|  |  |  |
| --- | --- | --- |
|  | **Universitatea *Transilvania* din Brașov**  **Facultatea de Inginerie Electrică și Știința Calculatoarelor**  **Departamentul Automatică și Tehnologia Informației** |  |

**PROIECT EGC**

**PONG**

Munteanu Maria-Cristiana

Tehnologia Informației, anul 2, grupa 4LF322

**BRAȘOV, 2024**

Cuprins

[1. Introducere 2](#_Toc166591808)

[1.1. Scurtă introducere despre C++ 2](#_Toc166591809)

[1.2. Scurtă introducere despre biblioteca OpenGL 2](#_Toc166591810)

[2. Proiect: jocul Pong 3](#_Toc166591811)

[2.1 Scurt istoric al jocului Pong 3](#_Toc166591812)

[2.2 (re)Crearea jocului Pong cu ajutorul bibliotecii OpenGL 3](#_Toc166591813)

[2.3 Probeleme întâmpinate la realizarea proiectului 14](#_Toc166591814)

[2.4 Concluzii 14](#_Toc166591815)

[3. Codul complet al proiectului 15](#_Toc166591816)

[4. Capturi de ecran la rularea programului 19](#_Toc166591817)

[5. Bibliografie 21](#_Toc166591818)

# Introducere

## Scurtă introducere despre C++

Limbajul de programare C++ este un limbaj puternic și versatil, utilizat în mod extensiv în dezvoltarea software. Este o extensie a limbajului C și oferă suport pentru programare orientată pe obiecte, programare generică și programare procedurală. C++ este folosit în diverse domenii, inclusiv dezvoltarea de aplicații desktop, dezvoltarea de jocuri, programarea sistemelor înglobate și multe altele.

Limbajul C++ a fost creat de Bjarne Stroustrup în anul 1983, ca o extensie a limbajului C. El a dorit să adauge suport pentru programarea orientată pe obiecte în limbajul C, fără a pierde eficiența și controlul oferite de acesta. Astfel, a luat naștere C++, o limbaj de programare care combină caracteristicile de nivel înalt ale programării orientate pe obiecte cu performanța și controlul de nivel scăzut oferite de limbajul C.

C++ aduce numeroase caracteristici și avantaje în dezvoltarea software precum programarea orientată pe obiecte, performanța, compatibilitatea cu limbajul C și bogăția de biblioteci.

## Scurtă introducere despre biblioteca OpenGL

OpenGL a fost creată de firma Silicon Graphics în anul 1992 cu scopul de a oferi programatorilor o bibliotecă portabilă pentru aplicații 2D/3D. În zilele noastre este una dintre tehnologiile cele mai des implementate în diverse aplicații și domenii precum grafica asistată de calculator, realitate virtuală, vizualizare științifică, simulări de zboruri sau jocuri video.

Biblioteca se împarte în trei sub-biblioteci cu funcționalități distincte:

* Biblioteca OpenGL propriu-zisă care oferă tot ce este necesar pentru a utiliza funcțiile fundamentale,
* Biblioteca GLU (OpenGL Utility Library) ce conține funcții utilitare pentru realizarea rapidă a unor operații des folosite precum crearea și manipularea matricilor de vizualizare și proiecție și desenarea formelor geometrice de bază,
* Biblioteca GLX (OpenGL Extension to the X Windows System) permite OpenGL-ului să interacționeze cu serverul de interfață grafică X Windows.

Pe lângă acestea, biblioteca GLUT (OpenGL Utility Toolkit) oferă o interfață independentă de sistemul de operare, suficient pentru crearea de aplicații grafice protabile (independente de platformă.

# Proiect: jocul Pong

În cadrul acestui proiect am decis să realizez o recreere a jocului video Pong. În cele ce urmează vă voi prezenta o scurtă istorie a acestui joc, etapele re-realizării acestuia, oferindu-vă explicații ale codului și capturi de ecran. La finalul acestei documentații se va afla codul complet al acestui proiect.

## 2.1 Scurt istoric al jocului Pong

Un joc video lansat în 1972 de Atari Inc., Pong imită un meci de tenis sau ping-pong. Este primul joc lansat de Atari, contribuind astfel la începutul industriei jocurilor video. Pong se joacă într-un mod foarte simplu: jucătorul folosește paleta pentru a lovi mingea, trecând-o pe partea adversarului pentru a obține un scor.

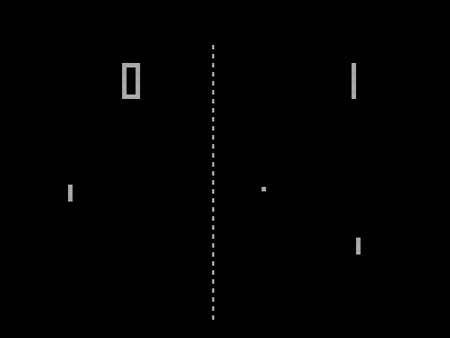


Fig. 1 Captură de ecran a jocului Pong (1972)

Până în 1976, a devenit extrem de popular, generând multiple variante produse de diverse companii, atât pentru arcade cât și pentru console. Aceste companii adăugau adesea modificări minore sau majore la joc pentru a evita asocierile directe cu Atari, rezultând în titluri precum Pong-Tron, Super Pong, Super Pong Ten, Pong Double, etc. Popularitatea jocului a scăzut după 1976, dar jocul a rămas cunoscut ca unul dintre cele mai influente jocuri video din istorie.

## 2.2 (re)Crearea jocului Pong cu ajutorul bibliotecii OpenGL

După cum am precizat anterior, în cele ce urmează voi ilustra, prin secvențe de cod și capturi de ecran, modul prin care am recreat acest joc istoric folosind limbajul C++ și a bibliotecii OpenGL.

Secvență de cod 1

|  |
| --- |
| #include <GL/glut.h>  #include <bits/stdc++.h> |

Primul pas a fost includerea bibliotecilor OpenGL și GLUT pentru desenarea paletelor, mingilor și textului. Biblioteca <bits/stdc++.h> include toate funcțiile care se pot realiza pentru numere și șiruri de caractere. Am adăugat această bibliotecă, deoarece este utilă în calculul pozițiilor atât pentru elementele jocului (mingea și paletele), cât și pentru textele ce vor apărea în joc (precum scorul jucătorilor.

Secvență de cod 2

|  |
| --- |
| // Game states  enum GameState { MENU, GAME, END, PAUSE };  // Current game state  GameState gameState = MENU; |

Această secvență de cod inițializează o enumerare (se poate observa cuvântul-cheie enum) prin care sunt definite stările jocului astfel:

* MENU: fiind inițializat în această stare în următoarea linie de cod, aceasta afișează un meniu de bun venit la rularea programului (detalii despre această funcție puțin mai încolo),
* GAME: aceasta este stare în care jocul începe. Sunt afișate paletele, mingea și textul care arată scorul fiecărui jucător,
* PAUSE: acastă stare se activează atunci când este apăsata tasta ESC în timpul stării anterioare. În timpul pauzei, mingea nu se mai deplasează, scorul nu mai este afișat, iar jucătorii nu vor mai putea mișca paletele. Se poate ieși din starea de pauză apăsând din nou tasta ESC,
* END: această stare apare la sfârșitul jocului. Pe ecran se va afișa jucătorul câștigător și două opțiuni, una de restart și una de ieșire (detalii puțin mai încolo).

Secvență de cod 3

|  |
| --- |
| // Define the initial positions and sizes of the paddles and the ball  float paddle1Y = 0.0f, paddle2Y = 0.0f;  float paddleWidth = 0.1f, paddleHeight = 0.4f;  float ballX = 0.0f, ballY = 0.0f, ballRadius = 0.05f;  float ballXSpeed = 0.005f, ballYSpeed = 0.002f;  // Define the dimensions of the screen  const int screenWidth = 800, screenHeight = 600; |

Variabilele reale (de tip float) definesc dimensiunile și coordonatele inițiale pentru palete și minge. Aceste coordonate reale sunt utile în definirea funcțiilor care vor desena aceste elemente.

Variabilele întregi (de tip int) definesc dimensiunea ecranului când codul va fi executat. Având în față cuvantul-cheie const, aceste variabile nu pot fi modificate în cod după inițializare, deoarece se va genera o eroare. Într-adevăr, se pot modifica doar în declararea acestora într-o secvență de genul:

Secvență de cod 4

|  |
| --- |
| const int screenWidth = 900, screenHeight = 650; |

dar o modificare ulterioară declarării precum:

Secvență de cod 5

|  |
| --- |
| screenHeight = 600; |

va genera datorită cuvântului cheie const, eroarea:



Fig. 2 Eroarea generată de modificările specificate

Secvență de cod 6

|  |
| --- |
| // Number of rounds and current score  int score1 = 0, score2 = 0;  // Winner player  int winner = 0; |

Variabilele score1 și score2, de tip întreg, reprezintă scorul fiecărul jucător care se vor incrementa atunci când mingea depășește marginea stângă sau marginea dreaptă (cea stângă aparținând de jucătorul 1, iar cea dreaptă de jucătorul 2).

Variabila winner, tot de tip întreg, va putea lua una din cele două valori:

* 1, dacă jucătorul 1 va câștiga jocul, sau
* 2, dacă jucătorul 2 va câștiga jocul.

Secvență de cod 7

|  |
| --- |
| // Function to draw a rectangle  void drawRectangle(float x, float y, float width, float height)  {  glPushMatrix();  glTranslatef(x, y, 0.0f);  glBegin(GL\_QUADS);  glVertex2f(-width / 2, -height / 2);  glVertex2f(width / 2, -height / 2);  glVertex2f(width / 2, height / 2);  glVertex2f(-width / 2, height / 2);  glEnd();  glPopMatrix();  } |

Această funcție are rolul de a desena un dreptunghi. Funcția va avea rolul de a desena mingea și paletele pentru joc. Conținutul acestei funcții utilizează următoarele funcții predefinite în biblioteca OpenGL:

* Funcția glPushMatrix(); salvează starea curentă a matricei de transformare. OpenGL folosește un sistem de matrici pentru a gestiona cum sunt transformate și poziționate obiectele în spațiul 3D (și în acest caz, 2D). Prin salvarea stării, putem modifica și aplica transformări fără a afecta alte desene din scena globală,
* Funcția glTranslatef(x, y, 0.0f); aplică o transformare de translație la matricea curentă, mutând toate desenele ulterioare cu x unități pe axa orizontală și y unități pe axa verticală. Al treilea parametru este 0.0f deoarece, în contextul unui joc 2D ca Pong, nu este necesară mișcarea pe axa z (adâncimea),
* Funcția glBegin(GL\_QUADS); începe definirea unui set de primitivi, în acest caz un „quad” (dreptunghi), care este format din patru vertex-uri (colțuri),
* Cele patru vertex-uri (glVertex2f), ințializate în această ordine: stânga-jos, dreapta-jos, dreapta-sus și stânga-sus, sunt definite relativ la centrul dreptunghiului, care este mutat la poziția (x, y) de funcția glTranslatef();. Astfel, dreptunghiul este centrat în punctul (x, y),
* Funcția glEnd(); marchează sfârșitul definirii primitivelor (în acest caz, sfârșitul quad-ului). OpenGL acum va procesa și va renderiza dreptunghiul conform vertex-urilor și transformărilor aplicate.
* Funcția glPopMatrix(); restabilește starea matricei de transformare la cea salvată anterior cu glPushMatrix();. Acest lucru este util pentru a asigura că transformările aplicate pentru acest dreptunghi nu afectează alte elemente ale scenei care sunt desenate ulterior.

Un exemplu de apelare a funcției ar fi:

Secvență de cod 8

|  |
| --- |
| drawRectangle(0.0f, 0.0f, 0.4f, 0.6f); |

Această apelare va genera următorul dreptunghi:

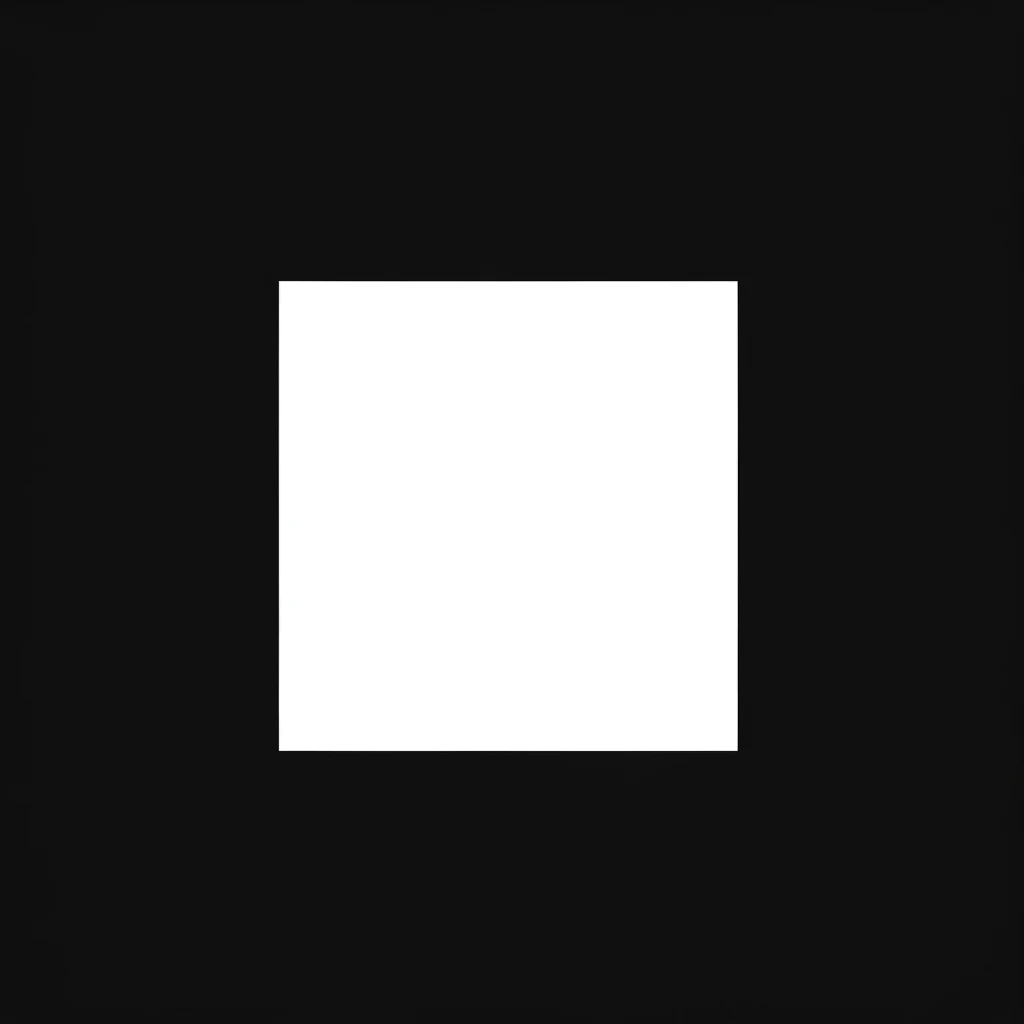


Fig. 3 Exemplu de dreptunghi generat cu funcția de desenare

În acest exemplu:

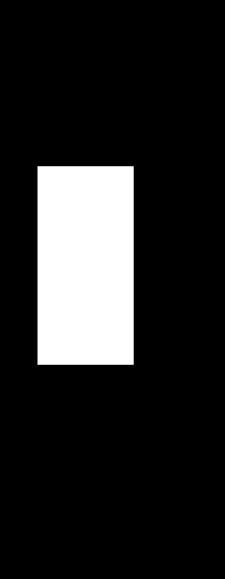
* 0.0f, 0.0f sunt coordonatele mijlocului dreptunghiului care este plasat în centrul ecranului,
* 0.4f, 0.6f reprezintă lățimea, respectiv înălțimea dreptunghiului.

Secvență de cod 9

|  |
| --- |
| // Function to draw the ball  void drawBall()  {  glColor3f(1.0f, 1.0f, 1.0f); // White color  drawRectangle(ballX, ballY, ballRadius \* 2, ballRadius \* 2);  }    // Function to draw the paddles  void drawPaddles()  {  glColor3f(1.0f, 1.0f, 1.0f); // White color  drawRectangle(-0.9f, paddle1Y, paddleWidth, paddleHeight);  drawRectangle(0.9f, paddle2Y, paddleWidth, paddleHeight);  } |

Aceste funcții au rolul de a desena mingea și paletele utilizând funcția de desenare anterior definită. De asemenea, funcția glColor3f(1.0f, 1.0f, 1.0f); colorează mingea și paletele în alb utilizând codul RGB pentru această culoare. După cum se poate observa, pentru desenare s-au utilizat coordonatele anterior definite. De asemenea, pentru paleta din stânga s-a setat coordonata -0.9f, iar pentru cea din dreapta coordonata 0.9f.

Paletele și mingea care sunt desenate în joc arată astfel:



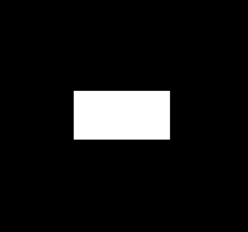


Fig. 4 Mingea generată în joc

Fig. 5 Paleta generată în joc

Secvență de cod 10

|  |
| --- |
| // Function to reset the ball's position  void resetBallPosition()  {  ballX = 0.0f;  ballY = 0.0f;  }  // Function to check game over  void checkGameOver()  {  if (score1 >= 5 || score2 >= 5)  {  gameState = END;  winner = (score1 > score2) ? 1 : 2;  }  }  // Function to update the position of the ball  void updateBall()  {  if (gameState != PAUSE)  {  ballX += ballXSpeed;  ballY += ballYSpeed;  // Check for collision with walls  if (ballY + ballRadius > 1.0f || ballY - ballRadius < -1.0f)  ballYSpeed = -ballYSpeed;  // Check for collision with paddles  if ((ballX - ballRadius < -0.85f && ballY > paddle1Y - paddleHeight / 2 && ballY < paddle1Y + paddleHeight / 2) ||  (ballX + ballRadius > 0.85f && ballY > paddle2Y - paddleHeight / 2 && ballY < paddle2Y + paddleHeight / 2))  ballXSpeed = -ballXSpeed;  // Check if ball went out of bounds  if (ballX + ballRadius < -1.0f)  {  score2++;  resetBallPosition();  checkGameOver();  }  else if (ballX - ballRadius > 1.0f)  {  score1++;  resetBallPosition();  checkGameOver();  }  }  } |

Prima funcție, resetBallPosition();, resetează poziția mingii astfel încât aceasta să fie plasată în centrul ecranului.

Funcția checkGameOver(); verifică care dintre jucători este câștigător. Condiția de câștig este ca unul dintre jucători să atingă scorul de 5 puncte.

Funcția updateBall(); este folosită pentru actualizarea mingii în joc. Prima dată, această funcție verifică dacă jocul nu este pus pe pauză, executând astfel liniile de cod care urmează după această condiție. Funcția actualizează poziția mingii în timpul jocului. Dacă mingea lovește un perete sau o paletă, viteza mingii este inversată, acest lucru indicând deplasarea acesteia în sens opus. În plus, dacă mingea depășește limita ecranului, scorul se va actualiza în funcție de limita care a fost depășită.

Secvență de cod 11

|  |
| --- |
| // Function to handle key presses  void handleKeypress(unsigned char key, int x, int y)  {  switch (gameState)  {  case MENU:  if (key == 13) // 13 is ASCII code for Enter  gameState = GAME;  else if (key == 27) // ESC key  exit(0);  break;  case GAME:  // Handling game keypress  if (key == 'w' && paddle1Y < 0.8)  paddle1Y += 0.1f;  else if (key == 's' && paddle1Y > -0.8)  paddle1Y -= 0.1f;  else if (key == 'i' && paddle2Y < 0.8)  paddle2Y += 0.1f;  else if (key == 'k' && paddle2Y > -0.8)  paddle2Y -= 0.1f;  else if (key == 27) // ESC key  gameState = PAUSE;  break;  case PAUSE:  if (key == 27) // ESC key  gameState = GAME;  break;  case END:  if (key == 'r' || key == 'R')  {  gameState = MENU;  score1 = 0;  score2 = 0;  resetBallPosition();  }  else if (key == 'e' || key == 'E')  exit(0);  break;  } |

Această funcție este concepută pentru a gestiona acțiunile jucătorilor bazate pe apăsările tastelor în timpul jocului. Funcția folosește un switch care determină starea curentă, făcând programul să răspundă corespunzător la intrările de la tastatură.

* MENU: dacă tasta Enter (care are codul ASCII 13) este apăsată, starea curentă devine GAME. Dacă se apasă tasta ESC (care are codul ASCII 27), programul se încheie și programul se oprește.
* GAME: În timpul acestei stări, funcția permite mișscarea verticală a paletelor. Tastele 'W' și 'S' controlează paleta jucătorului 1, prima tastă mutând paleta sus, iar cea de-a doua jos. Tastele 'I' și 'K' controlează paleta jucătorului 2 în mod similar. De asemenea, apăsarea tastei ESC punei jocul pe pauză, schimbând starea în PAUSE.
* PAUSE: apăsarea tastei ESC reia jocul, schimbând starea înapoi în GAME.
* END: în această stare sunt prezente două opțiuni. Tasta 'R' repornește jocul, resetând scorul și poziția mingii, și revine la meniu. Apăsarea tastei 'E' închide jocul.

Secvență de cod 12

|  |
| --- |
| // Function to update the screen  void update(int value)  {  if (gameState == GAME) updateBall();  glutPostRedisplay();  glutTimerFunc(10, update, 0);  } |

Această funcție este apelată periodic pentru a actualiza starea jocului și pentru a redesena ecranul. Prima condiție actualizează poziția mingii dacă jocul este în starea GAME. Funcția glutPostRedisplay(); redesenează ecranul, declanșând și funcția de desenare draw(); care gestionează redesenarea. Apelul glutTimerFunc(10, update, 0); setează un temporizator care apelează din nou funcția update(); după un interval de 10 milisecunde, asigurând rata constantă de actualizare a jocului.

Secvență de cod 13

|  |
| --- |
| // Function to draw text  void drawText(const char\* text, float x, float y)  {  glPushMatrix();  glColor3f(1.0f, 1.0f, 1.0f);  glRasterPos2f(x, y);  while (\*text)  {  glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, \*text);  text++;  }  glPopMatrix();  }  // Function to draw the scene  void drawMenu()  {  drawText("Welcome to Pong! =)", -0.3f, 0.4f);  drawText("First one to 5 points wins!", -0.3f, 0.3f);  drawText("Press Enter to start", -0.3f, 0.1f);  drawText("Press ESC to exit", -0.3f, 0.0f);  }  // Function to draw the pause logo in the top left corner  void drawPauseLogo()  {  glColor3f(1.0f, 1.0f, 1.0f); // Set color to white  // Calculate normalized coordinates for the top left corner  float normX = -1.0f + 20.0f / screenWidth; // Adjust x position a bit right from the edge  float normY = 1.0f - 20.0f / screenHeight; // Adjust y position a bit down from the top  drawText("PAUSE", normX, normY);  } |

Funcția drawText(); este utilizată pentru a afișa diverse texte pe ecran. Apelurile glPushMatrix(); și glPopMatrix(); salvează și apoi restabilesc starea curentă a matricei de transformare, asigurându-se că afișarea textului nu afectează alte elemente grafice. Apelul glRasterPos2f(x, y); setează poziția la care va începe afișarea textului (coordonatele (x, y)). Bucla while(\*text) este parcursă pentru fiecare caracter din textul ce trebuie desenat. Comanda glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, \*text); afișează un caracter la poziția curentă folosind un font bitmap predefinit (8 x 13 pixeli)

Funcția drawMenu(); desenează patru texte pentru starea MENU a jocului, apelând de patru ori funcția precedentă.

Funcția drawPauseLogo(); desenează textul „PAUSE” când starea jocului este PAUSE. S-au folosit coordonate normalizate pentru afișarea adecvată a textului desenat în limitele ecranului setate la începutul codului.

Secvență de cod 14

|  |
| --- |
| // Main drawing function  void draw()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glLoadIdentity();  if (gameState == MENU) drawMenu();  else if (gameState == GAME)  {  char scoreDisplay[50];  sprintf(scoreDisplay, "Player 1: %d Player 2: %d", score1, score2);  drawText(scoreDisplay, -0.1f, 0.9f);  drawBall();  drawPaddles();  }  else if (gameState == PAUSE)  {  drawPauseLogo(); // Draw pause logo when the game is paused  drawBall();  drawPaddles();  }  else if (gameState == END)  {  char winText[30];  sprintf(winText, "Player %d wins! :-p", winner);  drawText(winText, -0.1f, 0.0f);  drawText("Press R to Restart", -0.15f, -0.1f);  drawText("Press E to Exit", -0.12f, -0.2f);  }  glutSwapBuffers();  } |

Aceasta este funcția principală de desenare, fiind utilizată pentru a actualiza vizualizarea ecranului în funcție de starea curentă a jocului. Task-urile pe care le realizează această funcție în cadrul proiectului sunt:

* **Curățarea ecranului**: Prin apelul glClear(GL\_COLOR\_BUFFER\_BIT); se șterge ecranul, curățând buffer-ul de culoare pentru a pregăti un nou cadru de desenat. Acest apel este esențial pentru evitarea suprapunerii imaginilor de la un cadru la altul.
* **Resetarea transformărilor**: Apelul glLoadIdentity(); resetează matricea curentă la matricea identitate, eliminând efectele transformărilor anterioare și stabilind un nou punct de referință pentru desenare.
* **Desenarea în funcție de starea jocului**: Dacă jocul este în starea MENU, funcția drawMenu(); este apelată pentru a desena meniul principal. În timpul stării GAME se afișează și se formatează scorul jucătorilor, iar minge și paletele sunt desenate prin intermediul funcțiilor drawBall(); și drawPaddles();. Când jocul este în starea PAUSE, se afișează logo-ul de pauză, iar în starea END sunt afișate un mesaj care indică câștigătorul și opțiuni pentru repornire sau ieșire.
* **Schimbarea buffer-elor**: Apelul glutSwapBuffers(); schimbă buffer-urile în contextul unei ferestre cu buffering dublu. În această situație se comută între două zone de memorie care stocază imagini: una este vizibilă pe ecran, iar cealalta este în care se desenează următorul cadru. Prin aceasta metodă, utilizatorii văd o imagine completă în timp ce următoarea este pregătită în fundal, eliminând astfel problemele de aliasing și oferind o animație mai lină.

Secvență de cod 15

|  |
| --- |
| // Main function  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutInitWindowSize(screenWidth, screenHeight);  glutCreateWindow("Pic-Poc :)");  glutDisplayFunc(draw);  glutKeyboardFunc(handleKeypress);  glutTimerFunc(10, update, 0);  glutMainLoop();  return 0;  } |

Această funcție inițializează și configurează fereastra de afișare folosind biblioteca GLUT, intrând apoi în bucla principală a jocului. Pașii pe care îi parcurge această funcție sunt:

* **Inițializarea GLUT**: Comanda glutInit(&argc, argv); inițializează bibliteca GLUT, permițând aplicației să primească argumentele de la linia de comandă, fiind utilă pentru configurările avansate ale GLUT.
* **Setarea modului de afișare**: glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); este comanda care configurează fereastra GLUT pentru utilizarea unui mod de afișare cu buffer dublu și color în format RGB (Red, Green, Blue). Cum am precizat anterior, buffering-ul dublu ajută la prevenirea efectelor de aliasing și asigură o tranziție lină între cadre.
* **Dimensiunile ferestrei**: Apelul glutInitWindowSize(screenWidth, screenHeight); setează dimensiunile ferestrei de afișare conform variabilelor screenWidth și screenHeight setate la începutul codului.
* **Crearea ferestrei**: glutCreateWindow("Pic-Poc :)"); deschide o fereastră cu titlul ”Pic-Poc :)” unde se va desfășura jocul.
* **Funcțiile de callback**: Apelul glutDisplayFunc(draw); indică funcția draw(); ca fiind funcția de callback pentru redesenarea ferestrei. Comanda glutKeyboardFunc(handleKeypress); setează funcția handleKeypress(); pentru a gestiona evenimentele de la tastatură. Comada glutTimerFunc(10, update, 0);, explicată anterior, setează un temporizator care actualizează ecranul la fiecare 10 milisecunde.
* **Intrarea în bucla principală GLUT:** Apelul glutMainLoop(); intră în bucla de procesare a evenimentelor GLUT, care gestionează toate intrările de la utilizator, actualizările de afișsare și alte evenimente. Această buclă rulează până când programul este închis.

Această funcție este punctul de intrare în jocul Pong, configurând mediul grafic și gestionând fluxul de control al jocului.

## 2.3 Probeleme întâmpinate la realizarea proiectului

Inițial, am dorit să adaug în meniu câteva opțiuni pentru alegerea numărului de runde (de exemplu, un meci să aibă fie 3, fie 5, fie 7, fie 10 runde) prin intermediul unui click. În timpul jocului, la scor egal, programul adăuga runde în plus, lungind astfel jocul. Am încercat să controlez și să previn acest eveniment, însă, din păcate, programul făcea la fel. Pentru o implementare mai simplă a jocului, am decis ca primul jucător care atinge scorul de 5 puncte să câștige.

Un alt bug micuț apare atunci când paleta lovește spatele mingii, făcând mingea să se ciocnească între paletă și marginea ecranului. Deși probabilitatea ca acest bug să se declanșeze este mică, încă nu am reușit să elimin acest eveniment.

Iar ca o mică notă haioasă, am decis să numesc fereastra „Pic-Poc”, deoarece mi se pare că sună mai drăguț decât „Ping-Pong”

## 2.4 Concluzii

În documentația aceasta am parcurs o varietate de funcții și metode utilizate în recrearea jocului Pong, un clasic atemporal al industriei jocurilor video. Am explorat cum fiecare componentă a codului contribuie la dinamica jocului, de la inițializarea stărilor jocului, gestionarea interacțiunilor utilizatorului, până la logica de mișcare a bilei și scorurile jucătorilor. Implementarea acestor funcții demonstrează nu doar fundamentele programării în OpenGL, dar și cum gestionarea eficientă a statelor și interactivitatea sunt esențiale în dezvoltarea jocurilor.

Pong nu este doar un simplu joc, ci o parte a istoriei tehnologice, reprezentând unul dintre primele jocuri video care au captivat publicul larg. Prin înțelegerea și recrearea acestui joc, dezvoltatorii nu doar că își îmbunătățesc abilitățile tehnice, dar se și conectează la rădăcinile și evoluția tehnologiei în divertisment. Încurajăm fiecare cititor să utilizeze acest exemplu nu doar ca un exercițiu de învățare, ci și ca o sursă de inspirație pentru a explora noi posibilități creative în programarea jocurilor, îmbinând astfel istoria cu inovația în proiectele lor viitoare.

# Codul complet al proiectului

#include <GL/glut.h>

#include <bits/stdc++.h>

// Game states

enum GameState { MENU, GAME, END, PAUSE };

// Current game state

GameState gameState = MENU;

// Define the initial positions and sizes of the paddles and the ball

float paddle1Y = 0.0f, paddle2Y = 0.0f;

float paddleWidth = 0.1f, paddleHeight = 0.4f;

float ballX = 0.0f, ballY = 0.0f, ballRadius = 0.05f;

float ballXSpeed = 0.005f, ballYSpeed = 0.002f;

// Define the dimensions of the screen

const int screenWidth = 800, screenHeight = 600;

// Number of rounds and current score

int score1 = 0, score2 = 0;

// Winner player

int winner = 0;

// Function to draw a rectangle

void drawRectangle(float x, float y, float width, float height)

{

glPushMatrix();

glTranslatef(x, y, 0.0f);

glBegin(GL\_QUADS);

glVertex2f(-width / 2, -height / 2);

glVertex2f(width / 2, -height / 2);

glVertex2f(width / 2, height / 2);

glVertex2f(-width / 2, height / 2);

glEnd();

glPopMatrix();

}

// Function to draw the ball

void drawBall()

{

glColor3f(1.0f, 1.0f, 1.0f); // White color

drawRectangle(ballX, ballY, ballRadius \* 2, ballRadius \* 2);

}

// Function to reset the ball's position

void resetBallPosition()

{

ballX = 0.0f;

ballY = 0.0f;

}

// Function to check game over

void checkGameOver()

{

if (score1 >= 5 || score2 >= 5)

{

gameState = END;

winner = (score1 > score2) ? 1 : 2;

}

}

// Function to draw the paddles

void drawPaddles()

{

glColor3f(1.0f, 1.0f, 1.0f); // White color

drawRectangle(-0.9f, paddle1Y, paddleWidth, paddleHeight);

drawRectangle(0.9f, paddle2Y, paddleWidth, paddleHeight);

}

// Function to handle key presses

void handleKeypress(unsigned char key, int x, int y)

{

switch (gameState)

{

case MENU:

if (key == 13) // 13 is ASCII code for Enter

gameState = GAME;

else if (key == 27) // ESC key

exit(0);

break;

case GAME:

// Handling game keypress

if (key == 'w' && paddle1Y < 0.8)

paddle1Y += 0.1f;

else if (key == 's' && paddle1Y > -0.8)

paddle1Y -= 0.1f;

else if (key == 'i' && paddle2Y < 0.8)

paddle2Y += 0.1f;

else if (key == 'k' && paddle2Y > -0.8)

paddle2Y -= 0.1f;

else if (key == 27) // ESC key

gameState = PAUSE;

break;

case PAUSE:

if (key == 27) // ESC key

gameState = GAME;

break;

case END:

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

{

gameState = MENU;

score1 = 0;

score2 = 0;

resetBallPosition();

}

else if (key == 'e' || key == 'E')

exit(0);

break;

}

}

// Function to update the position of the ball

void updateBall()

{

if (gameState != PAUSE)

{

ballX += ballXSpeed;

ballY += ballYSpeed;

// Check for collision with walls

if (ballY + ballRadius > 1.0f || ballY - ballRadius < -1.0f)

ballYSpeed = -ballYSpeed;

// Check for collision with paddles

if ((ballX - ballRadius < -0.85f && ballY > paddle1Y - paddleHeight / 2 && ballY < paddle1Y + paddleHeight / 2) ||

(ballX + ballRadius > 0.85f && ballY > paddle2Y - paddleHeight / 2 && ballY < paddle2Y + paddleHeight / 2))

ballXSpeed = -ballXSpeed;

// Check if ball went out of bounds

if (ballX + ballRadius < -1.0f)

{

score2++;

resetBallPosition();

checkGameOver();

}

else if (ballX - ballRadius > 1.0f)

{

score1++;

resetBallPosition();

checkGameOver();

}

}

}

// Function to update the screen

void update(int value)

{

if (gameState == GAME) updateBall();

glutPostRedisplay();

glutTimerFunc(10, update, 0);

}

// Function to draw text

void drawText(const char\* text, float x, float y)

{

glPushMatrix();

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

glRasterPos2f(x, y);

while (\*text)

{

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, \*text);

text++;

}

glPopMatrix();

}

// Function to draw the scene

void drawMenu()

{

drawText("Welcome to Pong! =)", -0.3f, 0.4f);

drawText("First one to 5 points wins!", -0.3f, 0.3f);

drawText("Press Enter to start", -0.3f, 0.1f);

drawText("Press ESC to exit", -0.3f, 0.0f);

}

// Function to draw the pause logo in the top left corner

void drawPauseLogo()

{

glColor3f(1.0f, 1.0f, 1.0f); // Set color to white

// Calculate normalized coordinates for the top left corner

float normX = -1.0f + 20.0f / screenWidth; // Adjust x position a bit right from the edge

float normY = 1.0f - 20.0f / screenHeight; // Adjust y position a bit down from the top

drawText("PAUSE", normX, normY);

}

// Main drawing function

void draw()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

if (gameState == MENU) drawMenu();

else if (gameState == GAME)

{

char scoreDisplay[50];

sprintf(scoreDisplay, "Player 1: %d Player 2: %d", score1, score2);

drawText(scoreDisplay, -0.1f, 0.9f);

drawBall();

drawPaddles();

}

else if (gameState == PAUSE)

{

drawPauseLogo(); // Draw pause logo when the game is paused

drawBall();

drawPaddles();

}

else if (gameState == END)

{

char winText[30];

sprintf(winText, "Player %d wins! :-p", winner);

drawText(winText, -0.1f, 0.0f);

drawText("Press R to Restart", -0.15f, -0.1f);

drawText("Press E to Exit", -0.12f, -0.2f);

}

glutSwapBuffers();

}

// Main function

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(screenWidth, screenHeight);

glutCreateWindow("Pic-Poc :)");

glutDisplayFunc(draw);

glutKeyboardFunc(handleKeypress);

glutTimerFunc(10, update, 0);

glutMainLoop();

return 0;

}

# Capturi de ecran la rularea programului

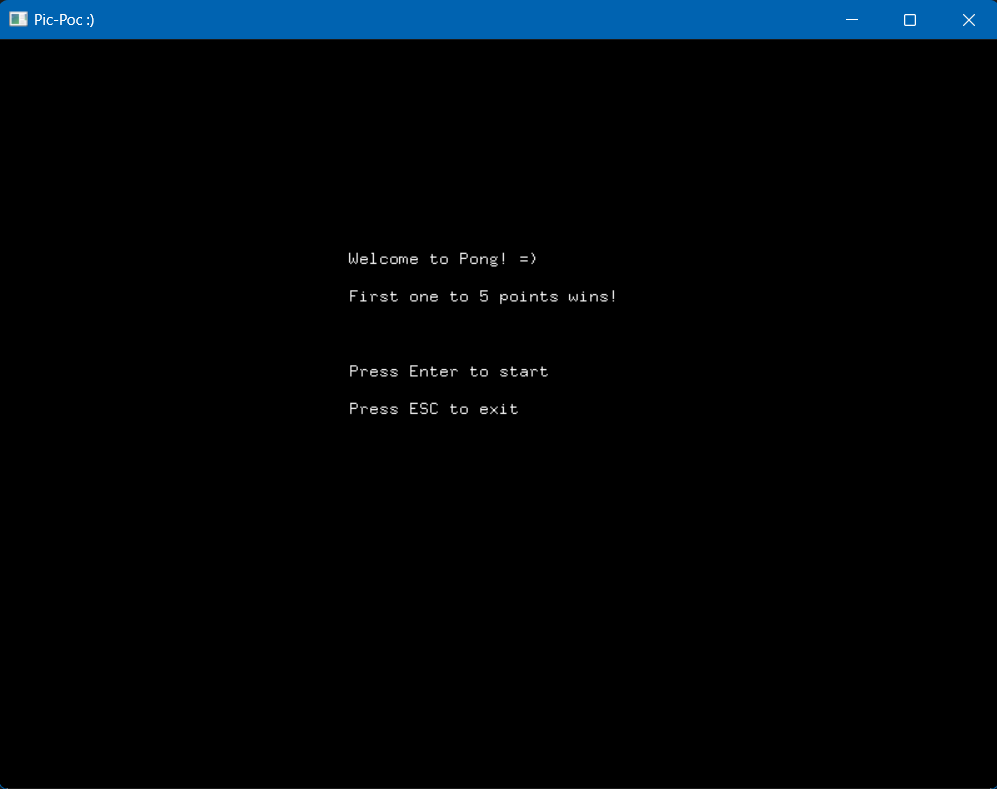


Fig. 6 Meniul jocului

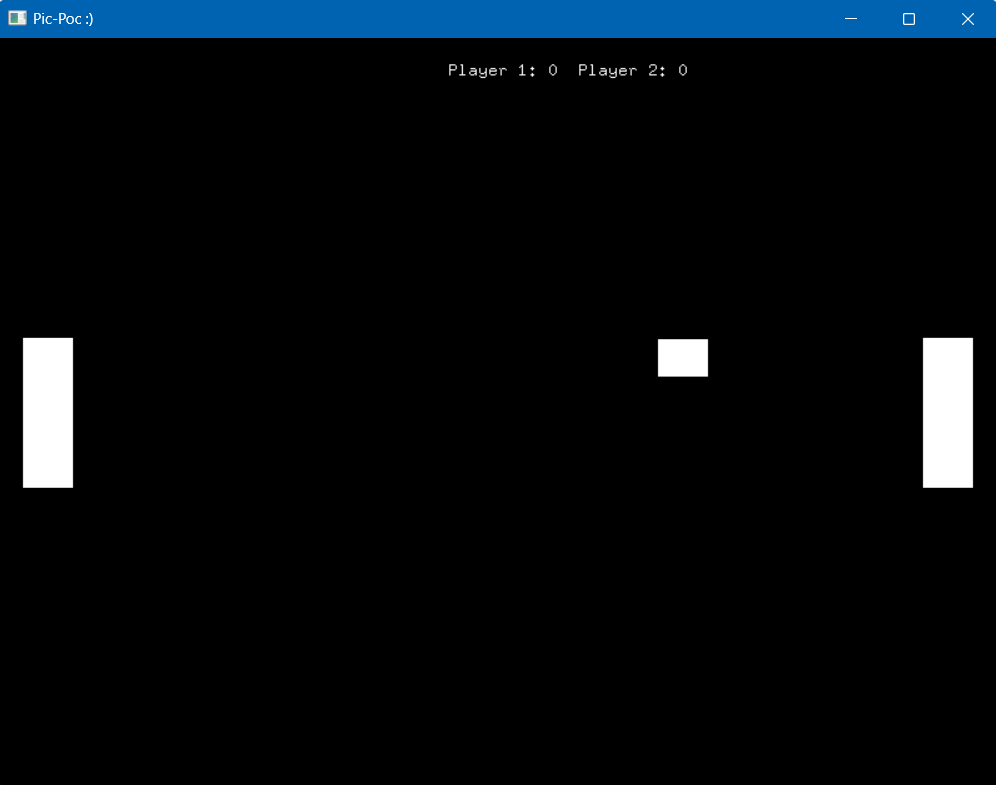


Fig. 7 Fereastra stării GAME a jocului

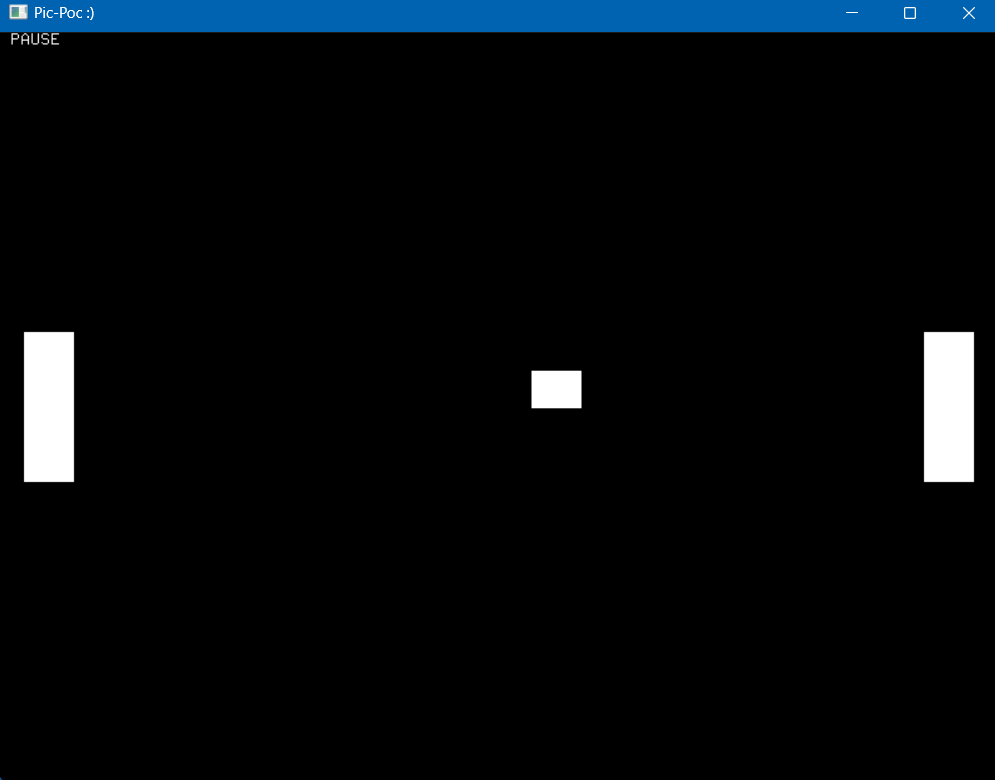


Fig. 8 Starea de pauză a jocului (se poate observa textul „PAUSE” în colțul din stânga sus)

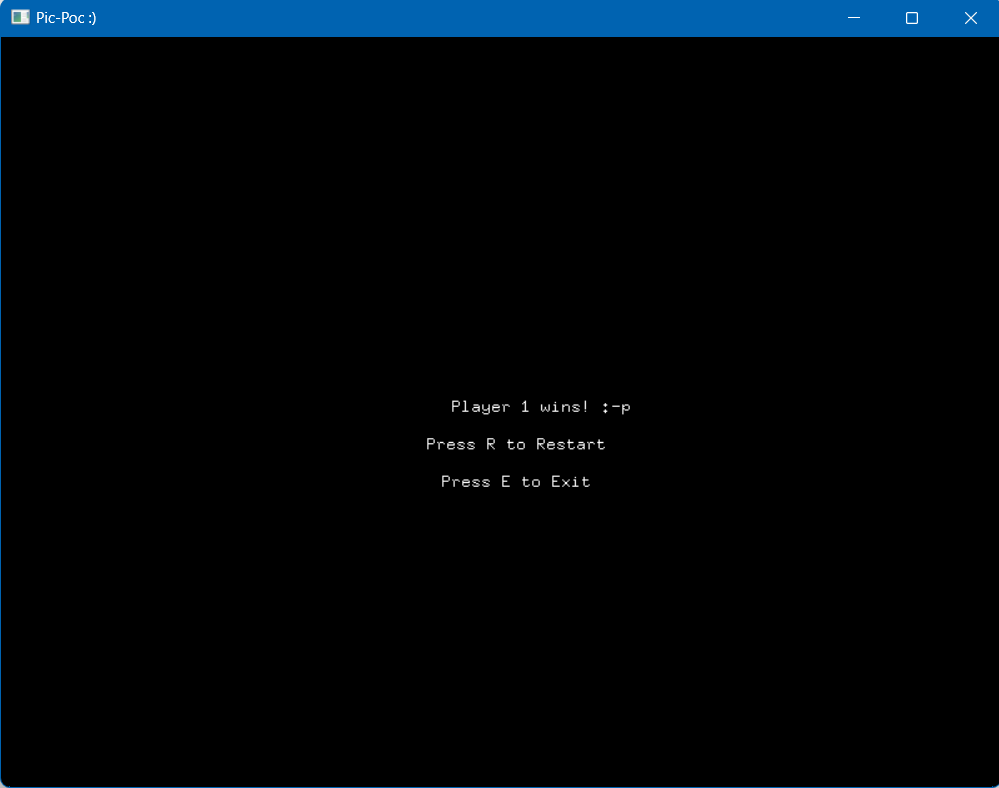


Fig. 9 Starea END, cu cele trei texte care ilustrează câștigătorul și cele două opțiuni de ieșire și de restart.

# Bibliografie

* Luminița Popa, Curs de Programare Grafică
* Luminița Popa, Laborator de Programare Grafică
* Jorje Garcia, Dan Popa, „Manual introductiv de OpenGL”, editura Matrix Rom București 2023
* ChatGPT
* <https://videogamehistory.fandom.com/wiki/Pong>
* <https://ro.wikipedia.org/wiki/OpenGL>
* <https://dcweb.ro/limbajul-de-programare-c>