FB KINECT PAINTER

Dokumentacja projektu

Autorzy:
Wojciech Bełka
Łukasz Florczak

Spis treści

Wstęp **4**
- Opis aplikacji 4
- Dlaczego wybrano temat? 4
- Podobne aplikacje 4

Cel i zakres prac **5**
- Zadania aplikacji 5
- Podział Prac 5

Funkcjonalność **6**
- Zapisywanie 6
- Wczytywanie 6
- Malowanie 7
  - Wybór kolorów 7
  - Wybór narzędzia 7
  - Zmiana rozmiaru narzędzia 8
  - Cofanie ruchów 8

Wykorzystane technologie i urządzenia **9**
- WPF 9
- UWP 9
- Kinect for Windows SDK 9
- AdDuplex 10
- Sensor Kinect 360 10

Środowisko i narzędzia **11**
- Środowisko 11
- Narzędzia 11

Model Aplikacji **12**
- Architektura 12
- Struktury danych 12

Problemy (w tym rozwiązania) **13**
- Moduł RTB 13
- Nazewnictwo plików i rozszerzenie fbkp 14
- Problemy z wyświetlaniem interfejsu 14
- Problemy z przechwytywaniem ruchów 14

Instrukcja **16**
- Instalacja 16
- Obsługa 16

Podsumowanie **19**
Możliwości dalszego rozwoju aplikacji 19
Cele zrealizowane 19
Cele niezrealizowane 19
Bibliografia **20**

1. Wstęp

1.1. Opis aplikacji

Aplikacja FB Kinect Painter to uproszczona wersja programów typu MS Paint, która do swojego działania wymaga urządzenia Kinect. Zadaniem aplikacji jest malowanie obrazów przy użyciu ruchów dłoni, lewej lub prawej w zależności od predyspozycji danego użytkownika. Dodatkowymi funkcjonalnościami są wybór narzędzia, zmiana koloru narzędzia, grubość narzędzia oraz wczytywanie i zapisywanie obrazu.

1.2. Dlaczego wybrano temat?

Głównym powodem było zainteresowanie techniką rozpoznawania ruchów przy wykorzystaniu sensora Kinect przy wykorzystaniu bibliotek dostarczonych wraz z Kinect for Windows Developer Toolkit w wersji 1.8. Dodatkowym bodźcem stała się chęć podniesienia swoich umiejętności programowania w języku C# z wykorzystaniem silnika graficznego WPF (ang. Windows Presentation Foundation), z którym zapoznaliśmy się na studiach.

1.3. Podobne aplikacje

W Internecie istnieje niewiele tego typu aplikacji wykorzystujących urządzenie Kinect, prawdopodobnie dlatego, że urządzenie Kinect jest mało popularne wśród użytkowników PC. Najpopularniejszą tego typu aplikacją jest Kinect Paint[1] z 2012 roku, która jest oprogramowaniem typu OpenSource.

[1] *https://paint.codeplex.com/*

2. Cel i zakres prac

2.1. Zadania aplikacji

- malowanie obrazów - każdy użytkownik, by rozpocząć malowanie musi podnieść rękę powyżej łokcia, która nie jest używana do malowania.

- tworzenie nowego obrazu - każdy użytkownik ma możliwość utworzenia nowego obrazu, klikając w odpowiedni przycisk w oknie głównym aplikacji, następnie musi potwierdzić swoją decyzję.

- zapisywanie obrazu - utworzony obraz można zapisać, klikając w odpowiedni przycisk w oknie głównym aplikacji.

- wczytywanie obrazu - obrazy zapisane w formacie przewidzianym dla naszej aplikacji możemy wczytać z folderu, który użytkownik wskazał przy konfiguracji aplikacji.

- wybór narzędzia - użytkownik ma do wyboru pięć narzędzi, które opisane zostaną w rozdziale 3.

- zmiana grubości narzędzia - w celu zwiększenia lub zmniejszenia wielkości narzędzia, użytkownik musi kliknąć w odpowiednie przyciski.

- zmiana koloru - użytkownik ma do wyboru paletę barw składającą się z 22 kolorów.

2.2. Podział Prac

Wojciech Bełka

- Zaawansowana obróbka ruchów dłoni
- Logika aplikacji
- Moduł RTB

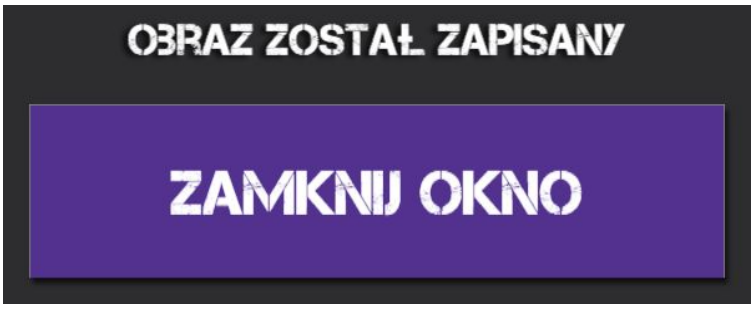
Łukasz Florczak

- Nawiązanie połączenia z urządzeniem Kinect
- Interfejs aplikacji
- Wstępne przechwytywanie ruchów dłoni
- Moduł RTB

3. Funkcjonalność

3.1. Zapisywanie

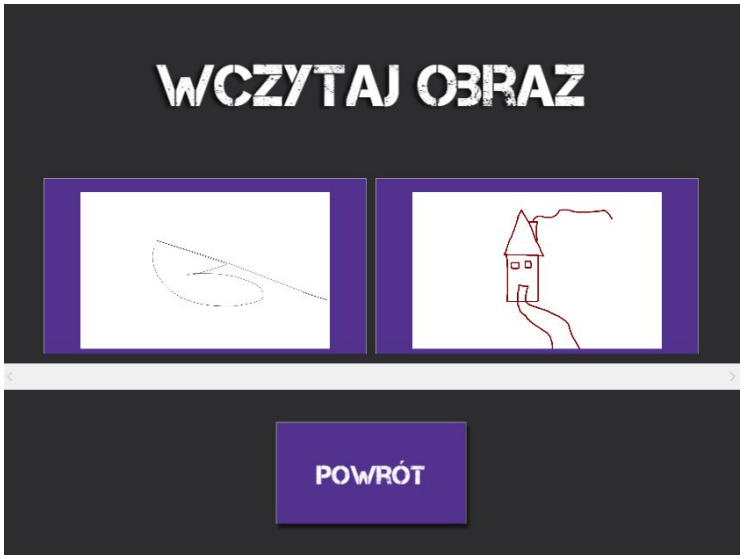
Użytkownik może zapisać efekty swojej pracy klikając w przycisk zapisz. Po pomyślnym zapisaniu pojawi się okno (Rys 1). Opis realizacji zapisu (od środka) został opisany w podrozdziale 7.2.



Rys 1. Okno pomyślnego zapisu.

3.2. Wczytywanie

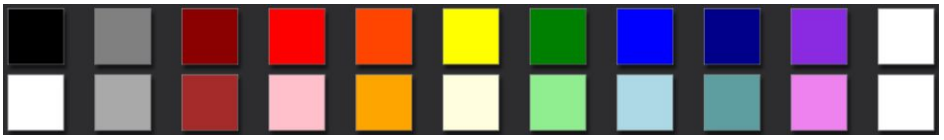
Użytkownik może wczytać zapisane przez siebie obrazy. Wczytywanie ogranicza się tylko do wczytywania obrazów utworzonych w naszej aplikacji, obrazy te posiadają specjalne rozszerzenie, o którym mowa w rozdziale 7.2.



Rys 2. Okno wczytywania.

3.3. Malowanie

3.3.1. Wybór kolorów



Rys 3. Paleta kolorów.

Użytkownik może dowolnie zmieniać kolor narzędzi (spray, pędzel i ołówek), na jeden z 22 kolorów. Paleta kolorów (Rys 3.) umieszczona jest w dolnej części interfejsu użytkownika.

3.3.2. Wybór narzędzia

Rys 4. Menu narzędzi.

Użytkownik może przełączać się między dostępnymi narzędziami (spray, ołówek, gumka, pędzel, zaznaczenie/wycinanie) dostępnymi pod przyciskami w menu w lewej części interfejsu użytkownika (Rys 4).

3.3.3. Zmiana rozmiaru narzędzia

Rys 5. Panel zmiany rozmiaru narzędzia.

Użytkownik może zmieniać rozmiar trzech narzędzi (spray, gumka, pędzel), przy użyciu panelu (rys 5.) dostępnym, w lewym menu interfejsu użytkownika. Rozmiar przyjmuje wartości z zakresu od 1 do 30. W przypadku pędzla i gumki wartość ta jest mnożona przez 4, w ten sposób narzędzie otrzymuje faktyczny rozmiar w pikselach XxX. Natomiast w przypadku narzędzia spray rozmiar narzędzia wpływa na wielkość promienia w jakim “rozpylana” jest farba (promień w pikselach jest równy ilorazowi parametru rozmiar narzędzia oraz liczby 4).

3.3.4. Cofanie ruchów

Rys 6. Przycisk cofania ruchu.

Rys 7. Przycisk przywracania ruchu.

Użytkownik może cofnąć każdy swój ruch, za pomocą przycisku (Rys 6) dostępnego w lewym menu interfejsu użytkownika. Każdy cofnięty ruch może zostać przywrócony po kliknięciu w przycisk (Rys 7).

4. Wykorzystane technologie i urządzenia

4.1. WPF

Windows Presentation Foundation - silnik graficzny i API bazującego na .NET3[2]. API w WPF opiera się na języku XML, dokładniej na jego implementacji o nazwie XAML. Graficzna część GUI wykorzystuje grafikę wektorową, budowaną z użyciem akceleratorów grafiki 3D i efektów graficznych udostępnianych przez WGF[3].

4.2. UWP

Universal Windows Platform - uniwersalna platforma systemu Windows, której głównym założeniem jest opracowywanie uniwersalnych aplikacji, które działają w systemach Windows 10 i Windows 10 Mobile. API zostało zaimplementowane w języku C++, natomiast obsługuje takie języki jak C++, Visual Basic, C#, F# i JavaScript.

4.3. Kinect for Windows SDK

Kinect for Windows SDK pozwala na obsługę Kinecta z poziomu systemu Windows. Pierwsza wersja ukazała się w 2011 roku, jego użycie wymaga spełnienia poniższych wymagań:

- system Windows 7 lub nowszy,
- procesor posiadający co najmniej 2 rdzenie z taktowaniem co najmniej 2.66GHz,
- minimum 2GB RAM,
- karta graficzna zgodna z Directx 9.0c.

Powyższy zestaw deweloperski umożliwia w łatwy sposób wykorzystać czujniki zainstalowane w urządzeniu Kinect i dostarcza deweloperowi między innymi:

- śledzenie w trybie szkieletu,
- możliwość śledzenia twarzy,
- nieprzetworzony strumień danych,
- funkcje audio,
- skanowanie pomieszczeń.

Ponadto Microsoft udostępnia bardzo bogatą dokumentację, która opisuje każdą dostępną funkcję w ramach SDK.

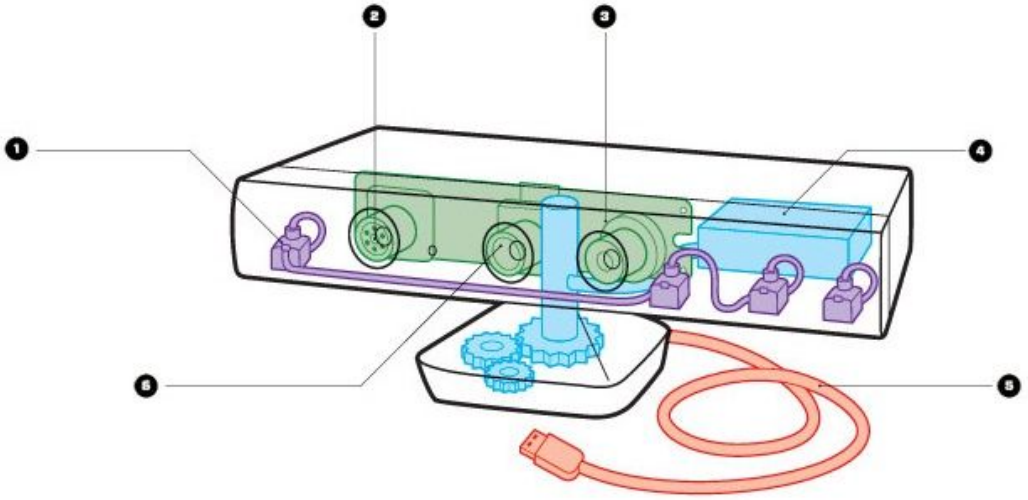
[2] *Platforma programistyczna opracowana przez Microsoft*
[3] *ang. Windows Graphics Foundation*

4.4. AdDuplex

AdDuplex - jedna z platform SSP (Sell Side Platform) modelu RTB (Real Time Bidding) odpowiedzialną za licytacje reklamy pod dostępne miejsce w czasie rzeczywistym. Platforma ta została wybrana ponieważ jako jedyna spośród wielu wspiera aplikacje dla systemu Windows.

4.5. Sensor Kinect 360

Kinect - czujnik ruchu dla konsoli Xbox360 wyprodukowane przez Microsoft. Urządzenie pozwala użytkownikowi na interakcję z konsolą bez konieczności używania kontrolery, natomiast przy pomocy ruchów ciała, a także głosu.



Rys 8. Budowa sensora Kinect.[4]

1. Zestaw mikrofonów.
2. Emiter podczerwieni.
3. Kamera głębokości.
4. Automatyczny kontroler nachylenia.
5. Kabel połączeniowy.
6. Kolorowa kamera RGB.

[4] Źródło: https://www.wired.com/wp-content/uploads/blogs/magazine/wp-content/images/19-07/mf_kinect2_f.jpg

5. Środowisko i narzędzia

5.1. Środowisko

Visual Studio 2015 Enterprise - zintegrowane środowisko programistyczne firmy Microsoft, umożliwiające tworzenie dużych projektów oraz aplikacji w łatwy i przyjemny dla programisty sposób. Dużym atutem tego oprogramowania jest integracja z systemami kontroli wersji takimi jak np. GitHub.

5.2. Narzędzia

GitHub - ogólnodostępny zdalny system kontroli wersji. Narzędzie to umożliwiło nam wspólną, zdalną pracę nad projektem. Wykonanie niezależnych od siebie modułów i następne ich scalenie w jedną aplikację. Dodatkowym atutem tego serwisu jest możliwość udostępnienia projektu publicznie lub wybranym osobom do wglądu.

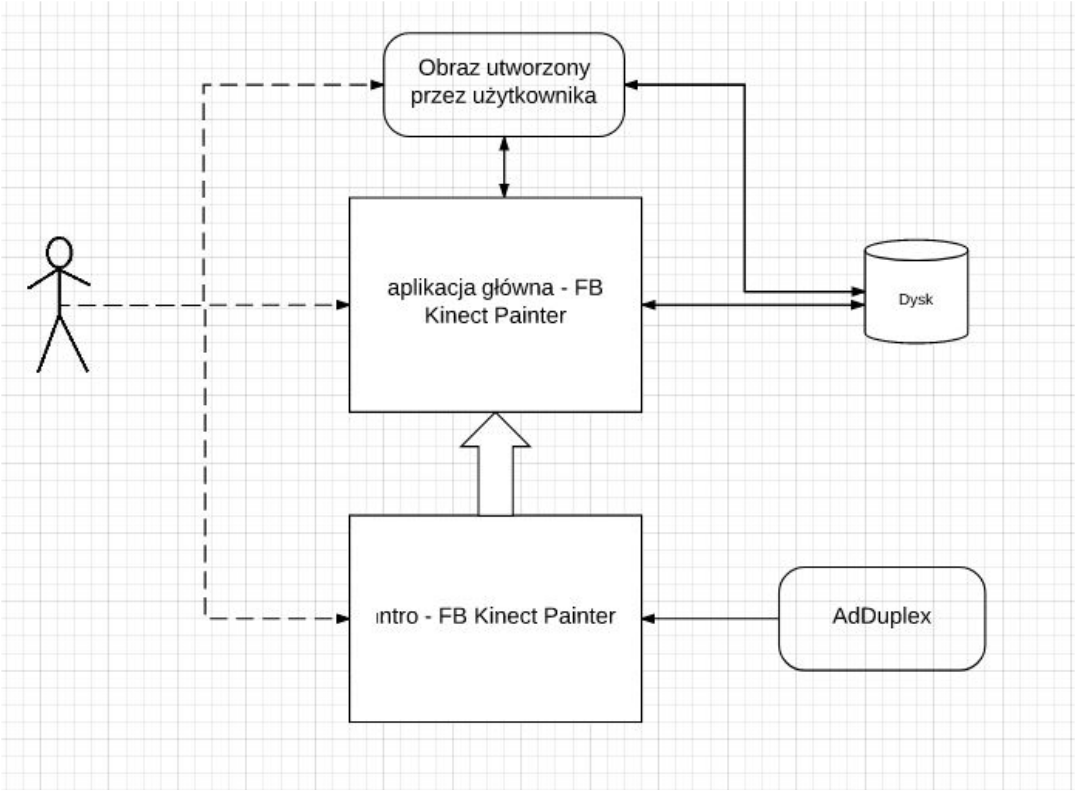
Kinect Studio - aplikacja udostępniona wraz z Kinect for Windows SDK umożliwiająca podgląd strumienia obrazu pochodzącego z kamery umieszczonej w urządzeniu Kinect w trzech trybach:

- zwykłego strumienia wideo (Color Viewer),
- podglądu 3D zeskanowanego pomieszczenia i obiektów (3D Viewer),
- sensora głębi (Depth Viewer).

W celu korzystania z oprogramowania należy uruchomić aplikację wykorzystującą urządzenie Kinect i podłączyć ją do Kinect Studio. Podpięta aplikacja musi mieć zaimplementowaną aktywację powyższych trybów.

6. Model Aplikacji

6.1. Architektura



Rys 9. Architektura aplikacji.

Użytkownik obsługuje za pomocą myszy komputerowej aplikację Intro. Z poziomu tej aplikacji może uruchomić aplikację główna FB Kinect Painter, w której interfejs jest obsługiwany poprzez sensor Kinect 360. Przy pomocy ruchów dłoni i dostępnych w programie narzędzi do malowania, użytkownik może utworzyć dowolny obraz. Utworzony przez użytkownika obraz może zostać zapisany na dysku lokalnym. Zapisany wcześniej obraz może zostać wczytany do aplikacji w celu dalszej edycji.

Aplikacja Intro komunikuje się z platformą AdDuplex w celu pobrania i wyświetlenia treści reklamowych.

6.2. Struktury danych

Wiele obiektów danej klasy jest statyczna z publicznym modyfikatorem dostępu (public), głównym celem tego rozwiązania była możliwość uzyskania dostępu do obiektów z poziomu każdej klasy, dotyczy to głównie obiektów odpowiadających za narzędzia i zmianę kolorów. Wobec obiektu odpowiadającego za nawiązanie połączenia z urządzeniem Kinect wykorzystaliśmy wzorzec projektowy Singleton, gdyż chcieliśmy mieć 100%

pewność istnienia tylko jednej instancji tego typu. Obiekt ten tworzony jest w momencie uruchamiania aplikacji.

Klasa WorkSeheet - zawiera dostępne narzędzia oraz obiekt typu InkCanvas, który umożliwia tworzenie obrazów. Zmienne przechowujące ścieżkę do aktualnie edytowanego obrazu i nazwę pliku (w przypadku nowego obrazu zmienne te są puste).

Klasa Painting_Tool - przechowuje dane narzędzia plik kursora, wielkość narzędzia oraz informację o sposobie edycji obiektu InkCanvas.

Klasa Settings - przechowuje ustawienia, które są wczytywane z pliku.

Klasa User - przechowuje informacje o użytkowniku. Tutaj zdefiniowana jest ręka główna i pomocnicza.

7. Problemy (w tym rozwiązania)

7.1. Moduł RTB

Problem z modułem reklam polegał na tym, że platformy reklamowe nie wspierają aplikacji desktopowych (bynajmniej nie bezpośrednio).

Wyświetlanie reklam w aplikacjach WPF (Windows Presentation Foundation):

- brak usługi udostępniającej reklamy dla WPF

- możliwość wyświetlenia reklamy ze strony internetowej (AdSense)

  AdSense umożliwia wyświetlanie reklam na stronach internetowych, po zweryfikowaniu treści przez Googla. Z takiej strony można by pobrać reklamę i wyświetlić ją w aplikacji WPF przy użyciu kontrolki WebBrowser (kontrolka umożliwiająca wyświetlanie treści stron internetowych). Dużym utrudnieniem tego rozwiązania są wysokie wymogi co do treści naszej strony WWW. Musi ich być dużo, muszą być unikalne, nie mogą zawierać treści niedozwolonych oraz muszą pozwalać na jednoznaczne określenie tematyki. Przy naszym projekcie okazało się to bardzo trudne. Kilkukrotnie zgłaszaliśmy stronę z opracowanymi treściami na temat naszej aplikacji i za każdym razem otrzymywaliśmy negatywną odpowiedź.

- utworzenie odrębnej aplikacji wyświetlającej reklamy (AdDuplex)

  AdDuplex umożliwia wyświetlanie reklam w aplikacjach Windows Store. Taką aplikacją jest aplikacja UWP, którą można uruchamiać z poziomu Windows 10. Daje nam to możliwość stworzenia aplikacji która byłaby wywoływana podczas działania głównego programu FB Kinect Painter w celu prezentacji reklam. Po zmaganiach z AdSense AdDuplex okazał się lepszym rozwiązaniem. Nie wymaga weryfikacji naszej aplikacji, ręcznie przydzielamy tematykę reklam (kategorię). Ograniczeniem są jednak formaty reklam. Są to: baner w jednym rozmiarze (tekstowy lub grafika) oraz reklama pełnoekranowa (grafika).

Po wielu próbach podpięcia reklamy bezpośrednio do naszej aplikacji zdecydowaliśmy się utworzyć aplikację “Intro” pośredniczącej w uruchamianiu aplikacji właściwej. W tej aplikacji umieściliśmy reklamy z AdDuplex oraz dodaliśmy krótką informację na temat projektu oraz wymagań sprzętowych i programowych. “Intro” umożliwia użytkownikowi konfigurację ustawień takich jak:

- wybór głównej ręki
- wybór lokalizacji zapisu obrazów

7.2. Nazewnictwo plików i rozszerzenie fbkp

Przy tworzeniu obsługi zapisu obrazu pojawił się problem z nazewnictwem pliku, musieliśmy odpowiedź sobie na pytanie jak je zrealizować:

- utworzyć nowe okno z klawiaturą, przy pomocy której użytkownik będzie mógł podać nazwę pliku,
- nadawać nazwy domyślnie generowane przez program.

Po przemyśleniach doszliśmy do wniosku, że najwygodniejszym sposobem z punktu widzenia użytkownika i nas programistów będzie zaimplementowanie automatycznego nadawania nazw przez program. Przyjętą przez nas nazwą pliku została nazwa “FB_Kinect_Painter_x+1”, gdzie x oznacza ilość plików z rozszerzeniem fbkp znajdujących się folderze wskazanym w opcjach programu, która jest powiększana o 1. W momencie zapisywania obrazu aplikacja tworzy dwa pliki jeden z rozszerzeniem fbkp i drugi z rozszerzeniem bmp. Pierwszy plik wykorzystywany jest przez nasz program do wczytywania obrazu, natomiast drugi, użytkownik może wykorzystać do edycji w dowolnym programie do edycji grafiki rastrowej obsługującej rozszerzenie bmp.

Plik z rozszerzeniem fbkp przechowuje listę obiektów (strokes), posiadających swoje punkty charakterystyczne. Dzięki tym punktom każdy obiekt jest wyświetlany w kontrolce InkCanvas tworząc razem jeden obraz. Tak powstały obraz jest rzutowany na bitmapę.

7.3. Problemy z wyświetlaniem interfejsu

W trakcie tworzenia interfejsu natknęliśmy się na problem z jego wyświetlaniem. Interfejs tworzony był przy wykorzystaniu monitorów, których proporcje wynosiły 16:10. W momencie, gdy uruchamialiśmy aplikację na laptopach w których współczynniki ekranu wynoszą 16:9, interfejs odbiegał wyglądem od tego jak go zaprojektowaliśmy wcześniej. Minęło trochę czasu zanim spostrzegliśmy się że ekran monitora i laptopa mają inne współczynniki. Rozwiązaniem problemu było uzależnienie wielkości elementów tj. przycisków od szerokości. W wyniku czego nasz interfejs wygląda równie dobrze na ekranach Full HD jak i tych ze współczynnikiem 16:10.

7.4. Problemy z przechwytywaniem ruchów

Sensor Kinect 360 zwraca wyniki w maksymalnej rozdzielczości 640x480 pikseli. Jest to rozdzielczość znacznie mniejsza od najpopularniejszych dziś rozdzielczości (1366x768 - większość laptopów, 1920x1080 - Full HD, 1680x1050 - monitory 16:10). W związku z tym zostaliśmy zmuszeni do przeskalowania wyników, do rozdzielczości naszej aplikacji - było to niezbędne aby kontrolować cały interfejs przy pomocy

urządzenia Kinect. Efektem przeskalowania była propagacja błędu odczytu pozycji dłoni. W wyniku nasz kursor zaczął intensywnie drgać, uniemożliwiając obsługę aplikacji i utworzenia jakiegokolwiek obrazu.

Problem ten rozwiązaliśmy poprzez filtrowanie i wygładzanie wyników zwracanych przez sensor. W tym celu posłużyliśmy funkcją z Kinect SDK przyjmującą następujące parametry (Rys. X.):

- Smoothing
  - zwiększenie wartości powoduje zwrócenie bardziej wyrównanych wartości pozycji szkieletu
  - zwiększenie wartości powoduje zwiększenie opóźnień
  - przyjmuje wartości z zakresu 0 - 1.0

- Correction
  - korekta pomiaru (0 - dane mocno wygładzone, 1.0 - zwraca surowe dane)
  - przyjmuje wartości z zakresu 0 - 1.0

- JitterRadius
  - redukcja krótkotrwałych odchyleń

- MaxDeviationRadius
  - maksymalny promień w metrach odchylenia od bieżącej pozycji
  - większe odchylenia są zaciskane w promieniu

Parametry zostały dobrane metodą prób i błędów.

```
//smoothing
TransformSmoothParameters smoothingParam = new TransformSmoothParameters();
{
    smoothingParam.Smoothing = 0.7f;
    smoothingParam.Correction = 0.3f;
    smoothingParam.Prediction = 1.0f;
    smoothingParam.JitterRadius = 1.0f;
    smoothingParam.MaxDeviationRadius = 1.0f;
};
```

Rys 10. Parametry filtrów.

8. Instrukcja

8.1. Instalacja

8.1.1. Minimalne wymagania

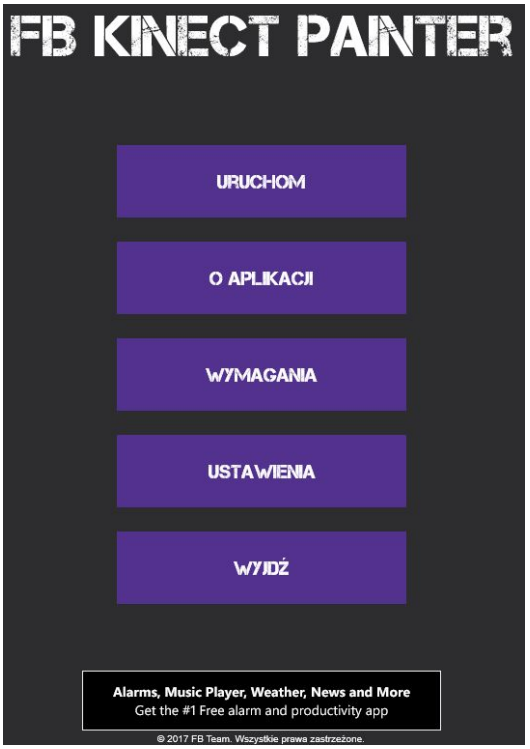
- sensor Kinect 360,
- system Windows 10 x64,
- procesor posiadający co najmniej 2 rdzenie z taktowaniem co najmniej 2.66GHz,
- minimum 2GB RAM,
- karta graficzna zgodna z Directx 9.0c.

8.1.2. Opis instalacji

Zanim przystąpimy do użytkowania aplikacji (intro) należy ją zainstalować z poziomu developera, gdyż nie jest dostępna z poziomu Windows Store. Istnieje również możliwość uruchomienia głównej aplikacji bez aplikacji pośredniczącej, jednakże w tym wypadku nie istnieje możliwość zmiany domyślnych ustawień, czyli zmiany ręki do malowania oraz katalogu zapisu obrazu, którym w tym wypadku będzie katalog aplikacji i utworzony w nim folder saved_pictures.

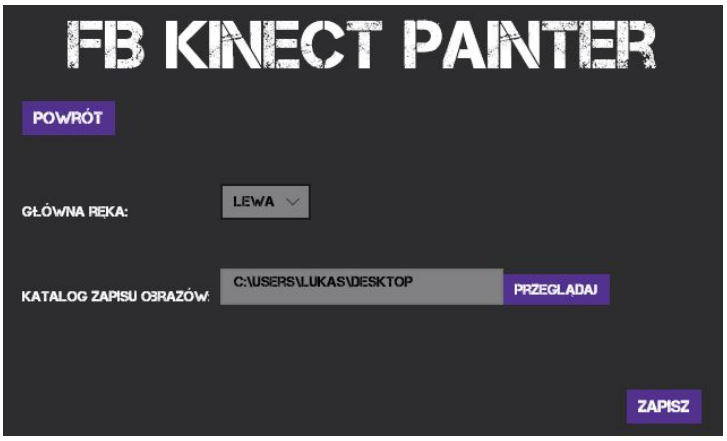
Główna aplikacja nie wymaga instalacji - jest to program typu portable.

8.2. Obsługa



Rys 11. Widok główny aplikacji UWP

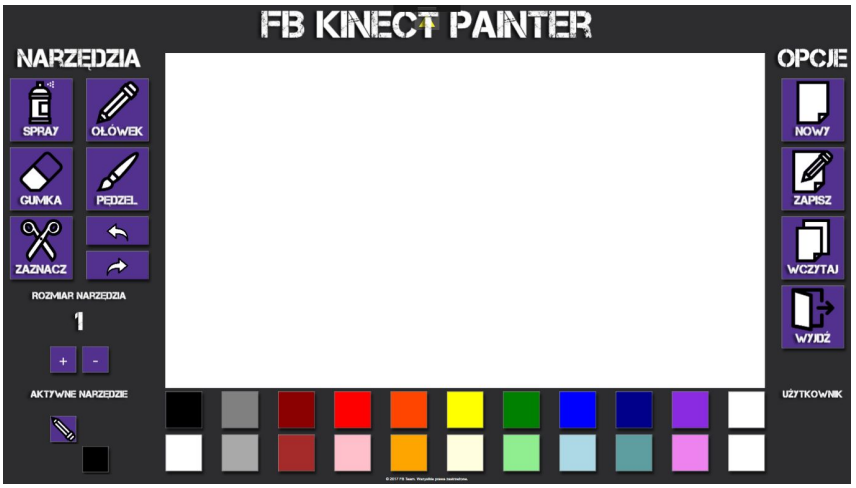
Przed przystąpieniem do uruchomienia głównej aplikacji należy ją wpierw dostosować do swoich preferencji. W tym celu klikamy w przycisk ustawienia (rys 11). Po wykonaniu powyższej operacji pojawi się okno (rys 12).



Rys 12. Ustawienia.

W tym miejscu wybieramy rękę, którą chcemy malować oraz wskazujemy ścieżkę zapisu tworzonych przez nas obrazów. W celu dodania ścieżki klikamy w przycisk przeglądaj (rys 12), wybieramy

odpowiadający nam katalog i klikamy ok. Jeżeli wszystko zostało skonfigurowane klikamy w przycisk zapisz. Teraz możemy śmiało przystąpić do uruchomienia aplikacji. Po skonfigurowaniu aplikacji wyświetli nam się okno główne (rys 13):



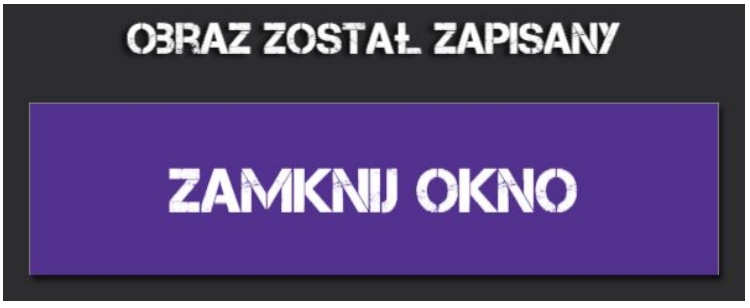
Rys 13. Okno główne aplikacji.

1. Stań w odpowiedniej odległości (do 1.5 m) od urządzenia Kinect, zaczekaj aż urządzenie Cię wykryje. W momencie zakończenia wykrywania Twoja zmapowana postać pojawi się pod napisem “Użytkownik”. Pamiętaj, aby dłonie zawsze były w polu widzenia urządzenia.

2. W celu poruszania kursorem wystarczy, że ruszysz ręką, którą wybrałeś w momencie konfiguracji.

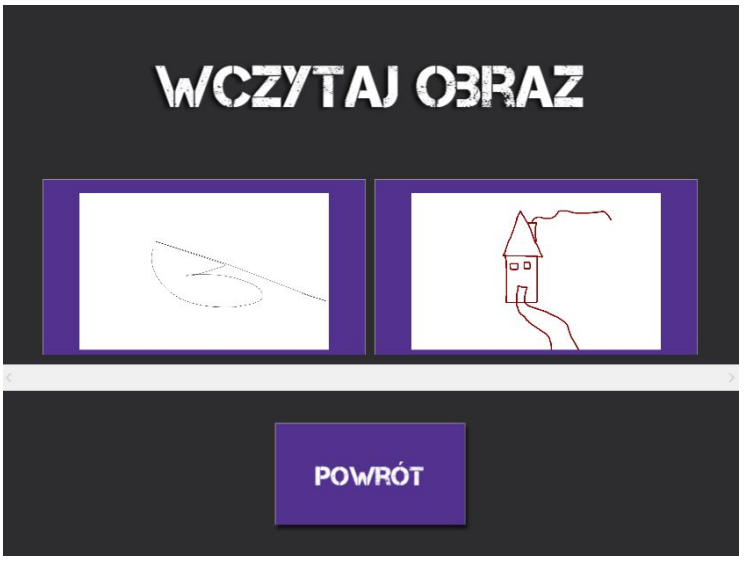
3. Aby wdusić dowolny przycisk należy na niego najechać i unieść drugą dłoń powyżej łokcia.

4. W momencie zakończenia prac możemy zapisać efekty naszej pracy, w tym przypadku klikamy w przycisk zapisz. Po pomyślnym zapisaniu pojawi się okno (rys 14):



Rys 14. Okno pomyślnego zapisu.

5. Aby wczytać obraz należy kliknąć w przycisk wczytaj, pojawi się okno (rys 15) z którego możemy wybrać obraz, który chcemy wczytać. W przypadku braku obrazów pojawi się stosowny napis.



Rys 15. Okno wczytywania.

6. W celu utworzenia nowego obrazu klikamy w przycisk nowy, po którego kliknięciu pojawi się okno (rys 16), którego celem jest potwierdzenie naszej decyzji:

Rys 16. Okno potwierdzające utworzenie nowego obrazu.

9. Podsumowanie

9.1. Możliwości dalszego rozwoju aplikacji

Warto byłoby się zastanowić nad dalszym rozwojem aplikacji, jeżeli miałoby do niego dojść to warto byłoby wykorzystać tutaj biblioteki OpenCL i OpenCV, które mogą znacznie przyspieszyć działanie programu. W późniejszym etapie dostosować aplikację, aby działała pod urządzeniem Xbox360 i udostępniać ją za darmo użytkownikom konsoli posiadających sensor Kinect.

9.2. Cele zrealizowane

Udało nam się zrealizować wszystkie podstawowe cele jakie przed sobą postawiliśmy:

- możliwość malowania przy wykorzystaniu urządzenia Kinect,
- wybór narzędzi malowania,
- wybór kolorów,
- możliwość zapisu i wczytania obrazów.

9.3. Cele niezrealizowane

Jednym z głównych niezrealizowanych celów jest brak możliwości edycji obrazów utworzonych w innych programach do edycji grafiki rastrowej. Zostało to spowodowane pewnymi ograniczeniami obiektu InkCanvas zastosowanego w naszej aplikacji, który przy malowaniu tworzy specjalne obiekty (scopes), które mogą być zaznaczane przy pomocy narzędzia wycinania. W przypadku wczytania obrazu z innego programu zaznaczanie byłoby niemożliwe co wyglądałoby nieelegancko.

10. Bibliografia

- Jarrett Webb, James Ashley: *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012.
- Abhijit Jana: *Kinect for Windows SDK Programming Guide*. Packt Publishing, 2012.
- http://dotneteers.net/blogs/vbandi/archive/2013/03/25/kinect-interactions-with-wpf-part-i-getting-started.aspx
- https://en.wikipedia.org/wiki/AdDuplex
- https://pl.wikipedia.org/wiki/Windows_Presentation_Foundation
- https://en.wikipedia.org/wiki/Universal_Windows_Platform
- https://pl.wikipedia.org/wiki/Kinect
- https://pl.wikipedia.org/wiki/Microsoft_Visual_Studio