Projekt nr 21: Napisy na zdjęciach

Autorzy:
Karolina Kotłowska (wypisywanie EXIF, zapisywanie IPTC)
Kinga Miszczak (szata graficzna)
Michał Ogorzałek (obsługa plików)

Opis Projektu

Celem projektu było stworzenie przeglądarki do zdjęć, w której będzie istniała możliwość odczytania danych EXIF i IPTC, zapisanych w pliku ze zdjęciem cyfrowym. Po wybraniu folderu ze zdjęciami, po lewej stronie wyświetlają się miniaturki zdjęć. Po pojedynczym kliknięciu na wybrane zdjęcie, wyświetlają się dane IPTC i EXIF, a po dwukrotnym kliknięciu, zdjęcie wyświetla się w powiększeniu. Ponadto dane IPTC można będzie dowolnie edytować, zapisywać i czyścić. Dodatkowym atutem naszego programu jest możliwość zapisywania zdjęć z wypisywanymi na nich danymi EXIF dla wybranego zdjęcia lub w sposób zautomatyzowany dla wszystkich zdjęć. Dostępna jest również opcja zapisu kilku podstawowych informacji

Założenia wstępne przyjęte w realizacji projektu

Program, po wczytaniu folderu, wyświetla tylko pliki z rozszerzeniem .jpg. Minimalna wielkość okna ustawiona jako 900x675 pikseli.

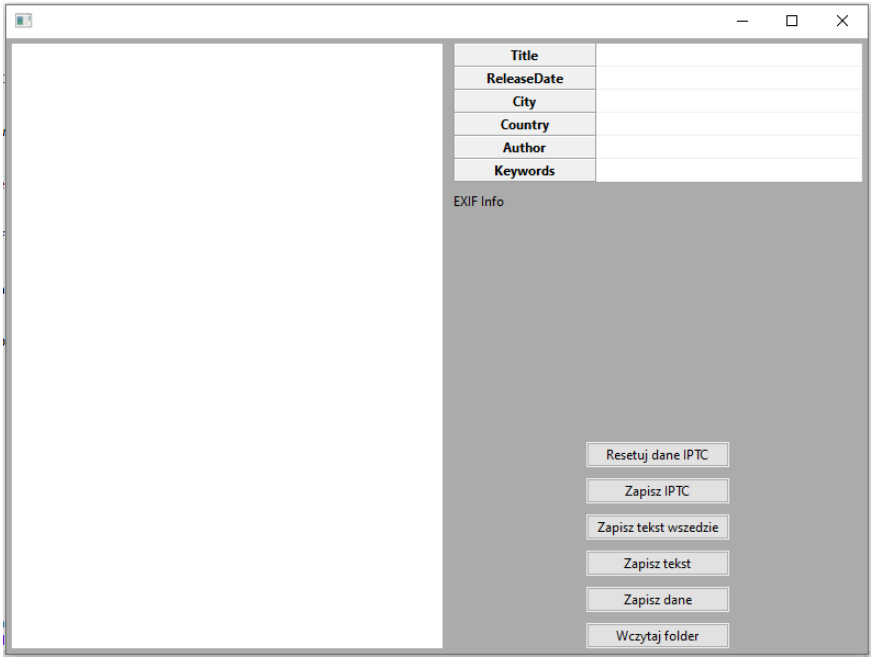
Pomysł na wyświetlanie zdjęć zrealizowaliśmy następująco. Zdjęcia wczytujemy z wybranego przez użytkownika folderu, skalujemy je (w celu ograniczenia zużycia pamięci) – zapisujemy je jako przeskalowane bitmapy. Wczytywanie za każdym razem zdjęć z dysku obciążyło by znacząco nasz program. Następnie na podstawie przeskalowanych bitmap tworzymy „przyciski" – `BitmapButtony` – które dodajemy do `FlexGridSizera`. Obsługa pojedynczego kliknięcia na zdjęcie jest prosta – stworzenie klasy `MyButton` dziedziczącej po `wxBitmapButton`, umożliwiło obsługę takiego eventu. Zachowanie ścieżki do aktualnie wybranego zdjęcia pozwala na wyświetlenie odpowiednich danych EXIF oraz IPTC. Natomiast obsługa podwójnego kliknięcia na zdjęcie działa na zasadzie chowania się i pokazywania dwóch paneli. Jeden panel – `wxPanel parent` odpowiedzialny za wyświetlanie `FlexGridSizera` jest chowany w momencie podwójnego kliknięcia na miniaturkę, natomiast drugi panel `wxPanel display` z dużym zdjęciem, w tym momencie się pokazuje. Po ponownym „double-clicku" na powiększone zdjęcie – mechanizm działa odwrotnie.

Analiza projektu

Aby program działał zgodnie z oczekiwaniami, należy wybrać folder, w którym znajdują się pliki z rozszerzeniem .jpg. Jeżeli w folderze znajdują się pliki z innymi rozszerzeniami, zostaną one zignorowane i nie wyświetlą się ich miniaturki.

Podczas działania programu, użytkownik może wygenerować plik lub pliki (w zależności od kliknięty przycisku) o rozszerzeniu .jpg. Obrazki będą zawierały oryginalne zdjęcie wraz z ich danymi EXIF, wypisanymi w lewym górnym rogu. Możliwym jest również wygenerowanie pliku .txt, zawierającego ścieżki do wszystkich wyświetlonych miniaturek wraz z ich rozmiarami. Podczas działania programu możliwe jest również dowolne zmienianie danych IPTC obrazka.

Po uruchomieniu programu, pojawia się interfejs użytkownika.



Rysunek1: Interfejs użytkownika

Składa się on z 6 przycisków, Grida i StaticTextu.

Przyciski służą do kontroli poszczególnych funkcji programu:

- Resetuj dane IPTC – służy do usuwania danych IPTC z aktualnie wybranego zdjęcia
- Zapisz IPTC – służy do zapisu edytowanych danych IPTC
- Zapisz tekst wszędzie – służy do zapisywania danych EXIF na wszystkich zdjęciach w folderze
- Zapisz tekst – służy do zapisu danych EXIF na wybranym zdjęciu
- Zapisz dane – zapisuje w pliku txt ścieżki do otworzonych w programie zdjęć oraz informacje EXIF na temat każdego ze zdjęć
- Wczytaj folder – służy do wyboru folderu z którego są wczytywane zdjęcia do programu

Grid służy do wypisywania danych IPTC. Istnieje możliwość ich edytowania, jednak po wpisaniu nowych danych, żeby je zapisać należy wcisnąć przycisk „Zapisz IPTC”.

StaticText służy do wypisywania danych EXIF. Wypisuje się ich tyle, ile jest zapisanych w pliku.

Program rozłożyliśmy na następujące problemy:

- Wyświetlenie zdjęć z katalogu
- Wyświetlanie IPTC po jednokrotnym kliknięciu na zdjęcie
- Wyświetlanie dużego zdjęcia po dwukrotnym kliknięciu
- Wyświetlanie EXIF
- Zapisywanie danych na zdjęcia lub zdjęcie
- Edytowanie IPTC
- Resetowanie IPTC
- Zapisywanie informacji o zdjęciach do pliku

Podczas pisania programu, do budowy interfejsu użyliśmy programu **wxFormBuilder,** ułatwiło to nam zaplanowanie wyglądu całej aplikacji. Dodatkowo jest on bardzo intuicyjny i w szybki sposób mogliśmy rozpocząć budowę całego projektu. Całość programowaliśmy przy pomocy biblioteki **wxWidgets** oraz **FreeImage.** Obie biblioteko pozwoliły nam zoptymalizować i usprawnić naszą pracę, ich obsługę znaliśmy z laboratoriów. Wykorzystanie biblioteki FreeImage było kluczowe w naszym projekcie, ponieważ mogliśmy w stosunkowo prosty sposób „dostać się" do danych IPTC oraz EXIF. Wszyscy używaliśmy środowiska programistycznego **VisualStudio 2019**. Używaliśmy go na laboratoriach, co zdecydowanie usprawniło naszą pracę pod względem czasowym, ponieważ mieliśmy już podstawową wiedzę na temat obsługi tego środowiska. Ustaliśmy, że każdy z nas będzie odpowiedzialny za inną część programu. W celu łączenia naszych części posługiwaliśmy się platformą **Github**, gdzie każdy z nas mógł edytować stworzone w tym celu repozytorium. Dzięki temu praca szła o wiele sprawniej i każdy z nas mógł w każdej chwili ściągnąć najnowszą wersję programu oraz zgłaszać ewentualne uwagi.

Podział pracy i analiza czasowa

Cały projekt tworzyliśmy etapami, spotykając się w celu omówienia postępów pracy i podziału kolejnych zadań. Projekt tworzyliśmy około 3 tygodnie. Podział pracy wyglądał następująco:

I tydzień
Tworzenie koncepcji, dyskusja nad warunkami wstępnymi, zdefiniowanie problemów, obranie optymalnej ścieżki realizacji
Karolina – analiza plików pod względem danych EXIF
Kinga – stworzenie szaty graficznej programu
Michał – wczytywanie folderu i wypisywanie miniaturek zdjęć

II tydzień
Karolina – wypisywanie danych EXIF oraz analiza plików pod względem danych IPTC
Kinga – powiększanie zdjęcia po dwukrotnym kliknięciu na miniaturkę
Michał – wykonanie dynamicznie zwiększającego się FlexGridu, reakcje programu na zmianę wielkości okna głównego

III tydzień
Karolina – wypisywanie, zapisywanie, resetowanie danych IPTC
Kinga – wstępne przygotowanie dokumentacji i optymalizacja kodu
Michał – zapisywanie danych na zdjęciu/zdjęciach i zapisywanie danych w pliku tekstowym

Czwarty tydzień poświęciliśmy na wspólne testowanie i dokończenie dokumentacji.

Struktury danych

Klasa GUIMyFrame1 odpowiedzialna jest za kontrole nad programem i przechowywanie ogólnych danych programu. Z ważniejszych dla działania prgoramu przechowywanych danych należy wymienić:
Podczas odczytywania zdjęć z folderu, ścieżki do każdego zaakceptowanego pliku przechowywane zostają w tablicy typu wxArrayString o nazwie path_array. Wszystkie bitmapy stworzone z obrazów przechowywane zostają w wektorze biblioteki standardowej std::vector<wxBitmap> bitmapVector.
Powstałe z bitmap obiekty klasy MyButton przechowywane są za pomocą obiektu wxFlexGridSizer.
int window_width - ostatni zapisany rozmiar szerokości okna
int window_height - ostatni zapisany rozmiar wysokości okna
size_t file_count - ilość zaakceptowanych przez program plików
static Panel2* currentFullDisplay – obiekt Panel2 aktualnie pokazujący zdjęcie w większym rozmiarze

static MyButton* currentPic - aktualnie wybrany obiekt MyButton
wxBoxSizer* masterSizer – sizer odpowiedzialny za odpowiednie ustawienie wxFlexGridSizer na ekranie

W klasie MyButton, dziedziczącej po klasie wxBitmapButton przechowywane zostają dane potrzebne do wypisu / dostępu do danych EXIF i IPTC oraz zainicjowania zamiany wyświetlanego panelu. Dane przechowywane w klasie:
wxString path - ścieżka do obrazu przechowywanego w klasie
wxPanel* parent1 – aktualny panel
wxGrid* EXIF1 - dostęp do wyświetlania danych EXIF
wxGrid* IPTC1 - dostęp do wyświetlania danych IPTC
wxFlexGridSizer* fgSizer – sizer używany w programie
wxPanel* DisplayPanel – drugi panel do wyświetlania całości obrazów
wxStaticText* text – miejsce tymczasowego przechowywania danych EXIF.

Klasa Panel2 dziedzicząca po wxPanel służy do zmiany wyświetlanego panelu przy potrzebie wyświetlenia zdjęcia. Przechowuje:
wxPanel* _parent – panel główny
wxString _path - ścieżka do zdjęcia które ma zostać wyświetlone
wxPanel* _DisplayPanel – panel na którym zdjęcie zostanie wyświetlone

Opracowanie i opis niezbędnych algorytmów

- Algorytm odpowiedzialny za zmianę wielkości paneli przy zmianie wielkości okna programu:

Podczas nastąpienia wxUpdateUIEvent:
If(pokazany jest główny panel)

    If (wcześniejszy rozmiar tego panelu jest inny niż aktualny)

        ustaw wcześniejszy rozmiar na aktualny

        aktualizuj aktualny panel (printBitmapButtons())

Else if (pokazywany jest panel ze zdjęciem)

    If (wcześniejszy rozmiar tego panelu jest inny niż aktualny)

        ustaw wcześniejszy rozmiar na aktualny

        aktualizuj aktualny panel (currentFullDisplay->PaintFD())

- Algorytm wczytywania zdjęć:

Ustawiane są wartości związane z wczytywaniem plików na wartości domyślne.
Otwierane jest okno wyboru folderu z plikami i pobierana jest do niego ścieżka.
If (wybieranie folderu się powiodło)

W tablicy path_array zapisywane są ścieżki do wszystkich typu .jpg.
Zliczana jest ilość wczytanych ścieżek.
wywoływana jest funkcja odpowiedzialna za konwersję zdjęć do bitmap.

Else

Wywoływane jest okno komunikatu z informacją o błędzie otwierania folderu.

- Algorytm tworzenia bitmap

Podczas działania algorytmu na ekranie wyświetlana jest wiadomość "LOADING IMAGES..."
If(wczytana ilość ścieżek jest większa niż 0)

Dla każdej ze ścieżek:

Tworzony jest obraz
Obraz jest skalowany do rozmiaru 240x180
Na podstawie obrazu tworzona jest bitmapa
Bitmapa zostaje przechowana w wektorze bitmap bitmapVector.

Ekran z napisem jest czyszczony
wywoływana jest funkcja odpowiedzialna za wstawienie bitmap do wxFlexGridSizer

- Algorytm ustawienia wxFlexGridSizer i wczytania plików

Do obliczenia dynamicznego rozłożenia FlexGridu , pod warunkiem ,że plików jest więcej niż 0 i zakładając, że zmienne window_width i file_count mają odpowiednie wartości dla aktualnego stanu programu, użyto:

window_width - szerokość panelu
file_count - ilość plików rozpoznana przez program
SzerokośćObrazu = 240

Ilość kolumn: cols = (window_width - 35) / SzerokośćObrazu
window_width -35 jest wzięciem pod uwagę szerokości suwaków

Ilość wierszy: rows = file_count / cols + 1

Tworzony jest nowy obiekt wxFlexGridSizer o nazwie fgSizer1 z ilością wierszy i kolumn wyliczoną powyżej

Tworzony jest nowy obiekt wxBoxSizer o nazwie masterSizer, dodany jest do niego fgSizer1 i ustawiany jest odstęp od lewej krawędzi panelu według wzoru:

(window_width - 35 - 10*(cols-1) - cols* SzerokośćObrazu)/2

10*(cols-1) jest wzięciem pod uwagę szerokości odstępu pomiędzy plikami

Dla każdego z plików:

Tworzony jest nowy obiekt klasu MyButton
obiekt umieszczany jest w obiekcie fgSizer1

Główny sizer programu ustawiany jest na masterSizer.

Kodowanie

Klasy:

- GUIMyFrame1
- MyButton
- Panel2
- PanelBox

Metody klasy GUIMyFrame1:
`void WriteInIPTCData (wxCommandEvent& event)` – metoda wpisująca nowe, zmienione dane IPTC do zdjęcia, jeśli chcemy nasze zdjęcie uzupełnić o tytuł, datę powstania zdjęcia, miejsce w którym zdjęcie zostało zrobione, miasto, kraj czy słowa – klucze, możemy to zrobić wpisując w tabelę IPTC nowe dane, a następnie kliknąć w przycisk „Zapisz IPTC”
`void IPTCReset(wxCommandEvent& event)` – metoda obsługująca przycisk Resetuj IPTC, zerująca dane IPTC dla zdjęcia, na które wcześniej kliknęliśmy
`void DisplayMetaData(wxGrid* EXIF,wxStaticText* m_textCtrl1, wxGrid* IPTC, wxPanel* parent, wxPanel* display, wxString path)` – wypisująca dane EXIF w panelu po prawej stronie

`void LoadImgOnClick(wxCommandEvent& event)` - metoda reagująca na przycisk “Wczytaj folder”. Pobiera ścieżki do plików z roszerzeniem .jpg ze wskazanego folderu, zapisuje je w tablicy path_array, wywołuje metodę loadBitmaps(). W przypadku niepowodzenia informuje użytkowanika wyświetlając wiadomość "Cannot open a directory".

`void loadBitmaps()` - Dla każdej ścieżki z path_array tworzy obraz wskazywany przez ścieżkę, skaluje obraz do rozmiaru (240, 180), z tak przeskalowanego obrazu tworzy bitmapę i zapisuję ją w wektorze bitmap bitmapVector. Wywołuje metodę printBitmapButtons().

`void printBitmapButtons()` - Zwracając uwagę na ilość bitmap oraz ich wielkość, tworzy nowy obiekt FlexGrindSizer o nazwie fgSizer1 i odpowiednio ustawia ilość kolumn i wierszy. Dla każdego elementu bitmapVector tworzy odpowiedni dla niego obiekt klasy MyButton po czym dodaje go do stworzonego wcześniej obiektu fgSizer1. FlexGridSizer jest dynamicznie ustawiany na środek panelu poprzez masterSizer. Ostatecznie sizer panelu m_panel1 jest ustawiany na masterSizer. Na tej funkcji kończy się ładowanie plików do programu. Służy ona również do odpowiedniego ułożenia plików w programie w przypadku zmiany rozmiaru okna głównego.

`void window_update(wxUpdateUIEvent& event)` - metoda reagująca na event zmiany stanu GUI programu. Wykrywa który z paneli jest aktualnie pokazywany, obsługuje dostosowanie wielkości i rozłożenia elementów obu panelów w przypadku zmiany rozmiaru okna głównego.

`void WriteDataOnPic(wxCommandEvent& event)` - metoda reagująca na przycisk “Zapisz tekst”. Tworzy kopię dla aktualnie wybranego przez użytkownika pliku, zapisuje na niej w lewym górnym rogu 5 pierwszych informacji EXIF posiadanych przez plik oryginalny. Zapisuje stworzoną kopię w wybranym przez użytkownika folderze z wybraną przez użytkownika nazwą z rozszerzeniem .jpg.

`void GenerateTextOnAll(wxCommandEvent& event)` - zachowanie analogiczne do metody WriteDataOnPIc() z rozróżnieniem stworzenia kopii i zapisu danych na wszystkich wczytanych przez program plikach. Nazwy plików ustawiane są jako nazwa wybrana przez użytkowanika z dodatkiem numeru pliku liczonego od 0.

`wxString getDataOnBitmap(wxString path)` - metoda znajdująca 5 pierwszych informacji EXIF danego przez ścieżkę pliku, zwracająca wxString z danymi informacjami.

void WriteDataToFile(wxCommandEvent& event) - metoda reagująca na przycisk "Zapisz dane". Tworzy w wybranym prze użytkownika folderze dokument tekstowy ze ścieżkami i 5 pierwszymi informacjami EXIF dla każdego wczytanego przez program pliku.

Metody klasy MyButton:
`void OnMouseLeftDown(wxMouseEvent& event)` – metoda obsługująca pojedynczy klik na miniaturkę zdjęcia
`void OnMouseDoubleDown(wxMouseEvent& event)` - metoda obsługująca podwójny klik na miniaturkę zdjęcia
`wxString GetPathFromClick()` – metoda zwracająca ścieżkę do zdjęcia, które aktualnie jest wybrane przez użytkownika

Metody klasy Panel2:
`wxString getPath()` – metoda zwracająca ścieżkę do pliku
`void OnPanelDoubleDown(wxMouseEvent& event)` – metoda obsługująca podwójny klik na zdjęcie
`void PaintFD()` – metoda wyświetlająca zdjęcie przeskalowane do wielkości panelu, na którym się wyświetla. Jeżeli zdjęcie jest mniejsze niż wielkość panelu, wyświetla się w oryginalnych rozmiarach.

Testowanie

Każda osoba z grupy realizowała swoją część projektu. Na etapie dyskusji nad architekturą naszego programu, osobno implementowaliśmy i testowaliśmy swoje funkcje. Dopiero po sprawdzeniu czy części programu działają, linkowaliśmy cały projekt.

Elementy które wymagały sprawdzenia:

- Skalowalność okna
- Skalowalność FlexGrida wypełnionego zdjęciami
- Czas ładowania zdjęć z folderu

Test każdego z elementów przeszedł poprawnie, więc mogliśmy uznać projekt za gotowy.

Wdrożenie, raport i wnioski

Pomyślnie zrealizowaliśmy podstawowe wymagania. Dodatkowo wzbogaciliśmy projekt o funkcje, które mogą być użyteczne dla użytkownika. W naszym programie istnieje opcja zapisywania nowych danych IPTC, resetowania danych IPTC, zapisywanie danych EXIF na kopii wybranego zdjęcia lub na kopiach wszystkich wczytanych zdjęciach, generowanie pliku .txt, zawierającego ścieżki do wszystkich wyświetlonych miniaturek wraz z ich wybranymi danymi EXIF.

Problemem okazało się zbyt wolne ładowanie zdjęć, gdy w katalogu, z którego pobieraliśmy zdjęcia mieściło się powyżej 30 zdjęć bardzo dobrej jakości (12MB). Natomiast nie znaleźliśmy bardziej efektywnej metody wczytywania zdjęć.