DerpyDerp - Dokumentacja

Spis treści

Opis klas

Game

Klasa implementująca wzorzec projektowy *MVC*. Przechowuje jako pola wskaźniki do obiektów innych klas, które obsługują wyświetlanie obrazu oraz logikę gry.

- *void poll_events()* - obsługuje sterowanie klawiaturą oraz myszą
- *void update()* - sprawdza i odświeża stan wszystkich obiektów, wywołując metody z klasy *Updater*
- *void render()* - wyświetla w okienku wszystkie obiekty posiadające graficzną reprezentację (gracz, przeciwnicy, itd.) wywołując metody z klasy *Renderer*
- *const bool is_window_open() const* - wywołuje metodę *is_window_open* z klasy *Window*
- *void intialize_variables()* - odpowiada za inicjalizację pól klasy, wektorów obiektów oraz zainicjowanie okna

Window

Klasa odpowiedzialna za wyświetlanie obrazu

- *const bool is_window_open() const* - sprawdza stan wyświetlanego okna
- *void initialize_window(int FPS)* - odpowiada za utworzenie okna oraz ustawienie jego parametrów (rozmiar, ogranicznik klatek, tryb wyświetlania obrazu)
- *sf::RenderWindow* get_render_window()* - zwraca wskaźnik do wyświetlanego okna

HandleText

Klasa odpowiedzialna za generowanie tekstu * *void set_text(std::string name, int size, double pos_x, double pos_y, sf::Color color)* - ustawia parametry wyświetlanego tekstu (treść, rozmiar, pozycja, kolor)

Renderer

Klasa odpowiedzialna za wyświetlanie sprite'ów w okienku

- *void render_player(Player*, sf::RenderWindow*);* - odpowiada za wyświetlanie sprite'a gracza
- *void render_enemies(std::vector<Enemy*>&, sf::RenderWindow*)* - odpowiada za wyświetlanie sprite'ów przeciwników
- *void render_food(std::vector<Food*>&, sf::RenderWindow*)* - odpowiada za wyświetlanie sprite'ów bonusów oraz pożywienia
- *void render(Player*, std::vector<Enemy*>&, std::vector<Food*>&, sf::RenderWindow*, sf::Text)* - wywołuje wszystkie pozostałe metody klasy, renderując wszystkie sprite'y

Updater

Klasa odpowiedzialna za sprawdzanie oraz aktualizowanie stanu obiektów gracza, przeciwników oraz jedzenia/bonusów

- *void update_player(Player* player, std::vector<Enemy*>& enemies, std::vector<Food*>& food)* - odpowiada za detekcję kolizji gracza z przeciwnikami, jedzeniem oraz wyświetlanie informacji o wygranej lub porażce
- *void update_enemies(std::vector<Enemy*>& enemies, std::vector<Food*>& food)* - odpowiada za poruszanie przeciwnikami, detekcję kolizji z pożywieniem oraz graczem i odpowiednio nadawanie przeciwnikom określonych efektów lub usuwanie ich z gry
- *void update_food(std::vector<Food*>& food)* - odpowiada za usuwanie zjedzonego pożywienia oraz generowanie nowego w losowych miejscach na planszy
- *void update(Player* player, std::vector<Enemy*>& enemies, std::vector<Food*>& food)* - wywołuje pozostałe metody klasy

Entity

Klasa wirtualna, dziedzicząca po *sf::Sprite*. Przechowuje podstawowe pola i metody wspólne dla *Player*, *Enemy* oraz *Food*

- *virtual void draw(sf::RenderTarget&, sf::RenderStates) const = 0* - wirtualna funkcja odpowiedzialna za wyświetlanie sprite'a w oknie
- *float get_radius()* - zwraca promień obiektu (kółka)
- *virtual void grow(float r)* - wirtualna funkcja odpowiedzialna za zwiększenie rozmiaru obiektu
- *int is_collision(Entity*)* - funkcja wspólna dla wszystkich klas dziedziczących po *Entity*, odpowiada za detekcję kolizji z innymi obiektami na planszy

Player

Klasa implementująca wzorzec projektowy *Singleton*. Przechowuje podstawowe informacje o graczu.

- *virtual void draw(sf::RenderTarget&, sf::RenderStates) const* - odpowiada za rysowanie gracza na planszy
- *static Player* get_instance()* - sprawdza czy istnieje już instancja klasy i odpowiednio tworzy ją lub zwraca już istniejąca

- *void Player::kill()* - usuwa sprite'a gracza z planszy

Enemy

Klasa przechowująca podstawowe informacje o przeciwniku.

- *virtual void draw(sf::RenderTarget&, sf::RenderStates) const* - odpowiada za rysowanie przeciwnika na planszy

Food

Klasa przechowująca informacje o danym bonusie (jego efekty, stan, kolor)

- *virtual void draw(sf::RenderTarget&, sf::RenderStates) const* - odpowiada za rysowanie pożywienia na planszy
- *Effects get_effect()* - zwraca efekt nakładany przez obiekt

Autor: Krystian Jasionek