Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Vývoj mobilní aplikace s využitím UNITY

Bc. Miloš Veselý

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Miloš Veselý

Informatika

Název práce

Vývoj mobilní aplikace s využitím UNITY

Název anglicky

Mobile application development using UNITY

Cíle práce

Hlavním cílem práce je analyzovat, navrhnout a implementovat mobilní aplikaci využívající možnosti grafického engine Unity. Dílčími cíli je popsat možnosti využití Unity při tvorbě aplikací využívajících 2D a 3D grafiku v obecné rovině a dále se zvláštním zaměřením na mobilní aplikace a zhodnocení možností a výhod, které využití Unity přináší.

Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných poznatků budou popsány možnosti využití Unity při tvorbě aplikací. Dále bude vybranou metodou analyzována a posléze navržena a implementována ukázková mobilní aplikace využívající Unity. Aplikace bude otestována a bude provedeno zhodnocení výhod, které přináší využití Unity oproti vývoji bez podobné technologie. Dále budou nastíněny možnosti dalšího rozvoje aplikace, např. s využitím dalších funkcí Unity, případně souvisejících technologií.

Doporučený rozsah práce

60-80 stran

Klíčová slova

iPhone, Unity, SDK, C#, IOS, objekt, model, 3D aplikace

Doporučené zdroje informací

[1] FELICIA, Patrick. Getting started with Unity. Birmingham: Packt Publishing, 2013, 170 p. ISBN 978-1-84969-584-8.

[2] MARK, D., NUTTING, J., LAMARCHE, J. and OLSSON. F. Beginning iOS6 development: Exploring the iOS SDK, 1. vyd. New York: Apress, 2013, 179 s. Průvodce (Grada). ISBN 978-143-0245-131.

[3] PIERCE, Gregory. Unity iOS game development: beginners guide : develop iOS games from concept to cash ow using Unity : [learn by doing : less theory, more results]. Olton Birmingham [England]: Packt Publishing, 2012, 299 p. ISBN 978-1-84969-040-9.

[4] SHARP, John. Microsoft Visual C# 2010: Krok Za Krokem Vyd. 1. Brno: Computer Press, 2010, 696 s., ISBN: 9788025131473.

[5] VÁVRŮ, Jiří. IPhone: vývoj aplikací. 1. vyd. Praha: Grada, 2012, 179 s. Průvodce (Grada). ISBN 978-80-247-4457-5.

Předběžný termín obhajoby

2015/06 (červen)

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 24. 03. 2015

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Vývoj mobilní aplikace s využitím UNITY" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. Března 2015 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Jiří Brožkovi Ph.D. za rady poskytnuté během psaní diplomové práce. Dále mým rodičům a Matěji Mňoučkovi za pomocnou korekci chyb.

Vývoj mobilní aplikace s využitím UNITY

Mobile application development using UNITY

Souhrn

Diplomová práce pojednává o možnostech tvorby mobilních aplikací s využitím Unity 3D enginu. Teoretická část rozebírá konkurenční programy pro vývoj aplikací, přičemž vyhodnocuje, které z uvedených programů jsou spíše zaměřeny na mobilní platformy a ty které jsou naopak vhodnější pro vývoj aplikací na platformy PC, Xbox či Play Station. Součástí teoretické části práce je charakteristika Unity 3D enginu a C# jazyku, jenž je hlavním programovacím jazykem využitým v rámci vyvíjené aplikace.

Empirická část práce vysvětluje postupy tvorby aplikace od prázdné scény až po finální fungující hru Garbage ride. V této části jsou uvedeny analýzy předcházející vývoji. Proces tvorby fiktivního světa a všech nastavení, jež jsou součástí Unity. Hlavní kapitolou praktické části je detailní popis skriptů, které dávají hře funkčnost. Poslední kapitola obsahuje shrnutí celé práce, vypsání výhod Unity a návrhy na možná zlepšení vytvořené hry. Výsledná aplikace je cílena na zařízení iPhone 4, na kterém jsou prováděny všechny testy aplikace a je podle něj upravené rozvržení všech částí hry.

Summary

Diploma thesis deals with options of making applications using Unity 3D engine. Theoretic part describes competitive software appropriate for application development, while evaluates which of the software is more suitable for mobile application and which is better for other platforms like PC, Xbox and Play Station. The theoretical part includes characteristics of Unity 3d engine and C# language which is the main programming language used within developed application.

Empiric part contains procedure of development application from empty scene to final working game “Garbage ride”. In this part analysis of foregoing development are listed. Process of making imaginary world and all these settings which are connected to Unity. Main chapter of empiric part is a detailed description of scripts which give functionality to the game. Last chapter contains summary of whole work, states benefits of using Unity used during development and suggestions for next improvements of developed game. Final application is aimed to iPhone 4 on which are performed all tests and the application is adapted to.

Klíčová slova: iPhone, Unity, SDK, C#, iOS, objekt, model, 3D aplikace

Keywords: iPhone, Unity, SDK, C#, iOS, object, model, 3D application

1 Obsah

1. Úvod

V lednu roku 2007 byl oficiálně představen první iPhone od firmy Apple. Uveden na trh byl téhož roku v červnu. Tato událost odstartovala vlnu smartphonů, které jsou v současnosti jedním z nejžádanějších zařízení na trhu spotřební elektroniky. S příchodem iPhonu se otevřel tzv. iStore neboli obchod s aplikacemi. V roce 2008 Apple představil beta verzi SDK, která umožnila vytváření nativních aplikací třetím stranám. Pro mnoho vývojářů tento krok znamenal zásadní změnu v jejich životě, protože získali nový typ zařízení, na které lze naprogramovat nejrůznější aplikace. Jako první se začaly objevovat aplikace typu synchronizování kontaktů, záznam zvuků, messengery, předpovědi počasí, vytváření poznámek spojené s fotkami nebo druh poznámkového bloku vhodný třeba pro vytváření nákupního seznamu. Krátce na to přišel na tento trh nový segment aplikací mobilní hry. Možnost, zahrát si kdykoli a kdekoli pěkné, jednoduché a zábavné hry na dotykovém displeji s využitím multi-doteků, se stala obrovským hitem na celém světě a lze říci, že dodnes přetrvává.

V současnosti se trh aplikací rozděluje do dvou hlavních kategorií podle typu operačního systému (OS), existuje i třetí kategorie pro Windows mobile, ale kvůli nízkému zastoupení, zde není více zmiňována. První kategorie se věnuje aplikacím pro Apple zařízení, přesněji pro iOS systém. Druhou kategorii ovládl operační systém Android od Googlu. Z těchto důvodu jsou aplikace vyvíjeny podle cílového OS, čímž se situace pro vývojáře komplikuje, jelikož si musí vybrat, pro které přístroje budou aplikace vytvářet. Výhodami v softwarech jako je Unity, jsou volby mobilních zařízení a export do jejich kódového jazyka podle zmíněných systémů. Lze vytvořit jednu aplikaci, kterou je možné posléze exportovat a používat na více zařízeních s různými systémy. Nutností je aplikaci přizpůsobit na míru zařízení, pro které je vytvářena. Aplikace, jež bude vytvářena v praktické části této práce, bude zaměřena na mobilní telefon iPhone 4. Z tohoto důvodu se nebude rozebírat situace kolem konkurenčních zařízení a tvorby pro ně. Vše bude vztaženo k možnostem využití Unity 3D enginu pro vývoj aplikací na mobilní telefon iPhone.

V roce 2008 byl počet aplikací v iStore kolem pěti tisíc, kdežto o rok později toto číslo přesahovalo hranici dvaceti tisíc. Neuvěřitelný růst tohoto trhu signalizoval obrovský potenciál odvětví. Ke konci roku 2014 eviduje iStore kolem 1,3 miliónu aplikací s počtem stažení 75 miliard. (1) Toto neskutečné číslo desetkrát přesahující počet populace, jasně znázorňuje popularitu iPhonů a velikost trhu točícího se kolem mobilních aplikací.

V současné době vývoj aplikace sebou nese jisté komplexnosti, bez kterých je téměř nemožné docílit vyšších prodejů aplikace, hlavní roli hraje účinný marketing. Touto tématikou se tato práce zabývat nebude. V rámci práce je uvedení možností jak lze aplikace vytvořit, jaké existují použitelné jazyky a jejich výhody. Poslední kapitoly se věnují vytvoření aplikace s detailním popsáním průběhu tvorby.

Jak je uvedeno v názvu práce, hlavním nástrojem pro tvorbu aplikace je engine Unity 3D, představující vývojové prostředí pro různé platformy s možností velké podpory jak od tvůrců, tak od samotných uživatelů v podobě velkého množství blogů a webů zabývajících se tvorbou aplikací.

2. Cíl práce a metodika

Hlavním cílem práce je seznámení čtenáře s možnostmi tvorby mobilních aplikací pomocí Unity 3D enginu. Teoretická část se zabývá popisem různých univerzálních nástrojů pro vytvoření mobilní aplikace, uvedením možných programovacích jazyků a popisem frameworků. Součástí této kapitoly je porovnání vývojových prostředí s enginem Unity 3D formou analýzy dostupných zdrojů. Zjištěné informace jsou dále využívány v praktické části, která čerpá z kapitol věnujících se Unity 3D enginu.

Praktická část je zaměřena na vývoj mobilní aplikace, konkrétně se jedná o hru pro mobilní telefon iPhone, která demonstruje široké užití funkcí zmíněného prostředí Unity. Hra se zabývá svozem komunálního odpadu popelářským vozem. Cílem hry je sběr kontejnerů v co nejkratším časovém úseku za dosažením určitého počtu bodů pro postup do další úrovně. K řízení vozu je využito gyroskopu, což je vlastnost iPhonů od verze 4. Tento druh ovládání tj. pomocí náklonu mobilního zařízení ze strany na stranu je vhodný pro řízení automobilu z důvodu přesnější odezvy než v případě dotykových tlačítek. Zároveň se tím stává řízení jednodušší a zcela intuitivní pro každou věkovou skupinu.

Výstupem z praktické části je funkční aplikace v beta verzi, která bude splňovat stanovené předpoklady, definované v analýzách předcházejících vývoji. V budoucnu by dále mohlo být vyvinuto multiplayer prostředí, které by umožnilo hrát hru ve více hráčích, což by zajisté přineslo větší oblibu a zajímavější možnosti rozvoje hry.

K vytvoření aplikace je využit jazyk C# a javascript. Tyto dva programovací jazyky jsou hlavním prostředkem Unity pro tvorbu skriptů. Samotný Unity editor obsahuje Asset store, kde je k dispozici množství modelů pro vytváření fiktivního světa tzv. prostředí pro pohyb popelářského vozu. Z důvodu obsáhlosti jsou použity modely z Asset storu, které jsou uspořádány do podoby malého města. Hlavní část práce spočívá v tvorbě skriptů pro systém hry, popelářský vůz, uživatelské prostředí a celkovou kooperaci těchto prvků dohromady.

Přínosem této práce bude demonstrace možností Unity 3D enginu při tvorbě mobilní aplikace. Budou zde zmíněny postupy aplikované pro jednotlivé skripty a uvedeny funkce Unity ulehčující práci vývojáře. Skripty budou vytvářeny v MonoDevelop

Frameworku, který je součástí Unity 3D. V průběhu vývoje budou vkládány obrázky z grafického prostředí a úryvky skriptů, jež budou zajišťovat hlavní funkcionalitu hry.

3. Teoretická východiska

V teoretické části práce jsou představeny konkurenční softwary pro tvorbu aplikací. Jsou k nim uvedeny základní charakteristiky a poté je zhodnocena jejich vhodnost použití z autorova pohledu. Krom toho, tato kapitola obsahuje také základní popis Unity 3D enginu a způsob skriptování v něm. Poslední část se věnuje jazyku C#, jakožto hlavnímu jazyku, ve kterém je vyvíjena aplikace. Dále je popsána základní syntaxe C# a vysvětlena objektová logika programování v něm.

3.1. Vysvětlení základních pojmů

Součástí této práce je charakteristika konkurenčních programů k Unity 3D a proto jsou zde zmíněny základní pojmy, které jsou v této tématice hojně využívány a bez nichž by se čtenář neznalý problematiky mohl v textu ztrácet. Seznam těchto pojmů je vypsán níže i s popisem hlavních vlastností.

- IDE (Integrated Development Enviroment)

  Pojem IDE, z anglického překladu integrované vývojové prostředí, představuje program zaštiťující podporu funkcí pro vytváření jiných programů. Mezi hlavní vlastnosti patří editor zdrojového kódu, kompilátor, interpret a debugger. Většina IDE podporuje jeden čí více programovacích jazyků. Mezi nejznámější patří NetBeans, který podporuje jazyky jako Java, HTML5, PHP, C/C++. (2)

- Framework

  Jednoduše řečeno Framework rozšiřuje vlastnosti IDE, tím že přidává funkce a knihovny, které usnadňují tvorbu korektních aplikací. Vývojáři ve frameworkách využívají implementované funkce na kontrolu kódu a správnou kompilace do konečné podoby. Velmi populárním Frameworkem pocházejícím z České republiky je Nette Framework od Davida Grudla. (3)

- API (Application Programming Interface)

  Aplikační programovací rozhraní je formát jazyka a zprávy použitý programem pro zajištění komunikace s operačním systémem nebo dalšími programy. Ve zkratce je

API zprostředkovatelem mezi uživatelem a operačním systémem. API je definován použitým jazykem a vlastnostmi svázanými s ním. (4)

- SDK (Software Development Kit)

  Jedná se o celek nástrojů pro tvorbu aplikací na určitou platformu. SDK přidává takové vlastnosti IDE, API a Frameworků, které umožňují vývoj aplikací podle druhu zařízení. (5)

- Objektové programovací jazyky

  Je to druh programovacích jazyků využívající objektový přístup pro tvorbu programů. Prvotní snahou objektových jazyků bylo přiblížit myšlení programu více reálnému světu. Základními kameny objektového programování jsou objekty, třídy a vztahy mezi nimi. Hlavními představiteli objektového programování jsou jazyky Perl, Smalltalk, Java, C++, C#, Visual Basic .NET, Lisp, PHP, Python, Ruby Object a Pascal. (6)

- GUI (Graphical User Interface)

  Graphical User Interface do češtiny přeloženo jako grafické uživatelské rozhraní je grafické prostředí, které lze intuitivně ovládat pomocí grafických ovládacích prvků. První známější GUI vzniklo v roce 1984 v Americké společnosti Apple a velmi rychle získalo oblibu u uživatelů. V současnosti je nejpoužívanější a nejrozšířenější GUI Windows od firmy Microsoft. (7) (8)

- Kompilátor

  Kompilátor slouží pro převod programovacího kódu do strojového jazyka, tak aby byl čitelný pro procesor. Jelikož jsou programy psány v programovacím jazyce, který je srozumitelnější člověku, je nutno je převézt do podoby srozumitelné pro procesor tj. do jedniček a nul.

- Simulátor

  V počítačové terminologii je simulátor nástroj pro simulaci programu či hry. V této práci bude simulátor použit pro testování aplikace, aby se mohly odladit chyby, které nelze zjistit v průběhu vývoje.

- Debugger

  Debugger je nástroj, součást většiny IDE a frameworků pro hledání chyb v programovacím kódu.

3.2. Mobilní aplikace

Název mobilní aplikace pochází z anglického překladu mobile app, přičemž pod tímto názvem se skrývá program vytvořený pro chytré telefony (smartphony), tablety a phablety. Poslední uvedený přístroj představuje mobilní telefon ve velikosti zapadající mezi dva známější přístroje, a to smartphony a tablety.

Mobilní aplikace se snaží co možná nejvíc využít možností dotykového ovládání s kombinací gyroskopu a kompaktních rozměrů těchto zařízení. Oproti počítačovým hrám je kladen větší důraz na hratelnost a jednodušší ovladatelnost. Právě zmíněné technologie doteku a náklonu velmi přispívají k docílení těchto vlastností ve spoustě vydávaných aplikací. Třetí vlastnost tzv. kompaktnost hraje důležitou roli v momentech, kdy není k dispozici jakýkoli jiný prvek zábavy, je vždy jednoduché sáhnout do kapsy a brázdit po webu nebo si na chvíli zastřílet malými ptáčky na různé druhy křehkých konstrukcí. (8)

Celá historie aplikací se odehrála v posledních šesti letech. Během této doby se stáhlo neuvěřitelných 75 miliard aplikací na všechny přístroje značky Apple. Prvenství v této kategorii hraje aplikace Facebook, jenž je věrnou kopií své webové předlohy sociální sítě přizpůsobené pro mobilní zařízení. Aktuální žebříček podle počtu stažení uvedený níže, je rozdělen do dvou sloupců na aplikace placené a zdarma.

Aplikace zdarma	Placené aplikace
Facebook	WhatsApp Messenger
Skype	Angry Birds
Viber	Fruit Ninja
YouTube	Angry Birds Seasons
Instagram	Camera+
Shazam	Cut the Rope
Facebook Messenger	Angry Birds Space
Angry Birds Free	Sleep Cycle alarm clock
Google Translate	Angry Birds Star Wars
Fruit Ninja Free	Hipstamatic

3.3. Možnosti programů pro tvorbu aplikací

Současný svět je do značné míry ovlivňován internetem, který zasahuje do života téměř celé populace. Někteří lidé patří do skupiny vývojářů, kteří vytvářejí celé weby, programy, aplikace a ostatní software. Druhá skupina jsou ti, kteří využívají internet a různé aplikace v dennodenním životě za účelem jeho usnadnění. Chtějí, aby vše fungovalo tak jak je v popisu, ale netuší, jakým způsobem se vše provádí, ani je to vlastně nezajímá, hlavní je, že to funguje.

První zmiňovaná skupina vytváří taktéž programy pro ty, jež chtějí současně přispívat vlastní tvorbou, ale nemají dostatečné znalosti, aby byli schopni si programy sami napsat. Pro ty v dnešním světě existuje velké množství programů. Tyto programy (často celá SDK) podporují jednodušší tvorbu aplikací, oproti kompletnímu naprogramování celé aplikace v programovacím jazyku. Nabízejí uživateli prostředí, které má možnosti jednoduchého vkládání objektů a následného přiřazování předdefinovaných funkcí. Tímto způsobem lze vytvořit např. ikonu na obrazovce mobilního zařízení odkazující na webovou stránku. Přidat objektu vlastnosti, které by jinak bylo nutné složitě naprogramovat, lze pouhým přidáním vhodné funkce. Takto je možné vytvořit reálnou

fungující aplikaci bez znalosti konkrétního programovacího jazyka. Samozřejmě to není tak lehké, ale s jistými základními znalostmi, to možné je. (9) (10)

Většina programů pro tvorbu mobilních aplikací využívá objektově orientované jazyky spadající do kategorie vyšších jazyků, které jsou blíže abstrakci z pohledu lidského myšlení. Jednou z výhod těchto jazyků je např. oproštění od problému s přiřazováním paměti procesoru, jež se objevuje v nižších jazycích. Mezi objektově orientované jazyky patří:

- C++
- Java
- JavaScript
- C#
- HTML 5
- CSS

Do kategorie SDK programů lze zařadit i Unity 3D engine vhodný pro vývoj aplikací na různé platformy. Celá SDK čerpají výhody hlavně z celistvosti, obsahují vše potřebné k vytvoření a nasazení aplikace do běhu. Tím odpadá potřeba kombinovat více softwarů pro docílení stanoveného cíle. Mezi hlavní alternativy Unity 3D enginu se řadí následující programy:

- Xcode
- MonoGame
- Xamarin
- GameMaker
- Source engine
- Unreal engine
- CryEgine

Níže jsou charakterizovány konkurenční programy k Unity enginu, přičemž jsou zdůrazněny především vlastnosti, jež zpříjemňují práci vývojáři. Řazení programů

reflektuje zaměření softwarů pro tvorbu aplikací od více specifikovaných k univerzálním, proto se začíná s nativním Xcode a mezi posledními jsou uvedeny programy nejpodobnější Unity 3D, jedná se o celé enginy tj. programy s vlastním způsobem vykreslování grafické stránky aplikace. Unreal engine, CryEngine a Source engine jsou objemné graficky propracované SDK pro tvorbu moderních aplikací se zaměřením na výkonné platformy. (10) (11)

3.3.1. Xcode

Xcode je soubor nástrojů od společnosti Apple pro vytváření softwarových aplikací na platformy iOS a OS X. Tyto operační systémy jsou podporovány zařízeními Macbook, iPhone a iPad. Prostředí Xcode obsahuje Cocoa a Cocoa Touch framework založený na objektově orientovaném prostředí využívajícím základní knihovny C jazyka. Právě Cocoa framework je ideální balík funkcí vylepšující vlastnosti Xcode z hlediska přidání základních funkcionalit pro iPhone zařízení.

Xcode podporuje celou řadu programovacích jazyků C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, Rez a Swift. Z nich nejpoužívanější jsou C, C++, Objective-C, Objective-C++, Java a AppleScript.

Velkou předností Xcode jsou nástroje integrované v instalaci, a to především tyto:

LLVM kompilátor

LLVM kompilátor funguje jako klasický kompilátor, ale je navržen, tak aby převedený strojový kód byl přesně optimalizován pro procesory v zařízení Apple.

iOS Simulator

Představuje nástroj pro možnost testování aplikace přímo skrze mobilní zařízení, které se zobrazí na displeji počítače a tím ukáže vývojáři, jak bude aplikace reálně vypadat na iPhone telefonu. Jedná se o jednu z hlavních vlastností Xcode umožňující pohled na vytvářené aplikace v průběhu vývoje.

Interface Builder

Nástroj pro vytváření uživatelského prostředí bez nutnosti použít jakýkoli programovací jazyk. Pomocí jednoduchého spojování navrhnutých obrazovek lze vytvořit prostředí aplikace, ve kterém se uživatel bude pohybovat.

Instruments

Instruments patří mezi nástroje pro testování a analýzu výkonu aplikace. Obsluha spočívá v nadefinování vlastních instruments, tak aby zaznamenávaly potřebné informace o aplikaci a sbíraly data, která lze zkoumat za účelem zlepšení chodu celé aplikace. (10)

3.3.2. MonoGame

Monogame vznikl jako otevřená implementace Microsoft XNA 4 Frameworku. Tento Framework, soubor knihoven tříd určených pro vytváření her, je rozšířením k programu Microsoft Visual Studio. Spojením těchto funkcí vzniklo takzvané XNA Game studio, které je zdarma ke stažení, ale má podporu pouze pro systémy Windows. Cílem vývojářů Monogame bylo umožnit převod vytvořených her pro Xbox, Windows a Windows phone na systémy iOS, Android, Mac OS X, Linux a Windows 8 Metro. V podstatě se jedná rozšíření základního Frameworku o podporu dalších zařízení. Tím docílili větší užitelnosti programu v rámci dostupných platforem. Tvůrcem těchto rozšíření je José Antonio Leal de Farias, aktivní člen XNA komunity.

Název Monogame byl vytvořen v roce 2011, kdy už existoval Mono XNA, které nabízelo export aplikací na iPhone. Jakmile vyšlo oficiálně Monogame, byla podpora rozšířena na Android, Mac, Linux a OpenGL pro Windows. Monogame je založen na jazycích C# a Visual Basic. Je to open-source projekt, takže je možné provádět úpravy kódů dle libosti uživatele, pokud na to má dostatečné vědomosti. Je oblíben, protože při vytváření jedné hry je možné si zvolit více platforem, aniž by bylo potřeba složitě měnit kód hry. (11) (12)

3.3.3. Xamarin

Xamarin je dalším univerzálním nástrojem pro tvorbu aplikací. Obdobně jako Monogame představuje i Xamarin Framework nástavbu na Microsoft Visual Studio nebo MonoDevelop, který je téměř stejný jako Xamarin studio. Vývojáři se zaměřili na uživatelskou přívětivost a klasické dodržení standardů. Díky kvalitní podpoře platforem bylo docíleno možnosti vytvořit aplikaci na míru zařízení. Pomocí pluginů (doplňků) na Android či iPhone se přistupuje k hlavním funkcím, umožňujícím vývoj aplikací na zmíněné

platformy. Xamarin má jako hlavní programovací jazyk C#, o kterém tvrdí, že to je nejlepší jazyk pro vývoj mobilních aplikací. S jistotou lze říci, že Xamarin míří především na mobilní aplikace na platformách iOS, Android a Windows phone. Dle vzhledu se nejvíce podobá nativnímu nástroji Xcode od Applu. Velkou výhodou Xamarinu je podpora různých knihoven jako jsou SQlite, Json.NET nebo ReactiveUI na všech platformách. Tím se umožnilo vytvářet jednu aplikaci pro více zařízení.

Xamarin zároveň nabízí pro vývojáře možnost použít Xamarin Studio, které má podporu Windows i Mac OS. Xamarin umožňuje zobrazení vytvářené aplikace v reálném čase v průběhu vývoje přímo v IDE. Tato možnost, vyzkoušet si každý přidaný skript, aniž by bylo potřeba kompilovat celou aplikaci, je pro vývojáře nesmírným zjednodušením a úsporou času, proto je součástí většiny programů pro tvorbu aplikací. (13)

3.3.4. GameMaker

Další z programů ulehčující tvorbu mobilních aplikací je GameMaker v současnosti vytvářen pod záštitou YoYo Games. Na webových stránkách je v popisu GameMakeru uvedeno: „Studio s plně integrovaným vývojovým prostředím, dává sílu vytvářet profesionální hry bez jakékoli znalosti programování.“ (14) Toto prohlášení není možné brát doslova, jelikož komplexní herní mechanismy vždy potřebují jistou úpravu kódu pro zajištění celkové funkcionality. Ale i přesto je pravda, že vytvořit jednoduchou hru bez komplexních znalostí programování, je možné. Pomocí techniky Drag and drop, která je hlavním systémem tvorby her v GameMakeru je možné objektům např. přiřazovat funkce jako pohyb, reakce na kliknutí či změnu vlastností objektů. V posledních letech je citelná snaha od tvůrců těchto programů nabídnout uživatelům, co možná nejlehčí způsob, jak vyvinout svou vlastní aplikaci s využitím implementovaných funkcích. GameMaker obsahuje podporu vlastního jazyka GML, který slouží pro větší unifikování hry, oproti nastavení vlastností pouze pomocí ikon. Jazyk GML patří mezi vyšší jazyky, jeho syntaxe je odvozena z jazyka C.

Velkým usnadněním pro vývojáře je možnost získat modely, obrázky, animace, zvuky a fonty pouhým stažením assetu (volně česky přeloženo jako balíku) z webu podporovaného programem. Dlaší výhodu představuje GameMaker v možnosti

exportovat vytvořenou aplikaci na téměř všechny známé platformy včetně těch nejnovějších (iOS8, Windows phone 8, PS4, XBOX One). Program GameMaker je především vhodný pro začínající uživatele, kteří chtějí získat snadno a rychle zkušenosti s tvorbou základních her a hlavně pochopit logiku tvorby. (14)

3.3.5. Source Engine

Herní engine Source byl vyvinut společností Valve a v poslední době je velmi oblíben, zejména díky titulům, které vévodí herním statistikám. Prvně se objevil ve známé hře Half Life 2 v roce 2004. Během následujících let byl Source engine vylepšen a použit v titulech Team Fortress a Left 4 dead. Životní cyklus programů společnosti Valve se dosti vymyká obecným zvyklostem, protože Source engine byl vždy pouze vylepšován, doposud ale nebyla oznámena nová verze. Valve také vydala Source Development Kit, který obsahuje takové funkce jako je HDR (High Dynamic Range), což je dynamický rozsah zobrazení. Jedná se kompletní balík funkcí pro vývoj moderních 3D her včetně fyzických vlastností, síťové podpory, flexibility. Z pohledu autora je grafická stránka Source enginu poněkud méně detailní oproti jiným enginům. Zato kvalita vykreslování a systém materiálů je na vysoké úrovni. Zajímavou změnou je zajisté odlišný způsob použití materiálů. Material v Source enginu obsahuje informace nejenom z čeho je vyroben, ale také jak se bude rozpadat při zničení, jak bude znít a jakou hodnotu hustoty a váhy má.

Podpora platforem v Source enginu je skromnější a spíše zaměřena na výkonnější zařízení. V Source enginu lze tvořit hry pro XBOX 360, Play Station 3, Mac OS X, Linux, Windows a nově od roku 2014 i pro Android. (15)

3.3.6. Unreal engine

Pro hráče velmi známý engine, na kterém funguje velká část graficky propracovaných her. Unreal engine byl prvně použit pro stejnojmennou hru Unreal, jež v 90. letech 20 století patřila mezi nejlepší střílečky z pohledu první osoby. V té době se jednalo o verzi Unreal engine 1. Postupem času přišly další verze s označením 2, 3, 4. V současnosti jsou pomocí Unreal enginu vytvářeny kvalitní hry s různou tématikou jako Batman: Arkham City, Bioshock, Borderlands, Gears of War, Mass Effect, XCOM: Enemy

Unknown. Původním návrhářem prvního Unreal enginu byl Tim Sweeney, on sám dokázal sjednotit do jediného enginu rendering, detekci kolizí, umělou inteligenci, viditelnost, networking i správu souborového systému. Ostatní verze už byly vytvářeny pod společnostmi, jež zakoupily práva na Unreal engine. V roce 2015 se majitelé Unreal enginu rozhodli nabídnout nejnovější verzi zdarma ke stažení. Jediná podmínka je, že 5 % z příjmů z vydané aplikace půjde na vrub společnosti. Za tímto krokem jistě stojí snaha konkurovat ostatním softwarům a rozhodnutí zpřístupnit vysoce profesionál nástroj pro vývoj všem. (16)

Bezesporu je možné prohlásit, že se jedná o jeden z nejlepších grafických enginů současnosti. Jádro enginu je napsané v C++, za účelem zjednodušení tvorby her, byl vyvinut skriptovací jazyk UnrealScript, jež usnadňuje práci vývojářům, jelikož není nutné zasahovat do jádra systému, nýbrž pouze měnit skripty. Hlavní předností je zejména podpora všech důležitých platforem, výborná grafická propracovanost a kvalitní editor nabízející možnosti profesionálům vytvářet kvalitní aplikace rychle a snadno. (17)

3.3.7. CryEngine

Jedním z nejpropracovanějších enginů, co se týče grafické stránky, je CryEngine. Původ CryEnginu je zajímavý, protože byl prvně použit jako technologické demo pro Nvidii (výrobce grafických karet) a jelikož byl grafický výstup natolik kvalitní, rozhodli se v něm vytvořit počítačovou hru. CryEngine je původem z Německého vývojářského studia Crytek. První hrou vyvinutou v CryEnginu byla střílečka Far Cry, která dominovala zejména grafickým pokrokem zobrazovaného světa.

Celé CryEngine SDK obsahuje balík pro vývoj na platformy PlayStation3, PlayStation 4, Xbox 360, Xbox one, Wii U, Windows, Linux, iOS a Android. Kompletní SDK se originálně nazývá Sandbox editor a je k dispozici za měsíční poplatek necelých 10 eur. Jako většina SDK nabízí řadu nástrojů pro vývoj obsáhlých her. Těmito nástroji jsou např. fyzické vlastnosti, skriptování, pokročilé osvětlování, 3D audio, shadery, animace, dynamický zvuk. Díky chytrým algoritmům nabízí ze všech enginů nejlepší možnost vytvářet neuvěřitelně obrovské světy, které se kvalitou zobrazení nejvíce podobají

reálnému světu. Hry vyvíjené v CryEnginu jsou většinou graficky velmi náročné, a proto jsou spíše vhodné pro konzole a výkonnější počítače. (18) (19)

3.3.8. Unity 3D engine

Stěžejní vývojový nástroj požívaný v této práci je Unity 3D engine verze 4.6. Tento program byl vybrán autorem práce z důvodů dřívějších zkušeností s tímto programem a pro demonstrování jeho možností při tvorbě aplikací. Na stránkách tohoto enginu je uvedeno: „ Unity je herní vývojový ekosystém“. Tato věta mluví za vše. Unity obsahuje celou sadu nástrojů pro tvorbu aplikací všeho druhu. Pokud již není něco součástí, lze snadno a jednoduše stáhnout asset rozšiřující tyto možnosti stejně jako v GameMakeru. V Unity je možné vytvářet aplikace pro většinu platforem. Díky rozmanitosti stovek druhů kvalit a zejména pak podpoře od velké komunity, soustředěné kolem tohoto nástroje, lze relativně rychle vyvinout svou první aplikaci. (20)

V brzké době se má publikovat nová verze Unity 5. Velkým přínosem pro vývojáře bude 64 bitový editor, který zajistí dostatek operační paměti pro obsáhlejší hry. Právě nedostatek paměti způsoboval časté pády programu, v nové verzi je tento problém již vyřešen. Unity 5 zároveň přináší řadu vylepšení, mezi která lze zařadit především lepší výpočet stínů, komplexnější audio systém, zmíněný 64 bitový editor a spoustu dalších prvků. Současně se vedení Unity rozhodlo zpřístupnit některé z funkcí, jež byly k dispozici pouze v placené verzi Unity 4.6. (21)

Unity podporuje vývoj i 2D her, které jsou z pohledu mobilních technologií stále žádané. V Unity není problém vytvořit karetní hry nebo 2D arkády. Opět je možné využít implementovaných funkcí, jež ulehčí vývoj. Pomocí animace spritů (2D obrázky) lze udělat postavu hýbající nohama a rukama. Postup při vytváření ovládacích prvků je stejný jako ve 3D hrách. Často jsou 2D hry více zaměřené na logickou stránku než na grafiku. Celkově se vývoj 2D a 3D her až tolik neliší, spíše je hlavní rozdíl v náročnosti 3D titulů, jež se snaží dostat kvalitního zobrazení reálného či smyšleného světa v podání složitých modelů vytvářených na míru aplikace.

Základními jazyky, které Unity nabízí, jsou C#, JavaScript a Boo. První dva jsou známé a již zmiňované programovací jazyky vhodné pro tvorbu skriptů v celku

vytvářejících moduly pro ovládání jednotek, umělé inteligence či interakce mezi objekty. Unity k tvorbě skriptů využívá IDE MonoDevelop, jenž je univerzálním programovacím prostředím pro jazyky C#, F#, Visual Basic, .NET, C/C++, Vala. Psaní skriptů je tím pádem o to snazší s pomocí implementovaných nástrojů předcházející vzniku chyb v kódu. Uvedený jazyk Boo je objektově orientovaný programovací jazyk pro .NET a Mono, inspirován syntaxí Pythonu (starší logický programovací jazyk). Jelikož se autor rozhodl zaměřit se na C# a JavaScript, jiný výše zmiňovaný jazyk nebude v práci použit. (22)

3.4. Porovnání Unity 3D enginu

Vzhledem k tomu, že v práci jsou uvedeny konkurenční programy pro tvorbu aplikací, je vhodné uvést jejich výhody a nevýhody v porovnání s Unity enginem. Z charakteristik programů výše vyplývá, že Unity patří mezi univerzály tj. programy nabízející podporu všech platforem. Z vybraných programů je multiplatformní také Unreal engine a GameMaker. MonoGame nemá podporu iOS 8, kdežto Xamarin se zaměřuje pouze na mobilní platformy. Samozřejmostí je separace Xcode pramenící z uzavřenosti Apple produktů, která se nezapře ani v tomto případě. Jak již bylo napsáno Xcode běží na Mac OS a není jiné cesty jak vyvíjet v tomto programu. Mezi další bonusy Unity patří celková podpora všech různých modulů potřebných k vytvoření aplikace. Přes debugger, kompilátor, analyzátor až po všelijaká API na všechny platformy. Touto univerzálností disponuje také Unreal engine a CryEngine ostatní nástroje nejsou tak komplexní. U většiny je k dispozici možnost doinstalování pluginů s knihovnami podporujícími dané funkce, ale už to, lze brát jako jistou nevýhodu.

Vybrané programy jsou rozděleny podle použitého jazyka. Z předchozího popisu programů pro tvorbu aplikací vyplývá, že nejčastějším jazykem pro psaní mobilních aplikací je v současnosti C#. Pokud se někdo rozhodne vytvářet aplikace, vhodný program by se měl odvíjet podle jazyka kódování a platformy, na kterou se chystá cílit. Právě tyto dva aspekty tvoří ideální pomůcku k výběru vhodného softwaru pro začínající vývojáře. Na základě potřeby programovat aplikace pro zvolené zařízení, je důležité vybrat program umožňující maximální využití vlastností pro zajištění kvalitní podpory funkcí pro vyvíjenou

aplikaci a cílové zařízení. Podle těchto kritérií je možné rozdělit programy zaměřené na tvorbu mobilních aplikací a na odlišné platformy. Nejvhodnějším vývojovým prostředím pro začínající vývojáře je GameMaker, který vysvětlí základní principy tvorby a umožní vyzkoušet tyto funkce na jednoduchých hrách vytvořených prostřednictvím Drag and Drop. Pokud má někdo zájem tvořit pouze na Apple zařízení je vhodné naučit se s Xcode. V momentě získaní základních znalostí a potřeby pro více platforem lze přejít na Xamarin či MonoGame. V případě vývoje obsáhlých a graficky propracovaných her pro PC, PS4 a XBOX je na místě sáhnout po Unreal enginu, Source engine, CryEnginu nebo Unity 3D enginu. V současnosti je nepopiratelně velmi výhodné využít Unreal enginu, jelikož mít k dispozici takhle výkonový nástroj a k tomu zadarmo, to je nabídka, jež se neodmítá. Zajisté z pohledu vývoje 2D her se složité a funkcemi přeplněné enginy příliš nehodí na jejich vytváření. Naopak v tento moment je ideálním nástrojem Monogame, GameMaker, Xamarin či Xcode.

Platformy	Xcode	Mono Game	Game Maker	Xamarin	Unreal Engine	Cry Engine	Unity 3DEngine	Source Engine
Windows	✗	✓	✓	✓	✓	✓	✓	✓
Mac OS	✓	✓	✓	✓	✓	✗	✓	✓
Linux	✗	✓	✓	✗	✓	✓	✓	✓
PlayStation 4	✗	✗	✓	✗	✓	✓	✓	✗
Xbox one	✗	✓	✓	✓	✓	✓	✓	✓
Nintendo Wii	✗	✗	✗	✗	✗	✓	✓	✗
iOS	✓	✓	✓	✓	✓	✓	✓	✗
Android	✗	✓	✓	✓	✓	✓	✓	✓
Windows phone	✗	✓	✓	✓	✗	✗	✓	✗
Web	✗	✗	✓	✗	✓	✗	✓	✗
Cena	Zdarma**	Zdarma	Zdarma*	Zdarma*	Zdarma***	9,90 € /měsíc	Zdarma*	Zdarma****

Tabulka 1 Porovnání podpory platforem Zdroj: Vlastní.

Vysvětlivky k tabulce: ** Základní verze bez pokročilejších funkcí*

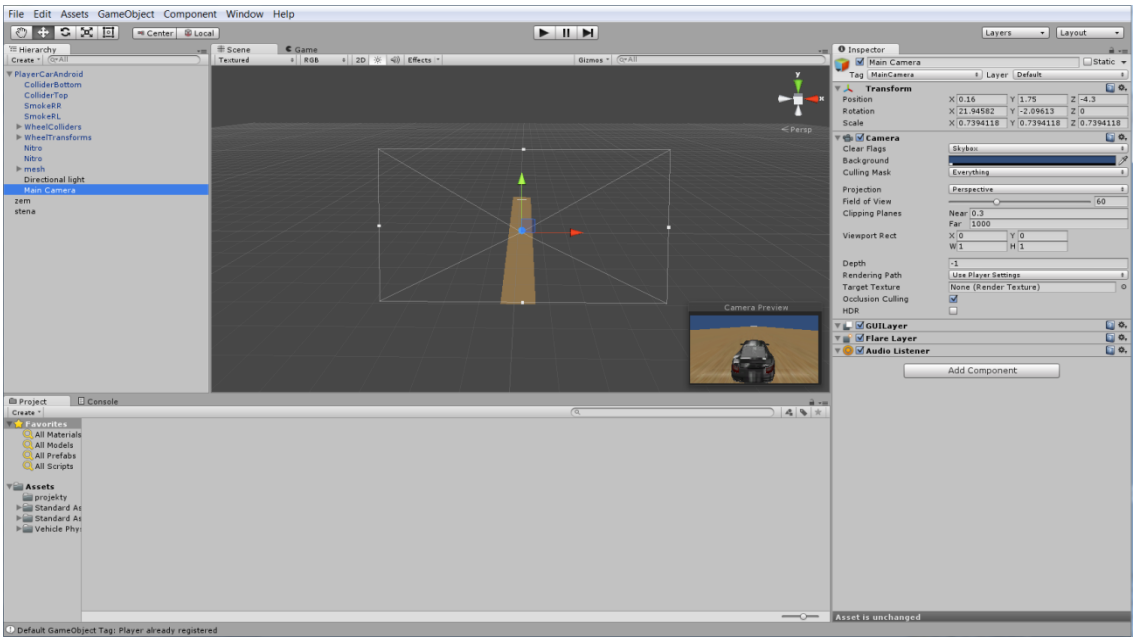
*** Zdarma, ale pouze na MacOS*

**** Zdarma, přičemž 5% ze zisku se odvádí společnosti Epic Games*

***** Zdarma pouze pro majitele her v Source enginu*

3.5. Základní vlastnosti Unity

Unity vyniká ve výborné podpoře komunitou uživatelů, která vznikala od prvního uvedení na trh v roce 2005. První krokem k vytvoření jakékoli aplikace je nainstalování softwaru a následně naučením se jeho základních ovládacích prvků. Důležité je umět se orientovat v grafickém rozhraní editoru. K tomuto účelu je dobré ukázat, jak vlastně takový editor vypadá viz. obrázek níže.



Obrázek 1 Unity editor Zdroj: vlastní

Na uvedeném obrázku je k vidění základní rozdělení editoru do podoken. Na levé straně je okno Hierarchie znázorňující pomocí stromové struktury objekty vložené na scénu. Spodní okno představuje strukturu složek s objekty, skripty, texturami, materiály. Vpravo se nalézá Inspector sloužící k zobrazování informací o vybraném objektu. Prostředním a hlavním oknem je Scéna, kde se sestavuje grafická podoba vytvářené aplikace. (23)

3.5.1. Unity dokumentace

Unity poskytuje pro vývojáře dokumentaci, kde je možné najít popis většiny funkcí. Nejenom funkcí, ale také všech vlastností a prvků Unity. K tomu ještě nabízí tzv. Scripting API, ve kterém lze nalézt vysvětlení všech tříd a s nimi související metody a vlastnosti.

Součásti jsou také ukázky možného použití kódu a syntaxe. Během vývoje aplikace bylo hojně využíváno této dokumentace, která je přehledně zpracována a poskytuje velmi dobrou nápovědu. Taková podpora není ve světě vytváření aplikací ojedinělá, podobné dokumentace nabízí také většina zmíněných konkurenčních prostředí. (24)

3.5.2. Herní objekty

Všeobecná lidská představa o objektu je většinou věc nebo nemovitost např. postava, dům, auto, strom, pistole atd., nicméně objekt ve hře je více abstraktní koncept. Základní objekt je cokoli bez jakéhokoli chování či funkce. Objekt získává vlastnosti k určitému chování, tak že se k němu přidá funkce. Unity v základu obsahuje komponenty, které je možné k objektu přiřadit. Komponenty podle Unity názvosloví přidávají objektu pravý význam a definují ho. Vlastnosti přidané objektu, dávají objektu život tím, že umožňují určit pozici, rotaci a měřítko.

Komponenty jsou prvky, jež upravují základní chování objektu a přidávají funkce vylepšující chování, tak aby objekt sloužil k tomu, pro co je určen. Těmito funkcemi jsou např. fyzické vlastnosti (gravitace), audio, světlo, podmínky renderování. Pro usnadnění práce je vhodné tyto komponenty využívat a pouze si je upravovat k docílení kýženého efektu. Komponenty jsou v podstatě předvytvořené skripty, proto přidání vlastního skriptu např. pro ovládání objektu, probíhá stejným postupem tj. přidáním skriptu k objektu. Tyto vlastnosti je možné vidět v okně Inspectora, kde jsou všechny přehledně vypsány s možností úpravy základních vlastností komponenty pomocí nadefinovaných proměnných. Většinou lze upravovat intenzitu, rychlost, možnosti renderování (tvorba reálného obrazu podle počítačového modelu) a další vlastnosti. (25)

Objekty jsou vykreslovány na obrazovku pomocí skladby základních komponent. Každý dům na scéně je sestaven z meshe, textury a materiálu.

- **Mesh**

  Tento prvek objektu lze připodobnit k síti ohraničující tvar objektu. Mesh dává objektu tvar, na který je možné aplikovat další vrstvy přidávající objektu specifičnost a poznatelnost. Jedná se o technický termín, proto bude dále používán anglický název mesh, místo českého (síť).

- **Textury**

  Pomocí textur je dán objektu vizuální vzhled. Textura jednoduše řečeno obalí mesh vrstvu barevnou fólií. Dle kvality existuje velké množství textur, Unity dokonce vlastní typ textury vhodné pro mobilní zařízení, právě tento typ je používán na vyvíjenou hru. Textury obsahují tzv. shadery, které udávají, jakým způsobem se daný objekt vykreslí na displeji zařízení a jak bude odrážet světlo. Pro vyvíjenou aplikaci jsou u všech modelů použity Mobile/Diffuse shadery.

- **Materiál**

  V Unity je nutné prvně vytvořit materiál a až poté přidat texturu. Pokud není dodrženo toto pravidlo, objekt bez textury je mnohem náročnější na výpočetní výkon. Materiál sám o sobě dodává objektu povrch, tak aby každý objekt nebyl pouze hladká kostka, ale byl hrubý např. jako kámen, či látka. Z hlediska reálnosti se jedná o zásadní prvek kvality, jelikož detail přidaný materiálem ovlivňuje věrohodnost celého objektu. (25)

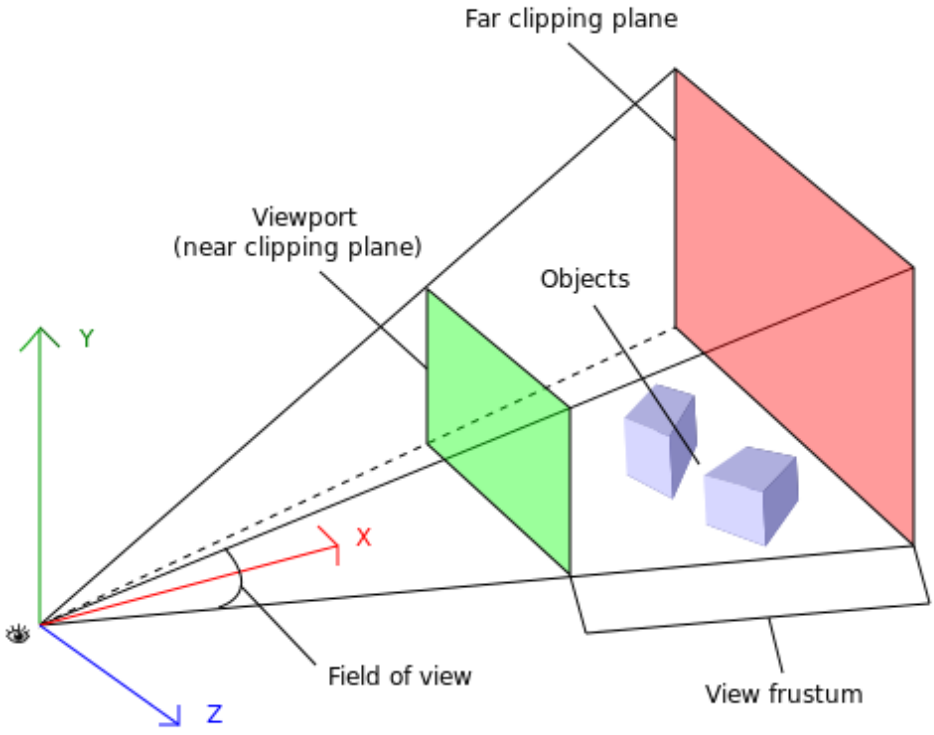
3.5.3. Prefaby

Pod pojmem prefab se skrývá kolekce skriptů, animací, mřížek, textur atd., které lze použít na scéně. Hlavní výhodou těchto prefabů je možnost znovupoužití bez nutnosti opakovat proces výroby objektu s přidanými vlastnostmi. Jednoduše řečeno, prefab je kompletní objekt se všemi komponenty a vlastnostmi. Pomocí jednoduchého přetahování na scénu se prefaby přidávají do hry. Jejich využití se ocení nejvíce, pokud se přidávají na scénu stejné objekty s určitými vlastnostmi a funkcemi. Takový prefab může představovat objekt s přiřazeným skriptem či s nastaveným tlačítkem. Po přetažení na scénu vznikají instance prefabu se všemi vlastnostmi původního objektu. Nemusí se nutně jednat jenom o jeden objekt, nýbrž lze udělat prefab celé scény včetně všeho co obsahuje.

Ve vyvíjené hře jsou zastoupeny modely či prefaby většinou jako domy, silnice, světla a další předměty objevující se ve fiktivním světě. (24)

3.5.4. Kamera

Pomocí kamery lze zobrazovat obsah scény z pohledu uživatele. Kamera umožňuje snímat vše, co je zobrazeno v tzv. frustum neboli vše v rámci komolého jehlanu, určujícího, které objekty jsou právě viděny. Toto frustum je poté vykresleno jako 2D obraz pro uživatele s perspektivou 3D prostoru. V obecném mínění frustum definuje geometrický tvar, který je určen pro zobrazení zóny viditelných objektů, ostatní objekty mimo dosah této zóny jsou neviditelné. Zmíněné vlastnosti kamery je opět možné uzpůsobit požadavkům v okně inspektora pomocí nadefinovaných proměnných např. pole zobrazení, blízká rovina, vzdálená rovina.



Obrázek 2 Frustum. Zdroj: http://answers.unity3d.com/questions/589485/understanding-3d-world-and-gui-camera-space.html

V Unity existují dva typy projekce kamer ortografická a perspektivní projekce. Tyto projekce slouží k určení, jak je svět viděný kamerou zobrazován. První typ projekce je

definován následovně: *„Kamera s ortografickou projekcí odstraňuje perspektivní úpravu z pohledu kamery a zajišťuje, že nezáleží jak daleko, je objekt od kamery, jejich vzdálenost zůstává stejná.“* (25) Tato kamera bude použita při tvorbě 2D her, což je zřejmé z její definice. Druhý typ projekce tzv. perspektivní kamera je určena pro zobrazení 3D světa. Objekty vzdálenější od kamery jsou zobrazeny jako menší, než ve skutečnosti jsou.

V rámci jednoho projektu lze použít oba typy projekce. Pro klasické zobrazení hry z pohledu první osoby je vhodná perspektivní kamera, kdežto pro vykreslení mapy světa se spíše hodí ortografická kamery. Unity umožňuje kombinaci těchto projekcí v rámci jednoho projektu. (25)

3.5.5. Světla

Důležitou součástí každé scény jsou světla. Scéna bez světel vykresluje objekty tmavé a bez života. Pomocí světel je možné přidat atmosféru scéně a hloubku světu využitím stínů. Existují tři základní typy světel: směrové světlo, bodové světlo, prostorové světlo.

- **Směrové světlo**

  Obecně použitelným světlem v Unity je směrové světlo. Na základě pozice světla pomocí směru je určeno, jak jsou objekty ve scéně osvíceny. Jedná se o nejméně náročné světlo tj., světlo s nejmenší náročností na výpočetní výkon.

- **Bodové světlo**

  Tento druh světla představuje zdroj světelných paprsků s pozicí na scéně, ale bez směru světla. Šíření paprsků světla je vyrovnané ve všech směrech. Z hlediska výpočetní náročnosti se řadí mezi průměrné a obecně je nejvíce využívané pro většinu efektů ve hře.

- **Prostorové světlo**

  Nejnáročnějším typem světla je prostorové světlo, jež simuluje stejný efekt jako světlo z baterky. Toto kuželovité světlo má vlastnost spádu, která vykresluje střed světla světlejší barvou, než okraje. Bohužel se nejedná o obecně použitelné světlo, jelikož vysoká náročnost zamezuje častému výskytu na scéně. Použití na mobilních

platformách je velmi nevhodné, opět z důvodu vysoké náročnosti na výpočetní výkon.

Pokud je potřeba osvětlit scénu za podmínek nízké náročnosti na výpočetní výkon, lze využít tzv. světelných map. Definice těchto světel zní: „Světelné mapy slouží k osvětlení scény a zapečení těchto efektů světla přímo do scény.“ Je to jednoduchý způsob jak snížit náročnost aplikace a neomezit se snížením kvality grafické stránky. Unity nabízí podporu pro vytváření těchto světelných map skrze Beast light mapping system. (25)

3.5.6. Zvuky

Nedílnou součástí každé hry jsou zvukové efekty. Díky zvukům v pozadí hry lze jednoduše navodit atmosféru prostředí. Hlasité zvukové efekty během střelby na protivníky zvyšují emoce hráče a dávají větší pocit reálného světa. Unity obsahuje spousty zvukových schopností pro pokrytí celého spektra.

- **Zvukový posluchač**

  Tento Unity komponent přidává posluchače zvuků objektům, ke kterým je přiřazen. Zvukový posluchač funguje jako mikrofon, vše co slyší, slyší i hráč skrze reproduktory. Jako v reálném světě i v Unity má tento komponent vlastnost hrát zvuky pouze pokud je hráč dostatečně blízko objektu. V momentě, kdy se hráč vzdálí, nic neslyší. Většinou se používá na postavu, za kterou hráč hraje hru.

- **Zdroj zvuku**

  Jedná se o komponent šířící zvuky daného objektu. Aplikuje se jednoduchým přidáním na objekt, který je slyšet pokud se k němu hráč přiblíží.

- **Zvukový klip**

  Po vytvoření zdroje zvuku je nutné mu přiřadit zvukový klip tj. zvukovou stopu v různých formátech. Unity nabízí podporu celé škály známých formátů, jako jsou: mp3, wav, aif, Olg, xm, mod, it a s3m. (25)

3.5.7. Skripty

Skripty slouží k vytváření nového chování, které se přiřadí k objektu. Unity využije tento skript, když bude komunikovat s objektem během fáze vykreslování. V unity existují předdefinované skripty pro provozování základních operací např. ovládání pro první osobu, řízení auta, ovládání pro mobilní zařízení. Bohužel tyto skripty jsou obecně použitelné, ale pro specifickou aplikaci je vhodnější si naprogramovat vlastní.

Pro skriptování obsahuje Unity dva editory Unitron a Monodevelop. V minulosti byl hlavním editorem Unitron, jež byl přímo dodáván s Unity. Unitron nenabízí tak kvalitní IDE jako jsou Visual Studio, ale pro základní skripty je zcela dostačující. V současnosti je implicitně nastaven editor MonoDevelop. Tvůrcem tohoto editoru je komunita Mono, která v současnosti běží pod záštitou Xamarinu. Monodevelop je open source project (volně ke stažení) IDE na úrovni Microsoft Visual Studio. MonoDevelop integruje klasické funkce IDE např.: automatické dokončování kódu, zdrojová kontrola, GUI a webovou grafiku. (25) (26)

3.5.8. Specifické skriptování pro iOS

Unity obsahuje množství základních tříd (kolekce předdefinovaných funkcí) určených přímo pro iOS. Pomocí těchto tříd je psaní skriptů zjednodušeno a lze naplno využít vlastností mobilního zařízení. Zároveň se jedná o velký přínos pro vývojáře, jelikož volané metody (funkce v rámci třídy) není potřeba vytvářet, nýbrž je stačí zavolat. Přístup k vlastnostem telefonu probíhá pomocí zmíněných tříd. Informace o těchto třídách jsou k dispozici na internetových stránkách Unity dokumentace, kde jsou všechny charakterizovány pro usnadnění použití.

3.6. C Sharp

Microsoft Visual C# je výkonným programovacím jazykem, jež přebírá množství toho nejlepšího z jazyků C++ a Microsoft Visual Basic. Výsledkem je čistší a logičtější jazyk určený nejvíce pro vývojáře na platformě .NET Framework. Zejména díky odstranění nesrovnalostí a anachronismů, mohl vzniknout tento jazyk vhodný pro vývoj v 21 století.

Jelikož se jedná o jazyk vyššího typu, je blíže lidskému myšlení a vývoj v něm pomáhá vytvářet lepší logiku celého kódu k psaní moderních aplikací. První premiéra jazyka C# byla v roce 2001 a jednalo se o verzi 1.0. V roce 2005 vyšla verze 2.0, která zavedla nové prvky jako generické typy, iterátory a anonymní metody. Další verze oficiálně vyšla v roce 2008 a přidala rozšiřující metody, lambda výrazy a technologii LINQ (Language Integrated Query).

Technologie LINQ představuje most mezi objekty a daty. Pomocí této technologie je umožněno psát příkazy na různé data použitím jedné syntaxe. Do příchodu LINQ, bylo nezbytně nutné znát dotazové jazyky pro každý typ zdrojových dat, ať už se jedná o SQL databáze, XML dokumenty či různé webové služby.

Ve verzi 4.0 byla vylepšena zejména provázanost ostatních jazyků a technologií. Současně s verzí 4.0 přišla verze .NET Framework 4.0, která jako hlavní přínos, přidává podporu TPL (Task Parallel Library) neboli knihovny, jež dokážou vytvářet vysoce škálovatelné aplikace za využití nejnovějších více jádrových procesorů. Poslední verze jazyka C# s označením 5.0 byla uvedena v roce 2012 a přinesla vylepšení v podpoře asynchronního programování přidáním klíčových slov „async“ a „await“. V unity je možné využít async pro zavolání jiné operace během jedné běžící. (27) (28) (29)

3.6.1. Základní vlastnosti jazyka C#

V této kapitole jsou charakterizovány vlastnosti jazyka C#. V předchozí části bylo zmíněno, že jazyk C# je objektovým jazykem, proto se skládá ze tříd, metod a atributů tříd. Syntaxe jazyka je ovlivněna tímto uspořádáním, protože se vždy musí dodržet konstrukce kódu na základě objektových vlastností. Jakožto ve většině odborné literatury na téma programovacích jazyků, tak i v knize Microsoft Visual C# 2010 je prvním programem „Hello, World!“. Dobře známý, jako první program na světě, který navrátil řetězec „Hello, World!“. Čistě z tradice je proto uveden tento program jako první.

```
public class Hello
{
	static void Main ()
	{
	Console.WriteLine("Hello, World!");
	}
}
```

Na ukázce kódu výše je možné demonstrovat syntaxi jazyk C#. První řádka kódu obsahuje definici veřejné třídy tj. třídy, do které lze přistupovat, přestože není přímo obsažena v daném souboru. Poté je v kódu definována statická metoda Main, která je specifická svým použitím. Jedná se o vstupní bod programu. To, že je definována jako statická, umožňuje volat tuto metodu, aniž by bylo nutné vytvářet instance této třídy. Z toho vyplývá, že statická třída nepotřebuje ukládat parametry, nýbrž pouze provede určenou metodu a navrátí výsledek. V opačném případě nestatické třídy musí vždy operovat nad zadanými parametry.

Posledním záznamem v kódu je „Console.WriteLine(„Hello, World!“);“, která slouží k vypsání řetězce napsaného v uzavřené závorce a ohraničeného uvozovkami. Samotný příkaz se skládá z vestavěné třídy Console. Za tečkou jsou k dispozici všechny metody, vlastnosti a datové pole této třídy. Celý zápis tedy volá metodu WriteLine ze třídy Console s přidanou vlastností tj. řetězcem „Hello, World!“. (27)

3.6.2. Obor názvů

Mezi další vlastnosti jazyka C# patří obory názvů. V podstatě to jsou kontejnery pro identifikátory, jako jsou např. třídy. V této práci jsou nejvíce používány obory názvů:

- UnityEngine
- System.Collections
- UnityEngine.UI

První zmíněný obor názvů odkazuje na všechny třídy spojené s Unity. Druhý obsahuje třídy vztahující se k .NET tj. obecně používané třídy např. list, queues, bit arrays, hash tables a dictionaries. (30) Poslední ze zmíněných oborů názvů definuje třídy zabývající se

uživatelským prostředím. Jakmile se v Unity vytváří menu pomocí implicitních tříd, je nezbytné deklarovat UnityEngine.UI, aby MonoDevelop věděl, kde tyto třídy hledat. (27)

3.6.3. Identifikátor

Identifikátor slouží pro rozpoznávání částí programu, podle použitých názvů. Mezi identifikátory lze zařadit obory názvů, třídy, metody a proměnné. Jazyk C# obsahuje základní pravidla pro používání identifikátorů. Pro pojmenování je možné použít pouze malá/velká písmena, číslice a podtržítko, přičemž identifikátor musí vždy začínat písmenem nebo podtržítkem. Celkově je v jazyce C# k dispozici pro svou vlastní potřebu 77 identifikátorů. Těmto identifikátorům se také říká klíčová slova a nelze je použít pro pojmenování proměnných. Každé klíčové slovo má svůj přesný význam. V MonoDevelop jsou obarveny do modra, aby byly jasně poznatelné. Identifikátory v jazyce C# jsou např. if, do, class, int, true, void atd. (27)

3.6.4. Proměnné

Společným základem většiny programovacích jazyků jsou proměnné. Obecně lze proměnné považovat za místo přechovávající nějakou hodnotu. Přesněji to je část paměti určená k dočasné úschově informací. Každá proměnná musí mít jedinečný název, který ji definuje v prostředí, v němž se používá. Pomocí názvu se odkazuje na informaci uloženou v paměti.

Jazyk C# má několik vestavěných typů proměnných, které jsou označovány jako primitivní datové typy. Jsou to všeobecně známé proměnné jako: int, long, float, double, decimal, string, char a bool. (27)

3.6.5. Třídy

Třída je základním pojmem klasifikace, slouží k systematickému uspořádání informací a chování do logické entity. Nejlepším způsobem vysvětlení třídy je ukázka použití v praxi. Třída automobil obsahuje všechny společné rysy chování pro všechny vozy např.: řízení, zastavení, zatáčení atd. a společné vlastnosti jako volant, kola, motor atd.. Bez klasifikaci by nemohl existovat reálný svět. Lze tvrdit, že dělení do skupin je lidská

vlastnost dávající uspořádanost, bez níž by nedávala debata o automobilu smysl, jelikož by si každý představil jiný objekt s odlišnými vlastnostmi a chováním.

3.6.6. Metody

Název metoda je synonymem k funkci či proceduře v jiných jazycích jako je C nebo Microsoft Visual Basic. Základní vlastností metody je unikátní název a tělo. Název, by měl odpovídat smyslu celé metody, tzn. přesně popisovat funkci např. „sečtiDvaInt“. Samotné tělo metody poté obsahuje příkazy, jež budou provedeny po zavolání kromě toho, v těle lze definovat proměnné, se kterými se posléze bude pracovat nebo je napsat jako parametr do závorky za název metody. Pokud má metoda vracet hodnotu, musí být zadán typ návratové hodnoty a k tomu v těle umístěn příkaz return. V opačném případě, je nutné použít klíčové slovo void, jež signalizuje, že metoda nevrací žádnou hodnotu. Metody představují základní stavební kámen jazyka C#.

„Metody existují, proto aby byly volány.“ (27) Samotné zavolání metody probíhá vypsáním jejího jména a přidáním parametru pokud byl zadán. Jestliže nebyl přiřazen žádný parametr, je stejně potřebné napsat kulaté závorky za jménem metody. (27)

4. Praktická část

V teoretické části byl čtenář seznámen s Unity 3D enginem a jeho konkurenty, s popisem základních vlastností Unity a také s jazykem C#. Nyní se dozví více o praktickém fungování tohoto enginu. Tato část diplomové práce se zabývá tvorbou aplikace pro iOS zařízení, konkrétně pro iPhone 4. V jednotlivých kapitolách jsou rozepsány kroky vedoucí k finální podobě vyvíjené hry (Garbage Ride). Na začátku práce byla jednoduše popsána hra a její herní prvky. V nadcházejících kapitolách jsou detailně uvedeny všechny procesy spojené s vývojem.

4.1. Analýzy předcházející vývoji

Samotnému vývoji aplikace vždy předchází analýza a navržení hlavních grafických a funkčních částí. Z tohoto důvodu jsou zde uvedeny základní požadavky na aplikaci, charakteristika cílového zařízení, grafický wireframe a vývojový diagram systému hry.

4.1.1. Požadavky na aplikaci

Vývoj aplikace není jednoduchá ani krátkodobá záležitost, proto je důležité si vše rozmyslet a naplánovat. V této sekci jsou obsaženy požadavky, které by měla vyvíjená aplikace splňovat. Přestože se nejedná o aplikaci vytvářenou na míru, je vhodné si předem zvolit některé ze základních vlastností.

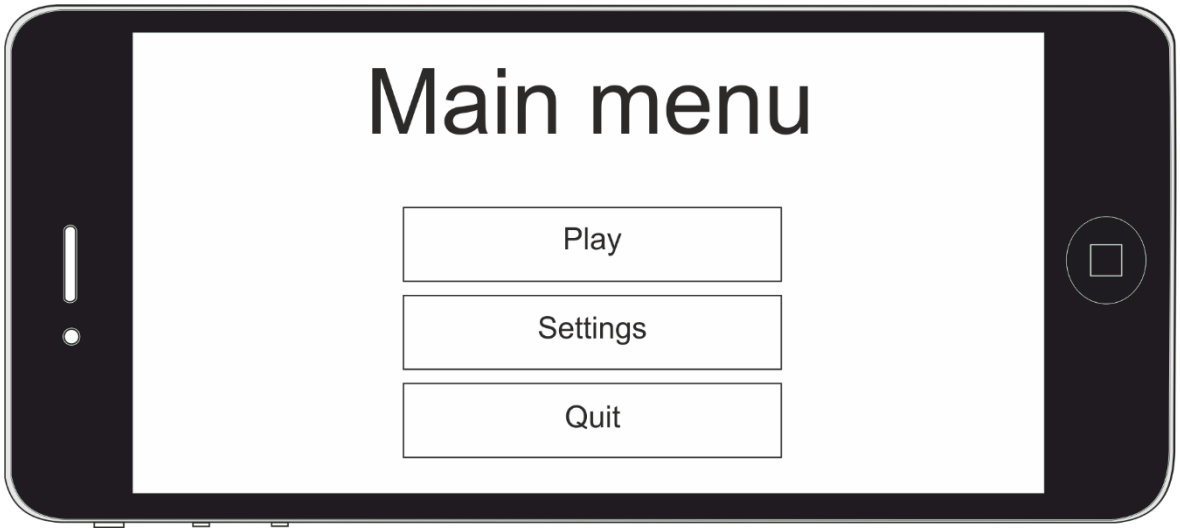
- Výkonové požadavky
  - Hratelnost na iPhone 4
- Funkční požadavky
  - Jízda dopředu/dozadu pomocí tlačítek na displeji
  - Zatáčení náklonem zařízení
  - Započítávání bodů za naložení kontejneru/popelnice
  - Časový limit
  - Mini-mapa
  - Systém menu
- Logické požadavky

- Bodová hranice pro postup do další úrovně
- Sčítání bodů ze všech úrovní
- Ukončení po doběhnutí stanoveného času
- Ukončení při převrácení vozu

4.1.2. WireFrame

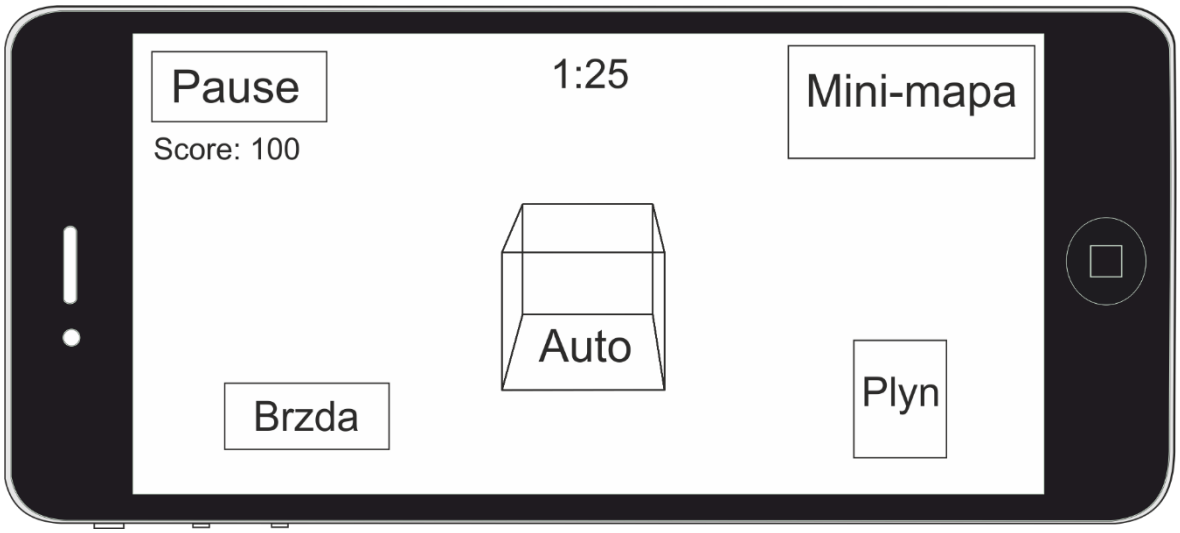
Tuto kapitolu lze rozdělit do menších částí na základě zobrazovaných informací na displeji zařízení. Tím je míněno, že níže jsou představeny návrhy na úvodní obrazovku, herní GUI, hlavní menu a ukončovací obrazovku.

- Úvodní obrazovka se skládá ze základních tlačítek tvořících tzv. Main menu. V aplikaci se neobjevují obsáhlá menu na nastavení funkcí, nýbrž pouze jednoduchá spouštěcí tlačítka pro vstup do hry, ukončení a nastavení zvuku. Pozadí se bude skládat z vytvořeného obrázku ze hry.



- Menu nastavení a Pause menu bude vypadat obdobně jako hlavní menu. Opět bude složené pouze z názvu aktuální obrazovky a tlačítek na ovládání hry.

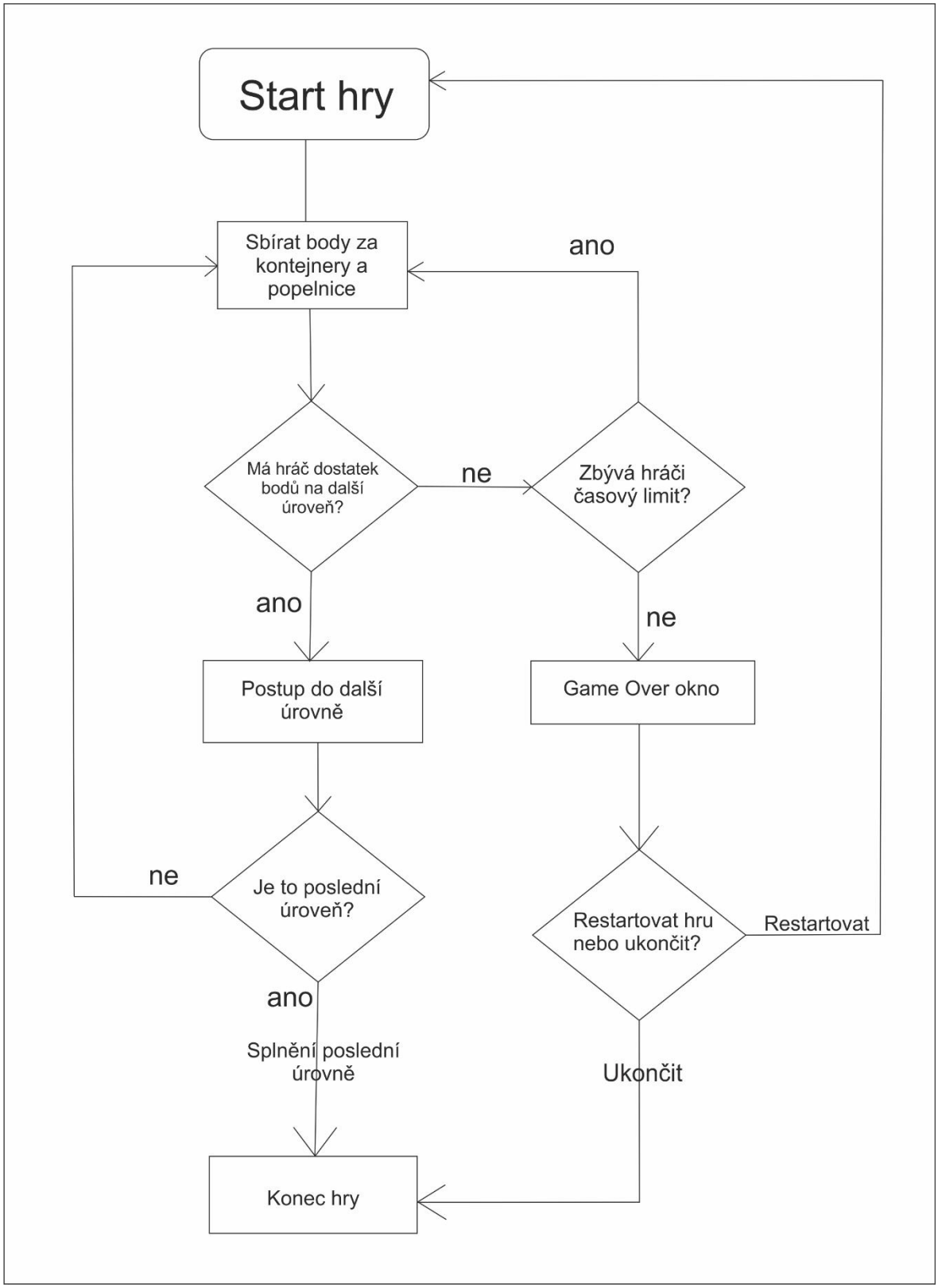
- Herní interface je základem celé aplikace, proto je důležité si důkladně rozvrhnout rozložení všech tlačítek a funkcí zobrazených na displeji zařízení. Spuštěná aplikace by měla mít následující rozvržení:



Na grafickém návrhu výše, jsou zobrazeny všechny prvky, jenž vidí hráč na displeji zařízení. Tři z nich jsou tlačítka (Pause, Brzda, Plyn) vykonávající funkci na základě toho, zda je hráč stiskl či ne. Ostatní položky jsou informativního původu a slouží k doplnění logiky hry. Časomíra umístěna nad vozidlem určuje limit, do kterého musí hráč posbírat dostatek bodů. Skóre zaznamenává nasbírané body. Mini-mapa slouží k navádění ke kontejnerům nebo popelnicím. Na displeji jsou pouze dvě tlačítka na ovládání, jelikož zatáčení je prováděno náklonem zařízení.

4.1.3. Vývojový diagram

Za účelem znázornění systému hry je nakreslen vývojový diagram vysvětlující, jaké možnosti jsou ve hře k dispozici. Logika hry čerpá z klasických bodových systémů her. Pokud hráč dosáhne určité bodové hranice ve stanoveném časovém limitu, postupuje do další úrovně. Tento systém je pomocí diagramu znázorněn níže:


Start hry
Sbírat body za kontejnery a popelnice
ano
Má hráč dostatek bodů na další úroveň?
ne
Zbývá hráči časový limit?
ano
ne
Postup do další úrovně
Game Over okno
ne
Je to poslední úroveň?
Restartovat hru nebo ukončit?
Restartovat
ano
Splnění poslední úrovně
Ukončit
Konec hry

4.1.4. iPhone 4

Hra je navržena pro mobilní telefon iPhone 4, proto je dobré zmínit technické vlastnosti tohoto zařízení. Již v úvodu byla řečena krátká historie iPhonů, následující kapitola se více zabývá funkčním popisem. Hlavní vlastnosti iPhone 4 jsou: dotyková kapacitní obrazovka, uzavřený systém iOS, gyroskop, akcelerometr a relativně obstojný výpočetní výkon vzhledem ke stáří zařízení. Vzhledem k tomu, že se jedná o jediné zařízení, které má autor k dispozici, vše se odehrává právě na něm a o něm.

Unity má podporu pro platformu iOS, bohužel dostat aplikaci do telefonu, aby bylo možné ji testovat, není zcela jednoduché a je k tomu zapotřebí vlastnit počítač od společnosti Apple např. MacBook. Cesta aplikace z Unity do telefonu začíná exportem aplikace na zvolenou platformu. Poté je nezbytné stáhnout Xcode do Macbooku a importovat do něj aplikaci. Aby byl umožněn export do zařízení, musí mít uživatel vytvořený účet na Apple Developers za poplatek 99 dolarů. Následně po schválení účtu má uživatel status developer a možnost zaregistrovat své zařízení, na které poté může nahrát a testovat vytvořenou aplikaci.

Mezi výhody spojení Unity a iOS patří podpora technologie IL2CPP. Jedná se skok dopředu, jelikož IL2CPP představuje Compiler a Virtual Machine, který přenáší větší náročnost z GPU (Graphic Processor Unit) na CPU (Computer Processor Unit). Díky této funkci je možné spouštět náročnější aplikace, aniž by to poznamenalo výkon zařízení. Tato funkce je současně dostupná pouze pro platformu iOS, ale Unity tvrdí, že pracují na podpoře dalších platforem.

Jelikož autor práce neměl k dispozici MacBook, celé testování probíhá pomocí nástroje Unity Remote 4, což je aplikace pro iPhone volně ke stažení. Jakmile je mobilní zařízení připojeno k počítači a je spuštěna vyvíjená aplikace, postačí zapnout Unity a dát volbu „play“. Unity rozpozná zapnutou aplikaci na telefonu a umožní testování všech dostupných funkcí. Tímto způsobem je možné volně používat aplikaci na zařízení, přičemž současně probíhá zobrazení i na displeji počítače.

- **Displej**

iPhone 4 má tzv. kapacitní systém využívající kapacitní materiály, které drží elektrický náboj. Pokud se displeje dotkne prst, změní se nabití vrstvy a zařízení dokáže poznat,

na jakém místě prst spočinul a tuto informaci odeslat procesoru. Jistá nevýhoda těchto displejů spočívá v tom, že nelze vyhodnotit dotek, pokud není statické elektřiny, která vyvolá reakci kapacitních materiálů. Což znamená, že telefon nelze používat v rukavicích nebo přes jiné nevodivé materiály. Jiné způsoby záznamu dotyků na displeji jsou rezistivní a infračervený ty však iPhone 4 nevyužívá.

- **Akcelerometr**

Akcelerometr je součástka měřící zrychlení, které je převedeno na elektrické napětí. Akcelerometry se dělí na dva typy dynamický a statický. Přičemž první vyhodnocuje rychlý pohyb a druhý reaguje na zemskou přitažlivost. Současně s gyroskopem podávají informace o poloze a směru v prostoru. Kombinace použití obou technologií je z důvodu přesnosti, protože akcelerometr je náchylný na odchylky v magnetickém poli a zároveň je občas zmaten dynamickým pohybem. Akcelerometr s gyroskopem v iPhonu 4 dohromady dávají nejpřesnější informaci o současném stavu přístroje v prostoru. (31)

4.2. Vytváření světa

Při startu Unity je prvním krokem vytvoření scény, na kterou se postupně přidávají další objekty. Na začátku je scéna prázdná, pouze rozdělená mřížkou na rovinu v bodě nula, pro zajištění lepší orientace.

Popelářská hra se bude odehrávat ve fiktivním světě, vytvořeném ze staženého assetu Cartoon city. Jelikož mobilní zařízení nenabízí výkon jako stolní počítač, je důležité hlídat si náročnost vytvářené hry na výkon. Hlavními prvky, které mohou výrazně ovlivnit výkonnost hry, jsou počty polygonů (mnohoúhelníků, ze kterých je složen mesh objektu). Právě jejich počet je rozhodujícím faktorem pro výpočet náročnosti zobrazení objektů ve hře. Kvůli snaze tento problém, co nejvíce eliminovat, jsou použity jednoduché modely budov s minimálním počtem polygonů. Jedna budova má přibližně kolem 12 polygonů, což se pozitivně odráží na spotřebě systémových zdrojů zařízení a zároveň umožňuje navrhnout obsáhlejší svět s více modely.

4.2.1. Terén

Unity obsahuje sadu nástrojů pro vytváření terénu nazvanou příznačně terrain. Tato sada obsahuje základní pomůcky pro vyrovnávání, vyvyšování, snižování, zabarvení, vyhlazování. Dále obsahuje obecné nástroje pro úpravu textury, jakmile jednou změníte texturu, je to kompletní krok k obarvení celého terénu, následné změny pouze segmentu se provádějí pomocí paint texture, kde jsou k dispozici barvy jako štěrk, hlína, písek. Samozřejmostí je možnost přidání více textur dle libosti vývojáře. Nedílnou součásti sady terrain jsou možnosti pro vkládání stromů a rostlin na scénu. Velká výhoda tohoto assetu je, že jakmile je potřeba vložit více stromů na větší prostor, je možné iniciovat terrain s modely stromů a pak hromadně vkládat velké počty stromu na vybrané území. Zároveň lze provést hromadné vkládání na základě podkladové mapy. Pokud je k dispozici mapa s kontrastním oddělením krajin se stromy a bez, lze vytvořit skript, který se postará o takové vložení a tím neuvěřitelně usnadní práci v případě velmi rozsáhlých světů. Okolí vytvářeného světa je obehnáno horami, které zamezují jízdu mimo hranice města. Pomocí nástroje terrain, byl vyvýšen okolní terén, aby napodoboval hory a poté byla nanesena textura imitující pohoří.

4.2.2. Modely

Pomocí jednoduchého vkládání byla vytvořena síť silnic. Unity umožňuje duplikovat objekty na scéně, čímž se ulehčí práce při tvorbě okolího světa, jako jsou silnice, chodníky atd. Vše je možné označit a posléze duplikovat. V případě silnic se označí více částí, ty se duplikují a poté vyrovnají, tak aby na sebe pasovaly.

Situace u modelů budov se opakuje, jelikož se provádí stejným způsobem jako vkládání silnic. Vybraný model se přetáhne na scénu a umístí. Tímto způsobem lze vytvořit celá města. Čím více prvků je vloženo na scénu, tím více se fiktivní svět podobá reálnému. Zajisté záleží na druhu modelů, protože např. vyvíjená hra má záměrně nádech animovaných prvků, takže modely nekopírují přesně reálný svět, i přesto je možné pomocí modelů, jako jsou billboardy, stromy, elektrické vedení, chodníky atd. vytvořit pěkný model fiktivního světa.

4.2.3. Obloha

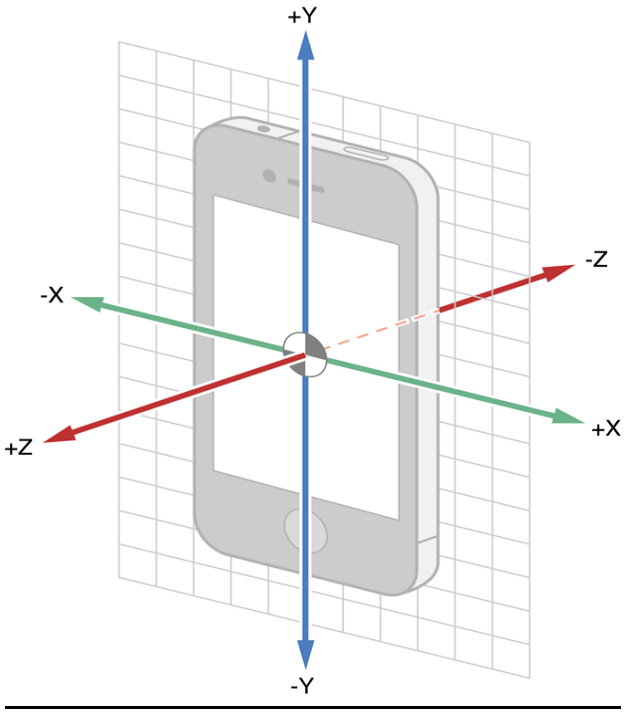
Součástí reálného světa je obloha s mraky. V unity je více možností jak docílit stejného efektu, zásadní je být rozhodnut do jaké míry dávat přednost grafice před výkonovou nenáročností. To je otázka, která by měla být položena při startu vytváření světa a podle toho by se měly odvíjet modely, obloha, grafické efekty atd. Ve hře Garbage ride je toto zabezpečeno implementovaným Skyboxem, který představuje krychli s texturou oblohy a je vložen přes celé město. Existují také skyboxy, jež imitují cyklus dne, tj. ráno vyjde slunce, večer měsíc. Ovšem tyto složitější napodobeniny oblohy jsou také náročnější na výkonové prostředky, se kterými je snaha šetřit. Zejména z důvodu staršího zařízení, pro které by se mohla propracovaná obloha stát "brzdou" z hlediska výkonových možností.

4.3. Herní skripty

Pomocí Skriptů se vytváří chování, které dohromady dává herní mechanismus. V Unity se nepíše jeden dlouhý zdrojový kód, nýbrž krátké skripty pro danou činnost. Skripty jsou vzájemně provázané, čímž se propojují i herní mechanismy a vzniká kompletní hra. V následujících kapitolách jsou jednotlivé skripty vysvětleny včetně vzájemného propojení.

4.3.1. Zatáčení pomocí náklonu

Zatáčení v připravované hře využívá ovládání pomocí gyroskopu, který poskytuje aktuální informace o náklonu mobilního zařízení. Hraní tímto způsobem jistě navozuje přesvědčivější atmosféru hry pro hráče. Zároveň řeší problém s intenzitou zatáčení, jelikož pouhý stisk neumí brát v potaz lehké zatočení, na rozdíl od této funkcionality.



Obrázek 3 Gyroskope iPhone 4 Zdroj: https://developer.apple.com/library/prerelease/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/motion_event_basics/motion_event_basics.html

Mobilní zařízení jsou užívána v 3D prostoru a proto je nutné zamezit reakci na náklon v jiných osách. Hra má pevně nastaveno omezení na otáčení displeje, protože je od počátku vytvářena na hraní v horizontální poloze. IPhone knihovna v Unity má nastaveno přetočení os, při převrácení zařízení o 90 stupňů. Proto je názorný obrázek rozvržení os pravdivý i při otočení a náklon je brán pouze z osy x. Hodnoty získané ze zařízení se pohybují v intervalu <-1; 1> a představují hodnoty G-přetížení. Skript na zatáčení pomocí náklonu vypadá následovně:

```
public void Steer()

{

turn = carTurn * Input.acceleration.x;

WheelColliderFL.steerAngle = turn;

WheelColliderFR.steerAngle = turn;

}
```

Vytvořená metoda pro zatáčení náklonem je díky C# jazyku a podpoře Unity relativně snadná. Do proměnné turn typu float se ukládá zvolená hodnota carTurn, která je veřejná

a lze ji skrze inspektora měnit podle požadavků. Posléze je proměnná turn uložena do hodnoty steerAngle, jež je vlastností WheelColliderů, čímž je předána hodnota vygenerována mobilním zařízením při naklonění, do WheelColliderů na předních kolech, které zajistí zatočení ve hře.

Kvůli zajištění nejenom fyzických vlastností vozu, ale také vizuálních je použita funkce transform.Rotate převádějící rychlost pohybu vozu do otáček kol za metr. Tímto způsobem je zajištěno otáčení kol při jízdě. Grafická podoba kol je vložena do objektů, které rotují kolem své osy na základě výpočtu rychlosti otáčení WheelColliderů.

```
WheelFR.transform.Rotate (WheelColliderFR.rpm * 60 / 360 *
Time.deltaTime * 20, 0, 0);
```

Stejný zápis je aplikován pro všechna kola a nachází se v update metodě v CarScriptu.

4.3.2. Dotykové tlačítko

Některé třídy, jsou přístupné pouze, pokud je na začátku skriptu deklarován tzv. Namespace, neboli sada identifikátorů, česky obor názvů. V případě hry Garbage ride je nutno deklarovat ve všech skriptech obsahující dotykové funkce následující obor názvů:

```
using Unityengine.UI;
```

V Unity je funkčnost tlačítka myši naprosto stejná jako jeden dotek prstu na displeji iPhonu. Celý mechanismus funguje pomocí metody Ray, která vyvolá paprsek na místo doteku a poté sleduje chování uživatele na základě čtyř různých stavů.

Při stisku displeje existují čtyři stavy:

- OnTouchDown

  Co se stane, když se prst dotkne displeje tj. okamžitá změna po doteku.

- OnTouchUp

  Změna po zvednutí prstu.

- OnTouchStay

  Stav, který se realizuje, když prst zůstane na obrazovce.

- OnTouchExit

  Konec doteku.

Vždy když displej rozezná dotek, projde všechny možnosti a určí, který ze stavů se právě děje. Tímto způsobem se provádí vstup do aplikace pro ovládání tlačítek plynu a brzdy.

Unity 4.6 má nový systém pro uživatelské rozhraní, který by měl usnadnit a zlepšit možnosti vytváření UI (User Interface). Podle dostupných informací je zřetelná snaha unity vývojářů o přiblížení tvorby aplikací širšímu okolí. Jelikož všechny čtyři stavy byly implementovány do grafického rozhraní, není již více potřeba vypisovat do kódu, co se má stát v každém ze stavů. V novém UI oproti starému není potřeba definovat činnost pro všechny možnosti doteku. Postačí si zvolit jednu z možností a poté na ní navázat metodu, která se provede po stisknutí. Ono navázání probíhá způsobem Drag and Drop do okna Inspectora k vytvořené události. Funkčnost tohoto systému je zajištěna pomocí EventSystemu, který obstarává Input příkazy, tj. provádí to, co v minulosti bylo nutné definovat pomocí čtyř stavů doteků. Důležité je nezapomenout přidat na začátek kódu zmíněný obor názvů UnityEngine.UI, díky němuž jsou přístupné dotykové funkce.

4.3.3. Kontejnery

Cílem hry je sesbírat určitý počet popelnic v časovém horizontu. Sběr popelnic probíhá způsobem zastavení ve vyznačené části v blízkosti popelnice a vyčkání po dobu animace vykládání. Pokud hráč nepočká, popelnice se nevysype a hráč pouze ztratil čas, protože musí opět čekat, než proběhne animace vykládání.

Animace naložení kontejneru je vytvořena pomocí nástroje Animation a upravena k použití nástrojem Animator. Celý proces animování se sestává z jednotlivých kroků, jež přiřadí části nebo celému objektu změnu souřadnic pohybem, otočením nebo změnou měřítka. Operace naložení je poněkud složitější, protože to jsou dvě animace na různých objektech navázané do sebe. Neboli v momentě, kdy se otevře víko zadní části popelářského vozu, je spuštěna animace na objektu kontejneru. Po proběhnutí této animace pokračuje první, která zavře víko.

Proces rozpoznání vozu v oblasti naložení je zajištěn skriptem přiřazeným k objektům popelnic. Funkčnost tohoto skriptu je založena na metodách, které zjistí, zda je objekt s tagem („tag“ definuje objekt, podle originálního názvu, tak že je dohledatelný)

„Car“ ve vyznačeném místě před popelnicí. Následně se spustí animace a po proběhnutí je objekt vymazán, přičemž se hráči připíší body za vyložení popelnice. Vyznačené místo před popelnicí je zprostředkováno pomocí objektu kostky, ze které byl odstraněn mesh vykreslující grafickou stránku objektu. Jedná se o neviditelný objekt, jež má nastaven collider(síť ohraničující objekt a přidávající zhmotnění) se zapnutým „is Trigger“. Právě tato funkce slouží jako spouštěč na objektu, tak aby reagoval podle toho, zda je „is Trigger“ true nebo false.

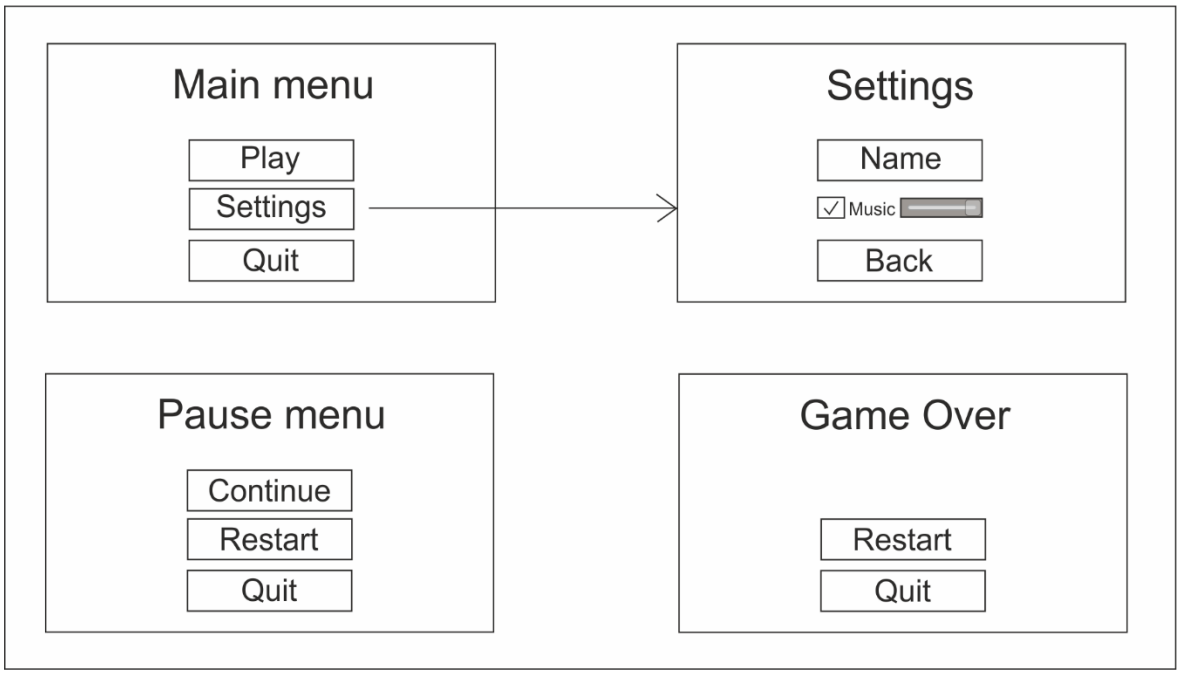
```
void OnTriggerEnter (Collider pop)
	{
		if (pop.gameObject.tag == "Car")
		{
			isTrigger = true;
		}
	}
```

Tato metoda zjistí, zda se objekt s tagem „Car“ nachází v collideru objektu, na který je aplikován skript. Aby hráč rychle a jednoduše našel místo, kde zastavit pro naložení popelnice, byl do hry přidán doplněk particle systém, jenž je součásti Unity. Jeho využití je široké, od efektu ohně či vodopádu až k napodobení deště či sněhu. V případě hry Garbage ride slouží ke zvýraznění místa pro zastavení vozidla k naložení popelnice. Particle system se přidává jako komponent k objektu, aby vypadal jako místo pro zastavení, je nastaven podle čtvercového meshe, u kterého je nastaveno zobrazení pouze na hranách. Díky tomu je zajištěn efekt osvětlující místo pro zastavení vozidla.

4.3.4. Main Menu

Hlavní menu hry je první, co hráč uvidí, když spustí hru. V případě vyvíjené hry není potřeba mít široké možnosti nastavení, proto je menu tvořeno z jednoho obrázku a tlačítek pro spuštění, nastavení a ukončení hry. Autor se snažil vytvořit hlavní menu animovaných stylem, proto při spuštění aplikace nejdříve přijedu název hry a posléze

tlačítka pro ovládání. Pro tyto animace je opět využit animační systém integrovaný v Unity. Systém menu je názorně zobrazen pomocí diagramu níže:



Obrázek 4: Systém menu Zdroj: Vlastní

Posledním menu, které se ve hře vyskytuje, je settings menu. Jedná se o podmenu, z tohoto důvodu nemá vlastní kapitolu. V menu nastavení je k dispozici ovládání zvuku, možnost nastavení jména profilu (Nickname) a návrat do hlavního menu. Z důvodu dodržení jednotného stylu má menu nastavení animovaná tlačítka obdobným způsobem jako hlavní menu.

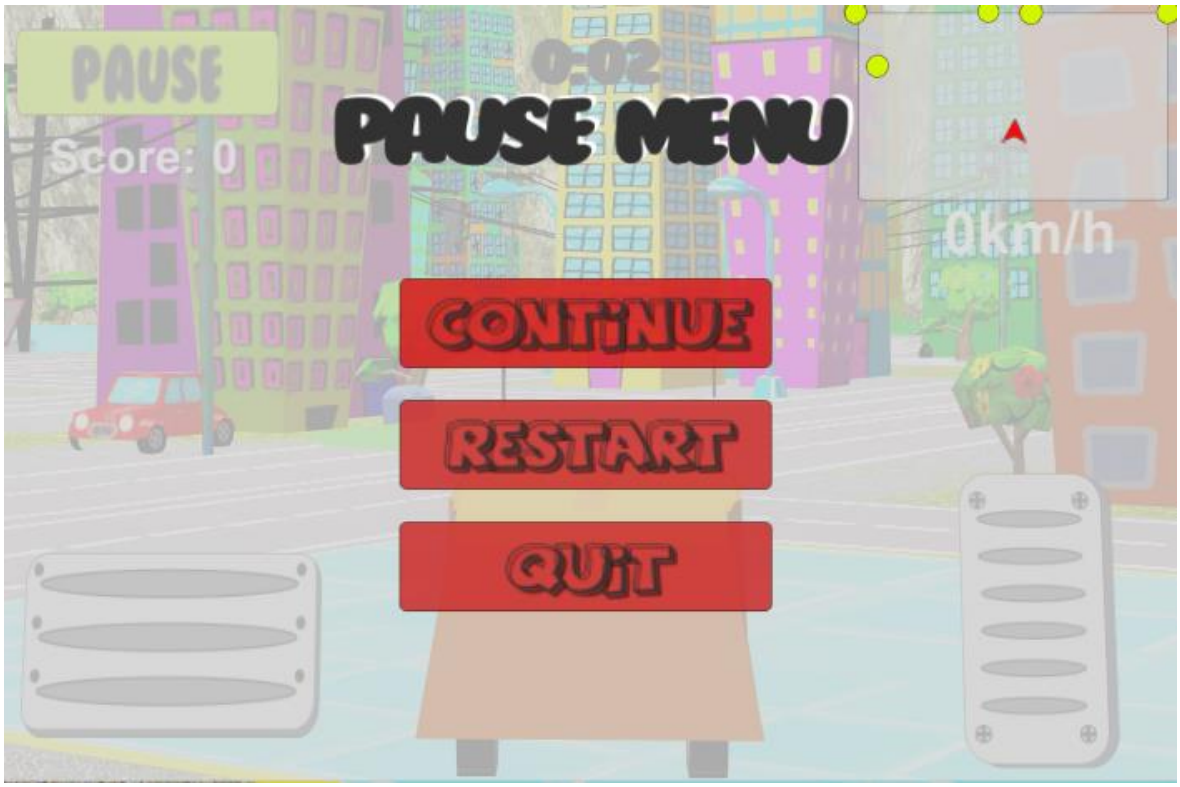
4.3.5. Pause Menu

Vytvoření pause menu je spojeno s funkcí Canvas, která se stará o zobrazování UI elementů. Přidáním panelu lze vytvořit nové menu s různými položkami. V případě pause menu jsou těmito položkami tlačítka. Celkově se jedná o tři tlačítka continue, restart, quit. Podle názvů je možné odvodit účel. Každé z tlačítek je nastaveno, aby volalo funkci, která provede daný úkon. Důležitou vlastnosti skriptu zajišťující tyto funkce je následující proměnná:

```
Time.timeScale = 0;
```

Pokud je nastavena na 1, čas ve hře běží normální rychlostí, jakmile se tato hodnota změní na 0, je čas zastaven, takže se neprovádí žádné operace ani výpočty. Právě pomocí této proměnné byla vytvořena pauza, což znamená, že v momentě kliknutí se tato hodnota změní na 0 a čas je zastaven. V opačném případě kliknutí na tlačítko „Continue“ změní tuto hodnotu zpět na 1 a hra se opět spustí.

Zmíněná nastavení jsou vytvořena ve skriptu „UIManagerScript“, který nejenom obsahuje funkce na volání pause metody, ale také funkce, jež se provádí po stisku některého zvýše zmíněných tlačítek. Tento skript je vlastně celou třídou, která se stará o chod tlačítek a všech menu obsažených ve hře. Pro představu je níže uveden obrázek displeje v pause menu.



Obrázek 5: Pause menu Zdroj: Vlastní

4.3.6. Game Over

Pokud doběhne odpočet času, znamená to, že hráč nestihl nasbírat dostatečný počet bodů a musí úroveň opakovat nebo došlo k převrácení vozu. V takovém případě se

přes displej zobrazí okno s nápisem „Game Over“ a s tlačítkem „Restart“ a „Quit“. Tohoto chování bylo docíleno použitím funkce Image v rámci již známého systému Canvas. Obrázek překryje celý displej šedou barvou s lehkou průhledností a nápisem, dodatečně přidaným. Pro zobrazení obrázku byl opět použit animátor, jež je zavolán po doběhnutí času, kdy provede animaci vyskočení okna a textu.

4.3.7. Časomíra

Objekt časomíra je zobrazen v horní části obrazovky. Jedná se o čas, ve kterém je potřeba splnit všechny úkoly tj. naložit kontejnery za určitý počet bodů ve městě. Opět se jedná o součást canvasu, tedy uživatelského prostředí pro hraní hry. Tento doplněk byl vytvořen přes objekt Text, na který byl aplikován napsaný skript. Hlavní částí tohoto skriptu je odpočet času od zvolené proměnné float. Algoritmus zajišťující tuto funkci vypadá následovně:

```
if (Seconds <= 0)
        {
            Seconds = 59;
            if (Minutes >= 1)
                {
                    Minutes--;
                }
            else
            {
                Minutes = 0;
                Seconds = 0;
            }
        }
        else
        {
            Seconds -= Time.deltaTime;
        }
```

Uvedený kód není potřeba komentovat, jeho konstrukce je pochopitelná i pro laika. Aby byla zaručena funkčnost tohoto kódu je potřeba ho umístit do třídy Update, která pravidelně volá funkce, jež jsou v ní obsaženy. Pro správné zobrazení je ještě potřeba kód

převést do textového řetězce, jelikož komponent Text může vypisovat pouze textové proměnné. V jazyce C# je převod do textu proveden následovně:

```
textObject.text = Minutes.ToString("f0") + ":" +
Seconds.ToString("f0");
```

Na levé straně zápisu je objekt, který odkazuje na komponent text, do kterého je vypsán převedený odpočet času.

4.3.8. Mini-mapa

Všechny kontejnery, které se nachází v dané lokalitě a je třeba je v časovém rozmezí vyprázdnit, jsou zobrazeny na mini-mapě. Mini-mapa představuje jednoduchou navigaci v prostředí, kde červená tečka určuje pozici auta, žluté tečky ukazují postavení popelnic a zelené pozici kontejnerů. Pokud se kontejner nenachází v blízkosti auta, žlutá značka se pohybuje na hranici mini-mapy vždy ve směru, kterým by měl hráč jet pro nalezení daného kontejneru. Barevné označení popelnic a kontejnerů je z důvodu rozdílných bodů získaných za vyložení.

Způsob vytvoření mini-mapy v Unity se provádí využitím Canvas funkce panel. Obdobně jako se vytvářelo PauseMenu, tak i mini-mapa využívá GUI rozhraní Unity. Přidáním panelu, u kterého se upraví velikost, aby byl zobrazen pouze v pravém horním rohu. Takto je zajištěno vykreslení malého obdélníku, do kterého se vykreslují tečky odkazující na objekty na scéně. Tečky jsou zobrazeny jako obrázky s barvou, podle druhu objektu.

Mini-mapa se skládá ze dvou skriptů, přičemž jeden je přiřazen k panelu mini-mapy a druhý k obrázkům představujícím kontejnery. Tyto skripty jsou vzájemně provázány a pomocí metod upravující pozice objektů na mini-mapě vytvářejí funkční systém navigace. Předlohou k tomuto skriptu byl návod na vytváření mini-mapy v Unity 4.6 ze stránek Game Pie. (32) Hlavní metoda zajišťující převedení souřadnic auta a kontejnerů do 2D prostoru mini-mapy je zobrazena níže:

```
public Vector2 TransformPosition(Vector3 position)

      {

            Vector3 offset = position - Target.position;

            Vector2 newPosition = offset.x * XRotation;

            newPosition += offset.z * YRotation;

            newPosition *= ZoomLevel;

            return newPosition;

      }
```

Veřejná metoda s názvem TransformPosition s parametrem position typu Vector3, převádí pozici auta a kontejnerů do 2D zobrazení na mini-mapu. Převod je zprostředkován pomocí proměnné offset, která reprezentuje Vector3 a do níž je uložena aktuální pozice mínus pozice cílového objektu. K proměnné Target typu GameObject je přiřazen objekt kontejneru. Do newPosition je poté uložena pozice offset na ose x vynásobena XRotation, která je vlastností Vector2.left. Tento zápis zajišťuje správnou pozici kontejneru i při otáčení s vozem a převádí pozici kontejnerů do zobrazení vy vytyčeném obdélníku. V proměnné newPosition typu Vector2 je uložena převedená pozice z 3D prostoru do 2D na ose z a vynásobena proměnnou YRotation, jež obsahuje Vector2.right neboli změny v 2D prostoru na ose x. Aby body v případě, že prostředí bude rozsáhlé, nebyly zobrazeny na mini-mapě ve velmi malém měřítku, je do metody přidána proměnná ZoomLevel, která násobí hodnoty v 2D prostoru přednastavenou velikostí. Poslední řádek pouze navrací novou pozici objektů na scéně.

Další metoda, jež je součástí mini-map skriptu, vytváří hranice mini-mapy, tak aby všechny body znázorňující objekty zájmu na scéně, byly vždy zobrazeny. Přestože se kontejner nachází mimo zobrazenou část mini-mapy, stále je znázorněn na okraji mini-mapy ve směru, který míří na daný objekt. Středem mini-mapy je vždy bod představující hráče (vozidlo). Kódový zápis této metody je zobrazen níže:

```
public Vector2 MoveInside(Vector2 point)
	{
		Rect mapRect = GetComponent<RectTransform> ().rect;
		point = Vector2.Max (point, mapRect.min);
		point = Vector2.Min (point, mapRect.max);
		return point;
	}
```

Znázorněná metoda výše provádí omezení pohybu objektů zobrazených na mini-mapě. Jakmile je zavolána tato metoda je jí předán parametr newPosition, jež je poté metodou převeden do bodu vykresleného v obdélníku ohraničujícího mini-mapu.

4.3.9. Skóre

Součástí většiny her je proměnná zaznamenávající průběžné skóre. V případě hry Garbage ride je tomu obdobně. Skóre je zobrazováno na displeji zařízení a poskytuje tím průběžně informaci hráči, kolik bodů již nasbíral. Ve vyvíjené hře se sbírají body za naložení popelnice, když je to malá popelnice získává hráč menší počet bodů oproti naložení velkého kontejneru. Vždy je potřeba dosáhnout určitého skóre, aby hráč mohl postoupit do další úrovně. Tím vzniká pro hráče taktická otázka, jelikož se musí rozhodnout, zda absolvovat delší cestu za ziskem více bodů, či sbírat menší popelnice a postupně dosáhnout na cílovou hranici. V následujícím skriptu byla poprvé využita statická proměnná, která je ideální k použití na záznam nasbíraných bodů. Díky klíčovému slovu static, lze do této proměnné přistupovat odkudkoli je potřeba. Právě vlastnost, že není nutné vždy vytvářet novou instanci objektu, nýbrž se pouze odkazovat na statickou proměnnou, má největší uplatnění, když je použita jedinečně. Pokud totiž dojde ke změně hodnoty této proměnné, vždy se změní i všechny hodnoty ostatních instancí. Skript na záznam skóre je vypsán níže:

```
public class ScoreMan : MonoBehaviour
{
	public static int score = 0;
	private Text text;

	void Awake()
	{
		text = GetComponent<Text> ();
	}
	void FixedUpdate()
	{
		text.text = "Score: " + score;
	}
}
```

Na skriptu ScoreMan je zřejmé, že zásadní odlišnou informací je veřejná statická proměnná score typu int. Zbytek kódu pouze získá komponent Text a vypíše do něj aktuální skóre na základě proměnné skóre. Tento jednoduchý skript provádí zobrazení skóre na displej. Stačí pouze přiřadit nějakému objektu hodnotu, která bude navyšovat skóre a poté pomocí přičítání se do statické proměnné score ukládá hodnota ve scoreValue.

Pokud by se nejednalo o statickou proměnnou, bylo by nezbytné provést vytvoření nové instance objektu a tím propojit dva nezávislé skripty. První skript je přiřazen k textovému objektu Score, přičemž druhý zápis je vložen do skriptu napojeného na Kontejnery. V momentě, kdy je kontejner naložen a zmizí, přičte se hráči určitý počet bodů, podle velikosti kontejneru.

4.3.10. Ukazatel rychlosti

Obdobným způsobem jako byl vytvořen ukazatel skóre, proběhlo přidání tachometru. Jedná se digitální ukazatel aktuální rychlosti, přičemž je na něj aplikováno

omezení rychlosti na hodnotu 90 km/h. Opět bylo využito statické proměnné, aby se do ní ulehčil přístup z ostatních skriptů. Výpis rychlosti na displej zařízení probíhá totožným způsobem jako výpis skóre. Do textového objektu je vypsána hodnota ze statické proměnné. Skript na výpočet rychlosti vypadá následovně:

```
actualSpeed = 2 * Mathf.PI * WheelColliderRL.radius *
WheelColliderRL.rpm * 60/1000;

actualSpeed = Mathf.Round (actualSpeed);
```

Do proměnné actualSpeed se ukládá vypočtená hodnota obvodu kola (2*Pí*r) vynásobená rychlostí otáčení, která je převedená na km/h. Další řádek pouze zaokrouhluje rychlost na celé číslo.

4.3.11. Skript na ovládání auta

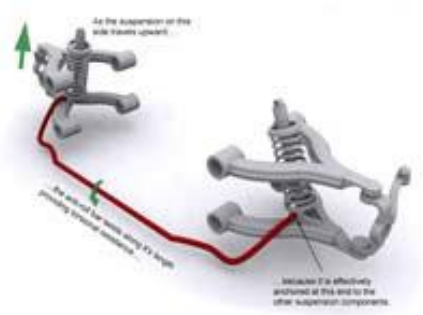
Klíčovým bodem vyvíjené hry je ovládání popelářského auta prostřednictvím implementování fyzických vlastností vozu na herní model. Unity obsahuje metody zajišťující část z této logiky. Jedná se o použití WheelCollider, jež je speciální collider na kola vozu vlastnící základní fyzické vlastnosti reálných kol. Možnostmi ve vlastnostech tohoto komponentu lze nastavit přilnavost a pérování pro každé kolo zvlášť.

Další pomocnou metodou je motorTorque, který se stará o reálnou jízdu auta. Pomocí Newtonových zákonů, dává sílu kolům, na které působí hnací síla. Je možné si vybrat, zda auto bude mít náhon na přední, zadní nebo na všechna čtyři kola. Skript na zajištění pohonu je k dispozici níže:

```
if (verticalInput == 1)
      {
            if (actualSpeed <= maxSpeed)
                  {
                              speed = CarForce * verticalInput;
                              WheelColliderRL.motorTorque = speed;
                              WheelColliderRR.motorTorque = speed;
                  }
                  else
                  {
                              WheelColliderRL.motorTorque = 0;
                              WheelColliderRR.motorTorque = 0;
                  }
      }
```

Na skriptu je zřejmé, že popelářské auto má náhon pouze na zadní kola, což by opět mělo odpovídat reálnému popelářskému vozu. Hodnota síly dodávána motorTorque je vypočtena jednoduše podle zvolené proměnné CarForce a násobena hodnotou verticalInput, která je jedna, když je stisknuto tlačítko plynu a nula, když není. Dále je přidána podmínka omezující nejvyšší možnou rychlost podle hodnoty v proměnné maxSpeed, aktuálně nastavené na 90.

Pomocí výše zmíněných postupů je možné provést simulaci základního pohonu automobilu rovnající se skutečnému automobilu. Poté ovšem vyvstává problém při zatáčení. Jelikož auto v zatáčce přenáší velkou váhu na venkovní kolo, je nutné vytvořit fyzické vlastnosti napodobující stabilizátor. V reálném autě stabilizátor pomáhá přenášet sílu vyvinutou rychlostí a váhou vozu v zatáčení na jedno z předních kol či zadních kol.



Obrázek 6 Stabilizátor Zdroj: http://www.carbibles.com/suspension_bible_pg4.html

Na obrázku výše je znázorněn skutečný stabilizátor auta. Červeně označená tyč spojující pružení předních kol, přenáší působení síly z jednoho kola na druhé. Tím je zamezeno převrácení vozu při extrémních průjezdech zatáček. Obdobným způsobem byl naprogramován stabilizátor pro popelářské auto, aby se zabránilo převrácení vozu při zatáčení. Jelikož se jedná o relativně složitý mechanismus, byl jako vzor použit skript antiRoll bar od Edy's vehicle physics, který je napsán v JavaScriptu. (33)

V C# je tento skript tvořen pomocí podmínek upravující působení sil. Pokud při zatáčení směřuje velká síla na jedno z předních kol. Skript zavolá metodu, která tuto sílu převede částečně na druhé kolo a tím zabrání převrácení vozu. Zároveň je upraven moment tření, který způsobí při velké rychlosti, že auto jede ve skluzu místo, aby se převrátilo. Nejlépe jde tento mechanismus vysvětlit na přímo použitém skriptu níže:

```
groundedFL = WheelColliderFL.GetGroundHit (out hitFL);

groundedFR = WheelColliderFR.GetGroundHit (out hitFR);

if (groundedFL){

      travelFL = (-WheelColliderFL.transform.InverseTransformPoint
(hitFL.point).y - WheelColliderFL.radius)/
WheelColliderFL.suspensionDistance;}

      else {

                  travelFL = 1.0f;

            }

if (groundedFR){

      travelFR = (-WheelColliderFR.transform.InverseTransformPoint
(hitFR.point).y - WheelColliderFR.radius)/
WheelColliderFR.suspensionDistance;}

      else {

                  travelFR = 1.0f;

            }

antiRollForce = (travelFL - travelFR) * antiRoll;
```

Stabilizátor neustále počítá sílu v bodě, kde se kolo dotýká země. Tento princip je v kódu zapsán jako podmínka, když je kolo na zemi, spočítej travelFL nebo travelFR hodnotu. Pokud se kolo nedotýká země, nastav travel hodnotu na jedna. Jelikož se výpočet pohybuje v intervalu <-1, 1>, znamená hodnota travel nastavená na jedna převedení nejvíce síly na opačné kolo. Následně se vypočtené hodnoty pro každé kolo od sebe odečítají a násobí proměnnou antiRoll, která je inicializována na vysokou hodnotu 30000 a zvyšuje tím hodnotu síly působící na jedno z kol. Po těchto výpočtech se ukládá vypočtená hodnota do proměnné antiRollForce, jež určuje, kolik síly je nutné převést na druhé kolo. Přenesení síly je zprostředkováno v následující částí skriptu:

```
if (groundedFL)
                    {
      rigidbody.AddForceAtPosition (WheelColliderFL.transform.up * -
antiRollForce, WheelColliderFL.transform.position);
                    }
if (groundedFR)
                    {
      rigidbody.AddForceAtPosition (WheelColliderFR.transform.up *
antiRollForce, WheelColliderFR.transform.position);
                    }
```

Na uvedeném skriptu je znázorněn přenos síly na dané kolo, podle působení sily v zatáčení zmíněné výše. Funkce AddForcePosition má dva parametry. Prvním z nich je Vector3 dávající sílu, a druhý představuje pozici, na kterou síla působí. Tyto dva parametry jsou v kódu zastoupeny jako změna souřadnic WheelColliderFL/FR směrem nahoru násobena vypočtenou hodnotou antiRollForce. Poté je tato síla aplikována na pozici WheelColliderFL/FR. Tímto způsobem je docíleno podobných vlastností stabilizátoru přibližující se opravdovému stabilizačnímu systému v autě. Uvedený příklad je pouze na přední nápravu. V případě zadní nápravy se situace opakuje, a proto zde není tento skript uveden.

4.4. Výsledná podoba hry Garbage ride

Cílem této práce bylo vytvořit mobilní aplikaci s využitím Unity 3D enginu. Dle autorova rozhodnutí, padla volba na mobilní hru. Aby bylo možné demonstrovat široké možnosti mobilních zařízení, byla zvolena závodní hra využívající gyroskop a akcelerometr.

Neocenitelná pomůcka při vývoji takové hry je skvělá dokumentace Unity i se Scripting API, bez něhož by nebylo možné práci dokončit, jelikož mnoho problémů vzniklých během vývoje se vyřešilo důkladným přečtením těchto pomocných zdrojů. Autor tuto pomoc využil zejména v momentě programování fyzických vlastností vozu, kde celá problematika WheelCollider je přehledně vysvětlena ve zmíněném Scripting API.

V praktické části diplomové práce je vysvětlen celý postup vývoje od prázdné scény, až po finální skript na ovládání vozu. Následující část obsahuje shrnutí celkového dojmu z aplikace s grafickými ukázkami.

Výsledná hra je zcela funkční, není sice ve stavu, že by se dala publikovat na AppStoru, ale v případě dodělání více úrovní a pár navrhnutých vylepšeních, je tato možnost stále otevřena. Hra je jednoduchá po grafické stránce, zejména z důvodu použitých modelů, jež by mohly být propracovanější a tím pozvednout hru na vyšší úroveň. Přestože je hra graficky jednoduchá, namítaje nutno konstatovat, že Unity vykresluje pěknou grafiku i v případě jednoduchých modelů a textur. Modely samy o sobě představují téma pro samostatnou diplomovou práci a právě z tohoto důvodu byly použity již vytvořené modely z Asset storu a nově byl vytvořen pouze jednoduchý model auta. Finální podoba hry je zobrazena na obrázku níže:

Obrázek 7 Hlavní menu Zdroj: Vlastní

Hlavní menu je ve skutečnosti animované a při startu hry postupně přijíždějí objekty název hry a tlačítka. Volby menu není třeba opakovat, jsou popsány v praktické části práce. Jako vhodná ukázka ze hry, byl vybrán moment zastavení v požadovaném místě, znázorněném pomocí efektu svítícího obdélníku. Jakmile je vůz ve vyznačené oblasti spustí se animace nakládání kontejneru. Pro zpestření hry byly využity netradiční fonty korespondující s grafickou stránkou hry podporující kreslené prostředí.

Autor se snažil dodržet navržené rozvržení obrazovky dle WireFrame návrhu. Popelářský vůz se nachází v centru displeje, po stranách jsou tlačítka plynu a brzdy. V horní části displeje je mini-mapa, pod ní ukazatel rychlosti, ten nebyl ve WireFemu zohledněn. Nahoře uprostřed je časomíra a v levé části tlačítko Pause, přičemž pod ním se nalézá ukazatel skóre.

Vůz je dobře ovladatelný, zatáčení náklonem splňuje požadavky. S čím si není autor zcela jist, je měřítko auta vůči okolnímu světu. Některé zatáčky jsou příliš ostré a zatočení je nutné provádět ve velmi nízké rychlosti, což na druhou stranu odpovídá reálnému chování v městských uličkách. Orientace pomocí mini-mapy je po lehkém

tréninku bezproblémová. Ačkoli má mobilní telefon iPhone 4, v porovnání s nynějším trendem, relativně malý displej. Tlačítka jsou dostatečně ergonomická a nestává se, že by se hráč uklikl vedle.

Obrázek 8 Ukázka ze hry Zdroj: Vlastní

Grafická podoba hry Garbage ride na mobilním zařízení není zcela adekvátní, protože probíhá skrze Unity Remote, jež z důvodu nižších přenosových rychlostí není schopen vykreslit hru s dostatečnou grafickou kvalitou. Pokud by se celá aplikace zkompilovala a přenesla na cílové zařízení, vypadala by jako obrázky z Unity výše. Aktuální zobrazení je o něco více rozkreslené ze zmíněných důvodů.

Obrázek 9 Ukázka na mobilním zařízení Zdroj: Vlastní

4.4.1. Výhody Unity 3D enginu

Nebýt Unity 3D, projekt Garbage ride by představoval dlouhodobý projekt s časovou náročností minimálně jeden rok. Právě přínos Unity umožnil navrhnout a vytvořit hru v rámci měsíců. Dle odhadu autora dva měsíce dolaďování by umožnily projekt publikovat na Apple store. Velkým přínosem ze strany Unity, byly již implementované funkce v programu, jako jsou: terrain, skybox, canvas, particle systém a určitě možnost vše rovnou přizpůsobovat pro zvolenou mobilní platformu. Všechny materiály mají shadery vhodné pro mobilní zařízení a sprite jsou přímo určené na iPhone 4 rozlišení.
Zajisté jednou z nejvíce používaných vlastností Unity během vývoje byla možnost spouštět vytvořené skripty přímo po uložení a tím vyzkoušet správnou funkčnost. Pokud by bylo zapotřebí pokaždé kompilovat celou hru, proces vývoje by se jistě velmi zpomalil. Nový systém UI, přidaný od verze Unity 4.6, byl přínosný z hlediska ušetřeného času, umožnil místo psaní alternativ ke každému dotykovému tlačítku, pouze vybrat eventTrigger k vybranému objektu a zvolit si, jaká metoda bude zavolána po stisku příslušného tlačítka. Tímto způsobem je navržen celý systém menu a dotykových tlačítek.

Nesporně prospěšnou funkcí Unity je podpora dotykových displejů skrze EventSystem, jenž zprostředkovává fungování dotyku na mobilních platformách, ale zároveň umožňuje využívat myš, jako zdroj doteku v případě, že hra běží na počítači.

4.4.2. Možnosti vylepšení

Vždy existují možnosti vylepšení, ať už se jedná o sebe dokonalejší produkt. Někdy tato vylepšení představují změny drobných herních prvků, jindy může jít o změny v celé logice hry. Následující návrhy jsou zásadními úpravami, které by mohly být vytvořeny v průběhu měsíců. Dvě změny, které by výrazně ovlivnily celou hru, jsou zlepšení grafické stránky modelů a přidání multi-playeru. Propojení aplikace s účty na sociálních sítích je přínosem hlavně ze strany marketingu, jelikož se hra zviditelní pouhým používáním. Pokud by uživatel hrál, automaticky by se přidal příspěvek na Facebook či twitter a všichni kamarádi by viděli, že právě hraje hru Garbage ride. V tento moment je zásadním nedostatkem počet úrovní. Jak se taková úroveň vytváří, je popsáno v kapitole vytváření světa. Bohužel z důvodu omezených časových možností byla vytvořena pouze jedna úroveň. Mezi další možnosti vylepšení, zajisté patří:

- Přidání umělé inteligence okolním autům a postavám, by proměnilo doposud statickou scénu na živý svět, hemžící se samovolně pohyblivými objekty.
- Součástí menu by mohl být globální žebříček nejrychlejších časů ke každé úrovni. Soutěživost je vždy zdrojem větší popularity a zájmu o danou hru.
- Mini-mapu lze vylepšit přidáním kamery, která snímá povrch a zobrazuje pouze silnice. Hráč by poté přesněji věděl, kudy má jet za kontejnerem.
- Jelikož ve skutečném světě jsou kontejnery vyváženy v ranních hodinách. Aplikovat do hry reálný východ slunce, čímž by během jízdy, byly vykresleny některé části tmavě a na některé by pomalu začínalo svítit sluneční světlo. To vše by vytvářelo pěkné scenérie poskytující možnosti pro využití kvalitního stínování objektů, které Unity nabízí. Bohužel by tento přídavek potřeboval více výkonu a tím i jiné cílové zařízení.
- Problematika svozu odpadu má v posledních letech trend třídit odpad podle druhu a tím pomáhat recyklovat. Hra by pomocí změny vozu na např. modrý vůz svážející pouze plasty, mohla získat edukativní náboj, protože by učila, jaké kontejnery jsou vhodně pro určitý odpad.

Možností vylepšení je celá řada, ale z pohledu autora nelze vytvářet takový projekt pouze jednou osobou. Není reálné skloubit všechny činnosti vztahující se k vytvoření kvalitní hry na mobilní zařízení. Být jednou osobou programátorem, grafikem, návrhářem, testerem, je velmi složité a dle autora to není šťastný způsob jak se snažit prosadit. Na vysoké škole byla velká část projektů sdílená pro skupinu studentů, kteří si rozdělili práci a tím dokázali kvalitně splnit zadání. S vývojem aplikace je to obdobné, proto jako doporučení je uvedeno spolupracovat na projektech s ostatními a docílit tím kvalitnějších výstupů. Toto zajisté neplatí u jednoduchých 2D her, kde může snadná hra jako Flappy Bird, která měla úspěch zejména díky své grafické jednoduchosti, ale jinak celkové náročnosti hry. Úspěch hry Flappy Bird byl takový, že její vývojář pobíral měsíčně milion dolarů. Někdy je jednoduchý nápad více, než kvalitní, ale nezáživná 3D hra.

5. Závěr

Současná doba je plná technologických vymožeností, jež lidem usnadňují život. Mezi tyto vymoženosti se řadí mobilní zařízení. Tato práce pojednává o možnostech těchto zařízení a poukazuje na fakt, že potenciál vidí, nejenom prodejci, ale také vývojářské společnosti nabízející všemožné programy pro tvorbu aplikací na tyto přístroje. V praktické části byly zmíněny konkurenční softwary i s výsledným shrnutím. Konkurenční programy k Unity 3D lze dělit na dvě skupiny. První z nich jsou nástroje přizpůsobené pro tvorbu aplikací na mobilní platformy. Druhá skupina představuje univerzální nástroje pro vývoj her, programů na různé platformy. Tyto sofistikované pomůcky pomáhají vytvářet nepřeberné množství aplikací, jež se snaží uspět v přetechnizovaném světě. Někdy se to podaří, jindy se ovšem pouze zařadí na seznam neúspěšných, řadících se na neviditelná místa v hierarchie všech aplikací.

Hlavním cílem diplomové práci bylo demonstrovat využití Unity 3D enginu pro vytvoření mobilní hry. Využít dostupné funkce tohoto programu a dosáhnout skutečné hry, spustitelné na reálném zařízení. Někdy se může tento cíl stát během na dlouhou trať. Nejenže to nelze provést, pokud je cíleno na iOS, bez registrace na AppStoru, ale také pro přístup k většině funkcí Unity je třeba zakoupit tento software ve verzi Pro. Z těchto důvodů je vytvořená hra ve stavu Beta verze, tedy lze ji spustit na zařízení, ale pouze jako emulátor běžící zároveň na počítači, tak na mobilním zařízení.

Průběh vývoje obnáší zaručeně znalosti programovacího jazyka C# či JavaScript a schopnost správně aplikovat vytvořené skripty v prostředí editoru. Skriptování v Unity má své specifika a ne vždy je vše vyřešeno klasickou cestou. V tento moment je vhodné vyzdvihnout především skvělou dokumentaci Unity a podporu ze strany velké uživatelské komunity. Díky těmto vlastnostem Unity předčí konkurenční softwary, jelikož nenabízejí takové možnosti nalezení pomoci v případě nesnází. Přestože programy jako CryEngine, Source engine nebo Unreal engine předčí Unity po grafické stránce, práce v nich je ale o poznání těžší a nelze tak snadno své záměry úspěšně finalizovat.

Teoretická část pojednává nejenom o konkurenčních softwarech, ale také charakterizuje vlastnosti Unity 3D a skriptovací jazyk C#. Tyto kapitoly obsahují vysvětlující

informace k základní práci v prostředí Unity editoru včetně definování druhů modelů, světel, zvuků a kamer. Poslední část je zaměřena na základní vlastnosti jazyka C#, jež je jediným jazykem použitým k vývoji aplikace. Jazyk je vysvětlen z hlediska objektového programování a jsou zmíněny hlavní přednosti a krátká historie vzniku.

Praktická část je zcela zaměřena na vývoj mobilní aplikace. Jsou zde uvedeny kroky předcházející vývoji s obecným grafickým zobrazením aplikace a systémem hry vysvětleným pomocí vývojového diagramu. Zbývající část se věnuje jednotlivým krokům vedoucím k finální podobě hry. Ta představuje poslední kapitolu shrnující předchozí vývoj s výstupy z aplikace odkazující na výslednou podobu. Kroky jsou řazeny od procesů probíhajících v editoru Unity, až po skriptování v přidruženém frameworku MonoDevelop, kde jsou napsány všechny skripty v aplikaci. Tvorba a popis většiny skriptů obsahuje ukázky jednotlivých skriptů zajišťující náhled do útrob celé hry. Aplikace se skládá z počtu menších skriptů vzájemně propojených do uceleného komplexu herního systému. Právě provázanost mezi skripty tvoří podstatu celé hry, jelikož reakce jednoho úkonu vyvolá akci na jiném objektu. Tím je míněno, že pokud se vůz nachází ve vyznačeném obdélníku, je spuštěn skript, který objekt zničí a připíše body do skóre, přičemž další skript představující bod kontejneru na mini-mapě je zničen, když neexistuje cílový objekt, na který je navázán.

Z pohledu autora je Unity kvalitní program pro začínající vývojáře, jež trochu znají programování v C# a chtějí proniknout do segmentu tvorby, ať už her pro počítače, konzole nebo webový prohlížeč, tak pro tvorbu aplikací na mobilní platformy. Poté zejména široká podpora platforem umožňuje jeden projekt využít na více platformách a tím získat větší možnosti distribuce. Celkově je Unity prostředkem, který nabízí možnosti tvořit originální software pro různá zařízení. Přesto je nejdůležitějším prvkem nápad, jež dává aplikaci to hlavní kouzlo, díky kterému má aplikace příležitost proniknout do zařízení většiny domácností. Příkladem mohou být i čeští vývojáři. Někteří z nich dokázali vytvořit natolik kvalitní aplikace, že se dostali do světového podvědomí a díky tomu například dostali zajímavé pracovní nabídky.

Výsledkem celé diplomové práce je ukázka možností Unity. Nejlepším možným prostředkem k tomu, je vytvořená aplikace. Při vývoji bylo využito většiny vlastností mobilního telefonu iPhone 4. Pohyb vozu je zprostředkován skrze dotyková tlačítka na

displeji. Zatáčení vozu zajišťuje gyroskop s akcelometrem, takže pouhé natočení přístroje zatočí s vozem ve hře. Tento způsob zatáčení je ideální pro ovládání, jelikož je umožněno převést požadovanou intenzitu, kterou stiskem jednoho tlačítka nelze nahradit. Vytvořená hra je výkonově na úrovni iPhone 4, pokud by se cílilo na výkonnější zařízení, byly by možnosti z hlediska grafického zobrazení, mnohem rozsáhlejší. Aplikace na moderní telefony se velmi rychle blíží úrovni počítačových her. Čím více se zmenšuje mezera mezi aplikací pro mobilní platformu a programem pro počítač, tím více se otevírají možnosti vývoje jedné aplikace pro všechny platformy. V blízké budoucnosti se tyto platformy jistým způsobem doženou a poté hra pro počítač bude totožná s aplikací na mobilní platformu. Unity tento trend podporuje a nejspíše v něj také věří.

6. Bibliografie

1. **Perez, Sarah**. itunes app store now has 1,2 million apps hes seen 75 billion downloads to date. *techcrunch.* [Online] 2. Červen 2014. [Citace: 20. Leden 2015.] *http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/.*

2. **NetBeans**. *netbeans.* [Online] Oracle. [Citace: 23. Leden 2015.] *https://netbeans.org/.*

3. **Grudl, david.** Nette. *nette framework.* [Online] [Citace: 12. Únor 2015.] *http://nette.org/cs/.*

4. **PcMag**. *api.* [Online] [Citace: 15. Únor 2015.] *http://www.pcmag.com/encyclopedia/term/37856/api.*

5. **SDK.** *techterms.* [Online] 15. Duben 2010. [Citace: 30. Leden 2015.] *http://techterms.com/definition/sdk.*

6. **Ondřej, Čada**. *Objektové programování: naučte se pravidla objektového myšlení.* Praha : 1. vyd., Grada, 2009. ISBN 978-80-247-2745-5..

7. **Galitz, Wilbert**. *The essential guide to user interface design an introduction to GUI design principles and techniques: naučte se pravidla objektového myšlení.* Indianapolis : 3rd ed., Wiley Pub, 2007. ISBN 04-701-4622-2.

8. **McWherter, Jeff. Gowell, Scott**. *Professional mobile application development: naučte se pravidla objektového myšlení.* 3rd ed., Indianapolis : John Wiley, 2012. ISBN 11-182-4068-5.

9. **Vávrů, Jiří.** *IPhone: vývoj aplikací.* Praha : 1. vyd., Průvodce (Grada), 2012. ISBN 978-80-247-4457-5..

10. **xCode** - features. *developer.apple.com.* [Online] Apple. [Citace: 14. Únor 2015.] *https://developer.apple.com/xcode/features/.*

11. **Monogame.** *monogame.* [Online] [Citace: 16. Únor 2015.] *http://www.monogame.net/.*

12. **Getting Started.** *GitHub.* [Online] [Citace: 5. Únor 2015.] *https://github.com/mono/MonoGame/wiki/Tutorials%3A-Getting-Started.*

13. ***Xamarin.*** platforms [Online] [Citace: 28. Leden 2015.] *http://xamarin.com/platform.*

14. ***YOYOgames.*** Workflow [Online] [Citace: 2. Únor 2015.] *http://www.yoyogames.com/studio/workflow.*

15. **Source.** *Moddb.* [Online] [Citace: 16. Únor 2015.] *http://www.moddb.com/engines/source.*

16. **Maximumpc**. *10 best unreal engine games.* [Online] 14. Březen 2014. [Citace: 6. Únor 2015.] *http://www.maximumpc.com/10_best_unreal_engine_3_games#slide-9.*

17. **Unreal engine.** *What is Unreal engine 4.* [Online] [Citace: 16. Únor 2015.] *https://www.unrealengine.com/what-is-unreal-engine-4.*

18. **Cry engine.** *features* [Online] [Citace: 4. Únor 2015.] *http://cryengine.com/features.*

19. **CryEngine.** *en.wikipedia.org.* [Online] [Citace: 19. Únor 2015.] *http://en.wikipedia.org/wiki/CryEngine.*

20. **Unity - features.** *unity3d.* [Online] [Citace: 28. Leden 2015.] *http://unity3d.com/unity.*

21. **Unity 5.** *unity3d.com.* [Online] [Citace: 18. Březen 2015.] *http://unity3d.com/5.*

22. **Boo.** *boo.codehouse.org.* [Online] [Citace: 22. Únor 2015.] *http://boo.codehaus.org/.*

23. **Felicia, Patrick**. *Getting Started with Unity.* Birmingham : Packt publishing, 2013. ISBN: 978-1-84969-584-8.

24. **Unity Manual.** *Unity Documentation.* [Online] [Citace: 5. Únor 2015.] *http://docs.unity3d.com/Manual/index.html.*

25. **Pierce, Gregory.** *Unity iOS game development beginners guide.* London : Packt Publishing, 2012. ISBN-10: 1849690405.

26. **MonoDevelop.** *mono.wikia.com.* [Online] [Citace: 15. Únor 2015.] *http://mono.wikia.com/wiki/MonoDevelop.*

27. **Sharp, John.** *Microsoft Visual C# 2010: Krok za krokem.* Brno : Computer Press, a.s., 2010. ISBN: 978-80-251-3147-3.

28. **Script Reference.** *docs.unity3d.com.* [Online] [Citace: 25. Únor 2015.] *http://docs.unity3d.com/ru/current/ScriptReference/.*

29. **New features in C# 5.** *blogs.msdn.com.* [Online] MSDN, 26. Březen 2012. [Citace: 27. Únor 2015.] *http://blogs.msdn.com/b/mvpawardprogram/archive/2012/03/26/introduction-of-new-features-in-c-5-0.aspx.*

30. **System Collection**. *msdn.microsoft.com.* [Online] Microsoft. [Citace: 28. Únor 2015.] *https://msdn.microsoft.com/en-us/library/system.collections.aspx.*

31. **Josefnav**. *Arduino akcelerometr.* [Online] [Citace: 5. Březen 2015.] *http://www.josefnav.cz/Arduino_akcelerometr.html.*

32. **Gamepie.co**. *Creating a minimap using Unity 4.6 UI.* [Online] [Citace: 2. Březen 2015.] *http://gamepie.co/topic/20/creating-a-minimap-using-unity-4-6-ui/3.*

33. **Garcia, Angel.** *The Stabilizers Bars. projects.edy.es.* [Online] 3. Září 2012. [Citace: 22. Březen 2015.] *http://projects.edy.es/trac/edy_vehicle-physics/wiki/TheStabilizerBars.*

Seznam Obrázků

7. Seznam příloh

K práci je přiloženo DVD se zdrojovými kódy a zkompilovanou aplikací.