Studenckie RPG - dokumentacja końcowa projektu

Antoni Przybylik
PROI 23L
GRUPA 103

Zoja Hordyńska
PROI 23L
GRUPA 103

30 maja 2023

Opis gry

Studencka gra RPG to wyzwanie dla miłośników przygód, którzy chcą zmierzyć się z największą zmorą inżynierów - Politechniką. Czy uda Ci się przeżyć te studia? Przejdź się korytarzami uczelni wśród gąszczu książek, pustych biurek i płaczących nad kodami studentów, na ścianach widząc istny koszmar programisty – wypisany czerwonymi literami ERROR. Niech nie zgasi Twego zapału oschłość Pani z dziekanatu, dbałość o szczegóły prowadzącego czy bezwzględność, wobec Twego zdrowia psychicznego, sesji. Legenda miejsca głosi, iż nikt jeszcze nie przeżył Politechniki bez uszczerbku na zdrowiu. Czy będziesz pierwszym, który dokona niemożliwego?

Instrukcja obsługi

Po uruchomieniu programu, na ekranie ukaże się menu startowe. Aby wybrać jedną z dostępnych opcji należy użyć strzałek (górnej lub dolnej), a następnie wcisnąć klawisz ENTER. “Play” rozpocznie grę, a “Exit” spowoduje zamknięcie okna menu.

Po wciśnięciu “Play” i pojawieniu się postaci na mapie, możesz swobodnie się po niej poruszać, używając, intuicyjnie działających, strzałek. Po spotkaniu na swej drodze jednego z trzech bossów, kliknij na niego, aby rozpocząć quiz. Wybierz odpowiedź za pomocą strzałek, a następnie zatwierdź swój wybór klawiszem ENTER. Wciśnij spację, aby przejść do kolejnego pytania. Po odpowiedzeniu na wszystkie pytania (bądź utracie wszystkich punktów zdrowia) zamknij okno za pomocą klawisza ESC. Nie tracąc wszystkich punktów zdrowia, odpowiedz na wszystkie pytania i przeżyj studia! Podczas rozgrywki możesz wczytać, bądź zapisać aktualną rozgrywkę.

Zmiany względem wcześniejszych założeń

W pierwotnym planie gry gracz miał walczyć z przeciwnikami, używając do tego przedmiotów, jednak jako zwolennicy polubownego rozwiązywania konfliktów, stwierdziliśmy, że dużo ciekawsze i bardziej pasujące do klimatu studiów będzie przeprowadzenie mini quizów z, odpowiednimi dla każdego bossa, pytaniami.

Architektura

Projekt jest podzielony na cztery moduły:

- Silnik gry
- Ciało gry
- Launcher
- Launcher pytań

- Aplikacja gry

Silnik

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. Polimorfizm w języku C++ pozwala na nadpisywanie w klasach dziedziczących po klasie Sprite (duszek) funkcji wirtualnych i dzięki temu można w duszkach umieszczać kod który zostanie wykonany na odpowiednich zdarzeniach takich jak zderzenie dwóch duszków lub kodu, który jest wykonywany w każdym tyknięciu zegara.

Ciało Gry

Ciało gry to kod implementujący właściwą rozgrywkę.

Klasa Player przechowuje liczbę punktów zdrowia gracza.

Klasa Enemy jest klasą, opisującą wrogów. Każdy wróg posiada wektor indywidualnych pytań oraz punktów zdrowia, które zabierają graczowi za niepoprawną odpowiedź. Metody klasy pozwalają na zarządzanie zestawem pytań, wyświetlaniem ich treści oraz odpowiedzi, a także modyfikację parametrów gracza.

Klasa Question przechowują informacje dotyczące pytań - ich treść oraz dostępne odpowiedzi.

Launcher

Launcher jest programem służącym do uruchomienia gry. Z jego pomocą rozpoczyna się rozgrywkę, zapisuje stan gry do pliku, a także wczytuje stan wcześniej zapisanej gry.

Launcher pytań

Launcher jest programem odpowiednio zarządzającym, dostępnym w grze, quizem. Wyświetla się tam treść pytań wraz z dostępnymi odpowiedziami. Zła odpowiedź odbiera życie graczowi, a odpowiedzenie na wszystkie pytania przeciwnika sprawia, że znika z mapy.

Aplikacja Gry

Aplikacja gry jest programem który służy do grania w grę. W oknie aplikacji gry jest renderowana symulacja gry. Aplikacja gry pozwala na zapis gry oraz wczytanie wcześniej zapisanej mapy.

Wymagania

Gra wymaga zainstalowanej biblioteki SFML. Założone jest działanie gry na systemach operacyjnych Windows i Linux.

Biblioteki i Narzędzia

Projekt w całości został zrealizowany w języku C++, w standardzie C++20. Dopuszczalne jest używanie rozszerzeń GCC.

Skrypty pozwalające na zbudowanie projektu są kompatybilne z programem make i do zbudowania projektu został użyty toolchain GCC. Do kompilacji gry na Windowsa został użyty MinGW.

Projekt został zrealizowany przy użyciu biblioteki SFML, biblioteki standardowej języka C++ (libstdc++), biblioteki standardowej języka C (libc) i biblioteki matematycznej (libm).

W projekcie nie jest wymagane ścisłe trzymanie się kanonicznej wersji tych bibliotek. Dopuszczalne jest użycie rozszerzeń GLIBC.

Obsługa wyjątków i testy

Zidentyfikowane sytuacje wyjątkowe to np. utworzenie gracza o niedodatnich punktach życia czy ujemnej pozycji, próba odwołania się do nieistniejącego obiektu Question w wektorze questions w klasie Enemy, dodanie do niego obiektu już istniejącego, bądź próba usunięcia obiektu, który się w wektorze nie znajduje. W takim wypadku w terminalu ukazuje się odpowiedni komunikat wraz z nieprawidłową wartością, która spowodowała sytuację wyjątkową. Ponadto pod uwagę brane są wyjątki rzucane w trakcie działania programu - "runtime error". Testy jednostkowe sprawdzają prawidłowe działanie klas: Enemy, Player, Question oraz interfejsów pytań i menu.

Podział Pracy

Antoni

- Silnik
- Zapisywanie oraz wczytywanie rozgrywki
- Generator mapy
- Sprite'y związane z właściwą rozgrywką
- Wybrane części gry

Zoja

- Menu
- Interfejs pytań
- Tekstury i obrazki
- Ciało gry; Klasy: Player, Enemy, Question
- Testy jednostkowe i obsługa wyjątków