Informatyka w Edukacji, XVI
UMK Toruń, 2019

PROGRAMOWANIE I URZĄDZENIA MOBILNE NA LEKCJACH INFORMATYKI

Andrzej Polewczyński
Wydział Matematyki i Informatyki UMK
ap@mat.umk.pl

Abstract. This article contains information about the possibilities of using the MIT App Inventor in school lessons. MIT App Inventor is a web application for users with minimal programming experience to easily and quickly build mobile applications, and allow users to build advanced applications. Visual blocks language allows users to create mobile applications for Android.

1. Wstęp

Współczesne aplikacje pisane na urządzenia mobilne można podzielić na trzy grupy: aplikacje internetowe, hybrydowe, oraz natywne. Aplikacje internetowe wykorzystują wyłącznie teksty kodów znakowania i formatowania warstwy wizualnej i są wspomagane często kodami skryptów np. JavaScript. Aplikacje natywne wykorzystują natywny SDK(ang. Software Development Kit) konkretnej platformy, aplikacje hybrydowe pozycjonuje się pomiędzy internetowymi i natywnymi ze względu na stosowane w nich rozwiązania mieszane. Warto nadmienić, że zwłaszcza piąta wersja HTML (ang. Hyper Text Markup Language) umożliwiła szerszej grupie zainteresowanych budowanie uniwersalnych aplikacji internetowych dostosowanych również do urządzeń mobilnych. Połączenie technologii HTML, CSS(ang. Cascading Style Sheets) oraz użycie języka skryptowego wystarcza, aby utworzyć aplikacje webowe i mobilnych jednocześnie. Ze względu na solidną obsługę różnych przeglądarek i dużą szybkość reakcji każda aplikacja utworzona z udziałem HTML 5 działa bezproblemowo na prawie wszystkich urządzeniach.

Kodowanie aplikacji mobilnej na lekcjach w szkole w wersji wymagającej korzystania z kilku technologii, może jednak stanowić pewną barierę organizacyjną. Dzięki wizualnym środowiskom projektowania proces powstawania programów na urządzeniach mobilnych stał się łatwiejszy i możliwy do realizacji na lekcjach. Wiadomo – nie od dziś – że warto sięgać po narzędzia, które umożliwią łatwe budowanie interfejsu aplikacji i jednocześnie proste i wizualne jej oprogramowanie przynajmniej na niektórych etapach nauki. Narzędziem, które zyskało sobie sporą grupę użytkowników,

przeznaczonym do tworzenia w sposób wizualny aplikacji dedykowanych na platformy z Androidem jest rozwijany przez Massachusetts Institute of Technology App Inventor [1] i występujący też pod nazwą App Inventor 2.

Aplikacja budowana za pomocą tego narzędzia tworzona jest w dwóch powiązanych ze sobą logicznie warstwach. Warstwa App Inventor Designer odpowiada za wybór komponentów warstwy graficznej interfejsu, ekranów, przycisków pól, i wielu innych elementów interfejsu zarówno widocznych jak i ukrytych. Warstwa App Inventor Blocks Editor, czyli program tworzony za pomocą bloków i określający zachowanie wybranych elementów interfejsu. Aplikacja może być tworzona systematycznie i uruchamiana bezpośrednio na urządzeniach mobilnych, albo za pomocą emulatora systemu Android. Środowisko App Inventor jest obsługiwane w systemie Mac OS X, GNU / Linux i systemach operacyjnych Windows i telefonach z systemem Android. App Inventor 2 jest rozwijany przez zespół MIT i jest naturalną kontynuacją edukacji dla wszystkich tych, którzy wcześniej mieli do czynienia ze Scratchem. Za pomocą App Invertor 2 można tworzyć nawet dość zaawansowane aplikacje, ponieważ środowisko łączy możliwości profesjonalnych środowisk programistycznych z prostotą i przystępnością dla osób początkujących znanego projektu Scratch.

Środowisko pozwala zapisywać wynikowe aplikacje w postaci plików .apk, które można łatwo instalować na smarftonie z Androidem – pod warunkiem włączenia odpowiedniej opcji pozwalającej na instalację aplikacji spoza Sklepu Play. App Inventor można zaliczyć do grupy przełomowych technologii wspierających kodowanie i jego naukę szerokiej grupie zainteresowanych. Może być używane do inspirowania i zwiększenia zainteresowania programowaniem wśród uczniów. Z punktu widzenia realizacji programu nauczania, sam pomysł tworzenia aplikacji mobilnej stanowi dodatkową okazję do demonstracji tematyki i zasad programowania obiektowego. Warto zauważyć też, że przy okazji takiego programowania uczeń wychodzi z własnoręcznie napisanym kodem poza ramy lokalnego komputera klasy PC. Programuje inne urządzenie. Programuje urządzenie, którego na co dzień używa, a wyniki jego działań mają charakter praktyczny i wzbudzający chęć uzyskania coraz lepszych rezultatów.

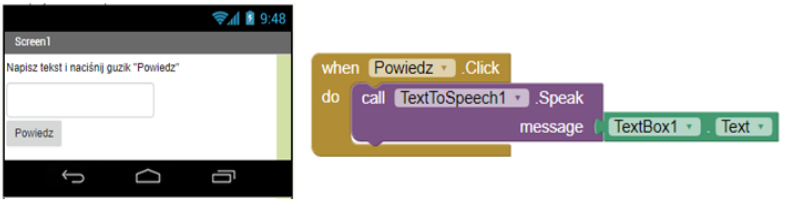
2. Początek pracy z MIT App Inventor

W celu rozpoczęcia pracy z App Inventor potrzebujemy: komputera podłączonego do Internetu za pośrednictwem stabilnego połączenia sieciowego, urządzenia mobilnego z Androidem lub jego emulatora. Na smartfonie lub tablecie trzeba zainstalować aplikację MIT AI2 Companion, która umożliwi testowanie tworzonego programu. Natomiast na stronie ai2.appinventor.mit.edu należy utworzyć nowy projekt. W ramach projektu będzie tworzony właściwy kod aplikacji. Połączenie z projektem może odbyć się na kilka sposobów. Najprostszy sposób to skorzystanie z kodu QR za pomocą

zainstalowanej na telefonie lub tablecie aplikacji MIT AI2 Companion. Dopuszczalne są też inne sposoby połączenia się z projektem jak np. kabel USB i korzystanie z emulatora Androida.

3. Pierwsze ćwiczenia wprowadzające

Użytkownicy urządzeń mobilnych często korzystają z wielu praktycznych i charakterystycznych funkcji swoich urządzeń. Należą do nich: możliwości odtworzenia napisanego tekstu jako dźwięku, rozpoznawanie mowy, wykrywanie ruchu telefonu, rysowanie palcem, przesuwanie obiektu na ekranie np. ruch w grze, reakcja obiektu na zmianę orientacji telefonu. Kilka przykładów dla zobrazowania sposobów wykorzystania wymienionych funkcji w kontekście programowania zdarzeń i uruchamiania sensorów przedstawię poniżej w charakterze ćwiczeń wprowadzających. Każdą aplikację tworzymy w dwóch trybach. Na rysunku 1 widać obie warstwy prostej aplikacji, której celem jest wypowiadanie tekstu wprowadzonego w polu tekstowym. Pierwsza warstwa aplikacji tworzona jest w trybie Blocks. W warstwie pierwszej wybierane są komponenty: Etykieta z tekstem zachęty do wprowadzenia tekstu w polu tekstowym, pole tekstowe oraz przycisk.

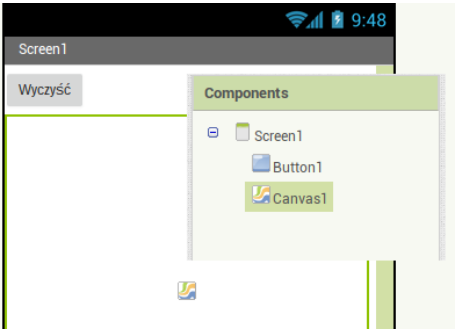


Rysunek 1

Każdy z komponentów trybu Designer ma swój bardziej lub mniej rozbudowany system właściwości ustawianych w bardzo prosty sposób. Tryb Blocks zawiera program realizowany za pomocą bloczków. Mamy tutaj obsługę zdarzenia wciśnięcia przycisku. Metoda Speek działa w ramach obiektu TextToSpeech1 uruchamianego w kontenerze call z argumentem, którego wartością jest wprowadzony tekst w polu tekstowym TexBox1. Przykład może być punktem wyjścia do korzystania z pola tekstowego w roli konsoli wejściowej i pisania prostych przykładów kształcących obsługę bloków sterujących, logicznych czy obsługę zmiennych.

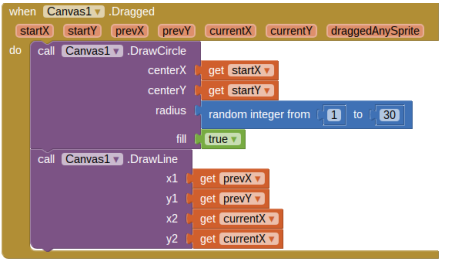
Na rysunku 1 zastosowano dwa Komponenty. Każdy z nich – czyli pole tekstowe i przycisk – projektowany jest w trybie Designer. W trybie Blocks, każdy z komponentów posiada pewien repertuar bloków, będących lista możliwych do zrealizowania operacji, właściwych danemu komponentowi. Taka filozofia konstrukcji środowiska programistycznego i jednocześnie projektowego sprzyja skupianiu uwagi na elemen-

tach istotnych i przyspiesza proces powstawania aplikacji. Zasada podobna do tej zopisanego i zilustrowanego na rysunku 1 przykładu będzie funkcjonować przy tworzeniu kolejnych prostych aplikacji wprowadzających. Drugi przykład będzie również bazować na projekcie dwóch komponentów przedstawionych na rysunku 2.



Rysunek 2

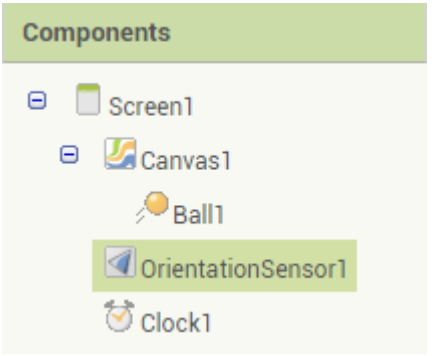
Komponenty, które są osadzone na ekranie Screen 1 (Rysunek 2) to: przycisk Wyczyść, którego celem jest czyszczenie zawartości kanwy. Na kanwie będzie powstawać rysunek. Rysowanie odręczne oznacza tutaj korzystanie z metody Dragged. Kanwa jest zdefiniowanym obszarem prostokątnym, któremu można przypisać różne właściwości. Należą do nich: kolor tła, rysunek tła, parametry czcionki, wysokość i szerokość kanwy, kolor pisaka, sposób wyrównywania tekstu. Program przedstawiony jest na rysunku 3.



Rysunek 3

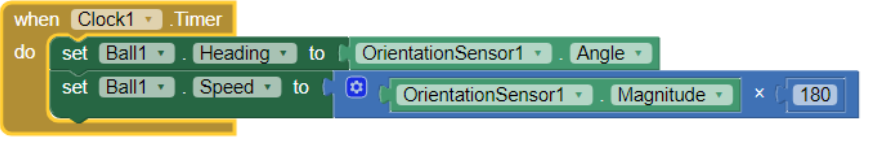
Program (Rysunek 3) uruchamiany jest w momencie przesuwania palcem po ekranie dotykowym urządzenia. Obsługa zdarzenia następuje w bloku ”when ...” z metodą Dragged. Na zilustrowanym przykładzie wywoływane są dwa efekty. Pierwszy to rysowanie koła w punkcie dotknięcia ekranu o promieniu wylosowanym z zakresu 1-10 wypełnionego kolorem wskazanym we właściwościach kanwy w trybie Designer. Drugi efekt sprowadza się do rysowania linii z punktu do punktu o współ-

rzędnych wskazywanych dotknięciem ekranu. Przycisk Wyczyść po wybraniu uruchamia metodę clear na wskazanej w kontenerze call kanwie. Warto zwrócić uwagę na możliwości korzystania z innych sensorów. Komponenty na rysunku 4 ilustrują sytuację, w której wykorzystujemy możliwość kontroli położenia (nachylenia) urządzenia. Przykład jest też demonstracją możliwości korzystania z komponentów niewidocznych na ekranie urządzenia. Należą do nich: komponent obsługi czujnika położenia oraz komponent obsługi czasu. Jedynym komponentem widocznym na kanwie jest obiekt piłeczka (Ball).



Rysunek 2

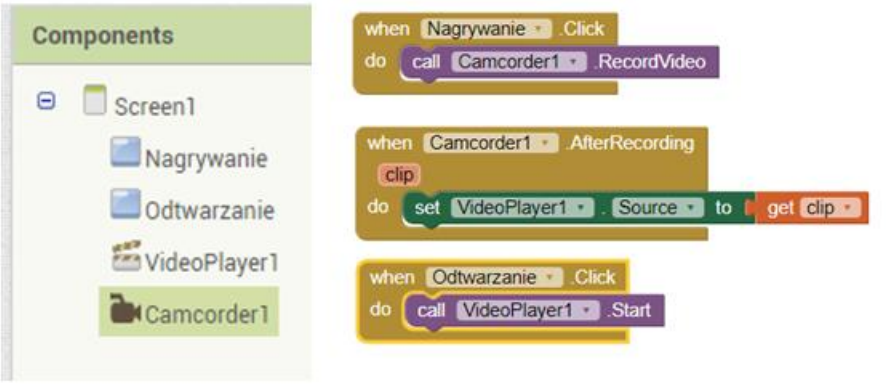
Program w trybie Blocks przedstawiony jest na rysunku 5.



Rysunek 3

Po uruchomieniu aplikacji, w czasie jej działania (when Clock1.Timer) są kolejno wykonywane dwie ważne czynności: ustawienie kierunku ruchu piłeczki w zależności od sposobu nachylenia urządzenia i przesuwanie piłeczki z prędkością zależną od tego nachylenia. Kolejny przykład związany jest również z obsługą wyposażenia technicznego każdego współczesnego smartfona. Przykład pokazuje sposoby obsługi zdarzeń i akcji inicjowanych przez użytkownika, który korzysta z konkretnego wyposażenia swojego urządzenia mobilnego. Na rysunku 6 zawarto komponenty projektu. Należą do nich widoczne: dwa przyciski inicjujące określone akcje oraz VideoPlayer, oraz ukryty komponent urządzenia nagrywającego wideo (Camcorder). Przykład pokazuje możliwość przechwytywania obrazu ruchomego i przekazywania do aplikacji.

Podobnie można wyobrazić sobie przechwytywanie zdjęć z aparatu fotograficznego (komponent Camera).



Rysunek 4

Odpowiednio dobierane proste przykłady dość szybko przeprowadzą uczniów przez podstawy korzystania z App Inventor 2. Uczniowie widząc prawie natychmiast efekty swojej pracy, będą chętniej podejmować próby sięgania po bardziej skomplikowane konstrukcje, będą chętniej stawiać sobie cele i przyswajać algorytmy przydatne przy ich tworzeniu.

4. Projekty, algorytmy, programowanie

W podstawie programowej nauczania informatyki występują tematy, których realizacja w postaci projektu aplikacji mobilnych może przynieść bardzo dobre rezultaty. Z jednej strony osiągamy cel dydaktyczny z drugiej przygotowujemy ucznia do przyszłych wyzwań, pokazując, że algorytmy z kanonu maturalnego są rzeczywiście przydatne. Jednym z większych tego typu projektów może być symulacja działania gry edukacyjnej w wieże Hanoi. Zadanie jako bardziej rozbudowana propozycja zajęć porusza wiele wątków i daje możliwości trenowania ważnych umiejętności. Zaliczamy do nich umiejętność dekompozycji problemu, abstrakcyjnego myślenia, umiejętności ewaluacji rozwiązania[3]. W trakcie projektu mamy możliwość kształtowania umiejętności korzystania z przycisków, kanwy, aranżacji poziomej i pionowej obrazu, ikon, etykiet, komponentów notyfikacji, korzystanie z wirtualnych ekranów. Projekt może kształtować też zrozumienie zmiennych lokalnych i globalnych, stosowanie procedur z parametrami i bez parametrów, stosowanie myślenia komputacyjnego przy rozwiązywania problemów. Każdy projekt rozpoczynamy od doboru różnych komponentów na podstawie stworzonego wcześniej planu – schematu poglądowo-funkcjonalnego

interfejsu naszej aplikacji. Etap tworzenia interfejsu w trybie Designer obejmuje różne czynności, ale wszystkie można ująć ogólnie w punktach.

1. Ustawianie właściwości określonych ekranów, w tym wirtualnych ekranów aplikacji.
2. Określenie parametrów kanwy – jeśli będzie używana.
3. Wybór komponentów z wymaganych kategorii i opracowanie ich właściwości.
4. Import i ustawienia właściwości plików graficznych, dźwiękowych, filmowych. Ta czynność często jest powiązana z czynnościami: 1, 2, 3

Przy okazji czynności samego projektu w trybie Designer uczeń może się zapoznać z poszczególnymi pojęciami w obszarze środowiska programistycznego App Inventor 2 oraz lepiej zrozumieć funkcje poszczególnych komponentów. Schemat funkcjonalny powinien zawierać scenariusz wywoływania i przebiegu akcji związanych z samym programem. Etap tworzenia schematu jest też etapem określania algorytmu. Jego kodowanie nastąpi w warstwie Blocks. Trening opisanej procedury postępowania można przeprowadzić na bardzo prostych pomysłach.

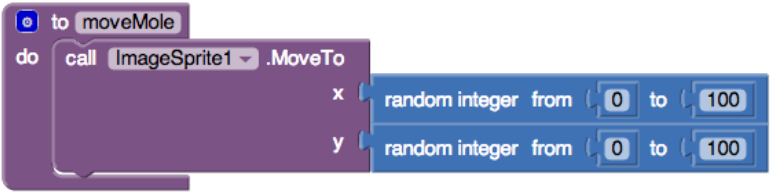
5. Zmienne, dane złożone, procedury

Przykładem aplikacji, w której jest zawarta obsługa danych złożonych i dynamicznych jednocześnie, może być aplikacja umożliwiająca wprowadzanie notatek[2]. Dane zapisane za pomocą pełnej aplikacji mają charakter trwały. W aplikacji tej może być wykorzystana następująca konstrukcja bloków (Rys. 7).

Rysunek 5

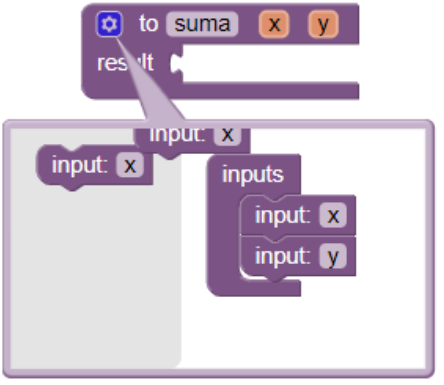
Na użytek programu, tworzona jest zmienna noteList, która ma charakter listy. Elementy listy pochodzą z pola tekstowego NoteTextBox i są dodawane do niej po każdym wybraniu przycisku SubmitButtom. Jednocześnie mamy tutaj przykład obsługi

konsoli wyjściowej ListLabel oraz zerowania samego pola tekstowego po zatwierdzeniu wprowadzonych danych. Widzimy też wyraźnie globalny charakter zmiennej noteList. Przykład (rys. 7) pokazuje, że środowisko MIT App Inventor udostępnia wystarczający zakres obiektów i metod umożliwiających obsługę danych i zdarzeń, również tych, z którymi uczeń ma kontakt przy okazji innych środowisk czy języków programowania. MIT App Inventor podobnie jak Scratch, Blokly i podobne środowiska programowania wizualnego umożliwia budowę kodów zgrupowanych w bloki proceduralne z regulowanymi opcjami liczby parametrów wejściowych i możliwościami zwracania wartości(procedura funkcyjna).



Rysunek 6

Na rysunku 8 przykładowa klasyczna procedura wykonująca zleconą czynność, ale bez możliwości zwrócenia wartości.



Rysunek 7

Rysunek 9 ilustruje możliwość zdefiniowania listy argumentów (danych wejściowych) i jednocześnie określenie wyrażenia – żądanego wyniku w bloku proceduralnym z opcją result. Mamy więc klasyczną procedurę funkcyjną. Tego typu konstrukcje łącznie ze zmiennymi o zasięgu globalnym oraz hermetyczne zmienne lokalne, łącznie z możliwościami komunikacji między obiektami(komponentami projektu) stanowi bar-

dzo wygodny system nie tylko do uczenia się kodowania, ale również jest cenną pomocą dydaktyczną dla nauczyciela.

6. Podsumowanie

Środowisko App Inventor zostało zaprojektowane dla szerokiego grona użytkowników chcących podjąć wyzwanie programowania aplikacji dla Androida. W rezultacie oferuje bogaty zestaw narzędzi i jest dość wygodnym środowiskiem dydaktycznym. Kolejnym atutem jest dostęp do wskazówek i narzędzi stanowiących cenną pomóc dla nauczycieli, chcących atrakcyjnie i skutecznie prowadzić naukę programowania w szkole. Związana z systemem, rozbudowana warstwa instrukcji, przykładów i społeczności[2] stanowią też duże wsparcie dla osób niebędących informatykami, a chcących skutecznie poznać zasady kodowania na przystępnym poziomie. Głównym atutem stosowania MIT App Inventor na lekcjach w szkole jest duża łatwość tworzenia kodów oraz możliwość zainteresowania szerszego grona uczniów nauką programowania. Środowisko uznane i używane również w ramach globalnych akcji edukacyjnych np. Godziny Kodowania. Bardzo ważną cechą środowiska jest też możliwość tworzenia projektów w trybie zespołowym, importowania i eksportowania plików, uruchamiania aplikacji na urządzeniach mobilnych podłączonych logicznie do projektu poprzez sieć lub korzystanie z emulatora urządzenia z Androidem.

Literatura

1. D.Wolber, H.Abelson*, E. Spertus, L. Looney, App Invetor 2*, 2015, O'Reyly Media Inc
2. http://www.appinventor.org , ostatni dostęp: 1.06.2019
3. D. Harel*, Rzecz o istocie informatyki*, WNT, Warszawa 2008.