J. F. DiMarzio

Programujeme hry pro Android 4

Návrh a vývoj hry ve 2D i 3D

Tvorba prostředí a postav

Umělá inteligence nepřátel, reálná fyzika

WWW

Ke stažení zdrojové kódy příkladů z knihy

computer press

Apress®

J. F. DiMarzio

Programujeme hry pro Android 4

Computer Press
Brno
2012

Programujeme hry pro Android 4

J. F. DiMarzio

Překlad: Lukáš Krejčí
Obálka: Martin Sodomka
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Original edition copyright © 2011 by J. F. DiMarzio. All rights reserved.
Czech edition copyright © 2012 by Albatros Media a.s. All rights reserved.
Translation © Lukáš Krejčí, 2012.

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN 978-80-251-3754-3

Vydalo nakladatelství Computer Press v Brně roku 2012 ve společnosti Albatros Media a. s.
se sídlem Na Pankráci 30, Praha 4. Číslo publikace 16 338.
© Albatros Media a. s. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného souhlasu vydavatele.

1. vydání

ALBATROS MEDIA a.s.

Obsah

Předmluva **9**
Poděkování **10**
Úvod **11**
Zpětná vazba od čtenářů **12**
Zdrojové kódy ke knize **12**
Errata **12**

ČÁST I
PLÁNOVÁNÍ A TVORBA DVOJROZMĚRNÝCH HER

KAPITOLA 1
Vítejte v herním světě platformy Android **15**
Programování her pro Android **16**
Začněte dobrým příběhem **17**
Proč záleží na příběhu 18
Zapsání příběhu 19
Co vás čeká **22**
Příprava vývojových nástrojů pro platformu Android **22**
Instalace knihovny OpenGL ES 24
Vyberte si verzi platformy Android **26**
Shrnutí **26**

KAPITOLA 2
Star Fighter: Dvojrozměrná střílečka **27**
Příběh v pozadí hry Star Fighter **27**
Co dělá hru hrou? **29**
Porozumění hernímu enginu 30
Pochopení kódu specifického pro hru 32
Průzkum herního enginu pro hru Star Fighter 34
Vytvoření projektu Star Fighter **35**
Shrnutí **38**

KAPITOLA 3
Stiskněte Start: Tvorba nabídky **39**
Sestavení úvodní obrazovky **40**
Vytvoření aktivity 41
Vytvoření obrázku pro úvodní obrazovku 46
Práce se souborem R.java 48
Vytvoření souboru rozvržení 49
Vytvoření efektů prolnutí 56
Práce s vlákny ve hře 58
Tvorba hlavní nabídky **64**
Přidání obrázků pro tlačítka 65
Nastavení rozvržení 66
Propojení tlačítek 69
Přidání posluchačů klepnutí na tlačítka 70
Přidání hudby **72**
Vytvoření hudební služby 74
Přehrávání hudby 79
Shrnutí **82**

KAPITOLA 4
Kreslení prostředí **83**
Vykreslování pozadí **84**
Vytvoření aktivity pro hru 85
Vytvoření vykreslovací jednotky 89
Nahrávání obrázku pomocí OpenGL 96
Posouvání pozadí 106
Přidání druhé vrstvy **112**
Nahrávání druhé textury 114
Posouvání druhé vrstvy 115
Práce s maticemi 116
Dokončení metody scrollBackground2() 118
Běh rychlostí 60 snímků za vteřinu **120**
Pozastavení cyklu hry 121
Vyčištění mezipamětí knihovny OpenGL 123
Úprava hlavní nabídky **125**
Shrnutí **126**

KAPITOLA 5
Tvorba postavy **127**
Animování spritů **127**
Nahrávání postavy **130**
Vytvoření polí pro mapování textury 131
Načtení textury do postavy 135
Příprava herního cyklu 140
Pohybování postavou **141**
Nakreslení výchozího stavu postavy 143
Programování akce PLAYER_RELEASE 145
Pohybování postavou doleva 147
Nahrání správného spritu 148
Nahrání druhého snímku animace 151
Pohybování postavou doprava 154
Nahrání animace náklonu vpravo 155
Pohybování postavou pomocí události doteku **158**
Informace v objektu typu MotionEvent 159
Zachycení akcí ACTION_UP a ACTION_DOWN 160
Úprava prodlevy při vykreslování **163**
Shrnutí **164**

KAPITOLA 6
Přidání nepřátel **165**
Úklid uprostřed hry **165**
Vytvoření třídy pro textury **166**
Vytvoření třídy pro nepřátele **170**
Přidání nového archu spritů 170
Vytvoření třídy SFEnemy 172
Bézierova křivka 176
Shrnutí **181**

KAPITOLA 7
Přidání jednoduché umělé inteligence nepřátel **183**
Příprava nepřátel pro umělou inteligenci **183**
Vytvoření logiky každého z nepřátel 185
Inicializace nepřátel 188
Nahrání archu spritů 189

Přezkoumání umělé inteligence **190**
Vytvoření metody moveEnemy() 190
Vytvoření cyklu nad polem enemies[] 191
Pohyb každého z nepřátel pomocí jeho logiky umělé inteligence 191
Tvorba umělé inteligence Stíhače **192**
Úprava vrcholů 193
Zaměření na pozici hráče 194
Implementování vzorce pro sklon 196
Tvorba umělé inteligence Průzkumníka **202**
Příprava náhodného bodu pro pohyb Průzkumníka 204
Pohyb po Bézierově křivce 205
Tvorba umělé inteligence Válečné lodě **208**
Shrnutí **209**

KAPITOLA 8
Braňte se! **211**
Vytvoření archu spritů se zbraněmi **211**
Tvorba třídy pro zbraň 212
Dráha střely **215**
Vytvoření pole zbraní 215
Přidání druhého archu spritů 216
Inicializace zbraní 217
Pohyb zbraňových střel 218
Detekování okraje obrazovky 220
Volání metody firePlayerWeapons() 222
Implementování detekce kolizí **223**
Aplikování kolizního poškození 223
Tvorba metody detectCollisions() 224
Samotné detekování kolize 226
Odebrání neplatných střel 226
Rozvíjení získaných dovedností **229**
Shrnutí **229**
Kontrola nejdůležitějšího kódu pro dvojrozměrnou hru **229**

KAPITOLA 9
Publikování hry **247**
Příprava manifestu **247**

Příprava pro podepsání, zarovnání a vydání **248**
Kontrola připravenosti souboru AndroidManifest.xml 250
Vytvoření úložiště klíčů 252
Shrnutí **254**

ČÁST II
TVORBA TROJROZMĚRNÝCH HER

KAPITOLA 10
Blob Hunter: Tvoříme trojrozměrné hry **257**
Srovnání dvojrozměrných a trojrozměrných her **257**
Vytvoření projektu pro trojrozměrnou hru **258**
Soubor BlobhunterActivity.java 259
Třída BHGameView 260
Třída BHGameRenderer 260
Třída BHEngine 261
Vytvoření testu trojrozměrného objektu **262**
Přidání konstanty 262
Přidání třídy BHWalls 263
Tvorba instance třídy BHWalls 265
Namapování obrázku 266
Použití metody gluPerspective() 268
Vytvoření metody drawBackground() 269
Finální úpravy 271
Shrnutí **273**

KAPITOLA 11
Tvorba pohlcujícího prostředí **275**
Použití třídy BHWalls **275**
Vytvoření chodby z více instancí třídy BHWalls 276
Třída BHCorridor **277**
Sestavení třídy BHCorridor 278
Přidání textury pro zeď 287
Použití třídy BHCorridor **288**
Shrnutí **289**

KAPITOLA 12
Pohyb v trojrozměrném prostředí **291**
Vytvoření ovládacího rozhraní **291**
Úprava třídy BHEngine 292
Úprava třídy BlobhunterActivity 293
Pohyb hráče dopředu 294
Pohyb chodbou **295**
Úprava výhledu hráče 297
Shrnutí **298**
Kontrola nejdůležitějšího kódu pro trojrozměrnou hru **298**

Rejstřík **305**

Předmluva

Když jsem byl mladý, snil jsem jako téměř každý kluk mého věku o tvorbě videoher, neměl jsem však žádnou představu ani o tom, kde začít. Každý má možná nápad na skvělou hru, ale dát dohromady nástroje pro její vytvoření je úplně něco jiného. Internet byl ještě v plenkách a k dispozici bylo jen málo zdrojů o vývoji her, poněvadž také ti, kteří se v tomto průmyslu pohybovali, se i nadále zabývali tím, jak něco vyřešit. Pro mě se situace změnila těsně po dvacítce, kdy jsem zjistil, že univerzity začaly učit design a vývoj her.

Pamatuji si, že i po dokončení studií bylo pro mne stále velmi málo příležitostí, jak ukázat svým potenciálním zaměstnavatelům své dovednosti. Byl jsem dobrým programátorem, neexistovalo však vesměs nic takového jako software pro vývoj her, který by mi umožnil zaměřit se na tvorbu samotné hry. Pro vytvoření čehokoli většího než té nejjednodušší hry byl zapotřebí tým vývojářů. Pro jediného vývojáře neexistovalo nic, co by mu umožňovalo živit se prací na volné noze, neovládal-li všechny typy programování, umění a designu a neměl-li možnost vydržovat se při několikaleté práci na hře.

Situace se začala rychle měnit s rozmachem trhu s hrami na sociálních sítích a s takovým nárůstem výkonu mobilních zařízení, že na nich bylo možné spouštět zábavné hry. Věci se dále vyvíjely až do té míry, že dnes mohu vidět hry, které jsem před deseti lety hrál na konzole, plně funkční na mobilním zařízení. Spolu s tím přišla softwarová prostředí pro vývoj her, která umožnila vývojářům snadno vytvářet hry a soustředit se na zábavu a funkce, a ne na to, jak vše sestavovat dohromady.

Dnes mají vývojáři her tolik možností, že se musejí rozhodovat, čemu budou věnovat svůj čas. Je-li vaším cílem flexibilita, sáhněte po platformě Android s jejím otevřeným prostředím, jež podporuje vývojáře a dává jim možnosti ohledně toho, jak a kde nabídnout obsah spotřebitelům. Kromě toho je jednoduché vytvořit obsah, který je použitelný jak na tabletech s Androidem, tak i na mobilních zařízeních, díky čemuž je při stejném vynaloženém úsilí vaše šance na zisk mnohem vyšší.

Je-li pro vás vývoj pro platformu Android jen odrazovým můstkem, pak vězte, že Java je široce používaný jazyk, takže budete moci využít získané znalosti i v budoucnu. Java je kromě toho tím nejjednodušším, s čím můžete začít. Kéž bych měl takové nástroje a platformy, když jsem začínal svoji kariéru! Nyní je nejlepší čas naskočit a splnit si sen o tvorbě her!

Jameson Durall
Game Designer
@siawnhy (Twitter)
www.jamesondurall.com

Poděkování

Chtěl bych poděkovat každému, kdo umožnil vznik této knihy: můj agent Neil Salkind a každý ve Studiu B; Steve Anglin, Corbin Collins, James Markham, Yosun Chang, Tony Hillerson a osazenstvo v nakladatelství Apress; Ben Eagle (grafika ve hře); MD, JS, CL, DL, MB, JK, CH, BB, DB a KK z DWSS; a všichni ostatní, na které jsem zapomněl a kteří mi na této cestě také pomáhali.

Úvod

Vítejte v knize *Programujeme hry pro Android 4*. Tato kniha vás provede krok za kokem vývojem dvou odlišných mobilních her – od koncepce ke kódu. Naučíte se, jak začít tvorbu hry základní představou a pokračovat komplexním programováním enginu pro převedení této představy do hratelné podoby.

Rozhodl jsem se napsat tuto knihu, abych se s vámi mohl podělit o dovednosti nezbytné pro tvorbu vašich vlastních dvojrozměrných a trojrozměrných her pro platformu Android 4. Android 4 sjednocuje operační systémy mobilních telefonů a tabletů založených na Androidu pod jednou společnou sadou SDK. To znamená, že vámi vytvořené hry lze hrát na nejnovějších tabletech a telefonech a tom nejlepším možném hardwaru. Tutéž hru je možné nyní hrát na jakémkoli druhu zařízení – stačí jen učinit první krok a vytvořit neodolatelnou hru.

Při prvním vydání sady Android SDK s podporou dvojrozměrné a trojrozměrné grafiky knihovny OpenGL ES jsem okamžitě začal hledat způsoby, jak vytvářet hry, které by byly podmanivé a zábavné. V té době jsem si uvědomil, že ačkoliv zvládnutí dovedností nutných pro jejich tvorbu sice není obtížné, zcela jistě není pro každého snadné, aby je objevil. Ve skutečnosti je to tak, že pokud jste neměli nějaké dřívější zkušenosti s knihovnou OpenGL, konkrétně s knihovnou OpenGL ES, bylo velice obtížné pustit se jen tak do příležitostného vývoje her pro platformu Android.

Rozhodl jsem se tedy vzít, co jsem naučil se při vývoji příležitostných her na platformě Android, a rozdělit tyto znalosti na ústřední skupinu základních dovedností, které si lze snadno osvojit a rozvíjet je během vývoje hry. Tyto základní dovednosti vás možná nepovedou k tomu, že ihned po dočtení této knihy vytvoříte další Red Faction: Armageddon, dají vám však znalosti nezbytné pro porozumění tomu, jak se takové hry dělají, abyste je tak možná jednou mohli vytvářet se správným nasazením a praxí.

Bezpochyby máte svoji první hru pro platformu Android již v hlavě. Naprosto přesně víte, jak má vypadat a jak se má hrát. Co ovšem nevíte, je to, jak tuto představu dostat ze své hlavy do telefonu či tabletu. Je sice hezké mít nápad na hru, obtížné je však právě přechod od fáze nápadu k fázi „hratelné na mobilním zařízení".

Moje rada pro čtenáře této knihy zní: udržuje své nápady co nejjednodušší. Nepokoušejte se překomplikovat dobrou hru jen proto, že tak můžete učinit. Chci říci, že některé z „nejnávykovějších" her nepatří nutně mezi nejsložitější. Obvykle se jedná o hry, které si snadno osvojíte a zahrajete, ale těžko se od nich odtrhnete. Mějte toto na paměti, jakmile si začnete vytvářet představu o hrách, které chcete vytvořit. V této knize uděláte jednoduchý engine, který bude pohánět posouvací střílečku. Jde o jednoduchý typ hry, který může obsahovat velice obtížné a náročné hry. Dlouhou dobu byl řazen mezi ty za návykovější arkádové hry, protože nabízí rychlou akci a téměř neomezené množství hraní. Je velice jednoduché vrátit se do doby posouvacích střílček a znovu si dobře zahrát. Z tohoto důvodu začneme s tímto typem her

i v této knize. Koneckonců, pokud se snažíte vytvářet hry, které byste si jako hráči rádi zahráli, bude výsledek rozhodně stát za to. Doufám, že si cestu do nádherného světa vývoje her pro platformu Android náležitě užijete!

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/K2030* 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 pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2030* po klepnutí na odkaz Soubory ke stažení.

ČÁST I

PLÁNOVÁNÍ A TVORBA DVOJROZMĚRNÝCH HER

První část této knihy, kapitoly 1 až 9, vás provede procesy plánování a tvorby hratelné dvojrozměrné hry pro platformu Android s názvem Star Fighter. Tvorba této hry bude postupovat podle odlišné, avšak logické cesty. Nejdříve naplánujete a napíšete příběh v pozadí hry. Dále vytvoříte pozadí pro hru. Potom vytvoříte hratelné a nehratelné postavy. Nakonec vytvoříte zbraňové systémy a detekci kolizí. Před provedením kroků v kapitole 9, nezbytných pro nasazení vaší hry na mobilní zařízení, budete mít na konci kapitoly 8 k dispozici kompletní výpisy kódu nejdůležitějších souborů, které jste vytvořili nebo upravili v první části této knihy. Tyto výpisy použijte pro porovnání s vaším kódem a pro zajištění, aby každá hra fungovala tak, jak má. Tím budete připraveni na vývoj trojrozměrných her, jemuž se věnuje druhá část knihy (kapitoly 10 až 12).

KAPITOLA 1

Vítejte v herním světě platformy Android

V této kapitole:

- Programování her pro Android
- Začněte dobrým příběhem
- Proč záleží na příběhu
- Zapsání příběhu
- Co vás čeká
- Příprava vývojových nástrojů pro platformu Android
- Instalace knihovny OpenGL ES
- Vyberte si verzi platformy Android
- Shrnutí

S vývojem pro platformu Android jsem začal na začátku roku 2008, kdy byla k dispozici beta verze platformy. V té době nebyly pro tento nový operační systém ohlášeny žádné telefony a my vývojáři jsme opravdu cítili, jako bychom stáli na začátku něčeho vzrušujícího. Android na sebe strhl veškerou energii a nadšení prvních dnů vývoje s otevřeným zdrojovým kódem. Vývoj pro tuto platformu silně připomínal posedávání v prázdné studentské hale ve 2 hodiny ráno se silně kofeinovým nápojem a čekání na přidělení času v systému VAX pro spuštění svého nejnovějšího kódu. Bylo úžasné sledovat zhmotnění této platformy a jsem rád, že jsem mohl být u toho.

S tím, jak Android rostl a společnost Google vydávala další aktualizace pro upevnění finální architektury, začalo být jasné, že by platforma Android, která je založená na Javě a obsahuje řadu dobře známých balíčků Javy, mohla představovat snadný přechod pro příležitostné vývojáře her. Většina znalostí, jimiž vývojáři Javy již disponují, by se totiž dala na této nové platformě recyklovat. Velice rozsáhlá základna vývojářů her v Javě by mohla tyto znalosti použít pro docela plynulý přechod na platformu Android.

Jak tedy vývojář v Javě začne vyvíjet hry na platformě Android a jaké nástroje k tomu potřebuje? Tato kapitola se zaměřuje na odpověď na tyto otázky, avšak nejen na to. Naučíte se totiž také to, jak poskládat příběh vaší hry do kousků, které lze plně realizovat jako části vaší hry. Prozkoumáme některé z nepostradatelných nástrojů nezbytných pro provádění úkolů v budoucích kapitolách.

Tato kapitola je velice důležitá, neboť vám dává něco, co mnoho jiných knih o programování her nemá: skutečné soustředění na vznik hry. Přestože znalost toho, jak napsat kód, který přivede hru k životu, je velice důležitá, skvělý kód vám nepomůže, pokud nemáte žádnou hru, kterou byste mohli přivést k životu. Povědomí o tom, jak dostat nápad na hru z vaší hlavy jasným a čistým způsobem, činí rozdíl mezi dobrou hrou a hrou, kterou hráč nemůže odložit.

Programování her pro Android

Vyvíjení her na platformě Android má své výhody i nevýhody, kterých byste si měli být vědomi ještě před tím, než začnete. Za prvé, hry na platformě Android se vyvíjejí v jazyce Java, Android však není kompletní implementací Javy. Spousta balíčků, které jste možná používali pro OpenGL a další grafické dekorace, je obsažená v sadě SDK platformy Android. „Spousta" ovšem neznamená „všechny", takže některé skutečně užitečné balíčky pro vývojáře her, zvláště pak pro vývojáře trojrozměrných her, zde obsaženy nejsou. Ne každý balíček, na který jste při sestavování svých předchozích her spoléhali, budete mít na platformě Android k dispozici.

S každým vydáním nové sady Android SDK je k dispozici více a více balíčků a starší mohou být označeny jako zastaralé. Musíte si tedy dávat pozor, s kterými balíčky je nutné pracovat, čemuž se budeme postupně věnovat v průběhu jednotlivých kapitol.

Další výhodou je dobrá znalost platformy Android a nevýhodou nedostatek jejího výkonu. Co může Android nabídnout na straně dobré znalosti a snadnosti programování, postrádá na straně rychlosti a výkonu. Většina videoher, jako jsou ty napsané pro PC nebo konzoly, se vyvíjí v jazycích nízké úrovně, jako je C, nebo dokonce jazyk symbolických adres. Vývojáři tak obdrží největší kontrolu nad způsobem, jak daný kód provádí procesor a prostředí, v němž běží. V procesorech běží kód na velmi nízké úrovni a čím blíže se můžete dostat k nativnímu jazyku procesoru, tím méně interprety musí vaše hra pro svůj běh projít. Přestože platforma Android nabízí omezenou možnost programovat na nízké úrovni, interpretuje a proplétá váš kód Javy skrze svůj vlastní prováděcí systém. Vývojář tak má menší kontrolu nad prostředím, v němž jeho hra běží.

Tato kniha vás neprovede přístupy k vývoji her na nízké úrovni. Proč? Protože Java, a zvláště způsob, jakým je prezentována pro obecný vývoj pro Android, je široce známá, snadno použitelná a dokáže vytvářet velice zábavné a vděčné hry.

V zásadě platí, že pokud jste již zkušenými vývojáři v Javě, tak zjistíte, že při přechodu na Android nebudou vaše dovednosti ztracené. Pokud dosud nejste ostříleným vývojářem v Javě, ničeho se nebojte. Java je pro zahájení výuky naprosto skvělý jazyk. Z tohoto důvodu se budeme při psaní našich her držet nativního vývojového prostředí Javy na platformě Android.

Uvedli jsme si několik výhod a nevýhod při vývoji her na platformě Android. Nicméně jednou z největších výhod pro nezávislé a příležitostné vývojáře her při vytváření a publikování her na platformě Android je svoboda, kterou máte zajištěnou při vydávání svých her. Zatímco některé elektronické obchody s aplikacemi mají velice přísná pravidla ohledně toho, co v nich

lze prodávat a za kolik, Android Market je jiný. Kdokoli zde může vystavovat a prodávat téměř cokoliv, co chce. To dává vývojářům mnohem větší míru kreativní svobody.

V kapitole 2 vytvoříte svoji první hru založenou na Androidu, třebaže bude velice jednoduchá. Ze všeho nejdříve je ovšem nutné nahlédnout za oponu a podívat se, co inspiruje jakoukoli přínosnou hru: příběh.

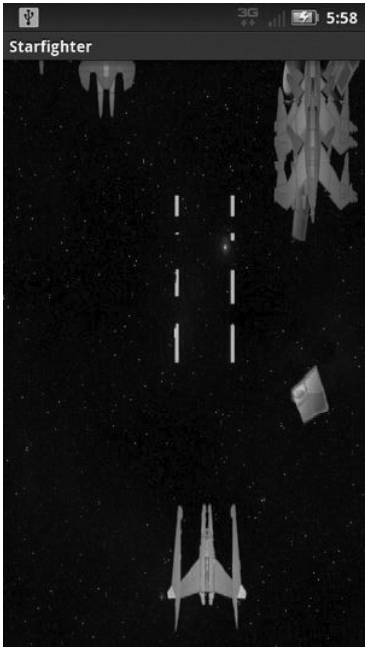
Začněte dobrým příběhem

Každá hra, od nejjednodušší arkádové hry po nejkomplexnější hry typu RPG, začíná příběhem. Příběhem nemusí být nic více nežli věta, například: Představte si, že máme obrovskou vesmírnou loď, která dokáže střílet.

Příběh ovšem může být i dlouhý, jako kniha, a popisovat každé území, osobu a zvíře v prostředí hry. Mohl by dokonce popisovat i každou zbraň, úkol a výsledek.

Poznámka: Příběh načrtává akci, účel a tok hry. Čím více podrobností do něj vložíte, tím snazší bude vaše práce při vývoji kódu.

Podívejte se na hru na obrázku 1.1. Říká vám to něco? Jedná se o obrazovku ze hry Star Fighter, což je hra, kterou budete vyvíjet v rámci počátečních kapitol této knihy. Za touto hrou je však také příběh.



Obrázek 1.1: Obrázek ze hry Star Fighter

Většina z nás si nikdy nepřečte příběhy, jež pomohly vytvořit některé z našich oblíbených her, protože tyto příběhy jsou ve skutečnosti podstatné pouze pro lidi, kteří danou hru vytvářejí. A za předpokladu, že vývojáři a tvůrci odvedou dobrou práci, vstřebá hráč příběh při hraní hry.

V malých, nezávislých vývojových studiích se může stát, že tyto příběhy nebude číst nikdo, kromě hlavního vývojáře. Ve větších společnostech zabývajících se vývojem her by na takovém příběhu mohlo ještě před tím, než skončí v rukou hlavního vývojáře, pracovat několik návrhářů, spisovatelů a inženýrů.

Každý má odlišný způsob psaní a tvorby příběhu pro hry, které hodlá vytvořit. Neexistuje žádný správný nebo špatný způsob, jak se vypořádat s příběhem hry, snad kromě tvrzení, že musí existovat, než začnete psát jakýkoliv kód. V následující části se dozvíte, proč je příběh tak důležitý.

Proč záleží na příběhu

Nelze popřít, že v raných dobách videoher se příběhům nepřikládal takový význam jako nyní. Bylo mnohem snazší uvést na trh hru, která nabízela rychlou zábavu, aniž by bylo nutné dostat se hlouběji do jejího smyslu.

To však již zcela jistě neplatí. Lidé, ať už hrají Angry Birds nebo World of Warcraft, očekávají, že každá akce bude mít definovaný smysl. Toto očekávání může být i na úrovni podvědomí, avšak vaše hra musí hráče vtáhnout tak, aby chtěl pokračovat v hraní. Toto vtažení do hry je hybným smyslem příběhu.

Příběh v pozadí vaší hry je důležitý z několika odlišných důvodů. Podívejme se přesně na to, proč byste měli před tím, než začnete psát jakýkoliv kód pro svou hru, věnovat čas vývoji příběhu.

Prvním důvodem, proč je příběh v pozadí vaší hry důležitý, je to, že vám dává šanci hru plně pochopit, od začátku do konce, ještě než začnete programovat. Bez ohledu na to, čím se živíte, ať už jste vývojáři her na plný úvazek nebo to děláte jen jako koníčka, má váš čas určitou cenu.

V případě vývojáře her na plný úvazek bude s každou hodinou strávenou programováním a tvorbou hry spojena určitá finanční hodnota. Vytváříte-li nezávislé hry ve svém volném čase, můžete svůj čas měřit ve věcech, které byste mohli dělat místo toho: rybaření, setkávání s ostatními a tak dále. Ať už se na to podívat jakkoli, váš čas má definovanou, konkrétní cenu a čím více času strávíte programováním vaší hry, tím větší jsou náklady.

Pokud svoji hru plně nepochopíte, ještě než začnete pracovat na zdrojovém kódu, nevyhnutelně narazíte na problémy, které vás mohou přinutit vrátit se zpět kvůli úpravě nebo kompletnímu přepsání kódu, který byl již hotový. To vás bude stát čas, peníze nebo zdravý rozum.

Poznámka: Myšlenka musí být pro plné pochopení kompletní. Je nutné promyslet a pečlivě zvážit každý její aspekt.

Poslední věcí, kterou potřebujete, je nutnost vracet se zpět a měnit kód, který je hotový a možná i otestovaný. Váš kód by měl být pokud možno dostatečně rozšiřitelný, abyste s ním mohli manipulovat bez větší námahy – zvláště pak tehdy, chcete-li později do hry přidávat úrovně nebo nepřátele. Může se však stát, že musíte přeprogramovat něco menšího, jako je jméno postavy nebo prostředí, nebo musíte něco změnit poněkud výrazněji. Můžete si například uvědomit, že jste své hlavní postavě nikdy nedali zbraň nezbytnou pro dokončení hry, protože jste při zahájení její tvorby nevěděli, jak skončí.

Plně rozvinutý příběh pro vaši hru vám dá lineární plán, jehož se můžete držet při psaní kódu. Takovéto rozplánování hry a jejích detailů vám ušetří řadu problémů, které by mohly způsobit přeprogramování již hotových částí hry. Tím se dostáváme k dalšímu důvodu, proč byste měli mít před zahájením programování hotový příběh.

Příběh, na němž je vaše hra založená, bude navíc sloužit jako referenční materiál při psaní kódu. Ať už se potřebujete podívat na správný zápis jména postavy nebo skupiny padouchů nebo si připomenout rozvržení městské ulice, měli byste být schopni tyto informace vytáhnout právě z vašeho příběhu.

Možnost dohledat si podrobnosti v příběhu je zvláště klíčová, bude-li na dané hře spolupracovat více lidí. V příběhu mohou existovat místa, která jste nenapsali. Programujete-li něco, co se odkazuje na jedno z takovýchto míst, představuje plně realizovaný dokument příběhu neocenitelný referenční materiál.

Rozvinutí příběhu do této míry a rozsahu znamená, že se na stejný zdroj může nahlížet více lidí, přičemž všichni získají stejný obrázek toho, co je zapotřebí udělat. Máte-li více lidí spolupracujících v kolaborativním prostředí, je naprosto zásadní, aby se každá osoba pohybovala stejným směrem. Pokud by každý začal hru programovat podle toho, jak si ji sám představuje, pak by každý programoval něco jiného. Dobře napsaný příběh, na který se může podívat každý vývojář pracující na dané hře, pomůže udržet pohyb týmu směrem k témuž cíli.

Jak ale dostat příběh z hlavy a připravit jej tak, aby se na něj mohl někdo podívat, ať už vy sami nebo někdo jiný? Odpověď na tuto otázku přinese následující část.

Zapsání příběhu

Zapsání příběhu není žádná věda. Může být tak propracovaný nebo primitivní, jak to považujete za nutné. Stačí cokoliv od několika stručných vět v zápisníku vedle PC po několik stránek v pěkně naformátovaném dokumentu aplikace Microsoft Word. Smyslem je nesnažit se publikovat příběh jako knihu. Místo toho totiž potřebujete jen dostat příběh z hlavy do čitelného formátu, na který se můžete odkazovat a který by se pokud možno neměl měnit.

Čím déle zůstane příběh ve vaší hlavě, tím více času budete mít na změnu detailů. Pokud totiž v příběhu změníte jakékoli detaily, pak riskujete, že budete muset přepsat kód (nebezpečí takového počínání jsme si již vysvětlili). Proto i tehdy, jste-li sami, věnujete-li se jen příleži-

tostnému vývoji a domníváte se, že není nutné příběh zapsat jen pro vás, znovu se zamyslete. Zapsání příběhu totiž zajistí, že na žádný detail nezapomenete ani jej omylem nezměníte.

Bezpochyby máte v hlavě hru, kterou chcete vyvinout ihned, jakmile si osvojíte dovednosti prezentované v této knize. Možná jste ale dosud ve skutečnosti nezvážili, jak by měl vypadat příběh pro takovou hru. Věnujte nějaké myšlenky tomuto příběhu.

Tip: Věnujte nějaký čas zapsání stručného nástinu vaší hry, máte-li nějakou ve své mysli. Jakmile skončíte, porovnejte ji se smyšleným příběhem, který následuje.

Podívejme se na krátký příklad příběhu, který lze použít pro vývoj hry.

John Black ukradne z lokálního skladu obstavených věcí poněkud rychlé, avšak odolné auto. Padouši se na něj rychle pověsí. Nyní se musí dostat pryč ze Zlosynova i s penězi, vyhnout se policii a přemoci gang, jemuž peníze ukradl. Auta gangu jsou rychlejší, John ale naštěstí může střílet a jezdit současně. Snad budou světla v konspiračním bytě i nadále rozsvícená.

V tomto stručném příběhu, který sice obsahuje jen několik málo podrobností, je i tak dostatek informací k tomu, aby jeden příležitostný vývojář začal pracovat na docela jednoduché hře. Co se z tohoto odstavce dá vyčíst?

První představou, která z tohoto stručného příběhu vytane na mysli, by byla shora viděná hra, v níž se v arkádovém stylu jezdí autem (např. jako Spy Hunter). Řidič nebo auto by mohli mít zbraň pro střílení na nepřátelská vozidla. Hra by mohla skončit, když se hráč dostane na okraj města nebo do konspiračního bytu nebo třeba do nějaké garáže.

Tento krátký příběh má dokonce dostatek detailů k tomu, aby bylo hraní hry o něco zábavnější. Hlavní postava má jméno John Black. Je nutné dávat si pozor na dvě skupiny nepřátel: policie a gang. Prostředí tvoří ulice Zlosynova a většina nepřátelských vozidel jezdí rychleji než auto hlavní postavy. Je zde bezpochyby dostatek dobrého materiálu pro vytvoření rychlé, příležitostné hry.

Metaforická kolečka ve vaší hlavě by již měla chrlit nápady pro tuto hru. V tomto jediném krátkém odstavci je popsáno značné množství dobré akce v arkádovém stylu. Dokážete-li popsat hru, kterou chcete vytvořit, v krátkém odstavci, jako je tento, potom jste jako samostatný, příležitostný vývojář na dobré cestě k vytvoření docela zábavné hry.

Jestliže jeden krátký odstavec může mít dostatek podrobností pro docela přesvědčivou příležitostnou hru, představte si, co by mohl nabídnout delší příběh. Čím více podrobností můžete do svého příběhu nyní dát, tím snazší bude vaše práce při programování a o to lepší hru vytvoříte.

Věnujte nějaký čas navíc svému příběhu, abyste se náležitě vypořádali s nezbytnými detaily. Krátký odstavec, jako je výše uvedený, je pro další práci zajisté dostatečný, nicméně více podrobností by vám při programování rozhodně pomohlo. Zde je seznam otázek, které byste si měli položit po přečtení tohoto příběhu:

- Jaký druh auta John ukradl a řídí?
- Proč ukradl peníze?
- Jaký druh zbraně vlastní?
- Jaké druhy zbraní jsou na autě, jsou-li na něm nějaké?
- Je Zlosynov městské, nebo venkovské prostředí?
- Je na konci souboj s bossem?
- Jak se počítají body, počítají-li se vůbec?

Vrátíme-li se zpět a odpovíme-li na tyto otázky, může příběh vypadat takto.

John Black, obviněný za zločin, který nespáchal, využije příležitosti, aby se dostal ke gangu, který ho do této situace dostal. Zachytí 8 milionů, které byly na cestě k Velkému šéfovi, což je vůdce Padouchů. Ví, že pěšky, se nikam nedostane, a tak z lokálního skladu obstavených věcí ukradne poněkud rychlý, avšak odolný černý sedan.

Toto auto má vše: dvojité kulomety, olejovou skvrnu a minirakety.

Padouši se na něj rychle pověsí. Nyní se musí dostat pryč z přelidněných městských ulic Zlosynova i s penězi. Ulice lemují zchátralé a prkny zabedněné budovy. Čím rychleji John pojede, tím má větší šanci, že vyvázne živý. Stačí mu vyhnout se policii a přemoci gang, jemuž ukradl peníze.

Auta gangu jsou možná rychlejší, John ale naštěstí může střílet a jezdit současně. Tyto dovednosti bude potřebovat, jakmile jej na okraji města dostihne Velký šéf ve svém opětovně zprovozněném tanku americké armády.

Dokáže-li John Velkého šéfa porazit, ponechá si peníze, pokud ale na cestě dostane zásah, budou kumpáni Velkého šéfa brát vše, co mohou, dokud John nebude mít vůbec nic. John tedy musí být opatrný, protože kumpáni Velkého šéfa na něj budou přicházet se vším, co mají: sportovní auta, motorky, kulomety, a dokonce helikoptéry.

Snad budou světla v konspiračním bytě i nadále rozsvícená.

Podívejme se nyní znovu na tento příběh. Máte toho k dispozici o poznání více a je jasné, že podrobnější příběh by mohl být předpokladem zajímavější hratelnost. Kdokoli, kdo bude tuto hru programovat, by nyní byl schopen rozlišit následující detaily hry:

- Autem hlavní postavy je černý sedan.
- Toto auto má jako zbraně dva kulomety, rakety a olejové skvrny.
- Prostředím je zalidněná městská ulice lemovaná zchátralými, prkny zabedněnými budovami.
- Hráč začíná s 8 000 000 Kč (8 000 000 body).
- Hráč ztratí peníze (body), pokud ho nepřítel chytí nebo zasáhne.
- Nepřátelskými vozidly budou sportovní auta, motorky a helikoptéry.
- Na konci města je velká bitva s tankem.
- Hra skončí, když hráči dojdou peníze (body).

Jak můžete vidět, je nyní mnohem jasnější, co se je nutné udělat. Ve způsobu hraní této hry by tedy neměly být žádné nejasnosti. Proto je důležité dát do příběhu, na němž bude vaše hra založena, co možná nejvíce detailů. Zcela jistě budete těžit ze všeho toho času, který jste si vyhradili ještě před zahájením programování.

Nyní jsme si tedy vysvětlili některé z důvodů, proč je dobré vyvíjet hry na platformě Android, a přezkoumali jsme filozofii v pozadí tvorby hry. Můžeme se tedy podívat na přístup, který budeme používat, a na nástroje, které budete potřebovat, abyste se stali úspěšnými vývojáři her pro platformu Android. Poslouží to jako základ pro všechny projekty ve zbývajících kapitolách.

Co vás čeká

V této knize se naučíte vyvíjet dvojrozměrné i trojrozměrné hry. Začnete-li od začátku této knihy a projdete si základní příklady, při čemž si sestavíte vzorovou dvojrozměrnou hru, mělo by pro vás být pochopení kapitol o trojrozměrné grafice snazší. A naopak, pokud se pokusíte přejít přímo ke kapitolám o vývoji trojrozměrných her a nejste-li ostřílenými vývojáři v OpenGL, může být pro vás obtížnější porozumět tomu, co se vlastně děje.

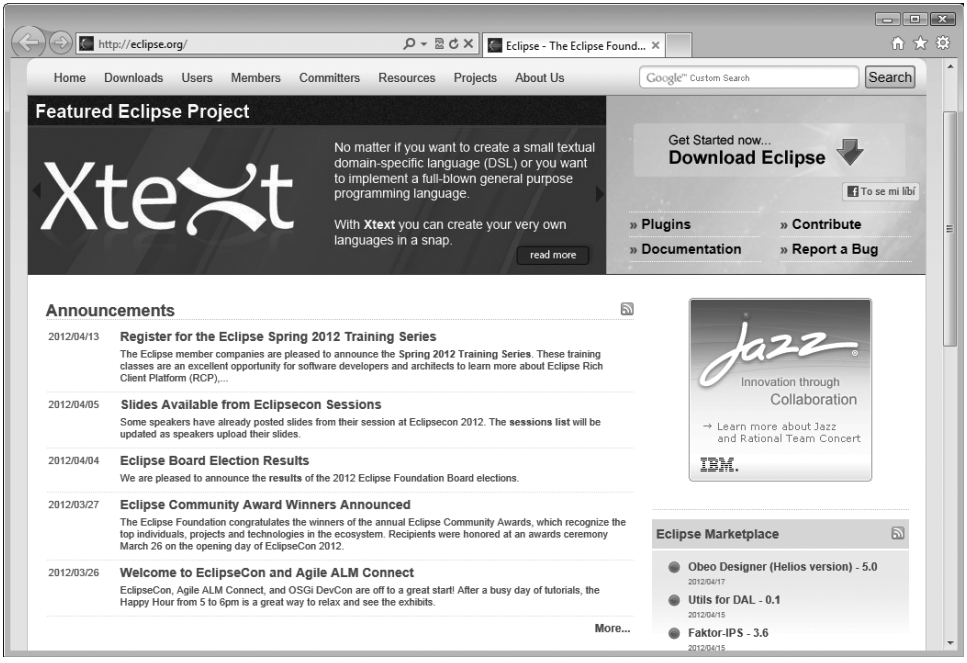
Stejně jako v jakékoli lekci, hodině či výuce bude pro vás nejlepší postupovat v této knize od začátku do konce. Pokud nicméně zjistíte, že některé z prvních příkladů jsou vzhledem k vaší úrovni zkušeností příliš jednoduché, klidně přejděte k dalším kapitolám.

Příprava vývojových nástrojů pro platformu Android

V tomto okamžiku se již nejspíše nemůžete dočkat, až se ponoříte do vývoje své hry pro platformu Android. Jaké nástroje tedy potřebujete pro zahájení své cesty?

Ze všeho nejdříve budete potřebovat kvalitní, plně vybavené integrované vývojové prostředí (Integrated Development Environment – IDE). Veškerý kód pro platformu Android píšu v prostředí Eclipse Indigo (je k dispozici bezplatně ke stažení). Všechny příklady v této knize budou prezentovány pomocí vývojového prostředí Eclipse. Přestože pro psaní kódu pro platformu Android můžete používat téměř jakékoli vývojové prostředí pro Javu, osobně preferuji vývojové prostředí Eclipse, a to kvůli kvalitním zásuvným modulům, které pevně integrují spoustu jednotvárných ručních operací pro kompilování a ladění kódu pro systém Android.

Nemáte-li vývojové prostředí Eclipse a chcete jej vyzkoušet, můžete si jej bezplatně stáhnout z webu Eclipse.org (*http://eclipse.org*), který je zachycený na obrázku 1.2:



Obrázek 1.2: Web Eclipse.org

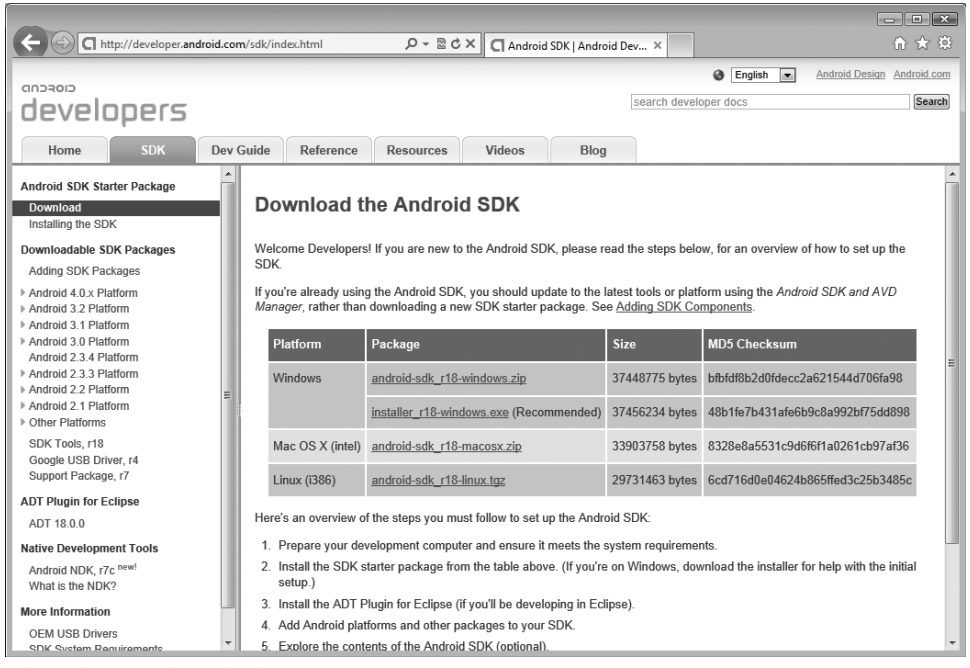
Nebudeme si vysvětlovat proces stažení a přípravy vývojového prostředí Eclipse. V případě, že potřebujete pomoci s instalací svého vývojového prostředí, máte k dispozici spoustu zdrojů, včetně těch na samotném webu Eclipse a v diskuzním fóru na webu Android Developers.

Tip: Pokud jste nikdy Eclipse nebo podobné vývojové prostředí neinstalovali, postupujte pozorně podle pokynů pro instalaci. Poslední, co potřebujete, je nesprávně nainstalované vývojové prostředí bránící vám v psaní skvělých her.

Dále budete potřebovat nejnovější sadu Android SDK. Stejně jako u všech sad Android SDK najdete nejnovější na webu Android Developers (*http://developer.android.com/sdk/index.html*), který je zachycený na obrázku 1.3.

Stejně jako u vývojového prostředí máte i zde k dispozici řadu zdrojů, které vám pomohou se stažením a instalací sady SDK (a odpovídajících komponent Javy, které mohou být potřeba), budete-li to potřebovat.

A nakonec byste měli mít alespoň základní povědomí o vývoji softwaru, zvláště pak v jazyce Java. Přestože si spoustu koncepcí a postupů používaných při tvorbě kódu pro tuto knihu vysvětlíme, není možné vysvětlovat principy vývoje softwaru na základnější úrovni. Stručně řečeno, samotné vysvětlení v této knize by mělo stačit, abyste dokázali pracovat s kódem v této knize, zatímco pokročilejší vývojáři v Javě budou schopni sami vzít příklady a dále je rozvádět.



Obrázek 1.3: Web Android Developers

Instalace knihovny OpenGL ES

Pravděpodobně jedním z nejdůležitějších prvků, které budete používat, je OpenGL ES, což je knihovna, kterou vyvinula společnost Silicon Graphics v roce 1992 pro použití v aplikacích typu CAD (Computer-Aided Design – počítačem podporované projektování). Od té doby se o ni stará konsorcium Khronos Group a najdete ji na většině platforem. Jedná se o velice výkonný a neocenitelný nástroj pro kohokoli, kdo chce vytvářet hry.

Poznámka: Stojí za zmínku, že verze knihovny OpenGL, která je poskytována a podporována platformou Android, se ve skutečnosti označuje jako OpenGL ES (OpenGL for Embedded Systems – OpenGL pro vestavěné systémy). Knihovna OpenGL ES není tak bohatě vybavená jako standardní knihovna OpenGL. I tak se ovšem jedná o výjimečný nástroj pro vývoj softwaru na platformě Android. V rámci této knihy se na funkce a knihovny systému OpenGL ES budeme pro jednoduchost odkazovat jako na OpenGL. Mějte tedy na paměti, že ve skutečnosti používáme OpenGL ES.

Většina lidí si knihovnu OpenGL ze všeho nejdříve spojí s trojrozměrnou grafikou. Faktem je, že knihovna OpenGL je velmi dobrá při vykreslování trojrozměrné grafiky a lze ji použít pro tvorbu přesvědčivých trojrozměrných her. Knihovna OpenGL je ovšem velice dobrá také při vykreslování dvojrozměrné grafiky. Ve skutečnosti dokáže vykreslovat a manipulovat s dvojrozměrnou grafikou mnohem rychleji než nativní volání systému Android. Nativní volání

systému Android jsou pro většinu vývojářů aplikací dostatečná, avšak pro hry, které vyžadují maximální možnou míru optimalizace, je OpenGL ta nejlepší volba.

Ti z vás, kteří nemají mnoho zkušeností s knihovnou OpenGL, se nemusejí ničeho obávat. V kapitolách, kde budeme pracovat s větším množstvím vykreslování grafiky pomocí knihovny OpenGL, si v maximální míře vysvětlíme každé volání, které je potřeba provést. Pokud vám tedy následující kód knihovny OpenGL připomíná cizí jazyk, ničeho se nebojte; na konci knihy budete vědět přesně, co říká:

```
gl.glViewport(0, 0, width, height);
gl.glMatrixMode(GL10.GL_PROJECTION);
gl.glLoadIdentity();
GLU.gluOrtho2D(gl, 0.0f, 0.0f, (float)width,(float)height);
```

Knihovna OpenGL je skvělý nástroj, který budete používat a učit se v této knize, protože jde o knihovnu pracující na více platformách. To znamená, že knihovnu OpenGL a znalosti pro její používání, které se zde naučíte, můžete používat v mnoha prostředích a oborech. Ve všech systémech od iPad a iPhone až po Microsoft Windows a Linux můžete používat mnoho stejných volání knihovny OpenGL.

Používání knihovny OpenGL pro vaši dvojrozměrnou herní grafiku v rámci této knihy má ještě jednu výhodu.

Knihovna OpenGL se prakticky nestará, zda pracujete s dvojrozměrnou nebo trojrozměrnou grafikou. V obou případech se často používají stejná volání. Jediný rozdíl je v tom, jak OpenGL vykresluje polygony, nastane-li čas kreslit na obrazovku. To znamená, že váš přechod od dvojrozměrné k trojrozměrné grafice bude při použití knihovny OpenGL mnohem plynulejší a snazší. Mějte na paměti, že tato kniha nemá být kompletní příručkou knihovny OpenGL ES ani vám neukáže komplexní maticovou matematiku a jiné optimalizační triky, které byte jinak používali v profesionálním herním studiu. Pravdou je, že pro příležitostné vývojáře her jsou metody knihovny OpenGL poskytované pro takové věci, jako je maticová matematika, i přes určitou zátěž navíc dostatečně dobré pro výuku lekcí v této knize.

V této knize budete používat verzi OpenGL ES 1.0. Pro uživatele platformy Android jsou k dispozici tři verze knihovny OpenGL: OpenGL ES 1.0, 1.1 a 2.0. Proč používat verzi 1.0? Za prvé, o verzi OpenGL ES 1.0 je na Internetu již spousta odborného materiálu. Když se tedy dostanete do problému nebo chcete rozšířit své znalosti, máte k dispozici spoustu míst, kam se můžete obrátit s prosbou o pomoc. Za druhé, tato verze je odzkoušená a otestovaná. Jakožto nejstarší z platforem OpenGL ES bude dostupná na většině zařízení a bude značně otestovaná. A nakonec, je velice jednoduché s ní začít pracovat a naučit se ji. Kromě toho přechod na verzi 1.1 a možná i 2.0 bude po osvojení verze 1.0 mnohem snazší.

Vyberte si verzi platformy Android

Jeden z půvabů vývoje pro platformu Android spočívá v tom, že se široce používá v mnoha různých zařízeních, jako jsou mobilní telefony, tablety a přehrávače MP3. Hry, které vyvinete, tak mají šanci běžet na tuctu různých mobilních telefonů a tabletů, a dokonce i na elektronických čtečkách knih. V celém spektru nejrůznějších bezdrátových nosičů a nejrůznějších výrobců se hardware, na němž vaše hra může běžet, docela liší.

Na tuto všudypřítomnost lze naneštěstí pohlížet také jako na překážku, kterou musíte proskočit. V kterémkoli okamžiku by totiž mohlo existovat až 12 odlišných verzí platformy Android běžících na tuctu odlišných kousků hardwaru. Na nejnovějších tabletech a telefonech poběží verze 2.3.3, 3.0, 3.1, nebo 4.0 – což jsou nejnovější verze, které běží na nejvýkonnějších zařízeních. Jedná se tedy o verze, na které se v této knize zaměříme.

Poznámka: Nemáte-li zařízení se systémem Android, na němž byste mohli testovat, můžete používat emulátor pro PC. Vřele ovšem doporučuji zkusit použít k otestování kódu skutečný telefon nebo tablet se systémem Android. Při spouštění kódu v emulátoru a v telefonu či tabletu se totiž mohou objevit drobné odchylky.

Nejdůležitější je, abyste se při procházení procesem tvorby her dobře bavili. Hry jsou koneckonců o zábavě, takže byste se při jejich vytváření měli skvěle bavit!

Shrnutí

V této kapitole jsme probrali, co byste měli očekávat, že od této knihy získáte. Dozvěděli jste se o významu příběhu pro tvorbu hry a také to, jak vám může držení se takového příběhu pomoci při vytváření lepšího kódu. Dále jste se dozvěděli o procesu tvorby her na platformě Android, o verzích systému Android a o vývojovém prostředí pro platformu Android. Nakonec jste objevili klíč k tvorbě her na platformě Android, kterým je knihovna OpenGL ES, a podívali jste se na několik relevantních detailů o jednotlivých vydáních systému Android.

KAPITOLA 2

Star Fighter: Dvojrozměrná střílečka

V této kapitole:
- Příběh v pozadí hry Star Fighter
- Co dělá hru hrou?
- Porozumění hernímu enginu
- Pochopení kódu specifického pro hru
- Průzkum herního enginu pro hru Star Fighter
- Vytvoření projektu Star Fighter
- Shrnutí

Hra, kterou budete v průběhu práce s touto knihou vytvářet, nese název Start Fighter. Jedná se o dvojrozměrnou, shora viděnou posouvající se střílečku. I když akce je poměrně omezená, příběh je překvapivě detailní. V této kapitole uvidíte, jak hra bude vypadat a jaký je příběh v jejím pozadí. Kromě toho se dozvíte o nejrůznějších částech herního enginu a o tom, co vlastně herní engine dělá.

Příběh v pozadí hry Star Fighter

Příběh pro hru Start Fighter vypadá následovně (v průběhu knihy se na něj budeme pravidelně odkazovat):

Kapitán John Straka je prošedivělý galaktický válečný veterán. Probojoval si cestu z každé bitvy, do níž byla Planetární koalice zapojená. Na cestě zpět na Zemi, kde se z letité služby chystá odejít do důchodu na malé tiché farmě v Beskydech, se ocitne uprostřed překvapivé nepřátelské invazní flotily.

Kapitán Straka se připravuje na bitvu. To ale není běžná invazní flotila Kordarkianů – něco je jinak.

Kapitán Straka nakopne motory na své AF-718 a nastaví zbraně na automatickou střelbu. AF-718 je naštěstí lehká a hbitá. Pokud se dokáže vyhnout nepřátelským střelám a občasné kolizi, měly by automatické kanóny udělat s menšími stíhačkami Kordarkianů krátký proces.

Jenže co má AF-718 v mrštnosti a automatické střelbě, to postrádá ve štítech. Kapitán Straka udělá nejlépe, když se zcela vyhne nepřátelské lodi. Utrpí-li nějaké poškození, bude po třech zásazích po něm. AF-718 nezvládne mnoho přímých zásahů bez kvalitních štítů. Co se přímé kolize s nepřítelem týče, je to naneštěstí tak, že stačí jednou, a je po Kapitánovi.

Zatímco se Kapitán Straka snaží proletět skrze další a další vlny nepřátelských lodí, může mít štěstí a narazit na nějaké trosky jiných zničených AF-718 – pozůstatky předchozí skupiny překvapené invazní flotilou. Pokud jej při tom nezničí, může Kapitán Straka nalézt pro tyto části nějaké použití.

AF-718 má velmi užitečnou vlastnost, která bude pomáhat Kapitánu Strakovi v boji. Poslední verze AF-718, zvláště ty, které byly vyrobené pro poslední vzpouru na planetě Centelum, jsou vybavené samoopravným režimem. Pokud se Kapitán Straka dostane do problémů a začne ztrácet štíty nebo pokud zjistí, že potřebuje silnější střely, stačí mu jen přiletět k nějakým částem AF-718 volně poletujícím v bitevní vřavě. Měl by být schopen získat cokoliv, od silnějších štítů, které by zdvojnásobily nebo ztrojnásobily množství poškození, které jeho loď vydrží, až po silnější zbraně, které jsou rychlejší a pro zničení nepřítele stačí méně zásahů.

Kapitán Straka a jeho AF-718 nejsou jediný, kdo má esa v rukávu. Invazní flotilu Kordarkianů tvoří tři odlišné lodě:

- *Kordarkianský Průzkumník*
- *Kordarkianský Stíhač*
- *Larashská Válečná loď*

Kordarkianští Průzkumníci jsou nejpočetnější ze všech lodí v invazní flotile. Jsou rychlí – stejně rychlí jako AF-718 Kapitána Straky. Průzkumník létá rychlým, avšak předvídatelným způsobem. Díky tomu by se měli dát snadněji rozpoznat a mělo by jít lépe předvídat jejich pohyb. Pro Straku je dobré, že Průzkumník má díky přesměrování veškeré energie do motorů jen velice slabé štíty. Jedna dobrá trefa od AF-718 by měla stačit pro sestřelení kordarkianského Průzkumníka. V přední části lodi mají přimontovaný kanón, který střílí pomalé dávky o jedné střele. Rychlá střelba a manévrování by měly dostat AF-718 mimo nebezpeční a dát Kapitánu Strakovi prostředek, který potřebuje pro zničení Průzkumníka.

Kordarkianští Stíhači jsou na druhou stranu velmi přímí a opatrní nepřátelé. Poletí pomalu, ale přímo na AF-718 Kapitána Straky. Stíhač je bezobslužný a používá se jako počítačem naváděné beranidlo. Jsou naprogramovaní tak, aby sestřelili všechny nepřátele, jakmile mohou zaměřit jejich pozici.

Stíhač byl postavený tak, aby proniknul silným trupem masivních bitevních křižníků Planetární koalice. Jejich štíty jsou proto velmi silné. Pro zastavení této lodě jsou zapotřebí až čtyři přímé zásahy od nejlepší zbraně AF-718. Nejlepším útokem Kapitána Straky je v tomto případě obrana. Kordarkianský Stíhač si svůj cíl zaměří velice brzy a je naprogramován tak, aby po zaměření svoji trajektorii již neměnil. Je-li Kapitán Straka v otevřené oblasti, neměl by mít problém provést úhybný manévr ještě před tím, než dojde ke kontaktu se Stíhačem. Má-li štěstí, může jednoho či dva sestřelit svými kanóny, k tomu je však zapotřebí určité zručnosti.

Posledním typem nepřítele, jemuž bude Kapitán Straka čelit, je larashská Válečná loď.

Přítomnost larashských Válečných lodí je to, čím se tato invazní flotila liší od ostatních, které Kapitán Straka dosud viděl. Larashské Válečné lodě jsou silné jako kordarkianské Stíhače, mají však vpřed směřující zbraně jako Průzkumníci. Mohou manévrovat náhodným způsobem a pro Kapitána Straku by měly představovat jeho největší výzvu. Naštěstí pro něj je těchto Válečných lodí relativně málo, což mu dává čas, aby se mezi jejich přílety dal zase dohromady.

Počítač AF-718 bude sledovat, kolik lodí se nachází v invazní vlně. Bude Kapitána Straku informovat, jakmile eliminuje všechny potenciální nepřátele. Tyto statistické údaje se budou odesílat do předsunutého velení na Zemi, kde budou mít přehled, jak si proti invazi vede.

Pomozte Kapitánu Strakovi eliminovat co možná nejvíce vln invazní flotily a vrátit se v pořádku na Zemi.

Toto je tedy příběh, z něhož budete vycházet při programování hry Star Fighter. Jaké údaje o hře z něj můžete vytáhnout? Zapišme si je do seznamu, podobně jako v případě vzorového příběhu v kapitole 1:

- Hlavní postava Kapitán John Straka bude pilotovat vesmírnou loď AF-718.
- Hráč nebude muset řídit žádný střílecí mechanizmus, protože loď má automatickou střelbu.
- Hráč může zlepšovat loď získáváním dalších štítů a zbraní.
- Pokud hráč obdrží tři zásahy od nepřátelského kanónu bez opravy, hra skončí.
- Pokud se hráč srazí s nepřátelskou lodí, hra skončí.
- Existují tři odlišné typy nepřátelských lodí:
  - Průzkumníci se pohybují rychle předvídatelným způsobem a střílejí jediným kanónem.
  - Stíhači nemají kanóny, ale vydrží čtyři přímé zásahy od hráče. Jakmile se zaměří na hráčovu pozici, tak už nemohou změnit svoji trajektorii.
  - Válečné lodě mají kanóny a vydrží čtyři přímé zásahy od hráče. Pohybují se náhodným způsobem.
- Hra bude sledovat počet nepřátel v každé vlně. Pokaždé, když hráč zničí jednoho z nepřátel, sníží se o jedničku počítadlo, dokud vlna neskončí.
- Body se budou nahrávat do centrální oblasti.

Zní to jako docela zábavná, vzrušující a detailní hra. Nejlepší ovšem je, že kód potřebný pro vytvoření této hry nebude příliš komplikovaný, tedy alespoň ne tak komplikovaný, jak byste možná čekali.

V další části se seznámíte s herním enginem pro hru Star Fighter. Naučíte se, k čemuž slouží nejrůznější části herního enginu a co tento engine jako celek dělá pro vaši hru. Nakonec začnete dávat dohromady základní funkčnost enginu a sestavovat svoji hru.

Co dělá hru hrou?

Nyní již tedy víte, o čem hra Star Fighter vlastně má být, a proto se můžeme podívat na nejrůznější části nezbytné pro sestavení hry. Pro vytvoření konečného produktu, který bude hra-

telnou a zábavnou hrou pro platformu Android, musí spousta částí pasovat dohromady velmi těsným a kompaktní způsobem.

Když se zamyslíte nad vším, co musí hra dělat pro poskytnutí skutečně zábavné hratelnosti, začnete si cenit času a úsilí, které je nezbytné pro vytvoření i té nejjednodušší hry. Typická hra bude provádět následující:

- Vykreslování pozadí
- Pohybování pozadím dle potřeby
- Vykreslování libovolného počtu postav
- Vykreslování zbraní, střel a podobných věcí
- Pohybování postav nezávislým způsobem
- Přehrávání zvukových efektů a podbarvující hudby
- Interpretování příkazů vstupního zařízení
- Sledování postav a pozadí pro zajištění, aby se nikdo nepohyboval tam, kde by neměl být schopen se pohybovat
- Vykreslování jakékoli předem definované animace
- Zajištění, aby se věci pohybovaly uvěřitelným způsobem (jako např. odrážení balónu)
- Sledování bodového hodnocení hráče
- Sledování a správa hráčů propojených přes počítačovou síť
- Sestavování systému nabídek, aby si hráč mohl zvolit, zda chce hrát nebo ukončit hru

Možná se nejedná o vyčerpávající seznam, obsahuje však všechny věci, které většina her dělá. Jak taková hra všech těchto věcí v seznamu docílí?

Pro účely této knihy můžeme rozdělit veškerý kód ve hře do dvou kategorií: herní engine a kód specifický pro hru. Vše v předchozím seznamu se řeší v jedné nebo obou kategoriích kódu. Znalost toho, kde se co řeší, je kritická pro osvojení dovedností prezentovaných v této knize. Začněme zkoumat tyto dvě kategorie kódu pohledem na herní engine.

Porozumění hernímu enginu

V srdci každé videohry je herní engine (motor hry). Jak již jeho název napovídá, herní engine je kód, který pohání hru. Každá hra, bez ohledu na svůj typ (RPG, FPS, skákačka nebo RTS), vyžaduje pro svůj běh nějaký engine.

Poznámka: Engine jakékoliv hry je záměrně sestavený tak, aby byl generický a umožňoval své použití ve více situacích a možná i více různých hrách. To je v přímém protikladu ke kódu specifickému pro hru, který, jak již jeho název napovídá, je specifický pouze pro jedinou hru.

Jedním z velice oblíbených herních enginů je Unreal Engine. Unreal Engine, poprvé vyvinutý kolem roku 1998 společností Epic pro její střílečku z vlastního pohledu (FPS) s názvem Unreal, byl nasazený ve stovkách her. Unreal Engine lze snadno adaptovat a pracuje s nejrůznějšími

typy her, ne jen s hrami typu FPS. Díky této generické struktuře a flexibilitě je oblíbený nejen profesionály, ale také příležitostnými vývojáři.

Všeobecně lze říci, že herní engine se stará o veškerou rutinní práci herního kódu. To může znamenat cokoliv od přehrání zvuku po vykreslení grafiky na obrazovku. Zde je krátký seznam funkcí, které provádí typický herní engine:

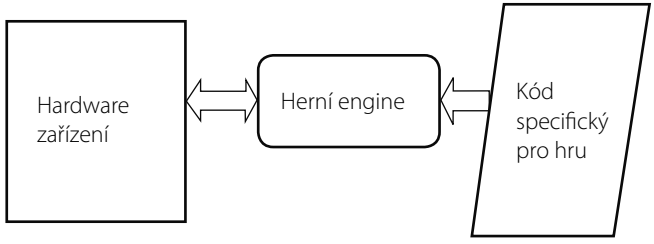
- vykreslování grafiky,
- animace,
- zvuk,
- detekce kolizí,
- umělá inteligence,
- fyzika (bez kolizí),
- správa vláken a paměti,
- komunikace přes síť,
- interpret příkazů (vstup a výstup).

Proč k provádění těchto věcí potřebujeme herní engine? Stručnou odpovědí je, že k tomu, aby hra běžela efektivně, nemůžeme se spoléhat na to, že by tento druh zátěžové práce prováděl operační systém hostitelského systému. Ano, většina operačních systémů má vestavěné prvky, které se dovedou postarat o každou položku z tohoto seznamu. Nicméně tyto systémy operačního systému pro vykreslování, zvuk a správu paměti jsou sestavené tak, aby na nich běžel operační systém a aby se dovedly přizpůsobit jakémukoli počtu nepředvídatelných použití, aniž by se na jakýkoli z nich specializovaly. To je báječné, jestliže píšete podnikové aplikace, ne však píšete-li hry. Hry vyžadují něco s trošku větším výkonem.

K tomu, aby hra běžela plynule a rychle, musí kód obejít zátěž, kterou standardní operační systémy vytvářejí, a běžet přímo nad hardwarem vyžadovaným pro specifický proces. To znamená, že hra by měla komunikovat pro provádění grafických funkcí přímo s grafickým hardwarem, pro přehrávání efektů přímo se zvukovou kartou a tak dále. Pokud byste použili standardní paměťové, grafické a zvukové systémy, které jsou dostupné prostřednictvím většiny operačních systémů, vlákna vaší hry by se chovala stejně jako vlákna ostatních funkcí operačního systému a aplikací běžících na tomto systému. Vaše interní zprávy by se navíc mohly řadit do fronty s jakoukoli další systémovou zprávou. To by způsobilo, že by hra vypadala trhaně a běžela by velice pomalu.

Z tohoto důvodu se herní enginy téměř vždy programují v jazycích nízké úrovně. Jak jsme si řekli již dříve, jazyky nízké úrovně nabízejí přímější cestu k hardwaru systému. Herní engine musí být schopen vzít kód a příkazy z kódu specifického pro hru a předat je přímo určitému hardwaru. Díky tomu může hra běžet mnohem rychleji a s veškerou kontrolou, kterou potřebuje k tomu, aby poskytovala uspokojující zážitek.

Obrázek 2.1 znázorňuje zjednodušenou verzi vztahu mezi herním enginem, hardwarem zařízení a kódem specifickým pro hru.



Obrázek 2.1: Vztah mezi herním enginem, kódem specifickým pro hru a hardwarem zařízení

Herní engine nebude dělat nic speciálně pro určitou hru. To znamená, že herní engine nebude na obrazovku vykreslovat kotě. Herní engine na obrazovku něco vykreslí, protože obstarává vykreslování grafiky, sám od sebe ale nic určitého nevykresluje. Je úkolem kódu specifického pro hru, aby předal enginu kotě, které se má vykreslit, a je úkolem enginu, aby vykreslil cokoliv, co se mu předá.

V herním enginu tedy proto nikdy neuvidíte následující funkci:

```
DrawFunnyKitten();
```

Místo toho byste mohli mít funkci, která vypadá spíše takto:

```
DrawCharacter(funnyKitten);
```

Faktem je, že finální funkce pro vykreslování grafiky, které v této knize vytvoříte, budou vyžadovat více parametrů než jen objekt obrázku, který je zapotřebí vykreslit. Podstatné je pochopit, že engine je velice obecný, zatímco kód specifický pro hru nikoliv.

Nyní již tedy máte dobrý přehled o tom, co engine dělá. Podíváme se tedy na kód specifický pro hru, abyste tak měli kompletní představu o tom, z čeho všeho se hra skládá.

Pochopení kódu specifického pro hru

Prozkoumejme nyní roli kódu specifického pro hru. Jak jsme si řekli již dříve, kód specifický pro hru je kód, který běží pouze v jediné hře, na rozdíl od herního enginu, který lze sdílet a adaptovat pro různé hry.

Poznámka: Při vytváření malé, příležitostné hry (např. ty, které vytváříme v této knize) mohou být herní engine a kód specifický pro hru tak těsně propojené, že může být obtížné je rozlišit. Přesto je velice důležité pochopit koncepční rozdíl mezi oběma částmi.

Kód specifický pro hru se skládá z veškerého kódu, který tvoří postavy ve vaší hře (A-718, Průzkumník, Stíhač atd.), zatímco herní engine tyto postavy jen vykresluje. Kód specifický pro hru ví, že hlavní postava vystřelila z kanónu, a ne raketu, zatímco herní engine danou věc vykreslí. Kód specifický pro hru je kód, který zničí hlavní postavu, srazí-li se s Průzkumníkem, ne však, narazí-li na vylepšení. Herní engine totiž jen testuje kolizi dvou objektů na obrazovce.

Kolize A-718 a Průzkumníka by ve zjednodušeném úryvku kódu mohla vypadat třeba takto:

```
GameCharacter goodGuy;
GameCharacter scout;
GameCharacter arrayOfScouts[] = new GameCharacter[1];

arrayOfScouts[0] = scout;

// Přesuň postavy
Move(goodGuy);
Move(arrayOfScouts);

// Otestuj kolizi
if (TestForCollision(goodGuy, arrayOfScouts))
{
  Destroy(goodGuy);
}
```

Přestože jde pouze o zjednodušenou verzi toho, jak může vypadat část herní rutiny, je z ní patrné, že jsme vytvořili A-718 a Průzkumníka, přesunuli jsme je po obrazovce a otestovali, zda mezi nimi došlo ke kolizi. Pokud ano, objekt `goodGuy` je zničen.

V tomto příkladu jsou objekty `goodGuy` a `arrayOfScouts` a funkce `Destroy()` náleží do kódu specifického pro hru, zatímco funkce `Move()` a `TestForCollision()` jsou částí herního enginu. Na tomto krátkém příkladu lze snadno vidět, že místo `goodGuy` a `arrayOfScouts` byste mohli použít jakékoli jiné postavy v téměř jakékoliv jiné hře a přitom by funkce `Move()` a `TestForCollision()` i nadále pracovaly tak, jak mají. Z toho je patrné, že objekty `goodGuy` a `arrayOfScout` jsou specifické pro hru a nejsou součástí enginu a že funkce enginu `Move()` a `TestForCollision()` fungují v libovolné hře.

U většího projektu, jako je třeba hra, na níž pracují desítky nebo stovky lidí, se nejdříve vyvine engine a potom se vytvoří kód specifický pro hru tak, aby pracoval s tímto enginem. V případě malých příležitostných her, jako jsou ty, které se nacházejí v této knize, je možné současně vyvíjet herní engine a kód specifický pro hru. Díky tomu získáte jedinečnou šanci sledovat vztah mezi dvěma bloky kódu při jejich vytváření.

V průběhu čtení této knihy se naučíte, že některé funkce herního enginu pro menší hry mohou být téměř neodlišitelné od kódu specifického pro hru. U menších her se nemusíte příliš starat o hranici mezi enginem a kódem specifickým pro hru, pakliže hra funguje tak, jak chcete. Na druhou stranu byste se měli snažit udržovat hranici mezi oběma částmi tak zřetelnou, jak je to jen možné, abyste tak mohli lépe podporovat opakovatelnou použitelnost vašeho vlastního kódu a udržovat své vývojářské dovednosti na maximální úrovni. Jinými slovy: snažte se vyhnout línému kódu a praktikám spojeným s líným kódem.

V dřívější části této kapitoly jste viděli seznam prvků, které tvoří téměř jakoukoli hru. Nyní se na tento seznam podíváme znovu s tím, že určíme, o které z prvků se stará herní engine a o které se stará kód specifický pro hru (viz tabulka 2.1).

Tabulka 2.1: Herní prvky

Herní prvek	Prvek herního enginu	Kód specifický pro hru
Vykreslování pozadí.	vykreslování grafiky	Vytvoření hvězdného pozadí.
Pohybování pozadí.	vykreslování grafiky	Posouvání pozadí shora dolů.
Vykreslování postav.	vykreslování grafiky	Umístění A-718 na obrazovku.
Vykreslování zbraní, střel atd.	vykreslování grafiky	Vykreslení trosek A-718 a střel z kanónu.
Pohybování postav nezávislým způsobem.	vykreslování grafiky a umělá inteligence	Pohyb Stíhače směrem k A-718. Pohyb Průzkumníka pomalým, předvídatelným způsobem.
Přehrávání zvukových efektů a podbarvující hudby.	zvuk	Vytvoření exploze při zásahu nepřítele. Přehrávání podbarvující hudby.
Interpretování příkazů vstupního zařízení.	interpret příkazů	
Sledování postav a pozadí pro zajištění, aby se nikdo nepohyboval tam, kde by neměl být schopen se pohybovat.	detekce kolizí	Pokud A-718 narazí do Průzkumníka, exploduje. Pokud se ale srazí dva Průzkumníci, tak je to v pořádku.
Vykreslování jakékoli předem definované animace.	animace	Když hráč vyhraje, vykreslí se vítězná animace.
Zajištění, aby se věci pohybovaly uvěřitelným způsobem (jako např. odrážení balónu).	fyzika	
Sledování bodového hodnocení hráče.	vykreslování grafiky a správa paměti	Při zničení nepřítele se odečte jednička od celkového počtu zbývajících nepřátel.
Sledování a správa hráčů propojených přes počítačovou síť.	komunikace přes síť	
Sestavování systému nabídek, aby si hráč mohl zvolit, zda chce hrát nebo ukončit hru.	vykreslování grafiky a interpret příkazů	Spuštění nové hry, ukončení hry nebo nahrání bodového hodnocení na server.

Jak je z tabulky 2.1 patrné, i ta nejmenší hra obsahuje spoustu částí. O všechny prvky hry se do jisté míry stará herní engine, při čemž některé se týkají pouze enginu. Z toho byste měli mít lepší představu o významu herního enginu a hranice mezi enginem a kódem specifickým pro hru.

Nyní již tedy víte, co herní engine dělá na obecné rovině. Co ale bude dělat náš herní engine pro hru Star Fighter?

Průzkum herního enginu pro hru Star Fighter

Herní engine pro hru Star Fighter bude malinko odlišný od obecného herního enginu, který můžete používat. Mějte na paměti, že platforma Android je postavená na linuxovém jádře a že vývoj softwaru se zde provádí pomocí mírně upravené verze Javy. To znamená, že Android je sám o sobě dostatečně rychlý k tomu, aby na něm snadno běžely některé příležitostné hry. Toho budeme využívat ve hře Star Fighter, aby naše programování nebylo příliš obtížné.

V této knize nebudeme sestavovat skutečný, nízkoúrovňový herní engine jednoduše proto, že pro hry, které budeme vytvářet, není nutný. Řekněme si to na rovinu: čím více času strávíte psaním své hry, tím méně času si budete užívat její hraní. Android obsahuje systémy, které můžeme využívat, a přestože nemusejí být optimální pro běh špičkových her, snadno se osvojují a krásně se hodí pro ten druh her, který budeme vytvářet.

Herní engine pro hru Star Fighter bude využívat sadu Android SDK (a související balíčky Javy) k provádění následujících činností:

- vykreslování grafiky,
- přehrávání zvuku a efektů,
- interpretování příkazů,
- detekce kolizí,
- obsluha umělé inteligence nepřátel.

Po přečtení dřívější části této kapitoly jste si možná všimli, že v našem herním enginu chybějí některé funkce, jako je fyzika bez kolizí, animace a komunikace po síti/sociální média. Důvodem je to, že hra, kterou sestavujeme, tyto funkce nepotřebuje, a proto je nemusíme vytvářet.

Aby bylo čtení této knihy co nejplynulejší a nejlogičtější, budeme sestavovat engine a kód specifický pro hru současně. Například jak vytvořit vykreslování grafiky se naučíte při vytváření pozadí a postav. Díky tomu budete mít na konci každé kapitoly plně funkční části enginu a kódu specifického pro hru.

Vytvoření projektu Star Fighter

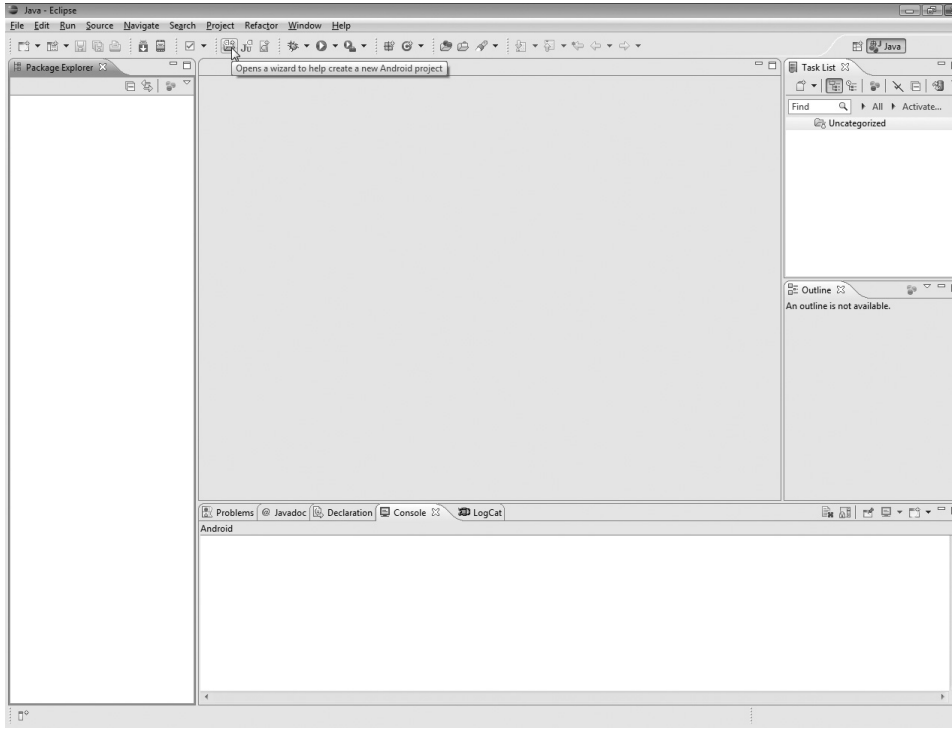
Jako počáteční úkol, v němž si vše připravíte, si v této části vytvoříte projekt, který budete používat pro hru Star Fighter. Tento projekt budeme používat v rámci celé knihy.

Nejdříve otevřete prostředí Eclipse a klepněte na tlačítko nabídky pro otevření nového průvodce projektu pro platformu Android (viz obrázek 2.2).

Jakmile otevřete průvodce, budete moci vytvořit projekt. Máte-li zkušenost s vytvářením projektů pro platformu Android, neměl by to pro vás být žádný problém.

Tip: Používáte-li pro vytváření aplikací pro platformu Android prostředí NetBeans nebo jakékoli jiné vývojové prostředí pro Javu, pak vám tento krátký návod nepomůže. K dispozici máte řadu zdrojů, pomocí nichž můžete vytvořit projekt ve vašem vývojovém prostředí.

Obrázek 2.3 ukazuje volby, které byste měli vybrat při vytváření svého projektu. Projekt nese název **starfighter**. Vzhledem k tomu, že se veškerý kód pro hru Star Fighter bude vytvářet ve stejném projektu, je vhodné projekt pojmenovat **starfighter**. Výsledkem bude navíc to, že se veškerý kód umístí do balíčku `starfighter`.

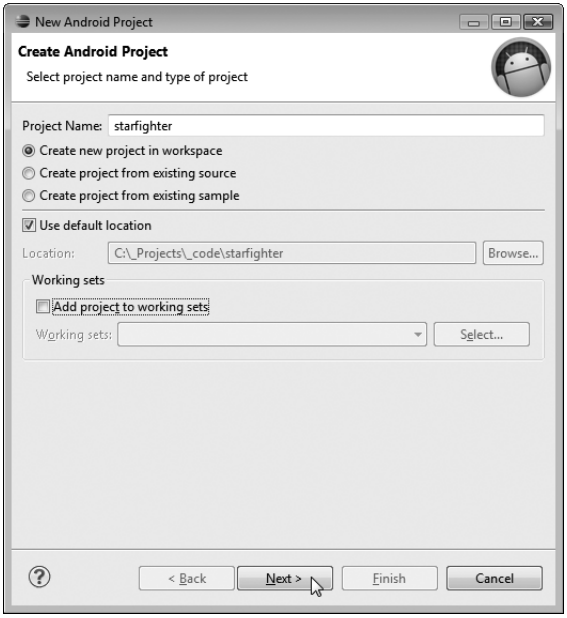


Obrázek 2.2: Spuštění nového průvodce projektu pro platformu Android

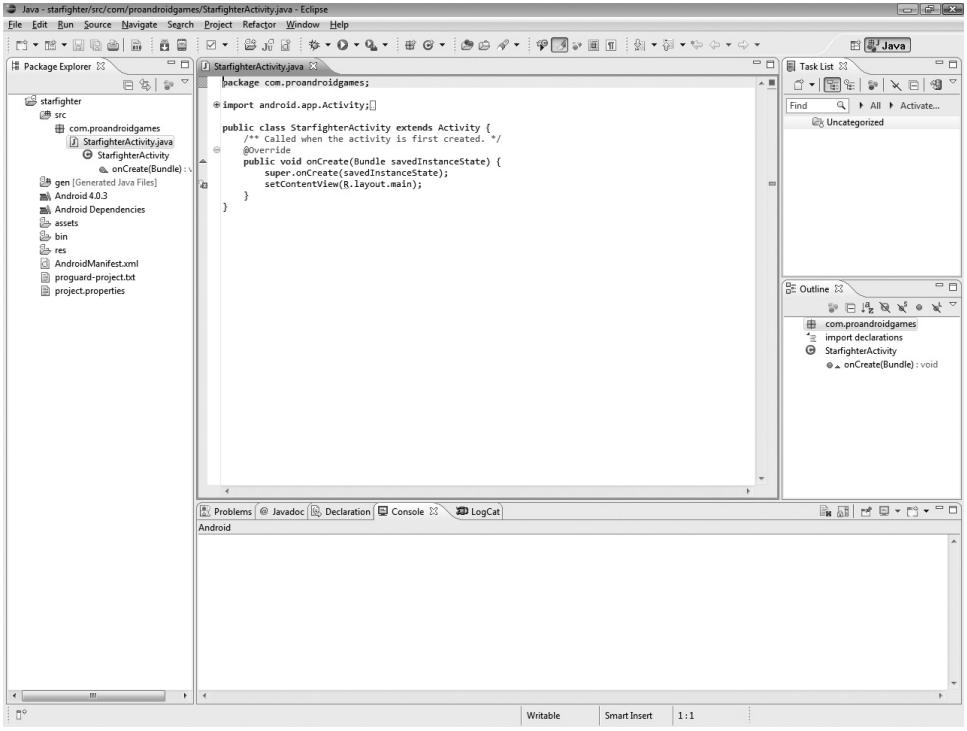
Tip: Pokud jste dosud nikdy nevytvářeli projekt pro Android (nebo pro Javu), existuje několik konvencí pro pojmenování, které byste měli mít na paměti. Při pojmenování vašeho balíčku si jej představte jako adresu URL, avšak zapsanou pozpátku. Měl by tedy začínat označením (např. `com` nebo `net`) a končit názvem vaší entity. V tomto případě budeme používat `com.proandroidgames`.

Nyní můžete vybrat možnost **Create new project in workspace**. Tím zajistíte, že se váš projekt vytvoří ve standardním pracovním prostoru prostředí Eclipse, který jste si pro sebe zvolili při instalaci prostředí Eclipse. Zaškrtávací políčko **Use default location** je ve výchozím stavu zvolené. Pokud nechcete měnit umístění svého pracovního prostoru pro tento projekt, ponechejte jej tak, jak je.

Vaším dalším krokem je výběr nejnovější verze sady Android SDK a klepnutí na tlačítko **Finish**. Obrázek 2.4 ukazuje vytvořený projekt. V následující kapitole jej začneme upravovat.



Obrázek 2.3: Průvodce New Android Project a zvolené možnosti



Obrázek 2.4: Projekt je náležitě připravený

Shrnutí

V této kapitole jste se dozvěděli o příběhu v pozadí hry Star Fighter. Prozkoumali jste nejen různé části generického herního enginu, ale také ty, které budou začleněné do herního enginu hry Star Fighter. Nakonec jste vytvořili projekt, který bude uchovávat kód pro vaši hru.

V následujících pěti kapitolách budete dávat dohromady kód, který bude tvořit hru Star Fighter. Začnete tak budovat svoji sadu dovedností jako příležitostný vývojář her a naučíte se více o platformě Android.

KAPITOLA 3

Stiskněte Start: Tvorba nabídky

V této kapitole:

- Sestavení úvodní obrazovky
- Vytvoření aktivity
- Vytvoření vašeho obrázku pro úvodní obrazovku
- Práce se souborem R.java
- Vytvoření souboru rozvržení
- Vytvoření efektů prolnutí
- Práce s vlákny ve vaší hře
- Tvorba hlavní nabídky
- Přidání hudby
- Přehrávání hudby
- Shrnutí

V této kapitole začnete s vývojem dvojrozměrné arkádové střílečky Star Fighter. Vytvoříte první řádky kódu ve vašem enginu a vyvinete první dvě obrazovky, které uživatel ve vaší hře uvidí: úvodní obrazovku hry a herní nabídku se dvěma herními volbami. V rámci této kapitole se naučíte několik nepostradatelných dovedností z oblasti vývoje her na platformě Android.

Naučíte se:

- zobrazovat grafiku,
- vytvářet aktivity a záměry,
- vytvářet službu systému Android,
- spouštět a zastavovat vlákna systému Android,
- přehrávat hudební soubory.

Kromě úvodní obrazovky a herní nabídky vytvoříte podbarvující hubu, která bude hrát při zobrazení nabídky.

Je toho opravdu hodně. Začněme tedy první obrazovkou, kterou hráč ve vaší hře uvidí – úvodní obrazovkou.

Sestavení úvodní obrazovky

Úvodní obrazovka je první část hry, kterou hráč uvidí. Můžete si ji představit jako úvodní titulky nebo vizitku vaší hry. Měla by zobrazovat název hry, nějaké obrázky ze hry a třeba i pár informací o tom, kdo hru vytvořil. Úvodní obrazovka pro hru Star Fighter je zachycená na obrázku 3.1.

Obrázek 3.1: Úvodní obrazovka hry Star Fighter

Pro hry, které jsou sestavené více lidmi v několika vývojových studiích, se může před začátkem hry objevit více než jedna úvodní obrazovku. To není výjimečné, neboť každé vývojové studio, distributor i producent by mohli mít svoji vlastní úvodní obrazovku, kterou by chtěli zobrazit před začátkem hry. Pro naši hru nicméně vytvoříme jednu úvodní obrazovku, protože budete jediným vývojářem.

Při hraní jakékoli typické hry uvidíte, že úvodní obrazovka většinou automaticky přejde do hlavní nabídky hry. Ve hře Star Fighter vytvoříte úvodní obrazovku, která se plynule objeví a plynule přejde do hlavní nabídky. Pro vytvoření úvodní obrazovky tedy bude nutné navíc vytvořit aktivitu, která bude uchovávat hlavní nabídku, abyste tak mohli správně připravit efekt prolnutí úvodní obrazovky bez jakýchkoli chyb.

Vytvoření aktivity

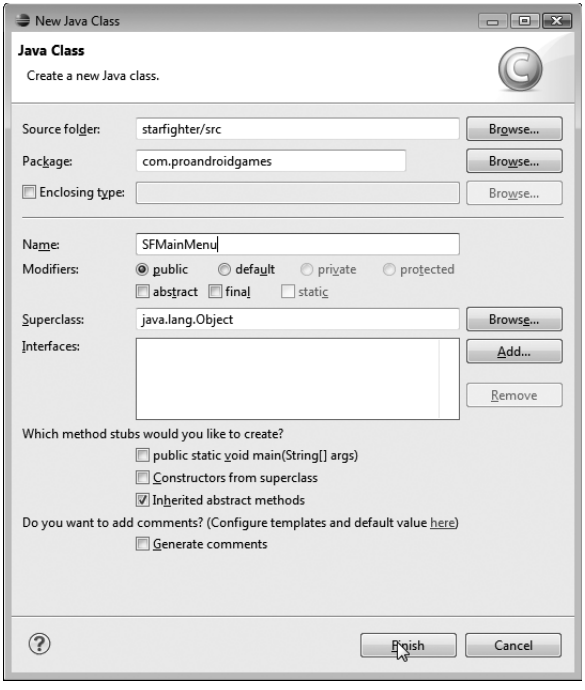
Ze všeho nejdříve otevřete projekt pro hru Star Fighter, který jste vytvořili v předchozí kapitole. Pokud jste jej dosud nevytvořili, vraťte se prosím zpět a učiňte tak nyní. Zbývající část kapitoly totiž předpokládá, že pracujete v projektu hry Star Fighter.

Váš projekt by měl v současném stavu obsahovat jednu aktivitu – `StarfighterActivity`. Tuto aktivitu automaticky vytvořil průvodce a jedná se o vstupní bod do projektu. Kdybyste svůj projekt nyní spustili, rozběhla by se aktivita `StarfighterActivity`. Ve skutečnosti ale pro tuto kapitolu potřebujete dvě aktivity: jednu pro úvodní obrazovku a jednu pro hlavní nabídku hry. Pro úvodní obrazovku již tedy aktivitu máte, takže v následující části vytvoříte novou aktivitu pro hlavní nabídku.

Přestože bude aktivita pro hlavní nabídku v současnosti prázdná, umožní vám plně implementovat plynulý přechod úvodní obrazovky, což je úkol, do něhož se vzápětí pustíte.

Vytvoření nové třídy

Pro vytvoření nové aktivity vytvořte nejdříve ve vašem hlavním balíčku novou třídu jazyka Java. Používáte-li stejný název balíčku, jaký jsme si uvedli v předchozí kapitole, je vaším hlavním balíčkem `com.proandroidgames`. Klepněte na název balíčku pravým tlačítkem a zvolením příkazu **New → Class** otevřete okno **New Java Class**, uvedené na obrázku 3.2.



Obrázek 3.2: Okno New Java Class

Většinu výchozích voleb ponechejte beze změny. V tomto okamžiku stačí jen zadat název třídy. Vaše třída by se měla jmenovat `SFMainMenu`. Pro vytvoření třídy klepněte na tlačítko `Finish`.

Nyní je nově vytvořená třída obyčejnou třídou jazyka Java s následujícím kódem:

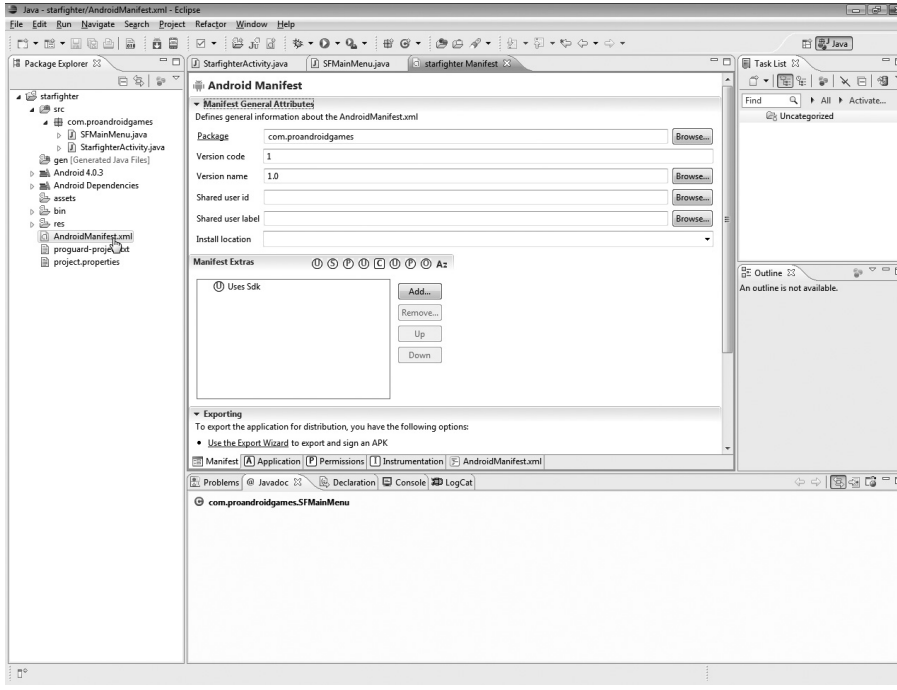
```
package com.proandroidgames;

public class SFMainMenu {

}
```

Tato třída ovšem ještě není aktivitou. K tomu je nutné přidat do ní určitý kód. Jakmile z této třídy bude aktivita, můžete začít vytvářet úvodní obrazovku a její efekty.

Transformování třídy na aktivitu

Importujte balíček `Activity` a rozšiřte svoji třídu `SFMainMenu`, čímž z ní uděláte aktivitu systému Android. Kód vaší třídy by měl nyní vypadat takto:

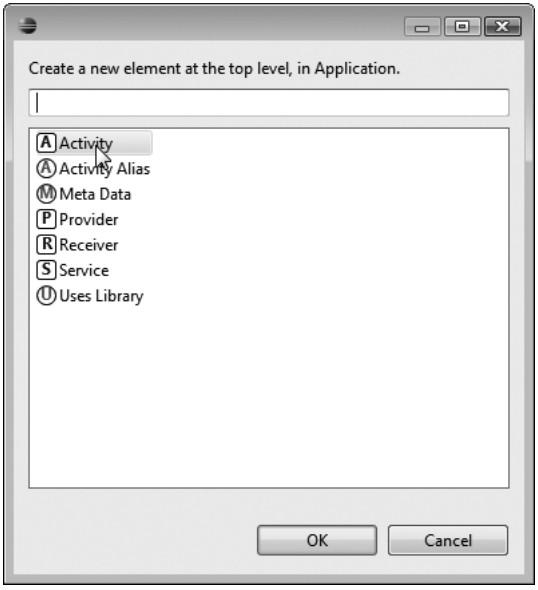
```
package com.proandroidgames;

import android.app.Activity;

public class SFMainMenu extends Activity {

}
```



Obrázek 3.3: Soubor AndroidManifest.xml

Nyní tuto aktivitu spojíme s projektem hry Star Fighter, abychom tak mohli vytvořit úvodní obrazovku. Otevřete soubor `AndroidManifest.xml`, v němž spojíte aktivitu `SFMainMenu` se svým projektem (viz obrázek 3.3).

Přejděte do spodní části karty **Application** a vyhledejte oblast označenou jako **Application Nodes**. Tato oblast manifestu uvádí všechny uzly aplikace, které jsou spojené s vaším projektem. V současnosti by měl být jediným uzlem aplikace uzel **.StarfighterActivity**. Pro přidání nové aktivity klepněte na tlačítko **Add** a na obrazovce zachycené na obrázku 3.4 zvolte položku **Activity**.



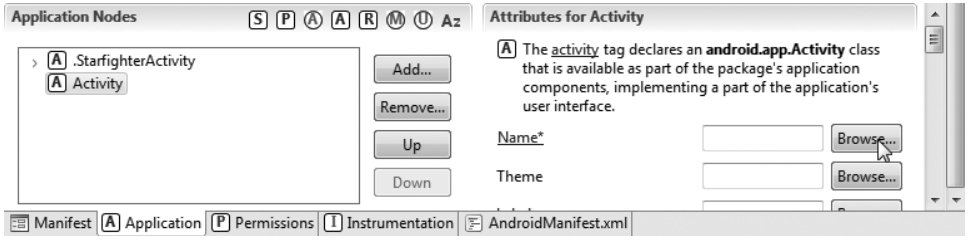
Obrázek 3.4: Vytvoření nového prvku Activity

Tím se vytvoří prázdný prvek **Activity**. Prázdný prvek, který vidíte v grafickém uživatelském rozhraní souboru `AndroidManifest.xml`, je grafickou reprezentací elementu jazyka XML v souboru `AndroidManifest.xml`. Po klepnutí na kartu **AndroidManifest.xml** byste měli vidět následující úryvek kódu jazyka XML:

```
<activity></activity>
```

Je jasné, že tento prázdný element toho moc nenadělá. Musíte nějakým způsobem vyjádřit, že tento element aktivity reprezentuje aktivitu `SFMainMenu`. To lze samozřejmě provést ručně. Podívejme se však na to, jak se to provádí automatizovaným způsobem.

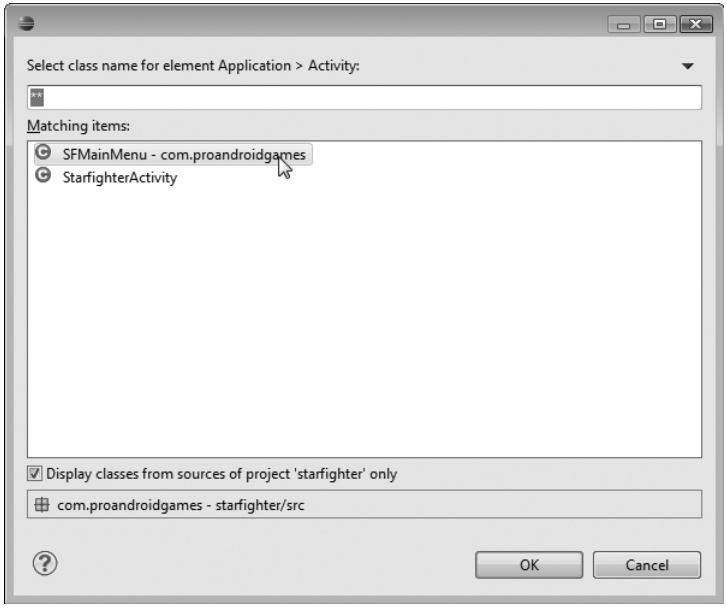
Jakmile vytvoříte nový prvek **Activity**, musíte jej spojit se skutečnou, dříve vytvořenou aktivitou `SFMainMenu`. V části **Application Nodes** klepněte na prvek **Activity** – tím jej zvýrazníte. Po pravé straně je část, která je nyní označena jako **Attributes for Activity** (viz obrázek 3.5).



Obrázek 3.5: Atributy pro aktivitu

Poznámka: Pokud byste klepli na prvek **.StarfighterActivity**, byla by tato část označená jako **Attributes for .StarfighterActivity**.

Klepnutím na tlačítko **Browse**, jež se nachází vedle atributu **Name**, otevřete prohlížecí nástroj, který zobrazuje všechny dostupné třídy ve vašem projektu odvozené od třídy `Activity`. Volby tohoto nástroje by měly vypadat jako na obrázku 3.6.



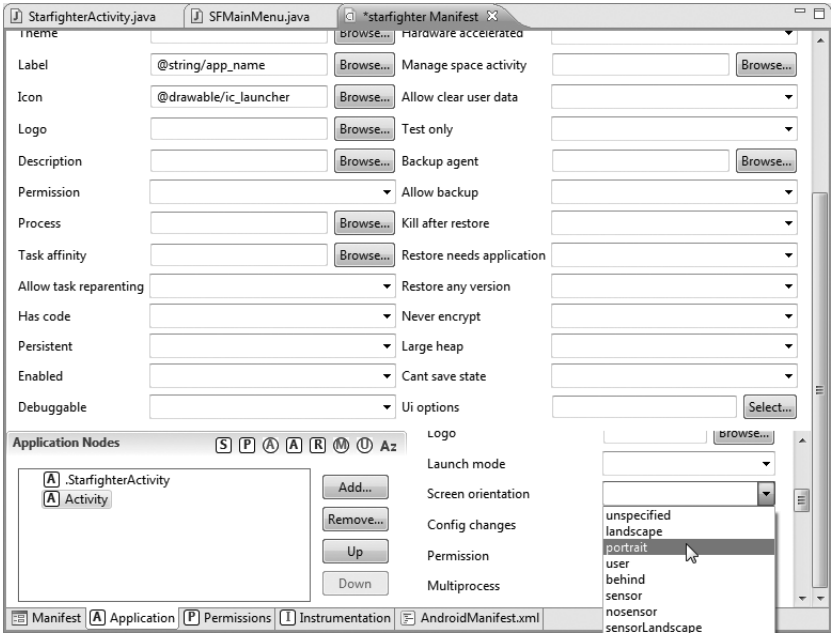
Obrázek 3.6: Selektor pro atribut Name

Všimněte si, že v poli **Matching items** je uvedena aktivita `SFMainMenu`. Zvolte tuto aktivitu a klepněte na tlačítko **OK**.

Tip: Pokud aktivitu `SFMainMenu` v poli **Matching items** nevidíte, zkuste se v prostředí Eclipse vrátit na kartu **SFMainMenu.java**. Je-li před popiskem této karty hvězdička, není tento soubor dosud uložený. Uložte jej a poté znovu otevřete prohlížeč pro atribut **Name**.

Pokud aktivitu `SFMainMenu` stále nevidíte, zkontrolujte, zda vaše třída `SFMainMenu` rozšiřuje třídu `Activity`. V případě, že vaše třída skutečně rozšiřuje třídu `Activity`, a přesto nemáte možnost vybrat aktivitu `SFMainMenu`, můžete soubor `AndroidManifest.xml` upravit ručně vyplněním nezbytných atributů (uvidíte je v pozdější části této kapitoly).

Po výběru aktivity `SFMainMenu` jako názvu pro tuto aktivitu nastavte orientaci obrazovky pro obě aktivity **.StarfighterActivity** a **SFMainMenu** na hodnotu **portrait** (viz obrázek 3.7).



Obrázek 3.7: Nastavte atributu Screen orientation na hodnotu portrait

Nastavení orientace obrazovky pro aktivity **.StarfighterActivity** (vaše úvodní obrazovka) a **SFMainMenu** (hlavní nabídka hry) uzamkne orientaci obrazovky na výšku. S ohledem na styl této hry chcete, aby ji hráč mohl používat pouze v režimu na výšku. Proto i v případě, že se hráč pokusí zařízení otočit do režimu na šířku, zůstanou obrazovky pro vaši hru orientované na výšku.

Hotový kód jazyka XML vaší nové aktivity **SFMainMenu** by měl nyní vypadat takto:

```
<activity android:name="SFMainMenu" android:screenOrientation="portrait">
</activity>
```

Aktivita hlavní nabídky je nyní spojená s projektem hry Star Fighter, a můžete tedy vytvořit úvodní obrazovku. Mějte na paměti, že veškerý kód pro hlavní nabídku se bude přidávat až v následující části – pro správnou přípravu efektu prolnutí nyní stačí, máte-li vytvořenou aktivitu.

Upozornění: Jednou z nejčastějších příčin způsobujících pády a selhání aplikace Android je nesprávné nastavení souboru `AndroidManifest.xml`, což je jeden z nejdůležitějších souborů ve vašem projektu.

Pojďme si stručně připomenout, kde se v tomto okamžiku nacházíme a proč. Úvodní obrazovka, kterou vytváříte pro hru Star Fighter, plynule přejde do hlavní nabídky. Vytvořili jste aktivitu, která bude uchovávat hlavní nabídku, a nyní nastal čas vytvořit úvodní obrazovku pro efekt prolnutí.

Vytvoření obrázku pro úvodní obrazovku

Nyní je nutné do vašeho projektu importovat grafiku, kterou použijete pro obrázek úvodní obrazovky. Systém Android je schopný pracovat s většinou běžně používaných obrázkových formátů. Pro tuto se hru se ovšem budete držet pouze dvou: `.png` a `.9.png`. Pro všechny sprity a ostatní obrázky hry budete používat standardní obrázky `.png`, zatímco pro úvodní obrazovku a hlavní nabídku použijete soubory `.9.png`.

Obrázek ve formátu `.9.png` se označuje také jako obrázek ve formátu nine-patch. Formát nine-patch je zvláštním druhem formátu, který umožňuje systému Android roztáhnout obrázek dle potřeby, protože kolem levého a horního okraje obrázku je černá hranice široká 1 pixel.

Poznámka: Většina obrázků, které do vaší hry začleníte, nebude ve formátu nine-patch, protože budete chtít sami řídit manipulaci s většinou obrázků. Nicméně pro úvodní obrazovku a hlavní nabídku je naprosto ideální použít formát nine-patch.

Rozdíl mezi formátem nine-patch a jinými procesy pro změnu velikosti obrázku spočívá v tom, že manipulací s černou hranicí můžete řídit způsob, jakým může systém Android roztahovat obrázek. Obrázek 3.8 ukazuje naši úvodní obrazovku ve formátu nine-patch.

Podíváte-li se pozorně na levou stranu obrázku 3.8, uvidíte tenkou čáru z černých teček. Tato černá čára je to, čím se obrázky ve formátu nine-patch liší od ostatních obrázkových formátů.

Poznámka: Formát nine-patch, který používáme v tomto příkladu, se může roztahovat volně ve všech směrech. Pokud se v obrázku nacházejí části, které se nemají roztahovat, nekreslete kolem nich hranici. Nástroj `draw9patch` vám může pomoci s vizualizací způsobu, jakým se váš obrázek bude roztahovat v závislosti na tom, jak nakreslíte hranici.

Obrázek 3.8: Úvodní obrazovka ve formátu nine-patch

Aplikace vyvinuté pro platformu Android se naneštěstí dají spouštět v mnoha různých velikostech obrazovky na mnoha různých zařízeních, od malých mobilních telefonů po větší tablety. Váš projekt se proto musí být schopen přizpůsobit všem nejrůznějších velikostem obrazovky. Použijete-li pro úvodní obrazovku obrázek ve formátu nine-patch, může Android (s pomocí určitého kódu v jazyce XML) změnit jeho velikost tak, aby krásně pasoval na libovolnou velikost obrazovky.

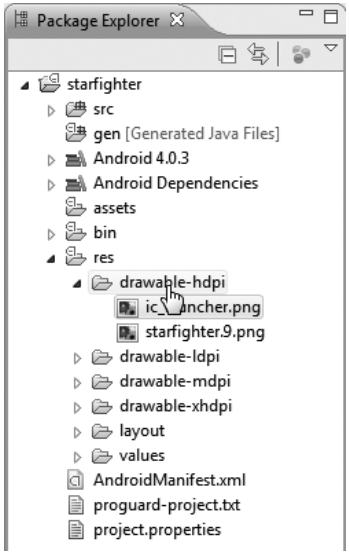
Tip: Pokud jste s grafikou ve formátu nine-patch nikdy nepracovali, může vám pomoci nástroj, který nabízí sada Android SDK. Ve složce `\tools` najdete nástroj `draw9patch`. Spusťte jej a budete moci importovat libovolný obrázek, nakreslit svou hranici pro formát nine-patch a obrázek zase uložit s příponou `.9.png`.

Importování obrázku

Nyní již tedy máte svůj obrázek ve formátu nine-patch připravený. Přetáhněte jej z místa, kam jste jej uložili, do složky `\res\drawable-hdpi` ve vašem projektu v prostředí Eclipse (viz obrázek 3.9). Možná jste si všimli, že se zde nacházejí tři složky: `drawable-hdpi`, `drawable-ldpi` a `drawable-mdpi`. Tyto složky obsahují obrázky pro tři různé typy zařízení na platformě Android: HDPI (High Density Per Inch – vysoká hustota na palec), MDPI (Medium Density Per Inch – střední hustota na palec) a LDPI (Low Density Per Inch – malá hustota na palec).

Proč používáme grafiku ve formátu nine-patch pro změnu velikosti obrázku tak, aby pasoval na libovolnou obrazovku, když systém Android poskytuje mechanizmus pro začlenění různých obrázků pro různé velikosti obrazovky? Stručnou odpovědí je, že tyto dva scénáře se ve skutečnosti vzájemně vylučují. Ano, formát nine-patch umožňuje změnu velikosti obrázku pro obrazovku určitého zařízení, to však má jen málo co dělat s hustotou obrazovky zařízení

v pixelech. Obrázky, které používáte (použijete-li obrázek z tohoto projektu), mají vysokou hustotu a jako takové se i zobrazí. Ale ani tato velikost nestačí na velikost obrazovky Motorola Xoom s úhlopříčkou 10,1 palce. Formát nine-patch proto umožňuje jejich správné roztažení.



Obrázek 3.9: Přetažení obrázku do složky drawable-hdpi

Skutečný přínos oddělení složek pro obrázky s vysokou, střední a nízkou hustotou začne být patrný v okamžiku, kdy chcete používat různá rozvržení a hustoty obrázků pro využití větší plochy obrazovky nebo naopak pro ústupky vůči obrazovkám, které mají menší plochu. Chcete-li vytvořit obrazovku nabídky, která má čtyři tlačítka umístěna v případě obrazovek tabletů nad sebou a u menších zařízení vedle sebe, můžete to díky těmto složkám realizovat jen s minimálním úsilím.

Pro účely našeho současného projektu umístěte úvodní obrazovku ve formátu nine-patch do složky `drawable-hdpi`. Ostatní složky v této hře používat nebudete. Klidně s nimi ovšem experimentujte a zkoušejte, jaké výsledky obdržíte na různých zařízeních.

Práce se souborem R.java

Po přetažení obrázku do složky pro něj Android vytvoří ukazatel na prostředek. Tento ukazatel je umístěný v souboru `R.java`, který se generuje automaticky a neměl by se ručně měnit. Nachází se ve složce `gen` pod názvem vašeho balíčku. Otevřete-li soubor `R.java` po přidání vašeho obrázku, měl by obsahovat kód podobný následujícímu:

```
package com.proandroidgames;

public final class R {
    ...
```

```
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
        public static final int starfighter=0x7f020001;
    }
    ...
}
```

Poznámka redakce českého vydání: Pokud soubor `R.java` ve složce `gen` nevidíte, zkuste svůj projekt zkompilovat – stačí v prostředí Eclipse zvolit příkaz **Project → Build Project**.

Soubor `R.java` bude spravovat všechny obrázky, identifikátory, rozvržení a ostatní prostředky používané vaším projektem. Protože tento soubor nyní obsahuje ukazatel na váš obrázek, můžete se na něj odkazovat kdekoliv ve vašem projektu pomocí následujícího kódu:

```
R.drawable.starfighter
```

Upozornění: Buďte opatrní, abyste soubor `R.java` nevymazali nebo nějakým způsobem ručně nezměnili. Například šestnáctková hodnota pro ukazatel na obrázek `startfighter` může být ve vašem systému odlišná. Váš soubor bude fungovat na vašem stroji, protože byl vygenerován ve vašem vývojovém prostředí. Pokud byste změnili vaši šestnáctkovou hodnotu tak, aby odpovídala výše uvedené, váš soubor by již nefungoval dle očekávání.

Nyní již tedy máte ve svém projektu obrázek, který chcete zobrazit jako úvodní obrazovku. Musíte ovšem ještě říci systému Android, aby jej zobrazil na obrazovce. To se dá provést mnoha způsoby. Protože však chcete na tento obrázek aplikovat efekt prolnutí, použijete rozvržení (layout).

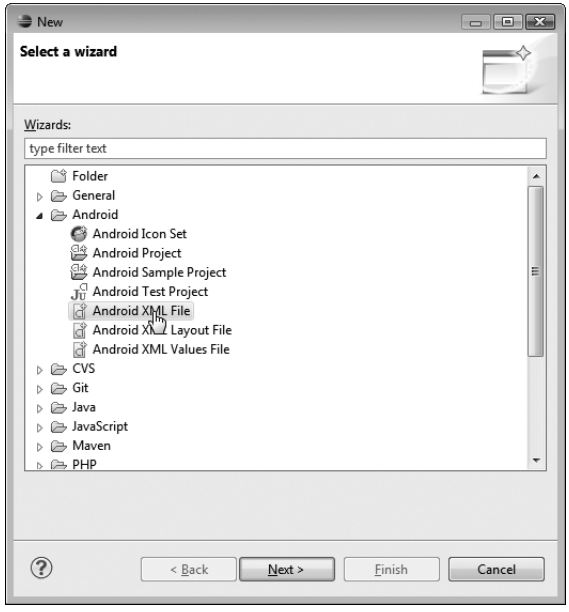
Rozvržení je soubor XML, který se používá k tomu, aby systém Android věděl, jak umístit daný prostředek na obrazovce. Vytvořme tedy rozvržení pro úvodní obrazovku.

Vytvoření souboru rozvržení

Budete používat jednoduchý soubor rozvržení pro zobrazení obrázku úvodní obrazovky (`starfighter`), jakmile hráč poprvé načte vaši hru. Vaše úvodní obrazovka bude jasná a výstižná – obrázek fascinujícího vesmírného prostředí a název hry.

Nyní musíte tento obrázek dostat na obrazovku, aby jej mohl hráč náležitě ocenit. Nejdříve klepněte pravým tlačítkem na složku `res\layout` a zvolte příkaz **New → Other**. V průvodci **New** vyberte **Android → Android XML File** (viz obrázek 3.10).

Nový soubor XML pojmenujte **splashscreen.xml** a dokončete průvodce. Do vaší složky `layout` se uloží nový soubor rozvržení a ukazatel na něj se přidá do souboru `R.java`.



Obrázek 3.10: Volba Android XML File

V tomto okamžiku můžete pracovat s rozvržením dvěma způsoby. Můžete použít návrháře grafického uživatelského rozhraní nebo soubor XML přímo upravovat. My budete upravovat soubor XML přímo, abyste tak měli lepší povědomí o tom, co přesně se do rozvržení vkládá a proč.

Úprava souboru XML

Poklepejte na soubor `splashscreen.xml` umístěný ve vaší složce `res\layout`, čímž jej otevřete v návrháři grafického uživatelského rozhraní. Když se ovšem podíváte na spodní část okna návrháře v prostředí Eclipse, uvidíte dvě podřízené karty. Jedna karta, aktuálně zvolená, je označená jako **Graphical Layout**. Druhá karta je označena jako **splashscreen.xml**. Tato karta je textový editor pro soubor XML. Klepněte na ni a vstupte do textového editoru.

Váš soubor XML by měl vypadat takto:

```
<?xml version="1.0" encoding="utf-8"?>
```

Jedná se o prázdný XML soubor. Nyní do něj přidáme rozvržení.

Při obecném vývoji pro platformu Android lze používat několik odlišných typů rozvržení. Vzhledem k tomu, že vyvíjíte hru, nemusíte se ve skutečnosti o 75 procent těchto rozvržení vůbec starat, protože na ně při vytváření hry Star Fighter jednoduše nenarazíte. Jsou zde ovšem dvě rozvržení, která byste mohli použít pro tuto úvodní obrazovku: `LinearLayout` a `FrameLayout`. Pro hru Star Fighter budete používat rozvržení `FrameLayout`, protože je skvělé pro vycentrování elementů a pro jejich přichycení na okraj.

Rozvržení `LinearLayout` se používá pro zobrazení několika prvků na obrazovce a pro jejich umístění za sebou ve svislém nebo vodorovném směru. Rozvržení `LinearLayout` si můžete představit jako jednosloupcovou nebo jednořádkovou tabulku. Můžete jej použít pro umístění libovolného množství prvků na obrazovku, včetně dalších rozvržení, do uspořádaného, lineárního stylu.

Rozvržení `FrameLayout` se používá pro uchovávání jednoho prvku. Jeden prvek lze gravitačně nastavit tak, aby byl vycentrovaný, vyplňoval celý prostor nebo byl přichycený k libovolnému okraji. Rozvržení `FrameLayout` působí, jako by bylo záměrně vytvořené pro zobrazování úvodní obrazovky, kterou tvoří jediný obrázek.

Použití rozvržení FrameLayout

Budete používat rozvržení `FrameLayout` pro zobrazení úvodní obrazovky a pole s textem, jenž bude identifikovat vás jakožto vývojáře. Faktem je, že jsme si právě řekli, že rozvržení `FrameLayout` bylo vytvořeno pro zobrazování jednoho prvku, a to také platí. Nicméně pokud rozvržení `FrameLayout` řeknete, aby zobrazilo dva prvky, budou se oba překrývat, pro jejich zobrazení ale bude zapotřebí mnohem méně kódu než při použití jakéhokoli jiného typu rozvržení.

Vraťte se ke svému souboru `splashscreen.xml` a níže uvedeným způsobem vytvořte rozvržení `FrameLayout`:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android">

</FrameLayout>
```

Rozvržení `FrameLayout` přijímá pouze dvě vlastnosti, které by vás nyní mohly zajímat: `layout_width` a `layout_height`. Tyto dvě vlastnosti řeknou systému Android, jak dané rozhraní napasovat na aktivitu, kterou jste vytvořili.

V této situaci nastavíte vlastnosti `layout_width` a `layout_height` na hodnotu `match_parent`. Konstanta `match_parent` říká systému Android, že šířka a výška zobrazení by měly odpovídat šířce a výšce rodičovského prvku tohoto zobrazení, což je v tomto případě samotná aktivita.

Tip: Pokud jste již pro platformu Android programovali, možná si vzpomenete na konstantu s názvem `fill_parent`. Tuto konstantu nahradila právě konstanta `match_parent`, obě však fungují stejným způsobem.

Vlastnosti rozvržení `FrameLayout` nastavte následujícím způsobem:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent">

</FrameLayout>
```

Nyní máte funkční rozvržení `FrameLayout`, které je ale zatím prázdné. Přidejme tedy obrázek a text.

Přidání obrázku a textu

Uvnitř rozvržení `FrameLayout` vytvořte zobrazení `ImageView` a přidělte mu ID `splashScreenImage`.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  <ImageView android:id="@+id/splashScreenImage">

  </ImageView>
</FrameLayout>
```

Vytvořili jste zobrazení `ImageView`, které bude uchovávat obrázek vaší úvodní obrazovky. Nyní musíte nastavit vlastnost `src` tak, aby ukazovala obrázek, který chcete zobrazit – v tomto případě obrázek `starfighter` ve složce `res\drawable-hdpi`. Dále musíte nastavit vlastnosti `layout_width` a `layout_height` stejně jako u rozvržení `FrameLayout`.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  <ImageView android:id="@+id/splashScreenImage"
             android:src="@drawable/starfighter"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  </ImageView>
</FrameLayout>
```

Všimněte si, že vlastnost `src` ukazuje na `"@drawable/starfighter"`, což pro systém Android znamená, že má zobrazit obrázek `starfighter` ze složky `drawable`. Nyní se podíváme na něco, co možná není tak zřejmé. Při výkladu o obrázcích ve formátu nine-patch jsme si řekli, že potřebujeme nějaký kód pro využití možností škálování formátu nine-patch. Nastavení vlastnosti `layout_width` anebo `layout_height` na hodnotu `match_parent` zajistí použití formátu nine-patch pro správné škálování vašeho obrázku vámi stanovaným způsobem.

Vytvořte nyní ve svém rozvržení zobrazení `TextView`. Použijete jej pro zobrazení jakéhokoliv textu, který chcete zobrazit na úvodní obrazovce.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  <ImageView android:id="@+id/splashScreenImage"
             android:src="@drawable/starfighter"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
```

```
  </ImageView>
  <TextView android:text="hra: j.f.dimarzio - grafika: ben eagel"
            android:id="@+id/creditsText">
  </TextView>
</FrameLayout>
```

Vytvoření tohoto zobrazení nevyžaduje žádná kouzla a mělo by být docela přímočaré. Opět musíte sdělit systému Android šířku (`layout_width`) a výšku rozvržení (`layout_height`). Pokud bychom ovšem tyto vlastnosti nastavili na hodnotu `match_parent` jako v případě elementů `ImageView` a `FrameLayout`, text by nevhodným způsobem překryl obrázek.

Místo toho nastavíte vlastnosti `layout_width` a `layout_height` níže uvedeným způsobem na hodnotu `wrap_content`. Konstanta `wrap_content` oznámí systému Android, aby velikost zobrazení `TextView` určil podle velikosti v něm obsaženého textu. Čím více textu tedy přidáte, tím bude zobrazení `TextView` větší.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  <ImageView android:id="@+id/splashScreenImage"
             android:src="@drawable/starfighter"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  </ImageView>
  <TextView android:text="hra: j.f.dimarzio - grafika: ben eagel"
            android:id="@+id/creditsText"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content">
  </TextView>
</FrameLayout>
```

Nakonec potřebujete zajistit, aby text nebyl příliš rušivý, a proto nastavíte vlastnost `gravity` zobrazení `TextView` tak, aby se přitáhl doprostřed spodní části rozvržení `FrameView`.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  <ImageView android:id="@+id/splashScreenImage"
             android:src="@drawable/starfighter"
             android:layout_width="match_parent"
             android:layout_height="match_parent">
  </ImageView>
  <TextView android:text="hra: j.f.dimarzio - grafika: ben eagel"
            android:id="@+id/creditsText"
            android:layout_gravity="center_horizontal|bottom"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content">
  </TextView>
</FrameLayout>
```

Úspěšně jste vytvořili rozvržení, které zobrazí vaši úvodní obrazovku. Nyní již stačí oznámit aktivitě `StarfighterActivity`, aby toto rozvržení použila.

Propojení aktivity StarfighterActivity s rozvržením

Propojení aktivity `StarfighterActivity` s rozvržením je velice jednoduché a vyžaduje jen jeden řádek kódu.

Uložte soubor `splashscreen.xml`. Tím se vytvoří další záznam v souboru `R.java`, díky čemuž se na toto rozvržení můžete odkazovat ve vašem dalším kódu.

Poznámka redakce českého vydání: Pokud soubor `R.java` záznam pro rozvržení `splashscreen` neobsahuje, zkuste svůj projekt zkompilovat – stačí v prostředí Eclipse zvolit příkaz **Project → Build Project**.

Otevřete soubor `StarfighterActivity.java`, umístěný v kořenovém adresáři zdrojových souborů vašeho projektu. Tento soubor byl vytvořený automaticky při vytváření projektu.

Tip: Pokud soubor s názvem `StarfighterActivity.java` nemáte, pak zkontrolujte, zda jste postupovali dle pokynů v předchozí kapitole pro vytvoření projektu. Pokud jste svůj projekt pojmenovali jinak než `starfighter`, bude mít vaše aktivita odlišný název.

Při otevření souboru `StarfighterActivity.java` uvidíte automaticky vygenerovaný kód, který zobrazuje předem vytvořené rozvržení s názvem `main`.

```
package com.proandroidgames;

import android.app.Activity;
import android.os.Bundle;

public class StarfighterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Změňte volání metody `setContentView()` tak, aby se místo zobrazení rozvržení `main` zobrazilo rozvržení `starfighter`, které jste právě vytvořili. Hotový kód aktivity by měl vypadat takto:

```
package com.proandroidgames;

import android.app.Activity;
import android.os.Bundle;

public class StarfighterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /* zobraz úvodní obrazovku z rozvržení */
        setContentView(R.layout.splashscreen);
    }
}
```

Klepnutím na zelené kolečko s bílou šipkou uvnitř umístěné na panelu nabídky svůj kód zkompilujte a spusťte. Můžete také stisknout Ctrl+F11 nebo v nabídce zvolit příkaz **Run → Run**.

Pokud jste ještě nikdy dříve nekompilovali nebo neladili aplikaci pro platformu Android, může se objevit obrazovka, která se vás ptá, zda si přejete spustit svoji aplikaci jako test frameworku JUnit nebo jako aplikaci pro platformu Android. V takovém případě spusťte aplikaci jako aplikaci pro platformu Android. Potom si můžete zvolit, ve které verzi emulátoru nebo v jakémkoli připojeném zařízení se systémem Android v ladicím režimu svoji aplikací spustíte.

! **Upozornění**: Pokud se rozhodnete spustit svůj kód v emulátoru systému Android, a ne ve skutečném telefonu se systémem Android, můžete obdržet nečekané výsledky. Mějte na paměti, že emulátor je jen emulátor a není přesnou reprezentací toho, jak vaše hra bude vypadat na skutečném zařízení (a emulátor byste používat vůbec neměli!). Buďte tedy opatrní, dokud svoji práci neuvidíte na skutečném zařízení.

Po spuštění hry byste měli vidět úvodní obrazovku jako na obrázku 3.11. To je velký úspěch a první milník při vytváření vstupního bodu do vaší hry. Tato obrazovka toho ovšem v současné chvíli ve skutečnosti moc nedělá. Kromě zobrazení nedělá vůbec nic jiného. Musíte vytvořit efekty plynulého objevení a zmizení, které povedou z úvodní obrazovky do hlavní nabídky.

Obrázek 3.11: Úvodní obrazovka hry Star Fighter

Ukončete aktivitu `StarfighterActivity` a vraťte se zpět ke svému kódu. Nastal čas vytvořit efekty plynulého objevení a zmizení.

Vytvoření efektů prolnutí

Pro vytvoření efektu plynulého objevení se úvodní obrazovky a následného prolnutí do hlavní nabídky použijete animaci. Platforma Android obsahuje několik vestavěných efektů animace, které se snadno používají a velice snadno implementují.

Proč použít animaci pro plynulé objevení a zmizení? Jednoduchou odpovědí je, že jde o snadný způsob, jak vylepšit vzhled hry. Pokud byste měli pouze statickou obrazovku, která by se jen přepnula do hlavní nabídky, dosáhli byste sice téhož cíle, avšak prolnutí obou obrazovek dodá vaší hře mnohem profesionálnější vzhled.

Vytvořte dva další soubory ve složce `res\layout`: jeden pojmenovaný `fadein.xml` a druhý `fadeout.xml`. Jak již jejich názvy napovídají, soubor `fadein.xml` se převede na animaci, která plynule zobrazí úvodní obrazovku na vašem zařízení. Soubor `fadeout.xml` bude řídit animaci, která prolne úvodní obrazovku do hlavní nabídky.

Typ animace, kterou vytvoříte, se označuje jako alfa. „Alfa“ zde znamená hodnotu alfa obrázku neboli jeho průhlednost. Hodnota alfa 1 je neprůhledná a hodnota alfa 0 je průhledná. K tomu, aby se obrázek objevil plynule, je tedy nutné vytvořit animaci, která během určitého času změní hodnotu alfa vašeho obrázku z 0 na 1. A naopak, chcete-li, aby obrázek plynule zmizel, potřebujete animaci, která během určitého času změní hodnotu alfa vašeho obrázku z 1 na 0. Z tohoto důvodu vytvoříte dvě různé animace alfa pro řízení plynulého objevení a zmizení vaší úvodní obrazovky.

Po vytvoření souborů `fadein.xml` a `fadeout.xml` ve vaší složce `res\layout` poklepejte na soubor `fadein.xml` pro jeho otevření v editoru. Soubor by měl být až na níže uvedený řádek prázdný. Pokud není, vymažte jeho obsah s výjimkou tohoto řádku:

```
<?xml version="1.0" encoding="utf-8"?>
```

Nyní vytvořte animaci alfa následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android" />
```

Pro dokončení této animace je nyní zapotřebí definovat čtyři vlastnosti: typ interpolátoru, který se má použít, počáteční a konečné hodnoty alfa a celkovou dobu trvání animace.

Nejdříve definujte interpolátor. Interpolátor říká animaci, jak má postupovat vpřed. To znamená, že animace může běžet normálně, může začít pomalu a zvyšovat rychlost, může začít rychle a snižovat rychlost nebo se může opakovat. Pro efekt plynulého objevení začneme animaci pomalu a poté ji necháme dokončit během vteřiny.

Pomocí interpolátoru `accelerate_interpolator` řeknete animaci, že chcete začít pomalu a poté zrychlit. Níže uvedený kód ukazuje, jak použít interpolátor `accelerate_interpolator` v souboru `fadein.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/accelerate_interpolator" />
```

Vaše animace plynulého objevení začne pomalu a postupně zrychluje, dokud není objevení hotové. Jak dlouho ale poběží?

Pomocí vlastnosti `android:duration` sdělte animaci alfa, jak dlouho má běžet. Vlastnost `android:duration` přijímá hodnotu v milisekundách. Nastavením vlastnosti `android:duration` na hodnotu 1000 tedy oznamte animaci, aby běžela 1 vteřinu.

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/accelerate_interpolator"
       android:duration="1000" />
```

Posledním krokem při vytváření této animace je nastavení vlastnosti pro její počáteční a konečnou hodnotu alfa. V tomto případě dochází k plynulému přechodu průhledného obrázku na úplně neprůhledný. To ovšem neznamená, že to jsou vaše jediné možnosti. Můžete zvolit počátek i konec v libovolných hodnotách mezi tím. Mohli byste tak mít například animaci, která začne na 25 procentech neprůhlednosti a skončí se 100procentní neprůhledností.

Nastavte vlastnosti `android:fromAlpha` a `android:toAlpha` tak, aby animace věděla, s jakými hodnotami alfa má začít a skončit.

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/accelerate_interpolator"
       android:duration="1000"
       android:fromAlpha="0.0"
       android:toAlpha="1.0" />
```

! **Upozornění**: Hodnoty vlastností `fromAlpha` a `toAlpha` nejsou celá čísla, ale čísla s plovoucí desetinnou čárkou. To je důležité, protože hodnota alfa se pohybuje pouze od 0 do 1.

Zde jste nastavili vlastnost `fromAlpha` na hodnotu `0.0`, což signalizuje, že animace začne s plně průhledným zobrazením. Vlastnost `toAlpha` je nastavená na hodnotu 1.0, což značí, že animace skončí s plně neprůhledným zobrazením. Tato animace vám tedy poskytne efekt plynulého objevení.

Nastal čas vytvořit efekt plynulého zmizení.

Představte si, jak by mělo plynulé zmizení fungovat ve vztahu k plynulému objevení. Plynulé zmizení by mělo fungovat podobně jako plynulé objevení, ale obráceně. To znamená, že animace by měla používat interpolátor, který začne rychle a pak zpomaluje, dokud neskončí. Animace by navíc měla začít s plně neprůhledným objektem a přejít do plně neprůhledného.

Uložte soubor `fadein.xml` a otevřete soubor `fadeout.xml`. Také zde byste měli mít pouze následující řádek kódu:

```
<?xml version="1.0" encoding="utf-8"?>
```

Nyní musíte nastavit vlastnosti `android:interpolator`, `android:duration`, `android:fromAlpha` a `android:toAlpha` pro `fadeout.xml`.

V animaci plynulého objevení jste použili interpolátor `accelerate_interpolator` pro pomalé zahájení efektu a jeho postupné zrychlování. Pro obrácení animace pro plynulé zmizení tedy použijete interpolátor `decelerate_interpolator`. Tento interpolátor zahájí animaci vyšší rychlostí, kterou pomalu zpomaluje, dokud animace neskončí.

A také pro plynulé zmizení nastavíte dobu trvání animace na 1 vteřinu (1 000 milisekund).

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/decelerate_interpolator"
       android:duration="1000" />
```

Pro dokončení animace ještě nastavte vlastnosti `android:fromAlpha` a `android:toAlpha`. Protože provádíte plynulé zmizení z neprůhledného obrázku na plně průhledný, nastavíte vlastnost `android:fromAlpha` na plně neprůhlednou a vlastnost `android:toAlpha` na plně průhlednou. Tím se spustí animace neprůhledného obrázku, který plynule přejde do průhledného.

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/decelerate_interpolator"
       android:duration="1000"
       android:fromAlpha="1.0"
       android:toAlpha="0.0"/>
```

Nyní můžete svůj hotový soubor `fadeout.xml` uložit.

V tomto okamžiku máte rozvržení a dvě animace pro řízení a definování vaší úvodní obrazovky. Potřebujete tedy nějaký způsob, jak těmto třem prvkům říci, aby spolupracovaly a vytvořily animace úvodní obrazovky.

K pochopení, jak tuto animaci budete vytvářet a spouštět, musíte rozumět tomu, jak funguje práce s vlákny ve vztahu k vaší hře.

Práce s vlákny ve hře

Jednou z největších překážek, kterou musíte jakožto vývojáři her překonat, je způsob, jakým vaše hra poběží na libovolné platformě. V nejzákladnějším kořenovém prvku je hra pro platformu Android stále jen základní aktivitou systému Android. Každá druhá „aplikace“, která je napsaná pro platformu Android, je také napsaná jako aktivita. Jediným rozdílem mezi vaší aktivitou a jakoukoli jinou spočívá v tom, že vaše obsahuje hru, zatímco jiné mohou být obchodními, mapovými nebo sociálními nástroji.

Problém s touto architekturou tkví v tom, že vzhledem k tomu, že všechny aktivity Androidu jsou stejné, zachází se s nimi jako se stejnými. To znamená, že každá aktivita systému Android, kterou napíšete, poběží v hlavním prováděcím vláknu systému. To ale není dobré pro hry. Běh hry v hlavním prováděcím vláknu systému znamená, že hra musí soupeřit o prostředky se všemi ostatními aktivitami běžícími v tomto vláknu. To vede v lepším případě k trhané nebo pomalé hře a v horším případě ke hře, která způsobí zastavení nebo zamrznutí zařízení.

Ničeho se ale neobávejte, neboť existuje způsob, jak tuto noční můru s jedním vláknem obejít. Máte možnost vytvářet libovolný počet vláken a spouštět v nich cokoliv, co chcete. V ideálním případě budete chtít, aby vaše hra běžela v jednom vláknu, které je oddělené od všeho, co běží na zařízení, abyste tak měli jistotu, že vaše hra poběží tak plynule, jak je to jen možné, a že bude mít přístup k prostředkům, které potřebuje.

Ve zbytku této kapitoly ve skutečnosti vytvoříte dvě samostatná vlákna pro provádění vaší hry. První vlákno, o němž je řeč v této části, bude pro běh hry a druhé vlákno (které vytvoříte v pozdější části této kapitoly) bude pro běh podbarvující hudby, která má hrát během hry.

Proč dvě samostatná vlákna? S výjimkou animace a herní logiky je jednou z věcí nejvíce zatěžujících procesor zařízení přehrávání médií, jako je třeba hudba. Musíte tedy zajistit, aby hra a hudba běžely plynule a současně bez vzájemného rušení. Spuštěním hudby v jiném vláknu, než ve kterém běží hra, budete navíc schopni ukončit hudbu bez zásahu do hraní hry, pokud byste zjistili, že zařízení docházejí prostředky.

Nyní již tedy rozumíte tomu, proč potřebujete pro svoji hru vytvořit různá vlákna, můžete tedy jedno vytvořit pro hlavní hru a úvodní obrazovku. Toto herní vlákno sváže dohromady úvodní obrazovku, kterou jste vytvořili, animace objevení a zmizení a hlavní nabídku.

Vytvoření herního vlákna

Znovu otevřete soubor `StarfighterActivity.java`. Pro připomenutí: váš soubor by měl nyní být schopen spustit úvodní obrazovku a měl by obsahovat následující kód.

```
package com.proandroidgames;

import android.app.Activity;
import android.os.Bundle;

public class StarfighterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /* zobraz úvodní obrazovku z rozvržení */
        setContentView(R.layout.splashscreen);
    }
}
```

Vzhledem k tomu, že `StarfighterActivity` je aktivita, která se spouští jako výchozí a která spouští úvodní obrazovku, jedná se o skvělé místo pro vytvoření vašeho herního vlákna. Vlákno, které nyní vytvoříte, bude vláknem, v němž nakonec poběží hra.

Vytvořte novou instanci třídy `Thread` a přepište metodu `run()` pro vytvoření nového vlákna. Uvnitř metody `run()` zavoláte hlavní nabídku, takže hra poběží v novém vláknu. Toto jsou základní kroky, podle nichž zde budete postupovat.

Poznámka: Během sestavování hry budeme kód v tomto vláknu upravovat, a dokonce přesouvat kvůli přizpůsobení složitějším procesům.

Následující kód ukazuje, kde uvnitř kódu třídy `StarfighterActivity` vytvořit nové vlákno:

```
package com.proandroidgames;

import android.app.Activity;
import android.os.Bundle;

public class StarfighterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /* zobraz úvodní obrazovku z rozvržení */
        setContentView(R.layout.splashscreen);

        /* spusť nové herní vlákno */
        new Thread() {
            @Override
            public void run() {

            }
        };
    }
}
```

S tímto kódem je ovšem spojený jeden problém. Tak jak je napsaný, vytvoří totiž nové herní vlákno v rámci milisekund po zobrazení úvodní obrazovky. To by sotva stačilo jen na její vykreslení. Proto musíte vytvoření herního vlákna odložit tak, aby úvodní obrazovka měla dostatek času na vykreslení.

Odpověď spočívá v použití pozdržené obsluhy (`Handler`). Android má obsluhy, které mohou spravovat vlákna a aktivity. Metoda `postDelay()` třídy `Handler` přijímá dva parametry: vlákno, které se má pozdržet, a dobu, po kterou má být pozdržené.

Nyní vytvoříte novou konstantu pro uchovávání doby, o kterou chcete pozdržet vlákno. Tato konstanta `GAME_THREAD_DELAY` bude prvním řádkem kódu ve vašem herním enginu. Tím, že ji umístíte sem, můžete upravovat zdržení vlákna z jediného místa, aniž byste ji museli hledat v celém kódu.

Vytvořte ve svém balíčku nový soubor pro třídu s názvem `SFEngine.java`. Jedná se o prázdný soubor pro třídu, která bude nakonec uchovávat většinu vašeho herního enginu. Přidejte do této třídy následující konstantu:

```
package com.proandroidgames;

public class SFEngine {
    /* konstanty, které se budou používat ve hře */
    public static final int GAME_THREAD_DELAY = 4000;
}
```

Konstantu `GAME_THREAD_DELAY` jsme nastavili na 4 vteřiny, což by mělo být dostatečné pro zobrazení úvodní obrazovky, než se objeví hlavní nabídka.

Uložte soubor `SFEngine.java` a znovu otevřete aktivitu `StarfighterActivity`. Nové herní vlákno zabalte do instance `Handler()` a pomocí metody `postDelay()` pozdržte jeho provádění.

Tip: Věnujte zvýšenou pozornost také balíčkům, které je nutné importovat. Pokud se totiž pokusíte zavolat metodu, jež se nachází v balíčku, který jste dosud neimportovali, objeví se chyby. Můžete použít klávesovou zkratku Ctrl+Shift+O pro automatické importování jakýchkoli odkazovaných balíčků, na které jste mohli zapomenout.

```
package com.proandroidgames;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;

public class StarfighterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /* zobraz úvodní obrazovku z rozvržení */
        setContentView(R.layout.splashscreen);

        /* spusť úvodní obrazovku a hlavní nabídku v pozdrženém vláknu */
        new Handler().postDelayed(new Thread() {
            @Override
            public void run() {

            }
        }, SFEngine.GAME_THREAD_DELAY);
    }
}
```

Nyní jste vytvořili nové vlákno a nastavili časovou prodlevu pro pozdržení spuštění vlákna o 4 vteřiny. Nakonec nastal čas pro sdělení vláknu, co má dělat.

Toto je pouze náhled elektronické knihy. Zakoupení její plné verze je možné v elektronickém obchodě společnosti eReading.