Hangman

Hangman, cunoscut și sub numele de "Spanzurătoarea", este un joc clasic de cuvinte în care un jucător trebuie să ghicească un cuvânt, literă cu literă, înainte ca un "hangman" (omul atârnat) să fie complet desenat. Jocul este de obicei jucat de două persoane, dar poate fi adaptat și pentru a fi jucat individual sau în grupuri mai mari.

Unul dintre jucători alege un cuvânt secret și îl scrie sub formă de liniuțe, câte una pentru fiecare literă a cuvântului. Celălalt jucător încearcă să ghicească litera cu litera. Dacă ghicește o literă corect, aceasta este dezvăluită în cuvânt. Dacă greșește, o parte a "hangman"-ului este desenată. Obiectivul este ca jucătorul care ghicește să dezvăluie cuvântul complet înainte ca "hangman"-ul să fie desenat complet.

Regulile exacte ale jocului pot varia, iar "hangman"-ul poate fi desenat sub diverse forme, cum ar fi o ghioagă, un om atârnat, o buclă etc. Hangman este un joc popular datorită simplității sale și faptului că necesită atât gândire logică, cât și cunoștințe literare pentru a ghici cuvintele.

Pentru jocul Hangman creat în limbajul C, am integrat cu succes biblioteca suplimentară raylib.h, care ne-a furnizat un set bogat de funcții pentru manipularea graficii și interacțiunea cu utilizatorul. Prin intermediul acestei biblioteci, am putut realiza un joc captivant și interactiv.

Pentru afișarea textului pe ecran, am folosit funcția DrawText, care ne-a permis să prezentăm informațiile necesare într-un mod clar și estetic. Pentru a adăuga elemente grafice, cum ar fi liniile pentru desenarea conturului ghicitorului sau alte elemente vizuale, am apelat la funcția DrawLine.

Pentru a menține codul organizat și eficient, am utilizat funcțiile BeginDraw și EndDraw, care ne-au ajutat să delimităm procesul de desenare a cadrelor jocului și să gestionăm schimbările grafice cu ușurință.

Interactivitatea jocului a fost realizată prin funcția IsKeyPressed, care ne-a permis să detectăm dacă o anumită tastă a fost apăsată de către utilizator. Această funcție a fost esențială pentru a permite jucătorului să introducă litere pentru a ghici cuvântul.

Pentru a asigura o experiență de joc fluentă și plăcută, am implementat funcția IsWindowOpen pentru a verifica dacă fereastra jocului este deschisă și funcția CloseWindow pentru a închide fereastra în mod corespunzător la finalul jocului.

Integrarea bibliotecii raylib.h și utilizarea funcțiilor sale predefinite au facilitat dezvoltarea jocului Hangman în limbajul C, oferind o experiență interactivă și vizual plăcută pentru utilizatori.

Pentru a asigura variabilitatea și relevanța întrebărilor în jocul Hangman, am formulat întrebările noi pornind de la conceptele și informațiile din mediul de învățare furnizat. Aceste întrebări au fost create pentru a se potrivi cuvintelor și noțiunilor cheie din materialele de învățare, astfel încât să fie captivante și educative pentru jucători. După ce am stabilit întrebările, am identificat cuvintele cheie din răspunsurile aferente și le-am utilizat pentru a crea o listă variată de cuvinte potențiale care pot fi ghicite în joc. Am integrat aceste întrebări și cuvinte cheie în documentul Google Docs, unde am păstrat o structură clară și organizată pentru a putea fi accesate și utilizate eficient în dezvoltarea jocului nostru Hangman. Astfel, acest document a servit ca resursă esențială pentru a ne asigura că întrebările și răspunsurile din joc sunt relevante și adaptate la conținutul educațional furnizat, contribuind astfel la îmbogățirea experienței jucătorilor noștri într-un mod interactiv și educativ.

Explicare pe secvențe de cod

```
void DrawHangman(int tries) {
    int startX = 550;
    int startY = 200;
    // Draw gallows in white for contrast
    DrawLine(startX, startY, startX, startY + 150, WHITE); // Vertical line
    DrawLine(startX, startY, startX + 100, startY, WHITE);  // Horizontal line
    DrawLine(startX + 100, startY, startX + 100, startY + 30, WHITE); // Rope

    // Hangman parts also in white
    if (tries < 6) {
        DrawCircle(startX + 100, startY + 40, 10, WHITE); // Head
    }
    if (tries < 5) {
        DrawLine(startX + 100, startY + 50, startX + 100, startY + 90, WHITE); // Body
    }
    if (tries < 4) {
        DrawLine(startX + 100, startY + 60, startX + 120, startY + 70, WHITE); // Right arm
    }
    if (tries < 3) {
        DrawLine(startX + 100, startY + 60, startX + 80, startY + 70, WHITE);  // Left arm
    }
    if (tries < 2) {
        DrawLine(startX + 100, startY + 90, startX + 120, startY + 120, WHITE); // Right leg
    }
    if (tries < 1) {
        DrawLine(startX + 100, startY + 90, startX + 80, startY + 120, WHITE);  // Left leg
    }
}
```

Funcția `DrawHangman(int tries)` reprezintă o parte crucială a logicii jocului Hangman, deoarece este responsabilă pentru afișarea stadiului curent al atârnatului pe baza numărului de încercări rămase. Această funcție contribuie la crearea unei experiențe vizuale interactive și captivante pentru jucători, oferindu-le un indicator vizual al progresului în ghicirea cuvântului.

Pentru a începe, variabilele `startX` și `startY` sunt definite pentru a specifica poziția de start a desenării elementelor. Apoi, o furcă simplă este desenată folosind linii orizontale și verticale, evidențiind contrastul cu figura atârnatului. Această furcă servește ca fundal pentru figură și contribuie la claritatea vizuală a întregului tablou.

Pentru fiecare parte a atârnatului (cap, corp, brațe și picioare), funcția verifică numărul de încercări rămase (`tries`) și desenează doar părțile corespunzătoare care nu au fost încă ghicite. De exemplu, dacă jucătorul are mai multe încercări rămase, doar capul

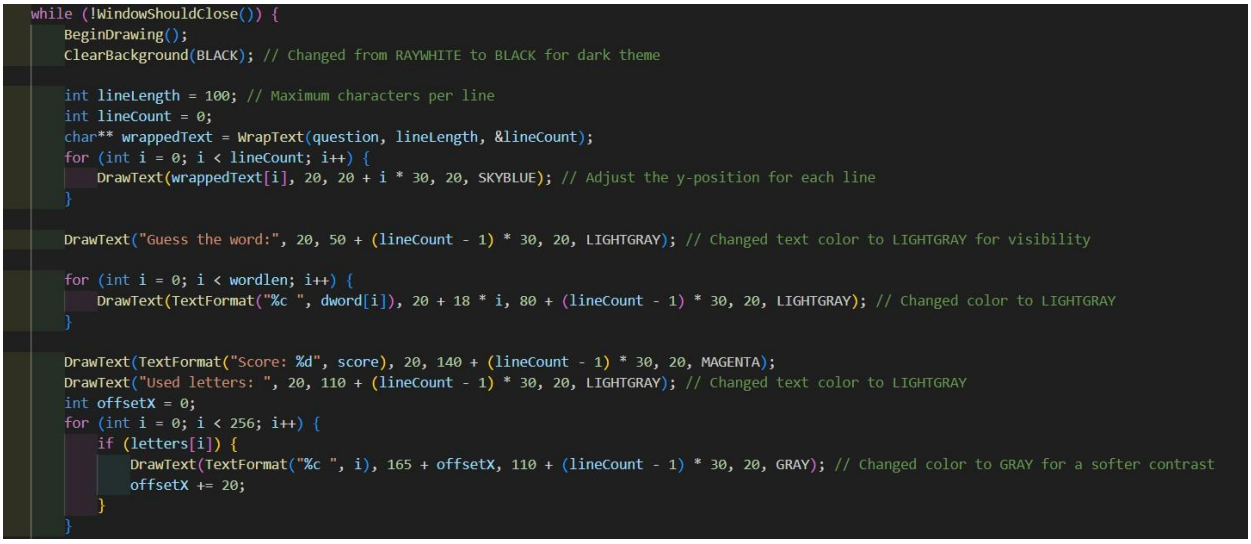
atârnatului va fi desenat. Pe măsură ce numărul de încercări scade, alte părți ale atârnatului sunt adăugate, reflectând progresul jucătorului în joc.

Prin utilizarea culorii albe (`WHITE`) pentru desenarea tuturor elementelor, se asigură un contrast clar cu fundalul jocului, ceea ce permite jucătorului să distingă ușor figura atârnatului. Această funcție contribuie la crearea unei atmosfere interactive și stimulante în cadrul jocului Hangman, oferind jucătorilor o experiență vizuală plăcută și intuitivă în timp ce își testează abilitățile de ghicire a cuvintelor.

```
// Function to wrap text without breaking words
char** WrapText(const char* text, int lineLength, int* lineCount) {
    int textLength = strlen(text);
    int maxLines = textLength / lineLength + 1; // Maximum possible lines
    char** lines = (char**)malloc(maxLines * sizeof(char*));
    *lineCount = 0;

    const char* start = text;
    while (*start) {
        const char* end = start + lineLength;
        if (end >= text + textLength) {
            end = text + textLength;
        } else {
            const char* space = end;
            while (space > start && *space != ' ') space--;
            if (space > start) {
                end = space;
            }
        }

        int actualLineLength = end - start;
        lines[*lineCount] = (char*)malloc((actualLineLength + 1) * sizeof(char));
        strncpy(lines[*lineCount], start, actualLineLength);
        lines[*lineCount][actualLineLength] = '\0';

        (*lineCount)++;
        start = end;
        while (*start == ' ') start++; // Skip spaces
    }

    return lines;
}
```

Funcția `WrapText` reprezintă un instrument esențial în gestionarea și formatarea textului pentru afișarea adecvată în cadrul jocului sau a altor aplicații care necesită prezentarea textului într-un mod structurat și plăcut. Această funcție primește un șir de caractere (`text`), lungimea maximă a unei linii (`lineLength`), și un pointer la o variabilă

pentru a stoca numărul total de linii generate (`lineCount`). În primul rând, se calculează lungimea totală a textului și numărul maxim posibil de linii în funcție de lungimea dorită a fiecărei linii. Apoi, este alocat spațiu de memorie pentru un array bidimensional de caractere (`lines`), care va conține textul împărțit în linii. Parcurgerea textului se face cu ajutorul unui pointer (`start`), care marchează începutul fiecărei linii. În timpul parcurgerii, se găsește sfârșitul fiecărei linii (`end`), astfel încât să nu se rupă cuvintele. Textul este împărțit în linii, iar fiecare linie este stocată în array-ul `lines`. Procesul se repetă până când toate caracterele din text au fost parcurse și împărțite corespunzător. La final, array-ul de linii este returnat pentru a fi utilizat în afișarea textului pe ecran. Această funcție oferă flexibilitate și control asupra modului în care textul este prezentat, permițând dezvoltatorilor să creeze experiențe interactive și estetice pentru utilizatori.



```
while (!WindowShouldClose()) {
    BeginDrawing();
    ClearBackground(BLACK); // Changed from RAYWHITE to BLACK for dark theme

    int lineLength = 100; // Maximum characters per line
    int lineCount = 0;
    char** wrappedText = WrapText(question, lineLength, &lineCount);
    for (int i = 0; i < lineCount; i++) {
        DrawText(wrappedText[i], 20, 20 + i * 30, 20, SKYBLUE); // Adjust the y-position for each line
    }

    DrawText("Guess the word:", 20, 50 + (lineCount - 1) * 30, 20, LIGHTGRAY); // Changed text color to LIGHTGRAY for visibility

    for (int i = 0; i < wordlen; i++) {
        DrawText(TextFormat("%c ", dword[i]), 20 + 18 * i, 80 + (lineCount - 1) * 30, 20, LIGHTGRAY); // Changed color to LIGHTGRAY
    }

    DrawText(TextFormat("Score: %d", score), 20, 140 + (lineCount - 1) * 30, 20, MAGENTA);
    DrawText("Used letters: ", 20, 110 + (lineCount - 1) * 30, 20, LIGHTGRAY); // Changed text color to LIGHTGRAY
    int offsetX = 0;
    for (int i = 0; i < 256; i++) {
        if (letters[i]) {
            DrawText(TextFormat("%c ", i), 165 + offsetX, 110 + (lineCount - 1) * 30, 20, GRAY); // Changed color to GRAY for a softer contrast
            offsetX += 20;
        }
    }
```

Această secvență de cod este responsabilă pentru afișarea elementelor de interfață grafică (UI) în fereastra jocului. În primul rând, se verifică dacă fereastra jocului este încă deschisă. Apoi, începe desenarea cu ajutorul funcției `BeginDrawing()` și se șterge fundalul cu negru folosind `ClearBackground(BLACK)`, schimbând tema de la `RAYWHITE` la `BLACK`.

Pentru a afișa întrebarea asociată cuvântului, textul întrebării este împărțit în linii pentru a se potrivi în fereastra de joc. Aceasta este realizată prin apelul funcției `WrapText()`, care returnează un vector de șiruri de caractere împărțite conform unei lungimi maxime specificate pentru fiecare linie. Apoi, fiecare linie a textului împărțit este desenată pe ecran folosind `DrawText()`, ajustând poziția verticală pentru fiecare linie.

În continuare, este afișat mesajul "Guess the word:", urmat de cuvântul necunoscut, reprezentat de caracterele necunoscute și spații goale. Aceste spații goale sunt înlocuite treptat cu literele corecte ghicite de jucător. Similar, textul "Score:" urmat de scorul actual și "Used letters:" urmat de literele utilizate sunt afișate pentru a oferi jucătorului informații relevante în timpul jocului. Literele utilizate sunt afișate în partea de jos a ferestrei de joc, fiecare literă fiind urmată de un spațiu și afișată în culoarea gri (`GRAY`) pentru a oferi un contrast mai blând față de textul principal.

```
DrawHangman(lives);

int key = GetKeyPressed();
while (key > 0) {
    if ((key >= 65 && key <= 90) || (key >= 97 && key <= 122)) { // A-Z or a-z
        char ch = toupper(key);
        if (!letters[ch]) {
            letters[ch] = 1;
            int found = 0;
            for (int i = 0; i < wordlen; i++) {
                if (word[i] == ch) {
                    dword[i] = ch;
                    found = 1;
                }
            }
            if (!found) lives--;
        }
    }
    key = GetKeyPressed(); // Check next pressed key
}
if (lives <= 0 || strcmp(dword, word) == 0) {
    DrawText((lives <= 0) ? "You lost!" : "You won!", 300, 350, 20, (lives <= 0) ? DARKBLUE : DARKGREEN);
    DrawText(TextFormat("The word was: %s", word), 300, 380, 20, LIGHTGRAY);
    if (strcmp(dword, word) == 0 && scoreAdded == false) {
        score += 10;
        scoreAdded = true;
    }
    gameFinished = true; // Game ends, waiting for reset
    DrawText("Press [BACKSPACE] to exit or [SPACE] to go to the next round", 300, 410, 20, LIME);
    if (IsKeyPressed(KEY_BACKSPACE)) {
        break;
    }
}
```

Această parte a codului este esențială pentru funcționarea jocului. În primul rând, se afișează grafica pentru omul atârnat (Hangman), în funcție de numărul de încercări rămase (`lives`). Apoi, jocul așteaptă introducerea unei taste de către utilizator prin intermediul funcției `GetKeyPressed()`.

În timp ce utilizatorul apasă taste, se verifică dacă aceasta reprezintă o literă din alfabet (`A-Z` sau `a-z`). Dacă este, litera este convertită în majusculă folosind funcția `toupper()` și este verificat dacă a fost deja folosită. Dacă litera nu a fost utilizată anterior,

aceasta este marcată ca folosită și se verifică dacă se potrivește cu vreuna dintre literele din cuvântul de ghicit. Dacă este găsită o potrivire, litera este afișată în locul corespunzător în cuvântul de ghicit (`dword`) și se actualizează starea `found`.

În caz contrar, dacă litera nu se potrivește cu nicio literă din cuvântul de ghicit, numărul de încercări rămase (`lives`) este decrementat. Acest lucru duce la afișarea de elemente corespunzătoare ale omului atârnat în funcție de numărul de încercări rămase.

Jocul continuă până când jucătorul ghicește cuvântul corect sau până când numărul de încercări rămase devine zero. Dacă jucătorul a pierdut, se afișează un mesaj corespunzător, indicând faptul că a pierdut și cuvântul corect. Dacă jucătorul a ghicit corect cuvântul și punctele nu au fost încă adăugate la scor, punctele sunt adăugate la scor și se afișează un mesaj corespunzător, indicând câștigul. Jocul este marcat ca încheiat și jucătorul are opțiunea de a continua la următoarea rundă sau de a ieși din joc, folosind tastele `BACKSPACE` sau `SPACE`.

```
    if (IsKeyPressed(KEY_SPACE)) {
        scoreAdded = false; // Reset the game when SPACE is pressed
        rand_idx = rand() % wc;
        Node *selectedWordNode = word_list;
        for (int i = 0; i < rand_idx; i++) {
            selectedWordNode = selectedWordNode->next;
        }
        word = selectedWordNode->content;
        question = selectedWordNode->linkedNode->content;
        // Move to the next word
        wordlen = strlen(word);
        for (int i = 0; i < wordlen; i++) {
            dword[i] = (word[i] == '-') ? '-' : '_';
        }
        dword[wordlen] = '\0';
        memset(letters, 0, sizeof(letters)); // Reset used letters
        lives = MAX_TRIES;
        gameFinished = false; // Allow new game input
    }

    EndDrawing();
}

CloseWindow();
```

Această secțiune de cod reacționează la apăsarea tastei `SPACE`. Atunci când utilizatorul apasă tasta `SPACE`, jocul este resetat pentru o nouă rundă.

Mai întâi, variabila `scoreAdded` este setată la `false`, pentru a asigura că punctele nu sunt adăugate din nou la scor înainte de a începe o nouă rundă. Apoi, este generat un nou index aleatoriu pentru a selecta un nou cuvânt din lista de cuvinte. Se parcurge lista de cuvinte până la indexul generat, iar apoi se actualizează cuvântul și întrebarea curentă cu noile valori. Lungimea noului cuvânt este calculată și se reinițializează cuvântul afișat (`dword`) cu liniuțe subliniate pentru fiecare literă, exceptând cazul în care litera este un cratimă.

Se resetează, de asemenea, vectorul `letters` care ține evidența literelor folosite și se reinițializează numărul de încercări rămase (`lives`) cu valoarea maximă `MAX_TRIES`. De asemenea, se setează `gameFinished` la `false`, permițând astfel introducerea de noi intrări în joc.

După ce jocul este resetat, se încheie desenarea ferestrei și se așteaptă următoarea interacțiune cu utilizatorul. Dacă utilizatorul închide fereastra, jocul se încheie complet prin apelarea funcției `CloseWindow()`.

De asemenea, o descriere a codului total, nu luat pe secvențe, ar fi:

În codul furnizat pentru jocul Hangman, inițializarea este un element fundamental. Se începe prin definirea și inițializarea ferestrei de joc, stabilind dimensiunile și titlul acesteia. Aceste detalii sunt esențiale pentru a oferi o experiență vizuală plăcută și coerentă jucătorului. De asemenea, variabile importante sunt declarate și inițializate, precum numărul maxim de încercări (MAX_TRIES) și listele de cuvinte și întrebări. Aceste variabile constituie baza logicii și a funcționării jocului.

Odată ce inițializarea a fost completă, urmează crearea și legarea listelor de cuvinte și întrebări. Folosind funcția InitializeList(), se construiește o structură de date pentru a stoca cuvintele și întrebările asociate. Apoi, prin intermediul funcției LinkNodes(), se asigură că fiecare cuvânt este corect asociat cu întrebarea corespunzătoare. Această legătură este esențială pentru a oferi context jucătorului în timp ce își ghicește cuvântul.

Secvența următoare este dedicată desenării elementelor vizuale ale jocului. Funcția DrawHangman() este responsabilă pentru desenarea spânzurătoarei în funcție de numărul de încercări rămase, oferind o reprezentare vizuală a progresului jucătorului și a consecințelor fiecărei ghiciri greșite. În același timp, cuvântul necunoscut este afișat sub spânzurătoare, cu spații pentru literele neatinse. Acest lucru facilitează urmărirea progresului jucătorului și oferă un cadru clar pentru ghicirea cuvântului.

Ultima secțiune se concentrează pe interacțiunea jucătorului cu jocul. Prin intermediul tastaturii, jucătorul poate încerca să ghicească litere din cuvânt. Implementarea acestei funcționalități implică verificarea tastelor apăsate și evaluarea dacă litera introdusă este prezentă în cuvântul necunoscut. În funcție de rezultat, jucătorul poate pierde o viață sau poate avansa în ghicirea cuvântului. Această interacțiune dinamică între jucător și joc adaugă un element de provocare și angajament.

Note către studenții participanți în echipă:

1. Alexandru Gabriel: 10
2. Dumitru Claudia Ștefania: 10
3. Pralea Bogdan-Ștefan: 10
4. Odoroaga Vlad-Ionuț: 10
5. Manole Daniel: 10
6. Dobrescu Diana: 10