Polynomial Fighter - dokumentacja

Projekt realizowany w ramach przedmiotu Programowanie III

Piotr Wawrzyńczyk, Bartosz Ścigała
Wydział Matematyki Stosowanej
Politechnika Śląska

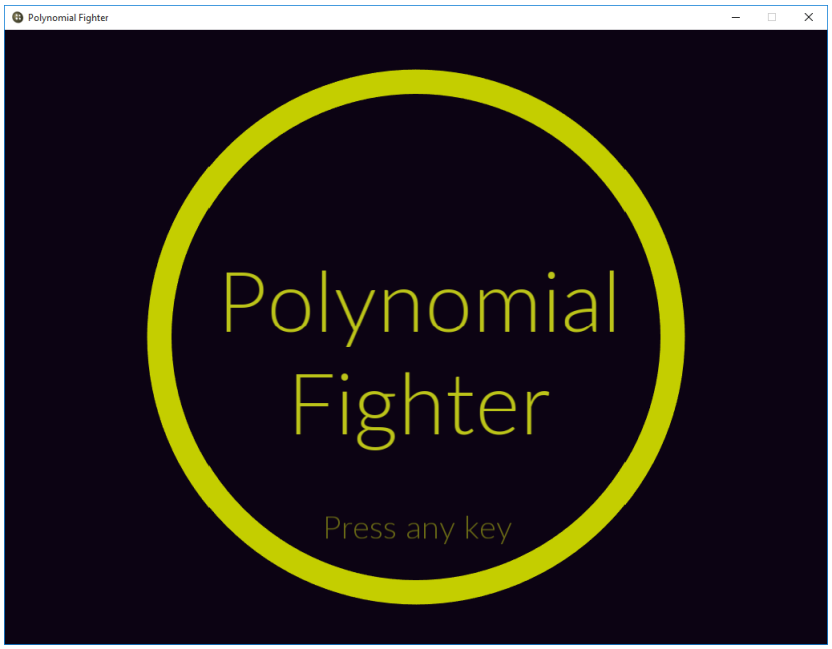
16 stycznia 2018

Spis treści

1 Opis programu

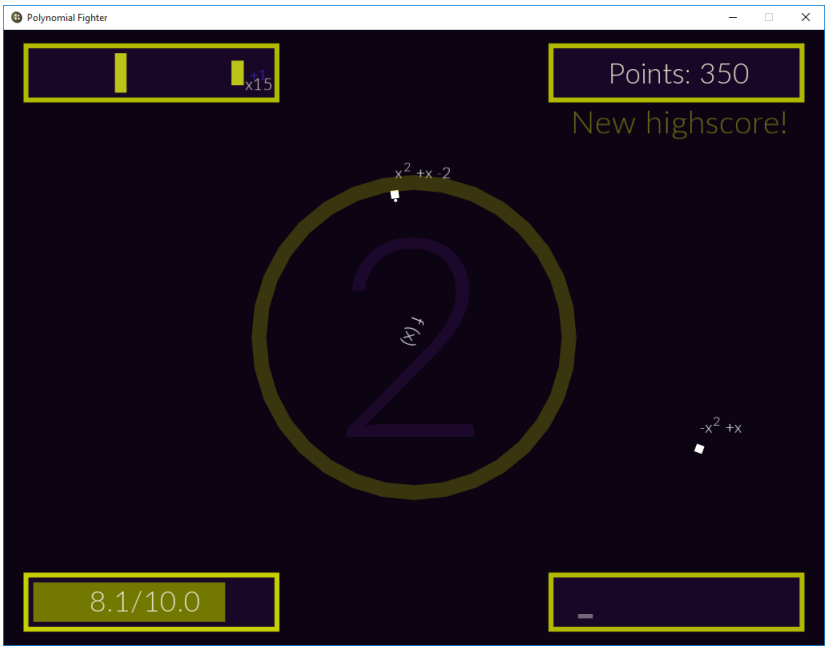
Program “Polynomial Fighter” jest edukacyjną grą komputerową, mającą na celu umożliwienie nauki płynnego rozwiązywania nieliniowych równań diofantycznych w zakresie do stopnia VII oraz współczynników poszczególnych iloczynów nie większych niż 6.

1.1 Opis świata przedstawionego



Rysunek 1: Ekran wstępny gry"

Gracz jest przedstawiony jako samotna f(x), broniąca się przed nadciągającymi ze wszystkich stron[1] wielomianami. Wielomiany, po dotarciu do okolicy pierścienia rozpoczynają ostrzał - tym cięższy i szybszy, im wyższy jest ich stopień. Punkty życia gracza są przedstawione w lewym dolnym rogu ekranu (są one tracone w momencie trafienia i odzyskiwane w momencie zniszczenia przeciwnika). Gracz musi baczyć na to, by nie zużyć całej amunicji, której licznik jest widoczny w lewym górnym rogu ekranu. Amunicja jest dodawana cyklicznie oraz - w większych ilościach - w momencie przejścia do następnego poziomu.



Rysunek 2: "Przykładowy ekran gry"

1.2 Interakcja z użytkownikiem

Użytkownik może oddziaływać z grą, używając klawiatury - do poruszania się w menu używając klawiszów strzałek / WASD, do zatwierdzania klawiszy spacji lub enter, do wychodzenia / przerywania klawisza escape. Dodatkowo, w czasie gry można ją zatrzymać klawiszem 'p'.

1.3 Obsługa pola tekstowego

Rysunek 3: "Pole tekstowe z przykładowym wejściem"

Pole tekstowe jest swoistym źródłem obrony wobec nadciągających wielomianów. Gdy grający wpisze właściwy pierwiastek (wyliczony z postaci wielomianu obecnego na scenie), umieszona pośrodku planszy manifestacja gracza wyceluje w odpowiadające mu wielomiany oraz odda strzał. Gdy "życie"wielomianu - rozumiane jako jego stopień dobiegnie końca (czyli zostanie zredukowane do zera),

[1]Nie do końca - są oni produkowani tak, by nie nachodzili na elementy GUI.

ten jest niszczony, a gracz otrzymuje punkty (widoczne w prawym górnym rogu okna).

2 Opis programu

2.1 Algorytmy

- Interpolacja liniowa - została ona wykorzystana do stworzenia płynnych przejść w menu oraz na ekranie rozgrywki.
- Tworzenie wielomianów i przejścia między ich postaciami - wielomianowe API godne osobnej biblioteki pozwoliło nam na płynne i szybkie tworzenie podstawowych mechanik gry.
- Kontenery standardowe - użycie kontenerów biblioteki standardowej STL (takich jak *vector*,*array* czy *map*) znacznie ułatwiło tworzenie gry, pozwalając nam skupić się na samej produkcji gry.

2.2 Wykorzystanie *smart pointers*

Jedną z wielu funkcjonalności wprowadzonych w nowoczesnym C++ są tak zwane sprytne wskaźniki, odciążające programistę od dbania o zarządzanie pamięcią. Ich użycie pozwoliło na skupienie się nad rozwiązywaniem problemów postawionych w projekcie.

2.3 Przegląd klas oraz funkcjonalności

`AssetManager` Klasa singleton, reprezentująca kontener zasobów graficznych oraz dźwiękowych. Wczytane zasoby - przechowywane w *std::map* oraz identyfikowane po nazwie - są współdzielone, co pozwala na zdecydowane zmniejszenie ilości wykorzystywanej pamięci.

`FloatColor` Odpowiednik klasy *sf::Color*. Wykorzystywany z powodu niskiej dokładności składowych wbudowanej w bibliotekę SFML klasy przedstawiającej kolor.

`Stopwatch` Byt pomocniczy dający możliwość ustawienia stopera i odebrania zdarzenia po upływie określonego czasu. te klasy co trzymaja dane

`RandomGenerator` Klasa statyczna zapewniająca odpowiedni stopień losowści wymaganych danych, razem z opracowanymi specjalizacjami do typów (wliczono tutaj także typ *sf::Color*.

`Przestrzeń nazw Time` Zawiera klasę singleton Timer, uruchamianą w momencie aktywacji okna, umożliwiającą uzyskanie informacji o czasie jaki upłynął od ostatniego restartu zegara gry oraz o czasie, który upłynął od ostatniej klatki. Dane te przekazywane są w strukturze TimeData, udostępniającej metody zwracające czas z uwzględnieniem skali czasowej.

Utility Zbiór globalnych funkcji udostępniający wiele popularnych i niezbędnych do działania wielu innych jednostek algorytmów. Występują tu na przykład interpolacja liniowa, operacje na wektorach czy funkcje pomocnicze do obsługi rotacji.

Debug Klasa ze statycznymi metodami, umożliwiająca wypisywanie komunikatów na ekran konsoli. Posiada metody *PrintFormatted* oraz *PrintErrorFormatted*, powodujące wypisanie żądanego komunikatu – odpowiednio zatrzymującej i niezatrzymującej działania programu. Użyto tu szablonu funkcji, który daje możliwość wypisywania dowolnych zmiennych – pierwszym argumentem jest łańcuch znaków, dalej opcjonalnie można podać zmienne o dowolnych typach. Ich wartości zostaną wpisane do miejsc oznaczonych '%' w pierwszym argumencie.

Delegate Nazwana na cześć podobnej funkcjonalności z języka C#, zmniejsza zapotrzebowanie na wiązanie klas poprzez opakowywanie wskaźników na funkcje. Klasa Entity przedstawia obiekt, udostępniający przeciążalne funkcje do aktualizacji (update), wywoływania przy zniszczeniu obiektu (onDestroy), rysowania (draw) oraz aktywacji/dezaktywacji obiektu (set/get Enabled). Dodatkowo przechowywane są tu tag oraz nazwa obiektu (jako łańcuchy znaków).

EntityManager Klasa singleton posiada kolekcję (*std::vector*) wskaźników na obiekty o typie *Entity*. W momencie gdy obiekt dziedziczący po wyżej wymienionej klasie jest tworzony (poza klasą), można zrzutować gona *std::shared_ptr<Entity>* oraz dodać do kolekcji. W pętli gry uruchamiane są metoda *EntityManager::update*, która aktualizuje stan wszystkich aktywnych obiektów w wektorze, oraz metoda *EnityManager::draw*, która rysuje na ekranie aktywne obiekty. Dodatkowo udostępnione zostały metody umożliwiające wyszukiwanie obiektów po tagu, nazwie lub jego numerze ID.

SoundManager Kolejny przedstawiciel singletonu, ściśle współpracujący z *AssetManager*em, używana do utrzymywania dźwięków w programie.

2.4 Zastosowane wzorce projektowe

Singleton Dość kontrowersyjny przykład wzorca projektowego, w projekcie występował głównie w wszelkiej maści managerach, na zmianę z klasami statycznymi.

Builder Wzorzec budowniczego został użyty przy tworzeniu systemów cząsteczkowych, co znacznie uprościło ich tworzenie - zamiast jedenastoparametrowego konstruktora, można wykorzystać wartości domyślne oraz wybrać tylko te właściwości, które nas interesują.

Łańcuch zobowiązań Wykorzystywany do parsowania wejścia w polu tekstowym gry, pozwolił na schludne wybieranie pierwiastków na podstawie żądań użytkownika.

Strategia Szeroko wykorzystywany, pozwolił na uniknięcie instrukcji warunkowych na rzecz większej liczby klas. Przykładem jest obsługa klas dziedziczących po *Entity*.

2.5 Biblioteki

SFML 2.4.2 Biblioteka multimedialna o szerokim możliwościach, umożliwiająca rysowanie i przetwarzanie grafiki, odtwarzanie dźwięków oraz interakcję z systemem oraz użytkownikiem.

NGUI Biblioteka oparta na SFMLu, służąca do obsługi interfejsu graficznego użytkownika. Pomysł z jej dodaniem został odrzucony na etapie projektowania GUI, gdy okazało się, że jej możliwości są niewystarczające do wymagań szaty graficznej.

3 Testy

Program został napisany pod 64-bitową wersją systemu Windows 10 oraz 64-bitową wersją systemu Linux. Testy zostały przeprowadzone na 64-bitowych Windowsach 7, 8 oraz 10 oraz na Linuxach Debian oraz Arch Linux. Ważnym wymaganiem było zapewnienie obecności bibliotek oraz multimediów w folderze z plikiem wykonywalnym. Przekazano program testerom oraz wprowadzono niezbędne poprawki, dotyczące głównie szaty graficznej oraz samej rozgrywki. Te pierwsze okazały się szczególnie cenne dla naszego doświadczenia oraz wiedzy o projektowaniu interfejsów graficznych, drugie pozwoliły uprzyjemnić oraz zbalansować rozgrywkę. Testowano również zużycie pamięci. Program początkowo nie zużywał wiele pamięci operacyjnej, jednakże wprowadzenie grafik, animacji oraz bardziej wyszukanego kroju czcionki znacznie zwiększyło jego zapotrzebowanie na RAM.

4 Możliwości dalszego rozwoju programu

- Dodanie muzyki (przyspieszającej w miarę rozgrywki.).
- Wprowadzenie Ńowej Gry +ź dodatkową fabułą, przeciwnikami oraz mechanikami.
- Stworzenie wersji na telefony działające pod kontrolą systemu Android; interesującym konceptem okazało by się wprowadzenie liczb za pomocą rysowania ich.

5 Źródła multimediów

- Dźwięki - `opengameart.org`, źródła własne
- Grafika - źródła własne

6 Wnioski końcowe

Projekt okazał się sukcesem, zarówno w kwestia spełnienia założeń, jak i efektu końcowego. Nauka wypływająca z niego pozwoli nam lepiej spełniać swoje role jako profesjonalni deweloperzy oprogramowania.