

Studenckie RPG - Dokumentacja Projektu

Antoni Przybylik

PROI 23L

GRUPA 103

Zoja Hordyńska

PROI 23L

GRUPA 103

10 kwietnia 2023

Cel i opis projektu

Przedmiotem projektu jest gra „Studenckie RPG”- RPG inspirowane grą Diablo. Gracz (student) przechodzi przez mapę Politechniki, zbierając przedmioty i walcząc z potworami godnymi studiów. Każdy gracz posiada pewne umiejętności, zdrowie i manę. Podczas walki, gracz rzuca umiejętności, które zadają losowe obrażenia z danego przedziału (obrażenia są zwiększane przez przedmioty oraz inne umiejętności). Celem gry jest zabicie wszystkich bossów i tym samym przeżycie studiów.

Projekt ma zostać zrealizowany w języku C++ z użyciem jego zaawansowanych mechanizmów i ze szczególnym uwzględnieniem paradygmatu obiektowego. Planowane jest stworzenie dedykowanego silnika gry zaimplementowanego przy użyciu biblioteki SFML, który będzie warstwą abstrakcji między wysokopoziomowym kodem właściwej gry operującym na „duszkach”, a interfejsem biblioteki SFML pozwalającym na proste operacje takie jak wyświetlanie obrazów.

Architektura

Projekt jest podzielony na cztery moduły:

- Silnik gry
- Ciało gry
- Launcher
- Aplikacja gry

Silnik gry dostarcza interfejs zaprojektowany w paradygmacie obiektowym. Jest on oparty o model „Sprite’owy”. Użytkownik tworzy „duszki” (ang. sprite) i określa ich interakcje z otoczeniem. Następnie dodaje duszki do silnika i uruchamia silnik. Silnik symuluje zachowanie duszków podane przez użytkownika. Programowanie obiektowe pozwala na przetwarzanie klasy Sprite (duszek) i umieszczenie w nim kodu który zostanie wykonany na odpowiednich zdarzeniach takich jak zderzenie dwóch duszków lub kodu który jest wykonywany w każdym tyknięciu zegara.

```
1      /* Utworzenie duszka. */
2      skin = new SpriteSkin(*tiles, 3, 500, 1);
3      sprite = new Sprite(skin,
4                          Rect(0, 0, win_x, win_y),
5                          BA_BOUNCE);
6      sprite->set_position(Rect(100, 100, 75, 75));
7      sprite->set_velocity(Vector(120, 250));
8
9      /* Silnik. */
10     engine = new Engine();
11     engine->bind_window(window);
12     engine->add_sprite(sprite);
13
14     engine->exec();
```

Listing 1: Interfejs silnika - kod poglądowy

Działanie tego kodu jest zademonstrowane na filmie.

Ciało gry to kod implementujący właściwą rozgrywkę. To w nim znajdują się klasy takie jak: Player, Enemy, Spell, czy Item.

Klasa Player przechowuje stan gracza: liczbę punktów zdrowia, manę, moc i ekwipunek. Metody klasy będą pozwalać na zarządzanie ekwipunkiem i modyfikację parametrów gracza. TODO: Dziedziczy po Sprite.

Klasa Enemy jest klasą bazową dla wrogów. Jej interfejs powinien pozwalać na walkę niezależnie od tego, z jakim przeciwnikiem mamy do czynienia.

Klasa Spell jest pomniejszą klasą bazową dostarczającą interfejs czar. Wszelkie parametry czar są wirtualne, a interfejs ma być taki sam dla wszystkich klas pochodnych.

Item jest pomniejszą klasą bazową implementującą interfejs przedmiotu.

```
1 class Sword : public Item {  
2 public:  
3     virtual Sword(void);  
4     virtual ~Sword(void);  
5  
6     virtual void action(void);  
7 };
```

Listing 2: Klasa „Sword” - kod poglądowy

Launcher jest programem służącym do uruchomienia gry. Można w nim modyfikować ustawienia rozgrywki. Zakładane możliwości dostosowania ustawień rozgrywki obejmują: zmianę mapy, wczytanie stanu gry z pliku. <TODO>

Aplikacja gry jest programem który służy do grania w grę. W oknie aplikacji gry jest renderowana symulacja gry. Aplikacja gry powinna dodatkowo pozwalać na zatrzymanie gry, zmianę ustawień i wyjście z gry. <TODO: Napisz coś o aplikacji gry.>

Wymagania

Gra wymaga zainstalowanej biblioteki SFML. Założone jest działanie gry na systemach operacyjnych Windows i Linux. Opcjonalnie gra mogłaby wspierać system MacOS. Pewnie Makefile dla Linuksa będzie działał na MacOS, ale nie mam tego systemu i nie zobowiązuje się że gra będzie na nim prawidłowo działać i kompilować bez ostrzeżeń.

<TODO: Napisać więcej.>

Instrukcja użycia

<TODO>

Planowane biblioteki i narzędzia

<TODO>

Podział pracy

Antoni

- Silnik
- Części ciała gry

Zoja

- Menu
- Interfejs gry
- Tekstury i obrazki
- Części ciała gry: Player, Enemy, Spell, Item