Brick game: tetris. Документация

SIONAPAE

05.10.24

1 Описание проекта

Реализация игры «Тетрис» на языке программирования C с использованием структурного подхода.

2 Требования

Для запуска программы требуются:

- Компилятор GCC или любой другой поддерживающий C++
- GNU Make для сборки программы
- библиотека check.h, ncurses.h

3 Как собрать и запустить игру

1. Выполните команду ‘make all‘ для сборки программы.
2. Запустите игру командой make run.
    - all
    - install — компиляция исходного файла tetris
    - run — запуск игры
    - uninstall — удаление tetris
    - clean — очистка файлов сборки
    - dvi — документация проекта
    - dist — архивирование
    - test — запуск тестов
    - gcov_report — формирование html-отчета с информацией о покрытии функций тестами

4 Управление

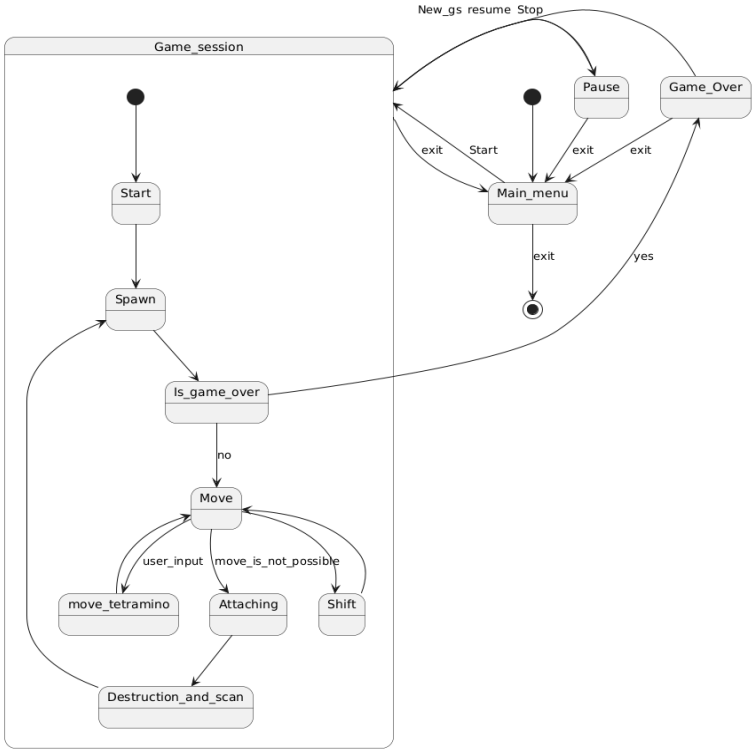
Игрок использует клавиши на клавиатуре, имитирующие физические кнопки на реальной консоли, для управления падающими фигурками:

- Клавиша D — Стрелка влево — перемещение фигуры влево.
- Клавиша A — Стрелка вправо — перемещение фигуры вправо.
- Клавиша F — Стрелка вниз — падение фигуры.
- Клавиша S — ускорение падения фигуры.
- Клавиша W — Стрелка вверх — поворот фигуры.

- Клавиша P/SPACE — пауза.
- Клавиша N/ENTER — старт новой игры или продолжение.
- Клавиша q/ESCAPE — завершение текущей игры, выход из игры.

5 Архитектура программы

Конечный автомат для конкретной реализации игры Тетрис.



Программа построена вокруг конечного автомата (Finite State Machine, FSM), который управляет логикой игры. Функция `main()` настраивает окружение, запускает игровой цикл через `game_loop()` и обрабатывает завершение игры.

Весь код организован следующим образом:

5.1 Функция `main()`: Точка входа

Эта функция инициализирует окружение игры, запускает основной игровой цикл и завершает работу игры при выходе.

- **Инициализация генератора случайных чисел:**

```
srand(time(0));
get_random();
```

  Здесь инициализируется генератор случайных чисел для генерации случайных фигур Тетромино.

- **Инициализация игровых объектов:**

```
Game_Objects_t* params = get_instanse();
*params = init_empty_game_objects();
```

  Объект `params` содержит информацию о текущем состоянии игры: текущее состояние, действия пользователя, игровые объекты и т.д.

- **Инициализация ncurses и запуск цикла игры:**

```
#ifndef debug_bro
init_bro_ncurses(&params->views);
game_loop();
terminate_ncurses_bro(&params->views);
#else
game_loop();
#endif
```

  В зависимости от режима компиляции, игра либо использует библиотеку ncurses для работы с консольным интерфейсом, либо запускается в режиме отладки без ncurses.

5.2 Функция `game_loop()`: Основной игровой цикл

Цикл `game_loop()` выполняется до тех пор, пока игра активна. Он обрабатывает все основные состояния игры и пользовательские действия.

- **Инициализация состояния игры:**

```
State prev = START;
```

  Переменная `prev` хранит предыдущее состояние игры для возврата из паузы или других состояний.

- **Главный цикл игры:**

```
while (params->game_is_running) {
    draw_static(params);
    main_game_fsm(params);
    game_session(params, &prev);
}
```

  Основной цикл игры продолжает выполняться до тех пор, пока `game_is_running == true`. Внутри цикла обрабатываются состояния игры и действия пользователя.

5.3 Функция `game_session()`: Управление игровой сессией

Функция `game_session()` управляет непосредственно игровым процессом (например, движением Тетромино, отсчетом времени и обработкой паузы).

- **Инициализация игровой сессии:**

```
if (params->state == START) {
    session_is_running = true;
    params->state = *prev;
}
```

  Если игра начинает с состояния `START`, то восстанавливается предыдущее состояние игры, и начинается новая игровая сессия.

- **Цикл игровой сессии:**

```
while (params->state != PAUSE && params->state != GAME_OVER && params->state != MAIN_MENU) {
    fsm_game_session(params);
    key = getch();
    userInput(getSignal(key), session_is_running);
    countTime(params);
}
```

Цикл продолжается до тех пор, пока игра не завершена (GAME_OVER) или не поставлена на паузу. Внутри цикла обрабатываются пользовательские действия (движение, пауза, завершение игры) и обновляется состояние игрового мира.

- **Запись рекордов:**

```
if (params->gameInfo.score > params->gameInfo.high_score)
    write_high_score(params->gameInfo.score);
```

Если игрок набрал новый рекордный счет, он записывается.

6 Используемые технологии и библиотеки

- **Компилятор и версия:** Рекомендуемая версия — GCC 9.3
- **Внешние библиотеки:**
  - `ncurses.h` — библиотека для работы с консольным интерфейсом, которая предоставляет функции для работы с текстовым терминалом, обработки ввода и вывода.