This paper should be cited as: Wichrowski, M. (2008). Techniki projektowania dostępnych interfejsów w Adobe Flash Wprowadzenie Historia aplikacji Flash. Proceedings of the Conference: Interfejs użytkownika - Kansei w praktyce, Warszawa 2008. Wydawnictwo PJWSTK.

Techniki projektowania dostępnych interfejsów w *Adobe Flash*

MARCIN WICHROWSKI

Polsko – Japońska Wyższa Szkoła Technik Komputerowych,
Koszykowa 86, 02-008 Warszawa

Wprowadzenie

Celem tego artykułu jest zaprezentowanie podstawowych zasad, jakie muszą być spełnione przy konstruowaniu dostępnych interfejsów użytkownika w programie *Adobe Flash*. Poza omówieniem technik projektowych wspomagających tworzenie interfejsów dla osób niepełnosprawnych, skupiono także uwagę na podstawach testowania filmów *Flash*.

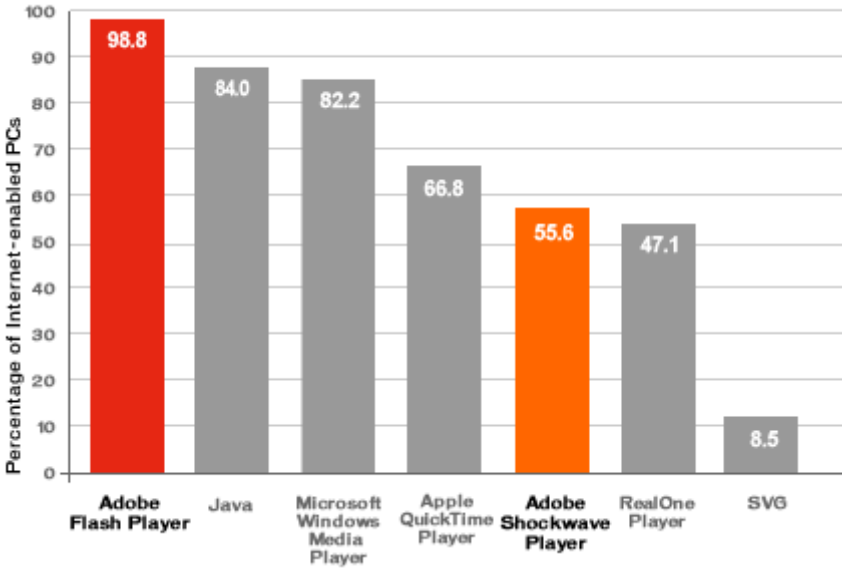
Historia aplikacji *Flash*

Droga do sukcesu programu *Flash* była długa i burzliwa. Obecnie uważany jest on za jeden z najszerzej rozpowszechnionych formatów multimedialnych w Internecie. Stanowi jedną z podstaw konstrukcji RIA (*Rich Internet Applications*). Według wyników podanych przez firmę *Adobe* (dane z marca 2008 roku) aż 98,8% komputerów z dostępem do Internetu ma zainstalowany w systemie operacyjnym *Flash Player* (Rys. 1).

Cechy, które przyczyniły się do zdobycia przez *Flasha* tak dużej popularności to m.in.:

- wzbogacenie stron internetowych o pełne spektrum form prezentacyjnych (narracja, muzyka, animacja, video, bogata interakcja itd.),
- możliwość kreacji atrakcyjnych wizualnie i interakcyjnie różnorodnych projektów w formie *online* i *offline*,
- skalowalność i mały rozmiar plików dzięki zastosowaniu grafiki wektorowej (uniezależnienie tym samym od rozdzielczości ekranów użytkowników),
- niezależność wyglądu od używanej przeglądarki,
- wciąż rozwijany język *ActionScript*, dający duże możliwości programistyczne,
- współpraca z wieloma innymi technologiami (w tym m.in. obsługa baz danych, XML, streamingu wideo, *Flex, AIR* itd.)

- dostępność użycia klawiatury jako interfejsu sterującego prezentacją,
- darmowa dystrybucja *Flash Player*a na wiele systemów operacyjnych i przeglądarek internetowych oraz łatwość jego instalacji (już w roku 1999 *Flash Player 4* osiągnął 100 milionów instalacji – głównie, dzięki dołączeniu go do *Microsoft Internet Explorera 5*, w roku 2000 był także dystrybuowany z *AOL* i *Netscape*, a później razem z każdą wersją *Windows XP*).[1]

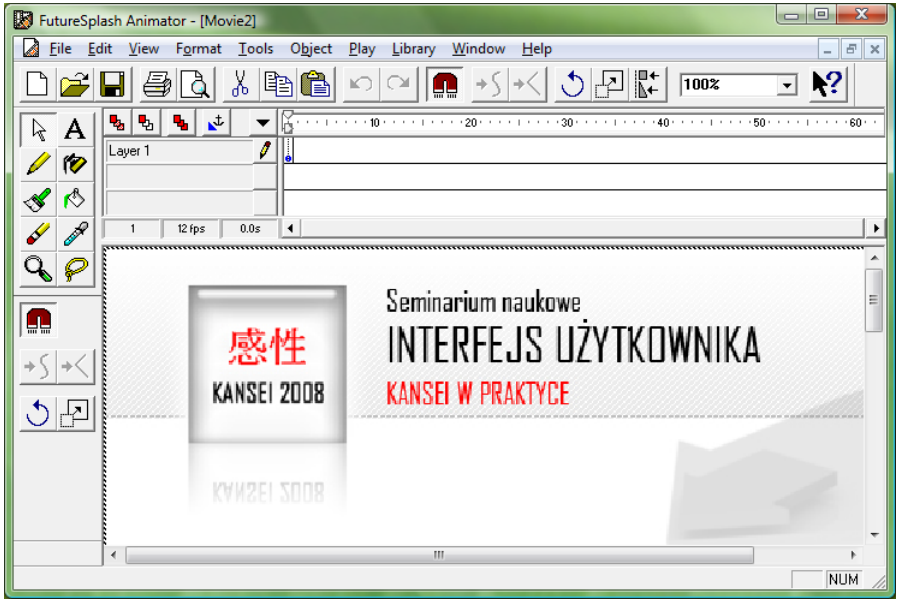


Rys. 1. Procentowy rozkład instalacji rozszerzeń multimedialnych w systemach komputerowych na (dane dotyczą USA, Kanady, Wielkiej Brytanii, Francji, Niemiec i Japonii; marzec 2008) [2]

Zanim jednak *Flash* stał się tak powszechnym i prawdziwie dynamicznym, zaawansowanym wizualno – programistycznym środowiskiem twórczym, jego początek sięga aplikacji *SmartSketch* stworzonej w 1993 roku przez firmę *FutureWave Software*. Było to proste wektorowe narzędzie rysownicze. W roku 1995 dodano do niego możliwość animacji poklatkowej i zmieniono nazwę na *FutureSplash Animator*. Podjęto próbę sprzedaży aplikacji firmie *Adobe*, ale jej wolne działanie zniechęciło potencjalnego nabywcę.[3] Po wydaniu programu w 1996 roku dużą

[1] http://en.wikipedia.org/wiki/Adobe_Flash
[2] http://www.adobe.com/products/player_census/flashplayer/
[3] http://www.flashmagazine.com/news/detail/the_flash_history/

popularność przyniosło jej wykorzystanie przez *Microsoft* przy produkcji serwisu *MSN* oraz przez *Disney Online*. W tym samym roku *Macromedia* nabyła tę aplikację i łącząc słowa *„Future”* oraz *„Splash”* stworzyła nazwę *„Flash”.*



Rys. 2. *FutureSplash Animator / Macromedia Flash 1*

Od tej pory rozpoczęła się multimedialna rewolucja sieci, która często wymykała się spod kontroli. Wielu projektantów wybrało niewłaściwą ścieżkę – tworzyło oszałamiające wizualnie strony zachwycające swoim urokiem, pomijając całkowicie ich użyteczność i dostępność. Skutkiem były serwisy doprowadzające odbiorców do prawdziwej frustracji. Wielokrotnie ważniejsze w nich było jak coś wygląda, a nie jak działa. Długie intra, fantazyjne przejścia między stronami, nieintuicyjne sztucznie skomplikowane interfejsy, nieracjonalny czas oczekiwania na załadowanie strony to tylko najczęstsze przykłady takich strategii projektowych. Odbiorcy dość szybko poczuli się tym znużeni. Słowo *Flash* stawało się synonimem „nieużyteczny”. W roku 2000 ekspert od użyteczności - Jakob Nielsen opublikował artykuł "*Flash 99% Bad*”.[4] *Macromedia* obserwując tę sytuację poczyniła wiele starań, by nie zaprzepaścić szansy jaką dawała rozwijająca się technologia. Jednym z nich było zatrudnienie właśnie samego Nielsena w celu ulepszenia użyteczności *Flasha*. W kolejnych wersjach aplikacji poświęcano więcej uwagi na

[4] http://www.useit.com/alertbox/20001029.html

wbudowanie mechanizmów zapewniających większą użyteczność tworzonych projektów i ich dostępność. Wersje demonstracyjne wszystkich wydań *Flasha* można odszukać w Internecie.[5]
Odpowiedzią firmy *Microsoft* na technologię *Flash* jest rozwijany początkowo pod kodową nazwą *Windows Presentation Foundation/Everywhere (WPF/E)* projekt *Silverlight* (aktualnie wersja 2.0 w fazie alpha).

Rozwój technik dostępności w aplikacji *Flash*

Pomimo tak szybkiej ekspansji technologii *Flash*, kwestia dostosowania do potrzeb osób niepełnosprawnych przez długi czas była pomijana przez jej producentów. Rozwój technik dostępności jest związany w dużym stopniu z ewolucją projektowania interakcji we *Flash*u. Pierwsze trzy wersje aplikacji oferowały bardzo ograniczone możliwości interakcyjne. W czwartym wydaniu programu zestaw prostych komend przerodził się w język skryptowy. Jego rozszerzona wersja zadebiutowała pod nazwą *ActionScript* w wersji piątej. Techniki dostępności pojawiły się dopiero w następnej – szóstej wersji (oznaczonej *MX*). Z tego też powodu minimalnym wymaganiem dla działania projektów wykorzystujących te funkcje jest *Flash Player 6.* Wprowadzone zostały również podstawowe tagi *HTML*[6] w polach tekstowych, typowe elementy używane przy konstruowaniu formularzy oraz obsługa wideo. Premiera w pełni obiektowego języka *ActionScript 2.0* w wersji siódmej (*MX 2004*) pozwoliła na łatwiejsze i bardziej wydajne przenoszenie raz stworzonych rozwiązań sprzyjających użyteczności do wielu nowych projektów. W roku 2005 nastąpiła fuzja firm *Macromedia* i *Adobe Systems Incorporated*. Najnowsza wersja *Flash* CS3 została wydana już przez *Adobe*. Wzbogacono ją o nowy *ActionScript* 3.0, mający być bezpieczniejszym, prostszym, wydajniejszym i bardziej kompatybilnym modelem programistycznym. Zawiera też nowy komponent *FLVPlaybackCaptioning* wspierający standard dla podpisów w filmach – *DFXP*[7] opracowany przez *World Wide Web Consortium (W3C).* Adobe nawiązało również współpracę z wieloma firmami produkującymi programy wspierające dodawanie podpisów do filmów *FLV*.[8] *Flash CS3* zawiera także wiele komponentów interfejsu obsługujących techniki dostępności. Są to elementy typu: *Button, Checkbox, Radio button, Text input, Text area, Combo box, List box, Data grid, Tile list.*

Podstawy projektowania dostępnych interfejsów

Jedną z najbardziej wartościowych cech, którą powinni posiadać projektanci graficznych interfejsów jest umiejętność przekształcania i prezentowania informacji o zasadach działania interakcji w formy wizualne zrozumiałe dla użytkowników. Tworząc dostępne interfejsy powinno się rozważyć wszelkie możliwe przeszkody utrudniające osobom niepełnosprawnym komunikację z komputerem, uwzględnić zasady działania narzędzi ułatwiających tę komunikację i stosować takie rozwiązania interakcji z komputerem, które będą najłatwiejsze w obsłudze.

[5] http://flash-ascript.blogspot.com/2007/06/first-steps-of-flash-futuresplash.html
[6] http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_14808
[7] Timed Text (TT) Authoring Format 1.0 – Distribution Format Exchange Profile
[8] http://www.adobe.com/accessibility/products/flash/captioning_tools.html

W uproszczonej wersji możemy przyjąć, iż niepełnosprawność dotyczy osób:

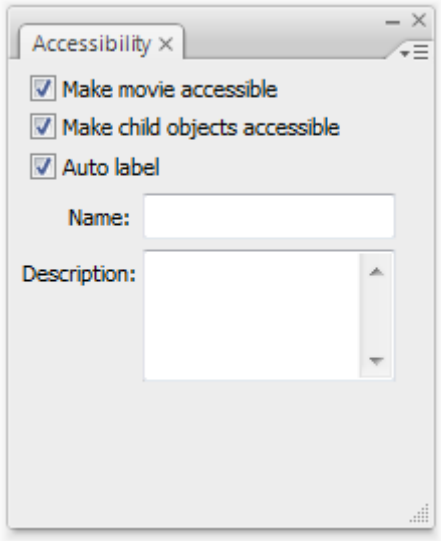
- niedosłyszących/niesłyszących – konieczność zapewnienia zsynchronizowanych podpisów opisujących zawartość dźwiękową,
- niewidomych – zawartość ekranu musi być dostępna dla programów typu *screen reader*; umożliwienie dostępu przez klawiaturę,
- niedowidzących – zapewnienie dużego kontrastu; możliwość powiększenia elementów ekranu,
- z zaburzeniami percepcji barwnej – unikanie zestawień barw uniemożliwiających prawidłową percepcję zawartości,
- z epilepsją – unikanie migotania zawartości ekranu,
- niesprawnych ruchowo – umożliwienie dostępu przez klawiaturę, unikanie skomplikowanych wymagań ruchowych przy obsłudze interfejsu
- z zaburzeniami poznawczymi – wykorzystanie prostych i spójnych schematów nawigacyjnych, stosowanie łatwych mechanizmów obsługi interfejsu, unikanie skomplikowanego języka, zapewnienie kontroli nad elementami zmieniającymi się w czasie.

Podstawą komunikacji aplikacji *Flash* z technikami wspomagającymi jest technologia *Microsoft Active Accessibility (MSAA). Flash Player* przygotowuje listę dostępnych obiektów, która jest odświeżana gdy zmienia się zawartość filmu lub nastąpi jakaś interakcja. Decyzja o dostosowaniu interfejsu do potrzeb osób niepełnosprawnych powinna być uwzględniona już na etapie koncepcyjnym, gdyż z wielu względów (np. ograniczeń programów czytających zawartość ekranu) należy odpowiednio wcześnie zaplanować układ i sposób komunikacji. W przypadkach złożonych interakcji, gdzie wiele zadań dzieje się jednocześnie, często konieczne jest jej uproszczenie. Wykonanie podobnych zadań w świecie rzeczywistym nie musi być wcale kłopotliwe dla osób niepełnosprawnych, jednak gdy pośrednikiem jest np. *screen reader*, odczytujący każdy element pojedynczo, obsługa interfejsu może się znacząco skomplikować.

Zapewnienie ekwiwalentów tekstowych dla elementów graficznych

Domyślnie obiekty tekstowe w prezentacji *Flash* (także te zawarte w przyciskach) są identyfikowane i czytane przez *screen reader*. Obiekty graficzne i animacje muszą być jednak opisane indywidualnie przez projektantów. Szczególną uwagę należy zwrócić na elementy tekstowe, które zostały zamienione na kształty poleceniem *Break Apart*. Tego rodzaju konwersję stosuje się czasem w celu zapewnienia identycznego wyglądu prezentacji na wszystkich komputerach. Obiekty tego typu stają się jednak niedostępne dla programów odczytujących zawartość ekranu i konieczne jest dodanie do nich odpowiednika tekstowego. Dodatkowo łatwo je pominąć w procesie dostosowywania dla potrzeb technik wspomagających, gdyż pozornie nie odróżniają się od obiektów tekstowych. Podpisy dodawać można tylko do symboli typu *button* lub

movie clip. Mogą one opisywać cały film, pojedyncze obiekty lub grupy obiektów zawarte wewnątrz. Panel *Accessibility* wywołuje się przez menu *Window / Other Panels / Accessibility* lub *Shift + F11*.



Rys. 2. Panel *Accessibility* w *Adobe Flash CS3*

Panel ten udostępnia następujące funkcje:

- *Make Movie Accessible* – wybranie tej opcji powoduje dostępność danego filmu *Flash* dla aplikacji typu *screen reader*; gdy nie jest włączona program czytający informuje tylko o tym, że napotkał obiekt *Flash* i nie analizuje jego zawartości,

- *Make Object Accessible* (dostępne gdy wybrany jest obiekt) - wybranie tej opcji powoduje dostępność danego obiektu w filmie *Flash* dla aplikacji typu *screen reader;* zaleca się jej wyłączenie jeśli obiekt nie zawiera istotnych informacji (np. pełni funkcje dekoracyjne) i ma być niewidoczny dla programu czytającego,

- *Make Child Objects Accessible* – jeżeli symbol złożony jest z innych obiektów (np. w przypadku bardziej złożonych animacji) można je ukryć przed programem odczytującym wyłączając tę opcję; pozwoli to na odczytanie tylko opisu tej animacji jako całości, bez analizy jej wnętrza,

- *Auto label* - wybranie tej opcji powoduje, że tekst zawarty w elementach *buttons* i *movie clip* będzie odczytywany jako tekst alternatywny; opcja ta dotyczy pojedynczych obiektów tekstowych; działa również z innymi komponentami typu *radio buttons, list boxes*,

- *Name* - przechowuje tekst alternatywny opisujący obiekt *Flash,* który będzie odczytywany przez *screen reader*; powinien być on krótki i jednoznaczny skupiając uwagę raczej na funkcji obiektu niż na dosłownym opisie zawartości,

- *Description* - przechowuje dłuższy opis tekstowy charakteryzujący obiekt *Flash*; najlepiej wykorzystywać go tylko wówczas gdy opis w polu *Name* ma przekroczyć 50 znaków; należy jednak mieć na uwadze fakt, że obydwa pola są odczytywane automatycznie i nadużywanie długich definicji może być męczące dla słuchającego,

- *Shortcut* – przechowuje informacje o skrócie klawiaturowym dla danego obiektu (nie przyporządkowuje tego skrótu, lecz jedynie informuje o nim).

Zasady stosowania animacji

Aplikacja *Flash* została zaprojektowania przede wszystkim jako narzędzie do tworzenia animowanych, bogatych multimedialnie zaawansowanych prezentacji, mających na celu zainteresować odbiorcę i zachęcić go do interakcji. Niestety technologie wspomagające dostępność nie działają w podobny sposób i nie tworzono ich z myślą obsługi tego rodzaju skomplikowanych rozwiązań. Dlatego należy mieć na uwadze ograniczenia jakie narzucają na prezentacje. Gdy *screen reader* napotyka obiekt *Flash* informuje wówczas użytkownika o załadowaniu zawartości filmu i przechodzi do analizy jego wnętrza. Jakakolwiek zmiana w tym obiekcie powoduje wysłanie sygnału przez *Flash Player* do programu czytającego i rozpoczyna on działanie od początku strony. Łatwo wyobrazić sobie jakie utrudnienia może powodować przedstawiona sytuacja. Rozwiązaniem powyższego problemu w przypadku bardziej złożonych animacji jest zapewnienie dla nich opisu jako całości i wyłączenie parametru *Make Child Objects Accessible*, dzięki czemu *screen reader* nie będzie analizował zawartości filmu. Pomoże to lepiej przedstawić odbiorcy zależności jakie zachodzą między zagnieżdżonymi elementami i spowoduje ich ignorowanie przez *screen reader*.
Wskazane jest unikanie częstego migotania ekranu i ciągłego poruszania elementów w prezentacji szczególnie w sytuacjach gdy prezentuje się użytkownikowi tekst do odczytania. Efekty te mogą powodować nie tylko utrudnienia w percepcji, ale nawet wywoływać epilepsję. Można oczywiście wprowadzać proste animacje, jednak po chwili zawartość ekranu powinna stać się statyczna.
Ponieważ programy czytające zawartość ekranu mogą nie nadążać za zmieniającą się zawartością prezentacji, użytkownicy powinni mieć kontrolę nad prędkością wyświetlania kolejnych ekranów i móc nimi sterować np. przez przyciski następny, poprzedni.

Zapewnienie informacji o strukturze prezentacji i działaniu elementów kontrolnych

Gdy układ prezentacji jest skomplikowany użytkownicy polegający jedynie na informacjach dostarczanych przez *screen reader* mogą poczuć się zdezorientowani. W takich przypadkach warto stosować osobny ekran opisujący sposób nawigacji i strukturę prezentacji lub zamieścić taką informację na poziomie nadrzędnym prezentacji.
Kolejnym ważnym aspektem w projektowaniu dostępnej interakcji jest odpowiednie informowanie użytkowników o właściwościach elementów kontrolnych takich jak:

- Rola – pozwala zdefiniować za co odpowiada dany element i jak działa (czy jest to przycisk, suwak, gałka itp.),
- Aktualny stan – w jakim aktualnie stanie jest dany element (włączony/wyłączony, na jaką wartość jest ustawiony i ile jest możliwych wartości np. poziom 3 z 24),
- Struktura i kontekst – w jakich relacjach interakcyjnych i lokalizacyjnych pozostaje element sterujący z innymi obiektami.

Dostęp przez klawiaturę

Niezwykle ważna jest możliwość interakcji z prezentacją za pomocą klawiatury, gdyż nie wszyscy odbiorcy mogą używać myszki. Bardzo pomocne jest przypisanie skrótów klawiaturowych najistotniejszym poleceniom.

Ustalenie kolejności odczytywania elementów

Dostosowanie kolejności w jakiej odczytywana będzie zawartości filmów *Flash* jest jednym z najbardziej czasochłonnych i trudnych zadań w procesie integracji z programami typu *screen reader*, które niestety nie działają w przewidywalny sposób (np. czytając od lewej do prawej, z góry do dołu czy w porządku alfabetycznym). W celu zapewnienia poprawnego porządku prezentowanych elementów stosuje się następujące rozwiązania:

- Ograniczenie rozmiaru filmu – obiekty *Flash* mniejsze niż 300 pikseli szerokości złożone z pojedynczej kolumny lub pojedynczego wiersza obiektów odczytywane są we właściwej kolejności,

- Kontrola za pomocą *ActionScript* – najbardziej precyzyjna z metod używająca parametru *.tabindex,* który musi być przypisany do każdego elementu w prezentacji. Wymusza to nadanie nazwy każdej instancji symbolu na scenie. Ponieważ nie można przypisać nazw instancji elementom typu *static text* konieczne jest używanie dynamicznych pól tekstowych. Lista parametrów *.tabindex* dotyczy także tych obiektów, które są niewidoczne (znajdują się poza sceną, są przykryte innymi obiektami lub pojawią się w późniejszej części filmu). Jeśli element ma być ignorowany przez *screen reader* należy przypisać mu parametr *.silent.* Dodatkowych zabiegów wymagają zewnętrzne pliki *SWF* ładowane do wnętrza głównego filmu. Powinny one również zawierać parametry *.tabindex.* Przykładowo pierwszy zewnętrzny plik powinien mieć przyporządkowane do elementów wartości w postaci 1,2,3 zaś następny 4,5,6 itd. Nie muszą być one ułożone sekwencyjnie, ale muszą być unikalne. Programem ułatwiającym realizację przytoczonych powyżej procedur jest *AccRepair* firmy *HiSoftware*. [9] Odszukuje on nienazwane instancje, konwertuje tekst na postać dynamiczną i buduje porządek czytania obiektów.

- Kontrola za pomocą dodatkowego zestawu elementów sterujących poza sceną – wymaga to wyłączenia dostępności wszystkich elementów widocznych na scenie i umieszczenia poza nią pojedynczej kolumny obiektów. Minusami tego rozwiązania jest zwiększenie wielkości filmu i duże prawdopodobieństwo braku kompatybilności z narzędziami powiększającymi ekran (ich działanie nie ogranicza się tylko do powiększania zawartości ekranu, ale także powoduje wycentrowanie tych elementów na scenie).

Kontrola elementów dźwiękowych

Bardzo ważne jest zapewnienie kontroli odtwarzania dźwięków zawartych w prezentacji ponieważ mogą one zagłuszać program czytający. Często spotykanym rozwiązaniem tego problemu jest zastosowanie skrótów klawiaturowych dla poleceń graj/pauza (np. przycisk P), wycisz (np. przycisk M lub cyfra 0) i regulacji głośności.

Zasady stosowania barw

Projektując schematy kolorystyczne dla interfejsu należy uwzględnić trudności jakie mogą napotkać osoby niedowidzące lub z zaburzeniami percepcji barwnej. Podstawą zasadą jest zapewnienie odpowiedniego kontrastu między tekstem a tłem. Jedną z metod pozwalającą sprawdzić poprawność kontrastu jest wyświetlenie danego ekranu jedynie w odcieniach szarości. Jeśli elementy stają się wówczas nieczytelne to jest to wskazówka, ze kontrast między nimi jest za mały. Innym problemem jest fakt, że pewna część ludzi dotknięta jest brakiem rozróżniania kolorów. Najczęstszą odmianą tego zaburzenia jest nierozpoznawanie barw czerwonej i zielonej gdy ich nasycenie i jasność są na zbliżonym

[9] http://www.adobe.com/products/flash/extensions/accrepair/

poziomie. Z tego też powodu nie powinno stosować się kodowania informacji jedynie za pomocą barwy. Doskonałym narzędziem umożliwiającym sprawdzenie prezentacji pod względem charakterystyki barwnej jest *Fujitsu ColorDoctor*.[10]

Testowanie interfejsów *Flash*

Testując interfejsy stworzone we *Flash*u należy zwrócić uwagę na fakt, iż mechanizmy ich działania często różnią się od tradycyjnych interfejsów *HTML*. Za przykład posłużyć może chociażby fakt, że interfejsy *Flash* działają na ogół dynamiczne – zarówno ich wygląd, zawartość czy położenie może się zmieniać w trybie natychmiastowym na skutek działań użytkownika. Standardowy model interfejsu *HTML* opiera się natomiast na założeniu, że każda interakcja prowadzi do kontaktu z serwerem, który dopiero po przetworzeniu polecenia zwraca odpowiedź. Ta i wiele innych różnic powinny skłaniać projektantów flashowych do testowania swoich prac z uwzględnieniem specyficznych wymagań jakie stawia ten format.

Pragnąc zapewnić efektywne i stabilne działanie aplikacji flashowych warto podczas ich testowania zwrócić uwagę na:

- Działanie przy różnych szybkościach łącza – projektanci często pomijają ten aspekt i bazują jedynie na sprawdzaniu aplikacji lokalnie, co nie oddaje prawdziwych warunków pracy. W ten sposób łatwo jest nieświadomie stworzyć kod, którego poprawność działania będzie uzależniona od szybkiej komunikacji z systemem. W rzeczywiści jednak może się okazać, że mała prędkość transferu będzie powodować problemy nie tylko z ładowaniem zewnętrznych plików, przesyłaniem strumieni wideo, ale wywoła poważniejsze błędy związane z niewłaściwym działaniem poleceń *ActionScript*. Przykładowo: próba przejścia do klatek na osi czasu, które nie zostały jeszcze pobrane do pamięci; dynamiczny dostęp do danych przed ich załadowaniem; wywoływanie metod na niezainicjalizowanych obiektach; złe dane lub ich brak w asynchronicznych obliczeniach itp. Rozwiązaniem tej kwestii jest odpowiednio wcześnie przygotowany plan podziału prezentacji na mniejsze elementy i dodanie preloaderów pokazujących stopień załadowania aplikacji do systemu i odpowiednia komunikacja między wszystkimi elementami aplikacji. Warto też posłużyć się wbudowanym we *Flasha* symulatorem pobierania danych (*Simulate Download*) jednak nie może on całkowicie zastąpić testowania w realnych warunkach sieciowych.

- Zachowanie w różnych przeglądarkach internetowych – testując strony *HTML* przede wszystkim zwraca się uwagę na różnicę w ich wyglądzie i problemy skryptowe po stronie klienta. W przypadku obiektów typu *Flash* przeglądarki różnią się kodem potrzebnym do załadowania i oglądania ich zawartości. Podstawą jest więc sprawdzenie: poprawności osadzenia pliku SWF w kodzie *HTML*, dostępu do lokalizacji wewnętrznych/zewnętrznych (adresy URL, pliki, dane XML) oraz komunikacji *JavaScript* między *Flash*em a przeglądarką.

[10] http://www.fujitsu.com/global/accessibility/assistance/cd/

- Wersje odtwarzacza *Flash Player* – testując interfejs *Flash* konieczne jest uwzględnienie minimum dwóch wersji *Flash Player*a - poza wersją deklarowaną dla prawidłowego działania, warto jest sprawdzić także wersję poprzedzającą. W przypadku testowania filmów wykorzystujących techniki dostępności należy zwrócić uwagę, że minimalną wersją jest *Flash Player* 6 (poprzedzające wersje nie obsługują tych technik). Firma *Adobe* udostępnia archiwalne wersje *Flash Player*a do celów testowych.[11] Przeprowadzenie tego rodzaju testów pozwoli uniknąć zarówno prostych błędów jak pomyłkowa publikacja w innej wersji, jak również problemów związanych wykrywaniem *Flash Player*a i różnicami w funkcjonalności danej wersji.

  Jeśli priorytetem jest bezproblemowe dotarcie do największej liczby odbiorców wskazane jest unikanie publikacji w najnowszej wersji dostępnej na rynku. Użytkownicy zazwyczaj zwlekają z jej zainstalowaniem np. z obawy przed błędami, a wielu z nich nie ma po prostu uprawnień do takiej aktualizacji (np. w miejscu pracy). *Flash Player* Developer Center[12] oraz *Flash Support Center*[13] oferują pomoc techniczną w przypadku najczęściej spotykanych problemów związanych z działaniem projektów *Flash*.

- Wydajność działania na starszych systemach komputerowych – w przeciwieństwie do stron *HTML*, które nie wymagają dużej mocy obliczeniowej komputera do prezentacji zawartości, interfejsy *Flash* mogą w znaczącym stopniu wykorzystywać zasoby RAM, procesor czy kartę graficzną. Dlatego wskazane jest na już na początku procesu projektowego określenie minimum sprzętowego do prawidłowego działania aplikacji i systematyczne sprawdzanie wydajności budowanego projektu.

Podsumowanie

Podstawą projektowania dostępnych interfejsów jest nie tylko stosowanie technologii wspomagających, ale przede wszystkim świadomość twórców o możliwych ograniczeniach jakich doświadczają ludzie niepełnosprawni w interakcji z interfejsami. Proces dostosowywania aplikacji *Flash* pod względem dostępności powinien zacząć się już na etapie koncepcyjnym i być testowany w czasie swojego rozwoju przez potencjalnych użytkowników. Stosowane rozwiązania muszą być poprawnie skonstruowane pod względem technicznym (właściwie wspierać techniki dostępności) jak i praktycznym (działać zgodnie z oczekiwaniami odbiorców).

[11] http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_14266&sliceId=1
[12] http://www.adobe.com/devnet/flashplayer/
[13] http://www.adobe.com/support/flash/troubleshooting.html

Literatura

1. Ka Wai Cheung, Craig Bryant; „Flash Application Design Solutions: The Flash Usability Handbook”; Friends of ED Apress Inc.; 2006
2. http://www.flashmagazine.com/news/detail/the_flash_history/
3. Bob Regan; „Best Practices for Accessible Flash Design”; Macromedia 2005
4. http://www.adobe.com/accessibility/products/flash/
5. http://www.webaim.org/techniques/flash/#flashoverview
6. Kristopher Schultz; „Four steps to improving your *Flash* interface testing”; http://www.*adobe*.com/devnet/*flash*/articles/*flash*_interface_testing.html