0, 0);
    else
        glColor3ub(139, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2d(-45, 0);
    glVertex2d(25, 0);
    glVertex2d(15, 20);
    glVertex2d(-25, 20);
    glEnd();

    //ferestre masina
    glColor3ub(220, 220, 220);
    glBegin(GL_POLYGON);
    glVertex2d(-30, 0);
    glVertex2d(-7, 0);
    glVertex2d(-7, 10);
```

```
        glVertex2d(-25, 10);
        glEnd();

        glColor3ub(255, 220, 220);
        glBegin(GL_POLYGON);
        glVertex2d(-2, 0);
        glVertex2d(20, 0);
        glVertex2d(15, 10);
        glVertex2d(-2, 10);
        glEnd();

        //roata masinii
        glColor3ub(0, 0, 0);
        glPushMatrix();
        glTranslatef(-30, -20, 0.0);
        circle(10);
        glPopMatrix();

        glPushMatrix();
        glTranslatef(20, -20, 0.0);
        circle(10);
        glPopMatrix();

        glColor3ub(245, 245, 245);

        glPushMatrix();
        glTranslatef(-30, -20, 0.0);
        circle(6);
        glPopMatrix();

        glPushMatrix();
        glTranslatef(20, -20, 0.0);
        circle(6);
        glPopMatrix();

        //farurile masinii
        if (color == 1)
                glColor3ub(255, 255, 0);
        else
                glColor3f(0.9, 0.9, 0.9);
```

```
    glPushMatrix();
    glTranslatef(80, -14, 0.0);
    glBegin(GL_POLYGON);
    glVertex2d(-45, 5);
    glVertex2d(-36, 5);
    glVertex2d(-32, 12);
    glVertex2d(-40, 12);
    glEnd();
    glPopMatrix();

    //stopuri masina
    if(color == 0)
        glColor3ub(139, 0, 0);
    else
        glColor3ub(200, 0, 0);
    glPushMatrix();
    glTranslatef(-35, -12, 0.0);
    glBegin(GL_POLYGON);
    glVertex2d(-12, 0);
    glVertex2d(-5, 0);
    glVertex2d(-5, 10);
    glVertex2d(-12, 10);
    glEnd();

}

void masina_contrasens() {
    //desenam a doua masina (adversara)
    glPushMatrix();
    glTranslated(loc_vert, height, 0.0);

    if (color == 0)
        glColor3ub(0, 0, 255);
    else
        glColor3ub(0, 0, 138);
    glBegin(GL_POLYGON);
    glVertex2d(45, 0);
    glVertex2d(-50, 0);
    glVertex2d(-40, -20);
    glVertex2d(45, -20);
```

```
glEnd();

if (color == 0)
	glColor3ub(0, 0, 255);
else
	glColor3ub(0, 0, 138);
glBegin(GL_POLYGON);
glVertex2d(45, 0);
glVertex2d(-25, 0);
glVertex2d(-15, 20);
glVertex2d(25, 20);
glEnd();

glColor3ub(220, 220, 220);
glBegin(GL_POLYGON);
glVertex2d(30, 0);
glVertex2d(7, 0);
glVertex2d(7, 10);
glVertex2d(25, 10);
glEnd();

glColor3ub(255, 220, 220);
glBegin(GL_POLYGON);
glVertex2d(2, 0);
glVertex2d(-20, 0);
glVertex2d(-15, 10);
glVertex2d(2, 10);
glEnd();

//roata masinii
glColor3ub(0, 0, 0);
glPushMatrix();
glTranslatef(-24, -20, 0.0);
circle(10);
glPopMatrix();

glPushMatrix();
glTranslatef(30, -20, 0.0);
circle(10);
glPopMatrix();
```

```
glColor3ub(245, 245, 245);

glPushMatrix();
glTranslatef(-24, -20, 0.0);
circle(6);
glPopMatrix();

glPushMatrix();
glTranslatef(30, -20, 0.0);
circle(6);
glPopMatrix();

//farurile masinii
if (color == 1)
	glColor3ub(255, 255, 0);
else
	glColor3f(0.9, 0.9, 0.9);
glPushMatrix();
glTranslatef(-80, -14, 0.0);
glBegin(GL_POLYGON);
glVertex2d(45, 5);
glVertex2d(36, 5);
glVertex2d(32, 12);
glVertex2d(40, 12);
glEnd();
glPopMatrix();

//stopuri masina
if (color == 0)
	glColor3ub(139, 0, 0);
else
	glColor3ub(200, 0, 0);
glPushMatrix();
glTranslatef(35, -12, 0.0);
glBegin(GL_POLYGON);
glVertex2d(12, 0);
glVertex2d(5, 0);
glVertex2d(5, 10);
glVertex2d(12, 10);
```

```
    glEnd();
}

void delimitareLaterala() {
    if (color == 0)
        glColor3f(1.0, 1.0, 1.0);
    else
        glColor3f(0.3, 0.3, 0.3);

    // linia continua de sus
    glBegin(GL_POLYGON);
    glVertex2i(-115, 220);// Stanga jos
    glVertex2i(700, 220); // Dreapta jos
    glVertex2i(700, 240);  // Dreapta sus
    glVertex2i(-115, 240); // Stanga sus
    glEnd();

    // linia continua de jos
    glBegin(GL_POLYGON);
    glVertex2i(-100, -60);// Stanga jos
    glVertex2i(700, -60); // Dreapta jos
    glVertex2i(700, -80);  // Dreapta sus
    glVertex2i(-100, -80); // Stanga sus
    glEnd();

}

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-100.0, 700.0, -140.0, 460.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void miscasus(void)
{
    if (ok != 0)
```

```
    {
        if (j < 170)
        {
            contor = 1;
            j += 1;
        }

        glutPostRedisplay();
    }
}

void miscajos(void)
{
    if (ok != 0)
    {
        if (j > -30)
        {
            contor = -1;
            j -= 1;
        }

        glutPostRedisplay();
    }
}

void miscare_nori(int value) {
    posNor1 = posNor1 - 2.0;
    if (posNor1 < -520)
    {
        posNor1 = 520;
    }

    posNor2 = posNor2 - 2.0;
    if (posNor2 < -520) {
        posNor2 = 700;
    }

    posNor3 = posNor3 - 3.0;
    if (posNor3 < -520) {
        posNor3 = 550;
```

```
    }

    posNor4 = posNor4 - 1.0;
    if (posNor4 < -520) {
        posNor4 = 1000;
    }

    posNor5 = posNor5 - 3.5;
    if (posNor5 < -520) {
        posNor5 = 1500;
    }

    glutPostRedisplay();
    glutTimerFunc(20, miscare_nori, 0);
}

void keyboard(int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_UP:
        miscasus();
        break;
    case GLUT_KEY_DOWN:
        miscajos();
        break;
    }
}

void callback_Color(int key)
{
    color = key;
}

void processNormalKeys(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'z':
        callback_Color(0);//zi
        break;
```

```
    case 'n':
            callback_Color(1);//noapte
            break;
    }
    if (key == 27)
            exit(0);
}

void drawScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    iarba();

    cer();

    if (color == 0)
            sun();
    else
            moon();

    nor1();

    nor2();

    nor3();

    nor4();

    nor5();

    int currentTime = glutGet(GLUT_ELAPSED_TIME);

    if (currentTime - startTime < 3000)
            RenderString(200.0f, 425.0f,
GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"Watch
Out!");

    glPushMatrix();
    glPushMatrix();
```

```
        linieJ(x1Jos, x2Jos, yJos);
        glPopMatrix();
        glPushMatrix();
        linieS(x1Sus, x2Sus, ySus);
        glPopMatrix();
        glPopMatrix();

        delimitareLaterala();

        copaci(posXCopaci[0], posYCopaci, posFormC1);
        copaci(posXCopaci[1], posYCopaci, posFormC1);
        copaci(posXCopaci[2], posYCopaci, posFormC1);
        copaci(posXCopaci[3], posYCopaci, posFormC1);
        copaci(posXCopaci[4], posYCopaci, posFormC1);
        copaci(posXCopaci[5], posYCopaci, posFormC1);
        copaci(posXCopaci[6], posYCopaci, posFormC1);
        copaci(posXCopaci[7], posYCopaci, posFormC1);
        copaci(posXCopaci[8], posYCopaci, posFormC1);
        copaci(posXCopaci[9], posYCopaci, posFormC1);
        copaci(posXCopaci[10], posYCopaci, posFormC1);
        copaci(posXCopaci[11], posYCopaci, posFormC1);
        copaci(posXCopaci[12], posYCopaci, posFormC1);
        copaci(posXCopaci[13], posYCopaci, posFormC1);

        copaci(posXCopaci[0], posYCopaci, posFormC2);
        copaci(posXCopaci[1], posYCopaci, posFormC2);
        copaci(posXCopaci[2], posYCopaci, posFormC2);
        copaci(posXCopaci[3], posYCopaci, posFormC2);
        copaci(posXCopaci[4], posYCopaci, posFormC2);
        copaci(posXCopaci[5], posYCopaci, posFormC2);
        copaci(posXCopaci[6], posYCopaci, posFormC2);
        copaci(posXCopaci[7], posYCopaci, posFormC2);
        copaci(posXCopaci[8], posYCopaci, posFormC2);
        copaci(posXCopaci[9], posYCopaci, posFormC2);
        copaci(posXCopaci[10], posYCopaci, posFormC2);
        copaci(posXCopaci[11], posYCopaci, posFormC2);
        copaci(posXCopaci[12], posYCopaci, posFormC2);
        copaci(posXCopaci[13], posYCopaci, posFormC2);
```

```
	lampi(xL, yL, becX, becY, posFormLampa1);

	lampi(xL + 380, yL, becX + 380, becY, posFormLampa2);

	lampi(xL + 380, yL2, becX + 380, becY2, posFormLampa3);

	lampi(xL, yL2, becX, becY2, posFormLampa4);

	masina_principala();

	if (ok == 0)
	{
		rsj = 8;
		rss = -8;
		rdj = -8;
		rds = 8;
	}


	glPopMatrix();
	glPopMatrix();


	if (ok == 0) {
		RenderString(250.0f, 200.0f, GLUT_BITMAP_8_BY_13, (const
unsigned char*)"GAME OVER");
	}

	if (contor == 1 && (j != 70 && j != 170))
		j = j + 1;
	else if (contor == -1 && (j != 70 && j != -30))
		j = j - 1;
	else {
		contor = 0;
	}

	glPushMatrix();

	masina_contrasens();
```

```
    glPopMatrix();

    glPopMatrix();

    startgame();
    high();
    glutPostRedisplay();
    glutSwapBuffers();
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Depaseste masinile - mini game");
    init();
    glutDisplayFunc(drawScene);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(processNormalKeys);
    glutSpecialFunc(keyboard);
    miscare_copaci(0);
    miscare_nori(0);
    miscare_linii(0);
    miscare_lampi(0);
    startTime = glutGet(GLUT_ELAPSED_TIME);
    glutMainLoop();
}
```