dokumentacja projektu

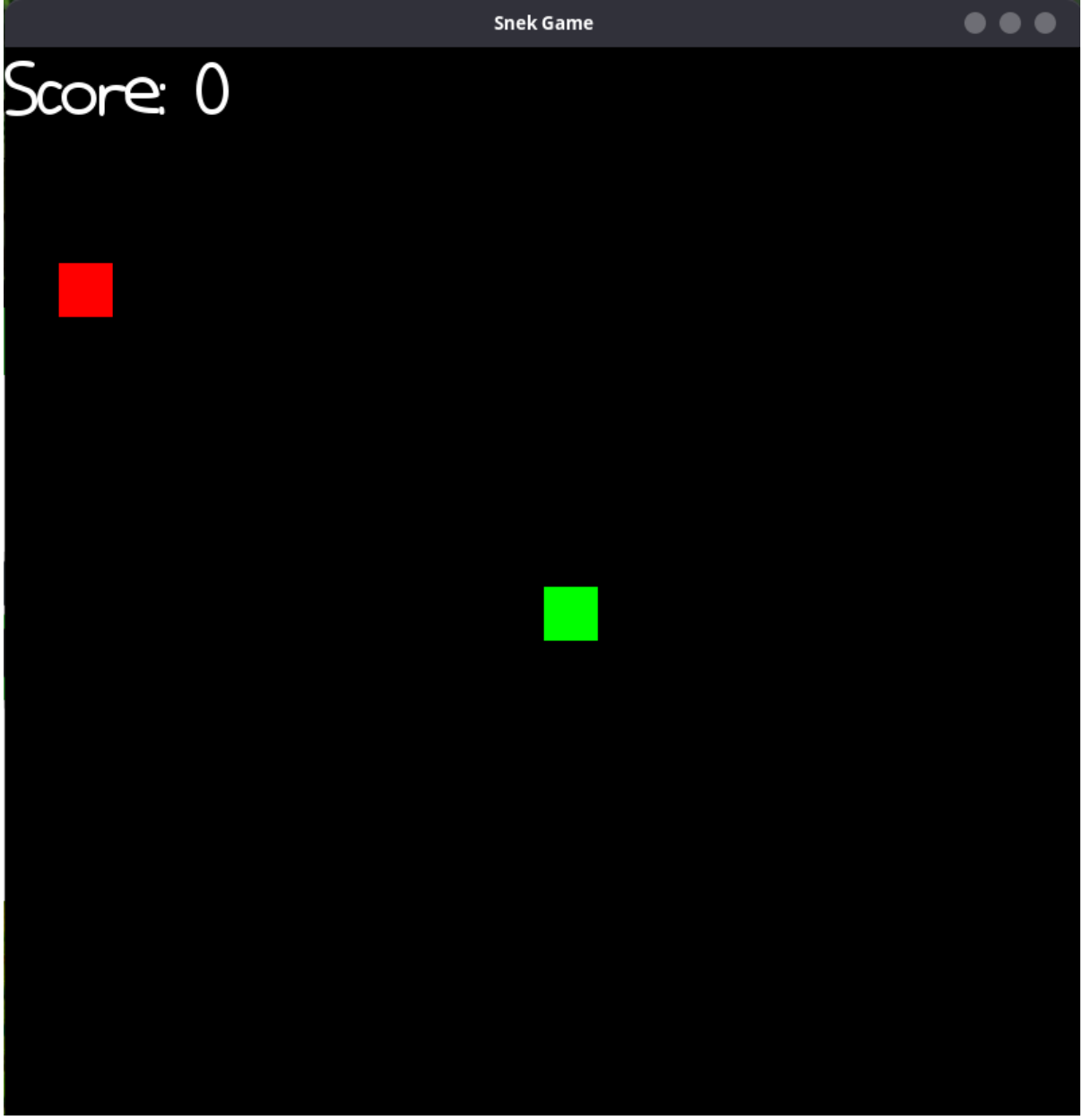
opis gry

Prosta gra w węża, zadaniem gracza jest za pomocą strzałek poruszać wężem nie udeżyć w samego siebie ani w ściany i zdobyć jedzenie dla węża. Wąż jest tym dłuższy im więcej jedzenia zje i wychodowanie jak największego węża jest celem gry. Gra kończy się gry gracz udezy wężem w ściane lub doprowadzi do ikolizji węża z samym sobą.

instrukcja

Po uruchomieniu gry gracza przywita ekran powitalny:

Aby rozpocząc gre należy wcisnąc Enter. Następnie gracz zobaczy ekran z zielonym i czerwonym kwadratem



zielony kwadrat jest wężem czerwone kwadraty to jedzenie dla węża. Wąż ruszy się dopiero po wciśnięciu któregoś z klawiszy strzałek i będzie się poruszał nie przerwanie w tym kierunku. Gracz ma za zadanie tak pokierowac głowe węża żeby trafiła w czerwony kwadracik jedzenia, przy jednoczesnym unikaniu krawędzi okna oraz udeżenia węża w samego siebie. Po zjedzeniu jedzonka snake urośnie o 1 segment a licznik punktów wzrośnie

po udeżeniu w krawędź okna bądź w samego siebie gracz zobaczy ekran

aby wyjśc z gry nalezy kliknąć x w prawym górnym rogu okna lub za pomocą skrótu klawiszowego alt+f4

dokumentacja kodu

1. Klasa Controller (`controller.cpp`)

Klasa `Controller` zarządza główną pętlą gry, inicjalizuje komponenty gry i obsługuje dane wejściowe użytkownika.

Kluczowe Metody

- `int launch()`:
  - Inicjalizuje komponenty gry (`Board` , `Snake` , `UI`).
  - Wyświetla ekran startowy.
  - Tworzy okno gry.
  - Zarządza główną pętlą gry, obsługuje wydarzenia, aktualizuje stan gry i renderuje grę.

Obsługa Zdarzeń

- Wykrywa zdarzenia zamknięcia okna.
- Rozpoczyna grę po naciśnięciu dowolnego klawisza.

- Zmienia kierunek węża na podstawie wejścia z klawiszy strzałek.

Pętla Gry

- Aktualizuje pozycję węża na podstawie zegara.
- Sprawdza warunki zakończenia gry.
- Renderuje komponenty gry (plansza, wąż, wynik) przy użyciu klasy `UI`.

2. Klasa UI (`ui.cpp`)

Klasa `UI` obsługuje interfejs graficzny, w tym wyświetlanie ekranu startowego, ekranu końca gry oraz rysowanie komponentów gry.

Kluczowe Metody

- `bool displayStartScreen()`:
  - Wyświetla ekran startowy z animacjami.
  - Czeka na dane wejściowe od użytkownika, aby rozpocząć grę.
- `void displayGameOver()`:
  - Wyświetla ekran końca gry.
- `void drawBoard(sf::RenderWindow &window, Board board, sf::RectangleShape rectangle)`:
  - Rysuje planszę gry.
- `void drawSnake(sf::RenderWindow &window, Snake snake, sf::RectangleShape &rectangle)`:
  - Rysuje węża.
- `void displayScore(sf::RenderWindow &window, Snake snake)`:
  - Wyświetla bieżący wynik.

Metody Animacji

- `float easeExpIn(float time, float delay = 0)`:
  - Funkcja easingowa wykorzystywana do animacji.
- `float easeSinInOut(float time)`:
  - Funkcja easingowa wykorzystująca sinusoidę do animacji.

3. Klasa Snake (`snake.cpp`)

Klasa `Snake` reprezentuje węża i zarządza jego ruchem, wzrostem i wykrywaniem kolizji.

Kluczowe Metody

- `Snake()`:
  - Konstruktor inicjalizuje węża.
- `std::vector<std::pair<int, int>> getBody()`:
  - Zwraca współrzędne ciała węża.
- `std::pair<int, int> getHead()`:
  - Zwraca współrzędne głowy węża.
- `void setDirection(int dx, int dy)`:
  - Ustawia kierunek ruchu węża.
- `void move()`:
  - Porusza węża w bieżącym kierunku.
- `void grow()`:
  - Ustawia węża do wzrostu przy następnym ruchu.

- `bool checkSelfCollision()`:
  - Sprawdza, czy wąż zderzył się sam ze sobą.
- `void addPoints(int points)`:
  - Dodaje punkty do wyniku węża.

kompilowanie i uruchamianie

wymagania

- wymagana biblioteka SFML
  - program był pisany na dystrybucji linuxa "Fedora" należy zainstaalowac paczki wymienione w tym tutorialu Compiling SFML with CMake (SFML / Learn / 2.6 Tutorials)
    - najlepiej budowac projekt w vscode lub clionie, podczas kompilowania za pomocą samych komend cmake i make w terminalu mogą wystąpić problemy
- kompilator cpp