**Universitatea Tehnică de Construcții din București**

**Facultatea de Hidrotehnică**

**Specializarea: Automatică și Informatică Aplicată**

PROGRAMAREA CALCULATOARELOR SI LIMBAJE DE PROGRAMARE – C++

DOCUMENTATIE TEHNICA DETALIATA

JOC STIL PACMAN

**Profesori coordonatori: Student:**

Neculoiu Giorgian Coles Sasha Mihai

Olteanu Gabriela

**1. Introducere**

Jocul Pac-Man, dezvoltat inițial în 1980 de Namco, este unul dintre cele mai iconice jocuri video din istorie. Acest proiect își propune să recreeze experiența clasică a jocului utilizând limbajul de programare C++ și să ofere o înțelegere profundă a mecanismelor jocului, a arhitecturii software și a implementării tehnice.

**2. Obiectivele Proiectului**

Scopul principal al acestui proiect este de a implementa un joc Pac-Man funcțional în terminal, utilizând C++ standard, fără a depinde de biblioteci externe. Obiectivele specifice includ:

* Crearea unei hărți de joc reprezentate printr-o matrice bidimensională.
* Implementarea mișcării personajului principal (Pac-Man) și a inamicilor (fantomele).
* Gestionarea interacțiunilor dintre Pac-Man și fantome.
* Implementarea unui sistem de scor și a unui meniu de joc.
* Asigurarea portabilității aplicației pe platforme Windows și Unix-like.

**3. Descrierea Jocului**

Jocul se desfășoară pe o hartă reprezentată printr-o matrice de caractere, unde:

* # reprezintă un perete.
* . reprezintă un punct de colectat.
* C reprezintă Pac-Man.
* G reprezintă o fantomă.

Obiectivul jucătorului este de a colecta toate punctele de pe hartă fără a fi prins de fantome. Jocul se termină atunci când toate punctele sunt colectate sau când Pac-Man este prins de o fantomă.

**4. Arhitectura Software**

**4.1 Structuri de Date**

* **Matricea hărții**: Reprezintă structura statică a labirintului.
* **Structura Ghost**: Reprezintă fiecare fantomă, incluzând poziția sa pe hartă.
* **Variabile globale**:
  + pacX, pacY: Poziția lui Pac-Man.
  + score: Scorul curent al jucătorului.
  + gameOver: Starea jocului (în desfășurare sau terminat).

**4.2 Funcții Cheie**

* **drawMap()**: Afișează harta curentă a jocului, inclusiv pozițiile lui Pac-Man și ale fantomelor.
* **isGameOver()**: Verifică dacă jocul s-a încheiat, fie prin capturarea lui Pac-Man de către o fantomă, fie prin colectarea tuturor punctelor.
* **move(char direction)**: Permite lui Pac-Man să se miște în direcția specificată, actualizând scorul și harta.
* **moveGhosts()**: Mișcă fantomele aleator pe hartă, evitând coliziunile cu pereții și cu Pac-Man.
* **menu()**: Afișează meniul principal al jocului și gestionează selecția utilizatorului.

**5. Detalii Tehnice**

**5.1 Afișarea în Terminal**

Pentru a simula o interfață grafică în terminal, se utilizează comenzi specifice sistemului de operare:

* **Windows**: system("cls") pentru a curăța ecranul.
* **Unix-like**: system("clear") pentru a curăța ecranul.

Aceste comenzi sunt incluse în funcția drawMap() pentru a actualiza vizualizarea jocului la fiecare pas.

**5.2 Detectarea Tastelor**

Pentru a permite interacțiunea în timp real cu utilizatorul, se implementează funcții pentru detectarea tastelor apăsate:

* **Windows**: \_kbhit() și \_getch() din biblioteca <conio.h>.
* **Unix-like**: Funcții personalizate kbhit() și getch() utilizând bibliotecile <termios.h>, <unistd.h> și <fcntl.h>.

Aceste funcții permit citirea tastelor fără a bloca execuția programului, esențială pentru un joc în timp real.

**5.3 Mișcarea Fantomelor**

Fantomele se mișcă aleator pe hartă, evitând coliziunile cu pereții și cu Pac-Man. Algoritmul de mișcare este simplu:

1. Se generează un număr aleator între 0 și 3, corespunzând celor patru direcții posibile (sus, jos, stânga, dreapta).
2. Se verifică dacă mișcarea este validă (nu duce într-un perete sau în poziția lui Pac-Man).
3. Dacă mișcarea este validă, se actualizează poziția fantomei.

Această abordare asigură un comportament imprevizibil al fantomelor, crescând dificultatea jocului.

**5.4 Sistemul de Scor**

Scorul este gestionat printr-o variabilă globală score, care este incrementată de fiecare dată când Pac-Man colectează un punct. La finalul jocului, scorul este afișat utilizatorului, oferind un feedback asupra performanței.

**5.5 Meniu și Interacțiune cu Utilizatorul**

La începutul jocului, utilizatorului i se prezintă un meniu simplu:

markdown

CopiazăEditează

======================

PAC-MAN C++

======================

1. Start joc

2. Iesire

Utilizatorul poate selecta opțiunea dorită prin apăsarea tastei corespunzătoare. Dacă se alege să se joace, jocul începe; altfel, aplicația se închide.

**6. Considerații privind Performanța și Limitările**

**6.1 Performanță**

Având în vedere că jocul rulează în terminal și utilizează funcții de sistem pentru a curăța ecranul, performanța poate fi afectată pe sisteme mai lente sau cu resurse limitate. De asemenea, utilizarea funcțiilor system("clear") sau system("cls") poate introduce întârzieri vizibile între cadre.

**6.2 Limitări**

* **Interfață grafică**: Lipsa unei interfețe grafice

**6. Considerații privind Performanța și Limitările (continuare)**

**6.3 Limitări ale AI-ului fantomelor**

Algoritmul folosit pentru mișcarea fantomelor este unul complet aleator. Deși adaugă un anumit nivel de imprevizibilitate, lipsește orice fel de inteligență strategică. În jocul original Pac-Man, fiecare fantomă are un comportament specific (de exemplu, urmărire directă, ambuscadă, rătăcire etc.). Această implementare nu oferă acest nivel de sofisticare, dar oferă o bază ușor de extins.

**6.4 Dimensiunea fixă a hărții**

Harta este definită static într-un tablou de caractere. Acest lucru limitează posibilitățile de a crea nivele dinamice sau multiple nivele. Pentru a permite o extensibilitate mai mare, ar fi recomandabilă utilizarea unui fișier extern din care să fie citită harta sau crearea unei clase dedicate care să gestioneze labirinturile.

**7. Posibile Îmbunătățiri și Extensii**

**7.1 Adăugarea mai multor nivele**

Pentru a îmbunătăți rejucabilitatea jocului, se pot crea mai multe hărți ce pot fi încărcate secvențial. Acestea pot fi salvate într-un fișier .txt, iar programul poate să le citească dinamic folosind operații de fișiere (ifstream, getline etc.).

**7.2 Implementarea unei clase Game**

Actualmente, jocul este scris procedural, ceea ce limitează scalabilitatea și lizibilitatea. Refactorizarea codului într-un stil orientat pe obiecte ar îmbunătăți semnificativ structura aplicației. O clasă Game ar putea conține metode precum update(), render(), processInput(), run() etc.

**7.3 Introducerea unui sistem de vieți**

Pac-Man ar putea începe cu un număr de vieți (ex: 3). De fiecare dată când este prins de o fantomă, pierde o viață. Jocul se termină doar când toate viețile sunt epuizate. Acest sistem ar putea fi reprezentat de o variabilă globală lives.

**7.4 Salvarea și încărcarea scorurilor**

Un sistem simplu de înregistrare a scorurilor maxime ar putea fi implementat utilizând fișiere text. La sfârșitul jocului, scorul este comparat cu cel salvat anterior și, dacă este mai mare, înlocuiește valoarea anterioară. Acest lucru creează o motivație suplimentară pentru jucători.

**7.5 Grafică avansată cu librării externe**

Folosirea librăriilor precum SFML, SDL2 sau chiar ncurses pentru terminal ar aduce o îmbunătățire semnificativă a interfeței utilizatorului. Prin SFML, s-ar putea crea un joc cu animații, sunete și interacțiuni fluide.

**8. Exemple de Cod Detaliate**

**8.1 Cod pentru Mișcarea Pac-Man**

cpp

CopyEdit

void move(char direction) {

int newX = pacX;

int newY = pacY;

if (direction == 'w') newY--;

else if (direction == 's') newY++;

else if (direction == 'a') newX--;

else if (direction == 'd') newX++;

if (map[newY][newX] != '#') {

pacX = newX;

pacY = newY;

if (map[pacY][pacX] == '.') {

score++;

map[pacY][pacX] = ' ';

}

}

}

Această funcție interpretează inputul de la tastatură, actualizează poziția lui Pac-Man dacă mișcarea este validă și modifică scorul dacă a fost colectat un punct.

**8.2 Cod pentru Mișcarea Fantomelor**

cpp

CopyEdit

void moveGhosts() {

for (auto& g : ghosts) {

int dir = rand() % 4;

int newX = g.x;

int newY = g.y;

if (dir == 0) newY--;

else if (dir == 1) newY++;

else if (dir == 2) newX--;

else if (dir == 3) newX++;

if (map[newY][newX] != '#' && !(newX == pacX && newY == pacY)) {

g.x = newX;

g.y = newY;

}

}

}

Acest cod asigură deplasarea fantomelor într-un mod aleator, cu verificări simple pentru a preveni mișcările invalide.

**9. Compararea cu Jocul Original Pac-Man**

| **Caracteristică** | **Jocul original (1980)** | **Versiunea C++ Terminal** |
| --- | --- | --- |
| Grafică | 2D, color | ASCII, monocrom |
| IA fantome | Comportament definit | Mișcare aleatorie |
| Nivele multiple | Da | Nu (doar o hartă) |
| Sunet | Da | Nu |
| Interfață utilizator | Complexă | Minimală în terminal |
| Dificultate progresivă | Da | Nu |

Deși versiunea realizată în C++ este mult mai simplificată, aceasta reușește să păstreze esența jocului original: colectarea punctelor și evitarea inamicilor. Este o bază bună pentru extindere și învățare.

**10. Testare și Debugging**

Testarea jocului se poate face manual, verificând următoarele scenarii:

* Pac-Man se mișcă corect în toate direcțiile.
* Nu se poate trece prin pereți.
* Punctele sunt colectate corect.
* Scorul este afișat corect.
* Fantomele nu apar în pereți.
* Jocul se termină la capturarea lui Pac-Man sau colectarea tuturor punctelor.

Pentru debugging, se pot insera mesaje de tip cout în funcții critice pentru a verifica valorile pozițiilor și starea variabilelor globale.

**11. Extinderea codului într-un motor de joc**

Jocul poate fi transformat într-un proiect mai complex, încorporând:

* Sisteme de coliziune avansate.
* O structură MVC (Model-View-Controller).
* Unit testing folosind framework-uri precum Google Test.
* Posibilitatea de multiplayer local (2 jucători).

Acest tip de extensii transformă un joc simplu într-un proiect educațional complet și aplicabil în portofoliu.

**12. Concluzie Finală**

Jocul Pac-Man implementat în C++ pentru terminal este un exercițiu de programare excelent, care combină elemente de programare procedurală, manipulare de matrice, interacțiune cu utilizatorul și compatibilitate multiplatformă. Deși simplificat, acest proiect oferă o bază solidă pentru învățarea fundamentelor dezvoltării de jocuri.

Cu o structură clară și extensibilă, el poate fi îmbunătățit în numeroase moduri, de la interfață grafică până la IA avansată, ceea ce îl face potrivit atât pentru începători, cât și pentru programatori avansați care doresc să exerseze gândirea algoritmică și arhitectura de software.