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 nebezpečí 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:

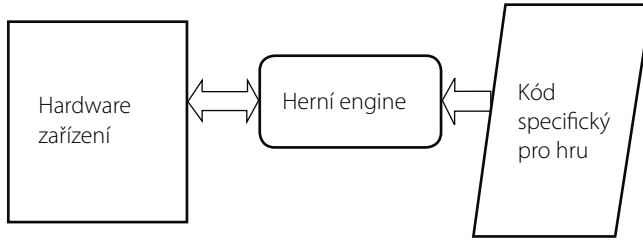
- vykreslování grafiky,
- animace,
- zvuk,
- detekce kolizí,
- umělá inteligence,
- fyzika (bez kolizí),
- správa vláken a paměti,
- komunikace přes síť,
- interpret příkazů (vstup a výstup).

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

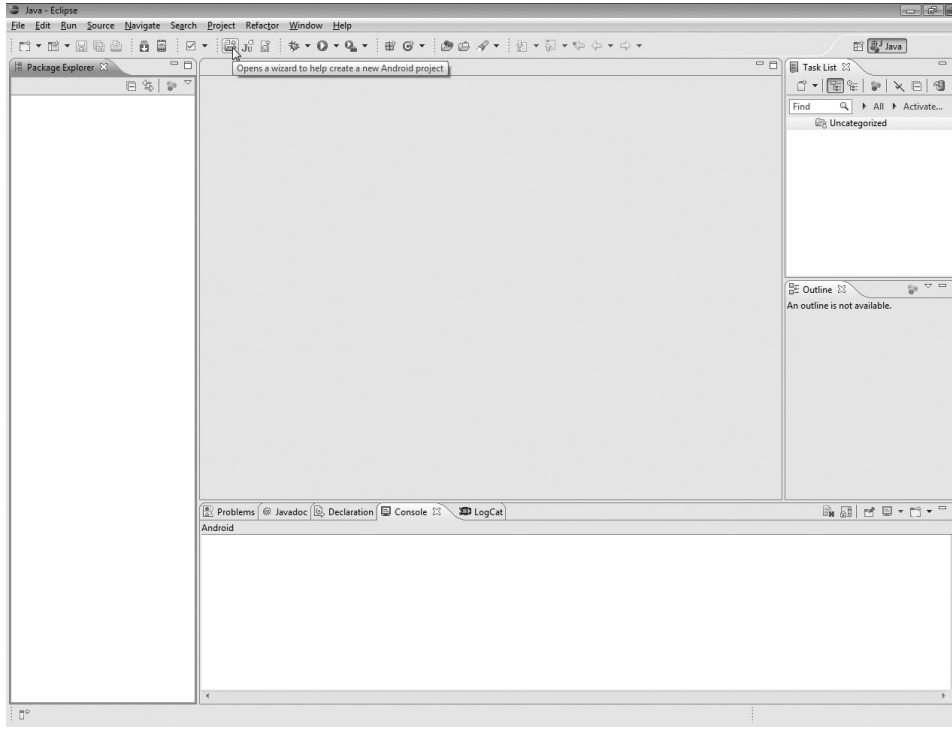
Jako počáteční úkol, v němž si vše připravíte, si v této části vytvoříte projekt, který budete používat pro hru Star Fighter. Tento projekt budeme používat v rámci celé knihy.

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`.

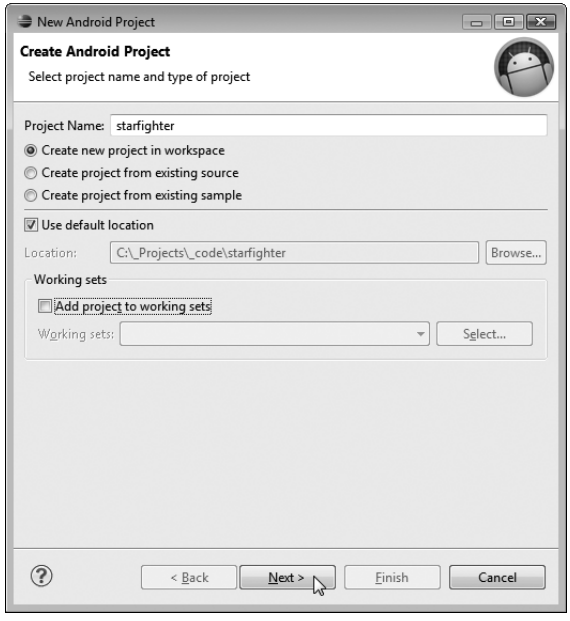


Obrázek 2.2: Spuštění nového průvodce projektu pro platformu Android

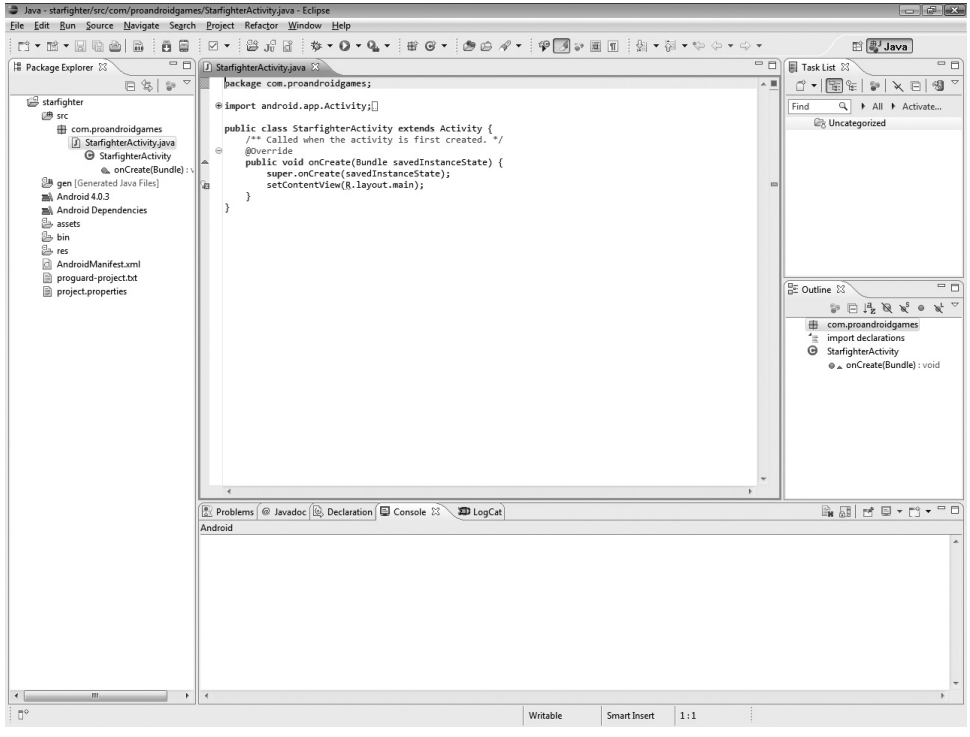
Tip: Pokud jste dosud nikdy nevytvářeli projekt pro Android (nebo pro Javu), existuje několik konvencí pro pojmenování, které byste měli mít na paměti. Při pojmenování vašeho balíčku si jej představte jako adresu URL, avšak zapsanou pozpátku. Měl by tedy začínat označením (např. `com` nebo `net`) a končit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