POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Tomasz Kuczyński

Nr albumu 9833

Nieliniowe prezentowanie informacji z uwzględnieniem zainteresowań grupy docelowej

Praca magisterska

napisana pod kierunkiem:

dr inż. Mariusz Trzaska

Streszczenie

Praca dotyczy zagadnienia nieliniowych prezentacji, w tym tworzenia oraz wyświetlania określonym grupom odbiorców. Odejście od dobrze znanego, liniowego sposobu rozmieszczania slajdów pozwala na budowanie ciekawszych oraz bardziej dopasowanych pokazów, skierowanych do osób o konkretnych potrzebach. W niniejszej pracy podjęto próbę wypracowania rozwiązania, które pozwoliło by ułatwić, przyśpieszyć oraz unowocześnić cały proces.

Zaproponowana koncepcja porusza aspekty generowania oraz renderowania slajdów, z uwzględnieniem nieliniowości oraz środowiska 3D. Prototypy wykorzystują pomysły z zaproponowanego rozwiązania i demonstrują jedną z wielu dróg, które mogły by prowadzić do stworzenia aplikacji ułatwiających powyższe zadanie.

Spis treści

1. Wstęp

Najpopularniejszą formą przekazu informacji do szerokiego grona osób jest prezentacja multimedialna. Wypowiedzią, na której treść, oprócz tekstu mogą składać się elementy takie jak zdjęcia, wykresy, filmy czy dźwięki.

Najbardziej znanym programem umożliwiającym zbudowanie pokazu jest produkt firmy Microsoft – PowerPoint [1], pozwalający na tworzenie slajdów, których zmiana następuje po akcji użytkownika lub automatycznie po określonym czasie. Oprócz podejścia zastosowanego w PowerPoint, istnieją inne, ciekawsze rozwiązania, pozwalające na generowanie pokazów, takie jak Prezi – webowe oprogramowanie wykorzystujące technologie Flash i innowacyjne połączenia pomiędzy slajdami.

Budowanie prezentacji w programie PowerPoint odbywa się zazwyczaj w sposób liniowy to znaczy, że po slajdzie pierwszym, nastąpi slajd drugi i tak do ostatniego. Powyższe podejście jest przestarzałe, jednak wciąż szeroko stosowane. Istnieje również możliwość zastosowania przycisków z podpiętymi akcjami, które pozwolą na obranie innej ścieżki przepływu informacji. Wykorzystanie omawianego rozwiązania spowoduje utworzenie pokazu wykorzystującego nieliniowe ułożenie slajdów i umożliwi wzrost użyteczności całej prezentacji.

1.1. Cel pracy

Zbudowanie prezentacji jest pierwszym etapem dostarczenia informacji. Forma przekazu oraz dopasowanie treści do odbiorców odgrywają kluczową rolę w sukcesie całego pokazu. Oprogramowanie dostępne na rynku znacząco rozszerza możliwości generowania i wyświetlania przygotowanych slajdów, jednakże wciąż istnieje potrzeba uproszczenia i unowocześnienia omawianego procesu.

Celem pracy jest przedstawienie koncepcji usprawniającej tworzenie nieliniowych prezentacji dla różnych grup docelowych oraz wykorzystanie środowiska 3D do wyświetlenia utworzonego pokazu. Zaproponowane rozwiązanie skupia się na rozszerzeniu właściwości slajdu oraz sposobie prezentowania, co umożliwi inne spojrzenie na proces budowania prezentacji.

Produkty uboczne oparte na omawianej idei, to dwa narzędzia pozwalające na tworzenie i wyświetlanie nieliniowych prezentacji.

Edytor pozwala na:

- tworzenie grup docelowych i slajdów;

- rozmieszczenie elementów na slajdzie;

- ustawianie kolejności ich wyświetlania;

- zapis do pliku XML;

Prezenter umożliwia:

- odczytanie prezentacji z pliku XML;

- wyświetlenie slajdów prezentacji;

- zmianę grupy docelowej gdy slajd to umożliwia;

1.2. Rozwiązania przyjęte w pracy

Oba prototypy powstały z wykorzystaniem środowiska programistycznego Visual Studio firmy Microsoft, umożliwiającego tworzenie różnego typu aplikacji: od okienkowych, po strony internetowe, a kończąc na oprogramowaniu oferowanym na konsole Xbox, czy telefony komórkowe. Język wybrany do implementacji rozwiązania to C#, oparty o platformę .NET, podobny do Javy, umożliwiający sprawne wykorzystanie możliwości oferowanych przez framework Microsoftu.

Edytor został napisany w oparciu o Windows Presentation Foundation - silnik graficzny wykorzystujący język XAML do opisu interfejsu użytkownika. Dzięki rozdzieleniu warstwy prezentacyjnej od logiki biznesowej, WPF był idealnym do omawianego celu rozwiązaniem.

Prezenter zbudowany został z wykorzystaniem frameworku XNA przygotowanego głównie jako zbiór bibliotek dla aplikacji 3D, w tym gier na systemy Windows oraz konsolę Xbox. Microsoft zadbał o wiele pomocniczych funkcji na przykład umożliwiające przekształcenia macierzowe, renderowanie za pomocą akceleratora graficznego, czy obsługę kontrolera lub klawiatury.

Oba prototypy bazują na pliku XML zawierającym definicję struktury prezentacji, który mieści informacje dotyczące grup docelowych oraz slajdów wraz z ich zawartością. Zastosowanie powyższego formatu umożliwiło łatwy odczyt i zapis danych.

1.3. Rezultaty pracy

Rezultatem pracy jest rozwiązanie mające usprawnić i unowocześnić tworzenie nielinowych prezentacji multimedialnych, z uwzględnieniem zainteresowań grup docelowych. Przedstawiona

koncepcja zawiera między innymi propozycję struktury prezentacji oraz sposobu renderowania, co zostało wykorzystane do zbudowania obu aplikacji.

W omawiany sposób uzyskano prototypy narzędzi do tworzenia, edycji oraz wyświetlania pokazu w środowisku 3D, które zawierają treści skierowane w kierunku różnych typów odbiorców. Wybór ścieżki poprowadzenia prezentacji zależy wyłącznie od prowadzącego oraz tego, która grupa w danym momencie powinna otrzymać informację.

1.4. Organizacja pracy

Rozdział pierwszy stanowi krótki wstęp poruszający tematykę nieliniowych pokazów slajdów. Następnie „Istniejące rozwiązania do tworzenia prezentacji" zawiera rozwinięcie wstępu, definicję prezentacji multimedialnej oraz porównanie rozwiązań zastosowanych w aplikacjach dostępnych na rynku. Trzeci rozdział dotyczy koncepcji usprawnienia procesu generowania treści, uwzględniających zainteresowania grup docelowych. Rozdział czwarty porusza tematykę technologii i rozwiązań wykorzystanych w realizacji pracy, kolejny dotyczy dokładnego opisu obu prototypów. Ostatni rozdział stanowi podsumowanie zawierające opis zalet i wad zaproponowanego rozwiązania oraz możliwe plany dalszego rozwoju.

2. Istniejące rozwiązania do tworzenia prezentacji

Prezentowany rozdział przedstawia i omawia obecne na rynku rozwiązania do tworzenia pokazów multimedialnych. Na początku zostanie zdefiniowane pojęcie, czym jest prezentacja i omówione potrzeby jej tworzenia, a także warunki jakie powinna spełniać, by zostać dobrze odebrana przez słuchaczy.

2.1. Czym jest prezentacja multimedialna

Prezentacja multimedialna stanowi formę przekazu informacji, która łączy elementy takie jak jak tekst, grafika, wideo czy animacja. Multimedia wykorzystane podczas tworzenia mają za zadanie zwrócić uwagę odbiorcy. Ludzie są wzrokowcami, w związku z tym znaczna większość przyswajanych informacji dociera do nas za pośrednictwem obrazu. Okazuje się, że możemy zapamiętać o wiele więcej, jeżeli materiał został ubarwiony elementami graficznymi. Najbardziej skuteczną formą przekazu jest infografika – wykorzystująca obrazy, wykresy i tekst, z wykorzystaniem ludzkiej możliwości widzenia schematów i zauważania trendów.

Zaprezentowanie wiedzy szerokiej grupie odbiorców stało się dużo prostsze poprzez wykorzystanie prezentacji multimedialnych. Potrzeba przekazania informacji jest głównym celem tworzenia. Wraz z pojawieniem się najbardziej znanego programu do budowania pokazów slajdów – PowerPoint, omawiana forma przekazu jest obecna na każdym szkoleniu, wykładzie czy spotkaniu biznesowym. Rozwiązanie Microsoftu od lat króluje na rynku, a nauka jego używania w wielu szkołach prowadzona jest już od klas podstawowych na pierwszych zajęciach z informatyki.

Nie należy zapominać, że nawet najlepiej przygotowana prezentacja nie wygłosi się sama. Trzeba pamiętać, że to osoba prezentująca stanowi główne medium przekazu informacji, a referowany materiał to jedynie dodatek mający na celu zaciekawienie odbiorcy i przedstawienie rzeczy niemożliwych do zaprezentowania przez element ludzki. Cechami dobrze przygotowanej prezentacji są między innymi [2]:

- prosty i czytelny wygląd – powinien być prosty i spójny, nie może odwracać uwagi od treści które chcemy przekazać. Zastosowanie dużych czcionek dla ważnych elementów ułatwi zapamiętanie i przykuje uwagę;

- wspieranie przekazu – pokaz slajdów ma być tylko pomocą dla prowadzącego, a nie zawierać treści, które wygłasza. Najgorszą możliwą prezentacją jest czytana przez prowadzącego;

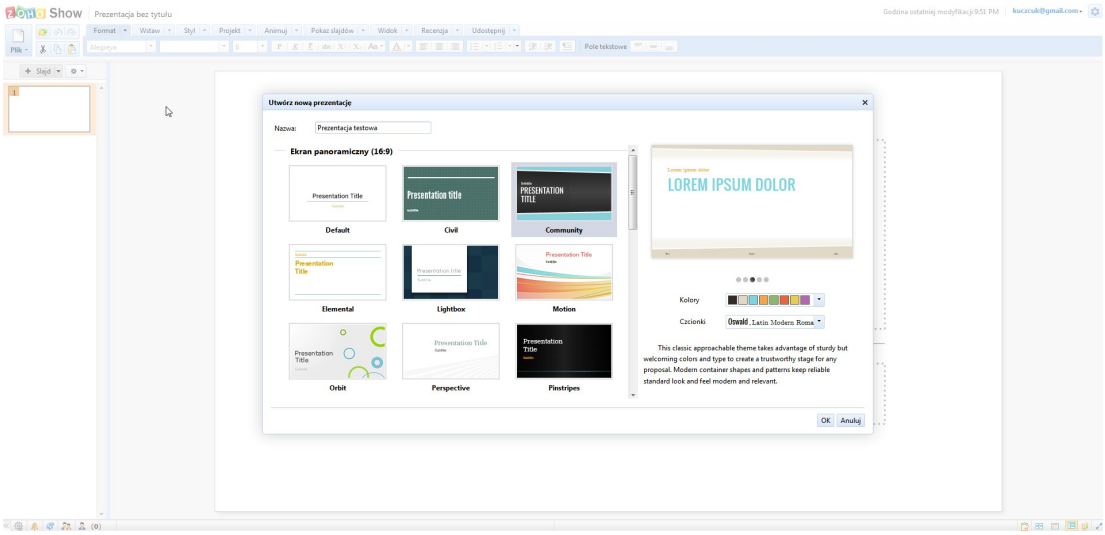
- odpowiednia ilość informacji – umieszczenie dużej ilości informacji nie poprawia jej przyswajalności. Jeżeli dany slajd będzie zbyt rozbudowany to słuchacz będzie miał problemy ze zrozumieniem przekazu;

2.2. Zoho Docs

Zoho Docs jest webową aplikacją zawierającą moduły do tworzenia dokumentów tekstowych, arkuszy, notatek, zarządzania projektem i tworzenia prezentacji. Wykorzystuje model SaaS, polegający na udostępnieniu przechowywanego na hostingu oprogramowanie, które jest uruchamiane na komputerze klienta. Omawiany model powoduje przeniesienie obowiązków aktualizacji, zarządzania i pomocy technicznej na dostawcę – klienta nie interesuje infrastruktura oraz zaplecze techniczne. Powyższe możliwości porównywalne są z pakietem Office od Microsoftu czy OpenOffice.org.

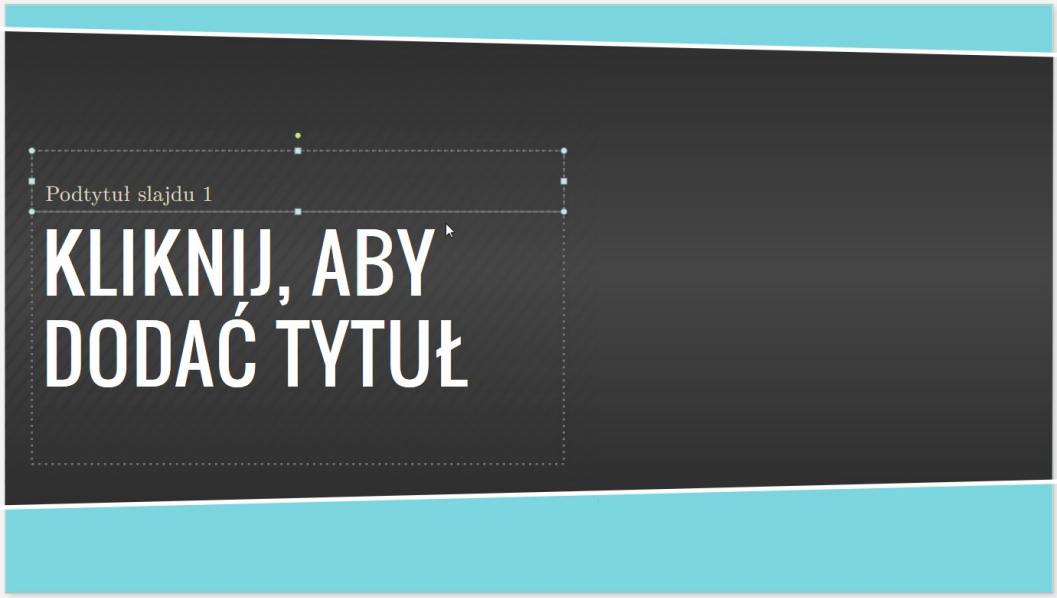
Budowanie prezentacji w Zoho Docs jest zbliżone do tworzenia w programie PowerPoint. Widok listy slajdów znajduje się po lewej stronie, menu pozwala na dodanie nowego slajdu, edycje zawartości, animacji i uruchomienie pokazu. Wykorzystanie intuicyjnego układu elementów interfejsu użytkownika umożliwia szybkie rozpoczęcie pracy bez zbędnego poznawania aplikacji. Powyższy program posiada możliwość importu z formatu .PPT lub .PPS a także .ODP lub .SXI. Po wykonaniu prezentację możemy udostępnić innym użytkownikom lub osadzić na stronie internetowej. Utworzony pokaz wykorzystuje kod HTML, natomiast obsługą animacji zajmuje się JavaScript.

Podczas rozpoczęcia pracy nad pokazem slajdów, kreator umożliwia wybór szablonu prezentacji (Rysunek 2.1.1). Możemy wybrać spośród wielu gotowych szablonów, a następnie wykorzystać wybrany schemat kolorów i czcionek. Omawiane podejście pozwala użytkownikowi szybko przejść do pracy nad zawartością pokazu i znacząco przyśpiesza cały proces tworzenia.



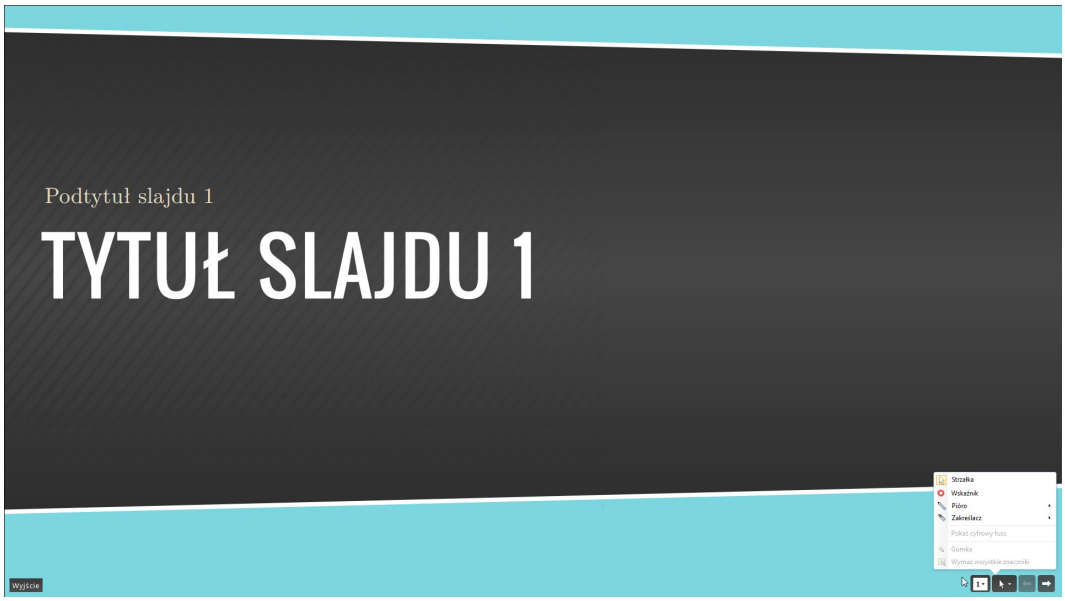
Rysunek 2.2.1: Zoho Docs – nazwa i wybór szablonu nowej prezentacji (źródło: https://docs.zoho.com)

Po zamknięciu pierwszego okna ukazuje się widok, pokazujący slajdy po lewej stronie oraz przestrzeń po prawej, gdzie możemy dodać tytuł i podtytuł. Szczegóły przedstawia Rysunek 2.2.2. W górnej części aplikacji widać menu – podobne do tego jakie znamy z PowerPoint. Każdy kto tworzył prezentacje w programie PowerPoint nie powinien mieć trudności z wykonaniem pokazu slajdów w Zoho Docs. Wszystko jest intuicyjne i nie wymaga korzystania z pomocy technicznej lub specjalnych kursów.



Rysunek 2.2.2: Zoho Docs – widok slajdu, dodawanie tytułu i podtytułu (źródło: https://docs.zoho.com)

Rysunek 2.2.3 przedstawia podobieństwo w sposobie wyświetlania pokazu slajdów, do aplikacji PowerPoint. Umożliwia przejścia do następnego lub poprzedniego slajdu jak i wybór z listy, w celu pójścia dalej, a także zakończenia pracy. Stworzona prezentacja jest stroną internetową zbudowaną w HTML. Każdy element zawiera odpowiednie style CSS, które upodabniają wygląd do znanego z PowerPoint. Duża ilość znaczników HTML, jak również rozbudowane skrypty języka JavaScript są wykorzystane do sterowania przebiegiem prezentacji i animowania przejść oraz elementów interaktywnych.



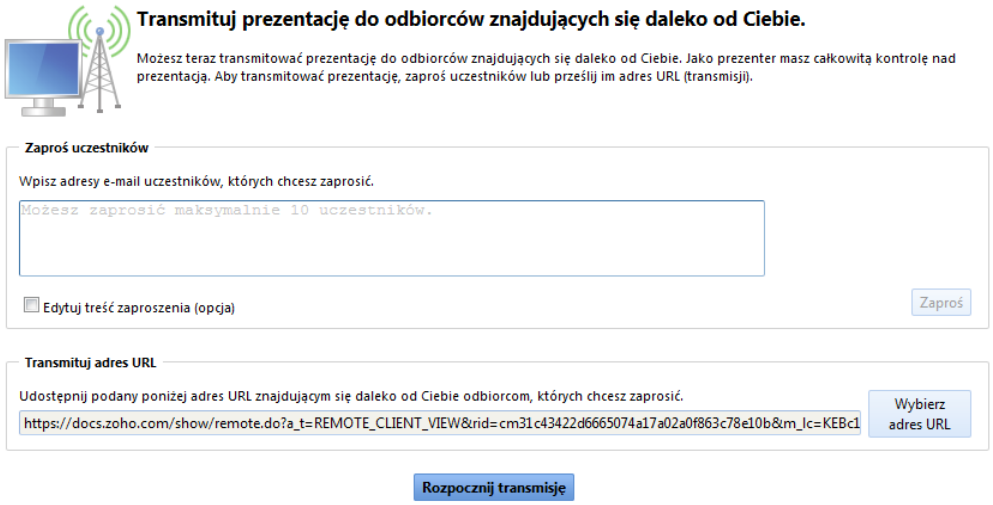
Rysunek 2.2.3: Zoho Docs – widok uruchomionej prezentacji (źródło: https://docs.zoho.com)

W porównaniu do dobrze znanej aplikacji PowerPoint, Zoho Docs różni się głównie tym, że wykorzystuje model SaaS. Po zarejestrowaniu się w serwisie można od razu rozpocząć pracę nad pokazem slajdów.

Rozmieszczenie elementów w aplikacji jest intuicyjne. Autorzy nie pokusili się o nowatorskie podejście do wyglądu interfejsu użytkownika, ponieważ inne ustawienie paneli widoku wymagałoby zapoznania się użytkownika z dokumentacją.

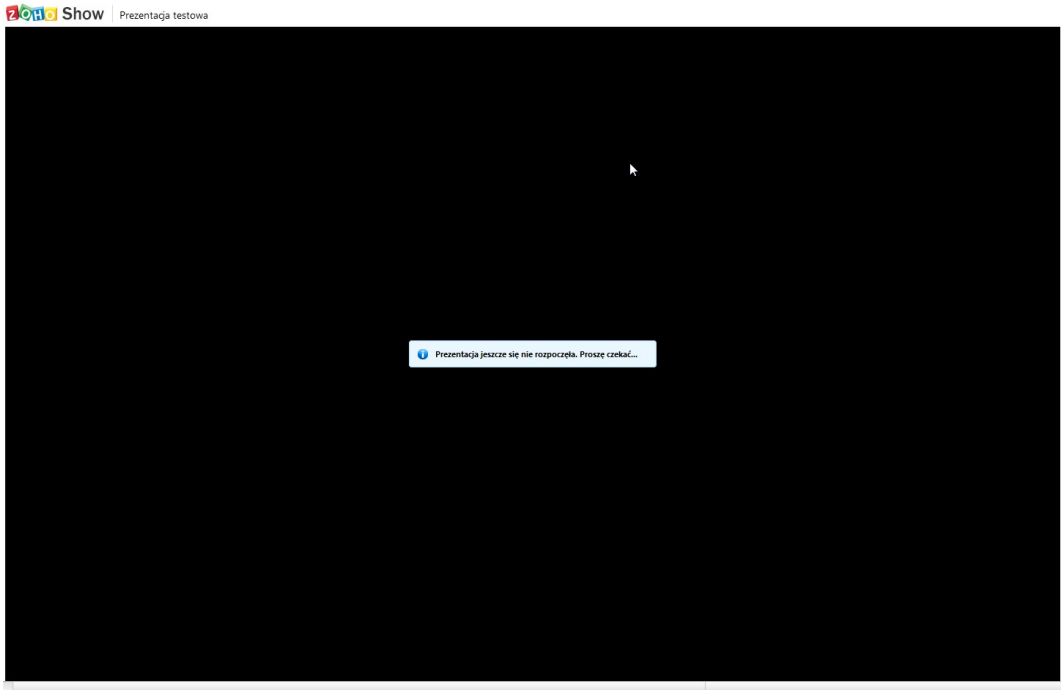
Zoho Docs jest odpowiednikiem znanego pakietu Google Docs. Funkcjonalność wyróżniająca powyższy produkt od konkurencji, dzięki czemu aplikacja firmy Zoho została wybrana do tego porównania, to konferencja połączona z pokazem slajdów.

Pozwala zaprosić dowolnego użytkownika, niekoniecznie posiadającego konto w serwisie Zoho. Umożliwia wysyłanie powiadomień do potencjalnych odbiorców za pomocą formularza (Rysunek 2.2.4), a następnie przeprowadzenie pokazu slajdów równocześnie rozmawiając na chacie z użytkownikami.



Rysunek 2.2.4: Zoho Docs – zaproszenie do konferencji (źródło: https://docs.zoho.com)

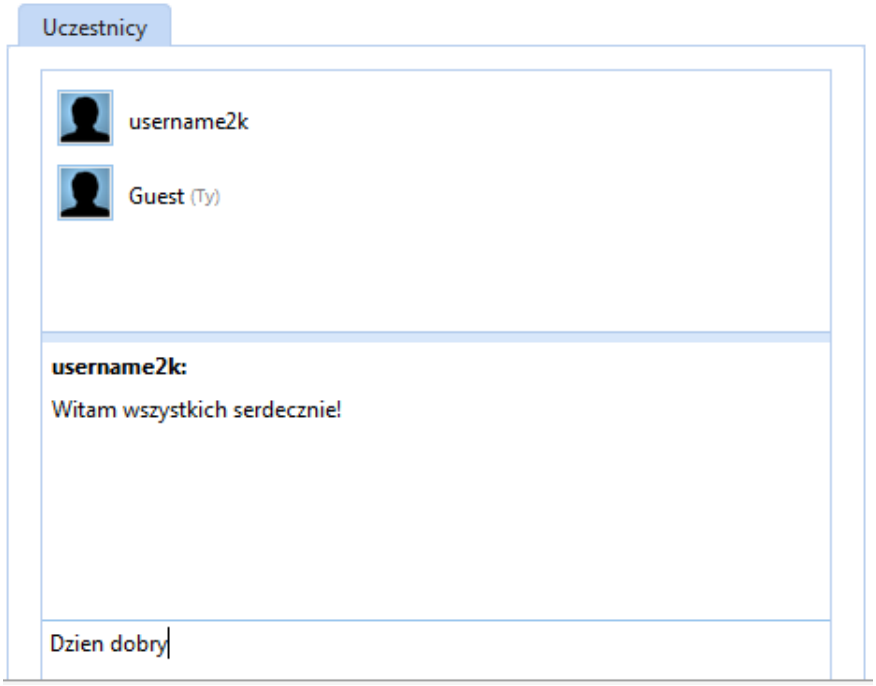
Innowacyjna funkcjonalność pozwala w czasie rzeczywistym kontrolować przebieg prezentacji oraz rozmawiać na chacie z uczestnikami w celu na przykład wyjaśnienia niejasności lub odpowiedzi na pytania. Umożliwia rezygnację z dodatkowego systemu do prowadzenia rozmów poprzez chat na rzecz jednego oprogramowania. Rysunek 2.2.5 demonstruje widok użytkownika zaproszonego do pokazu. Ekran ten będzie wyświetlony aż do momentu startu prezentacji.



Rysunek 2.2.5: Zoho Docs – widok zaproszonej osoby, oczekującej na rozpoczęcie prezentacji (źródło: https://docs.zoho.com)

Rysunek 2.2.6: Zoho Docs – widok rozpoczętej prezentacji (źródło: https://docs.zoho.com)

Po rozpoczęciu pokazu slajdy są automatycznie zmieniane przez prowadzącego. Osoba oglądająca prezentację ma możliwość przechodzenia pomiędzy dostępnymi slajdami. Po prawej stronie widzimy wszystkich uczestników i możemy prowadzić rozmowę na żywo za pomocą chatu. Szczegóły przedstawia Rysunek 2.2.6. W górnej części znajduje się lista osób zaproszonych, wraz z osobą prezentującą, poniżej obszar, w którym pojawiać się będzie treść rozmowy, a na samym dole możemy wprowadzać tekst (Rysunek 2.2.7).



Rysunek 2.2.7: Zoho Docs – okno chatu (źródło: https://docs.zoho.com)

Rozwinięcie omawianej funkcjonalności o elementy wideokonferencji byłoby na pewno dobrze odebrane przez użytkowników aplikacji. Zastosowanie powyższej interakcji pomiędzy osobami biorącymi udział w pokazie przeniosło by poziom przekazywania wiedzy na kolejny poziom, upodabniając go do wykładu na auli w rzeczywistym świecie.

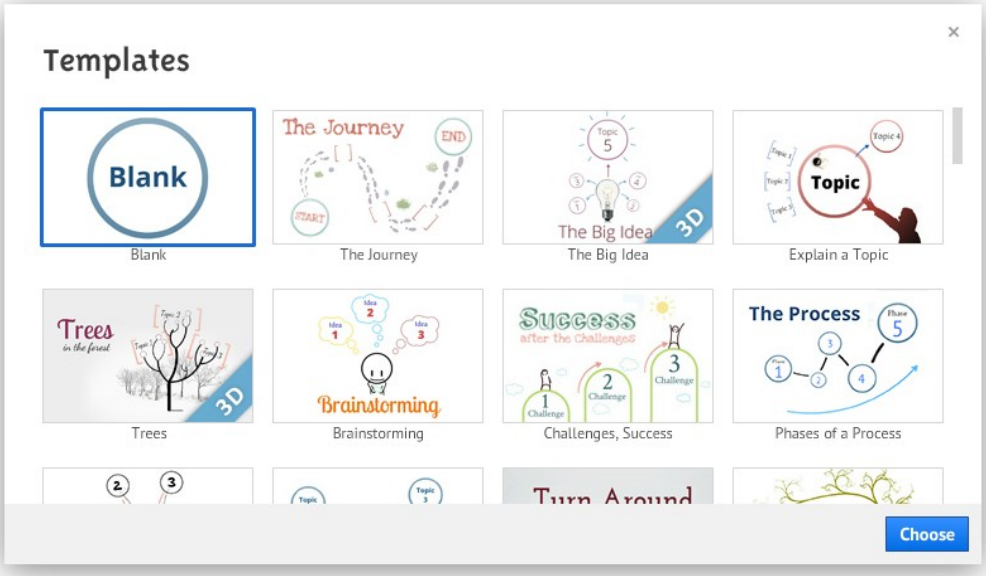
Podejście do tworzenia i wyświetlania pokazu slajdów zaprezentowane w Zoho Docs powinny być wzorem dla oprogramowań do tworzenia prezentacji. Upodobnienie wyglądu do dobrze znanego programu PowerPoint pozwoliło na skupieniu się nad wprowadzeniem nowych funkcjonalności, takich jak chociażby rozmowy z uczestnikami. Wykorzystanie modelu dystrybucji SaaS uniezależniło omawiany produkt od platformy sprzętowej pozwalając na uruchomieniu na dowolnym komputerze z zainstalowaną przeglądarką internetową i dostępem do internetu.

2.3. Prezi

Prezi jest aplikacją webową dostępną z poziomu przeglądarki internetowej. Analogicznie do poprzedniego oprogramowania wykorzystuje model SaaS jako metodę dystrybucji. Zbudowana jest w oparciu o Flash i ActionScript, co powoduje, że wymaga zainstalowanej wtyczki Flash w przeglądarce. Wykorzystanie powyższych technologii umożliwiło przygotowanie czytelnego interfejsu i innowacyjnego systemu prezentowania informacji. Sposób w jaki pokazywana jest prezentacja wyróżnia omawiane oprogramowanie spośród innych rozwiązań. Podejście do tworzenia pokazów jest zupełnie inne w porównaniu do PowerPoint czy Zoho Docs.

Materiały stworzone za pomocą Prezi przypominają wielką bitmapę z interaktywnymi elementami, które mogą zostać przybliżone, kamera może zmieniać swoją orientacje – obrócić się tak, aby dana część prezentacji była odpowiednio czytelna. Animacje pomiędzy przejściami są płynne i nie męczą użytkownika. Powyższe rozwiązanie bardzo dobrze sprawdza się oraz posiada szerokie możliwości tworzenia rozbudowanych pokazów, które z pewnością nie znużą odbiorców.

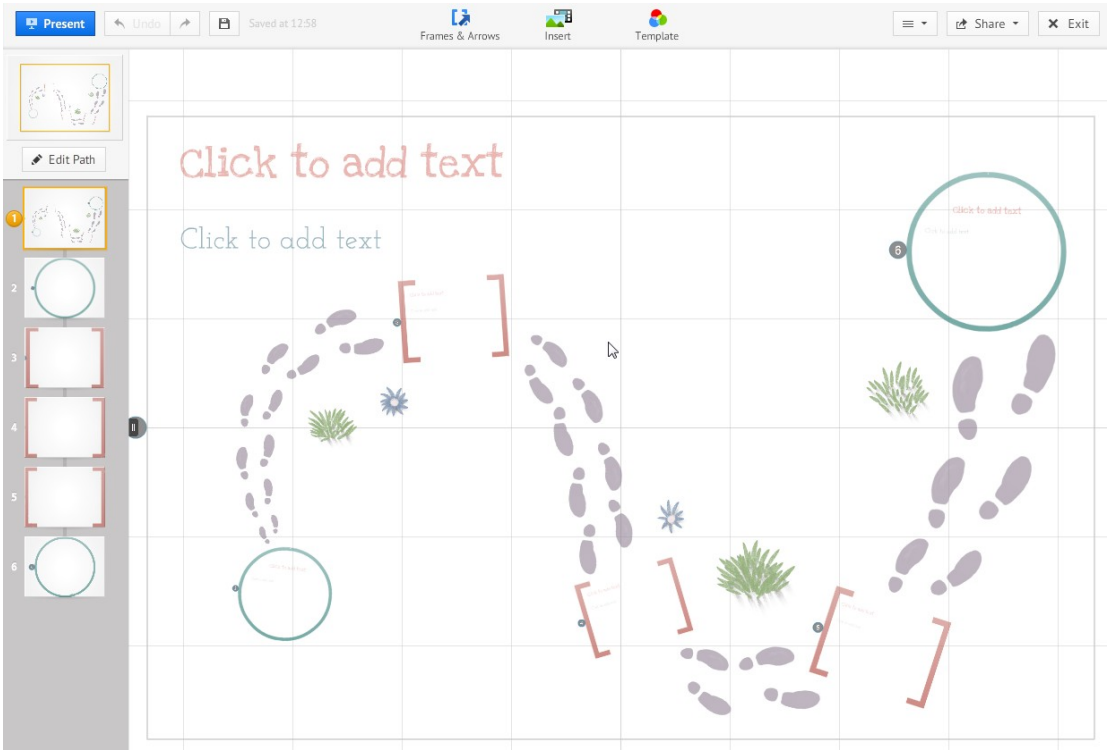
Rozpoczęcie tworzenia prezentacji poprzedzone jest wyborem szablonu, który stanowi dużą przestrzeń roboczą, gdzie możemy umieszczać stacje przystankowe slajdów oraz inne elementy takie jak bitmapy czy tekst. Szczegóły przedstawia Rysunek 2.3.1.



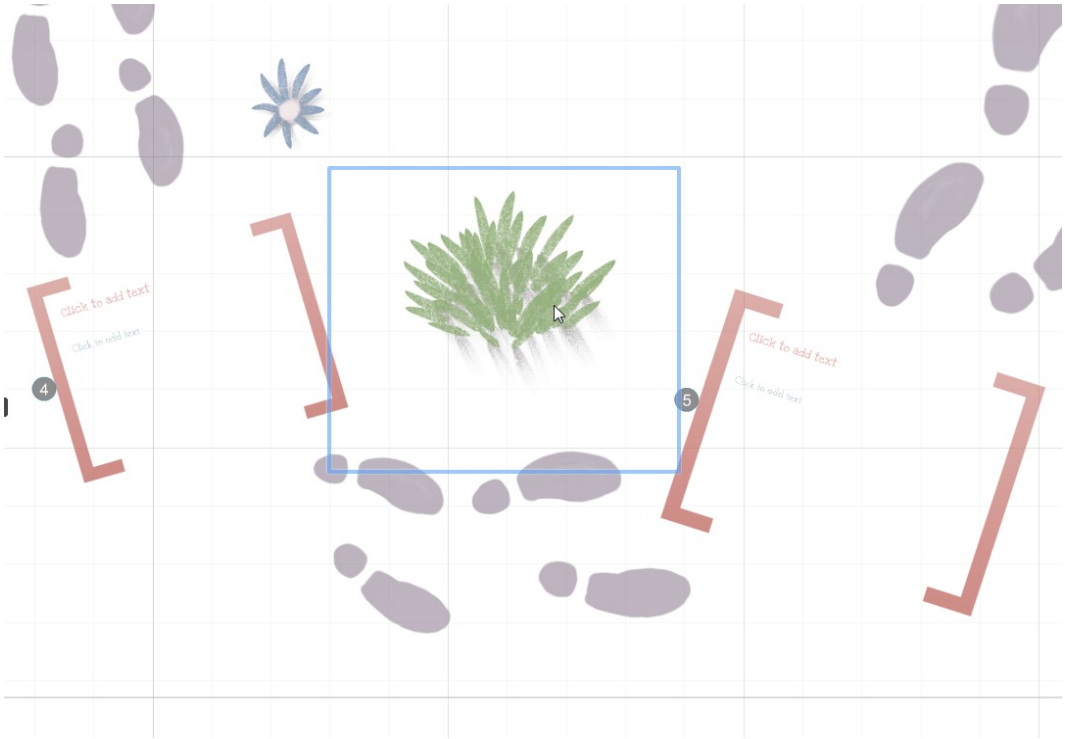
Rysunek 2.3.1: Prezi – ekran wyboru szablonu (źródło: http://prezi.com)

Do wyboru jest wiele gotowych szablonów z przygotowanymi wcześniej przystankami. Każdy obiekt w szablonie można edytować, dodać nowe przystanki i przejścia między nimi. Możliwości tworzenia prezentacji wydają się nieskończenie wielkie, z pewnością omawiane podejście znalazło wielu zwolenników.

Po dokonaniu wyboru szablonu ukazuje się widok na okno aplikacji (Rysunek 2.3.2). Tradycyjnie po lewej stronie widać slajdy (przystanki), możemy je tutaj również usuwać. U góry znajduje się menu zwierające niezbędne elementy pozwalające stworzyć imponujące prezentacje. Większość widoku zajmuje obszar roboczy. Poszczególne elementy na przystanku mogą zostać edytowane lub usunięte. Z pozycji menu można dodawać nowe obiekty takie jak grafika czy tekst, (Rysunku 2.3.3).



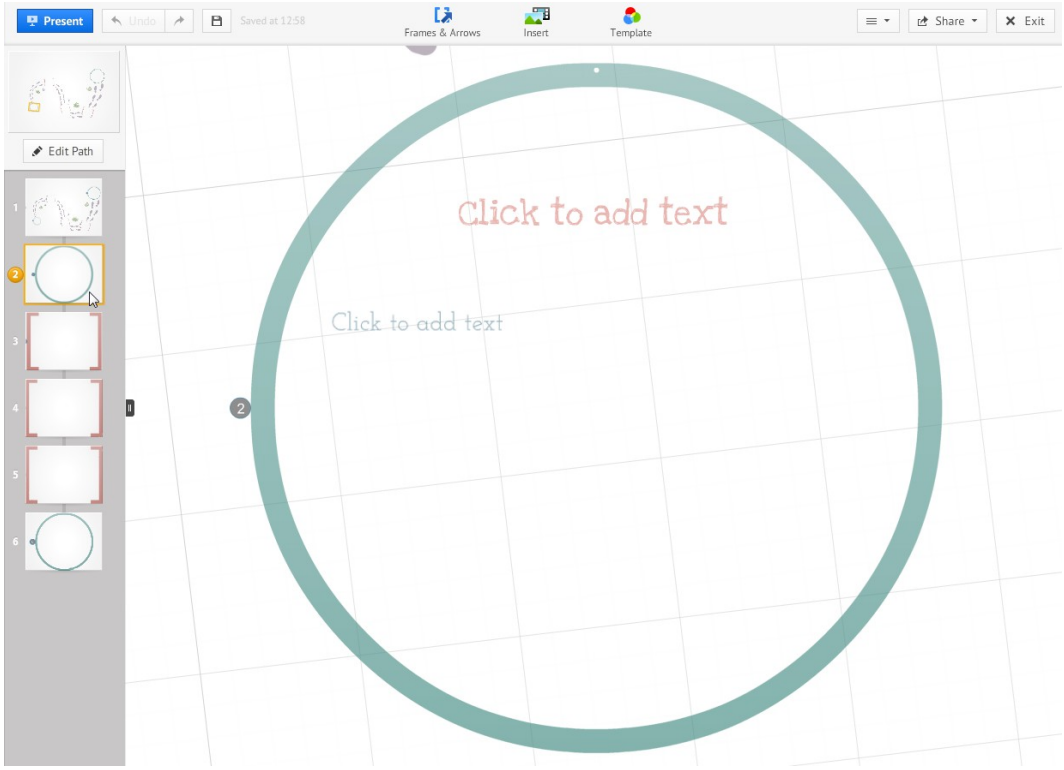
Rysunek 2.3.2: Prezi – okno edycji prezentacji (źródło: http://prezi.com)



Rysunek 2.3.3: Prezi – edycja elementów na szablonie (źródło: http://prezi.com)

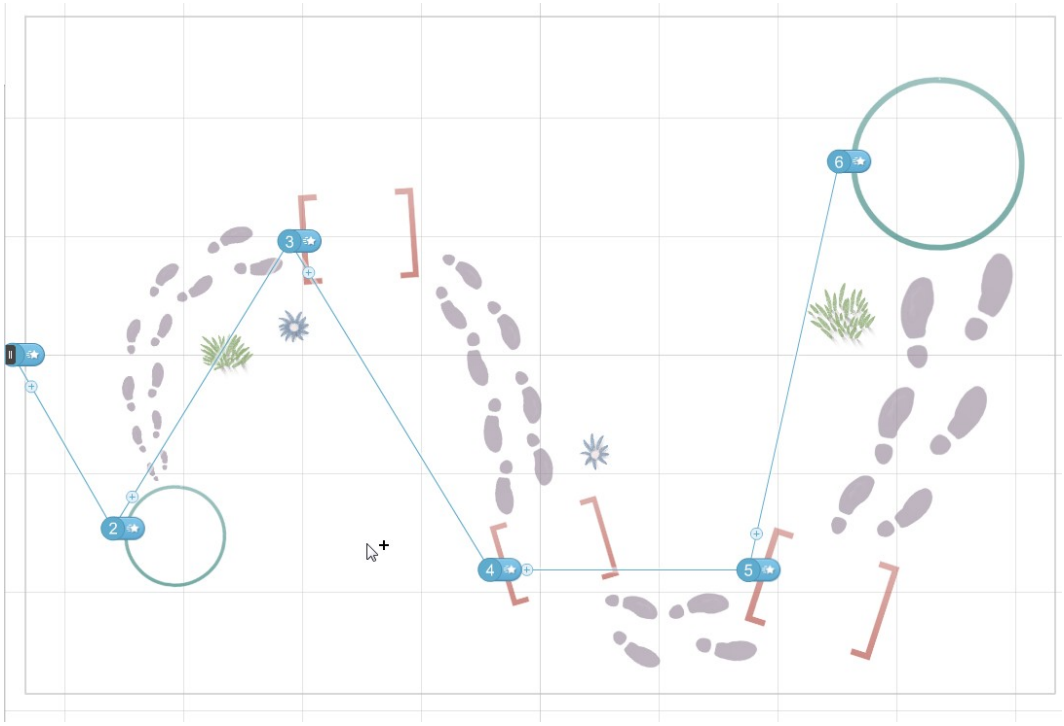
Wybranie slajdu z lewej strony spowoduje zmianę widoku roboczego. Nastąpi najazd kamery analogiczny do tego jaki będzie później obserwowany podczas prezentowania naszego dzieła. Po

zbliżeniu udostępniona zostanie możliwość dodania elementów do aktualnego widoku, jak również ustawienie ich sposobu pojawienia się czy ruchu. Ponadto oprócz animacji pomiędzy widokami możemy animować elementy na konkretnym przystanku (Rysunek 2.3.4).



Rysunek 2.3.4: Prezi – widok wybranego przystanku (źródło: http://prezi.com)

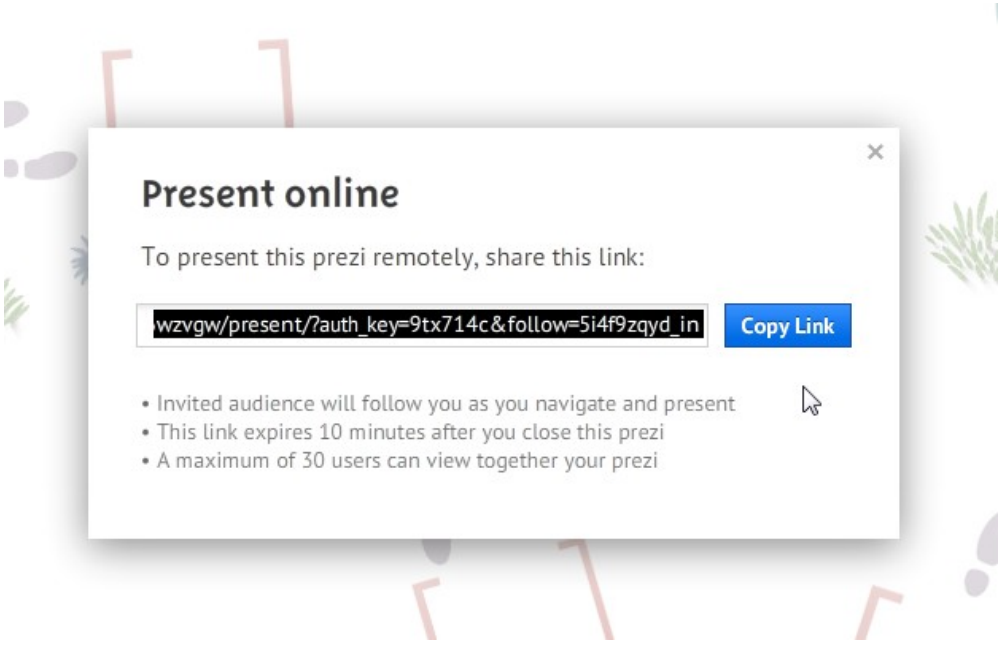
Każdy element obszaru roboczego może być przystankiem, na który zostanie wykonany najazd kamery. Po wyborze opcji edycji ścieżki na widok prezentacji zostanie nałożona linia odzwierciedlająca zachowanie się kamery. Linia utworzona z połączenia przystanków pokazuje przebieg całego pokazu, co daje ogólne pojęcie o kolejności slajdów. Szczegóły przedstawia Rysunek 2.3.5. Dodatkowo po kliknięciu na przystanek możemy edytować najazdy na elementy wewnątrz aktualnego slajdu.



Rysunek 2.3.5: Prezi – edycja kolejności slajdów (źródło: http://prezi.com)

Analogicznie do Zoho Docs ze swoją zdalną prezentacją, Prezi posiada własną wersję omawianej funkcjonalności z wyjątkiem możliwości prowadzenia rozmowy z użytkownikami.

Zapraszanie użytkowników rozwiązane jest za pomocą linku, zawierającego parametry identyfikujące sesję, który możemy rozesłać maksymalnie do 30 osób (Rysunek 2.3.6).

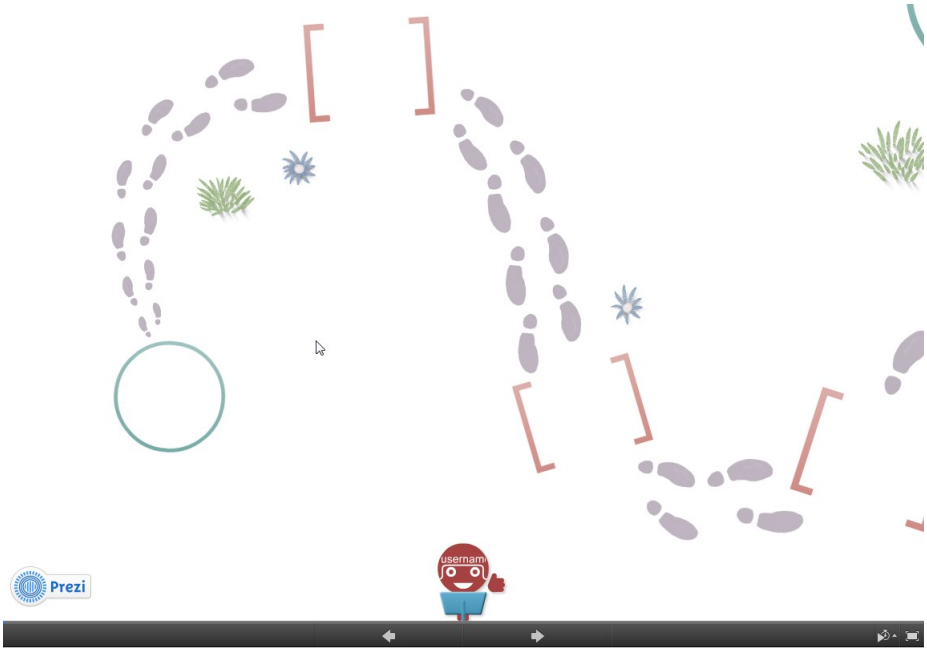


Rysunek 2.3.6: Prezi – zapraszanie do pokazu online (źródło: http://prezi.com)

Użytkownik zaproszony do pokazu może czekać na zmianę widoku przez prowadzącego lub samodzielnie sterować przebiegiem prezentacji we własnym tempie, pozwalając na cofnięcie się oraz przejście dalej w zależności od upodobania (Rysunek 2.3.7).

Rysunek 2.3.7: Prezi – widok pokazu online widziany przez osobę prezentującą (źródło: http://prezi.com)

Rysunek 2.3.8 przedstawia widok, jaki wyświetlony jest osobie sterującej pokazem. Widoczną różnicą w stosunku do pokazu widzianego przez osobę zaproszoną, jest inna grafika w dolnej części prezentacji.



Rysunek 2.3.8: Prezi – widok pokazu online widziany przez osobę zaproszoną (źródło: http://prezi.com)

2.4. Podsumowanie istniejących rozwiązań

Powyżej wymieniono tylko niektóre z obecnych na rynku aplikacji do tworzenia prezentacji. Większość programów zawiera identyczne rozwiązania znane z PowerPoint. Potrafią obsługiwać jego formaty i niekiedy nawet nie odróżniają się interfejsem.

Można zauważyć, że coraz popularniejsze wydaje się być użycie SaaS, jako modelu dystrybucji oprogramowania. Zalety omawianego rozwiązania przekładają się na zadowolenie klienta, który nie musi instalować żadnych programów na swojej maszynie, martwić się o aktualizowanie programu czy doinstalowywanie potrzebnych sterowników lub innych wymaganych bibliotek. Użytkownika końcowego interesuje tylko używanie aplikacji, sprawne wykonanie pracy i efekty jego działań w postaci prezentacji.

Powyższe aplikacje posiadają wiele wspólnych cech wyglądu. Analogicznie do PowerPoint posiadają widok slajdów w lewej części interfejsu oraz po prawej stronie znajduje się aktualny obszar roboczy. Stosowanie podobnego układu pozwala użytkownikowi poczuć się pewniej w nowym środowisku, dzięki czemu może bez zbędnej nauki podjąć pracę nad pokazem slajdów.

Każda z omawianych aplikacji posiada możliwość edycji obiektów na slajdzie, takich jak tekst czy nagłówek, a także dodawania elementów multimedialnych w postaci grafiki czy dźwięku - podstawowe możliwości jakie powinno oferować oprogramowanie do tworzenia prezentacji multimedialnych.

Interesującym rozwiązaniem jest możliwość prowadzenia zdalnego pokazu widoczne w obu aplikacjach. Po wcześniejszym udostępnieniu prezentacji lub wysłaniu zaproszenia na adres Email, możemy kontrolować przebieg i samemu sterować przejściami do kolejnych slajdów. Rozwiązanie zastosowane w Zoho Docs jest rozszerzone o możliwość prowadzenia rozmowy na chacie w trakcie prezentowania materiału. Powyższa funkcjonalność pozwala na interaktywną rozmowę z uczestnikami spotkania, możliwe stało się udzielanie odpowiedzi na pytania, które mogłyby się pojawić podczas referowania. Kolejnym etapem rozwoju zdalnego pokazu mógłby być przekaz z kamery skierowanej na osobę referującą i uczestników.

Pod względem podejścia procesu budowy prezentacji, aplikacja Prezi wyprzedza Zoho Docs i PowerPoint. Slajdy są tylko umownymi scenami, przystankami na ścieżce, która została poprowadzona od widoku całego pokazu do ostatniego określonego elementu. Omawiana funkcja pozwala na zbudowanie ciekawych przejść pomiędzy slajdami, a płynna animacja wraz ze zbliżeniami i oddaleniami kamery urozmaica cały pokaz. Umożliwia olbrzymie możliwości do tworzenia pokazów, które zainteresują odbiorców nie tylko zawartością, ale i sposobem przedstawienia.

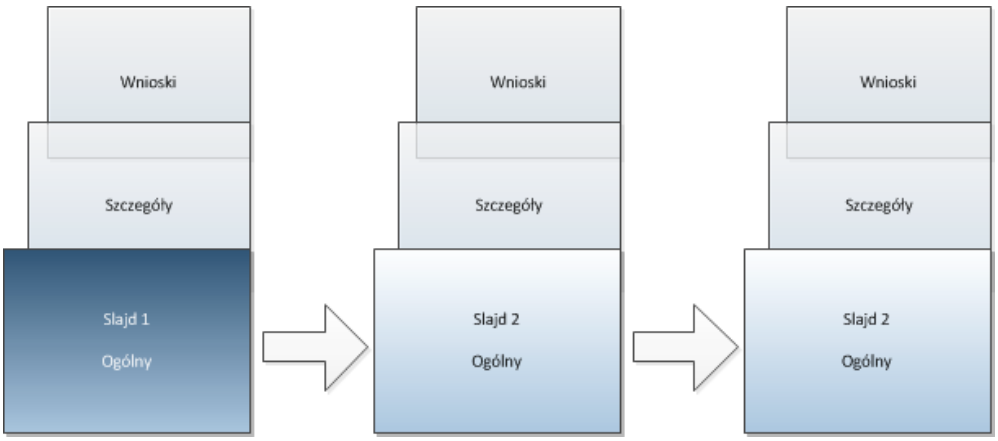
Porównane wyżej oprogramowanie, oferuje wiele możliwości tworzenia pokazów slajdów, jednak żaden w znaczącym stopniu nie upraszcza tego procesu dla prezentacji skierowanych do różnych grup odbiorców. Oczywiście można stworzyć slajdy, których treść będzie podzielona pod względem zainteresowań słuchaczy, jednak wykonanie takiego pokazu jest skomplikowane i pracochłonne. Zoho Docs jako webowy odpowiednik programu PowerPoint nie wnosi żadnych nowych sposobów ustawienia przejść do określonych grup odbiorców. Aplikacja Prezi posiada swój unikalny system przechodzenia między slajdami. Możliwości budowania nieliniowych slajdów wyglądają w powyższym programie zupełnie inaczej. Ścieżka prezentacji jest określona liniowo pomiędzy scenami lub elementami na obszarze roboczym, co uniemożliwia zastosowanie rozwiązań z Zoho Docs lub z aplikacji PowerPoint.

3. Propozycja systemu prezentowania informacji

Zaproponowana koncepcja skupia się na procesie generowania oraz renderowania slajdów, z uwzględnieniem nieliniowości oraz środowiska 3D. Poniższa propozycja jest tylko jedną z wielu dróg, które można obrać w celu ulepszenia procedury budowania prezentacji.

3.1. Grupy docelowe

Jednym z problemów współczesnych prezentacji jest brak ukierunkowania na konkretną grupę docelową. Użytkownik tworzący prezentacje na przykład projektu informatycznego, zmuszony będzie do podzielenia materiału na kilka pokazów. Umieszczenie informacji dla właściciela projektu, programistów, grafików czy innych grup, spowoduje chaos i trudności z odbiorem przygotowanej wypowiedzi. Rysunku 3.1.1 przedstawia przykład slajdów wykorzystujących grupy docelowe.



Rysunek 3.1.1: Przykład zastosowania grup docelowych (źródło: opracowanie własne)

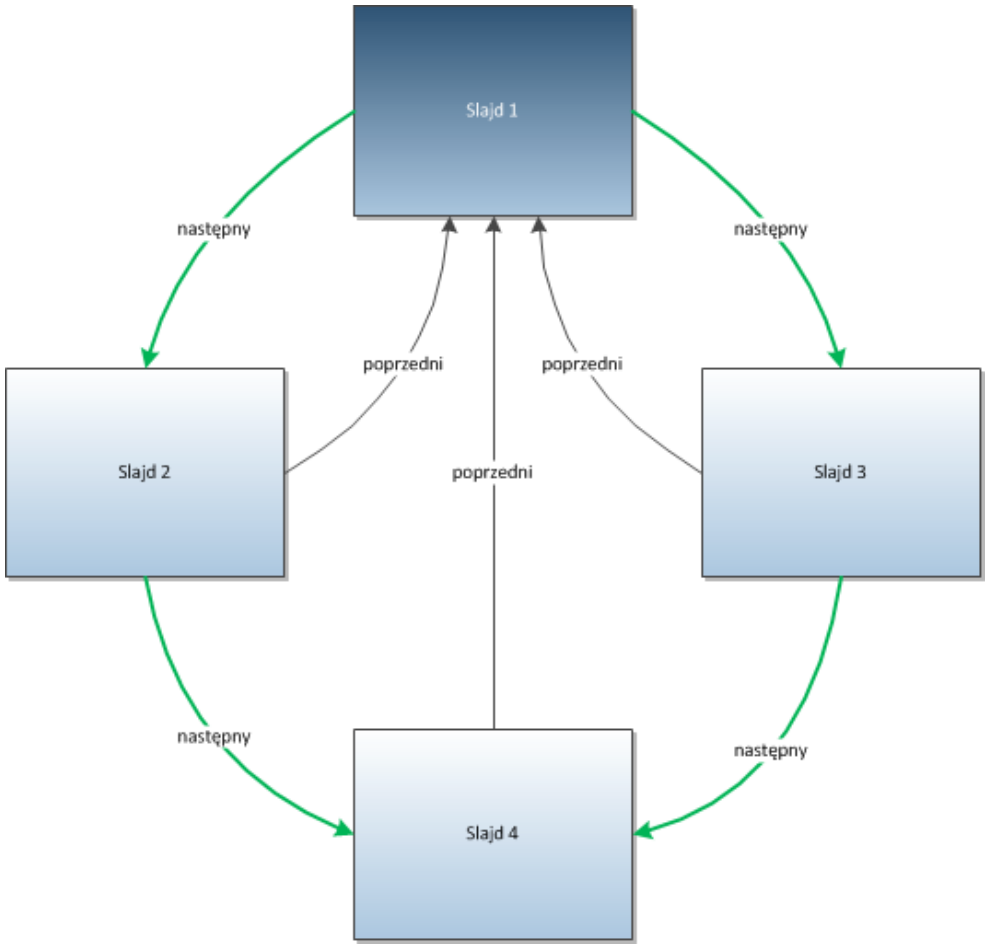
Rozwiązanie, które można zastosować polegałoby na opisaniu slajdu dodatkowo informacją o grupie docelowej, do której jest kierowany. Definicja powinna składać się co najmniej z unikalnego identyfikatora grupy i nazwy, która byłaby potrzebna do wyświetlenia w aplikacji. Omawiane podejście dałoby możliwość wyboru slajdu na podstawie nie tylko kolejności, ale i grupy odbiorców, do których jest skierowany.

Dodatkowo należy zwrócić uwagę na zaprojektowanie odpowiedniego rozmieszczenia slajdów na prezentacji oraz sposób pokazania w edytorze. Należy pamiętać o czytelnym zaprezentowaniu możliwych przejść pomiędzy poszczególnymi scenami. Wykorzystanie atrybutów, takich jak identyfikator slajdu oraz identyfikator grupy, mogą zapewnić informacje, na podstawie których ułożony i sterowany będzie slajd w prezentacji oraz edytorze.

Unikalny identyfikator byłby głównie potrzebny po stronie aplikacji przy tworzeniu obiektów. Nazwa grupy powinna być jedyną właściwością, która interesuje użytkownika oglądającego lub referującego prezentacje.

3.2. Nieliniowość w prezentacji

Określenie kolejności slajdów jest kolejnym zagadnieniem poruszonym w procesie tworzenia nieliniowych prezentacji. Jeżeli każdy slajd posiadałby unikalny identyfikator, możliwe byłoby zastosowanie dwóch dodatkowych cechy: identyfikatora slajdu poprzedniego oraz następnego. Omawiane własności sterowałyby pokazem i umożliwiały nawigację podobną do obecnych rozwiązań. Dodatkową funkcjonalnością, która pojawia się przy zastosowaniu powyższych cech jest możliwość skierowania pokazu do dowolnego miejsca w prezentacji. Możliwe byłoby stworzenie zakończenia wspólnego dla kilku ścieżek pokazu, a także zbudowanie innych slajdów końcowych w zależności od grupy docelowej. Możliwości wynikające z zastosowania omawianych cech przedstawia Rysunek 3.2.1.



Rysunek 3.2.1: Schemat zmiany przebiegu prezentacji (źródło: opracowanie własne)

Rozszerzenie definicji slajdu o dodatkowe cechy zwiększyłoby możliwości oprogramowania, jednak należy zadbać również o odpowiednie rozmieszczenie slajdów w przestrzeni oraz podczas procesu budowania pokazu. Użytkownik powinien bez zbędnego pogłębiania wiedzy być w stanie zrozumieć działanie aplikacji, a następnie stworzyć swoją prezentację.

3.3. Plik prezentacji

Plik, z którego odtwarzana byłaby prezentacja powinien zawierać w informacje o grupach docelowych oraz definicje slajdów składających się na cały pokaz. Rozwiązaniem byłoby zastosowanie dobrze znanego formatu XML, w którym znajdowałyby się sekcje dla grup oraz slajdów.

Każda grupa posiadałaby definicję pól niezbędnych do jej określenia, czyli co najmniej identyfikator i nazwę (Rysunek 3.3.1). Również slajdy powinny być opisane, co najmniej identyfikatorem oraz jednoznacznym numerem slajdu poprzedniego oraz następnego i grupy docelowej. Ponadto w opisie slajdu powinny być opisane elementy określające zawartość slajdu, czyli rozmieszczenie elementów wraz z ich typem i treścią.

```
<?xml version="1.0" encoding="utf-8" ?>
<presentation>
    <groups>
        <group>
            <id>1</id>
            <name>Właściciel projektu</name>
        </group>
        <group>
        <group>
        <group>
    </groups>
    <slides>
</presentation>
```

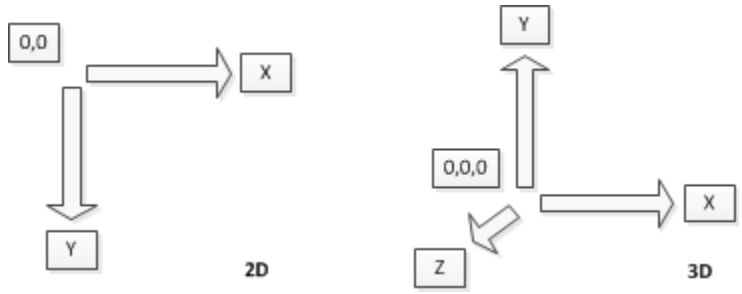
Rysunek 3.3.1: Struktura pliku XML zawierającego definicję pokazu slajdów (źródło: opracowanie własne)

Na potrzeby zaprezentowania koncepcji, slajdy będą posiadały bardzo ubogi zestaw obiektów możliwych do umieszczenia. Dostępne komponenty to tytuł, opis oraz bitmapa - podstawowe składniki slajdu, które powinny znaleźć się w prezentacji multimedialnej.

Każdy element znajdujący się na slajdzie, będzie posiadał pozycję oraz rozmiar. Obiekty tekstowe będą mogły używać kolorów z przestrzeni barw ARGB oraz posiadać definicje ich treści. Bitmapa zostanie określona relatywną ścieżką do pliku, który ma zostać narysowany na powierzchni slajdu.

3.4. Wyświetlanie prezentacji

Pomijając aspekty związane z opisem prezentacji, treści czy kolejności slajdów, ważnym elementem pokazu jest estetyka i sposób wyświetlenia przygotowanego materiału. Aktualne rozwiązania korzystają głównie z systemu dwu-wymiarowego, określając położenie slajdu za pomocą wartości X oraz Y, czyli odległości w poziomie oraz pionie od górnego lewego rogu panelu wyświetlającego slajd (Rysunek 3.4.1).



Rysunek 3.4.1: Porównanie określania położenia w środowisku 2D i 3D (źródło: opracowanie własne)

Proponowane rozwiązanie będzie wykorzystywać środowisko trzy-wymiarowe, które rozszerzy powyższy sposób określania slajdów o wartość głębokości sceny. Omawiane podejście otwiera możliwość zupełnie innego rozmieszczenia slajdów w prezentacji oraz zastosowania ciekawych efektów przejść. Środowisko renderowane za pomocą karty graficznej pozwala zastosować rozbudowane animacje i odciąża procesor w obliczeniach, przekładając obliczenia na GPU.

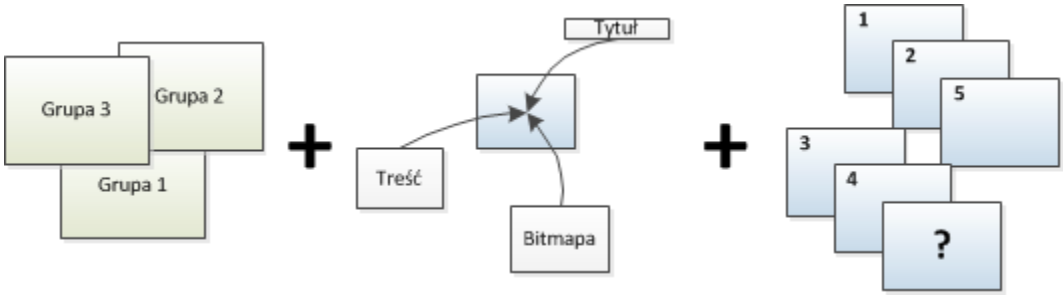
Slajd określony będzie za pomocą płaszczyzny, na której wyrenderowana zostanie bitmapa zawierająca wczytane wcześniej elementy: tekst czy grafika. Następnie slajdy zostaną umieszczone w przestrzeni w miejscach związanych z cechami, które je określają: identyfikatorem, slajdem następnym i poprzednim oraz grupą, do której należą.

Przejście pomiędzy slajdami następować będzie poprzez zmianę położenia kamery, wykonując przy tym płynny ruch ze slajdu aktualnego do wybranego. Dodatkowo animowany będzie sam slajd, co jest demonstracją pokazującą możliwości zastosowania środowiska 3D. Oczywiście istnieje wiele możliwości, które mogą zostać wykorzystane przy uwzględnieniu omawianego sposobu renderowania. Najważniejsza funkcjonalność, która jest dostępna dzięki kartom graficznym - shadery. Użycie shaderów, czyli krótkich programów opisujących właściwości pikseli oraz wierzchołków rysowanych elementów, pozwoliłoby na dodanie dynamicznych efektów świetlnych, wielokrotnego teksturowania, a także zjawiska refrakcji, czy odbić lustrzanych.

3.5. Tworzenie prezentacji

Proces tworzenia pokazu slajdów nie powinien być bardziej skomplikowany od omawianego w rozwiązaniach dostępnych na rynku. Interfejs aplikacji do tworzenia prezentacji musi zawierać elementy, do których użytkownicy są przyzwyczajeni. Widok aktualnego slajdu oraz pokazujący inne slajdy są nieodzowną częścią interfejsu, biorąc pod uwagę nawyki osób tworzących pokazy za pomocą innego oprogramowania.

Dodatkowymi elementami, które należy uwzględnić w aplikacji będą: widok grup, które można będzie przypisać do danego slajdu oraz widok pokazujący całą prezentacje ze wszystkimi slajdami. Ustanowienie odpowiednich przejść między slajdami i kolejności ich wyświetlania powinno być łatwe i nie zmuszać użytkownika do czytania dokumentacji. Widok pokazujący całą prezentacje może więc dodatkowo oferować omawianą funkcjonalność.



Rysunek 3.5.1: Proces tworzenia prezentacji (źródło: opracowanie własne)

Po zapewnieniu powyższych wymagań, takich jak tworzenie grup, slajdu oraz określenie przebiegu prezentacji w połączeniu z intuicyjnym interfejsem proces budowania pokazu slajdów nie powinien sprawiać problemów nawet początkującym użytkownikom (Rysunek 3.5.1).

4. Narzędzia i technologie użyte w pracy

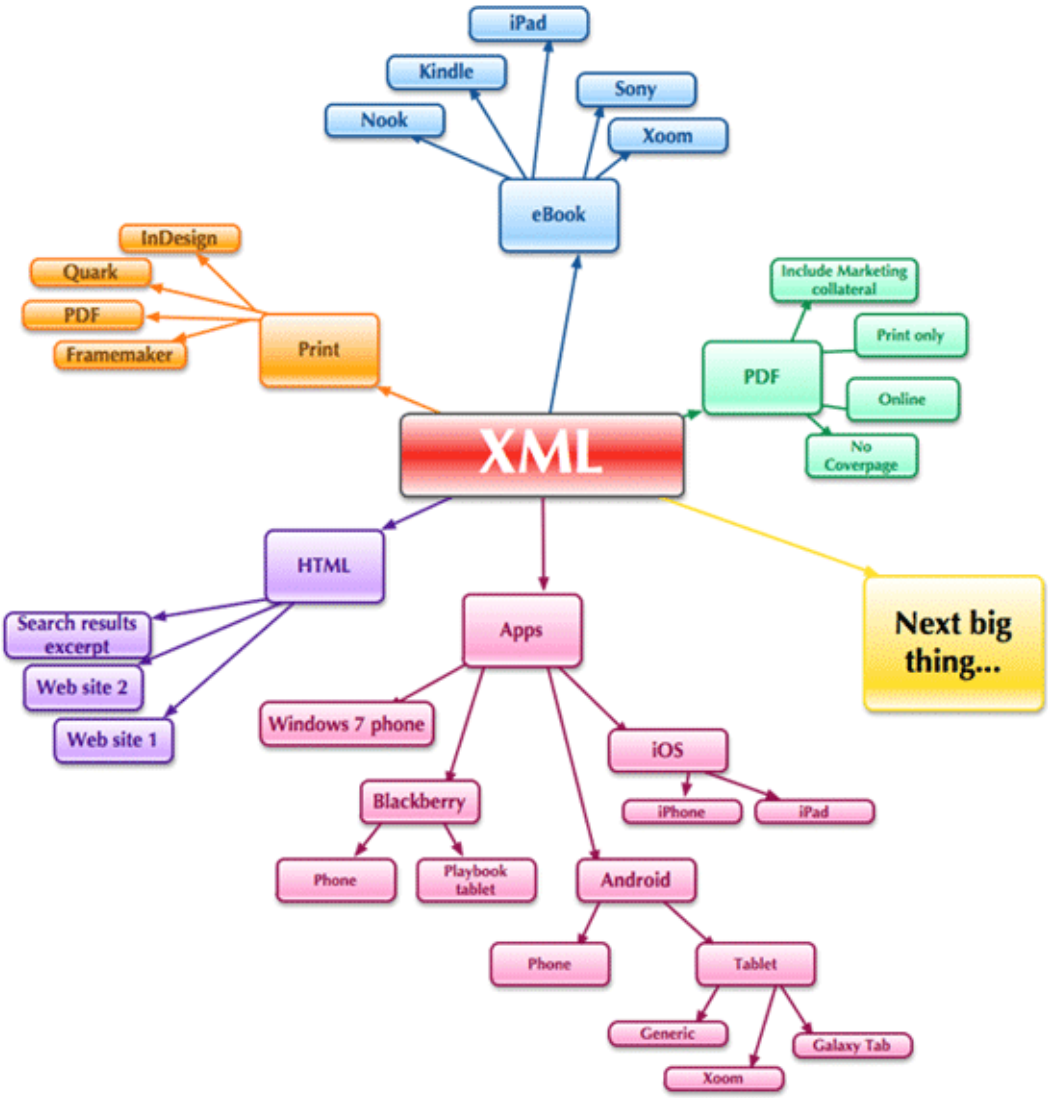
Poniższy rozdział zawiera informacje dotyczące technologii i rozwiązań wykorzystanych do zbudowania prototypów aplikacji, które posłużyły jako demonstracja zaprezentowanej koncepcji.

4.1. XML

XML, czyli rozszerzalny język znaczników (ang. *Extensible Markup Language*) umożliwiający budowanie struktur danych. Jest zestawem reguł pozwalających na projektowanie formatów, na których oparte są dane [3].

Dzięki wykorzystaniu tagów i atrybutów przypomina składnią język HTML. Interpretacja informacji zawartych w pliku XML leży po stronie aplikacji odczytującej. Z powodu wykorzystania tekstowej postaci do reprezentacji danych pliki XML posiadają znacznie większy rozmiar niż binarne zawierające identyczne informacje. Przy obecnych rozmiarach dysków twardych różnica w objętości nie odgrywa znaczenia, a podczas przesyłania plików XML poprzez sieć można zastosować metody kompresji, takie jak Gzip czy Deflate.

Istnieje wiele zastosowań omawianego formatu, począwszy od tworzenia stron internetowych, opisu zasobów, komunikacji czy opisu elementów multimedialnych. Grafika wektorowa (SVG – Scalable Vector Graphics) wykorzystuje XML do bezpośredniego zapisu informacji. Kolejnym przykładem jest zastosowanie w protokołach SOAP, używanych przez usługi sieci telekomunikacyjnych (Rysunek 4.1.1).



Rysunek 4.1.1: Język XML (źródło: http://blog.reallysi.com)

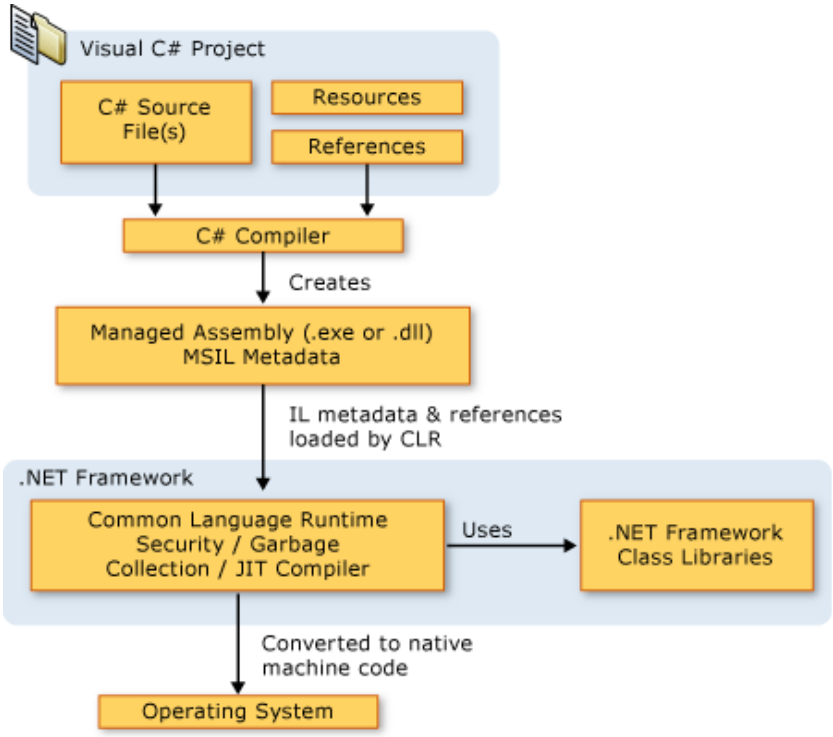
Język XML nie posiada ograniczeń w definiowaniu nowych tagów i atrybutów [4]. Omawiana funkcja pozwala na przechowywanie jakichkolwiek danych. Jest wolny od opłat licencyjnych, co nie generuje dodatkowych kosztów podczas tworzenia oprogramowania. Dodatkowo niezależność od platformy stawia powyższy język wysoko w hierarchii rozwiązań do przechowywania danych. Problem niepowołanego dostępu do danych, czy rozmiaru może być łatwo rozwiązany przez wykorzystanie szyfrowania lub algorytmów kompresujących. Prostota edycji za pomocą dowolnego edytora tekstu jest nieoceniona, gdy dane muszą być szybko zmodyfikowane nawet przez osobę, która nie posiada szerokiej wiedzy informatycznej.

4.2. C#

C# jest nowoczesnym językiem programowania, zaprojektowanym do tworzenia aplikacji wykorzystujących .NET Framework i uruchamianych wewnątrz systemu Common Runtime Environment (CLR). Obiektowy, silnie typowany język wywodzi się od powstałych znacznie wcześniej języków C oraz C++ [5]. Kod programu jest kompilowany do kodu pośredniego, który jest uruchamiany w środowisku .NET. Posiada wbudowany Garbage Collector (odśmiecacz), dzięki któremu programista nie musi martwić się o ręczne zwalnianie pamięci [6].

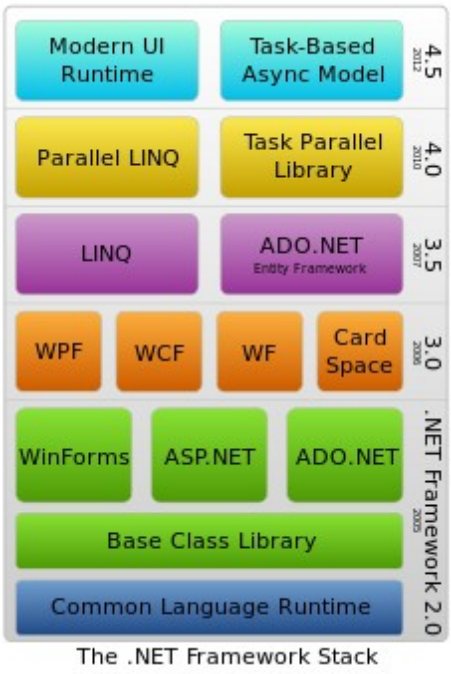
C Sharp jako język obiektowy [7] pozwala na używanie enkapsulacji, dziedziczenia i polimorfizmu. Nie mamy możliwości używania wielodziedziczenia znanego z C++. Możemy dziedziczyć tylko po jednej klasie, jednak implementować wiele interfejsów. Każdy obiekt w języku C# dziedziczy po klasie podstawowej Object.

Aplikacja stworzona w C# jest kompilowana do kodu pośredniego – Intermediate Language (IL), który jest zapisywany w pliku .EXE lub .DLL. Następnie po uruchomieniu aplikacji, CLR wykonuje kompilacje w locie (JIT), która przekształca kod pośredni na wykonywalny [8]. Dodatkowo CLR dba o zwalnianie nieużywanych zasobów, zapewnia obsługę błędów oraz zarządzanie zasobami (Rysunek 4.2.1).



Rysunek 4.2.1: C# - elementy środowiska .NET (źródło: http://msdn.microsoft.com)

Dzięki wykorzystaniu .NET Framework [9], używając języka C# mamy dostęp do biblioteki klas posiadających ponad 5000 publicznych elementów w wersji .NET 2.0. Każda kolejna aktualizacja frameworku rozszerzała powyższą listę o nowe elementy i funkcjonalności oraz poprawiała błędy. Obecna wersja .NET Framework to 4.5. Skupia się głównie na nowych metodach asynchronicznych oraz tworzeniu aplikacji dla Modern UI systemu Windows 8 (Rysunek 4.2.2).



Rysunek 4.2.2: NET Framework – składniki platformy .NET (źródło: http://en.wikipedia.org/wiki/.NET_Framework)

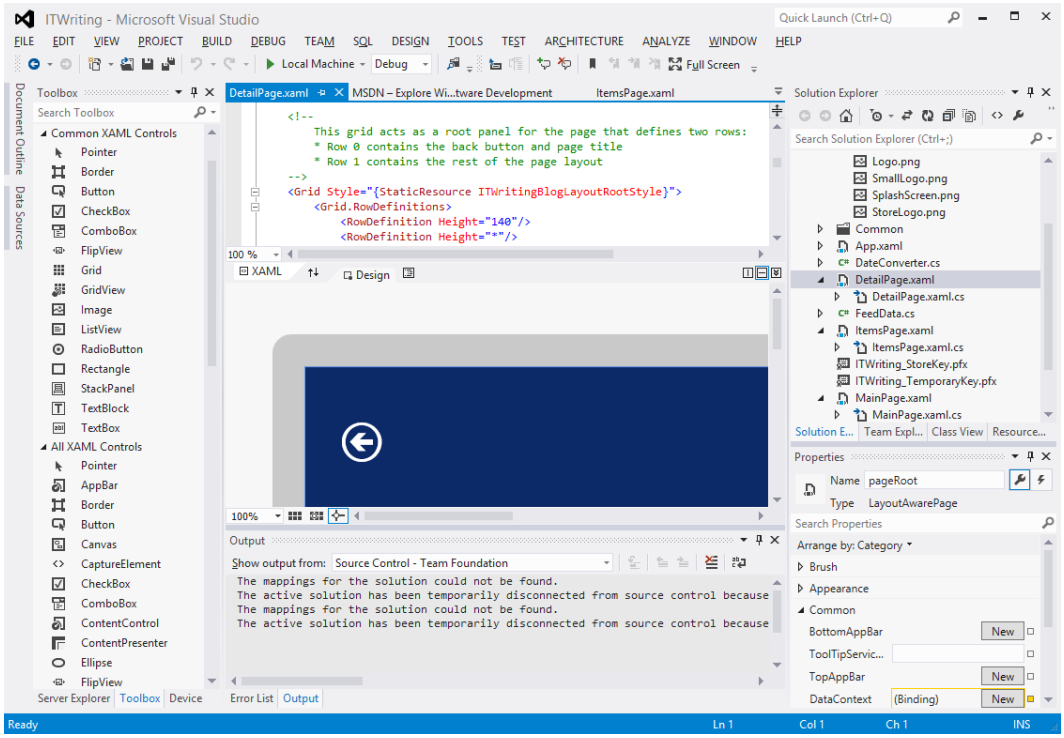
C# jest popularnym i nowoczesnym językiem posiadającym szeroką społeczność, zapewniająca wsparcie. Powstało wiele bibliotek rozszerzających funkcjonalności omawianego rozwiązania na przykład o obsługę innych typów baz niż Microsoftu, jak również inne środowiska uruchomieniowe niż .NET Framework, którego przykładem jest Mono - darmowe środowisko oparte na licencji open source mogące być zainstalowane na systemach innych niż Windows.

4.3. Microsoft Visual Studio

Microsoft Visual Studio jest zintegrowanym środowiskiem programistycznym pozwalającym na tworzenie stron www, aplikacji webowych, okienkowych, konsolowych oraz usług systemu Windows, a także bibliotek .DLL i aplikacji na konsole oraz telefony [10]. Omawiane programy mogą być napisane dowolnym językiem z rodziny .NET w tym C# czy VB.NET.

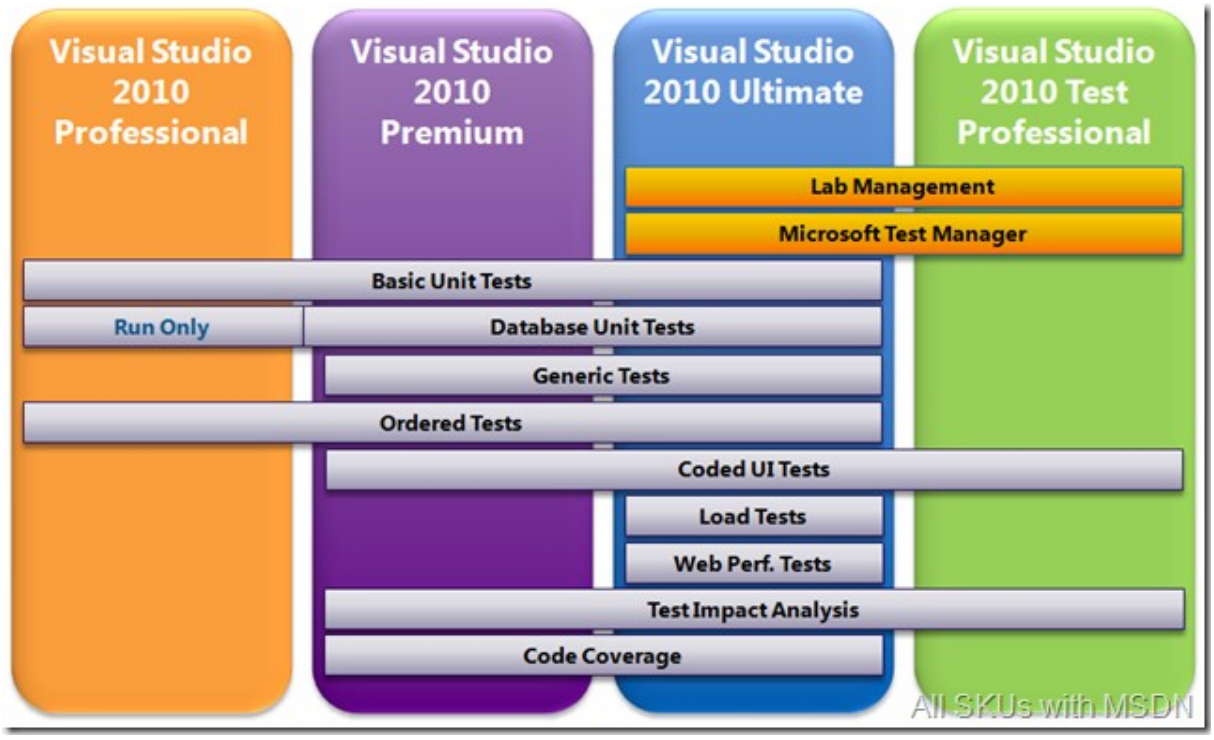
Budowanie aplikacji wspomaga aktywne poprawianie składni zapewnione przez IntelliSense. Powyższa funkcjonalność pozwala na szybsze pisanie kodu poprzez auto-uzupełnianie i oznaczanie błędów. Wbudowany debugger umożliwia uruchomienie aplikacji w środowisku testowym, dającym dostęp do wartości zmiennych, zatrzymywania wykonywania kodu w określonych miejscach, czy wykorzystania parametrów startowych.

Omawiana aplikacja posiada intuicyjny interfejs, z możliwością dowolnego konfigurowania okien wchodzących w skład programu. Powyższe rozwiązanie zapewnia dużą swobodę w dostosowaniu narzędzia do własnych i typów tworzonego oprogramowania (Rysunek 4.3.1).



Rysunek 4.3.1: Visual Studio – widok interfejsu użytkownika (źródło: http://visualstudio2012.wordpress.com/)

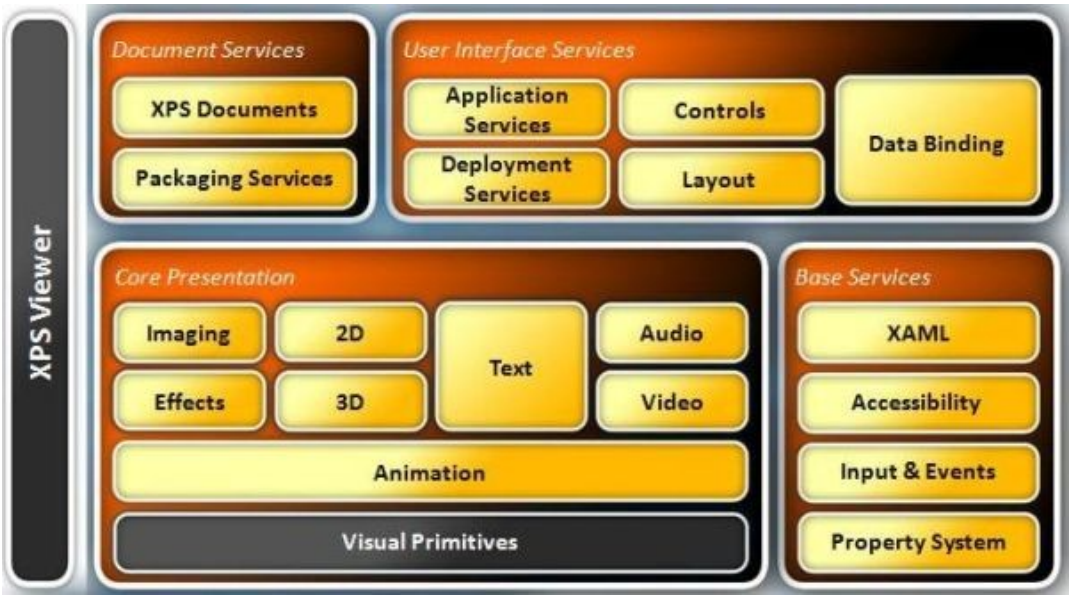
Visual Studio zostało wprowadzone na rynek w wielu edycjach dostosowanych do wymagań klienta. Wersja Express jest darmowa i pozwala na tworzenie aplikacji komercyjnych, dzięki czemu zapewniła sobie wysoką popularność. Kolejne edycje rozszerzają funkcjonalności na przykład o profesjonalne wsparcie pomocy technicznej, czy możliwość obsługi wtyczek zwiększających możliwości tego środowiska (Rysunek 4.3.2).



Rysunek 4.3.2: Porównanie funkcjonalności oferowanych przez różne wersje Visual Studio (źródło: http://devmatter.blogspot.com)

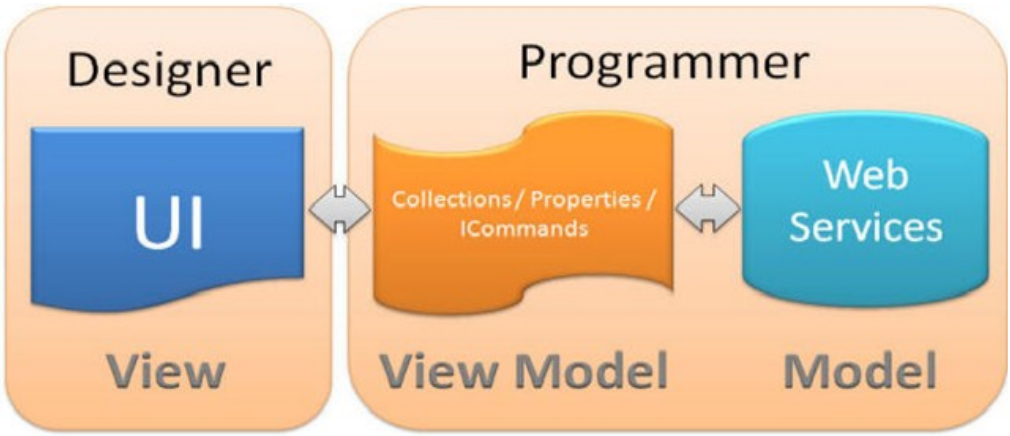
4.4. WPF

Windows Presentation Foundation stanowi podsystem renderowania interfejsu użytkownika dla aplikacji opartych o system Windows [11]. Został dołączony do .NET Framework w wersji 3.0 i porzuca renderowanie GDI na rzecz DirectX. Aplikacje stworzone w oparciu o WPF domyślnie pozwalają oddzielić logikę biznesową od wyglądu. Do tego celu wykorzystano XAML, który jest opartym na XML językiem stworzonym do opisywania elementów interfejsu użytkownika (Rysunek 4.4.1).



Rysunek 4.4.1: Architektura WPF (źródło: http://madhukaudantha.blogspot.com)

WPF do renderowania wykorzystuje bibliotekę DirectX, co pozwala na przerzucenie odpowiedzialności za wyświetlenie aplikacji na procesor graficzny. Dodatkowo omawiane podejście umożliwia budowanie o wiele bardziej zaawansowanego graficznie oprogramowania zawierającego wiele interaktywnych elementów i animacji. Cechą wyróżniającą ten podsystem jest głównie bindowanie modeli do odpowiednich widoków. Dzięki zastosowaniu wzorca projektowego Model-View-ViewModel uzyskano rozdzielenie kodu odpowiedzialnego za logikę w aplikacji od wyglądu interfejsu użytkownika [12] (Rysunek 4.4.2).



Rysunek 4.4.2: Wzorzec projektowy MVVM (źródło: http://abramlimpin.wordpress.com)

W porównaniu do WinForms, WPF oferuje nowsze podejście do generowania aplikacji dzięki zastosowaniu XAML oraz renderowania za pomocą DirectX. Programy stworzone z wykorzystaniem WPF posiadają nowocześniejszy wygląd, a proces ich tworzenia można łatwo rozdzielić na logikę biznesową i tworzenie interfejsu. Problemem aplikacji tworzonych w WPF jest wymóg zainstalowania

co najmniej .NET Framework 3.0 oraz posiadania odpowiedniej karty graficznej, co uniemożliwia uruchomienie omawianych aplikacji na starszych komputerach.

4.5. Microsoft XNA

Powyższy zestaw narzędzi został stworzony w celu ułatwienia pracy programistom gier wideo. Opiera się na .NET Framework i pozwala na uruchamianie aplikacji na systemie Windows, Windows Phone oraz na konsoli Xbox [13]. Silnik wykorzystuje bibliotekę DirectX do renderowania grafiki oraz obsługi kontrolerów i dźwięku. Głównym celem powstania XNA było stworzenie lekkiego środowiska, silnika graficznego i narzędzi, które pozwoliłoby na pisanie prostych gier dla platformy Xbox Live Indie Games – zbioru produktów możliwych do pobrania przez użytkowników konsol (Rysunek 4.5.1).

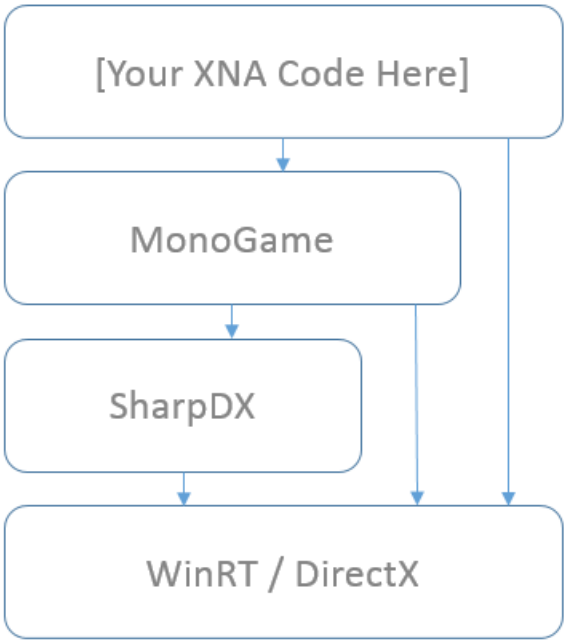
Rysunek 4.5.1: Framework XNA (źródło: http://uditha.wordpress.com)

Framework XNA opiera się na implementacji .Net Framework 2.0, dzięki czemu zapewniona została kompatybilność pomiędzy konsolami Xbox oraz systemami Windows. W tworzonych aplikacjach istnieje możliwość wykorzystania zestawów bibliotek ułatwiających tworzenie gier, pozwalających na łatwe przekształcenia macierzowe, generowanie widoków czy ustawień kamer i światła. Domyślny szablon projektu posiada predefiniowaną strukturę, podzieloną na metody odpowiedzialne za ładowanie zasobów takich jak grafika, dźwięki i czcionki oraz główną pętle gry wraz z metodami odpowiedzialnymi za rysowanie i aktualizacje sceny [14].

XNA Game Studio 4.0 jest najnowszą i ostatnią wersją rozwiązania do tworzenia gier. Firma Microsoft nie planuje rozwijania tego narzędzia, gdyż obecnie pracuje nad szeroko wykorzystywaną

biblioteką DirectX i systemach z rodziny Windows 8. Do tej pory zbudowane aplikacje będą mogły być dalej uruchamiane. W kwietniu 2014 roku zostanie zamknięta możliwość uzyskania tytułu MVP w omawianej technologii. Wspomniana kwestia stanowić będzie ogromny problem dla twórców aplikacji, szczególnie małych firm tworzących gry na telefony z system Windows Phone i konsole Xbox.

Na rynku jednak zaczęło pojawiać się dużo otwarto-źródłowych rozwiązań takich jak MonoGame, implementujących ostatnią wersję XNA i pozwalającą uruchamiać stworzone programy na Mac OS, iOS, Linux czy Android (Rysunek 4.5.2).



Rysunek 4.5.2: MonoGame – multiplatformowy framework wykorzystujący XNA (źródło: http://blogs.msdn.com/b/bobfamiliar/archive/2012/08/01/)

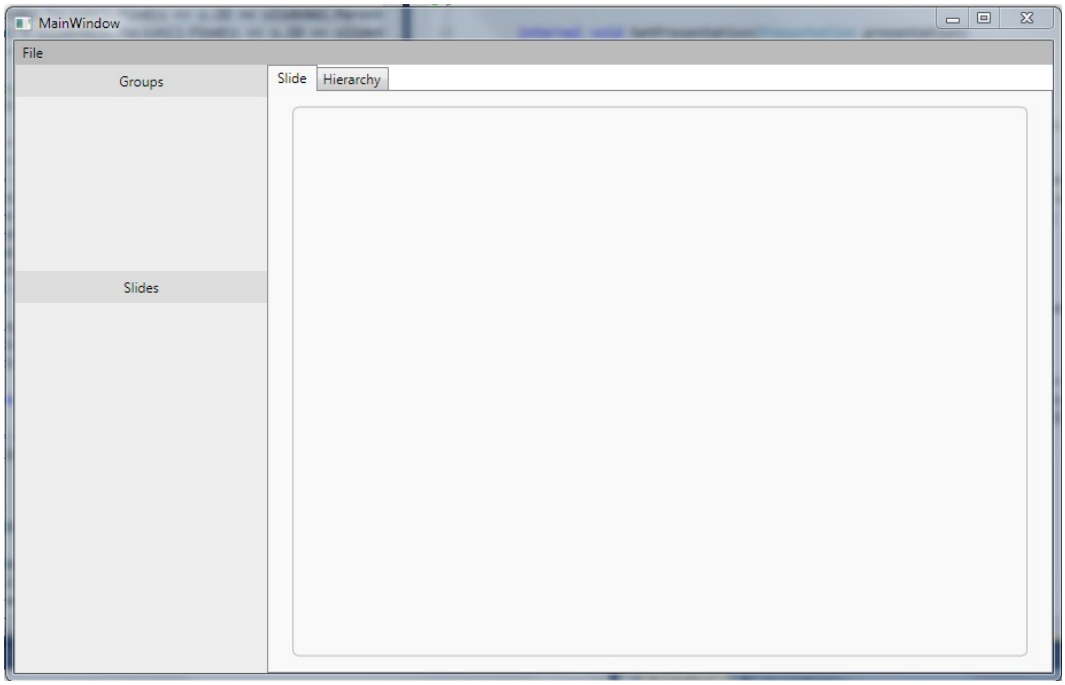
5. Prototyp aplikacji

Prototypy do tworzenia i prezentowanie nieliniowych prezentacji powstały w oparciu o koncepcję przedstawioną w rozdziale trzecim. W niniejszym rozdziale opisane zostały obie aplikacje, czyli Edytor i Prezenter oraz wybrane szczegóły implementacyjne.

5.1. Edytor

Program do tworzenia i edytowania slajdów powstał bazując na Windows Presentation Foundation. Wykorzystanie języka XAML do opisu interfejsu użytkownika pomogło w szybkim ukończeniu prototypu.

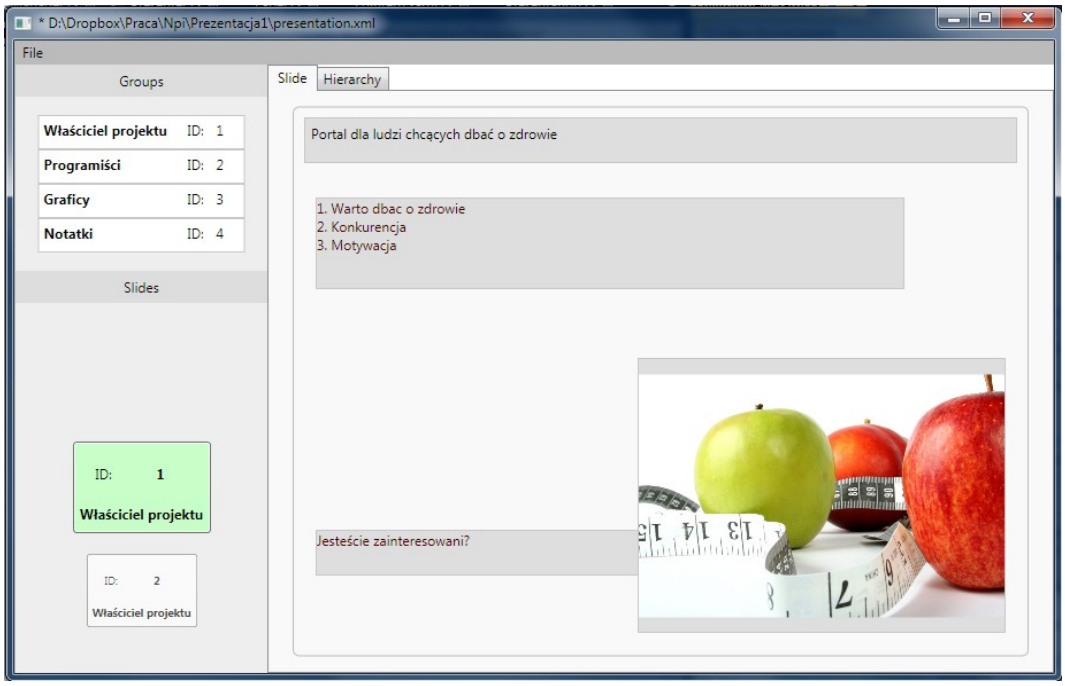
Edytor umożliwia tworzenie i edycję prezentacji zapisywanej do pliku XML, zawierającego definicję rozmieszczenia slajdów i grup docelowych. Okno omawianego programu składa się z menu, w którym znajdują się pozycje niezbędne do utworzenia nowej, otwarcia czy zapisania zbudowanej prezentacji oraz wyjście z programu. Są to podstawowe funkcjonalności zaimplementowane w niniejszej aplikacji. Po lewej stronie znajdują się dwa panele: pierwszy do zarządzania grupami oraz drugi pokazujący otoczenie aktywnego slajdu (Rysunek 5.1.1).



Rysunek 5.1.1: Edytor – widok okna prezentacji (źródło: opracowanie własne)

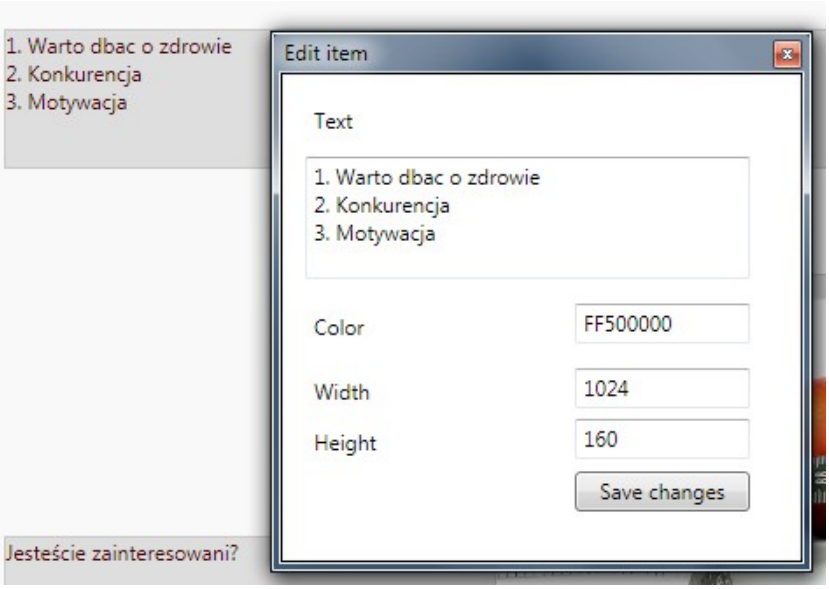
Po wybraniu z menu opcji tworzenia nowej prezentacji domyślnie zostaje wygenerowana pierwsza grupa docelowa oraz slajd. Od tego momentu istnieje możliwość edycji zawartości oraz dodanie nowych elementów. Dodatkowo w panelu Slides zaprezentowane jest aktualne otoczenie wraz z możliwymi przejściami.

Dla celów demonstracyjnych została przygotowana prosta prezentacja zawierająca informacje podzielone na cztery grupy: Właściciela projektu, Programistów, Grafików oraz Notatki. Pokaz posiada 9 slajdów skierowanych do różnych typów odbiorców.



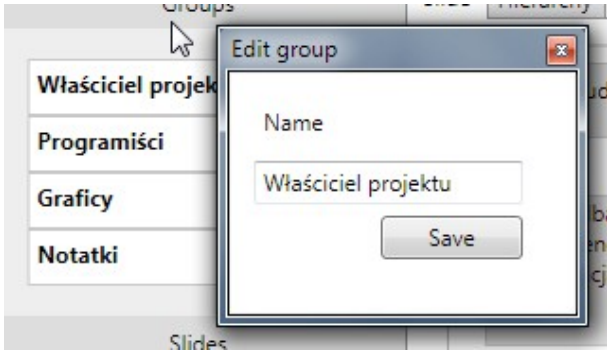
Rysunek 5.1.2: Edytor – widok okna prezentacji po wczytaniu dokumentu (źródło: opracowanie własne)

Aktualnie wybrany slajd jest wyświetlony po prawej stronie aplikacji. Każdy jego element może zostać przeciągnięty w dowolne miejsce. Istnieje również możliwość edycji rozmiaru danej kontrolki powodujące zmianę zawijania tekstu lub wymiarów renderowanej grafiki (Rysunek 5.1.3).



Rysunek 5.1.3: Edytor – widok edycji elementu slajdu (źródło: opracowanie własne)

W analogiczny sposób istnieje możliwość edycji grup docelowych. Wartością dostępną do zmiany jest wyłącznie nazwa wyświetlana (Rysunek 5.1.4).

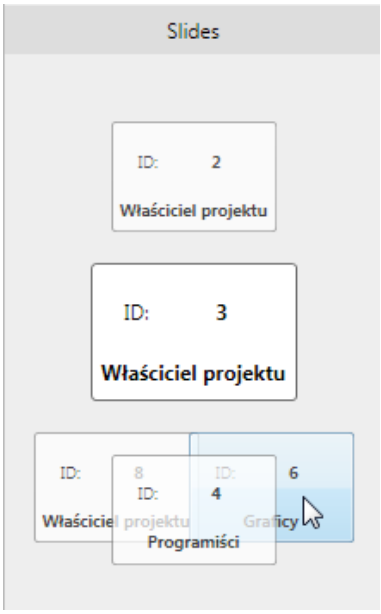


Rysunek 5.1.4: Edytor – widok edycji grupy (źródło: opracowanie własne)

Widok otoczenia slajdów zawiera w swoim centrum aktualnie wybrany element prezentacji. Zmiana aktualizuje widok, demonstrując nowe otoczenie slajdu aktywnego w tym poprzednika powyżej oraz inne możliwe przejścia poniżej. Obiekt o identyfikatorze 1 przedstawia slajd poprzedni, znajdujący się powyżej aktualnie wybranego. Dodatkowo poniżej znajdują się dwie opcje przejścia. Taka sytuacja oznacza, że z aktualnego slajdu istnieje możliwość zmiany grupy docelowej. Kolejnym wyborem może stać się więc slajd 3 lub 9 z grupy docelowej notatki (Rysunek 5.1.5).

Rysunek 5.1.5: Edytor – widok otoczenia slajdu 1 (źródło: opracowanie własne)

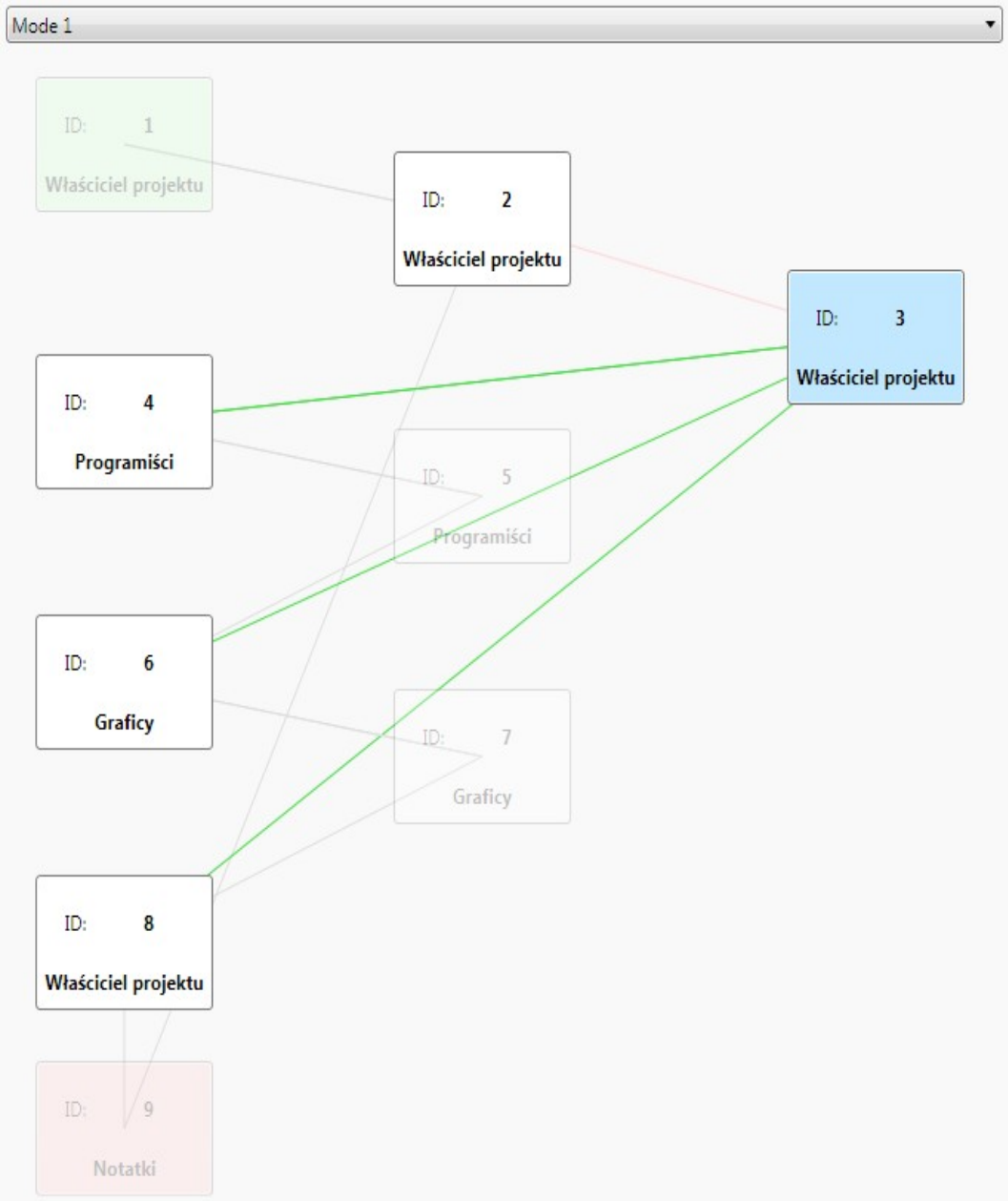
Wybór slajdu 3 spowoduje ustawienie go w centralnej części panelu otoczenia slajdów. Zaktualizowany widok demonstruje możliwości przejścia do kolejnych, z wykorzystaniem ścieżki Właściciela projektu - slajd 8 lub zmiany grupy docelowej na Programistów czy Grafików (Rysunek 5.1.6).



Rysunek 5.1.6: Edytor – widok otoczenia slajdu 2 (źródło: opracowanie własne)

Widok otoczenia jest pomocniczym elementem pokazującym możliwości przejścia z aktualnego slajdu do pozostałych znajdujących się w prezentacji. Do zarządzania wszystkimi elementami powstał specjalny widok odzwierciedlający cały pokaz wraz z możliwymi przejściami. Widok ten nazywa się

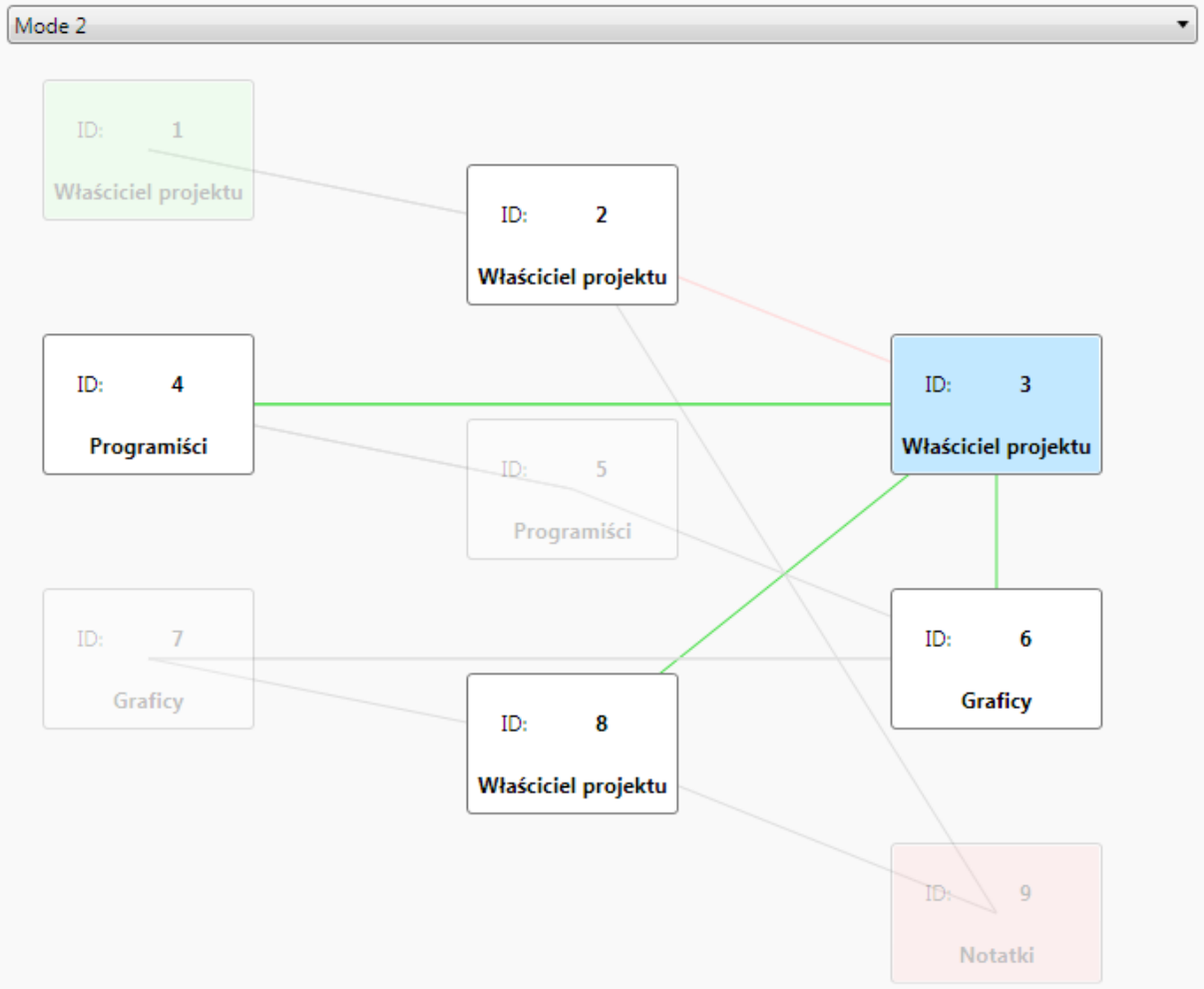
Hierarchy i wywoływany jest poprzez kliknięcie na drugą zakładkę panelu w panelu edycji slajdu (Rysunek 5.1.7)



Rysunek 5.1.7: Edytor – widok hierarchii slajdów, tryb pierwszy (źródło: opracowanie własne)

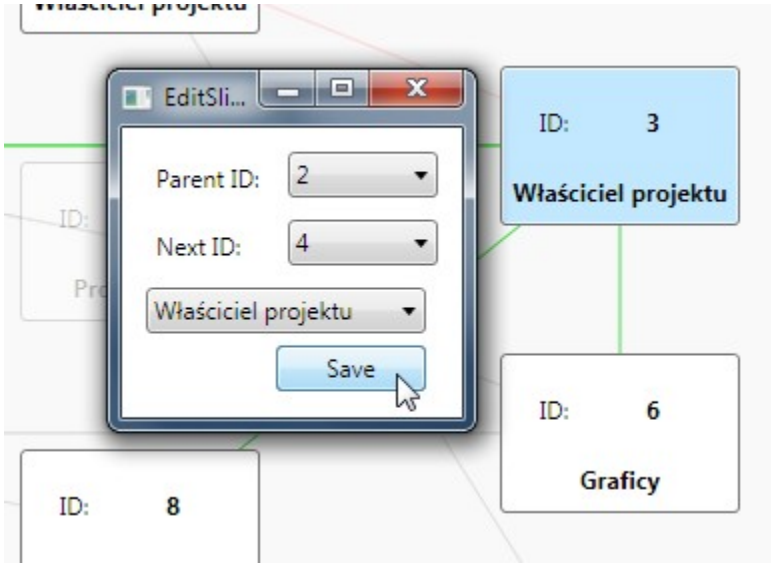
Omawiany widok zawiera wszystkie slajdy znajdujące się w prezentacji. Pokazuje układ wraz z możliwymi przejściami, które są zaznaczone zielonymi liniami. Linie czerwone informują, z których miejsc można dostać się do aktualnie wybranego. Widok otoczenia slajdu pokazuje tylko najbliższe otoczenie i możliwości zmiany ścieżki prowadzenia pokazu, w przeciwieństwie do widoku hierarchii, który pokazuje wszystkie przejścia pomiędzy slajdami. Dodatkowo powyższy widok posiada rozwijane menu, gdzie możliwa jest zmiana trybu wyświetlania. Tryb pierwszy został

zademonstrowany na Rysunku 5.1.7 , natomiast Rysunek 5.1.8 przedstawia ułożenie elementów w trybie drugim.



Rysunek 5.1.8: Edytor – widok hierarchii slajdów, tryb drugi (źródło: opracowanie własne)

Oba widoki wykorzystują kontrolki użytkownika pokazujące slajd z jego identyfikatorem i nazwą grupy. Po kliknięciu otworzy się menu kontekstowe umożliwiające usunięcie elementu z prezentacji lub edycji. Przy wybraniu opcji edycji ukaże się okno pozwalające na zmianę właściwości takich jak identyfikator slajdu poprzedniego, następnego oraz grupy docelowej (Rysunek 5.1.9).



Rysunek 5.1.9: Edytor – okno edycji właściwości slajdu (źródło: opracowanie własne)

Przy zmianie dowolnego z parametrów, widok hierarchii slajdów i otoczenia zostaną zaktualizowane, odzwierciedlając nowy układ wszystkich elementów prezentacji. Analogiczne menu jest wywoływane po kliknięciu na obiekt reprezentujący slajd w oknie najbliższego otoczenia. Omawiana funkcjonalność pozwala na szybką edycje bez wchodzenia do widoku hierarchii.

Można zauważyć, że dwa slajdy w widoku hierarchii są wypełnione dodatkowo kolorami. Zielone wypełnienie reprezentuje element początkowy, który jest wybierany na podstawie identyfikatora rodzica lub własnego, natomiast czerwone – końcowy, określany jest na podstawie unikalnej wartości opisującej numer slajdu, który zostanie po nim wybrany oraz w przypadku, gdy żaden slajd nie wykorzystuje powyżej omawianego jako rodzica.

Prezentacja może zostać zapisana, nastąpi wtedy zapis obiektów do XML, który zostanie umieszczony na dysku. Na podstawie pliku pokaz jest ładowany do Prezentera i wyświetlany. Wykorzystanie XML jako formatu zapisu zostało podyktowane przejrzystością struktury oraz możliwością modyfikacji bez uruchamiania Edytora, w dowolnym programie do edycji tekstu.

5.2. Prezenter

Program do wyświetlania stworzonej prezentacji został napisany w oparciu o XNA. Przekształcenie powyższego frameworku przystosowanego do tworzenia gier pozwoliło na zbudowanie prostego narzędzia do renderowania slajdów wczytanych z wcześniej zdefiniowanego pliku XML.

Po uruchomieniu użytkownik jest proszony o wskazanie pliku do wczytania. Następnie zostaje odtworzona struktura prezentacji i zasilone odpowiednie klasy. Każdy slajd zbudowany jest z powierzchni zawierającej cztery wierzchołki, na której wyświetlana jest tekstura zawierająca zawartość slajdu. Powyższa tekstura jest tworzona dla każdego slajdu w momencie ładowania. Elementy zdefiniowane w pliku XML zostają odczytane, a następnie narysowane na teksturze. Przygotowany slajd jest umieszczany w przestrzeni trójwymiarowej na podstawie cech takich jak identyfikator rodzica i grupy.

Slajd jest obiektem w prezentacji, składającym się ze struktury `Quad` – posiadającej opis wierzchołków, efektu `BasicEffect` – zawierającego teksturę, macierze `View`, `World` oraz `Projection` oraz wektora określającego jego położenie. Opisane elementy są niezbędne do wyświetlenia w aplikacji. Posiada metodę `Draw()` renderującą obiekt na podstawie tekstury efektu oraz struktury `Quad`.

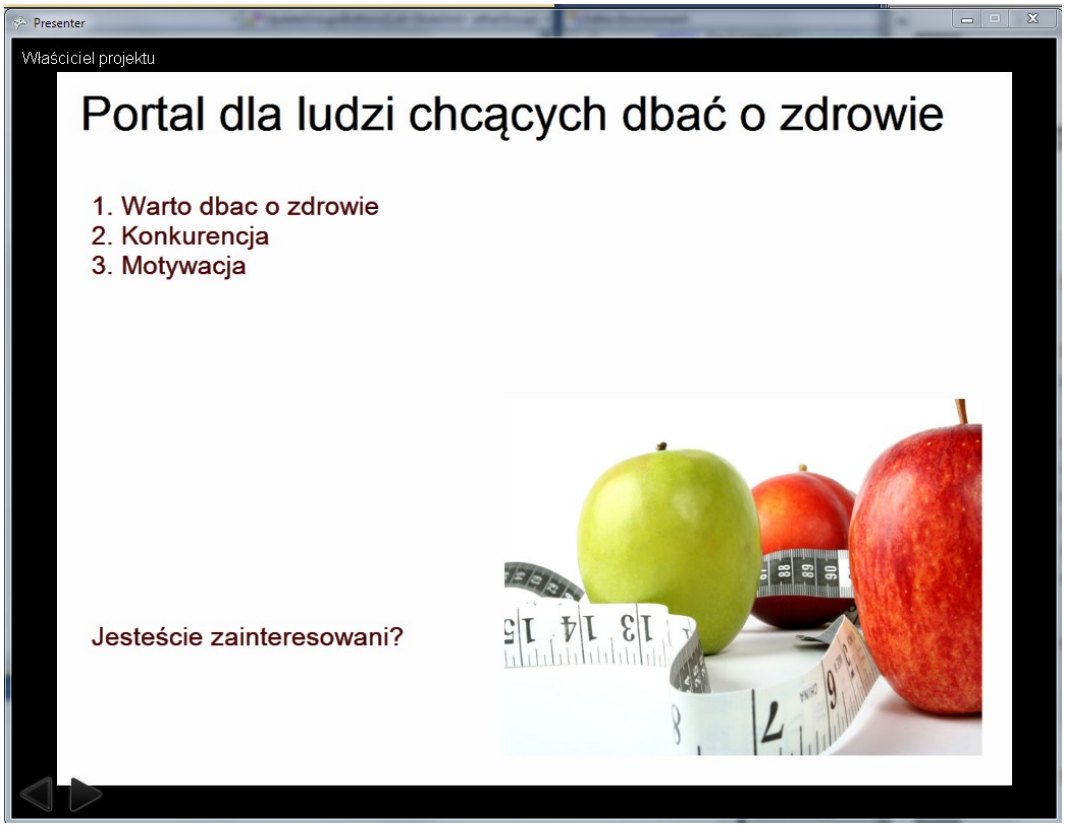
Wykorzystywana tekstura jest rozmiaru 1280x960 pikseli i używa 32 bitowego formatu kolorów z zastosowaniem przezroczystości. Na podstawie bitmapy tworzony jest obiekt typu `Graphics`, który rysuje na teksturze elementy slajdu pobrane z pliku XML. Omawiano proces jest najwolniejszy w aplikacji – nawet z wykorzystaniem rysowania równoległego za pomocą `Pararell.ForEach` trwa on parę sekund.

Pętla programu cyklicznie wywołuje metody `Update()` oraz `Draw()`, które zostały utworzone w momencie tworzenia nowego projektu. Metoda `Update()` jest odpowiedzialna za aktualizacje kamery oraz menu zawierającego przyciski oraz etykiety z nazwą aktualnej grupy i możliwymi do zmiany.

Każdy obiekt klasy `Button` wewnątrz swojej metody `Update()` sprawdza położenie kursora myszy oraz stan lewego przycisku i przy spełnieniu odpowiednich warunków wywołuje akcje.

Aktualizacja kamery polega na zmianie wartości parametru `View` dla efektu znajdującego się wewnątrz slajdu. Parametr `View` jest generowany wbudowaną metodą `Matrix.CreateLookAt()`, tworzącej widok na podstawie pozycji kamery oraz obiektu, na który jest skierowana.

Interfejs aplikacji do prezentowania jest bardzo uproszczony. Posiada przyciski prowadzące do następnego lub poprzedniego slajdu oraz nazwę grupy, do której kierowany jest slajd. Umożliwiają również zmianę aktywnego elementu prezentacji, gdy jest taka możliwość oraz dezaktywują się – wykorzystują inną teksturę kiedy zmiana jest niedostępna (Rysunek 5.2.1).



Rysunek 5.2.1: Prezenter – widok uruchomionej prezentacji (źródło: opracowanie własne)

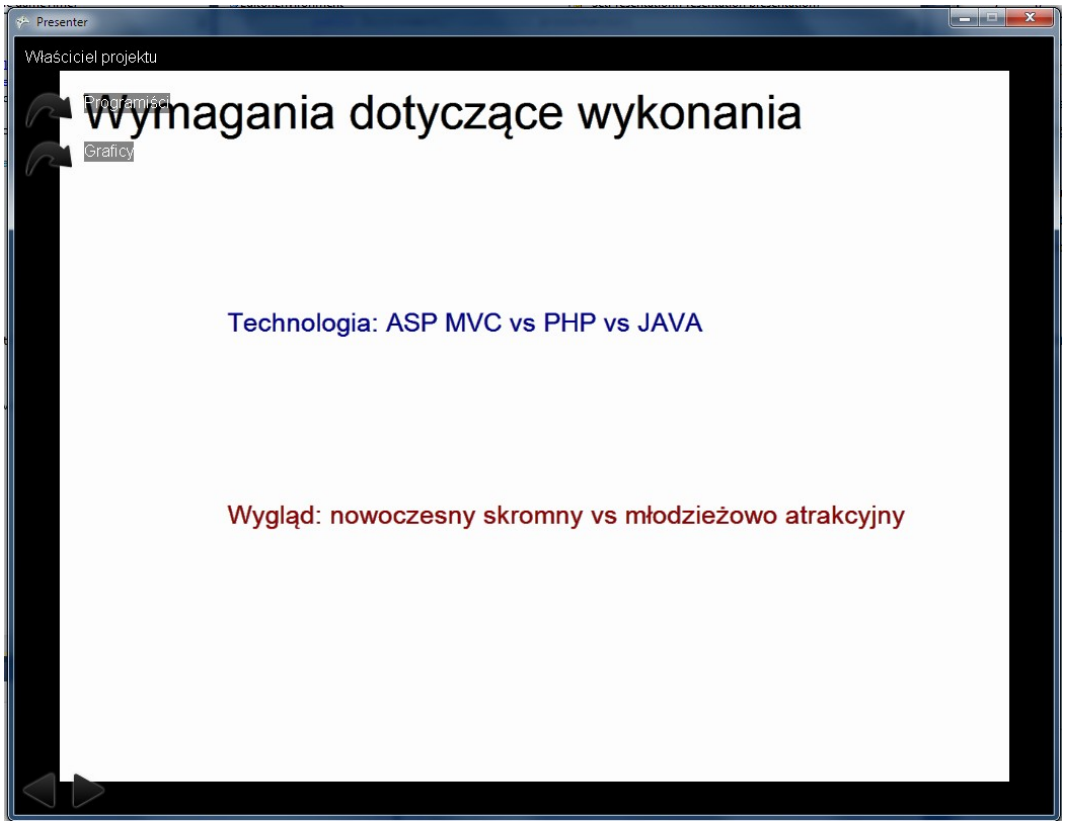
Przejście do kolejnego slajdu następuje poprzez zmianę właściwości `View`, przypisanego do każdego obiektu `Effect`. Dzięki metodzie `Update()` wewnątrz aplikacji, w określonych odstępach czasu wywoływana jest aktualizacja obiektu odpowiedzialnego za płynne, animowane przejście. Dodatkowo powyższy animator obraca elementy prezentacji wokół własnych osi.

Slajd aktualny jest obracany względem osi X, natomiast nowy względem osi Y. Połączenie tych animacji generuje sekwencje, którą przedstawia Rysunek 5.2.2.

Rysunek 5.2.2: Prezenter – animacja przejścia do następnego slajdu (źródło: opracowanie własne)

Dodatkowym efektem wykorzystanym podczas renderowania jest mgła, której parametry ustawione zostały w taki sposób, aby demonstrować głębie przestrzeni. Elementy bardziej oddalone od kamery są ciemniejsze, zanikają. Efekt może wykorzystywać shadery napisane przez użytkownika do bardziej zaawansowanych modyfikacji pikseli lub wierzchołków obiektu. Użycie `BasicEffect` pozwala tylko na prostą konfigurację, czyli ustawienie mgły, oświetlenia czy przezroczystości.

Przy zmianie slajdu oprócz aktualizacji widoku, następuje sprawdzenie, czy nie posiada innych grup docelowych, do których może przejść użytkownik. Jeżeli okaże się, że istnieje możliwość dopasowania treści do odbiorcy, pod etykietą aktualnej grupy pokażą się dostępne opcje: Programiści lub Graficy (Rysunek 5.2.3).



Rysunek 5.2.3: Prezenter – slajd oferujący zmianę grupy docelowej (źródło: opracowanie własne)

Wszystkie slajdy w prezentacji posiadają rozmieszczenie uwarunkowane przez ich identyfikator, rodzica oraz grupę docelową. Obiekt posiadający grupy docelowe jest rozmieszczony w inny sposób niż nieposiadający grup (Rysunek 5.3.4).

Rysunek 5.2.4: Prezenter – animacja przejścia do slajdu z innej grupy docelowej (źródło: opracowanie własne)

Aplikacja do prezentowania posiada podstawowe funkcje takie jak przechodzenie do slajdu następnego i poprzedniego oraz zmiany grup docelowych, gdy zaistnieje możliwość. Demonstruje jak może być wyświetlana prezentacja, która posiada nieliniową strukturę. Zastosowane animacje są

proste, ale pokazują możliwości wykorzystania środowiska 3D do obsługi prezentacji multimedialnych.

5.3. Opis wybranych szczegółów implementacyjnych

Zaproponowane rozwiązanie zostało podzielone na dwie aplikacje: Edytor do tworzenia i edycji prezentacji oraz Prezenter – stworzony do wyświetlania slajdów. Oba prototypy wykorzystują plik XML reprezentujący pokaz oraz definiują slajdy i ich elementy na podstawie wczytanych danych. Prezenter wykorzystuje grafikę 3D do renderowania, co wymusiło zastosowanie odmiennego sposobu rysowania slajdu niż wykorzystanego w Edytorze. Poniżej zostały wybrane i przedstawione ciekawsze elementy zbudowanych prototypów.

5.3.1 Definicja prezentacji na podstawie pliku XML

Prezentacja stworzona za pomocą Edytora jest określona danymi zapisanymi w pliku XML. Wykorzystanie możliwości formatu XML zostało podyktowane łatwością w serializacji oraz deserializacji za pomocą wbudowanej w .NET Framework klasy `XmlSerializer`. Wspomniana klasa pozwala na odczytanie zawartości pliku XML i utworzenie na jego podstawie odpowiedniego obiektu. Umożliwia także zapis stworzonego wcześniej pokazu z powrotem na dysku (Rysunek 5.3.1.1).

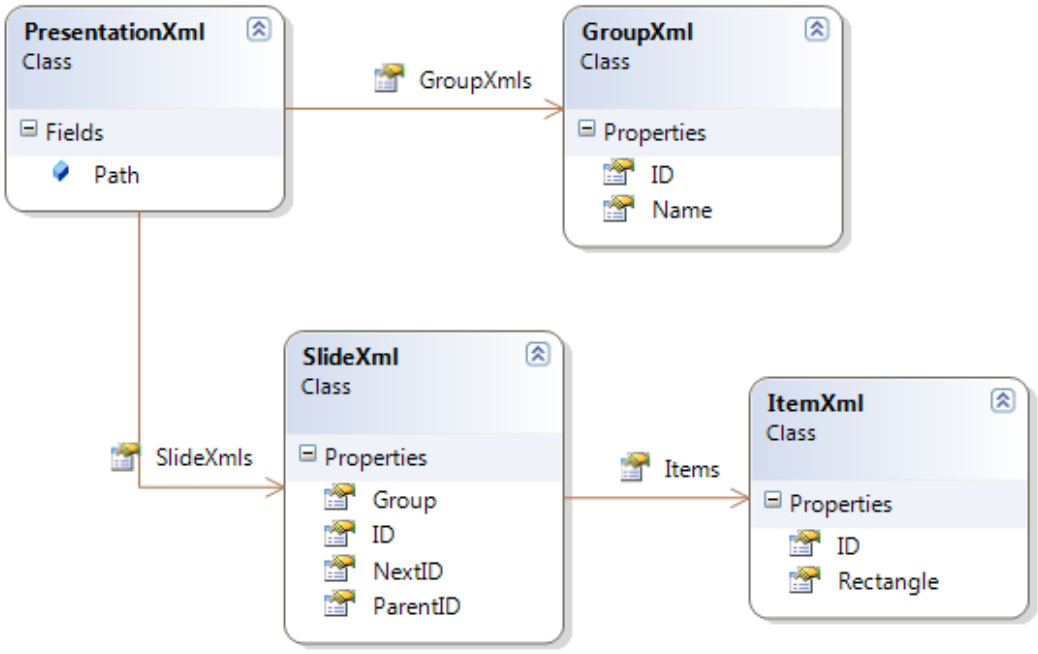
```
public static void ReadXmlFile(string name,out PresentationXml presentation)
{
    XmlSerializer serializer = new XmlSerializer(typeof(PresentationXml));
    using (XmlReader reader = XmlReader.Create(name))
    {
        presentation = (PresentationXml)serializer.Deserialize(reader);
        System.IO.FileInfo info = new System.IO.FileInfo(name);
        presentation.Path = name.Replace( info.Name,"");
    }
    if (presentation == null)
    {
        throw new Exception("presentation is null");
    }
}

public static void SaveXmlFile(string filename, PresentationXml presentation)
{
    XmlSerializer serializer = new XmlSerializer(typeof(PresentationXml));
    using (XmlWriter writer = XmlTextWriter.Create(filename))
    {
        serializer.Serialize(writer, presentation);
    }
}
```

Rysunek 5.3.1.1: Metody związane z odczytywaniem i zapisem prezentacji do pliku XML (źródło: opracowanie własne)

Po wybraniu pliku XML następuje odczyt i deserializacja do klasy `PresentationXml`. Powyższa klasa posiada definicję obiektów znajdujących się w prezentacji. Każda grupa składa się z identyfikatora oraz nazwy. Slajd jest bardziej rozbudowany, ponieważ oprócz identyfikatora posiada dodatkowe ważne atrybuty takie jak identyfikator poprzedniego i następnego slajdu oraz grupy docelowej.

Struktura klasy `PresentationXml` zawiera ścieżkę do prezentacji oraz listę grup oraz slajdów. Można zauważyć, że każdy obiekt, we właściwości `Items`, posiada elementy do renderowania typu `ItemXml`.

Klasa `ItemXml` jest bazową zawierającą identyfikator oraz określenie pozycji i rozmiaru za pomocą czworokąta. Bardziej rozbudowane elementy takie jak tytuł, opis czy bitmapa dziedziczą po klasie `ItemXml` (Rysunek 5.3.1.2).



Rysunek 5.3.1.2: Diagram klasy PresentationXml (źródło: opracowanie własne)

5.3.2 Generowanie slajdu na potrzeby środowiska 3D

Po zasileniu klasy `PresentationXml` danymi z pliku, Prezenter dla każdej definicji slajdu generuje teksturę wraz ze znajdującymi się na niej elementami. Następnie tworzy efekt oraz czworokąt reprezentujący slajd. Omawiany proces jest dosyć obciążający zatem aplikacja wykorzystuje klasę `ConcurrentBag` do przechowywania oraz przetwarzania równoległego za pomocą klasy `Parallel` (Rysunek 5.3.2.1).

```
using (Bitmap bitmap = new Bitmap(width, height,
    System.Drawing.Imaging.PixelFormat.Format32bppPArgb))
{
    Graphics g = Graphics.FromImage(bitmap);
    g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceOver;
    g.CompositingQuality = System.Drawing.Drawing2D.CompositingQuality.HighQuality;
    g = DrawBackground(presentation, slideXml, g);

    //renderowanie elementow slajdu
    foreach (ItemXml i in slideXml.Items)
    {
        if (i.GetType() == typeof(TitleXml))
        {
            g = DrawTitle(i, g, fonts);
        }
        else if (i.GetType() == typeof(DescriptionXml))
        {
            g = DrawDescription(i, g, fonts);
        }
        else if (i.GetType() == typeof(ImageXml))
        {
            g = DrawImage(i, presentation, g);
        }
    }

    //zapis do tekstury
    using (MemoryStream ms = new MemoryStream())
    {
        bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
        texture = Texture2D.FromStream(presenter.GraphicsDevice, ms);
    }
}
```

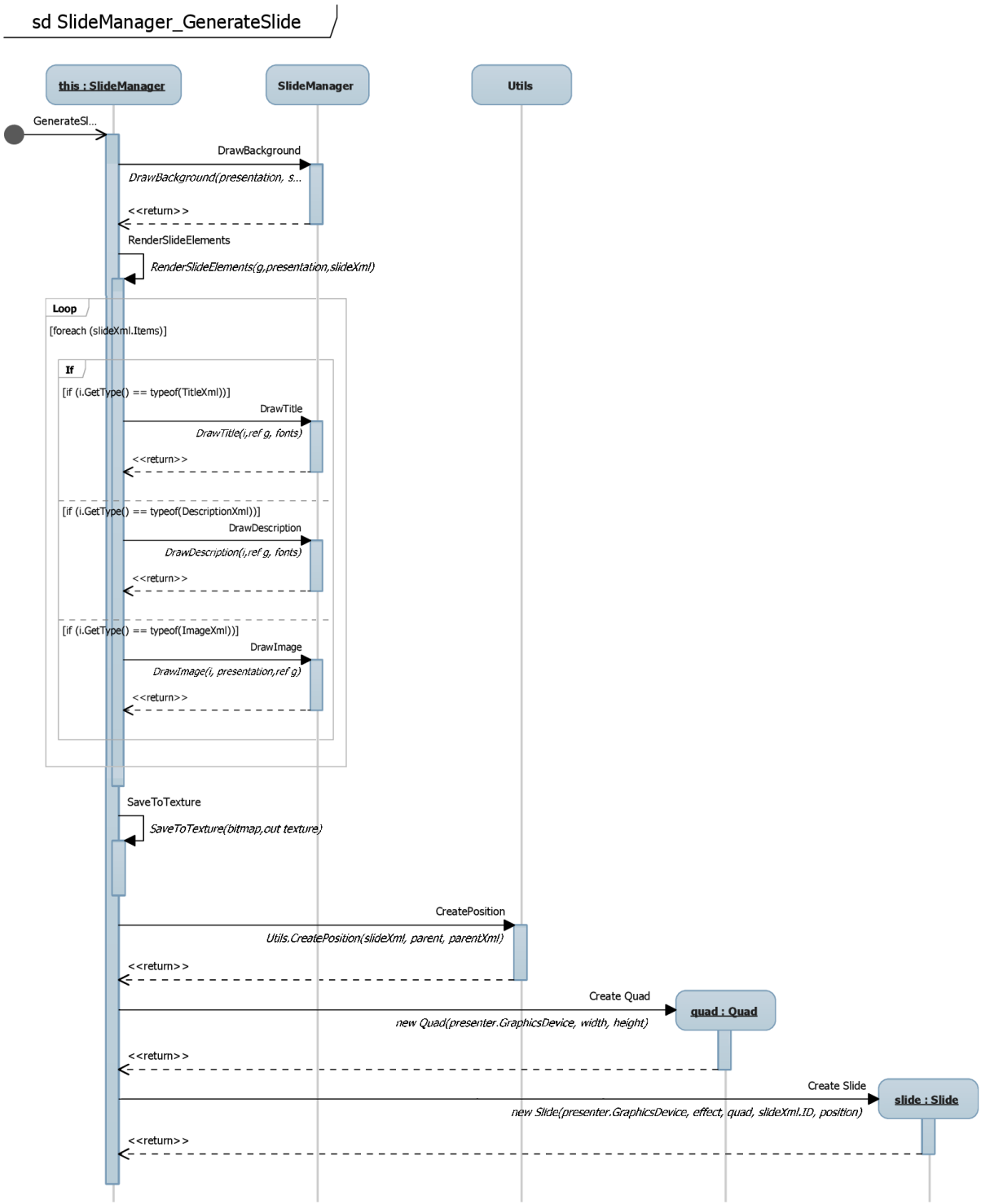
Rysunek 5.3.2.1: Rysowanie elementów slajdu na bitmapie (źródło: opracowanie własne)

Po zapisaniu bitmapy do tekstury, możliwe jest przypisanie jej do obiektu `BasicEffect`, odpowiedzialnego za wyświetlanie slajdu. Oprócz tekstury należy utworzyć odpowiednie macierze, reprezentujące świat, projekcję oraz widok, żeby klasa `BasicEffect` spełniła swoją funkcję. Funkcją macierzy świata jest określenie położenia slajdu w przestrzeni. Najprostsze do omawianego celu było wykorzystanie wbudowanej metody klasy `Matrix – CreateTranslation()`, przyjmującej trzy-składnikowy wektor pozycji w przestrzeni (Rysunek 5.3.2.2).

```
//Generowanie pozycji na podstawie slajdu
Slide parent = slides.ToList().Find(s => s.ID == slideXml.ParentID);
SlideXml parentXml = slideXmls.ToList().Find(s => s.ID == slideXml.ParentID);
Vector3 position = Utils.CreatePosition(slideXml, parent, parentXml);

//tworzenie efektu, z tekstura i polozeniem
BasicEffect effect = new BasicEffect(presenter.GraphicsDevice);

Matrix world = Matrix.CreateTranslation(position);
effect.World = world;
effect.View = view;
effect.Projection = projection;
effect.Texture = texture;
effect.TextureEnabled = true;
effect.FogEnabled = true;
effect.FogStart = 2;
effect.FogEnd = 5;

effect.LightingEnabled = true;
effect.DirectionalLight0.DiffuseColor = new Vector3(1, 1, 1);
effect.DirectionalLight0.Direction = new Vector3(0, 0, -1);

Quad quad = new Quad(presenter.GraphicsDevice, width, height);

Slide slide = new Slide(presenter.GraphicsDevice, effect, quad, slideXml.ID, position);
slides.Add(slide);
```

Rysunek 5.3.2.2: Tworzenie slajdu (źródło: opracowanie własne)

Na podstawie pozycji tworzony jest obiekt `World` typu `Matrix`, który został wykorzystany do zasilenia klasy `BasicEffect`. Dodatkowo ustawiono podstawowe oświetlenie oraz mgłę powodująca zanikanie slajdów oddalonych od kamery. Następnie został utworzony czworokąt, na którym wyświetlana będzie tekstura. Ostatnim krokiem było wygenerowanie obiektu typu `Slide`, składającego się z wcześniejszych elementów, umieszczonego w liście wszystkich slajdów prezentacji (Rysunek 5.3.2.3).



Rysunek 5.3.2.3: Diagram sekwencji tworzenia slajdu (źródło: opracowanie własne)

5.3.3 Animacja slajdów

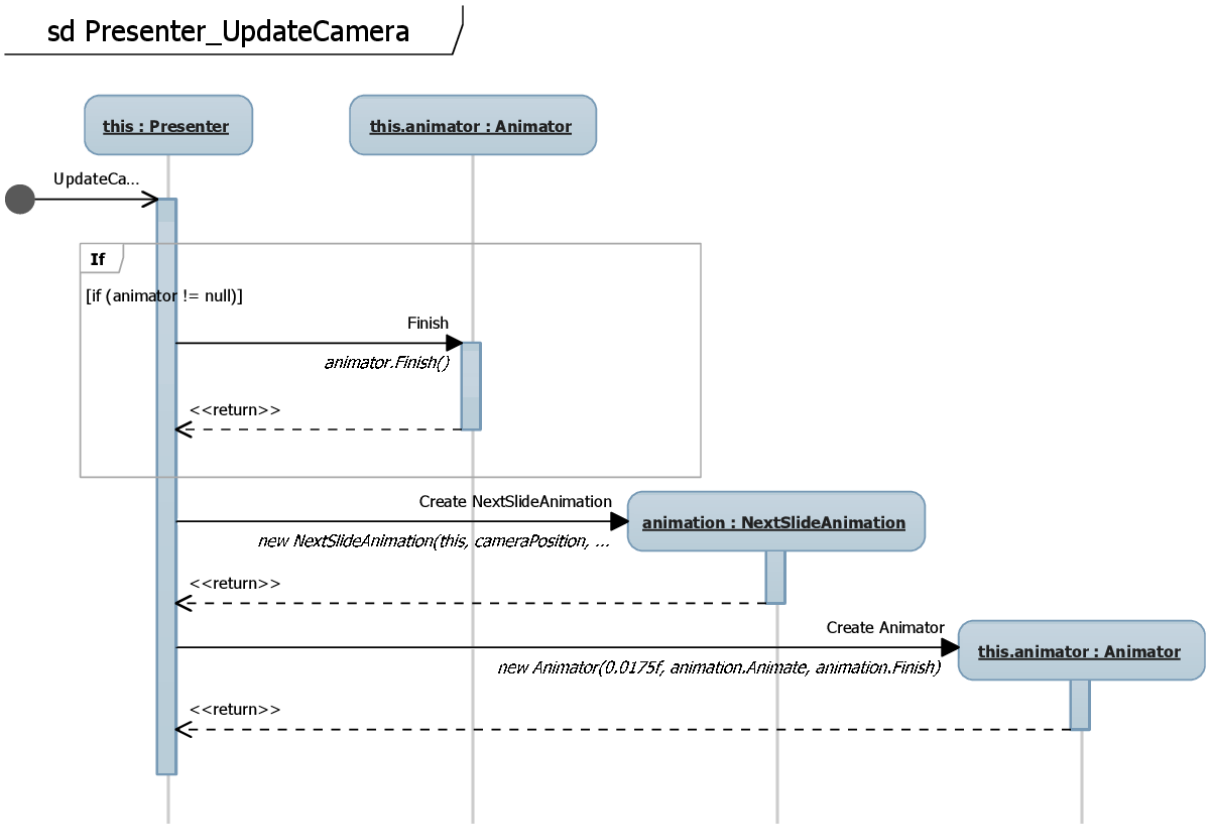
Przy zmianie aktywnego slajdu, Prezenter odgrywa animację, która sprawia wrażenie przesuwania się kamery. Dodatkowo elementy prezentacji obracają się częściowo wokół

odpowiednich osi, co razem daje interesujący efekt przejścia. Osiągnięcie efektu animacji opiera się na aktualizacji macierzy widoku dla slajdów w prezentacji. Widok utworzony jest za pomocą metody `CreateLookAt()`, znajdującej się w klasie `Matrix`. Przyjmowane parametry to trzy wektory 3-komponentowe: pozycja kamery, pozycja celu oraz określenie położenia górnej części widoku. Klasa pomocnicza `NextSlideAnimation` zawiera definicję powyższej animacji, która wraz z klasą `Animator` odpowiedzialna jest za animowanie przejść między slajdami (Rysunek 5.3.3.1).

```
internal void UpdateCamera(Matrix matrix, Slide activeSlide, Slide newActiveSlide)
{
    if (animator != null)
    {
        animator.Finish();
        animator = null;
    }
    IViewAnimation animation = new NextSlideAnimation(this, cameraPosition,
        cameraTarget, matrix,
        activeSlide, newActiveSlide);
    animator = new Animator(0.0175f, animation.Animate, animation.Finish);
}
```

Rysunek 5.3.3.1: Metoda ustawiająca animację przejścia między slajdami (źródło: opracowanie własne)

Metoda `UpdateCamera()` na podstawie slajdu poprzedniego oraz nowego, odpowiedzialna jest za animację przejścia. Jeśli aktualna animacja nie została ukończona zostaje wymuszone zakończenie poprzez wywołanie metody `Finish()` na obiekcie typu `Animator`. Następnie zostaje utworzona nowa animacja za pomocą klasy `NextSlideAnimation`, której dwie główne metody czyli `Animate()` oraz `Finish()` są przekazywane do animatora. Klasa `Animator` w określonych odstępach czasu, wykorzystując wbudowaną w szkielet projektu metodę `Update()`, zwiększa parametr stanu animacji o 0.0175 jednostek i wywołuje metodę `Animate()`. Omawiana zmienna jest odpowiedzialna za prędkość przejścia pomiędzy slajdami (Rysunek 5.3.3.2).



Rysunek 5.3.3.2: Diagram sekwencji metody UpdateCamera() (źródło: opracowanie własne)

Metoda `Finish()` wywoływana jest, gdy wartość zmiennej `progress` zostanie ustawiona na 1.0. Powyższa wartość oznacza koniec animacji. Wewnątrz metody `Animate()` znajduje się kod odpowiedzialny za obracanie slajdu aktywnego oraz następnego. Omawiany efekt został osiągnięty za pomocą przekształcenia macierzy `World`. Uzyskanie efektu przejścia wymaga aktualizacji zmiennej `view`. Wspomniana zmienna ta odpowiedzialna jest za macierz `View`, określonej tak samo jak `World`, wewnątrz klasy `BasicEffect` przypisanej do obiektu `Slide` (Rysunek 5.3.3.3).

```
public void Animate(float progress)
{
    Vector3 cameraPosition = Vector3.SmoothStep(this.cameraPosition,
        this.cameraPosition + world.Translation, progress);
    Vector3 cameraTarget = Vector3.SmoothStep(this.cameraTarget,
        this.cameraTarget + world.Translation, progress);

    activeSlide.World = Matrix.Identity * Matrix.CreateRotationX(progress * 4);
    activeSlide.World *= Matrix.CreateTranslation(activeSlide.Position);

    newActiveSlide.World = Matrix.Identity * Matrix.CreateRotationY(-(1 - progress) * 2);
    newActiveSlide.World *= Matrix.CreateTranslation(newActiveSlide.Position);

    view = Matrix.CreateLookAt(cameraPosition, cameraTarget, Vector3.Up);
    updateView.UpdateView(view);
}

public void Finish()
{
    updateView.UpdateCameraPosition(cameraPosition + world.Translation);
    updateView.UpdateCameraTarget(cameraTarget + world.Translation);
}
```

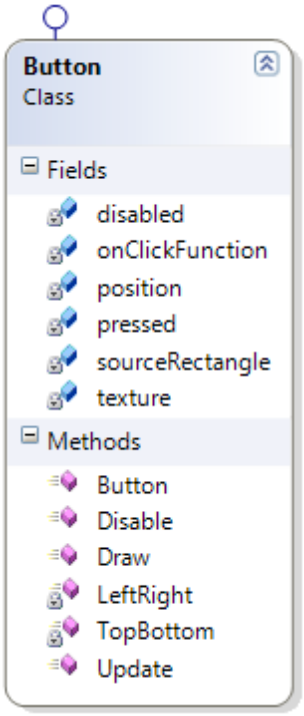
Rysunek 5.3.3.3: Szczegóły metod Animate() oraz Finish() (źródło: opracowanie własne)

Na podstawie postępu animacji, metoda `SmoothStep` klasy `Vector3` generuje nowy wektor znajdujący się pomiędzy początkowym a końcowym. Następnie tworzony jest odświeżony widok przy pomocy metody `CreateLookAt()`, który zostaje ustawiony dla każdego slajdu w prezentacji.

5.3.4 Przyciski w pokazie slajdów

W aplikacji do prezentowania wymagane było utworzenie przycisków do przechodzenia między slajdami oraz zmiany grupy docelowej. Posiadają własne tekstury, które renderowane są jako ostatnie dzięki czemu nie zostają zasłonięte przez slajdy. Mogą zostać wyłączone co spowoduje zmianę ich wyglądu. Omawiana funkcjonalność jest potrzebna, aby pokazać na przykład brak możliwości cofnięcia się ze slajdu początkowego.

Przyciski stanowią instancję klasy `Button`, która wewnątrz metody `Draw()` rysuje teksturę na odpowiedniej pozycji. Dodatkowym parametrem wykorzystywanym podczas wykonania wspomnianej metody jest czworokąt, na podstawie którego pobierany jest wycinek tekstury. Omawiane rozwiązanie pozwala na renderowanie różnych wyglądów z jednej tekstury. Zmienna `sourceRectangle` domyślnie wskazuje na pozycję 0,0 oraz posiada wymiary identyczne do wymiarów przycisku (Rysunek 5.3.4.1).



Rysunek 5.3.4.1: Diagram klasy Button (źródło: opracowanie własne)

Wcześniej przygotowana tekstura zbudowana jest z dwóch bitmap znajdujących się jedna pod drugą. Pierwsza reprezentuje wygląd aktywnego przycisku, natomiast druga nieaktywnego. Po dezaktywacji za pomocą metody `Disable()`, następuje aktualizacja zmiennej `sourceRectangle` wykorzystywanej do określenia obszaru bitmapy. Pozycja Y zostaje zwiększona o wysokość przycisku. Omawiany zabieg spowoduje, że do rysowania brana pod uwagę będzie druga bitmapa (Rysunek 5.3.4.2).

```
public void Disable(bool disable)
{
    sourceRectangle.Y = sourceRectangle.Height * (disable ? 1 : 0);
    this.disabled = disable;
}
```

Rysunek 5.3.4.2: Metoda Disable() (źródło: opracowanie własne)

Jednym z parametrów konstruktora klasy `Button` jest akcja uruchamiana przez zdarzenie kliknięcia przyciskiem myszy. Metoda `Update()` `jest` odpowiedzialna za wywołanie metody `Invoke()na` akcji, jeżeli spełnione zostaną warunki takie jak odpowiedni stan oraz pozycja kursora. Dodatkowo ze względu na fakt, że aktualizacja przycisku jest wywoływana wiele razy na sekundę, niezbędne było zablokowanie możliwości wielokrotnego wywołania akcji (Rysunek 5.3.4.3).

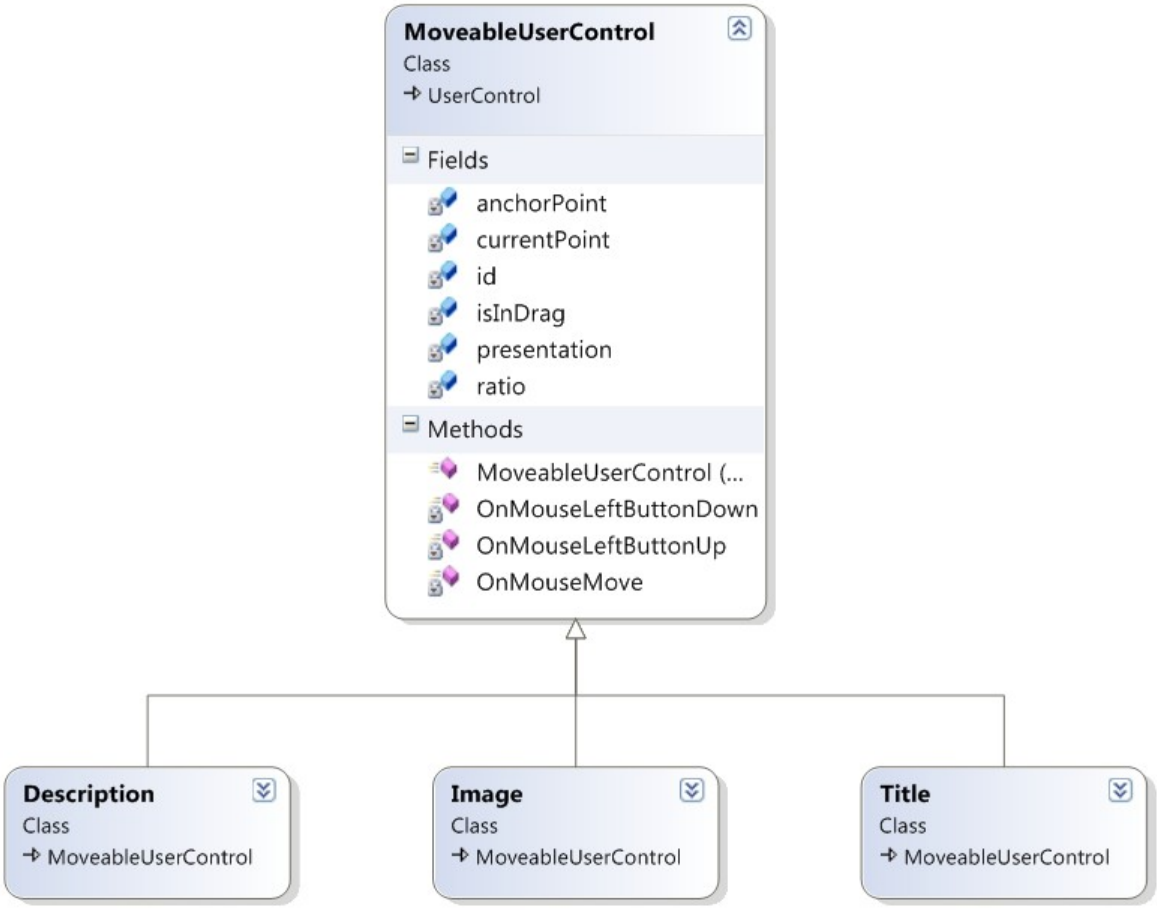
```
public void Update(GameTime gameTime)
{
    MouseState state = Mouse.GetState();
    if (LeftRight(state) && TopBottom(state) && !disabled)
    {
        if (state.LeftButton == ButtonState.Pressed)
        {
            pressed = true;
        }

        if (state.LeftButton == ButtonState.Released && pressed)
        {
            pressed = false;
            onClickFunction.Invoke();
        }
    }
}
```

Rysunek 5.3.4.3: Metoda Update() (źródło: opracowanie własne)

Zastosowanie zmiennej `pressed` umożliwiło zablokowanie wielokrotnych wywołań metody `Invoke()` na akcji, przekazanej przez konstruktor klasy `Button`.

5.3.5 Przemieszczanie elementów slajdu poprzez przeciągnięcie

Przy tworzeniu prezentacji możliwość rozmieszczania elementów na slajdzie nie powinna różnić się sposobem od dostępnych w konkurencyjnych rozwiązaniach. Na potrzeby Edytora powstała klasa `MovableUserControl`, wykorzystywana jako bazowa dla opisu, tytułu i obrazka czyli elementów które mogą zostać umieszczone na slajdzie (Rysunek 5.3.5.1).



Rysunek 5.3.5.1: Diagram klasy MovableUserControl (źródło: opracowanie własne)

Przedstawiona klasa wykorzystuje zdarzenia myszy, takie jak wciśnięcie i puszczenie lewego przycisku oraz ruchu. Metoda wywoływana podczas zmiany pozycji kursora, czyli `OnMouseMove()` sprawdza stan zmiennej `isInDrag`, która przyjmuje wartość `True`, gdy został dodatkowo wciśnięty przycisk oraz `False`, w przypadku puszczenia, a następnie aktualizuje marginesy elementu, który wywołał zdarzenie (Rysunek 5.3.5.2).

```
void OnMouseMove(object sender, MouseEventArgs e)
{
    if (isInDrag)
    {
        var element = sender as FrameworkElement;
        currentPoint = e.GetPosition(null);

        element.Margin = new Thickness()
        {
            Left = element.Margin.Left + currentPoint.X - anchorPoint.X,
            Top = element.Margin.Top + currentPoint.Y - anchorPoint.Y
        };

        anchorPoint = currentPoint;
    }
}
```

Rysunek 5.3.5.2: Szczegóły metody OnMouseMove() (źródło: opracowanie własne)

Po pobraniu pozycji kursora tworzony jest nowy margines, składający się z wartości poprzedniego oraz aktualnego przesunięcia. Nowe położenie zostało obliczone na podstawie pozycji kursora, od którego odjęto poprzednią pozycję.

Klasa `MovableUserControl` posiada dodatkowo elementy wspólne dla dziedziczących po niej klas takie jak identyfikator zawierający unikalną wartość, odróżniającą dany element ze wszystkich umieszczonych na slajdzie oraz zmienną zawierającą referencje do obiektu prezentacji reprezentującego model danych.

6. Podsumowanie

Usprawnienie procesu tworzenia i wyświetlania nieliniowych prezentacji z uwzględnieniem grup docelowych może zostać osiągnięte przez zmiany w podejściu do budowania pokazu i struktury slajdów. Obecnie stosowane rozwiązania przedstawione na przykład w Zoho Docs pozwalają na budowanie omawianych prezentacji, jednak proces jest pracochłonny, gdyż wymaga umieszczenia na każdym slajdzie przycisków, których kliknięcie wywołuje akcję przejścia do innego miejsca w pokazie. Inne rozwiązanie, takie jak wspomniana webowa aplikacja Prezi, nie pozwala na budowanie prezentacji o strukturze umożliwiającej wykorzystanie grup docelowych, ale przedstawia nowatorski sposób wyświetlenia przygotowanego materiału.

W ramach pracy zostały zaprezentowane dwie aplikacje: Edytor i Prezenter, które wykorzystują założenia zaproponowanej koncepcji. Oba programy zostały napisane w języku C# i opierają się na środowisku XNA oraz podsystemie renderowania WPF. Do przechowywania danych zastosowano format XML, zawierający grupy docelowe oraz slajdy wraz z ich elementami multimedialnymi. Zbudowane oprogramowanie pozwala na tworzenie, edycję i wyświetlanie pokazu. Dodatkowo użycie środowiska 3D oraz animacji pokazuje możliwości płynące z przejścia z przestarzałych technologii dwu-wymiarowych do nowoczesnego świata akceleratorów graficznych.

6.1. Zalety i wady przyjętych rozwiązań

Wykonane prototypy aplikacji wykorzystują zaprezentowaną koncepcję ulepszenia tworzenia nieliniowych prezentacji.

Aplikacja Edytor, wykorzystująca WPF pozwala na tworzenie i edytowanie pokazu z uwzględnieniem grup docelowych. Proces budowy wymaga niskiego nakładu pracy. Generowanie zawartości i edycja elementów jest intuicyjna i nie sprawia problemów początkującym użytkownikom.

Edytor wykorzystuje system WPF do renderowania okna, który używa akceleratora graficznego wraz z DirectX zamiast przestarzałego i powolnego, sprzętowego GDI+. Definicja pokazu slajdów posługuje się powszechnie znanym formatem XML do zapisu informacji o grupach docelowych oraz slajdach i powiązań między nimi.

Aplikacja Prezenter oparta o framework XNA jest odpowiedzialna za wyświetlanie. Wykorzystując akceleracje sprzętową generuje trójwymiarowe środowisko, w którym umieszczone zostały slajdy. Dzięki różnym pozycjom oraz animacjom przejść między nimi, udało się uzyskać efekt przemieszczania kamery.

Przejścia do slajdu następnego i poprzedniego różnią się od tych wykorzystanych przy wyborze grupy. Interfejs aplikacji jest prosty i czytelny oraz zawiera podstawowe funkcjonalności niezbędne do przeprowadzenia pokazu.

W porównaniu do rozwiązań zaprezentowanych przez konkurencyjne programy dotyczące tworzenia slajdów najważniejszą wadą jest brak webowych wersji aplikacji. W obecnych czasach największą popularność zyskują aplikacje, które możemy wykorzystywać bez instalowania ich na komputerze. Model SaaS zaprezentowany przez konkurencje pokazuje, jak bardzo wygodne jest omawiane rozwiązanie. Uzależnienie użytkownika od systemu Windows i potrzeby instalacji ograniczają potencjalnych odbiorców aplikacji.

Kolejną wadą jest mało nowoczesny wygląd obu aplikacji. Jeżeli przedstawiony produkt miałby podbić rynek, powinien wykorzystywać najnowsze osiągnięcia i trendy w dziedzinie tworzenia interfejsów użytkownika. Zastosowane w pracy rozwiązanie jest jedynie demonstracją i jedną z wielu możliwości, jak mogły by wyglądać aplikacje do budowania, edycji i wyświetlania nieliniowych prezentacji z uwzględnieniem grup docelowych.

6.2. Proponowane plany rozwoju

Oba prototypy aplikacji, Edytor oraz Prezenter realizują założenia, które powinny być spełnione przez oprogramowanie do tworzenia i prezentowania pokazów slajdów. Demonstrują, jak mogłoby wyglądać jedno z rozwiązań ułatwiajcie budowanie nieliniowych prezentacji. Możliwości rozwoju zostały opisane poniżej:

Plik definiujący prezentacje

Obecnie w pliku XML definicja bitmapy wykorzystuje tylko relatywną ścieżkę do pliku z grafiką. Zastosowanie zapisu danych pliku graficznego do XML pozwoliłoby na korzystanie z jednego, dużego pliku, zawierającego w sobie wszystkie potrzebne zasoby.

Aplikacja webowa

Obserwując trendy można zauważyć tendencję do przechodzenia na model SaaS. Nowoczesna aplikacja powinna znajdować się w chmurze i umożliwiać pracę bez instalowania na dysku twardym.

Zbudowanie webowej wersji oprogramowania pozwoliłoby na szerszy dostęp z różnego rodzaju urządzeń, nie tylko komputerów PC.

Elementy slajdu

Na obecnym etapie zaprezentowane prototypy potrafią obsłużyć jedynie kilka elementów prezentacji multimedialnych takich jak: tytuł, opis oraz bitmapa. Dodanie obsługi filmów wideo czy dźwięku, a także innych interaktywnych struktur posiada wysoki priorytet i powinno być wdrożone jak najszybciej.

Nowoczesny wygląd

Obie aplikacje prezentują bardzo minimalistyczny wygląd i zawierają podstawowe funkcjonalności. Otwierając możliwość używania przedstawionego oprogramowania dla szerszej grupy odbiorców należałoby wykorzystać najnowsze trendy w projektowaniu interfejsu użytkownika, takie jak chociażby kafelkowe układy interfejsu znane z Windows 8.

Optymalizacja ładowania Prezentera

Wykorzystanie bitmap, na których renderowane są elementy slajdu jest dobrym rozwiązaniem w przypadku wyświetlania ich na płaszczyźnie w przestrzeni 3D. Jednak obecnie wspomniany proces jest dość powolny, gdyż bitmapy generowane są dla wszystkich slajdów, co jest dość obciążające dla procesora. Nawet zastosowanie przetwarzania równoległego nie wyeliminowało problemu oczekiwania na załadowanie się prezentacji. Rozwiązaniem mogłoby być renderowanie tylko slajdu początkowego oraz tych w najbliższym otoczeniu. Pozostałe byłyby ładowane w momencie, gdy będą potrzebne.

Bibliografia

[1]. Pat McGee, Stephen Cawood. *Microsoft XNA Game Studio Creator's Guide, Second Edition*. Wydawnictwo McGraw-Hill Osborne Media, ISBN 0071614060.

[2]. Doug Lowe. *PowerPoint 2010 For Dummies*. Wydawnictwo For Dummies, ISBN 0470487658.

[3]. David Hunter, Kurt Cagle, Chris Dix, Roger Kovack, *Beginning XML, 2nd Edition*. Wydawnictwo Wrox, ISBN 0764543946.

[4]. Microsoft. *XML Standards Reference*. http://msdn.microsoft.com/en-us/library/ms256177.aspx. Dostęp: 2013-06-17.

[5]. Joseph Albahari, Ben Albahari. *C# 5.0 Pocket Reference: Instant Help for C# 5.0 Programmers*. Wydawnictwo O'Reilly Media.

[6]. Herbert Schildt. *C# 4.0 The Complete Reference*. Wydawnictwo McGraw-Hill Osborne Media, ISBN 007174116X, ISBN 1449320171.

[7]. Andrew Stellman, Jennifer Greene. *Head First C#.* Wydawnictwo O'Reilly Media, ISBN 0596514824.

[8]. Barbara Doyle. *C# Programming: From Problem Analysis to Program Design*. Wydawnictwo Cengage Learning, ISBN 0538453028.

[9]. Microsoft. *.Net Framework 4.5.* http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2.aspx. Dostęp: 2013-06-21.

[10]. Jacob Vibe Hammer, Karli Watson, Morgan Skinner, Jon Reid, Christian Nagel. *Beginning Visual C# 2012 Programming*. Wydawnictwo Wrox, ISBN 1118314417.

[11].Matthew MacDonald. *Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5.* Wydawnictwo Apress, ISBN 1430243651.

[12]. Microsoft. *Building a WPF Application*. http://msdn.microsoft.com/en-us/library/aa970678.aspx. Dostęp: 2013-06-22.

[13]. Microsoft. *Podstawy XNA – Wprowadzenie*. http://msdn.microsoft.com/pl-pl/library/podstawy-xna—wprowadzenie.aspx. Dostęp: 2013-06-22.

[14]. A.T. Chamillard. *Beginning C# Programming with XNA Game Studio*. Wydawnictwo Burning Teddy, ASIN B0076RMPGM.

Spis rysunków