Akademia Górniczo – Hutnicza im. Stanisława Staszica w Krakowie

Wydział Fizyki i Informatyki Stosowanej

Informatyka Stosowana

Rok II

Projekt zaliczeniowy z przedmiotu: Podstawy Grafiki Komputerowej

Tytuł projektu: Kalejdoskop 2D
Wykonawcy: Karol Rojek, Mariusz Niwiński, Dariusz Rębisz

1. Opis projektu

Celem projektu było wykonanie programu, który za pomocą wczytanego obrazka symulowałby obrazy powstające w kalejdoskopie oraz zapisywałby efekt jego działania. Projekt został wykonany za pomocą biblioteki wxWidgets w języku C++.

2. Założenia wstępne

W programie pobieramy obraz i wycinamy z niego fragment odpowiadający polu jednego z obszarów, który powstaje poprzez podzielenie obrazka, za pomocą osi. Osie przecinają się w jednym punkcie, będącym środkiem okręgu, w którym symulujemy działanie kalejdoskopu. Użytkownik może wybrać liczbę osi (1-20). Symulacja kalejdoskopu polega na symetrycznym odbiciu elementów względem kolejnych obszarów. Użytkownik ma możliwość obrotu osi, powrotu do obrazka wyjściowego oraz zapisu wyniku działania programu w postaci ciągu bitmap w formacie png.

3. Analiza projektu

Interfejs użytkownika (ramka wraz z przyciskami i suwakami) zostały wykonane za pomocą narzędzia wxFormBuilder. Za pomocą przycisku wczytaj pobieramy obraz w formacie jpg – obraz jest przycinany do kwadratu o rozmiarze określonym rozmiarem ramki. Następnie wycinane z obrazka jest koło w którym symulowane będzie działanie kalejdoskopu. Następnie użytkownik ma możliwość zapisu stworzonego kalejdoskopu w postaci ciągu bitmap w formacie png.

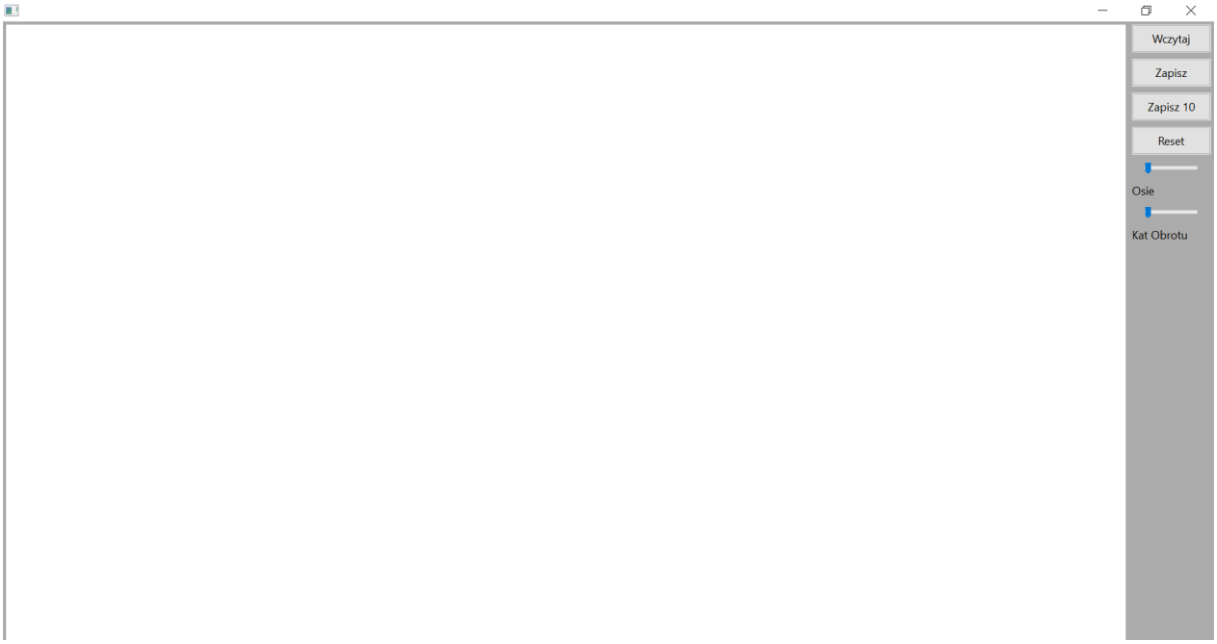
Dane przechowywane są w klasach biblioteki wxWidgets, głównie za pomocą wxBitmap i wxImage, ilość osi jest wczytywana do programu na bieżąco, kąt natomiast jest przechowywany w pamięci jako typ zmiennoprzecinkowy – „double”.

Program komunikuje się z użytkownikiem za pomocą jednowyrazowych napisów, reprezentowanych przez elementy klasy wxFileDialog opisujących działanie przycisków.

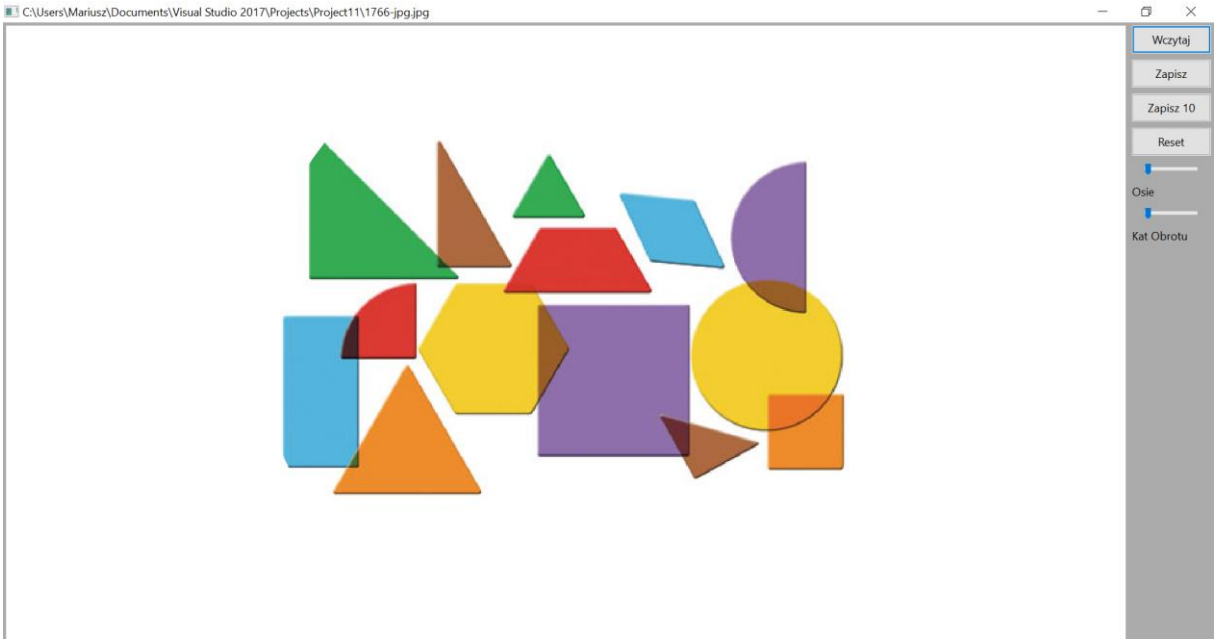
Sama obsługa programu jest podzielona na kilka funkcji, obejmujących takie zagadnienia jak: wczytanie i zapis obrazka, reprezentowanie go jako okręgu, rysowanie kalejdoskopu, interpolacja punktów obrót osi o kąt oraz reset działania programu.

Cały projekt powstał w środowisku Visual Studio 2017, korzystając z standardowej biblioteki C++ (pliki nagłówkowe <iostream> i <cmath>) oraz wxWidgets, która odpowiadała za obsługę środowiska graficznego.

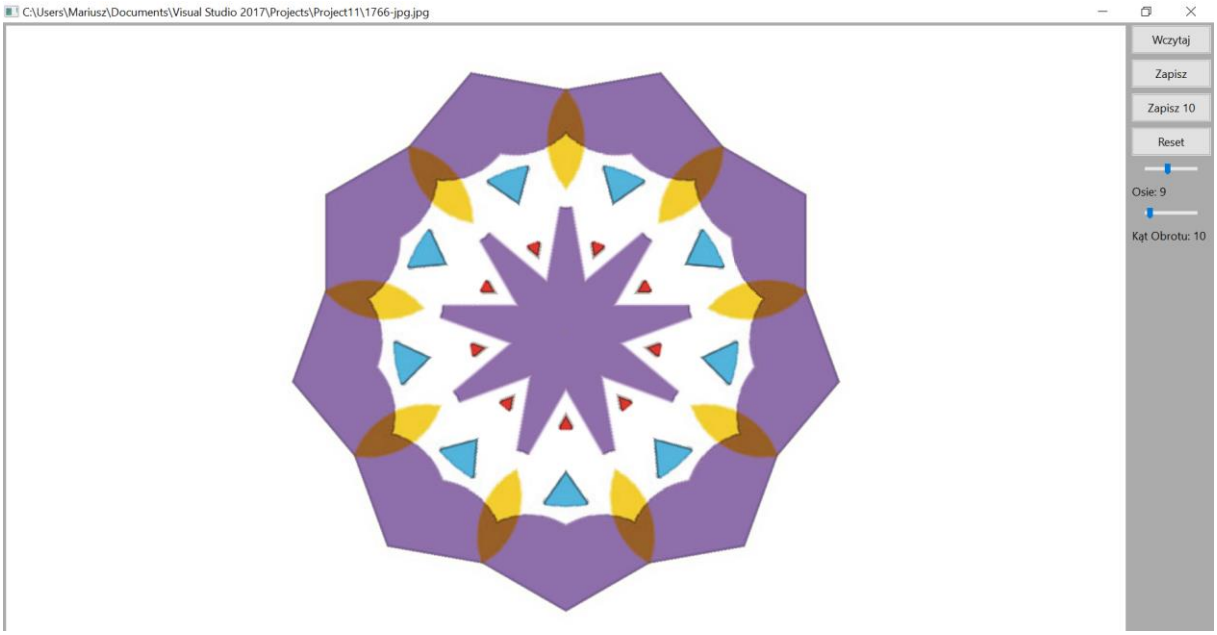
Okno programu:



Okno po wczytaniu obrazka:



Efekt działania:



4. Podział pracy i analiza czasowa.
   Dariusz Rębisz – stworzenie szkieletu interfejsu, obróbka obrazka w programie, funkcje odpowiadające za tworzenie i obrót kalejdoskopu, obliczenia teoretyczne

   Karol Rojek – funkcje wczytujące obrazek, przechowywanie pobranych danych w programie, wybór i implementacja funkcji interpolującej, obliczenia teoretyczne

   Mariusz Niwiński – funkcje odpowiadające za zapis efektów działania programu, dokumentacja programu, optymalizacja działania algorytmów, obliczenia teoretyczne

   Pierwsze etapy pracy – stworzenie interfejsu, struktur danych, zapis i wczytanie obrazka, przebiegały w dniach 28.05.2018 – 30.05.2018. Każdy z członków grupy pracował wtedy oddzielnie.
   Drugi etap pracy – właściwy program oraz udokumentowanie wyników jego działania powstawały podczas wspólnej pracy członków zespołu w dniach 10.06.2018 – 13.06.2017.

5. Opracowanie i opis użytych algorytmów

   Jedynym gotowym algorytmem użytym w naszym programie była interpolacja liniowa, wyliczająca współrzędne R, G, B w przestrzeni kolorów dla danego pixela jako średnią ze współrzędnych R, G i B czterech pixeli, znajdujących się u góry, z dołu i po bokach.

   Główny algorytm symulujący działanie kalejdoskopu bierze uzależniony od ilości osi wycinek i symetrycznie odbija punkty względem kolejnych osi. Same osie nie są zaimplementowane w kodzie – wycinek koła jest przepisywany o zadany kąt.

Algorytm obracający kalejdoskop o kąt znajduje nowy wycinek koła, skąd pobierane są dane, na następnie jest on rysowany podobnie jak w funkcji rysującej kalejdoskop – dzięki czemu obrazek jest uzupełniany nowym wycinkiem.

Pozostałe funkcje odpowiadają za obsługę podstawowych funkcjonalności programu.

6. Opis kodu
   Funkcje publiczne klasy Frame, która odpowiada za obsługę okna i programu

```
virtual void Wczytaj(wxCommandEvent& event);
virtual void Zapisz(wxCommandEvent& event);
virtual void Repaint();
virtual void Osie(wxScrollEvent & event);
virtual void Reset(wxCommandEvent& event);
virtual void Zmiana_Rozmiaru(wxSizeEvent & event);
virtual void Kalejdoskopuj(void);
virtual void Wypelnij();
virtual void Interpoluj(int i, int j, int x, int y);
```

Zmienne prywatne klasy Frame

```
wxBitmap Bitmapa;
wxImage Wycinek;
wxImage Kopia;
int size;
wxFileDialog* File_Dialog;
wxImage Wczytany_Obrazek;
wxImage Przeskalowany_Obrazek;
bool Czy_Jest_Wczytany;
bool Reset_Flaga;
```

Opis funkcji i zmiennych

virtual void Wczytaj(wxCommandEvent& event);
Funkcja wczytująca obrazek.

virtual void Zapisz(wxCommandEvent& event);
Funkcja zapisująca obrazek w wybranym folderze.

virtual void Repaint();
Funkcja rysująca obrazek o zadanym rozmiarze.

virtual void Osie(wxScrollEvent & event);
Funkcja pobierająca liczbę osi i wywołująca rysowanie kalejdoskopu.

virtual void Reset(wxCommandEvent& event);
Funkcja resetująca wynik działania programu.

virtual void Zmiana_Rozmiaru(wxSizeEvent & event);
Funkcja dopasowująca rozmiar obrazka do okna.

virtual void Kalejdoskopuj(void);
Funkcja odpowiadająca za wykonanie kalejdoskopu na zadanym obrazku.

virtual void Wypelnij();
Funkcja wyznaczająca wycinek koła, który będzie poddany obróbce

virtual void Interpoluj(int i, int j, int x, int y);
Funkcja interpolująca kolory pixeli, minimalizująca błędy podczas symulacji kalejdoskopu.

wxBitmap Bitmapa;
Bitmapa rysowana w oknie.

wxImage Wycinek;
Zmienna przechowująca wycinek

wxImage Kopia;
Zmienna w której przechowujemy kopie wycinka.

int size;
Rozmiar obrazka

wxFileDialog* File_Dialog;
Zmienna odpowiadająca za wyświetlanie

wxImage Wczytany_Obrazek;
Zmienna do której wczytujemy obrazek po wczytaniu.

wxImage Przeskalowany_Obrazek;
Zmienna do której wczytujemy obrazek po przeskalowaniu.

bool Czy_Jest_Wczytany;
Zmienna sprawdzająca czy obrazek jest wczytane – pozwala na uniknięcie błędów podczas prób wywołania funkcji, gdy obrazek nie jest wczytany.

bool Reset_Flaga;
Zmienna dla przycisku resetującego

7. Testowanie i wdrożenie

Osobnych testów wymagała przede wszystkim funkcja Wypełnij. Okazało się, że kwestia obrotu osi jest zagadnieniem nietrywialnym. Wymagało to testowania wszystkich jej funkcjonalności oddzielnie od całego programu.

Problemy wystąpiły również podczas implementowania funkcji Kalejdoskopuj i Reset, nie wymagały one jednak oddzielnych testów, jedynie drobnych poprawek w kodzie. Podczas ich pisania okazało się, że potrzebujemy dodatkowej zmiennej przechowującej obrazki.

Problemy pojawiły się również z wydajnością – wymagały od nas zmiany sposobu przechodzenia po pixelach obrazka.

8. Wnioski

Podstawowe wymagania projektu zostały spełnione, jednak część okazała się trudna w implementacji – szczególnie zagadnienia obrotów. Istotna podczas pisania projekty była optymalizacja zagadnień i z teoretyczne rozumienie zagadnienia.