PROGRAMOVANIE MOBILNÝCH ZARIADENÍ

ĽUBOMÍR ŠNAJDER, GABRIELA LOVÁSZOVÁ, VIERA MICHALIČKOVÁ, JÁN GUNIŠ

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE

MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY

it akadémia

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu
v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

Programovanie mobilných zariadení

Spracované v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Bratislava 2020

Programovanie mobilných zariadení

Spracované s finančnou podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Autori: Ľubomír Šnajder, Gabriela Lovászová, Viera Michaličková, Ján Guniš

Editor: Ľubomír Šnajder

Recenzenti: Ján Mazák, Róbert Novotný

Neprešlo jazykovou úpravou

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava

Rok vydania: 2020

Vydanie: 1. vydanie

Bratislava 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.

OBSAH

ÚVOD

Milí žiaci,

do rúk sa Vám dostáva učebnica k výučbe samostatného informatického predmetu **programovania mobilných zariadení** (v blokovom prostredí MIT App Inventor 2, ďalej len Ai2). Našou snahou je ukázať, že androidové mobilné zariadenie (ďalej len MZ) nemusíte používať len ako konzumenti, ale aj ako autori vlastných softvérových aplikácií. Rovnako, že pri programovaní môžete vytvárať užitočné softvérové aplikácie, ktoré poslúžia vám a tiež ostatným ľuďom na zábavu, vzdelávanie a podporu rôznych aktivít. Aby sme oslovili čo najviac z vás, vybrali sme blokové programovacie prostredie, ktoré „skrýva“ viaceré menej podstatné technické detaily a umožní vám vytvárať jednoduché a stredne náročné softvérové aplikácie.

Autormi publikácie sú vysokoškolskí učitelia z Univerzity Pavla Jozefa Šafárika v Košiciach a Univerzity Konštantína Filozofa v Nitre, ktorí sa podieľali pri tvorbe jednotlivých kapitol, resp. podkapitol nasledovne: Ľubomír Šnajder (Úvod, 1, 2, 3.1, 4.2, 6.1, Prílohy), Gabriela Lovászová (3.2, 3.3, 4.5, 5.1), Viera Michaličková (4.3, 4.4, 5.2, 6.2) a Ján Guniš (4.1).

Ďakujeme recenzentom tejto publikácie a tiež učiteľom stredných škôl zapojených do **národného projektu IT Akadémia – vzdelávanie pre 21. storočie**, ktorí overovali či používali tento učebný text spolu s pracovnými súbormi, za poskytnutie cenných pripomienok a návrhov, ktoré veľkou mierou prispeli k požadovanej úrovni finálnej verzie tohto učebného textu.

Autori

Štruktúra a obsah učebnice

Obsah učebnice

Učebnica je rozčlenená do šiestich kapitol a príloh (textovej a elektronickej).

1 **Prvé kroky pri práci s OS Android a tvorba prvej aplikácie v Ai2**

1.1 **Spoznávajme androidové mobilné zariadenie**
(Zisťovanie parametrov MZ včítane vstavaných senzorov a zmena nastavení MZ, práca so súborovým systémom a správa procesov OS Android, prehľad a použitie vybraných aplikácií na získanie, uloženie a prenos údajov z/na MZ (napr. čítačky QR kódov), diskusia o existujúcich aplikáciách a o návrhu vlastných aplikácií)

1.2 **Poďme vytvoriť prvú aplikáciu v MIT App Inventore 2**
(Životný cyklus vývoja aplikácie pre MZ v cloudovom prostredí Ai2, prihlásenie sa do Ai2, vytváranie aplikácie v režimoch *Designer* a *Blocks*, kompilácia programového kódu, uloženie inštalačného balíka a jeho inštalácia na MZ, export a import zdrojových kódov programov z Ai2 na lokálny počítač)

2 **Tvorba malých aplikácií**
Kapitolu tvorí 12 malých aplikácií, pri programovaní ktorých sa využívajú vybrané funkcionality Ai2:

2.1 **Kresliaci editor**

2.2 **Hra Postreh**

2.3 **Hra Guľka**

2.4 **Kalkulačka**

2.5 **Zbierka vtipov**

2.6 **Čítačka QR kódu**

2.7 **Asistent pri cvičení**

2.8 **Generátor náhodných viet**

2.9 **Zobrazovač aktuálnej polohy**

2.10 **Asistent aktuálnej polohy**

2.11 **Hlasovanie na internete**

2.12 **Komunikačný asistent**

V ďalších štyroch kapitolách je uvedená tvorba komplexných užitočných projektov

3 **Multimédiá** – animácia, zvuk, reč, video, QR kódy

3.1 **Multimediálny zápisník pre mladého reportéra**
Aplikácia na záznam zvukov a fotografií zo vstavaného mikrofónu a fotoaparátu. Ďalšími možnosťami aplikácie sú dokreslenie a doplnenie textu do získanej fotografie, uloženie a načítanie upravenej fotografie. V aplikácii sú použité

multimediálne komponenty: `Camera`, `ImagePicker`, `Player`, `SoundRecorder`.

3.2 **Dychový tréner**
Aplikácia na manažovanie dychového tréningu pre potápačov. Cyklicky sa striedajú fázy zadržania dychu a predýchania. Aplikácia odpočítava čas, graficky znázorňuje fázu tréningu (zadržanie dychu alebo predýchavanie), graficky a zvukovo signalizuje koniec fázy, umožňuje rôzne nastavenia parametrov tréningu. V aplikácii sú použité multimediálne komponenty: `TextToSpeech`, `Sound`, `Ball`.

3.3 **Asistent prvej pomoci**
Aplikácia použiteľná pri záchrane pomoci. Umožňuje používateľovi prečítať si návody prvej pomoci, resp. si ich vypočuť syntetickou rečou. Navyše asistuje pri resuscitácii a pri vytočení tiesňovej telefónnej linky. V aplikácii sú použité multimediálne komponenty: `VideoPlayer`, `TextToSpeech`, `Sound`.

4 **Siete** – web prehliadač, lokálna a webová databáza, Bluetooth

4.1 **Záznamník terénnych dát**
Aplikácia pre zber dát v teréne. Dáta sa ukladajú do lokálneho CSV súboru a je možné ich posielať aj na online server. Budeme sa zaoberať aj použiteľnosťou aplikácie.

4.2 **Hlasovací systém**
Aplikácia pre hlasovanie žiakov a správu odpovedí. Využijeme online databázu *Firebase* pre záznam odpovedí. Aplikácia má dve časti – učiteľskú a žiacku.

4.3 **Pomocník pri učení sa cudzieho jazyka**
Prekladač viet. Aplikácia prekladá vety zadané textom alebo hlasom. Na preklad využijeme webovú službu *Yandex*.

4.4 **Spoločenská hra pre tablet**
Hráč háda slovo podľa opisu protihráča. Zoznam slov môže byť uložený v online databáze `TinyWebDB`. Rôzne časti aplikácie vyvíjajú v rámci tímu viacerí programátori. Ich časti aplikácie sa nakoniec spoja pomocou nástroja *App Inventor Merger* do jedného celku.

4.5 **Kockový poker**
Hra poker pre dvoch hráčov. Aplikácia riadi hru dvoch hráčov, vyhodnocuje hody a zobrazuje výsledky hráčov. Na vzájomnú komunikáciu hráčskych tabletov využijeme Bluetooth.

5 Geolokácia – senzor polohy

5.1 **Reverse caching**
Aplikácia, ktorá približuje princíp určovania geografickej polohy pomocou trilaterácie. Pomocou aplikácie používateľ hľadá miesto v teréne, kde môže byť ukrytá schránka s „pokladom“ (cache), na základe informácie o vzdialenosti k cieľu bez udania smeru. V aplikácii sú použité komponenty `Map` a `LocationSensor` na prácu s geodátami.

5.2 **Geolokačná hra**
Akčná pohybová exteriérová hra na získavanie bodov, ktorú môžu hrať formou súťaže medzi sebou jednotlivci alebo tímy. Žiaci ju majú remixovať (vytvoriť vlastnú verziu hry rozšírením, upravením, obmenou námetu pôvodnej hry). V aplikácii je použitý komponent `LocationSensor` na prácu s geodátami a komponent `Clock` na meranie času.

6 Senzory a aktuátory – senzor zrýchlenia, priblíženia, orientácie, ovládanie robotických modelov

6.1 **Tréner cvikov pre pacientov a športovcov**
Aplikácia podporujúca používateľa pri vykonávaní fyzickej aktivity. Aplikácia zaznamenáva počet cvikov a čas cvičenia, v priebehu vykonávania cvičení poskytuje pravidelnú zvukovú signalizáciu alebo hlasový komentár (pomocou komponentov `Sound` a `TextToSpeech`), umožňuje získať a uchovávať jednoduchú štatistiku o predchádzajúcich tréningoch (s využitím lokálnej databázy).

6.2 **Hra ovládaná dotykovými gestami**
Viacúrovňová hra typu „plošinovka“ založená na dotykovom ovládaní. Obsahuje statické aj pohybujúce sa grafické objekty a zvukové efekty. Zmenu stavu sprajtov, resp. plánovanie udalostí v hre zabezpečíme pomocou komponentu `Clock`.

Bibliografia

Register pojmov

Tlačená príloha

- **Inštalácia podporných nástrojov pre vývoj aplikácií v Ai2**
- **Prehľad komponentov a štandardných blokov Ai2**
- **Zdieľanie vyvinutej aplikácie ostatným používateľom**

Súčasťou učebnice je aj elektronická príloha s priečinkami pre každú kapitolu, resp. podkapitolu, ktoré obsahujú zdrojové kódy rozpracovaných riešení úloh vo formáte *.aia a multimediálne a iné dátové súbory.

Štruktúra kapitol a podkapitol

Jednotlivé kapitoly a podkapitoly učiva obsahujú nasledovné časti:

- Úvod do kapitoly
- Kľúčové slová
- Čo sa naučíme a čo si precvičíme (očakávané učebné ciele kapitoly)
- Motivačný úvod (pod)kapitoly
- Návrh tvorby aplikácie (návrh používateľského rozhrania, návrh správania)
- Postupnosť zadaní čiastkových úloh
- Vysvetlíme si (vysvetlenie nového učiva)
- Otázky na zamyslenie (k riešenému problému, k alternatívnym postupom ...)
- Poznámka (k riešenému problému, k vytváranej aplikácii, ku komponentom ...)
- Ako vylepšiť či rozšíriť našu aplikáciu? (námety na rozšírenia projektov, na nové projekty, na ďalšie študijné zdroje)
- Zamyslime sa, čo sme sa naučili (sumarizácia osvojeného učiva)

1 Prvé kroky pri práci s OS Android a tvorba prvej aplikácie v Ai2

1.1 Spoznávajme androidové mobilné zariadenie

Kľúčové slová

OS Android, nastavenia parametrov OS Android, vstavané senzory, súborový systém, správa procesov, softvérové aplikácie

Čo sa naučíme a čo si precvičíme

- Zistiť parametre a meniť nastavenia androidového MZ, včítane jeho vstavaných senzorov.
- Pomocou androidového MZ získať, uložiť, resp. preniesť údajové a programové súbory.
- V OS Android zistiť zoznam bežiacich procesov v pamäti a ukončiť ich vykonávanie.
- Kriticky diskutovať o funkcionalitách existujúcich softvérových aplikácií a navrhnúť funkcionality vlastnej aplikácie pre androidové MZ.

Aké parametre má moje androidové MZ?

Úloha 1 (prieskum o dostupnosti a používaní MZ)

Najprv si urobme malý prieskum o dostupnosti a používaní MZ. Prieskumom chceme zistiť: aké percento spolužiakov má vlastné MZ, aké percento spolužiakov používa zdieľané MZ s inou osobou, aké typy MZ sa používajú, aké zastúpenie majú jednotlivé OS. Zozbierajte údaje od spolužiakov pomocou zdieľaného *Google formulára* a zosumarizujte a interpretujte výsledky vášho prieskumu. Webovú adresu tohto formulára vám oznámi váš učiteľ.

Úloha 2 (parametre môjho MZ)

Zistite špecifikáciu vášho MZ. Zamerajte sa hlavne na parametre: názov a číslo modelu, verziu operačného systému, kapacitu vnútornej pamäte (RAM), kapacitu úložiska zariadenia/ ukladacieho priestoru a kapacitu prenosného úložiska (karty SD).

Napríklad:

- Galaxy S10+, SM-G975F, Android 10, pamäť 8 GB, úložisko 128 GB, karta SD 29,7 GB,
- Lenovo YT-X705F, Android 9, RAM 4 GB, ukladacie zariadenie 64 GB,
- Galaxy A5 (2016), SM-A510F, Android 7.0, pamäť RAM 2 GB, úložisko zariadenia 16 GB, prenosné úložisko karta SD 29,71 GB.

Vysvetlíme si

Uvedené časti špecifikácie MZ môžete zistiť pomocou ponuky Nastavenia a jej podponúk, ktoré sa odlišujú v jednotlivých verziách OS Android napr.

Android 10, smartfón

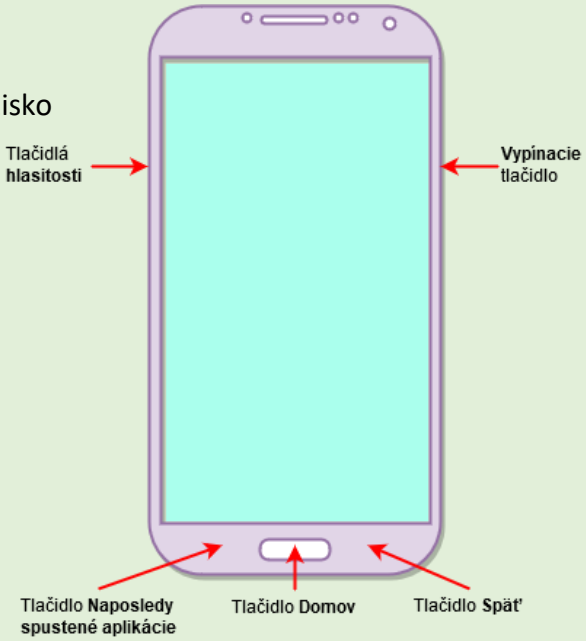
- Nastavenia / Informácie o telefóne / Informácie o softvéri
- Nastavenia / Starostlivosť o zariadenie
- Nastavenia / Starostlivosť o zariadenie / Úložisko

Android 9, tablet

- Nastavenia / Systém / Informácie o tablete
- Nastavenia / Úložisko /
- Nastavenia / Systém / Informácie o tablete / Hardvérové informácie

Android 7.0, smartfón

- Nastavenia / Informácie o telefóne
- Nastavenia / Údržba zariadenia / Pamäť
- Nastavenia / Údržba zariadenia / Úložisko



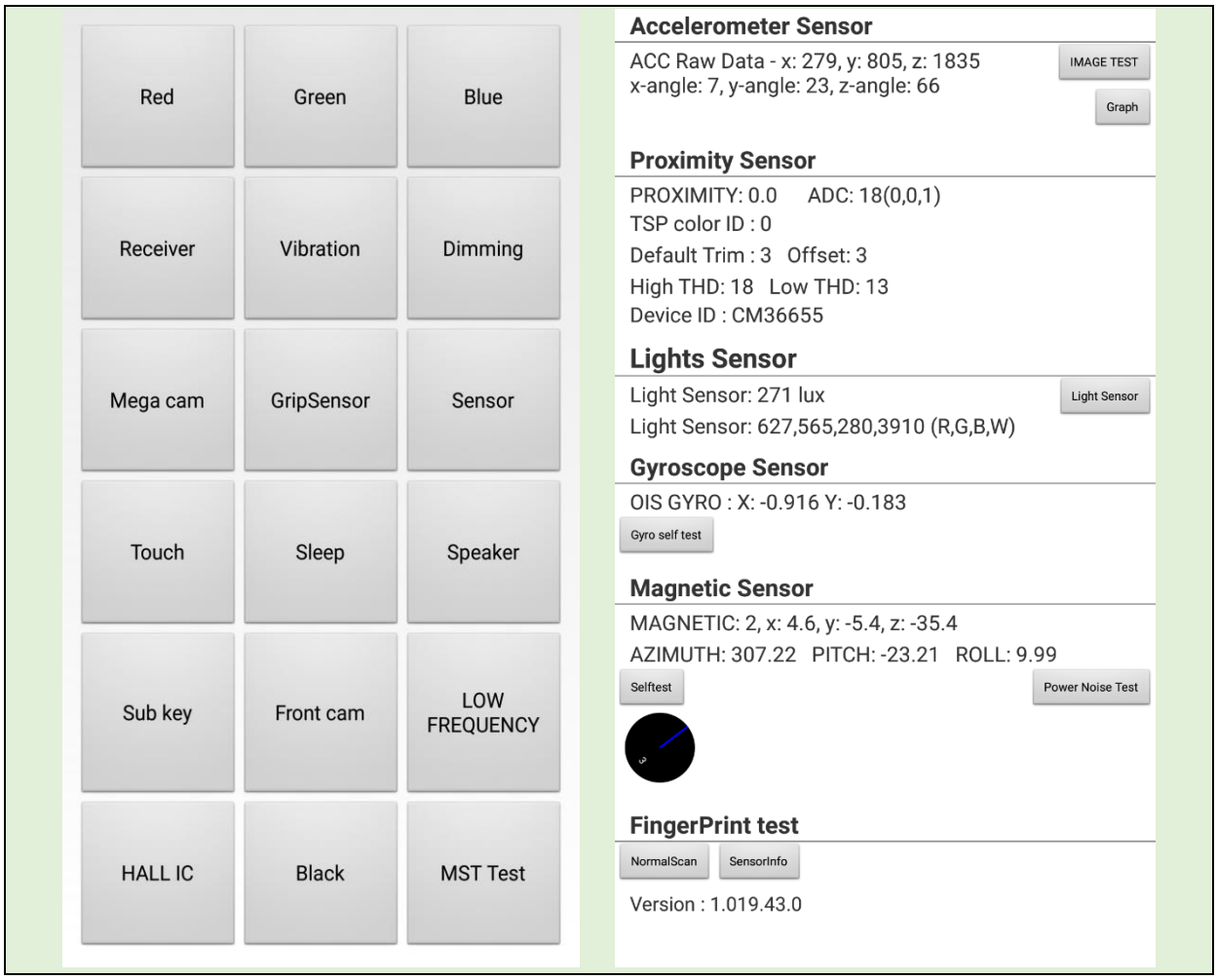
Úloha 3 (aké vstavané senzory má moje MZ)

Zistite, aké vstavané senzory má vaše MZ. Ako reagujú jednotlivé senzory na vašu manipuláciu s MZ? Prediskutujte aký význam majú jednotlivé senzory.

Poznámka

Ak nenájdete na MZ žiadnu aplikáciu na zobrazenie parametrov vstavaných senzorov, môžete si z Google Play nainštalovať a použiť vhodnú aplikáciu, napr. *Device Info : View Device Information* (Yasiru Nayanajith), či *Sensoroid – Sensor info* (3k Developers).

Špeciálne pre Samsung smartfóny sa po vytočení čísla ***#0*#** v telefóne zobrazí prehľad funkcionalít smartfónu a jeho senzorov, ktoré sa dajú otestovať.



Ktoré softvérové aplikácie a ako môžem využívať na mojom MZ?

Úloha 4 (získanie a uloženie údajov na MZ a ich prenos na iné zariadenie)

Otvorte v MZ svoju obľúbenú softvérovú aplikáciu, zosnímajte jej obrazovku, uložte ju do grafického súboru a pošlite učiteľovi. Zistite, kde v MZ sa uložil tento grafický súbor?

Vysvetlíme si

Aplikáciu **otvoríme** tak, že klikneme na jej ikonu na domovskej obrazovke alebo na obrazovke aplikácií. Prípadne ju vieme otvoriť zo zoznamu naposledy spustených aplikácií stlačením tlačidlá Naposledy spustené aplikácie.

Aplikáciu **uzavrieme**, ak v zozname naposledy používaných aplikácií presunieme jej ikonu doľava alebo doprava. Ak chceme zavrieť všetky (na popredí) spustené aplikácie, vyberieme položku Zavrieť všetko. Obvykle aplikácie v OS Android nie je nutné zatvárať, pretože sa o to „inteligentným“ spôsobom postará samotný operačný systém.

Existuje niekoľko spôsobov ako urobiť **kópiu obrazovky** (angl. screenshot):

- súčasným stlačeným tlačidiel Stíšenie zvuku a Vypnutie,
- gestom posunutia dlane ponad obrazovku zľava doprava alebo sprava doľava.

Zosnímaný grafický súbor môžeme byť **uložený** v priečinku Galéria (Obrázky / Nedávne)

alebo v Moje súbory (Obrázky):

- Interné úložisko / DCIM / Screenshots
- Moje súbory / Fotografie / Screenshots

Po zosnímaní a uložení kópie obrazovky máme možnosť **zdieľať** s niekým grafický súbor prostredníctvom niektorej z aplikácií (napr. odoslaním SMS, cez e-mail, Viber, Skype, WhatApp, Facebook, Hangout, skopírovaním do Schránky, spojením sa s iným zariadením cez Bluetooth, uložením do cloudu OneDrive či Google Disc).

Úloha 5 (prehľad obľúbených a užitočných softvérových aplikácií na androidových MZ)

Urobte prehľad obľúbených a užitočných softvérových aplikácií, ktoré využívajú na androidových MZ spolužiaci z vašej triedy. Zistite a prediskutujte, ktoré typy aplikácií sú v spoločnom prehľade najviac zastúpené? Ktoré funkcionality MZ tieto aplikácie využívajú?

Poznámka

Na zozbieranie a prediskutovanie zoznamu obľúbených androidových aplikácií môžeme použiť otázku 4 z online pracovného zošita. Pre vyhodnotenie odpovedí odporúčame použiť online generátor mrakov slov (https://www.wordclouds.com/).

Alternatívou je nástenka aplikácie *Padlet* (https://padlet.com/lubomirsnajder/ml19124lcnke) s niekoľkými stĺpcami (Komunikácia, Multimédiá, Nástroje, Vzdelávanie, Hry, Iné), do ktorých by ste mali umiestňovať vaše obľúbené androidové aplikácie. Táto nástenka je verejne prístupná pre všetkých.

Úloha 6 (domáca úloha – zriadenie Google účtu, inštalácia aplikácie na snímanie QR kódov)

- Ak nemáte, zriaďte si Google účet, ktorý budete používať pri programovaní aplikácií pre MZ.
- Skontrolujte si, či máte na MZ inštalovanú aplikáciu na snímanie QR kódov. Ak nie, nainštalujte ju. (Poznámka: V najnovších verziách Androidu aplikácia Fotoaparát dokáže rozpoznať a zosnímať QR kódy.)

Úloha 7 (aký mám vzťah k programovaniu a predstavu o naprogramovaní vlastných aplikácií pre MZ)

Pomocou posledných troch otázok pracovného listu vyjadrite svoj vzťah k programovaniu a svoju predstavu a záujem o naprogramovanie vlastných aplikácií pre MZ.

- (7a) **Radi programujete?** Vyberte jednu z uvedených možností.
  áno – skôr áno – neviem povedať – skôr nie – nie
- (7b) Pokladáte za **užitočné** naučiť sa **programovať aplikácie pre MZ**?
  áno – skôr áno – neviem povedať – skôr nie – nie
- (7c) Uveďte, **aké aplikácie** by ste **chceli naprogramovať pre svoje MZ**:
  .....

Zamyslíme sa, čo sme sa naučili

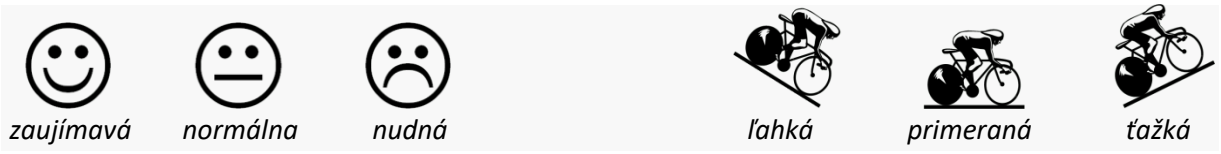
Sebahodnotiaca karta

Zapísaním symbolu ✓ na príslušné miesta tabuliek vyjadrite, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Vymenovať aspoň tri **základné parametre** androidového MZ.			
Vymenovať aspoň tri **vstavané senzory** androidového MZ a **vysvetliť ich využitie**.			
Vysvetliť význam a využitie aspoň piatich **softvérových aplikácií** androidového MZ.			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Zistiť parametre androidového MZ, včítane vstavaných senzorov.			
Inštalovať aplikáciu z Google Play na androidové MZ a **spustiť ju**.			
Otvoriť a **uzavrieť aplikáciu** v androidovom MZ.			
Zosnímať kópiu obrazovky MZ a **preniesť ju** na iné zariadenie.			

Aká bola pre vás táto hodina? Zaujímavá? Ľahká? Zafarbite/zakrúžkujte niektorú z uvedených možností:



1.2 Poďme vytvoriť prvú aplikáciu v MIT App Inventore 2

Kľúčové slová

Cloudové vývojové prostredie, režim návrhu prostredia aplikácie, režim programovania aplikácie, životný cyklus vývoja aplikácie), udalosťami riadené programovanie

Čo sa naučíme a čo si precvičíme

- Vysvetliť postup tvorby aplikácie pre MZ.
- Vytvoriť jednoduchú aplikáciu využívajúcu vybrané funkcionality MZ (dotykovú obrazovku a senzor zrýchlenia), skompilovať ju do inštalačného balíka a nainštalovať na MZ.
- Vysvetliť rozdiel medzi zdrojovým súborom aplikácie (AIA) a jej inštalačným balíkom (APK).
- Exportovať, resp. importovať zdrojové súbory aplikácie z resp. na cloud Ai2.

Akú zaujímavú aplikáciu môžeme vytvoriť? Ako budeme postupovať pri jej programovaní?

Úloha 1 (poďme krok za krokom naprogramovať aplikáciu pre MZ)

Vytvorte aplikáciu, ktorá bude na obrazovke vykresľovať kruhy na miestach nášho dotyku.

Vysvetlíme si

Pri tvorbe aplikácie pre MZ budeme postupovať nasledovne:

❶ Prihlásime sa na cloud Ai2 pomocou Google účtu (http://ai2.appinventor.mit.edu/).

❶ Začneme vytvárať nový projekt, ktorý pomenujeme **pmz_1_kreslicka**.

❷ Navrhneme rozhranie a správanie aplikácie.

❸ Vytvoríme rozhranie aplikácie v režime **Designer**.

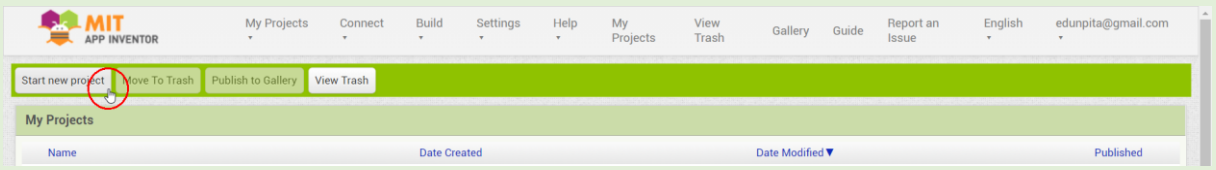
❹ Naprogramujeme správanie aplikácie v režime **Blocks**.

❺ Zostavíme inštalačný balík aplikácie (APK súbor).

❻ Stiahneme APK súbor na MZ, nainštalujeme a spustíme.

Vysvetlíme si

❶ Po prihlásení sa na cloud Ai2 začneme vytvárať nový projekt, ktorý pomenujeme názvom **pmz_1_kreslicka**.



Poznámka

Prihlasovanie sa do cloudu Ai2 na MIT je realizované cez Google účet. Pri prvom prihlásení sa na cloud Ai2 treba prečítať a akceptovať licenčnú zmluvu (https://appinventor.mit.edu/about/termsofservice). MIT nemá prístup ku nášmu Google účtu, ani k informáciám, ktoré sme poskytli Google. MIT má len našu e-mailovú adresu, prostredníctvom ktorej nás môže kontaktovať. Stránky MIT obsahujú aj dokumentáciu a vzdelávací obsah, ktorý sa poskytuje používateľom podľa licencie Creative Commons Attribution 4.0 International (CC BY 4.0). Cloud Ai2 obsahuje aj galériu na zdieľanie aplikácií, kde sú uvedené aj informácií o používateľských profiloch a komentáre k zdieľaným aplikáciám. Všetok obsah a projekty na tejto stránke sú licencované pod licenciou Creative Commons Attribution-ShareAlike 4.0 (CC SA), ktorá umožňuje komukoľvek prezerať, upraviť a redistribuovať tieto materiály.

Vysvetlíme si

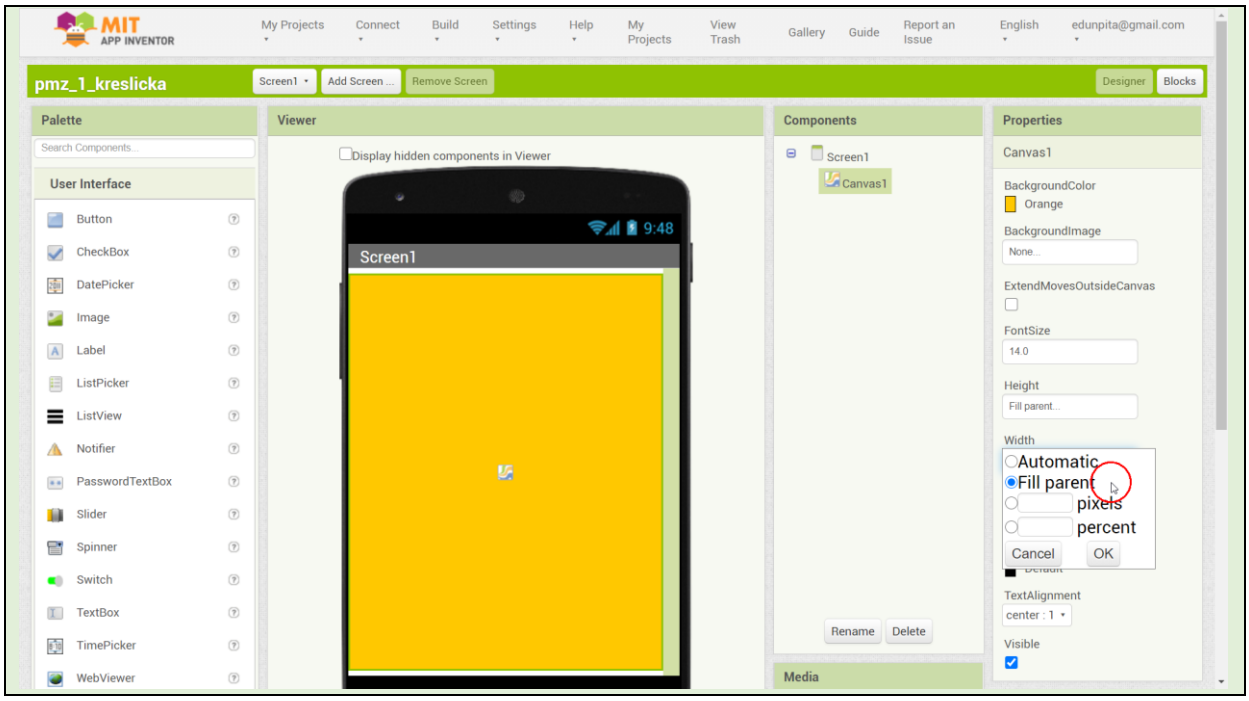
❷ POUŽÍVATEĽSKÉ ROZHRANIE aplikácie **pmz_1_kreslicka** je veľmi jednoduché – tvorí ho jeden viditeľný komponent `Canvas` (slov. *Plátno*), ktorý bude umiestnený na celej obrazovke MZ. Aplikácia v Ai2 je riadená udalosťami, ktoré vedia zachytiť jednotlivé komponenty a následne vykonať nejakú akciu.

SPRÁVANIE aplikácie je vhodné popísať do tabuľky ako trojicu Komponent – Udalosť – Akcia:

Komponent	**Udalosť**	**Akcia**
`Canvas` (Plátno)	`Touched` (Dotyk)	Na plátno sa vykreslí kruh so stredom v mieste dotyku a polomerom 10 bodov (`Canvas.DrawCircle`)

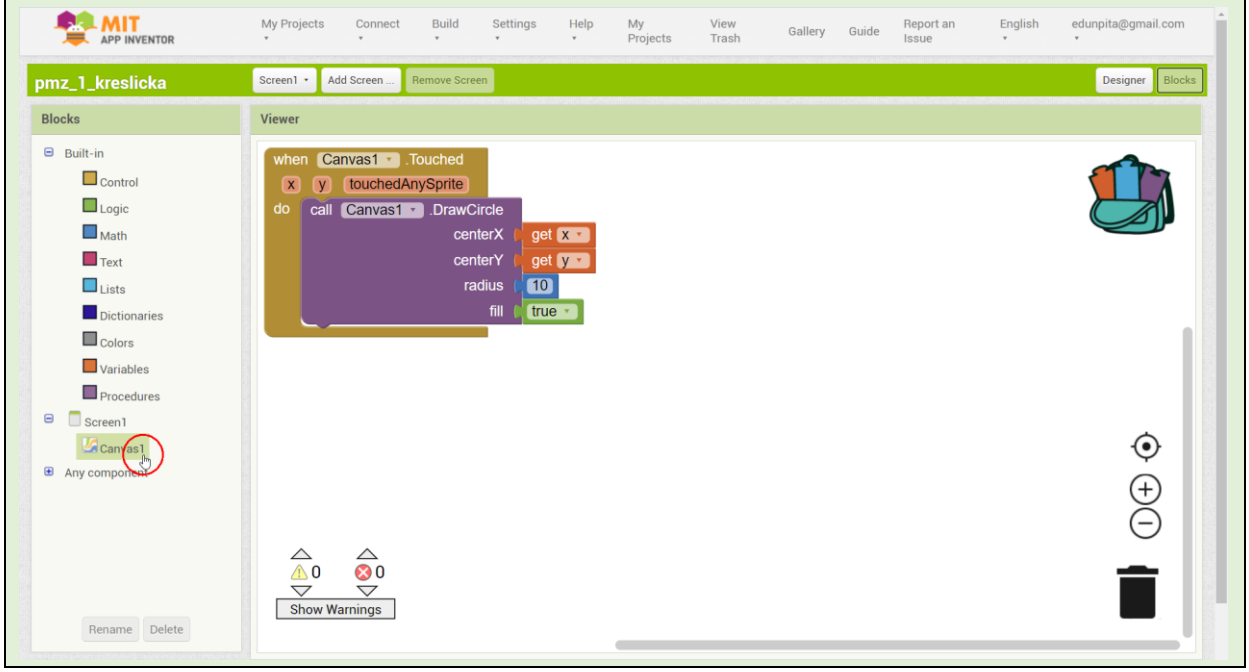
Vysvetlíme si

❸ V režime **Designer** vyberieme z časti **Palette** komponent `Canvas` a potiahneme ho do časti **Viewer**. V časti **Properties** nastavíme vlastnosť `Background` na hodnotu **Orange** a vlastnosti `Hight` a `Width` na hodnotu **Fill Parent**.



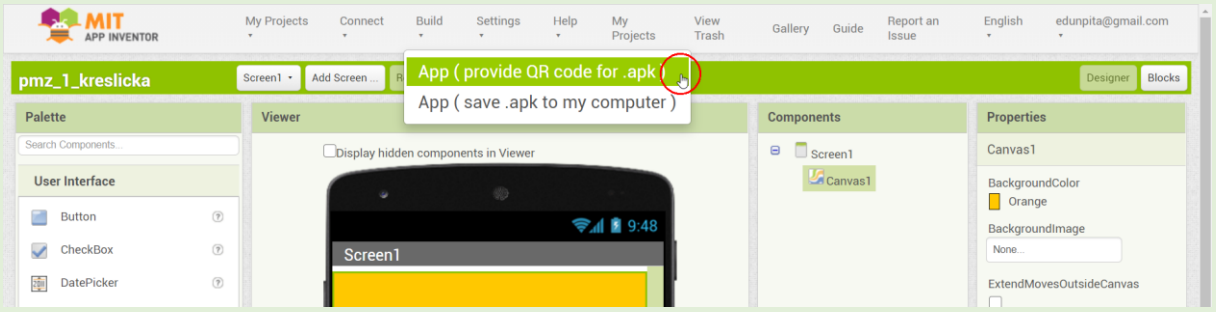
Vysvetlíme si

❹ V režime Blocks klikneme v časti **Blocks** na komponent `Canvas`. Z jeho ponuky vyberieme blok s udalosťou `Canvas.Touched` a potom blok s metódou `Canvas.DrawCircle`. Parametre `centerX` a `centerY` doplníme kliknutím na premenné `x` a `y` v bloku s udalosťou a ich ťahaním. Parameter `radius` doplníme číslom **10** zo skupiny **Math**.



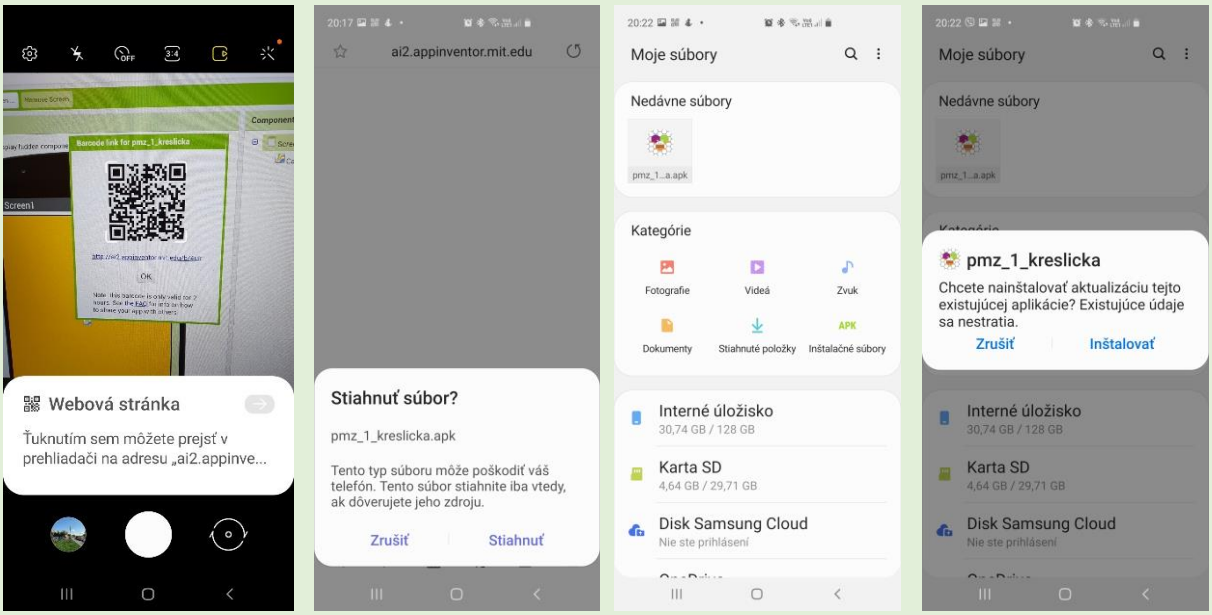
Vysvetlíme si

❺ Ak máme vytvorené používateľské rozhranie a aj programový kód aplikácie, môžeme vytvoriť inštalačný balík aplikácie – súbor s príponou `APK`. Výberom možnosti **Build / App (provide QR code for .apk)** hlavnej ponuky Ai2 sa o niekoľko sekúnd na cloude Ai2 zostaví `APK` súbor. Následne sa zobrazí okno s QR kódom, v ktorom je zakódovaná adresa `APK` súboru na cloude Ai2 (Pozor, táto adresa je platná len 2 hodiny, potom sa z cloudu Ai2 zmaže tento `APK` súbor).

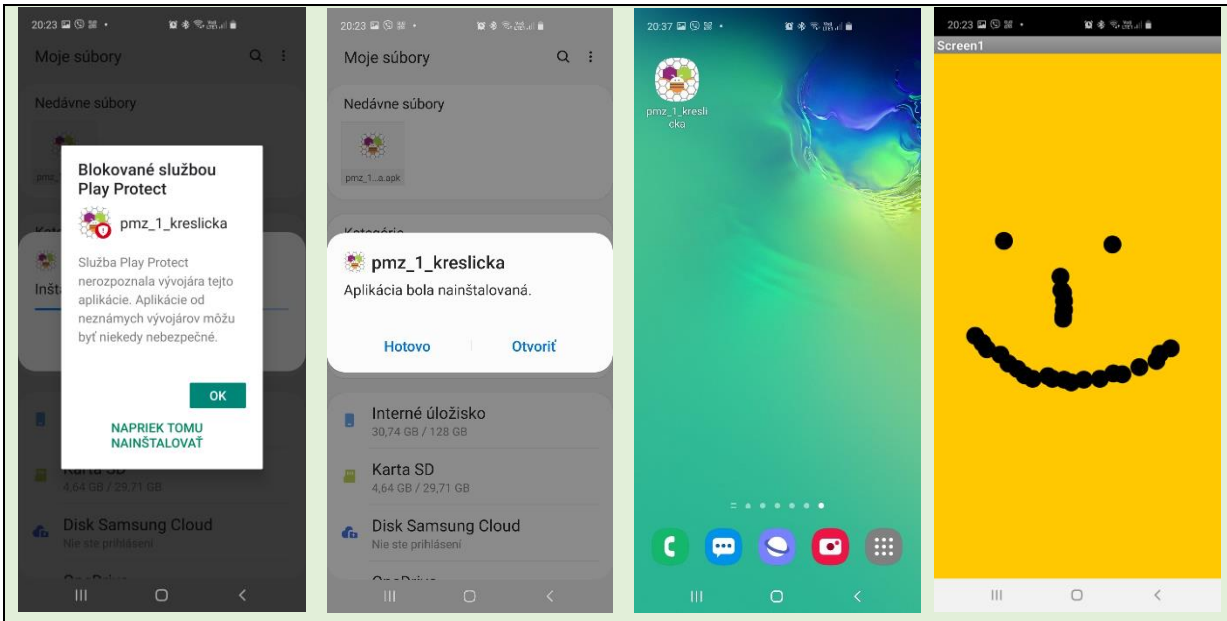


Vysvetlíme si

❻ Na prečítanie QR kódu a následné stiahnutie `APK` súboru do MZ použijeme aplikáciu na čítanie čiarových kódov, napr. *Barcode Scanner* (od ZXing Team). Na najnovších MZ netreba inštalovať ďalšie aplikácie, stačí spustiť fotoaparát, ktorý z obrázku prečíta QR kód.



Po stiahnutí aplikácie ju nainštalujeme na MZ. Pri inštalácii aplikácie je potrebné povoliť jej inštaláciu z neznámych zdrojov (na niektorých zariadeniach v ponuke **Nastavenia / Biometrické údaje a zabezpečenie / Inštalovať neznáme aplikácie** alebo na iných starších zariadeniach **Nastavenia / Systém / Zabezpečenie / Neznáme zdroje)**. Ak nás počas inštalácie služba *Play Protect* upozorní na nebezpečnosť aplikácii od neznámych vývojárov. V našom prípade vyberieme možnosť „Napriek tomu inštalovať“.



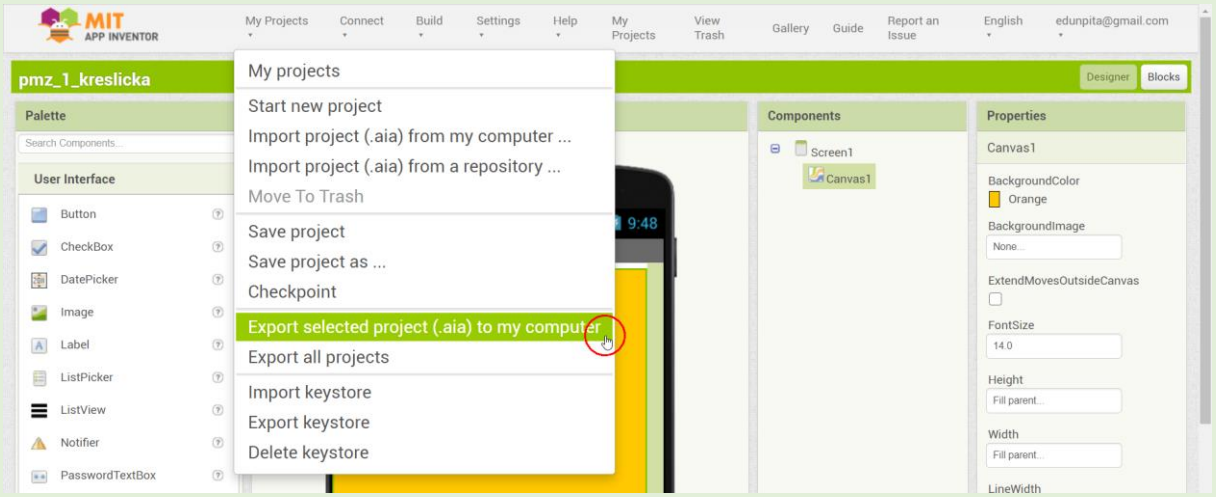
Ikonu aplikácie umiestnime na plochu, spustíme aplikáciu a nakreslíme milý obrázok, napr. usmievačika.

Ak ste to všetko dokázali, môžete spolužiakom a učiteľovi ukázať nakreslený obrázok vo vlastnej aplikácii pre MZ. Rovnako sa môžete s týmto svojím úspechom podeliť aj s nami, keď nám pošlete svoj obrázok na e-mailovú adresu edunpita@gmail.com.

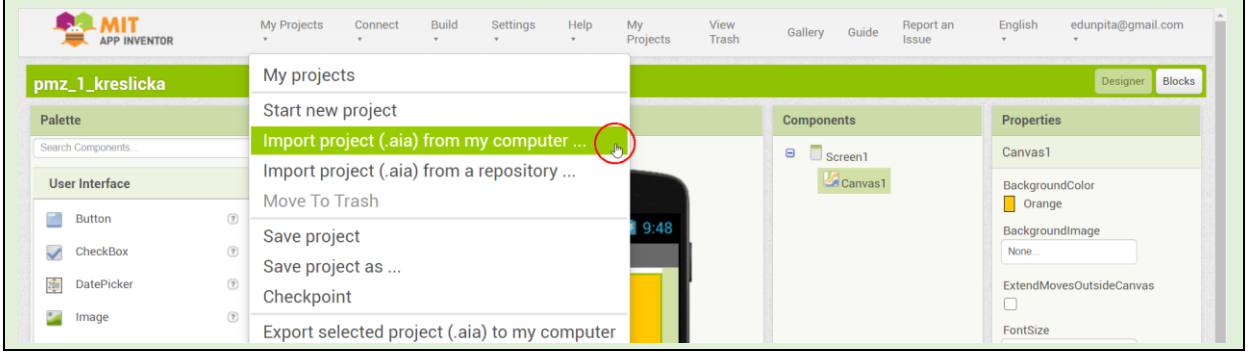
Ako pošleme učiteľovi svoju aplikáciu a ako nahráme do cloudu Ai2 učiteľovu aplikáciu

Vysvetlíme si

Zdrojový kód aplikácie, ktorú programujeme v cloude Ai2, môžeme uložiť na disk ako súbor s príponou `AIA` pomocou ponuky **Projects / Export selected projects (.aia) to my computer**. Tento súbor môžeme poslať učiteľovi či kamarátom, ktorí si ho môžu importovať a pozrieť v svojom účte na cloude Ai2.



Podobne, ak nám učiteľ či kamaráti pošlú zdrojový kód ich aplikácie (`AIA` súbor), môžeme tento súbor importovať do nášho účtu na cloude Ai2 pomocou ponuky **Projects / Import project (.aia) from my computer**. Takto sa môžeme od učiteľa či kamarátov naučiť ako programovať v Ai2. Pozor, rozlišujte **zdrojový súbor** *aplikácie* (má príponu `AIA`) a **inštalačný balík aplikácie** (má príponu `APK`).



Ako vylepšiť či rozšíriť našu aplikáciu?

Vo vytvorenej aplikácii dokážeme kresliť obrázky len bodkovaním. Ak by sme chceli pomocou nej nakresliť, napr. domček jedným ťahom, tak by sme mali do nej doplniť možnosť kreslenia ťahaním čiar. Z praktického hľadiska je dôležité doplniť do aplikácie aj zmazanie plátna, vyvolané napr. zatrasením MZ.

Úloha 2 (rozšírme kresliacu aplikáciu)

Rozšírte aplikáciu o možnosť kreslenia ťahaním čiar a zmazania plátna zatrasením MZ.

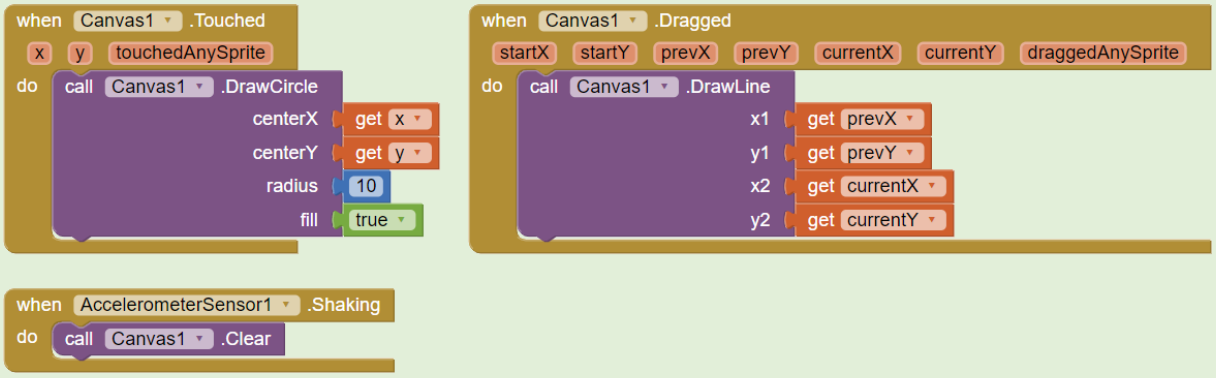
Vysvetlíme si

POUŽÍVATEĽSKÉ ROZHRANIE rozšírenej aplikácie doplníme o neviditeľný komponent `AccelerometerSensor` (slov. *Senzor zrýchlenia*), pomocou ktorého budeme zaznamenávať zatrasenie MZ.

SPRÁVANIE aplikácie rozšírime o ďalšie dve udalosti `Canvas.Dragged` (slov. *Ťahanie na plátne*) a `AccelerometerSensor.Shaking` (slov. *Zatrasenie*), na základe ktorých sa vykonajú akcie `Canvas.DrawLine`) (slov. *Vykreslenie úsečky*) a `Canvas.Clear` (slov. *Zmazanie obrazovky*).

Komponent	Udalosť	Akcia
`Canvas` (Plátno)	`Touched` (Dotyk)	Na plátno sa vykreslí kruh so stredom v mieste dotyku a polomerom 10 bodov (`Canvas.DrawLine`).
`Canvas` (Plátno)	`Dragged` (Ťahanie)	Na plátno sa vykreslí úsečk od predchádzajúcej do aktuálnej pozície (`Canvas.DrawLine`).
`AccelerometerSensor` (Senzor zrýchlenia)	`Shaking` (Zatrasenie)	Zmaže sa obsah plátna (`Canvas.Clear`)

Teraz si môžete overiť výsledný programový kód rozšírenej kresliacej aplikácie:



Zamyslíme sa, čo sme sa naučili

Sebahodnotiaca karta

Zapísaním symbolu ✓ na príslušné miesta tabuliek vyjadrite, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	rozumiem	čiastočne rozumiem	vôbec nerozumiem
Ai2 je cloudové programovacie prostredie na tvorbu aplikácií pre MZ.			
Pomocou **Google účtu** sa prihlasujeme na **cloud Ai2**, kde sú uložené naše aplikácie.			
Pri vytváraní aplikácie navrhujeme **používateľské rozhranie** v režime **Designer** a **správanie aplikácie** v režime **Blocks**.			
Používateľské rozhranie tvoria **Components** (komponenty) s nastavenými **Properties** (vlastnosťami).			
Komponenty môžu byť **Visible** (viditeľné) a **Non-visible** (neviditeľné).			
Príkladom **viditeľného** komponentu je `Canvas` (Plátno).			
Príkladom **neviditeľného** komponentu je `AccelerometerSensor` (Senzor zrýchlenia).			
Správanie aplikácie zabezpečíme pomocou **akcií**, ktoré sú odpoveďami na **Events** (udalosti) zachytené **komponentmi**.			
Zdrojový súbor v Ai2 má príponu **AIA** a inštalačný balík príponu **APK**.			
Príkladmi **udalosti** sú, napr. `Canvas.Touched` (Dotyk na Plátno), `Canvas.Dragged` (Ťahanie po Plátne), `AccelerometerSensor.Shaking` (Zatrasenie zariadením).			
Príkladmi **metód** sú, napr. `Canvas.DrawCircle` (Vykreslenie kruhu na Plátno), `Canvas.DrawLine` (Vykreslenie úsečky na Plátno), `Canvas.Clear` (Zmazanie obsahu Plátna).			
*Udalosť `Dragged` (Ťahanie) využijeme na kreslenie čiar pomocou **metódy** `DrawLine`**.**			
*Pri kreslení čiar ťahaním nastavíme v **metóde** `DrawLine` prvý bod úsečky na hodnotu predchádzajúceho miesta ťahania [`prevX`**,** `prevY`] a druhý bod úsečky na hodnotu aktuálneho miesta ťahania [`currentX`**,** `currentY`].			

Z uvedeného učiva viem vykonať nasledovné činnosti:	samostatne	s malou pomocou	s veľkou pomocou
Prihlásiť sa na cloud Ai2 a **začať vytvárať** nový projekt.			
Exportovať/importovať zdrojové kódy aplikácii (**AIA súbory**) na/z disku počítača.			
Vytvárať používateľské rozhranie aplikácie v režime Designer a **programový kód správania sa** aplikácie v režime Blocks.			
Zostaviť súbor s príponou APK na cloude Ai2 a **stiahnuť ho** do MZ.			
Povoliť inštaláciu aplikácie z neznámych zdrojov, resp. non-market applications.			
Inštalovať aplikáciu na MZ a **spustiť ju**.			

Aká bola pre vás táto hodina? Zaujímavá? Ľahká? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavá


normálna


nudná


ľahká


primeraná


ťažká

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2 Tvorba malých aplikácií

Kľúčové slová

Softvérová aplikácia, životný cyklus vývoja aplikácie, udalosťami riadené programovanie, údajové typy, riadiace konštrukcie, senzory, komponenty, udalosti, metódy, vlastnosti

Čo sa naučíme a čo si precvičíme

- Analyzovať funkcionality softvérovej aplikácie:
  - na základe prečítania jej programového kódu,
  - na základe jej spustenia.
- Modifikovať, resp. rozšíriť softvérovú aplikáciu o nové funkcionality.
- Prehĺbiť teoretické poznatky o životnom cykle vývoja softvérovej aplikácie pre MZ a prakticky precvičiť vývoj udalosťami riadenej aplikácie .
- Precvičiť základné údajové typy a riadiace konštrukcie prostredia Ai2.
- Rozvíjať tvorivosť pri rozširovaní funkcionality aplikácie.
- Vysvetliť význam využitia jednotlivých malých aplikácií v každodennom živote pre určité cieľové skupiny používateľov.

Zoznam malých aplikácií

Zbierka 12 malých aplikácií pokrýva vybrané funkcionality Ai2. Jednotlivé malé aplikácie sú rozpracované v rôznych verziách náročnosti, resp. s rôznym rozsahom rozšírenia (v zátvorkách sú tučným písmom zvýraznené použité komponenty Ai2):

2.1 **Kresliaci editor**
(**Canvas**.TouchDown, TouchUp; Screen.Title, AppName, Icon, BackgroundImage, BackgroundColor)

2.2 **Hra Postreh**
(**Clock**.Timer, TimerInterval, TimerEnabled; **Ball**.TouchDown, MoveTo, Enabled; **Button**.Click; **Label**.text; **HorizontalArrangement**; **Sound**.Play; **Screen**.Initialize; **Canvas**.Width, Height; **globálna premenná** – initialize global, set, get; close application; random integer from X to Y)

2.3 **Hra Guľka**
(**OrientationSensor**.Enabled; **Ball**.X, Y, PaintColor, Visible, CollidedWith; **Clock**.Now, Duration; **TinyDB**.GetValue, StoreValue; IF)

2.4 **Kalkulačka**
(**TextBox**.Text; **Slider**.PositionChanged; **Notifier**.ShowAlert, ShowTextDialog, ShowMessageDialog, AfterTextInput; **lokálna premenná**, **vlastná funkcia s parametrom**; IF-THEN-ELSE; join)

2.5 **Zbierka vtipov**
(**Screen.**OtherScreenClosed; open another screen, close screen, close screen with value; **Canvas.**Flung)

2.6 **Čítačka QR kódu**
(**BarcodeScanner.**DoScan, AfterScan, UseExternalScanner; **TextToSpeech.**Speak, Country, Language, SpeechRate, Pitch; **Spinner.**AfterSelecting; **SpeechRecognizer.**GetText, AfterGettingText)

2.7 **Asistent pri cvičení**
(**ProximitySensor.**ProximityChanged, Enabled; **Sound.**Vibrate; **CheckBox.**Checked; **Pedometer.**Start, Resume, Reset, Pause, Save, Stop, WalkStep, WalkSteps, Distance, ElapsedTime, StrideLength, StopDetectionTimeout, **AccelerometerSensor.**Enabled; **TableArrangement**)

2.8 **Generátor náhodných viet**
(make a list, select list item, length of list, remove list item, insert list item, create empty list, add items to list; **ListView.**Elements; FOR EACH NUMBER; FOR EACH ITEM)

2.9 **Zobrazovač aktuálnej polohy**
(**LocationSensor.**LocationChanged, latitude, longitude, altitude, speed, StatusChanged, status, LatitudeFromAddress, LongitudeFromAddress, ProviderName, HasAccuracy, Accuracy, CurrentAddress, TimeInterval, DistanceInterval, Enabled; **Map**)

2.10 **Asistent aktuálnej polohy**
(**ActivityStarter.**Action, DataUri, Extras, StartActvity; **ListPicker.**Elements, Selection, AfterPicking, BeforePicking, **TinyDB.**GetTags, ClearAll; **HorizontalArrangement.**Visible; pick a random item)

2.11 **Hlasovanie na internete**
(**FirebaseDB.**StoreValue, GetValue, GotValue, DataChanged, tag, value, FirebaseURL; index in list; max)

2.12 **Komunikačný asistent**
(**Texting.**MessageReceived, number, messageText, SendMessage, PhoneNumber, Message, ReceivingEnabled; **PhoneCall.**MakePhoneCall, PhoneNumber)

Pre úplnosť a orientáciu čitateľa uvádzame aj prehľad ďalších komponentov Ai2, ktoré nie sú pokryté uvedenými malými aplikáciami:

Skupina	**Komponenty**
User Interface	`DataPicker`, `PasswordTextBox`, `TimePicker`, `Switch`, `WebViewer`
Layout	`HorizontalScrollArrangement`, `VerticalScrollArrangement`
Media	`Camcorder`, `ImagePicker`, `Player`, `SoundRecorder`, `VideoPlayer`, `Yandex Translate`
Drawing and Animation	`ImageSprite`
Maps	`Circle`, `FeatureCollection`, `LineString`, `Marker`, `Navigation`, `Polygon`, `Rectangle`
Sensors	`Barometer`, `GyroscopeSensor`, `Hygrometer`, `LightSensor`, `NearField`, `Thermometer`
Social	`ContactPicker`, `EmailPicker`, `PhoneNumberPicker`, `Sharing`, `Twitter`
Storage	`CloudDB`, `File`, `TinyWebDB`
Connectivity	`BluetoothClient`, `BluetoothServer`, `Serial`, `Web`
LEGO® MINDSTORMS®	`NxtDirectCommands`, `NxtColorSensor`, `NxtLightSensor`, `NxtSoundSensor`, `NxtTouchSensor`, `NxtUltrasonicSensor`, `NxtDrive` `Ev3Motors`, `Ev3ColorSensor`, `Ev3GyroSensor`, `Ev3TouchSensor`, `Ev3UltrasonicSensor`, `Ev3Sound`, `Ev3UI`, `Ev3Commands`
Experimental	-

Niektoré z týchto komponentov sú vysvetlené a použité pri projektoch v kapitolách 3 až 6.

Pokrytie vybraných komponentov Ai2 problémami v malých aplikáciách

		Použ. rozhranie							Mmédiá			Senzory								Komunik.			Pamäť		Jazykové koncepty						
názov problému	**názov súboru s kódom**	Screen, Canvas, Ball, Image	MultipleScreens	Button, TextBox, Label, CheckBox	H/V/T Arrangements	ListView, ListPicker	Slider, Spinner	Notifier	Sound	TextToSpeech	SpeechRecognizer	Clock	AccelerometerSensor	LocationSensor	Map	OrientationSensor	BarcodeScanner	ProximitySensor	Pedometer	ActivityStarter	Texting	PhoneCall	TinyDB	FireBase	Cyklus	Vetvenie, podmienky	matematické operácie a funkcie	Globálne premenné (čísla, farby)	Zoznamy	Procedúry, funkcie	lokálne premenné	
Kresliaci editor (1.1, 1.2)	pmz_2_1_platno	■											■																			2
Kresliaci editor (1.3)	pmz_2_1_platno_R	■											■																			2
Hra Postreh (2.1)	pmz_2_2_hra_postreh1	■							■			■															■					4
Hra Postreh (2.2)	pmz_2_2_hra_postreh2	■		■	■				■			■															■	■				7
Hra Postreh (2.3)	pmz_2_2_hra_postreh2_R	■		■	■				■			■															■	■				7
Hra Guľka (3.1)	pmz_2_3_hra_gulka1	■							■							■											■					4
Hra Guľka (3.2)	pmz_2_3_hra_gulka1_R	■														■											■					3
Stopky (3.3)	pmz_2_3_stopky			■								■															■	■				4
Počítadlo (3.4)	pmz_2_3_pocitadlo	■		■																			■				■	■				5
Hra Guľka (3.5)	pmz_2_3_hra_gulka2_R	■		■								■				■							■			■	■	■		■		9
Kalkulačka BMI (4.1)	pmz_2_4_kalkulacka1a			■	■																					■	■				■	5
Kalkulačka BMI (4.2)	pmz_2_4_kalkulacka1b			■	■																					■	■			■	■	6
Kalkulačka BMI (4.3)	pmz_2_4_kalkulacka2_R			■	■		■																			■	■			■	■	7
Kalkulačka BMI (4.4)	pmz_2_4_kalkulacka3			■				■																		■	■	■		■		6
Zbierka vtipov (5.1)	pmz_2_5_vtipy1	■	■	■	■																					■						5
Zbierka vtipov (5.2)	pmz_2_5_vtipy1_R	■	■	■	■																					■						5
Zbierka vtipov (5.3)	pmz_2_5_vtipy2	■	■	■	■																					■						5
Čítačka QR kódu (6.1)	pmz_2_6_QR_kod1			■						■							■															3
Čítačka QR kódu (6.2)	pmz_2_6_QR_kod2			■			■			■	■						■															5
Čítačka QR kódu (6.3)	pmz_2_6_QR_kod2_R			■	■		■			■	■						■															6
Asistent cvikov (7.1)	pmz_2_7_cviky1	■							■				■					■														4
Asistent cvikov (7.2)	pmz_2_7_cviky1_R	■		■	■				■				■					■								■	■	■				9
Asistent cvikov (7.3)	pmz_2_7_krokomer1			■	■				■										■								■					5
Asistent cvikov (7.4)	pmz_2_7_krokomer2	■		■	■														■							■	■			■		7
Generátor viet (8.1)	pmz_2_8_vety			■						■																	■	■	■			5
Generátor viet (8.2)	pmz_2_8_vety_R			■						■																	■	■	■		■	6
Spracovanie zoznamov (8.3)	pmz_2_8_zoznam_cisel			■	■	■																			■	■	■	■	■	■	■	10
Spracovanie zoznamov (8.4)	pmz_2_8_zoznam_cisel_R			■	■	■																			■	■	■	■	■	■	■	10
Zobrazovač GPS polohy (9.1)	pmz_2_9_gps			■	■									■	■																	4
Zobrazovač GPS polohy (9.2)	pmz_2_9_gps_R			■	■		■							■	■											■						6
Asistent GPS polohy (10.1)	pmz_2_10_astarter_mapy	■		■	■	■								■						■								■	■			8
Asistent GPS polohy (10.2)	pmz_2_10_astarter_mapy_R	■		■	■	■		■						■						■			■		■	■	■	■	■	■		14
Spúšťač externých apiek (10.3)	pmz_2_10_astarter_rozne			■	■															■								■	■			5
Webový kliker (11.1)	pmz_2_11_kliker	■		■																				■			■					4
Webové hlasovanie (11.2)	pmz_2_11_hlasovanie_R	■		■	■																			■		■	■		■			7
Hlasná čítačka SMS (12.1)	pmz_2_12_sms_nahlas			■						■											■											3
Asistent SMS (12.2)	pmz_2_12_sms_telefon_R	■		■	■					■	■		■					■			■	■				■						10
		20	3	31	21	4	4	2	7	7	3	5	5	4	2	3	3	3	2	3	2	1	3	2	3	16	22	14	8	8	6	

2.1 Kresliaci editor

Prvky nového učiva

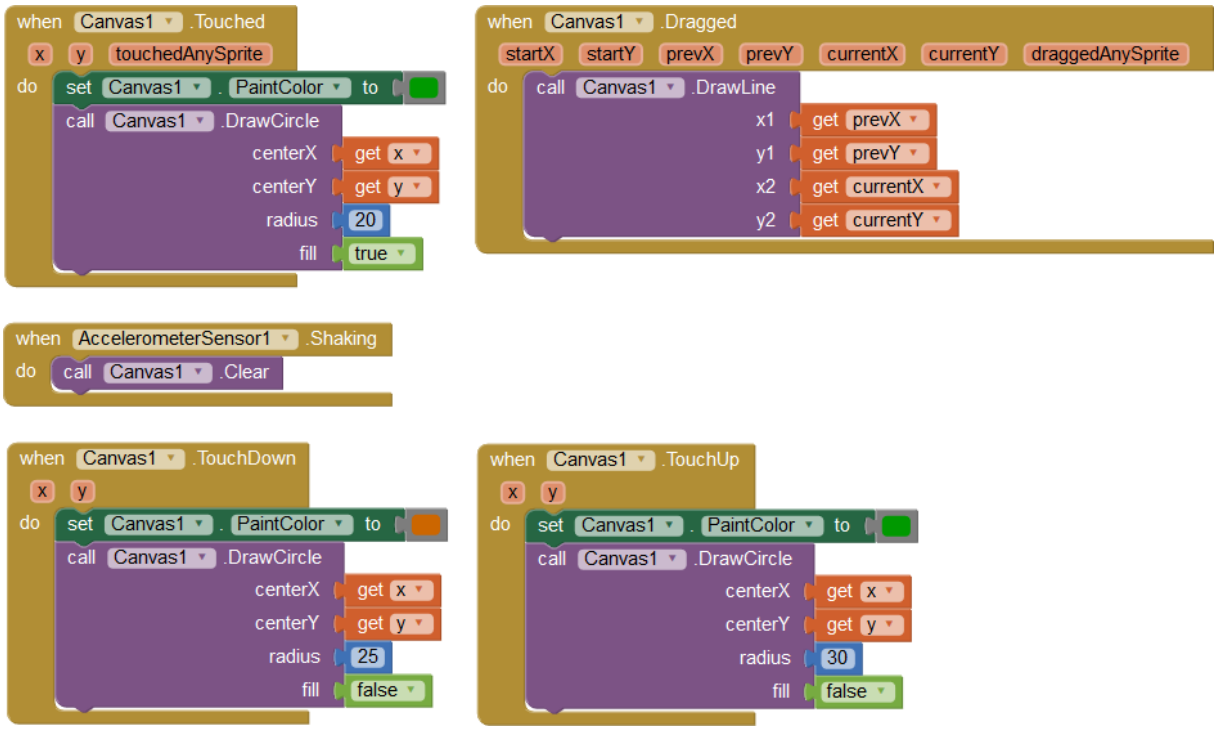
Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`Canvas`	`TouchDown` `TouchUp`		`PaintColor`

`Screen`			`Title`
			`AppName`
			`Icon`
			`BackgroundImage`
			`BackgroundColor`

Jazykové konštrukcie		**Iné prvky jazyka**	

Úloha 1

V aplikácii **pmz_1_kreslicka** z prvej kapitoly sme doplnili spracovanie ďalších dvoch udalostí `Canvas.TouchDown` a `Canvas.TouchUp`. V dvojiciach si preštudujte uvedený zdrojový kód a bez spustenia aplikácie prediskutujte jej nové funkcionality.



Úloha 2

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_1_platno.aia**. Zostavte z neho inštalačný balík a nainštalujte ho na MZ. Po jeho spustení preskúmajte správanie sa aplikácie pri spracovaní nových udalostí `Canvas.TouchDown` a `Canvas.TouchUp` a svoje zistenia zapíšte do posledných dvoch riadkov tabuľky.

Komponent	Udalosť	Akcia
`Canvas` (Plátno)	`Touched` (Dotyk)	Na plátno sa vykreslí kruh so stredom v mieste dotyku a polomerom 10
`Canvas` (Plátno)	`Dragged` (Ťahanie)	Na plátno sa vykreslí úsečka od predchádzajúcej do aktuálnej pozície
`AccelerometerSensor` (Senzor zrýchlenia)	`Shaking` (Zatrasenie)	Zmaže sa obsah plátna
`Canvas` (Plátno)	`TouchDown` ( )	
`Canvas` (Plátno)	`TouchUp` ( )	

Úloha 3

V aplikácii **pmz_2_1_platno.aia** urobte zmeny uvedené v prvom stĺpci tabuľky a do druhého stĺpca na základe vlastného experimentovania uveďte aký efekt spôsobili tieto zmeny. Pri tvorbe ikon odporúčame použiť rozmer 48×48 bodov a formát PNG (transparentný). Upravený kód uložte do súboru **pmz_2_1_platno_R.aia**.

Zmena	Efekt
V komponente `Screen` nastaviť vlastnosť: a. `Title` na hodnotu **Kreslička2** b. `AppName` na hodnotu **Kreslička2** c. `Icon` na hodnotu **farbicka_ikona.png** d. `BackgroundImage` na hodnotu **mriezka_16x9.png** e. `BackgroundColor` na hodnotu **yellow**	a. b. c. d. e.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.1 Kresliaci editor

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	rozumiem	čiastočne rozumiem	vôbec nerozumiem
Udalosť `Canvas.TouchDown` sa vyvolá po priložení prstu (pera) na plátno			
Udalosť `Canvas.TouchUp` sa vyvolá po zdvihnutí prstu (pera) z plátna			
Vlastnosť `Canvas.PaintColor` zodpovedá farbe, ktorou sa bude kresliť na plátno			
Vlastnosť `Screen.Title` zodpovedá textu v titulnom páse aplikácie			
Vlastnosť `Screen.AppName` zodpovedá názvu aplikácie po jej nainštalovaní na MZ			
Vlastnosť `Screen.Icon` zodpovedá obrázku ikony, ktorá bude reprezentovať aplikáciu nainštalovanú na MZ			
Vlastnosť `Screen.BackgroundImage` zodpovedá obrázku pozadia obrazovky			
Vlastnosť `Screen.BackgroundColor` zodpovedá farbe pozadia obrazovky			

Z uvedeného učiva viem vykonať nasledovné činnosti:	samostatne	s malou pomocou	s veľkou pomocou
Nastaviť vlastnosť `Canvas.PaintColor` na danú farbu			
Nastaviť vlastnosť `Screen.Title` na daný text			
Nastaviť vlastnosť `Screen.AppName` na daný názov aplikácie			
Nastaviť vlastnosť `Screen.Icon` na daný obrázok ikony aplikácie			
Nastaviť vlastnosť `Screen.BackgroundImage` na daný obrázok pozadia obrazovky			
Nastaviť vlastnosť `Screen.BackgroundColor` na danú farbu pozadia obrazovky			
Vytvoriť kresliacu aplikáciu využívajúcu udalosti komponentu `Canvas` (`TouchUp`, `TouchDown`)			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.2 Hra Postreh

Prvky nového učiva

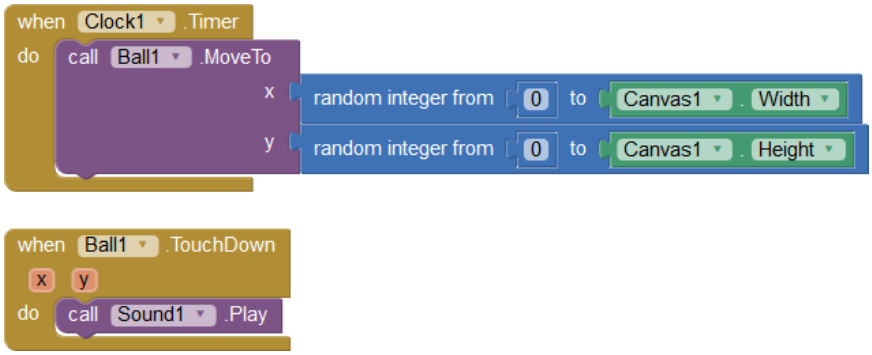
Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`Clock`	`Timer`		`TimerInterval` `TimerEnabled`
`Ball`	`TouchDown`	`MoveTo`	`Enabled`
`Button`	`Click`		
`Label`			`Text`
`HorizontalArrangement`			
`Sound`		`Play`	
`Screen`	`Initialize`		
`Canvas`			`Width` `Height`

Jazykové konštrukcie	Iné prvky jazyka
globálna premenná (`initialize global`, `get`, `set`)	príkaz `close application` funkcia `random integer from X to Y`

Úloha 1

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_2_hra_postreh1.aia**. Po jeho nainštalovaní a spustení na MZ preskúmajte správanie sa tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuliek.

(Poznámka: Pri skúmaní správania aplikácie odporúčame použiť referenčné materiály uvedené na stránkach: http://ai2.appinventor.mit.edu/reference/components/ a https://developer.android.com/guide/topics/media/media-formats)



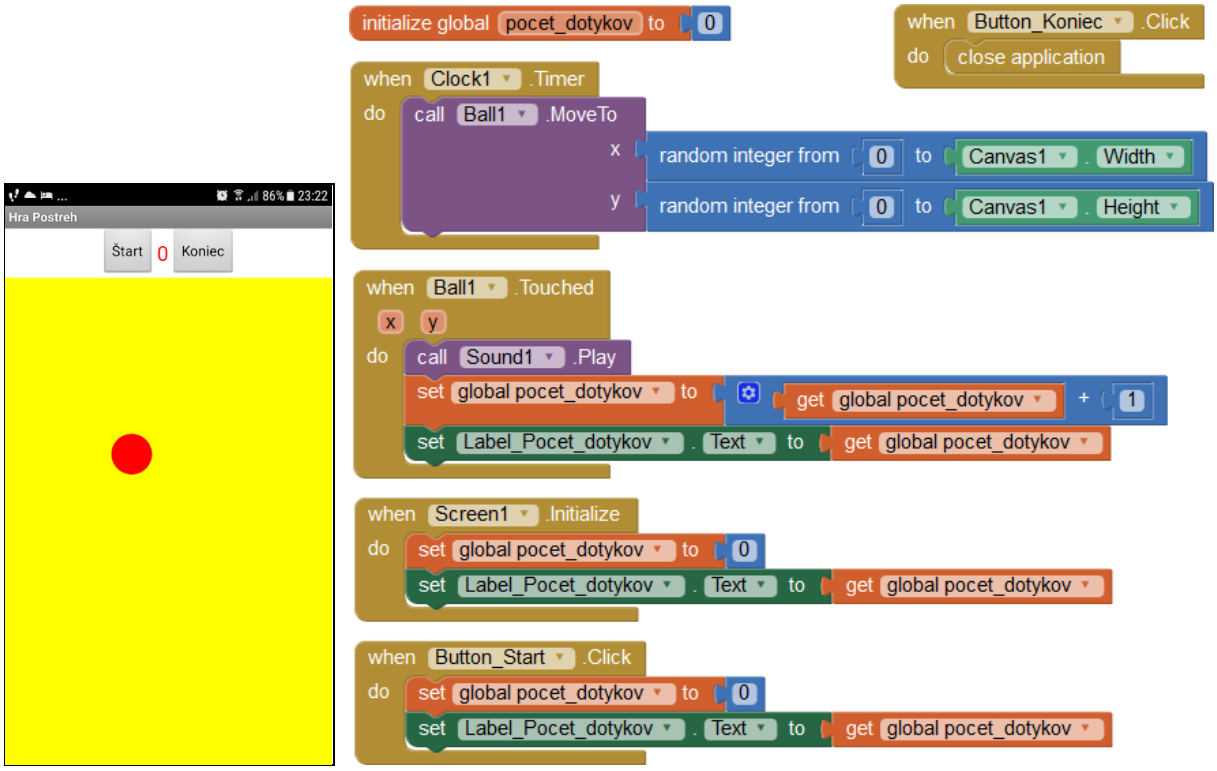
Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `Ball`? b. Ktoré vlastnosti má komponent `Ball`? c. Do ktorého komponentu môžeme vložiť komponent `Ball`? d. Na čo sa dá využiť komponent `Ball`? e. Ktoré zvukové formáty vie prehrávať komponent `Sound`? f. Čo predstavujú vlastnosti `Canvas.Width` a `Canvas.Height`?	a. b. c. d. e. f.

g. Čo vracia funkcia `random integer from X to Y`? h. Ako sa zmení správanie aplikácie (udalosti `Clock.Timer`) ak zmeníme v komponente `Clock` vlastnosti `TimerInterval` a `TimerEnabled`?	g. h.

Komponent	Udalosť	Akcia
`Ball` ( )	`TouchDown` ( )	`Sound.Play`
`Clock` ( )	`Timer` ( )	`Ball.MoveTo`

Úloha 2

Preskúmajte používateľské rozhranie a správanie aplikácie so zdrojovým kódom uloženým v súbore **pmz_2_2_hra_postreh2.aia**. Svoje zistenia zapíšte do voľných políčok tabuliek.



Otázka	Odpoveď
a. Ktoré vizuálne a nevizuálne komponenty tvoria **používateľské rozhranie** aplikácie? b. Do ktorého komponentu je vložený komponent `Ball` (lopta)? c. Na čo slúži komponent `Button` (tlačidlo)? d. Na čo slúži komponent `Label` (popisok)? e. Na čo slúži komponent `HorizontalArrangement`? f. V ktorej situácii sa spracuje udalosť `Screen.Initialize`?	a. b. c. d. e. f.

g. Čo urobí príkaz `close application` v rámci udalosti `Button_Koniec.Click`? h. V zdrojovom kóde aplikácie vyznačte časti, v ktorých sa pracuje s globálnou premennou **pocet_dotykov**. i. Čo sa stane s hodnotou premennej **pocet_dotykov**, ak sa viackrát rýchlo dotkneme lopty na danom mieste? j. Ktoré multimediálne súbory sú použité v aplikácii a akým spôsobom?	g. h. i. j.

Komponent	Udalosť	Akcia
`Ball`	`TouchDown`	
`Clock`	`Timer`	
`Button_Koniec`	`Click`	
`Button_Štart`	`Click`	
`Screen`	`Initialize`	

Úloha 3

Vylepšite aplikáciu **pmz_2_2_hra_postreh2.aia** zmenou nastavenia vlastnosti `Ball.Enabled` a `Clock.TimerEnabled` na hodnoty **true** a **false**. Výsledný kód uložte do súboru **pmz_2_2_hra_postreh2_R.aia**.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.2 Hra Postreh

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	rozumiem	čiastočne rozumiem	vôbec nerozumiem
Komponent `Clock` je nevizuálnym komponentom súvisiaci s meraním a vypisovaním času, dátumu a s časovým intervalmi			
Udalosť `Clock.Timer` sa opakovane vyvoláva po uplynutí časového intervalu zadaného vo vlastnosti `Clock.TimerInterval`			
Udalosť `Clock.Timer` sa nevyvolá, ak je vypnutá vlastnosť `Clock.TimerEnabled` (t. j. je nastavená na hodnotu **false**)			
Komponent `Ball` je vizuálnym komponentom často využívaným na animácie, ktorý je umiestnený na komponente `Canvas`			
Udalosť `Ball.TouchDown` sa vyvolá po dotyku prstu (pera) na komponent `Ball`			
Komponent `Ball` je neaktívnym, ak je vypnutá jeho vlastnosť `Ball.Enabled` (t. j. je nastavená na hodnotu **false**)			
Komponent `Button` je vizuálnym komponentom využívaným hlavne na spúšťanie rôznych aktivít pomocou udalosti `Button.Click`			
Komponent `Label` je vizuálnym komponentom využívaným na výpis hodnôt vlastností komponentov, či premenných. Tieto hodnoty sú uložené vo vlastnosti `Label.Text`			
Udalosť `Screen.Initialize` sa vyvolá hneď po otvorení aplikácie			
Komponent `HorizontalArrangement` sa využíva ako kontajner na vodorovné umiestnenie rôznych komponentov vedľa seba			
Príkaz `close application` ukonči beh aplikácie			
Komponent `Sound` je nevizuálnym komponentom, ktorý sa používa na prehrávanie krátkych zvukov uložených v rôznych zvukových formátoch, napr. **vzorky** (flac , wav, ogg, mp3,			

3gp, mp4, aac, mkv), **skladby** (mid, xmf, mxmf), **zvonenia** (rtttl, rtx, ota, imy). Na prehrávanie zvuku sa používa metóda `Sound.Play`			
Vlastnosti `Canvas.Width` a `Canvas.Height` predstavujú šírku a výšku komponentu `Canvas`			
Funkcia `random integer from X to Y` vráti náhodné celé číslo z intervalu <X,Y>			
Pri výpočtoch môžeme využívať **premenné**. Hodnota **globálnej premennej** sa **inicializuje** pomocou špeciálneho inicializačného bloku			
Priradenie hodnoty výrazu **do premennej** sa realizuje pomocou `set` bloku			
Hodnota premennej vo výraze je reprezentovaná `get` blokom			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť používateľské rozhranie aplikácie, ktoré obsahuje **plátno** s **loptou** a tiež **vodorovne** vedľa seba zarovnané **tlačidlá** a **popisok**			
Pre komponent `Sound` nastaviť **zvukový súbor** a použiť metódu `Sound.Play`			
Použiť udalosť `Clock.Timer` na spúšťanie určitých aktivít v pravidelných intervaloch			
Použiť zapínanie a vypínanie vlastností `Clock.TimerEnabled` a `Ball.Enabled` pre rôzne stavy bežiacej aplikácie			
Použiť udalosť `Ball.TouchDown` pre zachytenie dotyku prstu (pera) na komponente `Ball`			
Použiť udalosť `Screen.Initialize` pre počiatočné nastavenia aplikácie			
Použiť príkaz `close application` na ukončenie behu aplikácie			
Použiť **premenné** vo výpočtoch (**inicializáciu**, bloky `set` a `get`)			
Použiť **generovanie náhodných celých čísel** vo výpočtoch			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.3 Hra Guľka

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`OrientationSensor`	`OrientationChanged (roll, pitch)`		`Enabled`

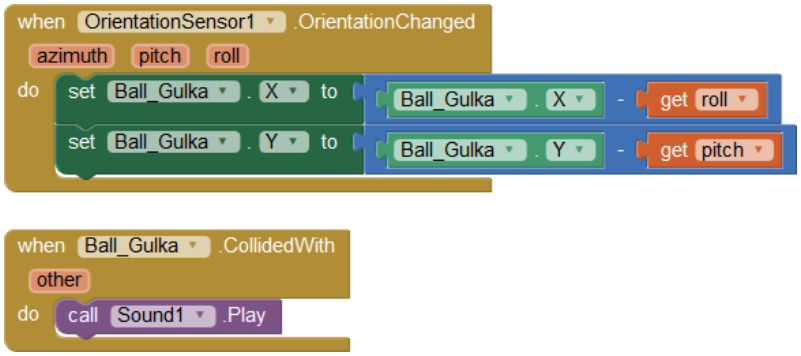
`Ball`	`CollidedWith`		`X`
			`Y`
			`Visible`
			`PaintColor`

`Clock`		`Now` `Duration`	
`TinyDB`		`GetValue` `StoreValue`	

Jazykové konštrukcie	Iné prvky jazyka
Príkaz vetvenia `IF`	

Úloha 1

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_3_hra_gulka1.aia**. Po jeho nainštalovaní a spustení na MZ preskúmajte správanie sa tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuliek.



Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `OrientationSensor`? b. Aký význam majú `roll` a `pitch` ako parametre udalosti `OrientationSensor.OrientationChanged`? c. Aký význam má udalosť `Ball.CollidedWith`? d. Ako sa správajú komponenty `Ball_Gulka` a `Ball_Jama`?	a. b. c. d. `Ball_Gulka` `Ball_Jama`

Komponent	Udalosť	Akcia

`OrientationSensor ( )`	`OrientationChanged ( )`	`set Ball_Gulka.X`
		
		`set Ball_Gulka.Y`

`Ball ( )`	`CollidedWith ( )`	`Sound.Play`

Úloha 2

Upravte aplikáciu **pmz_2_3_hra_gulka1.aia**, aby mala nasledovné funkcionality:

- Pri kolízii lopty Gulka s iným komponentom (napr. loptou Jama) sa **zmení farba lopty Jama** na červenú (vlastnosť `PaintColor`), **schová sa lopta Gulka** (vlastnosť `Visible`) a **vypne sa komponent** `OrientationSensor` (vlastnosť `Enabled`)

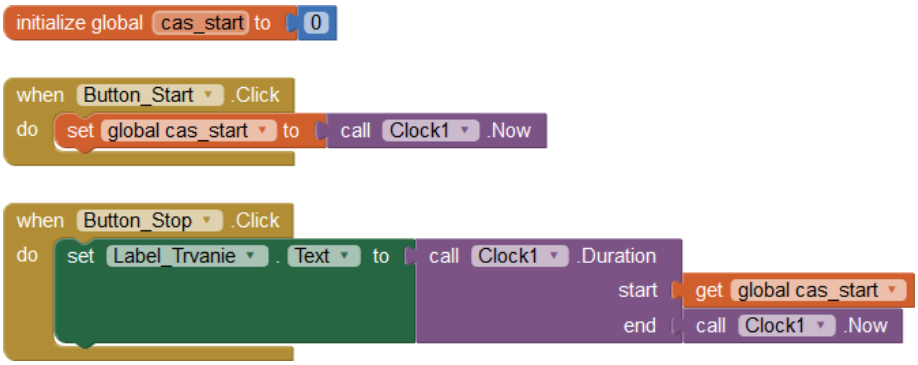
- Doplní sa tlačidlo ŠTART, ktoré **nastaví y-súradnicu lopty Jama** na dolný okraj (vlastnosť `Y`), **nastaví farbu lopty Jama** na sivú farbu, **ukáže loptu Gulka** a **zapne komponent** `OrientationSensor`

Preskúmajte rozdiel medzi vlastnosťami `Enabled` a `Visible` komponentu `Ball`.

Výsledný kód uložte do súboru **pmz_2_3_hra_gulka1_R.aia**.

Úloha 3

Preskúmajte používateľské rozhranie a správanie aplikácie so zdrojovým kódom uloženým v súbore **pmz_2_3_stopky.aia**. Svoje zistenia zapíšte do voľných políčok tabuliek.

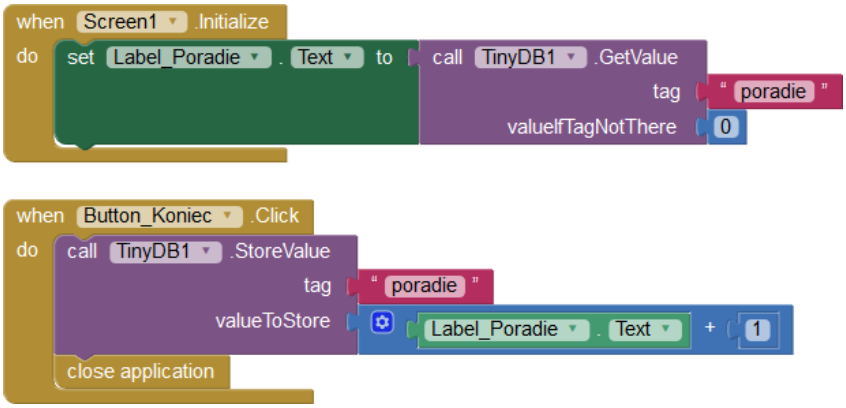


Otázka	Odpoveď
a. Aký význam majú funkcie `Clock.Now` a `Clock.Duration`? b. V akých **časových jednotkách** vracia výsledok funkcia `Clock.Duration`**?** c. V akých **situáciách** by ste **využili** tento programový kód?	a. `Clock.Now` `Clock.Duration` b. c.

Komponent	Udalosť	Akcia
`Button_Start`	`Click`	
`Button_Stop`	`Click`	

Úloha 4

Preskúmajte používateľské rozhranie a správanie aplikácie so zdrojovým kódom uloženým v súbore **pmz_2_3_pocitadlo.aia**. Svoje zistenia zapíšte do voľných políčok tabuliek.



Otázka	Odpoveď
a. Ako sa správa aplikácia po viacnásobnom spustení a ukončení?	a.
b. Akú máte skúsenosť s (nerelačnou) databázou typu **tag:value,** v ktorej sú údaje uložené ako dvojice **kľúč:hodnota**?	b.
c. Čo je **kľúčom** a čo **hodnotou** našej databázy v uvedenom programovom kóde?	c. kľúč hodnota
d. Na čo slúži metóda `TinyDB.GetValue`?	d.
e. Na čo slúži metóda `TinyDB.StoreValue`?	e.
f. Aký význam má `ValueIfTagNotThere`?	f.

Komponent	Udalosť	Akcia
`Screen`	`Initialize`	
`Button Koniec`	`Click`	

Úloha 5

Za pomoci programových kódov uvedených v predchádzajúcich úlohách vytvorte **hru Guľka**, ktorá bude mať nasledovné funkcionality:

- Nakláňaním MZ sa snažíme dostať malú guľku (loptu) do väčšej kruhovej jamky

- Ak dostaneme guľku do jamky, guľka sa schová, jamka sa zafarbí na červeno a hra končí

- Po skončení hry sa zaznamená čas trvania hry do databázy, ale len vtedy, ak je v hre dosiahnutý čas menší ako čas predtým uložený do databázy

Výsledný kód uložte do súboru **pmz_2_3_hra_gulka2_R.aia**.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.3 Hra Guľka

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Komponent `OrientationSensor` je nevizuálnym komponentom súvisiaci s naklonením MZ			
Komponent `OrientationSensor` je neaktívnym, ak je vypnutá jeho vlastnosť `OrientationSensor.Enabled` (t. j. je nastavená na hodnotu **false**)			
Udalosť `Ball.CollidedWith` sa vyvolá pri kolízii (zrážke) komponentu `Ball` s iným animačným komponentom na plátne			
Vlastnosti `Ball.X` a `Ball.Y` predstavujú súradnice komponentu `Ball` na rodičovskom komponente `Canvas` (ľavý horný bod plátna má súradnice (0,0), súradnica y narastá zhora nadol)			
Komponent `Ball` je viditeľným, resp. neviditeľným, ak je jeho vlastnosť `Ball.Visible` nastavená na hodnotu **true**, resp. **false**			
Vlastnosť `Ball.PaintColor` predstavuje farbu komponentu `Ball`			
Rozdiel medzi vlastnosťami `Ball.Enable` a `Ball.Visible` je v tom, že prvá znamená, že komponent `Ball` je povolený a reaguje na udalosti a druhá znamená, že komponent `Ball` je viditeľný			
Metóda `Clock.Now` vracia aktuálny čas (vo vlastnom formáte)			
Metóda `Clock.Duration` vracia čas v milisekundách uplynutý medzi časmi uvedenými v parametroch **start** a **end**			
Komponent `TinyDB` slúži na uloženie údajov do zariadenia, ktoré sú reprezentované ako dvojica **kľúč**:**hodnota**			
Na ukladanie a načítanie údajov z databázy sa používajú metódy `GetValue` a `StoreValue`			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť hru, ktorá využíva: • nakláňanie zariadenia (komponent `OrientationSensor`**)**			
• kolíziu lôpt (udalosť `Ball.CollidedWith`)			
• meranie uplynutého času (metóda `Clock.Duration`)			
• ukladanie hodnôt do/z databázy (komponent `TinyDB`)			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé

normálne

nudné

ľahké

primerané

ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.4 Kalkulačka

Prvky nového učiva

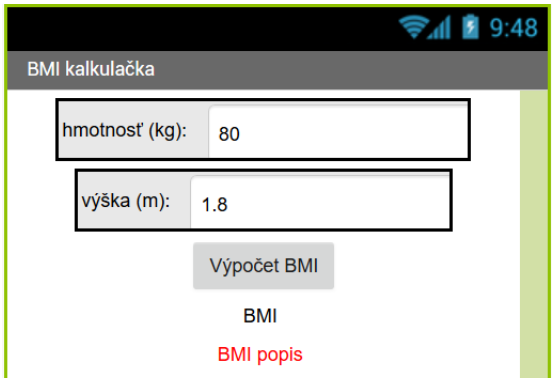
Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`TextBox`			`Text`
`Notifier`	`AfterTextInput (response)`	`ShowAlert` `ShowTextDialog` `ShowMessageDialog`	
`Slider`	`PositionChanged (minValue, maxValue, thumbPosition)`		

Jazykové konštrukcie	Iné prvky jazyka
lokálna premenná (`initialize local`, `get`, `set`) `IF-THEN-ELSE` (s výstupom, vnorený) vlastná funkcia s parametrom	`join`

Úloha 1

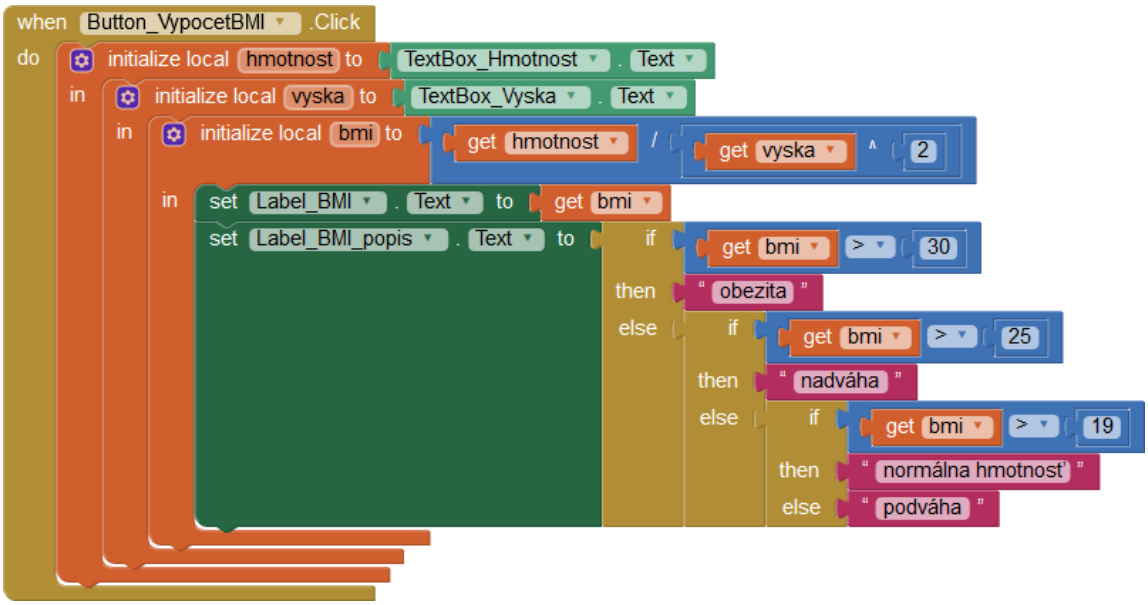
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_4_kalkulacka1a.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

Používateľské rozhranie:



Otázka	Odpoveď
a. Ktoré komponenty v tejto aplikácii zabezpečujú **načítanie vstupov**? b. Ktoré komponenty v tejto aplikácii zabezpečujú **spracovanie vstupov**? c. Ktoré komponenty v tejto aplikácii zabezpečujú **výpis výstupov**? d. Na **výpočet čoho** je vhodná táto aplikácia?	a. b. c. d.

Zdrojový kód:

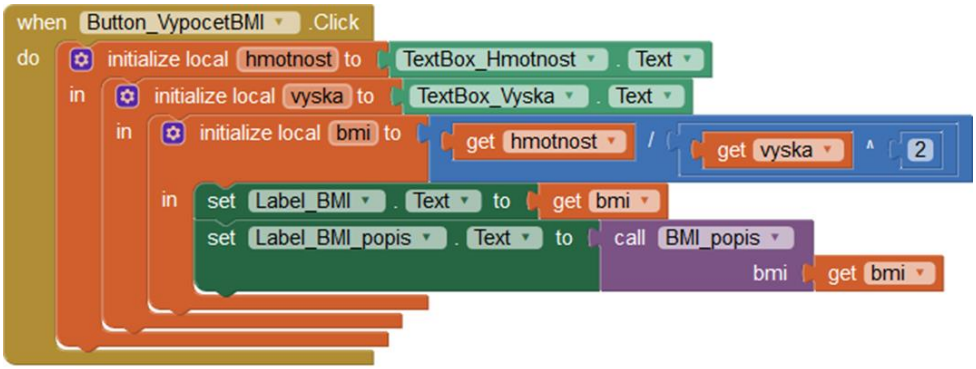


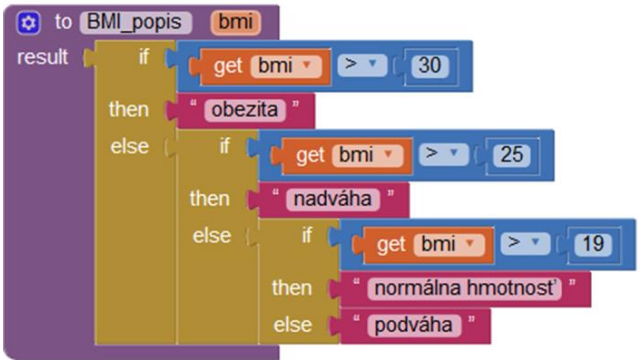
Otázka	Odpoveď
e. Ktoré **globálne premenné** a ktoré **lokálne premenné** sú použité v zdrojovom kóde?	e. globálne lokálne
f. Uveďte **matematický výraz**, ktorý je priradený do premennej **bmi**.	f.
g. Uveďte **výsledok**, ktorý bude uložený **v komponente** `Label_BMI_popis` pre nasledovné hodnoty **bmi**:	g. bmi = 24 Label_BMI_popis = bmi = 19 Label_BMI_popis = bmi = 31 Label_BMI_popis = bmi = 18 Label_BMI_popis =

Úloha 2

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_4_kalkulacka1b.aia** a preskúmajte ho. Svoje zistenia zapíšte do voľných políčok tabuľky.

Zdrojový kód:





Otázka	Odpoveď
a. **V čom sa líši** zdrojový kód aplikácie **pmz_2_4_kalkulacka1b** od kódu **pmz_2_4_kalkulacka1a**? b. Aké **výhody** prináša používanie **vlastných funkcií**?	a. b.

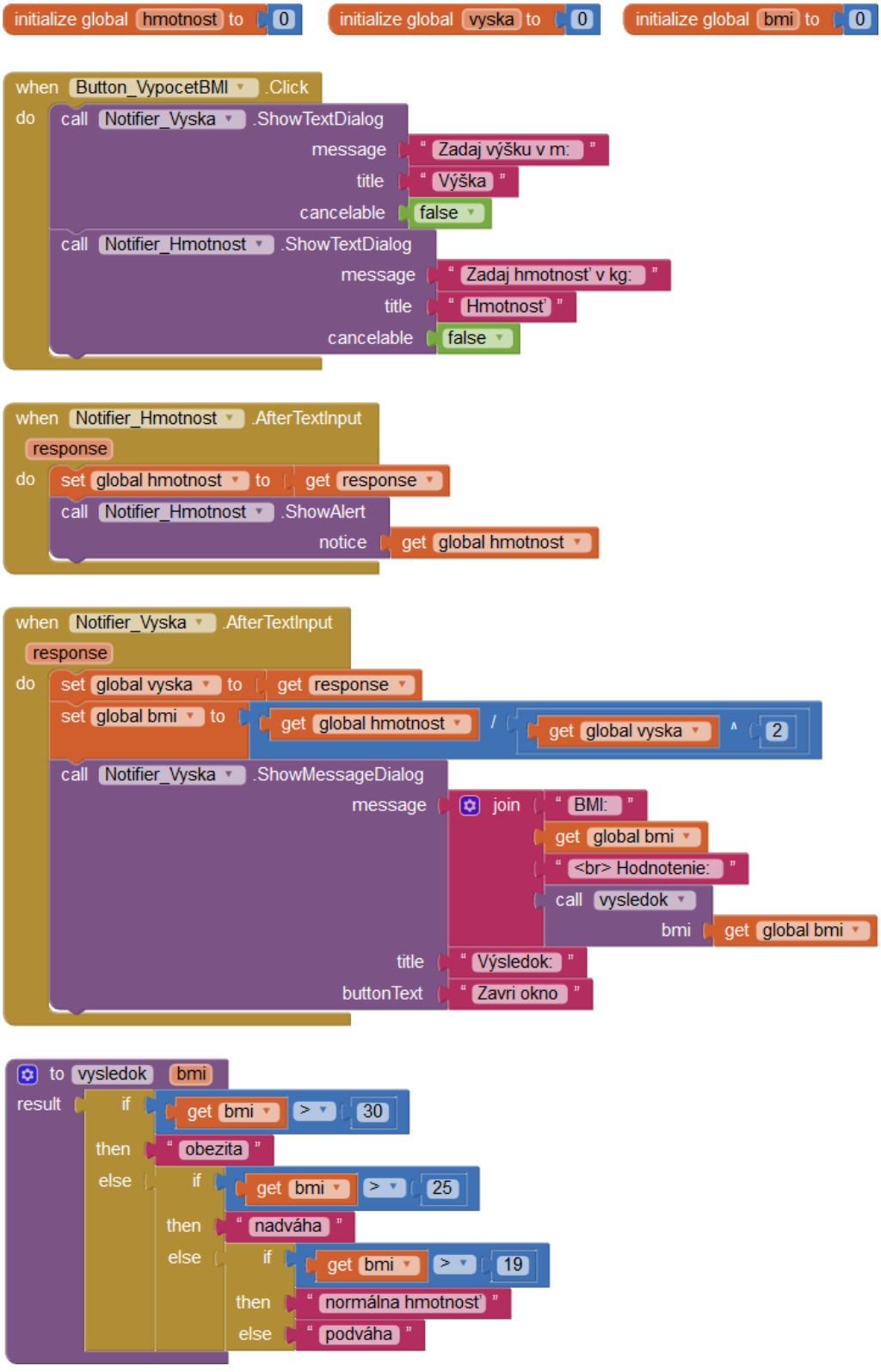
Úloha 3

Rozšírte aplikáciu **pmz_2_4_kalkulacka1b.aia**, aby sa vstupné hodnoty mohli zadať nielen pomocou komponentov `TextBox`, ale aj pomocou komponentov `Slider` (posúvačov). Výsledný kód uložte do súboru **pmz_2_4_kalkulacka2_R.aia**.

(Odporúčanie: V režime **Designer** rozšírime rozhranie aplikácie o dva komponenty `Slider`, v ktorých nastavíme dolnú a hornú hranicu povolených hodnôt. V režime **Blocks** doplníme dve udalosti `Slider.PositionChanged`, v ktorých nastavíme textové polia na aktuálnu hodnotu posúvača v parametri `thumbPosition` danej udalosti.)

Úloha 4

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_4_kalkulacka3.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.



Otázka	Odpoveď

a. Vysvetlite **funkcionalitu** použitých **metód** komponentu `Notifier`:	a. `Notifier.ShowTextDialog`
	`Notifier.ShowAlert`
	`Notifier.ShowMessageDialog`

b. Ako by sa zmenil výpočet programu, ak by sme v udalosti Button_VypocetBMI.Click **prehodili volania** oboch uvedených metód?	b.
c. Ktorá z **metód** komponentu `Notifier` **vyvolá** udalosť `Notifier.AfterTextInput`?	c.
d. Ktoré z **metód** komponentu `Notifier` by ste **použili** v uvedených **situáciách**:	d. len na výpis údajov na načítanie údajov

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.4 Kalkulačka BMI

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	rozumiem	čiastočne rozumiem	vôbec nerozumiem
Vo výpočte môžeme použiť **lokálne premenné**, ktoré sú dostupné len vo vymedzenej časti programu			
Príkaz `IF-THEN-ELSE` môžeme viackrát **vnoriť do seba**			
Príkaz `IF-THEN-ELSE` môže mať aj **výstup** z vetiev `THEN` a `ELSE`			
Vo výrazoch na výpočet **mocniny** sa používa **blok** `^`			
Vlastné funkcie (procedúry s výstupom) sa používajú na **sprehľadnenie** výpočtu, na **opätovné použitie** v programe, pri **tímovej tvorbe** programov			
Komponent `Slider` je vizuálnym komponentom súvisiaci so vstupom údajov podľa polohy posúvača			
Udalosť `Slider.PositionChanged` sa vyvolá zmenou polohy bežca posúvača medzi nastavenými krajnými pozíciami **minValue** a **maxValue,** pričom aktuálna hodnota posúvača je uložená v jej parametri `thumbPosition`			
Komponent `Notifier` je vizuálnym komponentom umožňujúcim vstupy a výstupy údajov pomocou dialógových okien			
Metóda `Notifier.ShowAlert` slúži na krátkodobý výpis správy v okne			
Metóda `Notifier.ShowMessageDialog` slúži na výpis správy v okne ukončený stlačením tlačidla okna			
Metóda `Notifier.ShowTextDialog` slúži na zadanie vstupných údajov pomocou dialógového okna			
Udalosť `Notifier.AfterTextInput` sa vyvolá po spustení metódy `Notifier.ShowTextDialog`, pričom			

aktuálne zadaný vstup je uložený v jej parametri **response**			
V metóde `Notifier.ShowMessageDialog` v parametri **message** môžeme na viacriadkový výpis použiť HTML značku ** **			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť výpočtovú aplikáciu využívajúcu na vstupy a výstupy: • komponenty `TextBox` a `Label`			
• komponent `Notifier`			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé

normálne

nudné

ľahké

primerané

ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.5 Zbierka vtipov

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`Screen`	`OtherScreenClosed (result)`		
`Canvas`	`Flung (xvel)`		

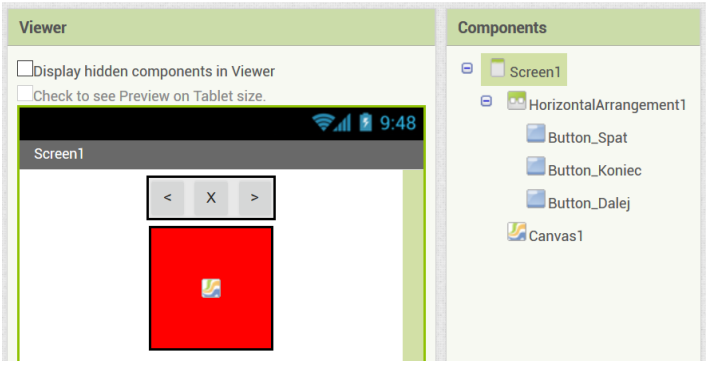
Jazykové konštrukcie	Iné prvky jazyka

	`open another screen`
	`close screen`
	`close screen with value`

Úloha 1

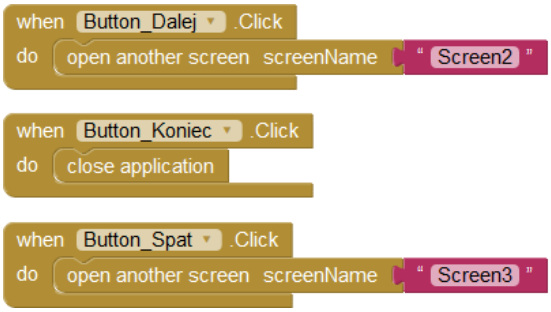
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_5_vtipy1.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

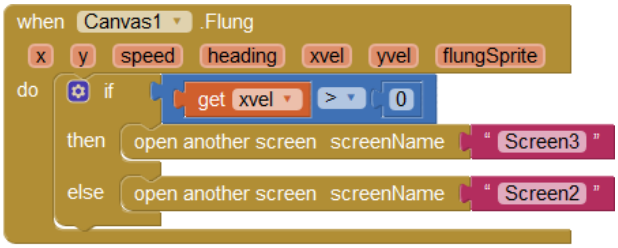
Používateľské rozhranie obrazovky `Screen1`:



Otázka	Odpoveď
a. Čo sa stane po **kliknutí na tlačidlá** označené symbolmi „**<**“ a „**>**“?	a.
b. Čo sa stane, ak **potiahneme prstom vľavo** (resp. vpravo) po farebnom plátne?	b.
c. **Koľko obrazoviek** obsahuje táto aplikácia?	c.
d. Má každá obrazovka svoje **multimediálne súbory**?	d. ÁNO / NIE
e. Má každá obrazovka svoje **programové kódy**?	e. ÁNO / NIE
f. Uveďte aspoň jeden vlastný **námet na aplikáciu** s viacerými obrazovkami:	f.

Zdrojový kód obrazovky `Screen1`:





Otázka	Odpoveď
g. Aké použitie má príkaz `open another screen`?	g.
h. Akou činnosťou používateľa sa spúšťa udalosť `Canvas.Flung`?	h.
i. Čo predstavuje parameter `xvel` udalosti `Canvas.Flung`?	i.

Úloha 2

Upravte aplikáciu **pmz_2_5_vtipy1.aia**, aby tlačidlá namiesto symbolov „**<**“, „**X**“ a „**>**“ boli reprezentované vhodnými obrázkami a komponenty `Canvas` namiesto rôznych farieb pozadia boli reprezentované vhodnými obrázkami s vtipmi či vtipnými zadaniami úloh. Aplikáciu rozšírte aspoň o jednu stranu s vtipom či vtipným zadaním úlohy. Výsledný kód uložte do súboru **pmz_2_5_vtipy1_R.aia**.

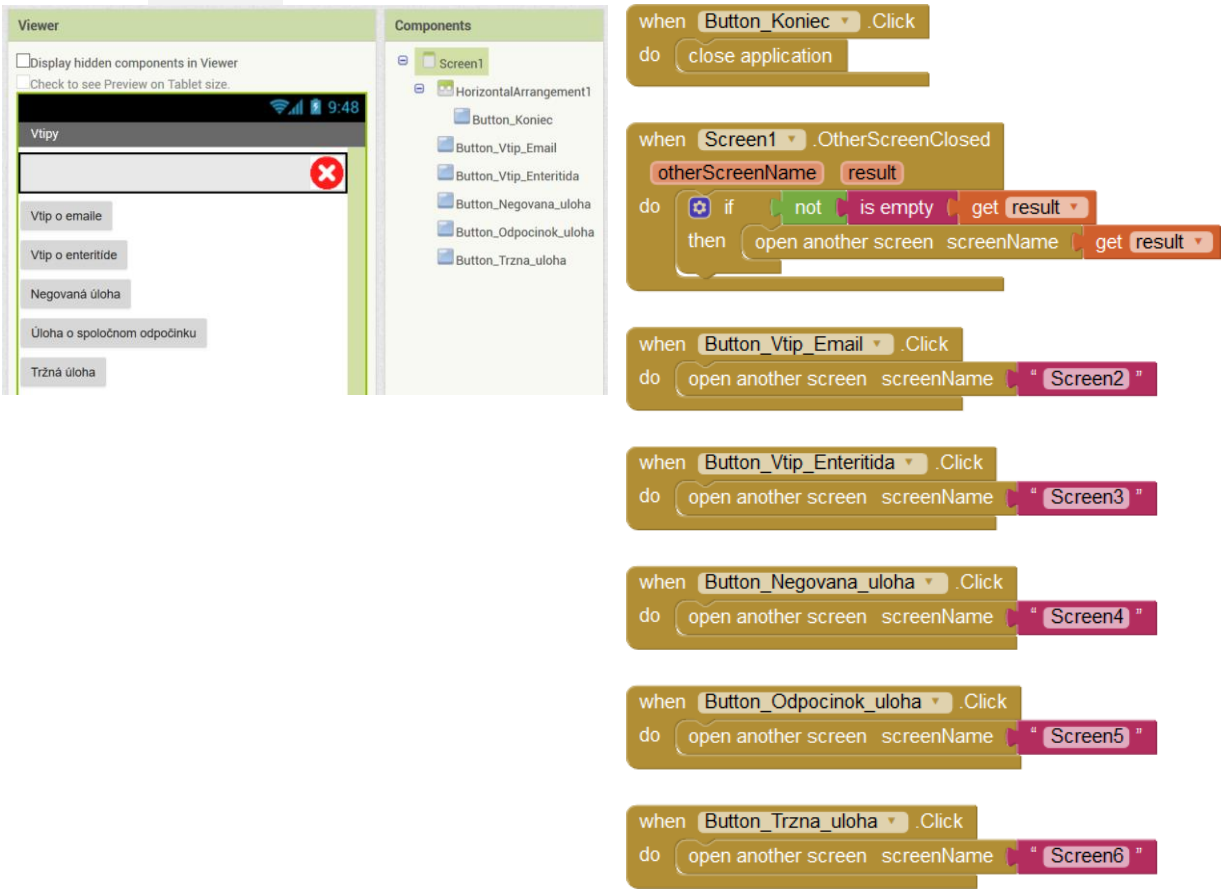
(Odporúčanie: Pri kopírovaní programového kódu z jednej obrazovky do druhej odporúčame použiť **schránku** (ikona modrozeleného batohu vpravo hore). Na kopírovanie objektov do batohu a z batohu používame kontextovú ponuku vyvolanú stlačením pravého gombíka myši. Obsah batohu zobrazíme stlačením ľavého gombíka myši.)

Úloha 3

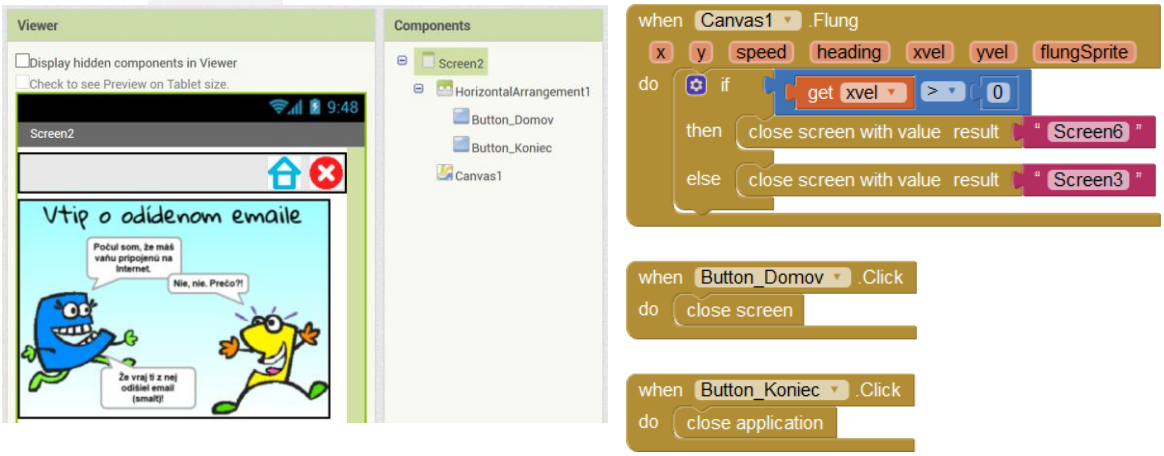
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_5_vtipy2.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie a zdrojový kód:

Obrazovka `Screen1`



Obrazovka `Screen2`



Otázka	Odpoveď
a. **Koľko obrazoviek** obsahuje táto aplikácia?	a.
b. Ako sa **líši prvá obrazovka** od ostatných?	b.
c. Akú funkcionalitu zastupuje udalosť `Screen.OtherScreenClosed`?	c.
d. Prečo je **v tejto udalosti** uvedená **podmienka**?	d.
e. Prečo je na Screen2 v udalosti `Canvas.Flung` použitý príkaz `close screen` a nie `open screen`?	e.
f. **V čom je lepšie** riešenie aplikácie **vtipy2** od aplikácie **vtipy1**?	f.
g. Uveďte ďalší vlastný **námet na aplikáciu** s viacerými obrazovkami:	g.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.5 Zbierka vtipov

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
V aplikácii s viacerými obrazovkami má **každá obrazovka** svoj vlastný **programový kód**			
V aplikácii s viacerými obrazovkami sú nahrané **multimediálne súbory** prístupné **pre každú obrazovku**			
Viaceré obrazovky je vhodné použiť v aplikáciách, ktoré potrebujú zobraziť **viacero** údajov s **rôznym rozmiestnením** a **účelom** (napr. Úvod, Pomoc, Nastavenia, Výsledky)			
Na **otvorenie** obrazovky sa používa príkaz `open another screen` so zadaným menom obrazovky			
Na **uzavretie** obrazovky sa používa príkaz `close another screen` so zadaným výsledkom (napr. menom obrazovky), resp. príkaz `close screen`, ak ide o aktuálnu obrazovku			
Udalosť `Screen.OtherScreenClosed` sa vyvolá po uzavretí nejakej obrazovky, pričom v jej parametri `result` je uložený výsledok (napr. meno nasledovnej obrazovky), resp. `result` je prázdny reťazec po uzavretí obrazovky príkazom `close screen`			
Na otváranie obrazoviek môžeme použiť rôzne komponenty, napr. `Button`, `Canvas`			
Udalosť `Canvas.Flunge` sa vyvolá pomocou dotykového gesta, napr. potiahnutím vpravo, čo indikuje kladná hodnota jej parametri `xvel`			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť aplikáciu s viacerými obrazovkami: • **s rovnakým** rozmiestnením obsahu **bez uzatvárania** obrazoviek			
• **s rôznym** rozmiestením obsahu obrazoviek **aj s uzatváraním** obrazoviek			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.6 Čítačka QR kódu

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`BarcodeScanner`	`AfterScan (result)`	`DoScan`	`UseExternalScanner`

`TextToSpeech`		`Speak`	`Country`
			`Language`
			`SpeechRate`
			`Pitch`

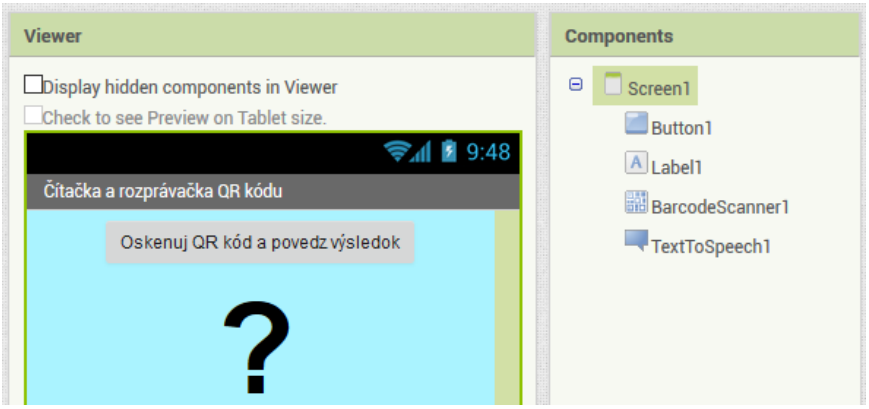
`Spinner`	`AfterSelecting (selection)`		
`SpeechRecognizer`	`AfterGettingText (result)`	`GetText`	

Jazykové konštrukcie	Iné prvky jazyka

Úloha 1

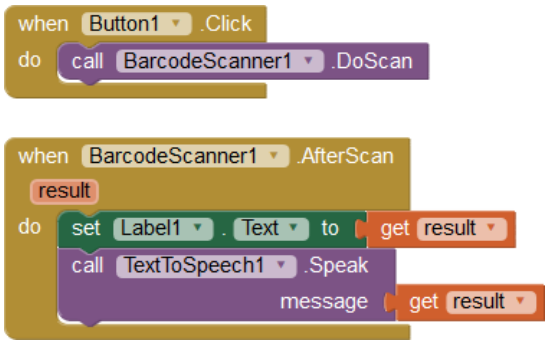
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_6_QR_kod1.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

Používateľské rozhranie:



Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `BarcodeScanner`?	a.
b. V ktorej skupine komponentov je uvedený komponent `TextToSpeech`?	b.
c. Aký význam má komponent `BarcodeScanner`?	c.
d. Aký význam má komponent `TextToSpeech`?	d.
e. Pomocou ktorej **aplikácie** v MZ sa skenujú čiarové kódy?	e.

Zdrojový kód:

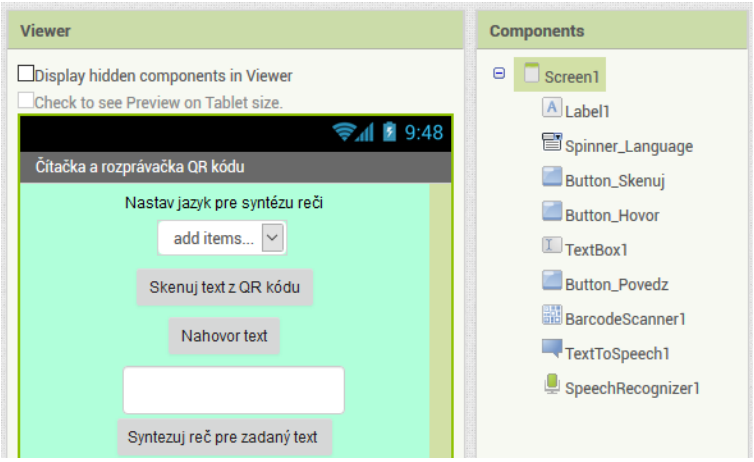


Otázka	Odpoveď
f. Čo sa stane po spustení metódy `BarcodeScanner.DoScan`?	f.
g. Čo vyvolalo udalosť `BarcodeScanner.AfterScan` a kedy?	g.
h. Aký význam v udalosti `BarcodeScanner.AfterScan` má parameter `result`?	h.
i. Čo sa stane, ak v komponente `BarcodeScanner` zaškrtneme vlastnosť `UseExternalScanner`?	i.
j. Čo robí metóda `TextToSpeech.Speak`?	j.

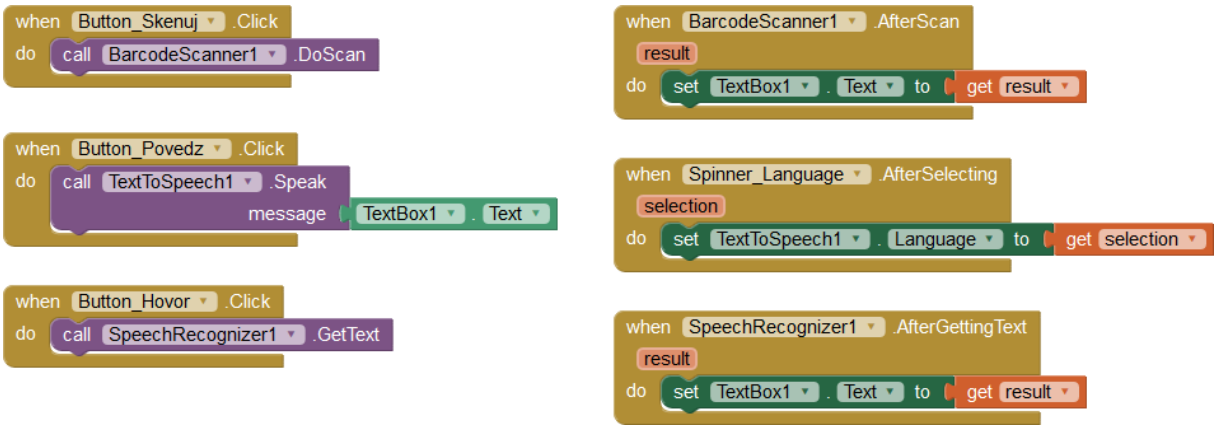
Úloha 2

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_6_QR_kod2.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie:



Zdrojový kód:



Otázka	Odpoveď
a. Na čo slúži komponent `Spinner`? b. Aký význam v komponente `Spinner` majú uvedené vlastností? c. Kedy sa vyvolá udalosť `Spinner.AfterSelecting`? d. Aký význam má jej parameter `selection`? e. Aký význam v komponente `TextToSpeech` má vlastnosť `TextToSpeech.Language`? f. Na čo slúži komponent `SpeechRecognizer`? g. Kedy sa vyvolá udalosť `SpeechRecognizer.AfterGettingText`? h. Aký význam má jej parameter `result`?	a. b. Prompt ElementsFromString Selection c. d. e. f. g. h.

Úloha 3

Doplňte aplikáciu **pmz_2_6_QR_kod2.aia** o ďalšie tri komponenty **Spinner** na nastavenie **rýchlosti reči**, **výšku hlasu** a **krajiny**, v ktorých nastavíte aspoň 1 ďalší jazyk. Výsledný kód uložte do súboru **pmz_2_6_QR_kod2_R.aia**.

(Odporúčanie: Pre nastavenie krajiny (Country) sa používajú trojpísmenové skratky, napr. SVK, USA, GBR, CZE a pre nastavenie jazyka (Language) sa používajú dvojpísmenové skratky, napr. sk, en, cz. Ďalšie kódy krajín a jazykov sa dajú nájsť na https://docs.thunkable.com/thunkable-classic-android/create/components/voice/text-to-speech
či https://cloud.google.com/speech-to-text/docs/languages)

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.6 Čítačka QR kódu

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Nevizuálny komponent `BarcodeScanner` slúži na skenovanie čiarových kódov			
Pomocou metódy `BarcodeScanner.DoScan` sa spustí interný alebo externý skener čiarových kódov, čo závisí od zaškrtnutia vlastnosti `BarcodeScanner.UseExternalScanner`			
Po ukončení skenovania sa vyvolá udalosť `BarcodeScanner.AfterScan`, ktorá výsledok skenovania má uložený v parametri `result`			
Nevizuálny komponent `TextToSpeech` slúži na rečovú syntézu			
Metóda `TextToSpeech.Speak` syntezuje reč pre text zadaný v parametri `message`			
Parametre syntezovanej reči (krajina, jazyk, rýchlosť reči, výška hlasu) sa dajú v komponente `TextToSpeech` nastaviť pomocou vlastností `Country`, `Language`, `SpeechRate`, `Pitch`			
Vizuálny komponent `Spinner` sa používa na vstup údajov z vymenovaného zoznamu prvkov, ktoré sú uvedené (oddelené čiarkou) vo vlastnosti `ElementsFromString`			
Po výbere hodnoty v komponente `Spinner` sa vyvolá udalosť `Spinner.AfterSelecting`, ktorá vybranú hodnotu má uloženú v parametri `selection`			
Nevizuálny komponent `SpeechRecognizer` slúži na analýzu reči			
Po rozpoznaní reči sa vyvolá udalosť `SpeechRecognizer.AfterGettingText`, ktorá má výsledok rozpoznávania uložený v parametri `result`			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť aplikáciu na **čítanie čiarových kódov**, ktorá využíva: • **syntézu reči**			
• **analýzu reči**			
• **výberový zoznam** (komponent Spinner)			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé

normálne

nudné

ľahké

primerané

ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.7 Asistent pri cvičení

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`ProximitySensor`	`ProximityChanged (distance)`		`Enabled`
`Sound`		`Vibrate`	
`CheckBox`			`Checked`

`Pedometer`	`WalkStep (walkSteps, distance)`	`Start`
		`Resume`
		`Reset`
		`Pause`
		`Save`
			`WalkSteps`
			`Distance`
			`ElapsedTime`
			`StrideLength`
			`StopDetection Timeout`

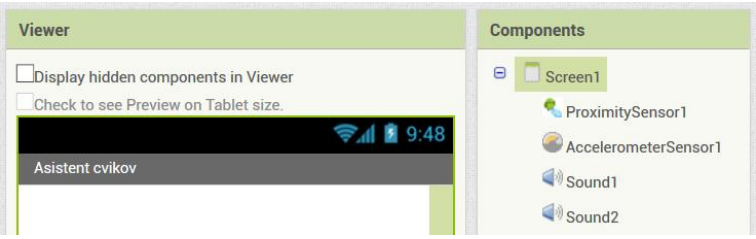
`AccelerometerSensor`			`Enabled`
`TableArrangement`			

Jazykové konštrukcie	Iné prvky jazyka

Úloha 1

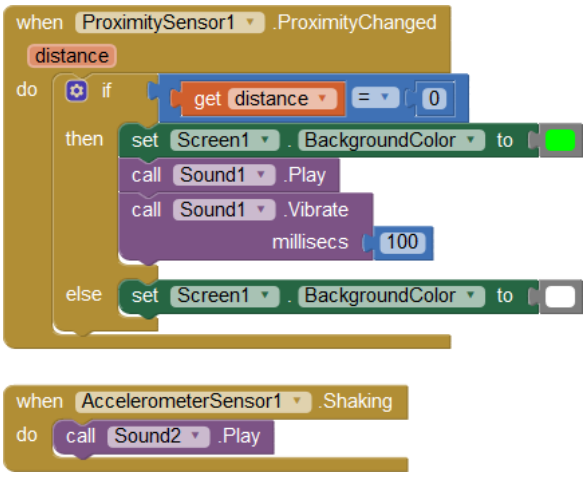
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_7_cviky1.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

Používateľské rozhranie:



Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `ProximitySensor`?	a.
b. Na čo sa využíva komponent `ProximitySensor`?	b.
c. Na **aký účel** by ste využili túto aplikáciu?	c.

Zdrojový kód:



Otázka	Odpoveď
d. Akou činnosťou sa vyvolá udalosť `ProximitySensor.ProximityChanged`? e. Aký význam má parameter `distance` udalosti `ProximitySensor.ProximityChanged`? f. Akú funkcionalitu predstavuje metóda `Sound.Vibrate`?	d. e. f.

Úloha 2

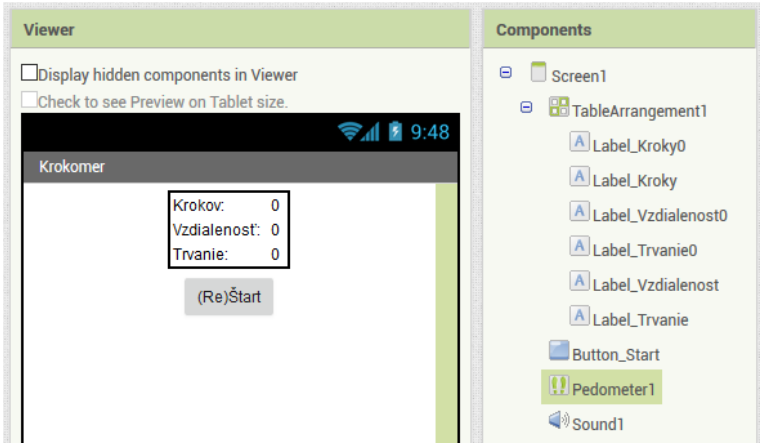
Doplňte aplikáciu **pmz_2_7_cviky1.aia**, aby zaznamenávala, resetovala a vypisovala počty priblížení a zatrasení MZ a tiež, aby umožňovala zapínať a vypínať senzory priblíženia a zrýchlenia. Výsledný kód uložte do súboru **pmz_2_7_cviky1_R.aia**.

(Odporúčanie: Na zapnutie a vypnutie senzorov priblíženia a zrýchlenia použite komponent `CheckBox`, pomocou ktorého sa dajú nastaviť vlastnosti `ProximitySensor.Enabled` a `AccelerometerSensor.Enabled`. Po zaškrtnutí a odškrtnutí komponentu `CheckBox` sa vyvolá udalosť `CheckBox.Changed`, stav komponentu `CheckBox` reprezentuje jeho vlastnosť `Checked`.)

Úloha 3

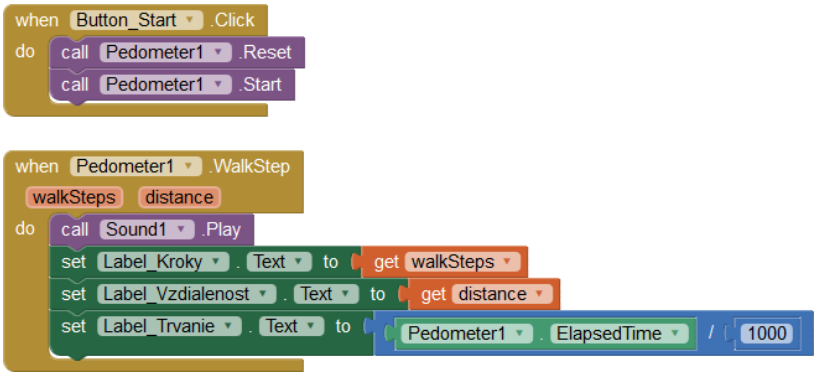
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_7_krokomer1.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie:



Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `Pedometer`? b. Na čo sa využíva komponent `Pedometer`? c. Aké hodnoty a aký význam v komponente `Pedometer` majú uvedené vlastnosti?	a. b. c. `StrideLength` `StopDetectionTimeout`

Zdrojový kód:



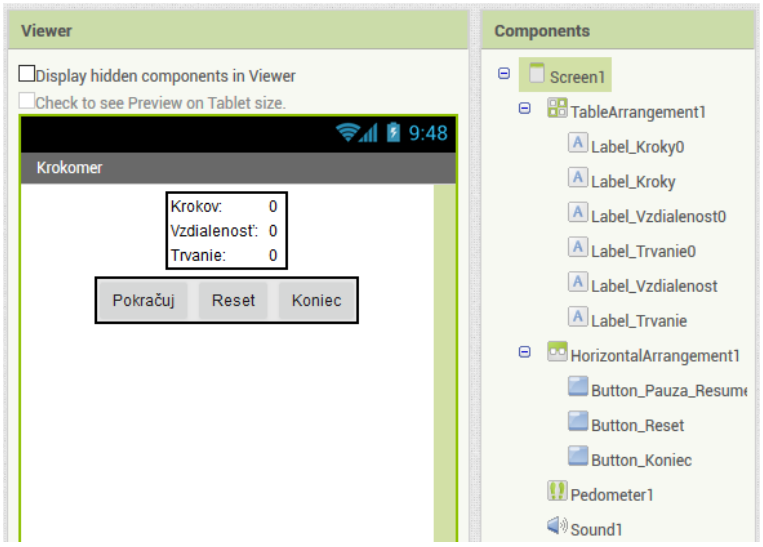
Otázka	Odpoveď
d. Čo robí metóda `Pedometer.Reset`? e. Ako by sa zmenilo správanie programu, ak by sa vynechala metóda `Pedometer.Reset`? f. Čo robí metóda `Pedometer.Start`? g. Akou činnosťou sa vyvolá udalosť `Pedometer.WalkStep`?	d. e. f. g.

h. Aký význam v udalosti `Pedometer.WalkStep` má jej parameter `walkSteps`? i. Aký význam v udalosti `Pedometer.WalkStep` má jej parameter `distance`? j. Ako by sa zmenilo správanie programu, ak by sa v udalosti `Pedometer.WalkStep` namiesto parametra `walkSteps` použila vlastnosť `Pedometer.WalkSteps`? k. Aký význam má vlastnosť `Pedometer.ElapsedTime`?	h. i. j. k.

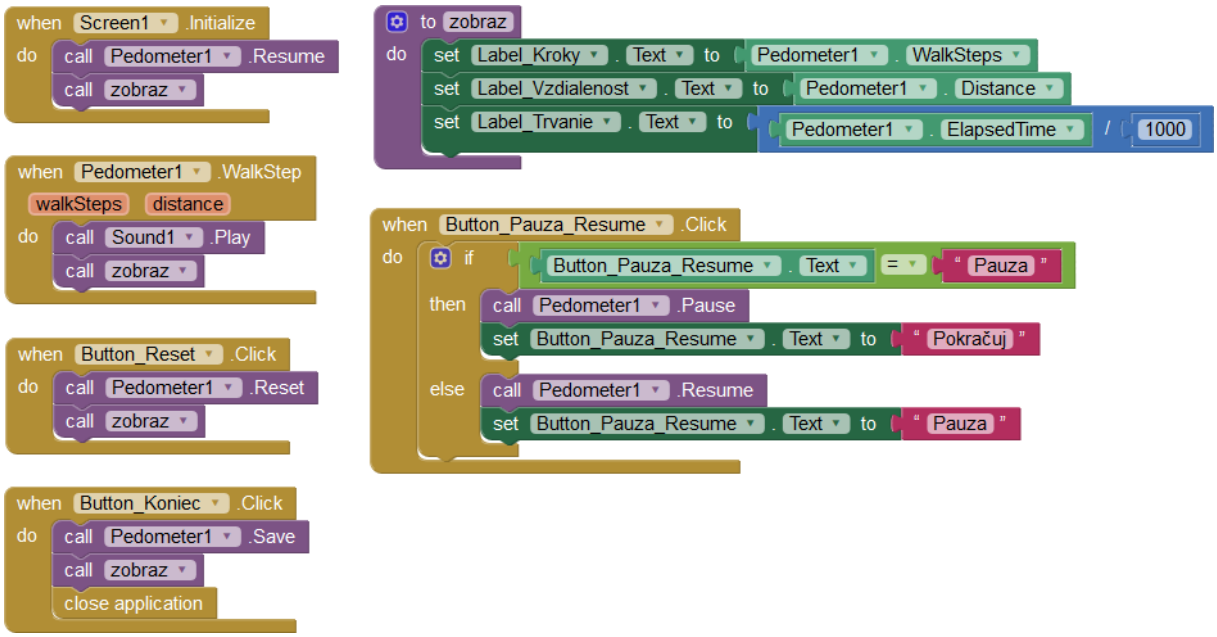
Úloha 4

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_7_krokomer2.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie:



Zdrojový kód:



Otázka	Odpoveď
a. Aký význam má metóda `Pedometer.Save` v súvislosti s **opätovným spustením** aplikácie?	a.
b. Prečo v programe chýba metóda `Pedometer.Start`? Ktorá metóda ju zastupuje?	b.
c. Čo sa deje po stlačení tlačidla `Button_Pauza_Resume`?	c.
d. Uveďte, ktoré **ďalšie užitočné funkcionality** by sa dali doplniť do tejto aplikácie?	d.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.7 Asistent pri cvičení

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	rozumiem	čiastočne rozumiem	vôbec nerozumiem
Nevizuálny komponent `ProximitySensor` sa používa na registrovanie objektu v blízkosti MZ			
Priblíženie objektu (ucha) k MZ (telefónu) vyvolá udalosť `ProximitySensor.ProximityChanged`, ktorá v parametri `distance` má uloženú jednu z dvoch hodnôt: **blízko** (0 cm) a **ďaleko** (napr. 8 cm)			
Vibrovanie zariadenia vyvoláme spustením metódy `Sound.Vibrate`			
Vizuálny komponent `CheckBox` sa používa na zaškrtnutie, resp. odškrtnutie možnosti			
Stav komponentu `CheckBox` reprezentuje jeho vlastnosť `Checked`			
Nevizuálny komponent `Pedometer` sa používa na meranie počtu krokov			
Chôdza s MZ vyvolá udalosť `Pedometer.WalkStep`, ktorá v parametri `walkSteps` má uložený počet prejdených krokov a v parametri `distance` prejdenú vzdialenosť			
Počet prejdených krokov, resp. prejdenú vzdialenosť sú uložené vo vlastnostiach `Pedometer.WalkSteps`, resp. `Pedometer.Distance`			
Čas chôdze je uložený vo vlastnosti `Pedometer.ElapsedTime`			
Spustenie krokomera sa vykoná pomocou metódy `Pedometer.Start`, resp. `Pedometer.Resume` po pozastavení			
Resetovanie hodnôt počtu krokov, prejdenej vzdialenosti a času chôdze sa vykoná pomocou metódy `Pedometer.Reset`			
Pozastavenie merania počtu krokov a vzdialenosti sa vykoná pomocou metódy `Pedometer.Pause`			
Uloženie stavu komponentu `Pedometer` do MZ sa vykoná pomocou metódy `Pedometer.Save`			

Dĺžka kroku je uložená vo vlastnosti `Pedometer.StrideLength` (štandardne je nastavená hodnota 0.73 m)			
Čas nečinnosti, po ktorom sa zastaví meranie krokov, je uložený vo vlastnosti `Pedometer.StopDetectionTimeout` (štandardne je nastavená hodnota 2000 ms)			
Na usporiadanie vizuálnych komponentov do tabuľky sa používa komponent `TableArrangement` zo skupiny **Layout**			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť aplikáciu registrujúcu pohyb zariadenia využitím: • komponentu `ProximitySensor`			
• komponentu `AccelerometerSensor`			
• komponentu `Pedometer`			
• komponentu `CheckBox`			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.8 Generátor náhodných viet

Prvky nového učiva

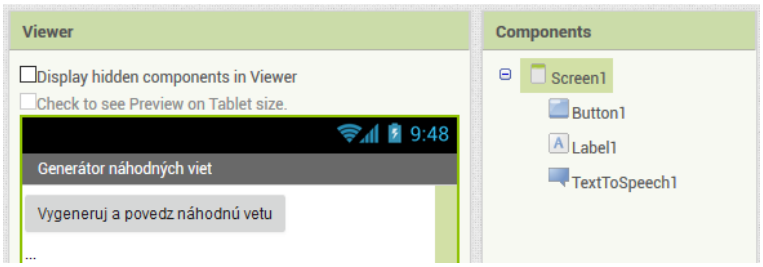
Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`ListView`			`Elements`
Jazykové konštrukcie		**Iné prvky jazyka**	

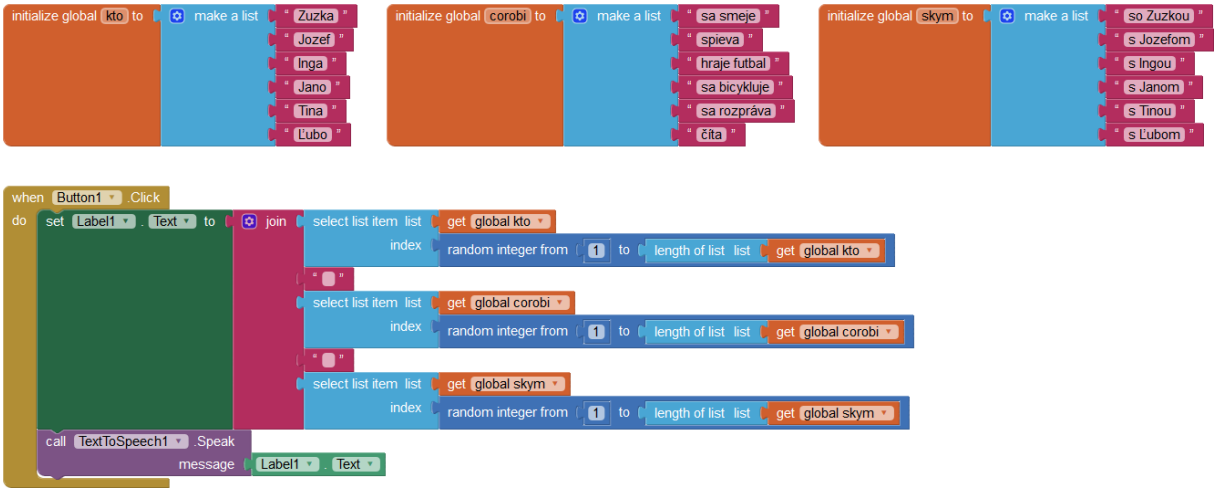
`list (make a list, select list item, length of list, remove list item, insert list item, create empty list, add items to list)`
`FOR EACH NUMBER`
`FOR EACH ITEM`			

Úloha 1

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_8_vety.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie a zdrojový kód aplikácie:





Otázka	Odpoveď
a. **Čo robí** daná aplikácia? b. Dáva **rovnaké výsledky** po opätovnom spustení? c. Prečo sú v zdrojovom kóde použité premenné typu **zoznam**?	a. b. c.

d. Aký význam má funkcia `make a list`? e. Aký význam má funkcia `select list item`? f. Aký význam má funkcia `length of list`?	d. e. f.

Úloha 2

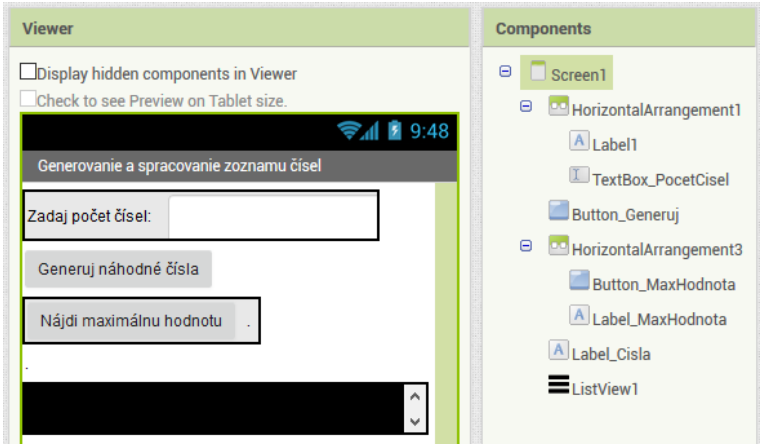
Upravte aplikáciu **pmz_2_8_vety.aia**, aby sa negenerovali dve rovnaké krstné mená v podmete a predmete. Aplikáciu rozšírte o generovanie ďalších vetných členov, napr. prívlastok, príslovkové určenie miest či času. Výsledný kód uložte do súboru **pmz_2_8_vety_R.aia**.

(Odporúčanie: Pri riešení tohto problému môžeme vhodne použiť odobranie prvku zoznamu pomocou funkcie `remove list item` a vloženie prvku do zoznamu pomocou funkcie `insert list item`.)

Úloha 3

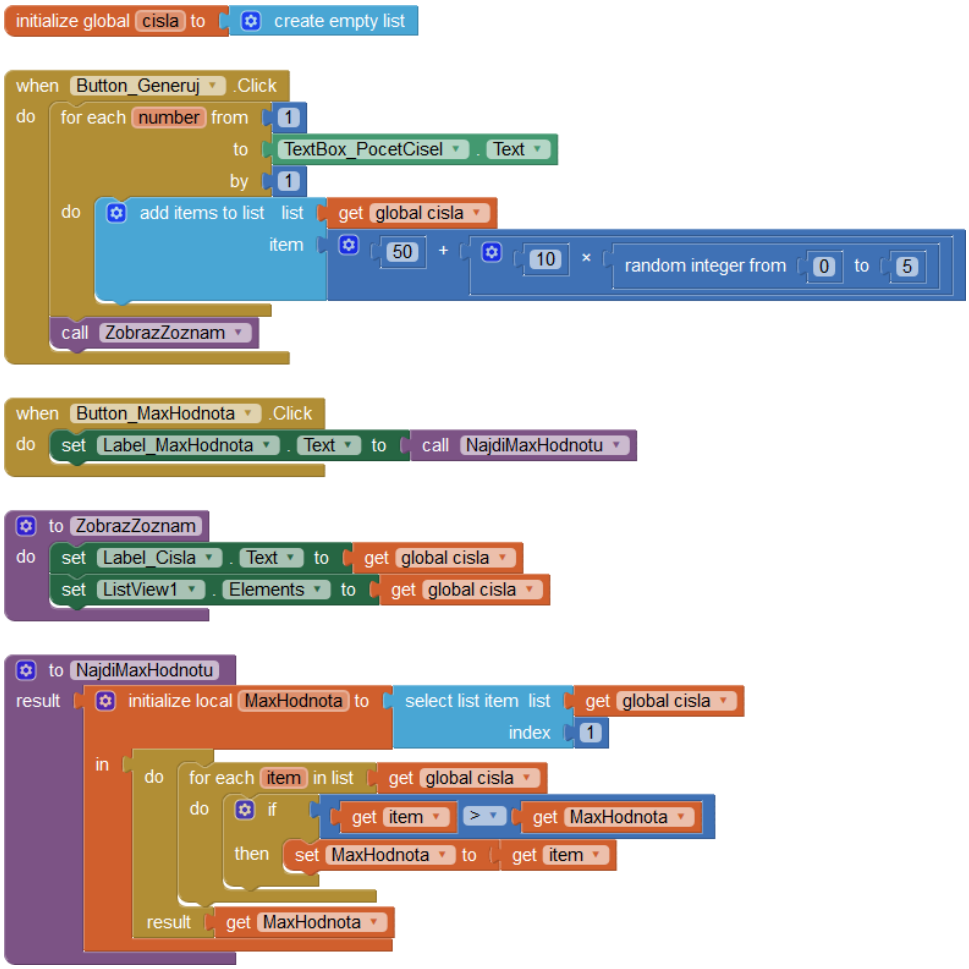
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_8_zoznam_cisel.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie:



Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `ListView`? b. Na čo sa využíva komponent `ListView`?	a. b.

Zdrojový kód:



Otázka	Odpoveď
a. Čo vracia funkcia `create empty list`? b. Čo sa nastavuje pomocou vlastnosti `ListView.Elements`? c. Čo vracia funkcia `add items to list`? d. Ako sa líšia od seba cykly `for each number` a `for each item in list`?	a. b. c. d. `for each number` `for each item in list`

Úloha 4

Doplňte aplikáciu **pmz_2_8_zoznam_cisel.aia** o tlačidlo, ktoré by **vyprázdnilo zoznam** čísel a tlačidlo, ktoré by spustilo **výpočet priemernej hodnoty** zadaného zoznamu. Výsledný kód uložte do súboru **pmz_2_8_zoznam_cisel_R.aia**.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.8 Generátor náhodných viet

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Zoznam (List) je štruktúrovaný údajový typ na spracovanie skupiny údajov			
Zoznam sa vytvára pomocou funkcie `make a list`			
Prvok zoznamu sa sprístupňuje funkciou `select list item`			
Dĺžku zoznamu vracia funkcia `length of list`			
Prvok sa odoberá zo zoznamu pomocou funkcie `remove list item`			
Prvok sa vkladá do zoznamu pomocou funkcie `insert list item`			
Na zobrazenie prvkov zoznamu sa používa komponent `ListView` zo skupiny komponentov **User Interface**			
Zoznam sa zobrazí v komponente `ListView` nastavením jej vlastnosti `ListView.Elements`			
Prázdny zoznam sa vytvorí pomocou funkcie `create empty list`			
Prvok sa pridá na koniec zoznamu pomocou funkcie `add items to list`			
Na spracovanie zoznamu sa používajú cykly `for each number` a `for each item`			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť aplikáciu na spracovanie zoznamov využívajúcu: • **výber** prvkov zoznamu podľa ich **indexu**			
• **postupný výber** prvkov zoznamu			
• **pridávanie** a **odstraňovanie** prvkov zoznamu			
• funkciu `length of list`			
• komponent `ListView` na zobrazenie prvkov zoznamu			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.9 Zobrazovač aktuálnej polohy

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti

`LocationSensor`	`LocationChanged (latitude, longitude, altitude, speed)`
	
	`StatusChanged (status)`
		`LatitudeFromAddress`
		`LongitudeFromAddress`
			`ProviderName`
			`HasAccuracy`
			`Accuracy`
			`CurrentAddress`
			`TimeInterval`
			`DistanceInterval`
			`Enabled`

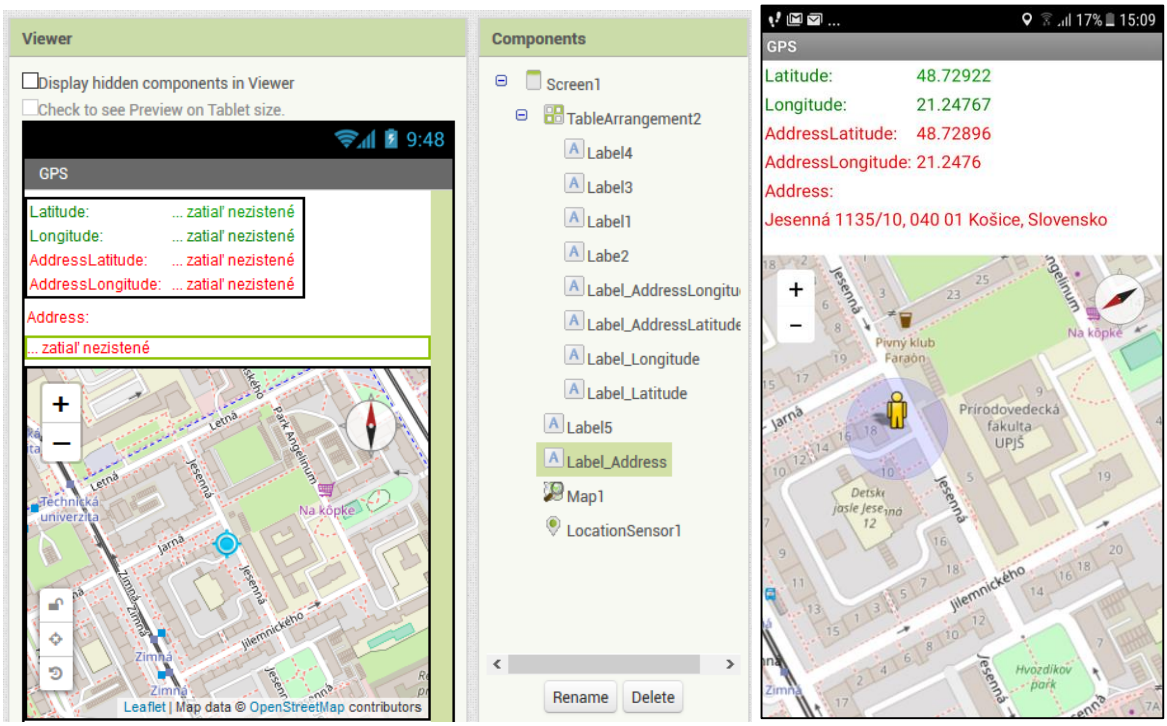
`Map`			
Jazykové konštrukcie		**Iné prvky jazyka**	

Úloha 1

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_9_gps.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

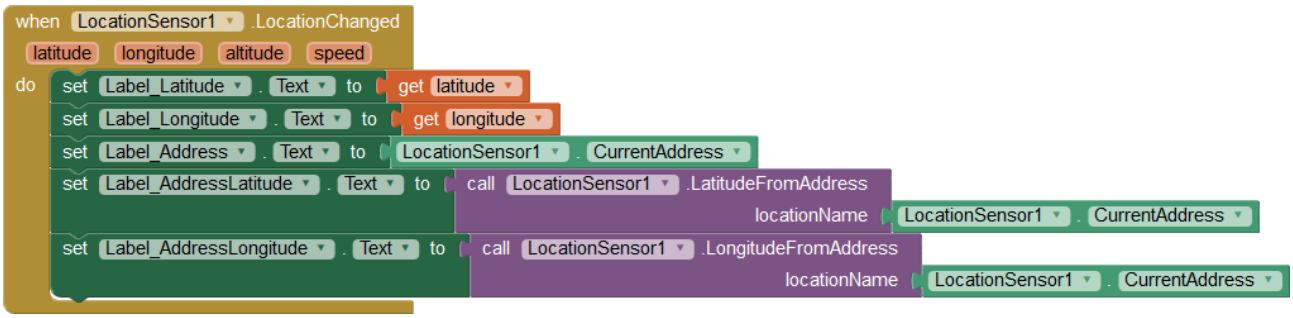
(Odporúčanie: V režime **Designer** v komponente `Map` nastavte svoju polohu zoomovaním a posúvaním mapy a tiež pomocou vlastnosti `ShowUser` a `CenterFromString`. Ak máte svoju polohu uprostred mapy, zafixujte toto zobrazenie v komponente `Map` stlačením tlačidla **Set initial map to current view**)

Používateľské rozhranie a kópia obrazovky bežiacej aplikácie:



Otázka	Odpoveď
a. V ktorej skupine komponentov je uvedený komponent `LocationSensor`? b. Na čo sa využíva komponent `LocationSensor`? c. V ktorej skupine komponentov je uvedený komponent `Map`? d. Na čo sa využíva komponent `Map`? e. Ako sa **líšia zemepisné súradnice** vášho obľúbeného miesta merané našou aplikáciou a inou aplikáciou? f. Ak predpokladáme, že Zem je guľa s dĺžkou poludníka 40000 km. Aký **dlhý** je **úsek poludníka** zodpovedajúci nasledovným uhlom?	a. b. c. d. e. **Naša apka:**zem. šírka zem. dĺžka **Iná apka:** zem. šírka zem. dĺžka f. **1° ≈** **0.001° ≈** **0.1°≈** **0.0001°≈** **0.01°≈** **0.00001°≈**

Zdrojový kód:



Otázka	Odpoveď
g. Akou aktivitou používateľa sa vyvolá udalosť `LocationSensor.LocationChanged`? h. Aký význam v tejto udalosti majú jej parametre `latitude` a `longitude`? i. Aký význam v komponente `LocationSensor` má vlastnosť `LocationSensor.CurrentAddress`? j. Čo vrátia metódy `LocationSensor.LatitudeFromAddress` a `LocationSensor.LongitudeFromAddress`?	g. h. `latitude` `longitude` i. j.

Úloha 2

Rozšírte aplikáciu **pmz_2_9_gps.aia**, aby zobrazovala aj ďalšie vlastnosti komponentu `LocationSensor`, napr. `ProviderName`, `HasAccuracy`, `Accuracy`. Skúmajte ako sa hodnoty týchto vlastnosti menia v závislosti od spôsobu lokalizácie (Vysoká presnosť, Šetrenie batérie, Iba telefón). Doplňte tiež nastavenie vlastností `DistanceInterval` (s hodnotami 0, 1, 10, 100) a `TimeInterval` (s hodnotami 0, 1000, 10000, 60000, 300000). Zabezpečte tiež zapínanie/vypínanie komponentu `LocationSensor` a zobrazovanie/ /nezobrazovanie časti údajov. Výsledný kód uložte do súboru **pmz_2_9_gps_R.aia**.

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.9 Zobrazovač aktuálnej polohy

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	rozumiem	čiastočne rozumiem	vôbec nerozumiem
Viditeľný komponent `Map` (uvedený v skupine **Maps**) sa používa na zobrazenie mapy vybraného miesta			
Nevizuálny komponent `LocationSensor` (uvedený v skupine **Sensors**) sa používa na určenie zemepisnej polohy zariadenia			
Zmena zemepisnej polohy zariadenia vyvolá udalosť `LocationSensor.LocationChanged`, ktorá má v parametroch `latitude` a `longitude` uloženú aktuálnu zemepisnú šírku a dĺžku			
Zapínanie a vypínanie komponentu `LocationSensor` umožňuje nastavenie vlastnosti `LocationSensor.Enabled`			
Pri zmene spôsobu lokalizácie a dostupnosti poskytovateľa GPS polohy sa vyvolá udalosť `LocationSensor.StatusChanged`			
Meno poskytovateľa GPS polohy (**gps**, **network**, **passive**) je uložené vo vlastnosti `LocationSensor.ProviderName`			
Presnosť určenia GPS polohy v metroch je uložená vo vlastnosti `LocationSensor.Accuracy`			
Ak má aktuálna pozícia zariadenia priradenú fyzickú adresu, tá bude uložená vo vlastnosti `LocationSensor.CurrentAddress`, inak je výsledkom správa “No address available”			
Zo zadanej `CurrentAddress` dostaneme zemepisnú pozíciu pomocou metód `LocationSensor.LatitudeFromAddress` a `LocationSensor.LongitudeFromAddress`			
Vyvolanie udalosti `LocationSensor.LocationChanged` určujú vlastnosti `LocationSensor.DistanceInterval` (prejdená vzdialenosť v metroch, napr. 5)			

a `LocationSensor.TimeInterval` (uplynutý čas v ms, napr. 10000)			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť aplikáciu zobrazujúcu aktuálnu GPS polohu zariadenia: • **bez** komponentu `Map`			
• **s** komponentom `Map`			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.10 Asistent aktuálnej polohy

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti

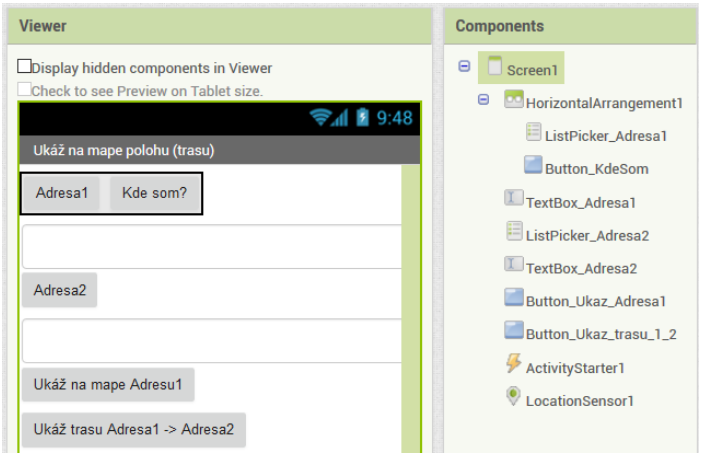
`ActivityStarter`		`StartActvity`	`Action`
			`DataUri`
			`Extras`

`ListPicker`	`BeforePicking` `AfterPicking`		`Elements` `Selection`
`TinyDB`		`GetTags` `ClearAll`	
`HorizontalArrangement`			`Visible`
Jazykové konštrukcie		**Iné prvky jazyka**	
		`pick a random item`	

Úloha 1

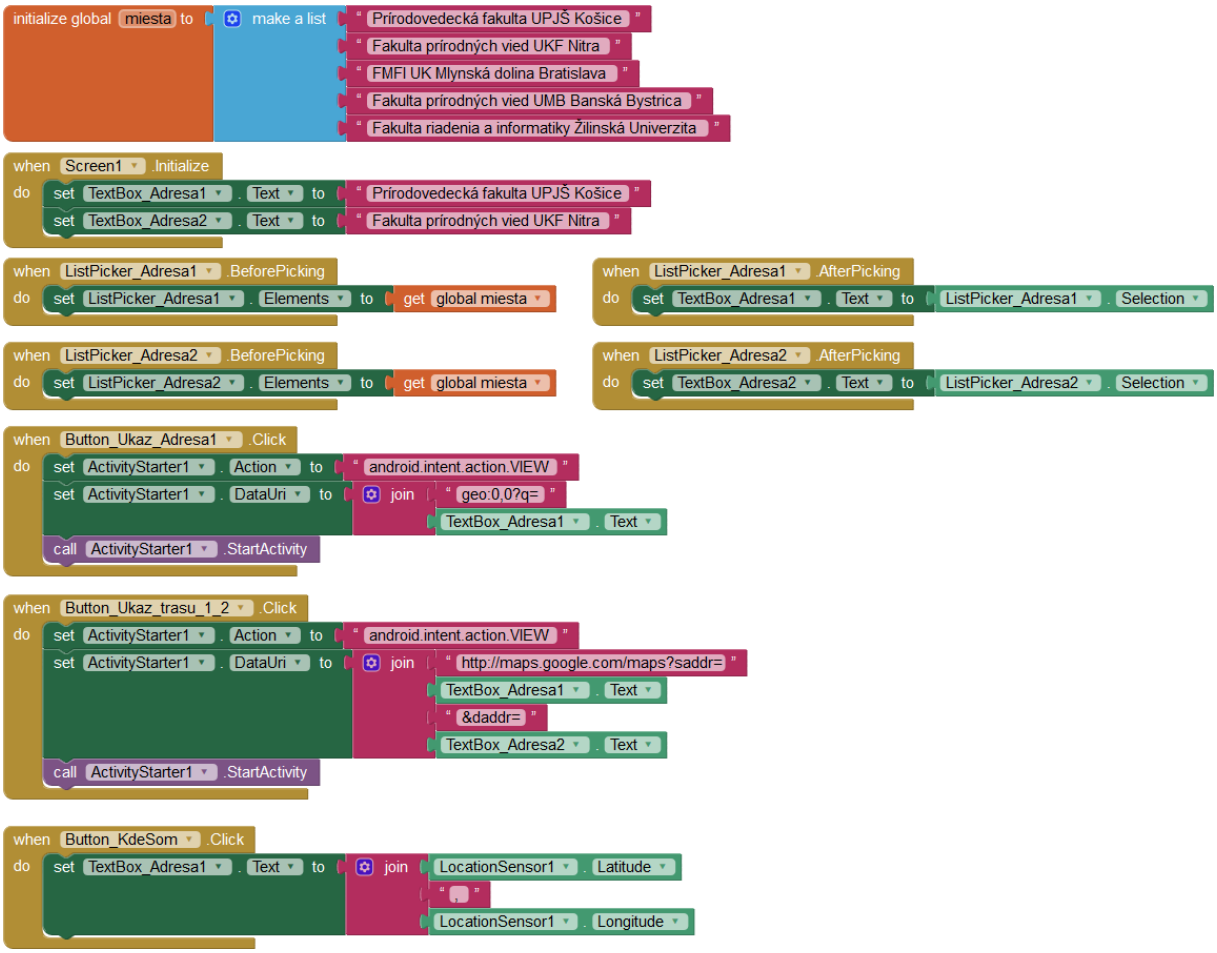
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_10_astarter_mapy.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

Používateľské rozhranie:



Otázka	Odpoveď
a. Do ktorej skupiny komponentov patrí komponent `ActivityStarter`?	a.
b. Spustenie ktorých **externých aplikácii** spôsobil komponent `ActivityStarter` po stlačení tlačidiel **Ukáž na mape ...** a **Ukáž trasu ...**?	b.
c. Do ktorej skupiny komponentov patrí komponent `ListPicker`?	c.
d. Ako sa správajú dva komponenty `ListPicker` po stlačení tlačidiel **Adresa1** a **Adresa2** vo vzťahu k hodnotám dvoch komponentov `TextBox`?	d.

Zdrojový kód:



Otázka	Odpoveď
e. Čím sa líšia udalosti `ListPicker.BeforePicking` a `ListPicker.AfterPicking`? f. Aký význam majú uvedené **vlastnosti** komponentu `ListPicker`? g. Ktoré **vlastnosti** komponentu `ActivityStarter` v tomto kóde sa nastavujú pred volaním metódy `ActivityStarter.StartActivity`? h. Pre zobrazenie miesta, resp. trasy na mape sa vo vlastnosti `ActivityStarter.DataUri` používa schéma **geo:0,0?q=*adresa1***, resp. **http://maps.google.com/maps?saddr=*adresa1*&daddr=*adresa2***. Čo by sa stalo, ak by sme v parametroch ***adresa1*** a ***adresa2*** zadali GPS súradnice namiesto textovej adresy?	e. f. `Elements` `Selection` g. h.

Úloha 2

Rozšírte aplikáciu **pmz_2_10_astarter_mapy.aia** o ďalšie funkcionality, napr.:

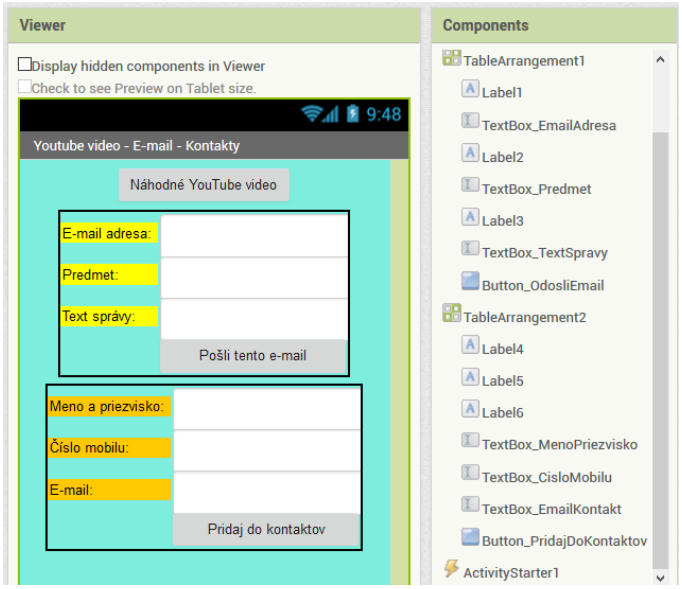
- zobrazenie textovej adresy aktuálnej polohy, ukončenie aplikácie,
- pridanie **Adresy1** na koniec zoznamu adries, zmazanie **Adresy1** zo zoznamu adries, zmazanie celého zoznamu,
- uloženie zoznamu adries do lokálnej databázy, zmazanie všetkých údajov z databázy. Výsledný kód uložte do súboru **pmz_2_10_astarter_mapy_R.aia**.

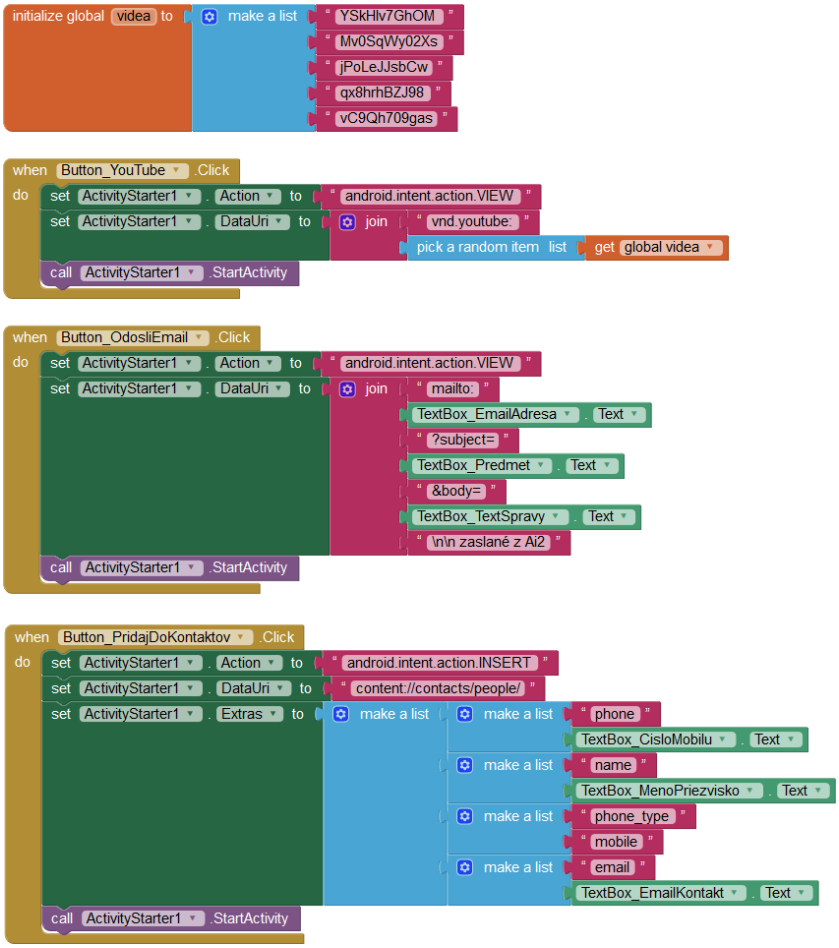
(Poznámka: Inšpirácie pre ďalšie rozšírenia tejto a návrhy ďalších aplikácii využívajúce komponent `ActivityStarter` nájdete na webe, napr. http://appinventor.mit.edu/explore/ai2/activity-starter.html)

Úloha 3

Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_10_astarter_rozne.aia** a preskúmajte používateľské rozhranie a správanie tejto aplikácie. Svoje zistenia zapíšte do voľných políčok tabuľky.

Používateľské rozhranie a zdrojový kód:





Otázka	Odpoveď
a. Ktoré **externé aplikácie** spúšťa táto aplikácia? b. Ktoré **spoločné** a ktoré **rozdielne** nastavenia majú jednotlivé spúšťače externých aplikácií?	a. b. spoločné rozdielne

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.10 Asistent aktuálnej polohy

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Nevizuálny komponent `ActivityStarter` (uvedený v skupine Connectivity) sa používa na spustenie externých aplikácií (napr. na zobrazenie mapy, spustenie videa, napísanie mailu, pridanie adresy do kontaktov)			
Pred samotným spustením externej aplikácie metódou `ActivityStarter.StartActivity` je potrebné nastaviť vlastnosti `ActivityStarter.Action` (napr. `VIEW`, `INSERT`) a `ActivityStarter.DataUri` na určenie typu (alebo konkrétnej) externej aplikácie a jej parametrov (napr. GPS poloha, video)			
Na výber prvku zo zoznamu sa používa komponent `ListPicker` (uvedený v skupine User Interface)			
Pred samotným výberom sa pomocou udalosti `ListPicker.BeforePicking` nastaví vlastnosť `ListPicker.Elements` na zoznam, z ktorého sa bude vyberať prvok			
Po výbere prvku sa vyvolá udalosť `ListPicker.AfterPicking`, ktorá vo vlastnosti `ListPicker.Selection` má uloženú hodnotu vybraného prvku a vo vlastnosti `ListPicker.SelectionIndex` index tohto prvku v zozname			

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s veľkou pomocou**
Vytvoriť aplikáciu využívajúcu komponent `ActivityStarter` na spustenie • **jednej** externej aplikácie s jedným parametrom			
• **viacerých** externých aplikácií **alebo** externej aplikácie s **viacerými** parametrami			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.11 Hlasovanie na internete

Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti

`FirebaseDB`	`GotValue (tag, value)`
	
	`DataChanged (tag, value)`
		`StoreValue`
		`GetValue`
			`FirebaseURL`

Jazykové konštrukcie		**Iné prvky jazyka**	
`list (index in list)`		`funkcia max`	

Úloha 1

Vytvorte aplikáciu **pmz_2_11_kliker.aia**, ktorá bude zaznamenávať a vypisovať kliknutia viacerých používateľov pripojených na internet (napr. učiteľov na výlete, ktorí spočítavajú svojich žiakov vo viacerých skupinách, aby vedeli chatárovi nahlásiť záujem o dané jedlo alebo pitie).

Poďme spoločne krok za krokom vyriešiť túto úlohu. Budeme postupovať nasledovne:

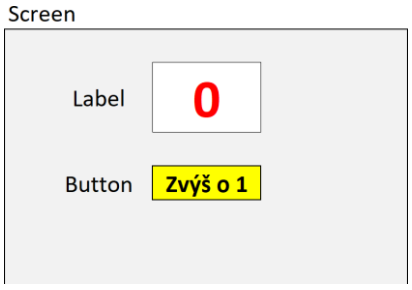
1. Navrhneme funkcionalitu a používateľské rozhranie aplikácie
2. Navrhneme štruktúru databázy
3. Na serveri **Firebase** pod svojim Google účtom zriadime vlastnú databázu s navrhnutou štruktúrou
4. Naprogramujeme aplikáciu s navrhnutými funkcionalitami

1 Návrh funkcionality a používateľského rozhrania aplikácie

Používateľské rozhranie bude obsahovať len dva vizuálne komponenty:

- tlačidlo na zvýšenie hodnoty spoločného počítadla o 1 (`Button`)
- popisok na výpis aktuálnej hodnoty spoločného počítadla (`Label`)

Na prácu s databázou použijeme nevizuálny komponent `FirebaseDB` (uvedený v skupine **Experimental**)



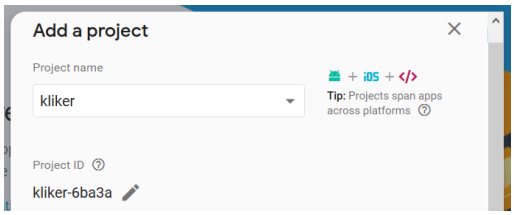
2 Návrh štruktúry databázy

Naša databáza bude obsahovať len jeden kľúč **pocet**, v ktorom bude uložená aktuálna hodnota počtu žiakov (na začiatku 0).

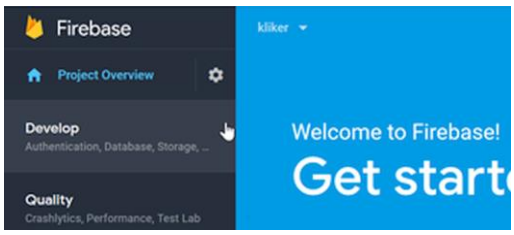
3 Zriadenie vlastnej databázy s navrhnutou štruktúrou na serveri Firebase

Navštívime webovú stránku https://console.firebase.google.com/ a prihlásime sa na ňu svojím Google účtom.

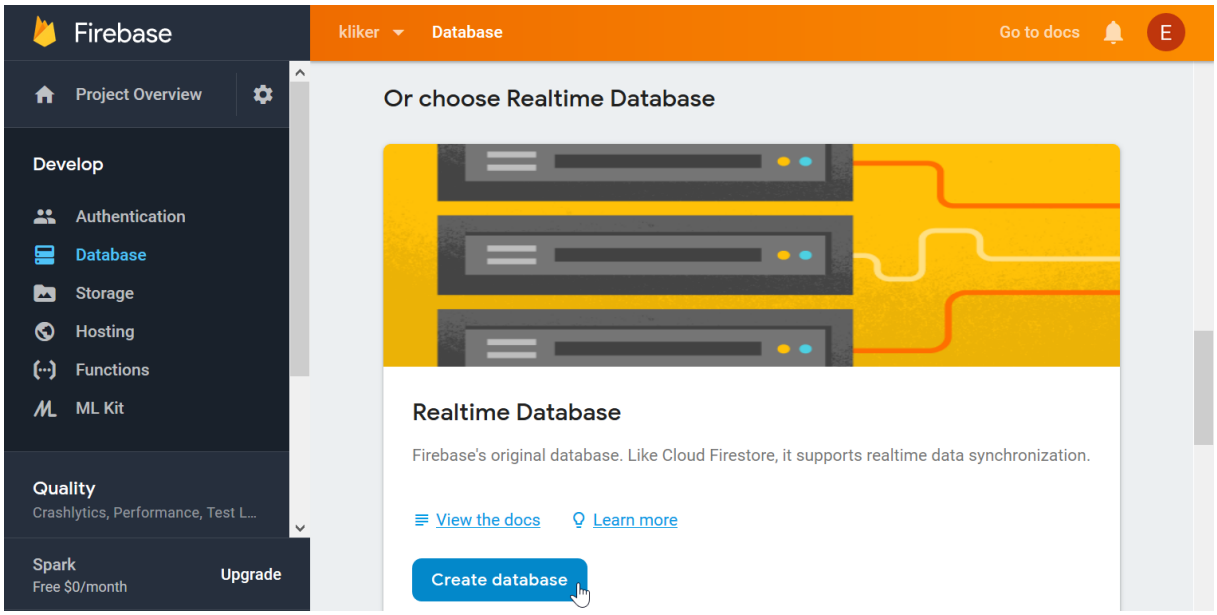
Stlačením tlačidla **Add project** na obrazovke vyvoláme okno, do ktorého uvedieme meno nášho projektu **kliker**. Nášmu projektu sa automaticky priradí **Project ID**, v našom prípade **kliker-6ba3a** (čo bude tiež menom našej vytváranej databázy).



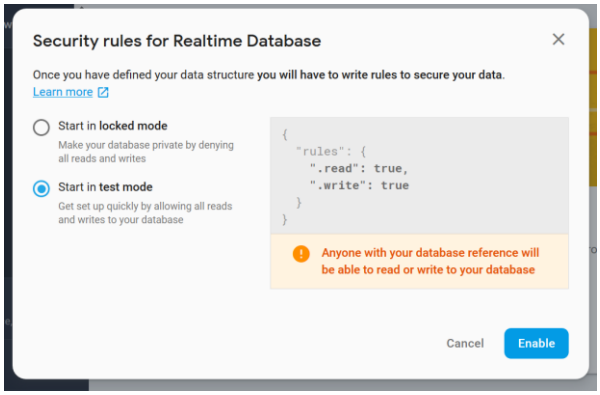
Po akceptovaní súhlasov bude v priebehu niekoľkých sekúnd vytvorený nový projekt, ktorý ďalej spravujeme.



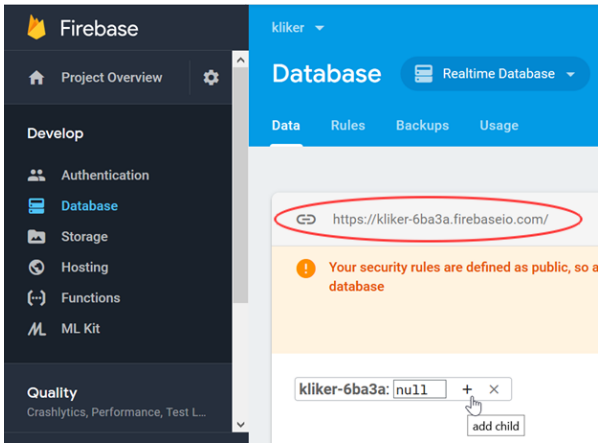
Teraz poďme v tomto projekte vytvoriť novú databázu. Rozbalíme ponuku **Develop** v ľavej časti okna a vyberieme podponuku **Database**. Môžeme si vybrať jednu z dvoch druhov databáz: **Cloud Firestore** alebo **Realtime Database**. Pre naše účely nám poslúži tradičná praxou overená **Realtime Database**, ktorú vyberieme tlačidlom v dolnej časti okna.



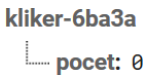
Počas vytvárania databázy sme vyzvaní, aby sme si vybrali jeden z dvoch bezpečnostných režimov – **zamknutý** (locked mode) alebo **testovací** (test mode). My si vyberieme **testovací režim**, aby sme umožnili používateľom aplikácie čítať a zapisovať do našej databázy.



Po odkliknutí tlačidla `Enable` sa objaví obrazovka, v strede ktorej je uvedené URL našej databázy (zvýraznené červenou elipsou). V dolnej časti môžeme do našej databázy **kliker-6ba3a** pridávať kľúče s hodnotami.



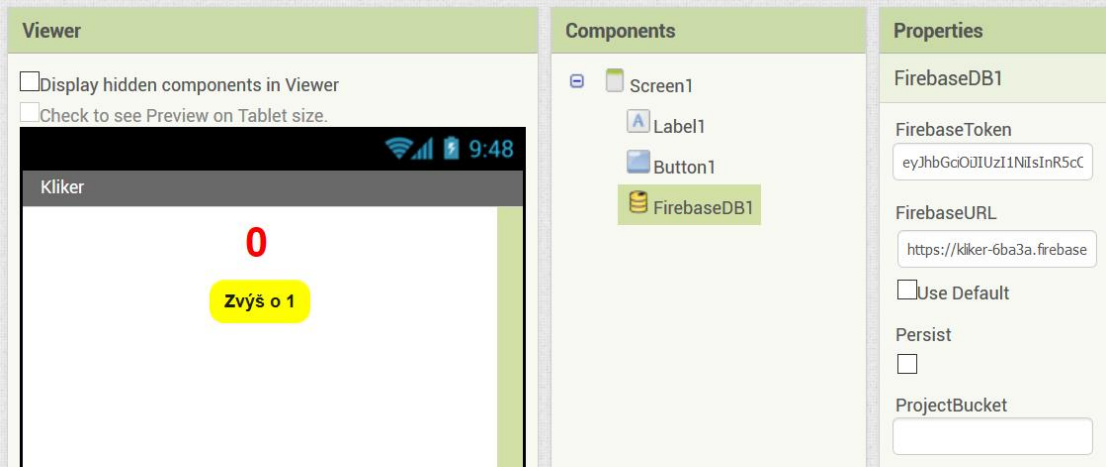
Pridaním kľúča **pocet** a jeho hodnoty **0** sme ukončili proces vytvárania štruktúry databázy a jej prvotného obsahu.



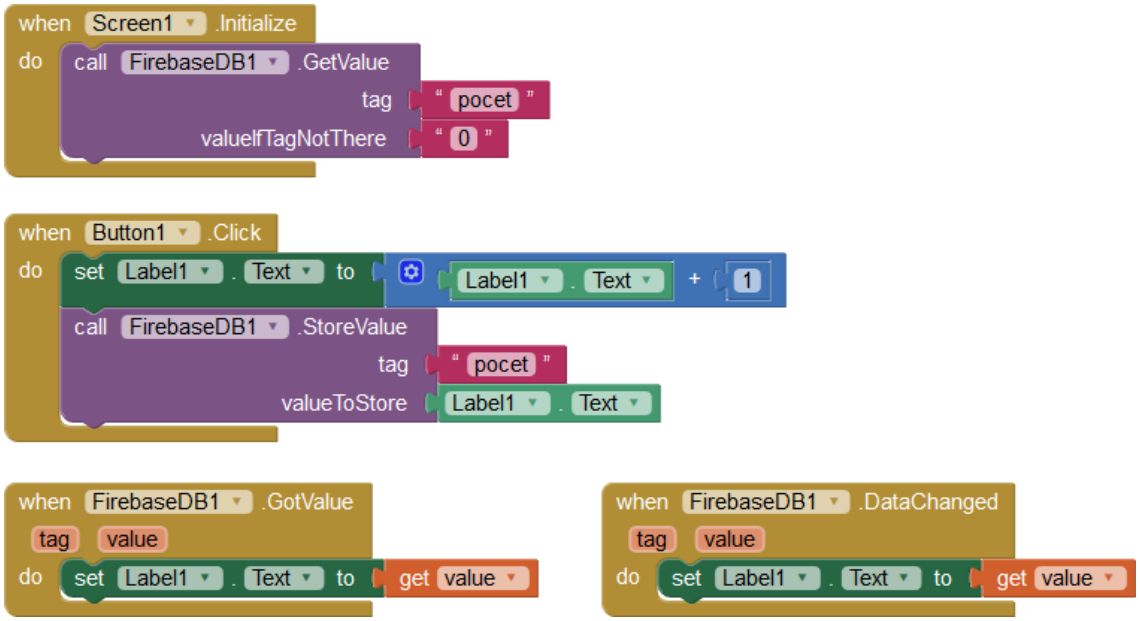
4 Naprogramovanie aplikácie s navrhnutými funkcionalitami

V režime **Designer** nastavíme komponent `FirebaseDB`:

- vlastnosť `FirebaseURL` z pôvodnej hodnoty **DEFAULT** na hodnotu **https://kliker-6ba3a.firebaseio.com/**
- odznačíme vlastnosť `Use Default`
- vlastnosť **project Bucket** z pôvodnej hodnoty **pmz_2_11_kliker** na prázdnu hodnotu.



V režime **Blocks** vytvoríme nasledovný zdrojový kód:



Po spustení aplikácie (vyvolaní udalosti `Screen.Initialize`) je zaslaná do databázy požiadavka na získanie hodnoty kľúča **pocet**. Ak databáza vie vrátiť hodnotu kľúča vyvolá sa udalosť `FirebaseDB.GotValue`, ktorá nastaví hodnotu popisku `Label1` na hodnotu získanú z databázy, ktorá je uložená v parametri `value` tejto udalosti.

Po stlačení tlačidla `Button1` sa zvýši hodnota popisku `Label1` o 1 a táto hodnota sa zapíše do databázy do kľúča **pocet**. Pri zmene kľúča **pocet** v databáze rôznymi používateľmi tejto aplikácie sa vyvolá udalosť `FirebaseDB.DataChanged`, ktorá nastaví hodnotu popisku `Label1` na hodnotu získanú z databázy, ktorá je uložená v parametri `value` tejto udalosti.

Otázka	Odpoveď
a. Prečo metóda `FirebaseDB.Getvalue` neposkytne programu priamo hodnotu daného kľúča ako to robí lokálna databáza `TinyDB`, ale tá hodnota sa získa až po vyvolaní udalosti `FirebaseDB.GotValue`?	a.

b. Čo majú spoločné a čo rozdielne udalosti `FirebaseDB.GotValue` a `FirebaseDB.DataChanged`?	b. spoločné rozdielne

Úloha 2

Vytvorte hlasovaciu aplikáciu pmz_2_11_hlasovanie.aia, ktorá umožni používateľom výber jednej z troch možností A, B, C. Bude ich tiež informovať, ktorá z možností hlasovania A, B, C je víťazom hlasovania.

(Odporúčanie: Pri spracovaní udalosti `FirebaseDB.GotValue` a `FirebaseDB.DataChanged` je potrebné pomocou podmienok rozlíšiť, ktorý kľúč (tag) sa načítal a ďalej spracovávať hodnotu (value) zodpovedajúcu tomuto kľúču. Pri výpočte víťaza hlasovania sa dajú použiť funkcie na spracovanie zoznamu `select list item`, `index in list`, `make a list` a matematickú funkciu `max`. Viac informácii o použití databázy Firebase pri programovaní v App Inventore nájdete na stránkach MIT, Experimental Components – App Inventor for Android:

http://ai2.appinventor.mit.edu/reference/components/experimental.html#FirebaseDB)

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.11 Hlasovanie na internete

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Pre uchovanie údajov prístupných pre viacerých používateľov sa používa webová databáza – komponent `FirebaseDB` uvedený v skupine **Experimental**			
Pre vytvorenie vlastnej webovej databázy sa dá použiť server `FirebaseDB`, na ktorom sa pridá nový projekt a v ňom nová databáza, napr. typu **RealTime Database**			
Pri vytváraní databázy treba nastaviť **testovací mód**, aby bola **prístupná** na čítanie aj zápis pre ostatných používateľov			
Poslednou fázou tvorby databázy je **pridávanie kľúčov** (tags) a **nastavovanie ich hodnôt** (values)			
Pri tvorbe používateľského rozhrania treba komponentu `FirebaseDB`: 1. nastaviť vlastnosť `FirebaseURL` na hodnotu webovej adresy uvedenej na serveri Firebase v našom projekte 2. odškrtnúť vlastnosť `Use Default` 3. nastaviť prázdnu hodnotu na vlastnosť `ProjectBucket`			
Pre načítanie a zapísanie hodnoty daného kľúča do webovej databázy používame metódy `FirebaseDB.GetValue` a `FirebaseDB.StoreValue`			
Po spustení metódy `FirebaseDB.GetValue` sa vyvolá udalosť `FirebaseDB.GotValue`, ktorá v parametri `value` má uloženú hodnotu daného kľúča			
Pre aktualizáciu údajov v aplikácii sa využíva udalosť `FirebaseDB.Datachanged`, ktorá sa vyvoláva pri každej zmene údajov databázy			

Z uvedeného učiva viem vykonať nasledovné činnosti:	samostatne	s malou pomocou	s veľkou pomocou
Vytvoriť na **Firebase** serveri **nový projekt** a **webovú databázu** s požadovanou štruktúrou a počiatočnými hodnotami			
Naprogramovať aplikáciu využívajúcu **webovú databázu** na **Firebase** serveri			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:


zaujímavé


normálne


nudné


ľahké


primerané


ťažké

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

2.12 Komunikačný asistent

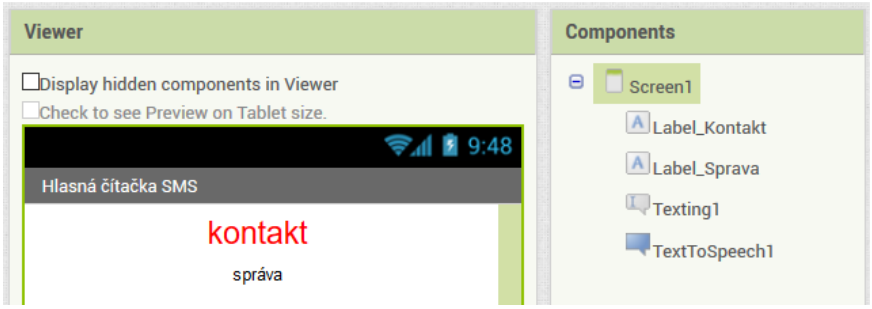
Prvky nového učiva

Komponenty	Udalosti (premenné)	Metódy	Vlastnosti
`Texting`	`MessageReceived (number, messageText)`	`SendMessage`	`PhoneNumber Message ReceivingEnabled`
`PhoneCall`		`MakePhoneCall`	`PhoneNumber`
Jazykové konštrukcie		**Iné prvky jazyka**	

Úloha 1

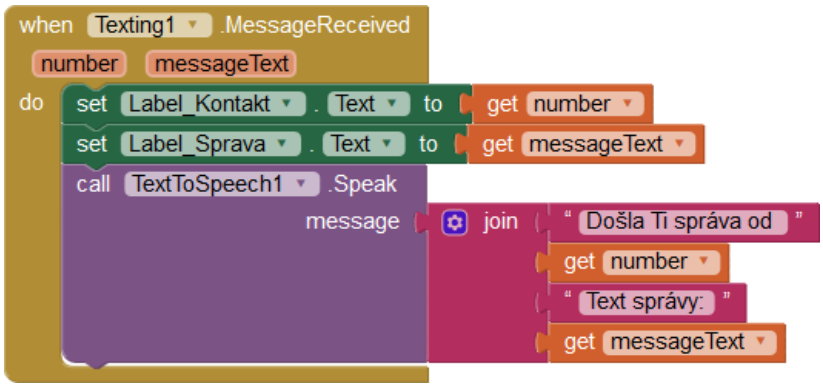
Do svojho Ai2 účtu importujte zdrojový kód aplikácie uložený v súbore **pmz_2_12_sms_nahlas.aia**. Po nainštalovaní a spustení aplikácie na MZ preskúmajte jej správanie. Svoje zistenia zapíšte do voľných políčok tabuliek.

Používateľské rozhranie:



Otázka	Odpoveď
a. V ktorej skupine komponentov sa nachádza komponent `Texting`? b. **Na čo slúži** komponent `Texting`? c. **Na ktorých** MZ sa dá použiť komponent `Texting`?	a. b. c.

Zdrojový kód:



Otázka	Odpoveď
d. Kedy sa vyvolá udalosť `Texting.MessageReceived`? e. Čo predstavujú parametre `number` a `messageText` tejto udalosti? f. **Čo robí** daný program a na čo by dal **použiť**?	d. e. f.

Úloha 2

Vytvorte aplikáciu **pmz_2_12_sms_telefon.aia**, ktorá:

- prečíta syntetickou rečou práve prijatú SMS správu,
- po zatrasení smartfónom sa vytočí telefónne spojenie s číslom uvedeným v došlej SMS správe,
- po priblížení ruky k hornej hrane smartfónu sa spustí analyzátor našej reči, ktorá sa opätovne spustí syntezátorom reči a pošle sa ako SMS správa pôvodnému odosielateľovi došlej SMS správy,
- umožní prepínanie režimu prijímania správ 2 (**Foreground**) a 3 (**Always**).

(Odporúčanie: Na prepínanie režimov prijímania SMS správ použite vlastnosť `Texting.ReceivingEnabled`. Pri odosielaní SMS správy nastavte vlastnosti `Texting.PhoneNumber` a `Texting.Message` a spustite metódu `Texting.SendMessage`. Pred vytočením hovoru metódou `PhoneCall.MakePhoneCall` nastavte vlastnosť `PhoneCall.PhoneNumber`)

Zamyslime sa, čo sme sa naučili

Sebahodnotiaca karta – Programujeme malú aplikáciu 2.12 Komunikačný asistent

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva nasledovným pojmom / princípom / tvrdeniam:	**rozumiem**	**čiastočne rozumiem**	**vôbec nerozumiem**
Nevizuálny komponent `Texting` je uvedený v skupine **Social** a slúži na posielanie a prijímanie **SMS správ**			
Nevizuálny komponent `PhoneCall` je uvedený v skupine **Social** a slúži na vytáčanie a prijímanie **telefónnych hovorov**			
Aplikácie využívajúce komponenty `Texting` a `PhoneCall` fungujú na **smartfónoch** a **iných MZ** schopných prijímať a posielať SMS a telefónne hovory			
Prijatím SMS správy sa vyvolá udalosť `Texting.MessageReceived` a v jej parametroch `number` a `messageText` sú uložené telefónne číslo odosielateľa a text prijatej SMS správy			
Na zaslanie SMS správy slúži metóda `Texting.SendMessage`, s nastavenými vlastnosťami `Texting.PhoneNumber` a `Texting.Message`			
Na prepínanie režimov prijímania SMS správ použite vlastnosť `Texting.ReceivingEnabled`, v režime **2** (**Foreground**) sa dajú prijímať správy, len keď je spustená aplikácia a v režime **3** (**Always**) aj keď aplikácia nebeží na popredí			
Na vytočenie telefonického hovoru slúži metóda `PhoneCall.MakePhoneCall` s nastavenou vlastnosťou `PhoneCall.PhoneNumber`			

Z uvedeného učiva viem vykonať nasledovné činnosti:	samostatne	s malou pomocou	s veľkou pomocou
Vytvoriť komunikačnú aplikáciu využívajúcu: • komponent `Texting`			
• komponent `PhoneCall`			
• komponent `TextToSpeech` alebo `SpeechRecognizer`			
• komponent `AccelerometerSensor` alebo `ProximitySensor`			

Aké boli pre vás tieto úlohy? Zafarbite/zakrúžkujte niektorú z uvedených možností:

zaujímavé *normálne* *nudné*

ľahké *primerané* *ťažké*

Uveďte, čo by ste ešte doplnili do tejto aplikácie, aby bola viac zaujímavá a využiteľná v praxi:

3 Multimédiá

V kapitolách 3 až 6 predstavujeme 12 náročnejších a komplexnejších projektov využiteľných v každodennej praxi. Tieto projekty nadväzujú na malé aplikácie uvedené v kapitole 2, ktoré pokrývajú vybrané funkcionality Ai2. Na druhej strane nám prinášajú aj nové poznatky nad rámec obsahu kapitoly 2 a dávajú možnosť samostatne či v tímoch vytvárať komplexnejšie aplikácie. Projekty v kapitole 3 sú zamerané prevažne na využívanie multimédií, v kapitole 4 na siete, v kapitole 5 na geolokáciu a v kapitole 6 na využitie rôznych senzorov a aktuátorov.

Pri spracovaní kapitol sme použili dva prístupy, ktoré by mali obohatiť žiakov aj ich učiteľov. Vo väčšine podkapitol sú projekty spracované ako postupnosť čiastkových úloh, ktoré sa postupne riešia spoločne s občasnou pomocou a vysvetleniami učiteľa. Niektoré podkapitoly (3.1, 4.2, 6.1) sú spracované pre projektové vyučovanie, v rámci ktorého sú žiaci viac samostatní v návrhu aj implementácii aplikácie. Pri takejto výučbe je učiteľ moderátorom úvodnej diskusie a brainstormingu k možným funkcionalitám aplikácie. Ďalej počas ich implementácie je konzultantom pre jednotlivé žiacke projekty, ktoré môžu mať navzájom rôzne funkcionality. A napokon je moderátorom záverečnej diskusie k prezentácii projektov. Žiakom sú podľa potreby ukázané časti vzorového riešenia projektov.

V podkapitolách 3.1 až 3.3 vytvoríme tri praktické aplikácie užitočné pre mladých reportérov, športovcov či pre budúcich záchranárov ľudských životov. Spoločným znakom týchto aplikácií je využitie rôznych multimediálnych funkcionalít mobilného zariadenia.

V tejto kapitole sa zameriame na vývoj 3 aplikácií využívajúcich multimédiá:

3.1 **Multimediálny zápisník pre mladého reportéra**
Aplikácia na záznam zvukov a fotografií zo vstavaného mikrofónu a fotoaparátu. Ďalšími možnosťami aplikácie sú dokreslenie a doplnenie textu do získanej fotografie, uloženie a načítanie upravenej fotografie. V aplikácii sú použité multimediálne komponenty: `Camera`, `ImagePicker`, `Player`, `SoundRecorder`.

3.2 **Dychový tréner**
Aplikácia na manažovanie dychového tréningu pre potápačov. Cyklicky sa striedajú fázy zadržania dychu a predýchania. Aplikácia odpočítava čas, graficky znázorňuje fázu tréningu (zadržanie dychu alebo predýchavanie), graficky a zvukovo signalizuje koniec fázy, umožňuje rôzne nastavenia parametrov tréningu. V aplikácii sú použité multimediálne komponenty: `TextToSpeech`, `Sound`, `Ball`.

3.3 **Prvá pomoc**
Aplikácia použiteľná pri záchrane pomoci. Umožňuje používateľovi prečítať si návody prvej pomoci, resp. si ich vypočuť syntetickou rečou. Navyše asistuje pri resuscitácii a pri vytočení tiesňovej telefónnej linky. V aplikácii sú použité multimediálne komponenty: `VideoPlayer`, `TextToSpeech`, `Sound`.

3.1 Multimediálny zápisník pre mladého reportéra

Kľúčové slová

multimédiá, komponent Camera, komponent ImagePicker, komponent Player, komponent SoundRecorder, responzívny dizajn, súbory, dekompozícia problému

Čo sa naučíme a čo si precvičíme

- Samostatne navrhnúť požadované funkcionality pre aplikáciu multimediálny zápisník.
- Vytvoriť aplikáciu s responzívnym dizajnomvyužívajúcu viaceré multimediálne komponenty (`Camera`, `ImagePicker`, `Player`, `SoundRecorder`).
- Vytvárať a používať grafické a zvukové súbory s časovou značkou a ukladať ich do vhodných priečinkov na MZ.
- Získať skúsenosti s tvorbou komplexnejších projektov vyžadujúcich dekompozíciu a hlbšiu analýzu čiastkových podproblémov.

Čo zaujímavé môžeme zistiť (o zaznamenávaní multimédií na mobilné zariadenie)?

Predstavme si situáciu, že chceme pre seba alebo pre našich priateľov vytvoriť aplikáciu, ktorá by nám pomohla zozbierať autentické multimediálne informácie s našim komentárom priamo z terénu, napr. školského výletu. Takto nazbierané informácie sa budú dať použiť pre vytvorenie, napr. reportáže do školského časopisu.

Otázky na zamyslenie

Prediskutujme nasledovné otázky:

- Z ktorých ďalších podujatí má zmysel zaznamenávať multimediálne informácie?
- Pre aké účely vieme využiť zaznamenané multimediálne informácie?
- Môžeme robiť multimediálne záznamy osôb bez ich súhlasu?
- Aké typy multimediálnych informácií vieme získavať pomocou MZ?
- Ktoré aplikácie na MZ využívate pri zázname multimédií?
- Aký zmysel má vytvárať vlastnú aplikáciu na záznam multimédií?

Akú zaujímavú aplikáciu môžeme vytvoriť?

Ak sme sa rozhodli pre tvorbu vlastnej aplikácie, mali by sme v triede formou brainstormingu zozbierať nápady k možným funkcionalitám multimediálneho zápisníka.

Ako budeme postupovať pri tvorbe aplikácie?

Pri tvorbe vlastného projektu môžeme postupovať podľa nasledovných krokov:

1. Spresnenie špecifikácie navrhovanej aplikácie
2. Návrh používateľského rozhrania aplikácie, zoznam komponentov a multimediálnych súborov
3. Návrh správania aplikácie
4. Tvorba používateľského rozhrania a programového kódu aplikácie
5. Prezentácia vlastnej aplikácie a diskusia využitiu aplikácie v praxi a jej prípadnému doladeniu
6. Doladenie aplikácie a jej publikovanie v rámci portfólia žiaka

1. Špecifikácia aplikácie

Multimediálny zápisník (verzia 1) má nasledovné funkcionality:

f 1. Kreslenie na plátno dotykom
f 2. Kreslenie na plátno ťahaním
f 3. Zmazanie plátna zatrasením
f 4. Nastavenie farby kresliaceho pera a farby pozadia obrazovky
f 5. Tlačidlo s ukončením aplikácie
f 6. Nastavenie prázdneho (bieleho) pozadia plátna
f 7. Uloženie obrázku na plátne do súboru
f 8. Načítanie obrázku pozadia z niektorého z grafických súborov
f 9. Načítanie uloženého obrázku pozadia pri spustení aplikácie
f 10. Spustenie fotoaparátu a nastavenie zosnímanej fotografie na pozadie plátna

Rozšírená verzia multimediálneho zápisníka (verzia 2) má navyše od verzie 1 doplnené funkcionality:

f 11. Responzívny dizajn aplikácie
f 12. Napísanie komentára do plátna spolu s dátumovou a časovou pečiatkou a GPS polohou
f 13. Uloženie obrázkového súboru s dátumovou a časovou značkou
f 14. Nahratie a prehratie zvukových komentárov, uloženie zvukového súboru s dátumovou a časovou značkou

2. Návrh používateľského rozhrania aplikácie, zoznam komponentov a multimediálnych súborov

Používateľské rozhranie (verzia 1)

V hornej časti aplikácie je skupina tlačidiel uložená vo vodorovnom kontajneri, v dolnej časti je plátno.

Zoznam komponentov (verzia 1)

Vizuálne komponenty:

- `HorizontalArrangement` (prípadne `HorizontalScrollArrangement`)
  - `Button_Black`, `Button_Blue`, `Button_Green`, `Button_Yellow`, `Button_Red`, `Button_White` – tlačidlá na zmenu farby pera kresliaceho po plátne a zmenu farby pozadia obrazovky (f4)
  - `Button_Photo` – tlačidlo na spustenie fotoaparátu a nastavenie zosnímanej fotografie na pozadie plátna (f10)
  - `Button_New` – tlačidlo na nastavenie prázdneho (bieleho) pozadia plátna (f6)
  - `Button_Save` – tlačidlo na uloženie obrázku na plátne do súboru (f7)
  - `ImagePicker` – tlačidlo s výberom súboru s obrázkom (f8)
  - `Button_Exit` – tlačidlo na ukončenie aplikácie (f5)
  - `Label1`, `Label2`, `Label3` – prázdne popisky na oddelenie tlačidiel medzi sebou
- `Canvas` – plátno na kreslenie (f1, f2) s možnosťou nastaviť obrázok do jeho pozadia

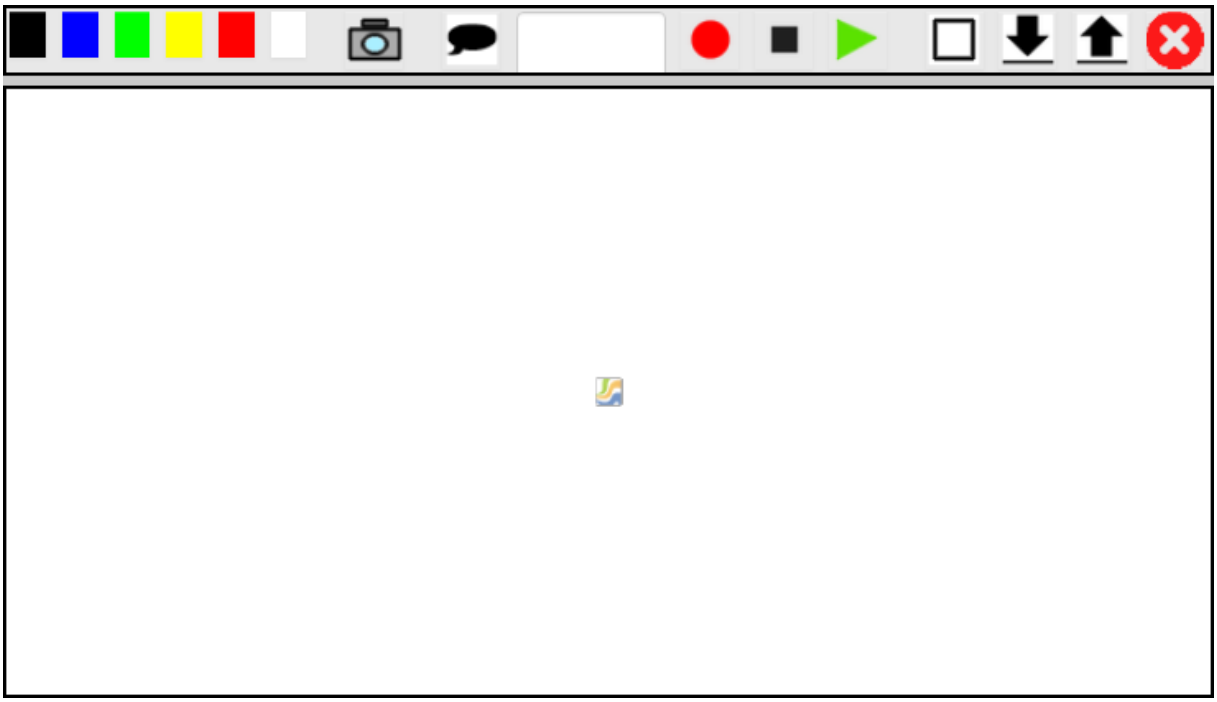
Nevizuálne komponenty:

- `AccelerometerSensor` – na zmazanie plátna zatrasením (f3)
- `Camera` – na spustenie fotoaparátu a nastavenie zosnímanej fotografie na pozadie plátna (f10)

Zoznam multimediálnych súborov (verzia 1)

- **crayon_icon.png** – ikona aplikácie
- **photo.png**, **new.png**, **save.png**, **load.png**, **exit.png** – obrázky tlačidiel Photo, New, Save, Load, Exit

Používateľské rozhranie (verzia 2)



Oproti verzii 1 sú v strednej časti skupiny tlačidiel doplnené: tlačidlo a textové pole pre tvorbu komentára (f12) a tri tlačidlá na nahrávanie, prehrávanie a ukladanie zvukových komentárov (f13). Vzhľadom na veľký počet tlačidiel je vodorovný kontajner rolovateľný.

Zoznam komponentov (doplnených vo verzii 2)

Vizuálne komponenty:

- `HorizontalScrollArrangement`
  - `Button_Remark` – tlačidlo na vypísanie uvedeného komentára na plátno (f12)
  - `TextBox_Remark` – textové pole na napísanie komentára (f12)
  - `Button_Start` – tlačidlo na spustenie nahrávania zvukového komentáru (f14)
  - `Button_Stop` – tlačidlo na ukončenie nahrávania zvukového komentáru (f14)
  - `Button_Play` – tlačidlo na prehratie nahratého zvukového komentáru (f14)
  - `Label4` – prázdny popisok na oddelenie tlačidiel medzi sebou

Nevizuálne komponenty:

- `Clock` – na vytvorenie názvov obrázkových aj zvukových súbor s dátumovou a časovou pečiatkou (f13, f14)
- `LocationSensor` – na získanie aktuálnej GPS polohy uvedenej v textovom komentári na plátne (f12)
- `SoundRecorder` – na záznam zvuku (f14)
- `Player` – na prehratie zaznamenaného zvuku (f14)

Zoznam multimediálnych súborov (doplnených vo verzii 2)

- **remark.png**, **record.png**, **stop.png**, **play.png** – obrázky tlačidiel Remark, Start, Stop, Play

3. Návrh správania aplikácie

Verzia 1

Komponent	Udalosť	Akcia
`Canvas`	`Touched`	(f1) Kreslenie bodov na plátne (`Canvas.DrawCircle`)
`Canvas`	`Dragged`	(f2) Kreslenie úsečiek na plátne (`Canvas.DrawLine`)
`AccelerometerSensor`	`Shaking`	(f3) Zmazanie plátna (`Canvas.Clear`)
`Button_Black`	`Click`	(f4) Nastavenie farby kresliaceho pera a farby pozadia obrazovky na čiernu (`Canvas.PaintColor`, `Screen.BackgroundColor`)
`Button_Blue`	`Click`	(f4) Nastavenie farby kresliaceho pera a farby pozadia obrazovky na modrú (`Canvas.PaintColor`, `Screen.BackgroundColor`)
`Button_Green`	`Click`	(f4) Nastavenie farby kresliaceho pera a farby pozadia obrazovky na zelenú (`Canvas.PaintColor`, `Screen.BackgroundColor`)
`Button_Yellow`	`Click`	(f4) Nastavenie farby kresliaceho pera a farby pozadia obrazovky na žltú (`Canvas.PaintColor`, `Screen.BackgroundColor`)
`Button_Red`	`Click`	(f4) Nastavenie farby kresliaceho pera a farby pozadia obrazovky na červenú (`Canvas.PaintColor`, `Screen.BackgroundColor`)
`Button_White`	`Click`	(f4) Nastavenie farby kresliaceho pera a farby pozadia obrazovky na bielu (`Canvas.PaintColor`, `Screen.BackgroundColor`)
`Button_Photo`	`Click`	(f10) Spustenie fotoaparátu (`Camera.TakePicture`)
`Camera`	`AfterPicture`	(f10) Nastavenie zosnímanej fotografie na pozadie plátna (`Canvas.BackgroundImage`)
`Button_New`	`Click`	(f6) Nastavenie prázdneho (bieleho) pozadia plátna (`Canvas.BackgroundImage`)
`Button_Save`	`Click`	(f7) Uloženie obrázku na plátne do súboru (`Canvas.Save`)

ImagePicker	AfterPicking	(f8) Načítanie obrázku pozadia z niektorého z grafických súborov (Canvas.BackgroundImage)
Button_Exit	Click	(f5) Ukončenie aplikácie (close application)
Screen	Click	(f9) Počiatočné nastavenie súboru s pozadím aplikácie (Canvas.BackgroundImage, Screen.BackgroundColor)

Verzia 2

Komponent	Udalosť	Akcia
Button_Remark	Click	(f12) Vypísanie komentára uvedeného v TextBox_Remark na plátno (Canvas.DrawText, LocationSensor.Latitude/ Longitude, Clock.Now)
Button_Save	Click	+ (f13) Uloženie obrázkového súboru s dátumovou a časovou značkou (Canvas.SaveAs, Clock.Now)
Button_Start	Click	(f14) Spustenie nahrávania zvukového komentáru (SoundRecorder.Start) a uloženie zvukového súboru s dátumovou a časovou značkou (Clock.Now, SoundRecorder.SavedRecording)
SoundRecorder	StartedRecording	(f14) Nastavenie viditeľnosti tlačidla Button_Stop
Button_Stop	Click	(f14) Ukončenie nahrávania zvukového komentáru (SoundRecorder.Stop)
SoundRecorder	StoppedRecording	(f14) Nastavenie viditeľnosti tlačidiel Button_Start, Button_Play
Button_Play	Click	(f14) Prehratie nahratého zvukového komentáru (Player.Start) Zrušenie viditeľnosti tlačidiel Button_Start, Button_Play
Player	Completed	(f14) Nastavenie viditeľnosti tlačidiel Button_Start, Button_Play
Screen	Initialize	+ (f14) Nastavenie viditeľnosti tlačidla Button_Start, a (f12) hodnoty textového poľa TextBox_Remark (f11) Nastavenie veľkosti plátna s pomerom 16:9 podľa veľkosti obrazovky zariadenia
		(f14) Inicializácia globálnej textovej premennej **zvuk_subor**

4. Tvorba používateľského rozhrania a programového kódu aplikácie

Pri tvorbe používateľského rozhrania aplikácie použijeme návrh grafického používateľského rozhrania obsahujúci vizuálne komponenty (`Screen`, `Canvas`, `Button`, `Label`, `HorizontalScrollArrangement`, `ImagePicker`), nevizuálne komponenty (`AccelerometerSensor`, `Camera`, `SoundRecorder`, `Player`) a multimediálne súbory (ikona a obrázky tlačidiel).

Programový kód vytvárame po jednotlivých funkcionalitách, ktorých riešenia uvedieme a okomentujeme po skupinách:

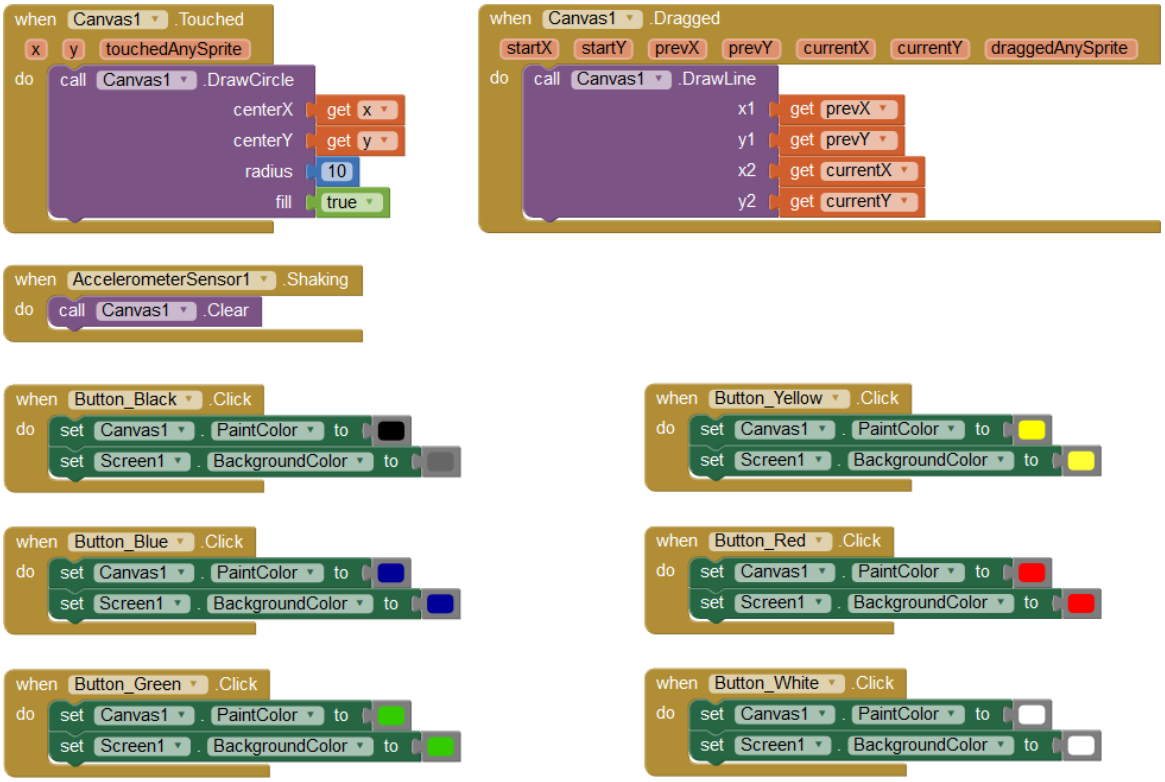
Verzia 1

- Kreslenie na plátne, mazanie plátna, zmena farby pera a farby pozadia obrazovky
- Nastavenie pozadia plátna (z fotoaparátu, žiadne pozadie), ukončenie aplikácie
- Uloženie obrázka do súboru, načítanie obrázka zo súboru, inicializácia aplikácie

Verzia 2

- Responzívny dizajn aplikácie
- Zápis komentára do plátna s údajmi o aktuálnom dátume, čase a polohe
- Uloženie obrázkového súboru s dátumovou a časovou značkou
- Nahratie, prehratie a uloženie zvukového komentára do súboru s dátumovou a časovou značkou

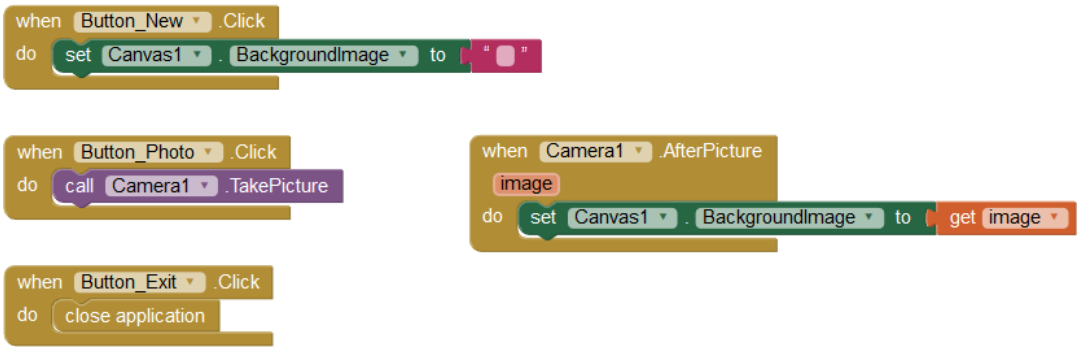
Kreslenie na plátne dotykom (f1) a ťahaním (f2), zmazanie plátna zatrasením (f3) a nastavenie farby pera (f4) sme vyriešili v malej aplikácii 2.1. V našom projekte nastavujeme farbu kresliaceho pera a farby pozadia obrazovky (ako indikácie vybranej farby kresliaceho pera) šiestimi tlačidlami:



Vyčistenie plátna (f6) dosiahneme nastavením vlastnosti `Canvas.BackgroundImage` na prázdny reťazec.

Nastavenie pozadia plátna na autentickú fotografiu získanú zo vstavaného fotoaparátu (f10) dosiahneme spustením metódy `Camera.TakePicture`. Táto metóda spustí aplikáciu fotoaparát a po uložení fotografie spustí udalosť `Camera.AfterPicture`, ktorá má v parametri image uložený práve zosnímaný obrázok. Tento vieme nastaviť ako hodnotu vlastnosti `Canvas.BackgroundImage`.

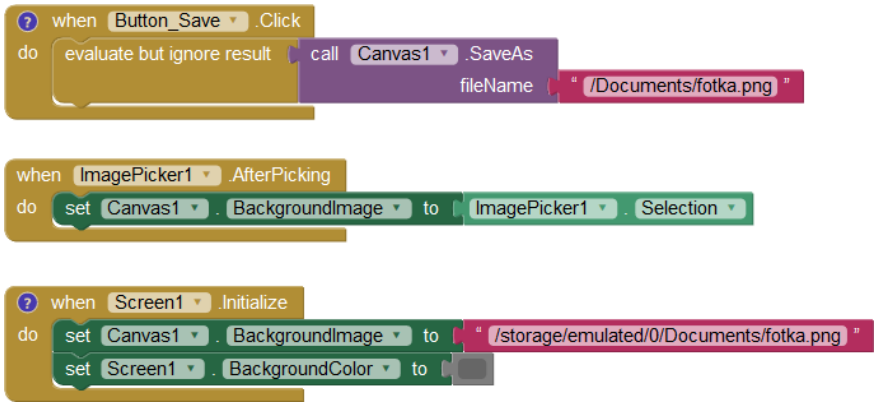
Ukončenie behu aplikácie (f5) dosiahneme príkazom `close application`, čo už bolo uvedené v malej aplikácii 2.2 Hra Postreh.



Obrázok zobrazený na plátne vieme uložiť do súboru (f7) pomocou metódy `Canvas.SaveAs`. Tu je dôležité premyslieť, do ktorého priečinku na MZ uložíme tento obrázok. Ak nechceme vytvárať nový priečinok, môžeme využiť existujúci priečinok, napr. priečinok **/storage/emulated/0/Documents** (alebo **/storage/emulated/0/Pictures**), do ktorého môžeme uložiť svoje súbory s obrázkami, zvukmi atď. V našom prípade v metóde `Canvas.SaveAs` nastavíme parameter `fileName` na hodnotu **/Documents/fotka.png**.

Ak chceme do pozadia plátna nastaviť nejaký iný obrázok uložený v MZ (f8), môžeme použiť udalosť `ImagePicker.AfterPicking`. Po výbere bude obsah tohto grafického súboru uložený vo vlastnosti `ImagePicker.Selection`, čo môžeme nastaviť ako hodnotu vlastnosti `Canvas.BackgroundImage`.

Pri spustení aplikácie nastavíme pozadie plátna (f9) na už predtým uložený obrázok v súbore **/storage/emulated/0/Documents/fotka.png** tak, že v udalosti `Screen.Initialize` nastavíme vlastnosť `Canvas.BackGroundImage` na absolútnu cestu na obrázkový súbor. Je zaujímavé, že pri uložení obrázku stačí použiť relatívnu cestu, pri jeho načítaní musíme uviesť absolútnu.



Týmto sme uzavreli popis programového kódu pre aplikáciu multimediálneho zápisníka verzia 1.

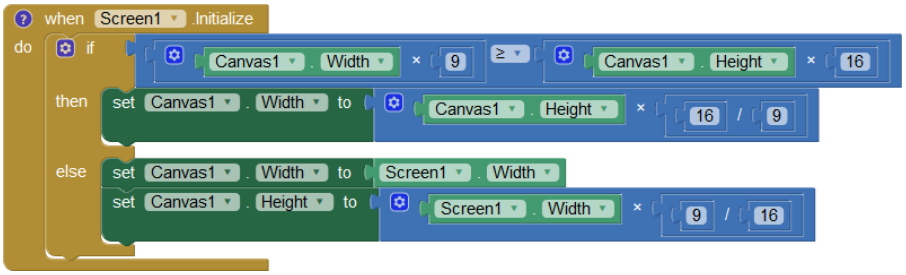
Ďalej uvedieme okomentované programové kódy ďalších funkcionalít aplikácie vo verzii 2.

Pri tvorbe webových stránok a aj iných aplikácií sa hovorí o tzv. **responzívnom dizajne**, čo znamená, že takáto aplikácia bude vyzerať a fungovať dobre na zariadeniach s rôznymi veľkosťami obrazovky. V našom prípade chceme vytvoriť aplikáciu, ktorá bude rovnako fungovať na tablete aj na smartfóne. Dosiahnuť dôsledný responzívny dizajn nemusí byť jednoduché, jednak pri špecifickom obsahu (napr. obrázok väčší ako zobrazovacia jednotka), jednak ho nemusí v plnej miere podporovať vybrané vývojové prostredie.

V Ai2 v časti **Designer** sa snažíme, aby jednotlivé vizuálne komponenty mali svoje vlastnosti nastavené na relatívne hodnoty (percentá), resp. boli automaticky nastavené alebo napasované do rodičovského komponentu (`Fill parent`). Ak máme vodorovný pás s viacerými tlačidlami, tie budú prístupné na každom zariadení, ak namiesto obyčajného kontajnera (`HorizontalArrangement`) použijeme rolovateľný kontajnér (`HorizontalScrollArrangement`). Ak máme viac komponentov na obrazovke, môžeme zaškrtnúť vlastnosť `Screen.Scrollable`. Aby naša aplikácia fungovala na zariadeniach s rôznym rozlíšením, je nevyhnuté nastaviť vlastnosť `Screen.Sizing` z hodnoty **Fixed** na hodnotu **Responsive**. Pre lepšiu predstavu ako bude vyzerať navrhnuté používateľské rozhranie na smartfóne, tablete, či monitore, môžeme v časti **Viewer** vybrať jednu z troch možností: **Phone Size** (505, 320), **Tablet Size** (675, 480), či **Monitor Size** (1024, 768).

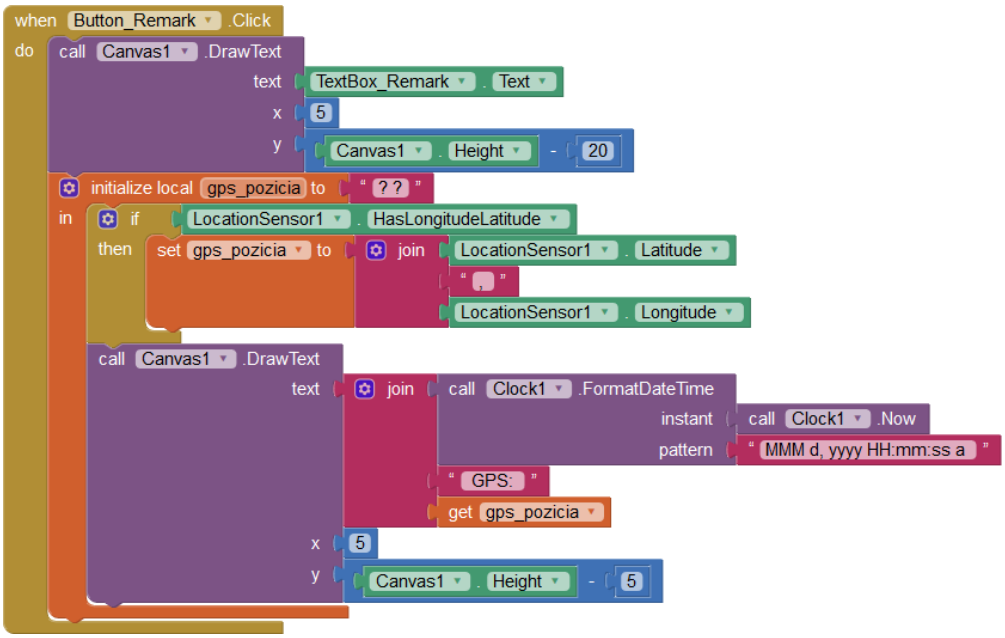
Pri písaní kódu pre responzívny dizajn odporúčame experimentovať s rôznymi zariadeniami a nechali vypisovať (napr. pomocou `Notifier.ShowAlert`) rozmery obrazovky (`Screen.Width`, `Screen.Height`) a rozmery plátna (`Canvas.Width`, `Canvas.Height`). Pri našom experimentovaní sme zistili, že na tablete má obrazovka rozmery 1280×800 a plátno pod vodorovným kontajnerom rozmery 1280×752. Na smartfóne sme zistili, že obrazovka má rozmery 640×360 a plátno pod vodorovným kontajnerom rozmery 640×312. Je zaujímavé, že pomer strán na tablete je 16:10 a na smartfóne 16:9. Keďže fotoaparáty robili snímky s rozlíšením 16:9, resp. 9:16, rozhodli sme sa, že veľkosti plátna na oboch zariadeniach budú 16:9. Pre zjednodušenie situácie sme uvažovali len vodorovné nastavenie obrazovky. Plátno sme nastavili na takú veľkosť, aby sa celé zmestilo na obrazovku

a malo pomer strán 16:9. To sme dosiahli doplnením nasledovného kódu (f11) do udalosti `Screen.Initialize`:

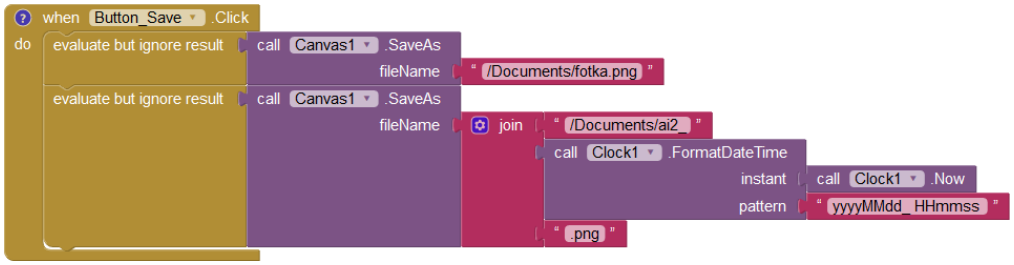


Pre ďalšie štúdium problematiky responzívneho dizajnu v prostredí Ai2 odporúčame prečítať dokument Responsive Design in App Inventor na webovej stránke MIT http://ai2.appinventor.mit.edu/reference/other/responsiveDesign.html (z 15. 8. 2015).

Pri implementácii ďalšej funkcionality – vypísaní komentára do dolnej časti plátna (f12), môžeme text zadať pomocou vyskakovacieho dialógového okna (`Notifier`) alebo pomocou textového poľa a tlačidla. Prvý spôsob šetrí miesto na obrazovke, druhý spôsob (ktorý sme vybrali) je pohodlnejší pre používateľa. Textový komentár na plátne bude pozostávať z dvoch častí v samostatných riadkov. V prvom riadku uvedieme samotný text zapísaný do textového poľa (`Textbox_Remark.Text`). V druhom riadku bude uvedená dátumová a časová značka (`Clock.Now`, `Clock.FormatDateTime`) spolu s GPS pozíciou (`LocationSensor.Latitude`, `LocationSensor.Latitude`).



Uloženie obrázkového súboru s dátumovou a časovou značkou (f13) implementujeme kódom:

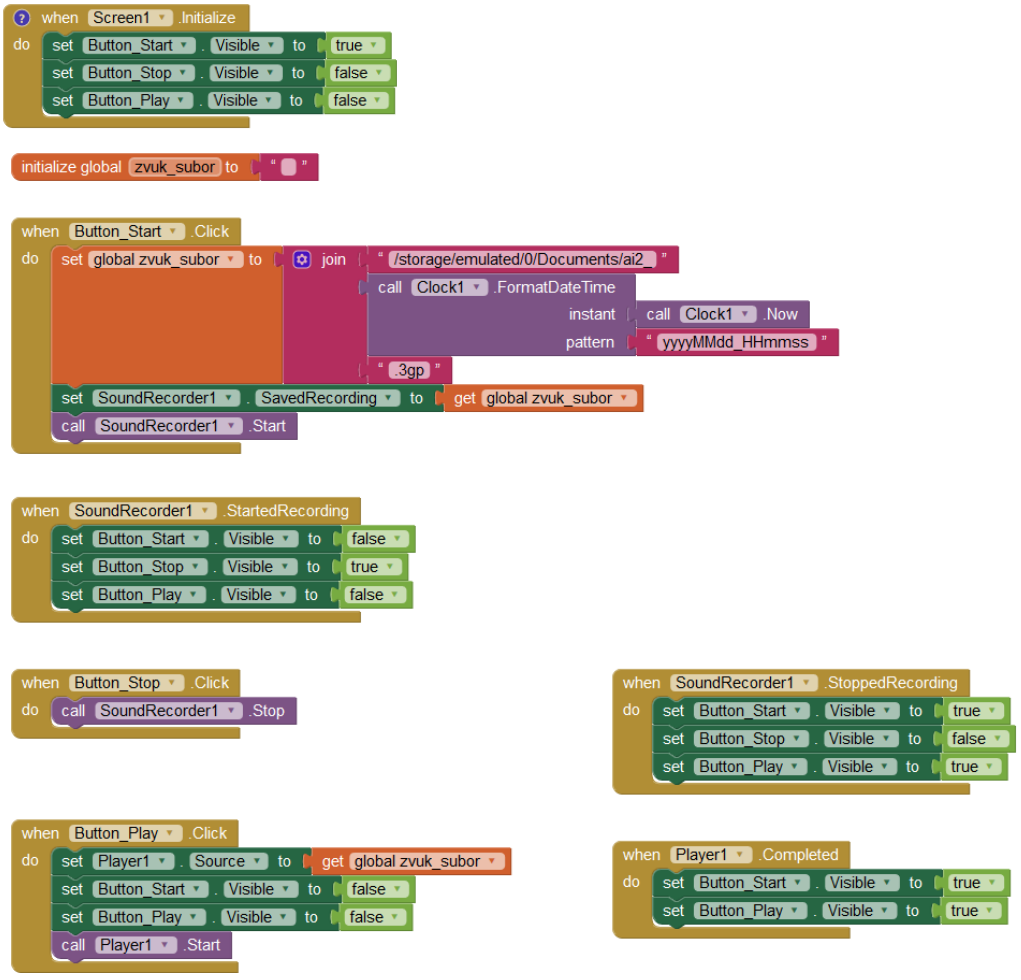


Spustenie a zastavenie nahrávania zvuku a jeho prehrávanie (f14) implementujeme príkazmi vyvolanými pomocou tlačidiel `Button.Start`, `Button.Stop`, `Button.Play`.

Aby sme používateľovi aplikácie zabezpečili pohodlné ovládanie, budeme podľa situácie meniť viditeľnosť všetkých troch tlačidiel na prácu so zvukom.

Na pomenovanie zvukového súboru s dátumovou a časovou značkou použijeme podobný kód ako pri pomenovaní grafického súboru. Aby sme mohli použiť meno tohto súboru pri nahratí komponentom `ScreenRecorder` a tiež pri prehratí komponentom `Player`, použili sme na to globálnu premennú **zvuk_subor**.

Pri stlačení tlačidiel sa spúšťajú jednotlivé metódy komponentov `ScreenRecorder` a `Player`, ktoré vyvolávajú jednotlivé udalosti (`SoundRecorder.StartedRecording`, `SoundRecorder.StoppedRecording`, `Player.Completed`).



Pri vývoji aplikácie odporúčame zabezpečiť farebný kontrast medzi písmom a pozadím. Na to nám poslúži rozšírenie webového prehliadača, napr. *WCAG Luminosity Contrast Ratio Analyzer* pre prehliadač *Google Chrome*. Pre malé písmo (Small Text) je v norme AA požadovaný minimálny kontrast 4,5:1 a v norme AAA kontrast 7,0:1. Pre veľké písmo (Large Text) je v norme AA požadovaný minimálny kontrast 3,0:1 a v norme AAA kontrast 4,5:1.

Obr. 3.1.1 Vybrané fialové písmo na bielom pozadí má kontrast 5,38:1, čo vyhovuje norme AA pre malé písmo a norme AAA pre veľké písmo.

Týmto uzatvárame komentár k možnému zdrojovému kódu aplikácie verzie 2. Pred odovzdaním a prezentáciou aplikácie je dôležité, aby sme aplikácii priradili ikonu (podľa možnosti vlastnú), vo vlastnosti `Screen.AboutScreen` uviedli meno autora aplikácie a vo vlastnostiach `Screen.VersionCode` a `Screen.VersionName` uviedli správne hodnoty.

5. Prezentácia vlastnej aplikácie a diskusia k jej využitiu v praxi a jej prípadnému doladeniu

Prezentujte svoj projekt Multimediálny zápisník v rozsahu 1–1,5 minúty. Predstavte doplnené funkcionality aplikácie spolu s komentárom k ich využitiu v praxi. Uveďte tiež, ktoré ďalšie funkcionality by ste ešte mohli doplniť do potenciálnej verzie 3 svojej aplikácie.

6. Doladenie aplikácie a jej publikovanie v rámci portfólia žiaka

Svoj prezentovaný projekt doladťe podľa návrhov učiteľa a spolužiakov a uložte ho do svojho projektového portfólia.

Zamyslime sa, čo sme sa naučili

- Navrhli sme funkcionality pre aplikáciu multimediálny zápisník.
- Vytvorili sme aplikáciu s responzívnym dizajnom využívajúcu viaceré multimediálne komponenty (`Camera`, `ImagePicker`, `Player`, `SoundRecorder`).
- Vytvárali a používali sme grafické a zvukové súbory s časovou značkou a ukladať ich do vhodných priečinkov na MZ.
- Získali sme skúsenosti s tvorbou komplexnejších projektov vyžadujúcich dekompozíciu a hlbšiu analýzu čiastkových podproblémov.

Sebahodnotiaca karta

Vyplňte uvedenú sebahodnotiacu kartu k tvorbe svojej aplikácie *Multimediálny zápisník*:

Meno a priezvisko	
Čo som sa nové naučil(a) pri programovaní tohto projektu?	
Ktoré funkcionality som doplnil(a) do svojej aplikácie?	
Ktoré funkcionality má zaujali v aplikáciách spolužiakov?	
Čo nové z problematiky Ai2 by som sa rád(a) naučil(a)?	
Čo nové by som rád/rada naprogramoval(a) v Ai2?	

3.2 Dychový tréner

Kľúčové slová

vizualizácia, animácia, zvuk, viac obrazoviek, hodiny, časovač, prepínač, farby, zoznamy.

Čo sa naučíme a čo si precvičíme

- Poskytovať informácie v rôznej forme – text, tabuľka, infografika, animácia, zvuk.
- Navrhnúť farebnú paletu aplikácie.
- Pracovať s dvomi obrazovkami (komponenty (`Screen`).
- Používať zoznamy údajov.
- Kresliť na plátno (komponent `Canvas`).
- Používať komponent `Clock` na generovanie udalostí v pravidelnom časovom intervale.
- Animovať pohyb guľôčky (komponent `Ball`).
- Použiť komponenty `CheckBox` (zaškrtávacie políčka) ako Radio Buttons (prepínače).
- Generovať hlasové a zvukové výstupy (komponenty `TextToSpeech`, `Sound`).

Čo zaujímavé sa môžeme dozvedieť?

Nádychové potápanie (free-diving) je potápanie bez dýchacieho prístroja len so zásobou vzduchu v pľúcach. Obľúbené je rekreačné potápanie na hladine mora s hlavou pod vodou so šnorchľom na dýchanie a s príležitostným ponorením do hĺbky, pri ktorom treba zadržať dych.

Nádychové potápanie je aj športová disciplína v rôznych súťažných kategóriách:

- statická apnea – zadržanie dychu na čas v plytkej vode (v bazéne),
- dynamická apnea – plávanie pod vodou na vzdialenosť (s plutvami alebo bez plutiev),
- hĺbkové potápanie – dosahovanie čo najväčšej hĺbky (so závažím, bez závažia, s plutvami, bez plutiev, s lanom, bez lana).

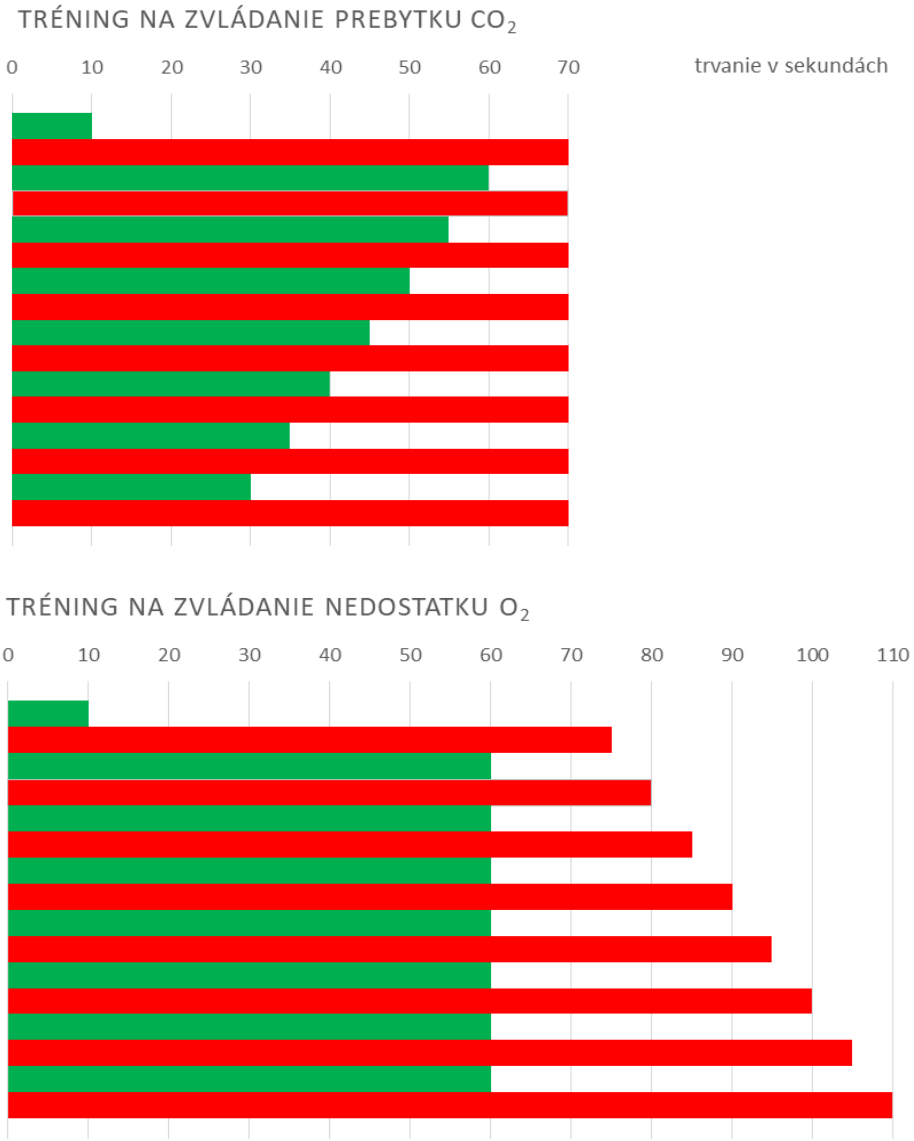
Viac o nádychovom potápaní a o rekordoch v tomto športe nájdete napríklad vo Wikipédii (Wikipédia, 2018).

Zlepšiť výkonnosť v dĺžke zadržania dychu sa dá tréningom (Yachtmeni, 2016). Pri zadržaní dychu ľudský organizmus bojuje s dvomi problémami:

- nedostatok kyslíka (O2) v krvi,
- prebytok oxidu uhličitého (CO2) v krvi.

Zvýšiť odolnosť organizmu voči nedostatku kyslíka v krvi sa dá tréningom, pri ktorom sa postupne predlžuje interval zadržania dychu pri zachovaní rovnakej dĺžky odpočinku až do 80 % hodnoty osobného maxima. Ak chceme trénovať odolnosť voči prebytku oxidu uhličitého v krvi, budeme postupne znižovať čas odpočinku pri zachovaní dĺžky zadržania dychu, ktorá by mala byť 50 % hodnoty osobného maxima.

Na obrázku 3.2.1 je sú grafy znázorňujúce tréningové plány na zvládanie prebytku CO2 a nedostatku O2 v krvi. Zelené sú časové intervaly dýchania, červené zadržania dychu.



Obr. 3.2.1 Grafy: Tréningové plány na zvládanie prebytku oxidu uhličitého a nedostatku kyslíka v krvi pri zadržaní dychu

Úloha 1

Vyplňte elektronický pracovný list podľa grafov na obrázku 3.2.1:

- Údaje z grafu zaznačte do tabuľky.
- Zistite, aké je osobné maximum zadržania dychu osoby, pre ktorú je určený tréningový plán.
- Určte parametre tréningov:
  - úvodné rozdýchanie,
  - začiatočná dĺžka zadržania dychu,
  - zmena dĺžky zadržania dychu,
  - začiatočná dĺžka vydýchania,
  - zmena dĺžky vydýchania,

    - počet opakovaní.
- Zostavte vzorce, ktoré podľa parametrov tréningu dynamicky počítajú tréningové časy dýchania a zadržania dychu. Experimentujte s rôznymi parametrami tréningu.

Akú zaujímavú aplikáciu môžeme vytvoriť?

Mobilné zariadenia môžu slúžiť športovcom na plánovanie alebo zaznamenávanie priebehu tréningov.

Otázky na zamyslenie

Poznáte nejaké aplikácie na podporu športových aktivít? Aké služby poskytujú?

Vytvoríme aplikáciu, ktorá bude asistovať svojmu používateľovi pri tréningu na zvýšenie výkonnosti v dĺžke zadržania dychu. V pracovnom liste sme už vytvorili tabuľku, ktorá počíta časové hodnoty jednotlivých úsekov tréningu. Mobilná aplikácia, ktorú vytvoríme, bude poskytovať používateľovi ďalšie užitočné služby. Číselné údaje v tabuľke názornejšie zobrazí vo forme infografiky – grafického zobrazenia dát, ktoré doplníme ešte animáciou zobrazujúcou priebeh tréningu. Mobilné zariadenie použijeme ako stopky na meranie času a okrem vizuálneho zobrazenia priebehu tréningu pridáme aj zvukové upozornenia v dôležitých úsekoch tréningu. Naša aplikácia bude:

- umožňovať nastavovanie parametrov tréningu,
- zobrazovať tréningový plán vo forme grafu,
- asistovať pri tréningu meraním času,
- zvukovo upozorňovať používateľa na zmenu akcie (dýchanie, zadržanie dychu),
- animovať priebeh tréningu pohybom kurzora po grafickom pláne.

Ako budeme postupovať pri tvorbe aplikácie?

Úloha 2

Vytvorte používateľské rozhranie aplikácie.

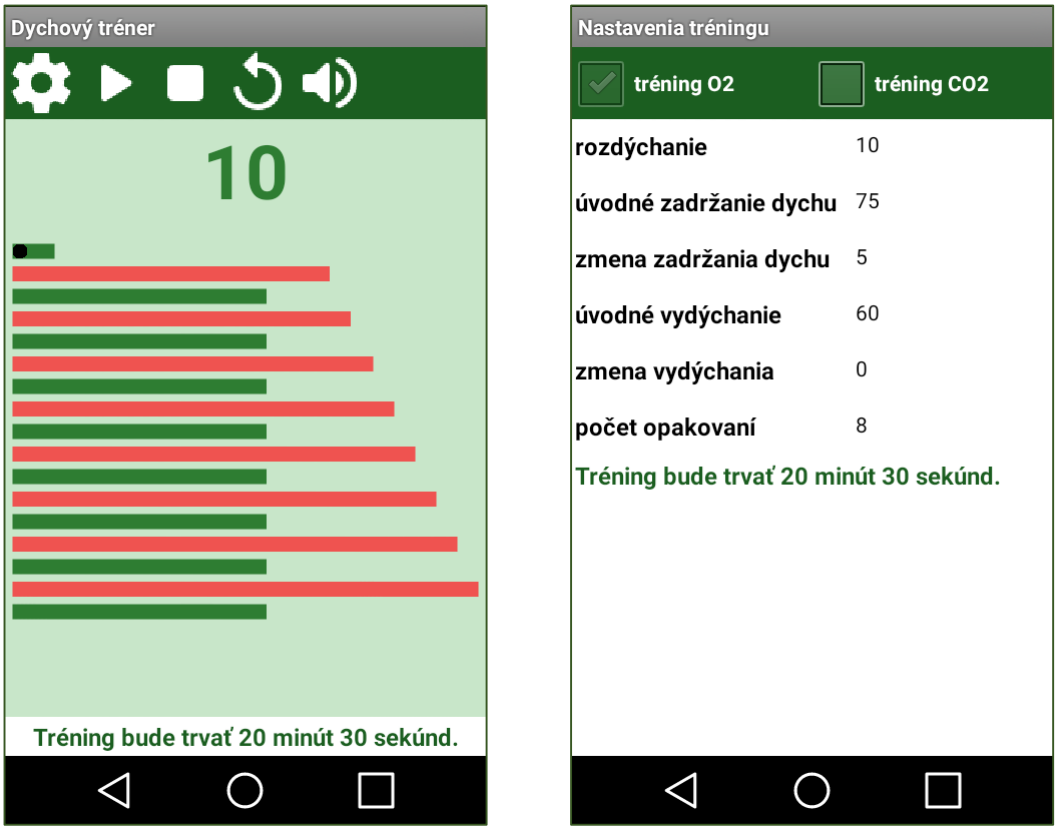
Vytvoríme dizajn aplikácie s dvomi obrazovkami napríklad ako na obr. 3.2.2. Na prvú obrazovku vložíme:

- pás (komponent `HorizontalArrangement`) ovládacích tlačidiel (komponenty `Button`) na nastavenie parametrov tréningu, spustenie tréningu, zastavenie tréningu, resetovanie tréningu, zapnutie/vypnutie zvuku,
- nápis (komponent `Label`) s odpočítavaním sekúnd počas tréningu,
- oblasť obrazovky (komponent `Canvas`), do ktorej nakreslíme tréningový plán podľa nastavených parametrov,
- kurzor (komponent `Ball`) označujúci miesto, kde sa nachádzame v pláne počas tréningu.

Na druhú obrazovku vložíme:

- dva komponenty `CheckBox` na voľbu typu tréningu (O2 alebo CO2),

- tabuľku (komponent `TableArrangement`) s názvami parametrov tréningu (komponenty `Label`) a ich hodnotami (komponenty `TextBox`).



Obr. 3.2.2 Vzhľad aplikácie: Obrazovka s priebehom tréningu, obrazovka na nastavovanie parametrov tréningu

Aby naša aplikácia mala pekný dizajn, vyberme vhodné farby a pre tlačidlá vhodné piktogramy. Vzájomne ladiace odtiene farieb a rôzne piktogramy nájdeme napríklad na stránke https://www.materialpalette.com/. Profesionálne aplikácie sa riadia pravidlami dizajnu definovanými v dizajnových systémoch. Jedným z nich je napríklad Material Design (Google, 2020).

Vysvetlíme si

Blok `make color` zo skupiny vstavaných blokov *Colors vytvára farbu zmiešaním trojice farieb (odtieňov červenej (Red), zelenej (Green) a modrej (Blue)). Odtiene farieb sa udávajú ako čísla veľkosti 1 bajt v desiatkovej sústave (0-255) vložené do trojprvkového zoznamu. Napríklad:*

Kódy farieb sa často zvyknú udávať v hexadecimálnom (šestnástkovom) kóde. Napríklad:

#1b5e20 je tmavozelená farba, kde 1b, 5e, 20 sú odtiene farieb RGB, ktoré pre použitie v bloku `make color` musíme previesť do desiatkovej sústavy na 27, 94, 32.

Správanie sa aplikácie začnime programovať grafickým zobrazením tréningového plánu.

Úloha 3

Naprogramujte funkciu `kresliGraf`, ktorá do komponentu `Canvas` vykreslí striedavo zelené a červené pruhy, ktorých dĺžky sú dané v globálnej premennej `rozvrh` typu zoznam (list).

Do globálnej premennej `rozvrh` si na testovacie účely vložme zoznam s časovými hodnotami úsekov tréningu z pracovného listu (10, 75, 60, 80, 60, 85, 60, 90, 60, 95, 60, 100, 60, 105, 60, 110). Každé číslo zo zoznamu vizualizujme ako vodorovnú čiaru, striedavo zelenou a červenou farbou (začneme zelenou farbou).

Vysvetlíme si

Vlastnosti komponentu `Canvas`:

`PaintColor` – farba, ktorou sa kreslí

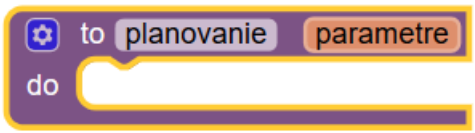
`LineWidth` – hrúbka čiary

Metóda komponentu `Canvas`:

`DrawLine` (x1, y1, x2, y2) – kreslí úsečku nastavenej hrúbky a farby s krajnými bodmi na súradniciach (x1, y1) a (x2, y2)

Úloha 4

Naprogramujte funkciu `planovanie`, ktorá pre zadaný vstupný zoznam `parametre` tréningu vygeneruje zoznam časových úsekov tréningu do globálnej premennej `rozvrh`.



`parametre` je zoznam parametrov tréningu podľa tabuľky v pracovnom liste: úvodné rozdýchanie, začiatočná dĺžka zadržania dychu, zmena dĺžky zadržania dychu, začiatočná dĺžka vydýchania, zmena dĺžky vydýchania, počet opakovaní.

Na testovacie účely nastavme zoznam `parametre` podľa údajov v pracovnom liste na (10, 75, 5, 60, 0, 8). Funkcia `planovanie` pre takéto `parametre` vygeneruje zoznam `rozvrh` = (10, 75, 60, 80, 60, 85, 60, 90, 60, 95, 60, 100, 60, 105, 60, 110)

Vygenerovaný rozvrh tréningu zobrazíme pomocou funkcie `kresliGraf`. Experimentujte s rôznymi parametrami tréningu a kontrolujte grafický výstup s hodnotami, ktoré vygenerujú vzorce v tabuľke pracovného listu.

Pri experimentovaní s rôznymi parametrami tréningu sa môže stať, že vygenerované časové hodnoty jednotlivých úsekov tréningu budú príliš veľké alebo príliš malé pre zobrazenie na displeji mobilného zariadenia. Problém s grafickým zobrazením údajov vo vhodnej mierke vyriešime naprogramovaním špeciálnej funkcie, ktorá bude číselný údaj v sekundách prevádzať na vhodný údaj v pixeloch, ktorý sa prispôsobí veľkosti displeja zariadenia.

Úloha 5

Naprogramujte funkciu `mierka`, ktorá pre zadané číslo (čas z tréningového rozvrhu) vypočíta dĺžku úsečky tak, aby sa maximálny časový úsek z tréningu zobrazil cez celú obrazovku.

Funkciu `mierka` použijeme vo funkcii `kresliGraf` pri vizualizácii tréningového plánu.

Vysvetlíme si

Mierka je pomer šírky plátna (`Canvas.Width`) a maximálneho času v tréningovom pláne (zoznam časových hodnôt v globálnej premennej `rozvrh`). Príklad výpočtu mierky a jej použitia pre výpočet dĺžky úsečky:

`rozvrh` = (10, 75, 60, 80, 60, 85, 60, 90, 60, 95, 60, 100, 60, 105, 60, 110)
max. čas v zozname `rozvrh` = 110
šírka plátna `Canvas.Width` = 320
mierka = 320/110
dĺžka čiary = čas * mierka
dĺžka čiary pre 110 sekúnd = 110 * 320/110 = 320
dĺžka čiary pre 10 sekúnd = 10 * 320/110 = 29

Poznámka: Maximálny čas v tréningovom pláne (zoznam `rozvrh`) treba zistiť výpočtom.

Teraz pridajme do nášho projektu interaktivitu. Parametre tréningu nebudú v aplikácii nastavené napevno, ale budú sa dať nastavovať a meniť priamo v aplikácii.

Úloha 6

Naprogramujte interaktívne nastavovanie parametrov tréningu pre funkciu `planovanie` na samostatnej obrazovke aplikácie.

Na druhú obrazovku aplikácie prejdeme stlačením tlačidla (udalosť `Button_Nastavenia.Click`). Po nastavení parametrov tréningu v editovacích poliach sa vrátime späť na prvú obrazovku (systémovým tlačidlom Back). Pri návrate na hlavnú obrazovku aplikácie (udalosť `Screen1.OtherScreenClosed`) odovzdáme zoznam parametrov tréningu ako výsledok v premennej `result` a spracujeme ho: vytvoríme `rozvrh` tréningu volaním funkcie `planovanie` a vizualizujeme ho volaním funkcie `kresliGraf`.

Úloha 7

Naprogramujte odpočítavanie času podľa tréningového plánu uloženého v globálnej premennej `rozvrh`:

- textové zobrazenie počtu zostávajúcich sekúnd daného úseku tréningu, farebne odlíšte text pri dýchaní zelenou a pri zadržaní dychu červenou farbou,
- hlasové upozornenie pri zmene: „Dýchaj“ alebo „Zadrž dych“,
- zvukové upozornenie pred zmenou: 5 sekúnd pred zmenou pridajte zvukové pípanie,
- animáciu pohybu kurzora (guľôčky) po grafickom zobrazení rozvrhu tréningu.

Ako časovač na odpočítavanie času v sekundách využijeme komponent `Clock` a jeho schopnosť generovať udalosti v pravidelných časových intervaloch. Necháme každú sekundu vygenerovať udalosť, na ktorú bude aplikácia reagovať vypísaním zostávajúceho času, posunom kurzora v grafickom pláne, za istých podmienok zvukovým signálom alebo hlasovým upozornením. Na textové zobrazenie odpočítavania sa hodí komponent `Label`. Na hlasové upozornenia pri zmene úseku tréningu využijeme komponent `TextToSpeech` na prevod textu do hovorenej reči. Zvukové pípanie pred koncom každého časového úseku tréningu realizujeme s využitím komponentu `Sound`. Ako pohybujúci sa kurzor na animovanie priebehu tréningu po grafickom pláne môžeme použiť komponent `Ball`.

V malých aplikáciách v kapitole 2 sme používali prvky, ktoré môžeme použiť aj v tomto projekte na realizáciu odpočítavania času podľa časového rozvrhu tréningu:

Komponenty:

- `Clock`
- `Label`
- `TextToSpeech`
- `Sound`
- `Ball`

Vlastnosti:

- `Clock.TimerEnabled`, `Clock.TimerAlwaysFires`, `Clock.Interval`
- `Label.TextColor`, `Label.Text`
- `Sound.Source`

Udalosť:

- `Clock.Timer`

Metódy:

- `TextToSpeech.Speak`
- `Sound.Play`
- `Ball.MoveTo`

Úloha 8

Naprogramujte nastavovanie predvolených parametrov dvoch typov tréningu (na zvládanie nedostatku O2 a na zvládanie prebytku CO2 v krvi) pomocou začiarkavacích políčok na druhej obrazovke aplikácie.

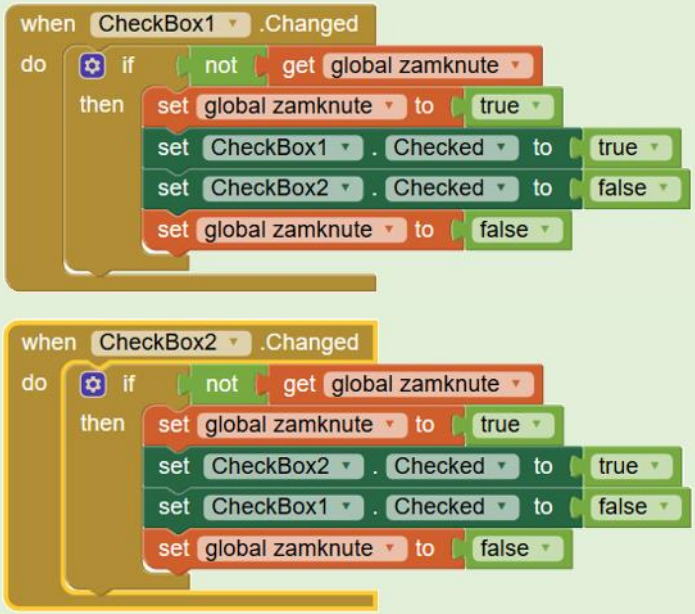
Vysvetlíme si

Check Box alebo začiarkavacie políčko je grafický ovládací prvok, pomocou ktorého sa nastavuje (*začiarkne*/*odčiarkne*) logická hodnota (*áno*/*nie*) nejakej premennej na opačnú.

Radio Button alebo tlačidlo možností je grafický ovládací prvok, pomocou ktorého sa vyberá jedna možnosť zo skupiny možností (nastavuje sa logická hodnota *áno* práve jednému zo skupiny tlačidiel, ostatné sa nastavia na *nie*).

Komponent `CheckBox` v App Inventore poskytuje funkcionalitu začiarkavacieho políčka. Pre tlačidlo možností (Radio Button) App Inventor nemá vstavaný komponent, jeho funkcionalitu môžeme naprogramovať pomocou políčok Check Box:

Dotykom na políčko `CheckBox` (udalosť `Changed`) nastavíme jeho stav vždy na `true` a stav ostatných políčok `CheckBox` v skupine na `false`. Táto zmena stavu ostatných políčok však tiež vyvolá udalosť `Changed` a kód sa vykoná aj pre ne. Aby sme tomu zabránili, podmienime reagovanie na túto udalosť pomocou globálnej logickej premennej, ktorou dočasne „uzamkneme“ reakcie na zmeny, až kým všetky políčka nebudú nastavené na správnu hodnotu. Potom opäť „odomkneme“ vykonávanie reakcií na udalosť `Changed` komponentov `CheckBox`. Riešenie pre výber z dvoch možností:



Poznámka: Všeobecné riešenie pre ľubovoľný počet tlačidiel možností v skupine preskúmajte v projekte **pmz_3_2_RadioCheck.aia**.

Ako vylepšiť či rozšíriť našu aplikáciu?

Dobrá aplikácia by mala mať tieto vlastnosti:

- *funkčnosť* – mala by byť dobre použiteľná na to, na čo je určená,
- *efektívnosť* – nemala by vyžadovať od používateľa príliš veľké úsilie na dosiahnutie cieľa (pri prvom kontakte s aplikáciou, ani pri dlhodobejšom používaní alebo po návrate k aplikácii po dlhšom čase),
- *odolnosť voči chybám používateľa* – mala by predchádzať chybám používateľa a ak k nim dôjde, účinne ich riešiť,
- *príjemnosť* – práca s aplikáciou by mala vyvolávať u používateľa spokojnosť, príjemné pocity.

Viac o použiteľnosti aplikácií sa dočítate napríklad na českej Wikipédii (Wikipedie, 2017).

Otázky na zamyslenie

Sú služby, ktoré aplikácia poskytuje, užitočné pre realizáciu dychového tréningu? Mohla by mať nejaké ďalšie užitočné funkcie?

Vie s aplikáciou jednoducho pracovať človek pri prvom kontakte? Je ovládanie aplikácie efektívne aj pre skúseného používateľa? Mohlo by sa ovládanie aplikácie nejako zjednodušiť, vylepšiť?

Je aplikácia odolná voči chybám používateľa? Rieši problémy s chybnými vstupmi?

Má aplikácia príjemný dizajn (farby, grafiku, zvuky)?

Niekoľko námetov na vylepšenie aplikácie:

- užitočná funkcia: vypočítanie a zobrazenie celkovej dĺžky tréningu,
- pomôcky pre zadávanie vstupov, napr. nápovede, z akého intervalu vyberať vstupy pre zvolený typ tréningu, informačné texty o dychovom tréningu,
- predchádzanie chybným vstupom, riešenie chybných vstupov,
- zladenie farebnej palety, starostlivý výber zvukových efektov, piktogramov.

Zamyslime sa, čo sme sa naučili

- Vytvárať aritmetické postupnosti údajov podľa vstupných parametrov.
- Programovať parametrizovanú grafiku.
- Miešať farby.
- Pracovať s dvomi obrazovkami, prenášať údaje medzi nimi.
- Naprogramovať funkcionalitu skupiny tlačidiel možností (Radio Button) pomocou začiarkavacích políčok (Check Box).
- Programovať zvukové výstupy s využitím časovača.

3.3 Prvá pomoc

Kľúčové slová

`video`, `zvuk`, `syntéza reči`, `telefonické volanie`, `viac obrazoviek`, `časovač`.

Čo sa naučíme a čo si precvičíme

- Prehrávať video v aplikácii (komponent `VideoPlayer`).
- Spustiť telefónny hovor z aplikácie (komponent `PhoneCall`).
- Pracovať s viacerými obrazovkami (komponenty `Screen`).
- Používať komponent `Clock` na generovanie udalostí v pravidelnom časovom intervale.
- Generovať hlasové a zvukové výstupy (komponenty `TextToSpeech`, `Sound`).

Čo zaujímavé sa môžeme dozvedieť?

Prvá pomoc je súbor opatrení, ktoré sa poskytujú pri ohrození života alebo postihnutí zdravia bezprostredne aj bez špeciálnych pomôcok a zdravotníckej kvalifikácie. Povinnosť poskytnúť prvú pomoc podľa svojich možností a schopností má každý, ak tým neohrozí vlastný život. Svoju schopnosť poskytnúť prvú pomoc zvýšime, ak sa aktívne na ňu pripravíme. V krízovej situácii môžu pomôcť aj okamžité a rýchlo dostupné informácie v mobilnom telefóne. Naprogramovaním vlastnej mobilnej aplikácie urobíme veľa pre svoju prípravu aj pre reálne poskytnutie pomoci v prípade potreby.

Život má v našej kultúre najvyššiu hodnotu, preto je záchrana života v krízových situáciách najvyššia priorita. Každý človek by mal poznať základné životné funkcie, život ohrozujúce stavy a úkony, ktoré môžu život zachrániť (Košická záchranka, 2016).

Základné životné funkcie sú:

- vedomie,
- dýchanie,
- činnosť srdca.

Život ohrozujúce stavy:

- Zastavenie krvného obehu (srdca) a dýchania – bez kyslíka, ktorý prijímame dýchaním a krvou sa rozvádza do celého tela, nastáva smrť po 4 až 6 minútach.
- Veľké vonkajšie krvácanie – hrozí šok až vykrvácanie (pri rozsiahlom krvácaní do 2 minút).
- Dusenie – je obmedzenie prístupu vzduchu do pľúc znepriechodnením dýchacích ciest.
- Bezvedomie – je strata schopnosti mozgu reagovať na vonkajšie aj vnútorné podnety, ktorá môže spôsobiť poruchy dýchania a smrť.
- Šok – je ťažký stav zníženého zásobovania orgánov krvou, ktoré postupne zlyhávajú.

Život zachraňujúce úkony:

- oživovanie (resuscitácia),

- zastavenie krvácania,
- uvoľnenie dýchacích ciest,
- stabilizovaná poloha,
- protišokové opatrenia,
- privolanie pomoci.

Viac o život ohrozujúcich stavoch, záchrane života a prvej pomoci si prečítajte na odkazoch odporúčaných na konci kapitoly vo Wikipédii (Wikipédia, 2017), (Wikipédia, 2019), (Wikipédia, 2020) a na Národnom portáli zdravia (Národné centrum zdravotníckych informácií, 2015-2020) alebo aj v iných zdrojoch. Pozrite si motivačné hudobné video (Baštrng - Kubovčík Michal, 2018) a ďalšie zábavné videá SEPRP – Sedlácka prvá pomoc z autorského projektu Baštrng o poskytovaní prvej pomoci (Baštrng - Michal Kubovčík, 2016-2020).

Úloha 1

Vyhľadajte a preskúmajte mobilné aplikácie na poskytovanie informácií o prvej pomoci.

Otázka na zamyslenie

Zamyslite sa a prediskutujte v skupine, ktoré funkcie skúmaných mobilných aplikácií by ste využili v krízovej situácii pri záchrane života. Snažte sa minimalizovať ich počet.

Akú zaujímavú aplikáciu môžeme vytvoriť?

Vytvorme aplikáciu, ktorá nám poskytne dôležité základné informácie a asistenciu v krízovej situácii pri záchrane života. Aby sme sa k dôležitým informáciám dostali čo najrýchlejšie, aplikácia nesmie byť príliš rozsiahla. Zamerajme sa **len na život ohrozujúce stavy a postupy, ktoré vedú k záchrane života, a na vysokú efektívnosť podávania informácií a pomoci.**

Naša aplikácia bude obsahovať:

- návody na prvú pomoc pri život ohrozujúcich stavoch (zástava srdca a dýchania, krvácanie, dusenie, bezvedomie a šok) vo forme:
  - krátke texty,
  - obrázky alebo fotografie,
  - krátke videá,
- asistenciu pri čítaní návodov – prečítanie textov (prevod textu na reč),
- asistenciu pri resuscitácii – časovač so zvukovými signálmi určujúcimi frekvenciu masáže srdca a umelého dýchania pre dospelého a dieťa,
- asistenciu pri volaní pomoci – vytočenie tiesňovej telefónnej linky.

Ako budeme postupovať pri tvorbe aplikácie?

Aplikácia má spĺňať dva dôležité ciele:

1. poskytovať dôležité informácie zrozumiteľne a efektívne, preto je dôležitý:
   - výber informácií,
   - vytvorenie prehľadnej štruktúry informácií, v ktorej sa ľahko orientuje,
   - voľba vhodnej formy prezentovania informácií.

2. užitočne asistovať pri poskytovaní prvej pomoci s využitím možností mobilného telefónu.

Najprv sa zamerajme na plnenie prvého cieľa (poskytovanie informácií). To si vyžaduje dôkladný návrh štruktúry aplikácie, veľa práce s prípravou kvalitných dát a vytvorenie prehľadného grafického používateľského rozhrania aplikácie. Potom sa pustíme do programovania ďalších asistenčných funkcií aplikácie.

Ak pracujeme vo dvojici alebo menšej skupine, spoločne diskutujme o koncepcii aplikácie a po prijatí rozhodnutia o koncepcii si môžeme úlohy rozdeliť.

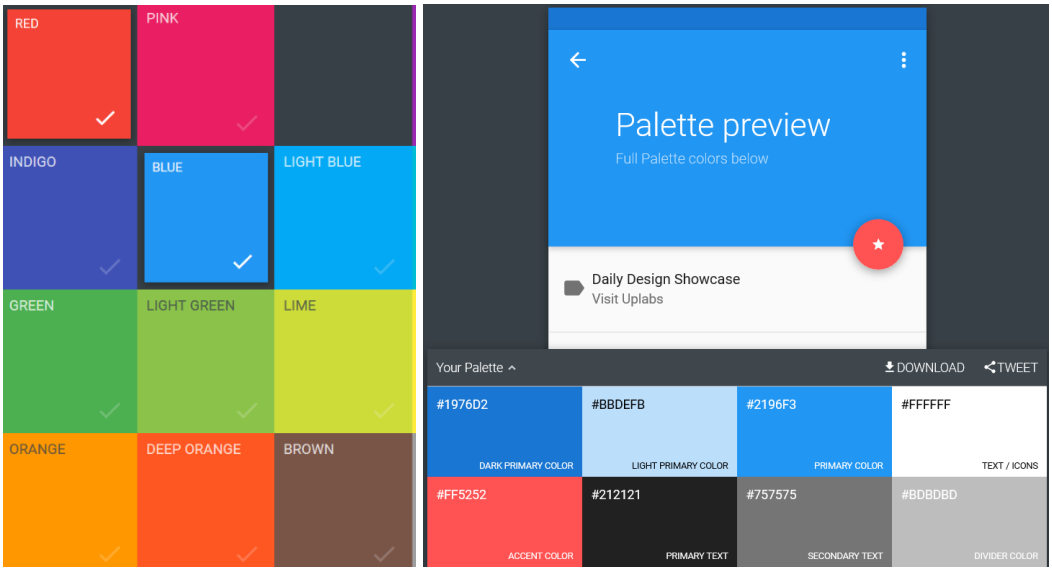
Úloha 2

Urobte konceptuálny návrh aplikácie – základnú štruktúru poskytovaných informácií. Na návrh použite softvér na tvorbu myšlienkových máp. Rozdeľte poskytovaný obsah a asistenčné služby do kategórií, znázornite štruktúru informácií v myšlienkovej mape, zvoľte a zaznamenajte formu, ktorou sa jednotlivé časti obsahu odovzdajú najefektívnejšie (text, obrázok alebo fotografia, video).

Úloha 3

Navrhnite grafické používateľské rozhranie aplikácie podľa pripraveného konceptuálneho návrhu. Vytvorte niekoľko obrazoviek, na ktoré vložte potrebné komponenty, nastavte ich vlastnosti.

Na výber farieb a ikon pri návrhu používateľského rozhrania aplikácie môžeme využiť stránky https://www.materialpalette.com/, https://material.io/tools/icons/.



Obr. 3.3.1 Návrh farebnej palety na interaktívnej stránke https://www.materialpalette.com/

Úloha 4

Pripravte texty, obrázky, videá a vložte ich do aplikácie.

Druh aplikácie, aký vytvárame, poskytuje veľa informácií, ktorými musíme aplikáciu naplniť. Údaje si pred vkladaním do aplikácie vopred pripravme a uložme do prehľadnej štruktúry, napríklad do tabuľky ako na obrázku 3.3.2.

	návod na prvú pomoc - text	obrázok	video	odkaz na	asistencia
Hlavná obrazovka					
Ďalšia obrazovka					
...					

Obr. 3.3.2 Tabuľka na prípravu a prehľad údajov v aplikácii

Do tabuľky vkladáme texty, názvy pripravených obrázkových a video súborov, názvy ďalších obrazoviek, na ktoré sa z daného miesta odkazujeme, formy asistencie, ktoré treba naprogramovať. Nie všetky bunky tabuľky budú niečo obsahovať, len kde sa to hodí. Ak pracujeme v skupine, vytvorme si zdieľanú tabuľku v cloude.

Vysvetlíme si

Na prehranie videa v App Inventore slúži vizuálny komponent `VideoPlayer` (videoprehrávač) v skupine *Media*. Počas behu aplikácie sa videoprehrávač zobrazí ako obdĺžnik, v ktorom sa pri dotyku zobrazia ovládacie prvky na prehrávanie videa.

Vlastnosť `Source` komponentu `VideoPlayer` obsahuje meno súboru s videom, ktoré sa má prehrať. Súbor musím byť vo formáte 3gp alebo mp4.

App Inventor povoľuje prehrávať len krátke videá. Veľkosť aplikácie nesmie presiahnuť 5 MB, preto videá a fotografie do aplikácie snímame s malým rozlíšením a upravíme v editore tak, aby mali čo najmenšiu veľkosť.

Úloha 5

Naprogramujte prečítanie dôležitých textov (návodov pri život zachraňujúcich úkonoch) aplikáciou.

Syntéza reči z písaného textu sa využíva v situáciách, keď človek nemôže zrakom prečítať písaný text (je nevidiaci alebo musí zrakom venovať pozornosť inému, napríklad vedeniu auta), keď kvôli postihnutiu nemôže rozprávať, alebo keď má rečou komunikovať stroj. V našom projekte je čítanie návodu na prvú pomoc užitočná funkcia v situácii, keď záchranca musí venovať pozornosť postihnutému.

Vysvetlíme si

App Inventor obsahuje v skupine *Media* nevizuálny komponent `TextToSpeech`, ktorý poskytuje funkcionalitu syntetizátora reči.

Metóda `Speak(text správy)` prečíta zadaný text (syntetizuje reč).

Úloha 6

Naprogramujte asistenciu pri resuscitácii (oživovaní). Aplikácia bude vydávať zvukové signály v požadovanom počte a frekvencii pre masáž srdca a pre umelé dýchanie v závislosti od toho, či sa zachraňuje dospelý alebo dieťa.

Pri resuscitácii sa robí masáž srdca a umelé dýchanie. Stláčanie hrudníka sa má robiť frekvenciou 100 stlačení za minútu, záchranný vdych má trvať asi 1 sekundu plus čas na nádych. U detí sa začína piatimi záchrannými vdychmi. Resuscitácia sa robí:

- až kým postihnutý nezačne sám dýchať,
- kým nepríde odborná zdravotná pomoc,
- do vyčerpania záchrancu.

Na generovanie zvukových signálov v pravidelných intervaloch využijeme komponenty `Clock` – jeden s frekvenciou 100 udalostí časovača za minútu pre stláčanie hrudníka, druhý s frekvenciou približne 30 za minútu pre záchranné vdychy.

Vysvetlíme si

Vybrané vlastnosti a udalosti komponentu `Clock`, ktoré poskytujú funkcionalitu časovača:

Vlastnosti

`TimerAlwaysFires` určuje, či časovač (Timer) bude generovať udalosti, aj keď aplikácia nie je na obrazovke zobrazená,

`TimerEnabled` určuje, či je časovač aktívny, teda či generuje udalosti v pravidelných časových intervaloch,

`TimerInterval` je interval generovania udalostí časovača v milisekundách.

Udalosť

`Timer()` nastáva pri uplynutí časového intervalu časovača.

Na odpočítavanie počtu stlačení hrudníka a záchranných vdychov použime premennú-počítadlo, ktoré pri každom tiknutí časovača znížime o jednu. Po vynulovaní počítadla opakovaní časovač deaktivujme (vlastnosť `TimerEnabled`), aktivujme druhý časovač a nastavme počítadlo na požadovaný počet opakovaní druhého časovača.

Úloha 7

Naprogramujte volanie na tiesňovú linku integrovaného záchranného systému 112 alebo záchrannú zdravotnú službu 155.

Vysvetlíme si

App Inventor obsahuje v skupine *Social* nevizuálny komponent `PhoneCall`, ktorý poskytuje funkcionality potrebné k realizovaniu telefónneho hovoru z aplikácie.

Vlastnosť `PhoneNumber` špecifikuje telefónne číslo, na ktoré sa má hovor uskutočniť, dá sa nastaviť v režime návrhu aj v programe.

Metóda `MakePhoneCall()` uskutoční volanie na číslo špecifikované vlastnosťou `PhoneNumber`.

Upozornenie: **Na tiesňové linky 112 a 155 sa volá len v tiesňových situáciách!** Neoprávnené zneužitie tiesňovej linky sa trestá pokutou, v prípade šírenia poplašnej správy aj odňatím

slobody. Pri vývoji a ladení používajme na otestovanie správneho fungovania aplikácie telefónne číslo majiteľa, ktorého nebudeme svojím volaním nepríjemne alebo neprípustne vyrušovať.

Ako vylepšiť či rozšíriť našu aplikáciu?

Na tiesňovú linku 112 je možné zaslať aj SMS so žiadosťou o pomoc. Táto služba je užitočná pre ľudí so sluchovým alebo rečovým postihnutím, ktorí nemôžu komunikovať telefonicky, alebo pre ľudí v tiesni, pre ktorých by mohol byť hlasový hovor nebezpečný. Aplikácia by mohla tiež generovať a odoslať SMS s údajmi o polohe človeka v tiesni.

Otázka na zamyslenie

Navrhnite ďalšie rozšírenia alebo vylepšenia aplikácie.

Zamyslime sa, čo sme sa naučili

- Vytvoriť konceptuálny návrh aplikácie s viacerými obrazovkami.
- Používať komponent `VideoPlayer` na prehratie krátkeho videa v aplikácii.
- Používať komponent `Clock` ako časovač na generovanie určitého počtu udalostí v pravidelných časových intervaloch.
- Používať komponent `PhoneCall` na uskutočnenie telefonického hovoru.

4 Siete

V podkapitolách 4.1 až 4.5 vytvoríme zaujímavé praktické aplikácie. Niektoré predstavujú nástroje pre prácu, iné slúžia k zábave. Spoločným znakom týchto aplikácií je využitie rôznych sieťových technológií pre vzájomnú komunikáciu a online riešení pre uchovávanie dát.

V tejto kapitole sa zameriame na vývoj 5 aplikácií využívajúcich počítačové siete na prenos dát:

4.1 **Záznamník terénnych dát**
Aplikácia pre zber dát v teréne. Dáta sa ukladajú do lokálneho CSV súboru a je možné ich posielať aj na online server. Budeme sa zaoberať aj použiteľnosťou aplikácie.

4.2 **Hlasovací systém**
Aplikácia pre hlasovanie žiakov a správu odpovedí. Využijeme online databázu *Firebase* pre záznam odpovedí. Aplikácia má dve časti – učiteľskú a žiacku.

4.3 **Pomocník pri učení sa cudzieho jazyka**
Prekladač viet. Aplikácia prekladá vety zadané textom alebo hlasom. Na preklad využijeme webovú službu *Yandex*.

4.4 **Spoločenská hra pre tablet**
Hráč háda slovo podľa opisu protihráča. Zoznam slov môže byť uložený v online databáze `TinyWebDB`. Rôzne časti aplikácie vyvíjajú v rámci tímu viacerí programátori. Ich časti aplikácie sa nakoniec spoja pomocou nástroja *App Inventor Merger* do jedného celku.

4.5 **Kockový poker**
Hra poker pre dvoch hráčov. Aplikácia riadi hru dvoch hráčov, vyhodnocuje hody a zobrazuje výsledky hráčov. Na vzájomnú komunikáciu hráčskych tabletov využijeme Bluetooth.

4.1 Záznamník terénnych dát

Kľúčové slová

záznamník dát, súbor, CSV formát, formát času, web, chyba, ergonómia aplikácie, odchytávanie chýb,

Čo sa naučíme a čo si precvičíme

- Analyzovať požiadavky na funkcionalitu a ovládanie aplikácie pre konkrétne potreby.
- Testovať aplikácie na chyby a ošetriť chybové stavy.
- Vytvárať štruktúrované dáta vo formáte CSV.
- Zvoliť vhodnú dátovú reprezentáciu pre záznamy obsahujúce viac položiek.
- Zaznamenávať dáta do lokálneho aj do vzdialeného súboru.
- Používať geolokačný senzor, čas.

Záznamník hustoty dopravy

Výhody mobilných zariadení už poznáme. Sú ľahko prenositeľné, nezávislé na externom zdroji energie, využívajú bezdrôtové pripojenie do siete, majú zabudované množstvo senzorov a pod. Nečudo, že vďaka týmto vlastnostiam upútali pozornosť aj terénnych výskumníkov a pozorovateľov.

Správa ciest pravidelne monitoruje hustotu dopravy. Vďaka tomu vie lepšie plánovať výstavbu nových ciest, predpovedať ich záťaž a v konečnom dôsledku znižovať ekologickú záťaž životného prostredia. Ich pozorovatelia stoja na vybraných úsekoch ciest a zaznamenávajú dopravné prostriedky, ktoré sledovaným úsekom cesty prechádzajú. Na záznam využívajú papierové záznamové hárky. Tento spôsob nie je veľmi efektívny, navyše je prácne spojiť záznamy od viacerých pozorovateľov. Oslovili nás, či by sme im vedeli navrhnúť a vyvinúť efektívnejší systém pre záznam dopravy. Pozorovateľ v záznamníku uvádza miesto pozorovania, čas záznamu, typ dopravného prostriedku a jeho smer.

Analyzujeme zadaný problém

Otázky na zamyslenie

Aké dáta potrebujeme zaznamenávať?

Ktoré z týchto dát vieme získať automaticky a ktoré sú na rozhodnutí pozorovateľa?

V akom formáte budeme dáta zaznamenávať?

Kde budeme dáta zaznamenávať (ukladať).

Akú funkcionalitu by mala výsledná aplikácia poskytovať?

Čo z predchádzajúceho vieme v prostredí AI2 implementovať a čo nie?

Vysvetlíme si

CSV formát (prispievatelia Wikipédie, 2017) (Comma-separated values) je jednoduchý súborový formát. Dáta sú zaznamenané vo formáte textu, pričom každý riadok súboru predstavuje jeden záznam. Položky záznamu v riadku sú oddelené čiarkou. Výhodou formátu CSV je, že je jednoduchý a vedia s ním pracovať rôzne aplikácie (napr. tabuľkový kalkulátor). Každý riadok by mal obsahovať rovnaký počet položiek. Ak by niektorá položka záznamu mala obsahovať čiarku, je potrebné celú položku uzavrieť do úvodzoviek.

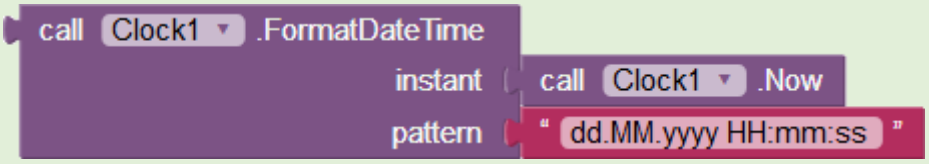
Ukážka súboru **vyplaty.csv**, ktorý obsahuje zoznam ľudí a výšku ich platu:

```
Jožko,Mrkvička,"580,32"
Karol,Petrík,702
Danka,Šikovná,"987,65"
```

Vysvetlíme si

Pri zaznamenávaní časovej značky by sme mali uvažovať, v akom formáte čas zaznamenať. Ak predpokladáme, že zaznamenané dáta o doprave budeme spracovávať v tabuľkovom tabulátore, mali by sme zvoliť formát, ktorý bude tabuľkový kalkulátor interpretovať ako čas, resp. ako dátum a čas. Ak preskúmame, aké formáty času podporuje tabuľkový kalkulátor, nájdeme medzi nimi aj formát: `22. 1. 2021 14:36:12`. V tomto formáte by sme mohli zaznamenávať čas aj my.

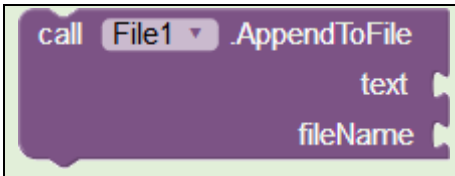
Pre prácu s časom použijeme komponent `Clock` (nájdeme ho v skupine Sensors) pomocou ktorého vieme pristupovať k systémovému času zariadenia (pozri: http://ai2.appinventor.mit.edu/reference/components/sensors.html#Clock). Súčasťou komponentu `Clock` je aj metóda pre formátovanie času (`FormatDateTime()`) ktorá upraví aktuálny čas podľa zadaného formátovacieho reťazca, resp. znakov tohto reťazca. (napr. hodina, mesiac apod.) Zoznam formátovacích znakov pre čas nájdeme na https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. Pre naše potreby môžeme čas formátovať nasledovne:



Výsledkom metódy `FormatDateTime()` je reťazec reprezentujúci čas v zadanom formáte. Výsledný reťazec použijeme ako súčasť záznamu.

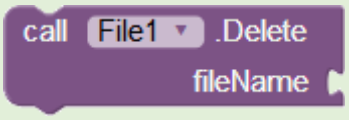
Vysvetlíme si

Pre prácu so súborom použijeme komponent `File` (nájdeme ho v skupine Storage). Komponent `File` ponúka niekoľko užitočných metód:



Pripojí text k obsahu súboru. Ak súbor neexistuje, vytvorí ho. Ak potrebujeme do súboru zapísať znak konca riadku, použijeme znak `\n`. Nasledujúci text tak bude pokračovať v ďalšom riadku.

Ak názov súboru začína prefixom lomka `/`, pokúsi sa systém lokalizovať súbor na SD karte. Ak prefix `/` vynecháme, systém lokalizuje súbor v súkromnom priestore našej aplikácie.



Zmaže súbor.

Úloha 1

Vytvorte aplikáciu pre zaznamenávanie hustoty dopravy v dvoch smeroch. Aplikácia by mala umožniť zaznamenávať všetky dáta tak, ako to robia pozorovatelia do papierového záznamníka.

Pomôcka: Premyslite si, ktoré hodnoty viete získať automaticky a ktoré sú na rozhodnutí pozorovateľa.

Pomôcka: Nezabudnite, že geolokačný senzor je potrebné po spustení aplikácie zapnúť.

Úloha 2

Pozorovateľ môže spraviť chybu a stlačiť nesprávne tlačidlo. Takýto záznam by zrejme znehodnotil prieskum hustoty dopravy.

Navrhnite a implementujte spôsob, ako postupovať v prípade chybného zápisu (chyby pozorovateľa).

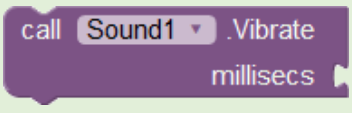
Pomôcka: Prediskutujte vzájomne rôzne riešenia a ich efektívnosť.

Úloha 3

Pozorovateľ pri zaznamenávaní musí sledovať dopravu a zároveň pracovať s aplikáciou. Niekedy si nie je istý tým, či tlačidlo stlačil alebo nie. Navrhnite a implementujte spôsob ako poskytnúť pozorovateľovi možnosť overiť si, či a aké tlačidlo stlačil.

Vysvetlíme si

Pri aplikáciách ovládaných dotykom na obrazovke je problém v tom, že primárne nemáme spätnú väzbu po stlačení tlačidla (resp. po dotyku na obrazovke). Riešením je spustiť nejakú relevantnú reakciu, napr. zmenu obrazovky, zobrazenie textu alebo zavibrovanie zariadenia.



Spustí vibrovanie zariadenia na zadaný počet milisekúnd.

Úloha 4

Pri zaškoľovaní pozorovateľov sa do súboru uloží množstvo testovacích záznamov. Zrejme by sa dali ignorovať použitím tlačidla `Button_chyba`, ale nie je to veľmi praktické riešenie. Upravte aplikáciu tak, aby sa dali testovacie dáta v súbore naraz zmazať.

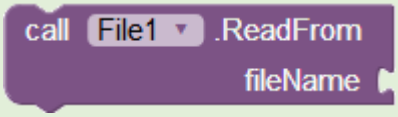
Úloha 5

Nami navrhnutý záznamník pracuje spoľahlivo a pozorovateľom značne uľahčuje ich prácu. Správa ciest pri monitorovaní dopravy však využíva viacero pozorovateľov súčasne. Každý zaznamenáva hustotu dopravy na inom mieste. Pre výsledné spracovanie by bolo potrebné tieto záznamy spojiť v jednom, centrálnom bode.

Správa ciest má vytvorenú webovú stránku, ktorá čaká na dáta od pozorovateľov a tie potom uloží do súboru na serveri. V našej aplikácii potrebujeme vyriešiť to, ako dáta zo súboru **doprava.csv** prečítať a ako ich webovej stránke poslať.

Navrhnite a implementujte spôsob ako dáta zo súboru prečítať a poslať ich webovej stránke. Ak webová stránka dáta akceptuje, odošle späť odpoveď `ok`.

Vysvetlíme si



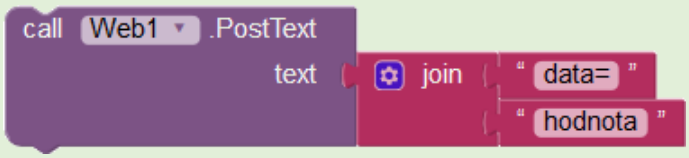
Prečíta obsah súboru. Následne je vyvolaná udalosť `GotText()`.

Udalosť nastane, ak prečítame obsah súboru. Obsah súboru je prístupný v premennej `text`.

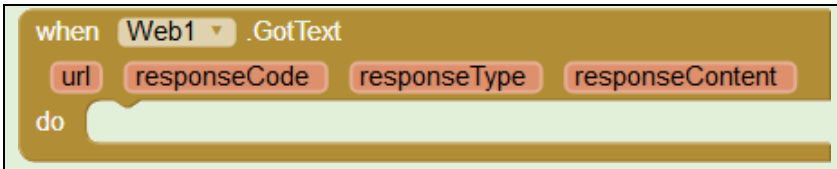
Vysvetlíme si

Komponent `Web` (v skupine Connectivity) je určený pre komunikáciu s webovou stránkou. Komponent `Web` umožňuje posielať stránke dáta a rovnako od stránky dáta prijímať.

Nastaví adresu stránky, s ktorou chceme komunikovať.



Pošle dáta stránke. Ak posielame dáta stránke, pomenujeme ich, napr. `data=`.



Udalosť nastane, ak stránka pošle nejaké dáta našej aplikácii (najčastejšie ako odpoveď, keď na stránku pristupujeme). Tento fakt využijeme, aby sme si potvrdili, že stránka naše dáta dostala, rozumie im a že si ich uložila do súboru na serveri.

Čo by sme mohli viac preskúmať? Ako vylepšiť či rozšíriť záznamník dopravy?

Nasledujúce úlohy predstavujú možné vylepšenia aplikácie. Niektoré zvyšujú komfort používania aplikácie, iné riešia problémové situácie. Je možné, že niektoré z problémov ste postrehli pri predchádzajúcich úlohách a už ich vyriešili.

Úloha 6

Otestujte vami naprogramovanú aplikáciu v rôznych situáciách. Použite ju spôsobom, ktorý ste nepredpokladali pri jej vývoji. Správa sa aplikácia vždy korektne alebo ste ju dokázali dostať do chybného stavu? Vie aplikácia v každej situácii vykonať vami požadovaný úkon?

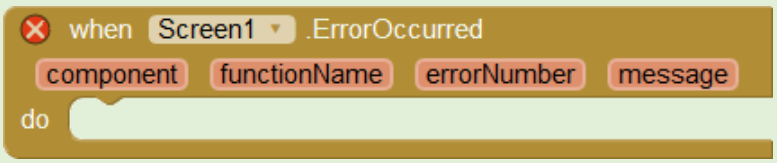
Úloha 7

Ak lokálny súbor neexistuje (pri prvom spustení aplikácie, po zmazaní testovacích dát), pri pokuse o odoslanie dát na server nastane chyba. Navrhnite spôsob, ako tejto chybe predísť.

Vysvetlíme si

Pri práci s aplikáciou môžu nastať chyby, ktoré je vopred ťažké predpovedať alebo im zabrániť. Výsledkom je, že aplikácia zobrazí nejakú chybovú, pre pozorovateľa nejasnú správu. Takéto správanie nie je žiadúce. Dobre naprogramovaná aplikácia by takéto chyby nemala zobrazovať. Používateľa by mala upozorniť vhodne zvoleným spôsobom a presne popísať, aká chyba nastala.

App inventor obsahuje jednoduchý mechanizmus pre odchytávanie chýb. Pomocou metódy `ErrorOccured()` komponentu `Screen` vieme odchytiť a zareagovať na chybný stav.



Úloha 8

Pri odosielaní dát na server odosielame aj chybné dáta (ak pozorovateľ urobil chybný záznam). Výhodnejšie by bolo uložené dáta analyzovať a chybné záznamy vynechať. Navrhnite a implementujte spôsob ako z lokálnych dát vynechať chybné záznamy a rovnako aj záznamy informujúce o chybe.

Vysvetlíme si

Štruktúrované dáta môžeme reprezentovať rôznymi spôsobmi. O formáte CSV sme hovorili v predchádzajúcej časti. Tento formát sa hodí skôr pre súbory. Pri práci s dátami CSV je vhodné ich transformovať do zoznamu – každý záznam (riadok) z CSV súboru bude samostatným prvkom zoznamu.

CSV text skonvertuje do zoznamu. Jednotlivé záznamy (riadky) textu sú transformované do samostatných položiek zoznamu.

Skonvertuje zoznam do textu v CSV formáte. Jednotlivé položky zoznamu sú transformované do samostatných riadkov v texte.

Úloha 9

Pozorovateľ mohol spraviť aj ďalší typ chyby. Tlačidlo pre zmazanie chybného záznamu stlačil viackrát než ako bol počet záznamov pred tým. Pri pokuse o odstránenie chybných záznamov (pred odoslaním na server) sa pokúsime odstrániť záznam z prázdneho zoznamu. To samozrejme spôsobí chybu. Navrhnite a implementujte spôsob ako tejto chybe predísť.

Úloha 10

Pri úprave záznamov sa môže stať, že výsledkom je prázdny zoznam. Súbor záznamov síce obsahoval nejaké záznamy, ale obsahoval aj informáciu o tom, že tieto záznamy sú chybné. Výsledkom je, že na server posielame prázdny reťazec (žiadne záznamy). Navrhnite spôsob ako tomuto zabrániť a ako informovať pozorovateľa, že nie sú žiadne záznamy pre odoslanie na server.

Zamyslime sa. čo sme sa naučili

- Analyzovať požiadavky používateľov a implementovať ich do návrhu aplikácie.
- Testovať aplikáciu s cieľom detegovať chybné stavy a ošetriť tieto chybné stavy.
- Popísať dáta v štruktúrovanom formáte CSV.
- Ukladať a čítať dáta zo súboru.
- Konvertovať dáta z formátu CSV do zoznamu a naopak.
- Komunikovať z webovou stránkou.

Sebahodnotiaca karta

Zapísaním symbolu ✓ do stĺpcov tabuľky uveďte, do akej miery ovládate uvedené prvky učiva.

Z uvedeného učiva viem vykonať nasledovné činnosti:	**samostatne**	**s malou pomocou**	**s malou pomocou**
Viem vytvoriť program, ktorý zapisuje dáta do súboru			
Viem vytvoriť program, ktorý zapisuje dáta do súboru v CSV formáte			
Viem vytvoriť program, ktorý zistí systémový čas			
Viem vytvoriť program, ktorý formátuje systémový čas			
Viem vytvoriť procedúru s parametrami			
Viem ako zistiť, že nastala chyba pri vykonávaní programu			
Viem vytvoriť program, ktorý pošle dáta webovej stránke			
Viem konvertovať dáta z CSV formátu do zoznamu a naopak			

4.2 Hlasovací systém

Kľúčové slová

`webová databáza`, `edukačná pomôcka`, `formatívne hodnotenie`, `komponent FirebaseDB`, `komponent ListView`, `hlasovanie`, `archivácia hlasovaní s dátumovou a časovou značkou`

Čo sa naučíme a čo si precvičíme

- Navrhnúť štruktúru databázy a funkcionality dvojice aplikácií na realizáciu viacerých hlasovaní žiakov a ich riadenie a archiváciu učiteľom využívajúcich navrhnutú databázu.
- Naprogramovať online učebnú pomôcku na formatívne hodnotenie – dvojicu aplikácií na realizáciu, riadenie a archivovanie hlasovania využívajúcu spoločnú webovú **databázu Firebase** (Realtime Database).
- Aplikovať komponent `FirebaseDB` a jeho metódy `GetValue`, `StoreValue`, `AppendValue` a udalosti `GotValue`, `DataChanged` a tiež funkcie na prácu s údajovým typom **zoznam** (`create empty list`, `make a list`, `replace list item`, `select list item`, `add item to list`).

Čo zaujímavé môžeme zistiť (o využití online hlasovania vo výučbe)?

Predstavme si situáciu, že chceme vytvoriť dvojicu aplikácii pre online hlasovanie vo výučbe, ktorú by sme mohli využiť vo vyučovaní informatiky či iných predmetov.

Otázky na zamyslenie

Prediskutujme nasledovné otázky:

- Aký význam pre učiteľa má, keď žiaci hlasujú (zdvihnutím ruky či pomocou aplikácie v smartfóne)?
- V ktorých situáciách v škole či mimo školy má zmysel urobiť hlasovanie?
- Ktoré aplikácie na MZ by sa dali použiť na online hlasovanie?
- Aký zmysel má vytvárať vlastnú aplikáciu na online hlasovanie?

Akú zaujímavú aplikáciu môžeme vytvoriť?

Ak sme sa rozhodli pre tvorbu vlastnej aplikácie, mali by sme v triede formou brainstormingu zozbierať nápady k možným funkcionalitám online hlasovania.

Ako budeme postupovať pri tvorbe aplikácií?

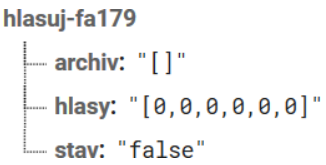
Pri tvorbe vlastného projektu môžeme postupovať podľa nasledovných krokov:

1. Spresnenie špecifikácie navrhovaných aplikácií (pre učiteľa aj pre žiaka)
2. Návrh používateľského rozhrania oboch aplikácií, zoznam komponentov a multimediálnych súborov
3. Návrh správania oboch aplikácií
4. Tvorba používateľského rozhrania a programového kódu oboch aplikácií
5. Prezentácia vlastných vytvorených aplikácií a diskusia ich využitia v praxi a ich prípadného doladenia
6. Doladenie aplikácií a ich publikovanie v rámci portfólia žiaka

1. Špecifikácia aplikácií

Obidve aplikácie (žiacka aj učiteľova) využívajú tú istú webovú databázu s kľúčmi:

- **hlasy** – (6-prvkový) zoznam početnosti vybraných možností
- **stav** – pravdivostná hodnota true/false predstavujúca stav hlasovania (otvorené = true, uzavreté = false)
- **archiv** – zoznam uložených hlasovaní, každé hlasovanie je zoznamom hodnôt: dátumu, času a zoznamu hlasovania v uvedenom dátume a čase



Žiacka aplikácia má nasledovné funkcionality:

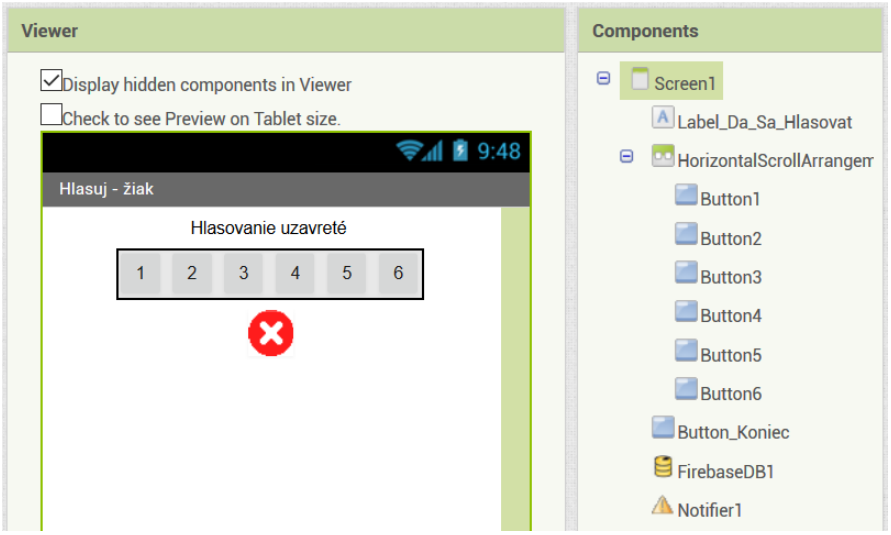
f 1. Kliknutie na jedno z hlasovacích tlačidiel a zvýšenie odpovedajúcej hodnoty vo webovej databáze
f 2. Zobrazenie stavu aplikácie a k tomu odpovedajúce zobrazenie, resp. skrytie hlasovacích tlačidiel

Učiteľská aplikácia má nasledovné funkcionality:

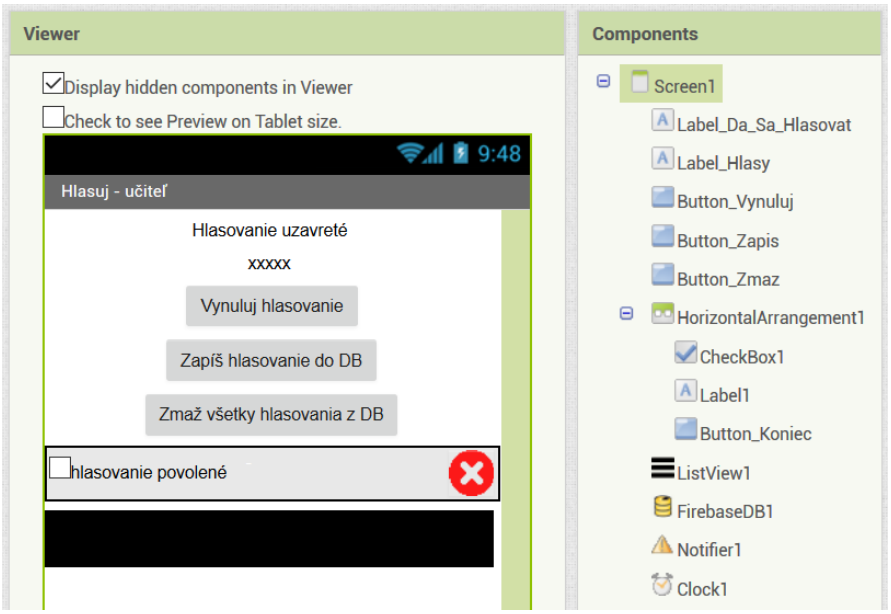
f 3. Zapnutie/vypnutie možnosti hlasovania
f 4. Zobrazenie aktuálneho stavu hlasovania (zoznam s početnosťami hlasovania jednotlivých možností)
f 5. Vynulovanie stavu hlasovania
f 6. Zapísanie výsledku hlasovania do databázy (dátum, čas, početnosti hlasovania jednotlivých možností)
f 7. Zobrazenie všetkých výsledkov hlasovania z databázy v textovej podobe
f 8. Zmazanie všetkých výsledkov hlasovania z databázy

2. *Návrh používateľského rozhrania aplikácií, zoznam komponentov a multimediálnych súborov*

Používateľské rozhranie (žiacka aplikácia)



Používateľské rozhranie (učiteľská aplikácia)



Zoznam komponentov (žiacka aplikácia)

Vizuálne komponenty:

- `Label_Da_Sa_Hlasovat` – textové pole na výpis stavu hlasovania podľa hodnoty kľúča **stav** webovej databázy (f2)
- `HorizontalScrollArrangement` – vodorovný kontajner zobrazený podľa hodnoty kľúča **stav** webovej databázy (f2)
  - `Button1`, `Button2`, `Button3`, `Button4`, `Button5`, `Button6` – hlasovacie tlačidlá na zvýšenie odpovedajúcej hodnoty vo webovej databáze (f1)
- `Button_Koniec` – tlačidlo na ukončenie behu aplikácie
- `Notifier` – vyskakovacie okno na výpis prípadnej chybovej hlášky o `FirebaseDB`

Nevizuálne komponenty:

- `FirebaseDB` – webová databáza s tromi kľúčmi (**stav**, **hlasy**, **archiv**) (f1, f2)

Zoznam komponentov (učiteľská aplikácia)

Vizuálne komponenty:

- `Label_Da_Sa_Hlasovat` – textové pole na výpis stavu hlasovania podľa kľúča **stav** webovej databázy (f2)
- `Label_Hlasy` – zobrazenie aktuálneho stavu hlasovania (f4)
- `Button_Vynuluj` – tlačidlo na vynulovanie stavu hlasovania (f5)
- `Button_Zapis` – tlačidlo na zapísanie výsledku hlasovania do databázy (f6)
- `Button_Zmaz` – tlačidlo na zmazanie všetkých výsledkov hlasovania z databázy (f8)
- `HorizontalArrangement` – vodorovný kontajner
  - `CheckBox` – zaškrtávacie pole na zapnutie/vypnutie možnosti hlasovania (f3)
  - `Label1` – oddeľovač vizuálnych komponentov
  - `Button_Koniec` – tlačidlo na ukončenie behu aplikácie
- `ListView` – zobrazovač zoznamov všetkých výsledkov hlasovania z databázy (f7)
- `Notifier` – vyskakovacie okno na výpis prípadnej chybovej hlášky o `FirebaseDB`

Nevizuálne komponenty:

- `FirebaseDB` – webová databáza s tromi kľúčmi (**stav**, **hlasy**, **archiv**) (f3-f8)
- `Clock` – hodiny na zapísanie výsledku hlasovania do databázy – dátum, čas, početnosti hlasovania jednotlivých možností (f6)

Zoznam multimediálnych súborov (žiacka aplikácia)

exit.png – obrázok tlačidla `Button_Koniec`

123_ziak_ikona.png – ikona aplikácie

Zoznam multimediálnych súborov (učiteľská aplikácia)

exit.png – obrázok tlačidla `Button_Koniec`

123_ucitel_ikona.png – ikona aplikácie

3. Návrh správania aplikácií

Žiacka aplikácia

Komponent(y)	Udalosť	Akcia

`Button1`
`Button2`
`Button3`
`Button4`
`Button5`
`Button6`
	`Click`	(f1, f2) zvýšenie odpovedajúcej hodnoty vo webovej databáze (`replace list item`, `FirebaseDB.StoreValue`, `Label_Da_Sa_Hlasovat`, `HorizontalScrollArrangement`)

`Button_Koniec`	`Click`	ukončenie behu aplikácie (`close application`)

FirebaseDB	GotValue	(f2) získanie hodnôt kľúčov **hlasy** a **stav** z webovej databázy, nastavenie globálnych premenných **hlasy** a **stav**
FirebaseDB	DataChanged	(f2) získanie hodnôt kľúčov **hlasy** a **stav** z webovej databázy, nastavenie globálnych premenných **hlasy** a **stav**
FirebaseDB	FirebaseError	zobrazenie prípadnej chyby pri práci s webovou databázou pomocou vyskakovacieho okna (Notifier.ShowAlert)
Screen	Initialize	spustenie metódy FirebaseDB.GetValue
		globálne premenné: **stav** = false **hlasy** = empty list

Učiteľská aplikácia

Komponent	Udalosť	Akcia
CheckBox	Changed	(f3) zapnutie/vypnutie možnosti hlasovania, zmena premennej **stav** aj kľúča **stav** vo webovej databáze (FirebaseDB.StoreValue, Label_Da_Sa_Hlasovat)
Button_Vynuluj	Click	(f5) vynulovanie premennej **hlasy** aj kľúča **hlasy** a nastavenie kľúča **stav** = false vo webovej databáze (FirebaseDB.StoreValue)
Button_Zapis	Click	(f6) zapísanie výsledku hlasovania v premennej **hlasy** spolu s dátumovou a časovou značkou do databázy (FirebaseDB.AppendValue, Clock.Now)
Button_Zmaz	Click	(f8) zmazanie všetkých výsledkov hlasovania z databázy FirebaseDB.StoreValue
Button_Koniec	Click	ukončenie behu aplikácie (close application)
FirebaseDB	GotValue	(f2) získanie hodnôt kľúčov **hlasy**, **stav** a **archiv** z webovej databázy, nastavenie globálnych premenných **hlasy**, **stav** a **archiv** (f4) zobrazenie aktuálneho stavu hlasovania (Label_Hlasy) (f7) zobrazenie všetkých výsledkov hlasovania z databázy (ListView.Elements)

`FirebaseDB`	`DataChanged`	(f2) získanie hodnôt kľúčov **hlasy**, **stav** a **archiv** z webovej databázy, nastavenie globálnych premenných **hlasy**, **stav** a **archiv** (f4) zobrazenie aktuálneho stavu hlasovania (`Label_Hlasy`) (f7) zobrazenie všetkých výsledkov hlasovania z databázy (`ListView.Elements`)
`FirebaseDB`	`FirebaseError`	zobrazenie prípadnej chyby pri práci s webovou databázou pomocou vyskakovacieho okna (`Notifier.ShowAlert`)
`Screen`	`Initialize`	spustenie metódy `FirebaseDB.GetValue`.
		globálne premenné: **stav** = false **hlasy** = empty list **archiv** = empty list

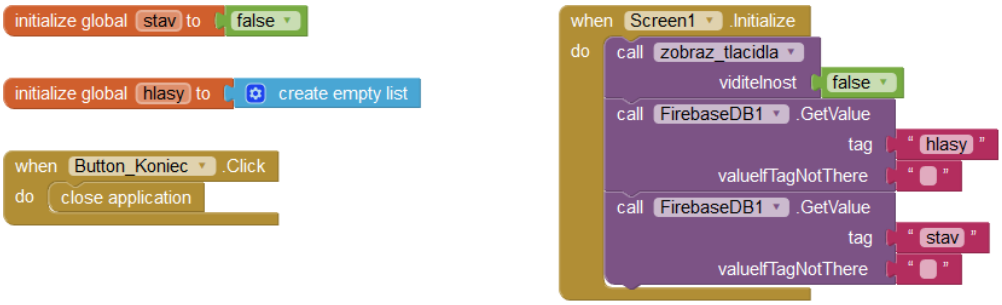
4. Tvorba používateľského rozhrania a programového kódu aplikácií

Pri tvorbe používateľského rozhrania aplikácií použijeme návrh grafického používateľského rozhrania obsahujúci vizuálne komponenty (`Button`, `Label`, `CheckBox`, `HorizontalScrollArrangement`, `HorizontalArrangement`, `ListView`, `Notifier`), nevizuálne komponenty (`FirebaseDB`, `Clock`) a multimediálne súbory (ikony a obrázky tlačidiel).

Programový kód vytvárame po jednotlivých funkcionalitách, ktorých riešenia uvedieme a okomentujeme po skupinách.

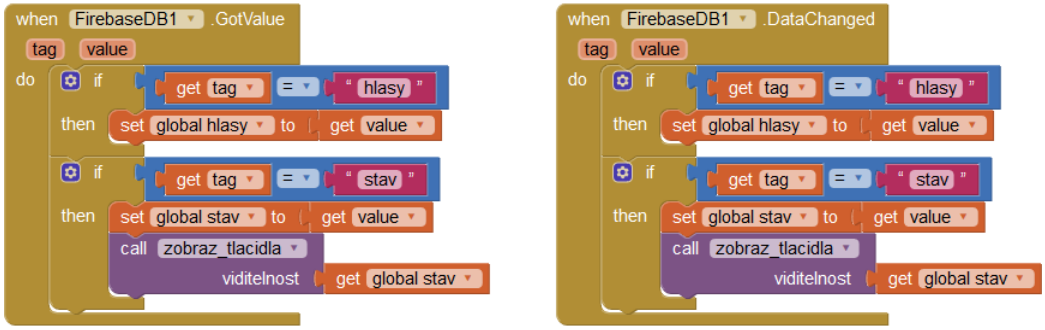
Výsledný programový kód žiackej aplikácie

Pri spustení žiackej aplikácie zabezpečíme inicializáciu premenných **stav** (logická hodnota) a **hlasy** (zoznam), skryjeme skupinu (šiestich) hlasovacích tlačidiel a vyžiadame z webovej databázy (pomocou metódy `FirebaseDB.GetValue`) aktuálne hodnoty kľúčov **stav** a **hlasy**.

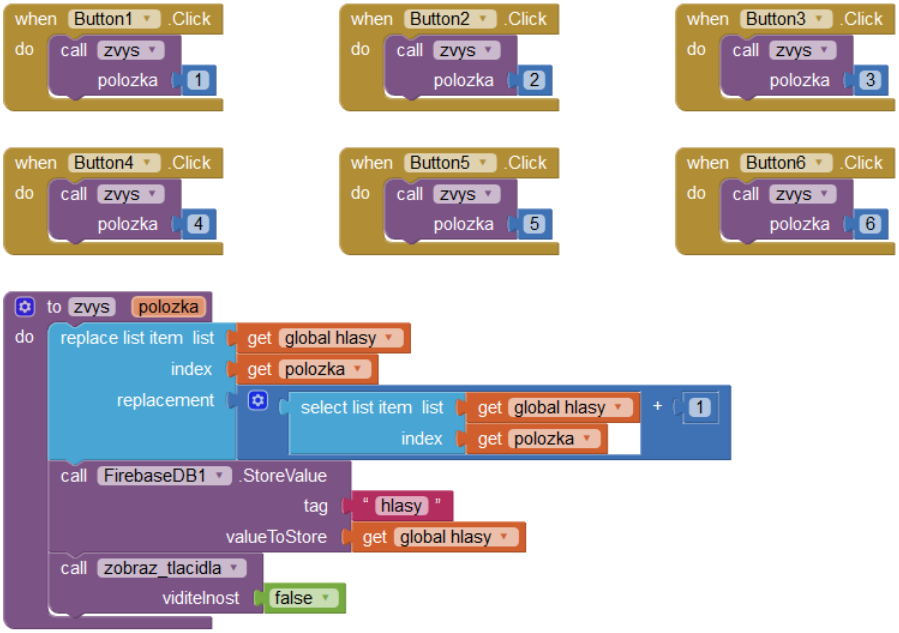


Po získaní hodnôt kľúčov z databázy sa vyvolá udalosť `FirebaseDB.GotValue`, v rámci ktorej získané hodnoty kľúčov načítame do globálnych premenných **stav** a **hlasy**. Podľa hodnoty stav sa zobrazia alebo skryjú hlasovacie tlačidlá. Rovnaké príkazy uvedieme aj keď

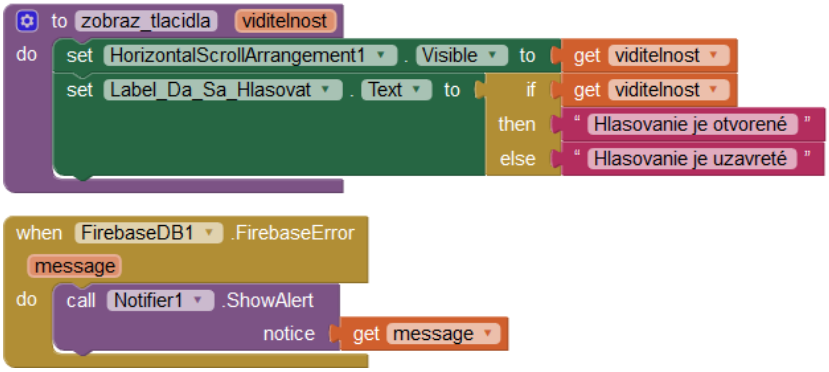
sa zmení hodnota niektorého z kľúčov vo webovej databáze, vtedy sa vyvolá udalosť `FirebaseDB.DataChanged`.



Samotné hlasovanie zabezpečíme pomocou udalostí `Button.Click` pre každý zo šiestich hlasovacích tlačidiel, ktoré spúšťajú rovnakú procedúru **zvys** s parametrom **polozka** určujúcim číslo vybranej voľby (f1). Táto procedúra na uvedenom mieste zoznamu zvýši jeho hodnotu o 1 a zapíše do webovej databázy (`FirebaseDB.StoreValue`) a schová tlačidlá.



Procedúra **zobraz_tlacidla** schová vodorovný kontajner s hlasovacími kľúčmi a nastaví do komponentu `Label_Da_Sa_Hlasovat` patričný text „Hlasovanie otvorené/uzavreté“(f2).

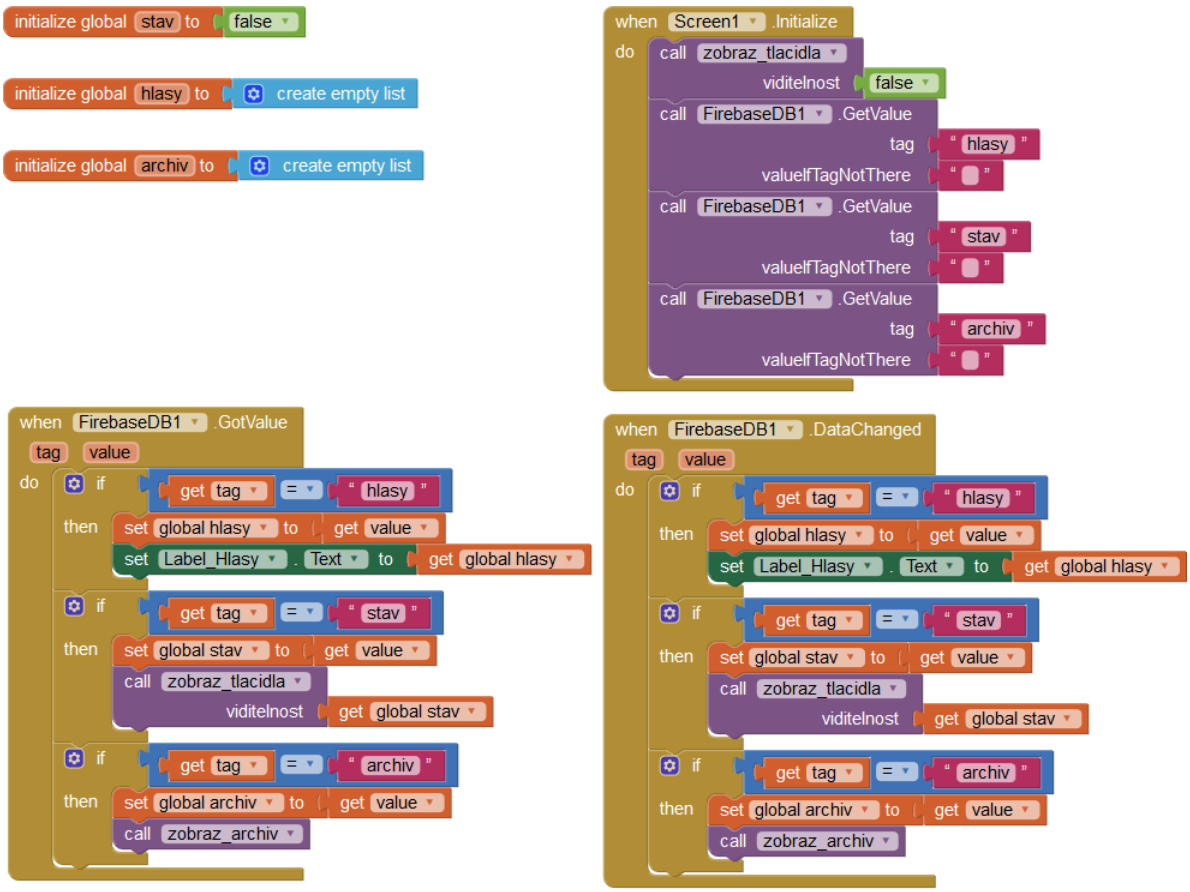


Prípadná chyba komunikácie s webovou databázou vyvolá udalosť `FirebaseDB.FirebaseError`, ktorá vypíše text tejto chyby pomocou vyskakovacieho (`Notifier.ShowAlert`).

Výsledný programový kód učiteľskej aplikácie

Pri spustení učiteľskej aplikácie zabezpečíme inicializáciu premenných **stav** (logická hodnota), **hlasy** (zoznam) a **archiv** (zoznam), vypneme `CheckBox` reprezentujúci stav hlasovania a vyžiadame z webovej databázy (pomocou metódy `FirebaseDB.GetValue`) aktuálne hodnoty kľúčov **stav**, **hlasy** a **archiv**.

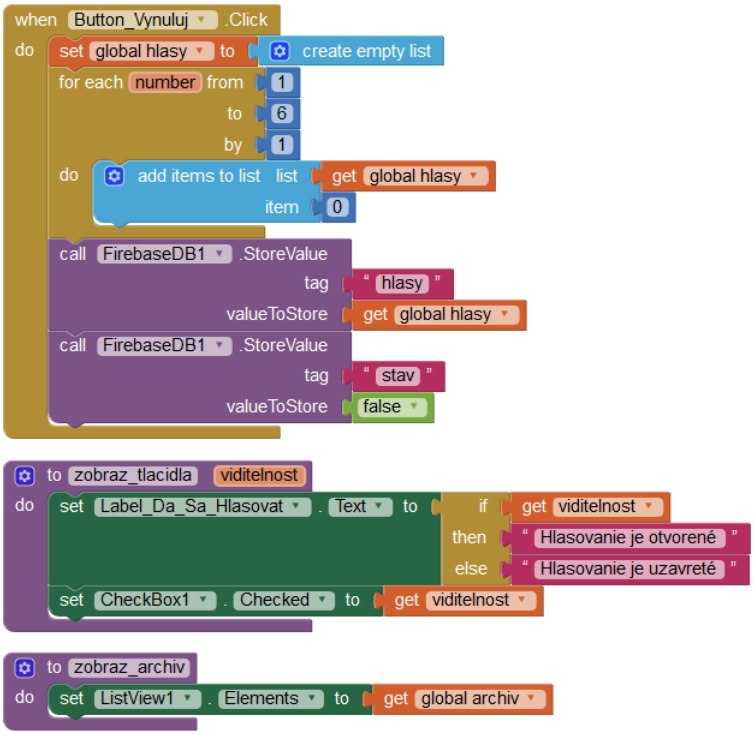
Po získaní hodnôt kľúčov z databázy sa vyvolá udalosť `FirebaseDB.GotValue`, v rámci ktorej získané hodnoty kľúčov načítame do globálnych premenných **stav**, **hlasy** a **archiv**. Podľa hodnoty stav sa zobrazí alebo skryje komponent `Checkbox` reprezentujúci stav hlasovania (otvorené/uzavreté) (f3). Zároveň zobrazíme zoznam s početnosťami hlasovania jednotlivých možností (f4). Rovnaké príkazy uvedieme, aj keď sa zmení hodnota niektorého z kľúčov vo webovej databáze, vtedy sa vyvolá udalosť `FirebaseDB.DataChanged`.



Pomocou kódu v tlačidle `Button_Vynuluj` vytvoríme zoznam **hlasy** s nulovými prvkami, ktorý zapíšeme do webovej databázy (`FirebaseDB.StoreValue`) (f5).

Procedúra **zobraz_tlacidla** zapne alebo vypne komponent `CheckBox` (reprezentujúci stav hlasovania) a nastaví do komponentu `Label_Da_Sa_Hlasovat` patričný text „Hlasovanie otvorené/uzavreté“(f2).

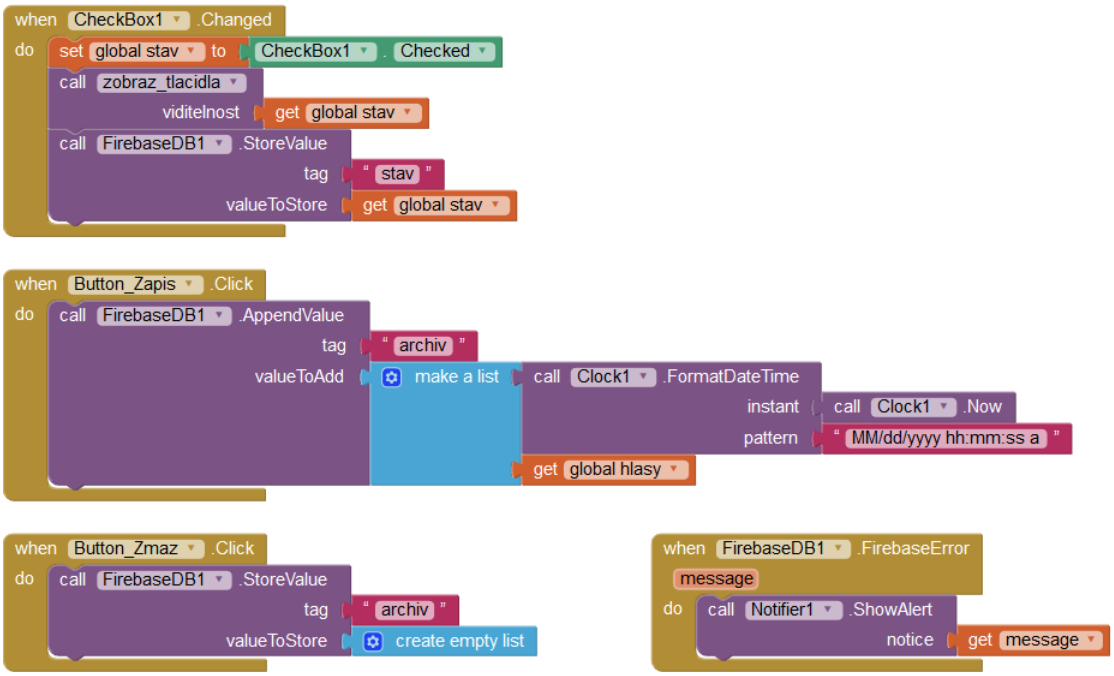
Výsledky všetkých hlasovaní zabezpečíme pomocou komponentu `ListView`, kde nastavíme vlastnosť `ListView.Elements` na hodnotu globálnej premennej **archiv** (f7).



Hlasovanie zapíname a vypíname pomocou komponentu `CheckBox` (f3), pričom zároveň do webovej databázy zapíšeme hodnotu kľúča **stav** uloženej v globálnej premennej **stav**.

Výsledky hlasovania zapisujeme do webovej databázy na koniec zoznamu zoznamov **archiv** pomocou metódy `FirebaseDB.AppendValue` (f6).

Výsledky všetkých hlasovaní môžeme zmazať pomocou metódy `FirebaseDB.StoreValue` s kľúčom **archiv** a s hodnotou valueToStore nastavenou na prázdny zoznam (f8).



Prípadná chyba komunikácie s webovou databázou vyvolá udalosť `FirebaseDB.FirebaseError`, ktorá vypíše text tejto chyby pomocou vyskakovacieho (`Notifier.ShowAlert`).

5. Prezentácia vlastnej aplikácie a diskusia k jej využitiu v praxi a jej prípadnému doladeniu

Prezentujte svoj projekt Hlasovací systém v rozsahu 1–1,5 minúty. Predstavte doplnené funkcionality aplikácie spolu s komentárom k ich využitiu v praxi. Uveďte tiež, ktoré ďalšie funkcionality by ste ešte mohli doplniť do potenciálnej verzie 3 svojej aplikácie.

6. Doladenie aplikácií a ich publikovanie v rámci portfólia žiaka

Svoj prezentovaný projekt doladťe podľa návrhov učiteľa a spolužiakov a uložte ho do svojho projektového portfólia.

Zamyslime sa, čo sme sa naučili

- Navrhli sme štruktúru databázy a funkcionality dvojice aplikácií na realizáciu viacerých hlasovaní žiakov a ich riadenie a archiváciu učiteľom využívajúcich navrhnutú databázu.
- Naprogramovali sme online učebnú pomôcku na formatívne hodnotenie – dvojicu aplikácií na realizáciu, riadenie a archivovanie hlasovania využívajúcu spoločnú webovú databázu **Firebase** (Realtime Database).
- Pri tvorbe aplikácií sme aplikovali komponent `FirebaseDB` a jeho sme metódy `GetValue`, `StoreValue`, `AppendValue` a udalosti `GotValue`, `DataChanged` a tiež funkcie na prácu s údajovým typom **zoznam** (`create empty list`, `make a list`, `replace list item`, `select list item`, `add item to list`).

Sebahodnotiaca karta

Vyplňte uvedenú sebahodnotiacu kartu k tvorbe svojej aplikácie *Hlasovací systém*:

Meno a priezvisko	
Čo som sa nové naučil(a) pri programovaní tohto projektu?	
Ktoré funkcionality som doplnil(a) do svojej aplikácie?	
Ktoré funkcionality má zaujali v aplikáciách spolužiakov?	
Čo nové z problematiky Ai2 by som sa rád(a) naučil(a)?	
Čo nové by som rád/rada naprogramoval(a) v Ai2?	

4.3 Pomocník pri učení sa cudzieho jazyka

Kľúčové slová

webová služba, strojový preklad, YandexTranslate, syntéza a rozpoznávanie reči, lokálna databáza, viac obrazoviek

Čo sa naučíme a čo si precvičíme

- Dozvieme sa, čo je webová služba.
- Spoznáme webovú službu zameranú na strojový preklad.
- Použijeme komponent `YandexTranslate` z kategórie *Media*.
- Budeme pracovať so zoznamami a lokálnou databázou.
- Vhodne využijeme syntézu a rozpoznávanie reči.
- Vytvoríme aplikáciu s viacerými obrazovkami.

Akú zaujímavú aplikáciu môžeme vytvoriť?

V aplikačnom obchode *Google Play* ľahko vyhľadáme rôzne mobilné aplikácie na podporu učenia sa cudzieho jazyka pre deti aj dospelých - slovníky, interaktívne cvičenia zamerané na slovnú zásobu, gramatiku, počúvanie, čítanie, hovorenie, podcasty, videá, hry pre jedného i viac hráčov. Mnohé populárne jazykové aplikácie integrujú viacero spôsobov učenia sa do jedného celku.

V našom prípade pôjde o aplikáciu, ktorá nám umožní:

- prekladať slová alebo slovné spojenia zo slovenčiny do anglického jazyka a naopak,
- zvoliť si medzi textovým alebo hlasovým ovládaním (slovo na preklad budeme môcť aj vysloviť, nielen napísať),
- zvoliť si medzi textovým a zvukovým výstupom prekladu (použijeme syntézu reči).
- zaznamenávať si vety s výskytom kľúčového slova do lokálnej databázy.

Príklady viet, v ktorých sa slovo používa v konkrétnom kontexte, sú veľmi užitočnou pomôckou pri trénovaní jazykových zručností (rozširovanie slovnej zásoby, zdokonaľovanie výslovnosti, plynulosť a gramatická správnosť vo vyjadrovaní).

Ako budeme postupovať pri tvorba aplikácie?

V našom projekte budeme musieť postupne vyriešiť rôzne podproblémy:

- navrhneme vzhľad aplikácie
  - zabezpečíme, aby texty zobrazené v rozhraní korešpondovali so smerom prekladu
  - smer prekladu bude možné ľahko zmeniť, napr. potrasením tabletu
- umožníme používateľovi zadať vstupné slovo
  - napísaním do textového poľa alebo
  - rozpoznaním hovorenej reči
- poskytneme používateľovo preklad
  - s využitím webovej služby zameranej na strojový preklad
  - výstup prekladu okrem zobrazenia aplikácia vysloví nahlas

- rozšírime aplikáciu o ďalšiu obrazovku určenú na prácu s databázou viet
- navrhneme a implementujeme ďalšie rozšírenia aplikácie

Niektoré komponenty, ktoré budeme potrebovať, sme už použili v malých aplikáciách či iných projektoch:

- vizuálne komponenty a správcov rozvrhnutia,
- `AccelerometerSensor`,
- `TextToSpeech`,
- `SpeechRecognizer`,
- lokálnu databázu `TinyDB`.

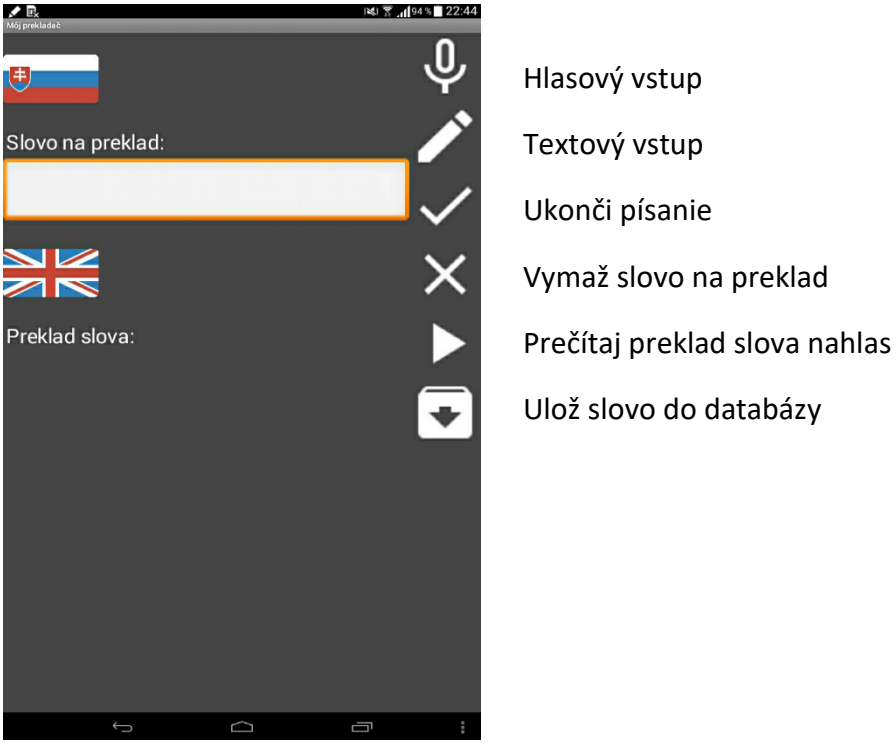
Na preklad slova využijeme webovú službu, s ktorou budeme komunikovať pomocou komponentu `YandexTranslate` z kategórie *Media*.

Úloha 1

Navrhnite vzhľad hlavnej obrazovky aplikácie, na ktorej budeme zadávať slová na preklad, získavať výsledok prekladu (a neskôr tiež umožníme otvorenie ďalšej obrazovky súvisiacej s databázou).

Ovládacie prvky aplikácie by mali byť rozmiestnené prakticky. Profesionálny dojem dosiahneme použitím vhodnej sady ikon. Ak chceme meniť smer prekladu (napr. potrasením tabletu), musíme aktuálnej situácii prispôsobiť aj vzhľad aplikácie.

Obrázok 4.3.1 obsahuje príklad rozloženia potrebných vizuálnych komponentov. V ľavej časti sú použité komponenty `Image`, `Label` a `Text`, v pravej časti tlačidlá s obrázkami nastavenými vo vlastnosti `Button.Image`.



Obr. 4.3.1 Návrh hlavnej obrazovky aplikácie

Úloha 2

Používateľ bude aplikáciu ovládať pomocou tlačidiel. Môže sa rozhodnúť pre hlasový vstup alebo písanie na klávesnici. V používateľskom rozhraní aplikácie sú aj tlačidlá na potvrdenie vstupu (neskôr ním vyvoláme zobrazenie prekladu), vymazanie vstupného textového poľa a prečítanie prekladu zadaného slova nahlas. Naprogramujte reakcie na udalosti vznikajúce pri stláčaní jednotlivých tlačidiel. Buďte dôslední pri testovaní používateľského rozhrania aplikácie.

Úloha 3

Doprogramujte hlavnú funkcionalitu aplikácie, ktorou je získanie prekladu zadaného slova. Výstup prekladu sa zobrazí na obrazovke. Po stlačení tlačidla ho aplikácia prečíta nahlas.

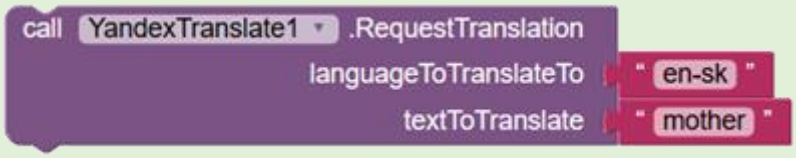
Vysvetlíme si

Na prekladanie slov zo zdrojového do cieľového jazyka použijeme komponent `YandexTranslate` z kategórie *Media*. Komponent potrebuje pre svoju funkčnosť internetové pripojenie, keďže pri každej žiadosti o preklad komunikuje s webovou službou Yandex – https://translate.yandex.com/ (MIT, 2020), (Yandex Translate, 2020).

Mnohé webové a mobilné aplikácie, ktoré bežne používame, sú založené na spracúvaní a remixovaní výstupov získaných od iných **webových služieb**, napr. Google Maps, Facebook, Twitter, YouTube, Flickr a pod.). Vývojári ich môžu využívať prostredníctvom verejného API (*Application Programming Interface,* rozhranie na programovanie aplikácií).

V dokumentácii si naštudujú, akým spôsobom je možné službe posielať **požiadavky** a v akom formáte sú **odpovede**, ktoré služba aplikácii vracia (napr. JSON, XML). Niektoré služby vyžadujú, aby vývojár najprv požiadal o pridelenie jedinečného kľúča (tzv. API key), na základe ktorého je možné aplikáciu identifikovať a sledovať, akým spôsobom službu využíva (počet prístupov, objem požadovaných dát a pod.).

App Inventor poskytuje špeciálny komponent, vďaka ktorému sa nemusíme zaoberať detailami komunikácie s webovou službou Yandex. Po vložení komponentu do aplikácie nenastavujeme žiadne vlastnosti. Požiadavku na preklad realizujeme zavolaním metódy, napr.:



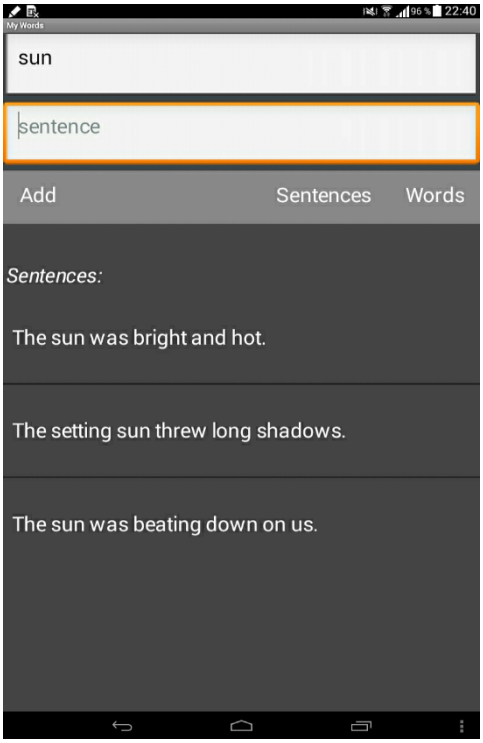
V parametroch metódy `RequestTranslation` určíme text na preloženie `textToTranslate` a jazyk, do ktorého chceme prekladať. V ukážke vyššie sme použili reťazec `"en-sk"`, čo znamená, že systém bude prekladať slovo `"mother"` z anglického do slovenského jazyka. Ak by sme prefix `"en-"`, vynechali a uviedli len kód cieľového jazyka, systém by sa najprv pokúsil zdrojový jazyk identifikovať sám. Dvojznakové kódy podporovaných jazykov nájdeme na stránke služby Yandex.

Preklad sa v aplikácii realizuje asynchrónne, na pozadí. Jeho ukončenie vygeneruje udalosť `YandexTranslate.GotTranslation`, na ktorú môžeme zareagovať. Výsledok prekladu získame z parametra `translation` (v našom príklade pôjde o reťazec `"matka"`).

Parameter `responseCode` obsahuje hodnotu, ktorá je dôležitá pre overenie úspešnosti prekladu. Ak má tento parameter hodnotu rôznu od 200, v komunikácii s webovou službou nastala chyba a preklad nie je k dispozícii. Význam kódov reprezentujúcich rôzne chybové stavy tiež nájdeme na stránke služby.

Úloha 4

Doplňte do projektu ďalšiu, samostatnú obrazovku (komponent `Screen`), ktorá sa v aplikácii otvorí po stlačení tlačidla dostupného na úvodnej obrazovke. Druhá obrazovka má používateľovi umožniť pridávanie, vyhľadávanie a editovanie príkladov viet obsahujúcich kľúčové slovo.



Obr. 4.3.2 Zobrazenie príkladov viet pre zadané kľúčové slovo (Michaličková, 2016)

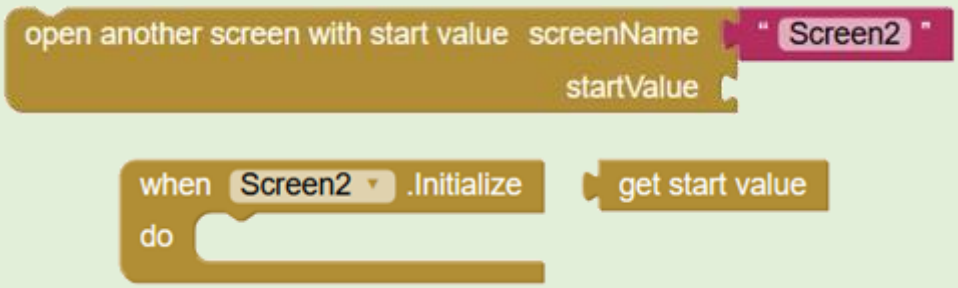
Pri navrhovaní používateľského rozhrania aplikácie a programovaní zvážte, že:

- do databázy chceme zapisovať slová a vety v anglickom jazyku, rozhranie druhej obrazovky by preto malo byť v angličtine,

- pri inicializácii druhej obrazovky je vhodné pripraviť ako kľúčové slovo posledné anglické slovo, s ktorým sa pracovalo na úvodnej obrazovke,
- zoznamy slov a viet pre zvolené slovo môžeme zobrazovať v komponente `ListView`,
- komponent `Notifier` je užitočný pri zobrazovaní správ a upozornení pre používateľa.

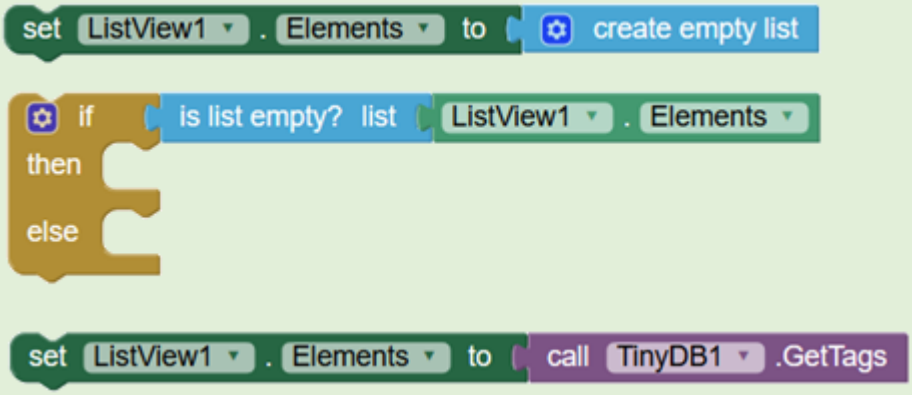
Pomôcky

Z jednej obrazovky môžeme otvoriť inú obrazovku a odovzdať jej hodnotu:



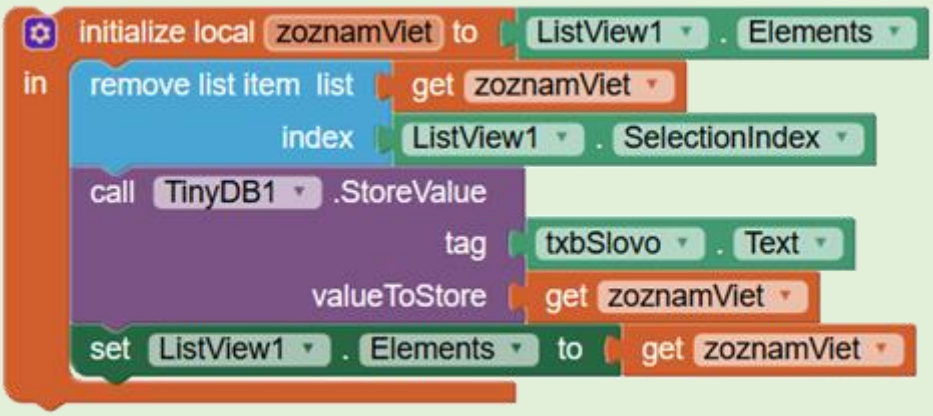
Ak sú údaje uložené v zozname, môžeme tento zoznam použiť ako zdroj dát pre komponent `ListView`.

S vlastnosťou `ListView.Elements` pracujeme ako so zoznamom, môžeme overiť či je prázdny, pridávať a odoberať prvky. Po každej operácii však nesmieme zabudnúť na synchronizáciu s lokálnou databázou `TinyDB`:

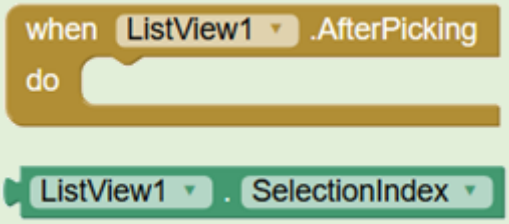


Pri získavaní zoznamu viet pre kľúčové slovo zadané v textovom poli sa nám zíde lokálna premenná:

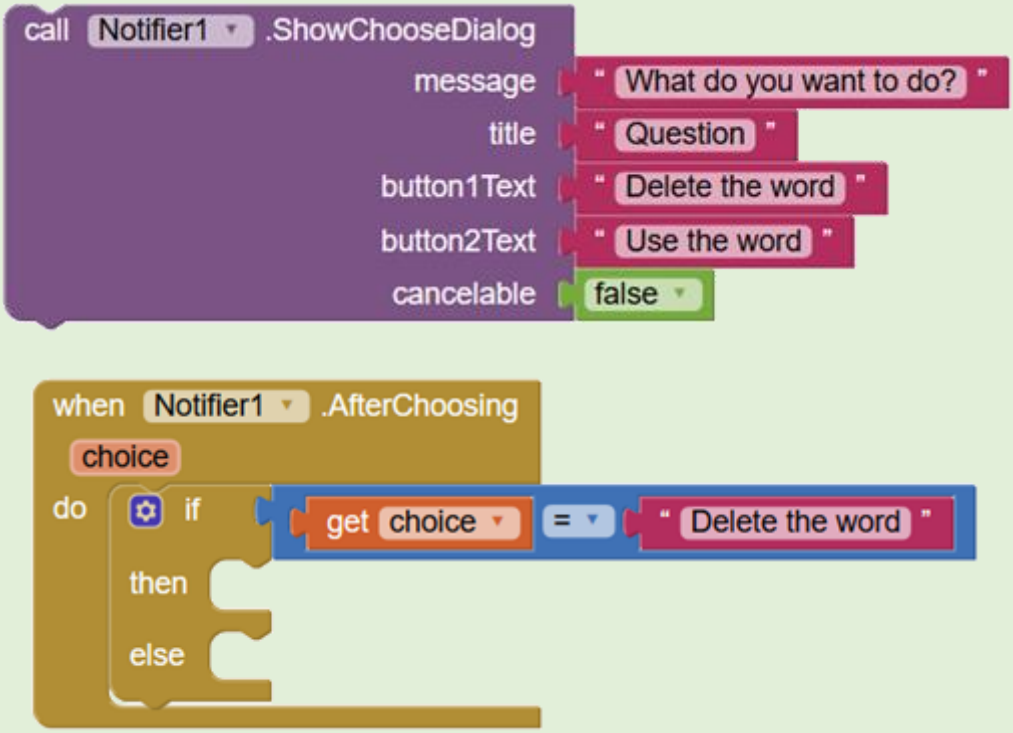
Podobne aj pri ukladaní aktualizovaného zoznamu viet do databázy (napr. po odstránení zvolenej vety) :



Ak chceme reagovať na výber slova zo zoznamu zobrazeného v komponente `ListView`, musíme naprogramovať reakciu na udalosť `ListView.AfterPicking`:



Ak chceme, aby používateľ potvrdil alebo zvolil zrealizovanie niektorej operácie (napr. vymazanie slova z databázy), zobrazíme dialógové okno a naprogramujeme reakciu na udalosť `Notifier.AfterChoosing`:



Ako vylepšiť či rozšíriť našu aplikáciu?

Základnú verziu aplikácie môžeme vylepšiť rôznym spôsobom, prispôsobiť si ju vlastným potrebám, napr.:

- pridať podporu pre viaceré jazyky,
- porovnávať vlastnú výslovnosť v nahrávke (použiť komponent `SoundRecorder`) s výstupom syntetizátora reči,
- naplniť databázu konverzačnými frázami, umožniť vyhľadávanie a hlasový výstup,
- exportovať obsah databázy do textového súboru,
- využiť obsah databázy ako zdroj dát pre minihru zameranú na precvičovanie slovnej zásoby alebo gramatiky,
- umožniť ovládanie celej aplikácie hlasom,
- upraviť GUI aplikácie, pridať ďalšie obrazovky atď.

Zamyslime sa, čo sme sa naučili

- Pomocou komponentu `YandexTranslate` vieme komunikovať s webovou službou zameranou na strojový preklad.
- V aplikáciách sme schopní vhodne využívať syntézu a rozpoznávanie reči,
- Dokážeme vytvárať aplikácie s viacerými obrazovkami a lokálnou databázou.

4.4 Spoločenská hra pre tablet

Kľúčové slová

spoločenská hra, ovládanie hry pomocou akcelerometra, webová databáza TinyWebDB, tímový projekt

Čo sa naučíme a čo si precvičíme

- Vytvoríme spoločenskú hru pre tablet.
- Údaje pre hru získame zo vzdialenej databázy, s ktorou budeme komunikovať pomocou komponentu `TinyWebDB`.
- Priebeh hry budeme riadiť nakláňaním tabletu vpred a vzad (pomôže nám akcelerometer).
- Pri tvorbe aplikácie budeme pracovať v tímoch a naučíme sa, ako je možné tímový projekt realizovať v App Inventore.
- Hotovú aplikáciu otestujeme v praxi (zahráme sa).

Spoločenskou hrou rozumieme hru pre dvoch a viac hráčov. Hráči sú pri hre fyzicky prítomní na jednom mieste, súperia proti sebe individuálne alebo v tímoch (hrajú sa spolu, relaxujú, zabávajú sa). Existujú rôzne typy spoločenských hier: doskové, kartové, slovné, papierové, konštrukčné. Pri niektorých hrách potrebujeme viacej pomôcok (hrací plán, hracie kocky, figúrky), pri iných menej (hracie karty, papier a pero) alebo dokonca žiadne. Pri spoločenskej hre sa nám môže zísť aj tablet (napr. na cestách). Dokáže ľahko zastúpiť hraciu kocku, kresliace plátno, zápisník, stopky a pod. Niektoré spoločenské hry môžu byť však na použití tabletu a jeho špecifických vlastnostiach aj založené. V takomto prípade hru riadi mobilná aplikácia: automaticky zobrazuje úlohy a vyhodnocuje úspešnosť. Zmysluplná interakcia s tabletom prispieva k dynamike a atraktívnosti hry. Zaujímavým spestrením býva tiež multimediálny výstup (zvukové a grafické efekty).

Otázky na zamyslenie

Ktorú spoločenskú hru ste sa hrávali/sa hrávate najčastejšie doma/v škole/na výletoch?

Uveďte príklad doskovej, kartovej, slovnej, papierovej a konštrukčnej hry.

Ktoré z vymenovaných spoločenských hier má zmysel hrať s vyžitím tabletu?

Použili ste už pri spoločenskej hre tablet?

Akú zaujímavú aplikáciu môžeme vytvoriť?

Naprogramujeme tabletovú verziu spoločenskej hry známej pod názvami *Šarády*, *Aktivity* alebo *Kartičky* (Warner Bros, 2016). Ide o *slovnú hru* pre 2 hráčov. Jeden hráč má k dispozícii kartičky so slovami a opisuje ich druhému hráčovi, ktorý má slovo uhádnuť. Hra má časový limit, cieľom oboch hráčov je, aby hádajúci hráč uhádol čo najviac slov. Hra je zaujímavejšia, keď proti sebe súperia viaceré dvojice hráčov. Pravidlá pre opisovanie slov môžu byť rôzne a ovplyvňujú obťažnosť a zábavnosť hry: hráč nesmie použiť v slovnom opise základ slova, musí hovoriť v cudzom jazyku, musí pri opisovaní spievať, veršovať, imitovať zvuky a pod. Použitie tabletu zefektívni organizačnú stránku hry: nebudeme musieť vopred pripravovať žiadne

papierové kartičky ani s nimi pri hre manipulovať, nemusíme merať čas ani počítať skóre. Na výber bude viacero kategórií slov, databázu slov bude možné upravovať a rozširovať.

Ako budeme postupovať pri tvorba aplikácie?

V našom projekte sa dajú ľahko identifikovať dva problémy, ktoré je možné vyriešiť nezávisle. Budeme preto pracovať v dvojčlenných tímoch:

Prvý člen tímu bude zodpovedný za to, aby si aplikácia po každom spustení z webovej databázy stiahla aktuálne dostupné kategórie slov a z kategórie, ktorú si hráč vyberie, uložila všetky slová na hádanie do lokálnej databázy.

Pripravené dáta použije **druhý člen tímu** vo svojej časti riešenia. Jeho úlohou bude zabezpečiť riadenie priebehu hry, jej riadne ukončenie a vyhodnotenie. Hra bude prebiehať takto: Hráč, ktorý má hádať, drží tablet v rukách tak, aby naň videl hádajúci hráč (t. j. zvislo, displejom smerom k spoluhráčovi). Ďalšie slovo sa na tablete zobrazí po uhádnutí slova (hádajúci hráč nakloní tablet na chvíľu vpred) alebo vtedy, keď sa hádajúci hráč vzdá a rozhodne sa radšej pre hádanie ďalšieho slova (v tejto situácii tablet nakloní vzad). Po skončení časového limitu sa zobrazí počet úspešne uhádnutých slov.

Prvý člen tímu bude pracovať na hlavnej obrazovke aplikácie (komponent `Screen1`). Druhý člen tímu vytvorí druhú obrazovku aplikácie (komponent `Screen2`). Z prvej obrazovky sa bude otvárať druhá obrazovka. Z druhej obrazovky sa bude dať vrátiť na prvú (hlavnú) obrazovku. Čiastkové projekty členov tímu sa v záverečnej fáze spoja do jedného spoločného projektu pomocou nástroja **App Inventor Merger**.

Niektoré komponenty, ktoré budeme potrebovať, sme už použili v malých aplikáciách či iných projektoch:

- vizuálne komponenty a správcov rozvrhnutia,
- `AccelerometerSensor`,
- `Notifier`,
- `Sound`,
- `Spinner`,
- lokálnu databázu `TinyDB`.

Dáta hry chceme uložiť vo vzdialenej databáze. Mobilná aplikácia k nim bude pristupovať prostredníctvom webovej služby, s ktorou budeme v App Inventore komunikovať pomocou komponentu `TinyWebDB` z kategórie *Storage*.

Najprv si vyskúšame, ako budeme zlučovať viaceré projekty do jedného:

Úloha 1

Vytvorte aplikáciu s dvomi obrazovkami. Na prvej obrazovke nech je obrázok psa, na druhej obrazovke nech je obrázok mačky. Keď na obrazovke so psom stlačíme tlačidlo, otvorí sa druhá obrazovka, na ktorej sa objaví alebo ozve správa pre mačku. Po návrate na hlavnú obrazovku uvidíme alebo si vypočujeme, čo na oplátku mačka odkazuje psovi. Správy môžete generovať aj náhodne. Pracujte v dvojčlennom tíme. Každý člen tímu nech je zodpovedný za

implementáciu jednej z obrazoviek. Na záver svoje projekty zlúčte do jedného a aplikáciu dokončite a otestujte.

Vysvetlíme si

V praxi programátori väčšinou pracujú v tímoch. Aj v App Inventore je možné, aby jednotlivé obrazovky aplikácie vyvíjali nezávisle dvaja alebo viacerí programátori. Jednotlivé projekty sa v záverečnej fáze dajú pomocou jednoduchého desktopového nástroja zlúčiť do finálneho produktu (MIT, 2020).

Členovia tímu sa musia na začiatku **dohodnúť na konvenciách**, ktoré budú dodržiavať pri pomenúvaní obrazoviek, zdieľaných komponentov (takým je napr. lokálna databáza `TinyDB`) a súborov médií, aby sa mohli v zdrojovom kóde svojich projektov na ne odvolávať. **Hlavná obrazovka aplikácie** sa nedá v App Inventore premenovať, preto sa **vždy musí volať** *Screen1*.

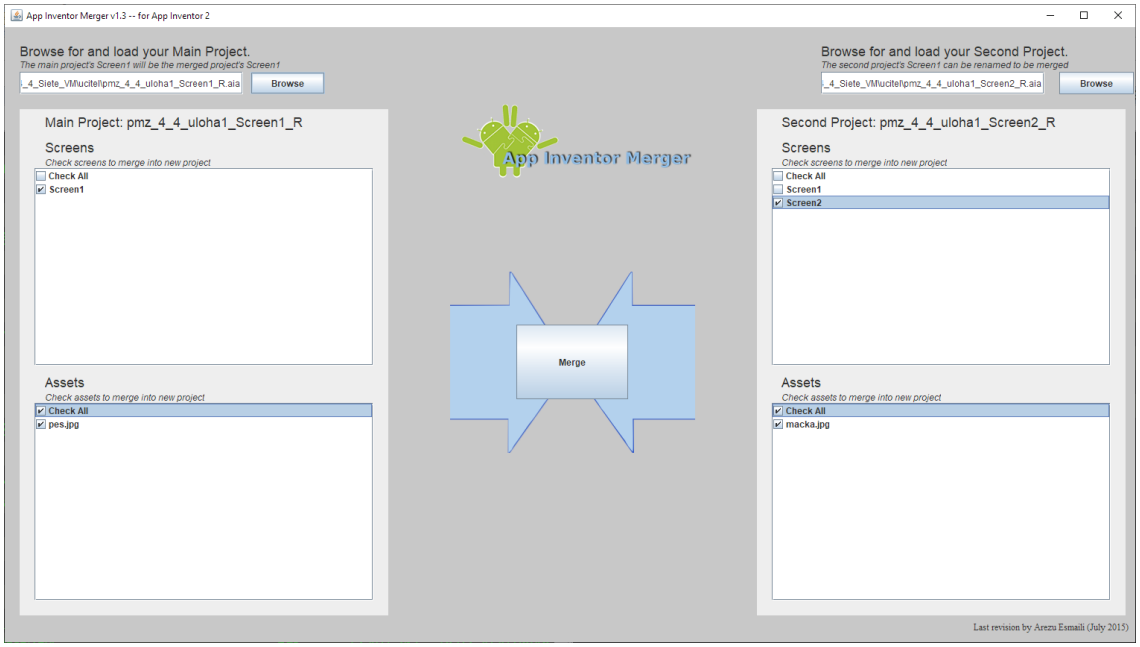
Uvažujme dvoch vývojárov, ktorých úlohou je naprogramovať aplikáciu s dvoma obrazovkami. Prvý z nich bude pracovať vo svojom projekte na hlavnej obrazovke aplikácie s názvom *Screen1*. Druhý z nich si vo vlastnom projekte **pridá novú obrazovku** a pomenuje ju napr. *Screen2.* Navrhne jej dizajn a naprogramuje jej funkcionalitu. Svoju hlavnú obrazovku s názvom *Screen1* môže ignorovať, nanajvýš na ňu umiestni pomocné tlačidlo na otvorenie druhej obrazovky, aby sa k nej dostal pri testovaní svojej časti aplikácie. Táto pomocná obrazovka sa pri zlučovaní projektov vynechá, nie je podstatné, čo obsahuje.

Po dokončení čiastkových riešení každý z vývojárov exportuje svoj projekt z cloudového úložiska do počítača. Po spustení nástroja App Inventor Merger (súbor aplikácie má názov *AI2MergerApp.jar*) sa zobrazí okno (Obrázok 4.4.1), v ktorom vyberáme projekty na zlúčenie. Po načítaní projektov (prvý projekt sa pokladá za hlavný projekt s obrazovkou *Screen1*) uvidíme zoznam komponentov a mediálnych súborov, ktoré ich tvoria. V zoznamoch pre jednotlivé projekty zvolíme, čo má byť súčasťou cieľového projektu po zlúčení. Niekedy totiž používame rovnaké obrázky a zvuky na viacerých obrazovkách. Alebo zatiaľ nechceme do výsledného projektu zahrnúť všetky obrazovky, na ktorých v projekte pracujeme.

Zlúčenie zrealizujeme kliknutím na tlačidlo *Merge (Zlúčiť)*. Vznikne nový projektový súbor *.aia*, ktorý v dialógovom okne vhodne pomenujeme a uložíme na disk. Následne ho importujeme do prostredia App Inventor (je jedno, ktorý z vývojárov), kde s ním môžeme ďalej pracovať.

Dokumentáciu k nástroju **App Inventor Merger** a súbor *AI2MergerApp.jar* na stiahnutie sa nachádza tu: http://appinventor.mit.edu/explore/resources/ai2-project-merger.html.

AI2MergerApp.jar je desktopová aplikácia napísaná v Jave, je preto potrebné, aby bola pred jej spustením v počítači nainštalovaná podpora pre Javu (*JRE, Java Runtime Environment*). Príslušný inštalačný súbor je na stiahnutie napr. tu:
https://www.java.com/en/download/



Obr. 4.4.1 Zlučovanie projektov v okne nástroja App Inventor Merger

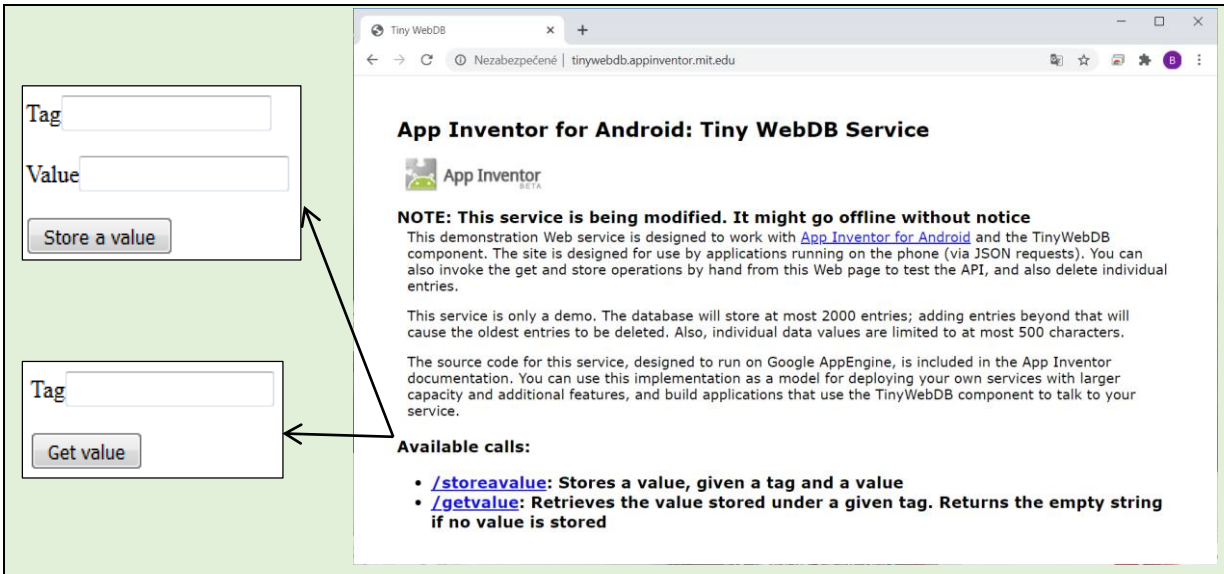
Úloha 2

Vyberte si kategóriu slov na hádanie (napr. Povolania, Hudobné skupiny, Rozprávkové bytosti a pod.). Zapíšte si na papier zoznam slov, ktoré chcete do tejto kategórie zaradiť. Uložte tieto údaje do vzdialenej databázy s využitím databázového komponentu `TinyWebDB`. Potom vytvorte aplikáciu, v ktorej údaje uložené v databáze načítate a zobrazíte.

Vysvetlíme si

S komponentom `TinyDB`, ktorý reprezentuje lokálne databázové úložisko aplikácie, sme už pracovali viackrát. Ak chceme, aby bola databáza umiestnená na serveri a mohli ju tak zdieľať viacerí používatelia (presnejšie inštancie rovnakej aplikácie nainštalované v mobilných zariadeniach rôznych používateľov), môžeme použiť komponent `TinyWebDB`.

Po vložení komponentu `TinyWebDB` do aplikácie zistíme, *že* má vo vlastnosti `ServiceURL` prednastavenú adresu http://tinywebdb.appinventor.mit.edu/. Ide o demo webovej služby s obmedzeniami na počet a veľkosť záznamov (Obrázok 4.4.2). Pre naše účely nám zatiaľ takéto riešenie postačí. Musíme si však uvedomiť, že rovnakú databázu, s ktorou budeme pracovať, používajú aj iní vývojári. Nie je preto vylúčené, že naše údaje niekto získa alebo prepíše. Stačí, že sa náhodou rozhodne použiť pri ukladaní hodnoty rovnaký *tag* (kľúč). Aby sa minimalizovalo toto riziko, zvyknú sa ako tagy voliť reťazce v tvare obrátenej doménovej adresy, napr.: `"sk.mojaskola.mojemeno.karticky.povolania"`.

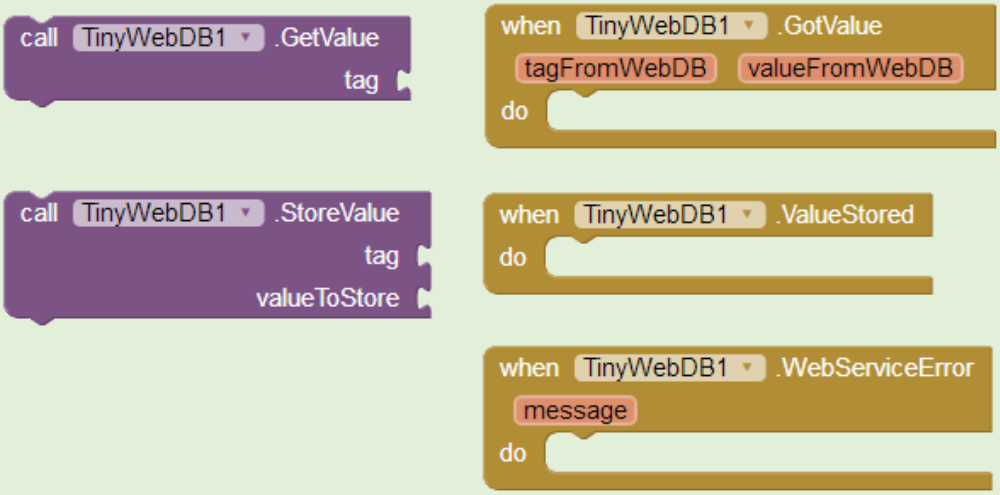


Obr. 4.4.2 Webová služba Tiny WebDB s formulármi pre uloženie a získanie údajov (Michaličková, 2016)

Návod na vytvorenie vlastnej webovej služby typu *Tiny WebDB* (bez spomínaných obmedzení demoverzie) nájdeme na: http://ai2.appinventor.mit.edu/reference/other/tinywebdb.html).

Údaje sa do databázy dajú ukladať aj manuálne, s využitím formulára vo webovom rozhraní služby (Obrázok 4.4.2). V našom prípade zvolíme vhodný `Tag` a do políčka `Value` zapíšeme zoznam slov oddelených čiarkami uzavretý v hranatých zátvorkách. Ak chceme overiť uloženie hodnoty v databáze, môžeme to urobiť zadaním tagu do vstupného poľa druhého formulára. Ak v databáze so zadaným tagom nie je asociovaná žiadna hodnota, vráti sa ako výsledok dopytu prázdny reťazec.

V App Inventore máme k dispozícii dve metódy a tri udalostné bloky súvisiace s komponentom `TinyWebDB`:



Ak poznáme tag, vieme z databázy *získať hodnotu*, ktorá je s ním asociovaná. Do databázy môžeme samozrejme *ukladať ďalšie záznamy*. V aplikácii sa dá zareagovať na úspešné získanie aj na uloženie hodnoty. V prípade, že komunikácia s webovou službou zlyhala, vznikne udalosť `TinyWebDB.WebServiceError`.

Úloha 3

Na detegovanie pohybu tabletu pri nakláňaní tabletu zo zvislej polohy vpred alebo vzad, resp. z okrajových polôh naspäť do zvislej polohy využijeme v našej hre senzor zrýchlenia, t. j. komponent `AccelerometerSensor`. Vytvorte jednoduchú aplikáciu na vypisovanie aktuálnych hodnôt, ktoré akcelerometer meria. **Uvažujte o tom, ako rozpoznať, aký pohyb používateľ s tabletom práve urobil**.

Poznámka

V aplikácii najprv nastavte napevno orientáciu obrazovky na *Landscape*. Komponentu `AccelerometerSensor` zmeňte v režime *Designer* vlastnosť *Legacy Mode* na `true`. Vypnete tým dodatočnú úpravu hodnôt poskytovaných aplikácii systémom Android, ktorú autori App Inventoru robia preto, aby zabezpečili rovnaké fungovanie aplikácie pre tablety aj smartfóny. Telefóny mávajú totiž východiskovú orientáciu nastavenú na *Portrait*.

Zrealizujte niekoľko experimentov s nakláňaním tabletu. Sledujte ako sa menia hodnoty parametrov `xAccel`, `yAccel`, `zAccel`, ku ktorým máme prístup v udalostnom bloku `when AccelerometerSensor1.AccelerationChanged`:

Experimenty s nakláňaním tabletu				
tablet je zvislo, držíme ho displejom otočeným od seba	zo zvislej polohy nakláňame tablet vpred až do vodorovnej polohy, v ktorej je tablet obrátený displejom nadol	z vodorovnej polohy s displejom natočeným nadol vraciame tablet späť do zvislej polohy	zo zvislej polohy nakláňame tablet vzad až do vodorovnej polohy, v ktorej je tablet obrátený displejom nahor	z vodorovnej polohy s displejom otočeným nahor vraciame tablet späť do zvislej polohy

Naprogramujeme aplikáciu **Kartičky** z úvodu kapitoly:

Úloha 4 (pre prvého člena tímu)

Navrhnite dizajn hlavnej obrazovky. Umožnite používateľovi vybrať zo zoznamu kategóriu slov na hádanie. Pre zvolenú kategóriu načítajte zoznam slov z webovej databázy. Uložte ho do lokálnej databázy aplikácie, ktorú budú zdieľať obe obrazovky. Pripravte tiež tlačidlo na spustenie hry. Dbajte však na to, aby bolo prístupné len v prípade, že slová na hádanie sú k dispozícii.

Úloha 5 (pre druhého člena tímu)

Naprogramujte automatické zobrazovanie slov a počítanie správnych odpovedí. Hru ovládajte nakláňaním tabletu vpred a vzad. Po skončení časového limitu alebo uhádnutí všetkých slov hru ukončite a oznámte výsledok. Po návrate na hlavnú obrazovku bude možné zvoliť inú kategóriu a spustiť ďalšie kolo hry.

Poznámka

V okamihu otvorenia druhej obrazovky už budú k dispozícii slová na hádanie. Pred spustením časomiery ale musíme mať istotu, že hráč drží tablet zvislo pred sebou tak, aby naň videl jeho spoluhráč. Až potom má zmysel hru odštartovať a zobraziť prvé slovo.

Úloha 6

Projekty členov tímu zlúčte do jedného celku, aplikáciu dokončite a otestujte.

V diskusii so spolužiakmi navrhnite pravidlá turnaja v hre **Kartičky**. Každý tím použije v súťaži vlastnú aplikáciu.

Poznámka

Aplikácie všetkých tímov by mali údaje zo vzdialenej databázy načítavať s použitím rovnakých tagov. Prefix identifikujúci projekt, napr. `"sk.mojaskola.karticky."`, je preto vhodné uložiť v globálnej premennej.

Pri dopytovaní sa na kategórie sa v aplikácii bude v tomto prípade používať tag:

`sk.mojaskola.karticky.kategorie`

Pri dopytovaní sa na slová z konkrétnej kategórie sa v aplikácii použije napr. tag:

`sk.mojaskola.karticky.povolania`

Údaj o časovom limite je tiež vhodné uložiť do globálnej premennej, keďže pri realizovaní súťaže tímov musí byť pre všetkých zúčastnených rovnaký.

Ako vylepšiť či rozšíriť našu aplikáciu?

Naším cieľom bolo vytvorenie jednoduchej, ale funkčnej a v praxi použiteľnej aplikácie. Môžete zvážiť ďalšie vylepšenia svojej verzie, napr.:

- upraviť grafické používateľské rozhranie aplikácie,
- zabezpečiť, aby sa hra dala hrať aj bez internetového pripojenia (aplikácia by mala mať v tomto prípade v lokálnej databáze všetky kategórie slov a realizovať synchronizáciu so vzdialenou databázou len na požiadanie),
- umožniť nastavovanie niektorých parametrov (napr. časového limitu, počtu kôl súťaže, náhodný výber kategórie, miešanie poradia slov a pod.),
- evidovať priebežné skóre pre viac tímov, ktoré sa pri hre striedajú,
- odpočítavať odštartovanie hry po rozpoznaní správnej polohy tabletu,
- signalizovať blížiaci sa koniec hry,
- umožniť editovanie dát vo vzdialenej databáze priamo v aplikácii atď.

Zamyslime sa, čo sme sa naučili

- Prostredníctvom komponentu `TinyWebDB` vieme údaje ukladať do a čítať z databázy umiestnenej na vzdialenom serveri.
- Pomocou akcelerometra dokážeme detegovať nakláňanie tabletu do strán a údaje zo senzora využívať na riadenie priebehu hry.
- Nástroj *App Inventor Merger* vieme použiť pri tímovej práci a zlúčiť viaceré čiastkové riešenia do jedného projektu.

4.5 Kockový poker

Kľúčové slová

náhodné čísla, sprajty, animácia, ťahanie, pravidlá hry, komunikácia, Bluetooth

Čo sa naučíme a čo si precvičíme

- použiť generátor pseudonáhodných čísiel na generovanie náhodných javov v aplikácii,
- animovať a gestami ovládať grafické objekty – sprajty,
- vyhodnocovať stav hry podľa pravidiel,
- komunikovať s iným zariadením prostredníctvom technológie Bluetooth.

Čo zaujímavé môžeme zistiť?

Kockový poker je spoločenská hra. Podobne ako hry s kartami je nenáročná na prípravu, na hranie nám stačí 6 kociek a niečo na zaznamenávanie výsledkov hry. Hra je založená na náhode (hádže sa kockami), ale hráč v nej robí aj strategické rozhodnutia, či a ako pokračovať v hre, ktorými významne ovplyvňuje priebeh hry.

Existujú rôzne variácie pravidiel kockového pokra. Vo Wikipédii nájdeme takéto (Wikipédia, 2018):

Ľubovoľný počet hráčov hrá so šiestimi kockami. Hráč začína hru hodom všetkých kociek. Hod sa vyhodnocuje takto:

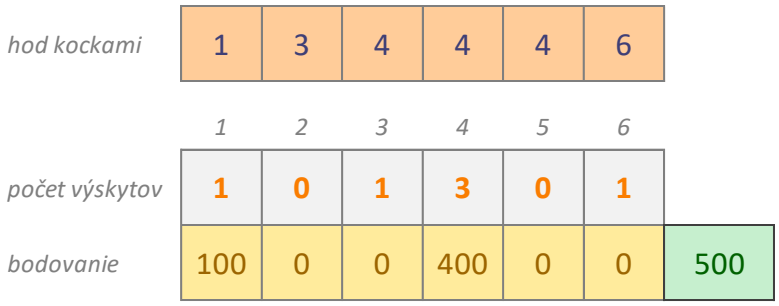
1. Ak padla trojica, násobí sa počet bodov na kocke stovkou, výnimkou sú tri jednotky, sú za 1000.
2. Každá ďalšia rovnaká kocka pri počte viac ako 3 v jednom hode body zdvojnásobuje.
3. Každá jednotka v počte menej ako 3 je za 100 bodov a každá päťka je za 50 bodov.
4. Postupka, teda čísla od 1 do 6, je za 2000 bodov.

Ak hráč v hode získal aspoň 50 bodov, so zvyšnými kockami môže hádzať ďalej. Ak pri nasledujúcom hode nič nezíska, stráca dovtedy nahraté body v danej hre, ak body získa, môže hádzať ďalej alebo si môže zapísať dosiahnutý výsledok. Ak všetkých šesť kociek získalo body, hráč môže znovu hádzať všetkými kockami, ale s rizikom, že príde o dosiahnuté body.

Hra končí v kole, v ktorom jeden z hráčov dosiahne dopredu dohodnutý súčet bodov (napríklad 5000, 10000 alebo 20000 bodov). Ak jeden z hráčov dosiahol dohodnutý počet bodov, zostávajúci hráči dokončia kolo a vyhráva ten, ktorý získal najviac bodov.

Úloha 1

Zoznámte sa s pravidlami kockového pokra. V elektronickom pracovnom liste **pmz_4_5_Kockovy poker_PL.xlsx** experimentujte so simuláciou hádzania kociek a sledujte vyhodnocovanie hodu. Zahrajte si kockový poker v skupine s ozajstnými kockami a vyskúšajte si stratégiu hry s viacerými hodmi.



Obr. 4.5.1 Príklad vyhodnotenia simulovaného hodu 6 kockami

Akú zaujímavú aplikáciu môžeme vytvoriť?

Hoci rekvizity na hranie kockového pokra nie sú objemné, predsa len nie je bežné, že človek nosí so sebou 6 hracích kociek. Problém pri hraní môže byť tiež s hádzaním kociek, ak nemáme na to vhodné prostredie. Mobilný telefón s aplikáciou môže simulovať náhodné hody kockami, asistovať pri vyhodnocovaní hry a zaznamenávaní výsledkov, a tak nahradiť všetky rekvizity potrebné k hre a uľahčovať administrovanie výsledkov.

Naprogramujme aplikáciu, ktorá bude:

- simulovať hod kockami,
- umožňovať rozhodovanie hráča o pokračovaní resp. ukončení hry,
- bodovo vyhodnocovať hru hráča: body za aktuálny hod, body za všetky hody v rámci jedného ťahu, celkový súčet bodov,
- zaznamenávať celkové skóre hráča.

Takáto aplikácia realizuje hru jedného hráča. Viacerí hráči môžu hrať každý so svojím telefónom a výsledky si evidovať každý vo svojom zariadení. Elegantnejšie riešenie kockového pokra ako spoločenskej hry je, ak mobilné aplikácie hráčov budú medzi sebou komunikovať a aplikácie budú riadiť priebeh hry a zaznamenávanie výsledkov skupiny hráčov. Naša aplikácia bude určená pre dvoch hráčov a bude:

- evidovať hráča na ťahu,
- blokovať hru pre hráča, ktorý nie je na ťahu,
- evidovať a zobrazovať výsledky oboch hráčov,
- oznamovať ukončenie hry a víťaza.

Ako budeme postupovať pri tvorbe aplikácie?

Najprv vytvoríme aplikáciu pre jedného hráča. Potom pridáme komunikáciu a riadenie hry pre dvoch hráčov.

Úloha 2

Vyskúšajte aplikáciu **kockovy_poker_v1.aia** a zistite, čo dokáže.

Potom preskúmajte kód aplikácie:

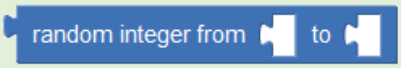
- Čo obsahuje globálna premenná `kocky` a v ktorom bloku programu sa jej priraďujú hodnoty? Na čo sa premenná v programe používa?

- Čo obsahuje globálna premenná `hodnotyKociek` a v ktorom bloku programu sa jej nastavujú hodnoty?

Vysvetlíme si

Hod kockou je náhodný jav, ktorý sa nedá predpovedať, možno iba určiť jeho pravdepodobnosť. Ak predpokladáme dokonalú kocku, tak pravdepodobnosť padnutia ľubovoľného čísla na kocke je rovnaká pre všetky čísla 1 až 6, a to 1/6. Algoritmicky sa dajú generovať postupnosti čísiel s vlastnosťami, ktoré zodpovedajú teoretickým pravdepodobnostiam. Takéto algoritmy sa nazývajú *generátory pseudonáhodných čísiel*.

V App Inventore sa generujú pseudonáhodné celé čísla zo zadaného intervalu pomocou funkcie



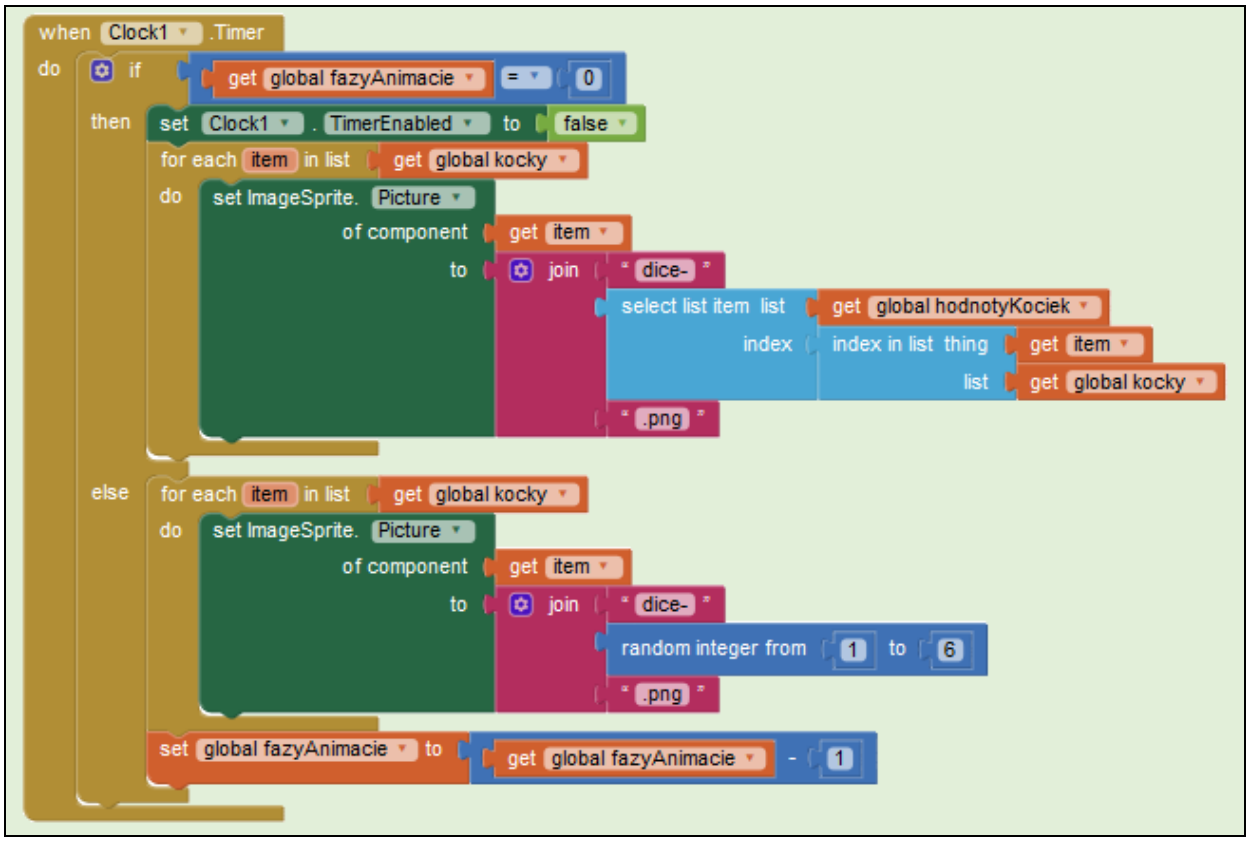
a pseudonáhodné reálne čísla z intervalu <0,1) pomocou funkcie

- Na čo slúži premenná `fazyAnimacie`?

Vysvetlíme si

Animácia je striedanie statických obrázkov tak, aby sa vytvoril dojem pohybu.

Striedanie obrázkov pri animácii robíme v projekte pomocou komponentu `Clock1` a jeho udalosti `Timer`. Pri hode kockami sa aktivuje časovač komponentu `Clock1` nastavením vlastnosti `TimerEnabled` na `true`. Komponent začne generovať v pravidelných intervaloch udalosti `Timer`. Pri každom tiknutí časovača sa náhodne nastavia obrázky hodených kociek a zníži sa hodnota premennej `fazyAnimacie` o 1. Po vynulovaní nastaveného počtu fáz animácie sa obrázok kocky nastaví podľa konečnej hodnoty uloženej v premennej `hodnotyKociek`, časovač komponentu `Clock1` sa deaktivuje nastavením vlastnosti `TimerEnabled` na `false` a animácia sa zastaví. Animuje sa hod všetkých kociek uložených v zozname `kocky` v cykle `for each item in list kocky`.



- Čo sa deje v aplikácii pri ťahaní a pri pustení kociek?

Vysvetlíme si

`ImageSprite` (sprajt) je vizuálny komponent, ktorý môže byť umiestnený v komponente `Canvas`. Môže sa pohybovať a reagovať na dotykové gestá.

Vybrané vlastnosti komponentu:

`Picture` – obrázok, ktorý definuje vzhľad sprajtu,

`Height`, `Width` – rozmery (výška, šírka) sprajtu,

`X`, `Y` – súradnice ľavého horného rohu sprajtu.

Vybrané udalosti:

`Dragged(startX, startY, prevX, prevY, currentX, currentY)` – ťahanie, parametre v obslužnej metóde udalosti udávajú pozíciu začiatku ťahania, predchádzajúcej a aktuálnej pozície sprajtu

`TouchDown(x, y)` – chytenie, parametre udávajú pozíciu, na ktorej bol sprajt uchopený

`TouchUp(x, y)` – pustenie, parametre udávajú pozíciu, na ktorej bol sprajt pustený

Vybraná metóda:

`MoveTo(x, y)` – premiestni sprajt na dané súradnice

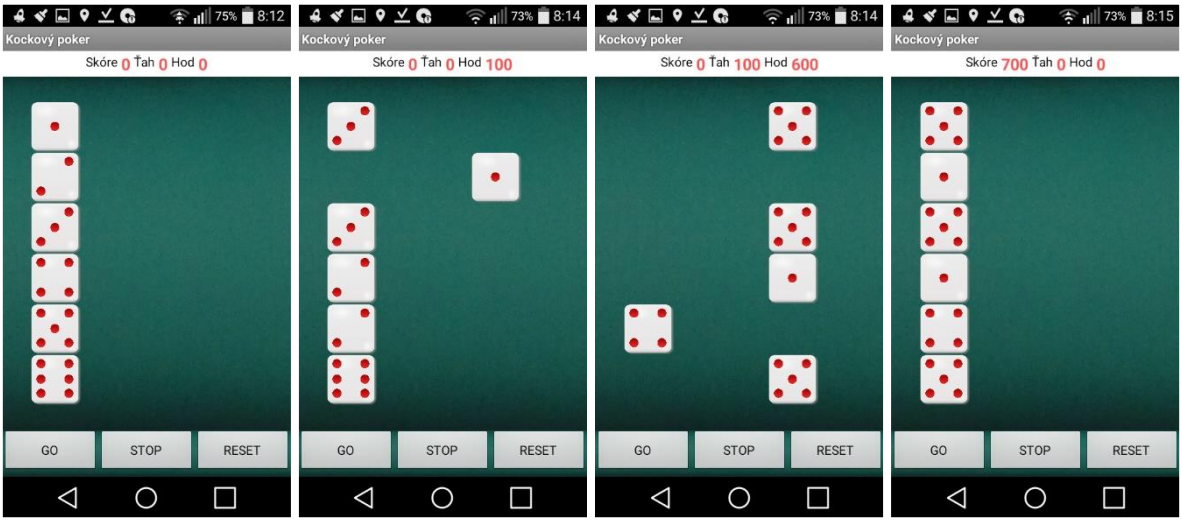
Úloha 3

Vyskúšajte aplikáciu **kockovy_poker_v2.aia**. Potom preskúmajte kód aplikácie.

- Aké nové funkcionality aplikácia obsahuje?

Na obrázku 4.5.2 sú snímky obrazovky, ktoré zachytávajú priebeh jednej hry:

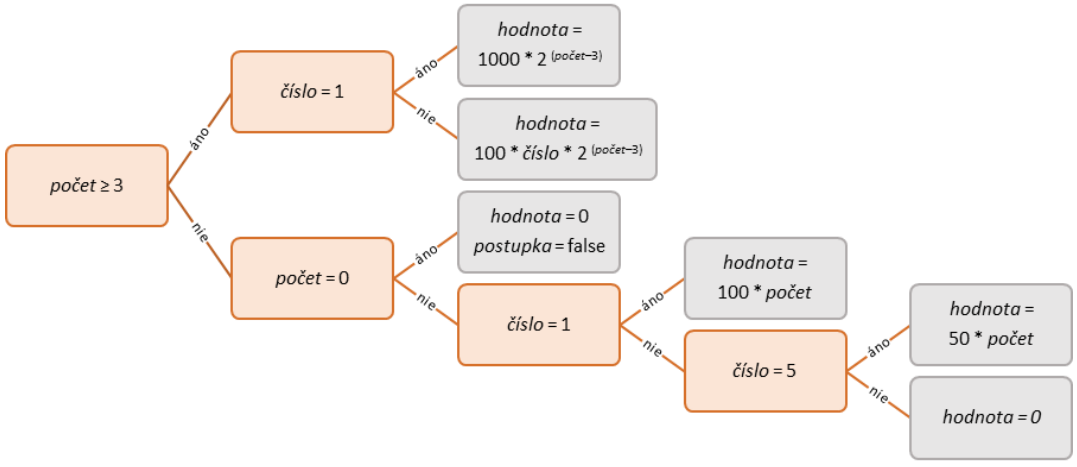
1. úvodné rozloženie kociek,
2. po prvom hode šiestimi kockami (stlačenie tlačidla **GO**): napravo kocka vybraná používateľom na bodovanie, naľavo kocky bez bodovej hodnoty, hodnota hodu je 100,
3. po druhom hode so zostávajúcimi piatimi kockami (po stlačení tlačidla **GO**): priebežná hodnota hry v tomto ťahu je 100 + hodnota aktuálneho hodu je 600
4. po ukončení ťahu (hry) stlačením tlačidla **STOP**: celkové skóre je 700



Obr. 4.5.2 Priebeh hry

- Ako sa vyhodnocuje hodnota hodu?

Na obrázku 4.5.3 je rozhodovací strom, podľa ktorého sa vyhodnocuje hodnota hodu v aplikácii. Zistite, ako sa vyhodnocuje postupka. Navrhnite svoj vlastný spôsob vyhodnocovania hodu.



Obr. 4.5.3 Rozhodovací strom pre vyhodnocovanie hodu

Úloha 4

Preskúmajte projekt **komunikacia_bluetooth.aia** – funkčnosť aplikácie a programový kód.

Aplikácia umožňuje komunikáciu medzi dvomi zariadeniami pomocou technológie Bluetooth (Pura Vida Apps, 2010-2020).

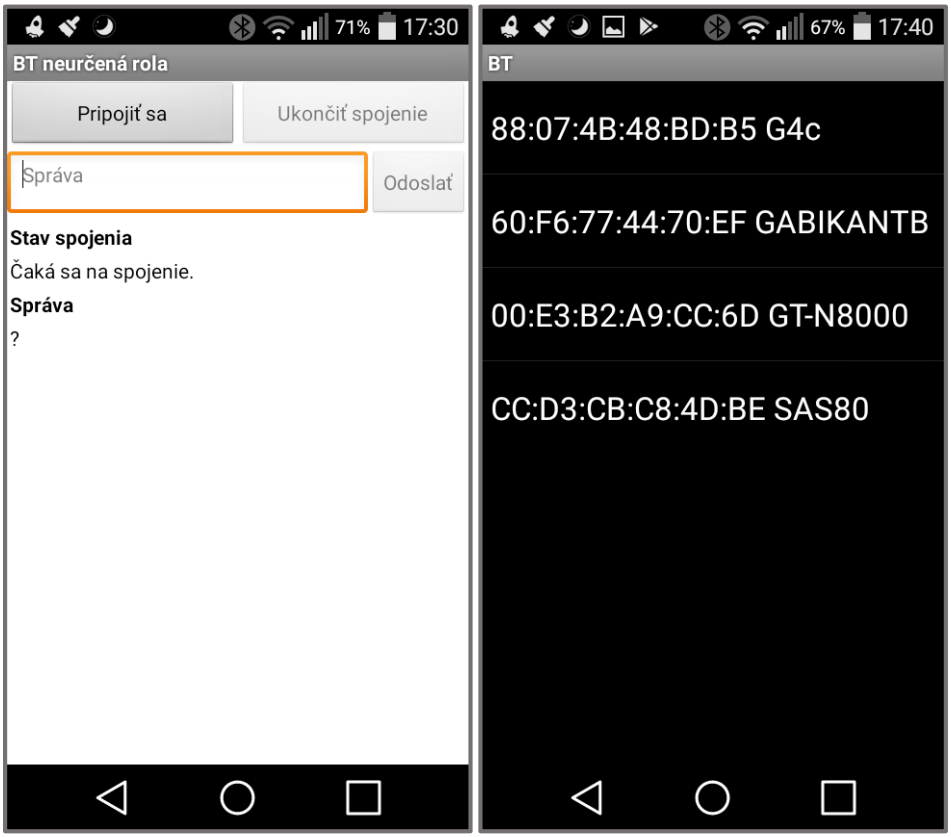
Vysvetlíme si

Bluetooth je technológia pre bezdrôtovú komunikáciu medzi dvomi alebo viac digitálnymi zariadeniami. Pri spojení komunikujúcich zariadení sa využíva architektúra klient-server. Zariadenie, ktoré začína komunikáciu, je klient, zariadenie, ktoré prijíma žiadosť o komunikáciu, je server.

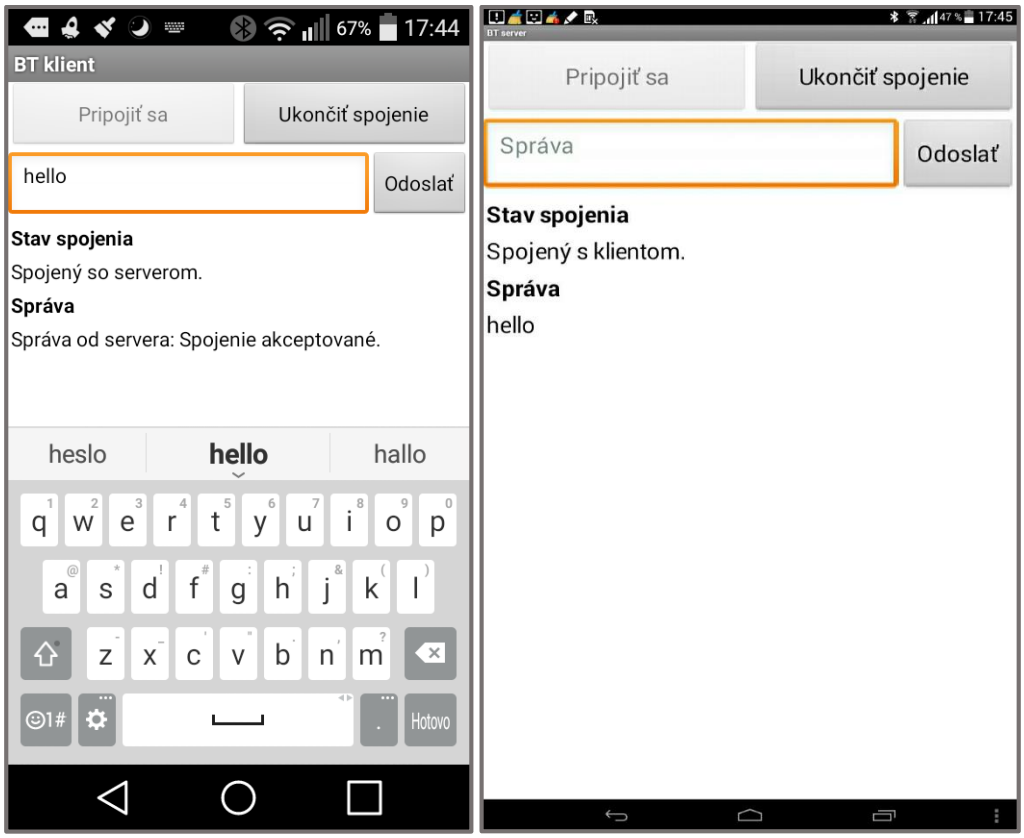
V App Inventore slúžia na vytvorenie spojenia bluetooth komponenty `BluetoothClient` a `BluetoothServer` zo skupiny komponentov *Connectivity*.

Na otestovanie funkčnosti aplikácie budeme potrebovať dve mobilné zariadenia. Príkazom **Build** vytvoríme Inštalačný balík aplikácie (.apk) a do oboch zariadení nainštalujeme a spustíme ukážkovú aplikáciu komunikacia_bluetooth. V oboch zariadeniach zapneme bluetooth a zariadenia spárujeme.

Spojenie začína jedno zo zariadení stlačením tlačidla s nápisom **Pripojiť**. Toto zariadenie bude vystupovať v úlohe klienta. Zo zoznamu sa vyberie zariadenie, ku ktorému sa chceme pripojiť. To bude v úlohe servera. Ak server akceptuje žiadosť klienta, spojenie sa úspešne nadviaže. Funkčnosť spojenia môžeme vyskúšať odoslaním správ z jedného do druhého zariadenia.

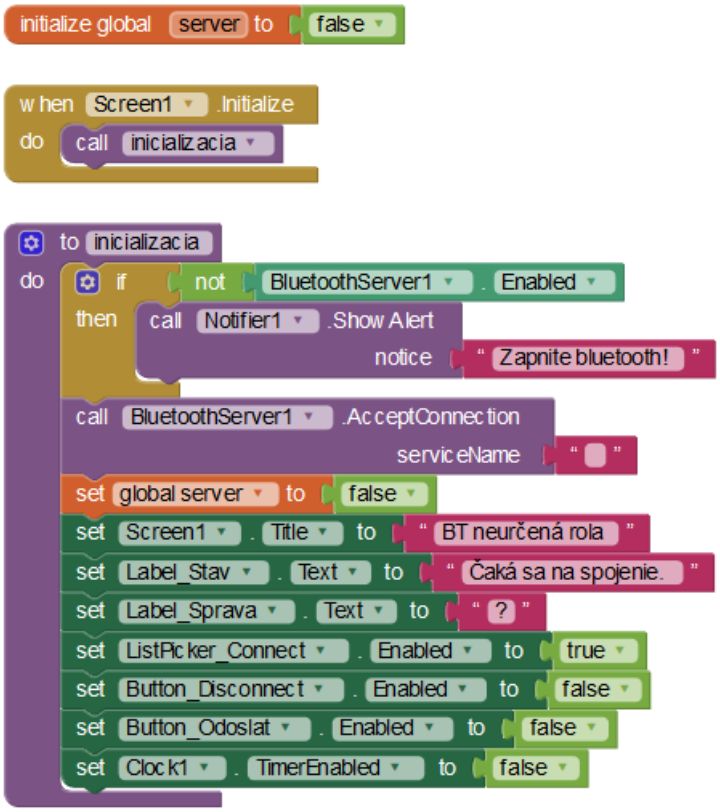


Obr. 4.5.4 Iniciovanie spojenia (tlačidlo Pripojiť), výber zariadenia na spojenie zo zoznamu



Obr. 4.5.5 Aplikácia v režime klient a server

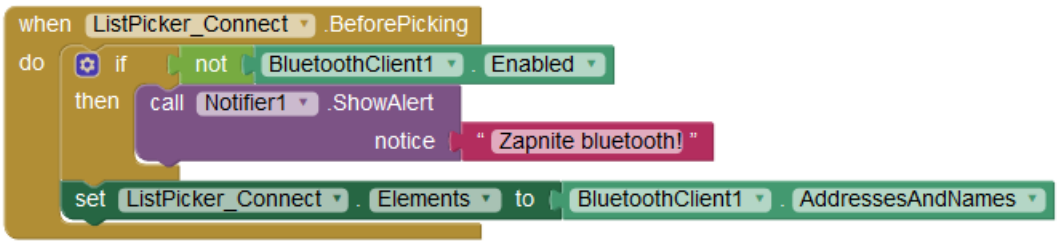
Preskúmajme program. Pri inicializácii aplikácia skontroluje, či je Bluetooth zapnutý (vlastnosť `BluetoothServer.Enabled`) a nastaví serverovú časť na akceptovanie spojenia metódou `BluetoothServer.AcceptConnection`.

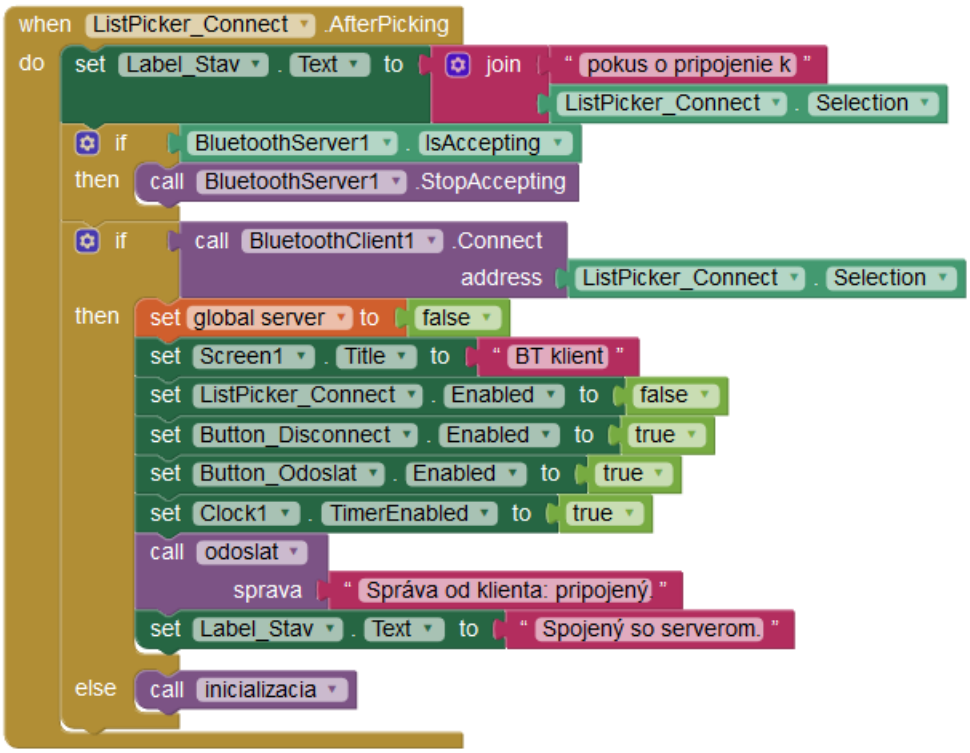


Oba režimy (klient aj server) sú naprogramované v jednej aplikácii. O tom, či bude aplikácia pracovať v režime klient alebo server, rozhodne to, kto z dvojice iniciuje spojenie. Na začiatku je aplikácia v režime server a čaká na žiadosť o spojenie.

Aplikácia bude pracovať v režime *klient*, keď používateľ iniciuje spojenie stlačením komponentu s textom **Pripojiť sa**. Je to komponent typu `ListPicker` – výber zo zoznamu. Reakcie na udalosti komponentu `ListPicker`:

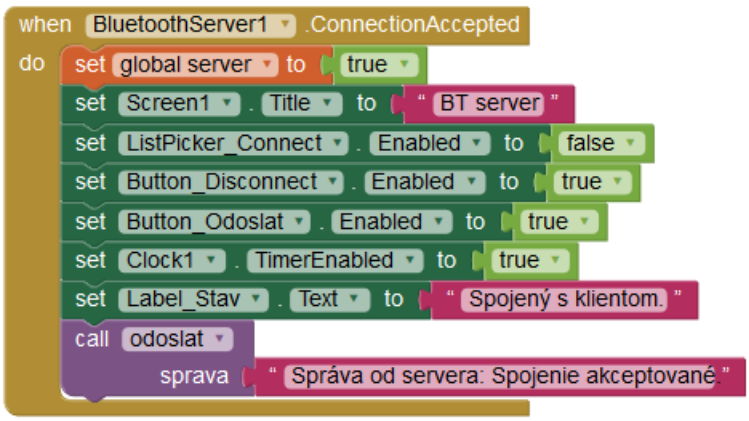
- udalosť `ListPicker.BeforePicking` (pred výberom) – zoznam (`ListPicker.Elements`) sa naplní menami a adresami dostupných a spárovaných bluetooth zariadení (`BluetoothClient.AddressesAndNames`).
- udalosť `ListPicker.AfterPicking` (po výbere)
  - ak medzitým serverová časť aplikácie akceptovala žiadosť o spojenie (`BluetoothServer.IsAccepting`), akceptácia sa zruší `BluetoothServer.StopAccepting`
  - ak sa spojenie s vybraným zariadením podarilo (`BluetoothClient.Connect (ListPicker.Selection)`), odošle sa správa serveru o úspešnom pripojení klienta a aplikácia sa nastaví do režimu klient, premenná `server` bude `false`
  - inak sa aplikácia znovu inicializuje



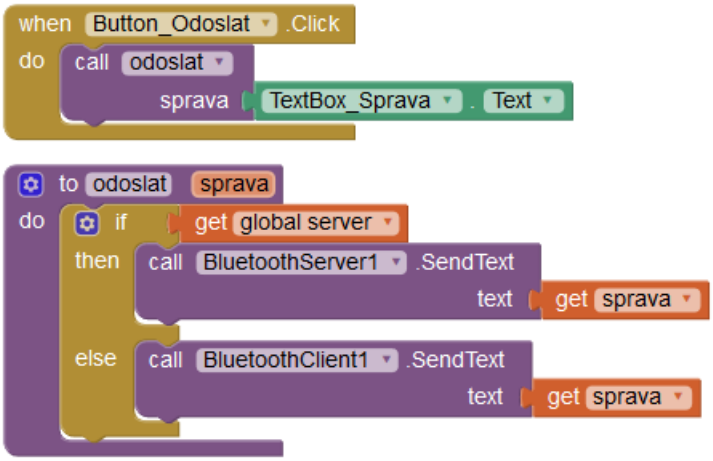


Aplikácia pracujúca v režime *server* neiniciuje spojenie, ale prijíma žiadosť o spojenie od druhého zariadenia, vtedy sa generuje udalosť:

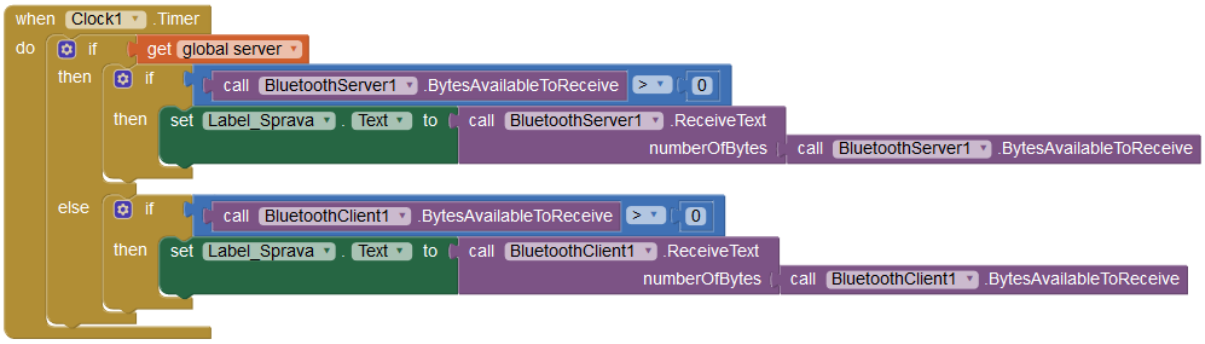
- `BluetoothServer.ConnectionAccepted` – aplikácia sa nastaví do režimu server (premenná `server` sa nastaví na `true`) a odošle sa správa klientovi o úspešnom pripojení.



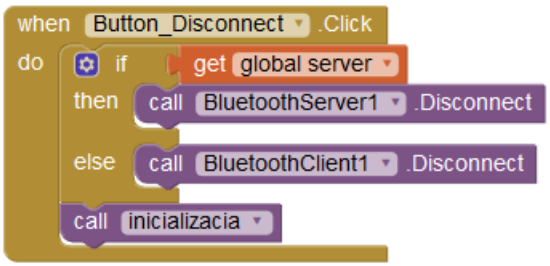
Po úspešnom spojení si zariadenia môžu vymieňať správy. Na odoslanie správy sa použije metóda `SendText` komponentu `BluetoothClient` alebo komponentu `BluetoothServer` podľa toho, v akom režime aplikácia pracuje.



Na prijímanie správ je použitý komponent `Clock` ako časovač. Časovač generuje pravidelné udalosti (`Timer`), pri ktorých sa zisťuje, či bola prijatá správa. Funkcia `BytesAvailableToReceive` komponentu `BluetoothClient` alebo komponentu `BluetoothServer` podľa toho, v akom režime aplikácia pracuje, zistí počet prijatých bajtov. Ak je väčší ako 0, funkcia `ReceiveText` vráti prijatý text.



Spojenie sa ukončí stlačením tlačidla s nápisom **Ukončiť spojenie**. V reakcii na udalosť `Click` komponentu `Button` sa zavolá metóda `Disconnect` komponentu `BluetoothClient` alebo komponentu `BluetoothServer` podľa toho, v akom režime aplikácia pracuje.



Úloha 5

Pridajte do projektu Kockový poker komunikáciu medzi dvomi hráčmi. Podľa vzoru ukážkovej aplikácie z úlohy 4 nadviažte spojenie dvoch zariadení. Doprogramujte funkčnosť aplikácie:

- po ukončení ťahu
  - aplikácia odošle pripojenému zariadeniu informácie: body za ťah a celkové skóre,
  - aplikácia obmedzí hráčovi ďalší ťah dovtedy, kým nedostane správu o ukončení ťahu súpera,
- po obdržaní správy o ukončení ťahu súpera:
  - aplikácia zobrazí prijaté informácie o skóre súpera a vyhodnotí priebežný stav hry,
  - ak nie je koniec hry, aplikácia sprístupní hráčovi nový ťah,
  - ak je koniec hry, vyhodnotí víťaza.

Ako vylepšiť či rozšíriť našu aplikáciu?

Hráči kockového pokra používajúci našu aplikáciu hrajú každý so svojím mobilným zariadením a odosielajú si prostredníctvom bluetooth komunikácie informácie o dosiahnutých bodoch vo svojom ťahu. Rozšírením aplikácie by mohlo byť „zdieľanie obrazovky“ súpera: Hráči by si posielali všetky informácie o hádzaní a výbere kociek, na základe ktorých by sa na obrazovke mohol zobrazovať celý priebeh hry súpera (animácia hodov, výber kociek), nielen výsledné bodové hodnotenie.

Zamyslime sa, čo sme sa naučili

- Rozumieť architektúre klient-server pri komunikácii prostredníctvom technológie Bluetooth.
- Použiť komponenty `BluetoothClient` a `BluetoothServer` na vytvorenie klientskej a serverovej aplikácie na komunikáciu dvoch mobilných zariadení.
- Vytvoriť univerzálnu aplikáciu s klientskou a serverovou časťou komunikácie Bluetooth.

5 Geolokácia

Geolokácia je určenie geografickej polohy. Mnohé mobilné zariadenia vedia určiť svoju polohu pomocou geolokačného senzora a/alebo pripojenia na internet. Údaje o geografickej polohe využívajú mobilné aplikácie rôznym spôsobom, napríklad na navigovanie do cieľa, zaznamenanie trasy, označenie fotografie údajom, kde bola zhotovená, hranie exteriérových geolokačných hier, zdieľanie polohy s priateľmi a podobne.

V tejto kapitole uvádzame námety na dva projekty, ktorých cieľom je vytvorenie aplikácie využívajúcej údaje o polohe. Pri vývoji takýchto aplikácií vzniká problém s priebežným testovaním a ladením aplikácie, keďže ako vstup sa spracovávajú dáta z geolokačného senzora, ktoré sa dajú získavať len v exteriéri. Priebežné testovanie aplikácie v exteriéri je časovo náročné, preto na získavanie testovacích vstupov o polohe zariadenia pri vývoji aplikácií použijeme emulátor GPS vstupov. Hotové aplikácie otestujeme v teréne so skutočnými dátami z geolokačného senzora. Aplikácie majú charakter hry, v prvom projekte edukačnej, v druhom zábavnej pohybovej hry:

5.1 Reverse caching

Aplikácia, ktorá približuje princíp určovania geografickej polohy pomocou trilaterácie. Pomocou aplikácie používateľ hľadá miesto v teréne, kde môže byť ukrytá schránka s „pokladom“ (cache), na základe informácie o vzdialenosti k cieľu bez udania smeru. V aplikácii sú použité komponenty `Map` a `LocationSensor` na prácu s geodátami.

5.2 Geolokačná hra

Akčná pohybová exteriérová hra na získavanie bodov, ktorú môžu hrať formou súťaže medzi sebou jednotlivci alebo tímy. Žiaci ju majú remixovať (vytvoriť vlastnú verziu hry rozšírením, upravením, obmenou námetu pôvodnej hry). V aplikácii je použitý komponent `LocationSensor` na prácu s geodátami a komponent `Clock` na meranie času.

5.1 Reverse caching

Kľúčové slová

GPS, trilaterácia, emulovanie GPS, digitálna mapa, kreslenie do mapy, skupina blokov Any component

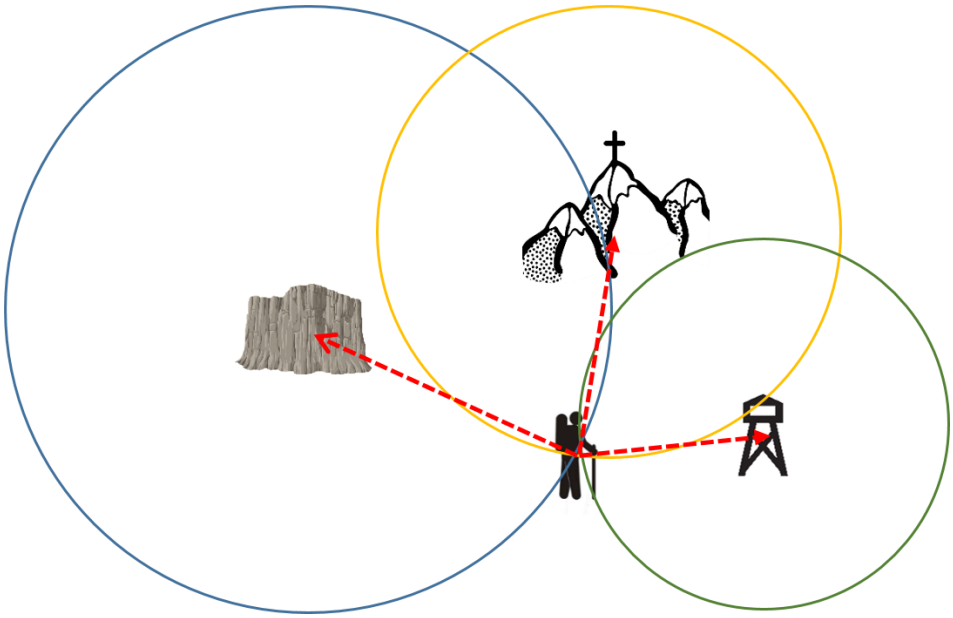
Čo sa naučíme a čo si precvičíme

- Aký je princíp GPS – určenie polohy na Zemi pomocou trilaterácie.
- Ladiť aplikáciu pomocou emulovania vstupov z geolokačného senzora.
- Spracovávať údaje o polohe z geolokačného senzora.
- Vypočítať vzdialenosť medzi aktuálnou polohou a danou geografickou polohou.
- Zobrazovať v aplikácii svoju polohu na mape a kresliť do mapy.
- Používať bloky zo skupiny *Any component*.

Čo zaujímavé môžeme zistiť?

Predstavme si človeka strateného niekde v horách, ktorý vysielačkou nadviazal spojenie so záchranármi. Zo svojej pozície vidí tri výrazné objekty: rozhľadňu, vrchol kopca s krížom a výraznú skalu. Záchranárom nahlási odhadom vzdialenosť od týchto troch miest. Ako ho záchranári nájdu?

Záchranári si na mape nájdu tri spomínané objekty. Stratený turista sa bude nachádzať niekde na kružniciach okolo objektov v danej vzdialenosti. Ak turista záchranárom správne odhadne svoju vzdialenosť od troch objektov, kružnice sa pretnú v jednom bode, na ktorom sa nachádza. Tento princíp určovania polohy sa volá *trilaterácia* (Obrázok 5.1.1). Ak turista neudá vzdialenosti celkom presne, kružnice sa nepretnú v jednom bode, ale aspoň bližšie vymedzia priestor, na prehľadávanie ktorého sa záchranári zamerajú.



Obr. 5.1.1 Princíp trilaterácie

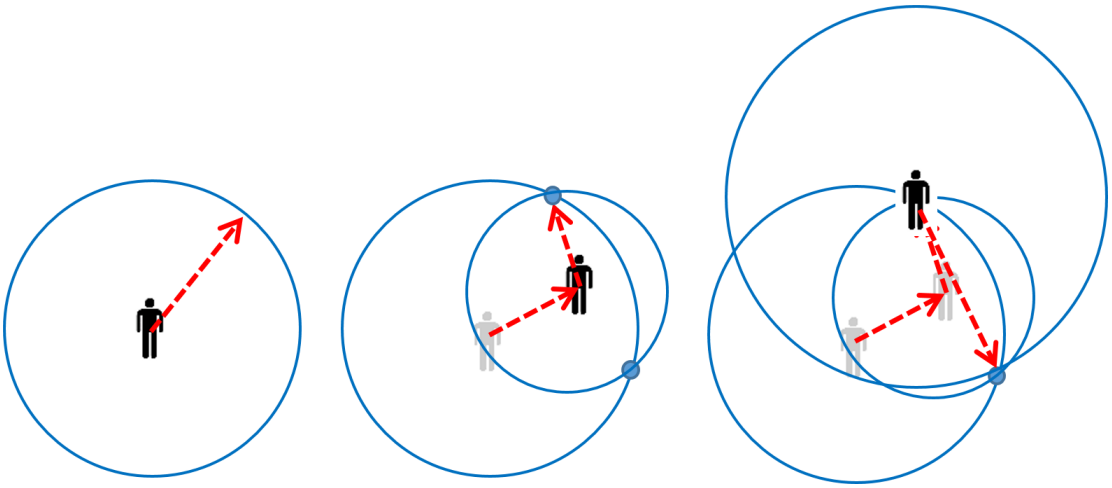
Podobne funguje princíp určovania polohy na Zemi pomocou satelitov. Pomocou mobilného zariadenia s prijímačom signálu satelitnej navigácie, napr. GPS (Geographical Positioning

System), náš prístroj zameria niekoľko satelitov a odmeria svoju vzdialenosť od nich. Na základe vzdialeností od aspoň troch satelitov (aspoň štyroch aj pre určenie nadmorskej výšky) vie určiť svoju polohu na Zemi.

Akú zaujímavú aplikáciu môžeme vytvoriť?

Vytvoríme si aplikáciu pre hru reverse caching. Je to geolokačná hra, v ktorej hráč hľadá nejaké miesto na Zemi s ukrytým „pokladom“ na základe informácie o vzdialenosti k cieľu bez udania smeru. Jednoduchá verzia tejto hry je známa ako „Zima–teplo“, v ktorej sa ale neudáva konkrétna vzdialenosť, len relatívna blízkosť k cieľu (zima, teplo, teplejšie, horúco). Meno hry reverse caching je odvodené od celosvetovej hry geolokačnej hry geocaching (Groundspeak, 2000-2020). Hráči z celého sveta v nej hľadajú schránky (cache) ukryté na rôznych zaujímavých miestach na Zemi. Adresy schránok spolu s opisom miesta sú zverejnené na internete a hráči na ich hľadanie používajú rôzne aplikácie na navigáciu do cieľa.

Mobilné aplikácie na navigáciu do cieľa nám ukazujú smer, ktorým sa nachádza náš cieľ, a vzdialenosť, ktorá nás od neho delí. Z princípu trilaterácie vieme, že aj bez udania smeru vieme nájsť cieľ na tri merania vzdialenosti (Obrázok 5.1.2). To sa využíva v hre reverse caching.



Obr. 5.1.2 Hľadanie cieľa meraním vzdialenosti

Naša aplikácia na navigovanie do cieľa určovaním vzdialenosti bude:

- určovať našu polohu a zobrazovať ju výpisom súradníc a graficky na mape,
- umožňovať načítanie a uloženie súradníc cieľa do pamäte,
- na požiadanie (stlačenie tlačidla) počítať vzdialenosť k cieľu,
- vypisovať vzdialenosť k cieľu,
- na mape vyznačovať okolo aktuálnej pozície kružnicu, na ktorej sa cieľ môže nachádzať,
- počítať počet meraní vzdialenosti.

Ako budeme postupovať pri tvorbe aplikácie?

V malej aplikácii 2.9 sme sa zoznámili s blokmi na prácu s údajmi z geolokačného senzora a na zobrazovanie geolokačných údajov v digitálnej mape. Použili sme a aj v tomto projekte použijeme tieto prvky:

- Komponenty:
  - `LocationSensor`
  - `Map`
- Udalosti:
  - `LocationSensor.Changed`
- Vlastnosti:
  - `LocationSensor.Latitude`
  - `LocationSensor.Longitude`
  - `LocationSensor.DistanceInterval`
  - `LocationSensor.TimeInterval`
  - `Map.LocationSensor`
  - `Map.ShowUser`
  - `Map.EnableZoom`
  - `Map.ZoomLevel`
  - `Map.EnablePan`

Pri ladení a testovaní aplikácií, ktoré spracovávajú údaje o polohe zariadenia, musíme zabezpečiť vstup týchto údajov buď premiestnením sa v teréne, čo je nepraktické, alebo emulovaním (napodobňovaním) zmeny polohy softvérovo. Počas ladenia aplikácie je pohodlnejšie emulovanie zmeny polohy bez presúvania sa v teréne.

Vysvetlíme si

Emulovanie (napodobňovanie) určovania polohy je nahradenie údajov o skutočnej polohe zariadenia falošnými údajmi, ktoré sú generované softvérovo. V aplikačnom obchode nájdeme aplikácie, ktoré emulujú zmenu polohy zariadenia, zadaním kľúčového slova *GPS Emulator*. Emulátor napodobňuje fyzickú zmenu polohy zariadenia vyznačením polohy na mape. Používanie emulovaných údajov v aplikáciách namiesto skutočných údajov o polohe povolíme v nastaveniach operačného systému v možnostiach pre vývojára.

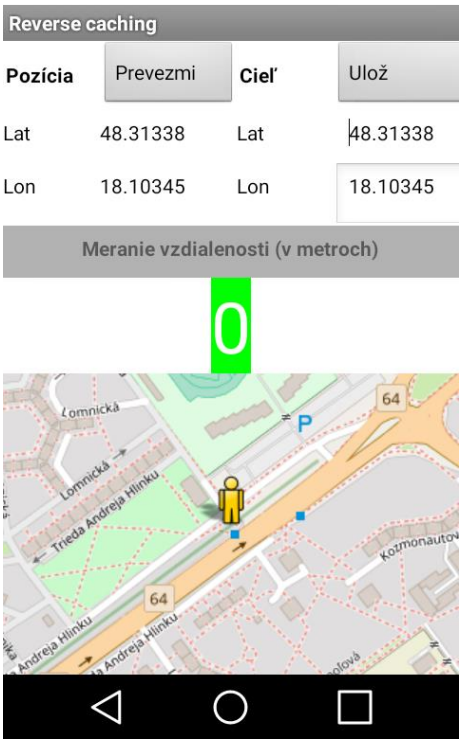
Úloha 1

Otestujte aplikáciu Zobrazovač aktuálnej polohy z malej aplikácie 2.9 pomocou emulátora GPS vstupov.

Úloha 2

Vytvorte používateľské rozhranie aplikácie Reverse caching podľa obrázka 5.1.3. Na obrazovke sa majú zobrazovať:

- aktuálne geografické súradnice označené príslušnými popismi,
- editovacie polia na zadanie súradníc cieľa označené popismi,
- tlačidlo s nápisom *Prevezmi* na prevzatie aktuálnych súradníc do cieľových súradníc (vyplní polia súradníc cieľa súradnicami aktuálnej polohy),
- tlačidlo s nápisom *Ulož* na uloženie cieľových súradníc do premenných,
- tlačidlo s nápisom *Meranie vzdialenosti (v metroch)* na výpočet vzdialenosti,
- veľký nápis na zobrazenie vzdialenosti k cieľu,
- mapa so zobrazením pozície používateľa.



Obr. 5.1.3 Dizajn aplikácie

Úloha 3

Naprogramujte zobrazovanie súradníc aktuálnej polohy a funkcie tlačidiel `Buttton_Prevezmi` a `Button_Uloz`. Aplikáciu testujte so vstupmi z emulátora. Sledujte, ako sa mení poloha používateľa na mape.

Úloha 4

Naprogramujte výpočet vzdialenosti od aktuálnej pozície k cieľu a výsledok zobrazte po stlačení tlačidla *Meranie vzdialenosti* v nápise pod tlačidlom.

Vypočítať najkratšiu vzdušnú vzdialenosť medzi dvomi bodmi na Zemi určenými geografickými súradnicami znamená určiť dĺžku oblúka na povrchu gule sploštenej na póloch. Nasledujúci vzorec počíta dĺžku oblúka na guli bez sploštenia, teda s určitou chybou:

$$d = \arccos(\sin \varphi_1 \cdot \sin \varphi_2 + \cos \varphi_1 \cdot \cos \varphi_2 \cdot \cos(\lambda_2 - \lambda_1)) \cdot R$$

φ_1, φ_2 sú geografické šírky aktuálnej pozície a cieľa (latitude), λ_1, λ_2 sú ich geografické dĺžky (longitude), R je polomer Zeme (asi 6371 km).[1]

Pri malých vzdialenostiach chyba výpočtu nie je veľká a môžeme ju zanedbať. Oveľa väčšiu chybu spôsobuje nepresnosť vstupných údajov z lokalizácie.

Vysvetlíme si

Jednotkou veľkosti uhla je *radián*. V praxi sa však častejšie na meranie uhlov používajú *stupne*. Aj geografické súradnice sa udávajú v stupňoch.

[1] https://www.movable-type.co.uk/scripts/latlong.html

V App Inventore sa používa ako jednotka uhla stupeň. To znamená, že všetky funkcie, ktorých parametrom sú uhly, alebo vracajú ako výsledok uhol, počítajú s údajmi v stupňoch. Bloky na prácu s uhlami nájdeme v skupine *Math*:

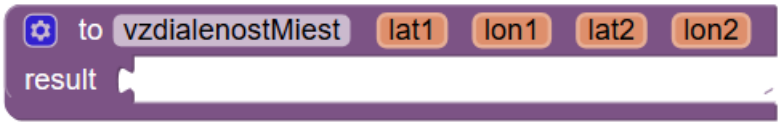
Bloky pre goniometrické funkcie `sin`, `cos`, `tan` majú vstupný parameter uhol v stupňoch a vracajú číslo.

Ich inverzné funkcie `asin`, `acos`, `atan` majú vstupný parameter číslo a vracajú uhol v stupňoch.

Na prevod stupňov na radiány a naopak slúži funkcia `convert radians to degrees`, `convert degrees to radians`.

V matematickom vzorci na výpočet vzdialenosti bodov na povrchu gule sú všetky uhly v radiánoch. Funkcie `sin`, `cos` však v App Inventore očakávajú vstupný uhol v stupňoch. To je výhodné, pretože nemusíme geografické šírky a dĺžky prevádzať v App Inventore na radiány. Výsledkom funkcie `acos` je v App Inventore tiež uhol v stupňoch. Aby sme ho mohli použiť vo vzorci na výpočet vzdialenosti dvoch bodov, musíme ho previesť na radiány.

Na sprehľadnenie kódu pri programovaní vzorca na výpočet vzdialenosti dvoch miest je užitočné naprogramovať výpočet ako funkciu so štyrmi parametrami (súradnice dvoch miest) a s návratovou hodnotou (vzdialenosťou vypočítanou podľa vzorca). Blok s hlavičkou funkcie:



Úloha 5

Do mapy zakreslite kružnicu so stredom v aktuálnej polohe používateľa s polomerom vzdialenosti od cieľa.

Na kreslenie kružnice do mapy so stredom v daných geografických súradniciach a s polomerom v metroch použijeme komponent `Circle`. Komponent sa nachádza v skupine *Maps*. V režime návrhu umiestnime kruh `Circle1` kdekoľvek do komponentu `Map1`, nastavíme farbu výplne na `None` (žiadna), zvolíme farbu a hrúbku obrysu – zostane kružnica. Kružnicu zatiaľ ponecháme neviditeľnou, zobrazovať ju budeme až počas behu programu. Nastavenie aktuálneho stredu a polomeru a následné zobrazenie kružnice naprogramujeme v reakcii na stlačenie tlačidla na výpočet vzdialenosti.

Vysvetlíme si

Vybrané vlastnosti komponentu `Circle`:

`Latitude` – geografická šírka stredu kruhu

`Longitude` – geografická dĺžka stredu kruhu

`Radius` – polomer kruhu v metroch

`FillColor` – farba výplne kruhu, 0 pre žiadnu farbu

`StrokeColor` – farba obrysu kruhu

`StrokeWidth` – hrúbka obrysu kruhu

`Visible` – viditeľnosť (true alebo false)

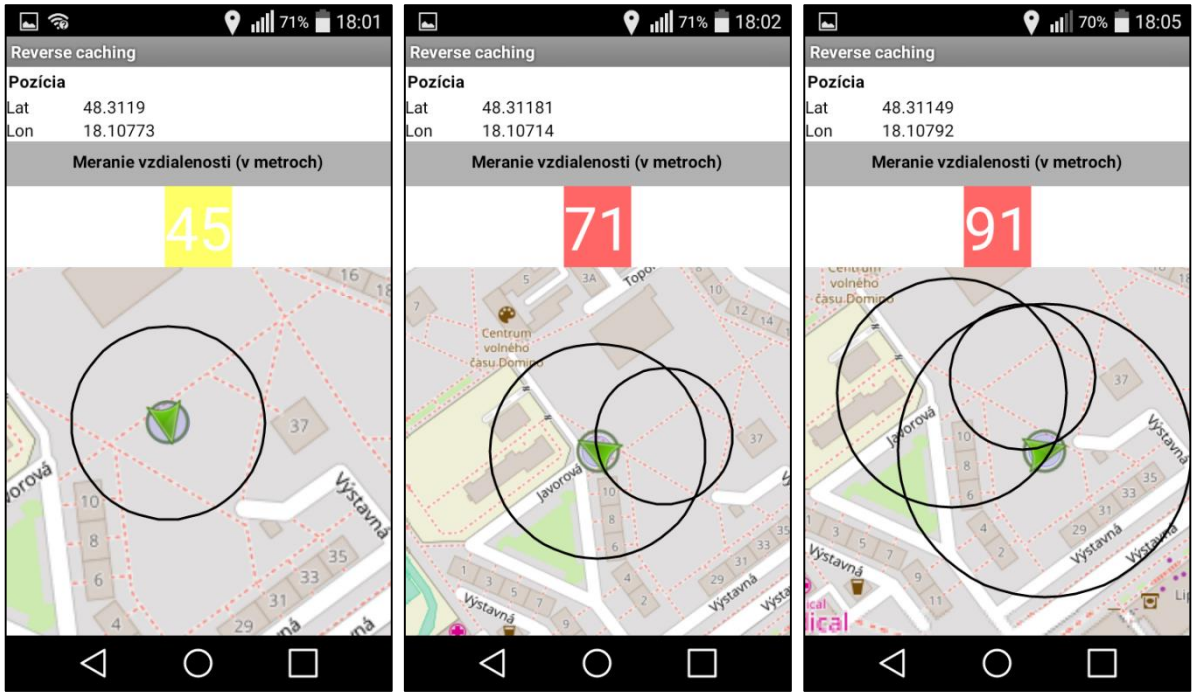
Metóda:

`SetLocation(number latitude, number longitude)` – nastaví pozíciu stredu kruhu na dané súradnice

Kreslenie kružnice do mapy nám pomáha pri hľadaní cieľa metódou trilaterácie. Na jednoznačné určenie polohy cieľa však potrebujeme odmerať vzdialenosť z aspoň troch rôznych miest a nakresliť aspoň tri kružnice. Do projektu preto pridajme viac komponentov `Circle`. Komponenty sa nedajú v App Inventore vytvárať počas behu programu, musíme si ich vložiť do projektu vopred v režime návrhu. Počet kružníc teda musíme obmedziť na konkrétny počet (aspoň 3, ale lepšie je pridať viac). Na začiatku ich zneviditeľníme.

Úloha 6

Pridajte do projektu počítadlo meraní vzdialenosti a po každom meraní zakreslite do mapy novú kružnicu (Obrázok 5.1.4). Počet meraní obmedzte a po dosiahnutí maxima neumožnite ďalšie merania.



Obr. 5.1.4 Trilaterácia – po treťom meraní je cieľ síce ďaleko, ale jeho poloha je už známa (v priesečníku kružníc)

Do mapy vložme niekoľko komponentov z triedy *Circle*, napr. 5: `Circle1`, `Circle2`, `Circle3`, `Circle4`, `Circle5`. Bolo by nepraktické pre každý z nich písať rovnaký kód pri inicializácii ich vlastností alebo zobrazovaní na mape. Namiesto toho napíšme všeobecný kód pre ľubovoľný komponent z triedy *Circle* (*Any Circle* – nejaký kruh), v ktorom špecifikujeme adresáta (konkrétny kruh) až počas behu programu.

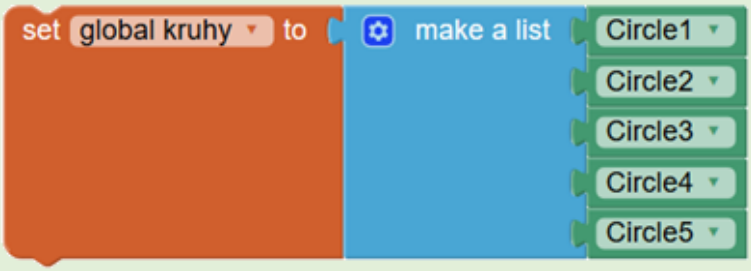
Vysvetlíme si

Bloky zo skupiny *Any component* (napr. Any Button, Any Label, Any Circle atď.) sú podobné, ako bloky konkrétnych komponentov s tým rozdielom, že umožňujú určiť adresáta (konkrétny komponent) až počas behu programu. Napríklad:

Blok pre komponent `Circle1`: a pre `Any Circle`:

Ak máme viac komponentov z rovnakej triedy, uložíme si ich do zoznamu (list). V cykle `for each item in list` vykonáme rovnaký kód zapísaný pomocou blokov zo skupiny *Any component* pre každý komponent zo zoznamu.

Napríklad vytvoríme zoznam `kruhy` komponentov `Circle1` až `Circle5`:



Nastavenie vlastnosti `Visible` postupne vo všetkých komponentoch `Circle` v zozname `kruhy` vykonáme v cykle:

Nastavenie vlastnosti `Visible` jedného komponentu, ktorý sa nachádza v zozname `kruhy` na indexe uloženom v premennej `cisloMerania` , urobíme takto:

Rovnaké vlastnosti pre každý kruh (`Visible`, `FillColor`, `StrokeWidth`) nastavíme v cykle pri inicializácii aplikácie. Pri kreslení kružnice do mapy upravíme pozíciu (metódou

`SetLocation`), polomer (`Radius`) a viditeľnosť (`Visible`) toho kruhu, ktorý je v zozname kruhov na indexe zodpovedajúcom číslu merania.

Úloha 7

Otestujte aplikáciu v teréne so skutočnými vstupmi zo senzora GPS.

Ako vylepšiť či rozšíriť našu aplikáciu?

Tu je niekoľko námetov na ďalšie rozšírenia aplikácie:

- Centrovať mapu podľa polohy používateľa (s použitím metódy `Map.PanTo`).
- Vložiť do mapy značky na miesta meraní vzdialenosti od cieľa (komponent `Marker`) v stredoch kružníc.
- Spojiť v mape lomenou čiarou miesta meraní (komponent `LineString`).
- Farebne odlišovať vzdialenosť do cieľa.
- Pri priblížení sa k cieľu na veľmi malú vzdialenosť (napr. menšiu ako 10 m) automaticky vypísať oznam.
- Po piatich meraniach vypísať správu o ukončení meraní.
- Prerobiť dizajn na aplikáciu s dvomi obrazovkami – s prvou na zadávanie a uloženie súradníc cieľa, s druhou na hľadanie cieľa pomocou určovania vzdialenosti a zobrazovania kružníc na mape.
- Pridať tlačidlo na ukončenie aplikácie.

Ako sa zahrať s našou aplikáciou?

V prvej fáze hry označíme cieľové miesto v teréne nejakou nenápadnou, ale viditeľnou značkou (napr. nálepkou) alebo na ňom ukryjeme schránku s pokladom (cache). Súradnice cieľa uložíme v našej aplikácii.

V druhej fáze odovzdáme mobilné zariadenie s našou aplikáciou niekomu, kto bude náš označený cieľ hľadať. Na nájdenie cieľa má obmedzený počet pokusov s meraním vzdialenosti podľa toho, aký sme zvolili v aplikácii.

Zamyslime sa, čo sme sa naučili

- Použiť emulátor na generovanie falošnej polohy mobilného zariadenia.
- Počítať vzdialenosť dvoch bodov určených geografickými súradnicami.
- Zobrazovať v aplikácii polohu na mape a kresliť do mapy.
- Vytvárať hromadné algoritmy pre komponenty pomocou blokov zo skupiny *Any Component*.

5.2 Geolokačná hra

Kľúčové slová

geolokačná hra, lokalizačný senzor, emulovanie GPS

Čo sa naučíme a čo si precvičíme

- Preskúmame podrobnejšie vlastnosti komponentu `LocationSensor`.
- Použijeme vzorec na výpočet vzdialenosť dvoch miest určených geografickými súradnicami.
- Navrhneme a naprogramujeme vlastnú geolokačnú hru.
- Aplikáciu budeme ladiť pomocou emulovania vstupov z lokalizačného senzora mobilného zariadenia, otestujeme ju aj v teréne.

Geolokačné hry sú počítačové hry pre mobilné zariadenia s prijímačom GPS, v ktorých je kľúčovou vstupnou informáciou geografická poloha hráča. Hráč sa v priebehu hry pohybuje v exteriéri (v parku, na ihrisku, v uliciach mesta a pod.) – presúva sa cieľavedome alebo sa náhodne ocitá na miestach, ktoré sú významné pre pokrok v hre. Hráč reaguje nielen na to, čo vidí na displeji mobilného zariadenia, ale aj na podnety zo skutočného prostredia, v ktorom sa práve nachádza: vedie dialóg s virtuálnymi postavami, zbiera a používa virtuálne objekty. Zároveň však dostáva úlohy, na splnenie ktorých je nevyhnutné navštíviť konkrétne miesto, získať informáciu alebo použiť skutočný objekt. Displej mobilného zariadenia predstavuje rozhranie, prostredníctvom ktorého hráč vstupuje do virtuálnej vrstvy hry.

Existujú špecializované platformy, ktoré sú zamerané len na tvorbu geolokačných hier. Dobrým príkladom sú hry *Wherigo*. Platforma Wherigo poskytuje nástroje na tvorbu hry, prehrávače hier pre rôzne mobilné operačné systémy a aj hry samotné (vytvárané členmi komunity hráčov). Na portáli www.wherigo.com nájdeme stovky hier rôzneho typu: turistických sprievodcov so zaujímavými informáciami o obci či meste, adventúry o hľadaní pokladu, športové hry, logické hry, fikcie inšpirované rozprávkou, filmom, historickými udalosťami atď. Niektoré hry je možné hrať kdekoľvek, iné sa viažu na konkrétnu lokalitu.

Geolokačné hry naprogramované priamo pre androidové zariadenia (t. j. natívne aplikácie) ľahko vyhľadáme v aplikačnom obchode Google Play. Niektoré z nich mohli byť vytvorené aj v prostredí App Inventor.

Otázky na zamyslenie

Hrali ste sa už niekedy hru, pri ktorej ste s mobilom či tabletom v ruke chodili či behali po parku, ulici alebo ihrisku? Máme na mysli len také mobilné hry, ktoré sú závislé na lokalizačnej technológii.

Hranie geolokačných hier môže byť poučné, zábavné aj zdraviu prospešné. Prečo?

Akú zaujímavú aplikáciu môžeme vytvoriť?

Chceme vytvoriť geolokačnú hru, ktorú by sme si radi zahrali v rodinnom kruhu, poskytli svojim priateľom alebo ponúkli na použitie pri voľnočasových aktivitách s mladšími deťmi. Obsah, dĺžku a náročnosť hry prispôsobíme cieľovej skupine. V našom prípade pôjde o hru hrateľnú

kdekoľvek, kde je k dispozícii dostatočne veľká hracia plocha (napr. ihrisko, park, námestie). Hra by mala byť zábavná, akčná, zmysluplná a bezpečná.

Ako budeme postupovať pri tvorba aplikácie?

Aby sme vedeli navrhnúť a naprogramovať vlastnú geolokačnú hru,

- pripomenieme si najprv, čo vieme o získavaní a spracovaní údajov z lokalizačného senzora v prostredí App Inventor,
- zahráme sa vzorovú geolokačnú hru v teréne,
- preskúmame zdrojový kód vzorovej hry.

Vzorová hra môže poslúžiť ako základ pre **remixovanie** – vytvorenie vlastnej, upravenej, rozšírenej, obmenenej verzie pôvodnej hry.

V priebehu programovania budeme chcieť hru testovať a ladiť. Priebežné testovanie v exteriéri je časovo náročné a nepraktické, pomôžeme si preto emulovaním GPS vstupov (podobne ako v kapitole *5.1. Reverse caching*).

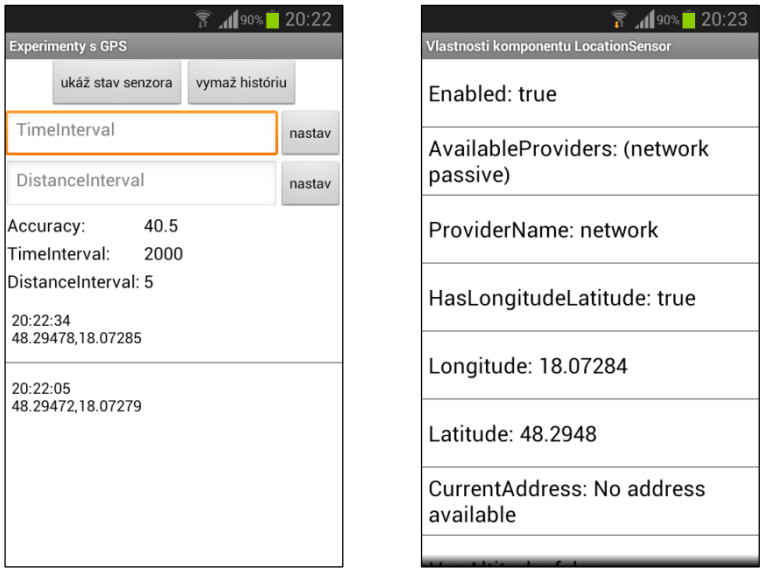
Niektoré komponenty, ktoré sú použité vo vzorovom projekte, ste už spoznali v malých aplikáciách alebo pri práci na iných projektoch:

- vizuálne komponenty a správcovia rozvrhnutia,
- komponent `Clock`,
- lokálna databáza `TinyDB`,
- komponent `Screen` (viac obrazoviek v aplikácii),
- komponent `LocationSensor`.

Úloha 1

Projekt **pmz_5_2_experimenty_s_gps.aia** obsahuje mobilnú aplikáciu pomocou ktorej môžeme realizovať experimenty s rôznymi nastaveniami komponentu `LocationSensor`. Nainštalujte si aplikáciu do mobilného zariadenia s GPS a vykonajte priamo v teréne niekoľko testov, aby ste zistili, ako sa lokalizačný senzor správa pri rôznych nastaveniach a používaní aplikácie v skutočnom teréne. Sformulujte závery svojich pozorovaní.

V aplikácii je k dispozícii tlačidlo „ukáž stav senzora“ (Obrázok 5.2.1). Jeho stlačením vyvoláme zoznam všetkých vlastností komponentu `LocationSensor1` s aktuálnymi hodnotami. Najdôležitejšie údaje o presnosti lokalizácie a parametroch riadiacich proces generovania udalostí vidíme aj priamo na hlavnej obrazovke. V histórii zobrazujeme čas výskytu udalosti `LocationSensor.LocationChanged` doplnený aktuálnymi geografickými súradnicami prečítanými pri jej spracovaní.



Obr. 5.2.1 Aplikácia na experimentovanie s nastaveniami komponentu `LocationSensor` (Michaličková, 2016)

Pri testovaní môžete postupovať napr. takto:

- Zapnite v mobilnom zariadení prijímač GPS, následne testovaciu aplikáciu. Ako dlho trvala prvá lokalizácia?
- V nastaveniach vyhľadajte položku *Lokalizačné služby*. Zapnite, resp. vypnite lokalizáciu s využitím bezdrôtových sietí (Wifi, mobilné siete) alebo/aj lokalizáciu pomocou GPS. Sledujte hodnoty vlastností `AvailableProviders` (dostupní poskytovatelia) a `ProviderName` (aktuálne zvolený poskytovateľ).
- Aké nastavenie majú vlastnosti `DistanceInterval` a `TimeInterval` po spustení aplikácie?
- Je hodnota `Accuracy` (presnosť lokalizácie) počas behu aplikácie stále rovnaká?
- Počas pohybu v teréne vyskúšajte rôzne kombinácie hodnôt `DistanceInterval` a `TimeInterval` a v spodnej časti obrazovky sledujte záznam o výskyte udalostí `LocationSensor.LocationChanged` (vyskúšajte napr. hodnoty 2000 ms a 50 m, 2000 ms a 0 m)

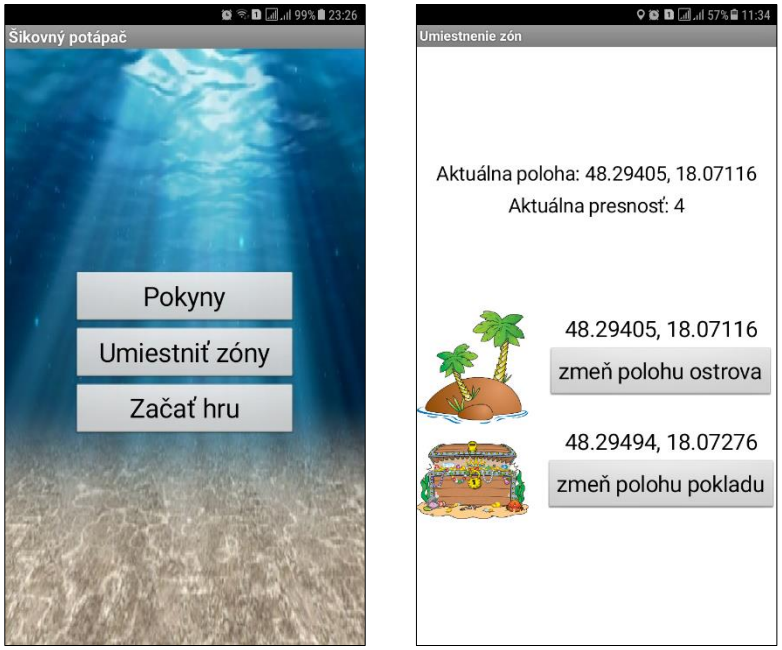
Úloha 2

Nainštalujte si geolokačnú hru naprogramovanú v projekte **pmz_5_2_sikovny_potapac.aia** do svojho mobilného zariadenia s GPS a **zahrajte sa ju vonku**, napr. na školskom ihrisku. Stanete sa na chvíľu potápačom, ktorého úlohou je zachrániť čo najviac z potopeného pokladu (Varouch, 2013). Kto bude rýchlejší a šikovnejší, ten vyhráva.

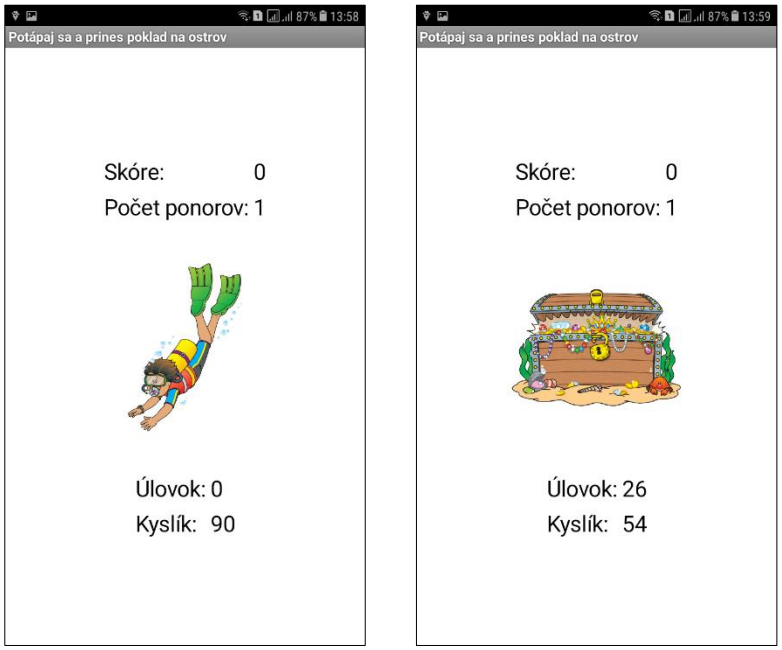
Obrázky 5.2.2 a 5.2.3 obsahujú pohľady na obrazovky aplikácie. Pred spustením hry sa uistite, či máte v zariadení zapnuté prijímanie GPS signálu. Potom postupujte takto:

1. Najprv sa rozhodnite, kde v teréne bude zóna *Ostrov* a kde sa bude nachádzať zóna *Poklad*. Zóny by mali byť od sebe primerane vzdialené (aspoň 40 metrov). Uložte ich geografické polohy pomocou tlačidiel.
2. Hráč začína hru v zóne *Ostrov*. Po vstupe do vody sa musí ponáhľať, aby sa stihol na *Ostrov* vrátiť skôr, ako sa mu v dýchacom prístroji minie kyslík. Po príchode do zóny *Poklad* si hráč-

potápač naberá z potopeného pokladu. Čím dlhšie je pri ňom, tým viac bodov (väčší úlovok) získa. Po návrate na *Ostrov* sa k celkovému skóre pripočíta práve prinesený úlovok a zásoba kyslíka sa doplní na maximum.



Obr. 5.2.2 Hlavná obrazovka a obrazovka pre umiestňovanie zón



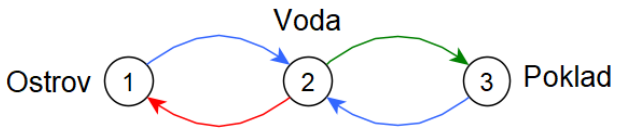
Obr. 5.2.3 Priebeh hry – plávanie k pokladu a pobyt pri poklade

3. Vo vzorovej aplikácii je časový limit nastavený na 5 minút. Po uplynutí tohto času hra skončí. Na obrazovke uvidíte svoje skóre a aj počet ponorov, ktoré ste vykonali.

Úloha 3

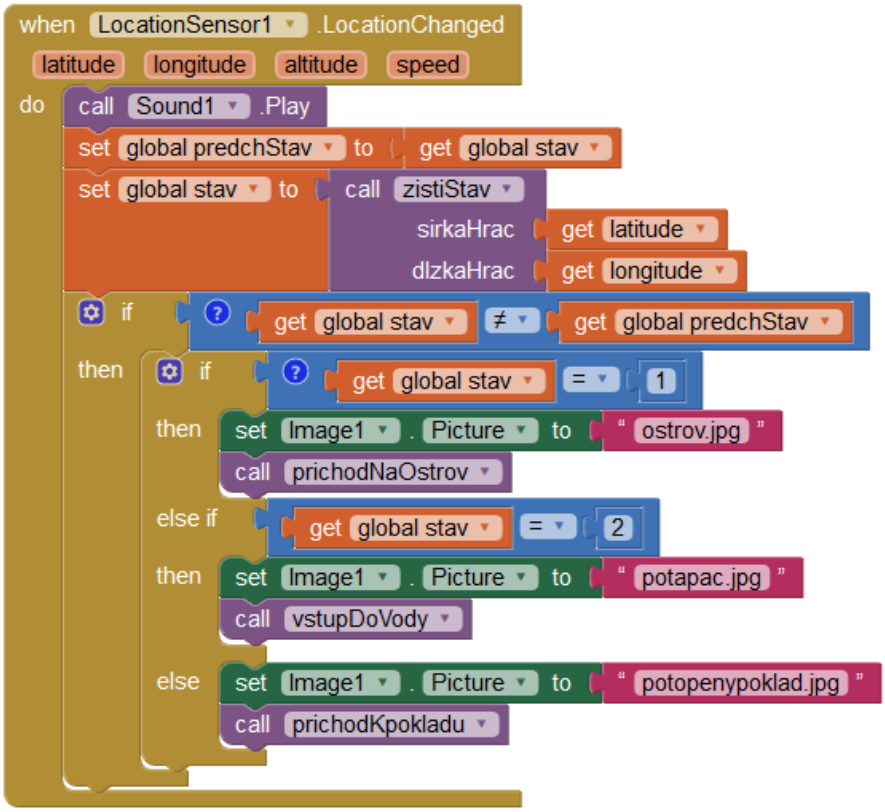
Preskúmajte zdrojový kód geolokačnej hry *Šikovný potápač* v projekte **pmz_5_2_sikovny_potapac.aia**. Odpovedzte na uvedené otázky:

1. Aplikáciu tvoria 3 obrazovky. V akom poradí ich po spustení hry uvidíme?
2. Komponent *Notifier* používame na zobrazovanie informácií a varovaní pre používateľa. Prezrite si reakcie na udalosti `when Screen1.BackPressed`, `when scrZony.BackPressed` a `when scrHra.BackPressed`. Kedy uvidíme okno so správou?
3. V lokálnej databáze *TinyDB* ukladáme geografické súradnice zón, ktoré v hre navštevujeme. Používame tagy `"sirkaOstrov"`, `"dlzkaOstrov"`, `"sirkaPoklad"`, `"dlzkaPoklad"` a rovnomenné **globálne premenné**. Na ktorých miestach zdrojového kódu tieto údaje z databázy čítame a na ktorých ich aktualizujeme?
4. Hráč sa môže v priebehu hry nachádzať v 3 stavoch – **je na ostrove** (vtedy vykladá „úlovok“ a dopĺňa kyslík), **je vo vode** (možno pláva k pokladu alebo sa vracia na ostrov) alebo **je pri poklade** (vtedy si z neho naberá). Orientovaný graf na obrázku 5.2.4 znázorňuje jednotlivé stavy a prechody medzi nimi:



Obr. 5.2.4 Stavy a prechody medzi nimi

V zdrojovom kóde rozlišujeme 3 udalosti - **vstup do vody**, **príchod na ostrov** a **príchod k pokladu**. V reakcii na udalosť `when LocationSensor1.LocationChanged` zistíme aktuálny stav. Ak **došlo ku zmene stavu** (pamätáme si aj predchádzajúci stav), zareagujeme na ňu zavolaním príslušnej procedúry:



Preskúmajte zdrojový kód v procedúrach `prichodNaOstrov`, `vstupDoVody` a `prichodKpokladu`. Kedy zapíname a kedy vypíname časovače súvisiace s ubúdaním kyslíka a pribúdaním úlovku?

Úloha 4

Navrhnite a naprogramujte vlastnú geolokačnú hru s niekoľkými zónami. Hru otestujte aj v exteriéri.

Rozhodnite sa, pre akú cieľovú skupinu bude hra určená. Vymyslite príbeh, pripravte vhodné obrázky a texty, ktoré sa budú v hre zobrazovať. Zvážte spôsob definovania zón (hra sa môže odohrávať aj na konkrétnom mieste, príslušné geografické súradnice budú v takom prípade uložené v aplikácii v premenných). Pri uvažovaní o priebehu hry (prípustné stavy a prechody medzi nimi) vám pomôže orientovaný graf. Reakcie na udalosti súvisiace so zmenou stavu naprogramujte ako samostatné procedúry.

Aby ste rozpracované verzie aplikácie nemuseli opakovane testovať v teréne, môžete fyzický pohyb hráča nahradiť emulovaním GPS vstupov (presúvaním značky na digitálne mape). Aplikáciu na emulovanie GPS vstupov vyhľadajte v aplikačnom obchode Google Play. Používanie emulovaných údajov namiesto skutočných údajov o polohe zariadenia je potrebné povoliť v nastaveniach operačného systému v možnostiach pre vývojára.

Ako vylepšiť či rozšíriť našu aplikáciu?

Vzorovú hru o *Šikovnom potápačovi* je možné vylepšiť, resp. modifikovať viacerými spôsobmi. Uvádzame niekoľko nápadov, ktoré vás môžu inšpirovať aj pri tvorbe vlastnej hry:

- v priebehu hry je vhodné, aby sa plynúci časový limit zobrazoval aj na obrazovke (vo vzorovom riešení sa nezobrazuje),
- potápača by mohol počas pobytu vo vody ohrozovať žralok,
- potápač môže mať možnosť zapojiť sa do čistenia dna od odpadkov a získavať bonusové body,
- po skončení hry zobraziť informáciu o prejdenej vzdialenosti a priemernú rýchlosť pohybu a pod.

Zamyslime sa, čo sme sa naučili

- Poznáme vlastnosti a špecifiká komponentu `LocationSensor` a sme schopní ho správne používať v geolokačnej aplikácii.
- Geolokačnú aplikáciu dokážeme ladiť aj bez fyzického pohybu v teréne, teda len na základe emulovania vstupov pre lokalizačný senzor.
- Vieme navrhnúť a naprogramovať vlastnú geolokačnú hru alebo tvorivo remixovať existujúce riešenie.

6 Senzory a aktuátory

Mobilné zariadenia sú prenosné, vybavené rôznymi senzormi (senzor zrýchlenia, gyroskop, senzor priblíženia, lokalizačný senzor, mikrofón, kamera a pod.), podporujú dotykové aj hlasové ovládanie, poskytujú prístup k internetu a jeho službám. Najmä zabudované senzory a pripojenie k sieti umožňujú vývojárom aplikácií prichádzať s originálnymi nápadmi (ovládanie dotykovými gestami, hlasom či nakláňaním zariadenia do strán, využitie rozšírenej reality, satelitnej navigácie, zapojenie viacerých používateľov a pod.). V tejto kapitole uvádzame námety na projektový vývoj dvoch zaujímavý aplikácií: nástroja užitočného pri vykonávaní fyzickej aktivity a arkádovej počítačovej hry založenej na ovládaní dotykovými gestami.

6.1 **Tréner cvikov pre pacientov a športovcov**

Aplikácia podporujúca používateľa pri vykonávaní fyzickej aktivity. Aplikácia zaznamenáva počet cvikov a čas cvičenia, v priebehu vykonávania cvičení poskytuje pravidelnú zvukovú signalizáciu alebo hlasový komentár (pomocou komponentov `Sound` a `TextToSpeech`), umožňuje získať a uchovávať jednoduchú štatistiku o predchádzajúcich tréningoch (s využitím lokálnej databázy).

6.2 **Hra ovládaná dotykovými gestami**

Viacúrovňová hra typu „plošinovka“ založená na dotykovom ovládaní. Obsahuje statické aj pohybujúce sa grafické objekty a zvukové efekty. Zmenu stavu sprajtov, resp. plánovanie udalostí v hre zabezpečíme pomocou komponentu `Clock`.

6.1 Tréner cvikov pre pacientov a športovcov

Kľúčové slová

senzory, multimédiá, komponent ProximitySensor, komponent TextToSpeech, komponent Sound, komponent TinyDB, komponent Clock, komponent ListView, komponent Spinner, komponent TableArrangement, dátový typ List.

Čo sa naučíme a čo si precvičíme

- Navrhnúť štruktúru databázy, používateľské rozhranie a správanie aplikácie tréner cvikov pre pacientov a športovcov.
- Naprogramovať užitočnú pomôcku pre pacientov a športovcov zameranú na zaznamenávanie počtu cvikov sprevádzaných nastaviteľným pravidelným zvukovým signálom (komponent `Sound`) a syntetickým hlasovým komentárom (komponent `TextToSpeech`).
- Precvičiť použitie lokálnej databázy `TinyDB` a jej metód `GetValue`, `StoreValue` a tiež funkcie na prácu s údajovým typom zoznam (`create empty list`, `make a list`, `add item to list`) a komponent `ListView` na jeho zobrazenie.

Čo zaujímavé môžeme zistiť (o podpore fyzického cvičenia pomocou mobilného zariadenia)?

Predstavme si situáciu, že chceme pre seba, pre našich priateľov, pre rehabilitujúcich pacientov po zlomeninách či pre športovcov vytvoriť aplikáciu, ktorá by bola užitočná pri fyzických cvičeniach.

Otázky na zamyslenie

Prediskutujme nasledovné otázky:

- Používate nejaké aplikácie pre podporu vašej fyzickej aktivity? Ak áno, ktoré?
- Pre ktorých používateľov a ktoré fyzické aktivity by mohlo pomôcť MZ?
- Ktoré senzory MZ by mohli byť použité na registrovanie fyzickej aktivity používateľa?
- Aký zmysel má vytvárať vlastnú aplikáciu pre podporu fyzickej aktivity?

Akú zaujímavú aplikáciu môžeme vytvoriť?

Ak sme sa rozhodli pre tvorbu vlastnej aplikácie, mali by sme v triede formou brainstormingu zozbierať nápady k možným funkcionalitám trénera cvikov pre pacientov a športovcov.

Ako budeme postupovať pri tvorbe aplikácie?

Pri tvorbe vlastného projektu môžeme postupovať podľa nasledovných krokov:

1. Spresnenie špecifikácie navrhovanej aplikácie
2. Návrh používateľského rozhrania aplikácie, zoznam komponentov a multimediálnych súborov
3. Návrh správania aplikácie

4. Tvorba používateľského rozhrania a programového kódu aplikácie
5. Prezentácia vlastnej aplikácie a diskusia využitiu aplikácie v praxi a jej prípadnému doladeniu
6. Doladenie aplikácie a jej publikovanie v rámci portfólia žiaka

1. Špecifikácia aplikácie

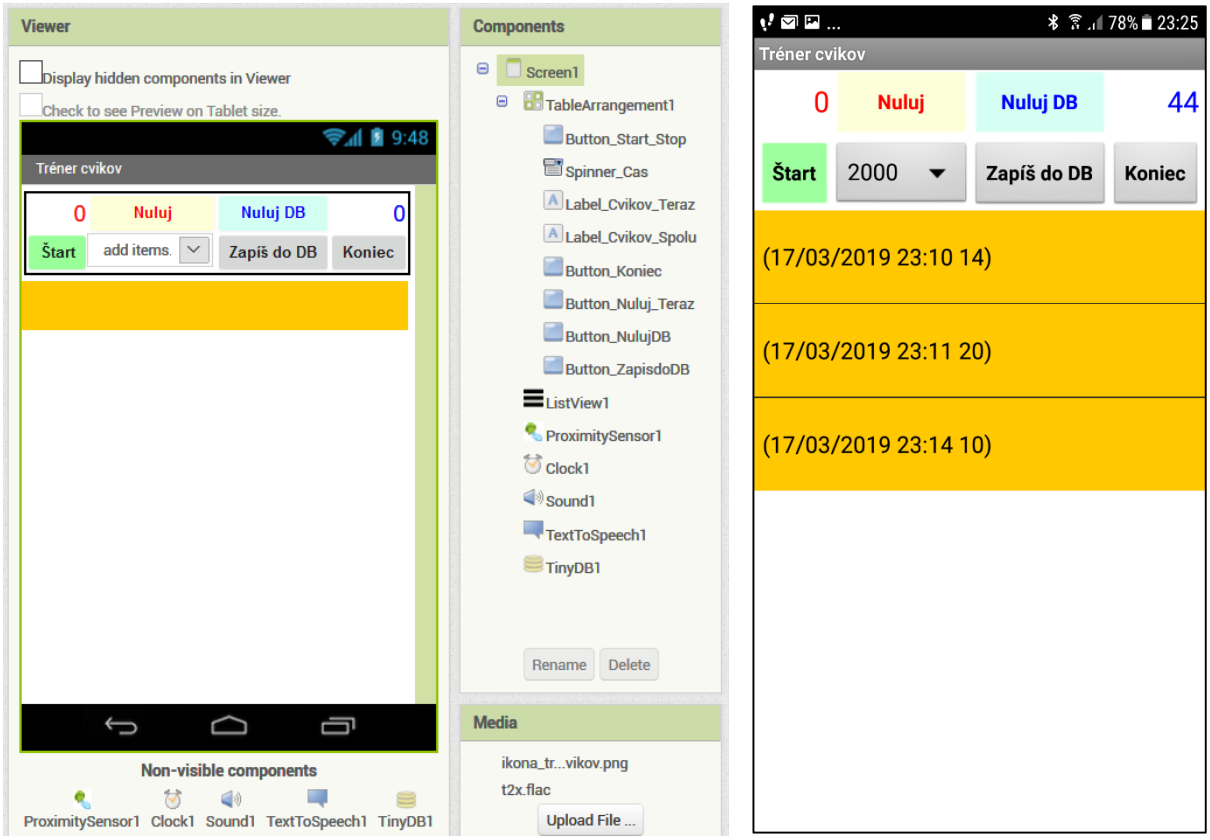
Základná verzia aplikácie využívajúcej na registrovanie pohybu (napr. cvičiacej ruky) `ProximitySensor` a zaznamenanie dátumu a času s počtom cvikov nonSQL databázu `TinyDB` má nasledovné funkcionality:

f 1. Spustenie a zastavenie pravidelnej zvukovej signalizácie trénera cvikov
f 2. Nastavenie časového intervalu pravidelnej zvukovej signalizácie trénera cvikov
f 3. Registrovanie cvikov sprevádzané graficky a syntetickým hlasom
f 4. Aktualizácia a zobrazenie počtu aktuálne vykonaných cvikov
f 5. Vynulovanie počtu aktuálne vykonaných cvikov
f 6. Zobrazenie celkového počtu cvikov, dátumu a času s počtom cvikov uložených v databáze
f 7. Zápis času cvičenia a počtu cvikov do databázy
f 8. Zmazanie všetkých záznamov databázy
f 9. Ukončenie behu aplikácie

2. *Návrh používateľského rozhrania aplikácie, zoznam komponentov a multimediálnych súborov*

Používateľské rozhranie

Na obrázku nižšie je uvedené používateľské rozhranie navrhovanej a spustenej aplikácie.



Zoznam komponentov

Vizuálne komponenty:

- `TableArrangement` (so 4 stĺpcami a 2 riadkami)
  - `Button_Start_Stop` – tlačidlo na spustenie resp. zastavenie časovača, ktoré zároveň mení stav cvičenia/necvičenia aj svoj text a farbu pozadia (f1)
  - `Spinner_Cas` – rolovacia ponuka umožňujúca nastaviť časový interval pravidelnej zvukovej signalizácie trénera cvikov (f2)
  - `Label_Cvikov_Teraz` – popisok zobrazujúci počet cvikov aktuálneho cvičenia (f4)
  - `Label_Cvikov_Spolu` – popisok zobrazujúci počet cvikov všetkých doterajších cvičení (f6)
  - `Button_Koniec` – tlačidlo na ukončenie aplikácie (F9) a prípadný zápis času aktuálneho cvičenia a počtu cvikov do databázy (f7)
  - `Button_Nuluj_Teraz` – tlačidlo na vynulovanie počtu aktuálne vykonaných cvikov (f5)
  - `Button_NulujDB` – tlačidlo na zmazanie všetkých záznamov databázy (f8)
  - `Button_ZapisdoDB` – tlačidlo na zápis času aktuálneho cvičenia a počtu cvikov do databázy (f7)

- `Screen` – zmena pozadia obrazovky na registrovanie práve vykonávaného cviku (f3)

Nevizuálne komponenty:

- `ListView` – zobrazovač zoznamu zobrazujúci dátumy a časy s počtami cvikov všetkých doterajších cvičení (f6)
- `ProximitySensor` – senzor priblíženia registrujúci práve vykonávaný cvik (f3)
- `Clock` – časovač vyvolávajúci pravidelnú zvukovú signalizáciu trénera cvikov (f1) s nastaviteľným časovým intervalom svojho spúšťania (f2)
- `Sound` – zvuková signalizáciu trénera cvikov (f1)
- `TextToSpeech` – syntéza reči na registrovanie poradia práve vykonávaného cviku (f3)
- `TinyDB` – databáza na registrovanie celkového počtu cvikov, zoznamu s dátumami a časmi spolu s počtami vykonaných cvikov (f6 až f8)

Zoznam multimediálnych súborov

- **ikona_trener_cvikov.png** – ikona aplikácie
- **t2x.flac** – zvuk signalizujúci trénera cvikov vydávaný v pravidelných časových intervaloch

3. Návrh správania aplikácie

Komponent	Udalosť	Akcia
`Screen`	`Initialize`	(f1, f4) vypnutie komponentov `ProximitySensor` a `Clock`, inicializácia globálnych premenných **stav**, **cviky**, **pocitadlo**, **spolu**
`Button_Start_Stop`	`Click`	(f1) zmena premennej **stav** a podľa nej zmena textu a farby pozadia tlačidla, zapnutie/vypnutie komponentov `ProximitySensor` a `Clock`
`Button_Nuluj_Teraz`	`Click`	(f5) vynulovanie premennej **pocitadlo** a aktualizácia hodnoty `Label_Cvikov_Teraz`
`Button_NulujDB`	`Click`	(f8) zmazanie všetkých záznamov databázy a aktualizácia hodnôt Label_Cvikov_Spolu, Label_Cvikov_Teraz a ListView.Elements
`Button_ZapisdoDB`	`Click`	(f7) v prípade nenulovej hodnoty premennej **pocitadlo** aktualizácia hodnôt kľúča **sucet** a kľúča **cviky**, aktualizácia hodnôt komponentov `Label_Cvikov_Spolu`, `Label_Cvikov_Teraz` a `ListView.Elements`,

		vynulovanie hodnoty premennej **pocitadlo**
`Button_Koniec`	`Click`	(f9, f7) ukončenie aplikácie a v prípade nenulovej hodnoty premennej **pocitadlo** aktualizácia hodnôt kľúča **sucet** a kľúča **cviky**, aktualizácia hodnôt komponentov `Label_Cvikov_Spolu`, `Label_Cvikov_Teraz` a `ListView.Elements`, vynulovanie hodnoty premennej **pocitadlo**
`Spinner_Cas`	`AfterSelecting`	(f2) nastavenie časového intervalu pravidelnej zvukovej signalizácie trénera cvikov pre komponent `Clock`
`ProximitySensor`	`ProximityChanged`	(f3) registrovanie vykonávaného cviku pomocou zvýšenia hodnoty premennej **pocitadlo**, a jej aktualizácie v komponente `Label_Cvikov_teraz` sprevádzané zmenou pozadia obrazovky a syntetickým hlasom vyslovujúcim hodnotu premennej **pocitadlo**
`Clock`	`Timer`	(f1) pravidelná zvuková signalizácia trénera cvikov

4. Tvorba používateľského rozhrania a programového kódu aplikácie

Pri tvorbe používateľského rozhrania aplikácie použijeme návrh grafického používateľského rozhrania obsahujúci vizuálne komponenty (`TableArrangement`, `Screen`, `Button`, `Label`, `Spinner`), nevizuálne komponenty (`Clock`, `ProximitySensor`, `Sound`, `ListView`, `TextToSpeech`, `TinyDB`) a multimediálne súbory (ikona aplikácie a zvuk trénera cvikov).

Programový kód vytvárame po jednotlivých funkcionalitách, ktorých riešenia uvedieme a okomentujeme po skupinách:

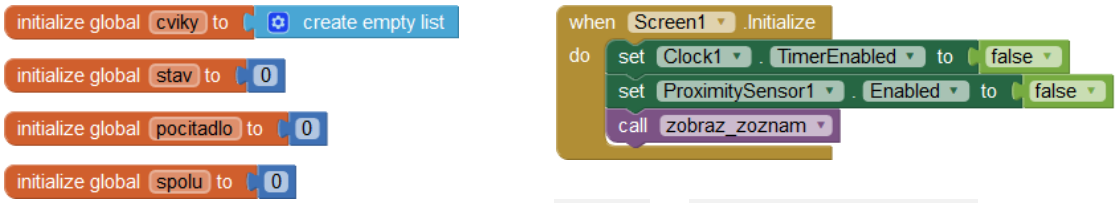
- Počiatočné nastavenie aplikácie
- Spustenie zvukovej signalizácie trénera cvikov a registrácia práve vykonávaných cvikov
- Aktualizácia hodnôt databázy

Počiatočné nastavenie aplikácie

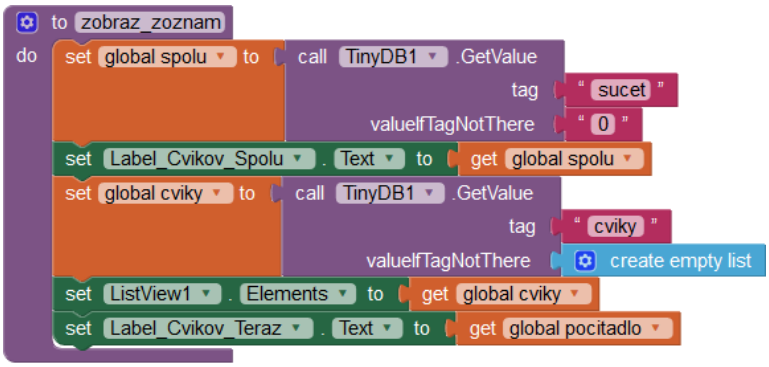
V aplikácii použijeme 4 globálne premenné:

- **stav** – pri hodnote 0 vypne `Clock` a `ProximitySensor`, pri hodnote 1 ich zapne, hodnota sa prepína pomocou `Button_Start_Stop` (počiatočná hodnota tejto premennej je 0),

- **cviky** – zoznam zoznamov dátumov a časov cvičenia spolu s časom tohto cvičenia, ukladá sa do databázy pod rovnomenným kľúčom **cviky** (počiatočná hodnota tejto premennej je prázdny zoznam),
- **pocitadlo** – počet aktuálne vykonávaných cvikov (počiatočná hodnota tejto premennej je 0),
- **spolu** – počet celkovo vykonaných cvikov, ukladá sa do databázy pod kľúčom **sucet** (počiatočná hodnota tejto premennej 0).

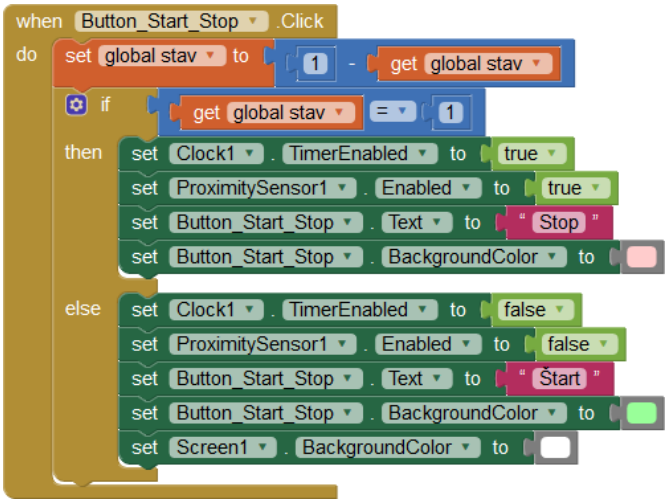


Po spustení aplikácie sa vypnú komponenty `Clock` a `ProximitySensor` a pomocou vlastnej procedúry **zobraz_zoznam** sa aktualizujú hodnoty globálnych premenných **spolu**, **cviky** a tiež hodnoty komponentov `Label_Cvikov_Spolu`, `Label_Cvikov_Teraz` a `ListView.Elements`.

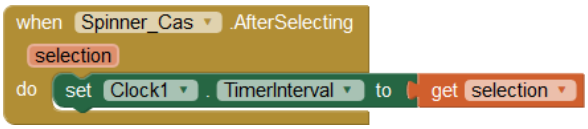


Spustenie zvukovej signalizácie trénera cvikov a registrácia práve vykonávaných cvikov

Tlačidlom `Button_Start_Stop` cyklicky prepíname hodnotu premennej **stav** medzi hodnotami 0 a 1. Podľa hodnoty premennej **stav** meníme text a farbu pozadia tohto tlačidla.



Po spustení aplikácie je časový interval pravidelnej zvukovej signalizácie trénera cvikov nastavený na 2000 ms. Pomocou rolovacej ponuky `Spinner_Cas` a jej udalosti `AfterSelecting` vieme zmeniť tento interval (vlastnosť `Clock.TimerInterval`).

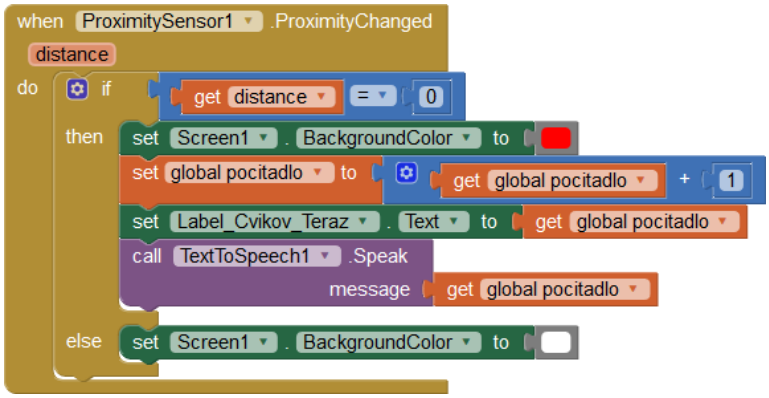


Pravidelnú zvukovú signalizáciu trénera cvikov dosiahneme pomocou časovača `Clock.Timer`.

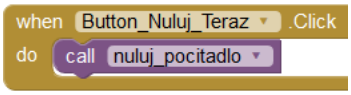
Registráciu práve vykonávaných cvikov zabezpečíme pomocou komponentu `ProximitySensor` a jeho udalosti `ProximityChanged`.

Ak sa priblížime k MZ (t. j. robíme cvik):

- zvýši sa hodnota premennej **pocitadlo**, ktorá sa zároveň aktualizuje v komponente `Label_Cvikov_teraz`,
- zmení sa pozadia obrazovky z bielej na červenú farbu,
- syntetickým hlasom sa vysloví hodnota premennej **pocitadlo**.



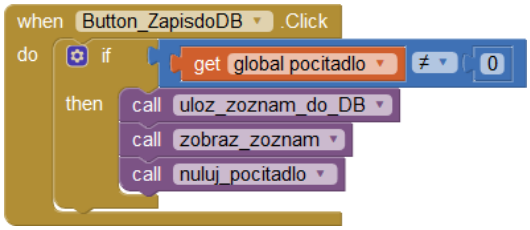
Vynulovanie počtu práve vykonávaných cvikov (t. j. hodnotu premennej **pocitadlo**) a aktualizácie komponentu `Label_Cvikov_Teraz` dosiahneme pomocou tlačidla `Button_Nuluj_Teraz`.



Aktualizácia hodnôt databázy

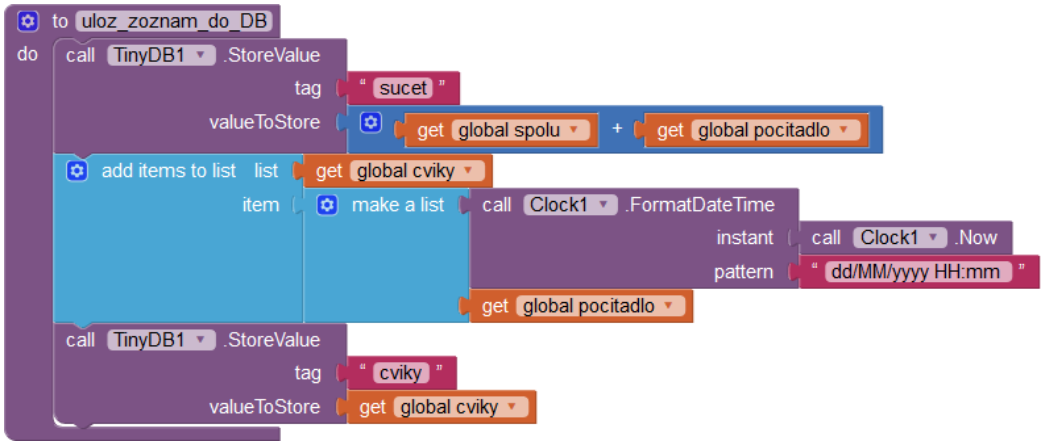
Veľmi dôležitou funkcionalitou je ukladanie výsledkov viacerých cvičení do databázy. V tejto verzii aplikácie sme použili lokálnu databázu `TinyDB`.

Tlačidlom `Button_ZapisdoDB` spustíme pomocné procedúry.



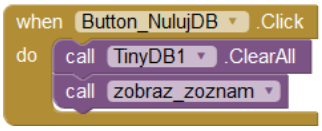
Procedúra **uloz_zoznam_do_DB**:

- nastaví kľúč **sucet** na hodnotu súčtu práve vykonaných cvikov (t. j. hodnotu premennej **pocitadlo**) a celkového počtu predtým vykonaných cvikov (t. j. hodnotu premennej **sucet**),
- pripojí do zoznamu dátumov a časov s počtami všetkých cvičení (t. j. do premennej **spolu**) aktuálny dátum a čas s počtom práve vykonaných cvikov,
- nastaví kľúč **cviky** na hodnotu aktualizovanej premennej **cviky**,
- aktualizované hodnoty oboch kľúčov **sucet** a **cviky** uloží do lokálnej databázy `TinyDB`.

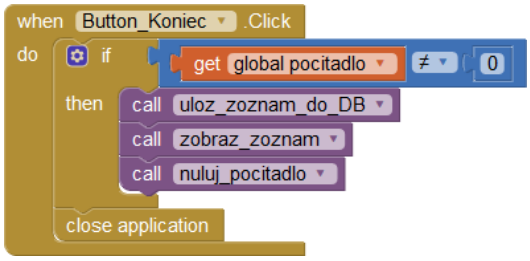


Vyvolaním procedúry **zobraz_zoznam** sa aktualizujú hodnoty globálnych premenných **spolu**, **cviky** a tiež hodnoty komponentov `Label_Cvikov_Spolu`, `Label_Cvikov_Teraz` a `ListView.Elements`. Vyvolaním procedúry **nuluj_pocitadlo** sa nastaví hodnota premennej **pocitadlo** na 0 a zaktualizuje sa podľa toho hodnota `Label_Cvikov_Teraz`.

Lokálnu databázu vyčistíme pomocou tlačidla `Button_NulujDB`.



Aplikáciu ukončíme stlačením tlačidla `Button_Koniec`, ktorý v prípade, že sme zabudli zapísať výsledky cvičenia do databázy vykoná rovnaké pomocné procedúry, ktoré by sa spustili pri stlačení tlačidlom `Button_ZapisdoDB`.



Týmto sme uzavreli popis zdrojového programového kódu základnej verzie aplikácie trénera cvikov pre pacientov a športovcov, ktorú by ste mali zvládnuť naprogramovať s minimálnou individuálnou pomocou a usmernením učiteľa.

Bude veľmi vítané, ak túto verziu aplikácie upravíte a rozšírite o ďalšie funkcionality podľa vlastných možností a záujmov. Pri vyvíjaní aplikácie môžete pracovať v dvojiciach, pričom môžete využiť referenčné príručky, textové či video tutoriály na internete, či pomoc a rady od spolužiakov prípadne učiteľa. Veríme že pri programovaní tejto aplikácie budete mať radosť z vlastného objavovania a výslednej aplikácie využiteľnej v praxi.

Pred odovzdaním a prezentáciou aplikácie nezabudnite, aby vaša aplikácia mala priradenú ikonu (podľa možnosti vlastnú), vo vlastnosti `Screen.AboutScreen` bolo uvedené meno autora aplikácie a vo vlastnostiach `Screen.VersionCode` a `Screen.VersionName` uvedené správne hodnoty.

5. Prezentácia vlastnej aplikácie a diskusia využitiu aplikácie v praxi a jej prípadnému doladeniu

Veľmi dôležitou súčasťou životného cyklu tvorby aplikácie sú prezentácie rozšírených aplikácií jednotlivých žiakov, resp. dvojíc žiakov.

Každý projekt by ste mali prezentovať v rozsahu cca 1 až 2 minúty, počas ktorých predstavíte vlastné doplnené funkcionality aj s ich využitím v praxi. Mali by ste tiež uviesť funkcionality, ktoré by ešte mohli doplniť do potenciálnej 2. verzie svojej aplikácie.

Po prezentáciách projektov prediskutujte, ktoré z uvedených funkcionalít vás zaujali, a tiež aké máte návrhy na úpravy a vylepšenia niektorých prezentovaných aplikácií.

6. Doladenie aplikácie a jej publikovanie v rámci portfólia žiaka

Svoj prezentovaný projekt doladťe podľa návrhov učiteľa a spolužiakov a uložte ho do svojho projektového portfólia.

Zamyslime sa, čo sme sa naučili

- Navrhli sme štruktúru databázy, používateľské rozhranie a správanie aplikácie tréner cvikov pre pacientov a športovcov.
- Naprogramovali sme užitočnú pomôcku pre pacientov a športovcov zameranú na zaznamenávanie počtu cvikov sprevádzaných nastaviteľným pravidelným zvukovým signálom (komponent `Sound`) a syntetickým hlasovým komentárom (komponent `TextToSpeech`).
- Pri tvorbe aplikácií sme precvičili použitie lokálnej databázy `TinyDB`, jej metódy `GetValue`, `StoreValue` a tiež funkcie na prácu s údajovým typom zoznam (`create empty list`, `make a list`, `add item to list`) a komponent `ListView` na jeho zobrazenie.

Sebahodnotiaca karta

Vyplňte uvedenú sebahodnotiacu kartu k tvorbe svojej aplikácie *Tréner cvikov*:

Meno a priezvisko	
Čo som sa nové naučil(a) pri programovaní tohto projektu?	
Ktoré funkcionality som doplnil(a) do svojej aplikácie?	
Ktoré funkcionality má zaujali v aplikáciách spolužiakov?	
Čo nové z problematiky Ai2 by som sa rád(a) naučil(a)?	
Čo nové by som rád/rada naprogramoval(a) v Ai2?	

6.2 Hra ovládaná dotykovými gestami

Kľúčové slová

hra, dotykové gestá, komponent Clock, grafické objekty, zvukové efekty, viac úrovní, viac obrazoviek

Čo sa naučíme a čo si precvičíme

- Navrhneme a naprogramujeme vlastnú hru s viacerými úrovňami ovládanú dotykovými gestami.
- Použijeme statické aj pohybujúce sa grafické objekty a zvukové efekty.
- Precvičíme si prácu s viacerými obrazovkami.
- Komponent `Clock` z kategórie *Sensors* zužitkujeme novým spôsobom: na riadenie zmeny stavu sprajtov, resp. ako prostriedok na plánovanie udalostí v hre.
- Vytvorenú hru zverejníme v *Galérii* prostredia Ai2.

Počítačové hry patria k najpopulárnejším a aj komerčne najúspešnejším aplikáciám. Platí to aj pre mobilné platformy. V online aplikačných obchodoch nájdeme hry rôzneho žánru a kvality – od jednoduchších hier od individuálnych (aj amatérskych) vývojárov po profesionálne produkty veľkých vývojárskych tímov. Tablety a smartfóny sú v porovnaní so stolnými počítačmi špecifické vo viacerých ohľadoch. Ľahko sa prenášajú, podporujú dotykové aj hlasové ovládanie, sú vybavené rôznymi senzormi (napr. kamera, gyroskop, akcelerometer, lokalizačný senzor) a poskytujú prístup k internetu a jeho službám v princípe kdekoľvek a kedykoľvek. Tieto možnosti prirodzene vedú k vzniku hier s originálnym námetom aj spôsobom ovládania.

Otázky na zamyslenie

Aké žánre počítačových hier poznáte?

Aké hry máte nainštalované vo svojom tablete alebo v smartfóne?

Malo by zmysel hrať ich aj na stolnom počítači?

Ktorú mobilnú hru považujete za najoriginálnejšiu, najzaujímavejšiu?

Akú zaujímavú aplikáciu môžeme vytvoriť?

Naším cieľom bude vyvinúť (t. j. vymyslieť a naprogramovať) vlastnú arkádovú hru typu „plošinovka“. Arkádové hry sú typické jednoduchým, ale zábavným a pútavým konceptom. Obsahujú viacero úrovní (levelov) so stupňujúcou sa náročnosťou. O úspešnosti hráča rozhoduje najmä dobrý postreh a zručnosť v ovládaní vstupného zariadenia, ale často tiež schopnosť koncentrovať sa, logicky myslieť, predvídať a rýchlo sa rozhodovať. V plošinových hrách prekonáva hlavný hrdina rôzne nástrahy virtuálneho sveta - skáče cez prekážky, vyhýba sa nepriateľom alebo s nimi bojuje. Popritom zbiera rôzne predmety, ktoré mu môžu pomôcť alebo za ktoré získava body.

Ako budeme postupovať pri tvorba aplikácie?

Pri tvorbe vlastnej plošinovej hry bude potrebné:

- zvoliť vhodný námet,
- premyslieť si pravidlá hry a nadväznosť levelov,
- navrhnúť grafické používateľské rozhranie hry, pripraviť obrázky a zvuky (príp. iné dáta potrebné pre jednotlivé levely),
- vyriešiť problém generovania grafických objektov (napr. pozadí, prekážok, nepriateľov) v hernom svete,
- vyriešiť problém ovládania hlavného hrdinu (beh, výskok, strieľanie a pod.),
- vyriešiť problém kolízie hráča s inými grafickými objektami.

Po otestovaní hry môžeme výsledok svojej práce zverejniť v *Galérii* prostredia MIT AI2, aby sme získali spätnú väzbu aj od iných používateľov.

V tejto kapitole budeme riešenia vybraných problémov ilustrovať na príklade jednoduchej vzorovej hry, v ktorej je hlavným hrdinom zajac bežiaci po lúke. Zajaca budeme ovládať dotykom prsta. Nad hlavou mu lietajú mrkvy, ktoré má zbierať, pod nohami sa mu gúľajú kapusty, ktoré má preskakovať. Keď zajac zakopne o kapustu, stráca život.

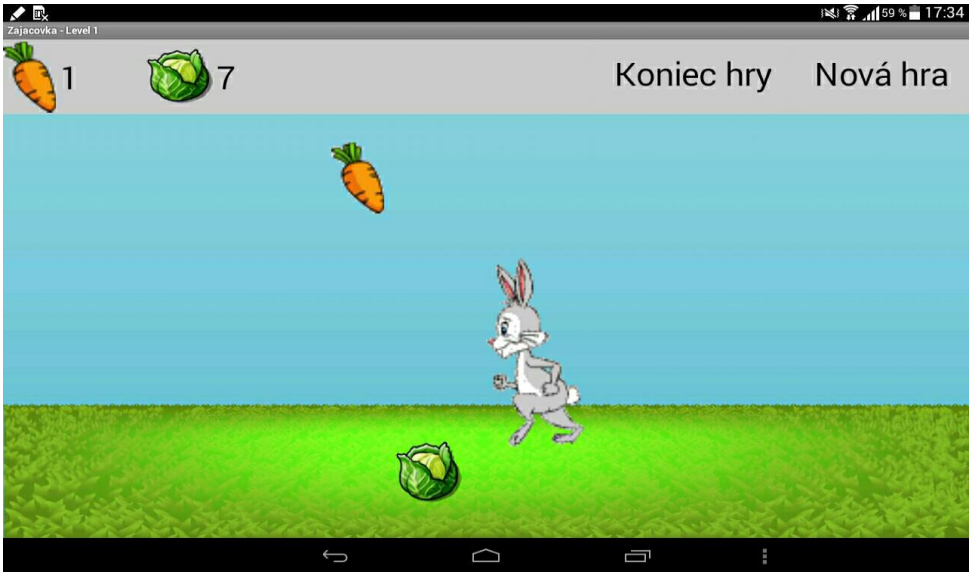
Na postup do nasledujúceho levelu musí hráč chytiť predpísaný počet mrkiev. Základné parametre hry (napr. počet životov) bude možné nastaviť na osobitnej obrazovke. Na pozadí sa má prehrávať rytmická hudba, pri zrážkach sprajtov zase vhodné zvukové efekty. Hra by sa mala dať v priebehu hrania aj pozastaviť.

Niektoré komponenty, ktoré použijeme, ste už spoznali v úvodných etudách alebo pri práci na iných projektoch:

- vizuálne komponenty a správcovia rozvrhnutia,
- komponenty `Canvas` a `ImageSprite`,
- komponenty `Sound` a `Player`,
- komponent `Clock`,
- lokálna databáza `TinyDB`, komponent `Screen` (viac obrazoviek v aplikácii).

Úloha 1

V projekte pmz_6_2_zajacovka_ver0.aia nájdete nultú verziu vzorovej hry s grafickými a zvukovými súbormi a základnými komponentami tvoriacimi používateľské rozhranie aplikácie:



Obr. 6.2.1 Používateľské rozhranie základnej verzie hry Zajacovka (Michaličková, 2016)

Všimnite si, že:

- obrazovka má nastavenú orientáciu na šírku (*Landscape*)
- na plátne sú umiestnené 3 komponenty typu `ImageSprite` (sprajty) s obrázkami zajaca, mrkvy a kapusty,
- horný pás nad plátnom obsahuje 2 statické obrázky (komponenty typu `Image`), nápisy s aktuálnym stavom počítadiel a tlačidlá pre spustenie novej hry a ukončenie práve bežiacej hry (vhodné umiestnenie komponentov sme dosiahli vložením jedného komponentu `HorizontalArrangement` do druhého),
- pre zajaca je k dispozícii 8 obrázkov s názvami zajac1.png, zajac2.png, ..., zajac8.png (ide o rôzne fázy animácie behu),
- v časti *Media* sú pripravené už aj zvukové súbory (budeme ich potrebovať neskôr).

Zajac sa zatiaľ nepohybuje, nebeží. Pomocou komponentu `Clock` zabezpečte, aby bol zajac animovaný (t. j. aby vyzeral, že beží).

Pomôcky

Komponent `Clock` (časovač) sme už na generovanie udalostí v pravidelných intervaloch používali. Fázy animačného cyklu budeme striedať v reakcii na udalosť `Clock.Timer` (teda v okamihu, keď časovač „tikne“). Na uloženie poradového čísla aktuálnej fázy využijeme globálnu premennú.

V tomto prípade je vhodné nastaviť vlastnosť `Clock.TimerEnabled` tak, aby zajac začal bežať až po odštartovaní novej hry. Hodnota vlastnosti `Clock.TimerInterval` by mala byť dostatočne malá na to, aby animácia pohybujúcich sa nôh pôsobila plynulo.

Po spustení novej hry už zajac beží na mieste. Zatiaľ však nevie vyskočiť, naučíme ho to:

Úloha 2

Pomocou ďalšieho časovača zabezpečte, aby zajac vyskočil nahor *primerane rýchlo* a *dostatočne vysoko*. Výskok zajaca ovládajte dotykom prsta v ľubovoľnom miesta plátna (t. j. lúky, po ktorej zajac beží).

Vysvetlíme si

Výskokom zajaca rozumieme plynulé posúvanie sa zajaca z jeho základnej pozície smerom nahor a následné plynulé zostúpenie smerom nadol.

Sprajty sú schopné (ak sú aktívne) pohybovať sa určeným smerom aj samé (každých `ImageSprite.Interval` milisekúnd sa posunúť v smere `ImageSprite.Heading` o `ImageSprite.Speed` pixelov). Tento prístup sa nám ale teraz nehodí. Pohyb zajaca budeme riadiť vlastným nezávislým časovačom (do projektu vložíme ďalší komponent typu `Clock`).

Po zapnutí časovača sa zajac presunie niekoľkokrát nahor (počet opakovaní treba zladiť s počtom pixelov posunu, dosiahnuť optimálnu rýchlosť a výšku). Následne zmeníme smer pohybu na opačný a postupne zajaca vrátime naspäť do východiskovej pozície. Po návrate zajaca nadol časovač pre riadenie skoku vypneme a zapneme znovu časovať riadiaci animáciu behu. Počas skoku môže mať zajac nastavenú fázu 4 naznačujúcu skok.

Úloha 3

Podľa zadania by zajacovi mali ponad hlavu lietať mrkvy a pod nohami sa mu gúľať kapusty. Naprogramujte automatický pohyb týchto sprajtov.

Pomôcky

Experimentuje s rôznymi hodnotami vlastností `ImageSprite.Interval`, `ImageSprite.Speed` a umiestnením sprajtov vzhľadom na pozíciu zajaca. Dokáže zajac kapustu preskočiť? Má možnosť dočiahnuť na mrkvu? Keď zajac kapustu preskočí, bude pokračovať v pohybe ďalej až po pravý okraj obrazovky. Podobne, keď zajac mrkvu nechytí, bude mrkva letieť ďalej k pravému okraju. Oba tieto sprajty sa po dosiahnutí pravého okraja obrazovky majú objaviť vľavo.

Zajac už vie skákať, ale pri zrážke s mrkvou ani s kapustou sa nič neudeje. Pohyb kapusty aj mrkvy je pravidelný a preto ľahko predvídateľný.

Úloha 4

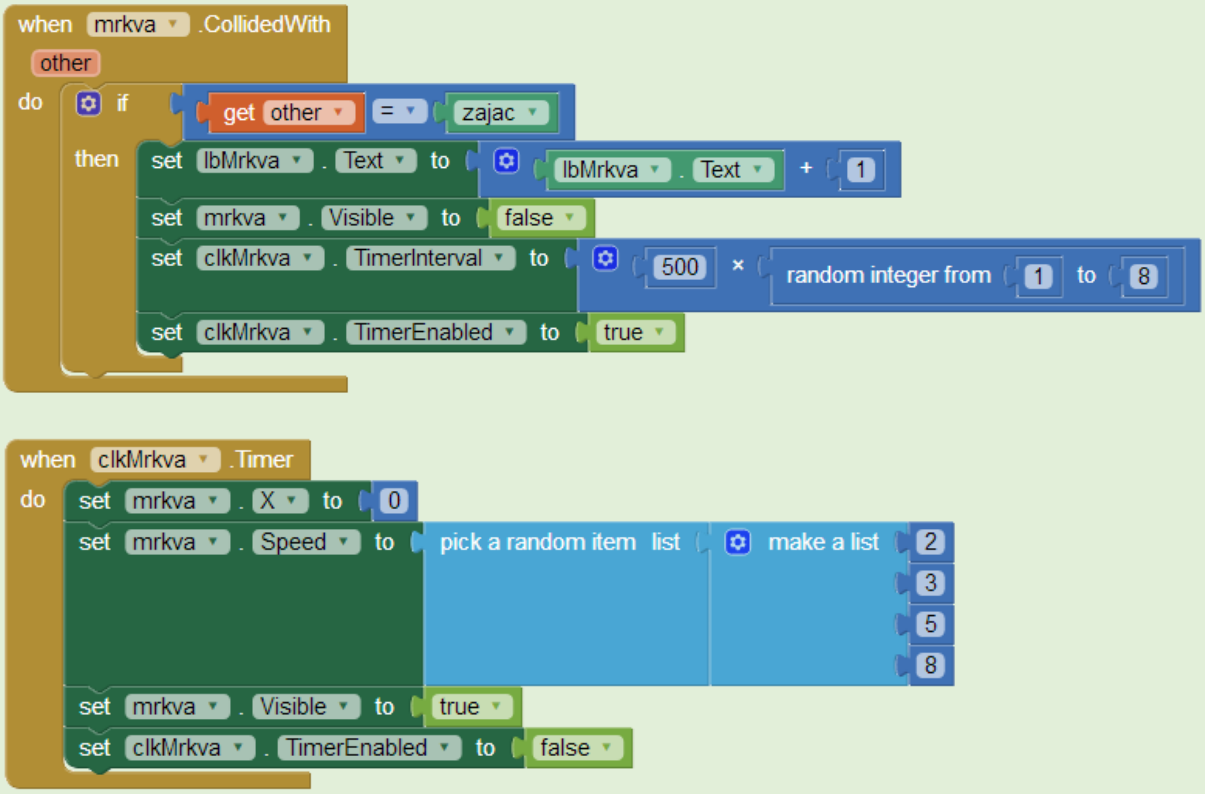
Upravte riešenie tak, aby sa ďalšia mrkva po zrážke so zajacom objavila na ľavom okraji obrazovky *o náhodne zvolený počet sekúnd* a letela vpravo *náhodne zvolenou rýchlosťou*. Kapusta by sa v prípade zrážky so zajacom mohla začať gúľať ihneď (t. j. v jej prípade nebudeme rozlišovať medzi nárazom na zajaca a nárazom na pravý okraj obrazovky).

Vysvetlíme si

Komponent `Clock` môžeme používať ako časovač generujúci udalosti v pravidelných intervaloch. Po zapnutí časovača časovač „tiká“ opakovane, až kým ho nevypneme. Na každé tiknutie môžeme zareagovať v udalostnom bloku `when Clock.Timer` (MIT, 2020).

Niekedy sa časovať hodí na naplánovanie vzniku jednorazových udalostí dôležitých pre riadenie behu aplikácie, zmenu stavu a pod. Časovaču môžeme nastaviť interval, po uplynutí ktorého sa vykonajú príkazy uvedené v jeho udalostnom bloku. K ďalšiemu tiknutiu časovača už nedôjde, keďže časovač na záver udalostného bloku vypneme.

Jednorazovou udalosťou (teda takou, ktorá nenastáva opakovane v pravidelných intervaloch) je v našej hre pokračovanie letu mrkvy z ľavej strany po predchádzajúcej zrážke so zajacom:



Naplánovať udalosť „objavenie sa mrkvy zľava“ znamená nastaviť pri zrážke príslušnému časovaču vlastnosti tak, aby tikol o požadovaný počet milisekúnd.

V ukážke zdrojového kódu vyššie vidíme, že sme časovaču `clkMrkva` nastavili vlastnosť `TimeInterval` na hodnotu, ktorá je výsledkom vyhodnotenia výrazu s náhodným číslom (možno pôjde o pol sekundy, možno až o 4 sekundy). Časovač sme vzápätí zapli nastavením jeho vlastnosti `Enabled` na `true`.

Druhý udalostný blok obsahuje príkazy na presun mrkvy z miesta zrážky na ľavú stranu obrazovky a nastavenie jej rýchlosti na náhodnú výberom niektorej hodnoty z pripraveného zoznamu možností. Posledným príkazom v tomto udalostnom bloku je vypnutie časovača `clkMrkva`.

Úloha 5

V komponentoch typu `Label` v hornom páse nad plátnom zobrazujte aktuálny stav počítadiel (počet chytených mrkiev, počet životov). Zabezpečte ukončenie hry a umožnite jej spustenie odznovu.

Pomôcky

Počítadlo súvisiace s gúľajúcimi sa kapustami slúži ako počítadlo životov. Po každej zrážke zajaca s kapustou sa jeho hodnota zníži o 1 (hráč stratí život). Po strate všetkých životov (napr. po desiatom zakopnutí o kapustu) sa hra skončí. Príkazy, ktoré je potrebné vykonať pri skončení hry alebo pri spúšťaní novej hry je vhodné umiestniť do samostatných procedúr.

Úloha 6

Po nazbieraní istého počtu mrkiev (na testovacie účely postačí napr. 5) bude možné zajaca kedykoľvek v priebehu hry (teda aj počas výskoku) ťahaním presúvať vpred alebo vzad po lúke. Vďaka tomu bude hráč schopný lepšie plánovať a realizovať výskoky a bude úspešnejší. Získanie schopnosti presúvať sa aj v horizontálnom smere budeme v našej vzorovej hre považovať za nový (druhý) level.

Otázky na zamyslenie

Zamyslite sa, čo znamená posúvať zajaca ťahaním vpred alebo vzad v horizontálnom smere. Ktorá súradnica sprajtu sa pri ťahaní bude meniť? Ktorý udalostný blok budeme potrebovať?

V titulkovom pruhu môžeme zobrazovať, v ktorom leveli sa hráč práve nachádza.

Úloha 7

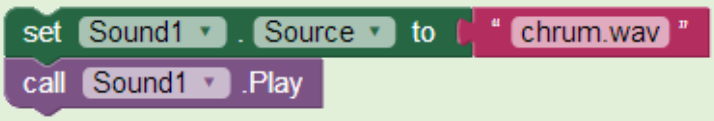
Doprogramujte v hre prehrávanie rytmickej hudby na pozadí a pridajte zvukové efekty pri zrážkach mrkvy a kapusty so zajacom.

Dovoľte tiež nastavovať niektoré parametre hry používateľovi (napr. počet životov a počet mrkiev, ktoré sú potrebné na prechod do ďalšieho levelu).

Pomôcky

Na prehrávanie hudby na pozadí je vhodné použiť komponent `Player`. V režime *Design* nastavíme vlastnosť `Player.Source` na názov súboru s rytmickou hudbou (napr. *podmaz.mp3*. V zdrojovom kóde doplníme na vhodné miesta príkazy na prehrávanie a zastavenie prehrávania:

Na prehrávanie krátkych zvukových efektov použite komponent `Sound`. Keďže chceme akusticky odlíšiť zakopnutie o kapustu od chytenia/zjedenia mrkvy, pri každej kolízii najprv nastavíme správny názov zdrojového súboru so zvukom a potom ho prehráme:

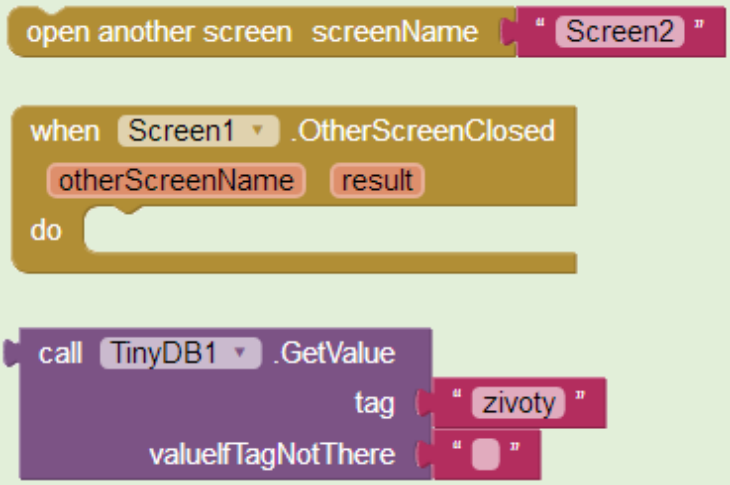


Nastavovanie parametrov hry by sa malo realizovať na samostatnej obrazovke. Pridajte preto do projektu ďalší komponent typu `Screen`, navrhnite vzhľad tejto obrazovky. Používateľovi môžete dať na výber rôzne hodnoty parametrov s využitím komponentu `Spinner`. Na hlavnú

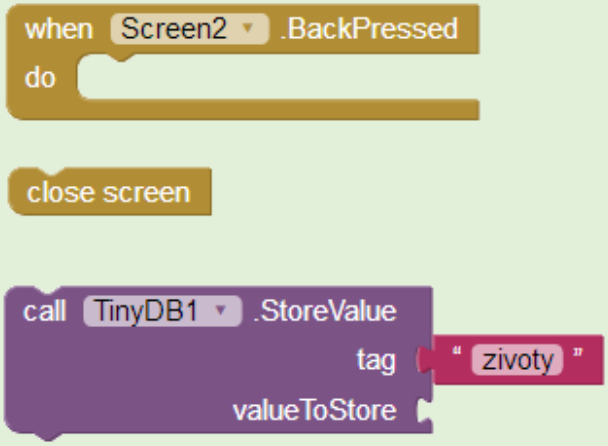
obrazovku sa môžete vrátiť stlačením tlačidla potvrdzujúceho zmenu alebo zabudovaným tlačidlom pre návrat späť.

Aktuálne hodnoty parametrov môžeme uchovávať v databáze `TinyDB`, ktoré obe obrazovky zdieľajú. Na implementovanie tejto funkcionality budete potrebovať aj tieto bloky:

Obrazovka `Screen1` (hlavná obrazovka aplikácie)



Obrazovka `Screen2` s ovládacími prvkami na výber hodnôt parametrov



Úloha 8

Porozmýšľajte nad námetom pre vlastnú hru alebo zrealizujte niektoré z nižšie uvedených námetov na vylepšenie základnej verzie hry.

Ako vylepšiť či rozšíriť našu aplikáciu?

Vzorovú hru by sme mohli vylepšovať ďalej. Uvádzame niekoľko nápadov, ktoré vás môžu inšpirovať aj pri tvorbe vlastnej hry:

- po spustení aplikácie nech sa zobrazí dialógové okno s informáciou o pravidlách a ovládaní hry,
- po každej x-tej mrkve, ktorú chytíme, získame pre zajaca jeden život späť,

- pridáme viac levelov, v každom ďalšom leveli sťažíme hráčovi situáciu (mrkvy začnú lietať rýchlejšie, kapusty sa gúľajú častejšie, pribudnú aj iné grafické objekty a pod.),
- zajac môže mať aj ďalšie schopnosti (zatiaľ s ním vieme skákať a presúvať ho v horizontálnom smere ťahaním, mohol by sa vedieť chvíľu vznášať vo vzduchu alebo zničiť kapustu vystrelením kalerábu),
- hra môže mať časový limit,
- pozadie za zajacom by sa mohlo v priebehu hry meniť,
- výsledky hráčov sa zapíšu do databázy a budú sa dať zobraziť na samostatnej obrazovke,
- v rozhraní v hornom páse bude k dispozícii tlačidlo na pozastavenie hry, aby ju hráč mohol prerušiť a dohrať neskôr,
- prehrávanie hudby na pozadí a zvukových efektov sa bude dať vypnúť,
- detegovanie kolízie zajaca a mrkvy bude presnejšie (nech zajac nechytí mrkvu aj vtedy, keď ju už má za ušami).

Zamyslime sa, čo sme sa naučili

- V aplikáciách dokážeme vhodne používať statické aj pohybujúce sa grafické objekty, zvukové efekty a viacero obrazoviek.
- Komponent `Clock` vieme zužitkovať na riadenie zmeny stavu grafických objektov (sprajtov), ale aj na plánovanie udalostí v hre.
- Sme schopní navrhnúť a naprogramovať vlastnú viacúrovňovú hru ovládanú dotykovými gestami a zverejniť ju v *Galérii* prostredia Ai2.

Bibliografia

Amin, Faisal. 2018. How to link Firebase to MIT App Inventor and Send or Retrieve data from Firebase database using MIT App Inventor 2. [Online] 5 2018. https://steemit.com/utopian-io/@faisalamin/how-to-link-firebase-to-mit-app-inventor-and-send-or-retrieve-data-from-firebase-database-using-mit-app-inventor-2.

Baštrng - Kubovčík Michal. 2018. SEPAR & SEPRP - KPR. [Online] 2018. https://www.youtube.com/watch?v=dh3QpBszxh0.

Baštrng - Michal Kubovčík. 2016-2020. *SEPRP – Sedlácká prvá pomoc.* [Online] 2016-2020. https://www.youtube.com/playlist?list=PLZSarXd25nWe0C7WAQH3mXkMieB4K-fwB.

Beer, Paula a Simmons , Carl. 2015. *Hello App Inventor! - Android programming for kids and the rest of us.* s.l. : Manning Publications Co, 2015.

beryko.cz. 2014. Senzory v mobilních telefonech od A do Z. [Online] 29. 11 2014. https://www.beryko.cz/blog/recenze/senzory-v-mobilnich-telefonech-od-a-do-z.html .

Edward, M. 2015. *Example of the new App Inventor "Responsive Design" Feature.* [Online] 17. 8 2015. https://appinventor.pevest.com/?p=836.

Google. 2020. Material Design. [Online] 2020. https://material.io/.

Groundspeak. 2000-2020. *Geocaching.* [Online] 2000-2020. https://www.geocaching.com/.

Chroust, Martin a Kůžel, Filip. 2015. Smartphony mají 19 smyslů. Znáte je všechny? [Online] 26. 2 2015. https://www.mobilmania.cz/clanky/smartphony-maji-19-smyslu-znate-je-vsechny/sc-3-a-1329584/default.aspx.

Irani, Romin. 2016. Tutorial : MIT App Inventor + Firebase. [Online] 9. 9 2016. https://rominirani.com/tutorial-mit-app-inventor-firebase-4be95051c325.

Košická záchranka. 2016. Základné životné funkcie a život ohrozujúce stavy. [Online] 2016. https://www.kezachranka.sk/306/Zakladne-zivotne-funkcie/.

Krishnendu, Roy. 2014. *App Inventor Activity-Calorie Tracker.* [Online] 3. 5 2014. https://youtu.be/Fq7B5WLRuHs.

Krupong. 2018. App Inventor 2 Ultimate - All in one App Inventor 2 personal server. [Online] 6. 6 2018. https://sourceforge.net/projects/ai2u/.

McGrath, Mike. 2014. *Building Android Apps In Easy Steps.* Second Edition. s.l. : In Easy Steps Limited, 2014.

Michaličková, Viera. 2016. *Programovanie mobilných aplikácií v prostredí MIT App Inventor 2.* Nitra : Univerzita Konštatnína Filozofa v Nitre, 2016.

MIT. 2015. *Responsive Design in App Inventor.* [Online] 15. 8 2015. http://ai2.appinventor.mit.edu/reference/other/responsiveDesign.html.

—. 2016. Experimental Components - App Inventor for Android. [Online] 2016. http://ai2.appinventor.mit.edu/reference/components/experimental.html.

—. 2020. MIT App Inventor. [Online] 2020. http://ai2.appinventor.mit.edu/.

—. 2018. MIT App Inventor 2. [Online] 2018. http://ai2.appinventor.mit.edu/.

—. 2016. The FirebaseDB Component (Experimental). [Online] 3 2016. http://ai2.appinventor.mit.edu/reference/other/firebase.html.

—. 2018. The MIT App Inventor Library: Documentation & Support. [Online] 2018. http://appinventor.mit.edu/explore/library.

Národné centrum zdravotníckych informácií. 2015-2020. *Národný portál zdravia.* [Online] 2015-2020. https://www.npz.sk/ .

Nield, David. 2017. All the Sensors in Your Smartphone, and How They Work. [Online] 23. 7 2017. https://fieldguide.gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002.

Pandey, Ranjan . 2015. *How to Make Android App for joggers using App Inventor 2 without writing codes.* [Online] 13. 10 2015. https://youtu.be/vMDObRR2MZ4.

prispievatelia Wikipédie. 2017. Comma-separated values. *Wikipédia, Slobodná encyklopédia.* [Online] 2017. https://sk.wikipedia.org/wiki/Comma-separated_values.

Pura Vida Apps. 2010-2020. A Simple Bluetooth Chat with App Inventor 2. [Online] 2010-2020. https://puravidaapps.com/btchat.php.

Refsnes Data. PHP 5 Tutorial. *W3Schools Online Web Tutorials.* [Online] https://www.w3schools.com/php/default.asp.

Škopek, Pavel. 2013. Techbox: váš telefon je prošpikovaný senzory. [Online] 13. 7 2013. https://mobilenet.cz/clanky/techbox-vas-telefon-je-prospikovany-senzory-12496 .

Šnajder, Ľubomír. 2018. Online pracovný list Spoznávajme Android mobilné zariadenie. [Online] 22. 6 2018. https://goo.gl/forms/8qnQjhmzkmUxLhRE2.

—. 2018. Zdieľaná nástenka na zozbieranie zoznamu obľúbených Android aplikácií. [Online] 22. 6 2018. https://padlet.com/lubomirsnajder/ml19124lcnke.

Varouch. 2013. Tajemství oceánu. [Online] 2013. https://www.wherigo.com/cartridge/details.aspx?CGUID=e561e853-6718-4dee-b2e3-4d5f663a0492.

Warner Bros. 2016. Heads Up! - The Best Charades Game! [Online] 2016. https://play.google.com/store/apps/details?id=com.wb.headsup&gl=SK.

Wikipédia. 2019. Bezvedomie. [Online] 2019. https://sk.wikipedia.org/wiki/Bezvedomie.

—. 2017. Dusenie (nedýchanie). [Online] 2017. https://sk.wikipedia.org/wiki/Dusenie_(nedýchanie).

—. 2018. Kockový poker. [Online] 2018. https://sk.wikipedia.org/wiki/Kockový_poker.

—. 2018. Nádychové potápanie. [Online] 2018. https://sk.wikipedia.org/wiki/Nádychové_potápanie.

—. 2020. Šok (medicína). [Online] 2020. https://sk.wikipedia.org/wiki/Šok_(medicína).

Wikipedie. 2017. Použitelnost. [Online] 2017. https://cs.wikipedia.org/wiki/Použitelnost#Definice.

Wolber, David, a iní. 2014. *App Inventor 2 - Create Your Own Android Apps.* s.l. : O'Reilly, 2014.

Wong, Emile. 2018. *Responsive Design: Drag a Canvas larger than screen size.* [Online] 18. 3 2018. http://appinventor.mit.edu/explore/blogs/karen/2018/03.html.

Yachtmeni. 2016. Apnea trénink. [Online] 2016. https://www.yachtmeni.cz/freediving-apnea/apnea-trenink/.

Yandex Translate. 2020. Developers. [Online] 2020. https://translate.yandex.com/developers.

Register pojmov

Textová príloha

Prílohy slúžia pri výučbe aj samoštúdiu vývoja aplikácie pre mobilné zariadenia v prostredí Ai2. Súčasťou príloh sú:

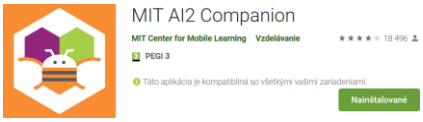
- **Inštalácia podporných nástrojov pre vývoj aplikácií v Ai2**
- **Prehľad komponentov a štandardných blokov Ai2**
- **Zdieľanie vyvinutej aplikácie ostatným používateľom**

Inštalácia podporných nástrojov pre vývoj aplikácií v Ai2

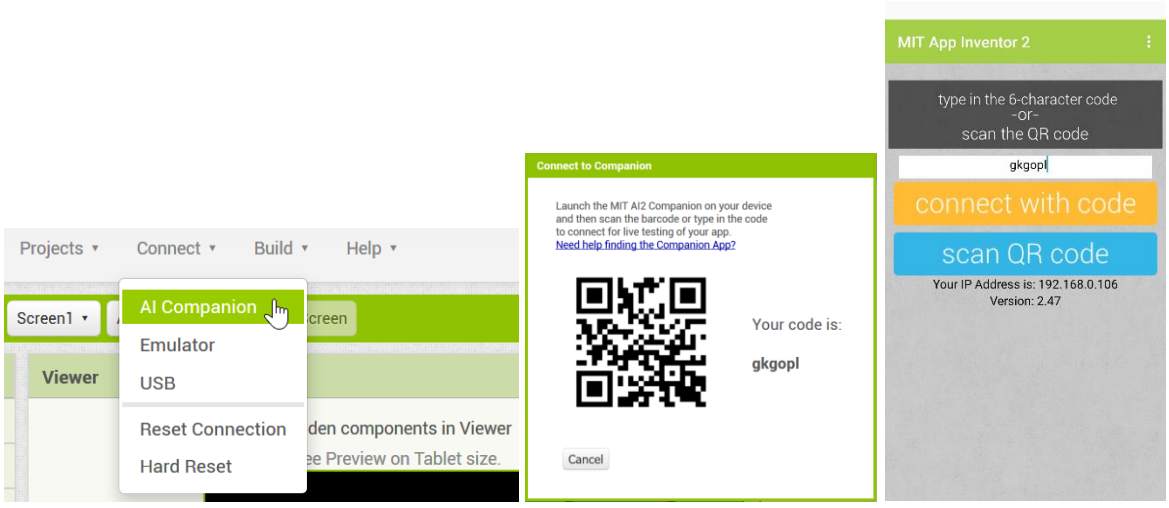
Inštalácia a používanie MIT Ai2 Companion

Aplikácia vyvíjaná na vývojovom počítači sa dá **naživo testovať na androidovom MZ**, ktoré je pripojené s vývojovým počítačom na **rovnakú wifi sieť**. Druhou alternatívou, ak nemáme androidové MZ, je použitie Emulátora priamo na obrazovke vývojového počítača (viac na webe http://appinventor.mit.edu/explore/ai2/setup-emulator.html). Treťou alternatívou je vzájomne prepojenie vývojového počítača a androidového MZ prostredníctvom USB kábla.

Odporúčame sa venovať prvej možnosti živého testovania aplikácie na androidovom MZ, na ktoré nainštalujeme aplikáciu **MIT Ai2 Companion**.



Živé testovanie aplikácie na androidovom MZ zabezpečíme tak, že najprv na vývojovom počítači vyberieme v hlavnej ponuke možnosť Connect/AI Companion, ktorá vygeneruje 6-znakový s odpovedajúcim QR kódom:



a na androidovom MZ spustíme aplikáciu MIT Ai2 Companion, v ktorej zapíšeme, resp. oskenujeme daný kód, čím sa následne prepoja obe zariadenia.

Takto môžeme na vývojovom počítači modifikovať zdrojový kód a pozorovať túto zmenu naživo na androidovom MZ.

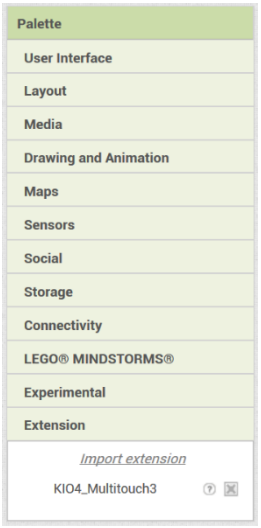
Inštalácia a používanie offline prostredia Ai2Offline

V prípade pomalého internetového pripojenia počas výučby v učebni s viacerými počítačmi alebo pri hodnotení viacerých žiackych projektov môžeme zvážiť inštaláciu vlastného offline Ai2 servera, napr. **AI2Offline** z webovej stránky https://sourceforge.net/projects/ai2offline/ (autor: Ramiro Prieto Alvarez).

Released /nb185a/AI2Offline_x64.exe	1 month ago
Released /nb185a/AI2Offline_nb185a.7z	1 month ago
Released /nb184/MIT_AI2_nb184.7z	2 months ago
Released /nb184/AI2Offline_x64.exe	2 months ago

Pozitívom takéhoto riešenia je rýchlejšia kompilácia APK súborov bez potreby internetového pripojenia. Nevýhodou je, že je to riešenie tretej strany, ktoré v určitom čase nemusí odpovedať aktuálnej online verzii od MIT umiestnenej na webovom sídle http://ai2.appinventor.mit.edu/.

Importovanie a používanie Extension



Pomocou tzv. **Extension** sa dajú rozšíriť funkcionality Ai2, napr. registrácia viacerých dotykov na plátne, generovanie tónov, využívanie svetelného senzora, Bluetooth LE komunikácia s IoT zariadeniami (Arduino 101, BBC micro:bit).

Rozšírenie uložené v súbore s príponou .aix (napr. com.KIO4_Multitouch3.aix) sa dá importovať do Ai2 projektu pomocou poslednej položky Extension v palete komponentov.

Po importovaní zdrojového súboru (.aia) obsahujúceho nejaké rozšírenie sa importuje aj toto rozšírenie s ním a zobrazí sa v položke Extension v palete komponentov daného projektu.

Podrobnejšie informácie o importovaní a používaní Extension sú uvedené na webe: http://ai2.appinventor.mit.edu/reference/other/extensions.html.

Further Extensions on other pages

1. MIT

- Vector Arithmetic Extension by Ethan: Takes in two vectors and can add them to return a result vector.
- Image Processor Extension by Justus: can do a weighted combine of two images, return the greyscale of an image.
- Sound Analysis Extension by Mouhamadou: analyzes the pitch of a sound through the microphone and returns it.
- Scale Detector Extension by Hal: Adds a multitouch scale gesture detector to a Canvas.
- Bluetooth Low Energy Extension (old) by MIT
- Bluetooth Low Energy Extension (new) by MIT
- Microbit Extension by MIT
- Rotation Detector Extension by Xinyue Deng, which reacts to two-finger rotation gestures.
- ChartMaker Extension by Kate Manning and Emily Kager
- Android Things Extension by Thilanka Munasinghe

2. Mad Robots

- Gyro Sensor Extension by Gareth
- Sound Extension by Gareth: offers Pitch effects, Volume Control, Speaker Balance

3. Makeblock

- Computer Vision Extension by Makeblock: for color detection, face detection and feature analysis (using online API).
- mBot Extension by Makeblock: to control Makeblock mBots.

4. thunkable

- Microsoft Emotion Recognizer by Thunkable
- Microsoft Image Recognizer by Thunkable

Zoznam viac ako stovky rozšírení od viacerých autorov sú uvedené na webe: https://puravidaapps.com/extensions.php.

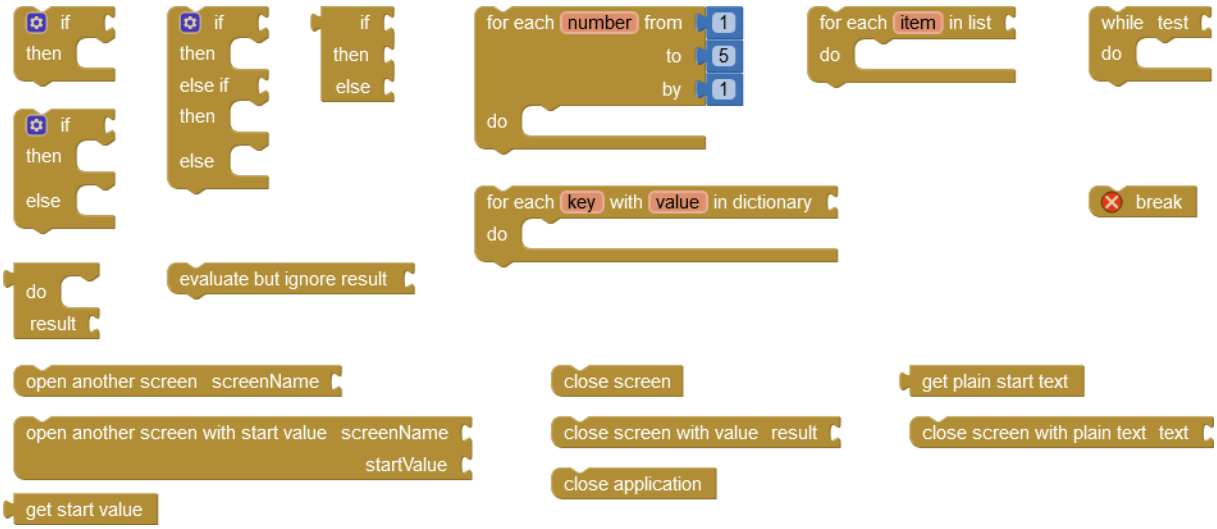
Prehľad komponentov a štandardných blokov Ai2

Prehľad komponentov Ai2 (nb185a, 2. 9. 2020)

User Interface	Layout	Media	Drawing and Animation
Button CheckBox DatePicker Image Label ListPicker ListView Notifier PasswordTextBox Slider Spinner Switch TextBox TimePicker WebViewer	HorizontalArrangement HorizontalScrollArrangement TableArrangement VerticalArrangement VerticalScrollArrangement	Camcorder Camera ImagePicker Player Sound SoundRecorder SpeechRecognizer TextToSpeech VideoPlayer YandexTranslate	Ball Canvas ImageSprite
Maps	**Sensors**	**Social**	**Storage**
Circle FeatureCollection LineString Map Marker Navigation Polygon Rectangle	AccelerometerSensor BarcodeScanner Barometer Clock GyroscopeSensor Hygrometer LightSensor LocationSensor MagneticFieldSensor NearField OrientationSensor Pedometer ProximitySensor Thermometer	ContactPicker EmailPicker PhoneCall PhoneNumberPicker Sharing Texting Twitter	CloudDB File TinyDB TinyWebDB
Connectivity	**LEGO® MINDSTORMS®**		**Experimental**
ActivityStarter BluetoothClient BluetoothServer Serial Web	NxtDrive NxtColorSensor NxtLightSensor NxtSoundSensor NxtTouchSensor NxtUltrasonicSensor NxtDirectCommands	Ev3Motors Ev3ColorSensor Ev3GyroSensor Ev3TouchSensor Ev3UltrasonicSensor Ev3Sound Ev3UI Ev3Commands	FirebaseDB

Prehľad štandardných blokov Ai2 (nb185a, 2. 9. 2020)

Control



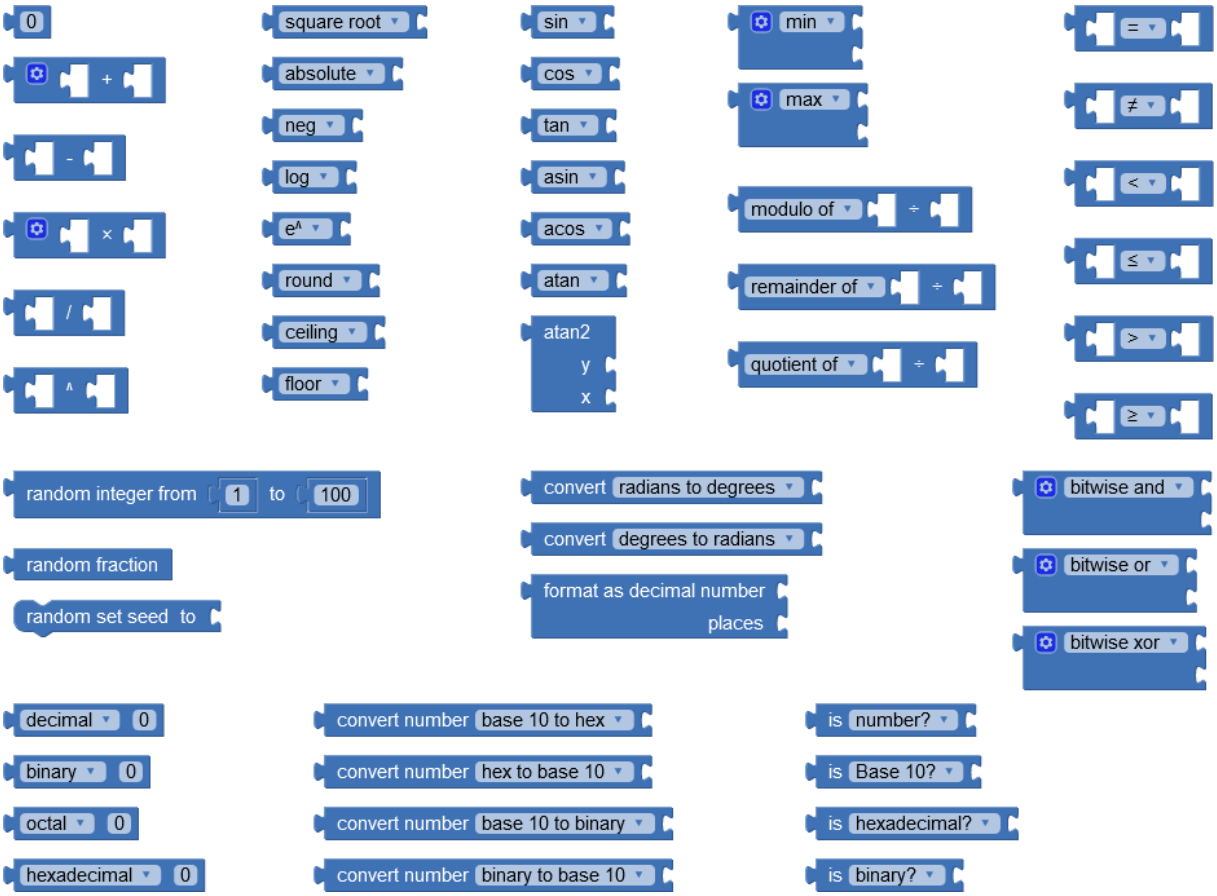
Variables

Procedures

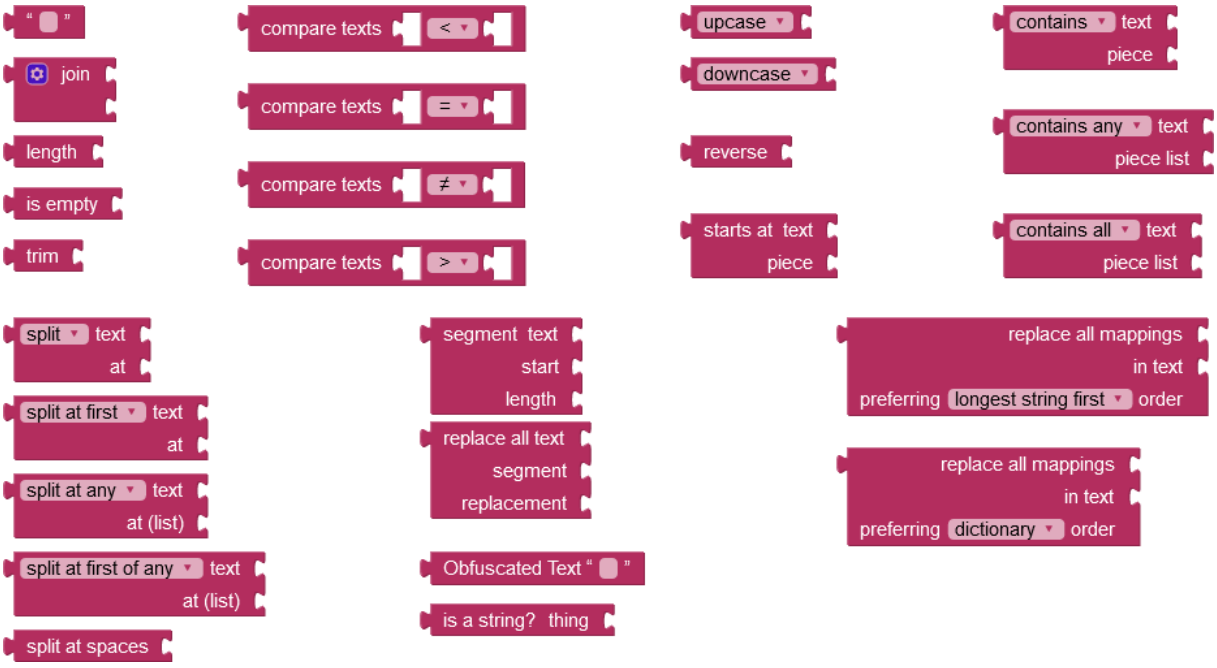
Colors

Logic

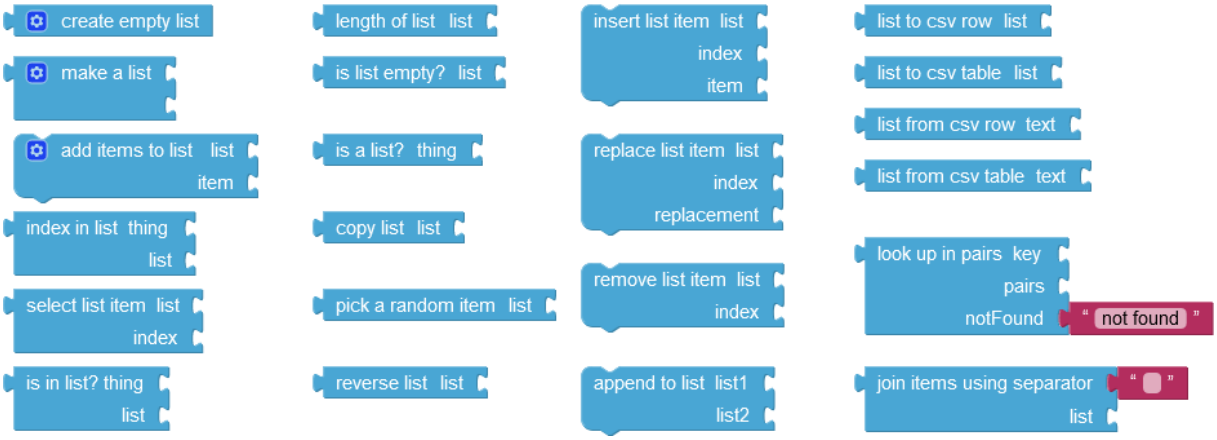
Math


0
+
-
×
/
^
square root
absolute
neg
log
e^
round
ceiling
floor
sin
cos
tan
asin
acos
atan
atan2
y
x
min
max
modulo of ÷
remainder of ÷
quotient of ÷
=
≠
<
≤
>
≥
random integer from 1 to 100
random fraction
random set seed to
convert radians to degrees
convert degrees to radians
format as decimal number places
bitwise and
bitwise or
bitwise xor
decimal 0
binary 0
octal 0
hexadecimal 0
convert number base 10 to hex
convert number hex to base 10
convert number base 10 to binary
convert number binary to base 10
is number?
is Base 10?
is hexadecimal?
is binary?

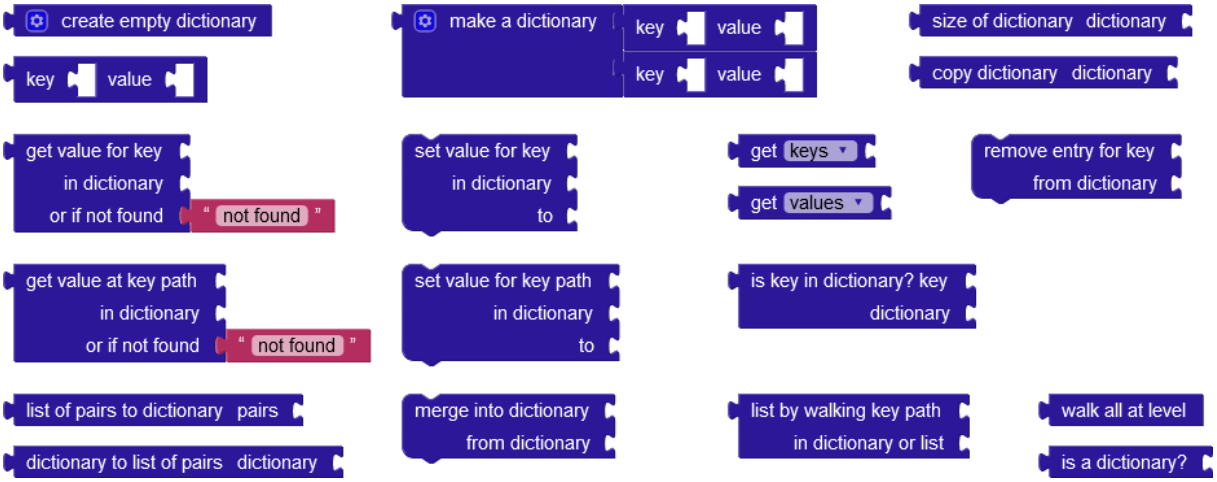
Text


" "
join
length
is empty
trim
compare texts <
compare texts =
compare texts ≠
compare texts >
upcase
downcase
reverse
starts at text piece
contains text piece
contains any text piece list
contains all text piece list
split text at
split at first text at
split at any text at (list)
split at first of any text at (list)
split at spaces
segment text start length
replace all text segment replacement
Obfuscated Text " "
is a string? thing
replace all mappings in text preferring longest string first order
replace all mappings in text preferring dictionary order

Lists



Dictionaries



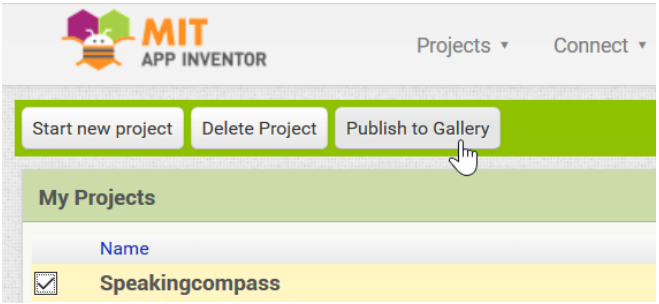
Zdieľanie vyvinutej aplikácie ostatným používateľom

Ak sme vyvinuli aplikáciu pre OS Android, môžeme ostatným používateľom dať zdieľať jej **zdrojový kód** (súbor s príponou **AIA**) alebo jej **inštalačný balík** (súbor s príponou **APK**).

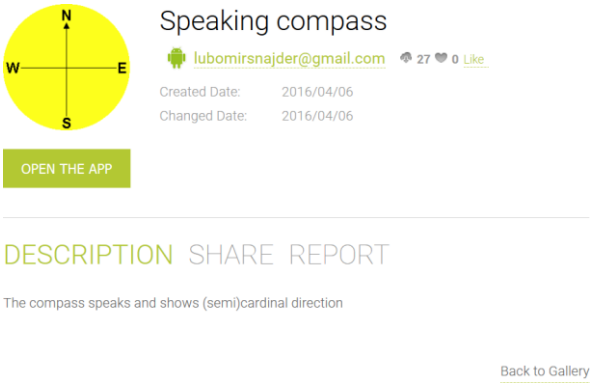
Zdieľanie zdrojového kódu má význam pre žiakov pri ich samoštúdiu cudzích zdrojových kódov a pre učiteľa pri hodnotení odovzdaných žiackych zdrojových kódov. Na importovanie cudzích zdrojových kódov, resp. exportovanie vlastných zdrojových kódov použijeme možnosti z hlavnej ponuky:

- Projects/**Import** project (.aia) from my computer, resp.
- Projects/**Export** selected project (.aia) to my computer

Zdrojový súbor (.aia) môžeme poslať e-mailom, alebo ho publikovať na vlastnej webovej stránke či zdieľať v Galérii Ai2 (v prehľade projektov stlačením tlačidla Publish to Gallery).



V Galérii Ai2 svoj projekt patrične zdokumentujeme a uchováme si jeho URL: http://ai2.appinventor.mit.edu/?galleryId=5852196860329984.



Z daného URL si vieme zostaviť QR kód, ktorý vygenerujeme pomocou niektorého z online generátorov, napr. http://goqr.me/.

Hotový inštalačný balík (súbor s príponou APK) môžeme tiež poslať e-mailom, či publikovať ho na svojej webovej stránke a zostaviť QR kód pre URL daného súboru.

Náš projekt môžeme publikovať aj v Google Play, podrobnejšie informácii o tom nájdeme na webovej stránke http://appinventor.mit.edu/explore/ai2/google-play.html.