A black background with white text

Description automatically generated

**PROIECT Elemente de Grafică pe Calculator**

**Cadru didactic:**

**Titulatura.Şef dr.ing.Luminiţa Popa**

**Student:**

**Simion Alexandru Gabriel**

**BRAȘOV, 2024**

**Departamentul Inginerie Electrică și Știința Calculatoarelor**

**Programul de studii: Tehnologia Informației**

Airplane Videogame

Brașov, 2024

**Cuprins**

1. **Introducere**

1.1 Motivație

1.2 Scopul Documentației

1.3 Descrierea Proiectului

1. **Dependențe**

2.1 Limbaj de Programare

2.2 Biblioteci

2.3 Mediu de Dezvoltare

2.4 Instalare si Configurare

1. **Structura Codului**

3.1 Variabile Globale

3.1.1 Variabile pentru Translatare și Viteză

3.1.2 Variabile pentru Gloanțe și Inamici

3.2 Inițializare și Configurare

3.2.1 Funcția 'init()'

3.2.2 Funcția 'reshape(int w, int h)'

3.3 Desenarea Elementelor

3.3.1 Funcția 'draw()'

3.3.2 Funcția 'drawGameOverScreen()'

3.4 Funcții pentru Logică și Actualizare

3.4.1 Funcția 'spawnFallingObjects()'

3.4.2 Funcția 'spawnFallingStars()'

3.4.3 Funcția 'update(int value)'

1. **Interacțiunea Utilizatorului**

4.1 Input de la Tastatură

4.1.1 Funcția 'keyPressed(unsigned char key, int x, int y)'

4.1.2 Funcția 'keyReleased(unsigned char key, int x, int y)'

4.2 Input de la Mouse

4.2.1 Funcția 'mouse(int button, int state, int x, int y)'

1. **Funcția Principală**

5.1 Funcția 'main(int argc, char\*\* argv)'

1. **Concluzie**

6.1 Rezumat al Proiectului

6.2 Direcții pentru Dezvoltare Viitoare

1. **Referințe**

7.1 Resurse Utilizate

1. **Anexe**

8.1 Capturi de Ecran

8.2 Codul Sursă Complet

# 

# 1 Introducere

1.1 Motivație

Proiectul acesta a fost conceput pentru a crea un joc simplu, dar captivant, folosind OpenGL și GLUT (OpenGL Utility Toolkit). Prin dezvoltarea acestui joc, dorim să demonstrăm modul în care tehnologiile grafice pot fi utilizate pentru a construi aplicații interactive. În plus, proiectul oferă o oportunitate de a învăța și aplica concepte fundamentale ale programării grafice și dezvoltării de jocuri.

1.2 Scopul Documentației

Scopul acestei documentații este de a descrie în detaliu proiectul jocului, inclusiv structura codului, funcționalitatea principală și modul de configurare a mediului de dezvoltare. Documentația se adresează studenților care doresc să înțeleagă și să extindă proiectul sau celor care doresc să învețe cum să dezvolte aplicații grafice interactive folosind OpenGL și GLUT.

1.3 Descrierea Proiectului

Proiectul este un joc de tip shooter în care jucătorul controlează un obiect (avion) și trebuie să evite și să distrugă obiectele căzătoare. Jocul include funcționalități pentru mișcarea avionului, tragerea de gloanțe, generarea și mișcarea obiectelor căzătoare și gestionarea scorului și a stării de final de joc. Codul este scris în C++ și utilizează bibliotecile OpenGL și GLUT pentru a realiza grafica 2D și interacțiunea cu utilizatorul.

# 2. Dependențe

2.1 Limbaj de Programare

Proiectul este scris în limbajul de programare C++. Este necesară o bună înțelegere a acestui limbaj pentru a modifica sau extinde funcționalitatea jocului.

2.2 Biblioteci

Proiectul utilizează următoarele biblioteci:

* OpenGL: OpenGL (Open Graphics Library) este o specificație standard pentru crearea de grafice 2D și 3D. În acest proiect, OpenGL este utilizat pentru a desena elementele grafice ale jocului.
* GLUT (OpenGL Utility Toolkit): GLUT este o bibliotecă care facilitează dezvoltarea aplicațiilor OpenGL, oferind funcționalități pentru gestionarea ferestrelor, evenimentelor de intrare și a altor elemente de interfață.

2.3 Mediul de Dezvoltare

Pentru a compila și rula proiectul, este necesar să aveți următoarele instrumente instalate pe sistemul dumneavoastră:

* Un compilator C++: Recomandăm utilizarea unui compilator compatibil cu standardele moderne C++, cum ar fi GCC (GNU Compiler Collection) sau Clang.
* Bibliotecile OpenGL și GLUT: Acestea trebuie să fie instalate și configurate corect pe sistemul dumneavoastră. Pe Windows, recomandăm utilizarea pachetului freeglut, care poate fi descărcat și instalat manual.

2.4 Instalare și Configurare

Pentru a configura mediul de dezvoltare, urmați pașii de mai jos:

Instalare Compilator C++:

* Descărcați și instalați GCC/Clang sau utilizați un IDE precum Visual Studio care include un compilator C++.

Instalare OpenGL și GLUT:

* Descărcați freeglut de pe site-ul oficial și urmați instrucțiunile de instalare.

Configurare Proiect:

* Asigurați-vă că proiectul dumneavoastră include corect directoarele și fișierele necesare pentru OpenGL și GLUT.
* Configurați variabilele de mediu și căile de includere și legare în compilatorul/IDE-ul pe care îl folosiți.

Acum, proiectul este pregătit pentru a fi compilat și rulat. Asigurați-vă că toate dependențele sunt instalate și configurate corect înainte de a începe dezvoltarea sau testarea jocului.

3. Structura Codului

3.1 Variabile Globale

Proiectul folosește mai multe variabile globale pentru a gestiona starea jocului, pozițiile obiectelor și alte aspecte esențiale. Aceste variabile sunt partajate între diferite funcții pentru a asigura consistența și pentru a simplifica accesul la date.

3.1.1 Variabile pentru Translatare și Viteză

Aceste variabile sunt utilizate pentru a controla mișcarea și viteza personajului principal și a altor obiecte din joc.

* float translateX = 0.0f;
* float translateY = -0.5f;
* float translationSpeed = 0.0175f;
  + 1. Variabile pentru Gloanțe și Inamici

Variabile pentru gestionarea gloanțelor și inamicilor.

* std::vector<float> bulletYs;
* std::vector<float> fallingObjectsY;
* std::vector<float> fallingObjectsX;
* std::vector<float> fallingStarsY;
* std::vector<float> fallingStarsX;
* float bulletSpeed = 0.35f;
* bool isShooting = false;
* clock\_t lastShotTime = 0;
* const float cooldownDuration = 0.125f;

3.2 Inițializare și Configurare

Inițializarea și configurarea sunt esențiale pentru pregătirea mediului de joc și setarea parametrilor inițiali.

3.2.1 Funcția init()

Funcția init() configurează setările inițiale ale jocului, cum ar fi culoarea de fundal și alte setări OpenGL necesare.

**void** init**()** **{**

***// Setează culoarea de fond a ferestrei la negru cu opacitate completă***

glClearColor**(**0.0**,** 0.0**,** 0.0**,** 1.0**);**

***// Activează amestecul de culori pentru a permite transparența***

glEnable**(**GL\_BLEND**);**

***// Schimbă matricea activă în matricea de proiecție***

glMatrixMode**(**GL\_PROJECTION**);**

glLoadIdentity**();** ***// Resetează matricea de proiecție la matricea identitate***

***// Setează proiecția ortogonală 2D cu coordonate fixe***

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);**

***// Revine la matricea de modelare/vizualizare***

glMatrixMode**(**GL\_MODELVIEW**);**

***// Inițializează generatorul de numere aleatoare folosind timpul curent ca sămânță***

srand**(static\_cast<unsigned** **int>(**time**(**0**)));**

**}**

3.2.2 Funcția reshape(int w, int h)

Funcția reshape(int w, int h) este apelată atunci când fereastra este redimensionată și ajustează viewport-ul și proiecția pentru a se adapta noii dimensiuni.

**void** reshape**(int** w**,** **int** h**)** **{**

***// Setează noua dimensiune a viewport-ului pentru a acoperi întreaga fereastră***

glViewport**(**0**,** 0**,** w**,** h**);**

***// Schimbă matricea activă în matricea de proiecție***

glMatrixMode**(**GL\_PROJECTION**);**

glLoadIdentity**();** ***// Resetează matricea de proiecție la matricea identitate***

***// Setează proiecția ortogonală 2D cu coordonate fixe***

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);**

***// Revine la matricea de modelare/vizualizare***

glMatrixMode**(**GL\_MODELVIEW**);**

**}**

3.3 Desenarea Elementelor

Desenarea elementelor este esențială pentru a afișa jocul pe ecran, incluzând jucătorul, gloanțele și alte obiecte.

3.3.1 Funcția draw()

Funcția draw() este responsabilă pentru desenarea elementelor de joc la fiecare cadru. Aceasta include desenarea jucătorului, gloanțelor, inamicilor, efectelor de background, etc.

**void** draw**()** **{**

glClear**(**GL\_COLOR\_BUFFER\_BIT**);** ***// Curăță bufferul de culoare***

glLoadIdentity**();** ***// Resetează matricea de modelare***

***// Desenează un dreptunghi translucid pentru fundal***

glColor4f**(**0.0f**,** 0.0f**,** 0.035f**,** 0.5f**);** ***// Albastru închis cu opacitate de 50%***

glBegin**(**GL\_QUADS**);**

glVertex2f**(-**1.0f**,** **-**1.0f**);**

glVertex2f**(**1.0f**,** **-**1.0f**);**

glVertex2f**(**1.0f**,** 1.0f**);**

glVertex2f**(-**1.0f**,** 1.0f**);**

glEnd**();**

***// Desenează stelele căzătoare***

**for** **(size\_t** i **=** 0**;** i **<** fallingStarsY**.**size**();** **++**i**)** **{**

glPushMatrix**();**

glTranslatef**(**fallingStarsX**[**i**],** fallingStarsY**[**i**],** 0.0f**);** ***// Translatează poziția stelei***

glScalef**(**0.05f**,** 0.05f**,** 1.0f**);** ***// Ajustează scala pentru a face stelele mai mici***

glBegin**(**GL\_TRIANGLES**);**

***// Primul triunghi***

glColor3f**(**1.0f**,** 1.0f**,** 1.0f**);** ***// Culoare albă***

glVertex3f**(-**0.60**,** 0.77**,** 0**);**

glVertex3f**(-**0.42**,** 0.77**,** 0**);**

glVertex3f**(-**0.58**,** 0.68**,** 0**);**

***// Al doilea triunghi***

glVertex3f**(-**0.64**,** 1**,** 0**);**

glVertex3f**(-**0.68**,** 0.77**,** 0**);**

glVertex3f**(-**0.60**,** 0.77**,** 0**);**

***// Al treilea triunghi***

glVertex3f**(-**0.68**,** 0.77**,** 0**);**

glVertex3f**(-**0.7**,** 0.68**,** 0**);**

glVertex3f**(-**0.86**,** 0.77**,** 0**);**

***// Al patrulea triunghi***

glVertex3f**(-**0.64**,** 0.63**,** 0**);**

glVertex3f**(-**0.7**,** 0.68**,** 0**);**

glVertex3f**(-**0.82**,** 0.43**,** 0**);**

***// Al cincilea triunghi***

glVertex3f**(-**0.64**,** 0.63**,** 0**);**

glVertex3f**(-**0.58**,** 0.68**,** 0**);**

glVertex3f**(-**0.51**,** 0.43**,** 0**);**

glEnd**();**

glColor3f**(**1.0f**,** 1.0f**,** 1.0f**);** ***// Culoare albă***

glBegin**(**GL\_POLYGON**);**

***// Poligon central pentru stea***

glVertex3f**(-**0.60**,** 0.77**,** 0**);**

glVertex3f**(-**0.68**,** 0.77**,** 0**);**

glVertex3f**(-**0.7**,** 0.68**,** 0**);**

glVertex3f**(-**0.64**,** 0.63**,** 0**);**

glVertex3f**(-**0.58**,** 0.68**,** 0**);**

glEnd**();**

glPopMatrix**();**

**}**

***// Desenează inamicii***

glColor3f**(**1.0**,** 0.0**,** 0.0**);** ***// Culoare roșie***

**for** **(size\_t** i **=** 0**;** i **<** fallingObjectsY**.**size**();** **++**i**)** **{**

glPushMatrix**();**

glTranslatef**(**fallingObjectsX**[**i**],** fallingObjectsY**[**i**],** 0.0f**);** ***// Translatează poziția***

glRotatef**(**180.0f**,** 0.0f**,** 0.0f**,** 1.0f**);** ***// Rotește cu 180 de grade***

glScalef**(**0.1f**,** 0.1f**,** 1.0f**);** ***// Ajustează scala***

glBegin**(**GL\_QUADS**);**

***// Desenează corpul***

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glEnd**();**

***// Desenează triunghiurile***

glBegin**(**GL\_TRIANGLES**);**

glVertex2f**(-**0.1f**,** 0.0f**);**

glVertex2f**(-**0.45f**,** 0.1f**);**

glVertex2f**(-**0.1f**,** 0.25f**);**

glVertex2f**(**0.1f**,** 0.0f**);**

glVertex2f**(**0.45f**,** 0.1f**);**

glVertex2f**(**0.1f**,** 0.25f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(**0.0f**,** 0.5f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glVertex2f**(-**0.1f**,** **-**0.1f**);**

glVertex2f**(-**0.3f**,** **-**0.4f**);**

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.1f**);**

glVertex2f**(**0.3f**,** **-**0.4f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glEnd**();**

glPopMatrix**();**

**}**

***// Translatează și scalează jucătorul***

glTranslatef**(**translateX**,** translateY**,** 0.0f**);**

glScalef**(**0.175f**,** 0.175f**,** 1.0f**);**

***// Desenează corpul obiectului jucătorului***

glBegin**(**GL\_QUADS**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);** ***// Culoare albă***

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glEnd**();**

***// Desenează triunghiurile pentru obiectul jucătorului***

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);** ***// Culoare albă***

glVertex2f**(-**0.1f**,** 0.0f**);**

glVertex2f**(-**0.45f**,** 0.1f**);**

glVertex2f**(-**0.1f**,** 0.25f**);**

glVertex2f**(**0.1f**,** 0.0f**);**

glVertex2f**(**0.45f**,** 0.1f**);**

glVertex2f**(**0.1f**,** 0.25f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);** ***// Culoare albă***

glVertex2f**(-**0.1f**,** 0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(**0.0f**,** 0.5f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);** ***// Culoare albă***

glVertex2f**(-**0.1f**,** **-**0.1f**);**

glVertex2f**(-**0.3f**,** **-**0.4f**);**

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.1f**);**

glVertex2f**(**0.3f**,** **-**0.4f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glEnd**();**

***// Desenează gloanțele***

**for** **(size\_t** i **=** 0**;** i **<** bulletYs**.**size**();** **++**i**)** **{**

glColor3f**(**1.0**,** 1.0**,** 1.0**);** ***// Culoare albă***

glPointSize**(**5.5f**);** ***// Dimensiunea punctelor***

glBegin**(**GL\_POINTS**);**

glVertex2f**(**0**,** bulletYs**[**i**]);**

glEnd**();**

**}**

***// Desenează scorul***

glMatrixMode**(**GL\_PROJECTION**);**

glPushMatrix**();**

glLoadIdentity**();**

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);** ***// Setează proiecția ortogonală 2D***

glMatrixMode**(**GL\_MODELVIEW**);**

glPushMatrix**();**

glLoadIdentity**();**

glColor3f**(**1.0**,** 1.0**,** 1.0**);** ***// Culoare albă pentru text***

**char** scoreText**[**20**];**

sprintf\_s**(**scoreText**,** **sizeof(**scoreText**),** "Score: %d"**,** score**);**

glScalef**(**2.0f**,** 2.0f**,** 1.0f**);** ***// Scalează textul***

renderBitmapString**(**0.35f**,** 0.45f**,** GLUT\_BITMAP\_HELVETICA\_18**,** scoreText**);**

glPopMatrix**();**

glMatrixMode**(**GL\_PROJECTION**);**

glPopMatrix**();**

glMatrixMode**(**GL\_MODELVIEW**);**

***// Verifică dacă jocul s-a terminat și desenează ecranul de finalizare***

**if** **(**gameOver**)**

**{**

drawGameOverScreen**();**

**}**

glutSwapBuffers**();** ***// Schimbă bufferul de ecran pentru a afișa conținutul desenat***

**}**

3.3.2 Funcția drawGameOverScreen()

Funcția drawGameOverScreen() desenează ecranul de Game Over atunci când jocul se încheie.

**void** drawGameOverScreen**()** **{**

***// Setează modul de matrice la GL\_PROJECTION pentru a manipula proiecția***

glMatrixMode**(**GL\_PROJECTION**);**

glPushMatrix**();** ***// Salvează starea curentă a matricei de proiecție***

glLoadIdentity**();** ***// Resetează matricea de proiecție***

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);** ***// Setează proiecția ortogonală 2D cu limitele specificate***

***// Comută la modul de matrice GL\_MODELVIEW pentru a manipula modelul vizualizat***

glMatrixMode**(**GL\_MODELVIEW**);**

glPushMatrix**();** ***// Salvează starea curentă a matricei de modelare***

glLoadIdentity**();** ***// Resetează matricea de modelare***

***// Setează culoarea pentru text (alb)***

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

***// Redă textul "Game Over" folosind fontul GLUT\_BITMAP\_TIMES\_ROMAN\_24 la coordonatele specificate***

renderBitmapString**(-**0.3f**,** 0.0f**,** GLUT\_BITMAP\_TIMES\_ROMAN\_24**,** "Game Over"**);**

***// Redă textul "Press 'R' to restart" folosind fontul GLUT\_BITMAP\_HELVETICA\_18 la coordonatele specificate***

renderBitmapString**(-**0.4f**,** **-**0.1f**,** GLUT\_BITMAP\_HELVETICA\_18**,** "Press 'R' to restart"**);**

***// Restaurează starea matricei de modelare anterioare***

glPopMatrix**();**

***// Comută înapoi la modul de matrice GL\_PROJECTION și restaurează starea anterioară***

glMatrixMode**(**GL\_PROJECTION**);**

glPopMatrix**();**

***// Comută înapoi la modul de matrice GL\_MODELVIEW pentru continuarea procesării normale***

glMatrixMode**(**GL\_MODELVIEW**);**

**}**

3.4 Funcții pentru Logică și Actualizare

Aceste funcții gestionează logica jocului și actualizează starea jocului la fiecare cadru.

3.4.1 Funcția spawnFallingObjects()

Funcția spawnFallingObjects() generează inamici noi care cad din partea de sus a ecranului.

**void** spawnFallingObjects**()** **{**

**int** numObjects **=** rand**()** **%** 5 **+** 1**;** ***// Generează un număr aleatoriu de obiecte care vor cădea (între 1 și 5)***

**for** **(int** i **=** 0**;** i **<** numObjects**;** **++**i**)** **{**

**float** randomX**,** randomY**;**

**bool** overlapping**;** ***// Variabilă pentru a verifica suprapunerea***

***// Generează poziții aleatorii până când se găsește o poziție care nu se suprapune***

**do** **{**

randomX **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;** ***// Generează o poziție X aleatorie între -0.9 și 0.9***

randomY **=** 1.0f**;** ***// Setează poziția Y la 1.0f (de obicei partea superioară a ecranului)***

***// Verifică dacă noua poziție se suprapune cu oricare dintre obiectele deja existente***

overlapping **=** **false;** ***// Resetează flag-ul de suprapunere***

**for** **(size\_t** i **=** 0**;** i **<** fallingObjectsY**.**size**();** **++**i**)** **{**

**if** **(**fabs**(**randomX **-** fallingObjectsX**[**i**])** **<** 0.2f **&&** fabs**(**randomY **-** fallingObjectsY**[**i**])** **<** 0.2f**)** **{**

***// Dacă diferența dintre pozițiile X și Y este mai mică de 0.2, se consideră suprapunere***

overlapping **=** **true;**

**break;** ***// Ieșire din buclă dacă se găsește o suprapunere***

**}**

**}**

***// Dacă există suprapunere, ajustează poziția și repetă procesul***

**if** **(**overlapping**)** **{**

***// Ajustează poziția (de exemplu, mută vertical)***

randomX **-=** 0.2f**;** ***// Muta poziția X în jos cu 0.2***

**}**

**}** **while** **(**overlapping**);** ***// Repetă până se găsește o poziție care nu se suprapune***

***// Adaugă noul obiect care cade cu poziția ajustată***

fallingObjectsY**.**push\_back**(**randomY**);** ***// Adaugă poziția Y a noului obiect***

fallingObjectsX**.**push\_back**(**randomX**);** ***// Adaugă poziția X a noului obiect***

**}**

**}**

3.4.2 Funcția spawnFallingStars()

Funcția spawnFallingStars() generează stele căzătoare. Aceasta poate fi similară cu spawnFallingObjects(), dar specifică pentru stele.

**void** spawnFallingStars**()** **{**

**if** **(**fallingStarsY**.**empty**())** **{**

***// Generează stele aleatoriu pe ecran la începutul programului***

**int** numStars **=** 30**;**

**for** **(int** i **=** 0**;** i **<** numStars**;** **++**i**)** **{**

***// Generează o poziție aleatorie pentru X și Y în cadrul ecranului***

**float** randomX **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;**

**float** randomY **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;**

***// Adaugă coordonatele la vectorul de poziții X și Y ale stelelor***

fallingStarsX**.**push\_back**(**randomX**);**

fallingStarsY**.**push\_back**(**randomY**);**

**}**

**}**

**else** **{**

***// Generează stele noi în partea de sus a ecranului pe măsură ce încep să cadă cele existente***

**int** numStars **=** rand**()** **%** 10 **+** 1**;** ***// Generează mai puține stele pe măsură ce se adaugă altele noi pentru a evita supraaglomerarea***

**for** **(int** i **=** 0**;** i **<** numStars**;** **++**i**)** **{**

***// Generează o poziție aleatorie pentru X și plasează stelele în partea de sus a ecranului***

**float** randomX **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;**

**float** randomY **=** 1.0f**;** ***// Generează stele la partea de sus a ecranului***

***// Adaugă coordonatele la vectorul de poziții X și Y ale stelelor***

fallingStarsX**.**push\_back**(**randomX**);**

fallingStarsY**.**push\_back**(**randomY**);**

**}**

**}**

**}**

3.4.3 Funcția update(int value)

Funcția update(int value) este apelată la intervale regulate pentru a actualiza starea jocului, incluzând pozițiile jucătorului, gloanțelor și obiectelor căzătoare.

**void** update**(int** value**)** **{**

***// Verifică dacă jocul nu s-a încheiat***

**if** **(!**gameOver**)** **{**

***// Verifică starea tastelor pentru mișcare diagonală***

**if** **(**keyStates**[**'w'**]** **&&** keyStates**[**'a'**])** **{**

translateY **+=** translationSpeed **\*** 0.7071f**;**

translateX **-=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **if** **(**keyStates**[**'w'**]** **&&** keyStates**[**'d'**])** **{**

translateY **+=** translationSpeed **\*** 0.7071f**;**

translateX **+=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **if** **(**keyStates**[**'s'**]** **&&** keyStates**[**'a'**])** **{**

translateY **-=** translationSpeed **\*** 0.7071f**;**

translateX **-=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **if** **(**keyStates**[**'s'**]** **&&** keyStates**[**'d'**])** **{**

translateY **-=** translationSpeed **\*** 0.7071f**;**

translateX **+=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **{**

***// Mișcare simplă pe orizontală și verticală***

**if** **(**keyStates**[**'w'**])** **{**

translateY **+=** translationSpeed**;**

**}**

**if** **(**keyStates**[**'s'**])** **{**

translateY **-=** translationSpeed**;**

**}**

**if** **(**keyStates**[**'a'**])** **{**

translateX **-=** translationSpeed**;**

**}**

**if** **(**keyStates**[**'d'**])** **{**

translateX **+=** translationSpeed**;**

**}**

**}**

***// Asigură că jucătorul rămâne în interiorul ecranului***

**if** **(**translateX **<** **-**0.90f**)** translateX **=** **-**0.90f**;**

**if** **(**translateX **>** 0.90f**)** translateX **=** 0.90f**;**

**if** **(**translateY **<** **-**0.90f**)** translateY **=** **-**0.90f**;**

**if** **(**translateY **>** 0.90f**)** translateY **=** 0.90f**;**

***// Inițializare joc la început***

**if** **(**gameStartTime **==** 0**)** **{**

spawnFallingObjects**();**

spawnFallingStars**();**

gameStartTime **=** clock**();**

**}**

***// Actualizare timp și dificultate pe parcursul jocului***

**clock\_t** currentTime **=** clock**();**

**float** elapsedTime **=** **static\_cast<float>(**currentTime **-** gameStartTime**)** **/** CLOCKS\_PER\_SEC**;**

**float** elapsetShotTime **=** **static\_cast<float>(**currentTime **-** lastShotTime**)** **/** CLOCKS\_PER\_SEC**;**

***// Generare obiecte căzătoare și stele în timpul jocului***

**if** **(**elapsedTime **>** spawnInterval**)** **{**

spawnFallingObjects**();**

spawnFallingStars**();**

gameStartTime **=** currentTime**;**

**if** **(**elapsedTime **>** harderOverTime**)** **{**

spawnInterval **\*=** 0.999f**;**

harderOverTime **+=** 30.0f**;**

**}**

**}**

***// Tragerea unui glonț dacă s-a trecut de timpul de răcire***

**if** **(**isShooting **&&** elapsetShotTime **>** cooldownDuration**)** **{**

bulletYs**.**push\_back**(**translateY **+** 0.80f**);**

lastShotTime **=** currentTime**;**

**}**

***// Detectarea coliziunilor între gloanțe și obiectele căzătoare***

**for** **(size\_t** i **=** 0**;** i **<** bulletYs**.**size**();** **++**i**)** **{**

**float** bulletRadius **=** 0.075f**;**

**for** **(size\_t** j **=** 0**;** j **<** fallingObjectsY**.**size**();** **++**j**)** **{**

**float** fallingObjectRadius **=** 0.125f**;**

**if** **(**bulletYs**[**i**]** **+** bulletRadius **>** fallingObjectsY**[**j**]** **-** fallingObjectRadius **&&**

bulletYs**[**i**]** **-** bulletRadius **<** fallingObjectsY**[**j**]** **+** fallingObjectRadius **&&**

translateX **+** bulletRadius **>** fallingObjectsX**[**j**]** **-** fallingObjectRadius **&&**

translateX **-** bulletRadius **<** fallingObjectsX**[**j**]** **+** fallingObjectRadius**)** **{**

bulletYs**.**erase**(**bulletYs**.**begin**()** **+** i**);**

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** j**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** j**);**

score **+=** 10**;**

**--**i**;**

**break;**

**}**

**}**

**}**

***// Detectarea coliziunilor cu obiectele căzătoare***

**for** **(size\_t** j **=** 0**;** j **<** fallingObjectsY**.**size**();** **++**j**)** **{**

**if** **(**fabs**(**translateY **-** fallingObjectsY**[**j**])** **<** 0.1f **&&** fabs**(**translateX **-** fallingObjectsX**[**j**])** **<** 0.1f**)** **{**

gameOver **=** **true;**

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** j**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** j**);**

**break;**

**}**

**}**

***// Actualizare poziției stelelor căzătoare și eliminarea lor când ies din ecran***

**for** **(size\_t** i **=** 0**;** i **<** fallingStarsY**.**size**();** **++**i**)** **{**

fallingStarsY**[**i**]** **+=** fallingStarSpeed**;**

**if** **(**fallingStarsY**[**i**]** **<** **-**1.0f**)** **{**

fallingStarsY**.**erase**(**fallingStarsY**.**begin**()** **+** i**);**

fallingStarsX**.**erase**(**fallingStarsX**.**begin**()** **+** i**);**

**--**i**;**

**}**

**}**

***// Actualizare poziției gloanțelor și eliminarea lor când ies din ecran***

**for** **(size\_t** i **=** 0**;** i **<** bulletYs**.**size**();** **++**i**)** **{**

bulletYs**[**i**]** **+=** bulletSpeed**;**

**if** **(**bulletYs**[**i**]** **>** 50.0f**)** **{**

bulletYs**.**erase**(**bulletYs**.**begin**()** **+** i**);**

**--**i**;**

**}**

**}**

***// Actualizare poziției și eliminarea obiectelor căzătoare când ies din ecran***

**for** **(size\_t** i **=** 0**;** i **<** fallingObjectsY**.**size**();** **++**i**)** **{**

fallingObjectsY**[**i**]** **+=** fallingObjectSpeed**;**

**if** **(**fallingObjectsY**[**i**]** **<** **-**1.0f**)** **{**

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** i**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** i**);**

**--**i**;**

score **-=** 10**;**

**}**

**}**

***// Detectare și eliminare obiecte căzătoare prin tragere***

**if** **(**isShooting**)** **{**

**for** **(size\_t** j **=** 0**;** j **<** fallingObjectsY**.**size**();** **++**j**)** **{**

**float** dist **=** distance**(**translateX**,** translateY**,** fallingObjectsX**[**j**],** fallingObjectsY**[**j**]);**

**if** **(**dist **<** destructionRadius**)** **{**

***// Distrugerea obiectului căzut***

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** j**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** j**);**

score **+=** 10**;**

**--**j**;** ***// Ajustare index deoarece obiectul căzut este eliminat***

**}**

**}**

**}**

**}**

**else** **{**

***// Reinițializare joc la apăsarea tastei 'r' sau 'R'***

**if** **(**keyStates**[**'r'**]** **||** keyStates**[**'R'**])** **{**

resetGame**();**

**}**

**}**

***// Reafișarea scenei și programarea actualizării ulterioare***

glutPostRedisplay**();**

glutTimerFunc**(**16**,** update**,** 0**);**

**}**

Această structură a codului evidențiază organizarea și funcționalitățile principale ale jocului, facilitând înțelegerea și modificarea proiectului.

4. Interacțiunea Utilizatorului

Interacțiunea utilizatorului este o componentă esențială în orice joc sau aplicație interactivă. În acest capitol, vom explora modul în care utilizatorii pot interacționa cu jocul tău, inclusiv introducerea de la tastatură și de la mouse.

4.1 Input de la Tastatură

Tastatura este una dintre cele mai comune modalități de interacțiune cu jocurile pe calculator. Prin aceasta, jucătorii pot controla acțiunile personajului sau alte elemente ale jocului.

4.1.1 Funcția 'keyPressed(unsigned char key, int x, int y)'

Această funcție este apelată atunci când o tastă este apăsată. Implementarea sa conține logica asociată cu gestionarea tastelor apăsate de către utilizator.

**void** keyPressed**(unsigned** **char** key**,** **int** x**,** **int** y**)** **{**

keyStates**[**key**]** **=** **true;** ***// Se setează starea tastei apăsate în map-ul de stări***

***// Verifică dacă tasta apăsată este 'r' sau 'R' pentru a reseta jocul***

**if** **(**key **==** 'r' **||** key **==** 'R'**)** **{**

resetGame**();** ***// Se apelează funcția de resetare a jocului***

**}**

**}**

4.1.2 Funcția 'keyReleased(unsigned char key, int x, int y)'

Această funcție este apelată atunci când o tastă este eliberată. Poate fi utilizată pentru a gestiona evenimentele asociate cu eliberarea unei taste.

**void** keyReleased**(unsigned** **char** key**,** **int** x**,** **int** y**)** **{**

keyStates**[**key**]** **=** **false;** ***// Se setează starea tastei eliberate în map-ul de stări***

**}**

4.2 Input de la Mouse

În unele jocuri sau aplicații, utilizatorii pot interacționa și cu ajutorul mouse-ului, permițându-le să efectueze acțiuni precum trageri.

4.2.1 Funcția 'mouse(int button, int state, int x, int y)'

Această funcție este apelată atunci când un eveniment de mouse are loc, cum ar fi apăsarea sau eliberarea unui buton al mouse-ului.

**void** mouse**(int** button**,** **int** state**,** **int** x**,** **int** y**)** **{**

**if** **(**button **==** GLUT\_LEFT\_BUTTON**)** **{** ***// Verifică dacă s-a apăsat butonul stâng al mouse-ului***

**if** **(**state **==** GLUT\_DOWN**)** **{** ***// Verifică dacă butonul stâng al mouse-ului este apăsat***

isShooting **=** **true;** ***// Marchează că se trage, de exemplu, dacă jocul utilizează mouse-ul pentru trageri***

**}**

**else** **if** **(**state **==** GLUT\_UP**)** **{** ***// Verifică dacă butonul stâng al mouse-ului este eliberat***

isShooting **=** **false;** ***// Marchează că nu se mai trage***

**}**

**}**

**}**

Aceste funcții pot fi adaptate și extinse în funcție de cerințele specifice ale jocului tău și de interacțiunile pe care dorești să le oferi utilizatorilor. Utilizarea lor corectă poate îmbunătăți semnificativ experiența jucătorului în jocul tău.

Capitolul 5. Funcția Principală:

5.1 Funcția main(int argc, char\*\* argv)

Această funcție este punctul de intrare în programul OpenGL. Ea inițializează și configurează fereastra OpenGL și apoi intră în bucla principală de procesare a evenimentelor OpenGL.

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

***// Inițializează GLUT și specifică modul de afișare și proprietățile ferestrei***

glutInit**(&**argc**,** argv**);**

glutInitDisplayMode**(**GLUT\_DOUBLE **|** GLUT\_RGB**);**

glutInitWindowSize**(**800**,** 600**);** ***// Setează dimensiunile ferestrei***

glutInitWindowPosition**(**150**,** 150**);** ***// Setează poziția ferestrei pe ecran***

glutCreateWindow**(**"Proiect"**);** ***// Creează fereastra cu titlul specificat***

***// Inițializează scenele OpenGL și alte configurații***

init**();**

***// Specifică funcțiile pentru desenare, input de la tastatură și mouse***

glutDisplayFunc**(**draw**);**

glutKeyboardFunc**(**keyPressed**);**

glutKeyboardUpFunc**(**keyReleased**);**

glutMouseFunc**(**mouse**);**

***// Inițializează și pornește timer-ul pentru actualizarea scenei***

glutTimerFunc**(**16**,** update**,** 0**);**

***// Intră în bucla principală a GLUT pentru procesarea evenimentelor***

glutMainLoop**();**

**return** 0**;**

**}**

Această funcție configurează fereastra OpenGL utilizând funcțiile glutInitDisplayMode, glutInitWindowSize, glutInitWindowPosition și glutCreateWindow. Apoi, sunt specificate funcțiile callback pentru desenare (glutDisplayFunc), input de la tastatură (glutKeyboardFunc și glutKeyboardUpFunc) și input de la mouse (glutMouseFunc). În final, bucla principală a GLUT (glutMainLoop) gestionează continuu evenimentele și interacțiunile din cadrul aplicației.

6.Concluzie

6.1 Rezumat al Proiectului

Proiectul meu constă în dezvoltarea unui joc simplu în OpenGL, în care jucătorul trebuie să evite obiectele care cad și să își mențină scorul cât mai mare. Jocul permite mișcarea în sus, jos, stânga și dreapta, trageri cu glonțul pentru a distruge obiectele care cad și oferă o interfață grafică simplă pentru utilizator. Acest proiect reprezintă o introducere în conceptele de bază ale programării grafice 3D și interacțiunii utilizatorului în OpenGL.

6.2 Direcții pentru Dezvoltare Viitoare

Pentru dezvoltarea ulterioară a acestui proiect, pot fi luate în considerare următoarele direcții:

* Adăugarea de mai multe niveluri pentru a menține interesul jucătorilor.
* Implementarea unui sistem de meniu și ecran de pornire pentru a oferi o experiență mai completă utilizatorului.
* Îmbunătățirea graficii și a animațiilor pentru a face jocul mai captivant vizual.
* Adăugarea de sunete și efecte sonore pentru a îmbunătăți experiența auditivă a jocului.

7.Referințe

7.1 Resurse Utilizate

Documentația OpenGL: <https://www.opengl.org/documentation/>

Documentația GLUT: <https://www.opengl.org/resources/libraries/glut/>

Tutoriale și exemple de cod OpenGL: <https://www.opengl.org/resources/libraries/glut/>

Laboratoarele de Elemente de Grafica pe Calculator

8.Anexe

* 1. Capturi de Ecran

O imagine care conține astronomie, captură de ecran, spațiu, întuneric

Descriere generată automat

O imagine care conține stea, spațiu, întuneric, captură de ecran

Descriere generată automat

O imagine care conține captură de ecran, spațiu, întuneric, Univers

Descriere generată automat

* 1. Codul Sursă Complet

#include <GL/glut.h>

#include <iostream>

#include <vector>

#include <ctime>

***// Variabile globale pentru controlul jocului***

**float** translateX **=** 0.0f**;** ***// Coordonata X a jucătorului***

**float** translateY **=** **-**0.5f**;** ***// Coordonata Y a jucătorului***

**float** translationSpeed **=** 0.0175f**;** ***// Viteza de deplasare a jucătorului***

**bool** keyStates**[**256**]** **=** **{** **false** **};** ***// Starea tastelor***

std**::**vector**<float>** bulletYs**;** ***// Lista cu coordonatele Y ale glonțurilor***

std**::**vector**<float>** fallingObjectsY**;** ***// Lista cu coordonatele Y ale obiectelor care cad***

std**::**vector**<float>** fallingObjectsX**;** ***// Lista cu coordonatele X ale obiectelor care cad***

std**::**vector**<float>** fallingStarsY**;** ***// Lista cu coordonatele Y ale stelelor care cad***

std**::**vector**<float>** fallingStarsX**;** ***// Lista cu coordonatele X ale stelelor care cad***

**float** bulletSpeed **=** 0.35f**;** ***// Viteza glonțurilor***

**bool** isShooting **=** **false;** ***// Indicator pentru trageri***

**clock\_t** lastShotTime **=** 0**;** ***// Timpul ultimei trageri***

**float** spawnInterval **=** 3.0f**;** ***//Intervalul dintre a aparea inamicii***

**const** **float** cooldownDuration **=** 0.125f**;** ***// Durata de răcire între trageri***

**float** harderOverTime **=** 30.0f**;** ***// Variabilă pentru creșterea dificultății în timp***

***// Viteza de cădere a obiectelor și a stelelor***

**float** fallingObjectSpeed **=** **-**0.01f**;**

**float** fallingStarSpeed **=** **-**0.0005f**;**

***// Variabile pentru gestionarea timpului și a scorului***

**time\_t** gameStartTime **=** 0**;**

**int** score **=** 0**;**

**bool** gameOver **=** **false;**

**bool** overlapping**;**

**const** **float** destructionRadius **=** 0.25f**;** ***// Raza de distrugere a obiectelor***

***// Funcție pentru calcularea distanței între două puncte***

**float** distance**(float** x1**,** **float** y1**,** **float** x2**,** **float** y2**)** **{**

**return** sqrt**(**pow**(**x2 **-** x1**,** 2**)** **+** pow**(**y2 **-** y1**,** 2**));**

**}**

***// Funcție pentru inițializarea OpenGL***

**void** init**()**

**{**

glClearColor**(**0.0**,** 0.0**,** 0.0**,** 1.0**);**

glEnable**(**GL\_BLEND**);**

glMatrixMode**(**GL\_PROJECTION**);**

glLoadIdentity**();**

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);**

glMatrixMode**(**GL\_MODELVIEW**);**

srand**(static\_cast<unsigned** **int>(**time**(**0**)));**

**}**

***// Funcție pentru redimensionarea ferestrei***

**void** reshape**(int** w**,** **int** h**)**

**{**

glViewport**(**0**,** 0**,** w**,** h**);**

glMatrixMode**(**GL\_PROJECTION**);**

glLoadIdentity**();**

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);**

glMatrixMode**(**GL\_MODELVIEW**);**

**}**

***// Funcție pentru afișarea textului pe ecran***

**void** renderBitmapString**(float** x**,** **float** y**,** **void\*** font**,** **const** **char\*** string**)** **{**

**const** **char\*** c**;**

glRasterPos2f**(**x**,** y**);**

**for** **(**c **=** string**;** **\***c **!=** '\0'**;** c**++)** **{**

glutBitmapCharacter**(**font**,** **\***c**);**

**}**

**}**

***// Funcție pentru afișarea ecranului de Game Over***

**void** drawGameOverScreen**()** **{**

glMatrixMode**(**GL\_PROJECTION**);**

glPushMatrix**();**

glLoadIdentity**();**

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);**

glMatrixMode**(**GL\_MODELVIEW**);**

glPushMatrix**();**

glLoadIdentity**();**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

renderBitmapString**(-**0.3f**,** 0.0f**,** GLUT\_BITMAP\_TIMES\_ROMAN\_24**,** "Game Over"**);**

renderBitmapString**(-**0.4f**,** **-**0.1f**,** GLUT\_BITMAP\_HELVETICA\_18**,** "Press 'R' to restart"**);**

glPopMatrix**();**

glMatrixMode**(**GL\_PROJECTION**);**

glPopMatrix**();**

glMatrixMode**(**GL\_MODELVIEW**);**

**}**

***// Funcție pentru resetarea jocului***

**void** resetGame**()**

**{**

gameOver **=** **false;**

fallingObjectsY**.**clear**();**

fallingObjectsX**.**clear**();**

bulletYs**.**clear**();**

score **=** 0**;**

gameStartTime **=** 0**;**

**for** **(int** i **=** 0**;** i **<** 256**;** **++**i**)** **{**

keyStates**[**i**]** **=** **false;**

**}**

translateX **=** 0.0f**;**

translateY **=** **-**0.5f**;**

**}**

***// Funcție pentru desenarea scenei***

**void** draw**()** **{**

***// Șterge bufferul de culoare și resetează matricea de modelare-vizualizare***

glClear**(**GL\_COLOR\_BUFFER\_BIT**);**

glLoadIdentity**();**

***// Desenează fundalul jocului***

glColor4f**(**0.0f**,** 0.0f**,** 0.035f**,** 0.5f**);** ***// Albastru închis cu 50% opacitate***

glBegin**(**GL\_QUADS**);**

glVertex2f**(-**1.0f**,** **-**1.0f**);**

glVertex2f**(**1.0f**,** **-**1.0f**);**

glVertex2f**(**1.0f**,** 1.0f**);**

glVertex2f**(-**1.0f**,** 1.0f**);**

glEnd**();**

***// Desenează stelele care cad pe fundal***

**for** **(size\_t** i **=** 0**;** i **<** fallingStarsY**.**size**();** **++**i**)** **{**

glPushMatrix**();**

glTranslatef**(**fallingStarsX**[**i**],** fallingStarsY**[**i**],** 0.0f**);**

glScalef**(**0.05f**,** 0.05f**,** 1.0f**);**

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0f**,** 1.0f**,** 1.0f**);**

glVertex3f**(-**0.60**,** 0.77**,** 0**);**

glVertex3f**(-**0.42**,** 0.77**,** 0**);**

glVertex3f**(-**0.58**,** 0.68**,** 0**);**

glVertex3f**(-**0.64**,** 1**,** 0**);**

glVertex3f**(-**0.68**,** 0.77**,** 0**);**

glVertex3f**(-**0.60**,** 0.77**,** 0**);**

glVertex3f**(-**0.68**,** 0.77**,** 0**);**

glVertex3f**(-**0.7**,** 0.68**,** 0**);**

glVertex3f**(-**0.86**,** 0.77**,** 0**);**

glVertex3f**(-**0.64**,** 0.63**,** 0**);**

glVertex3f**(-**0.7**,** 0.68**,** 0**);**

glVertex3f**(-**0.82**,** 0.43**,** 0**);**

glVertex3f**(-**0.64**,** 0.63**,** 0**);**

glVertex3f**(-**0.58**,** 0.68**,** 0**);**

glVertex3f**(-**0.51**,** 0.43**,** 0**);**

glEnd**();**

glColor3f**(**1.0f**,** 1.0f**,** 1.0f**);**

glBegin**(**GL\_POLYGON**);**

glVertex3f**(-**0.60**,** 0.77**,** 0**);**

glVertex3f**(-**0.68**,** 0.77**,** 0**);**

glVertex3f**(-**0.7**,** 0.68**,** 0**);**

glVertex3f**(-**0.64**,** 0.63**,** 0**);**

glVertex3f**(-**0.58**,** 0.68**,** 0**);**

glEnd**();**

glPopMatrix**();**

**}**

***// Desenează obiectele care cad***

glColor3f**(**1.0**,** 0.0**,** 0.0**);** ***// Roșu***

**for** **(size\_t** i **=** 0**;** i **<** fallingObjectsY**.**size**();** **++**i**)** **{**

glPushMatrix**();**

glTranslatef**(**fallingObjectsX**[**i**],** fallingObjectsY**[**i**],** 0.0f**);**

glRotatef**(**180.0f**,** 0.0f**,** 0.0f**,** 1.0f**);**

glScalef**(**0.1f**,** 0.1f**,** 1.0f**);**

glBegin**(**GL\_QUADS**);**

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glVertex2f**(-**0.1f**,** 0.0f**);**

glVertex2f**(-**0.45f**,** 0.1f**);**

glVertex2f**(-**0.1f**,** 0.25f**);**

glVertex2f**(**0.1f**,** 0.0f**);**

glVertex2f**(**0.45f**,** 0.1f**);**

glVertex2f**(**0.1f**,** 0.25f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(**0.0f**,** 0.5f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glVertex2f**(-**0.1f**,** **-**0.1f**);**

glVertex2f**(-**0.3f**,** **-**0.4f**);**

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.1f**);**

glVertex2f**(**0.3f**,** **-**0.4f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glEnd**();**

glPopMatrix**();**

**}**

***// Desenează jucătorul***

glTranslatef**(**translateX**,** translateY**,** 0.0f**);**

glScalef**(**0.175f**,** 0.175f**,** 1.0f**);**

glBegin**(**GL\_QUADS**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

glVertex2f**(-**0.1f**,** 0.0f**);**

glVertex2f**(-**0.45f**,** 0.1f**);**

glVertex2f**(-**0.1f**,** 0.25f**);**

glVertex2f**(**0.1f**,** 0.0f**);**

glVertex2f**(**0.45f**,** 0.1f**);**

glVertex2f**(**0.1f**,** 0.25f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

glVertex2f**(-**0.1f**,** 0.3f**);**

glVertex2f**(**0.1f**,** 0.3f**);**

glVertex2f**(**0.0f**,** 0.5f**);**

glEnd**();**

glBegin**(**GL\_TRIANGLES**);**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

glVertex2f**(-**0.1f**,** **-**0.1f**);**

glVertex2f**(-**0.3f**,** **-**0.4f**);**

glVertex2f**(-**0.1f**,** **-**0.3f**);**

glVertex2f**(**0.1f**,** **-**0.1f**);**

glVertex2f**(**0.3f**,** **-**0.4f**);**

glVertex2f**(**0.1f**,** **-**0.3f**);**

glEnd**();**

***// Desenează gloanțele trase de jucător***

**for** **(size\_t** i **=** 0**;** i **<** bulletYs**.**size**();** **++**i**)** **{**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

glPointSize**(**5.5f**);**

glBegin**(**GL\_POINTS**);**

glVertex2f**(**0**,** bulletYs**[**i**]);**

glEnd**();**

**}**

***// Afisează scorul pe ecran***

glMatrixMode**(**GL\_PROJECTION**);**

glPushMatrix**();**

glLoadIdentity**();**

gluOrtho2D**(-**1.0**,** 1.0**,** **-**1.0**,** 1.0**);**

glMatrixMode**(**GL\_MODELVIEW**);**

glPushMatrix**();**

glLoadIdentity**();**

glColor3f**(**1.0**,** 1.0**,** 1.0**);**

**char** scoreText**[**20**];**

sprintf\_s**(**scoreText**,** **sizeof(**scoreText**),** "Scor: %d"**,** score**);**

glScalef**(**2.0f**,** 2.0f**,** 1.0f**);**

renderBitmapString**(**0.35f**,** 0.45f**,** GLUT\_BITMAP\_HELVETICA\_18**,** scoreText**);**

glPopMatrix**();**

glMatrixMode**(**GL\_PROJECTION**);**

glPopMatrix**();**

glMatrixMode**(**GL\_MODELVIEW**);**

***// Afiseaza ecranul de Game Over, daca este cazul***

**if** **(**gameOver**)**

**{**

drawGameOverScreen**();**

**}**

***// Schimba bufferele pentru a afisa rezultatul***

glutSwapBuffers**();**

**}**

***// Funcție pentru generarea obiectelor care cad***

**void** spawnFallingObjects**()** **{**

**int** numObjects **=** rand**()** **%** 5 **+** 1**;**

**for** **(int** i **=** 0**;** i **<** numObjects**;** **++**i**)** **{**

**float** randomX**,** randomY**;**

***// Generate random position until a non-overlapping position is found***

**do** **{**

randomX **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;**

randomY **=** 1.0f**;**

***// Check if the new position overlaps with any existing falling objects***

overlapping **=** **false;**

**for** **(size\_t** i **=** 0**;** i **<** fallingObjectsY**.**size**();** **++**i**)** **{**

**if** **(**fabs**(**randomX **-** fallingObjectsX**[**i**])** **<** 0.2f **&&** fabs**(**randomY **-** fallingObjectsY**[**i**])** **<** 0.2f**)** **{**

overlapping **=** **true;**

**break;**

**}**

**}**

***// If overlapping, adjust the position and repeat the process***

**if** **(**overlapping**)** **{**

***// Adjust position (for example, move vertically)***

randomX **-=** 0.2f**;** ***// Move down by a certain amount***

**}**

**}** **while** **(**overlapping**);**

***// Add the new falling object with the adjusted position***

fallingObjectsY**.**push\_back**(**randomY**);**

fallingObjectsX**.**push\_back**(**randomX**);**

**}**

**}**

***// Funcție pentru generarea stelelor care cad***

**void** spawnFallingStars**()** **{**

**if** **(**fallingStarsY**.**empty**())** **{**

***// Generate stars randomly on the screen at the start of the program***

**int** numStars **=** 30**;**

**for** **(int** i **=** 0**;** i **<** numStars**;** **++**i**)** **{**

**float** randomX **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;**

**float** randomY **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;** ***// Generate Y positions randomly across the screen***

fallingStarsX**.**push\_back**(**randomX**);**

fallingStarsY**.**push\_back**(**randomY**);**

**}**

**}**

**else** **{**

***// Generate new stars at the top of the screen as they begin to fall***

**int** numStars **=** rand**()** **%** 10 **+** 1**;** ***// Generate fewer stars as new ones are added to avoid overcrowding***

**for** **(int** i **=** 0**;** i **<** numStars**;** **++**i**)** **{**

**float** randomX **=** **static\_cast<float>(**rand**())** **/** **static\_cast<float>(**RAND\_MAX**)** **\*** 1.8f **-** 0.9f**;**

**float** randomY **=** 1.0f**;** ***// Generate stars at the top of the screen***

fallingStarsX**.**push\_back**(**randomX**);**

fallingStarsY**.**push\_back**(**randomY**);**

**}**

**}**

**}**

***// Funcție pentru actualizarea jocului***

**void** update**(int** value**)** **{**

**if** **(!**gameOver**)** **{**

**if** **(**keyStates**[**'w'**]** **&&** keyStates**[**'a'**])** **{**

translateY **+=** translationSpeed **\*** 0.7071f**;**

translateX **-=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **if** **(**keyStates**[**'w'**]** **&&** keyStates**[**'d'**])** **{**

translateY **+=** translationSpeed **\*** 0.7071f**;**

translateX **+=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **if** **(**keyStates**[**'s'**]** **&&** keyStates**[**'a'**])** **{**

translateY **-=** translationSpeed **\*** 0.7071f**;**

translateX **-=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **if** **(**keyStates**[**'s'**]** **&&** keyStates**[**'d'**])** **{**

translateY **-=** translationSpeed **\*** 0.7071f**;**

translateX **+=** translationSpeed **\*** 0.7071f**;**

**}**

**else** **{**

**if** **(**keyStates**[**'w'**])** **{**

translateY **+=** translationSpeed**;**

**}**

**if** **(**keyStates**[**'s'**])** **{**

translateY **-=** translationSpeed**;**

**}**

**if** **(**keyStates**[**'a'**])** **{**

translateX **-=** translationSpeed**;**

**}**

**if** **(**keyStates**[**'d'**])** **{**

translateX **+=** translationSpeed**;**

**}**

**}**

**if** **(**translateX **<** **-**0.90f**)** translateX **=** **-**0.90f**;**

**if** **(**translateX **>** 0.90f**)** translateX **=** 0.90f**;**

**if** **(**translateY **<** **-**0.90f**)** translateY **=** **-**0.90f**;**

**if** **(**translateY **>** 0.90f**)** translateY **=** 0.90f**;**

**if** **(**gameStartTime **==** 0**)** **{**

spawnFallingObjects**();**

spawnFallingStars**();**

gameStartTime **=** clock**();**

**}**

**clock\_t** currentTime **=** clock**();**

**float** elapsedTime **=** **static\_cast<float>(**currentTime **-** gameStartTime**)** **/** CLOCKS\_PER\_SEC**;**

**float** elapsetShotTime **=** **static\_cast<float>(**currentTime **-** lastShotTime**)** **/** CLOCKS\_PER\_SEC**;**

**if** **(**elapsedTime **>** spawnInterval**)** **{**

spawnFallingObjects**();**

spawnFallingStars**();**

gameStartTime **=** currentTime**;**

**if** **(**elapsedTime **>** harderOverTime**)** **{**

spawnInterval **\*=** 0.999f**;**

harderOverTime **+=** 30.0f**;**

**}**

**}**

**if** **(**isShooting **&&** elapsetShotTime **>** cooldownDuration**)** **{**

bulletYs**.**push\_back**(**translateY **+** 0.80f**);**

lastShotTime **=** currentTime**;**

**}**

**for** **(size\_t** i **=** 0**;** i **<** bulletYs**.**size**();** **++**i**)** **{**

**float** bulletRadius **=** 0.075f**;** ***// Adjust as needed***

**for** **(size\_t** j **=** 0**;** j **<** fallingObjectsY**.**size**();** **++**j**)** **{**

**float** fallingObjectRadius **=** 0.125f**;** ***// Adjust as needed***

**if** **(**bulletYs**[**i**]** **+** bulletRadius **>** fallingObjectsY**[**j**]** **-** fallingObjectRadius **&&**

bulletYs**[**i**]** **-** bulletRadius **<** fallingObjectsY**[**j**]** **+** fallingObjectRadius **&&**

translateX **+** bulletRadius **>** fallingObjectsX**[**j**]** **-** fallingObjectRadius **&&**

translateX **-** bulletRadius **<** fallingObjectsX**[**j**]** **+** fallingObjectRadius**)** **{**

bulletYs**.**erase**(**bulletYs**.**begin**()** **+** i**);**

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** j**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** j**);**

score **+=** 10**;**

**--**i**;**

**break;**

**}**

**}**

**}**

**for** **(size\_t** j **=** 0**;** j **<** fallingObjectsY**.**size**();** **++**j**)** **{**

**if** **(**fabs**(**translateY **-** fallingObjectsY**[**j**])** **<** 0.1f **&&** fabs**(**translateX **-** fallingObjectsX**[**j**])** **<** 0.1f**)** **{**

gameOver **=** **true;**

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** j**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** j**);**

**break;**

**}**

**}**

**for** **(size\_t** i **=** 0**;** i **<** fallingStarsY**.**size**();** **++**i**)** **{**

fallingStarsY**[**i**]** **+=** fallingStarSpeed**;**

**if** **(**fallingStarsY**[**i**]** **<** **-**1.0f**)** **{**

fallingStarsY**.**erase**(**fallingStarsY**.**begin**()** **+** i**);**

fallingStarsX**.**erase**(**fallingStarsX**.**begin**()** **+** i**);**

**--**i**;**

**}**

**}**

**for** **(size\_t** i **=** 0**;** i **<** bulletYs**.**size**();** **++**i**)** **{**

bulletYs**[**i**]** **+=** bulletSpeed**;**

**if** **(**bulletYs**[**i**]** **>** 50.0f**)** **{**

bulletYs**.**erase**(**bulletYs**.**begin**()** **+** i**);**

**--**i**;**

**}**

**}**

**for** **(size\_t** i **=** 0**;** i **<** fallingObjectsY**.**size**();** **++**i**)** **{**

fallingObjectsY**[**i**]** **+=** fallingObjectSpeed**;**

**if** **(**fallingObjectsY**[**i**]** **<** **-**1.0f**)** **{**

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** i**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** i**);**

**--**i**;**

score **-=** 10**;**

**}**

**}**

**if** **(**isShooting**)** **{**

**for** **(size\_t** j **=** 0**;** j **<** fallingObjectsY**.**size**();** **++**j**)** **{**

**float** dist **=** distance**(**translateX**,** translateY**,** fallingObjectsX**[**j**],** fallingObjectsY**[**j**]);**

**if** **(**dist **<** destructionRadius**)** **{**

***// Destroy the falling object***

fallingObjectsY**.**erase**(**fallingObjectsY**.**begin**()** **+** j**);**

fallingObjectsX**.**erase**(**fallingObjectsX**.**begin**()** **+** j**);**

score **+=** 10**;**

**--**j**;** ***// Adjust index since falling object is removed***

**}**

**}**

**}**

**}**

**else** **{**

**if** **(**keyStates**[**'r'**]** **||** keyStates**[**'R'**])** **{**

resetGame**();**

**}**

**}**

***// Reapelează funcția de actualizare la fiecare cadru***

glutPostRedisplay**();**

glutTimerFunc**(**16**,** update**,** 0**);**

**}**

***// Funcție pentru tratarea apăsării tastei***

**void** keyPressed**(unsigned** **char** key**,** **int** x**,** **int** y**)** **{**

keyStates**[**key**]** **=** **true;**

**if** **(**key **==** 'r' **||** key **==** 'R'**)** **{**

resetGame**();**

**}**

**}**

***// Funcție pentru tratarea eliberării tastei***

**void** keyReleased**(unsigned** **char** key**,** **int** x**,** **int** y**)** **{**

keyStates**[**key**]** **=** **false;**

**}**

***// Funcție pentru tratarea evenimentelor mouse-ului***

**void** mouse**(int** button**,** **int** state**,** **int** x**,** **int** y**)** **{**

**if** **(**button **==** GLUT\_LEFT\_BUTTON**)** **{**

**if** **(**state **==** GLUT\_DOWN**)** **{**

isShooting **=** **true;**

**}**

**else** **if** **(**state **==** GLUT\_UP**)** **{**

isShooting **=** **false;**

**}**

**}**

**}**

***// Funcția principală***

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

glutInit**(&**argc**,** argv**);**

glutInitDisplayMode**(**GLUT\_DOUBLE **|** GLUT\_RGB**);**

glutInitWindowSize**(**800**,** 600**);**

glutInitWindowPosition**(**150**,** 150**);**

glutCreateWindow**(**"Proiect"**);**

init**();**

glutDisplayFunc**(**draw**);**

glutKeyboardFunc**(**keyPressed**);**

glutKeyboardUpFunc**(**keyReleased**);**

glutMouseFunc**(**mouse**);**

glutTimerFunc**(**16**,** update**,** 0**);**

glutMainLoop**();**

**return** 0**;**

**}**