Projekt nr 25
GENEROWANIE I ANIMACJA FRAKTALI

Mikołaj Baczyński
Bartłomiej Długosz
Igor Półchłopek

1. Opis projektu

Głównym zadaniem naszego programu jest stworzenie animacji : generowania poszczególnych fraktali - punkt po punkcie - oraz płynnego przechodzenia (morfowania) fraktali w inne.

Cały projekt został napisany w języku **C++**. Program pozwala użytkownikowi na wybranie interesującego go fraktala, a także wybranie (w dowolnej kolejności) pewnej sekwencji fraktali, które będą się w siebie przekształcać w zadanej liczbie kroków.

2. Założenia wstępne przyjęte w realizacji projektu

W planie mieliśmy stworzenie interfejsu przy pomocy którego będziemy mogli wygenerować wybrany przez użytkownika fraktal z puli zaimplementowanych fraktali oraz stworzyć animację polegającą na przemianie wybranego fraktala w inny.

Wstępnie nie zamierzaliśmy implementować wersji rozszerzonej projektu, który oferowałby generowanie fraktali w 3D. Nie podjęliśmy my się założeń rozszerzonych, więc przygotowaliśmy wersję podstawową, generującą fraktale dwuwymiarowe. W naszym programie została dodana opcja zapisu poszczególnych klatek animacji (tablice bitmap) do pliku.

3. Analiza projektu

a) specyfikacja danych wejściowych:

Naszymi danymi wejściowymi były tak naprawdę tylko ilość iteracji oraz ilość klatek animacji. Wszystkie inne dane wyliczone zostały w środku programu lub były narzuconymi stałymi.

b) opis oczekiwanych danych wyjściowych:

Na ekranie widnieją wybrane przez użytkownika fraktale - w wersji generowania punkt po punkcie, a także animacja przejść wybranych przez użytkownika fraktali.

Program również umożliwia wgląd na poszczególne klatki animacji - generuje serię bitmap - każda z nich to poszczególny krok animacji przejścia.

c) struktura programu:

Zdecydowaliśmy się na implementacji abstrakcji fraktala - klasa czysto wirtualna, której obiektu nie da się stworzyć. Każdy nowy rodzaj fraktala dziedziczył po klasie **Fractal**, zyskując dzięki temu potrzebny interfejs, jak również dostarczał pewnych nowych metod, które są specyficzne dla każdego z nich.

Dzięki temu zabiegowi program jest rozszerzalny - zaimplementowanie nowego typu fraktala jest bardzo wygodne, praktycznie w ogóle nie ma potrzeby refaktoryzacji kodu, wywołanie funkcji rysujących wygląda tak samo. Całość wygląda bardzo przejrzyście, kod jest przyjazny dla programisty oraz jest to sensowne i logiczne podejście do tego problemu.

Wszystko umieściliśmy w interfejsie przyjemnym dla użytkownika, utworzonym za pomocą wxFormBuilder, który zawiera przyciski odpowiadające generowaniu wybranego fractala, utworzenia animacji, pobierania ilości iteracji i kroków przejścia animacji. Zawiera również zapis do pliku, pasek postępu i przycisk czyszczenia ekranu.

d) specyfikacja interfejsu użytkownika:

Użytkownik ma możliwość za pomocą przycisków wygenerować wybrany fraktal klikając przycisk z jego nazwą, przeprowadzić animację przejścia z jednego fraktala w inny poprzez przyciski obok nazw fraktali, wybór ilości iteracji poprzez wpisanie w odpowiednie pole upatrzonej liczby iteracji, ilość kratek przejścia między fraktalami również poprzez wpisanie ilości w odpowiednie pole, zapis do pliku oraz wyczyszczenia ekranu przyciskiem ‘clear’. Interfejs posiada również pasek przedstawiający postęp animacji.

e) wyodrębnienie i zdefiniowanie zadań:

Zadania moglibyśmy podzielić następująco:

- zaplanowanie odpowiedniej struktury programu
- wygenerowanie pojedynczego fraktala:
- utworzenie klasy ‘fractal’, zawierającej metody potrzebne do generowania ogółu fraktali
- dodanie kilku nowych fraktali
- zapisanie kolejnych klatek animacji jaki bitmapę

- stworzenie animacji
- zbudowanie interfejsu użytkownika
- końcowe poprawki

f) decyzja o wyborze narzędzi programistycznych:

Naszym wyborem zostało środowisko Microsoft Visual Studio ze względu na wysoką jakość narzędzi, które ono oferuje oraz byliśmy już z nim zaznajomieni wcześniej.

Do zrealizowania projektu wykorzystaliśmy bibliotekę **wxWidgets.** Próbowaliśmy zastosować bibliotekę **SFML**, lecz program działał wtedy wolniej, a także napotkaliśmy trudność w zaimplementowaniu interfejsu. WxWidgets oferowały również wxFormBuilder, który znacząco ułatwił nam stworzenie interfejsu.

4. Podział pracy i analiza czasowa

Do pracy nad projektem staraliśmy się spotykać, by wspólnymi siłami dojść do jak najlepszej wersji programu. Każdy z nas starał się dołożyć coś od siebie. Jeśli chodzi o poświęcony czas to najdłużej zeszło nam z opracowaniem planu projektu oraz zbudowaniu struktury. Dzięki poświęceniu uwagi na to, mogliśmy zaoszczędzić go na rozwijaniu programu o kolejne opcje.

5. Opracowanie i opis niezbędnych algorytmów

Algorytm powstawania fraktala polegał na wypełnieniu tablicy par punktów poprzez charakterystyczne dla każdego fraktala funkcje wybierane odpowiednio z pewnym prawdopodobieństwem. Następnie wszystkie te punkty były mapowane. Wczytywanie ich stanowiło już proste odczytanie z tablicy.

Algorytm powstawania animacji polegał na na zapisywaniu do vectora bitmap pojedynczych klatek jakie elementy tego vectora. Przy tworzeniu animacji były one po prostu wczytywane. Ilość klatek może się zmieniać i odpowiada ona ilości kroków potrzebnych do przejścia z funkcji oraz prawdopodobieństwa początkowego fraktala do funkcji oraz prawdopodobieństwa fraktala oczekiwanego.

6. Kodowanie

Rozwinięcie tego podpunktu znajduje się w osobnym folderze ‘doc’, w którym są wygenerowane pliki ‘latex’ oraz ‘html’ przy pomocy generatora dokumentacji ‘doxygen’. Cały kod źródłowy został udokumentowany.

7. Testowanie

Jeśli chodzi o testy to takowe przeprowadzaliśmy na bieżąco. Próbowaliśmy generować wszystkie rodzaje fraktali oraz morfować je na każdy możliwy sposób. Sprawdzaliśmy również jakie fraktale i animacje otrzymamy dla zadanych przez nas: liczby iteracji oraz liczby kroków.

8. Wdrożenie, raport i wnioski

Program po uruchomieniu spełnia wszystkie postawione przez nas założenia. Udało się stworzyć siedem różnych fraktali oraz możliwość morfowania każdego z nich w inny wybrany. Ilość iteracji oraz kroków jest poprawnie pobierana. Pasek postępu odpowiednio odzwierciedla status animacji. Bitmapy odpowiadające pojedynczym klatką zapisują się tak jak powinny. W przyszłości moglibyśmy spełnić wymagania rozszerzone projektu i tym samym dodać fraktale 3D oraz spróbować nieco przyspieszyć działanie programu.