Dawid Bartosiak 318361

Zuzanna Godek 318373

Sprawozdanie z projektu PROI 22L.

„Maro The Albanian Electrician™”

Założenia projektu:

W projekcie chodziło o nauczenie się obsługi interfejsu z biblioteki graficznej, utrwalenie zdobytej na laboratorium wiedzy z zakresu projektowania obiektowego. Dodatkowym elementem projektu było sprawdzenie swoich umiejętności w zakresie stworzenia własnej implementacji fizyki w grze 2d.

Wykorzystane biblioteki:

1. SFML - obraz 2d, operacje na gotowych, zawartych w bibliotece wektorach, wsparcie w obsłudze gry.
2. chrono - odmierzanie czasu
3. vector - biblioteka do obsługi wektorów objektów
4. array - lepsze tablice dla mapy
5. fstream - zapis i odczyt do pliku
6. iostream - działanie na std::string

Podział projektu i opis klas

- Maro – klasa gracza, jest ona tworzona bezpośrednio po rozpoczęciu gry. Ma w sobie metody pozwalające na interakcje z otoczeniem oraz komunikację na poziomie użytkownik-klasa.
- Roomba – przeciwnicy Maro, odpowiednik Goomby. Klasa sama wykonuje ruch i ma metody, które pozwalają jej na interakcje z otoczeniem

- Mushroom – grzybki z Maro, działają podobnie do oryginału, czyli znikają po zjedzeniu ich przez maro, czym sprawiają, że Maro rośnie.
- LevelManager – klasa odpowiedzialna za zarządzanie poziomem. Obsługuje ona pliki formatu .png i przekształca je następnie na odpowiedni dla działania programu format.
- GameManager – odpowiada za ciągłe aktualizacje gry. Odpowiada zatem za ponawianie wywołań funkcji ruchów we wszystkich klasach oraz za zamieszczanie ich na ekranie, przechowując w sobie wszelkie obiekty
- Collision – Nasza implementacja fizyki, dokładniej kolizji z mapą, która dzieje się w systemie binarnym dla każdej komórki z którą w następnej klatce mamy część wspólną:

$$\begin{vmatrix} 1 & 2 \\ 4 & 8 \end{vmatrix}$$

- Animation – Nasza implementacja animacji, także na podstawie plików .png złożonych z kilku sąsiadujących obrazków – dla nas będących pojedynczymi klatkami
- Character – Klasa, która zawiera wszelakie gettery i atrybuty, które łączą klasy Maro, Roomba i Mushroom.

Dodatkowo w projekcie znajduje się plik Consts.h przechowujący zmienne globalne, biblioteki, typy wyliczeniowe oraz typedef i struct. W pliku main.cpp mamy zaimplementowany klasyczny model obsługi biblioteki graficznej do gier2d oraz odczyt, zapis i wszelakie komunikaty pojawiające się graczowi na ekranie.

Balans

W celu odpowiedniego zbalansowania gry przeprowadzane były przez nas testy na osobach z roku oraz koła naukowego robotyki „Bionik”. Po wielu próbach (bardziej i mniej udanych), przedstawiamy nowy system balansu:

Sekunda – 25pkt

Moneta – 200pkt

Grzyb – 200pkt

Roomba – 200pkt

Grzyb w czasie posiadania wzmocnienia – 1000pkt

Gracze po naniesionych przez nas zmianach, mieli większą satysfakcję z gry i możliwości pobijania rekordu. Próba doświadczalna liczyła 42 osoby. Aktualny najwyższy wynik zapisany jest w pliku highScore.txt

Część refleksyjna

Nasza praca z założenia była znacznie bardziej ambitna. Wiele mechanizmów rozgrywki musieliśmy wyciąć ze względu na ograniczenia czasowe, jednakże w planach mamy dalszy rozwój gry, jako otwarty projekt, w który będzie mógł zagrać każdy. W ramach tej rozbudowy gry planujemy, np. dodać nowych przeciwników, ale także zmienić system kolizji, ponieważ obecny ma ograniczenia. W czasie pracy nad projektem napotkaliśmy wiele problemów, jak chociażby błędne wskazywanie kolizji, błędy z teksturami, błędne interakcje w obrębie obiektów. Ze wszystkimi problemami udało nam się uporać, przez co aktualna wersja gry (v0.1) jest wolna od poważniejszych błędów i można w nią bezproblemowo grać. W naszej opinii wykonaliśmy projekt na poziomie zadowalającym, szczególnie biorąc pod uwagę ramy czasowe jakie na niego mieliśmy wyznaczone.

Struktura klas

W naszym projekcie nie była potrzebna zaawansowana struktura klas, zatem najważniejsze elementy to:

1. GameManager zarządza wszystkimi obiektami w czasie rozgrywki
2. Klasy Mushroom, Roomba, Maro i ewentualne przyszłe klasy, które planujemy dodać, dziedziczą po klasie Character.

Działanie programu

Nasz projekt jest prostą grą platformową, zatem w celu zagrania w nią wystarczy jedynie mieć pobraną bibliotekę SFML, a następnie wszystko razem skompilować (tutaj uwaga: Visual Studio 2022 wymaga, by zawrzeć w katalogu gry plik z OpenGL, zaś konfiguracja SFML u każdego może wyglądać inaczej).