VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝVOJ MULTIPLATFORMNÍ HRY VYUŽÍVAJÍCÍ PRUŽNÁ TĚLESA

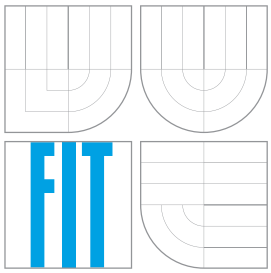
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ BASOVNÍK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝVOJ MULTIPLATFORMNÍ HRY VYUŽÍVAJÍCÍ PRUŽNÁ TĚLESA

DEVELOPMENT OF MULTI-PLATFORM GAME USING SOFT BODY ELEMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ BASOVNÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ZACHARIÁŠ
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této bakalářské práce je popsat proces vývoje her, následně navrhnout dvourozměrnou hru využívající pružná tělesa jako zábavný herní prvek a implementovat ji pomocí vhodných nástrojů, které zajistí její funkčnost napříč platformami. Problém s multiplatformitou je řešen vytvořením HTML5 hry v aplikaci Construct 2 a její následnou kompilací nástrojem Intel XDK na mobilní platformy (iOS, Android), které Construct 2 přímo nepodporuje. Chování pružných těles je popsáno dvěma modely vhodnými pro navrhovanou hru a jejich modifikacemi. Základní model využívá k udržení tělesa pohromadě pouze vlastnosti pružin (Spring-mass), pokročilý navíc tlakové síly (Pressure soft body). Celkově se použitým postupem povedlo úspěšně zprovoznit a otestovat hru na sedmi platformách a to jak stolních (Windows 7 a 8.1, Mac OS X, Linux), tak mobilních (Android, iOS, Windows Phone), navíc je hra funkční také ve webových prohlížečích. Testy bylo zjištěno, že pokročilejší model pružných těles je stabilnější a vykazuje méně chyb. Nakonec však byla použita modifikovaná verze původního modelu, která více vyhovovala požadovaným vlastnostem návrhu hry. Přínosem této práce je popsání a ukázání některých možností, které vývojář má při tvorbě multiplatformních her a také vytvoření zábavného herního prvku pomocí pružných těles a vhodně zvoleného modelu.

Abstract

The goal of this bachelor's thesis is to describe the process of game development, subsequently propose a two dimensional game using soft body objects as a fun game element and to implement it using appropriate tools to ensure its functionality across platforms. The problem with multiplatform is solved by creating HTML5 game in the application Construct 2 and its subsequent compilation by the tool Intel XDK for mobile platforms (iOS, Android) which Construct 2 application does not support directly. The behavior of the soft body objects is described by two models suitable for the proposed game and their modifications. The basic model uses to keep the body together only the properties of springs (Spring-mass), the advanced model uses extra pressure forces (Pressure soft body). Overall, the procedure used succesfuly managed to put into operation and test the game on seven platforms, both on desktop (Windows 7 and 8.1, Mac OS X, Linux) and mobile (Android, iOS, Windows Phone), in addition the game also works in web browsers. Tests releaved, that this more advanced model of soft body objects is more stable and exhibits fewer errors. Ultimately, however, was used a modified version of the basic model, which better suit the desired design characteristics of the game. The contribution of this bachelor's thesis is to describe and show some possibilities, that the developer has in creating multiplatform games and also create a fun game element using soft body objects and suitably chosen model.

Klíčová slova

pružné těleso, Spring-mass, Pressure soft body, multiplatformita, proces vývoje videoher, hybridní aplikace, Construct 2, Intel XDK, Cordova, HTML5, hra

Keywords

soft body element, Spring-mass, Pressure soft body, multi-platform, videogame development process, hybrid application, Construct 2, Intel XDK, Cordova, HTML5, game

Citace

Tomáš Basovník: Vývoj multiplatformní hry využívající pružná tělesa, bakalářská práce, Brno, FIT VUT v Brně, 2015

Vývoj multiplatformní hry využívající pružná tělesa

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Zachariáše.

.....
Tomáš Basovník
20. května 2015

Poděkování

Rád bych touto cestou poděkoval mému vedoucímu práce Ing. Michalu Zachariášovi, za zpracování zadání práce, rady v průběhu tvorby a profesionální přístup.

Obsah

Kapitola 1

Úvod

Cílem práce je vytvořit videohru využívající pružná tělesa tak, aby výsledná hra byla jen minimálně závislá na herní platformě či operačním systému.

Kapitola 2 popisuje problematiku vývoje multiplatformních videoher pomocí vhodně zvolených postupů a nástrojů tak, aby fungovala na nejrozšířenějších mobilních a stolních platformách. Patří sem platformy s operačními systémy Android, iOS, Windows Phone, Windows, Mac OS, Linux a další. Právě kvůli velkému počtu nesourodých herních systémů vzniká problém, kdy při nativním přístupu se vývoj musí opakovat pro každý systém zvlášť, což je časově velmi neefektivní. Účelem následujícího textu tedy je poskytnutí základních informací vedoucích k minimalizování tohoto problému. K tomuto účelu byl použit program Construct 2 usnadňující tvorbu HTML5 her. Pro platformy, které HTML5 nepodporují přímo, byla hra dále kompilována nástrojem Intel XDK.

Kapitola 3 se zaměřuje na druhou část práce, která se věnuje modelům pružných těles ve dvourozměrném prostředí. Tento prvek je stěžejní herní mechanikou výsledného produktu. S využitím pružnosti hráč zdolává překážky, které jinak znemožňují postup hrou. V práci popisuji a testuji dva modely vhodné pro navrhovanou hru a jejich modifikace. Základní model *Spring-mass* využívá k udržení stability tělesa pouze vlastnosti pružin, pokročilý model *Pressure soft body* navíc síly vnitřního tlaku.

Návrh hry je popsán v kapitole 4, implementace a testování těchto modelů a celé hry poté v kapitole 5.

V samotném závěru shrnuji dosažené výsledky mojí bakalářské práce a navrhuji možnosti dalšího postupu.

Kapitola 2

Proces vývoje multiplatformních videoher

Tato kapitola by měla čtenáře zasvětit do procesu vývoje multiplatformních videoher. V první části budou definovány některé pojmy, které se budou objevovat v dalších kapitolách. Stručně bude popsána i současná situace videoherního trhu se zaměřením na rozšířené herní platformy. Další část se věnuje rozdílnému přístupu k vývoji hry z hlediska dělení na produkty a služby. Poté následuje text věnovaný architektonickým vzorům vhodným k použití při tvorbě a údržbě her. Závěrečná část kapitoly představuje některé postupy a nástroje řešící problém s multiplatformitou.

2.1 Pojem HTML5 hra a současná situace videoherního průmyslu

Jelikož je zde multiplatformita řešena vytvořením HTML5 hry a následnými dalšími úpravami, je potřeba pojem *HTML5 hra* nejdříve upřesnit. Definice následujících pojmů nejsou úplně striktní, proto se v různých zdrojích můžou trochu lišit. Následují tak, jak je uvádí Nicholas Lovell [3].

Hra je dobrovolná činnost, kterou se snažíme dosáhnout jasně daného cíle, v jehož dosažení nám brání překážky za použití prostředků definovaných omezujícími pravidly a ve které nám při chybě hrozí penále. Důležitou součástí hry je i motivace — tedy důvod proč hru dobrovolně hrát. Podstatou hry je, že není těžká jako realita. Každá hra by měla mít podmínky (cíl, pravidla), mechanismy (pohybové, klidové), překážky (soupeři, zručnost), motivaci (zážitek, soutěživost) a penále (opakování, ztráta prostředků).

Videohra je hra v podobě softwaru, obvykle spuštěném na herní konzoli nebo počítači a zobrazovaném na displej nebo televizní obrazovku. Hráč hru ovládá pomocí připojených periferií. V textu bude dále používán kratší výraz „hra“ i pro označení „videohry“.

Multiplatformní hra je taková, která běží na více než jednom herním zařízení či operačním systému. Způsoby jak toho dosáhnout popisuje dále podkapitola 2.4.

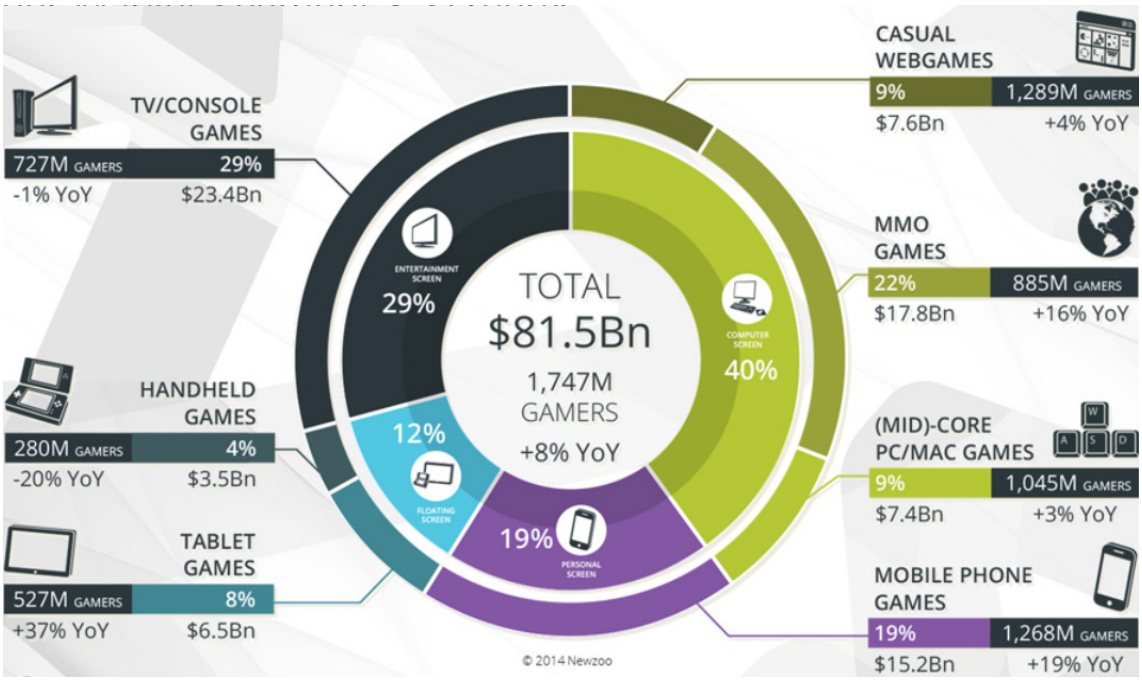
HTML5 hra je taková hra, která byla vytvořena pomocí webových technologií. HTML5 je nejnovější verzí značkovacího jazyka HTML, který se používá především k tvorbě webových stránek. HTML5 ve spojitosti s hrami a aplikacemi nám pouze poskytuje aplikační rozhraní k dalším technologiím, jako je Canvas umožňující kreslení 2D grafiky, WebGL pro 3D grafiku, Web Audio umožňující pokročilou práci se zvuky, Web Storage pro ukládání dat a další. Samotné hry a aplikace jsou pak programovány především v jazyce JavaScript.

Takto vytvořená hra či aplikace je poté spustitelná v internetových prohlížečích nebo při použití hybridního přístupu přímo v herním zařízení (viz podkapitola 2.4).

Herní průmysl a platformy

Jak uvádí společnost Newzoo, která se specializuje na průzkum herního trhu, má herní průmysl dlouhodobě rostoucí tendenci a to jak v počtu aktivních hráčů, tak ve velikosti finančního obratu. Technologie jsou levnější a dostupnější než kdy dřív. Tvorba her už díky snadné distribuci do internetových obchodů není jen doménou velkých firem.

Graf zobrazený na obrázku 2.1 znázorňuje podíl platforem na výdělku herního průmyslu za rok 2014 (celkový výdělek přibližně 81,5 miliardy dolarů) a také počet uživatelů využívajících dané platformy ke hraní her (celkem přibližně 1,7 miliardy hráčů).



Obrázek 2.1: Globální údaje o herním trhu za rok 2014[1].

V tabulce 2.1 jsou k dispozici statistiky mobilního trhu aktuální k březnu 2015. Uvedené částky nezahrnují příjem z nákupů uvnitř aplikací nebo zobrazovaných reklam, ale pouze jednorázové prodeje. Jedná se pouze o údaje z oficiálních obchodů daných platforem, tedy Google Play, Apple App Store, Windows Phone Store a BlackBerry World. Jak z tabulky vyplývá, u mobilních zařízení nejrozšířenější platforma nemusí být nejvýdělečnější. Třeba operační systém Android je nejrozšířenější, jeho uživatelé si v průměru stahují nejvíce apli-

[1] Převzato z http://www.newzoo.com/keynotes/newzoo-global-games-market-webinar-presentation/.

kací, ale nejsou za ně ochotni příliš platit. Co do počtu aplikací je na méně rozšířených platformách menší konkurence, přitom i tyto obchody dosahují nezanedbatelných příjmů.

	Android	**iOS**	**Windows Phone**	**BlackBerry**
Podíl mobilních operačních systémů	56,22 %	30,31 %	1,88 %	1,33 %
Celkový počet stažených aplikací v řádech miliard	31	29	5,1	3,4
Průměrný počet stažených aplikací na 1 telefon	88	68	57	49
Počet uživatelů, kteří nikdy neutratili více než 1 $ za aplikaci	60 %	43 %	56 %	61 %
Příjmy obchodů za rok 2014 v miliardách dolarů	1,8	6,9	1,25	0,75
Celkový počet dostupných aplikací v obchodech	950 000	1 205 000	320 000	230 000

Tabulka 2.1: Podíl mobilních operačních systémů a statistiky obchodů s mobilními aplikacemi aktuální k březnu 2015[3].

Pokud chce herní vývojář obsáhnout co největší část herního trhu, neměl by se omezovat jen na jednu platformu. Možnosti tvorby multiplatformní hry budou popsány později.

2.2 Dělení her z hlediska postupu při vývoji na produkty a služby

Hry se dají dělit podle mnoha faktorů například podle žánru, grafického stylu, podle hráčské cílové skupiny apod. Tato podkapitola se věnuje dělení na *produkty* a *služby*. Volbu jedné z variant by měli vývojáři učinit na úplném začátku, protože ovlivní celý průběh vývoje hry. Hlavní rozdíly, jak je uvádí Nicholas Lovell [3], jsou uvedeny v tabulce 2.2.

Herní produkt byl tradičně vyvíjen pro stolní počítače či konzole a prodáván v krabicové podobě prostřednictvím maloobchodních prodejen. Za takové hry se obvykle platí jednorázově. Vydavatel nebo vývojář neudržuje se svými zákazníky žádný trvalý vztah.

Digitální obchody jako Steam, GOG, Desura a další umožnili prodávat hry v digitální podobě. Tímto bylo možné odstranit roli distributora mezi vývojářem a samotným obchodem, což má v dnešní době za následek rostoucí počet nezávislých vývojářů. První vlna her na mobilní telefony se nesla ve stejném duchu — vytvořit, publikovat a začít další projekt. Rozdíl byl v tom, že se často jednalo o menší hry, jejichž vývoj byl rychlejší a méně nákladný.

Postupně se tyto nové platformy začaly měnit, když společnosti Microsoft a Sony začaly podporovat stahovatelný obsah rozšiřující původní hru (tzv. DLC) a Apple umožnil nákupy přímo v aplikaci. Vývojáři začali budovat úzké vztahy se svými zákazníky a na své hry pohlíželi více jako na služby.

[3]Statistiky podílu mobilních operačních systémů převzaty z http://gs.statcounter.com/#mobile+tablet-os-ww-monthly-201503-201503-bar ostatní z http://www.statisticbrain.com/mobile-phone-app-store-statistics/.

Herní služby se snaží své hráče udržet po dlouhou dobu tak, aby měli důvod se ke hře pravidelně vracet. Využívá se zde sociálních aspektů, hráči spolupracují se svými přáteli, což vede k nárůstu uživatelů. Vývojáři pravidelně rozšiřují obsah hry, aby měl hráč pořád co dělat. K hraní takové hry je většinou vyžadováno připojení k internetu. Dostupná je buď přímo skrze internetový prohlížeč nebo stažením herního klienta do počítače či mobilního telefonu. Provozovatel tedy musí mít určité zázemí, jako servery na kterých hra běží.

	Produkt	**Služba**
Distribuce	Fyzické/digitální médium	Online
Business model	Jednorázová platba	Prodej virtuálních předmětů Předplatné
Prodej	Obchody	Přímo zákazníkovi
Marketing	Reklama, PR	Sdílení samotnými hráči
Velikost týmu	Po vydání klesá	Po nasazení roste
Update	Jednorázový Opravuje chyby	Pravidelný Přidává funkčnost
Příjmy	Především v době vydání	Průběžně po dobu provozu
Výdaje	V době vývoje před vydáním	Průběžně po dobu provozu

Tabulka 2.2: Rozdíly mezi herním produktem a službou.

Jelikož je tvorba služby pro jednotlivce a malé týmy velmi náročná na prostředky a čas, věnuje se bakalářská práce tvorbě herního produktu.

2.3 Architektura Model-View-Controller-Service při tvorbě a údržbě her

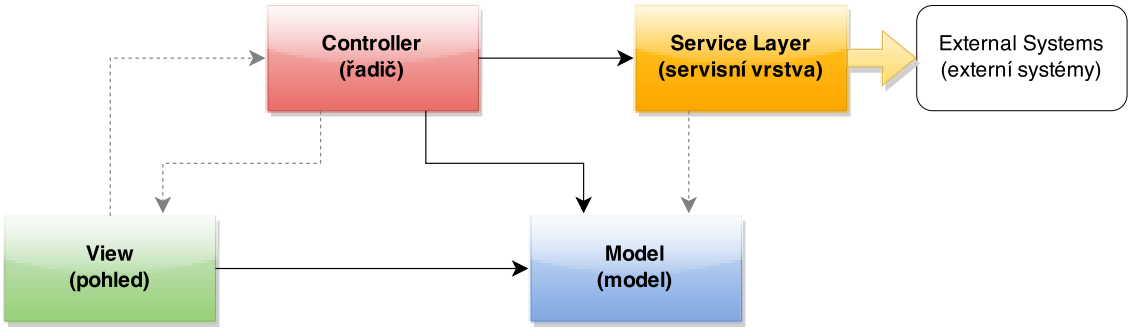
Nutnou podmínkou trvale udržitelné aplikace je zamezení nárůstu složitosti při jejích úpravách a rozšiřování. V průběhu posledních desetiletí proto bylo popsáno několik architektonických vzorů, které toto řeší.

Architekturu MVC rozebírá Borek Bernard ve své sérii článků [1]. Tato architektura dělí aplikaci na tři logické části tak, aby je šlo upravovat samostatně a dopad změn na ostatní části byl co nejmenší. Tyto tři části jsou:

- **Model (model)** – reprezentující data a doménovou logiku aplikace. Může se jednat o datovou strukturu obsahující číselné hodnoty jako počet nasbíraných bodů a doba hraní úrovně. Doménová logika je pak výpočet finálního skóre z těchto hodnot.
- **View (pohled)** – jedná se o zobrazení modelu a dalších prvků uživatelského rozhraní. Příkladem může být zobrazení skóre v tabulce nebo jinak v grafu, jde o dva různé pohledy na stejná data.
- **Controller (řadič)** – má na starosti tok událostí v aplikaci a obecně aplikační logiku. Na příkladu hry se projeví třeba ve chvíli, kdy hráč sebere další bod. Řadič má na starosti, aby se aktualizovala data v modelu a překreslily se všechny pohledy (zde pozor, promítnutí změny modelu do pohledu, nemusí mít s řadičem nic společného, řadič pouze tento proces spouští). Definice řadiče je poměrně volná, právě jeho pojetím se od sebe vzory nejvíce liší (viz vzor MVP).

Občas se přidává ještě čtvrtá vrstva. Přidáním servisní vrstvy vzniká architektura MVCS:

- **Service Layer (servisní vrstva)** – tato část odděluje komunikaci s externími systémy. V případě her bude v této části umístěna komunikace se službami třetích stran přes aplikační rozhraní, jakými jsou načítání reklam od zprostředkovatelů nebo odesílání skóre do internetových žebříčků Google Play Service a Apple Game Center. Stejně tak zde můžeme umístit komunikaci s vlastními externími databázemi a systémy.



Obrázek 2.2: Architektonický vzor MVCS.

Šipky v obrázku 2.2 naznačují několik přímých vazeb. Řadič má přímý odkaz na model, aby mohl upravit jeho data. Pohled má přímý odkaz na model, aby mohl jeho data zobrazit. V závislosti na konkrétní variaci MVC se může vyskytnout ještě vazba mezi řadičem a pohledem. V žádném případě však model nemůže držet přímý odkaz na pohled či řadič, jednalo by se o chybu v návrhu aplikace.

Tok událostí v aplikaci typicky začíná u uživatele, který vykoná nějakou akci na uživatelském rozhraní. Ta je následně zachycena řadičem. Ten rozhodne, jak na akci zareagovat. Typicky změní hodnoty v modelu nebo přímo ovlivní pohled. Nakonec pohled promítne změny uživateli.

2.4 Nativní a webové aplikace a možnosti sestavení hybridních aplikací

Tato podkapitola je věnována třem hlavním přístupům k vývoji her — nativní, webový a především hybridní. U hybridního přístupu budou popsány také aplikační rámce a programy umožňující jejich sestavení.

Nativní aplikace

Nativní přístup vyžaduje použití primárního programovacího jazyka dané platformy, u systému iOS jde o Objective-C nebo nově Swift, Java pro Android a pro Windows Phone to je převážně C#. Při programování používá vývojář specializované editory kódu, které obsahují pro cílové systémy různé pokročilé funkce, jako emulátor pro testování aplikace apod. Mezi takové editory patří Xcode pro Objective-C, Eclipse pro Javu a Visual Studio pro C#.

K funkcím koncového zařízení přes aplikační rozhraní se přistupuje odlišně podle operačního systému a toho co zařízení podporuje. V případě mobilů jsou to různé senzory, gyroskop, akcelerometr, fotoaparát a další. Liší se také doporučené zásady vzhledu aplikací, které by měl vývojář pro platformu dodržovat.

Distribuce takové aplikace probíhá skrze k tomu určené internetové obchody. Každá platforma má svůj vlastní oficiální obchod. Aplikace pro operační systém Android jsou nabízeny na Google Play, v případě iOS se jedná o App Store společnosti Apple a distribuce pro Windows Phone probíhá přes Windows Phone Store, který provozuje Microsoft. I méně rozšířené platformy jako BlackBerry, Tizen či Firefox OS mají takové obchody. Tyto obchody jsou většinou jedinou oficiální cestou k pořízení aplikace uživatelem. Aplikace či hra se po zakoupení stahuje a instaluje přímo do telefonu. Jedná se o výhodu oproti webovému přístupu především díky snadným a rychle proveditelným platbám. Protože jde o velké společnosti, uživatelé k nim chovají jistou důvěru a nebojí se provázat platební systém třeba s kreditní kartou.

Jelikož jsou aplikace spouštěny přímo v systému, nikoli skrze jádro webového prohlížeče, dosahují vyššího výkonu oproti webovému či hybridnímu přístupu.

Hlavní nevýhodou je tedy oddělený vývoj, z čehož vyplývá nutnost rozumět více technologiím, což vede k potřebě větší pracovní síly a zvyšování nákladů.

Webové aplikace

Nejjednodušší cestou k vytvoření multiplatformní hry je použití HTML5 (více 2.1). Taková hra nebo aplikace je naprogramována pouze jednou. V podstatě jde o webové stránky běžící v internetovém prohlížeči. Moderní mobilní i stolní platformy takové prohlížeče obsahují přímo nebo se do systému dají instalovat.

Jelikož se aplikace nenachází přímo v mobilním telefonu, tak uživatelé potřebují přístup k internetu. Také přístup k aplikačnímu rozhraní zařízení je poměrně omezený, závisí na typu a verzi systému a prohlížeče. Nelze se spolehnout, že bude vždy umožněn přístup k fotoaparátu, GPS modulu a dalším komponentám.

Výkon webových her oproti nativním je slabší, proto se dá použít jen na menší projekty. Uživatelské rozhraní často nebývá konzistentní s danou platformou, i když existují aplikační rámce, kterými toho lze dosáhnout.

Hry jsou nejčastěji zpeněženy pouze reklamou. Na druhou stranu odpadají schvalovací procesy a další práce navíc, která se vyskytuje u internetových obchodů. Zde jsou aktualizace plně v režii vývojářů.

Největší záporem je ale fakt, že samotní uživatelé raději používají nativní mobilní aplikace než webové. Podle statistik společnosti Flurry[4] z dubna 2014 tráví 86% času uživatelé v nativních aplikacích (z toho 32% času hrají hry) a jen zbylých 14% ve webovém prohlížeči.

Hybridní aplikace

Hybridní aplikace spojuje oba již zmíněné přístupy. Jak uvádí Patrick Rudolph ve svém článku [6], k tvorbě hybridní aplikace lze přistoupit dvěma způsoby:

- **WebView aplikace** – Základem je webová HTML5 aplikace popsaná v předchozí části, která ale běží v interním prohlížeči nazývaném WebView a je zapouzdřena do podoby nativní aplikace. Pomocí Apache Cordova, Trigger.io a podobných aplikačních rámců lze zpřístupnit aplikační rozhraní zařízení pro programování v JavaScriptu.
- **Kompilovaná hybridní aplikace** – Kód je napsán v jednom jazyku (např. C# nebo JavaScript) a je kompilován do nativního kódu každé podporované platformy.

[4] Údaje převzaty z http://flurry.com/bid/109749/Apps-Solidify-Leadership-Six-Years-into-the-Mobile-Revolution.

Výsledkem je nativní aplikace pro každou platformu, ale za cenu menší volnosti během vývoje. Příkladem takových aplikačních rámců jsou Xamarin, Appcelerator Titanium a Embarcadero FireMonkey.

Oba přístupy mají své využití a jsou široce používány, ale dále se práce zaměřuje pouze na WebView aplikaci. Po dokončení hry toho lze využít k propagaci pomocí demoverze spustitelné přímo v internetovém prohlížeči. Později bude také představen editor HTML5 her Construct 2 a běhové prostředí NW.js pro spustitelné stolní aplikace.

Mezi výhody, které tento přístup poskytuje, patří možnost distribuce her přes oficiální obchody platforem stejně jako u nativního přístupu. Dále již zmíněný lepší přístup k vybavení koncového zařízení. Kvůli využití webového jádra přetrvává nevýhoda ve slabším výkonu. Nicméně tyto technologie se neustále vyvíjejí a situace se postupně zlepšuje.

Tabulka 2.3 shrnuje výhody a nevýhody nativního, webového a hybridního přístupu. Dále budou popsány nástroje k tvorbě hybridní WebView aplikace.

	Nativní	**Webový**	**Hybridní**
Náklady	Nejvyšší, pokud vyvíjíme pro více platforem	Nejmenší, znalost jedné technologie	Jako webový + práce s hybridními nástroji
Přenositelnost	Zvlášť verze pro každou platformu	Omezena jen dostupností prohlížeče	Nástroji přidána podpora hlavních platforem
Přístup k zařízení	Dostupné celé rozhraní	Omezené, závislost na sys. a prohlížeči	Dostupné v závislosti na použitém nástroji
Konzistence uživ. rozhraní	Každá platforma má unikátní UI	Použití aplikačních rámců lze docílit nativního vzhledu	Použití aplikačních rámců lze docílit nativního vzhledu
Výkon	Vysoký	Nižší v závislosti na prohlížeči a připojení k internetu	Nižší kvůli abstrakci
Distribuce	Webové obchody, mají výhody, ale také pravidla	Žádná pravidla, ale žádné výhody obchodů	Oba způsoby, před zapouzdřením web a poté obchody
Zpeněžení	Nákupy v obchodě či hře, reklama, ale odvádění procent ze zisku obchodu	Převážně reklama	Oba způsoby, před zapouzdřením web a poté obchody

Tabulka 2.3: Výhody a nevýhody nativního, webového a hybridního přístupu k vývoji her.

Apache Cordova je knihovna s otevřeným zdrojovým kódem spravovaná v současné době firmou Apache. Jedná se o sadu aplikačních rozhraní pro přístup k možnostem mobilních zařízení (např. ukládání souborů, akcelerometr a další senzory) z jazyka JavaScript. Pracuje na principu zapouzdření HTML5 aplikace do nativní podoby, kde toto pouzdro je založeno na interním prohlížeči WebView, který tuto aplikaci zobrazuje.

Jeho součástí není vývojové prostředí, je to jen nástroj pro kompilaci. Podporované systémy a stav této podpory naleznete v dokumentaci[5] na stránkách projektu. V současné

[5]Dostupná z http://cordova.apache.org/docs/en/edge/guide_support_index.md.html

době plně podporuje systémy Android 4 a novější, iOS 6 a novější, Amazon Fire OS, až na drobnosti také Windows 8.0 a 8.1, Windows Phone 8.1 a Blackberry 10, částečně také Firefox OS a Tizen. Operační systémy se neustále vyvíjí, stejně tak se vyvíjí projekt Cordova.

Crosswalk Project je nástavba Cordovy pro Android. Crosswalk nepoužívá základní WebView dostupný na Androidu, ale jádro Chromium stejně jako prohlížeč Google Chrome. Díky tomu má lepší podporu WebGL a dalších technologií a o to větší výkon ve hrách a náročnějších aplikacích.

Intel XDK je vývojové prostředí pro tvorbu HTML5 aplikací. Dá se v něm přímo psát kód, obsahuje pokročilé emulační, testovací a ladící nástroje. Pro účely této bakalářské práce je nejdůležitější částí napojení na službu vzdáleného severu, která umožňuje vytvořenou hru převést do nativní podoby a aplikovat na ni potřebné části knihoven Cordova a Crosswalk. Díky tomu, že toto sestavení probíhá na vzdáleném serveru, nemusí mít vývojář nainstalovánu sadu nástrojů ve svém počítači. Například k sestavení aplikace pro iOS by potřeboval počítač Apple s operačním systémem OS X. Vývojář by také musel věnovat čas aktualizacím a nastavování těchto nástrojů. Vzdálený překlad tyto problémy odstraňuje. Podobné možnosti nabízí také Adobe PhoneGap.

Intel XDK také umožňuje živý náhled aplikace v mobilním telefonu přes USB či Wi-Fi bez nutnosti jejího sestavení. Do mobilního telefonu si vývojář nainstaluje aplikaci Intel App Preview, ta je poté synchronizována s vývojovým prostředím v počítači. Tato aplikace je vhodná pro základní testování. Dostupná je pro Android, iOS a Windows Phone.

Před samotným sestavením aplikace je potřeba vyplnit požadované informace, od názvu hry přes části zařízení s požadovaným přístupem (senzory, fotoaparát, souborový systém a další) až po výběr použitých zásuvných modulů Cordovy a také koncové operační systémy a další údaje. V případě testování je možné některé údaje přeskočit, ale pokud chceme aplikaci odeslat do obchodu, je potřeba vyplnit správně všechny údaje. Takto nachystaný projekt se odešle na vzdálený server, kde proběhne jeho zpracování. Poté vývojář obdrží odkaz ke stažení sestavených balíků aplikace, pro Android s koncovkou .apk, pro iOS s koncovkou .ipa a podobně. Finální balík aplikace je připraven k odeslání do internetového obchodu ke schvalovacímu procesu a k následné distribuci.

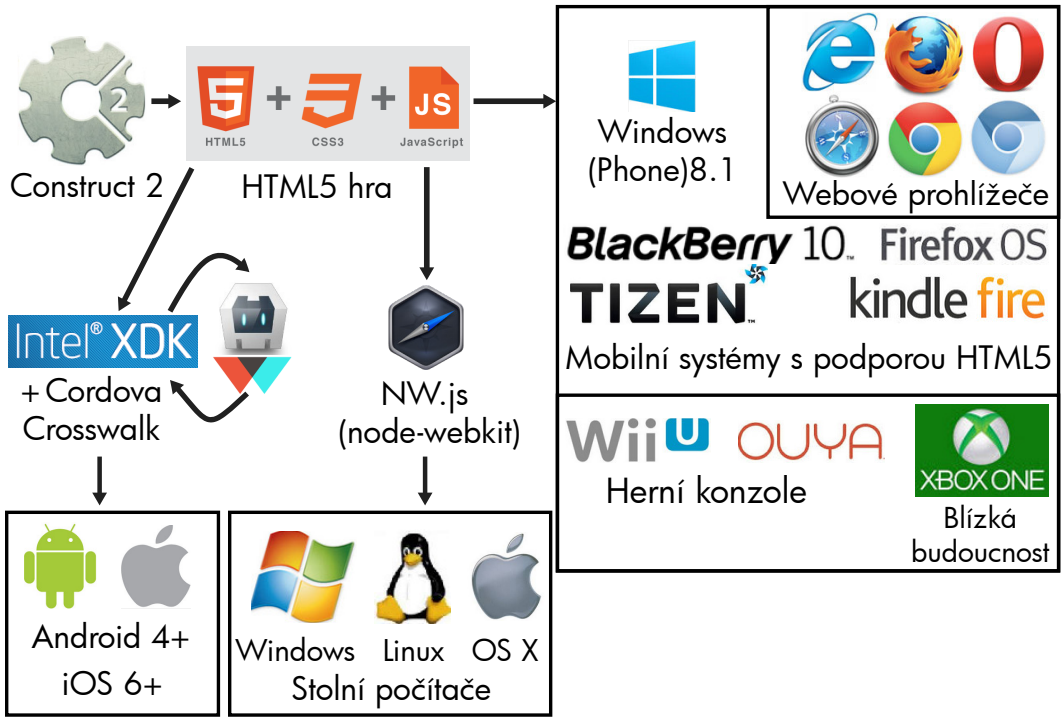
NW.js (dříve node-webkit) je řešení k vytvoření hybridní aplikace spustitelné na stolních počítačích. Doposud bylo popisováno řešení spíše pro mobilní platformy. Export pomocí NW.js umožňuje spuštění hry vytvořené pomocí webových technologií jako samostatnou desktopovou aplikaci pro Windows, OS X i Linux. Po uživateli nevyžaduje instalaci zvláštního prohlížeče. Pouze vývojář musí mít toto běhové prostředí nainstalováno.

Podobně jako Crosswalk u mobilních telefonů i NW.js používá k chodu jádro Chromium. V podstatě je tedy výkon hry podobný tomu, jako by byla spuštěna ve webovém prohlížeči Google Chrome, zároveň však vypadá jako nativní aplikace (je bez uživatelského rozhraní prohlížeče). Toto řešení je poměrně výkonné i pro náročnější aplikace jakými jsou třeba 3D hry. Navíc je takto možné zpřístupnit některé funkce, které prohlížeče neumožňují. Například přímo manipulovat se soubory v počítači (vytvářet, načítat, mazat, přejmenovat, spustit, apod.) nebo přepnout zobrazení hry do režimu celé obrazovky bez vyžádání uživatelem.

Verze hry pro stolní počítače otevírá vývojáři spoustu dalších možností k distribuci. Existuje řada internetových obchodů, které se specializují na prodej her pro počítače. Příkladem mohou být Steam, GOG, Desura, Itch.io, Humble Store a řada dalších.

Construct 2 je herní editor, jehož výstupem je HTML5 hra. Umí spolupracovat s již zmíněnými knihovnami a nástroji Intel XDK, NW.js a dalšími. Blíže popsán bude v následující podkapitole 2.5.

Výše uvedené nástroje společně tvoří jeden ze způsobů, jak udělat hru nezávislou na platformě napříč mobilními i stolními zařízeními. Dokonce lze obsáhnout i některé herní konzole podporující nativně HTML5 (Nintendo Wii U) nebo založené na systému Android (Ouya). S příchodem Windows 10 se počítá s podporou HTML5 her také pro konzole Xbox One. Postup vývoje hry a podporované platformy jsou zobrazeny na obrázku 2.3. Některé editory a nástroje mají konkurenční řešení, kterým se dají za určitých podmínek nahradit.



Obrázek 2.3: Jeden z možných postupů vývoje multiplatformní hry a cílové platformy.

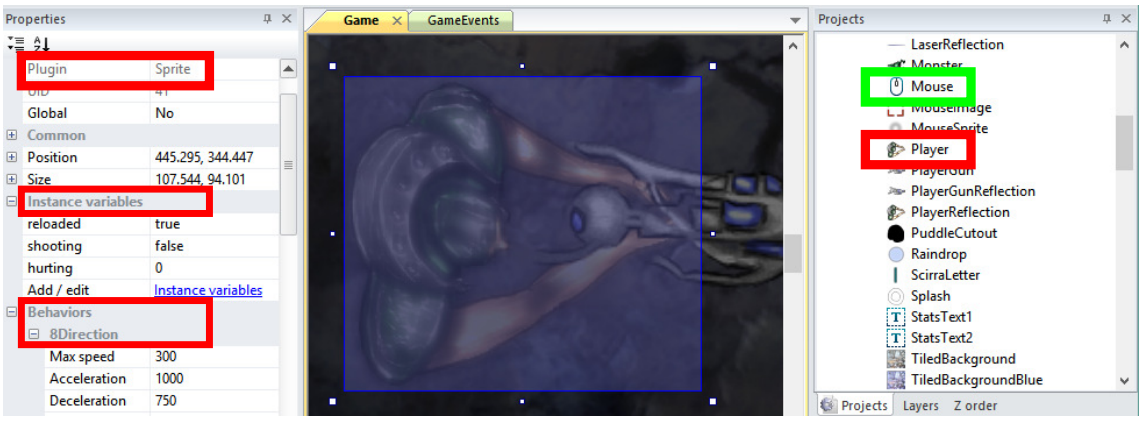
2.5 Construct 2 a jiné editory sloužící k tvorbě her

V předchozí části bylo uvedeno, jak z HTML5 aplikace vytvořit hybridní aplikaci nezávislou na platformě. Tato část je zaměřena na nástroje pro tvorbu takových HTML5 her, s kterými můžeme dále pracovat. Bude zde popsán převážně program Constuct 2[6] společnosti Scirra.

Construct 2 je herní editor specializovaný na tvorbu dvourozměrných HTML5 her. Mezi přednosti tohoto programu patří přehledné událostmi řízené programování, grafický i textový editor, sada zásuvných modulů, které řeší často se opakující herní problémy, možnost psaní vlastních modulů, vstřícná komunita a mimo jiné také cena, která může být pro začínající vývojáře důležitá. Následuje stručný popis principu programování v Construct 2.

[6] Webová stránka projektu https://www.scirra.com/construct2

Většina věcí, které lze pozorovat v Construct 2 hře, je zastoupena *objekty*. Při vkládání objektu do grafického náhledu *scény*, je mu potřeba přiřadit jeden z již vytvořených *zásuvných modulů*, podle toho získá objekt vlastnosti (např. sprite[7], zdroj světla, řádek textu, reklama, správce zvuku a mnoho dalších) a je buď viditelný nebo skrytý. Tyto zásuvné moduly se dají programovat v jazyku JavaScript. Takto je vytvořen objekt určitého typu. Při zvolení modulu sprite to může být třeba objekt typu „Player“. Vložením do scény vzniká jeho první *instance*, tu lze dále duplikovat a vytvářet další. Objektu lze přiřadit *instanční proměnné*, číselné nebo textové, jejichž hodnoty se pro každou instanci můžou lišit. Zajímavá je také možnost přiřazení chování. *Chování* přidává určitou předpřipravenou funkcionalitu. Například z něj lze učinit ovladatelný objekt, takže má vlastnosti jako maximální rychlost, zrychlení, sílu skoku, působí na něj gravitace apod. Hodnoty parametrů tohoto chování vývojář nastaví podle potřeby. Vlastní moduly chování jde také vytvořit v jazyku JavaScript. Objekty s podobnými vlastnostmi je možné sloučit do skupin pojmenovaných jako *rodina*, např. pro rodinu „Nepřátelé“ by byla společná vlastnost ta, že po zasažení hráčem přijdou o jeden život. Výřez části editoru je zachycen na obrázku 2.4.

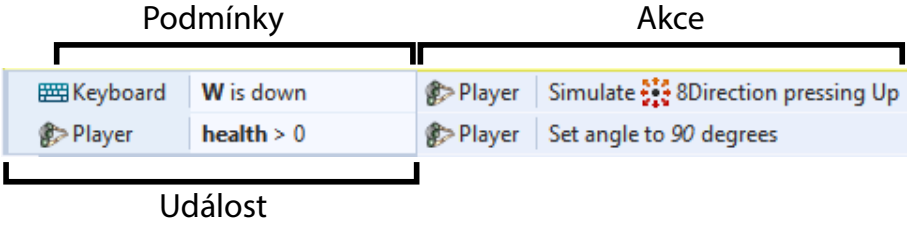


Obrázek 2.4: Výřez z programu Construct 2. Červeně je označen objekt Player. Přetažením do scény vznikla jeho instance. Objekt je založen na modulu Sprite. Má tři instanční proměnné a přiřazeno chování 8Direction, které usnadňuje jeho pohyb. Zeleně je označen jiný objekt Mouse. Ten je skrytý a zpracovává vstup z počítačové myši.

Hlavní logika hry je definována systémem *událostí* (viz obrázek 2.5). Intuitivní událostmi řízené programování umožňuje vývojáři zaměřit se na skutečně důležité problémy jeho hry. Událost zahrnuje *podmínky*, které musí nastat. Následně se spustí *akce* reagující na tuto událost. Události a akce se zapisují do strukturovaných *seznamů událostí*, ty můžou být podle potřeby logicky rozděleny. V seznamu událostí se vyskytují také globální či lokální proměnné a konstanty. Program podporuje také časové události, díky kterým je možné nějakou akci vyvolat třeba každou vteřinu nebo každý takt hry. Na dostatečně silném přístroji je provedeno 60 taktů za vteřinu, což zajišťuje plynulý chod aplikace.

Export na platformy používající HTML5 jako nativní jazyk (webové prohlížeče, Tizen, BlackBerry 10, Windows Phone) je možný přímo. Pro export na stolní počítače (Windows, OS X, Linux) musí mít vývojář nainstalováno běhové prostředí NW.js. Editor ho detekuje a použije k zapouzdření HTML5 aplikace do nativní podoby. U systémů Android a iOS je zde možnost připravení aplikace pro následný import do nástrojů Intel XDK a PhoneGap

[7]Sprite je označení používané v počítačové grafice pro dvourozměrný obrázek nebo animaci, která je integrována do větší scény.



Obrázek 2.5: Ukázka programování v Construct 2.

(oba popsány v předchozí podkapitole 2.4), kde je export na koncové platformy dokončen.

Další editory her

Mezi další rozšířené editory her patří GameMaker: Studio, Unity 5 a Unreal Engine 4. Nejpodobnější Constructu 2 je GameMaker: Studio. Oba slouží primárně k tvorbě 2D her, mají své programovací prostředí, chod hry je řízen událostmi a akcemi apod. GameMaker má navíc vlastní systém exportu na výsledné platformy, proto nemusí využívat dalších nástrojů. Za toto si ale vývojář připlatí, kompletní GameMaker: Studio stojí 640 €, oproti tomu Construct 2 vyjde levněji na 330 €. Oba poskytují nekomerční zkušební demo verzi zdarma.

Unity 5 a Unreal Engine 4 jsou komplexnější nástroje zaměřené na tvorbu 3D her používané i profesionálními firmami. Také mají vlastní vývojové prostředí i některé stejné principy a funkce, kromě JavaScriptu umožňují i programování v jazycích C++ a C#, ale kvůli větší složitosti jsou vhodné spíše pro týmy než jednotlivce. Tomuto odpovídá i cena, která se pohybuje od zkušební verze zdarma až po tisíce eur podle zahrnutých funkcí.

Kapitola 3

Simulace pružných těles

Tato kapitola se věnuje problematice počítačové simulace pružných těles. V první části bude čtenář seznámen s několika typy deformace objektů především s pružnou neboli elastickou deformací. Druhá část se věnuje numerickým integračním metodám, které budou použity k řešení diferenciálních rovnic a postoupení v simulaci hry dál o určitý časový krok. Poslední dvě části jsou věnovány dvěma rozdílným modelům pružných těles a jejich modifikacím. První model je založen na fyzikálních vlastnostech pružin, druhý i na vlastnostech tlaku.

3.1 Pružné a jiné deformace těles

V reálném světě se nevyskytují pouze pevná tělesa, ale také měkká třeba lidské a zvířecí orgány a tkáně nebo neživé objekty jako textil, gel a kapalina. Výzkum simulace měkkých neboli deformovatelných těles má v počítačové grafice dlouholetou historii. Díky dnešním výkonným počítačům se používají k realistické animaci postav v počítačových hrách a filmech, ale třeba také pro odbornou přípravu chirurgů.

V mechanice termín *deformovatelný objekt* odkazuje na objekt, jehož tvar může být změněn v důsledku působení sil, mezi které patří tah, tlak, ohyb a další. Jak je uvedeno v technické zprávě zabývající se tímto tématem [7], deformace se dělí v závislosti na reakci materiálu a působících sil do tří skupin:

- **Pružná** či **elastická** deformace (malá deformace) je vratná. Tvar objektu je deformován dočasně, dokud je aplikována síla, jakmile je síla odstraněna, vrací se do svého původního tvaru. Třeba těleso z pryže má velký rozsah pružné deformace, hedvábná tkanina má mírný rozsah a krystal nemá téměř žádný rozsah pružné deformace.

- **Plastická** deformace (mírná deformace) není vratná. Tvar objektu je deformován, dokud je aplikována síla, jakmile je síla odstraněna, vrací se do původního tvaru jen částečně nebo vůbec. Třeba stříbrné a gumové předměty mohou být nataženy do své původní délky, ale po deformaci nemůžou zcela obnovit svůj původní tvar.

- **Lomová** deformace (velká deformace) není vratná, ale od plastické se liší. Objekt je deformován trvale, pokud je nevratně ohnut, natržen nebo roztržen, když materiál přesáhne hraniční mez pružné deformace. Všechny materiály mohou mít lomovou deformaci, pokud je aplikována dostatečná síla.

Pružná tělesa patří do podskupiny měkkých deformovatelných těles. Jedná se o dynamické objekty, které v reakci na vnější a vnitřní síly mění výrazně tvar a udržují si konstantní objem. Po deformaci obnovují svůj původní tvar. Pružný objekt pro dynamickou simulaci, má typicky jednovrstvý elastický povrch s různým obsahem uvnitř. Za dobu výzkumů v této oblasti byla vytvořena široká škála modelů založených na fyzikálních zákonech od využití vlastností pružin až po vývoj modelů na bázi kapalin či tlaku (viz podkapitoly 3.4 a 3.5).

Jedna z možností využití pružných těles ve hrách je vytvoření postavičky hovorově označované jako „Blob“, jehož chování se dá přirovnat k balónku naplněném vodou. Na obrázku 3.1 jsou zobrazeny dva příklady takových her a simulace kusu látky dopadajícího na kouli.

Obrázek 3.1: Ukázka z her Locoroco a Gish používající pružné objekty jako herní postavy a simulace kusu pružné látky dopadajícího na kouli.

Fyzikální simulace pružných těles je formulována jako v čase proměnná parciální diferenciální rovnice, jejíž diskretizací získáme numericky řešitelnou běžnou diferenciální rovnici. K aplikování tohoto poznatku musí být spojitý objekt (pryž, textil apod.) diskretizován vytvořením abstrakce tohoto objektu v podobě *bodů* (particles) rozložených po povrchu tělesa. Informace o rozložení hmotnosti objektu je obsažena v těchto bodech. Tyto body jsou vzájemně propojeny *pružinami* (springs) s různými vlastnostmi, které lze upravovat pro dosažení požadovaného chování.

Na jednotlivé body poté působí v různých směrech *síly* (forces). Takto je získána samotná diferenciální rovnice, kterou je dále potřeba vyřešit a pokročit tak v simulaci dál o určitý časový krok. K tomuto výpočtu lze použít některou *numerickou integrační metodu*, jejímž použitím se získá ze vstupního stavu systému nový výstupní stav v novém čase.

3.2 Pohybové rovnice

Při simulaci pružného objektu se vychází ze znalosti polohy a rychlosti všech bodů, ze kterých se tento objekt skládá. Na tyto body působí vnější i vnitřní síly, ty se na základě druhého Newtonova pohybového zákona (rovnice 3.1) promítnou do výsledného zrychlení. Z upravené rovnice 3.2 lze vyvodit, že čím více objekt váží, tím méně se zrychlí ze stejného množství síly, které obdržel. Síla a zrychlení jsou vektory, proto jsou zvýrazněny tučně.

$$\mathbf{F} = m \cdot \mathbf{a} \tag{3.1}$$

$$\mathbf{a} = \frac{\mathbf{F}}{m} \tag{3.2}$$

Zrychlení je velikost změny rychlosti za určitý čas neboli zrychlení je derivace rychlosti (rovnice 3.3). Obdobně rychlost je velikost změny polohy v čase neboli rychlost je derivace polohy (rovnice 3.4).

$$\mathbf{a} = \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} \tag{3.3}$$

$$\mathbf{v} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \tag{3.4}$$

Z toho vyplývá, že pokud bude známa aktuální poloha a rychlost objektu, působící síly a hmotnost, je možné integrací najít novou polohu a rychlost v určitém budoucím okamžiku. Tyto hodnoty jsou pak aktuální po určitý časový úsek. Postupuje se po tzv. *časovém kroku* (integrační krok). Zdrojem informací k těmto rovnicím a následujícím integračním metodám byla kniha Jamese M. Vertha [8] zaměřená na aplikaci matematiky ve hrách a znalosti ze studijní opory předmětu Modelování a simulace [5].

3.3 Numerické integrační metody

Pro výpočet nové polohy a rychlosti je potřeba znát:

1. Počáteční podmínky — polohu x_0 a rychlost v_0 v počátečním čase t_0.

$$x_0 = x(t_0) \tag{3.5}$$

$$v_0 = v(t_0) \tag{3.6}$$

2. Tvar řešení diferenciálních rovnic. Funkce f a g vypočítají hodnotu zrychlení a rychlosti v čase t. Cílem však je pomocí současné hodnoty zrychlení získat novou rychlost (3.9) a pomocí současné hodnoty rychlosti novou polohu (3.10). Proto je nezbytné aproximovat řešení pomocí numerické integrace.

$$f(t, v) = v' \tag{3.7}$$

$$g(t, x) = x' \tag{3.8}$$

$$v(T) = v_0 + \int_0^T f(t, v)\,\mathrm{d}t \tag{3.9}$$

$$x(T) = x_0 + \int_0^T g(t, x)\,\mathrm{d}t \tag{3.10}$$

3. Integrační kroky h neboli intervaly mezi diskrétními časovými body $t_0, t_1, \ldots, t_n$, v kterých je řešení aproximováno pro zpracování počítačem. Pro plynulé hraní her by měla být hodnota h rovna alespoň $\frac{1}{30}$ sekundy lépe však $\frac{1}{60}$ sekundy, což je přibližně 16 ms. Za hodnotu pro plynulou grafiku, kdy lidské oko již nerozezná jednotlivé obrazy, se totiž považuje 30 snímků za vteřinu, avšak u her náročnějších na rychlé reakce je možné pozorovat znatelný rozdíl mezi 30 a 60 snímky za vteřinu[1].

$$h_i = t_{i+1} - t_i \tag{3.11}$$

[1] Převzato z
http://www.technologyx.com/featured/understanding-frame-rate-look-truth-behind-30v60-fps/.

Integrační metody lze rozdělit podle:

- řádu metody, tedy podle počtu použitých členů Taylorova rozvoje.
- počtu hodnot z předchozích kroků, které používají pro výpočet nové hodnoty na *jednokrokové* a *vícekrokové*. Na začátku výpočtu vícekrokové metody často nemají potřebné inicializační hodnoty. Pro výpočet těchto prvních kroků se obvykle používá některá z jednokrokových metod.
- toho jestli počítají s pozicí a rychlostí sousedních bodů na *explicitní* a *implicitní*. Explicitní metody jsou nezávislé a vycházejí pouze z předcházejícího stavu systému, na základě kterého vypočítají nový stav. Po každém integračním kroku vzniká odchylka od správné pozice bodu, což se snaží systém v dalším kroku vyvážit. To postupně vede k nadměrným úpravám a zvyšování energie až za maximální hranici. Systém se zhroutí. K zabránění hromadění energie se někdy používá *tlumící síla* (damping). Implicitní metody počítají nové polohy bodů s ohledem na sousední body. Místo nabírání energie ji ztrácejí až do klidového stavu, takže není potřeba použít tlumící sílu. Díky tomu jsou stabilnější.

Eulerova metoda

Tato metoda je jednou z nejjednodušších. V základní podobě je explicitní, vychází z prvních dvou členů Taylorova rozvoje a je jednokroková, tedy počítá výsledek pouze s využitím derivace výchozího (předchozího) kroku. Vzorec pro výpočet hodnoty na konci kroku:

$$y(t+h) = y(t) + hf(t, y(t)) + \mathcal{O}(h^2) \tag{3.12}$$

kde $f(t, y(t))$ je hodnota derivace na začátku kroku a funkce $\mathcal{O}$ značí velikost rozdílu skutečné hodnoty oproti aproximaci. Výpočet nových hodnot rychlosti a polohy bodu se provede se znalostí předchozích hodnot následovně:

$$v_{n+1} = v_n + ha \tag{3.13}$$
$$x_{n+1} = x_n + hv_n \tag{3.14}$$

Metoda je jednoduchá, ale trpí několika problémy. Hlavní problém metody je, že celková chyba $\mathcal{O}(h^2N)$ vzrůstá závisle na druhé mocnině velikosti kroku h a na počtu kroků N. Takže zmenšení kroku o faktor 2 má lokálně za následek zmenšení chyby o faktor 4, ale k dosažení stejného pokroku v simulaci musí být proveden dvojnásobný počet kroků. Z toho vyplývá, že celková chyba Eulerovi metody je přímo závislá na velikosti časového kroku neboli $\mathcal{O}(h)$. Zvýšení přesnosti se tedy dá dosáhnout zmenšením časového kroku. Nicméně bez ohledu na to, jak moc se krok sníží, vždy zde chyba bude a poroste v průběhu času.

Dalším problémem je značná nestabilita při překročení určité velikosti kroku. Tento problém lze částečně vyřešit použitím *semi-implicitní Eulerovy metody*. Původní metoda se změní pouze v místě, kde je získána hodnota rychlosti. Nejprve se přičte zrychlení k rychlosti a až poté se aktualizuje pozice pomocí této nové rychlosti.

$$v_{n+1} = v_n + ha \tag{3.15}$$
$$x_{n+1} = x_n + hv_{n+1} \tag{3.16}$$

Takto dostává integrátor jiné vlastnosti, je více stabilní a energie se zachovává po více kroků, avšak zůstává stále stejně nepřesná.

I přes zmíněnou chybovost lze najít případy, kde se tato metoda používá. Pro jednoduché simulace jako 2D hry obsahující tuhá tělesa, vysoký poměr snímků za vteřinu a poměrně malá zrychlení, bude pravděpodobně Eulerova metoda vhodná. U složitějších simulací se spíše používají metody vyšších řádů nebo vícekrokové.

Verletova metoda

Tato metoda je zajímavou volbou pro svou dobrou stabilitu a malé nároky na paměť při simulaci velkého počtu částic. V tomto případě integrátor nepožaduje ukládání rychlosti každé částice, protože novou rychlost určí z nynější a předchozí polohy bodu. Jedná se o vícekrokovou metodu. Lze jí snadno detekovat kolize částic, a pokud existují požadavky na vzdálenosti mezi nimi (např. nějaká částice vybíhá ze skupiny příliš daleko), dá se pozice snadno korigovat. Je také v čase neměnná, což znamená, že simulace může běžet dopředu a poté pozpátku a skončí na počátečním místě. Vzorce pro Verletovu metodu jsou získány sčítáním Taylorova rozvoje pro nový a předešlý krok, pak po vyjádření nového stavu získáme:

$$y(t+h) \quad = \quad 2y(t) - y(t-h) + h^2 y''(t) + \mathcal{O}(h^4) \tag{3.17}$$

Výpočet nové pozice se pak provede následovně:

$$x_{n+1} \quad = \quad 2x_n - x_{n-1} + h^2 a \tag{3.18}$$

Z rovnice vyplývá, že se znalostí předcházející polohy x_{n-1} a současné polohy x_n částice a jejího zrychlení a, je možné zjistit její novou polohu. Pro získání hodnoty prvního kroku můžeme použít Eulerovu metodu nebo upravenou Verletovu metodu počítající s rychlostí.

Lokální chyba je $\mathcal{O}(h^4)$ a vychází obdobně jako Eulerova metoda z míry použití Taylorova polynomu. Ze čtyř použitých členů se dva odečtou. Celková chyba metody je pak $\mathcal{O}(h^2)$.

Pro výpočet pozice není potřebná rychlost, takže se počítá o jeden údaj méně. Z tohoto důvodu se často používá v částicových systémech, které obvykle nejsou závislé na rychlosti. Pokud je však třeba aplikovat sílu tření na základě rychlosti nebo zpracovat točivé síly pevných objektů, tak chybějící údaj o rychlosti použití metody stěžuje. Tento problém lze částečně řešit použitím *Leapfrog Verletovi metody*, která vypočítá rychlosti v půli kroku, z kterých je se zpožděním jednoho kroku za pozicí získána požadovaná rychlost. Další možností je použití *Rychlostní Verletovy metody*, která počítá dvě hodnoty akcelerace, jednu na začátku intervalu a druhou na konci, výsledná rychlost opět není úplně přesná.

Zatímco pro částicové systémy (model textilie) Verletova metoda funguje dobře, pro pevná tělesa z důvodu obtížně získávaného údaje o rychlosti není příliš vhodná.

Metody Runge-Kutta

Mezi další běžné numerické integrační metody patří metody *Runge-Kutta* druhého a čtvrtého řádu a jiné varianty, které při výpočtu provádí další výpočty uvnitř kroku a tím dosahují při stejné délce kroku podstatně přesnějších výsledků než Eulerova metoda. Díky tomu také dovoluje používat velké časové kroky.

Výpočet RK2 používá jeden pomocný bod uprostřed kroku, jde o metodu druhého řádu:

$$k_1 = hf(t, y(t)) \tag{3.19}$$
$$k_2 = hf(t + \frac{h}{2}, y(t) + \frac{k_1}{2}) \tag{3.20}$$
$$y(t+h) = y(t) + k_2 \tag{3.21}$$

Celková chyba $\mathcal{O}(h^2)$ je závislá na druhé mocnině velikosti integračního kroku, což je pro hodnoty h menší než 1 lepší než Eulerova metoda. Standardní metoda s chybou $\mathcal{O}(h^4)$ je známa jako Runge-Kutta čtvrtého řádu. Aby se chyba udržovala ve stanovených mezích, je možné metody doplnit o výpočet odhadu chyby metody, což dovoluje průběžně měnit délku kroku. Díky tomu si při použití většího kroku metoda sama rozhodne o jeho zkrácení.

Tyto metody jsou náročnější na výkon, třeba RK4 potřebuje pro získání výsledku čtyřikrát vyhodnotit derivaci. Vývojář musí zvážit, zdali možnost použití delších časových kroků vyváží tyto zvýšené nároky nebo jestli se raději uchýlí k použití méně náročné metody.

3.4 Model Spring-mass a jeho modifikace

Fyzikálně založené simulace jsou v herní produkci stále důležitější. Poskytují hráči prostředky k umocnění jeho zážitku ze hry. Příkladem budiž zmiňované hry Locoroco a Gish (obrázek 3.1), které staví herní zážitek především na elastické 2D postavičce označované *Blob* (chováním se blíží kapce vody, balónku naplněném vodou nebo želatině). Vzhledem k tomu, že důraz je kladen především na hratelnost, má fyzika těchto her jen malý vztah k realitě. Následující řádky popisují modely vhodné pro tvorbu právě takového Bloba.

Obě hry používají model *spring-mass* (dále jen SM). Jak je uvedeno v článku Micka Westa [9], tento model je založen na skupině *hmotných bodů* rozmístěných a propojených *pružinami* podle požadovaného tvaru objektu.

Základní *hmotný bod* má tyto tři parametry:

1. Polohu ve 2D prostoru,
2. vektor rychlosti,
3. hmotnost (všechny body objektu většinou stejnou).

Navíc jsou zde *síly* působící na bod jmenovitě gravitace, odpor vzduchu, tření, síly vycházející z vlastností pružin a síly vzniklé srážkou s jiným objektem nebo jinak. Aktualizace polohy bodů je prováděna výpočty pomocí numerických integračních metod popsaných v předchozí podkapitole 3.3.

Základní *pružina* propojuje dva body a má následující čtyři parametry:

1. Délka pružiny v klidu (což je délka pružiny, když není ani natažena, ani stlačena),
2. minimální délka pružiny (když je plně stlačena),
3. maximalní délka pružiny (když je plně natažena),
4. síla vyvinutá pružinou, která je úměrná vychýlení z klidové délky.

Hookův zákon

Pružiny vyvíjejí v závislosti na tom, zda a jak moc jsou stlačené nebo natažené tomu úměrné síly. Toto chování popisuje *Hookův zákon*:

$$\mathbf{F} = -k \cdot \Delta\mathbf{r} \tag{3.22}$$
$$\Delta\mathbf{r} = \mathbf{r} - \mathbf{r}_0 \tag{3.23}$$

Kde $\Delta\mathbf{r}$ je vektor vychýlení konce pružiny z klidové polohy a k je kladná konstanta popisující tuhost pružiny. Větší hodnota k znamená větší tuhost a tím pádem menší natažení na jednotku síly. Hodnota $\Delta\mathbf{r}$ se získá rozdílem délky roztažené nebo stlačené pružiny $\mathbf{r}$ a délky v klidovém stavu $\mathbf{r}_0$. Síla $\mathbf{F}$ působí opačným směrem k síle, která pružinu stlačila nebo natáhla. Pokud platí $\mathbf{r} < \mathbf{r}_0$, pružina je stlačená a bude se snažit natáhnout — síla $\mathbf{F}$ je kladná. Pokud $\mathbf{r} > \mathbf{r}_0$ pružina je natažená a bude se snažit stlačit — síla $\mathbf{F}$ je záporná. Jinak pružina sílu negeneruje.

V této podobě nejsou pružné síly dobře použitelné. K realistické simulaci je potřeba přidat ještě *tlumení* (damping). Tlumení simuluje ztrátu energie a používá se k ujištění, že pružiny nebudou oscilovat navždy. Tlumící síla ovlivní původní rovnici následovně:

$$\mathbf{F} = -k \cdot \Delta\mathbf{r} - b \cdot \Delta\mathbf{v} \tag{3.24}$$

Kde b je součinitel tlumení a $\Delta\mathbf{v}$ je rozdíl rychlostí prvního a druhého bodu spojených touto pružinou. Větší hodnota b zvyšuje množství tlumení, takže pružina přejde rychleji do klidového stavu. Konstanty k a b jsou důležité pro nastavování konečných vlastností objektu.

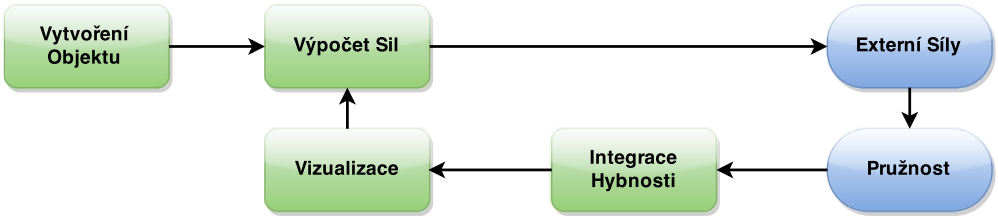
Jak uvádí Glenn Fiedler [2] ani tato rovnice ještě není plně vyhovující. Tyto dva body spojené pružinou by skončily na sobě. Je tedy potřeba rozšířit rovnici o vzdálenost odstupu:

$$\mathbf{F} = -k \cdot (|\Delta\mathbf{r}| - d) \cdot \frac{\Delta\mathbf{r}}{|\Delta\mathbf{r}|} - b \cdot \Delta\mathbf{v} \tag{3.25}$$

Zde je $|\Delta\mathbf{r}|$ aktuální vzdálenost dvou bodů, d je požadovaná vzdálenost odstupu bodů a $\frac{\Delta\mathbf{r}}{|\Delta\mathbf{r}|}$ je jednotkový vektor směřující z bodu a k bodu b při aplikaci sil směrem k bodu a a naopak. Výsledkem je, že v případě kdy jsou dva body spojené pružinou od sebe dále než na vzdálenost d, tak jsou tlačeny k sobě a naopak.

Algoritmus Spring-mass

Algoritmus SM má následující průběh (obrázek 3.2). V prvním kroku proběhne inicializace objektu, tedy rozmístění bodů a propojení pružinami. Je na vývojáři, jaký tvar napodobí. Pro tvar Bloba půjde o parametrické znázornění dvourozměrné kružnice. V druhém kroku proběhne výpočet všech sil v modelu. Síly budou počítány pro všechny hmotné body objektu. Prvně proběhne aplikace vnějších sil (gravitace, odpor vzduchu, tření, apod.) a poté vnitřních sil objektu, které generují pružiny. Pak následuje integrace hybnosti zvolenou integrační metodou (podkapitola 3.3) s využitím druhého Newtonova pohybového zákona (podkapitola 3.2) a získání nových pozic bodů. Nakonec přichází vizualizace pružného tělesa na obrazovku. Poté se algoritmus opakuje od výpočtu nových hodnot sil.



Obrázek 3.2: Průběh algoritmu Spring-mass.

Jednovrstvý Blob

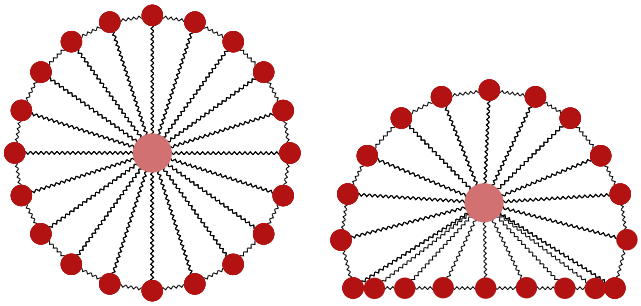
Vzhled i chování Bloba závisí na uspořádání pružin, jejich délkách a pružnosti a také na rozložení bodů a jejich hmotnosti. Získání požadovaného chování často stojí mnoho pokusů a omylů v nastavování těchto vlastností. Dále budou uvedeny některé z často používaných konstrukcí. Různé konstrukce mají své silné a slabé stránky.

Přirozený tvar kapky vody, když není vystavena vnějším silám, je koule. Blob je tedy tvořen soustavou bodů rovnoměrně rozložených po obvodu kruhu propojených pružinami. Působení gravitace poté vytváří jeho specifický tvar, který lze pozorovat na obrázku 3.3.

U jednovrstvé verze Bloba je každý bod po obvodu propojen se sousedem po levé a pravé straně jakož i s centrálním bodem uprostřed. Verze na obrázku má po obvodu 20 bodů tvořících pravidelný N-úhelník. Délky pružin přirozeně vyplývají z délek stran tohoto N-úhelníku. Tuto jednoduchou konstrukci lze realizovat s poměrně rozumnými nároky na výkon. Nicméně jednoduchost této konstrukce přináší také několik problémů.

Jednovrstvý obal se snadno zachytává a skládá přes sebe tím, že se body překřižují a nevracejí na původní místo. Špatné nastavení tuhosti pružin může způsobit úplné zhroucení do sebe. Celkově je vyvažování parametrů citlivé, Blob je buď moc tuhý, nebo příliš vratký.

Jediné výhody tohoto rozložení jsou tedy jeho jednoduchost a nízká náročnost na výkon, jinak je velmi nestabilní a chybové. Ve hře nejspíše nebude působit graficky příliš dobře.



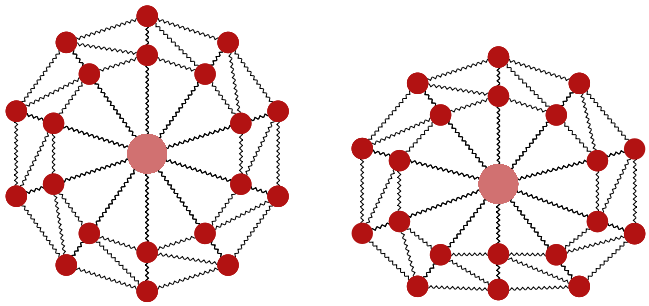
Obrázek 3.3: Jednovrstvý Blob v klidu a při působení gravitace.

Dvouvrstvý Blob

Mnohem více stabilní řešení poskytuje dvouvrstvé rozložení, které navrhl Mick West [9] a k vidění je na obrázku 3.4. Vnitřní vrstva je stejná jako u předchozího jednovrstvého řešení a obklopuje ji navíc vnější vrstva. Tyto dvě vrstvy spolu tvoří soustředné kružnice — mají různé poloměry a stejný střed. Jsou propojeny pružinami, které mají poměrně vysokou tuhost a vedou klikatě.

Při ladění vlastností tohoto rozložení je potřeba mimo jiné správně vyvážit počet bodů, sílu pružin a tloušťku vrstvy. Správným nastavením lze docílit stabilního a realistického chování. Díky tomu, že Blob drží lépe tvar je jeho pohyb více klouzavý, tedy není tak přilnavý k povrchu a pohybuje se podobně jako rtuť.

Mezi výhody tohoto rozložení patří stabilita a plynulost pohybu. Ale vzhledem ke složité struktuře je zapotřebí více času na vyladění chování. Také nároky na výkon prudce narůstají a velký Blob může simulaci velmi zpomalit, což může být problém pro plynulost hry.

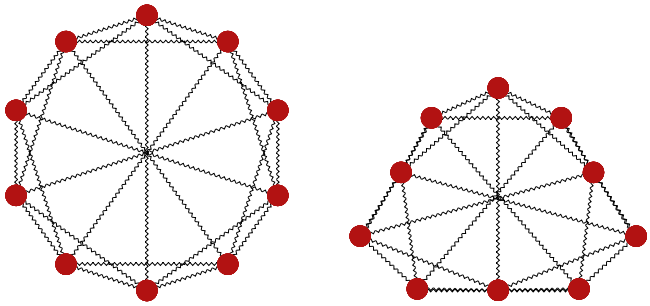


Obrázek 3.4: Dvouvrstvý Blob v klidu a při působení gravitace.

Blob bez centrálního bodu

Tato struktura na rozdíl od předešlých nemá hmotný středový bod (obrázek 3.5). Nedostatek vnitřní opory je vyvážen tím, že každý bod je spojen s nejbližšími dvěma sousedy nalevo i napravo a také s protilehlým bodem. Tyto dodatečné pružiny zajišťují dostatečnou stabilitu. Je nepravděpodobné, že dojde ke zhroucení do sebe.

Struktura je sice stabilnější než jednovrstvá a je méně náročná na výkon než dvouvrstvá, ale tvar Blobu je jiný než u předchozích typů. V čele a na bocích je více vypouklý, což může nebo nemusí být žádoucí. Tuto strukturu používá již zmíněná hra Gish (obrázek 3.1 uprostřed). Na druhou stranu je ve hrách častěji k vidění klasický tvar.

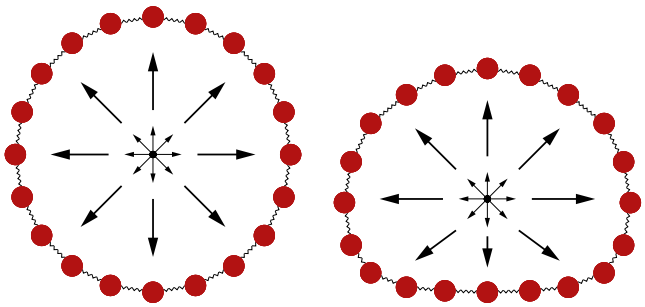


Obrázek 3.5: Blob bez centrálního bodu v klidu a při působení gravitace.

3.5 Model Pressure soft body

Předpokladem k pochopení modelu Pressure soft body (dále jen PSB) je znalost modelu Spring-mass (předchozí podkapitola), jelikož i tento model používá po obvodu hmotné body a pružiny. Tento model jako první popsal Maciej Matyka [4].

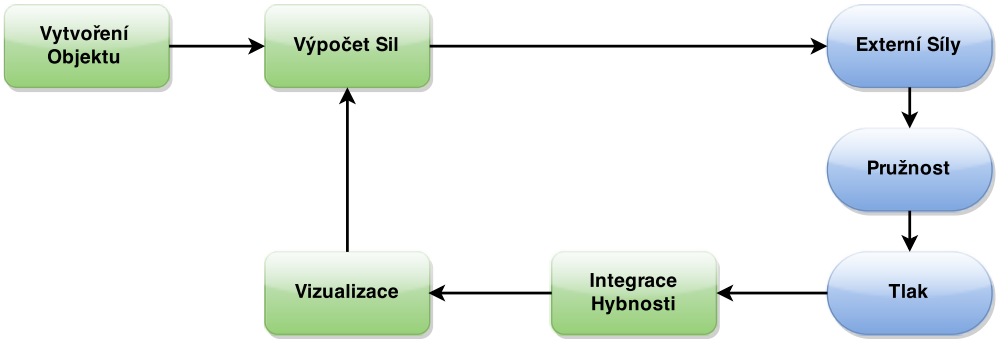
Oproti předchozím Blobům se tento odlišuje jeho vnitřním složením (obrázek 3.6). Místo pružin je tvar udržován vnitřním tlakem, jako u balónku naplněného plynem. Podle nastavení síla tlaku drží tvar více či méně konstantní.



Obrázek 3.6: Blob s vnitřním tlakem v klidu a při působení gravitace.

Algoritmus Pressure soft body

Algoritmus PSB má následující průběh (obrázek 3.7). Oproti SM se liší jen ve výpočtu sil. Po aplikaci externích sil a pružnosti probíhá ještě aplikace síly vnitřního tlaku.



Obrázek 3.7: Průběh algoritmu Pressure soft body.

Výpočet síly tlaku

Rovnice výpočtu síly vnitřního tlaku je následující:

$$\mathbf{F_P} = \frac{1}{V} \cdot A \cdot P \cdot \hat{\mathbf{n}} \tag{3.26}$$

kde $\mathbf{F_P}$ je vektor síly tlaku, V je celkový objem tělesa, A je část povrchu tělesa na který tlak působí (ve 2D délka hrany mezi dvěma body), P je skalární hodnota tlaku získaná experimenty a zvolená tak, aby vyhovovala účelům hry, $\hat{\mathbf{n}}$ je normálový vektor povrchu (ve 2D hrany) tělesa, v jehož směru tlak působí.

Výpočet objemu je proveden na základě Gaussova teorému. Integrál přes objem je u uzavřených objektů možné nahradit integrálem přes povrch tělesa. Důležitá je pak samotná rovnice pro výpočet objemu (u dvourozměrného tělesa mluvíme o obsahu, rovnice je stejná):

$$V = \sum_{i=1}^{NUMS} \frac{1}{2}|x_1 - x_2| \cdot n_x \cdot dl \tag{3.27}$$

kde V je celkový objem tělesa, $|x_1 - x_2|$ je absolutní hodnota rozdílu x-ové složky počátečního a koncového bodu pružiny, n_x je hodnota x-ové složky normálového vektoru a nakonec dl je délka pružiny. Provede se suma přes všechny pružiny (od $i = 1$ po $NUMS$), kde pružiny v tomto modelu tvoří hranu objektu.

Výše uvedený postup umožňuje poměrně snadný výpočet tlakových sil na základě objemu tělesa a dalších údajů. Ve spojení se silami pružin vzniká stabilní pružné těleso.

Kapitola 4

Návrh hry

Tato kapitola je věnována návrhu samotné hry a vybraným postupům, které budou zvoleny k její realizaci. První část se věnuje postupu, který bude zvolen k dosažení nezávislosti hry na cílové platformě. Druhá část je věnována principům navrhované hry. Třetí část pak popisuje mnou navrženou úpravu jednovrstvé verze modelu pružnosti. Závěrečná část se krátce věnuje návrhu uživatelského rozhraní a ovládání hry.

4.1 Navrhovaný postup při realizaci hry nezávislé na platformě

Vzhledem k získaným poznatkům ohledně nativního, webového a hybridního přístupu k programování, kterým je věnována podkapitola 2.4, jsem se rozhodl použít hybridní přístup. Takto ušetřím čas potřebný k naprogramování hry pro každou platformu zvlášť a zároveň budu moci využívat výhody internetových obchodů. Tvorba WebView hybridní aplikace umožňuje použití HTML5 a díky tomu i specializované herní editory, které toto rozhraní podporují. Vliv použití WebView na výkonost výsledné aplikace budu poté testovat. Zmiňovaná podkapitola obsahuje podrobné informace o programech, které budu používat. Výsledný postup bude téměř stejný jako ten na obrázku 2.3 (na konci podkapitoly 2.4).

Nejdříve naprogramuji HTML5 hru v herním editoru Construct 2 (popsán v podkapitole 2.5). Zde také použiji zásuvné moduly tohoto editoru, které obsahují funkce pro komunikaci s rozhraním knihovny Cordova. Nicméně samotný Construct 2 umí vyexportovat pouze HTML5 aplikaci, která ještě není zapouzdřená. Pro operační systémy Android a iOS, které nepoužívají HTML5 jako nativní jazyk, je toto finální zapouzdření a sestavení provedeno v aplikaci Intel XDK. Tento program obsahuje formulářového průvodce sestavením aplikace, kde vývojář vyplní požadované informace, certifikáty a náhledové ikony a projekt se poté odešle na vzdálený server, kde proběhne jeho sestavení. Vývojář poté v řádu jednotek minut obdrží na email odkaz ke stažení sestavené aplikace. K získání potřebných certifikátů pro dané platformy a také k distribuci her je potřeba být registrovaným vývojářem. K vytvoření desktopových aplikací použiji NW.js, které si Construct 2 sám najde, pokud je nainstalováno v počítači a umožní tak export aplikace na stolní počítače. Nakonec z Construct 2 vyexportuji univerzální Windows 8.1/Windows Phone 8.1 balík. Jelikož Windows (Phone) 8.1 aplikace podporují HTML5, není potřeba je zabalit pomocí Intel XDK. Místo toho se otevřou jako projekt ve Visual Studio 2013 a novější. Ve Visual Studiu poté probíhá podobná certifikace a sestavování jako v Intel XDK. Výsledkem jsou Windows 8.1 a Windows Phone 8.1 aplikace připravené k odeslání do obchodu či testování na testovacích zařízeních.

Jelikož mají zbývající podporované platformy jen minimální tržní podíl a navíc jsou v Česku těžce sehnatelné nebo zatím úplně nedostupné, nebudu se jim dále věnovat.

4.2 Základní principy a mechaniky hry

Základní koncept mnou navržené hry vychází z jiné hry jménem *Elastomania*. V této hře byl hráč v roli motorkáře, jehož motorka měla velmi pružné tlumiče a dva hmotné body v podobě kol. Hráč těchto vlastností motorky, pružnosti a hmotných bodů, využíval k překonávání překážek a procházení úrovněmi. Ve mnou navržené hře bude hráč také využívat tohoto principu, jenže hmotné body budou rozestavěny okolo centrálního bodu a vzájemně propojeny tak, aby tvořili Blob podle některého z modelů popsaných v podkapitolách 3.4 a 3.5. Výsledná implementace všech modelů je k vidění na obrázku 5.1 či v přílohách B, kde jsou obrázky, které zachycují hru v různých stádiích. Dalo by se tedy říci, že jde o kombinaci her *Elastomania* a *Gish*.

Ve hře se vyskytuje robot, kterého hráč ovládá a má vlastnosti Blobu. Hráčovým cílem je dostat se k druhému robotovi, který stojí na místě. Po kolizi těchto dvou robotů úspěšně končí úroveň a okamžitě začíná další. Dále se ve hře vyskytují fyzikální objekty různých tvarů, které mají tři druhy vlastností. Černé objekty jsou statické a nelze s nimi nijak pohnout. Bílé objekty jsou dynamické a reagují na fyzikální události prostředí, jako je gravitace nebo kolize s hráčem či jinými objekty. Červené objekty mohou být buď statické jako černé objekty nebo dynamické podobně jako bíle. Pokud však hráč dojde do kontaktu s takovým červeným tělesem úroveň skončí neúspěchem a restartuje se.

Restart hry způsobí, že se hráčovi zvětší počet jeho smrtí o 1. Dále se měří čas, který hráč strávil hraním úrovní a zaznamenává se kolikátou úroveň z kolika hráč hraje. Pokud dohraje poslední úroveň, jeho současné výsledky se porovnají s nejlepšími dříve získanými výsledky a pokud jsou lepší, tak se uloží jako nové nejlepší, jinak se zahodí.

V případě zařízeních s operačním systémem Android a iOS zde přichází na řadu služby příslušných obchodů. Nejlepší skóre a údaj o dokončení hry se v případě Androidu odešle do Google Play Game Services a v případě iOS do služby Game Center.

Návrh počítá také se hrou dvou hráčů. Kde si hráči pomáhají při zdolávání překážek a dosažení cíle. Oba hráči mají identického robota a odlišují se jen barevně.

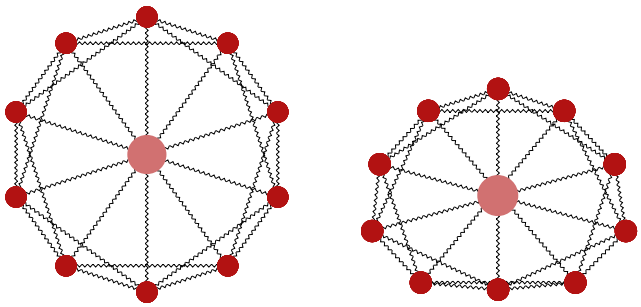
Toto je obecný návrh principů a herních mechanik mnou vyvíjené hry. O implementaci se toho dozvíte více v následující kapitole 5.

4.3 Předpoklady a navrhovaná úprava modelu pružnosti

Z informací, které jsem získal při studiu modelů pružnosti (podkapitoly 3.4 a 3.5) usuzuji, že ani jeden z popsaných modelů nebude plně vhodný pro použití v navrhované hře. Jednovrstvý model Spring-mass bude nejspíše velmi nestabilní. Dvouvrstvý zbytečně složitý a těžkopádný, navíc jeho druhá řada bude ve hře spíše na obtíž. Třetí modifikace, která je bez centrálního bodu, bude mít nevyhovující tvar. Navíc pro hru požaduji středový bod. Kvůli tomuto vylučuji i tlakový algoritmus PSB.

Rozhodl jsem se tedy zkusit zpevnit jednovrstvý Blob tak, že každému hmotnému bodu po obvodu přidám další dvě pružiny. Jednu spojenou s v pořadí druhým sousedem nalevo a druhou spojenou s druhým sousedem napravo. Takto vznikne jednovrstvý Blob s dvojitým spojením. Předpokládám, že požadavky na výkon se příliš nezvýší. Zato se zlepší stabilita jednovrstvého modelu. Nákres výsledné úpravy je vyobrazen na obrázku 4.1.

Uvedené předpoklady dále v textu potvrzují testy (více v podkapitole 5.2).



Obrázek 4.1: Jednovrstvý blob s dvojitým spojením v klidu a při působení gravitace.

4.4 Uživatelské rozhraní a ovládání

Hra je navržena tak, aby byla co nejjednodušší. Obsahuje jen dvě herní obrazovky. První obsahuje prvky hlavní nabídky (obrázek 4.2). Druhá obrazovka vyobrazuje samotné herní úrovně (příloha B). Uživatelské rozhraní obrazovky s úrovněmi je vždy stejné. Avšak rozhraní hlavního menu se bude měnit v závislosti na koncové platformě. Např. tlačítko pro přepínání režimu celé obrazovky je na mobilních telefonech zbytečné, protože hra bude vždy přes celou obrazovku. Také se budou měnit odkazy na cílové obchody v levém dolním rohu a tlačítko přihlášení k službám třetích stran.

Ovládání postavy hráče je také velmi snadné. Umožněn je pouze pohyb doleva, doprava a skok. Toho půjde snadno docílit na klávesnici, herním ovladači (gamepad) i dotykovém displeji. Na dotykovém displeji bude pohyb doleva proveden držení prstu v levé dolní čtvrtině obrazovky, pohyb doprava v pravé dolní čtvrtině a skok zmáčknutím kdekoliv v horní polovině displeje. V případě dvou hráčů bude na dotykovém displeji ovládání prvního hráče natlačeno do levé poloviny a pro druhého hráče do pravé poloviny obrazovky. U klávesnice se bude první hráč ovládat šipkami a druhý klávesami WASD. Druhý připojený ovladač hra také detekuje.

Obrázek 4.2: Obrazovka hlavního menu.

Kapitola 5

Implementace a testování hry

V této kapitole se věnuji praktické implementaci a testování výsledné hry, která byla navržena v předchozí části.

5.1 Implementace návrhu v herním editoru Construct 2

Stěžejní roli v implementované hře hrají fyzikální objekty. Mým cílem však není tvorba kompletního fyzikálního jádra. V práci se zaměřuji především na pružná tělesa a multiplatformitu. Předností editoru Construct 2 je, že obsahuje předefinovaná chování, které jde aplikovat na objekty. Jedním z těchto chování je i *Physics*. Přidělením tohoto chování objektu na něj začnou působit fyzikální síly okolí. Takovou silou může být např. gravitace, tření či výsledná síla při srážce s jiným takovým objektem. Toto chování tedy aplikuji na všechny fyzikální objekty ve hře.

V implementaci používám také několik předefinovaných zásuvných modulů, které umožňují vykonání pokročilých úkonů. Důležitými moduly jsou třeba *Touch* pro zpracování dotyků na displeji, *Keyboard* zajišťuje vstup z klávesnice a *Gamepad* pro zpracování vstupu z herního ovladače. Dále také *Localstorage* k ukládání dat na pevný disk příslušného zařízení. Takto ukládám údaje o pokroku hráče ve hře. Zásuvný modul *Audio* umožňuje práci se zvukem a hudbou.

Podle hodnoty globální proměnné, do které ukládám identifikátor cílové platformy, se automaticky přizpůsobuje vzhled uživatelského rozhraní. Převážně se takto změní výsledné odkazy do herních obchodů a zobrazení nebo skrytí některých nepotřebných tlačítek.

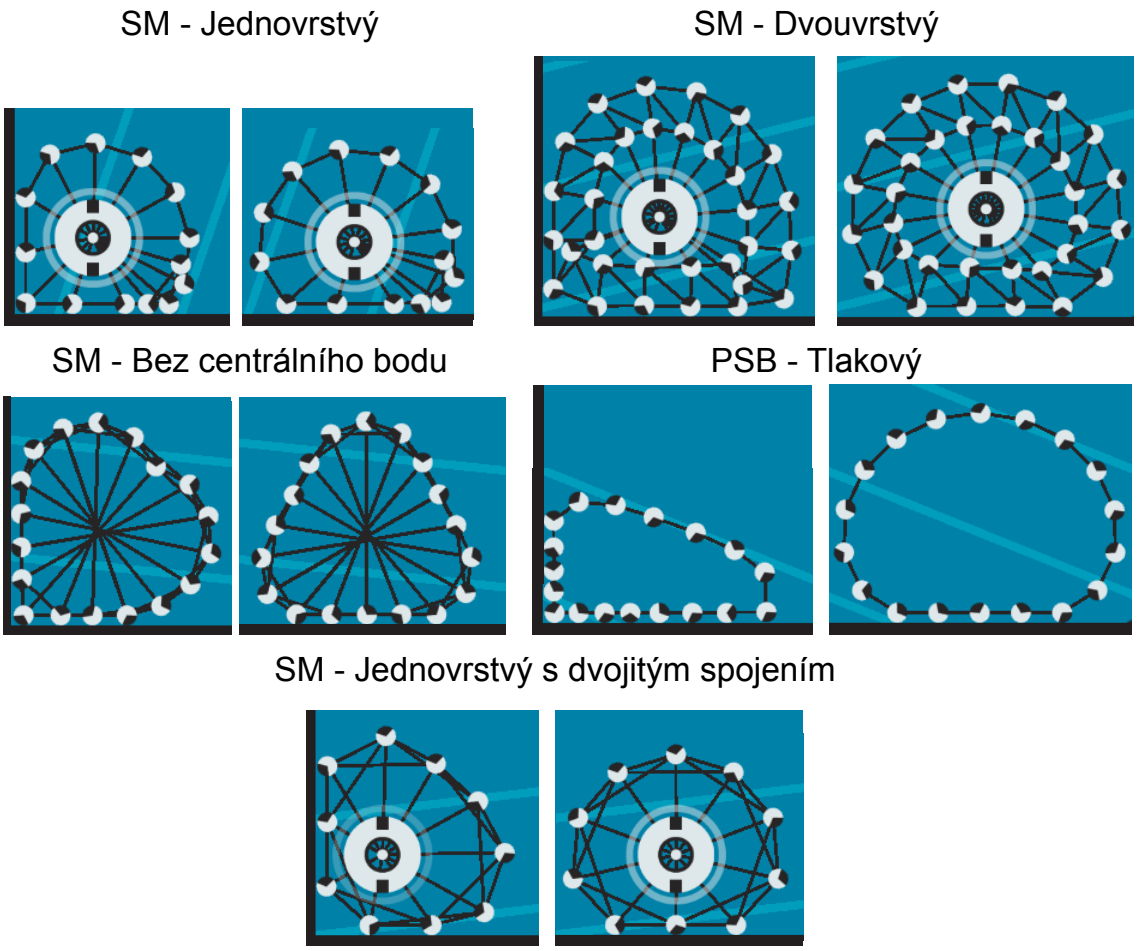
5.2 Implementace a výběr nejlépe vyhovujícího algoritmu modelů pružnosti

Jak již bylo naznačeno na hlavní bod i vedlejší body jsem v editoru aplikoval chování *Physics*. Díky tomu jsem se mohl plně věnovat algoritmům zajišťující pružnost a tlak.

Síly aplikované pružinami jsem počítal pomocí vzorce 3.25 pro Hookův zákon. V tomto vzorci jsem však mohl vynechat výpočet součinitele tlumení, protože fyzikální chování editoru používá semi-implicitní Eulerovu metodu doplněnou také o tlumící síly. Ve vzorci jsem tedy odečetl současnou vzdálenost bodů od požadované počáteční vzdálenosti a toto vynásobil tuhostí pružin tak, jak je ve vzorci uvedeno. Výslednou sílu jsem aplikoval v požadovaném směru od bodů či k nim.

K výpočtu tlaku podle vzorce 3.26 bylo nejprve potřeba aplikovat rovnici pro výpočet celkového obsahu (objemu) tělesa 3.27. Ve schématu průběhu algoritmu Pressure soft body 3.7 můžeme vidět, že se první aplikují síly pružin. V cyklu při procházení všech bodů a aplikování pružných sil toho využívám a postupně počítám celkový obsah. Po skončení cyklu navazuje další, ve kterém se na každý bod aplikují síly tlakové s použitím dříve vypočtené velikosti obsahu a zvolené konstanty tlaku.

Postupně jsem experimentoval s různým nastavením tuhosti pružin a s počtem bodů a jejich vzdáleností od sebe a také s tlakovou konstantou. Stabilní výsledky jsou k vidění na obrázku 5.1.



Obrázek 5.1: Všechny implementované modely v klidu a při nárazu do rohové zdi.

Výběr nejlépe vyhovujícího modelu je ovlivněn požadavky, které na hru kladu. Je možné, že jiné hře by lépe vyhovoval jiný model. Testováním jsem došel k tomu, že moje prvotní předpoklady z podkapitoly 4.3 byly správné.

Tlakový model PSB se vyznačuje největší možností deformace ze všech testovaných modelů. Po zdeformování se vždy vrátil do původního stavu. Jeho pohyb a chování je nejvíce podobné tekutině. Nicméně mu chybí centrální bod, kvůli čemuž se pro mě v tomto projektu stává nepoužitelným do finálního nasazení. Ze stejného důvodu nemůžu použít ani modifikaci modelu SM bez centrálního bodu. Dvouvrstvý model je příliš těžkopádný a složitý na

nastavování, navíc jeho druhá řada se pro účely hry také nehodí. Jednovrstvý model je zase příliš nestabilní.

Nakonec jsem tedy implementoval navrhovanou modifikaci z již zmíněné podkapitoly. Tím je jednovrstvý model s dvojitým spojením, který se napojuje na dva sousedy zleva i zprava. Testováním jsem si ověřil, že pružiny navíc nemají téměř žádný vliv na výkon, přičemž z nestabilního objektu udělaly stabilní. Proto jsem tuto variantu zvolil jako konečnou.

5.3 Přidání služeb třetích stran

Služby třetích stran funkčních pro Android verze 4 a novější, iOS 7 a novější a Windows Phone 8.1 lze zajistit pomocí dvou zásuvných modulů.

První zásuvný modul se nazývá *IAP*, což značí in-app purchase neboli platbu za reálné peníze přímo ve hře a podporují ho všechny tři platformy. Aby bylo možné platbu uskutečnit, musí být vývojář zaregistrovaný v příslušných internetových obchodech daných platforem. Poté zde tyto platby nastaví. V editoru Construct 2 nastaví parametry zásuvného modulu tak, že do nich uloží identifikátory těchto plateb. Nakonec nastaví vyvolání těchto plateb např. kliknutím na určité tlačítko.

Zásuvný modul zajišťující herní služby se nazývá *Phonegap Game*. Podobně jako u plateb i tyto služby je potřeba nastavit ve vývojářském účtu programátora. Je zde možné vytvořit žebříček dosaženého bodového skóre nebo zaznamenávat dosažené úspěchy (např. dokončení hry, zabití tisíce nepřátel a další). Poté vývojář musí i tomuto zásuvnému modulu nastavit atributy obsahující identifikátory těchto tabulek.

V přílohách B je k nahlédnutí několik obrázků z mnou vyvíjené hry, které zachycují, jak takové platby a herní služby vypadají.

5.4 Export hry pro dostupné platformy a testování

Postup při výsledném exportu hry byl již popsán v části předchozí kapitoly 4.1, která pojednává o navrhovaném postupu při realizaci multiplatformní hry. Tento postup jsem nakonec uplatnil. Obrázky z následujících testů jsou ke shlédnutí v příloze B.

Testování hry v internetovém prohlížeči *Chrome* a *Firefox* a také testování klasické *desktopové* a *Windows 8.1 Modern UI* aplikace probíhalo na počítači s operačním systémem Windows 8.1, jehož hardware obsahoval 8 GB paměti RAM, procesor Intel Core Duo o taktu 3 GHz a grafickou kartu NVIDIA GTX 460. Hra ve všech těchto případech běžela na tomto počítači naprosto plynule a počet snímků za vteřinu neklesal pod 60. Plynule běžela i hra pro dva hráče. Úspěšný byl i test herního ovladače, který byl také funkční.

Test hry na operačním systému *OS X Yosemite* proběhl na MacBook Pro (verze 2010), který je osazen procesorem Intel Core i5 o taktu 2,4 GHz, obsahuje 4 GB paměti RAM a grafickou kartu NVIDIA GT 330M. Také v tomto případě neklesal počet zobrazených snímků za vteřinu pod 60.

Testování verze pro *Android* probíhalo na tomto operačním systému ve verzi 4.4.2 na zařízení Lenovo A328 (půl roku starý model). Tento mobilní telefon patří do levnější cenové kategorie. Nicméně i on si se hrou poměrně dobře poradil. Počet snímků za vteřinu se držel přibližně na 40. Což je stále více než 30, takže hra se pohybovala plynule, avšak oproti stolním počítačům trochu pomaleji. V příloze je také možno vidět úspěšné přihlášení do služby Google Play Game Services (B.5) a započatý nákup uvnitř aplikace (B.6).

S o trochu horším výsledkem dopadl test hry na zařízení Nokia Lumia 810 (dva roky starý model) s operačním systémem *Windows Phone 8.1*. Zde byl počet snímků přibližně 30 za vteřinu. Hra byla stále plynulá, nicméně viditelně pomalejší než v předchozích testech. Na obrázku v příloze je zachycena započatá platba uvnitř aplikace.

Nejhůře skončil test mobilního telefonu iPhone 4 (pět let starý model) se systémem *iOS 7*. Hra byla prakticky nehratelná. Běžela rychlostí jen 7 snímků za sekundu. Navíc se vykreslovala pouze na výšku, takže byl po pravé straně černý pruh zabírající asi čtvrtinu plochy. Tento model je již značně zastaralý. U novějších verzí je vylepšeno zobrazovací jádro WebView, proto předpokládám, že na novějších zařízeních by měla být hra plynulejší. Nový iPhone 6 používá WKWebView, který může v některých případech způsobit několikanásobný nárůst výkonu[1].

Z výsledku testů lze vyvodit, že výkoné stolní počítače a notebooky si s hybridní hrou lehce poradí. U mobilních telefonů byla na zařízení starém pět let hra prakticky nehratelná, na zařízení starém dva roky plynulá, ale poměrně pomalá a na novějším zařízení starém jen půl roku plynulá a rychlá. To můžeme přisuzovat jak zvětšujícímu se výkonu mobilních telefonů, tak vzrůstající oblibě HTML5 a hybridního programování. Díky tomuto zájmu je snahou vývojářů těchto operačních systému postupně vylepšovat interní prohlížeč WebView.

[1] Informace převzata z
http://developer.telerik.com/featured/why-ios-8s-wkwebview-is-a-big-deal-for-hybrid-development.

Kapitola 6

Závěr

Tato práce byla věnována vývoji multiplatformní hry využívající pružná tělesa. Cílem bylo nalezení a otestování postupu k vytvoření takové hry a následná implementace modelů pružnosti a jejich zhodnocení.

Studiu procesu vývoje multiplatformních her je věnována kapitola 2, která popisuje problematiku od základních pojmů až po některé herní editory a další nástroje umožňující vývoj hry nezávislé na platformě. K tomuto účelu byl poté zvolen program Construct 2 umožňující tvorbu HTML5 her. Dále byla hra kompilována nástrojem Intel XDK pro platformy Android a iOS. Hru se takto podařilo zprovoznit a otestovat na sedmi platformách a to jak stolních (Windows 7 a 8.1, Mac OS X, Linux), tak mobilních (Android 4, iOS 7, Windows Phone 8.1). Hra na mobilních systémech také úspěšně podporuje některé pokročilé funkce, kterými jsou platby uvnitř aplikace a žebříčky hráčových úspěchů.

Jelikož vývojáři stačí znalost jednoho programovacího jazyka a celý projekt je uložen v jednom zdrojovém kódu, bylo dosaženo rapidního úbytku času potřebného na vývoj. Záporné vlastnosti se týkají především úbytku výkonu na starších zařízeních se zastaralou verzí WebView, který hybridní aplikace používají. Nicméně výkon hybridních aplikací se s každou další verzí operačních systému zlepšuje, proto do budoucna hodlám nadále sledovat a testovat situaci kolem hybridních aplikací. Také testování kvůli odlišným operačním systémům a přístrojům bylo poměrně složité. K některým platformám jsem se při testování nedostal, což bych chtěl v nejbližší době napravit a otestovat projekt třeba na novém iPhone 6 s operačním systémem iOS 8 a mobilních telefonech značky BlackBerry.

O simulaci pružných těles pojednává kapitola 3. Popisuji v ní typy deformace těles, numerické integrační metody potřebné k pochopení průběhu simulace a pak dva hlavní modely a jejich modifikace, které používám pro simulaci elastické herní postavičky. Model Spring-mass využívá pouze vlastností pružin. Jeho jednovrstvou modifikaci jsem shledal jako nestabilní, dvouvrstvou jako složitou a těžkopádnou, model bez centrálního bodu měl pro navrhovanou hru nevyhovující tvar. Navrhl jsem tedy úpravu jednovrstvého modelu a přidal každému bodu po obvodu tělesa dvě pružiny pro spojení s dalšími sousedy nalevo i napravo. Tímto bylo dosaženo zvýšení stability objektu za cenu nepatrného snížení výkonu. Druhým modelem byl Pressure soft body, který k silám pružin přidává ještě sílu vnitřního tlaku. Vyznačoval se největší pružnou deformací z testovaných modelů. Po zdeformování se vždy vrátil do původního stavu. Nyní se však nehodil do mnou vyvíjené hry. Nicméně v budoucnu bych se k němu chtěl vrátit a vytvořit hru, která bude využívat jeho vlastnosti.

Při tvorbě tohoto projektu mě stály také mnoho času různé neprogramátorské záležitosti. Zmínit můžu například registraci a údržbu vývojářských účtů pro různé obchody, kvůli kterým jsem si založil i živnost nebo tvorbu grafických a zvukových podkladů.

Spojením zmíněných vlastností vznikla multiplatformní hra plynule hratelná na novějších herních zařízeních, která podporuje i některé pokročilé funkce daných platforem. Využívá pružnost hráčem ovládané postavy jako stěžejní herní mechaniku, s jejíž použitím hráč zdolává překážky. Hra podporuje různorodé ovládání podle cílové platformy, jako je dotykové ovládání, kombinace klávesnice a myš či herní ovladač. Také podporuje možnost kooperace dvou hráčů. To vše z jednoho zdrojového souboru. Hra však ještě potřebuje dodatečné testování a další rozšiřování, aby mohl být naplno využit její potenciál.

Literatura

[1] BERNARD, B.: MVC a další prezentační vzory. *Zdroják*, 2009, naposled editováno 15.05.2009 [cit. 13.10.2014], dostupné z: http://www.zdrojak.cz/serialy/mvc-a-dalsi-prezentacni-vzory/.

[2] FIEDLER, G.: Spring Physics. *Gaffer on Games*, 2006, [cit. 21.03.2015], dostupné z: http://gafferongames.com/game-physics/spring-physics/.

[3] LOVELL, N.: *How To Publish a Game*. 1005 Gravenstein Highway North, Sebastopol, CA: GAMESbrief, první vydání, 2010, s. 13–20.

[4] MATYKA, M.: How To Implement a Pressure Soft Body Model. Technická zpráva, University of Wroclaw, Vratislav, Polská republika, March 2004, dostupné z: http://panoramix.ift.uni.wroc.pl/~maq/soft2d/howtosoftbody.pdf.

[5] PERINGER, P.: Modelování a simulace IMS: Studijní opora. online, rev. 12.12.2012 [cit. 28.01.2015], s. 64–70.

[6] RUDOLPH, P.: Hybrid Mobile Apps: Providing A Native Experience With Web Technologies. *Smashing Magazine*, 2014, naposled editováno 21.10.2014 [cit. 20.11.2014], dostupné z: http://www.smashingmagazine.com/2014/10/21/providing-a-native-experience-with-web-technologies/.

[7] SONG, M.; GROGONO, P.: A Framework for Dynamic Deformation of Uniform Elastic Two-Layer 2D and 3D Objects in OpenGL. In *Proceedings of the 2008 C3S2E conference on - C3S2E*, New York, USA: ACM Press, May 2008, ISBN 978-1-605-58101-9, s. 145–158, doi:10.1145/1370256.1370282, dostupné z: http://portal.acm.org/citation.cfm?doid=1370256.1370282.

[8] VERTH, J. M.; BISHOP, L. M.: *Essential Mathematics for Games and Interactive Applications: A programmer's Guide*. Burlington: MA: Morgan Kaufmann Publishers, druhé vydání, 2008, ISBN 978-0-123-74297-1, s. 601–621.

[9] WEST, M.: Blob Physics. *Cowboy Programming*, 2007, naposled editováno 05.01.2007 [cit. 11.02.2015], dostupné z: http://www.cowboyprogramming.com/2007/01/05/blob-physics/.

Příloha A

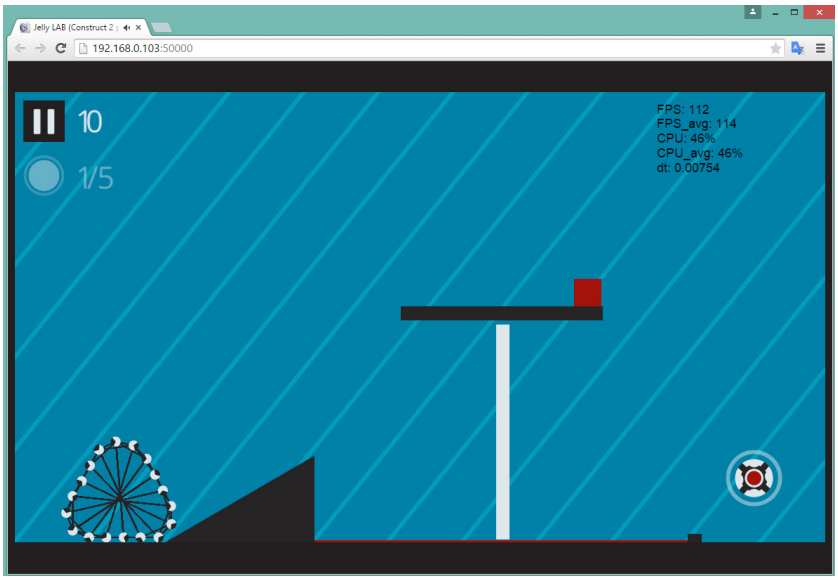
Obsah CD

- **Zdrojové soubory projektu** - K otevření projektu, který se nachází v této složce, je potřeba mít v počítači se systémem Windows 7, 8 nebo 8.1 nainstalován herní editor Construct 2 se zakoupenou osobní licencí (Personal License). Editor je ke stažení na stránkách https://www.scirra.com/. Podrobný manuál k editoru se nachází zde https://www.scirra.com/manual/1/construct-2. Použití tohoto nástroje je také popisováno v písemné části této práce.
- **Technická zpráva** - Zde se nachází písemná část bakalářské práce ve formátu pdf včetně zdrojových souborů ve formátu tex.
- **Ukázkové obrázky ze hry** - Obsahuje obrázky ze hry spuštěné na různých platformách.
- **Ukázkové video ze hry** - Obsahuje sestřih videí ze hry spuštěné na různých platformách.
- **Spustitelný soubor pro Android 4** - Ve složce se nachází soubor s koncovkou apk. Jelikož je Android otevřená platforma, podporuje jednoduchou instalaci aplikací. Stačí soubor přetáhnout do paměti telefonu či na paměťovou kartu a nainstalovat.
- **Spustitelný soubor pro iOS 7** - iOS je uzavřenější platforma a bez zásahu do operačního systému neumožňuje instalaci aplikací jinak než z autorizovaného obchodu App Store. Případně je instalace tohoto souboru možná na vývojářem autorizovaném zařízení, které musí on sám přidat do svého vývojářského účtu.
- **Spustitelný soubor pro Linux** - Hra se zde nachází ve verzích pro 32 bitový a 64 bitový systém. Hra se spouští přes konzoly příkazem ./jellylab ve složce se hrou.
- **Spustitelný soubor pro Mac OS X** - Aplikace se zde nachází ve verzích pro 32 bitový a 64 bitový systém. Hra je spustitelná běžným dvojklikem na jellylab.app.
- **Spustitelný soubor pro Webové prohlížeče** - Zde je podmínkou ke spuštění mít nainstalovaný některý běžně používaný internetový prohlížeč (Chrome, Firefox, apod.). Pokud je hra spuštěna přímo v počítači, tak ji internetové prohlížeče většinou blokují, jakožto důsledek ochrany proti nežádoucím útokům. Proto je potřeba hru nejprve nahrát na vzdálený webový server a až z něj hru spouštět.

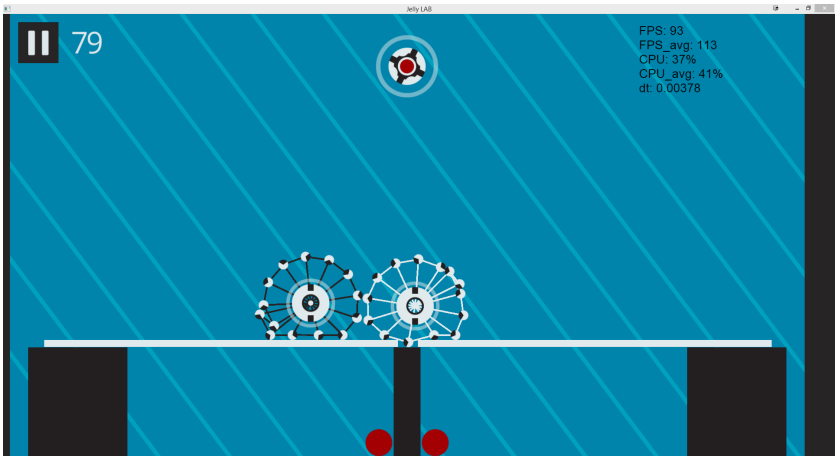
- **Spustitelný soubor pro Windows 7** - Aplikace se zde nachází ve verzích pro 32 bitový a 64 bitový systém. Hra je spustitelná běžným dvojklikem na jellylab.exe. Jedná se o běžnou desktopovou aplikaci spustitelnou na Windows 7, 8 a 8.1.
- **Spustitelný soubor pro Windows 8.1** - Tato verze je spustitelná na Windows 8.1 v prostřední Modern UI (dříve Metro). Instaluje se spuštěním souboru Add-AppDevPackage.ps1 v programu PowerShell. Stačí na Add-AppDevPackage.ps1 kliknout pravým tlačítkem myši a zvolit možnost „Run with PowerShell“, takto se hra nainstaluje a lze ji poté spustit.
- **Spustitelný soubor pro Windows Phone 8.1** - K nainstalování této verze je třeba mít registrovaný mobilní telefon jako vývojářský, čímž se zpřístupňuje možnost instalovat aplikace jinak než z obchodu Windows Store. Hru ve formátu appx lze na mobil poté nainstalovat přes aplikaci Windows Phone Application Deployment 8.1.

Příloha B

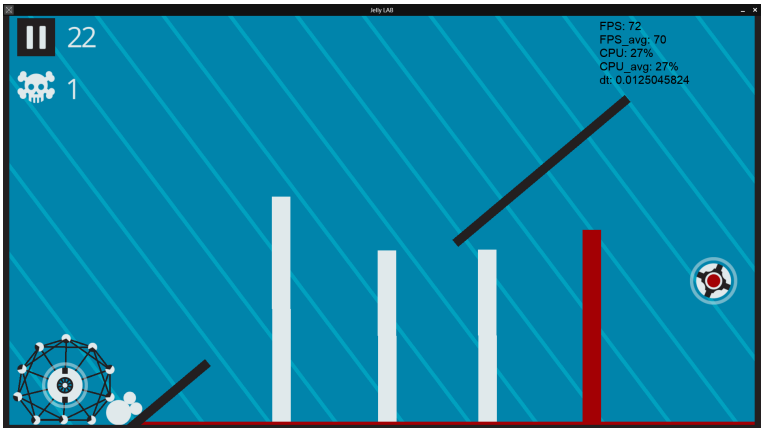
Ukázky ze hry na reálných zařízeních



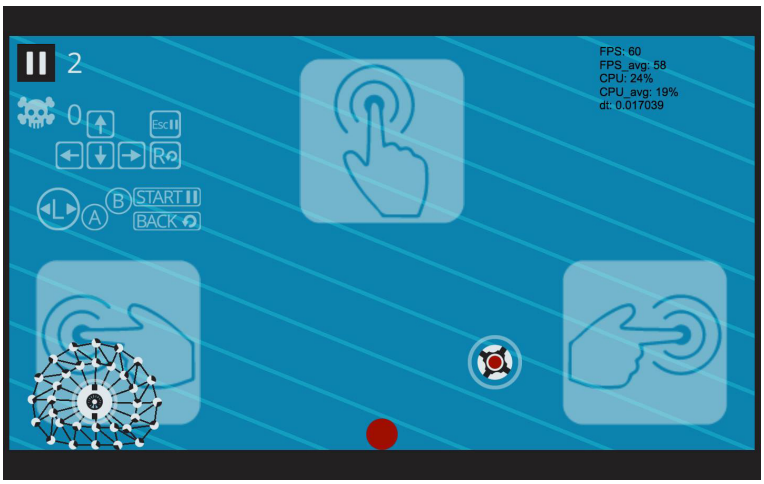
Obrázek B.1: Ukázka hry spuštěné ve webovém prohlížeči Chrome.



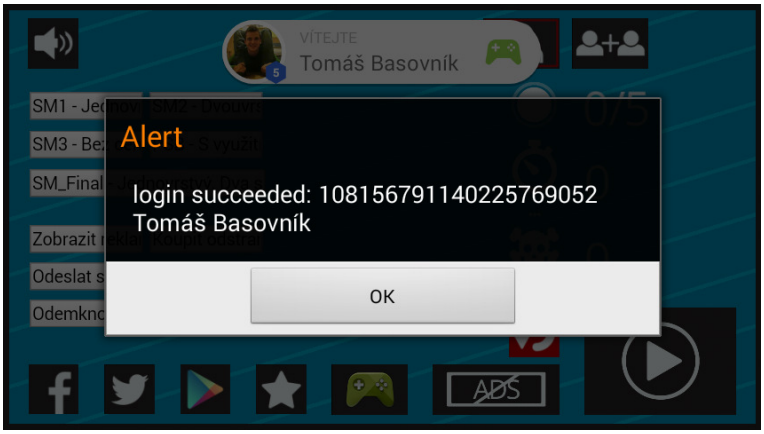
Obrázek B.2: Ukázka hry spuštěné na Windows 8.1 jako klasická spustitelná aplikace.



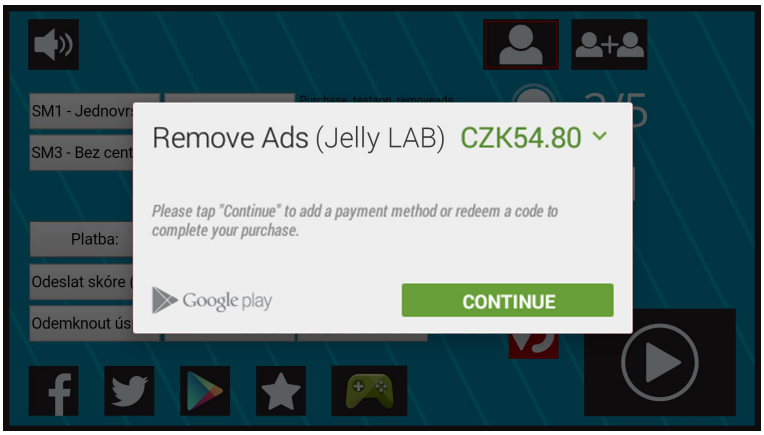
Obrázek B.3: Ukázka hry spuštěné na Windows 8.1 jako Modern UI (dříve Metro) aplikace.



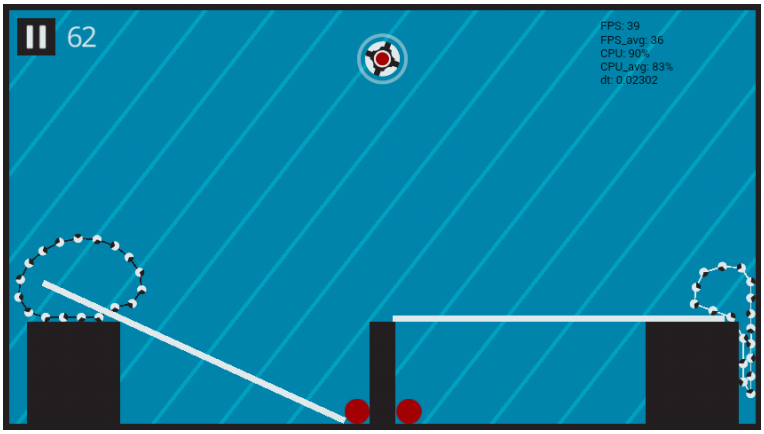
Obrázek B.4: Ukázka hry spuštěné na MacBook Pro s operačním systémem OS X Yosemite.



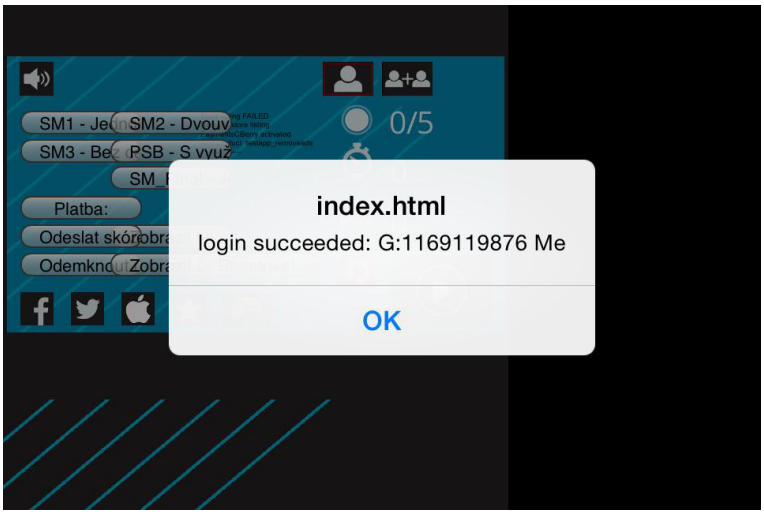
Obrázek B.5: Ukázka hry spuštěné na mobilním telefonu Lenovo A328 s operačním systémem Android 4.4.2. Obrázek zachycuje úspěšné přihlášení do služby Google Play Game Services.



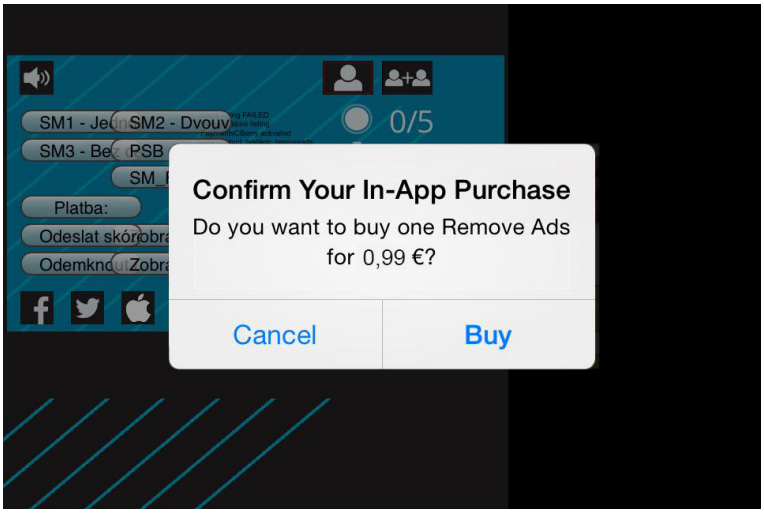
Obrázek B.6: Ukázka hry spuštěné na mobilním telefonu Lenovo A328 s operačním systémem Android 4.4.2. Obrázek zachycuje nákup položky uvnitř aplikace.



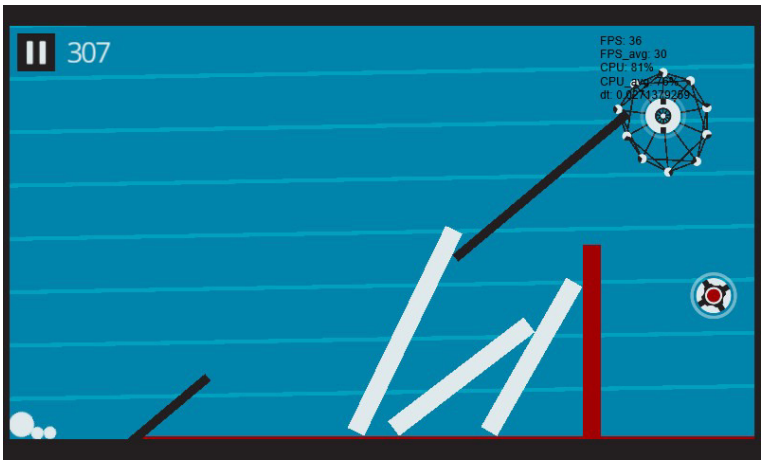
Obrázek B.7: Ukázka hry spuštěné na mobilním telefonu Samsung Galaxy S4 s operačním systémem Android 4.4.



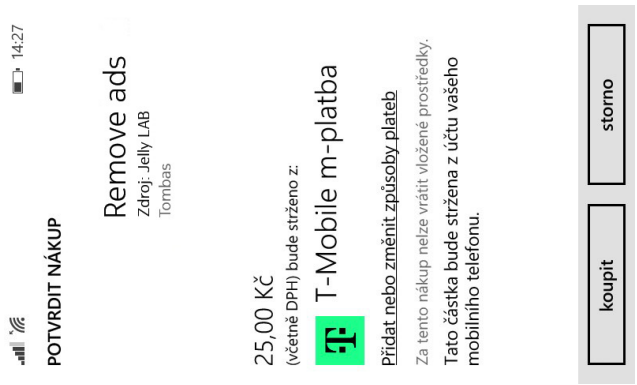
Obrázek B.8: Ukázka hry spuštěné na mobilním telefonu iPhone 4 s operačním systémem iOS 7. Obrázek zachycuje úspěšné přihlášení do služby Game Center.



Obrázek B.9: Ukázka hry spuštěné na mobilním telefonu iPhone 4 s operačním systémem iOS 7. Obrázek zachycuje nákup položky uvnitř aplikace.



Obrázek B.10: Ukázka hry spuštěné na mobilním telefonu Nokia Lumia 810 s operačním systémem Windows Phone 8.1.



Obrázek B.11: Ukázka hry spuštěné na mobilním telefonu Nokia Lumia 810 s operačním systémem Windows Phone 8.1. Obrázek zachycuje nákup položky uvnitř aplikace.