Dokumentacja do projektu

Opis projektu

Moim projektem zaliczeniowym, jest prosta **gra 2d w języku C++**, pisana przy użyciu biblioteki **SFML 2.4**. Gracz porusza się po mapie, walczy z potworami i rozwija swoją postać, by ostatecznie pokonać wielkiego potwora, który zagraża pobliskiej wiosce.

Instrukcja do gry

Gracz porusza się po mapie za pomocą klawiszy ‘w’ ‘a’, ‘s’, ‘d’, a z potworami walczy przy użyciu klawisza spacji. Za każdego ogra dostaje 50 pkt doświadczenia. Na każdy kolejny poziom potrzeba o 100 pkt doświadczenia więcej. Jednocześnie każdy kolejny poziom daje postaci gracza +1 do ataku oraz +1 do obrony.

Instalacja gry

Do gry potrzebna jest do biblioteka SFML 2.4, należy ja pobrać z oficjalnej strony i odpowiednio zainstalować. Następnie do kompilacji projektu należy użyć dołączonego pliku MakeFile.

Wymagania techniczne

Jako, że biblioteka SFML jest przenośna, projekt powinien działać na każdym systemie, który wspiera biblioteka. Gra pod względem graficznym jest bardzo prosta dla tego praktycznie każdy komputer da sobie z nią radę bez jakichkolwiek problemów. Połączenie z internetem nie jest wymagane. Gra przetestowana na:

- Ubuntu 16.04.2 LTS
- CPU: Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz
- RAM: 8GB DDR3 1600 MHz

Czego nie zdążyłem zaimplementować

- końcowego bossa
- poruszania się przeciwników
- gracz atakuje ogry, ale ogry gracza jeszcze nie

Dokumentacja kodu

Kod jest bardzo dobrze opisany w komentarzach, a samoopisujące się nazwy metod i atrybutów/zmiennych ułatwiają jego czytanie i zrozumienie, dlatego też przedstawię tutaj tylko ogólny opis zawartości plików, opis klas i ich hierarchię.

Plik main.h i main.cpp:

Struktury:

- Position – zawiera dwie współrzędne typu float, które określają pozycje x i y obiektów w grze
- ActiveDirection – zawiera cztery zmienne typu bool, które określają, w którym kierunku gracz chce się przemieścić

Klasy:

- Object – podstawowa klasa wszystkich obiektów w grze, zawiera strukturę Position, oraz dwie metody – do pobierania pozycji obiektu i do jej ustawiania.

Plik map.h i map.cpp:

Klasy:

- Map – kafelkowa mapa w tle, klasa posiada metodę służącą do wczytania mapy z pliku wczytania tekstur oraz metodę do wyświetlania jej w grze.

Plik obstacle.h i obstacle.cpp:

Klasy:

- Obstacle : public Object – klasa reprezentująca pojedynczą przeszkodę w grze, posiada tylko jeden atrybut – numer tekstury.
- Obstacles – klasa zawierająca vector obiektów Obstacle i metody do zarządzania nimi, wczytywania z pliku i rysowania. Taki „zarządca” przeszkód.

Plik player.h i player.cpp:

Klasy:

- Player : public Object – klasa reprezentująca gracza, ma wszystkie potrzebne atrybuty: HP, maxHP, exp, lvl, speed, str, def, kilka sf::Time potrzebnych do odmierzania cooldownu ataku i poruszania się niezależnie od liczby wyświetlanych klatek itd. Klasa zawiera metody do aktywowania i dezaktywowania kierunków przemieszczania gracza, metodę do faktycznej zmiany pozycji w zadanym kierunku, ustawiania pozycji gracza na zadaną pozycje oraz kilka prostych metod do pobierania lub ustawiania atrybutów.

Plik enemy.h i enemy.cpp:

Klasy:

- Enemy : public Object – klasa reprezentująca wszystkich przeciwników. Zawiera takie atrybuty jak: enemyHP, enemyStartHP (maxHP), enemySpeed, enemyStr, enemyDef, bool

deadflag, Position startPosition – pozycja na która ma się potwór respawnować, sf::Time deathTime – czas zgonu potwora, potrzebny do zrespawnowania go po minięciu określonego czasu. Zawiera metodę do wykrywania kolizji między graczem, a przeciwnikiem oraz wiele podstawowych metod do pobierania i ustawiania atrybutów.

- Ogr : public Enemy - klasa reprezentująca już konkretnego przeciwnika, w konstruktorze ustawia atrybuty z klasy Enemy, na odpowiednie wartości dla tego potwora, oraz posiada metodę do wyświetlania potwora w głównej pętli gry.

- Monsters - „zarządca" przeciwników. Zawiera sf::Time ogrRespawnTime – czas respawnu potworów typu ogr, teksturę ogra, vector ogrów oraz int ogrExp – ile expa gracz dostaje po zabiciu ogra. Zawiera metody:

  - Enemy* getMonsterOnPosition(Position position) - przeszukuje vector ogrów i sprawdza czy jest jakiś na zadanej pozycji, jeśli tak to zwraca go, jeśli nie to zwraca NULL

  - Enemy* getMonsterOnAtackArea(Position playerPosition, int attackDirection, float atackRange); - zwraca pierwszego w pamięci potwora który jest w zasięgu ataku

  - void loadOgrs(string fileOgrsPosition, string fileTextureOgr) - wczytuje potwory z pliku oraz wczytuje ich teksturę

  - void drawOgrs(sf::RenderWindow& window) – metoda rysująca żywe ogry

  - void updateDeadMonsters(sf::Time mainClockTime, Player& player) - aktualizuje potwory - uśmierca te co maja < 0 HP, daje exp dla gracza za zabicie potwora, respawnuje potwory

  - void drawMonstersHP(sf::RenderWindow& window) - metoda rysująca nad potworami ich paski HP

Plik game.h i game.cpp:

Klasy:

- Game – główna klasa w całym programie – zawiera wszystkie inne elementy oraz główną pętle programu. Zawiera dwa sf::Clock służące do mierzenia czasu (poprawne przemieszczanie się gracza oraz cooldown ataku i respawn ogrów), zawiera obiekt gracza – Player player, okno gry, widok gry, mapę, przeszkody, potwory itd. Zawiera metody:

  - Game() - ustawiamy wartości początkowe, wczytujemy tekstury, font itd.

  - void update(sf::Time timeBetweenFrames) – główna metoda aktualizująca grę co wyświetlaną klatkę

  - void drawUI() - metoda rysująca UI

  - void runGame() - główna pętla gry

Bibliografia:

- https://www.sfml-dev.org/index.php – tutoriale i dokumentacja