DOKUMENTACJA PROJEKTU
“GRA W STATKI”

Paweł Buczek, 173599
2EF-DI

Cel projektu

Celem projektu jest odtworzenie kultowej gry w statki poprzez napisanie jej w języku programowania C++. Gra będzie jednak będzie posiadać nowoczesne funkcjonalności oraz tryby.

Funkcjonalności

Gra powinna mieć opcję trybu klasycznego, znanego z oryginalnych statków, a także trybu własnego. W trybie własnym użytkownik powinien móc wybrać, czy chce zezwolić na łamane pozycje statków na planszy, a także, czy po zatopieniu statku pola wokół niego mają być także zatapiane. Do wyboru będzie także liczba statków danego rodzaju na planszy. Dostępne są także opcje losowego wybierania miejsca dla pojedynczego oraz dla wszystkich statków.

Technologie

Język programowania: **ISO C++ 14**

Biblioteka graficzna: **SFML v2.6.1**

Środowisko programistyczne: **Visual Studio 2022 (v143)**

Klasa Game

Opis

Klasa `Game` jest główną klasą gry w statki. Zarządza ona główną pętlą gry, obsługą zdarzeń, rysowaniem na ekranie, a także logiką gry, taką jak strzelanie do statków i sprawdzanie, czy gra została wygrana. Klasa `Game` korzysta z wielu innych klas, takich jak `Board`, `Menu`, `Ship5`, `Ship4`, itd., aby zorganizować różne aspekty gry.

Pola

`bool restart`: Flaga wskazująca, czy można wyświetlić “Od nowa!”

`sf::Image icon`: Ikona okna gry.

`sf::Music music`: Muzyka odtwarzana w tle gry.

`sf::Sound sound`: Dźwięk odtwarzany podczas strzelania.

`sf::SoundBuffer buffer`: Bufor dźwięku dla dźwięku strzału.

`sf::Sound soundEnd`: Dźwięk odtwarzany na końcu gry.

`sf::SoundBuffer bufferEnd`: Bufor dźwięku dla dźwięku końca gry.

`sf::Event event`: Zdarzenie SFML, używane do obsługi zdarzeń wejściowych.

`sf::Clock computerShootClock`: Zegar SFML, używany do kontrolowania strzałów komputera.

`Board board`: Obiekt klasy Board, reprezentujący planszę gry.

`Menu menu`: Obiekt klasy Menu, reprezentujący menu gry.

`Ship5 first5, first5Computer, ...`: Obiekty klasy `Ship5`, reprezentujące statki gracza i komputera.

`Ship4 first4, first4Computer, ...`: Obiekty klasy `Ship4`, reprezentujące statki gracza i komputera.

`Ship3 first3, first3Computer, ...`: Obiekty klasy `Ship3`, reprezentujące statki gracza i komputera.

`Ship2 first2, first2Computer, ...`: Obiekty klasy `Ship2`, reprezentujące statki gracza i komputera.

`Ship1 first1, first1Computer, ...`: Obiekty klasy `Ship1`, reprezentujące statki gracza i komputera.

`OccupiedField occupiedField, occupiedFieldComputer`: Obiekty klasy `OccupiedField`, reprezentujące zajęte pola na planszy gracza i komputera.

`HandleFieldAddition handleFieldAddition`: Obiekt klasy `HandleFieldAddition`, używany do obsługi dodawania pól do planszy.

`ReworkedVersion reworkedVersion`: Obiekt klasy `ReworkedVersion`, reprezentujący nowszą wersję gry.

`ClassicVersion classicVersion`: Obiekt klasy `ClassicVersion`, reprezentujący klasyczną wersję gry.

`HandleGame handleGame`: Obiekt klasy `HandleGame`, używany do obsługi logiki gry.

`HandleSunkShip handleSunkShip`: Obiekt klasy `HandleSunkShip`, używany do obsługi zatopionych statków.

`HandleCustomShips handleCustomShips`: Obiekt klasy `HandleCustonShip` obsługującą wybraną ilość statków.

Metody

`void run()`: Główna pętla gry. Zarządza obsługą zdarzeń, logiką gry i rysowaniem na ekranie.

`void restartGame()`: Metoda do restartowania gry. Resetuje wszystkie obiekty do ich stanu początkowego.

Klasa Board

Opis

Klasa `Board` reprezentuje planszę gry w statki. Jest odpowiedzialna za zarządzanie i wyświetlanie stanu planszy, a także obsługę niektórych zdarzeń związanych z kliknięciem na planszę.

Pola

`vector<sf::RectangleShape> userBoard`: Wektor przechowujący kwadraty planszy użytkownika.

`vector<sf::RectangleShape> userBoardBorder`: Wektor przechowujący granice planszy użytkownika.

`vector<sf::RectangleShape> computerBoard`: Wektor przechowujący kwadraty planszy komputera.

`vector<sf::RectangleShape> computerBoardBorder`: Wektor przechowujący granice planszy komputera.

`sf::Texture backgroundTexture`: Tekstura tła.

`sf::Sprite background`: Sprite tła.

`sf::Font font`: Czcionka używana do wyświetlania tekstu.

`sf::Text title`: Tytuł gry.

`sf::Text user`: Tekst reprezentujący użytkownika.

`sf::Text computer`: Tekst reprezentujący komputer.

`vector<sf::Text> userColumnLabel`: Wektor przechowujący etykiety kolumn planszy użytkownika.

`vector<sf::Text> userRowLabel`: Wektor przechowujący etykiety wierszy planszy użytkownika.

`vector<sf::Text> computerColumnLabel`: Wektor przechowujący etykiety kolumn planszy komputera.

`vector<sf::Text> computerRowLabel`: Wektor przechowujący etykiety wierszy planszy komputera.

Metody

`Board()`: Konstruktor klasy Board. Inicjalizuje wszystkie pola klasy.

`void resetAll()`: Resetuje wszystkie pola planszy do ich stanu początkowego.

`int getPosition(const sf::Event& event)`: Zwraca pozycję na planszy użytkownika, na którą kliknął użytkownik.

`int getPositionComputer(const sf::Event& event)`: Zwraca pozycję na planszy komputera.

`void update(int field)`: Aktualizuje stan określonego pola na planszy użytkownika.

`void updateComputer(int field)`: Aktualizuje stan określonego pola na planszy komputera.

`void cancelUpdate(vector <int> fieldsToDelete)`: Anuluje aktualizację określonych pól na planszy.

`void draw(sf::RenderWindow& window)`: Rysuje planszę na ekranie.

`void updateHit(int field)`: Aktualizuje stan określonego pola na planszy użytkownika po trafieniu.

`void updateHitComputer(int field)`: Aktualizuje stan określonego pola na planszy komputera po trafieniu.

`void updateHitCorrect(int field)`: Aktualizuje stan określonego pola na planszy użytkownika po poprawnym trafieniu.

`void updateHitComputerCorrect(int field)`: Aktualizuje stan określonego pola na planszy komputera po poprawnym trafieniu.

`void updateHitSunk(int field)`: Aktualizuje stan określonego pola po zatopieniu statku.

`void updateHitSunkComputer(int field)`: Aktualizuje stan określonego pola po zatopieniu statku komputera.

Klasa Menu

Opis

Klasa `Menu` jest odpowiedzialna za wyświetlanie menu gry. Zawiera metody do wyświetlania wiadomości i przycisków na ekranie.

Pola

`bool isClicked`: Flaga wskazująca, czy przycisk został kliknięty.

`bool isGameStarted`: Flaga wskazująca, czy gra została rozpoczęta.

`bool isVersionChoosen`: Flaga wskazująca, czy wybrano wersję gry.

`bool toMiddle`: Flaga wskazująca, czy menu przenieść na środek ekranu.

`bool isCheckboxFields`: Flaga wskazująca, czy zaznaczono checkbox.

`bool isCheckboxShips`: Flaga wskazująca, czy zaznaczono checkbox.

`bool customShips`: Flaga wskazująca, czy wyświetlić okno wyboru.

`button`, `buttonText`, `buttonRandom`, `buttonRandomText`, `number5`, `number4`, ..., `description`, `fieldsText`, `shipsText`, `checkboxShips`, `checkboxFields`: Obiekty `sf::RectangleShape` i `sf::Text` reprezentujące przyciski i ich etykiety.

`resetTexture`, `resetSprite`, `offTexture`, `offSprite`, `randomTexture`, `randomSprite`, `randomAllTexture`, `randomAllSprite`: Obiekty `sf::Texture` i `sf::Sprite` używane do wyświetlania przycisków resetowania i wyłączania.

`background`, `message`, `font`: Obiekty `sf::RectangleShape`, `sf::Text` i `sf::Font` używane do wyświetlania tła i wiadomości.

`arrowUp5`, `arrowDown5`, `arrowUp4`, ... : Obiekty `sf::ConvexShape` używane do wyświetlania strzałek w górę i w dół.

Metody

`Menu::Menu()`: Konstruktor klasy `Menu`. Inicjalizuje wszystkie pola klasy.

`void Menu::resetAll()`: Resetuje wszystkie flagi do stanu początkowego oraz ustawia pozycję przycisku `button` na wartości początkowe.

`void Menu::displayMessage(const std::string& messageToShow, sf::RenderWindow& window)`: Wyświetla wiadomość na ekranie, jeśli przycisk nie został kliknięty.

`void Menu::displayButtons(sf::RenderWindow& window)`: Wyświetla przyciski na ekranie, jeśli przycisk nie został kliknięty. Dodatkowo, przycisk `buttonRandom` jest wyświetlany tylko wtedy, gdy gra nie jest jeszcze rozpoczęta.

`void handleCheckboxFields()` : Zmienia wartość flagi `isCheckboxFields`, zmienia zaznaczenie checkboxa.

`void handleCheckboxShips()` : Zmienia wartość flagi `isCheckboxShips`, zmienia zaznaczenie checkboxa.

Klasa HandleFieldAddition

Opis

Klasa `HandleFieldAddition` jest odpowiedzialna za obsługę dodawania pól do gry. Zawiera metody do obsługi różnych typów statków.

Metody

`void handleShipX(sf::Event& event, ShipX& shipX, OccupiedField& occupiedField, Board& board, Menu& menu)`: Obsługuje dodawanie statku typu X do planszy gry. Sprawdza, czy statek może być dodany do planszy, a następnie aktualizuje planszę i ustawia odpowiednie flagi.

`void handleCancelChangesX(ShipX& shipX, Board& board)`: Obsługuje anulowanie zmian dla statku typu X. Jeśli statek nie jest poprawnie umieszczony na planszy, ta metoda usuwa statek i przywraca planszę do stanu sprzed dodania statku.

`void handleAcceptChangesX(ShipX& shipX, OccupiedField& occupiedField)`: Obsługuje akceptację zmian dla statku typu X. Jeśli statek jest poprawnie umieszczony na planszy, ta metoda dodaje statek do listy zajętych pól i ustawia flagę isAdded na true.

`void handleShipXComputer(ShipX& shipX, OccupiedField& occupiedField, Board& board)`: Obsługuje dodawanie statku typu X do planszy gry przez komputer.

`void handleShipXRandom(ShipX& shipX, OccupiedField& occupiedField, Board& board, Menu& menu)`: Obsługuje dodawanie statku typu X do planszy gry w sposób losowy.

W powyższych opisach, X oznacza rozmiar statku, który może wynosić 5, 4, 3, 2 lub 1. Każda metoda jest specyficzna dla danego rozmiaru statku.

Klasa OccupiedField

Opis

Klasa `OccupiedField` jest odpowiedzialna za przechowywanie informacji o polach zajętych na planszy gry. Zawiera metody do dodawania, usuwania i sprawdzania zajętych pól.

Pole

`vector<int> occupiedFields`: Wektor przechowujący numery pól, które są aktualnie zajęte.

Metody

`void resetAll()`: Czyści listę `occupiedFields`, usuwając wszystkie zajęte pola.

`void addOccupiedField(vector<int> fields)`: Dodaje nowe pola do listy `occupiedFields`. Pola są dodawane tylko wtedy, gdy nie są już zajęte.

`void writeOccupiedFields()`: Wyświetla wszystkie zajęte pola. Ta metoda jest obecnie zakomentowana.

`vector<int> getOccupiedFields() const`: Zwraca listę `occupiedFields`.

`int getOccupiedField(int field) const`: Zwraca konkretne pole z listy `occupiedFields`.

`bool isPositionOccupied(int position, vector<int> fields) const`: Sprawdza, czy dana pozycja jest zajęta.

`bool isPositionOccupiedComputer(vector<int> ship) const`: Sprawdza, czy dany statek jest na liście `occupiedFields`.

`void deleteFields()`: Usuwa wszystkie pola z listy `occupiedFields`.

`void repairFields()`: Usuwa duplikaty z listy `occupiedFields`.

Klasa HandleGame

Opis

Klasa `HandleGame` jest odpowiedzialna za obsługę logiki gry. Zawiera metody do zarządzania turami graczy, wybierania wersji gry, a także śledzenia i aktualizowania pól trafionych.

Pole

`bool isUserTurn`: Określa, czy jest tura gracza.

`bool isComputerTurn`: Określa, czy jest tura komputera.

`bool isVersionClassic`: Określa, czy wybrana została klasyczna wersja gry.

`bool isVersionReworked`: Określa, czy wybrana została nowa wersja gry.

`vector<int> additionalFieldsToUpdate`: Wektor przechowujący dodatkowe pola do zaktualizowania.

`vector<int> additionalFieldsToUpdateComputer`: Wektor przechowujący dodatkowe pola do zaktualizowania dla komputera.

`vector <int> hittedFields` : Wektor przechowujący aktualnie trafione pola użytkownika.

`vector <int> hittedFieldsComputer` : Wektor przechowujący aktualne trafione pola komputera.

Metody

`void resetAll()`: Resetuje wszystkie pola klasy do ich wartości początkowych.

`void addHittedField(int field)`: Dodaje pole do listy trafionych pól gracza.

`bool isFieldAvailable(int field) const`: Sprawdza, czy dane pole jest dostępne (nie zostało jeszcze trafione) dla gracza.

`void addHittedFieldComputer(int field)`: Dodaje pole do listy trafionych pól komputera.

`bool isFieldAvailableComputer(int field) const`: Sprawdza, czy dane pole jest dostępne (nie zostało jeszcze trafione) dla komputera.

`bool isShipSunk(vector<int> fields) const`: Sprawdza, czy dany statek gracza został zatopiony.

`bool isShipSunkComputer(vector<int> fields) const`: Sprawdza, czy dany statek komputera został zatopiony.

`void updateAdditionalFields(vector<int> fields)`: Aktualizuje listę dodatkowych pól do zaktualizowania dla gracza.

`void updateAdditionalFieldsComputer(vector<int> fields)`: Aktualizuje listę dodatkowych pól do zaktualizowania dla komputera.

`void simulateMouseClick(sf::RenderWindow& window)`: Symuluje kliknięcie myszą w oknie gry.

`void chooseVersion(Menu& menu, sf::RenderWindow& window)`: Pozwala graczowi wybrać wersję gry.

Klasa HandleSunkShip

Opis

Klasa `HandleSunkShip` jest odpowiedzialna za obsługę logiki zatapiania statków w grze. Zawiera metody do obsługi zatopienia statków o różnych rozmiarach dla gracza i komputera.

Metody

`void handleSunkShipX(HandleGame& handleGame, Board& board, ShipX& shipX)`: Obsługuje zatopienie statku o rozmiarze X dla gracza. Jeśli statek został zatopiony, metoda aktualizuje odpowiednie pola na planszy i ustawia flagę `isSunk` na true dla danego statku.

`void handleSunkShipXComputer(HandleGame& handleGame, Board& board, ShipX& shipX)`: Obsługuje zatopienie statku o rozmiarze X dla komputera. Działa podobnie do metody `handleSunkShipX`, ale dla statków komputera.

W powyższym opisie, X oznacza rozmiar statku, który może wynosić 5, 4, 3, 2 lub 1.

Klasa ClassicVersion

Opis

Klasa `ClassicVersion` jest częścią gry w statki. Jest odpowiedzialna za zarządzanie różnymi aspektami gry, takimi jak obsługa kliknięć użytkownika, strzałów, dodawanie statków do planszy, obsługa zatopionych statków i sprawdzanie, czy gra została wygrana.

Pola

`int lastHitPosition`: przechowuje ostatnią pozycję trafienia.

`int computerShotPosition`: przechowuje pozycję strzału komputera.

`vector<int> hittedFields`: przechowuje wektor trafionych pól.

Metody

`void resetAll()`: resetuje wszystkie pola klasy.

`void handleClick(sf::Event& event, ...)`: obsługuje kliknięcia użytkownika i dodaje statki do planszy.

`void showingMenu(sf::RenderWindow& window, ...)`: wyświetla menu.

`void handleChanges(...)`: obsługuje zmiany na planszy.

`void addingShipsComputer(...)`: dodaje statki komputera do planszy.

`void shootingUser(sf::Event& event, ...)`: obsługuje strzały użytkownika.

`void shootingComputer(sf::Event& event, ...)`: obsługuje strzały komputera.

`void handleSunkShipShips(HandleSunkShip& handleSunkShip, ...)`: obsługuje zatopione statki użytkownika.

`void handleSunkShipShipsComputer(HandleSunkShip& handleSunkShip, ...)`: obsługuje zatopione statki komputera.

`void handleRandomShips(...)`: obsługuje losowe pozycje statków.

`bool isUserWinner(...)`: sprawdza, czy użytkownik jest zwycięzcą.

`bool isComputerWinner(...)`: sprawdza, czy komputer jest zwycięzcą.

Klasa ReworkedVersion

Opis

Klasa `ReworkedVersion` jest częścią gry w statki. Jest bardzo podobna do `ClassicVersion`, ale obsługuje inne rozmiary i ilość statków.

Pola

zawiera te same pola co w `ClassicVersion`, oraz:

`int number5`: Określa liczbę statków 5-masztowych.

`int number4`: Określa liczbę statków 4-masztowych.

`int number3`: Określa liczbę statków 3-masztowych.

`int number2`: Określa liczbę statków 2-masztowych.

`int number1`: Określa liczbę statków 1-masztowych.

Metody

zawiera te same metody co w `ClassicVersion`, oraz:

`void winSound(sf::SoundBuffer& buffer, sf::Sound& sound)`: obsługuje odtwarzanie dźwięku po zwycięstwie.

`void lossSound(sf::SoundBuffer& buffer, sf::Sound& sound)`: obsługuje odtwarzanie dźwięku po przegranej.

`void setNumbers(HandleCustomShips& handleCustomShips, Menu& menu)` : ustawia wartości zmiennych `number5`, `number4` itd. na odpowiednią wartość wybraną przez użytkownika.

Klasa HandleCustomShips

Opis

Klasa `HandleCustomShips` jest częścią gry w statki. Odpowiada ona, za poprawne działanie przycisków strzałek, które ustalają ilość statków wybranych przez użytkownika w trybie własnym.

Metody

`handleAddX(Menu& menu)` : Obsługuje zwiększenie liczby odpowiedniego typu statku po wciśnięciu strzałki.

`handleDelX(Menu& menu)` : Obsługuje zmniejszenie liczby odpowiedniego typu statku po wciśnięciu strzałki.

`int shipsValue(Menu& menu)` : Zwraca obliczoną wartość wszystkich wybranych przez użytkownika statków.

`int valueX(Menu& menu)` : Zwraca obliczoną wartość pojedynczego statku.

W powyższym opisie, X oznacza rozmiar statku, który może wynosić 5, 4, 3, 2 lub 1.

Klasy Ship1, Ship2, Ship3, Ship4, Ship5

Opis

Klasy Ship1, Ship2, Ship3, Ship4, Ship5 są częścią gry w statki. Są odpowiedzialne za przechowywanie informacji o odpowiednio jedno-, dwumasztowym itd. statkach. Zawierają metody do dodawania, usuwania i sprawdzania pól statków.

Pola

`vector<int> fields`: Wektor przechowujący numery pól, które są aktualnie zajęte przez statek.

`bool isAdded`: Flaga wskazująca, czy statek został dodany do planszy.

`bool isSunk`: Flaga wskazująca, czy statek został zatopiony.

Metody

`void resetAll()`: Resetuje wszystkie pola klasy.

`void addField(int field)`: Dodaje nowe pole do listy `fields`. Pole jest dodawane tylko wtedy, gdy nie jest już zajęte.

`void deleteField()`: Usuwa ostatnie pole z listy `fields`.

`bool isHit(int field) const`: Sprawdza, czy dane pole jest trafione.

`int numberOfFields() const`: Zwraca liczbę pól statku.

`int getField(int field) const`: Zwraca konkretne pole z listy `fields`.

`vector<int> getFields() const`: Zwraca listę `fields`.

`bool isFieldsOK() const`: Sprawdza, czy wszystkie pola są poprawne. Pola są poprawne, jeśli ich liczba wynosi 1 i nie przekraczają 100.