App Inventor 2

NA PODSTAWIE MATERIAŁÓW STWORZONYCH PRZEZ TRENERÓW PROGRAMU „MISTRZOWIE KODOWANIA”

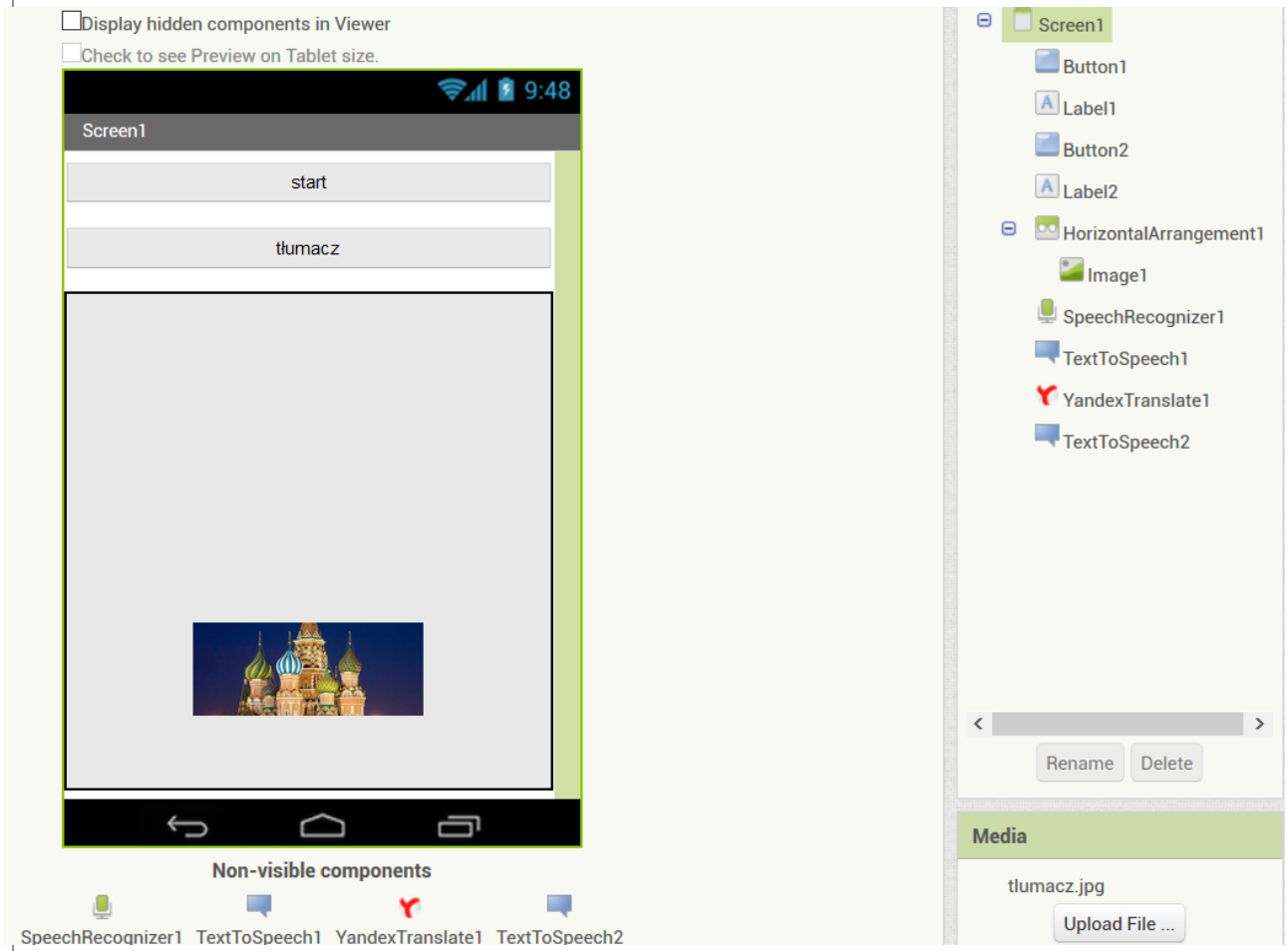
http://wiki.mistrzowiekodowania.pl

Tłumacz głosu

Tłumacz głosu to projekt pokazujący wykorzystanie telefonu jako podręcznego elektronicznego tłumacza, przy czym nasz tekst do przetłumaczenia nie będzie wpisywany, lecz mówiony.

Wejdźmy na stronę http://appinventor.mit.edu i wybierzmy Create, by uruchomić App Inventor. Następnie przejdźmy do zakładki My Projects (1), gdzie znajdziemy wszystkie nasze projekty - wystarczy tylko zalogować się na konto Google. W celu rozpoczęcia pracy nad nowym projektem kliknijmy na Start new project (2) i wpiszmy w oknie jego nazwę, np. Tlumacz_glosu.

Na początek przejdźmy do okna **Properties** i dla komponentu **Screen1**, w polu **Screen Orientation** wybierzmy **Portrait**. Dzięki temu nasz ekran zawsze będzie wyświetlany pionowo, zaś aplikacja będzie pracowała tylko w takiej orientacji ekranu. Dodatkowo w polu **Title** wpiszmy tytuł naszego bieżącego komponentu. Nazwijmy go np.: Tłumacz.
Teraz zacznijmy po kolei przeciągać na ekran button, label, button i jeszcze jeden label. Znajdziemy je w kategorii User Interface.
Następnie z kategorii Media przeciągnijmy: SpeechRecognizer, dwa razy TextToSpeech oraz YandexTranslate. Pojawią się one pod ekranem jako **Non-visible components**, czyli komponenty niewidoczne.



Na początek przejdźmy do okna **Properties** i dla komponentu **Screen1**, w polu **Screen Orientation** wybierzmy **Portrait**. Dzięki temu nasz ekran zawsze będzie wyświetlany pionowo, zaś aplikacja będzie pracowała tylko w takiej orientacji ekranu. Dodatkowo w polu **Title** wpiszmy tytuł naszego bieżącego komponentu. Nazwijmy go np.: Tłumacz.

Teraz zacznijmy po kolei przeciągać na ekran button, label, button i jeszcze jeden label. Znajdziemy je w kategorii User Interface.
Następnie z kategorii Media przeciągnijmy: SpeechRecognizer, dwa razy TextToSpeech oraz YandexTranslate. Pojawią się one pod ekranem jako **Non-visible components**, czyli komponenty niewidoczne.

Ustawimy teraz właściwości poszczególnych elementów:

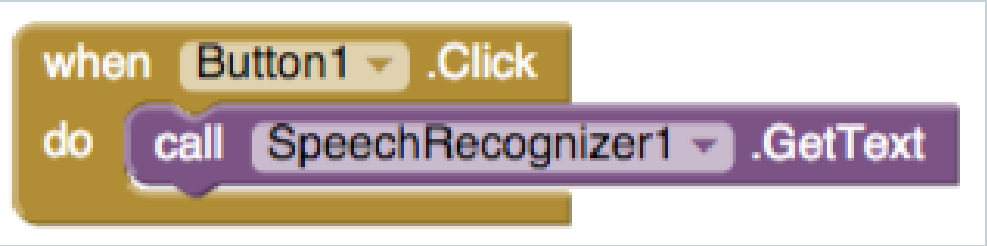
Button1 - tutaj zmienimy szerokość (width) na Fill Parent, w Text wpiszemy Start, Text Aligment ustawimy na center

Button2 - tutaj zmienimy szerokość (width) na Fill Parent, w Text wpiszemy Tłumacz, Text Aligment ustawimy na center

Label1 i **Label2** - tutaj wyczyścimy tekst w polu Text
Pozostałe komponenty nie wymagają od nas zmian w ustawieniach.

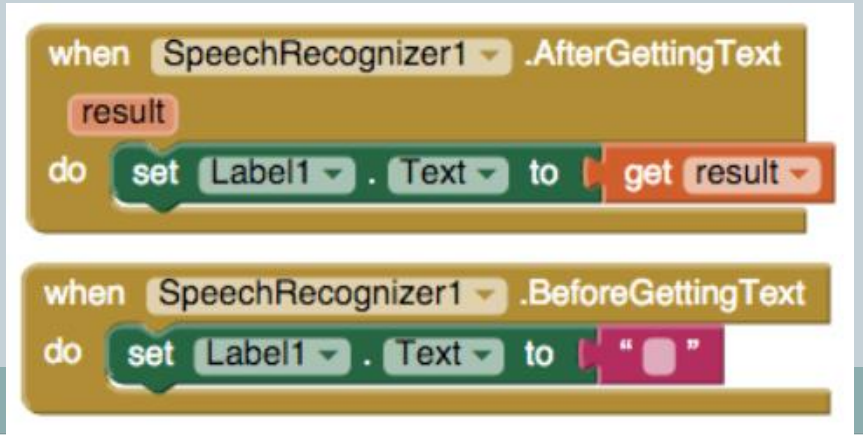
Kodujemy

- Czas przejść do zakładki Blocks. Programowanie naszej aplikacji zaczniemy od przypisania funkcji komponentowi button1. Zaczynamy od wybrania z kategorii Blocks/Screen1/Button1 **whenButton1Click**, do którego włożymy blok callSpeechRecognizer1.getText (znajdziemy go w SpeechRecognizer1). Spowoduje to wywołanie funkcji rozpoznawania mowy w telefonie.



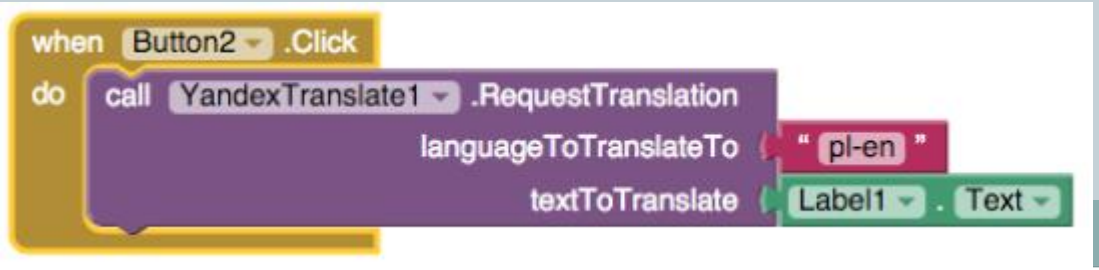
Kodujemy

- Kolejnym krokiem będzie wyświetlenie na ekranie tego, co powiedzieliśmy po kliknięciu Button1. Do tego posłuży nam Label1, za pomocą którego wyświetlimy jako tekst to, co usłyszał nasz telefon. Może się to okazać pomocne przy weryfikowaniu, czy funkcja rozpoznawania głosu dobrze nas usłyszała i rozpoznała, co powiedzieliśmy.
- Wybieramy **WhenSpeechRecognizer.afterGettingText** i do środka wstawiamy **setLabel1.textTo** oraz **getResult**. GetResult jest zmienną lokalną, więc wyciągniemy ją, najeżdżając myszką na result w klocku **WhenSpeechRecognizer.afterGettingText**.
- Do tego warto pomyśleć o tym, aby metka Label1 pozostała czysta przed kliknięciem Button1 – ustawimy więc wartość Label1 na pustą. Posłuży nam do tego pusty string (" ") czyli pole z pustym tekstem, które znajdziemy w kategorii Text.



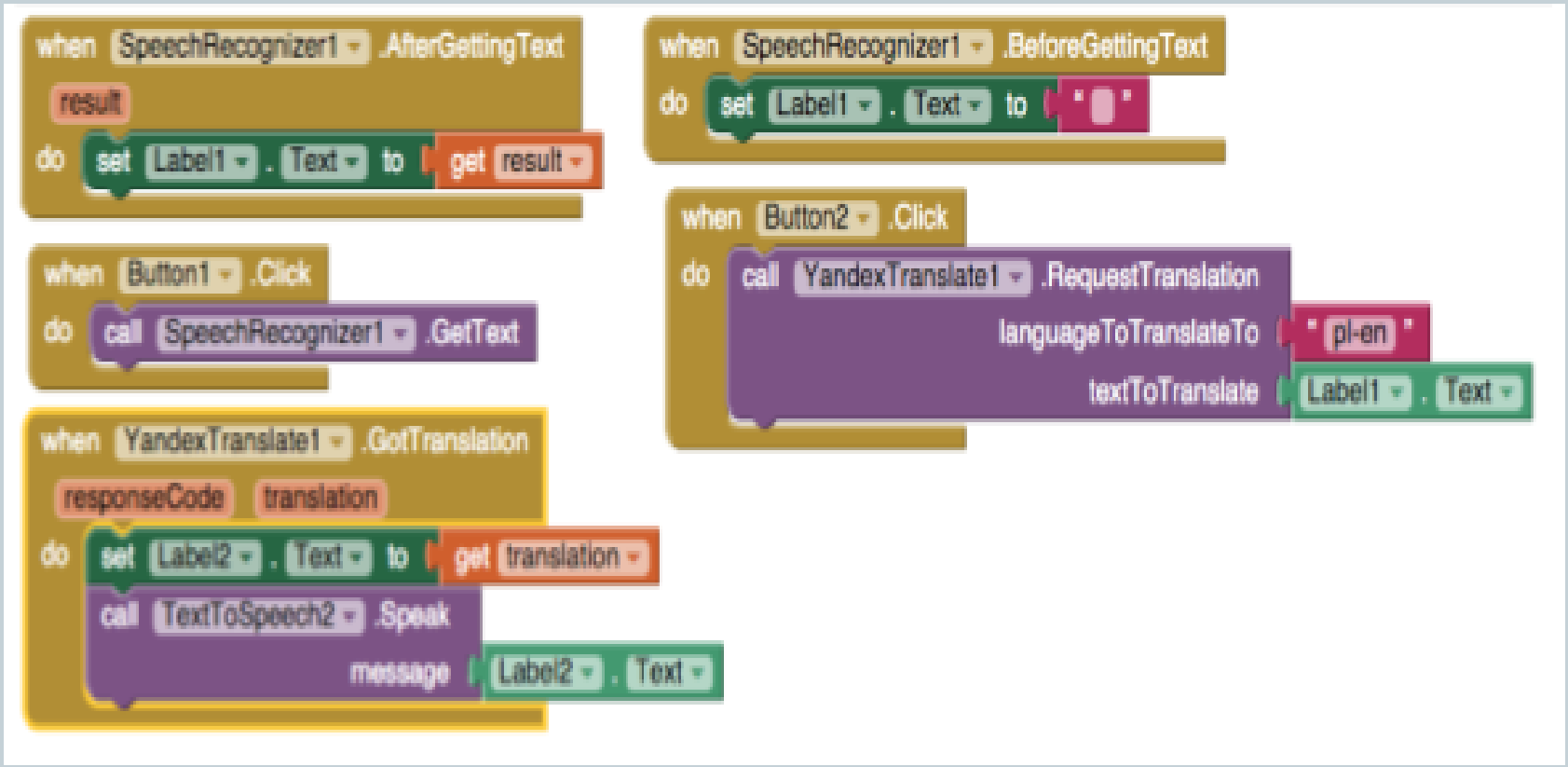
Tłumaczymy

- Jeżeli nasz program prawidłowo rozpoznaje i zapisuje to, co powiedzieliśmy, czas zaprogramować funkcję tłumaczenia. Będziemy chcieli, aby nasz program tłumaczył z języka polskiego na język angielski. Wykorzystamy do tego komponent **YandexTranslate**. Wywoływać go będziemy poprzez kliknięcie **Button2**, a tłumaczony będzie tekst zapisany w **Label1**.
- Potrzebujemy więc bloczka **WhenButton2.Click** oraz **CallYandexTranslate1.RequestTranslation**.
- CallYandexTranslate1.RequestTranslation ma miejsce na przyłączenie dwóch bloczków:
- **languageToTranslateTo -** tutaj wpiszemy z jakiego języka na jaki chcemy tłumaczyć np. "pl-en".
- **textToTranslate** - tutaj przyłączymy label1.text z którego pobierzemy tekst do tłumaczenia.



Powyższa operacja powoduje pobranie i przetłumaczenie tekstu z Label1. Nie widzimy jednak jeszcze wyniku tego tłumaczenia, gdyż nie zaprogramowaliśmy, żeby przetłumaczony tekst pojawiał się w aplikacji. Posłużyć do tego może nam kolejna metka Label2, w której wyświetlimy efekt tłumaczenia.
Programowanie ostatniego skryptu rozpoczniemy od znalezienia w YandexTranslate1 warunku **WhenYandexTranslate1.GotTranslation** i umiejscowienia w warunku setLabel2.TextTo połączonego ze zmienną lokalną getTranslation (wyciągniemy ją ze zmiennej lokalnej **translation** w **WhenYandexTranslate1.GotTranslation**).
Tak skomponowany kod wyświetli nam przetłumaczony tekst. Jednak chcąc, aby ten tekst był przeczytany, warto dodać jeszcze dwa bloczki: **callTextToSpeech2.SpeakMessage** oraz **label2.Text**. Spowodują one przeczytanie przez syntezator mowy przetłumaczonego tekstu.

Kompletny kod

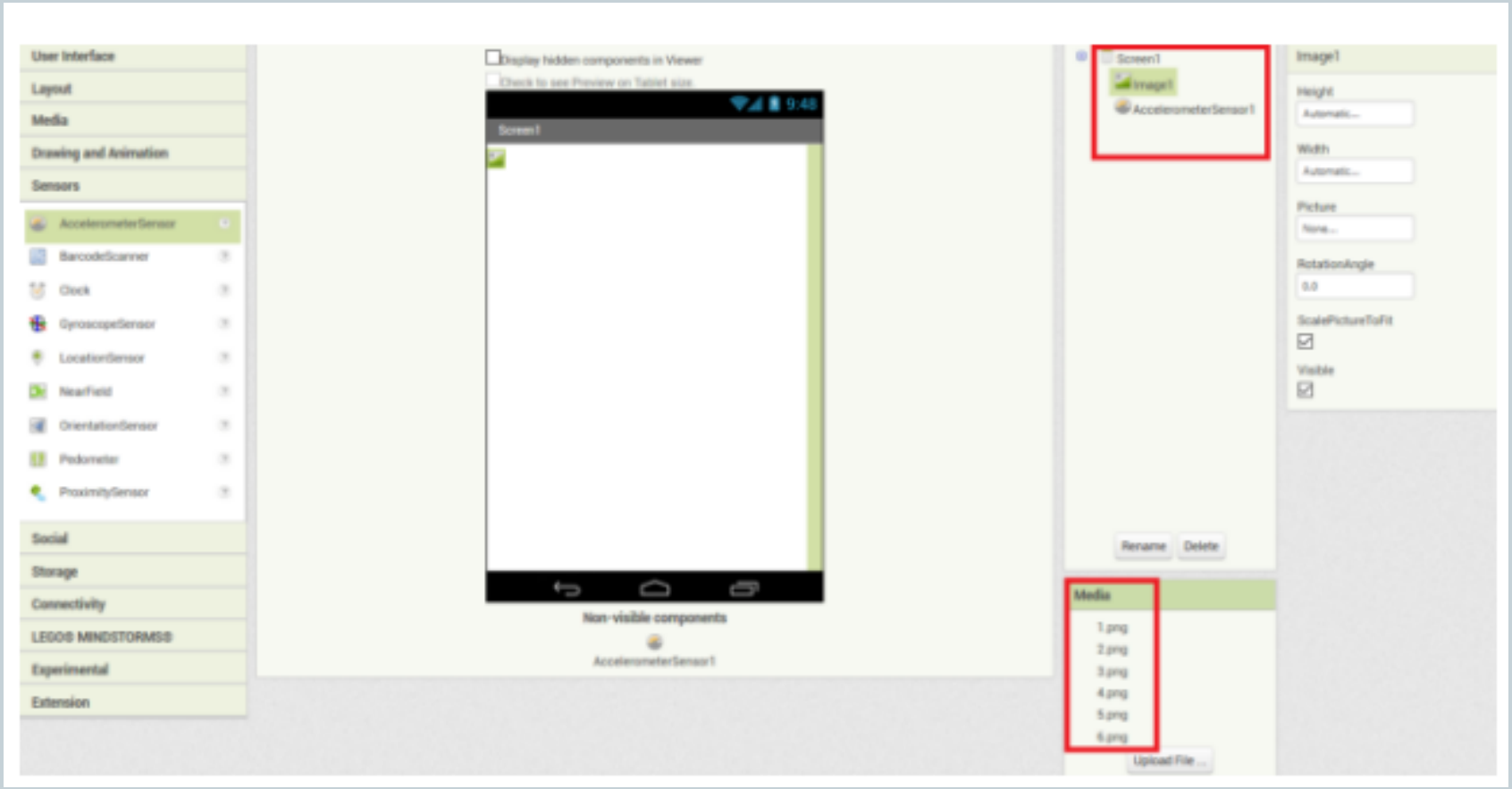


Rzucamy kostką

- Po włączeniu okna nowego projektu, zabieramy się za projektowanie wyglądu naszej aplikacji. Chcemy, by na ekranie - po potrząśnieciu telefonem czy tabletem - pojawił się jeden z sześciu obrazków, dlatego też z zakładki **User Interface** przeciągamy na **Screen 1** komponent **Image**, a z zakładki **Sensors** - **AccelerometerSensor** (jest to komponent niewidoczny na ekranie, jego dodanie zostanie odnotowane pod Screen1).
- Kolejnym etapem jest wczytanie sześciu grafik jako **Media** projektu Ważne dla naszej późniejszej pracy jest to, że ich nazwy pochodzą od liczby oczek na poszczególnych grafikach, a każdy plik ma rozszerzenie ".png" (1.png - 6.png). (około 600x600 pikseli)




Gotowy interfejs



Kodujemy

Czas przejść do zakładki Blocks. Skrypt składa się z zaledwie kilku bloczków. Skoro grafiki mają się ukazywać na ekranie w losowej kolejności po potrząśnięciu urządzeniem, wyciągnijmy z bloczków AccelerometerSensor - **when AccelerometerSensor1.Shaking** i włóżmy do jego wnętrza bloczek odpowiedzialny za ustawienie grafiki do wyświetlania na komponencie Image - **set Image1.Picture to**.

Teraz wystarczy wskazać, jak nazywają się poszczególne pliki z wczytanej biblioteki Mediów. Każda nazwa ma część wspólną - rozszerzenie ".png", natomiast różni się numerem. Dlatego też połączymy ze sobą (za pomocą bloczka join) losowo wybraną liczbę całkowitą z zakresu 1-6 (**random integer from 1 to 6**) oraz tekst ".png".