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

HRA NA PLATFORMĚ IOS

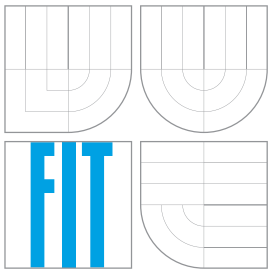
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB DOHNAL

BRNO 2012

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

HRA NA PLATFORMĚ IOS

THE IOS PLATFORM GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB DOHNAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAEL ANGELOV

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá vývojem multiplayer hry pro platformu iOS. Popisuje nástroje určené k vývoji aplikací pro platformu iOS a herní frameworky Cocos2d a Sparow. Dále rozebírá síťovou komunikaci, problém s časovou odezvou a možnosti síťového připojení mezi více zařízeními. Následně obsahuje návrh a implementaci jednoduché multiplayer hry využívající služby Game Center. Závěr je věnován testování a vyhodnocení průzkumu hraní multiplayer her na mobilních zařízení.

Abstract

This bachelor thesis is focused at the multiplayer games development. First chapters discuss the use of the game development tools and frameworks, specifically Cocos2d and Sparrow. After this introduction to the development for the iOS platform, study describes the communication between multiple devices, high latency problems and various network architectures. The succeeding chapters describe a design and implementation of a simple multiplayer game based on the Game Center social service. The end of this thesis reviews the user tests and the results of a survey on the multiplayer gaming on mobile devices.

Klíčová slova

hra, multiplayer, iOS, Cocos2d, Game Center

Keywords

game, multiplayer, iOS, Cocos2d, Game Center

Citace

Jakub Dohnal: Hra na platformě iOS, bakalářská práce, Brno, FIT VUT v Brně, 2012

Hra na platformě iOS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michaela Angelova. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Dohnal
16. května 2012

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Michaelu Angelovi za cenné rady, připomínky a metodické vedení práce.

Obsah

Kapitola 1

Úvod

Smartphone, z anglického překladu chytrý telefon, je malé přenosné zařízení, postavené na mobilní výpočetní platformě s více pokročilou výpočetní schopností a konektivitou, než klasický *hloupý telefon*. V současnosti je hardwarové vybavení smartphonů v určitých směrech pokročilejší než v noteboocích nebo desktopech, jedná se zejména o mobilní bezdrátové technologie. Programové vybavení Smartphonů se skládá z operačního systému (iOS, Android, Symbian OS, Windows Mobile, Bada, atd.) a aplikačního rozhraní, které umožňuje instalaci, nebo úpravu programů třetích stran. Spojením nejmodernějšího hardware a vysoce variabilního software do malého tělesa s displejem dostaneme zařízení, které dokáže provádět pokročilé výpočetní úkony, bez problémů obstojí v multimédiích, umožní 24 hodin denně připojení k internetu a v neposlední řadě nabídne hraní her v multiplayer režimu. Toto zařízení se právem nazývá smartphone.

Tato bakalářská práce se zabývá vývojem multiplayer hry na mobilních zařízení se systémem iOS. První část práce pojednává o grafických frameworcích a vývoji aplikací pro platformu iOS. V další kapitole je obsažen popis síťové komunikace, problém s časovou odezvou mezi zařízeními u různých typů multiplayer her a možnosti síťového připojení mezi více zařízeními na platformě iOS. Následující kapitola obsahuje návrh hry a její vývoj v čase, grafické a zvukové zdroje a práce s nimi. V předposlední části je uvedena implementace hry. Na konci je popis testování a hodnocení průzkumu hraní multiplayer her na mobilních zařízení.

Kapitola 2

Vývoj pro iOS

Tato kapitola popisuje, jakým způsobem je možné vyvíjet aplikace na operační systém iOS, jazyk, v němž se aplikace píší a nástroje používané k testování a optimalizaci. Jedna podkapitola je věnována i vrstvě Cocoa Touch poskytující infrastrukturu pro implementaci grafického rozhraní aplikace a interakci s uživatelem.

2.1 Objective-C

Je to objektově orientovaný programovací jazyk, definovaný jako malá, ale efektivní sada rozšíření standardu ANSI jazyka C. Rozšíření jazyka C jsou většinou založeny na jazyku Smalltalk, jednom z prvních objektově orientovaných programovacích jazyků. Podporuje tři mechanismy správy paměti a to automatickou správu, nebo počítání referencí *(více viz. kapitola 2.1.1)* . Objective-C se používá v operačních systémech Mac OS X a iOS a byl vytvořen ve společnosti Stepstone na začátku osmdesátých let dvacátého století dvěma muži Brad Coxem a Tom Lovem [5].

2.1.1 Paměť

Aplikace psané pro iOS musí sami zajišťovat správné uvolňování paměti.V iOS totiž není dostupné *Garbage Collection* z důvodu větší zátěže, která by se projevila na výdrži baterie v zařízení. Uvolňování objektů z paměti se provádí pomocí ručního, nebo automatického počítání referencí na objekt. Počet referencí je reprezentováno vlastností `retainCount` třídy `NSObjekt`, která je kořenovou třídou většiny Objective-C tříd. Reference se ovládají metodami `retain` pro zvýšení a `release` pro snížení její hodnoty. Používání těchto metod na správném místě musí zajistit buď programátor při ručním počítání referencí, nebo překladač při automatickém počítání referencí. Při vytvoření objektu je nastaven `retainCount` na hodnotu 1. Pokud tato hodnota klesne na 0, objekt je smazán z paměti a tu je možné znovu použít [1].

Sledování správného uvolňování paměti je možné instrumentem jménem Leaks obsaženém v prostředí Xcode IDE. V reálném čase sleduje běžící aplikaci a vypisuje informace o možných únicích paměti. Díky provázanosti s Xcode dokáže Leaks označit uniklé objekty a místo v kódu, kde k úniku došlo.

2.2 Cocoa Touch

Je tvořena sadou frameworků používaných k řízení interakce s uživatelem na zařízeních s operačním systémem iOS. Cocoa Touch je odvozen z frameworku Cocoa používaném v operačních systémech Mac. Byl spolu s grafickým rozhraním kompletně předělán pro vícedotekovou podporu. Je postaven na paradigmatu Model-View-Controller[1] a poskytuje solidní základ pro vytváření moderních aplikací. Obsahuje například framework Core Data pro správu dat, Core Animation k vytváření grafických efektů a engine WebKit k zobrazení internetových stránek.

2.3 Nástroje

Velmi důležitý a práci usnadňující nástroj pro programátora je vývojové prostředí. V případě vývoje aplikací v jazyce Objective-C se používá Xcode IDE. Je dostupný pouze pro platformu Mac a kromě obligátních věcí obsahuje navíc iOS simulátor a takzvané instrumenty. iOS simulátor dokáže simulovat zařízení typu iPhone a iPad. V obou případech je možné nastavit jeden ze dvou typů rozlišení, kterým disponují různé verze těchto zařízení. Simulovaná zařízení mají i softwarový akcelerometr, s nímž lze pracovat omezeně, pouze otáčením zařízení o 90 stupňů nebo provedením akce *zatřesení*. Instrumenty slouží k ladění aplikací a některé z nich jsou popsány v následujícím seznamu.

- Diagnostika energie poskytuje diagnostiku spotřeby energie a stav jestli jsou zapnuty nebo vypnuty hlavní komponenty zařízení.
- Network Connections analyzuje používání TCP/IP a UDP/IP spojení.
- Core Animation sleduje vytížení grafického procesoru v čase.
- Automation ten spouští skript simulující uživatelskou interakci s aplikací
- Alocation měří velikost používané paměti v haldě.
- Leaks sleduje úniky paměti.
- OpenGL ES Driver sleduje grafický výkon knihovny OpenGL ES.
- OpenGL ES Analysis měří a analyzuje činnost OpenGL ES. Zjišťuje problémy s výkonem a nabízí doporučení pro řešení problémů.

[1] Model-View-Controller je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní.

Kapitola 3

Herní frameworky

Herní framework zjednodušuje vývoj hry a obvykle obsahuje nástroje pro vykreslování *(anglicky rendering)* 2D a 3D grafiky, fyzikální simulace, detekci kolizí, přehrávání zvuku, vytváření animace a umělé inteligence. Většina herních frameworků používá knihovnu OpenGL, která poskytuje API pro tvorbu 2D a 3D grafických aplikací.

Pro operační systém iOS existuje řada takovýchto frameworků. V této sekci je popis dvou z nich, které jsou často používané a dostupné zdarma.

3.1 Sparrow

Sparrow je minimalistický open source herní framework pro operační systém iOS. Jeho cílem je být co nejvíce jednoduchý, rychlý a intuitivní. Podporuje zařízení typu iPhone, iPad a iPod touche s operačním systémem iOS ve verzi 3.0 a vyšší. Je psán čistě v jazyce Objective-C, takže není nutné se učit další jazyk. Vyvíjí ho firma Gamua pod licencí FreeBSD, která se inspirovala API Adobe Flash. Dále je popis základních funkcí tohoto frameworku rozdělených do tří částí.

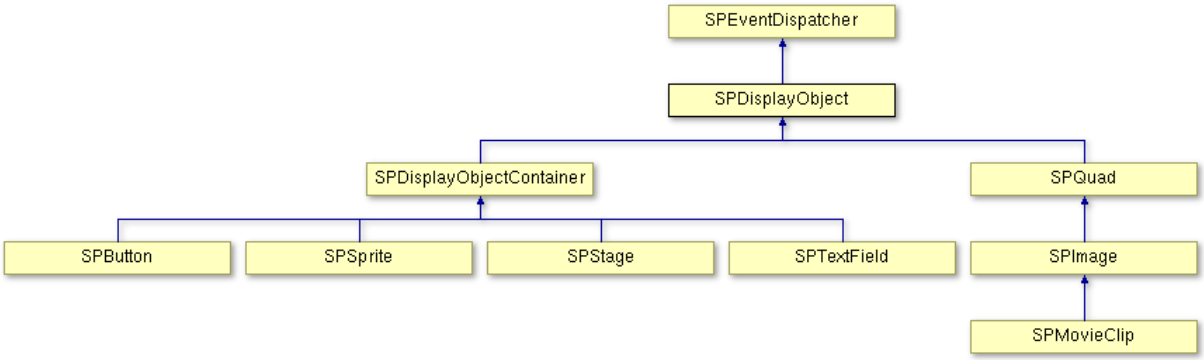
3.1.1 Zobrazované objekty

Objekty zobrazované na displeji musí obsahovat informace jako pozice, rozměry, měřítko, rotace a krytí. Třída `SPDisplayObject` obsahuje vlastnosti a metody poskytující tyto informace a používá se jako kořenová třída pro všechny zobrazované objekty. Na obrázku 3.1 je zobrazena hierarchie dědičnosti základních zobrazovacích tříd ve Sparrow.

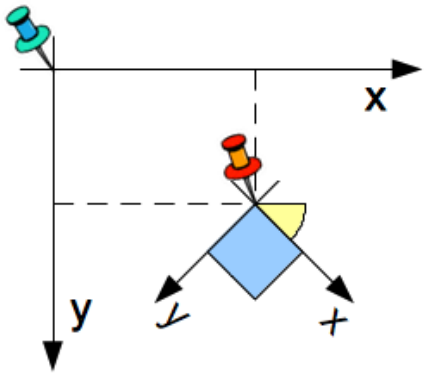
Třída `SPDisplayObject` má vlastnosti `pivotX` a `pivotY`, které dohromady tvoří bod určující počátek souřadnicového systému uvnitř objektu. Vůči tomuto bodu se vztahuje i poloha a rotace objektu na displeji. Vlastnosti `pivotX` a `pivotY` lze libovolně nastavit.

Pro převrácení objektu složí vlastnosti `scaleX` a `scaleY`. Při nastavení `scaleX` na hodnotu `-1` je objekt na displeji převrácen vodorovně. Při nastavení `scaleY` na `-1` je převrácen svisle.

Další důležitou třídou je `SPDisplayObjectContainer`, která slouží jako kontejner pro ostatní objekty třídy `SPDisplayObject`. Lze tak vytvářet stromovou architekturu a pracovat následně s celým stromem, jako by to byl jeden objekt. Každý kontejner má svůj vlastní souřadnicový systém nezávislý na okolních systémech. Objekty jsou tedy pozicovány podle souřadnicového systému kontejneru, ve kterém se nacházejí. Na obrázku 3.2 je zobrazen globální souřadnicový systém a souřadnicový systém uvnitř pootočeného kontejneru.



Obrázek 3.1: Diagram dědičnosti zobrazovaných objektů ve frameworku Sparrow. [4]



Obrázek 3.2: Zanořené souřadnicové systémy u grafických objektů. [4]

3.1.2 Události

Sparrow má flexibilní systém událostí. Ten umožňuje volat vlastní metody při výskytu již předdefinovaných nebo vlastních událostí. Jak je z obrázku 3.1 vidět, naprosto všechny zobrazované objekty dědí třídu `SPEventDispatcher`, která poskytuje metody k zachytávání událostí a jsou tak připraveny je zpracovávat. U zanořených objektů ve stromové hierarchii*(viz. kapitola 3.1.1)* je možné provádět takzvané probublávající *(anglicky Bubbling)* události. Takové události se pak dají zachytit kdekoliv, kde se strom používá. Mezi hlavní události patří dotek prstu na displeji. Sparrou poskytuje událost jako je začátek dotyku, tažení prstu po displeji, stání prstu na místě a konec dotyku prstu s displejem.

3.1.3 Textury

Zobrazit obrázek na displeji lze dvěma způsoby. Prvním způsobem je načtení obrázku třídou `SPTexture`, vytvoření textury přímo z něj a následně jeho zobrazení pomocí třídy `SPImage`. Druhým způsobem je obrázek rovnou načíst třídou SPImage. Pokud se ale bude zobrazovat obrázek na displeji vícekrát, je nutné z důvodu ušetření paměti použit první metodu.

OpenGL může mít textury o rozměrech v rozsahu 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048. Pokud má tedy obrázek rozměry 510×65 pixelů vytvoří se z něj textura o rozměrech 512×128 pixelů. To samozřejmě zabírá zbytečně mnoho místa v paměti. Jde tomu předejít používáním atlasu textur *(anglicky Texture Atlas)*, který vytváří ze všech obrázků jednu velkou texturu. Z takové textury je pak možné vykreslovat jednotlivé obrázky zvlášť. K tomu je ale nutné ještě vytvořit objekt třídy `SPTextureAtlas`, jenž obsahuje název, počáteční pozice a rozměry obrázků v atlasu textur. Vytvořením jediné velké textury se tak šetří velikost zabírané paměti.

Třída `SPQuad` představuje obdélník s jednou barvou nebo s barevným gradientem. Gradient je možné vytvořit ze čtyř barev, které jsou nastaveny k jednotlivým vrcholům obdélníku. Barva nebo gradient se při vykreslování násobí s číselnou reprezentací barvy každého pixelu textury a textura je tak přebarvena jinou barvou s tím, že se zachovají tmavé hrany a stíny.

`SPTextField` slouží pro vykreslení textu. Ten může mít přidělen jeden z fontů podporovaných v iOS. V případě nutnosti použití jiných než podporovaných fontů je možné načíst své vlastní. Je to velice podobné jako u atlasu textur. Je potřeba jeden velký obrázek s vlastním fontem a soubor s příponou *.fnt* kde jsou uloženy začátky a velikosti nakreslených znaků v obrázku s fontem.

Sparrow dokáže rozlišovat mezi zařízeními s rozlišením 480×320 a 960×640 pixelů a podle toho pak použít sadu obrázků načítaných do textur. Stačí uložit obrázky pro vysoké rozlišení do adresáře */graphics/2x* a pro nižší rozlišení do adresáře */graphics/1x* a Sparrow pak automaticky použije jednu ze sad.

3.1.4 Animace

Animace je základní součástí každé hry a Sparrow se snaží její použití co nejvíce zjednodušit. Rozlišuje mezi dvěma typy animací. První, kde od začátku je jasné, jak celá animace bude probíhat a druhá dynamická, kde nelze průběh animace předpovídat. K dynamické animaci se ve Sperrow používá volání vlastní metody při příchodu události *vykreslení snímku*. Ta je ve Sperrou předdefinována a používána pod názvem `SP_EVENT_TYPE_ENTER_FRAME`. U prvního typu animace se používá třída `SPTween`, která provádí plynulý přechod animovaného

objektu z počátečního stavu *(tj. hodnoty vlastností objektu)* do koncového stavu. Stačí předat objektu `SPTween` objekt, který se má animovat, koncový stav objektu a čas, za který se má animace provést. Ten pak automaticky mění stav objektu po krocích tak, aby se za nastavený čas dostal do koncového stavu. Přechod mezi stavy nemusí být vždy lineární, ale je možné použít jeden ze šestnácti předdefinovaných.

Animovat se dá nejen postupnou změnou některých vlastností zobrazovaného objektu, ale i rychlým střídáním obrázků. Ve Sparrow stačí vytvořit objekt třídy `SPMovieClip` a předat mu obrázky ve stejném pořadí, ve kterém se mají střídat, a dobu mezi výměnou obrázků.

3.1.5 Zvuk

Třída `SPSound` načítá zvuková data do paměti. Tyto data se následně přehrávají vytvořením zvukového kanálu `SPSoundChannel`, přes který lze přehrávání ovládat. Sparrow používá dvě knihovny pro přehrávání zvuku OpenAL a AVAudioPlayer. Automaticky mezi nimi vybírá podle formátu přehrávaného zvuku.

3.2 Cocos2d framework

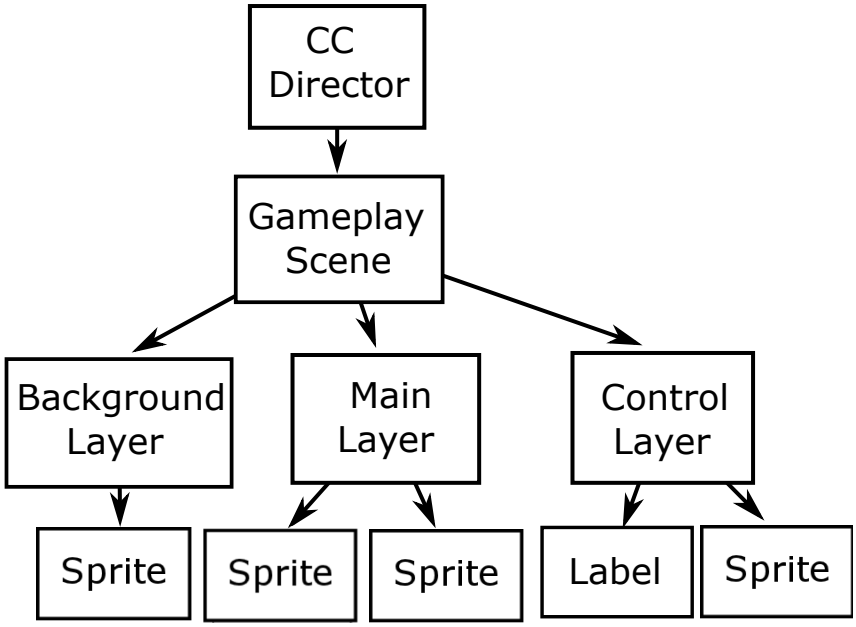
Cocos2d je open source 2D herní framework vydáváný pod licencí *Massachusetts Institute of Technology (MIT)*. Původně byl napsán v jazyce Python, ale po čase byl portován do dalších jazyků. V této kapitole je popsán jeho port do jazyku Objective-C pro platformu iOS a Mac OS X, který používá OpenGL ES verze 1.1. Je možné ho používat na zařízeních typu

- iPhone / iPod Touch 1. generace,
- Phone 3G a iPod Touch 2. generace,
- iPhone 3GS a iPod Touch 3. generace,
- iPhone 4 a iPod Touch 4. generace s podporou módu Vysoké-Rozlišení (anglicky High-Res),
- iPad,
- iPad 2,
- a operačním systému Mac OS X 10.5 a 10.6

Cocos2d požívá celá řada společností a to i celosvětově známých jako například Zynga, nebo Atari. O jeho oblíbenosti mezi vývojáři svědčí i to, že v *App Store* je už více jak 2500 aplikacích využívající tento framework a některé z nich jsou na prvních příčkách v oblíbenosti uživatelů.

Je podobný frameworku Sparrow v používání textur, atlasu textur, zanořování zobrazovaných objektů do sebe a používání zvukových knihoven. Cocos2d má mimo jiné navíc částicový systém, lepší práci se spojováním a řízením animací a je provázaný s fyzikálními enginy Box2d a Chipmunk.

Poloha objektů je v Cocos2d reprezentována jak pixely tak také takzvanými body, přičemž většinou se pracuje s body. Jeden bod je totiž u iPhone s rozlišením 480×320



Obrázek 3.3: Hierarchie zobrazovaných objektů. [8]

reprezentován jedním pixelem a na displeji s rozlišení 960×640 je reprezentován čtyřmi pixely. Při nastavení stejné pozice objektu v bodech se bude objekt na těchto dvou rozlišeních zobrazovat na displeji na stejném místě.

V tomto frameworku se nepožívá jedna samostatná smyčka, která řídí hru. Místo toho se používá plánovač (anglicky scheduler), který dokáže za běhu volat metody předané do plánovače pomocí selektoru. Ten je realizován třídou `CCScheduler` a umožňuje buď volání metody s určitým časovým intervalem nebo před každým vykreslením scény. Metoda je volána pořád dokola, dokud není z plánovače odstraněna.

3.2.1 Základní struktura

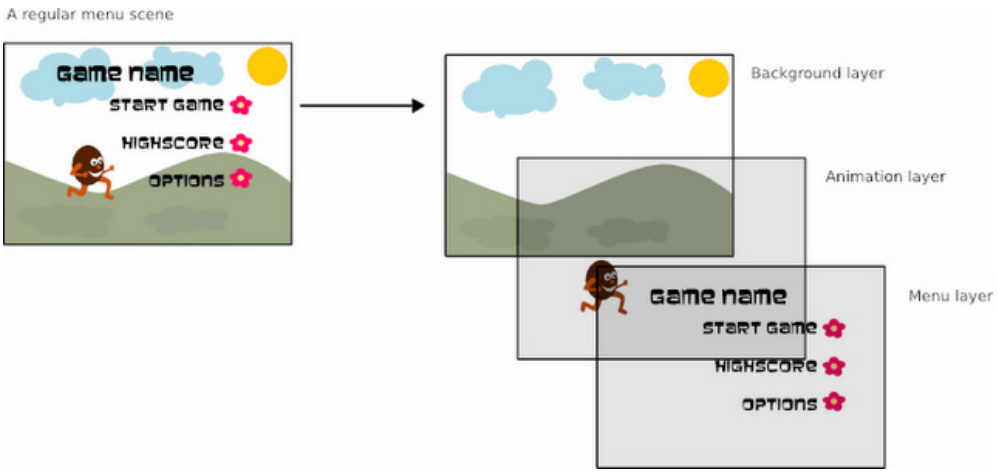
Cocos2d používá hierarchickou strukturu *(viz. obr. 3.3)*, která se skládá ze scén, vrstev a uzlů. Scénu tvoří jedna nebo více vrstev, které se přes sebe překrývají *(viz. obr. 3.4)*. Vrstvu tvoří obrázky, nápisy a další základní grafické objekty. Níže je popsáno několik tříd použitých v této hierarchii.

CCDirector

CCDirector řídí přepínání mezi scénami. Ví jaká scéna je zrovna zobrazována a řídí její aktivitu. Je na vrcholu hierarchie a v celé aplikaci existuje pouze jedna jeho instance.

CCScene

Aplikace je rozdělena do scén, jak již bylo uvedeno. Tyto scény jsou realizovány objekty třídy `CCScene`. Aplikace může mít více scén, ale pouze jedna z nich je zobrazena na displeji.



Obrázek 3.4: Použití vrstev ve frameworku Cocos2d. [3]

CCLayer

Objekty této třídy reprezentují vrstvy. Kromě obrázků, nápisů a jiných základních grafických objektů může obsahovat i jinou vrstvu. Cocos2d má několik již předdefinovaných vrstev jako je například menu hry a barevná vrstva pro ulehčení.

CCSprite

CCSprite je 2D obrázek, který lze přesunovat, otáčet, měnit měřítko a animovat. Může být složen z více objektů třídy `CCSprite`.

CCNode

Všechny tyto objekty, kromě `CCDirektor` jsou podtřídou `CCNode`. To je základní třída pro zobrazované objekty. Obsahuje vlastnosti jako je velikost, pozice, rotace a metodu `draw`. Ta je volána při každé obnově snímku na displeji. Pokud je třeba nějakého specifického nastavení OpenGL pro vykreslení daného objektu, je dobré k tomu použít právě tuto metodu.

3.2.2 Částicový systém

Částicový systém umožňuje realisticky simulovat Fuzzy jevy jako je kouř nebo oheň. Cocos2d má dva typy částicového systému, `CCParticleSystemPoint` a `CCParticleSystemQuad`. V prvním případě je částicový systém reprezentován obrázky do velikosti 64 pixelů. V druhém případě může být reprezentován většími obrázky, se kterými je možné rotovat kolem své osy, je dokonce rychlejší než první typ, ale spotřebovává mnohem více systémových prostředků.

3.2.3 Animace a akce

Animace je v Cocos2d rozdělena na *akce* a *animace*, které se dají provádět na jakémkoliv `CCNode` objektu. Akce obvykle mění některé atributy objektu jako jsou pozice, rotace a měřítko. Například při letu letadla do cílového bodu se při každém obnovovacím snímku mění pozice letadla tak, aby plynule přeletěl do cílové pozice. U akcí se pracuje pouze s již načteným obrázkem.

Animace je posloupnost obrázků, které se s časovým intervalem postupně střídají. Protože se u animací používá hodně obrázků, je dobré využívat `CCSpriteBatchNode` objekt, který má stejnou funkci jako atlas textur a k tomu použít `CCSpriteFrameCache` objekt, v němž jsou informace o umístění a velikosti obrázků v atlasu textur.

Akce jednoho objektu je možné řetězit nebo paralelizovat a to s jinou akcí nebo s animací. Zřetězené akce se provádí jedna podruhé, zato paralelizované se provádí všechny najednou. V Cocos2d je možné také vytvářet opakování akcí pomocí třídy `CCRepeat` při konečném počtu opakování a `CCRepeatForever` při nekonečném počtu opakování. Všechny tyto třídy se dají do sebe zanořovat takže se z jednoduchých akcí a animací dají skládat složité.

3.2.4 Zvuk

K přehrávání používá zvukový engine CocosDenshion, který je součástí frameworku Cocos2d a pracuje s knihovnami openAL a AVAudioPlayer. Dá se velice snadno používat pomocí SimpleAudioEngine a stejně jako Sparrow používá automaticky jednu ze dvou knihoven podle typu přehrávaného souboru.

Kapitola 4

Síť

Plynulost online her ovlivňuje nejen výkon zařízení, na kterém je hra spuštěna, ale také parametry síťového připojení, přes něž se hra hraje. Síť je velmi důležitou částí této práce. Je zde uveden graf a tabulka dopadu latence na hratelnost hry. Součástí kapitoly je i porovnání síťových architektur používaných u online her a postupy pro snížení dopadu latence na hratelnost. Na konci kapitoly jsou popsány nástroje používané k vývoji online a lokálních multiplayer her na platformě iOS.

4.1 Latence

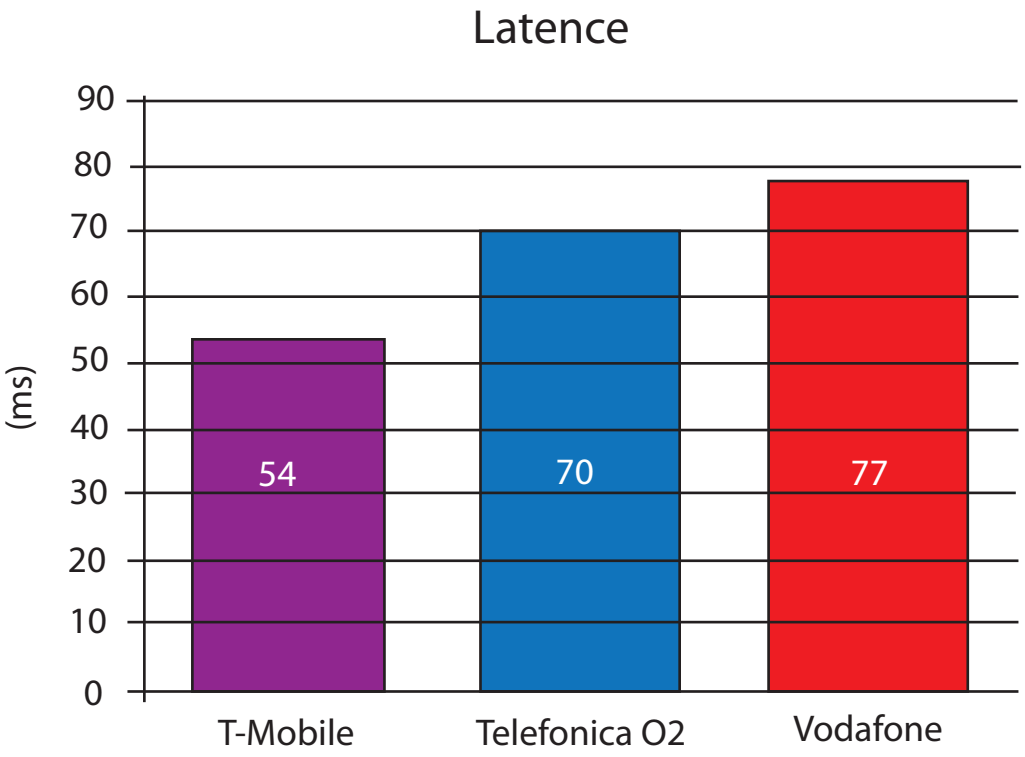
U online multiplayer her není potřeba odesílat přes síť velké data. Spíše je třeba často posílat krátké zprávy jako informace o stavu hry a akcích jednotlivých postav. Doba přenosu těchto dat je tedy závislá spíše než na šířce přenosového pásma, na zpoždění dosaženého v přenosové síti takzvané latenci. Latence je čas mezi odesláním dat z jednoho uzlu a příjmem dat uzlem druhým. Rozlišuje se na jednosměrnou a obousměrnou. U obousměrné latence *(tzv. roud-trip time)* se zahrnuje čas potřebný k přenosu paketu tam i zpět plus doba zpracování dat.

Nejčastěji se měří obousměrná latence, protože je možné ji měřit z jednoho uzlu. Používá se k tomu program Ping využívající ICMP protokol. Ten periodicky odesílá na cílovou adresu zprávy `Echo Request` a očekává odpovědi typu `Echo Reply`. Aby bylo možné zprávy mezi sebou odlišit, obsahují identifikátor `icmp_seq`. Po uplynutí určité doby bez odpovědi na danou zprávu je tato zpráva prohlášena za ztracenou.

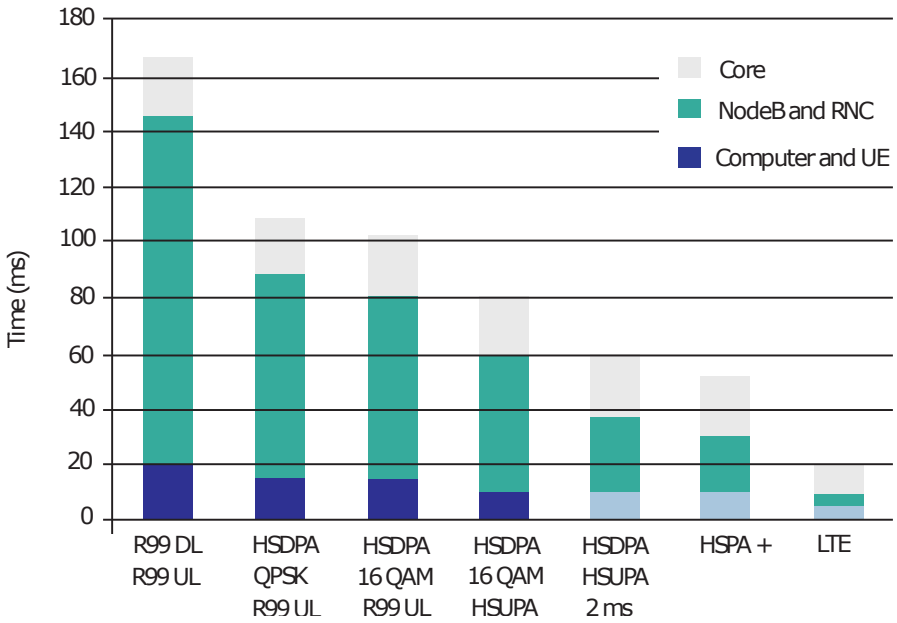
4.1.1 Latence v mobilní síti

Přenos dat v mobilních sítích je velice nespolehlivý. Je udáváno, že se ztratí až 1% odeslaných paketů. Nadměrné ztráty se řeší RLC[1] protokolem, který snižuje ztrátovost až na $0,0001\%$. Na druhou stranu se tím zvyšuje čas přenesení paketu sítí. Na obrázku 4.1 jsou vidět průměrné latence přenosových soustav jednotlivých operátorů v České republice. Měření bylo provedeno na mobilních sítích 3. generace. Při požadavku velikosti latence do 100 ms [10] z grafu vyplývá, že je možné využívat sítě 3. generace k hraní online her. Pro mobilní sítě 4. generace se počítá s výrazným snížením latence *(viz. obr. 4.2)* a to i díky podpoře RLC přímo v základních stanicích (antén) [2].

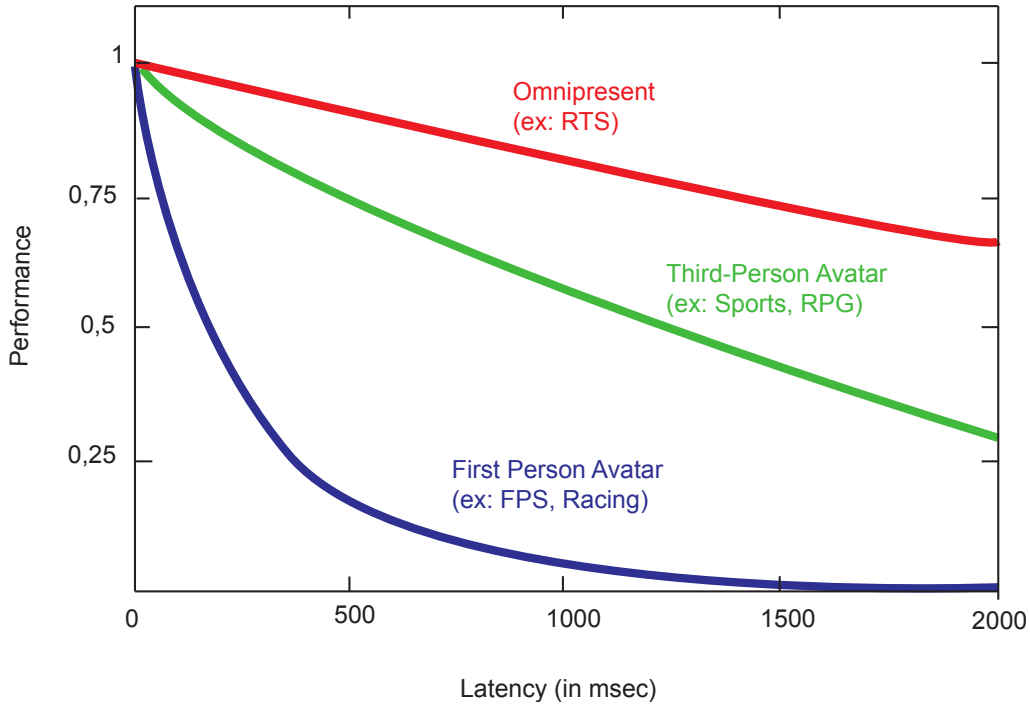
[1] je zkratka Radio Link Control, protokol používající se v mobilních sítích k zvýšení spolehlivosti přenosu dat sítí [7].



Obrázek 4.1: Průměrná latence jadnotlivých operátorů v ČR [9]



Obrázek 4.2: Dosahovaná latence na jednotlivých technologií mobilních sítí [2]



Obrázek 4.3: Dopad latence na hratelnost různých žánrů her [6]

4.1.2 Dopad Latence

Základní dělení online her je na Turn-Base(Tahové hry) a Real-Time(probíhající v reálném čase), přičemž pro tahové hry je význam časového zpoždění minimální a komunikace probíhá v nízké intenzitě. Protože hráč čeká až dohraje protivník, neovlivní jeho hru velké síťové zpoždění. U Turn-Base her je daleko důležitější spolehlivost přenosu všech dat než jeho rychlost.

U Real-Time her spíše než spolehlivost přenosu dat, je důležitý čas přenesených dat přes síť. Dopad latence na hratelnost Real-Time her se liší podle osoby, z níž se pohlíží na herní svět (viz. graf 4.3).

- z pohledu první osoby takzvané FPS - pohled na herní svět je skrz hlavní postavu.
- z pohledu třetí osoby takzvané TPS - na herní svět se díváme mimo pohled herní postavy, ale kamera se pohybuje s herní postavou.
- a všudy přítomný pohled - vidíme celý herní svět najednou.

Z grafu 4.3 vyplývá, že rapidní pokles hratelnosti s narůstající latencí je u prvních dvou typů her. Pro mobilní zařízení, která jsou většinou k internetu připojena přes mobilní sítě 3. generace je latence téměř na hranici přijatelných hodnot a kolísá.

4.1.3 Eliminace latence

Úspěšnost hry je závislá mimo jiné také na její dobré hratelnosti. Pokud se hra často trhá a pohyb hráčů není dostatečně plynulý, hratelnost prudce klesá a hra se může dokonce stát nehratelnou. Je dobré co nejvíce předejít tomuto jevu. To lze provést výběrem správné síťové architektury a použitím metody přemostění (anglicky bypasing) dočasně velké latence [10].

U Real-Time multiplayer her je na výběr ze dvou síťových architektur klient-server nebo peer-to-peer.

Klient-server

U multiplayer hry používající architekturu sítě klient-server probíhá celá simulace hry na serveru a klienti pouze provádějí příkazy, které od serveru dostává a odesílá informace o vstupech uživatele. Informace jsou tak posílány nejprve od klienta na server, který je zpracuje a potom odeslány všem klientům. Musíme tak počítat nejen s latencí, která vzniká při cestě od klienta na server, ale i s latencí vznikající při cestě od serveru ke klientovy. Toto však můžeme minimalizovat připojením serveru k rychlému internetovému připojení a jeho umístěním podle geolokace cílové skupiny hráčů. Pokud je cílová skupina celosvětová nebo se nachází na rozsáhlém území, použije se seskupení více serverů *(anglicky server cluster)* rozmístěných na tomto území.

K *přemostění* vysoké latence na straně klienta se u této architektury používá metoda, kdy klient není realizován pouze jako terminál, ale sám dokáže z uživatelských vstupů provést simulaci hry. Je tak možné na straně klienta jen předpokládat, že server obdržel jeho zprávu o pohybu hráče a pohyb realizovat lokálně. Tato metoda se nazývá *předpověď na straně klienta (anglicky client-side prediction)*. Po obnovení spojení se serverem proběhne synchronizace výsledků simulace, která může způsobit velký skok v posunu hráče. Aby byl efekt přemostění co největší, používá se na straně klienta i predikce pohybu protihráčů. Ta se realizuje interpolací nebo extrapolací historie pohybů [10].

Peer-to-peer

U peer-to-peer architektury jsou všichni klienti propojeni mezi sebou, každý s každým a simulace se provádí na každém z nich zvlášť. Informace jsou odesílány od jednoho klienta přímo všem najednou a počítá tedy pouze s latencí mezi klienty. K přemostění latence u této architektury stačí použít pouze predikci pohybu protihráčů.

Kapitola 5

GameKit

GameKit je součástí vrstvy Cocoa Touch. Poskytuje vývojářům nástroje na vytváření sociálních her. GameKit se dále dělí na tři části. Game Center, kde je možné sdílet názor na hru s ostatními hráči a vytvářet multiplayer hry s hráči připojenými přes Internet. Peer-to-Peer conectivity část, která poskytuje třídy pro připojení hráčů na lokální síti. In Game Voice pro hlasovou komunikaci mezi hráči.

5.1 Peer-to-Peer conectivity

Tato část, jak již bylo zmíněno, poskytuje třídy pro připojení hráčů v lokální síti. Síť může být buď vytvořena jedním ze zařízení pomocí bezdrátové technologie bluetooth a ostatní hráči připojení metodou ad-hoc, anebo připojením zařízení do stejné lokální sítě přes technologii WI-FI. Při vytváření sezení je možné nastavit typ zařízení na klienta, server, nebo peer (zařízení se chová jako klient a server zároveň). K zobrazení a konfiguraci dostupných sezení v síti můžeme použít standardní nebo vlastní uživatelské rozhraní. Není omezen formát odesílaných dat, ale jen jeho velikost a v jedné zprávě je možné odeslat data o velikosti 87kB. Doporučuje se však, aby přenášená data měla maximální velikost 1000B. Data vyššího objemu se automaticky rozdělí na menší části a v cíli složí dohromady. To vede ke zvýšení latence a spotřebě více času ke zpracování těchto dat. Pokud je spojení využíváno k posílání obsahu souborů nebo důležitých informací, použije se spolehlivý přenos. K přenosům dat potřebným v reálném čase se používá nespolehlivý přenos. Spolehlivý přenos zajistí doručení všech zpráv ve správném pořadí. U nespolehlivého přenosu může dojít ke ztrátě dat anebo doručení v jiném pořadí než byly odeslány. Na druhou stranu je tento přenos rychlejší a nevyžaduje tolik režijních nákladů. Je však možné kombinovat oba typy těchto přenosů.

5.2 Game Center

Je to online sociální služba, která je dostupná na zařízeních obsahujících operační systém iOS 4.1 a vyšší. Umožňuje hráčům sdílet dosažené cíle a zobrazovat žebříčky a úspěchy jednotlivých her, které hrají. Mohou se přes tuto službu připojit k ostatním hráčům v multiplayer hrách. Služba Game Center ukládá data o všech hráčích a podporuje propojení zařízení v různých sítích. Uživatel se může do služby dostat také skrz aplikaci Game Center, která je integrována v systému iOS. Ten může ve službě publikovat své status zprávy a přidávat kamarády z řad ostatních uživatelů. Do služby je nutné se přihlásit pomocí aliasu,

který má každý hráč unikátní. Umožňuje mu spravovat seznam svých přátel a publikovat své statusy, jež jsou viditelné pro přátele.

Framework Game Kit obsahuje nástroje umožňující spojení služby Game Center a využívání jejích služeb ve vlastní aplikaci. Aby aplikace mohla využívat Game Center musí být nejprve provedena autentizace uživatele. Tu je nutné implementovat do hry a provést jakmile je hra spuštěna. Pokud se autentizace neprovede, není možné se službou Game Center pracovat. V případě, že hráč není k účtu přihlášen nebo ještě nemá autentizační údaje, zobrazí se mu po spuštění hry rozhraní Game Center, kde si může vytvořit účet, nebo se přihlásit.

Až na pár vyjímek probíhá komunikace s Game Centrem asynchronně. Synchronní jsou pouze žádosti o zaslání dat z Game Center, nebo požadavky na vytvoření nějakého úkonu (například vytvoření zápasu pro multiplayer hru).

Aby bylo možné testovat aplikaci před jejím uvedením na App Store, poskytuje společnost Apple Inc. takzvané sandbox prostředí. Toto prostředí je oddělené od produkčního, ale umožňuje využívat služeb Game Centra. Hru je tak možné testovat, aniž by ji viděli uživatelé Game Centra. Game Center je dostupný i v iOS simulátoru, kde ale není možné používat přímé pozvání přátel do zápasu.

Dále je uveden popis některých nástrojů, které poskytuje služba Game Center. Na konci je samostatná kapitola věnující se používání Game Center k hraní multiplayer her.

Leaderboards (česky žebříčky) umožňují ukládat dosažené skóre ve hře do služby Game Center, kde jsou tyto žebříčky dostupné všem hráčům, kteří mají tuto hru nainstalovanou. Pomocí žebříčků hráči porovnávají své výsledky a je jen na vývojáři hry, jaké hodnoty se do žebříčku budou ukládat, jestli dosažený čas, vydělané peníze nebo obecně počet bodů. Není nutné se omezovat v jedné hře pouze na jeden žebříček. Je možné jich vytvářet více podle kategorií. Kromě ukládání hodnot z aplikace do žebříčků, je také možné tyto hodnoty zpětně číst.

Achivements (česky dosažené úspěchy) jsou specifické cíle, které hráč může dosáhnout ve hře. Například zabití určitého počtu protihráčů, dosažení počtu bodů atd. Je možné o tom informovat uživatele přímo ve hře, pomocí vyskakovací zprávy. Achivements jsou stejně jako Leaderboards dostupné přes Game Center. Uživatelé se tak mohou porovnávat s přáteli.

Perr-to-Peer poskytuje jednoduché rozhraní pro odesílání datových a hlasových služeb ostatním účastníkům zápasu. V zápasech, kde je více týmů, lze vytvořit hlasový kanál pro každý tým zvlášť. Ve všech případech se žádosti posílají pomocí tříd z GameKit frameworku.

5.2.1 Multiplayer

Službu Game Center je možné využít ke spojení více hráčů do jedné hry nebo jinak *zápasu*. K tomu slouží právě nástroj Multiplayer matchmaking (česky vytváření zápasu) dostupný ve frameworku GameKit, kterou může používat jakákoliv aplikace pracující s Game Center. V aplikaci je pak možné komunikovat mezi hráči přímo s použitím frameworku GameKit, nebo přes vlastní server, kde je nutné naprogramovat vlastní síťovou komunikaci. Před používáním Multiplayer matchmaking je nutné rozhodnout několik důležitých věcí, které se týkají multiplayer hry. Například kolik hráčů může být ve hře najednou, jak bude vypadat

architektura sítě mezi hráči a podle toho pak přizpůsobit využívání prostředků z Game Center.

Game Center poskytuje několik možností spojení hráčů do vytvářeného zápasu. Nejčastěji se požívá přímé pozvání. To buď skrz samotnou hru nebo aplikaci Game Center. Uživatel vybere ze seznamu přátel, s kým chce hrát, a odešle mu zprávu s požadavkem na připojení. Přítel obdrží požadavek prostřednictvím *push* notifikace zobrazené na displeji. Může se rozhodnout jestli pozvání přijme nebo odmítne. Pokud příjme, spustí se mu automaticky hra, ale pokud nemá hru nainstalovanou, dostane se skrz pozvánku do App Store, kde si příslušnou hru může stáhnout. Po dostatečném počtu příjmutých pozvánek může hra začít.

Další možností je automatické spojení hráčů. Jsou tak náhodně spojeni hráči, kteří zrovna používají automatické připojení. Lze nastavit i kritéria, podle nichž se budou hráči vyhledávat. Game Center pak najde hráče splňující tato kritéria a připojí je do zápasu. Hráč může být i automaticky připojen do již běžícího zápasu. Pokud třeba bude probíhat hra se třemi hráči a jeden se odpojí, může být nahrazený hráčem, který právě používá automatické připojení. Pokud nepoužíváme žádná kritéria na vyhledání hráčů, je připojení rychlé. U některých her je však potřeba připojovat podle určitých kritérií jako jsou skupiny a atributy, nebo oznámit uživateli, kolik hráčů čeká na automatické připojení.

Skupiny hráčů

Hráči jsou rozděleni do libovolných skupin, které si určí vývojář hry. Automatické připojování pak připojí pouze ty hráče, kteří se nacházejí ve stejné skupině. Závodní hra má například více map. Hráč si vybere mapu, na které chce závodit a při použití automatického vyhledávání ho Game Center připojí jen s hráči, kteří chtějí závodit na stejné mapě. Skupiny jsou reprezentovány 32 bitovým číslem takže jich může být až $2^{32} - 1$. Nula znamená, že hráč není v žádné skupině.

Atributy hráčů

Pomocí atributů je možné přiřadit hráči nějakou roli. Při automatickém vyhledávání jsou spojeni jen ti hráči, kteří nemají stejnou roli. Role je reprezentována 32 bitovým číslem a zápas je vytvořen jen tehdy, když exkluzivní OR atributů všech hráčů je dohromady roven $0xFFFFFFFF$.

Aktivity hráčů

Pomocí metod třídy GKMatchmaker můžeme hráči zobrazit aktuální počet čekajících hráčů na automatické připojení. Hráč je tak informován, jestli bude do zápasu připojen ihned, nebo bude čekat. Lze zobrazovat i počet čekajících hráčů podle skupin, ve kterých jsou.

Kapitola 6

Návrh

K vytvoření jednoduché online 2D hry je dobré rozdělit návrh do tří etap. V první etapě vyřešit koncept fungování nebo jinak popis hry. To znamená, co by se mělo ve hře odehrávat, určit základní pravidla, dějovou linií a charakteristiku postav. V druhé etapě, návrh síťové architektury a zasílaných zpráv. V poslední pak design a zvuk. V této kapitole jsou popsány jednotlivé etapy při návrhu hry JDGame, která je výsledkem této práce.

6.1 Popis hry

Rozhodl jsem se vytvořit jednoduchou online hru, žánrově spadající mezi Arkády. Děj hry probíhá ve vzduchu nad bojištěm, kde se hráči snaží pomocí vlastního vrtulníku sestřelit protihráče. Každý hráč má omezený počet nábojů. Po vystřílení všech nábojů musí hráč čekat než se mu sami dobijí. Hra končí po uplynutí časového limitu. V té chvíli se na displeji zobrazí statistyky všech připojených hráčů. Ty tvoří počet zabitých protihráčů a počet úmrtí. Vyhrává ten, který zabil nejvíce protihráčů. Při shodném počtu zastřelených hráčů rozhoduje o pořadí počet úmrtí. Statistika je zobrazena jen po určitou dobu a poté začíná hra zase od začátku. Takto běží hra pořád dokola dokud se neodpojí všichni hráči.

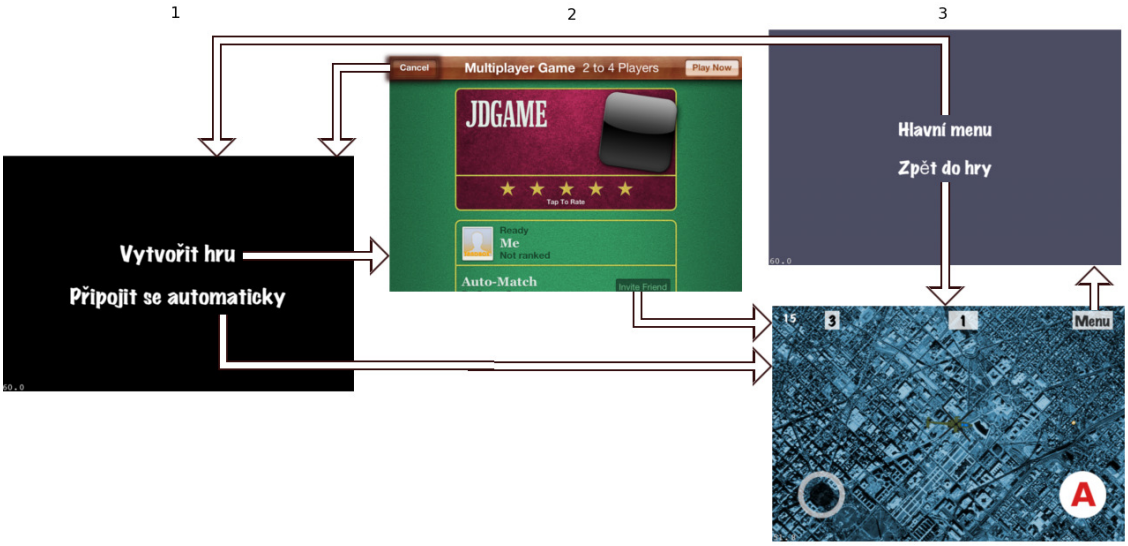
6.2 Struktura

Hra je rozdělena do tří hlavních částí *(viz. obr. 6.1)*. V první části je zobrazeno hlavní menu hry s možností nastavení zapnutí nebo vypnutí zvuku. Po zvolení položky *Vytvořit hru* je na displeji zobrazeno Game Center rozhraní, ve kterém hráč může pozvat kamarády do hry, nebo se do nějaké automaticky připojit. Po připojení dostatečného počtu hráčů je zobrazena herní scéna. Druhá položka *Připojit se automaticky* slouží k automatickému připojení do hry rovnou z menu.

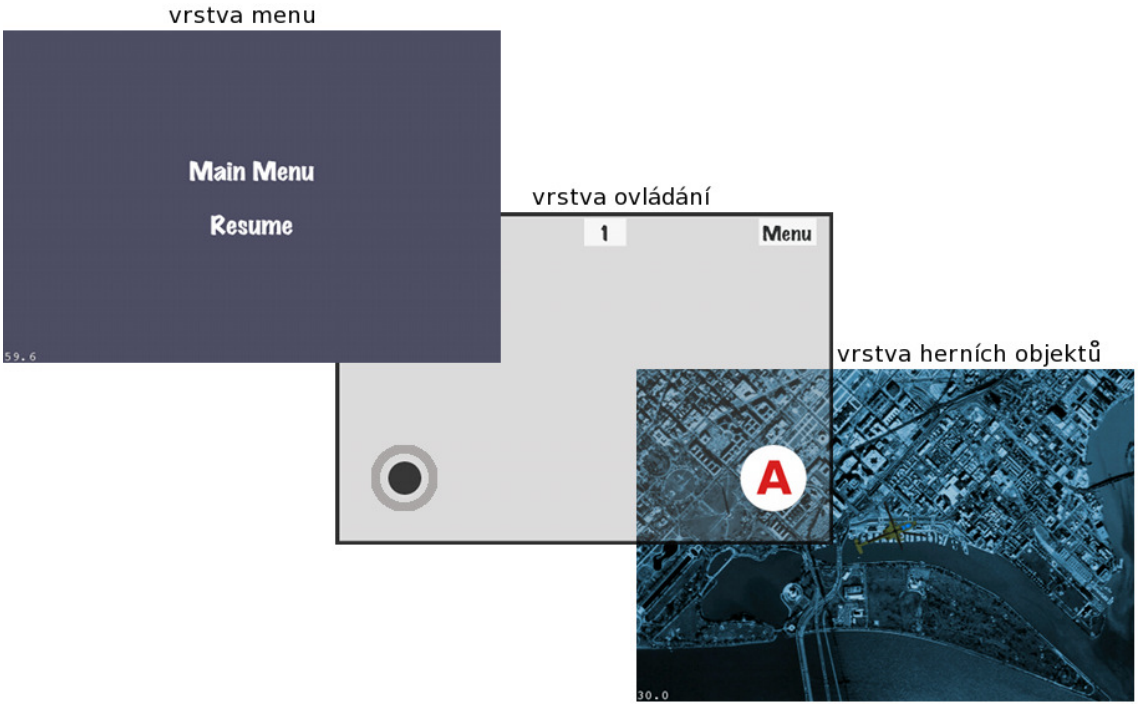
Herní scéna je složena z více vrstev *(viz. obr. 6.2)*. Nejníže je vrstva s vrtulníky vznášejícími se nad bojištěm. Nad ní je umístěná vrstva s ovládacími prvky a prvky zobrazujícími počet zásahů, střel v zásobníku a tlačítko menu hry. Nad tím vším se po stisknutí tlačítka "menu"zobrazuje vrstva s menu hry.

6.3 Síťová architektura

Síťovou architekturu je nutné zvolit podle typu hry. V tomto případě se jedná o Real-Time multiplayer hru, kde je možné použít architekturu klient-server nebo peer-to-peer. Pokud



Obrázek 6.1: Graf provázanosti scén



Obrázek 6.2: Vrstvy herní scény

```
/** Messages */
typedef enum {
    kMessageTypeRandomNumber = 0,
    kMessageTypeGameBegin,
    kMessageTypeMove,
    kMessageTypeShoot,
    kMessageTypeRotation,
    kMessageTypeGameOver,
    kMessageTypeImDead,
    kMessageTypeHit,
    kMessageTypeRevival,
    kMessageTypeTestLatency
} MessageType;
```

Obrázek 6.3: Výčet typů zpráv, kterými se klienti aplikace JDGame mezi sebou dorozumívají.

bych zvolil architekturu klient-server s vlastním serverem, musel bych udržovat server v chodu a s tím spojené náklady. U stejné architektury, kde funkci serveru plní jeden z klientů, bych sice ušetřil náklady, ale takové spojení přináší problém v podobě vysoké latence a možnosti rozpadnutí celého spojení. Rozpad spojení souvisí se spolehlivostí mobilní sítě, která je nestabilní a mohl by tak být server nedostupný *(viz. kapitola 4.1.1)*.

V případě architektury peer-to-peer je spojení udržováno i v případě odpojení jakéhokoliv hráče, nemusí se tak spravovat server připojený k Internetu. Rozhodl jsem se tedy použít architekturu peer-to-peer s topologií Full Mash.

6.4 Komunikace

Navrhl jsem množinu zpráv, kterou se mezi sebou klienti dorozumívají *(viz. obr. 6.3)* a rozdělil jsem zprávy podle typu nesoucí informace na dvě skupiny.

Klientské zprávy obsahují informace o pohybu lokálního hráče, jeho akcích, jako je výstřel a kolizích. Protože se kolize počítají na každém zařízení pouze pro lokálního hráče *(viz. kapitola 6.7)*, obsahují zprávy i informaci o tom, s jakým předmětem se hráč střetl.

Řídící zprávy odesílá pouze jeden z klientů takzvaný řídící klient, který oznamuje všem ostatním, že hra začala nebo skončila. Dostáváme se tím ale ke stejnému problému popsanému výše, úzkého hrdla. Navrhl jsem tedy hierarchické uspořádání klientů. Po připojení do sezení si každý klient vygeneruje číslo, které všem odešle. Pokud přijme od jiného klienta stejné číslo, klient generuje a odesílá číslo znovu. Klient, který má nejvyšší číslo se stává řídícím klientem. Pokud se tento klient odpojí ze sezení, přebírá řízení klient s následujícím nejvyšším číslem.

6.4.1 Spojovací server

Ke spojení hráčů do jedné hry je potřeba bod, k němuž se hráči připojí a oznámí, že jsou připraveni hrát. K tomu jsem využil sociální službu Game Center popsanou v kapitole 5.2. Uživatelé, kteří mají v této službě účet, jsou automaticky po připojení k internetu přihlášeni do Game Centra. Můžou být tedy tímto způsobem pozváni do hry i když nemají hru zrovna spuštěnou.

6.5 Grafika

Rozlišení displeje na iPhone 4 a 4S je opravdu velké. Objekty ve hře si ale musí udržet minimální reálnou velikost aby se nestaly příliš titěrnými. To na druhou stranu přináší problém v podobě příliš malého prostoru pro více hráčů. Rozhodl jsem se tedy udělat svět větší, než je velikost displeje. Pomyslná kamera, která snímá scénu, sleduje hráče a udržuje ho ve středu displeje a svět kolem se posouvá.

Grafiku jsem nevytvářel celou od začátku, ale upravil již existující. Podle svých představ jsem si vybral z návrhů jiných designérů na stránkách *www.deviantart.com* a *www.geekphilosopher.com*. Tu jsem upravil do konečné podoby tak, aby velikost a vzhled odpovídal mým představám. Z první zmíněné stránky jsem použil obrázek helikoptéry *(uživatele bagera3005)* , který jsem pouze zmenšil a rozdělil na dvě části, rotor a tělo helikoptéry. Pozadí jsem použil z druhých zmíněných stránek. To je tvořeno satelitním snímkem upraveným překrytím šedou barvou, aby více odpovídal nehostinnému prostředí bitevního pole. Ovládací prvky jsem navrhl a vytvořil sám.K úpravě grafiky do konečné podoby jsem použil program Gimp. Ten je velice účinný, multiplatformní a zdarma.

6.6 Zvuk

Zvuk dokresluje atmosféru hry a prohlubuje prožitek z jejího hraní. Při špatném použití může ale spíše hře uškodit a degradovat. Rozhodl jsem se použít reálné nebo velice podobné zvuky použitých objektů ve hře. U helikoptéry jsem použil opravdový zvuk rotoru spuštěný ve smyčce. Vypuštění střely doprovází zvuk stejný jako při vystřelení rachejtle. Na pozadí je zvuk bitevní vřavy, který má za úkol dokreslit podobu okolí, v němž se souboj odehrává. V podstatě je situace u zvuku stejná jako u grafiky. Existuje několik internetových stránek poskytujících zvukové efekty a to buď zcela zdarma, nebo za poplatek. Pro vlastní potřebu jsem využil stránky *www.freesfx.co.uk*, kde jsou k dispozici zvuky zdarma pod licencí EULA.

Zvuk bylo nutné upravit, k tomu výborně posloužil program Audacity. Je multiplatformní, má velké množství funkcí a je zdarma vydávaný pod licencí GNU General Public License (GPL).

6.7 Kolizní systém

Kolizní systém je potřeba k rozpoznávání střetů mezi herními objekty. Protože jsem použil síťovou architekturu peer-to-peer, kde každý klient provádí sám simulaci hry, použil jsem jednoduchý kolizní systém, kde každý klient sleduje kolize pouze s lokálním hráčem. Při použití složitých fyzikálních enginů jako je Box2D nebo Chipmunk, je nutné provádět simulaci pouze na jednom místě, jinak není zaručena konzistence průběhu simulace na více místech zároveň.

Kapitola 7

Implementace

Aplikace je naprogramována v jazyce Objective-C s využitím objektově orientovaného přístupu a požitím frameworku Cocos2d. Síťová část je implementována pomocí frameworku GameKit s využitím služeb Game Center.

K implementaci jsem vybral herní framework Cocos2d, protože poskytuje velkou škálu nástrojů, je přehledný, jednoduchý a multiplatformní *(viz. kapitola 3.2)*.

7.1 Třída GameManager

Ve hře existuje pouze jedna instance této třídy a pomocí ní lze přepínat scény hry. Dále načítat zvuk do paměti a ovládat jej.

7.2 Třída GameObjekt

Je to základní třída pro všechny herní objekty jako je helikoptéra a střela. Obsahuje vlastnosti vypovídající o stavu objektu `characterState`, jeho poškození `health` a typu `gameObjectType`. Dále má metodu, která usnadňuje vytváření animací jednotlivých objektů do paměti. Používá se hned při inicializaci objektu, aby byly animace připraveny v paměti k okamžitému použití. Této metodě stačí předat název animace a název třídy objektu, pro který se animace vytváří. Metoda pak ze souboru ve formátu *property list* [1] se stejným názvem jako je název třídy načte informace o animaci *(viz. obr. 7.1)* a podle nich ji vytvoří. Nakonec obsahuje metodu `deleteFromScreen`, která odstraňuje sama sebe z displeje.

7.2.1 Třída Shot

Třída `Shot` reprezentuje střelu vystřelenou jedním z hráčů. Při inicializaci je nutné ji předat identifikátor, pod kterým má být vytvořena (vlastnost `ID`) a identifikátor střelce vlastnost (`shooterID`), který střelu vytvořil. Tyto identifikátory jsou nutné pro určení komu střela patří a v případě zásahu mu započítat bod. Také se používají při odesílání zpráv o zásahu ostatním klientům. Identifikátory jedné střely na více klientech jsou totiž totožné.

Vypuštění střely provádí metoda
`shootActionStartPosition:(CGPoint)position andOrientation:(float)orientation`.

[1] formát souboru používaný v operačních systémech Mac OS X a iOS k ukládání objektů

▼ idleAnim	Diction...	(3 items)
animationFrames	String	1,6
delay	Number	0,03
filenamePrefix	String	shot_
▼ explAnim	Diction...	(3 items)
animationFrames	String	14,13,12,11
delay	Number	0,02
filenamePrefix	String	shot_

Obrázek 7.1: Ukázka properti-listu s informacemi o animacích

Ta vytvoří střelu na displeji, nastaví koncový bod střely a čas, za jaký má střela zasáhnout cíl.

Každému hernímu objektu je možné nastavit stav pomocí metody `changeState:(CharacterStates)newState`. Při nastavení stavu se rozhoduje o spuštění animace a zvuku.

7.2.2 Třída Copter

Jedná se o třídu reprezentující hlavní postavu ve hře helikoptéru. Pracuje s vybavením helikoptéry a jejím stavem. Vybavení tvoří dvě střely `shotRight`, `shotLeft` a `rotor`. Obě střely jsou implementovány jako prázdné kotvy určující počáteční pozici skutečných střel. Vypuštění střel provádí metoda `makeShoot`. Ta zjistí počáteční pozici střely a vytvoří nový identifikátor. Střelu následně inicializuje a vypustí. Tato třída také obsahuje statistiky zásahů a úmrtí helikoptéry `hits` a `dies`.

O dopad zásahu na helikoptéru se stará metoda `hitByDemage:(int)demage` a o dopadu kolize s jiným vrtulníkem metoda `colision` . Zničený vrtulník se vrací do hry pomocí metody `revival`. Úplnou inicializaci vlastností vrtulníku provádí `restart` používaný při nové hře.

7.3 Třída GamePlayLayer

Tato třída řídí hru. Probíhá zde hlavní herní smyčka a to v metodě update. Ta je volána před každým vykreslením scény. V herní smyčce se zjišťuje akce s joystikem a tlačítky. Podle stavů ovládacích prvků je ovládán vrtulník. Dále je zde inicializace vrtulníků. Vlastnost `localCopter` obsahuje referenci na helikoptéru lokálního hráče. Pomocí této reference je helikoptéra řízena ovládacími prvky. Ovládací prvky jsou součástí `ControlLayer` objektu, který je v této třídě dostupný přes vlastnost `control`.

K plánování událostí jsem použil metodu `schedule`. Ta je součástí třídy `CCNode` a dovoluje naplánovat spuštění zvolené metody v časovém intervalu. Stačí jí předat selektor plánované metody a délku intervalu. Zde je potřeba naplánovat některé události s časovým zpožděním pouze jednou. To se provede odstraněním plánované metody z plánovače pomocí unschedule.

7.4 Třída ControlLayer

Všechny ovládací a informační prvky zobrazené na displeji jsou součástí této třídy. Ta je vytvoří a jejich reference uchovává ve svých vlastnostech. Pouze přes `ControlLayer` objekt je možné s těmito prvky pracovat.

7.5 Třída GMLayer

Dědí ze třídy `GamePlayLayer` a přidává do hry multiplayer část. Přijímá a odesílá zprávy ostatním hráčům. O odesílání se stará pouze tato třída a to pomocí polymorfismu. Některé metody z třídy `GamePlayLayer` jsou v této třídě přepsány tak, že provedou příslušnou akci s vrtulníkem, ale také odesílají informace o této akci ostatním hráčům.

Dále obstarává příjem zpráv a provedení akce podle zprávy. Každý vrtulník se nachází v asociativním poli tvořeným referencí na objekt vrtulníku a identifikátorem hráče vlastnícím tento vrtulník. Když například přijde zpráva o výstřelu z vrtulníku, objekt `GMLayer` vybere z pole vrtulníků ten, který patří odesílateli zprávy a oznámí mu, aby provedl výstřel. Do asociativního pole jsou také vkládány střely. Zpráva o zásahu obsahuje identifikátor střely, která helikoptéru zasáhla. Tato střela je nalezena v poli a smazána z displeje.

U příjmu zpráv probíhá dvojité dekódování. Při prvním je z dat vytvořen objekt třídy `GMessage` a jeho vlastnost `messageDate` se porovná s časem přijetí předchozí zprávy. Pokud je čas nové zprávy starší, je zpráva zahozena, protože je neaktuální. V opačném případě je vyhodnocen typ zprávy a podle něj provedeno druhé dekódování do stejného objektu jako před zakódováním.

7.6 Třída GCHelper

Třída `GCHelper` je realizována jako singleton a implementuje protokoly `GKMatchmakerViewControllerDelegate` a `GKMatchDelegate`. Implementace prvního protokolu je potřebná, aby okno Game Centr mohlo delegovat zprávy o akcích této třídě, a druhého pro vytváření zápasů prostřednictvím služby Game Center a ke komunikaci s ostatními hráči.

Metoda `findMatchWithMinPlayers` zobrazuje okno Game Center. To je zobrazeno podle vlastnosti `pandingInvite` s nabídkou pozvání přátel nebo s informacemi o pozvánce, která přišla.

7.7 Zprávové třídy

Kořenová třída pro zprávy je `GMessage`, která obsahuje vlastnost `messageDate` obsahující čas vytvoření zprávy a `messageType` určující typ zprávy. Čas vytvoření zprávy používám k určení, zda je zpráva aktuální a typ zprávy k rozhodování o jejím dalším zpracování.

Všechny objekty zprávových tříd musejí být před odesláním převedeny na třídu `NSData`. K převedení zpráv využívám třídu `NSKeyedArchiver`, která zakóduje jakýkoliv objekt implementující protokol `NSCoding` a výsledek vrátí typu `NSData`. Protokol obsahuje metody nutné implementovat pro zakódování objektu do dat a k jeho rozkódování. V této hře se tak kódují všechny odesílané objekty.

Kapitola 8

Testování a vyhodnocení

K vývoji jakékoliv aplikace patří neodmyslitelně testování. To jsem prováděl metodou white-box (česky bílé skříňky) pomocí instrumentů popsaných v kapitole 2.3. Těmi jsem sledoval a analyzoval běh aplikace při jejím používání a z dosažených výsledků jsem zjišťoval konkrétní místa chyb v kódu a v samotných algoritmech. Po opravě jsem znovu prováděl testování, abych se přesvědčil, že chyby byly skutečně odstraněny.

Zkoumal jsem také, jak často lidé hrají multiplayer hry na mobilních zařízeních, jaké bezdrátové technologie k tomu používají a jejich zkušenosti s vysokou latencí na mobilních sítích. Průzkum jsem prováděl pomocí dotazníku, umístěného na diskuzních fórech zaměřených pro příznivce mobilních telefonů. Dotazník byl také zveřejněn na sociální síti Facebook na profilu internetového magazínu *Jablíčkář.cz*, díky kterému se mi celkově podařilo nasbírat odpovědi od 172 lidí.

Z výsledků *(viz. příloha A)* vyplývá, že alespoň jednu multiplayer hru na svém telefonu zná téměř 90% respondentů, ale téměř 50% respondentů multiplayer hry hraje zřídka nebo vůbec. Tento stav je podle mého názoru zapříčiněn malým množstvím jednoduchých a graficky dobře zpracovaných online her s nápaditou myšlenkou.

Dále jsem zjistil, že respondenti jsou k internetu při hraní online her převážně připojeni pomocí technologie WI-FI, více jak 1/3 je připojena přes mobilní sítě 3. generace a jen minimum jich je připojeno přes mobilní sítě 2. generace. Online hry se tak nejvíce hrají doma nebo v práci díky stabilnímu připojení. Jen minimum hráčů se připojuje k online hře na cestách. Možná i proto přes 80% z nich pozoruje zvýšenou latenci jen někdy nebo vůbec.

Kapitola 9

Závěr

Cílem práce bylo vytvořit zajímavou multiplayer hru na operačním systému iOS s použitím některého herního frameworku.

Výsledkem je plnohodnotná Real-Time multiplayer hra implementovaná pomocí frameworku Cocos2d. Hra je zajímavá především tím, že se hraje přes internet. Díky tomu je možné ji hrát například proti svému kamarádovi kdekoliv, kde je mobilní internetové připojení.

Z vyhodnocení průzkumu, který jsem provedl, je vidět, že lidé všeobecně znají multiplayer hry na své zařízení, ale málo kdy je hrají. To přináší výzvu vytvořit koncepčně zajímavou multiplayer hru, která by přinesla mezi herní tituly nějakou inovaci a stala se tak celosvětově úspěšnou. Příkladem je multiplayer hra Draw Something s velice jednoduchým konceptem, kterou v současnosti hraje 14,6 milionů hráčů a stala se tak celosvětovým fenoménem.

Možnost pro rozšíření se nabízí celá řada, například vytvoření vlastního serveru, který by nahradil službu Game Center aby se hra mohla implementovat pro další mobilní platformy. Přidat do hry možnost vybírat si mezi různými typy vrtulníků, které budou mít rozdílné vlastnosti.

Literatura

[1] Apple Inc.: About Memory Management. https://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/MemoryMgmt/Articles/MemoryMgmt.html, [cit. 2012-04-30], [online].

[2] Clara Serrano and Beatriz Garriga and Julia Velasco and Julio Urbano and Santiago Tenorio and Manuel Sierra: Latency in Broad-band Mobile Networks. http://202.194.20.8/proc/VTC09Spring/DATA/09-18-19.PDF, 2009 [cit. 2012-04-30].

[3] Cocos2d: cocos2d Basic Concepts. http://www.cocos2d-iphone.org/wiki/doku.php/prog_guide:basic_concepts, [cit. 2012-04-30], [online].

[4] Gamua: Display Objects. http://wiki.sparrow-framework.org/manual/display_objects, [cit. 2012-04-30], [online].

[5] Kochan, S. G.: *Objective-C 2.0 : výukový kurz programování pro Mac OS X a iPhone*. Computer Press, 2010, iSBN 978-80-251-2654-7.

[6] Mark Claypool and Kajal Claypool: On Latency and Player Actions in Online Games. ftp://ftp.cs.wpi.edu/pub/techreports/pdf/06-13.pdf, 2006-07-08 [cit. 2012-04-30].

[7] Ruedebusch, T.: *UMTS Signaling: UMTS Interfaces, Protocols, Message Flows and Procedures Analyzed and Explained*. Wiley, 2005-04-01, iSBN 978-0470013519.

[8] Strougo, R.; Wenderlich, R.: *Learning Cocos2D: A Hands-On Guide to Building iOS Games with Cocos2D, Box2D, and Chipmunk*. Addison-Wesley Publishing Company, 2011, iSBN 978-0-321-73562-1.

[9] Vysoké učení technické v Brně: Měření mobilních sítí 3G v České republice. http://www.feec.vutbr.cz/aktuality/info/3g_tiskova_zprava201103.pdf, [cit. 2012-04-30], [online].

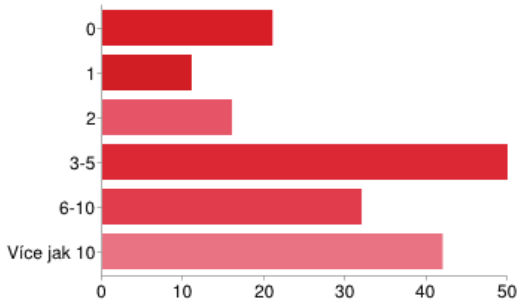
[10] Yahn W. Bernier: Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. https://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization, [cit. 2012-04-30], [online].

Příloha A

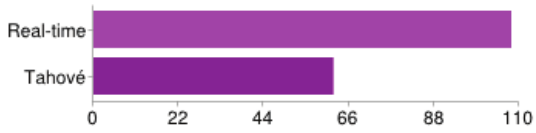
Plakát

VÝSLEDKY PRŮZKUMU HRANÍ ONLINE HER NA MOBILNÍCH TELFONECH A VÝSLEDNÝ VZHLED VYTVOŘENÉ HRY

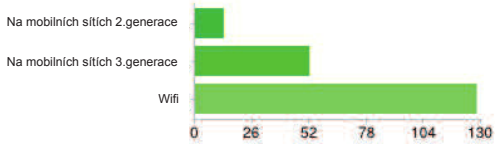
Kolik znáte online her na váš telefon?



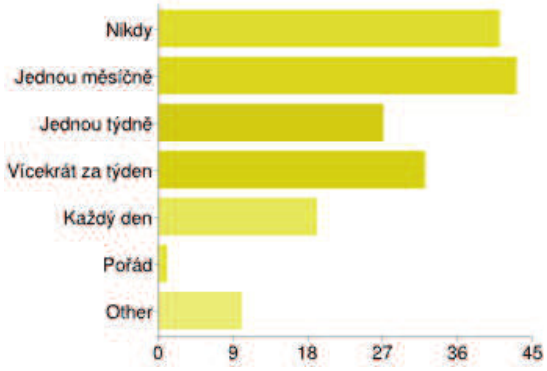
Jaký typ onlineher hrajete?



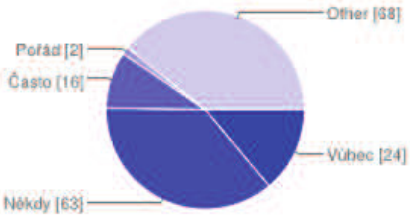
Na jakém připojení hrajete online hry v telefonu?



Jak často hrajete online hry?

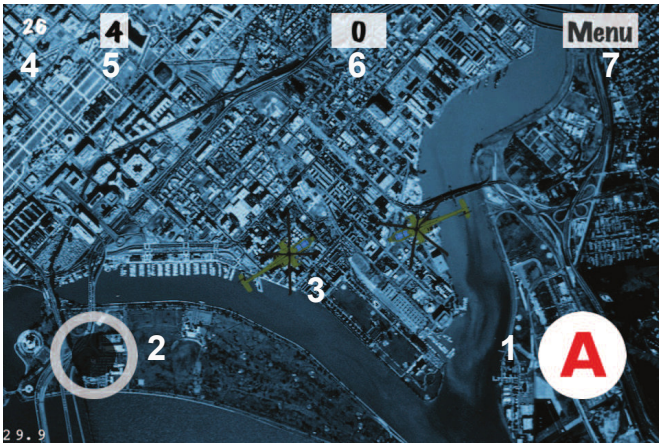


Pokud hrajete Real-Time online hry přes mobilní sítě, jak často pozorujete vysokou latenci?

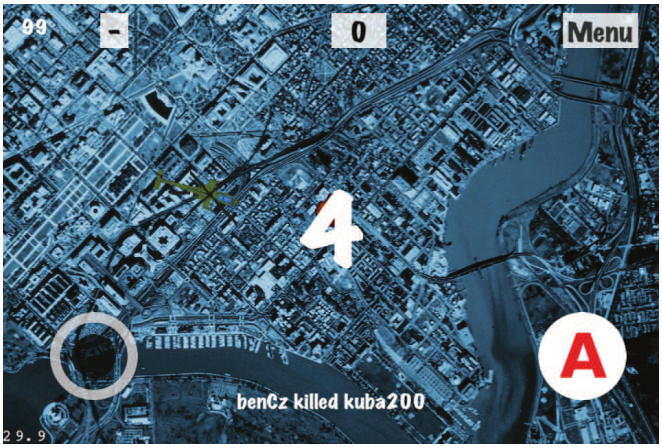


ovládáni hry

1- tlačítko výstřel 2- joystic - ovládání pozice helikoptéry
3- ovládaná helikoptéra 4- ping 5- počet střel v zásobníku
6- počet zabitých protivníků 7- tlačítko menu



zásah



konec hry