|  |
| --- |
| Projekt nr 25: Generowanie i animacja fraktali |
| Jan Paluch, Maciej Mikołajek, Wiktoria Szewczyk |

# Opis projektu

// skrócony opis czego projekt dotyczy

Głównym celem projektu jest napisanie programu, który wyświetla dwuwymiarowe fraktale. Dodatkowo tworzy animację przekształcając („morfując”) różne fraktale między sobą. Program umożliwia wyświetlanie dowolnych fraktali, jeśli dostarczymy mu dane na temat jego przekształceń afinicznych.

# Założenia wstępne przyjęte w realizacji projektu

// doprecyzowanie założeń podanych w zleceniu + dodatkowe założenia w zleceniu nie ujęte

Jednym z podstawowych założeń projektu jest wczytywanie z pliku parametrów animacji, między innymi rozmiar animacji, liczba klatek oraz liczba fraktali oraz ich parametry (odpowiednie współczynniki przekształceń afinicznych).

Program wyświetla na ekranie fraktale na podstawie dostarczonych danych.

Nasz program powinien sam zadbać o odpowiednie skalowanie rozmiaru fraktali, tak aby mieściły się na ekranie i nie były za małe. Dodatkowo wyświetlane fraktale dostosowują się dynamicznie do zmiany rozmiaru okna.

Kolor pikseli wyświetlanego fraktala powinien zależeć od tego, które przekształcenie zostało użyte do obliczenia jego pozycji.

# Analiza projektu

**Specyfikacja danych wejściowych**

// - specyfikacja danych wejściowych rodzaje, postać i formaty danych wejściowych

Rozmiar animacji: rozmiar x i y bitmap.

Ilość iteracji: ilość punktów, z których złożony jest fraktal.

Ilość fraktali.

Ilość przekształceń afinicznych dla konkretnego fraktala.

Przekształcenie: współczynniki A, B, C, D, E, F przekształcenia afinicznego.

**Oczekiwane dane wyjściowe**

- opis oczekiwanych danych wyjściowych rodzaj, postać i formaty danych wyjściowych

Na ekranie wyświetla się wybrany fraktal, zbudowany z odpowiednio obliczonych punktów, o odpowiednich kolorach. Dodatkowo po kliknięciu w przycisku „Morfuj” widzimy animacje przejścia między kolejnymi fraktalami (jeśli na wejściu podaliśmy dane dla więcej niż jednego fraktala).

**Struktury danych**

- zdefiniowanie struktur danych w jakich strukturach będą przechowywane danych

Zdecydowaliśmy się na stworzenie 4 klas: *Point*, *Trans*, *TransSet* oraz *Instruction*.

Klasa *Point* przechowuje przechowuję punkt w układzie kartezjańskim, razem z jego kolorem wyświetlania.

Klasa *Trans* przechowuję dane na temat jednej transformacji (współczynniki A, B, C, D, E, F) oraz kolor, który będą miały punkty stworzone za jej pomocą. Zajmuje się też obliczaniem punktu na podstawie poprzedniego.

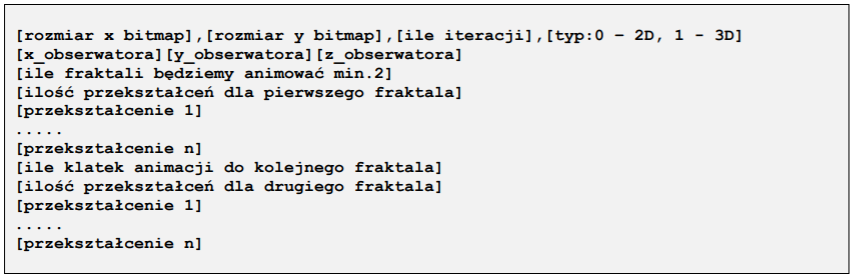
Klasa *TransSet* przechowuję wszystkie transformacje wymagane do wygenerowania jednego fraktala oraz ich ilość. Ponadto zajmuje się wyborem losowej transformacji.

Klasa *Instruction* przechowuje instrukcje generujące fraktale.

**Specyfikacja interfejsu użytkownika**

- specyfikacja interfejsu użytkownika w miarę możliwości szczegółowy opis komunikacji z użytkownikiem

Użytkownik może wprowadzić dane za pomocą wcześniej przygotowanego pliku, którego struktura jest następująca:



Zakładamy że tworzymy tylko fraktale 2D więc dostarczone dane dotyczące pozycji obserwatora są pomijane przez program. Przykładowy plik został dołączony do katalogu BIN.

Użytkownik może za pomocą przycisku „Następny” wybierać, który fraktal chce wyświetlić lub wyświetlić wszystkie w po kolei w postaci animacji klikając przycisk „Morfuj”.

**Wyodrębnienie i zdefiniowanie zadań**

- wyodrębnienie i zdefiniowanie zadań podział całego projektu na mniejsze, jednoznacznie zdefiniowane moduły

Projekt podzieliliśmy na następujące zadania, które kolejno wykonywaliśmy:

* Opracowanie planu struktury programu
* Zbudowanie interfejsu
* Wczytywanie danych z pliku
* Implementacja struktur danych
* Wyświetlanie pojedynczego fraktala
* Animacja przejść między fraktalami
* Skalowanie fraktali

**Wybór narzędzi programistycznych**

- decyzja o wyborze narzędzi programistycznych czego i dlaczego będziemy używać (kompilator, środowisko, biblioteki)

Wybraliśmy środowisko Microsoft Visual Studio ze względu na, że jesteśmy z nim już zaznajomieni oraz na wysokiej jakości narzędzia, które nam ono oferuje.

Używaliśmy języka C++ w standardzie C++14.

Dodatkowo korzystaliśmy z biblioteki wxWidgets, ponieważ umożliwiła nam ona łatwe zaimplementowanie interfejsu oraz wyświetlanie obrazów.

# Podział pracy i analiza czasowa

// kto co będzie robił i jak to będzie rozłożone w czasie

Regularnie spotykaliśmy się, żeby pracować nad projektem razem. Na początku zapoznaliśmy się z celami projektu i opracowaliśmy plan na struktury danych w projekcie. Praca nad projektem została rozłożona równomiernie. Aspekty wczytywania danych z pliku, tworzenia struktur danych lub proces wyświetlania fraktali na ekranie mogliśmy rozdzielić między siebie, jednak trudniejsze zagadnienia jak obliczanie i skalowanie kolejnych punktów lub sposób animacji omawialiśmy i implementowaliśmy wspólnie.

# Opracowanie i opis niezbędnych algorytmów

// opis użytych algorytmów, opracowanie własnych jeśli potrzebne

Przekształcenie afiniczne ma postać:

Pierwszy punkt zawsze jest punktem , a następne obliczane są na podstawie poprzedniego. Parametry do przekształcenia są dla każdego fraktala losowane z odpowiadającego mu zbioru przekształceń.

Parametry te, dostarczane są poprzez plik tekstowy. Plik ten następnie jest deserializowany do struktur danych, dla prostoty wykonywania procedur.

Skalowanie punków fraktala odbywa się za pomocą wbudowanej w wxWidgets procedury skalowania bufora.

Morfizm między dwoma fraktalami zaimplementowany jest jako liniowa transformacja odpowiadających sobie punktów w wyliczonych fraktalach.

# Kodowanie

// opis programu, schemat blokowy xDDDDDDDDD, wykresy przepływu danych, grafy ważniejszych algorytmów, szczegółowy opis klas, funkcji i zmiennych

# Testowanie

// opis procedury testowej, przygotowanie testowych zestawów danych

– testy niezależnych bloków

- testy powiązań bloków

- testy całościowe

- określenie niezmienników (jeśli to możliwe)

// co jeśli dostarczymy dane tylko jednego fraktala

# Wdrożenie, raport i wnioski

// uruchomienie z niezależnymi danymi, raport dotyczący wykonania (co się udało, co nie i dlaczego), co można poprawić lub zmienić w przyszłości