Dokumentácia - Hra Tetris

1. Predstavenie Hry

Tetris je klasická puzzle hra implementovaná v jazyku C s využitím knižnice ncurses. Hra ponúka kompletný herný zážitok s textovým grafickým rozhraním, koloračným zobrazením a moderným Tetrominovým systémom. Hráči musia usporaďovať padajúce štvorbloky (tetrominoes) tak, aby vytvorili horizontálne rady a tie následne vymazali. Hra sa s každým vymazaným radom stáva zložitejšou a rýchlejšou.

Hra obsahuje všetkých 7 štandardných tetromín: - **I-tetromína**: Modrá, lineárna (4 bloky v rade) - **O-tetromína**: Žltá, štvorcová (2x2) - **T-tetromína**: Purpurová, T-tvar - **S-tetromína**: Zelená, S-tvar - **Z-tetromína**: Červená, Z-tvar - **J-tetromína**: Modrá, J-tvar - **L-tetromína**: Biela, L-tvar

2. Preloženie a Spustenie Hry

Požiadavky

- Kompilátor GCC
- Knižnica ncurses (libncurses5-dev alebo ekvivalent)
- Unix-like terminálové prostredie (Linux, macOS, WSL, atď.)
- Terminálové okno aspoň 50 znakov šíreho a 24 riadkov vysokého

Spôsob Preloženia

1. **Navigácia do priečinka projektu:**

```
cd final
```

2. **Preloženie hry:**

```
make
```

Príkaz `make` spustí kompiláciu pomocou GCC s nasledovnými parametrami: - `-Wall -Wextra`: Zobrazenie všetkých upozornení - `-std=c99`: Použitie štandardu C99 - `-lncurses`: Linkovanie s knižnicou ncurses

3. **Spustenie hry:**

```
./tetris
```

4. **Čistenie súborov preloženia:**

```
make clean
```

Kompilácia na Sigma Serveri

Pre prácu na universitnom serveri:

```
ssh váš-login@sigma.cs.uchicago.edu
cd final
make
./tetris
```

3. Návod na Hru

Herné Ovládanie

Tlačítko	Akcia
← **Šípka doľava**	Posun časti doľava
→ **Šípka doprava**	Posun časti doprava
↓ **Šípka dole**	Soft drop - urýchlenie pádu
Medzerník	Hard drop - okamžitý pád na dno
Z	Otočenie časti v smere hodinových ručičiek
P	Pauza / Pokračovanie
Q	Ukončenie hry
R (pri Game Over)	Reštart hry

Systém Bodovania

Body sa udeľujú na základe počtu vymazaných radov naraz:

Počet Radov	Základné Body
1 rad	100
2 rady	300
3 rady	500
4 rady (Tetris!)	800

Násobitel podľa úrovne: Všetky body sa násobia aktuálnou hernou úrovňou. Napríklad, vymazanie 4 radov na úrovni 2 = 800 × 2 = 1600 bodov.

Úrovne a Ťažkosť

- Úroveň sa zvýši o 1 po každých 10 vymazaných radoch
- Počiatočná úroveň je 1
- Rýchlosť pádu časti sa s úrovňou zvyšuje
- Maximálna rýchlosť je dosahnutá na úrovni 16

Koniec Hry

Hra končí, keď sa nová tetromína nemôže spustiť na vrchu hernej dosky. To zvyčajne nastane vtedy, keď hráč nedokáže časti vymazávať dostatočne rýchlo a hracia plocha je zaplnená až po vrch.

Vizuálne Prvky

- **Náhľad ďalšej časti**: Vpravo sa zobrazuje nasledujúca tetromína
- **Fantómová čiastka**: Zelené bodky ukazujú, kde sa čiastka ocitne pri hard dropu
- **Farebné čiastky**: Každá tetromína má jedinečnú farbu
- **Herné Štatistiky**: Zobrazenie skóre, úrovne a počtu vymazaných radov

4. Programová Implementácia

Najdôležitejšie Funkcie

4.1 Inicializácia a Spúšťanie `void* init_game()` - Inicializuje nový herný stav - Prideľuje pamäť pre hernú štruktúru - Inicializuje hernú dosku na prázdny stav (všetky bunky nastavené na 0) - Nastavuje počiatočnú úroveň na 1 - Generuje prvú nasledujúcu čiastku

`int world_event(struct event* event, void* game)` - Hlavný event handler hry - Spracúva všetky vstupné udalosti (klávesy, timeout, zmena veľkosti okna) - Volá príslušné herné funkcie podľa typu udalosti - Vracia 1 ak sa má hra ukončiť, 0 ak sa hra pokračuje

`void destroy_game(void* game)` - Uvoľňuje pamäť hernému stavu - Volá sa pri ukončení hry

4.2 Mechanika Pohybu `int can_place_piece(GameState* state, int type, int x, int y, int rotation)` - Kontroluje, či je možné umiestniť čiastku na danú pozíciu s danou rotáciou - Overuje hranice hernej dosky - Overuje zrážky so zafixovanými čiastkami - Vracia 1 ak je umiestnenie možné, 0 ak nie

`void spawn_piece(GameState* state)` - Spúšťa novú tetromínu v strede vrchu dosky - Nastavuje rotáciu na 0 (pôvodná pozícia) - Kontroluje, či je hra skončená (ak sa čiastka nemôže spustiť)

`void move_piece_left(GameState* state)` - Posúva aktuálnu čiastku o jedno políčko doľava - Kontroluje, či je pohyb povolený

`void move_piece_right(GameState* state)` - Posúva aktuálnu čiastku o jedno políčko doprava - Kontroluje, či je pohyb povolený

`int move_piece_down(GameState* state)` - Posúva aktuálnu čiastku o jedno políčko nadol - Vracia 1 ak je pohyb povolený, 0 ak nie

`void hard_drop_piece(GameState* state)` - Spúšťa čiastku až na dno okamžite - Viackrát volá `move_piece_down()` až do chvíle, keď sa čiastka nemôže pohybovať

`void rotate_piece(GameState* state)` - Otáča aktuálnu čiastku o 90° v smere hodinových ručičiek - Overuje, či je otočenie povolené (bez zrážok)

4.3 Fixovanie a Interakcia `void lock_piece(GameState* state)` - Fixuje aktuálnu čiastku na hernú dosku (stáva sa jej trvalou súčasťou) - Spúšťa novú čiastku - Volá sa automaticky keď čiastka nemôže ďalej padať

`void check_and_clear_lines(GameState* state)` - Kontroluje, ktoré rady sú úplne zaplnené - Vymazáva úplné rady - Posúva všetky rady nad vymazaným radom nadol - Aktualizuje skóre na základe počtu vymazaných radov - Zvyšuje úroveň hry podľa počtu vymazaných radov

4.4 Fyzika a Časovanie `void apply_gravity(GameState* state)` - Aplikuje gravitáciu na čiastku (automatický pád) - Využíva čítač `frames_since_last_gravity` a porovnáva ho s `gravity_speed` - Keď čiastka nemôže padať ďalej, fixuje ju a vytvorí novú

`void render_game(GameState* state)` - Vykresľuje celú hernú scénu na obrazovku - Vykresľuje hernú dosku s rámčekom - Vykresľuje všetky fixované čiastky - Vykresľuje aktuálnu padajúcu čiastku - Vykresľuje fantómovú čiastku (náhľad pristátia) - Vykresľuje štatistický panel vpravo - Vykresľuje stavový riadok na spodku

Dôležité Štruktúry

4.5 Štruktúra GameState

```
typedef struct {
    int board[BOARD_HEIGHT][BOARD_WIDTH];  /* Hernú dosku - 1=obsadené, 0=prázdne */
    int piece_type;                         /* Typ aktuálnej čiastky (0-6) */
    int piece_x, piece_y;                   /* Pozícia čiastky na doske */
    int piece_rotation;                     /* Rotácia čiastky (0-3) */
    int next_piece_type;                    /* Typ ďalšej čiastky */
    int score;                              /* Aktuálne skóre */
    int level;                              /* Aktuálna úroveň */
    int lines_cleared;                      /* Počet radov vymazaných naraz */
    int total_lines;                        /* Celkový počet vymazaných radov */
    int game_over;                          /* Príznak konca hry */
    int paused;                             /* Príznak pauzy */
    int gravity_counter;                    /* Čítač gravitácie */
    int gravity_speed;                      /* Rýchlosť pádu čiastky */
    int frames_since_last_gravity;          /* Počet snímkov od posledného pádu */
    int screen_width;                       /* Šírka terminálu */
    int screen_height;                      /* Výška terminálu */
} GameState;
```

4.6 Tetromínové Tvary

```
static const int PIECE_SHAPES[NUM_PIECES][4][4][2] = { ... }
```

Táto trojrozmernná matica definuje 7 tetromín s 4 rotačnými stavmi každý: - **Rozmer 1** (0-6): Typ tetromíny (I, O, T, S, Z, J, L) - **Rozmer 2** (0-3): Rotačný stav (0°, 90°, 180°, 270°) - **Rozmer 3** (0-3): Štyri bloky tetromíny - **Rozmer 4** (x, y): Relatívne súradnice bloku od stredu

Algoritmické Princípy

4.7 Detekcia Zrážok Hra používa jednoduchý a efektívny systém detekcie zrážok: 1. Pre každý blok čiastky vypočíta pozíciu `(x + offset_x, y + offset_y)` 2. Kontroluje, či pozícia leží v hraniciach dosky (0 až BOARD_WIDTH-1, 0 až BOARD_HEIGHT-1) 3. Kontroluje, či je bunka na doske prázdna `board[y][x] == 0` 4. Ak všetky bloky prejdú overením, čiastka sa môže umiestniť

4.8 Vymazávanie Radov Algoritmus vymazávania: 1. Iteruje dosku od spodku k vrchu 2. Pre každý riad kontroluje, či sú všetky bunky obsadené 3. Neúplné rady sa kopírujú do novej dosky (shift dole) 4. Úplné rady sa vynechajú (vymazánia sa) 5. Horná časť dosky sa vyplní nulami

4.9 Systém Bodovania Bodovanie sa vypočítava ako:

```
skóre += body_za_rady * aktuálna_úroveň
```

Kde `body_za_rady` sú: - 1 riad: 100 bodov - 2 rady: 300 bodov - 3 rady: 500 bodov - 4 rady: 800 bodov

5. Modifikácie Knižnice World

Knižnica World bola pôvodne určená ako všeobecný framework na vytváranie interaktívnych aplikácií s textovým grafickým rozhraním. Pre Tetris bola použitá bez výrazných modifikácií, ale boli vyvinuté nasledovné rozšírenia a optimalizácie:

5.1 Prispôsobenie Farieb

Knižnica World bola rozšírená o podporu všetkých 8 systémových farieb (čierna, červená, zelená, žltá, modrá, purpurová, azúrová, biela). Hra používa tieto farby na odlíšenie jednotlivých tetromín.

5.2 Unicode Znaky pre Grafiku

Implementácia umožnila používanie Unicode znakov na lepšie grafické znázornenie: - `0x250C`, `0x2510`, `0x2514`, `0x2518`: Rohy rámčekov - `0x2500`, `0x2502`: Vodorovné a zvislé čiary - Vlastné znaky: `#`, `O`, `T`, `S`, `Z`, `J`, `L` pre tetromíny

5.3 Optimalizácia Výkonu

- **Event loop**: Implementácia efektívneho event loop-u s timeout mechanizmom

- **Inkrementálne vykresľovanie**: Hra vykresľuje len časti, ktoré sa zmenili
- **Gravitačný systém**: Časovací mechanizmus pre aplikovanie gravitácie

5.4 Interakcia s Terminálom

- Podpora šípok a špeciálnych kláves cez ncurses
- Detekcia veľkosti terminálového okna
- Bezproblémová práca s rôznymi typmi terminálov

6. Zdroje a Referencie

Dokumentácia a Knižnice

1. **ncurses dokumentácia** - https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/
2. **GCC kompilátor** - https://gcc.gnu.org/onlinedocs/
3. **C99 štandard** - https://en.wikipedia.org/wiki/C99
4. **Tetris pravidlá a mechanika** - https://tetris.wiki/
5. **Unicode Box Drawing Characters** - https://en.wikipedia.org/wiki/Box-drawing_character

Inšpirácia a Referencie

- Originálny Tetris od Alexeja Pajitnova (1984)
- Štandardný Tetris Guidelines - https://tetris.wiki/Tetris_Guideline

7. Použitie Umelej Inteligencie

7.1 Druh Umelej Inteligencie

Pri vytváraní tejto dokumentácie a v procese vývoja boli využité generatívne AI modely (konkrétne **Claude Haiku 4.5** a podobné jazykové modely).

7.2 Spôsob Použitia

1. **Dokumentácia**: Generácia textov v slovenčine, opisov funkcií a štruktúr
2. **Kódovanie**: Asistencia pri navrhovaní algoritámov a optimalizácií
3. **Testovanie**: Návrhy na testovacích prípadov a možné edge cases
4. **Vysvetľovanie**: Analýza a vysvetlenie existujúceho kódu

7.3 Kontextové Informácie

- AI bol používaný ako pomôcka pre zrýchlenie vývoja
- Všetok kód bol manuálne skontrolovaný a overený
- AI pomáhal pri brainstormingu a optimalizácii
- Dokumentácia bola vytvorená s podporou AI pre konzistentnosť a úplnosť
