Bez znalosti programování

App Inventor

Vytvořte si vlastní aplikaci pro Android

David Wolber, Hal Abelson,
Ellen Spertus & Liz Looney

O'REILLY®

computer press®

David Wolber, Hal Abelson, Ellen Spertus, Liz Looney

App Inventor

Computer Press
Brno
2014

App Inventor

David Wolber, Hal Abelson, Ellen Spertus, Liz Looney

Překlad: Jiří Huf
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN 978-80-251-4195-3

Vydalo nakladatelství Computer Press v Brně roku 2014 ve společnosti Albatros Media a. s.
se sídlem Na Pankráci 30, Praha 4. Číslo publikace 18 469.

1. vydání

ALBATROS MEDIA a.s.

Obsah

Předmluva

Konzumní společnost nám poskytuje bezpočet příležitostí bavit se a občas se i něco naučit. Ve velké většině případů se však jedná o pasivní aktivity. Ale to nevadí. Všichni si rádi občas oddechneme a pobavíme se. To nám však nestačí. Kromě spotřeby nás těší ještě něco – tvořit. Mám na mysli potěšení a pocit hrdosti, když nakreslíme obrázek, postavíme model letadla či něco upečeme.

Současné vyspělé technologie (mobilní telefony, tablety, televize apod.), jejichž prostřednictvím dnes hltáme zábavu a informace, jsou pro nás často velkou neznámou. Netušíme, jak fungují, a i když nám některé z nich umožní nakreslit obrázek, natočit video apod., nejsou to ze své podstaty tvůrčí média. Jinými slovy, většina lidí aplikace, které na takových zařízeních spouští, neumí napsat.

Co kdybychom tohle dokázali změnit? Co kdybychom dokázali z každodenně používaných nástrojů, jakými jsou mobilní telefony, vytáhnout tvůrčí potenciál? Co kdyby bylo napsat aplikaci pro mobilní telefon stejně snadné jako nakreslit obrázek či upéct chleba? Co kdybychom dokázali přemostit propast mezi spotřebními produkty a tvůrčími médii?

Zaprvé bychom tak z těchto nástrojů sňali závoj tajemna. Již by pro nás nebyly neuchopitelné, ale dokázali bychom s nimi pracovat. Dokázali bychom jim porozumět. Kdybychom pro ně uměli vytvářet aplikace, vybudovali bychom si k nim méně pasivní, zato však tvůrčí vztah, a mohli bychom si s nimi pohrávat mnohem rafinovaněji a účelněji.

Když jsme se s Halem Abelsonem o nápadu na prostředí App Inventor bavili poprvé, kladli jsme důraz na jedinečnou motivační sílu, kterou by mohly mobilní telefony vzdělávat. Hal uvažoval, zda bude možné pomocí této síly uvést studenty do problematiky počítačové vědy. Ve třídě Davea Wolbera jsme si pak uvědomili, že jsme dali vzniknout ještě něčemu většímu: App Inventor začal studenty konzumenty přetvářet na tvůrce. Studenty tvorba aplikací pro vlastní mobilní telefony bavila! Když pak jeden z Daveových studentů napsal jednoduchou, avšak velmi užitečnou aplikaci, díky které řidiči za jízdy nemusí psát textové zprávy, začali jsme uvažovat nad tím, co by se stalo, kdyby mobilní aplikace dokázali psát nejen profesionální programátoři.

Hodně jsme se proto snažili, aby bylo ovládání App Inventoru jednodušší a zábavnější. Rovněž jsme ho vybavili více funkcemi. A v práci neustáváme – App Inventor je stále v beta verzi a máme s ním velké plány.

Autoři této knihy jsou prvotřídními pedagogy a programátory. Rád bych jim zde osobně poděkoval za práci, kterou věnovali psaní, testování a dokumentaci prostředí App Inventor for Android, a samozřejmě také za to, že napsali tuto výtečnou knihu.

Nenechte se pobízet, nechte průchod tvůrčímu zápalu a napište si vlastní aplikaci.

– Mark Friedman
Vedoucí projektu App Inventor for Android, Google

Poděkování

Z hlediska výuky, což je hlavní motivátor App Inventoru, můžeme říci, že se prostřednictvím počítačů učíme poznávat velké myšlenky. App Inventor je v oblasti počítačů a vzdělávání součástí otevřeného hnutí, které se započalo prací Saymoura Paperta a skupiny MIT Logo Group v šedesátých letech dvacátého století a jehož vliv i dnes nalezneme v mnohých akcích a programech cílených na rozvoj počítačového myšlení.

Design App Inventoru staví na výsledcích výzkumů v oblasti výuky výpočetních systémů a na práci společnosti Google v oblasti online vývojových prostředí. Vizuální programovací rozhraní je úzce spjato s programovacím jazykem Scratch (MIT). Jeho konkrétní implementace je zde založena na technologii Open Blocks, šířené v rámci vzdělávacího programu Massachusettského technologického institutu (MIT), Scheller Teacher Education Program, a vychází z absolventské práce studenta MIT Ricarosa Roquea. Rádi bychom poděkovali Ericu Klopferovi a Danielu Wendelovi ze Schellerova programu za to, že nám technologii Open Blocks zpřístupnili, ale také za pomoc při práci s ní. Kompilátor překládající jazyk vizuálních bloků na Android využívá rozhraní Kawa Language Framework a Kawa verzi programovacího jazyka Scheme, které vytvořil Per Bothner a které je šířeno organizací Free Software Foundation pod licencí GNU Operating System.

Autoři by rádi poděkovali týmům Googlu a App Inventoru za podporu při práci a za pomoc při výuce na Sanfranciské univerzitě, Mills College i MIT. Zvláštní poděkování patří technickému vedoucímu App Inventoru Marku Friedmanovi, vedoucí projektu Karen Parkerové, ale také Sharon Perlové a Debby Wallachové.

Zvláštní poděkování přináleží i redaktorům v nakladatelství O'Reilly Courtney Nashové a Brianovi Jepsonovi. Za zpětnou vazbu a novou perspektivu děkujeme rovněž Kathy Riutzelové, Brianovi Kernighanovi, Debby Wallachové a Rafikiovi Caiovi.

Konečně bychom rádi poděkovali i svým manželkám a manželům: Elleninu manželovi Keithovi Goldenovi, Halově manželce Lynn Abelsonové, Lizině manželovi Kevinovi Looneyovi a Davidově manželce Minervě Novoaové. Čerstvá maminka Ellen je taktéž vděčna za pomoc pečovateli Neilovi Fullagarovi.

Úvodem

Právě běžíte svou obvyklou trasu, a v tom dostanete nápad na úžasnou novou aplikaci. Po zbytek cesty pak už neuvažujete, jaký budete mít čas, protože máte myšlenky jen pro novou aplikaci. Ale jak ji napsat? Nejste programátor a navíc by vám to trvalo roky, stálo spoustu času, peněz… A navíc takovou aplikaci už nejspíše někdo napsal. A najednou nevíte, co s nápadem dělat dál.

Nyní si představte trochu jiný svět, ve kterém k psaní aplikací nepotřebujete mít roky zkušeností s programováním. Svět, ve kterém mohou aplikace psát umělci, vědci, humanisté, zdravotníci, právníci, hasiči, maratonci, fotbaloví trenéři a lidé ze všech oblastí lidských činností. Představte si svět, ve kterém můžete myšlenku uvést v prototyp, aniž byste museli najímat programátory. Kde můžete vytvářet aplikace přímo sobě na míru, kde si výpočetní výkon svého telefonu můžete přizpůsobit vlastním potřebám.

Tak vypadá svět App Inventoru, nového nástroje Googlu určeného pro programování mobilních aplikací. Prostředí používá vizuální „blokové" programování, se kterým mohou slavit úspěch i děti. App Inventor boří bariéry, které při programování mobilních aplikací pro systém Android vyvstávají. Co si takhle napsat hru, ve které budete jako postavy vy a vaši přátelé? Anebo co si napsat aplikaci, která vám ve tři hodiny odpoledne připomene, že máte koupit mléko, pokud se zrovna nacházíte v blízkosti nějakého obchodu s potravinami? Nebo aplikace s kvízem pro vaši drahou polovičku, která bude ve skutečnosti žádostí o sňatek? „Otázka 4: Vezmeš si mě? Stisknutím tlačítka nabídku přijmeš a odešleš SMS." Takovou aplikaci už skutečně jeden muž napsal – a přítelkyně si ho vzala!

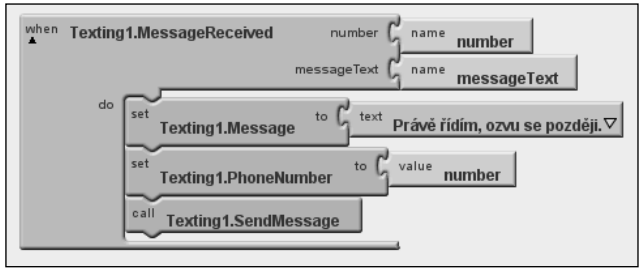
Blokový jazyk pro mobilní telefony

App Inventor je vizuálně zpracovaný přesunovací nástroj, který slouží k tvorbě mobilních aplikací na platformě Android. Pomocí GUI nástroje sestavíte uživatelské rozhraní (jak bude aplikace vypadat) a následně pomocí „bloků", které složíte dohromady jako skládačku, definujete, jak se bude aplikace chovat.

Na obrázku 0.1 vidíte několik bloků v úvodní fázi stavby aplikace, kterou vytvořil Daniel Finnegan, vysokoškolský student, který nikdy dříve neprogramoval. Dokážete říci, co aplikace dělá?

Aplikace slouží jako záznamník. Spustíte ji, když máte řídit, a ona automaticky odpovídá na přijaté textové zprávy.

Blokům porozumíte lépe než tradičnímu programovacímu jazyku, nebudete se muset nic učit a budete si jen klást otázky typu: Mohu si nechat přečíst přijaté zprávy nahlas? Mohu nějak přizpůsobit odpověď? Mohu napsat aplikaci, která umožní lidem pomocí textových zpráv hlasovat, například kdo se stane příští českou superstar? Odpověď na všechny tyto otázky je „ano". V této knize vám ukážeme, jak na to.



Obrázek 0.1 Bloky určují v App Inventoru funkcionalitu aplikace

Co můžete v prostředí App Inventor dělat?

Hrát si

Psát si vlastní aplikace je zábava a App Inventor se vás bude snažit vybízet ke zkoumání. Stačí prostředí otevřít ve webovém prohlížeči, připojit telefon a začít skládat bloky, které vidíte na obrázku 0.1. Vytvářenou aplikaci pak uvidíte přímo v telefonu. Takže zatímco programujete, můžete kamaráda e-mailem poprosit, aby vám poslal SMS a aplikaci tak vyzkoušel. Nebo můžete pomocí nové aplikace ovládat robota LEGO NXT či můžete telefon odpojit, vyjít ven a zkontrolovat, zda aplikace správně určuje, kde se telefon nachází.

Tvořit prototypy

Máte nápad na novou aplikaci? Nemusíte si ho psát na kapesník ani čekat, než se zcela rozplyne. Můžete si vytvořit její prototyp. Prototyp je nehotovým a jen částečně funkčním modelem. Vyjádřit myšlenku slovy je jako napsat kamarádovi či milované osobě báseň. Prototyp vytvořený v prostředí App Inventor můžete chápat jako báseň, kterou píšete investorovi. Svým způsobem vám App Inventor může v případě mobilních aplikací posloužit jako elektronický kapesník.

Vytvářet aplikace na míru

V současném světě mobilních aplikací máme na výběr jen z již existujících aplikací. Kdo z nás si někdy nestěžoval na aplikaci, kterou by měl rád přizpůsobenu vlastnímu způsobu práce? V prostředí App Inventor můžete vytvářet přesně takové aplikace, jaké chcete. Ve třetí kapitole si vytvoříte hru, ve které budete sbírat body, když prstem trefíte náhodně se pohybujícího krtka. Nemusíte však použít obrázek krtka z návodu. Namísto něj můžete vložit obrázek svého sourozence – tedy udělat něco, co má význam pouze pro vás, nikoliv pro ostatní. V osmé kapitole budeme vytvářet kvízovou aplikaci, která se táže na americké prezidenty, ale vy si ji můžete snadno přizpůsobit vlastním otázkám, ať už na téma oblíbené hudby či rodinné historie.

Vyvíjet plnohodnotné aplikace

App Inventor není jen prostředím pro tvorbu prototypů či designérem rozhraní. Můžete v něm vytvářet plnohodnotné aplikace pro širokou uživatelskou základnu. Jazyk prostředí vám nabídne všechny základní programátorské stavební prvky, jakými jsou smyčky a podmínky, avšak v podobě bloků.

Učit se a učit ostatní

Ať už jste na základní, střední nebo vysoké škole, App Inventor je skvělým výukovým prostředím. Výborně se hodí pro výuku IT, ale rovněž je úžasným nástrojem pro výuku matematiky, fyziky, podnikání a vlastně jakéhokoliv předmětu. Klíčem je, že se učíte tvůrčím způsobem. Namísto toho, abyste si museli pamatovat nazpaměť vzorečky, si například vytvoříte aplikaci, která vyhledá nejbližší nemocnici (nebo nákupní centrum). Namísto abyste museli psát písemnou práci na téma dějiny černochů, vytvoříte si multimediální kvízovou aplikaci s videoukázkami a proslovy Martina Luthera Kinga a Malcolma X. Jsme přesvědčeni, že App Inventor může, spolu s touto knihou, posloužit po celou dobu studia jako vynikající výukový nástroj.

Proč se v prostředí App Inventor dobře pracuje

Většina lidí říká, že se App Inventor snadno používá díky svému přesunovacímu rozhraní. Co to ale znamená? Proč se App Inventor snadno používá?

Nemusíte si pamatovat a zadávat příkazy

Jednou z věcí, která začínajícím programátorům činí největší potíže, je zadávat příkazy a čekat, až počítač vyplivne nesrozumitelné chybové hlášky. To mnohé ze začátečníků odradí dříve, než se dostanou k zábavnějším částem programování, k řešení problémů.

Volíte si z nabízených voleb

V App Inventoru jsou komponenty a bloky tříděny do přihrádek, které můžete ihned používat. Programujete tak, že vyhledáte blok – který vám pomůže ujasnit si požadovanou funkcionalitu – a přetáhnete ho do programu. Nebudete si muset pamatovat příkazy ani nahlížet do příručky programování.

Ne všechny bloky do sebe zapadají

Namísto toho, aby prostředí App Inventor trestalo programátory chybovými hláškami, v mnohých chybách jim zabraňuje už preventivně. Pokud například funkční blok očekává na vstupu číslo, nebudete do něj moci zapojit text. Nevyhnete se tak všem chybám, ale jistě vám to pomůže.

S událostmi pracujete přímo

Tradiční programovací jazyky dostávaly svou podobu v době, kdy se programování podobalo kuchyňským receptům, protože představovalo sled pokynů. V případě grafických rozhraní, a zvláště pak v případě mobilních aplikací, kde k jednotlivým událostem může dojít kdykoliv (například přijetí SMS či telefonního hovoru), většina programů již recepty nepřipomíná. Funguje spíše jako sada obslužných rutin událostí. Obslužná rutina události je způsob vyjádření, které říká: „Pokud dojde k této události, aplikace se zachová následovně." V tradičních jazycích, jakým je Java, musíte vědět, jak pracovat s třídami, objekty a speciálními objekty, které nazýváme naslouchacími procesy. Bez nich totiž nebudete schopni vyjádřit ani jednoduchou událost. V prostředí App Inventor však můžete říci „Když uživatel klepne na tohle tlačítko, ..." anebo „Když telefon přijme textovou zprávu, ..." pouhým přetažením podmínkového bloku „Když".

Jaké můžete vytvářet aplikace?

V prostředí App Inventor můžete vytvářet řadu různých typů aplikací. Nechte pracovat představivost a tvořte nejrůznější zábavné a užitečné aplikace.

Hry

Lidé často začínají u her, jakou je Krtek (kapitola 3), a u aplikací, které umožňují kreslit po tvářích přátel vtipné obrázky (kapitola 2). Postupně budete moci vytvářet vlastní verze složitějších her, například Pac-Man a Space Invaders. Můžete dokonce využít senzory telefonu a postavičkami pohybovat nakláněním telefonu (kapitola 5).

Vzdělávací software

Vytváření aplikací se neomezuje na jednoduché hry. Můžete také vytvářet informativní a vzdělávací aplikace. Můžete si napsat kvízovou aplikaci (kapitola 8), která vám i spolužákům pomůže naučit se na test, či dokonce jinou kvízovou aplikaci (kapitola 10), která uživatelům umožní sestavovat si vlastní kvízy. (Pamatujte na všechny rodiče, kteří by za takovou aplikaci na dlouhých cestách s dětmi byli neskonale vděční!)

Aplikace sledující, kde se nacházíte

App Inventor zpřístupňuje GPS senzor, takže si můžete vytvářet i aplikace, které budou vědět, kde se právě nacházíte. Budete si moci vytvořit aplikaci, která vám pomůže vzpomenout si, kde jste zaparkovali auto (kapitola 7), aplikaci, která vám při koncertě či konferenci ukáže, kde se právě nachází vaši přátelé, anebo vlastního virtuálního průvodce po vlastní škole, pracovišti či muzeu.

Aplikace využívající pokročilé technologie

Můžete si vytvořit aplikaci na snímání čárových kódů, aplikace, které umí mluvit, naslouchat (rozpoznat řeč), přehrávat hudbu, skládat hudbu (kapitola 9), přehrávat video, zjišťovat pozici a pohyby telefonu, fotit a volat. Chytré telefony jsou v oblasti technologií něco jako švýcarské kapesní nože a my můžeme být rádi, že se nám práci s technologiemi snaží v prostředí App Inventor usnadnit celá skupina pracovníků společnosti Google.

SMS aplikace

Aplikace, díky níž řidiči nemusí za volantem psát textové zprávy (kapitola 4), je pouze jedním z příkladů aplikací pracujících se SMS. Můžete si rovněž sestavit aplikaci, která bude milované osobě pravidelně odesílat zprávu „chybíš mi", anebo aplikaci pro koordinaci velkých akcí (kapitola 11). Chtěli byste mít aplikaci, která by vašim přátelům umožnila hlasovat textovými zprávami, například o budoucí české superstar? V prostředí App Inventor si ji můžete vytvořit.

Aplikace pro řízení robotů

V kapitole 12 si ukážeme, jak vytvořit aplikaci, která může sloužit jako ovladač LEGO robota. Telefon díky ní můžete použít jako dálkové ovládání anebo si ho můžete naprogramovat, aby sloužil jako „mozek", který si robot bude nosit s sebou. Robot bude s telefonem komunikovat prostřednictvím technologie bluetooth. Komponenty prostředí App Inventor vám umožní vytvářet i další aplikace pro řízení bluetooth zařízení.

Složité aplikace

App Inventor zásadním způsobem boří bariéru, která stojí před vstupem do světa programování, a i líbivou, složitou aplikaci vám umožní vytvořit během několika hodin. Jeho jazyk však nabízí i smyčky, podmínky a další programovací a logické nástroje, které jsou nezbytné při vytváření složitějších aplikací. Divili byste se, kolik zábavy vám mohou při vytváření aplikací přinést logické problémy.

Aplikace s přístupem k webu

App Inventor rovněž umožní vašim aplikacím komunikovat s webovým prostředím. Budete si moci sestavit aplikace, které vytáhnou data z Twitteru, z RSS kanálu či z vyhledávače v obchodě Amazon a umožní vám tak z čárového kódu knihy zjistit na Internetu její cenu.

Kdo může vytvářet aplikace?

App Inventor je volně dostupný všem. Je to online nástroj (nebudete ho tedy mít přímo na počítači) a budete s ním komunikovat prostřednictvím internetového prohlížeče. Nepotřebujete

k němu dokonce ani telefon, aplikace si můžete vyzkoušet na vestavěném emulátoru systému Android. V lednu roku 2011 mělo prostředí desítky tisíc aktivních uživatelů, kteří v něm vytvořili stovky tisíc aplikací.

Kdo jsou oni lidé? Byli to původně programátoři? Někteří ano, ale většina z nich nikoliv. Jedním z nejnázornějších příkladů jsou kurzy, které vyučoval na Sanfranciské univerzitě (USF) jeden ze spoluautorů, David Wolber. App Inventor se na této škole vyučuje v rámci běžné výuky IT, která se zaměřuje primárně na studenty obchodu a humanitních věd. Mnozí ze studentů si tento předmět zapíší, protože buď matematiku nemají rádi, či z ní mají strach. Absolvováním kurzu však splní nutný matematický základ. Naprostou většinu z nich nikdy ani nenapadlo, že by mohli někdy napsat počítačový program.

I přesto, že tito studenti nemají s programováním žádnou předchozí zkušenost, s prostředím App Inventor nemají problémy a dokážou v něm vytvářet skvělé aplikace. Jeden z posluchačů anglického jazyka vytvořil první aplikaci starající se o SMS namísto řidiče, dva posluchači komunikace vytvořili aplikaci, která uživateli řekne, kde má auto, a jeden posluchač mezinárodních studií napsal aplikaci pro organizování velkých událostí (kapitola 11). Když Wolberovi jednou večer, dlouho po konzultačních hodinách, zaťukal na dveře jeden ze studentů výtvarných umění s otázkou, jak napsat `while` smyčku, bylo zřejmé, že App Inventor značně přepsal hranice programování.

Důležitost tohoto prostředí si uvědomila i média. New York Times označil App Inventor jako „prostředí pro tvorbu aplikací pro kutily". San Francisco Chronicle píše o studentech Sanfranciské univerzity a říká, že „Google předkládá tvorbu aplikací širokým masám". Časopis Wired v článku o Danielu Finneganovi, autorovi aplikace pro automatické odesílání SMS za jízdy, napsal, že „Finneganův příběh ukazuje, že nadešel čas, aby se počítačové programování stalo věcí veřejnou".

Dnes je již App Inventor dobře známý. Učí ho nyní na středních školách, na odpoledním kurzu zaměřeném na technologie a inovace pro dívky v San Francisku, na Lakeside School v Seattlu, ale také coby součást nových předmětů na několika univerzitách. Stránky a fórum (*http://appinventor.googlelabs.com/forum/*) prostředí App Inventor již navštěvují tisíce nadšenců, podnikatelů, lidí, kteří chtějí prostřednictvím aplikace požádat o ruku, i ostatních zvědavců. Chcete se zapojit? S programováním nemusíte mít vůbec žádné zkušenosti!

Styly používané v této knize

Tato kniha pracuje s následujícími typografickými styly:

Tučný text

Označuje bloky, které se v prostředí App Inventor objevují v programech.

Kurzíva

Označuje e-mailové adresy, adresy URL, cesty k souborům a zdůrazňuje prvně použité termíny.

`Neproporcionální text`

Označuje kód v jazyce Python a názvy komponent, vlastností, proměnných a funkcí.

Otestujte svou aplikaci: Tato ikona označuje instrukce k testování vyvíjené aplikace.

Poznámka: Tato ikona označuje tip, návrh či poznámku.

Jak s touto knihou pracovat

Tuto knihu lze vnímat jako učebnici určenou středním a vysokým školám, ale také jako příručku pro ambiciózní vývojáře aplikací. Kniha je rozdělena do dvou oddílů: sady návodů k vytvoření konkrétních aplikací a oddílu manuálu prostředí App Inventor, který slouží spíše jako klasická učebnice programování. Příklady budou postupně složitější a složitější. V první kapitole začneme aplikací, v níž budete moci klepnout na obrázek kočky, která následně zamňouká, ale vytvoříme si taky webovou aplikaci, která dokáže prohledat knížku a najít o ní údaje pomocí webové služby obchodu Amazon (kapitola 13).

Z koncepčního hlediska je vhodnější si příklady vyzkoušet, ale jakmile se s prostředím trochu seznámíte, budete je moci přeskočit. V návodech najdete podrobné pokyny a snímky bloků, které vám pomohou. Rovněž zde najdete odkazy na konkrétní oddíly manuálu Inventoru, v nichž si své znalosti budete moci upevnit.

Jednou z výhod papírové knihy je skutečnost, že prostředí App Inventor obsadí většinu obrazovky a na další okno elektronické příručky byste neměli dostatek prostoru. Představujeme si, že knihu budete mít při procházení příkladů a zkoušení stále vedle sebe. Rovněž doufáme, že se vám kniha zalíbí natolik, že si ji vezmete s sebou, i když půjdete od počítače, abyste se mohli začíst do koncepčnějších kapitol manuálu Inventoru.

Učitelům a žákům může tato kniha sloužit jako učebnice úvodu do počítačových věd, ale také jako učebnice do libovolného předmětu, ve kterém se studenti učí praktickým zkoušením. Z našich zkušeností plyne, že nejúčinnější je sled návod → diskuse → tvorba. Takže můžete začít tím, že studentům uložíte vytvořit několik aplikací popsaných v návodech, aniž byste od studentů očekávali, že budou aplikace vytvářet hned zcela přirozeně. Následně můžete studentům ke studiu zadat kapitolu z oddílu manuálu a pomalu pokračovat diskusí ve třídě a přednáškou. Ve třetí fázi pak podnítíte zvědavost: Nechte studenty vytvořit některou z navržených variací aplikací, které najdete na koncích návodů, aniž byste jim podali podrobné pokyny. Nakonec nechte studenty, aby přišli s vlastními nápady a aplikacemi.

Soubory pro každou z kapitol si můžete společně s příklady kódu rovněž stáhnout na následující adrese: *http://examples.oreilly.com/0636920016632/*.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu přeložilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Zdrojové kódy ke knize

Z adresy *http://knihy.cpress.cz/K2131* si po klepnutí na odkaz Soubory ke stažení můžete přímo stáhnout archiv s ukázkovými kódy.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte. Ostatní uživatele tak můžete ušetřit frustrace a nám můžete pomoci zlepšit následující vydání této knihy.

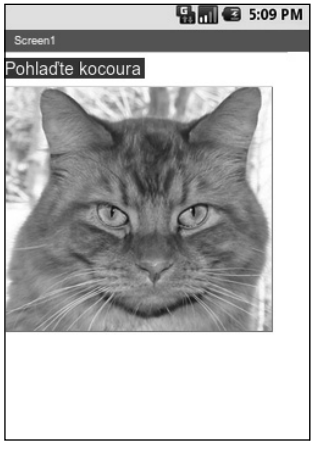
Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2131* po klepnutí na odkaz Soubory ke stažení.

KAPITOLA 1

Ahoj, kocoure

V této kapitole:
- Co se naučíte
- Prostředí App Inventor
- Design komponent
- Definování chování komponent
- Sbalení aplikace ke stažení
- Sdílení aplikace
- Variace
- Shrnutí

V této kapitole začneme s tvorbou aplikací. Představíme si v ní klíčové prvky App Inventoru – Designér komponent (Component Designer) a Editor bloků (Blocks Editor) – a při přípravě vaší první aplikace, AhojKocoure, si ukážeme několik základních kroků. Na konci kapitoly budete schopni vytvořit si vlastní aplikaci.



Obrázek 1.1 Aplikace AhojKocoure

V nových počítačových systémech se tradičně jako první program spouštěl „Hello World“ (Ahoj, světe), který ukázal, zda je vše správně propojeno. Tato tradice se datuje od sedmdesátých let minulého století, kdy tento program napsal Brian Kernighan v programovacím jazyce C v Bellových laboratořích. (Brian nyní přednáší v Googlu v týmu App Inventoru!) V App Inventoru dokážou i ty nejjednodušší aplikace více, než jen zobrazit text: Přehrají hudbu anebo zareagují na dotyk. Takže začněme přímo s něčím zábavnějším. Vaší první aplikací bude „AhojKocoure“, obrázek kocoura, který zamňouká, když se ho dotknete, a zapřede, když telefonem zatřesete.

Co se naučíte

V této kapitole se budeme věnovat následujícím tématům:

- Stavbě aplikací výběrem komponent a následným definováním toho, co a kdy mají dané komponenty provádět.
- Výběru komponent v Designéru komponent (Component Designer). Některé komponenty jsou na displeji telefonu viditelné, jiné nikoliv.
- Vkládání médií (zvuku a obrázků) do aplikací přímo z počítače.
- Práci s Editorem bloků (Blocks Editor), v němž budeme sestavovat bloky určující chování komponent.
- Testování aplikací v App Inventoru. Uvidíte tak, jak bude aplikace vypadat a jak se bude chovat v telefonu již při vývoji.
- Balíčkování vytvořených aplikací a jejich stažení do telefonu.

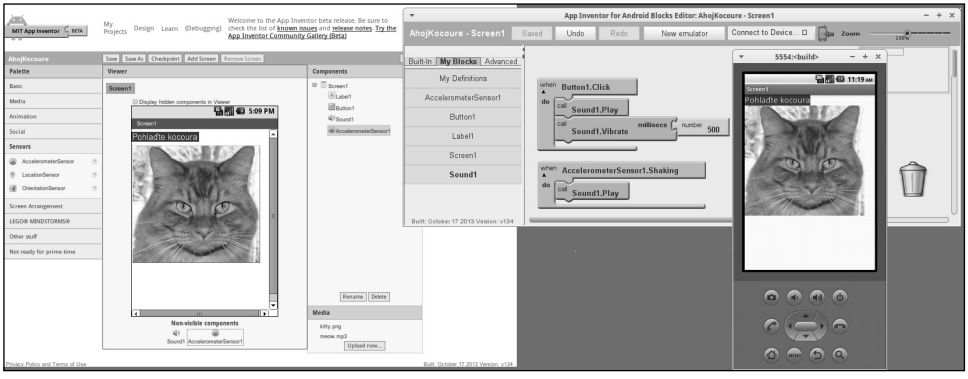
Prostředí App Inventor

App Inventor si můžete připravit dle pokynů na adrese *http://appinventor.googlelabs.com/learn/setup/*. Prostředí sice spouštíme primárně ve webovém prohlížeči, ale přesto si budete muset do počítače stáhnout příslušný software a upravit nastavení telefonu. Obvykle trvá celá operace pouze několik minut, i když někdy mohou nastat komplikace při nastavování ovladačů zařízení konkrétních telefonů Android. Pokud máte s telefonem potíže, doporučujeme vám použít emulátor prostředí Android, který je k App Inventoru přibalen.

Programovací prostředí App Inventoru nabízí tři základní části, viz obrázek 1.2:

- Designér komponent (Component Designer), který vidíte po levé straně na obrázku 1.2, je spuštěn v okně prohlížeče. Budete v něm vybírat komponenty pro své aplikace a nastavovat jim vlastnosti.
- Editor bloků (Blocks Editor) se spouští v samostatném okně – při práci s aplikací je často nejjednodušší přenést si ho napravo od okna Designéra komponent. V Editoru bloků definujeme chování komponent.
- V telefonu si můžete vyvíjené aplikace spustit a vyzkoušet. Pokud nemáte telefon s operačním systémem Android, můžete si aplikace otestovat v emulátoru (viz pravý spodní roh obrázku 1.2), který je součástí systému.

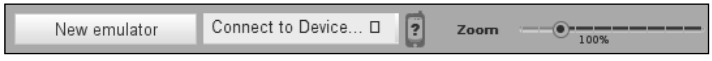
App Inventor spustíte na adrese *http://appinventor.googlelabs.com*. Pokud prostředí spouštíte poprvé, zobrazí se vám stránka projektů, která bude prázdná, protože jste doposud žádný projekt nevytvořili. Nový projekt vytvoříte klepnutím na New (nový) v levé horní části stránky. Zadejte název „AhojKocoure“ (jedno slovo bez mezer) a klepněte na OK.



Obrázek 1.2 Designér komponent, Editor bloků a emulátor systému Android

První okno, které se vám otevře, bude Designér komponent (Component Designer). Jakmile se tak stane, klepněte na Open the Blocks Editor (otevřít Editor bloků) v nabídce napravo. V samostatném okně se pomocí nástroje Java Web Start otevře Editor bloků. Zobrazené zprávy týkající se prostředí Java vás nemusí trápit – App Inventor pomocí tohoto prostředí, které byste na počítači již měli mít nainstalováno, spouští Editor bloků (Blocks Editor). Spuštění trvá zpravidla přibližně půl minuty.

Nedojde-li k chybě, zobrazí se okno Editoru bloků a vy uvidíte v horní části obrazovky napravo dvě tlačítka, viz obrázek 1.3.



Obrázek 1.3 Připojte telefon k počítači a klepněte na New emulator (nový emulátor), poté klepněte na Connect to Device (připojit k zařízení)

Pokud vlastníte telefon se systémem Android a máte k dispozici USB kabel, připojte telefon k počítači a klepněte na Connect to Device (připojit k zařízení). Pokud si však chcete vyzkoušet vlastní aplikace v emulátoru, klepněte na New emulator (nový emulátor) a počkejte asi půl minuty, dokud se emulátor nespustí. Až bude emulátor připraven k použití, klepněte na Connect to Device (připojit k zařízení), App Inventor spustí danou aplikaci v emulátoru.

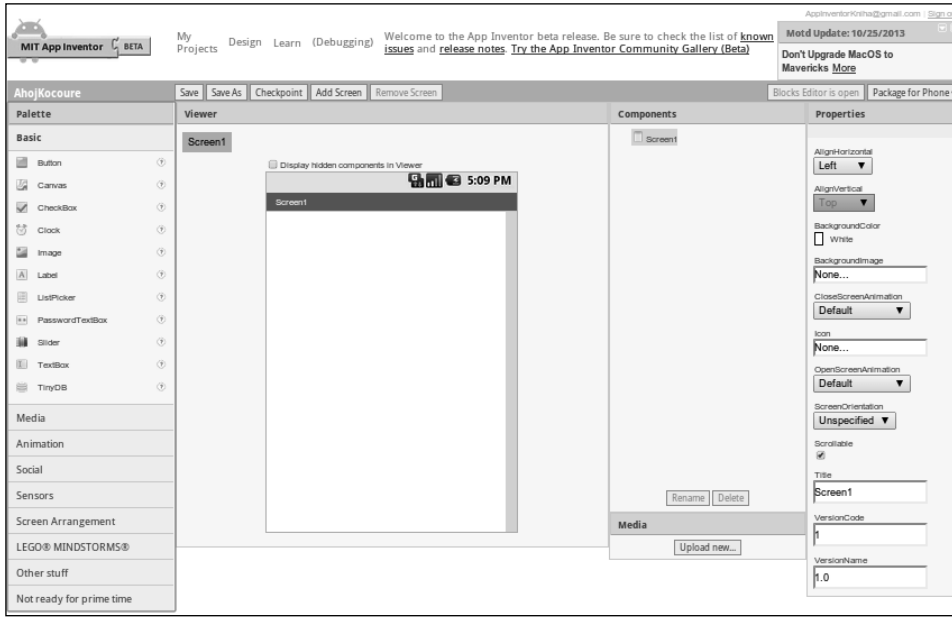
Pokud se vše podaří, mělo by se vám zobrazit okno Designéra komponent (Component Designer), okno Editoru bloků (Blocks Editor), a pokud jste si zvolili emulátor, tak i jeho okno. (Vaše obrazovka by měla vypadat podobně jako na obrázku 1.2 výše, ale měla by být z většiny prázdná.) Pokud narazíte na komplikace, projděte si pokyny k instalaci na adrese *http://appinventor.googlelabs.com/learn/setup/*.

Design komponent

Prvním nástrojem, s nímž budete pracovat, bude Designér komponent (Component Designer) nebo prostě jen Designér. Komponenty jsou prvky, jejichž kombinováním skládáme aplikace,

podobně jako ingredience receptu. Některé komponenty jsou velmi prosté, například komponenta `Label`, která na obrazovce zobrazí text, či komponenta `Button`, která klepnutím vyvolává akci. Jiné komponenty mohou být komplexnější: kreslicí plátno `Canvas` uchová obrázky či animace, akcelerometr, pohybový senzor, který funguje jako ovládání konzole Wii a rozpozná pohyb či zatřesení telefonem. Komplexnějšími komponentami jsou i prvky, které posílají textové zprávy, přehrávají hudbu a video, načítají údaje z webových stránek apod.

Když otevřete Designér, otevře se tak, jak to vidíte na obrázku 1.4.



Obrázek 1.4 Designér komponent

Designér je rozdělen do několik částí:

- Uprostřed najdeme bílou oblast nazvanou Viewer (prohlížeč). Zde umísťujeme komponenty a tvoříme z nich mapy, podle nichž chceme sestavovat aplikace. Prohlížeč nám vzhled aplikace nastíní jen v hrubých nárysech, takže kupříkladu řádek textu může být v aplikaci zalomen v jiném místě, než v jakém je zalomen v okně prohlížeče. Chcete-li vidět, jak bude aplikace vypadat skutečně, budete si ji buď muset stáhnout do telefonu (to si ukážeme zanedlouho v oddíle „Sbalení aplikace ke stažení“) anebo ji zobrazit v emulátoru App Inventoru.
- Nalevo od prohlížeče najdete paletu (Palette), kterou tvoří seznam komponent, z nichž vybíráme. Paleta je rozdělena do oddílů. Prozatím budou viditelné pouze základní komponenty, ale ostatní komponenty si můžete prohlédnout klepnutím na hlavičky oddílů Media, Animation (animace) apod.

- Napravo od prohlížeče najdeme seznam komponent (Components), který uvádí komponenty použité v daném projektu. Na seznamu se objeví všechny komponenty, které přetáhnete do prohlížeče. Momentálně se v projektu nachází pouze jedna komponenta: `Screen1`, která zastupuje samotnou obrazovku telefonu.
- Pod seznamem komponent se nachází oblast zobrazující média (obrázky a hudbu) použitá v projektu. V našem projektu se doposud žádná média nenachází, ale brzy nějaká vložíme.

Zcela napravo pak najdete oddíl ukazující vlastnosti komponent (Properties). Když na některou z komponent klepnete v prohlížeči, zobrazí se zde její vlastnosti. Vlastnosti jsou dílčí prvky komponent, které lze upravovat. (Například klepnutím na komponentu `Label` zobrazíte vlastnosti týkající se barvy, textu, fontu apod.) Nyní jsou zobrazeny vlastnosti obrazovky (`Screen1`), mezi nimiž najdeme barvu pozadí, obrázek na pozadí a titulek.

Pro aplikaci AhojKocoure budeme potřebovat dvě viditelné komponenty (tedy komponenty, které v aplikaci skutečně uvidíme): nápis `Label` s textem „Pohlaďte kocoura“ a tlačítko `Button` s obrázkem kočky. Rovněž budeme potřebovat neviditelnou komponentu `Sound`, která umí přehrávat zvuky, například „mňau“, a komponentu `Accelerometer`, která zaregistruje zatřepání telefonem. Nebojte se – všechny tyto komponenty si představíme postupně krok za krokem.

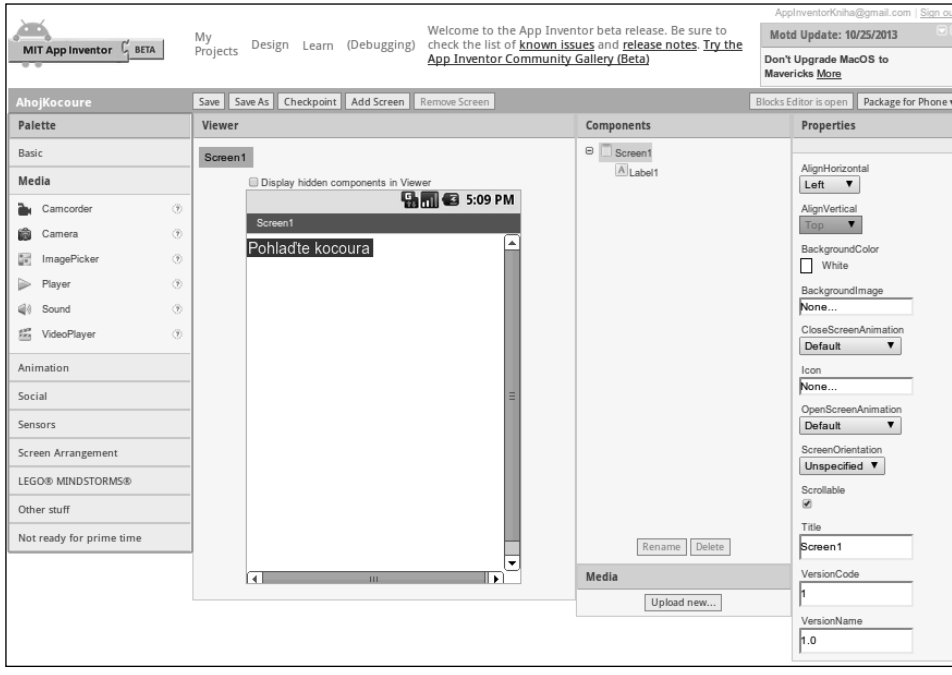
Nápis

Jako první budeme přidávat komponentu `Label` (nápis):

1. Přejděte do palety, klepněte na `Label` (na seznamu je asi na páté pozici) a přetáhněte ji do prohlížeče. V prohlížeči se zobrazí obdélníkový tvar s nápisem „Text for Label1“ (text nápisu).
2. Podívejte se do pole vlastností napravo od Designéra. Najdete zde vlastnosti nápisu. Přibližně v polovině seznamu se nachází vlastnost `Text`, která nabízí textové pole pro zadání nápisu. Sem zadejte „Pohlaďte kocoura“ a stiskněte Enter. V prohlížeči se text nápisu aktualizuje.
3. Klepnutím do pole vlastnosti `BackgroundColor`, které momentálně ukazuje hodnotu None (žádná), změňte barvu pozadí. V nabízeném seznamu vyberte modrou (Blue). Rovněž změňte barvu textu, `TextColor`, nápisu na žlutou (Yellow). Nakonec změňte velikost písma, `FontSize`, na 20.

Designér by nyní měl vypadat jako na obrázku 1.5.

Ujistěte se, zda máte připojen telefon, a otevřete Editor bloků (Blocks Editor). Nápis, který jste přidali v Designéru, by se vám měl zobrazit v telefonu. V App Inventoru tvoříte aplikace přímo výběrem komponent v Designéru. Proto ihned vidíte, jak aplikace vypadá. Tento proces, jenž označujeme jako živé testování (live testing), má přímý dopad na chování, které v Editoru bloků komponentám definujete. Za chvíli to uvidíte.



Obrázek 1.5 Aplikace má nyní nápis

Přidání tlačítka

Obrázek kocoura používáme v aplikaci AhojKocoure v podobě tlačítka `Button` – vytvoříme běžné tlačítko a necháme ho zobrazit obrázek kocoura. Nejprve vložíme běžné tlačítko. Přejděte proto v Designéru do palety a klepněte na komponentu `Button` (první v seznamu komponent). Přetáhněte ji do okna prohlížeče a umístěte ji pod nápis. V prohlížeči se zobrazí obdélníkové tlačítko. Přibližně za deset sekund by se mělo tlačítko objevit i v telefonu. Klepněte na tlačítko – myslíte, že se něco stane? Ne, protože jsme tlačítku doposud neřekli, co má provádět. Tohle je třeba se při práci s App Inventorem naučit: Jakmile přidáte v Designéru libovolnou komponentu, musíte se přesunout do Editoru bloků (Blocks Editor) a napsat kód, který s komponentou něco provede (k tomu se dostaneme, až do Designéra přidáme všechny komponenty).

Nyní máme tlačítko, které při klepnutí přehraje zvukový efekt. Chceme však, aby tlačítko vypadalo jako kocour, ne jako obyčejný obdélník. Z tlačítka uděláme kocoura následovně:

1. Nejprve si budeme muset stáhnout obrázek kocoura a uložit ho do počítače. Můžete tak učinit na stránce této knihy na adrese *http://examples.oreilly.com/0636920016632/*. Obrázek se nazývá kitty.png. (.png je běžný formát obrázků, podobně jako .jpg či .gif. V App Inventoru budou fungovat všechny jmenované typy a totéž platí i o běžných zvukových souborech, jako například .mpg či .mp3.) Taktéž si zde můžete stáhnout zvukový soubor, který budeme potřebovat, meow.mp3.

2. V okně vlastností byste měli vidět vlastnosti tlačítka. Pokud tomu tak není, klepněte v prohlížeči na obrázek tlačítka, zobrazíte tak jeho vlastnosti napravo. V okně vlastností klepněte na oblast pod obrázkem (momentálně je v ní uvedeno None, žádná). Zobrazí se pole s tlačítkem Add (přidat).
3. Klepněte na tlačítko Add (přidat), zobrazí se vám možnost nahrát soubor. Klepněte na Choose File (vybrat soubor), vyhledejte uložený soubor kitty.png a klepněte na tlačítko OK.
4. V horní části obrazovky se vám zobrazí žlutá oznámení, že se soubor nahrává na server AppInventor. Asi za půl minuty zpráva i nahrávací pole zmizí a soubor kitty.png byste měli vidět jako vlastnost tlačítka. Soubor uvidíte v okně Designéra i v oddílu Media, pod seznamem komponent. Když se podíváte na telefon, všimněte si, že se obrázek zobrazil i tam – tlačítko má nyní podobu kocoura.
5. Možná jste si také všimli, že obrázek na telefonu nese označení „Text for button 1“. To byste jistě v aplikaci neviděli rádi, takže upravte vlastnost `Text` tlačítka `Button1` například na „Pohladťe kocoura“, anebo text zcela odstraňte.

Designér by nyní měl vypadat stejně jako na obrázku 1.6.

Obrázek 1.6 Aplikace s nápisem a tlačítkem v podobě obrázku

Přidání mňoukání

Kocour v aplikaci zamňouká, když se ho dotknete. Budeme tedy muset přidat zvuk mňoukání a naprogramovat chování tlačítka tak, aby při stisknutí tlačítka došlo k přehrání zvuku:

1. Pokud jste si ještě z adresy *http://examples.oreilly.com/0636920016632/* nestáhli soubor meow.mp3, učiňte tak nyní.
2. Přejděte na paletu nalevo od okna Designéru a klepněte na hlavičku s označením Media, rozbalíte tak sekci médií. Uchopte komponentu Sound a přetáhněte ji do prohlížeče. Ať ji umístíte kamkoliv, zobrazí se ve spodní části prohlížeče s označením „Non-visible components" (neviditelné komponenty). Neviditelné komponenty jsou objekty, které aplikaci nějakým způsobem slouží, ale nezobrazují se v jejím grafickém uživatelském rozhraní.
3. Klepněte na komponentu `Sound1`, zobrazíte tak její vlastnosti. Vlastnost `Source` nastavte na meow.mp3. Budete muset postupovat stejně, jako když jste nahrávali obrázek kocoura. Jakmile skončíte, měli byste v sekci Media vidět dvě položky, kitty.png a meow.mp3.

Nyní byste v projektu měli mít stejné komponenty, jaké vidíte v tabulce 1.1.

Tabulka 1.1 Komponenty vložené do aplikace AhojKocoure

Typ komponenty	Skupina palety	Název komponenty	Účel
`Button`	Basic	`Button1`	Po stisknutí kocour zamňouká.
`Label`	Basic	`Label1`	Zobrazí text „Pohlaďte kocoura".
`Sound`	Media	`Sound1`	Přehraje zvuk mňoukání.

Definování chování komponent

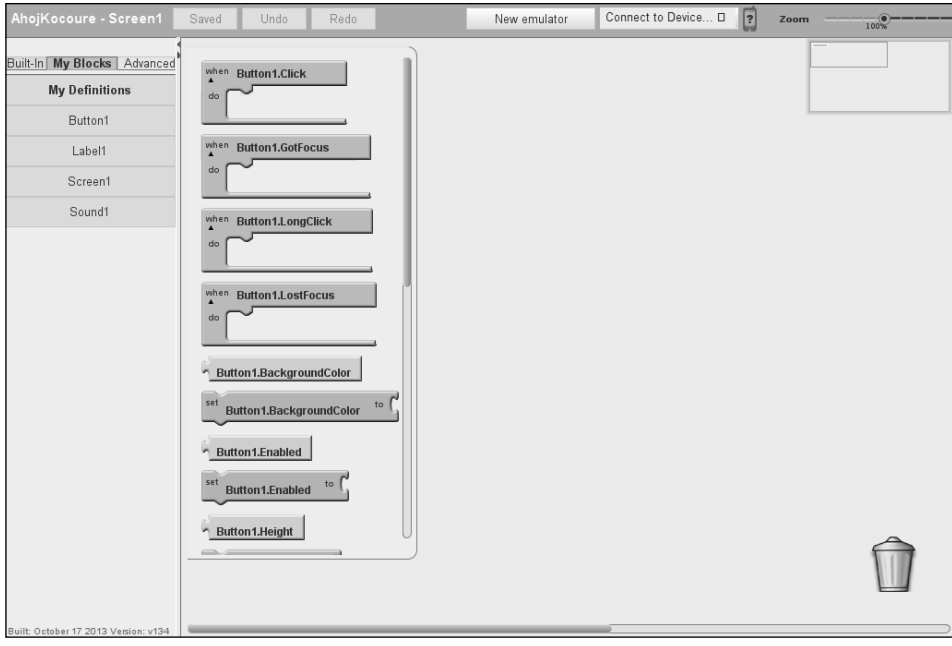
Právě jsme coby stavební bloky první aplikace přidali komponenty `Button`, `Label` a `Sound`. Nyní necháme kocoura při stisknutí tlačítka zamňoukat. Uděláme to v Editoru bloků (Blocks Editor). Pokud nemáte editor ještě otevřen, klepněte na Open the Blocks Editor (otevřít Editor bloků) v pravém horním rohu Designéra.

Podívejte se do okna Editoru bloků (Blocks Editor). Zde budeme komponenty instruovat, co a kdy mají provádět. Nyní tlačítku řekneme, že má při stisknutí zamňoukat. Pokud komponenty přirovnáme k ingrediencím v receptu, můžeme bloky chápat jako pokyny k tomu, jak podle receptu uvařit.

Mňoukající kocour

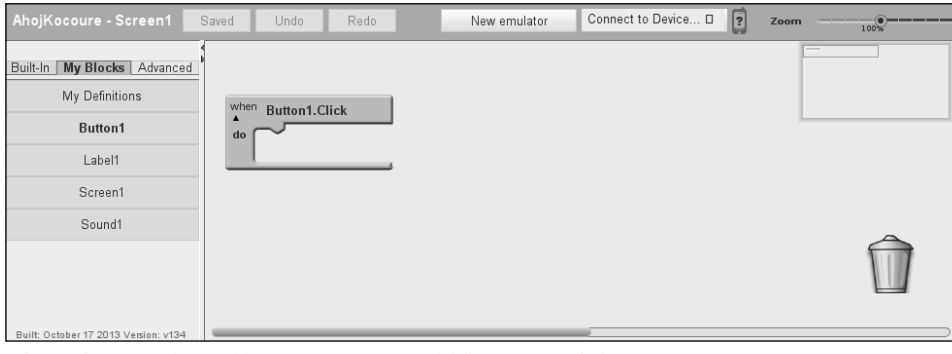
V levé horní části okna uvidíte tlačítka s nápisy „Built-In" (předdefinované bloky) a „My Blocks" (mé bloky). Klepněte na My Blocks, zobrazí se vám sloupec s přihrádkou pro každou komponentu, kterou jsme v Designéru vytvořili: `Button1`, `Label1`, `Screen1` i `Sound1`. Když na přihrádku klepnete, budete mít u každé z komponent na výběr z několika možností (bloků). (Prozatím se nezatěžujte sloupcem Built-In, k tomu se dostaneme v kapitole 2.) Klepněte na

přihrádku komponenty `Button1`. Přihrádka se otevře a nabídne vám několik bloků, jejichž prostřednictvím můžete tlačítku říct, co má dělat. První možnost je **Button1.Click**, viz obrázek 1.7.



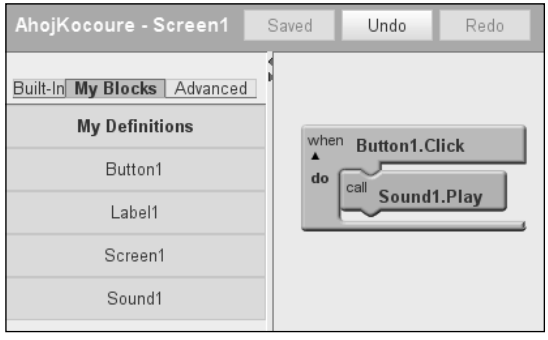
Obrázek 1.7 Klepnutím na tlačítko Button1 zobrazíte příslušné bloky

Klepněte na blok nazvaný **Button1.Click** a přetáhněte ho do pracovního prostoru. Když se na tento blok podíváte, všimnete si, že „when" (kdy) je na něm napsáno menším písmem než **Button1.Click**. Bloky s heslem „when" nazýváme obslužnými rutinami událostí. Říkají, co má komponenta udělat, když dojde k dané události. V tomto případě nás zajímá uživatelovo klepnutí na kocoura (což je ve skutečnosti tlačítko), viz obrázek 1.8. Následně do programu přidáme několik bloků, které programu řeknou, co má v reakci na událost udělat.



Obrázek 1.8 Reakci na klepnutí nastavíme v bloku Button.Click

Klepnutím na Sound1 v oddíle vlastních bloků (My Blocks) otevřete přihrádku zvukové komponenty. Přetáhněte blok **call Sound1.Play**. (Vzpomeňte si, jak jsme nastavovali vlastnost zvuku Sound1 na zvuk mňoukání, který jste si stáhli do počítače.) Nyní si můžete všimnout, že blok **call Sound1.Play** se tvarem hodí do mezery označené v bloku **Button1.Click** jako „do" (udělat). App Inventor je nastaven tak, abyste mohli spojovat jen určité typy bloků. V našem případě vyvolávají bloky s heslem „call" (volat) u komponent nějakou akci. Jak vidíte na obrázku 1.9, oba bloky by se spolu měly spojit. Při spojení uslyšíte krátký zvuk.



Obrázek 1.9 Když nyní někdo klepne na tlačítko, přehraje se zvuk mňoukání

Bloky v App Inventoru vypisují, na rozdíl od tradičních programovacích jazyků (které často vypadají jako směska náhodně poskládaných „slov"), definované chování slovy. V našem případě v podstatě říkáme: „Hele, App Inventore, když někdo klepne na tlačítko s kocourem, přehraj mňoukání."

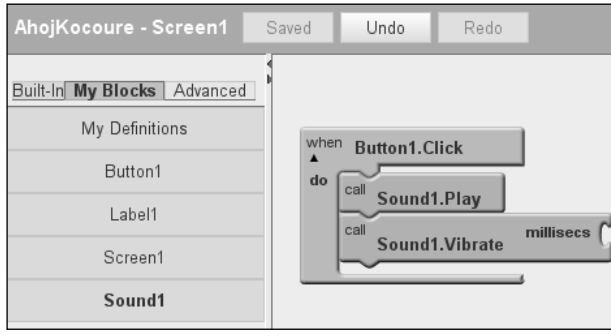
Otestujte svou aplikaci: Vyzkoušejme si, zda vše funguje tak, jak má – vždy, když do aplikace něco přidáte, měli byste to otestovat. Klepněte na tlačítko v telefonu (anebo v emulátoru). Měli byste slyšet zamňoukání. Výborně, vaše první aplikace funguje!

Přidání předení

Nyní nastavíme, aby kocour při klepnutí na tlačítko nejen zamňoukal, ale i zapředl. Předení nahradíme vibracemi telefonu. Možná to zní komplikovaně, ale ve skutečnosti je to snadné, protože komponenta Sound, kterou jsme použili k přehrání zamňoukání, umí telefon nechat také zavibrovat. App Inventor vám tento druh funkcí telefonu zpřístupní, aniž byste museli vědět, jak telefon vibruje. V Designéru nemusíte dělat nic jinak. Stačí tlačítku v Editoru bloků (Blocks Editor) přidat při stisknutí druhý typ chování:

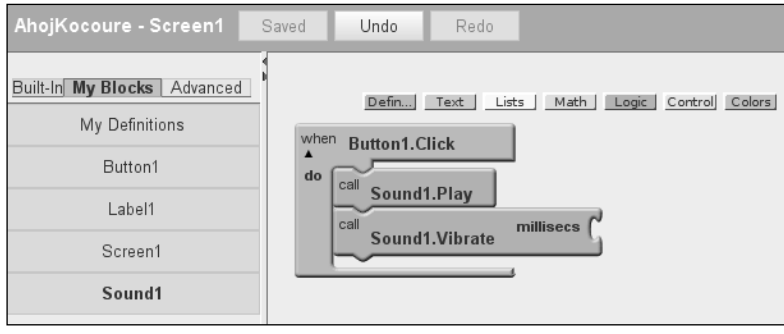
1. Přejděte do Editoru bloků (Blocks Editor) a v oddíle vlastních bloků (My Blocks) klepněte na zvuk Sound1, otevře se jeho přihrádka.
2. Vyberte blok **call Sound1.Vibrate** a přesuňte ho do přihrádky **Button1.Click** pod blok **call Sound1.Play**. Blok by se měl zapojit stejně jako na obrázku 1.10. Pokud se tak nesta-

ne, zkuste ho posunout tak, aby drobný zobáček na spodní části bloku **call Sound1.Play** zapadl do důlku v horní části bloku **call Sound1.Vibrate**.



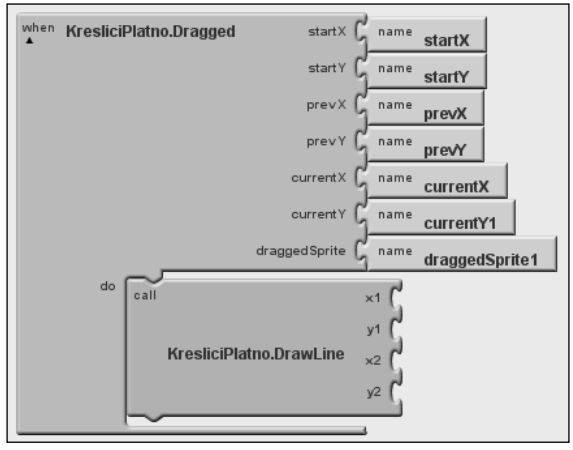
Obrázek 1.10 Přehrání zvuku a zavibrování při události Click

3. Zřejmě jste si všimli, že se v pravé horní části bloku **call Sound1.Vibrate** nachází text „millisecs“ (milisekundy). Volná přihrádka v bloku značí, že do ní můžete něco zapojit a definovat tak blíže, jak by se aplikace měla chovat. V tomto případě je třeba bloku **Vibrate** říct, jak dlouho má vibrovat. Tuto dobu zadáme v tisícinách sekundy (milisekundách), což je v mnohých programovacích jazycích běžné. Aby telefon vibroval půl sekundy, zadáme hodnotu 500. Uděláme to s pomocí bloku čísla. Klepněte v Designéru do prázdného prostoru a následně klepněte v zobrazené nabídce na zelené tlačítko Math (matematika), viz obrázek 1.11. Měla by se vám rozbalit nabídka, v níž je jako první položka 123. Ta představuje číselný blok.



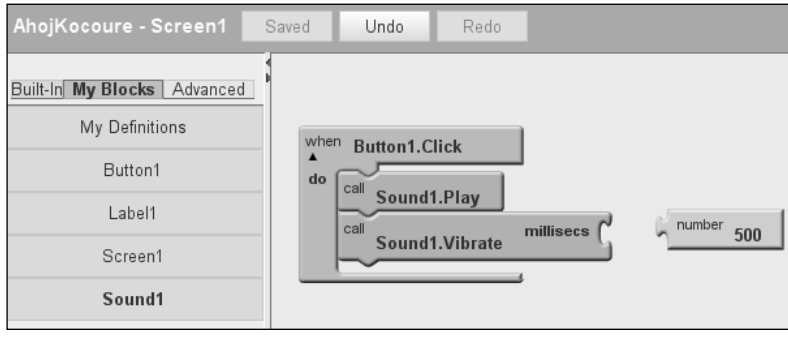
Obrázek 1.11 Otevření přihrádky Math

4. Klepněte na 123 zcela nahoře v seznamu. Měl by se vám ukázat blok s číslem 123, stejně jako na obrázku 1.12.



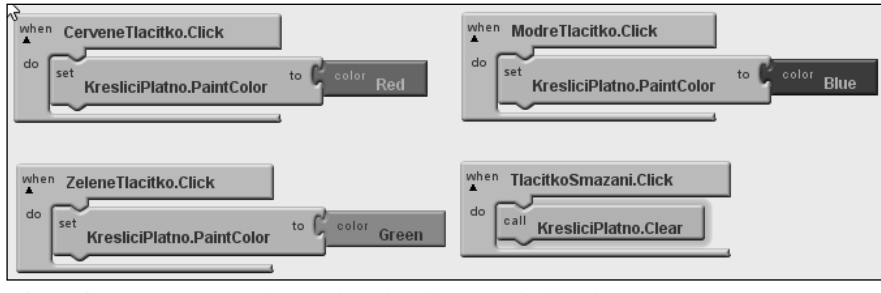
Obrázek 1.12 Výběr číselného bloku (123 je výchozí hodnota)

5. Klepnutím na 123 změňte hodnotu na 500, viz obrázek 1.13.



Obrázek 1.13 Změna hodnoty na 500

6. Zapojte číslo 500 do zásuvky napravo bloku **call Sound1.Vibrate**, viz obrázek 1.14.



Obrázek 1.14 Zapojení čísla 500 do přihrádky pro milisekundy.

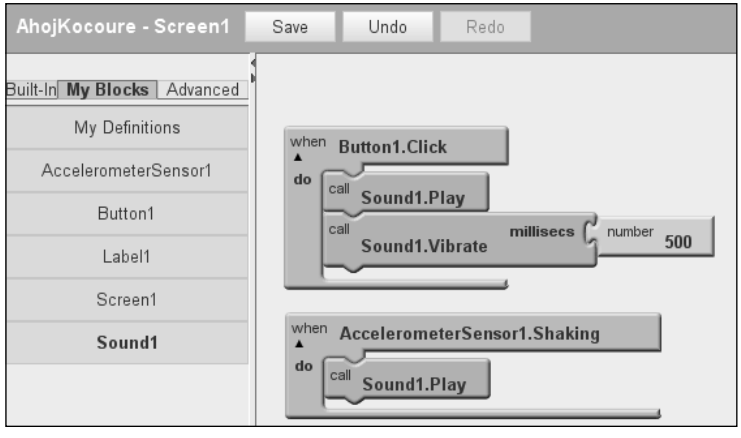
Otestujte svou aplikaci: Otestujte svou aplikaci. Vyzkoušejte si ji! Klepněte v telefonu na tlačítko, kocour bude půl vteřiny příst.

Zatřesení telefonem

Nyní přidáme poslední prvek, který využije další skvělou funkci telefonů se systémem Android: Necháme kocoura zamňoukat, když zatřepete telefonem. K tomu budeme potřebovat komponentu `AccelerometerSensor`, která rozpozná, když telefonem zatřepete.

1. V Designéru rozbalte v paletě oddíl Sensors (senzory) a přetáhněte z ní komponentu `AccelerometerSensor`. Nezáleží na tom, kam ji přesunete – stejně jako v případě všech ostatních neviditelných komponent, ať ji v prohlížeči přesunete kamkoliv, přesune se v prohlížeči do oddílu neviditelných komponent (Non-visible components).
2. K zatřepání telefonu je lepší se chovat jako k akci, která není závislá na stisknutí tlačítka. Proto budeme muset vytvořit novou obslužnou rutinu události. Přejděte do Editoru bloků (Blocks Editor). Zde byste měli mezi vlastními bloky (My Blocks) najít novou přihrádku `AccelerometerSensor1`. Otevřete ji a přetáhněte blok **AccelerometerSensor1.Shaking** – měl by to být druhý blok na seznamu.
3. Stejně jako v případě zvuku a klepnutí tlačítka přetáhněte blok **call Sound1.Play** a vložte ho do zdířky v bloku **AccelerometerSensor1.Shaking**. Vyzkoušejte zatřást telefonem.

Na obrázku 1.15 vidíte bloky hotové aplikace AhojKocoure.



Obrázek 1.15 Bloky aplikace AhojKocoure

Sbalení aplikace ke stažení

App Inventor je nástrojem provozovaným v cloudu, což znamená, že se samotná aplikace nachází na serverech Googlu. Když App Inventor zavřete, aplikace na vás bude čekat na serveru. Na počítači nemusíte nic ukládat, na rozdíl od běžné práce se soubory Word či hudebními stopami. Díky tomu můžete také aplikaci snadno testovat na připojeném telefonu (proto tomuto testování říkáme živé testování), aniž byste museli do telefonu cokoliv stahovat. Jediný

problém nastane v okamžiku, kdy telefon odpojíte od App Inventoru. Aplikace spuštěná na telefonu se vypne a nikde nenajdete její ikonu, protože ve skutečnosti není nainstalovaná.

Z dokončené aplikace můžete vytvořit balíček a nainstalovat ho do telefonu, aby aplikace fungovala i tehdy, když telefon odpojíte od počítače. Nejprve zkontrolujte, zda máte v telefonu povoleno stahovat aplikace i z jiných umístění než z Google Play. Obvykle lze toto nastavení změnit v Nastavení → Zabezpečení zaškrtnutím pole Neznámé zdroje. Poté se můžete v App Inventoru vrátit do Designéra, klepnout na „Package for Phone" (balíček pro telefon) a poté na „Download to Connected Phone" (stáhnout na připojený telefon). Měla by se vám zobrazit zpráva „Saving" (ukládám), následně „Packaging" (vytvářím balíček). Celý proces může trvat až minutu. Jakmile se zobrazí zpráva „Packaging", počkejte dalších 10–15 sekund, dokud se připravená aplikace nestáhne do telefonu. Jakmile se operace dokončí, zobrazí se vám potvrzení.

Jakmile aplikaci stáhnete, podívejte se mezi aplikace v telefonu. Uvidíte zde AhojKocoure, aplikaci, kterou jsme si právě vytvořili. Spustíte ji stejně jako jakoukoliv jinou aplikaci. (Spusťte novou aplikaci, nikoliv aplikaci App Inventor Phone.) Nyní můžete telefon odpojit, či dokonce restartovat, balíček nové aplikace z něj nezmizí.

Nezapomeňte, že zabalená aplikace již není součástí projektu v App Inventoru. V App Inventoru můžete na projektu pokračovat, stačí telefon zapojit pomocí USB kabelu do počítače jako dříve. Nainstalovaný balíček v telefonu se tak však nezmění. Pokud aplikaci změníte v App Inventoru, budete muset vytvořit balíček znovu a stáhnout si do telefonu jeho novou verzi.

Stáhněte si nyní zabalenou aplikaci AhojKocoure do svého telefonu. Jakmile skončíte, můžete se o aplikaci podělit i s rodinou a kamarády!

Sdílení aplikace

Aplikaci můžete sdílet dvojím způsobem. Chcete-li se podělit o spustitelnou aplikaci, klepněte na „Package for Phone" (balíček pro telefon) a následně „Download to this Computer" (stáhnout do počítače). V počítači tak vytvoříte soubor s příponou .apk. Pokud soubor chcete zpřístupnit na webu, budete ho muset do umístění nahrát. Jakmile se jednou aplikace objeví na webu, budou si ji moci na své počítače nainstalovat další lidé. Stačí, když si ji stáhnou ve webovém prohlížeči. Sdělte jim však, že si budou muset v telefonu povolit instalaci z neznámých zdrojů, aby si mohli aplikaci nainstalovat.

Rovněž se můžete podělit o zdrojový kód (bloky) aplikace s dalšími vývojáři, kteří pracují v App Inventoru. Klepněte na My Projects (mé projekty), označte aplikaci, kterou chcete sdílet (v tomto případě AhojKocoure), a vyberte More Actions (další akce) → Download Source (stáhnout zdrojový kód). Soubor, který se vám v počítači vytvoří, bude mít příponu .zip. Tento soubor pak můžete komukoliv zaslat e-mailem. Příjemce pak bude moci vaši aplikaci upravovat a přizpůsobit si ji, aniž by ovlivnil vaši verzi.

Sdílení aplikací bude brzy snazší a zábavnější – právě probíhají práce na komunitní stránce určené ke sdílení.

Variace

Vytvořili jste plnohodnotnou aplikaci a měli jste příležitost si s ní pohrát (a možná jste si ji i stáhli a podělili se o ni s dalšími), takže jste si již mohli všimnout několika věcí. Projděte si níže uvedené body a zvažte, jak byste je v aplikaci řešili. Jak zřejmě brzy zjistíte, často u hotové aplikace ještě přijdete na možná vylepšení a změny, k aplikaci se vrátíte a vylepšení zapracujete. Nijak se toho nebojte, je to tak správně – znamená to, že jste na dobré cestě stát se schopným vývojářem aplikací!

- Mňoukání bude při zatřepání telefonem znít zvláštně, jakoby z ozvěny. To proto, že senzor spouští událost třepání mnohokrát za sekundu, a tak se jednotlivá zamňoukání překrývají. Když se v Designéru zaměříte na komponentu `Sound`, uvidíte vlastnost `Minimum interval`. Tato vlastnost určuje, jak rychle po sobě mohou jednotlivé zvuky začínat. Momentálně je vlastnost nastavena na půl sekundy (500 milisekund), přičemž jedno zamňoukání trvá déle. Když si s tímto intervalem pohrajete, upravíte překrývání jednotlivých zamňoukání.
- Pokud aplikaci spustíte a s telefonem v kapse se trochu projdete, při každém prudším pohybu telefon zamňouká – a to by se vám možná nemuselo líbit. Aplikace jsou v systému Android zpravidla navrženy tak, aby zůstaly spuštěny, i když je nikdo nesleduje. Vaše aplikace bude nadále komunikovat s akcelerometrem a mňoukat. Pokud chcete aplikaci skutečně ukončit, přesuňte ji do popředí a stiskněte tlačítko nabídky. To vám umožní aplikaci ukončit.

Shrnutí

V této kapitole jsme se mimo jiné naučili následujícímu:

- Aplikace tvoříme výběrem komponent v Designéru. Komponentám říkáme, co a kdy mají provádět v Editoru bloků (Blocks Editor).
- Některé komponenty jsou viditelné, zatímco některé nikoliv. Viditelné komponenty se zobrazí v uživatelském rozhraní aplikace. Neviditelné komponenty například přehrávají zvuky.
- Chování komponent definujeme pomocí bloků v Editoru bloků (Blocks Editor). Nejprve přetáhneme obslužnou rutinu události, například **Button1.Click**, a poté do ní vložíme bloky s příkazy, například **Sound.Play**. Když uživatel stiskne tlačítko, program provede všechny bloky v rutině **Button1.Click**.
- Některé příkazy potřebují další parametry, bez kterých nebudou fungovat. Příkladem je příkaz `Vibrate`, který potřebuje vědět, jak dlouho má vibrovat. Těmto hodnotám říkáme *argumenty*.
- Čísla jsou zastoupena číselnými bloky. Ty můžete zapojovat do příkazů, které přijímají čísla coby argumenty.
- App Inventor nabízí komponenty senzorů. `AccelerometerSensor` dokáže registrovat pohyby telefonu.
- Z vytvořených aplikací si můžete vytvářet balíčky, které lze stáhnout do telefonu a spouštět nezávisle na App Inventoru.

ČÁST

12 APLIKACÍ NA MÍRU

V této části:

- Kapitola 2 – Kreslení
- Kapitola 3 – Krtek
- Kapitola 4 – Auto SMS
- Kapitola 5 – Beruška
- Kapitola 6 – Průvodce Paříží
- Kapitola 7 – Androide, kde mám auto?
- Kapitola 8 – Prezidentský kvíz
- Kapitola 9 – Xylofon
- Kapitola 10 – Tvorba a vyplňování kvízů
- Kapitola 11 – Centrála
- Kapitola 12 – Dálkový ovladač robota
- Kapitola 13 – Amazon v knihkupectví

V tomto oddíle najdete podrobné pokyny k vytvoření 12 aplikací pro systém Android. Ačkoliv se jednotlivé aplikace svou povahou značně liší, koncepčně na sobě postupně staví, přičemž poslední kapitoly jsou z hlediska potřebných vědomostí nejsložitější.

Na konci každé kapitoly najdete návrhy na úpravy a rozšíření daných aplikací. Vycházíme z toho, že nejlépe se naučíte programovat, když budete střídavě pracovat podle pokynů a sami zkoušet upravovat aplikace. Rovněž budete odkazováni na související kapitoly manuálové části této knihy, kde najdete podrobný popis podaných konceptů.

KAPITOLA **2**

Kreslení

V této kapitole:

- Co se naučíte
- Začínáme
- Design komponent
- Definování chování komponent
- Hotová aplikace: Kreslení
- Variace
- Shrnutí

V tomto návodu se seznámíme s komponentou `Canvas`*, která slouží k vytváření jednoduché dvourozměrné (2D) grafiky. Vytvoříme si aplikaci Kreslení, která uživatelům umožní kreslit po obrazovce různými barvami, a následně ji vylepšíme tak, aby se uživatelé mohli vyfotit a kreslit si po vlastním obličeji. Z historického hlediska se jedná o jednu z prvních aplikací, která v sedmdesátých letech ukazovala potenciál osobních počítačů. V té době byla i takováto aplikace velmi složitá a výsledky nebyly příliš reprezentativní. Nyní však může v App Inventoru poskládat jednoduchý kreslicí nástroj, který je dobrým odrazovým můstkem pro 2D hry, kdokoliv.*

V aplikaci Kreslení, kterou vidíte na obrázku 2.1, můžete:

- Namočit prst do virtuální plechovky s barvou.
- Tahem po displeji nakreslit čáru.
- Klepáním po displeji kreslit tečky.
- Pomocí tlačítka v dolní části displeje obrazovku smazat.
- Změnit velikost tečky pomocí tlačítek ve spodní části displeje.
- Pořídit telefonem snímek a kreslit po něm.

Obrázek 2.1 Aplikace Kreslení

Co se naučíte

Návod v této kapitole vás uvede do následující problematiky:

- Využití komponenty `Canvas` při kreslení.
- Práce s dotyky a tahy na displeji telefonu.
- Řízení rozložení obrazovky s aranžovacími komponentami.
- Práce s obslužnými rutinami událostí, které přijímají argumenty.
- Definování proměnných, které si budou pamatovat například velikost tečky, již si uživatel zvolí.

Začínáme

Ujistěte se, že máte počítač i telefon připraven na práci v App Inventoru, a přejděte na adresu *http://appinventor.googlelabs.com*. V okně Designéra komponent (Component Designer) vytvořte nový projekt a nazvěte ho „Kresleni". Otevřete Editor bloků (Blocks Editor) a v telefonu spusťte aplikaci App Inventor.

Začneme v panelu vlastností (Properties) napravo od Designéra tím, že změníme titulek obrazovky na „Kreslení" (a zbavíme se tak názvu `Screen1`). Změna by se měla projevit v telefonu změnou titulku aplikace.

Nemusíte se bát, že byste si pletli název projektu a obrazovky. V App Inventoru existují tři klíčové názvy:

- Název, který si zvolíte pro projekt, na němž pracujete. Tento název se současně použije jako název aplikace, kterou si budete stahovat do telefonu. Pokud chcete vytvořit novou verzi aplikace či ji přejmenovat, klepněte v Designéru na Save As (uložit jako).
- Název komponenty `Screen1`, který se zobrazí v panelu seznamu komponent aplikace. Tento název změnit v současné verzi App Inventoru nelze.
- Titulek obrazovky, který se zobrazí v záhlaví telefonu. Ten má na počátku hodnotu Screen1, viz aplikace AhojKocoure. Je však možné ho změnit, stejně jako v případě aplikace Kreslení.

Design komponent

Aplikaci budeme vytvářet pomocí následujících komponent:

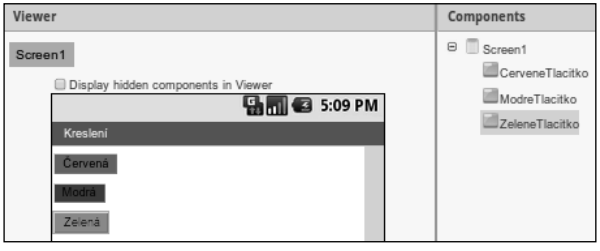
- Tři komponenty `Button` pro výběr červené, modré a zelené barvy a jedna komponenta `HorizontalArrangement`, která tlačítka seskupí.
- Jedna komponenta `Button`, která bude sloužit ke smazání nakresleného obrázku, a dvě pro změnu velikosti kreslených teček.
- Komponenta `Canvas`, která představuje kreslicí plochu. Komponenta `Canvas` má vlastnost `BackgroundImage`, jíž nastavíme soubor *kitty.png*, který znáte z aplikace AhojKocoure, po-

psané v první kapitole. Dále v kapitole upravíme aplikaci tak, aby uživatel mohl nahradit pozadí pořízeným snímkem.

Tlačítka barev

Nejprve vytvoříme tři tlačítka pro barvy:

1. Přetáhněte komponentu `Button` do prohlížeče, upravte jí atribut `Text` na „Červená“ a vlastnosti `BackgroundColor` nastavte červenou barvu.
2. Klepnutím v seznamu prohlížeče označte komponentu `Button1` (možná je již označena) a klepnutím na Rename (přejmenovat) změňte její název z `Button1` na `CerveneTlacitko`. Všimněte si, že v názvech komponent nelze používat mezery, takže běžně píšeme první písmena slov velkými písmeny.
3. Podobně vytvořte dvě další tlačítka pro modrou a zelenou barvu, a to s názvy `ModreTlacitko` a `ZeleneTlacitko`, a umístěte je vertikálně pod červené tlačítko. Porovnejte výsledek s obrázkem 2.2.



Obrázek 2.2 Prohlížeč zobrazuje tři vytvořená tlačítka

Všimněte si, že v tomto projektu upravujeme názvy komponent, nenecháváme jim výchozí názvy jako v aplikaci AhojKocoure. Popisné názvy vám zpřehlední projekty. Navíc vám skutečně pomohou, jakmile se přesunete do Editoru bloků (Blocks Editor), ve kterém budete muset komponenty hledat podle názvu. V této knize budeme v názvech komponent uchovávat jejich typy (například `CerveneTlacitko`).

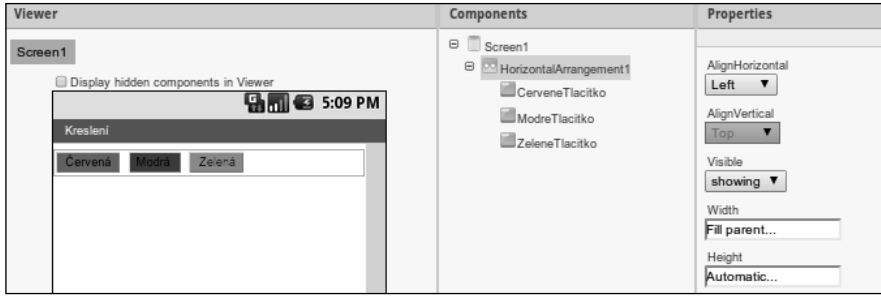
Otestujte svou aplikaci: Pokud jste doposud neklepli na „Connect to Device“ (připojit k zařízení), učiňte tak nyní a zkontrolujte si, jak aplikace vypadá na telefonu (pokud je zapojen do počítače) anebo v emulátoru.

Uskupení komponent

Nyní byste měli mít nad sebou tři tlačítka. V naší aplikaci však budeme chtít, aby se tlačítka vyrovnala v horní části obrazovky, viz obrázek 2.3. K tomu nám dopomůže komponenta `HorizontalArrangement`:

1. V kategorii Screen Arrangement (rozložení obrazovky) přetáhněte z palety komponentu `HorizontalArrangement` a umístěte ji pod tlačítka.

2. V panelu vlastností (Properties) změňte vlastnost `Width` komponenty `Horizontal Arrangement` na „Fill parent“ (vyplnit nadřazenou komponentu), komponenta tak vyplní celou šíři obrazovky.
3. Všechna tři tlačítka postupně přesuňte do komponenty `HorizontalArrangement`. **Nápověda:** modrá svislá čára vám ukáže umístění přetahované komponenty.



Obrázek 2.3 Tři horizontálně zarovnaná tlačítka

Když se zaměříte na seznam komponent použitých v projektu, všimnete si tří tlačítek, která jsou pod komponentou `HorizontalArrangement` zarovnána tak, aby bylo zřejmé, že se jedná o podřazené komponenty. Všimněte si, že jsou všechny komponenty zarovnány pod komponentou `Screen1`.

Otestujte svou aplikaci: Všechna tři tlačítka byste měli vidět zarovnaná v řadě také na obrazovce telefonu, i když nemusí vypadat tak jako v Designéru. Například v prohlížeči uvidíte ohraničení kolem komponenty `HorizontalArrangement`, ovšem to se v telefonu nezobrazí.

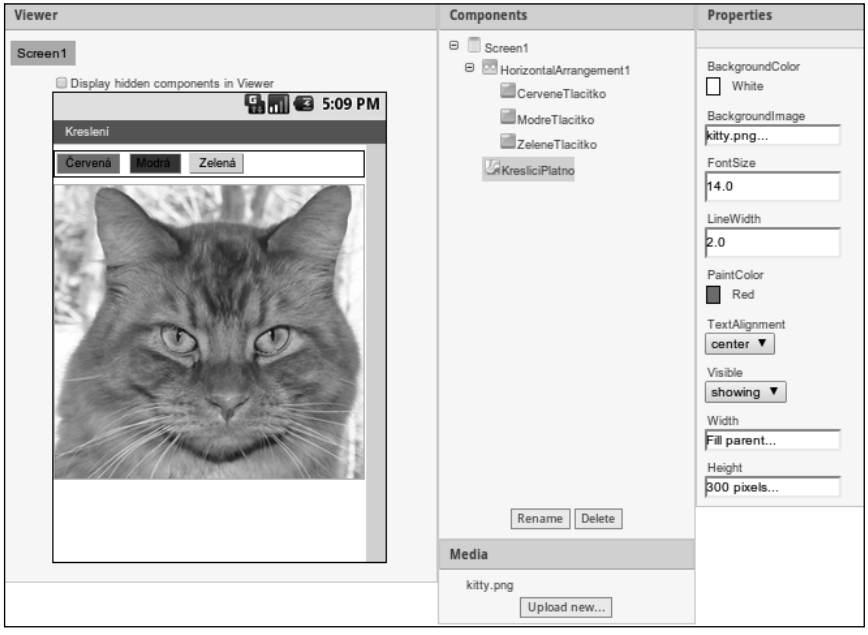
Zjednodušeně se dá říci, že na obrazovce můžeme pomocí podobných seskupení jednoduše rozkládat komponenty horizontálně, vertikálně, ale i do tabulek. Když navíc jednotlivá seskupení vložíme do sebe (zahnízdíme je), budeme moci vytvářet i složitější seskupení.

Přidání kreslicího plátna

Uživatel bude kreslit na kreslicí plátno. Přidejte ho do projektu a jeho vlastnosti `Background Image` nastavte soubor *kitty.png* z aplikace AhojKocoure:

1. Z kategorie Basic (základní) palety přetáhněte do prohlížeče komponentu `Canvas` (plátno). Její název změňte na `KresliciPlatno`. Vlastnost `Width` nastavte plátnu na „Fill parent“ (vyplnit nadřazenou komponentu). Vlastnost `Height` nastavte na 300 pixelů.
2. Pokud jste si vytvořili aplikaci AhojKocoure (kapitola 1), soubor *kitty.png* máte již stažen. Jestli ne, můžete si ho stáhnout z adresy *http://examples.oreilly.com/0636920016632/*.
3. Vlastnosti `BackgroundImage` plátna `Canvas` nastavte soubor *kitty.png*. V editoru vlastností bude mít vlastnost `BackgroundImage` nastavenu hodnotu None. Klepněte do tohoto pole a nahrajte soubor *kitty.png*.

4. Vlastnost `PaintColor` plátna `Canvas` nastavte na červenou, aby uživatel, který aplikaci spustí, ale ještě si nezvolí žádné z barevných tlačítek, mohl přímo kreslit červenou barvou. Zkontrolujte si, zda vaše aplikace vypadá jako na obrázku 2.4.

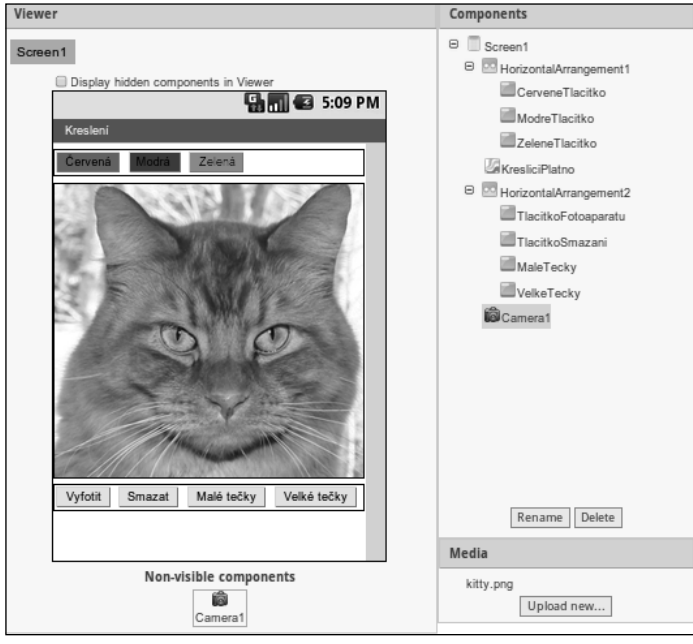


Obrázek 2.4 Komponenta Canvas má vlastnost BackgroundImage nastavenu na obrázek kocoura

Uspořádání spodních tlačítek a přidání snímku

1. Z palety přetáhněte druhou komponentu `HorizontalArrangement` a umístěte ji pod kreslicí plátno. Poté na obrazovku přetáhněte další dvě komponenty `Button` a umístěte je pod spodní komponentu `HorizontalArrangement`. Název prvního tlačítka změňte na `TlacitkoFotoaparatu` a jeho vlastnost `Text` změňte na „Vyfotit“. Název druhého tlačítka změňte na `TlacitkoSmazani` a jeho vlastnost `Text` změňte na „Smazat“.
2. Do komponenty `HorizontalArrangement` přetáhněte z palety další dvě tlačítka `Button` a umístěte je vedle komponenty `TlacitkoSmazani`.
3. Pojmenujte nová tlačítka `VelkeTecky` a `MaleTecky` a jejich vlastnosti `Text` nastavte na „Velké tečky“ a „Malé tečky“.
4. Z palety Media přetáhněte do prohlížeče komponentu `Camera`. Komponenta se zobrazí v oblasti určené neviditelným komponentám.

Nyní máte nastaven vzhled aplikace tak, jak vidíte na obrázku 2.5.



Obrázek 2.5 Kompletní uživatelské rozhraní aplikace Kreslení

Otestujte svou aplikaci: Vyzkoušejte si aplikaci v telefonu. Zobrazí se pod horní lištou tlačítek obrázek kocoura? Zobrazí se dolní řada tlačítek?

Definování chování komponent

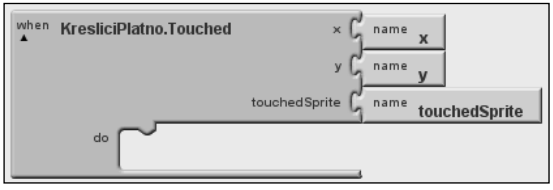
Dalším krokem je definování chování komponent. Možná vám přijde tvorba kreslicího programu složitá, ale věřte, že vám App Inventor v mnohém práci značně ulehčí. Uživatelovy dotyky a tahy po displeji stejně jako kreslení a pořizování snímků můžete řídit pomocí připravených bloků.

V Designéru jste přidali komponentu `Canvas` s názvem `KresliciPlatno`. Ta nabízí události `Touched` (dotyk) a `Dragged` (přetažení). Událost **KresliciPlatno.Touched** naprogramujeme tak, aby volala metodu **KresliciPlatno.DrawCircle**. Událost **KresliciPlatno.Dragged** bude volat metodu **KresliciPlatno.DrawLine**. Tlačítka pak naprogramujeme k tomu, aby nastavila vlastnost `KresliciPlatno.PaintColor`, smazala `KresliciPlatno` a změnila vlastnost `BackgroundImage` na snímek pořízený fotoaparátem telefonu.

Přidání události pro vykreslení tečky při dotyku

Nejprve zařídíme, aby se při dotyku plátna `KresliciPlatno` vykreslila tečka:

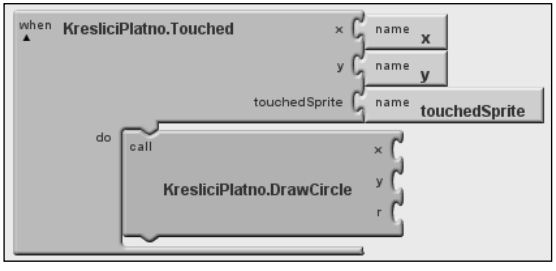
1. V Editoru bloků (Blocks Editor) klepněte na My Blocks (mé bloky), vyberte zásuvku `KresliciPlatno` a na pracovní plochu z něj přetáhněte blok **KresliciPlatno.Touched**. Jakmile blok přetáhnete, tři zásuvky napravo se automaticky vyplní názvy bloků `x`, `y` a `touchedSprite`, viz obrázek 2.6.



Obrázek 2.6 Událost obsahuje údaje o tom, kde na displeji došlo k dotyku

Poznámka: Pokud jste si v první kapitole vytvořili aplikaci AhojKocoure, již vám jsou známy události **Button.Click**, nikoliv však události komponenty `Canvas`. Události **Button.Click** jsou velmi jednoduché, protože o nich nemusíte vědět nic víc, než že k nim dojde. Některé obslužné rutiny událostí však obsahují i informace o samotné události, jež nazýváme argumenty. Událost **KresliciPlatno.Touched** vám sdělí souřadnice x a y dotyku na plátně. Rovněž vám sdělí, zda došlo k dotyku objektu komponenty `KresliciPlatno` (v App Inventoru je nazýváme sprity), ale to až do kapitoly 3 nebudeme potřebovat. Souřadnice x a y budeme jako argumenty využívat ke zjišťování polohy, ve které se uživatel dotkl displeje, abychom na daném místě mohli vykreslit tečku.

2. Z přihrádky KresliciPlatno přetáhněte do obslužné rutiny události **KresliciPlatno.Touched** příkaz **KresliciPlatno.DrawCircle**, viz obrázek 2.7.



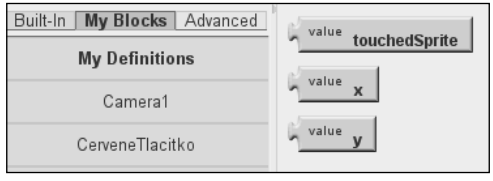
Obrázek 2.7 Jakmile se uživatel dotkne displeje, aplikace vykreslí tečku

Napravo od bloku **KresliciPlatno.DrawCircle** uvidíte tři přihrádky pro argumenty, jež je třeba vyplnit: `x`, `y` a `r`. Argumenty `x` a `y` určují místo, ve kterém se má vykreslit tečka, zatímco argument `r` určuje její poloměr (neboli velikost). Žlutě zbarvené upozornění v horní části obslužné rutiny události **KresliciPlatno.Touched** říká, že přihrádky ještě nejsou zaplněny. Vytvoříme proto bloky, které do přihrádek budeme moci zasunout.

Tato obslužná rutina události na vás může působit matoucím dojmem, protože přihrádky pro x a y má i událost **KresliciPlatno.Touched**. Pamatujte si ale, že argumenty x a y události **KresliciPlatno.Touched** ukazují, kde se uživatel obrazovky dotkl, zatímco v případě události **KresliciPlatno.DrawCircle** mají za úkol říci, kde se má vykreslit tečka.

Tečku chceme vykreslit v místě, ve kterém se uživatel dotkne displeje, a proto argumenty x a y události **KresliciPlatno.Touched** použijeme jako argumenty události **KresliciPlatno.DrawCircle**.

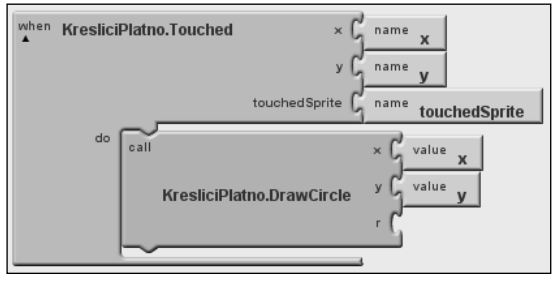
Poznámka: Argumenty nepřesouvejte z události `Touched` přímo, i když by vám to možná připadalo logické! Skutečnost, že je možné argumenty uchopit a přetáhnout, je jen nešťastným designovým aspektem App Inventoru. Namísto toho přetáhněte dané hodnoty z přihrádky My Definitions (mé definice), viz obrázek 2.8.



Obrázek 2.8 Systém vloží odkazy na argumenty události touchedSprite, x a y

3. Mezi My Blocks (mé bloky) otevřete přihrádku My Definitions (mé definice) a vyhledejte bloky **value x** a **value y**.

App Inventor tyto bloky automaticky vytvoří za vás, jakmile přetáhnete blok obslužné rutiny události **KresliciPlatno.Touched**: tedy odkazy na argumenty x a y dané události. Přetáhněte bloky **value x** a **value y** a zapojte je do odpovídajících zdířek bloku **KresliciPlatno.DrawCircle** tak, aby výsledek vypadal jako na obrázku 2.9.

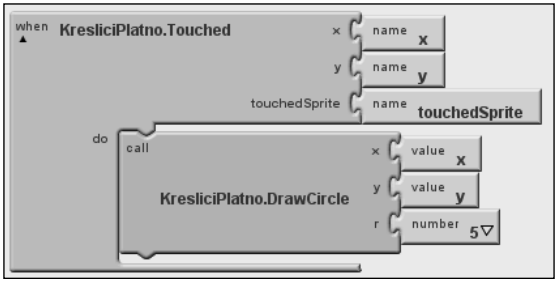


Obrázek 2.9 Aplikace ví, kam má vykreslovat (x, y), ale stále je ještě třeba definovat velikost tečky

4. Aby se mohla tečka vykreslit, budeme muset určit její poloměr `r`. Poloměr je udán v pixelech, což je nejmenší bod, který lze na obrazovce nakreslit. Prozatím nastavíme poloměr na 5: Klepněte do prázdné oblasti obrazovky, vyvoláte tak kontextovou nabídku. Vyberte složku Math. V rozbalovací nabídce vyberte položku 123, vytvoříte tak číselný blok. Hodnotu 123 upravte na 5 a blok zapojte do zdířky `r`. Jakmile tak učiníte, žluté pole

v levém horním rohu zmizí, protože již budou všechny zdířky zaplněny. Na obrázku 2.10 vidíte, jak by měla výsledná obslužná rutina události **KresliciPlatno.Touched** vypadat.

Poznámka: Všimněte si, že číselný blok je možné vytvořit pouhým zadáním čísla 5 v Editoru bloků (Blocks Editor) a stisknutím klávesy Enter. Můžeme takto filtrovat bloky: Jakmile začnete psát, Editor bloků zobrazí seznam bloků, jejichž názvy odpovídají zadávanému textu. Když zadáte číslo, vytvoří číselný blok.



Obrázek 2.10 Jakmile se uživatel dotkne kreslicího plátna, na souřadnicích x, y se vykreslí tečka o poloměru 5 pixelů

Otestujte svou aplikaci: Vyzkoušejte výsledek své dosavadní práce. Klepněte na kreslicí plochu – při doteku by měla na obrazovce zůstat tečka. Pokud jste vlastnost `KresliciPlatno.PaintColor` nastavili v Designéru komponent (Component Designer) na červenou, budou vykreslené tečky červené (v opačném případě budou černé, protože černá je výchozí barva).

Přidání události pro vykreslování křivek

Nyní přidáme obslužnou rutinu události potažení. Mezi dotykem a potažením je následující rozdíl:

- Při doteku se dotknete plátna, ale prst bez pohybu zvednete.
- Při potažení umístíte prst na plátno a posunete jím, přičemž ho necháte v kontaktu s displejem.

V kreslicím programu budete prstem při potažení kreslit tlustou křivku. Ve skutečnosti budete kreslit stovky tenkých úseček. Při každém pohybu prstu, byť jen o kousek, natáhnete čáru z poslední pozice prstu do nové pozice.

1. Z přihrádky KresliciPlatno přetáhněte na pracovní plochu blok **KresliciPlatno.Dragged**. Měli byste vidět stejnou obslužnou rutinu události jako na obrázku 2.11. Událost **KresliciPlatno.Touched** přijímá sedm argumentů:

   `startx,starty`

   Umístění prstu v místě, kde jste prstem začali táhnout.

   `currentx, currenty`

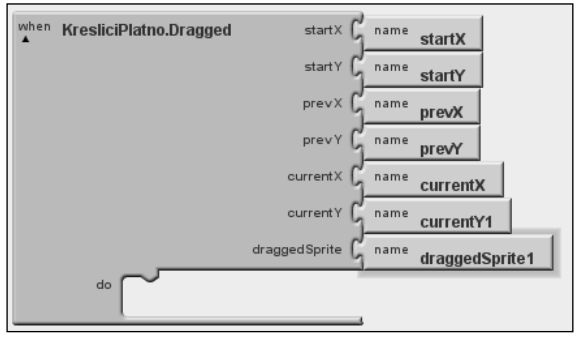
Aktuální pozice prstu.

`prevx,prevy`

Bezprostředně předchozí pozice prstu.

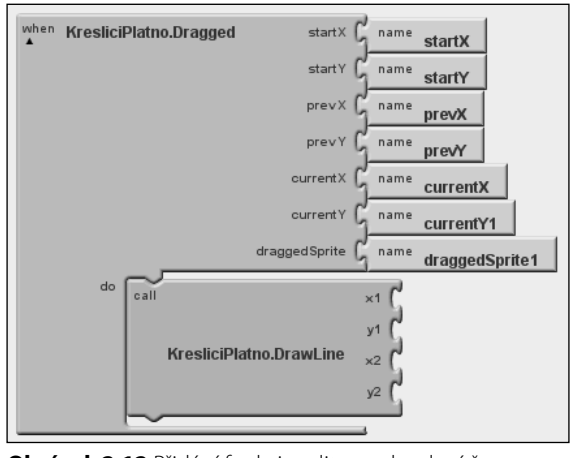
`draggedSprite`

Argument, který bude kladný, pokud uživatel potáhne prstem přímo na sprite obrázku. My v návodu tento argument nebudeme používat.



Obrázek 2.11 Událost Dragged má ještě více argumentů než událost Touched

2. Z přihrádky KresliciPlatno přetáhněte blok **KresliciPlatno.DrawLine** do bloku **KresliciPlatno.Dragged**, viz obrázek 2.12.

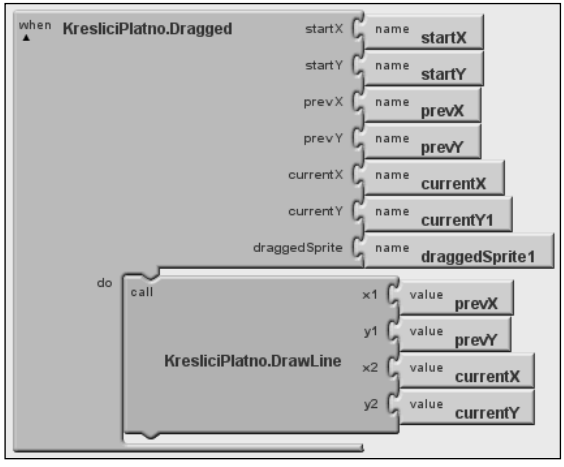


Obrázek 2.12 Přidání funkcionality pro kreslení čar

Blok **KresliciPlatno.Touched** má čtyři argumenty, dva pro každý bod definující úsečku: (`x1,y1`) zastupuje jeden bod, zatímco (`x2,y2`) druhý bod. Uhodnete, jaké hodnoty je třeba do těchto argumentů zapojit? Pamatujte si, že událost **Dragged** se zavolá tolikrát, kolikrát potáhnete prstem po kreslicím plátně: aplikace vykreslí drobnou linku vždy, když pohnete

prstem z bodu (`prevx`,`prevy`) do bodu (`currentX`,`currentY`). Tyto hodnoty tedy přidáme do bloku **KresliciPlatno.DrawLine**.

3. Klepněte na přihrádku My Definitions (mé definice). Měli byste vidět bloky potřebných argumentů. Odpovídající bloky s hodnotami přetáhněte do příslušných zdířek bloku `KresliciPlatno.Dragged`. Bloky **value prevX** a **value prevY** byste měli zapojit do přihrádek `x1` a `y1`. Bloky **value currentX** a **value currentY** byste měli zapojit do slotů `x2` a `y2`, stejně jako na obrázku 2.13.



Obrázek 2.13 Aplikace vykreslí při potažení úsečku z předchozího bodu do aktuálního bodu

Otestujte svou aplikaci: Vyzkoušejte si chování aplikace v telefonu. Potáhněte prstem po displeji a nakreslete několik čar. Dotykem vykreslete na obrazovce tečky.

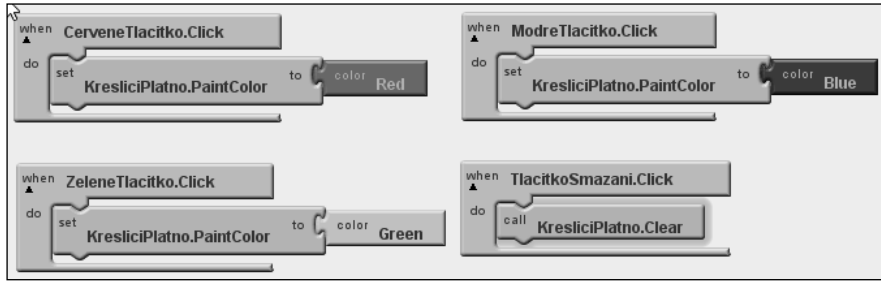
Přidání obslužných rutin událostí tlačítkům

Aplikace, kterou jste vytvořili, umožňuje uživateli kreslit, ale pouze červeně. V dalším kroku přidáme obslužné události rutin barevných tlačítek, které uživateli umožní změnit barvu, ale také další obslužnou rutinu události pro tlačítko `TlacitkoSmazani`, které umožní smazat obrazovku a začít kreslit znovu.

V Editoru bloků (Blocks Editor):

1. Přejděte do sloupce My Blocks (mé bloky).
2. Otevřete přihrádku `CerveneTlacitko` a přetáhněte blok **CerveneTlacitko.Click**.
3. Otevřete přihrádku KresliciPlatno. Přetáhněte blok **set KresliciPlatno.PaintColor to** (možná se budete muset v seznamu bloků přesunout níže, abyste blok našli) na oddílu „do“ (dělat) bloku **CerveneTlacitko.Click**.
4. Přejděte do sloupce Built-In (předdefinované bloky). Otevřete přihrádku Colors (barvy), přetáhněte z ní blok červené barvy a zapojte ho do bloku **KresliciPlatno.PaintColor**.

5. Opakujte kroky 2–4 s modrým a zeleným tlačítkem.
6. Poslední tlačítko, které je třeba nastavit, je `TlacitkoSmazani`. Přejděte zpět do sloupce My Blocks (mé bloky) a z přihrádky TlacitkoSmazani přetáhněte blok **TlacitkoSmazani.Click**. Z přihrádky KresliciPlatno přetáhněte blok **KresliciPlatno.Clear** a umístěte ho do bloku **TlacitkoSmazani.Click**. Ověřte si, zda vaše bloky vypadají jako na obrázku 2.14.



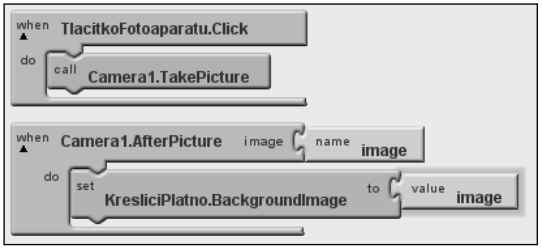
Obrázek 2.14 Klepnutím na tlačítka barev změníte vlastnost PaintColor kreslicího plátna, klepnutím na tlačítko smazání vyčistíte obrazovku

Uživatel si může pořídit snímek

Aplikace vytvořené v App Inventoru mohou využívat pokročilých funkcí zařízení Android, a to včetně fotoaparátu. Abychom aplikaci trochu vylepšili, umožníme uživateli nastavit do pozadí snímek pořízený fotoaparátem telefonu.

1. Komponenta `Camera` nabízí dva klíčové bloky. Blok **Camera.TakePicture** spustí na zařízení aplikaci fotoaparátu. Událost **Camera.AfterPicture** se vyvolá, jakmile uživatel pořídí snímek. Bloky obslužné rutiny události **Camera.AfterPicture** nastaví `KresliciPlatno.BackgroundImage` na právě pořízený snímek. Na pracovní plochu přetáhněte obslužnou rutinu události **TakePictureButton.Click**.
2. Z komponenty `Camera1` přetáhněte blok **Camera1.TakePicture** a umístěte ho do obslužné rutiny události **TakePictureButton.Click**.
3. Z komponenty `Camera1` přetáhněte na pracovní plochu obslužnou rutinu **Camera1.AfterPicture**.
4. Z komponenty `KresliciPlatno` přetáhněte blok **set KresliciPlatno.BackgroundImage to** a umístěte ho do obslužné rutiny **Camera1.AfterPicture**.
5. **Camera1.AfterPicture** má argument s názvem `image`, který zastupuje právě pořízený snímek. Můžete na něj odkázat pomocí položky **value image** v paletě My Definitions (mé definice). Přetáhněte blok a zapojte ho do bloku **KresliciPlatno.BackgroundImage**.

Bloky by měly vypadat jako na obrázku 2.15.



Obrázek 2.15 Pořízený snímek se nastaví jako pozadí kreslicího plátna

Otestujte svou aplikaci: Klepnutím na Pořídit snímek v telefonu aplikaci vyzkoušejte. Kocour by měl zmizet a naopak by se měl objevit pořízený snímek. Na ten můžete nyní kreslit. (Jak je vidět na obrázku 2.16, studenti rádi kreslí po obličeji profesora Wolbera.)

Obrázek 2.16 Aplikace Kreslení s „upraveným" obrázkem profesora Wolbera

Změna velikosti tečky

O velikosti teček vykreslovaných na plátně rozhoduje volání bloku **KresliciPlatno.DrawCircle**, ve kterém je poloměr `r` nastaven na 5. Chcete-li poloměr upravit, můžete argumentu `r` nastavit jinou hodnotu. Vyzkoušejte si to změnou poloměru z 5 na 10 a ověřte si, jak bude výsledek vypadat v telefonu.

Háčkem je zde skutečnost, že uživatel bude moci vykreslovat jen tak velké tečky, jak mu dovolíte argumentem poloměru. Co kdyby chtěl uživatel velikost změnit? Upravíme program tak, aby si velikost teček mohl určit nejen programátor, ale také uživatel. Aplikaci upravíme tak, aby se velikost tečky změnila na 8 při klepnutí na tlačítko „Velké tečky" a na 2 při klepnutí na tlačítko „Malé tečky".

Abychom mohli argumentu poloměru předat jinou hodnotu, musí ji aplikace znát. Musíme jí říci, že má použít konkrétní hodnotu, kterou si musí aplikace uložit (zapamatovat), aby ji mohla používat opakovaně. Když si aplikace potřebuje zapamatovat něco jiného než vlastnost, můžeme definovat *proměnné*. Proměnná je taková *paměťová buňka*. Můžeme ji chápat jako přihrádku, do které si uložíme data, jež se mohou měnit, kupříkladu velikost tečky (další informace o proměnných najdete v kapitole 15).

Začneme definicí proměnné `velikostTecky`:

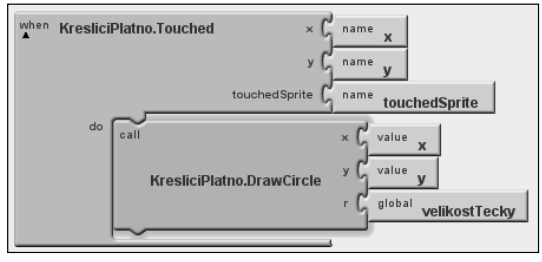
1. V Editoru bloků (Blocks Editor) otevřete ve sloupci Built-In (předdefinované bloky) přihrádku Definitions (definice). Přetáhněte z ní blok **def variable**. Slovo „variable“ změňte na „velikostTecky“.
2. Všimněte si, že blok **def velikostTecky** má jednu otevřenou přihrádku. Zde můžeme určit počáteční hodnotu proměnné, neboli hodnotu, kterou vrátí při spuštění aplikace. (V jazyce programátorů hovoříme o „inicializaci proměnné“.) V našem případě inicializujeme proměnnou `velikostTecky` hodnotou 2, a to tak, že vytvoříme blok **number 2** (buď začneme hodnotu psát anebo z přihrádky Math přetáhneme blok **number 123**) a připojíme ho do bloku **def velikostTecky**, viz obrázek 2.17.

Obrázek 2.17 Inicializace proměnné velikostTecky hodnotou 2

Práce s proměnnými

Dále budeme měnit argument bloku **KresliciPlatno.DrawCircle** v obslužné rutině události **KresliciPlatno.Touched** tak, aby se namísto pevně stanovené hodnoty vždy použila proměnná **velikostTecky**. (Může se zdát, že jsme proměnné `velikostTecky` stanovili hodnotu 2 napevno, nicméně za okamžik uvidíte, že hodnotu můžeme měnit – a tím pak i velikost vykreslené tečky.)

1. V Editoru bloků (Blocks Editor) přejděte do sloupce My Blocks (mé bloky) a otevřete přihrádku My Definitions (mé definice). Měli byste vidět dva nové bloky: (1) blok **global velikostTecky**, který zachycuje hodnotu proměnné, a (2) blok **set global velikostTecky**, který nastaví novou hodnotu proměnné. Tyto bloky se automaticky vytvoří spolu s proměnnou `velikostTecky`, podobně jako bloky hodnot argumentů `x` a `y`, které vznikly přidáním obslužné rutiny události **KresliciPlatno.Touched**.



Obrázek 2.18 Nyní závisí velikost každé z teček na hodnotě proměnné velikostTecky

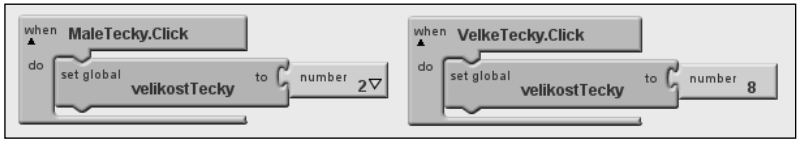
2. Přejděte do obslužné rutiny události **KresliciPlatno.Touched** a přetáhněte blok **number 5** z přihrádky `r` do koše. Poté ho nahraďte blokem **global velikostTecky** z přihrádky My Definitions (mé definice), viz obrázek 2.18. Jakmile se uživatel dotkne kreslicího plátna, aplikace načte poloměr z proměnné `velikostTecky`.

Změna hodnoty proměnné

Zde se dostává do hry skutečné kouzlo proměnných – proměnná `velikostTecky` umožňuje uživateli zvolit si velikost tečky, kterou pak obslužná rutina události příslušným způsobem vykreslí. Toto chování zapracujeme tak, že naprogramujeme obslužné rutiny **MaleTecky.Click** a **VelkeTecky.Click**:

1. Z přihrádky MaleTecky ve skupině My Blocks (mé bloky) přetáhněte obslužnou rutinu **MaleTecky.Click**. Poté z přihrádky My Definitions (mé definice) přetáhněte blok **set global velikostTecky to** a zapojte ho do obslužné rutiny **MaleTecky.Click**. Nakonec vytvořte blok **number 2** a zapojte ho do bloku **set global velikostTecky to**.
2. Podobným způsobem vytvořte obslužnou rutinu **VelkeTecky.Click**, nicméně hodnotu proměnné `velikostTecky` v ní nastavte na 8. Obě obslužné rutiny událostí by se nyní měly zobrazit v Editoru bloků (Blocks Editor), viz obrázek 2.19.

Poznámka: Slovo „global" v bloku **set global velikostTecky to** říká, že proměnnou můžeme použít ve všech obslužných rutinách událostí aplikace, tedy ***globálně***. Některé programovací jazyky umožňují definovat proměnné, které jsou „lokální" a lze je použít pouze v konkrétní části programu, App Inventor nikoliv.

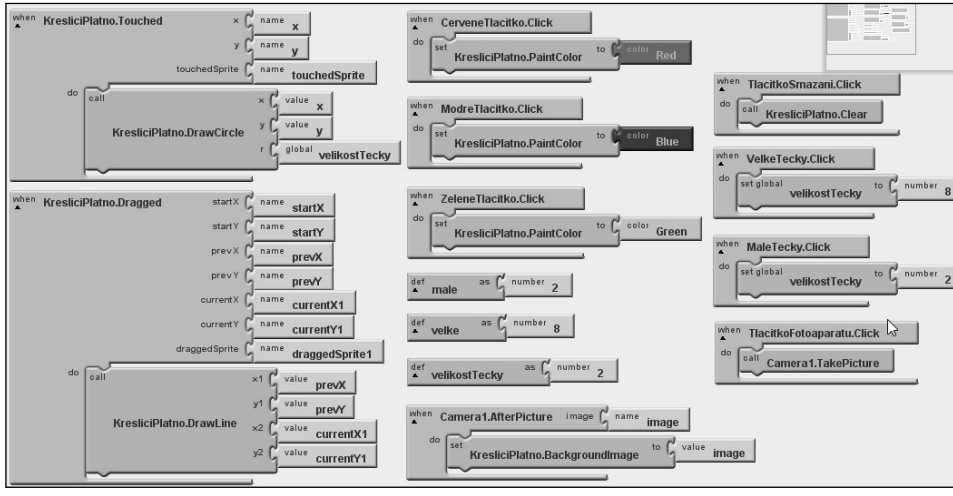


Obrázek 2.19 Hodnota proměnné velikostTecky se změní klepnutím na tlačítko, další klepnutí vykreslí tečku stanovené velikosti

Otestujte svou aplikaci: Zkuste klepnout na tlačítko určující velikost tečky a poté klepněte na kreslicí plochu. Postup opakujte s druhým tlačítkem. Vykreslují se vám různě velké tečky? A co čáry? Jejich velikost by se měnit neměla, protože proměnnou `velikostTecky` používáme pouze v bloku **KresliciPlatno.DrawCircle**. Dokážete si nyní již představit, jak byste upravili bloky, aby uživatelé mohli měnit i tloušťku čar? (V bloku ***Canvas*** byste použili vlastnost ***LineWidth***.)

Hotová aplikace: Kreslení

Na obrázku 2.20 vidíte hotovou aplikaci Kreslení.



Obrázek 2.20 Finální podoba bloků aplikace Kreslení

Variace

Můžete si vyzkoušet následující variace:

- Uživatelské rozhraní aplikace nenabízí mnoho údajů o aktuálním nastavení (například aktuální velikost tečky či barvu zjistíte až tehdy, když něco nakreslíte). Upravte aplikaci tak, aby se uživateli tyto informace zobrazily.
- Umožněte uživateli zadat velikost tečky prostřednictvím komponenty `TextBox`. Uživatel tak bude moci zadávat i jiné hodnoty než 2 a 8. Další informace o vstupních formulářích a komponentě `TextBox` najdete v kapitole 4.

Shrnutí

Následuje shrnutí toho, co jsme si popsali v této kapitole:

- Komponenta `Canvas` vám umožní kreslit. Rovněž rozpozná dotek a potažení prstem. Tyto události můžete přiřazovat kreslicím funkcím.
- Pomocí komponent pro úpravu rozložení obrazovky si můžete změnit rozvržení komponent, nemusíte je pouze pokládat pod sebe.
- Některé obslužné rutiny události uchovávají informace o události, například souřadnice místa, ve kterém došlo k doteku displeje. Tyto údaje se uchovávají v argumentech. Když

přetáhnete obslužnou rutinu události obsahující takové argumenty, App Inventor vytvoří bloky s hodnotami a umístí je do přihrádky My Definitions (mé definice).

- Proměnné vytváříme pomocí bloků **def variable**, které najdeme v přihrádce Definitions (definice). Proměnné umožňují aplikacím pamatovat si údaje, které se neuchovají ve vlastnostech komponent, například velikost tečky.
- App Inventor každé definované proměnné automaticky přiřadí blok **global value**, který předává hodnotu dané proměnné, ale také blok **set global variable to**, který změní hodnotu proměnné. Tyto bloky najdete v přihrádce My Definitions (mé definice). Chcete-li se o proměnných dozvědět více, nalistujte si kapitolu 16.

V této kapitole jsme si ukázali, jak můžeme použít komponentu `Canvas` v kreslicím programu. S její pomocí můžete také vytvářet animace, například ve 2D hrách. Chcete-li se dozvědět více, podívejte se na aplikaci Beruška v kapitole 5 a přečtěte si popis animací v kapitole 17.

KAPITOLA 3

Krtek

V této kapitole:
- Co budeme vytvářet
- Co se naučíte
- Začínáme
- Řízení chování komponent
- Hotová aplikace: Krtek
- Variace
- Shrnutí

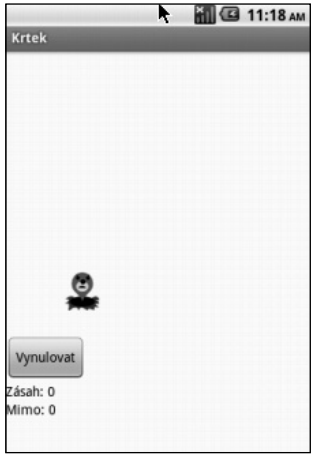
V této kapitole si ukážeme, jak vytvořit aplikaci Krtek, hru inspirovanou arkádovou klasikou Whac-A-Mole, ve které z děr rychle vykukují umělí krtci a hráči sbírají body tím, že je bouchají palicí. Hru Krtek vytvořila členka týmu App Inventoru, především proto, aby si vyzkoušela, jak fungují sprity obrázků (jsou implementovány), ale také proto, že má tuto hru ráda.

Když se Ellen Spertusová stala členkou týmu App Inventoru, chtěla vyvíjet hry, a tak se sama nabídla, že do prostředí zapracuje *sprity*. Tento pojem označuje v angličtině především mytologické bytosti, například víly. Do počítačové oblasti se výraz prodral v sedmdesátých letech, kdy označoval obrázky, které se na obrazovce pohybují (v počítačových hrách). Ellen se k práci se sprity dostala poprvé na setkání počítačových nadšenců na začátku osmdesátých let, kdy naprogramovala počítač TI 99/4. Při práci na spritech a hře Krtek ji pobízely dvě vzpomínky – na počítače i na hry z dětství.

Co budeme vytvářet

Ve hře Krtek, kterou vidíte na obrázku 3.1, použijeme následující funkce:

- Krtek se bude na obrazovce každou sekundu ukazovat na jiném, náhodném místě.
- Když se krtka dotknete, telefon zavibruje, skóre zásahů se zvýší (o jeden bod) a krtek se ihned přesune jinam.
- Když se dotknete displeje, ale krtka netrefíte, zvýší se počet minutí.
- Stisknutím tlačítka Vynulovat se vynuluje počet úspěšných a neúspěšných úderů.



Obrázek 3.1 Uživatelské rozhraní hry Krtek

Co se naučíte

Návod popisuje následující komponenty a koncepty:

- Komponentu `ImageSprite`, která ovládá na dotek citlivé pohyblivé obrázky.
- Komponentu `Canvas`, která bude fungovat jako plocha, na kterou umístíme komponentu `ImageSprite`.
- Komponentu `Clock`, která bude sprite přesouvat.
- Komponentu `Sound`, která při doteku krtka zavibruje.
- Komponentu `Button`, která spustí novou hru.
- Procedury obsluhující opakující se chování, například pohyb krtka.
- Generování náhodných čísel.
- Přičítání (+) a odečítání (-) bloků.

Začínáme

Připojte se ke stránce App Inventor a vytvořte nový projekt. Pojmenujte ho „Krtek“ a stejný název nastavte i titulku obrazovky. Otevřete Editor bloků (Blocks Editor) a připojte telefon.

Stáhněte si obrázek krtka z webové stránky knihy (*http://examples.oreilly.com/0636920016632*), pojmenujte ho *mole.png* a poznamenejte si, kam jste obrázek v počítači uložili. V Designéru komponent (Component Designer) klepněte v oddíle Media na Add (přidat) a přejděte k souboru umístěnému v počítači. Soubor nahrajte do App Inventoru.

Design komponent

Hru Krtek budeme tvořit pomocí následujících komponent:

- Komponenty `Canvas`, která slouží jako hrací pole.
- Komponenty `ImageSprite`, která zobrazuje obrázek krtka, může se pohybovat a registruje zásahy krtka.
- Komponenty `Sound`, která zavibruje, když krtka zasáhnete.
- Komponenty `Labels`, která zobrazí „Zásah:“ a „Mimo:“ a celkový počet zásahů a minutí.
- Komponenty `HorizontalArrangements`, která správně umístí komponenty `Label`.
- Komponenty `Button`, která vynuluje počitadla zásahů a minutí.
- Komponenty `Clock`, díky níž se krtek každou sekundu přesune.

Tabulka 3.1 zobrazuje kompletní seznam komponent.

Tabulka 3.1: Kompletní seznam komponent ve hře Krtek

Typ komponenty	Skupina palety	Název	Účel
`Canvas`	Basic	`Canvas1`	Kontejner komponenty `ImageSprite`.
`ImageSprite`	Animation	`Krtek`	Krtek se jí bude snažit dotknout.
`Button`	Basic	`TlacitkoVynulovani`	Uživatel bude jeho prostřednictvím vynulovávat skóre.
`Clock`	Basic	`Clock1`	Řídí krtkovy pohyby.
`Sound`	Media	`Sound1`	Vibruje, když se dotknete krtka.
`Label`	Basic	`PopisekZasahu`	Zobrazí „Zásah:“.
`Label`	Basic	`PopisekPocetZasahu`	Zobrazí počet zásahů.
`Horizontal Arrangement`	Screen Arrangement	`Horizontal Arrangement1`	Umístí komponentu `PopisekZasahu` vedle komponenty `PopisekPocetZasahu`.
`Label`	Basic	`PopisekMinuti`	Zobrazí „Mimo:“.
`Label`	Basic	`PopisekPocetMinuti`	Zobrazí počet minutí.
`Horizontal Arrangement`	Screen Arrangement	`Horizontal Arrangement2`	Umístí komponentu `PopisekMinuti` vedle komponenty `PopisekPocetMinuti`.

Umístění akčních komponent

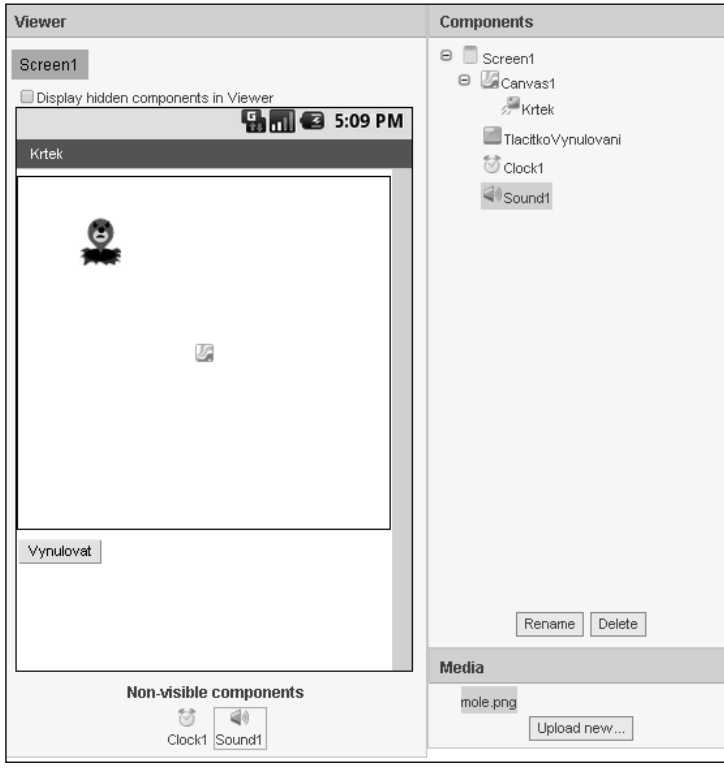
V tomto oddíle umístíme komponenty nezbytné k tomu, aby byla hra akční. V následujícím oddíle pak umístíme komponenty zobrazující skóre.

1. Přetáhněte komponentu `Canvas` a ponechte jí výchozí název Canvas1. Vlastnost `Width` jí nastavte na „Fill parent“ (vyplnit nadřazenou komponentu), čímž ji roztáhnete na celou obrazovku. Vlastnosti `Height` nastavte hodnotu 300 pixelů.
2. V paletě (Palette) přetáhněte ze skupiny Animation (animace) komponentu `ImageSprite`. Umístěte ji na plochu `Canvas1`. Klepněte na Rename (přejmenovat) ve spodní části seznamu

komponent (Components) a název změňte na „Krtek“. Vlastnost `Picture` změňte komponentě na nahraný soubor *mole.png*.

3. Ze skupiny Basic (základní) přetáhněte z palety komponentu `Button` a umístěte ji pod komponentu `Canvas1`. Přejmenujte ji na „TlacitkoVynulovani“ a vlastnost `Text` jí nastavte na „Vynulovat“.
4. Přetáhněte komponentu `Clock`. Komponenta se zobrazí ve spodní části prohlížeče, v oddíle neviditelných komponent („Non-visible components“).
5. Ze skupiny Media palety přetáhněte komponentu `Sound`. I ta se zobrazí v oddíle neviditelných komponent.

Na obrazovce byste nyní měli vidět něco podobného jako na obrázku 3.2 (i když krtek se může nacházet na jiném místě).



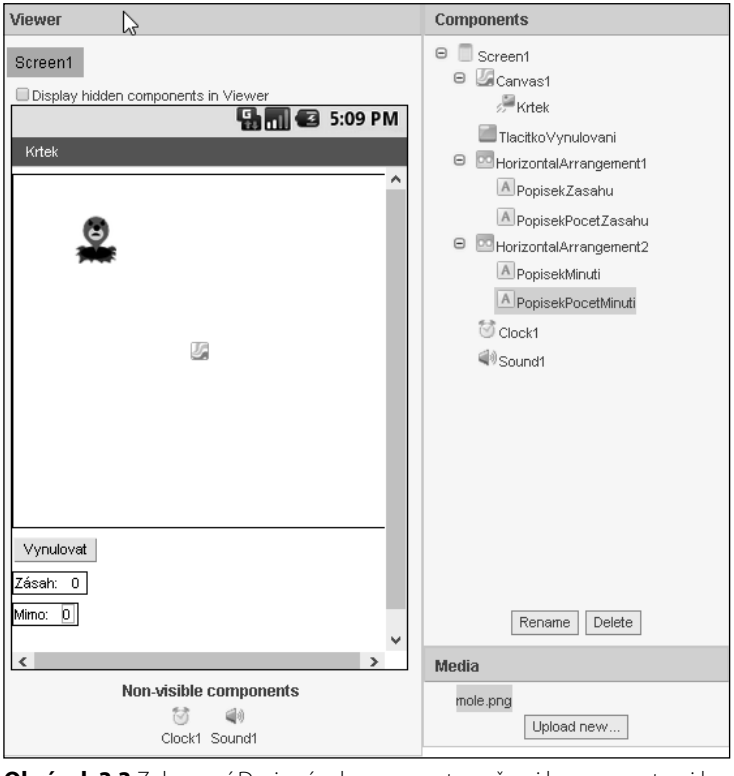
Obrázek 3.2 Zobrazení Designéra komponent a „akčních“ komponent

Umístění popisků

Nyní umístíme komponenty zobrazující uživatelovo skóre – konkrétně počet zásahů a minutí.

1. Ze skupiny Screen Arrangement (rozložení obrazovky) přetáhněte komponentu `HorizontalArrangement` a umístěte ji pod komponentu `Button`. Název `HorizontalArrangement1` neměňte.
2. Ze skupiny Basic přetáhněte do komponenty `HorizontalArrangement1` dvě komponenty `Label`.
3. Komponentu `Label` nalevo přejmenujte na „PopisekZasahu“ a její vlastnost `Text` nastavte na „Zásah: “ (za dvojtečkou udělejte mezeru).
4. Komponentu `Label` napravo přejmenujte na „PopisekPocetZasahu“ a její vlastnost `Text` nastavte na „0“.
5. Přetáhněte druhou komponentu `HorizontalArrangement` a umístěte ji pod komponentu `HorizontalArrangement1`.
6. Do komponenty `HorizontalArrangement1` přetáhněte dvě komponenty `Label`.
7. Komponentu `Label` nalevo přejmenujte na „PopisekMinuti“ a její vlastnost `Text` nastavte na „Mimo: “ (za dvojtečkou udělejte mezeru).
8. Komponentu `Label` napravo přejmenujte na „PopisekPocetMinuti“ a její vlastnost `Text` změňte na „0“.

Obrazovka by nyní měla vypadat přibližně jako na obrázku 3.3.



Obrázek 3.3 Zobrazení Designéra komponent se všemi komponentami hry Krtek

Řízení chování komponent

Komponenty již máme vytvořeny, takže se můžeme přesunout do Editoru bloků (Blocks Editor) a nastavit chování programu. Konkrétně chceme, aby se krtek každou sekundu náhodně přesunul na jiné místo. Uživatel bude mít za úkol klepnout na krtka vždy, když se někde objeví. Aplikace zobrazí počet zásahů a minutí. (*Poznámka*: Doporučujeme trefovat krtka prstem, ne palicí!) Klepnutím na tlačítko Vynulovat uživatel vynuluje počitadla zásahů a minutí.

Pohyb krtka

V programech, které jsme doposud vytvořili, jsme volali předdefinované procedury, například v aplikaci AhojKocoure to byla procedura `Vibrate`. Co kdyby App Inventor nabízel proceduru, která by komponentu `ImageSprite` po obrazovce přesouvala náhodně? Špatná zpráva je, že žádnou takovou proceduru nenabízí. Dobrá zpráva je, že si můžeme vytvářet své vlastní procedury! Stejně jako předdefinované procedury, i vlastní procedury se zobrazí v přihrádce a můžeme je použít kdekoliv v aplikaci.

Konkrétně vytvoříme proceduru, která bude krtka po obrazovce náhodně přesouvat. Nazveme ji `PresunKrtka`. Tuto proceduru budeme volat již při startu aplikace, když se uživatel dotkne krtka, ale také každou sekundu.

Tvorba procedury PresunKrtka

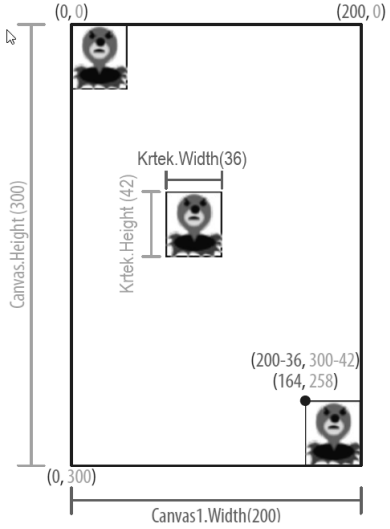
Abychom pochopili, jak budeme krtka přesouvat, budeme se muset podívat na to, jak na Androidu funguje grafika. Plochu (a obrazovku) můžeme chápat jako mřížku se souřadnicemi `x` (horizontální) a `y` (vertikální), které označují levý horní roh (0,0). Souřadnice `x` se navýší s přesunem doprava, zatímco souřadnice `y` se při pohybu dolů zmenší, viz obrázek 3.4. Vlastnosti `X` a `Y` komponenty `ImageSprite` ukazují, kde by se měl levý horní roh nacházet, takže budou mít hodnotu 0.

Ke zjištění maximálních hodnot `X` a `Y`, které zajistí, že se krtek vždy objeví na obrazovce, budeme muset použít vlastností `Width` a `Height` komponent `Krtek` a `Canvas1`. (Vlastnosti `Width` a `Height` krtka jsou shodné s velikostí nahraného obrázku. Když jsme vytvářeli komponentu `Canvas1`, nastavili jsme jí výšku `Height` na 300 pixelů a šířku `Width` na „Fill parent“ (vyplnit nadřazenou komponentu), takže zkopíruje šířku obrazovky.) Za předpokladu, že je krtek 36 pixelů a plocha 200 pixelů široká, souřadnice `x` nalevo od krtka může mít hodnotu 0 (zcela vlevo) až 164 (200-36, neboli `Canvas1.Width – Krtek.Width`), aniž by krtek překročil pravý okraj obrazovky. Podobně i souřadnice `y` nad krtkem může dosahovat 0 až `Canvas1.Height – Krtek.Height`.

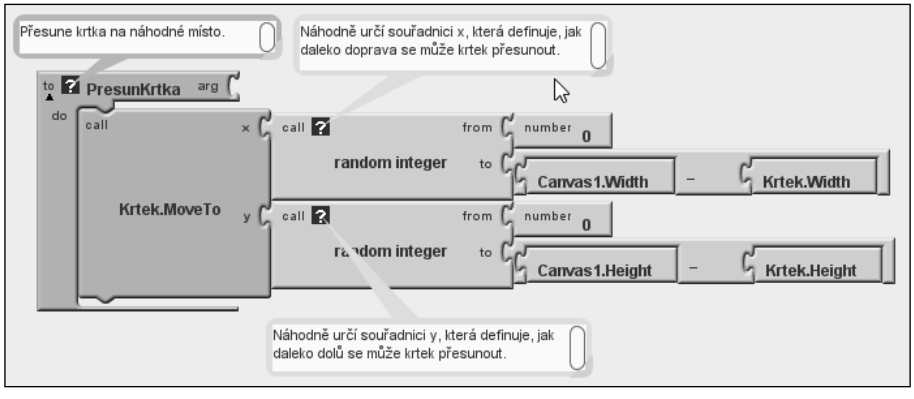
Obrázek 3.5 zobrazuje proceduru, kterou vytvoříme, s popisky (ty si můžete do procedury také přidat).

Aby se krtek mohl pohybovat náhodně, budeme souřadnici `x` nechávat vybírat z rozsahu `0` až `Canvas1.Width – Krtek.Width`. Podobně souřadnice `y` bude v rozsahu `0` až `Canvas1.Height –`

`Krtek.Height`. Náhodné číslo můžeme generovat pomocí předdefinované procedury `random integer`, kterou najdeme v přihrádce Math. Výchozí parametr „from“ (od) budeme muset změnit na 0 a parametry „to“ (do) budeme muset upravit dle obrázku 3.5.



Obrázek 3.4 Pozice krtka na obrazovce se souřadnicemi a údaji o výšce a šířce; souřadnice x a šířky jsou vyobrazený modře, zatímco souřadnice y a výšky oranžově



Obrázek 3.5 Procedura PresunKrtka, která přesune krtka do náhodného místa

Proceduru vytvoříme následujícím způsobem:

1. Klepněte na přihrádku Definition (definice), kterou v Editoru bloků (Blocks Editor) najdete na kartě Built-In (předdefinované bloky).
2. Přetáhněte blok **to procedure** (ne **to procedureWithResult**).
3. V novém bloku klepněte na text „procedure“ a zadejte „PresunKrtka“. Nastavíte tak název procedury.

4. Protože chceme krtka přesouvat, klepněte na kartu My Blocks (mé bloky). Dále klepněte na přihrádku Krtek a blok **Krtek.MoveTo** přesuňte do procedury, napravo od „do" (dělat). Všimněte si, že musíme zadat souřadnice `x` a `y`.
5. Abychom určili, že nová souřadnice `x` krtka musí být v rozmezí 0 až `Canvas1.Width – Krtek.Width`, budeme postupovat následovně:
6. Klepněte na kartu Built-In, zobrazíte tak předdefinované bloky.
7. Klepněte na přihrádku Math.
8. Přetáhněte blok **random integer**, přičemž výstupek nalevo zapojte do zásuvky „x" bloku `Krtek.MoveTo`.
9. Klepněte na blok **number 1** v zásuvce „from" a zadejte mu hodnotu 0.
10. Klepněte na blok **number 100** a stiskněte klávesu Delete anebo ho přetáhněte do odpadkového koše.
11. Klepněte na zásuvku Math a přetáhněte odečítací (-) blok do zásuvky „to".
12. Zobrazte své komponenty klepnutím na My Blocks (mé bloky).
13. Klepněte na zásuvku Canvas1 a přesuňte se dolů k bloku **Canvas1.Width**, který byste měli přetáhnout nalevo od odečítací operace.
14. Podobně klepněte i na zásuvku Krtek a přetáhněte blok **Krtek.Width** napravo od odečítacího bloku.
15. Podobným způsobem nastavte souřadnici `y`, která bude náhodným celým číslem v rozsahu 0 až `Canvas1.Height – Krtek.Height`.
16. Překontrolujte si výsledky oproti obrázku 3.5.

Vyzkoušejte si volání **Krtek.MoveTo**. Klepněte na blok pravým tlačítkem a vyberte možnost Do It (provést). (Možná budete muset aplikaci nejprve klepnutím na „Connect to Device" restartovat.) Měli byste vidět, jak se krtek pohybuje po obrazovce a pravidelně se přesouvá na náhodná místa (s výjimkou zcela nepravděpodobného případu, kdy generátor náhodných čísel dvakrát za sebou vydá stejné číslo).

Volání procedury PresunKrtka při spuštění aplikace

Nyní již máme proceduru `PresunKrtka` připravenu, takže ji použijeme. Při programování je obvyklé, že je třeba, aby k něčemu došlo již při spuštění aplikace. Pro tyto případy zde máme blok **Screen1.Initialize**.

1. Klepněte na My Blocks (mé bloky), poté na přihrádku Screen1 a vytáhněte z ní blok **Screen1.Initialize**.
2. Klepněte na přihrádku My Definitions (mé definice), v níž uvidíte blok **call PresunKrtka**. (Tvořit bloky je fajn!) Vytáhněte ho ven a umístěte ho do bloku **Screen1.Initialize**, viz obrázek 3.6.

Obrázek 3.6 Volání procedury PresunKrtka při spuštění aplikace

Volání procedury PresunKrtka každou sekundu

Aby se mohl krtek přesunout každou sekundu, budeme potřebovat komponentu `Clock`. Její vlastnosti `TimerInterval` jsme ponechali výchozí hodnotu 1000 (milisekund) neboli 1 sekundu. To znamená, že se každou sekundu provede to, co udává blok **Clock1.Timer**. Ten nastavíme následujícím způsobem:

1. Klepněte na My Blocks (mé bloky), poté na přihrádku Clock1 a vytáhněte z ní blok **Clock1.Timer**.
2. Klepněte na přihrádku My Definitions (mé definice) a do bloku **Clock1.Timer** přetáhněte blok **call PresunKrtka**, viz obrázek 3.7.

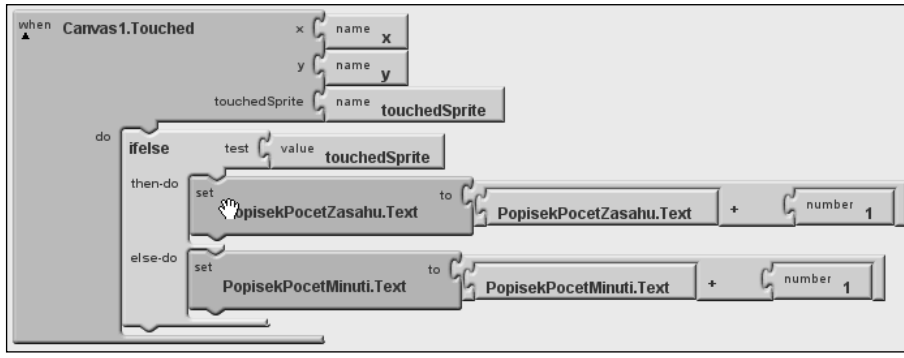
Obrázek 3.7 Volání procedury PresunKrtka podle časovače (každou sekundu)

Pokud je pro vás zadaný interval příliš krátký či dlouhý, můžete vlastnost `TimerInterval` komponenty `Clock1` v Designéru komponent (Component Designer) upravit a krtka nechat střídat pozice rychleji či pomaleji.

Počítání skóre

Jistě si vzpomenete, že jsme vytvořili dva popisky, `PopisekPocetZasahu` a `PopisekPocetMinuti`, s počáteční hodnotou 0. Čísla v těchto popiscích bychom rádi zvýšili vždy, když uživatel krtka zasáhne anebo mine. K tomu nám poslouží blok **Canvas1.Touched**, který indikuje, že došlo k doteku plochy, souřadnice x a y místa, kde k dotyku došlo (ty znát nepotřebujeme), a zda došlo k dotyku spritu (což naopak vědět potřebujeme). Připravený kód vidíte na obrázku 3.8.

Co se děje na obrázku 3.8? Kdykoliv dojde k dotyku obrazovky, zkontroluje se, zda došlo k dotyku spritu. Protože se v programu nachází pouze jeden sprite, musí to být `Krtek1`. Pokud dojde k dotyku spritu `Krtek1`, vlastnost `PopisekPocetZasahu.Text` se navýší o 1. V opačném případě se přičte 1 k vlastnosti `PopisekPocetMinuti.Text`. (Blok **touchedSprite** má hodnotu `false`, pokud nedojde k dotyku.)



Obrázek 3.8 K navýšení počtu zásahů (PopisekPocetZasahu) či minutí (PopisekPocetMinuti) dojde při doteku plochy Canvas1

Bloky vytvoříme následujícím způsobem:

1. Klepněte na My Blocks (mé bloky), poté na přihrádku Canvas1 a vytáhněte z ní blok **Canvas1.Touched**.
2. Klepněte na Built-In (předdefinované bloky), poté na přihrádku Control a do bloku **Canvas1.Touched** z ní přetáhněte blok **ifelse**.
3. Klepněte na My Blocks (mé bloky), následně na přihrádku My Definitions (mé definice) a do testovací zásuvky bloku **ifelse** z něj přetáhněte blok **touchedSprite**.
4. Pokud dojde k zásahu krtka, hodnota vlastnosti `PopisekPocetZasahu.Text` by se měla zvýšit o 1.
5. Z přihrádky PopisekPocetZasahu přetáhněte blok **set PopisekPocetZasahu.Text to** a umístěte ho napravo od „then-do“ (poté proveď).
6. Klepněte na Built-In (předdefinované bloky), klepněte na přihrádku Math a přetáhněte plusové znaménko (+) do zásuvky „to“ (do).
7. Klepněte na My Blocks (mé bloky), dále na přihrádku PopisekPocetZasahu a přetáhněte blok **PopisekPocetZasahu.Text** nalevo od plusového znaménka.
8. Klepněte na Built-In (předdefinované bloky), dále na přihrádku Math a přetáhněte blok **number 123** napravo od plusového znaménka. Klepněte na hodnotu 123 a změňte ji na 1.
9. Krok 4 opakujte s popiskem `PopisekPocetMinuti` v sekci „else-do“ (jinak proveď) bloku **ifelse**.

Otestujte svou aplikaci: Nový kód si můžete vyzkoušet na telefonu. Klepněte na plátno, na krtka i mimo a sledujte, jak se bude měnit skóre.