Středoškolská technika 2014

Setkání a prezentace prací středoškolských studentů na ČVUT

Hra vytvořená v platformě Unity 3D

Lukáš Cichý

Střední průmyslová škola na Proseku
Novoborská 2, 190 00 Praha 9

1 Obsah

2 Úvod

Cílem této dlouhodobé maturitní práce bylo vytvoření didaktické počítačové hry na platformě Unity 3D. Jako téma pro didaktickou hru si autor vybral simulátor autoškoly. Aby se jednalo o čistě autorskou práci, byly všechny použité prvky a 3D modely pro tento projekt vytvořeny v 3D grafickém programu. Pro tvorbu 3D modelů použil autor program od německé společnosti MAXON – Cinema 4D s registrovanou studentskou licencí. S programem a jeho základními funkcemi se autor seznámil při výuce na předmětu Počítačová grafika a multimédia.

3 Počítačová grafika

3.1 Základní informace

Počítačová grafika je jedno z podstatných odvětví informačních technologií. Počítačová grafika se stará o stylizaci například webových stránek apod. Počítačová grafika zejména zahrnuje 3D modelování, 2D kreslení, tvorbu vzhledů webových stránek, tvorbu animací a videí i s efekty. Pro práci s grafikou existuje široká nabídka grafických softwarů, většinou placených. Počítačová grafika lze dělit podle několika hledisek.

Příkladem je dělení z prostorového hlediska, kdy může být grafický objekt 3D (je možné ho vidět z více pohledů, než jen ze předu; pochopení jako prostorový, uchopitelný objekt), nebo 2D obrázek (objekt je možné vidět pouze tak, jak ho autor nakreslil; pochopení jako „placka“ s kterou se dá manipulovat pouze v jedné ose otáčení). Dalším důležitým dělením je dělení podle zobrazení obrázku, kdy existuje grafika rastrová (skládá se z jednotlivých bodů, při přiblížení ztrácí kvalitu), nebo grafika vektorová (obrázek se skládá z prvků jako křivky, čáry…, při přiblížení neztrácí kvalitu).

3.2 Dělení počítačové grafiky

Dělení podle dimenze – rozměru

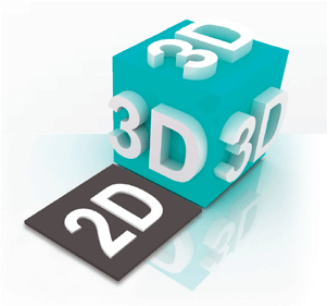
2D obrázky

3D objekty

(4D) – čtvrtý rozměr

(5D) – 3D zobrazení s dalšími efekty

Časové rozlišení – 2D nebo 3D animace



Obrázek 1 - Rozdíl mezi 2D a 3D grafikou (www.ptc.com)

Dělení podle stálosti zobrazení

Statické – jednoduchý obrázek, fotka…

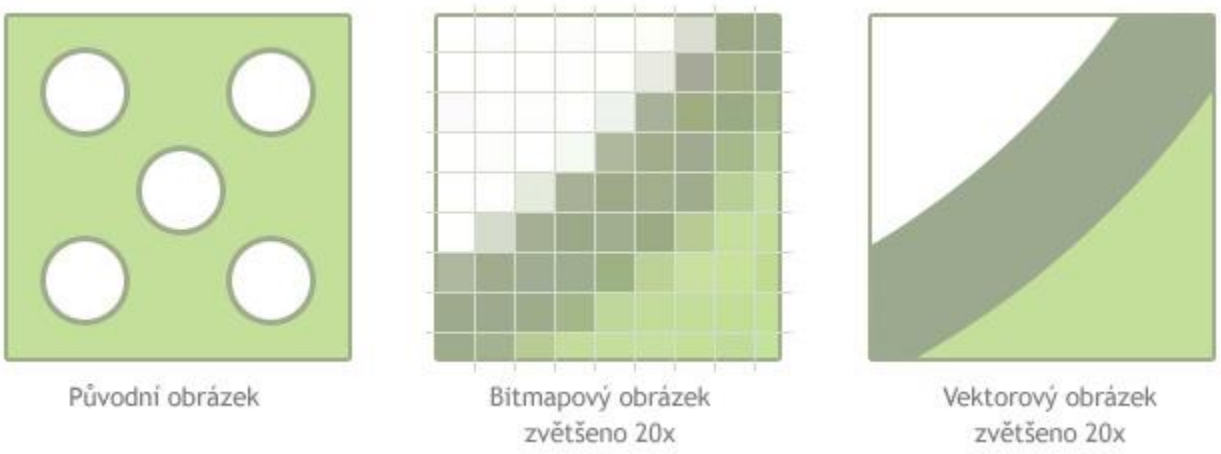
Dynamické – animace, video

Interaktivní – reagující na podnět uživatele (hra)

Dělení podle způsobu zobrazení

Rastrová grafika – mřížka jednotlivých bodů (pixelů), zobrazení závislé na rozlišení

Vektorová grafika – matematicky popsané polohy prvků jako čára, křivka…, nezávislé na rozlišení



Obrázek 2 - Rozdíl mezi rastrovou a vektorovou grafikou (www.optimal-marketing.cz)

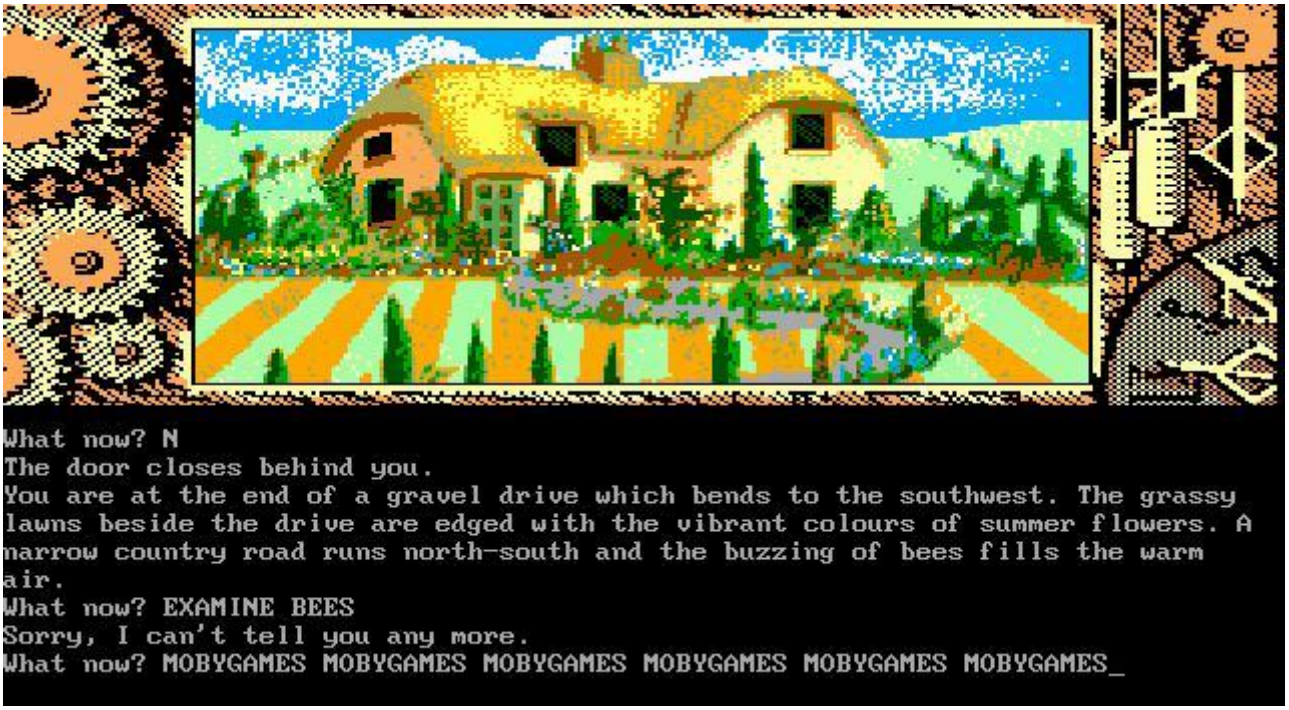
4 Herní enginy

4.1 Herní enginy úvodem

Jak už bylo v předchozí kapitole zmíněno, herní engine je program pro tvorbu her na desktopové operační systémy, mobilní operační systémy a herní konzole. Herní engine pomáhá vývojáři spojit herní prvky, konkrétně modely, materiály a dynamiku či fyziku obecně. Pokud se vývojář rozhodne použít herní engine, může si naprogramovat vlastní, nebo použít nějaký z běžně používaných. Lepší herní enginy mají uživatelské rozhraní, tudíž je práce s nimi snažší. Ve většině herních enginů existují různé licence, a to z důvodů využívání a výdělku. Herní engine a jeho jádro běžně poskytuje funkce renderování 2D i 3D grafiky. Mezi další poskytované funkce většinou patří také fyzický engine, detektor kolizí, zvuky, skripty a skriptování obecně, animace, umělou inteligenci, práci se sítí, streamování a v neposlední řadě struktura dat. Vývoj hry je běžně řešen ekonomicky tak, že je stejný engine použit na jinou hru, ale přizpůsobí se a pomůže tedy vytvořit hru pro více platforem. Engine tedy nabízí opětovné použití, které může být pozměněno a může přinést hru k životu. Na projektu se můžou účastnit umělci, designéři, skriptaři a další programátoři, kteří se hru, na které se podílejí, snaží udělat jedinečnou. Mezi běžné funkce herních enginů patří funkce nahrávání, zobrazování, animování modelů, kolize mezi objekty, fyzika, vstupy, grafické uživatelské rozhraní nebo dokonce i umělá inteligence. Tyto všechny komponenty tvoří herní engine. Téma hry, specifické modely a textury, myšlenka herního mechanismu a kolizí a způsob komunikace objektů s uživatelem je už složení skutečné hry.

4.2 Vznik herních enginů

První vývoj a programování her se datuje do dob osmibitových konzolí a osmibitových mikropočítačů. Tyto hry byly vyvíjeny v assembleru a až poté, co přišly šestnáctibitové a třiceti dvoubitové mikroprocesory, se přešlo na vyšší programovací jazyky. Ve vývoji byly také jazyky specializované přímo na vývoj her, které se používaly v textových aplikacích. Jinak tomu bylo u akčních her ze sedmdesátých a osmdesátých let. Tyto hry měly postup vývoje takový, že nebylo možné kód znovu použít na jinou hru. Bylo možné použít pouze podprogramy a ne celou logiku hry. To bylo hlavně kvůli nedostačující technice, tedy kvůli omezeným operačním pamětím. Z tohoto důvodu musela být logika hry přímo v kódu a ne v jiné vhodné datové struktuře jak je tomu dnes.



Obrázek 3 - Textová hra Price of Magik (www.old-games.com)

Výhodou tohoto typu vývoje jako přímého kódu je, že vývojáři mohli reagovat na rychlý rozvoj počítačového hardwaru, s kterým ani vývojáři nebyli přímo svázáni. Hlavní nevýhodou je vydání vysokého úsilí do vývoje hry. Tento problém ale v osmdesátých letech zajistil snahu o tvorbu univerzálních herních enginů, které na základě vstupních dat mohly být využity pro hry odlišné grafikou i příběhem, ale shodovaly se v systému zobrazení (2D, později i 3D) a zpracování. První herní enginy využívaly přímo vývojářské společnosti, kde enginy vznikaly. S rostoucím trhem a specializací se začaly rozvíjet další herní enginy, které se prodávaly vývojářům pro další práci. Pro běžné uživatele vznikl program construction kit, který poskytoval tvorbu her bez znalostí programování. Stačilo vytvoření mapy, designu a herních pravidel. Známým a populárním příkladem kitu byl Boulder Dash Construction Kit nebo Pinball Construction Set, který byl vytvořen pro tvorbu pinballových her.

Obrázek 4 - Boulder Dash Construction Kit (gamesdbase.com)

4.3 API a SDK

Tyto dva pojmy pochází rovněž z herního průmyslu a jsou spjaty s herními enginy v oblasti vývoje. API je rozhraní pro vývoj a programování aplikací. Pojem API je využíván v softwarovém inženýrství. Každý a cokoliv může API používat, pokud ví jak. SDK je přímo implementační nástroj, který vývojáři nebo programátorovi „povoluje“ tvorbu aplikace.

5 Unity3D

5.1 Unity3D Úvodem

Software Unity3D může mít více definic. Z jednoduchého hlediska se dá hovořit o počítačovém programu, který už z názvu napovídá, sjednocuje více prvků v jeden. Ze složitějšího hlediska se dá hovořit o herním enginu. Vývojář má zpočátku více vstupů – modely, textury, blokové schéma hry, ale hlavně myšlenku. Do programu se vkládají tyto jednotlivé prvky a pomocí programování v jazycích JavaScript, C# nebo méně používaném Boo se modelům přidá dynamika a vlastnosti. Cílem tohoto softwaru je tedy tvorba 2D či 3D jednoduchých i složitějších her, nebo interaktivních aplikací pro web nebo přímo operační systém.

5.2 Základní informace

Unity se používá pro tvorbu her a interaktivních aplikací pro webové pluginy, desktopové operační systémy, mobilní operační systémy a konzole. První verze vznikla v roce 2005 z verze, která byla podporována pouze systémem OS X. Unity je nyní multiplatformní aplikace, z tohoto důvodu mohou software používat uživatelé operačních systémů Windows, Linux nebo MAC OS. Program Unity má vývojové prostředí pro programátory zaměřené na programování v již zmíněném JavaScriptu, C# a Boo. Vývojové prostředí v Unity se nazývá Assembly - UnityScript Editor, a vývojáři nabízí barevně rozlišitelné části kódu, kompilaci a chybové hlášky což usnadňuje mnoho práce při vývoji a programování.

5.2.1 Multiplatformní program

Program Unity podporuje vývoj na většině platformách. Vývojáři v tomto programu mají plnou kontrolu nad vývojem pro desktopové stolní počítače, webové prohlížeče, mobilní zařízení a v poslední řadě herní konzole. Unity dále umožňuje optimalizaci textur jednotlivých 3D modelů a rozlišení tak, aby byly plně podporovány právě většinou běžných či herních zařízení.

5.3 Podporované platformy:

Desktopové operační systémy

Windows

MAC

Linux

Mobilní operační systémy

Android

iOS

Windows Phone

BlackBerry 10

Herní konzole

Xbox One

Xbox 360

Playstation 3

Wii, U

Playstation Vita

5.4 Funkce Unity

5.4.1 Fyzika

Unity má vestavěnou podporu PhysX od společnosti NVIDIA. PhysX je fyzický herní engine, který využívá grafické karty k výpočtu fyziky. Fyziku běžně vypočítává procesor, který je v případě PhysX od výpočtů ušetřen. To zajišťuje nejen lepší grafiku, ale také plynulejší chod aplikace. Další je podpora simulací, animací, materiálů a kolizních objektů v reálném čase. Od verze 4.3 byla zahrnuta podpora Box2D fyzického enginu, který je vytvořen pro 2D aplikace.

5.4.2 Skriptování

Skriptování v Unity je založeno na projektu Mono. Mono je projekt firmy Novell, jehož cílem je tvorba nástrojů, které jsou kompatibilní s prostředím .NET, to znamená, že vývoj a provoz je podporován na více operačních systémech, a to: Windows, Linux, OS X, Unix a Solaris. Vývojáři zde mohou použít UnityScript – vlastní jazyk založený na JavaScriptu. To znamená, že program má své přednaprorgamované skripty, které jsou slučitelné s běžnými funkcemi JavaScriptu. Dále se zde programuje v jazycích C#, nebo Boo. Boo má syntaxi skriptů založenou na syntaxi programovacího jazyku Python. Od verze 3.0 je zabudován MonoDevelop. Ten napomáhá a urychluje vývojáři čas našeptáváním skriptů při psaní.

5.4.3 Renderování

Grafický engine využívá engine Direct3D, OpenGL, OpelGL ES a vlastní API pro herní konzole. Direct3D je podporován firmou Microsoft, a je používán na platformách Windows a Xbox (Xbox je vyvíjen firmou Microsoft). OpenGL je taktéž podporován platformou Windows, ale dále také systémy MAC a Linux. OpelGL ES je podporován mobilními operačními systémy Android a iOS. Grafický engine podporuje bump mapping (iluze textury, která vypadá jako 3D objekt), reflection mapping (výpočet světla, odrazu), parallax mapping (vylepšený bump), okolní absorbování, shadow mapping (výpočet stínu objektu).

Aby se ve hře mohli použít 3D modely, musejí se někde vytvořit. Unity podporuje modely a formáty ze softwarů 3ds Max, Autodesk Maya, Softimage, Blender, modo, ZBrush, Maxon Cinema 4D, Cheetah 3D, Adobe Photoshop, Adobe Fireworks. Modely či jiné prvky vytvořené v těchto grafických programech mohou být importovány do enginu Unity a dále se s nimi může pracovat přes uživatelské grafické rozhraní

Unity. Toto jsou ale pouze programy, které lze použít pro tvorbu modelů či jiných grafických objektů. Unity podporuje formáty modelů .fbx, .dae, .3DS, .dxf, .obj. Nejvíce doporučované jsou .fbx a .obj z důvodu velikosti a bezproblémovosti.

5.4.4 Sledování prvků

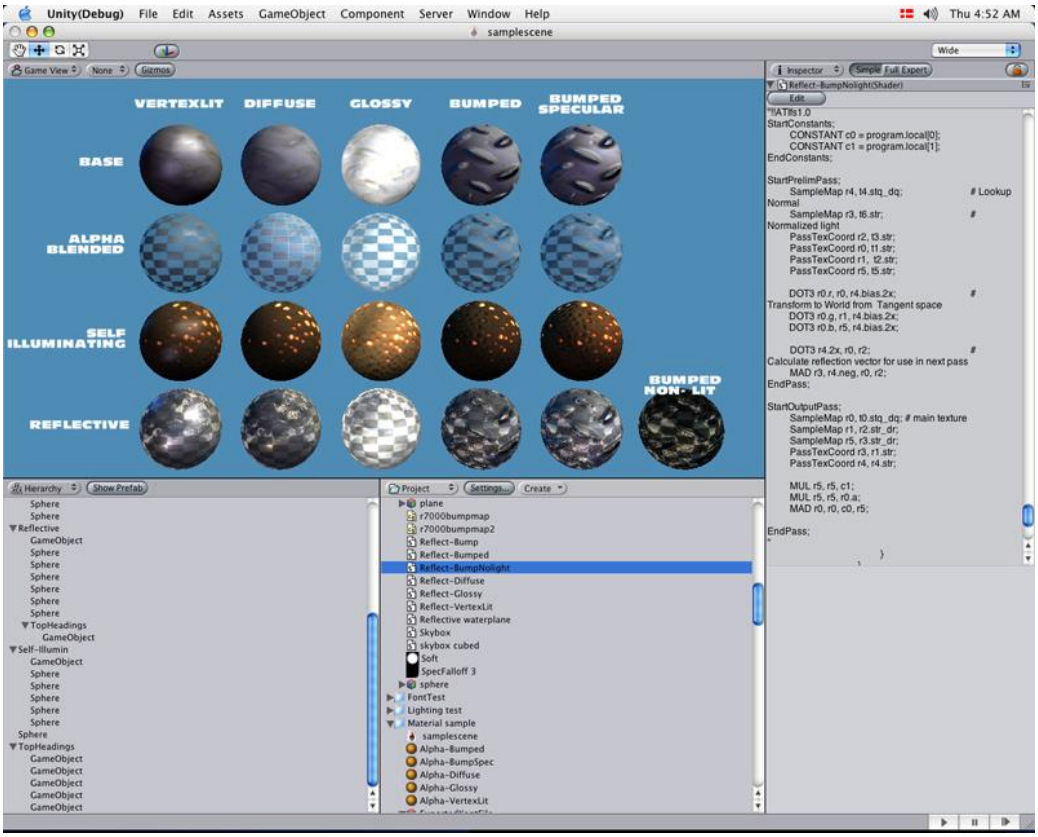
Unity obsahuje Unity Asset Server – řešení pro správu herních prvků a skriptů pro vývojáře. Tato správa využívá PostgreSQL – systém postavený na knihovně FMOD s možností přehrát komprimovaný zvuk. K přehrávání videa využívá kodeku Theora. Dále je zde engine pro tvorbu terénu a zázemí obecně s vestavěným globálním světlem a mapováním stínů. K multiplayerové hře se využívá RakNet.

5.5 Verze Unity

První verze Unity byla představena v roce 2005 na konferenci vývojářů firmy Apple. Tato první verze programu byla vyvinuta pro vývoj projektů na platformě MAC. Verze byla natolik úspěšná, že ji vývojáři začali vyvíjet i pro ostatní platformy. V září 2010 byla představena verze Unity 3, která byla zaměřena na více nástrojů pro možnosti dobře vybavených studií. Tento krok umožnil získat zájem více vývojářů a menších týmů s cenově dostupným balíčkem. Unity 4 bylo představeno v roce 2012, a podporuje animace Mecanim a DirectX 11.

5.5.1 Unity 1.0

První verze Unity, webový přehrávač je v sandboxu, což znamená, že některé funkce ve webovém přehrávači jsou vypnuté. V této verzi jsou přidány dokumenty s návody, rozhraní pro kódování, dále sada shaderů s hrubostí, rozptýlení světla, lesklost. V nástroji na programování jsou barevně zvýrazněny chyby a varování.



Obrázek 5 - Unity verze 1 (forum.unity3d.com)

5.5.2 Unity 2.0

Od předchozí verze obsahuje tato verze 2.0 terrain engine, což je nástroj pro tvorbu terénu. Terén se dá nyní tvořit jednoduše v editoru podobně jako při malování. Uživatel si tedy může sám namalovat vzhled terénu, tvořit kopce, údolí. Ve verzi pro se objevila funkce video playback. To umožňuje doher přidat video a zvuk, které mohou být přehrávány ve 2D či 3D. Dalším rozšířením je možnost tvorby multiplayerové hry. Síťová vrstva je založena na UDP protokolu a podporuje NAT. Síťová komunikace tu funguje dvěmi způsoby, a to synchronizací stavů a voláním procedur na dálku. Dalším vylepšením je tvorba dynamických stínů v reálném čase. Tato funkce je dostupná také pouze v pro verzi. Ve verzi 2.0 přišlo usnadnění pro tvorbu uživatelských rozhraní. Další novinkou ve verzi 2.0 je Unity Asset Server, což zajišťuje úplnou kontrolu nad skripty a herními prvky. Všechny záznamy o nastavení jsou ukládány. Unity Asset Server je funkce rovněž pouze pro verzi pro.

5.5.3 Unity 3.0

Mezi jedno z nejhlavnějších vylepšení oproti předchozí verzi patří podpora mobilního operačního systému Android. Další vylepšení se týkají grafiky. Ve verzi 3.0 jsou nové efekty obrázků, mezi ně patří paprsky slunce, odrazy objektivu, vinětování, upravování barev pomocí křivky, stínování, upravování kontrastu nebo rybí oko. Další vylepšení jsou předvytvořená nebe s alfa kanály a různými možnostmi efektů. Do editoru byly přidány skripty, které usnadňují práci. Další velkou novinkou je prvek CharacterMotor. CharacterMotor je skript, který je připraven pro použití se skriptem CharacterControllers. Už název napovídá, že skript CharacterMotor rozhýbe nějaký objekt a pomocí skriptu CharacterControllers je možné tento objekt ovládat.

V editoru jsou v této verzi navíc funkce jako například procedurální tvorba stromů. Další rozšíření přišlo s integrovaným mapováním světla. V grafické části je novinkou grafické statické absorbování využívající engine Umbra sPVS. Další novinkou je například geometrické dávkování, převzaté z verze iPhone 1.7 pro všechny platformy. V možnosti nastavení hráče jsou nové možnosti nastavení cest renderu. Tyto cesty jsou odložení osvětlení, Vertex shader lit, Forward (oproti unity 2 mnoho změn). Dalšího zlepšení se dočkalo také stínování, které využívá stínovacích map enginu Direct3D, což je rychlejší a používá méně paměti. Do možností terénu je přidán ovládací prvek na hustotu rozmisťování prvků. Dalším vylepšením tvoření terénu je zlepšení výkonu při barvení textur a detailních objektů.

Nové jsou možnosti nastavení zvuku. Ve verzi Unity 3.0 jsou zvukové filtry na nastavení zkreslení, přechodu nebo echo. Zlepšení ve fyzice je vylepšeno na verzi PhysX 2.8.3. Nová je simulace látky, ignorování kolizí na základě nastavení vrstev. Ve skriptování je nové rozšíření Mono Develop. Nový je také kompilátor v editoru skriptů.

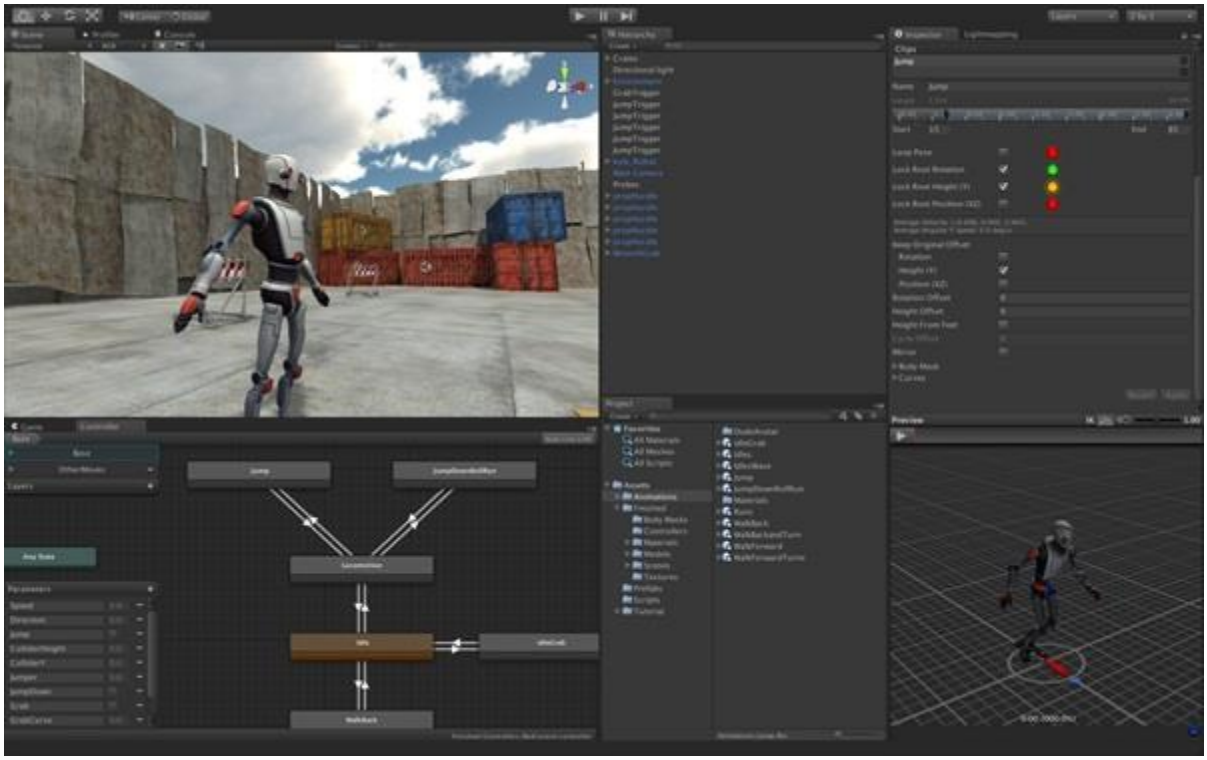
5.5.4 Unity 4.0

Unity 4.0 je podle vývojářů největším pokrokem ve vývoji tohoto enginu. Ve verzi 4.0 jsou opět nové funkce, které v předchozích verzích nebyly. Nový je například Mecanim, což je nový systém animování postavy nebo objektu. Další novinkou jsou stíny v reálném čase pro všechny platformy. Nový je také DirectX 11 rendering a generátor částic, který má funkci pro kolizi se světlem. Do rodiny platforem přibyla podpora platforem Linux a Adobe flash.

Mezi nové grafické funkce patří normálová mapa a vlastní shader při tvorbě terénu. Jsou podporovány také 3D textury, není možné je importovat. Tyto 3D textury je nutné vytvořit skriptem (Texture3D) a poté je použít v shaderech. V této verzi se rozšířily stíny i do mobilních zařízení, ale pouze ze směrového světla. Dalším novým grafickým prvkem jsou fonty, nové dynamické renderování na všech platformách s identickými výsledky. Déle přibyly efekty obrazu jako automatická oprava barvy, detekce rohů, šum, zkreslení.

Novinkou v uživatelském rozhraní je zobrazení miniatury náhledu objektu v seznamu objektů. To pomůže vývojáři s rychlejší možností práce než vyhledávat v seznamu ten správný objekt. Nové rozdělení hierarchie pro lepší orientaci je ve verzi 4.0 také novinkou. Nově je udělaný také favorites list, tedy senam oblíbených objektů pro snadný přístup k nejčastěji používaným položkám. Uživatel může toto okno prohlížení souborů a projektu přepnout i do starého stylu.

Bylo přepracováno i jádro uživatelského rozhraní. Přepracováním se myslí vyčištění, přepsání a optimalizování. Výsledkem je menší nárok na paměť. Je několikrát rychlejší než v předchozích verzích a přitom se nejedná o novou verzi, ale pouze přepracovanou již existující verzi. Nově je v uživatelském rozhraní také konzole ukazující chyby v programování.



Obrázek 6 - Unity verze 4 (indiegames.com)

5.6 Unity 5.0

Unity 5.0 bude k dispozici pravděpodobně až v roce 2015. Vývojáři si ale už nyní mohou vytvořit předobjednávky. V této verzi by se měl aktualizovat fyzikální engine PhysX na nejnovější verzi se zpětnou kompatibilitou s webovým přehrávačem. Dále aktualizace Mono editoru a nový 64 bitový editor.

5.7 Dostupné licence Unity

Licence herního enginu unity jsou rozděleny na dvě. Jednou verzí je Unity a druhou verzí Unity Pro. V tabulkách níže jsou uvedeny pouze ty funkce, ve kterých se tyto dva druhy licencí liší.

5.7.1 Hlavní funkce

Funkce	Unity	Unity Pro
Podpora LOD modelů	Ne	Ano
Zvukový filtr	Ne	Ano
Playback videa a streamování	Ne	Ano
Plně rozvinuté streamování s objekty	Ne	Ano
Používání firmou s obratem vyšším než 100 000 $	Ne	Ano

Tabulka 1 - Rozdíly licencí – funkce

5.7.2 Animace

Funkce	Unity	Unity Pro
Mecanim – IK Rigy	Ne	Ano
Mecanim – Vrstvy a křivky	Ne	Ano

Tabulka 2 - Rozdíly licencí – animace

5.7.3 Rozvíjení

Funkce	Unity	Unity Pro
Vlastní obrázek načítání	Ne	Ano

Tabulka 3 - Rozdíly licencí - rozvíjení

5.7.4 Grafika

Funkce	Unity	Unity Pro
Podpora 3D textur	Ne	Ano
Stínování v reálném čase	Ne	Ano
HDR, mapování tónů	Ne	Ano
Světelné sondy	Ne	Ano
Mapování světla s globální iluminací a světelnou oblastí	Ne	Ano
Statické dávkování	Ne	Ano
Efekt renderování na texturu	Ne	Ano
Grafické efekty na celou obrazovku	Ne	Ano
Absorbování výběru	Ne	Ano
Odložený rendering	Ne	Ano
Přístup k bufferování podle šablony	Ne	Ano
Skinování GPU	Ne	Ano

Tabulka 4 - Rozdíly licencí – grafika

5.7.5 Kód

Funkce	Unity	Unity Pro
Navmesh – dynamické překážky a priority	Ne	Ano
Přirozená podpora kódových pluginů	Ne	Ano

Tabulka 5 - Rozdíly licencí – kód

5.7.6 Editor

Funkce	Unity	Unity Pro
Profilování GPU	Ne	Ano
Přístup skriptu k objektům	Ne	Ano
Tmavý vzhled	Ne	Ano

Tabulka 6 - Rozdíly licencí – editor

5.8 Komunita

Komunita kolem softwaru Unity má okolo 2 milionů registrovaných uživatelů (vývojářů k datu 11. 1. 2014). Pro vývojáře bylo založeno Asset store. Asset store je obchod s modely, materiály, texturami, částicovými systémy, hudbou, zvuky, návody, projekty, skripty a různými balíčky s uvedenými položkami. Tyto objekty jsou od vývojářů pro vývojáře.

5.9 Skriptování v Unity

Skriptování v herním enginu je velice podstatným krokem vývoje počítačové hry. Bez skriptování by hra neměla žádnou logiku, ovládání ani fyziku. Jelikož se vývoj hry zabývá už od prvotní fáze objekty, interaktivními prvky, tlačítky, animacemi a podobnými efekty, jsou používány objektově orientované skriptovací jazyky. Nejpoužívanějším jazykem pro účel vývoje hry je objektově orientovaný skriptovací jazyk JavaScript. Druhým nejrozšířenějším nástrojem pro skriptování hry je objektově orientovaný programovací jazyk C#. Třetí možností jak na hru přenést logiku je objektovým jazykem Boo. Nejvíce používané jazyky jsou ale JavaScript a C#.

5.9.1 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk používaný zejména pro interaktivní účely. Interaktivními účely se myslí tlačítka, pole pro webové stránky, kde může být JavaScript zařazen přímo do HTML kódu. Skript začne běžet až po stažení uživatelem právě na jeho straně na rozdíl od php, které běží na straně serveru ještě před stáhnutím. Java je označením pouze z obchodních důvodů, s jazyk JavaScript spojuje s Javou pouze podobná syntaxe skriptů. V roce 1997 byl JavaScript normalizován společností ECMA, tato normalizovaná verze se nazývá ECMAScript, k kterého výchází například také ActionScript, který je možné použít při skriptování ve flashi. V roce 1998 byl standardizován normou ISO.

Ukázková funkce ve skriptovacím jazyku JavaScript

<script type="text/javascript">

document.write ("Hello World!")

</ script>

Skript na obrazovku vytiskne heslo „Hello World!“.

5.9.2 3.7.2 C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk založen firmou Microsoft na jazycích C++ a Java. Programovací jazyk C# je možné využít k tvorbě webových služeb, formulářových aplikací, k tvorbě databázových programů nebo webových aplikací. Stejně jako JavaScript je jazyk C# standardizován normami EMCA a ISO.

Ukázková funkce v programovacím jazyce C#

```
public class Hello1
{
   public static void Main()
   {
      System.Console.WriteLine("Hello World!");
   }
}
```

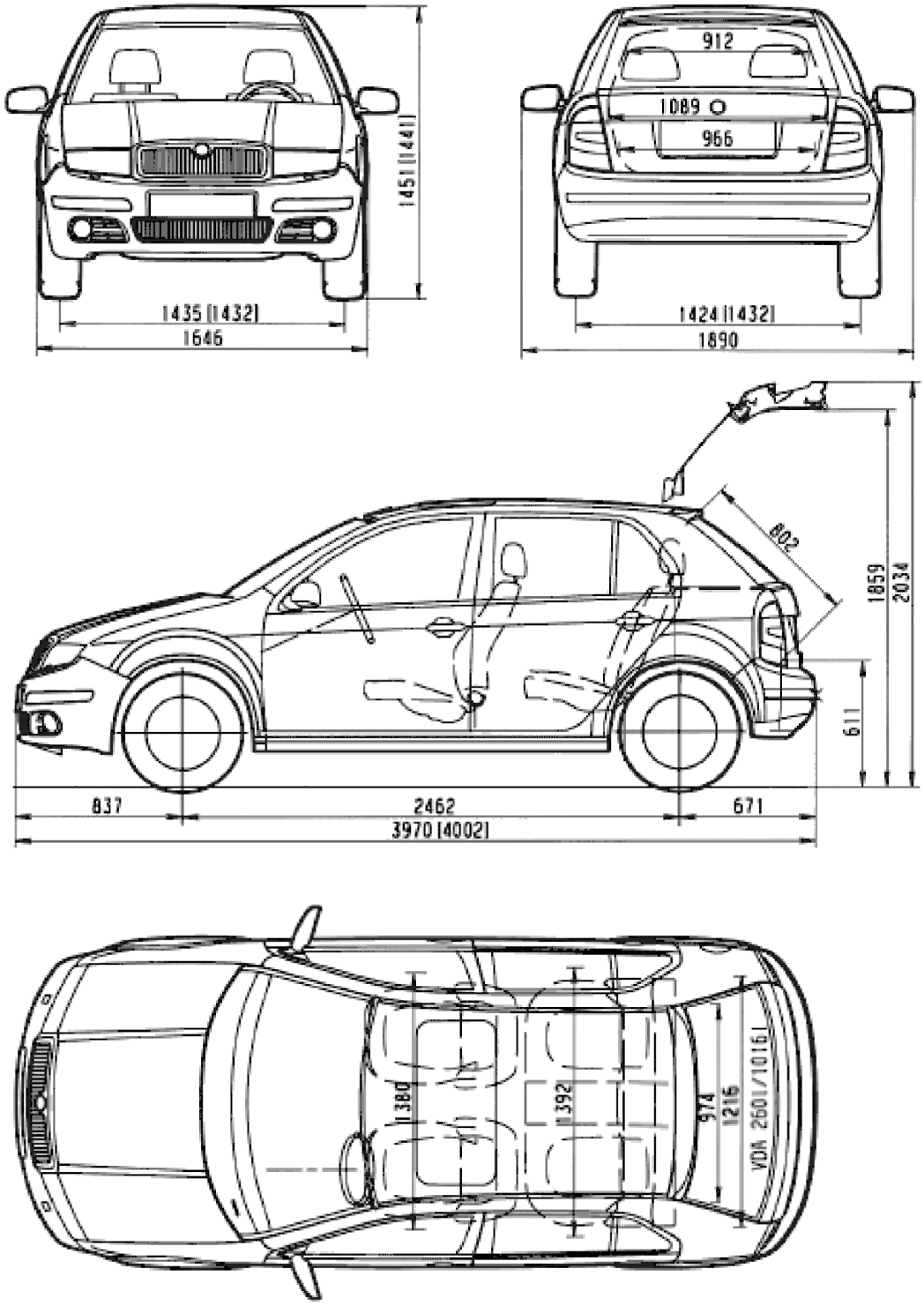
Skript na obrazovku vytiskne stejně jako u předchozí ukázky heslo „Hello World!“.

6 Technická dokumentace

6.1 Model automobilu

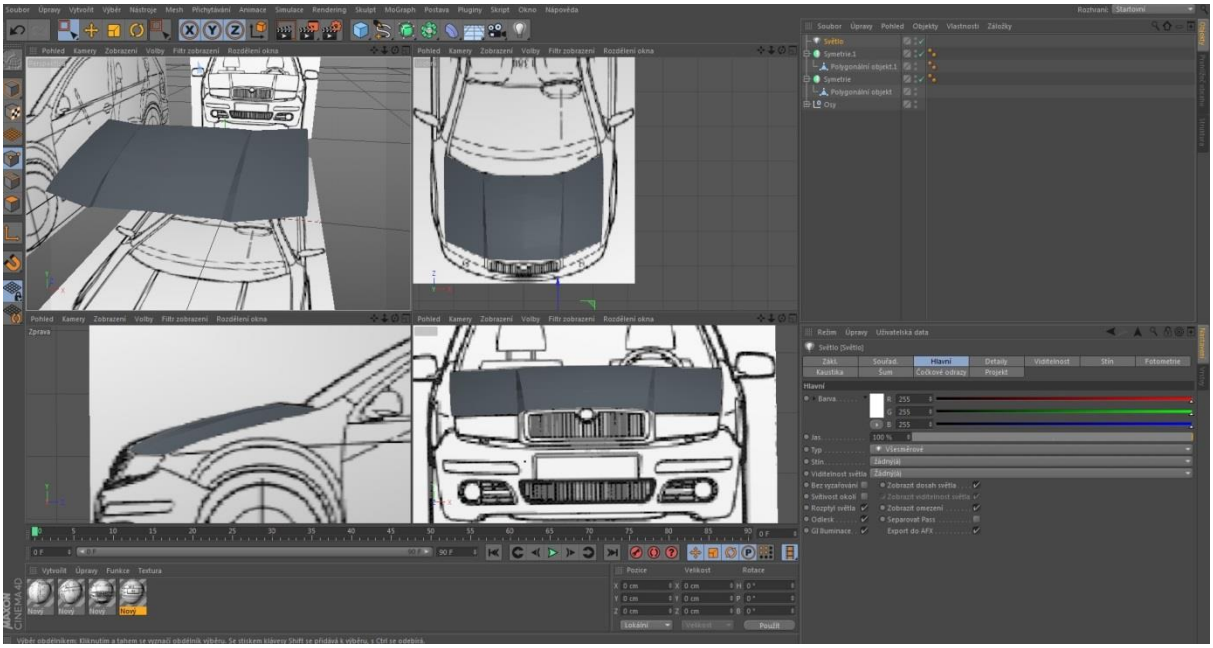
Tématem pro vývoj počítačové didaktické hry bylo zvoleno téma autoškola. Počáteční fází vývoje hry byla tvorba návrhů a blokových schémat jak by hra měla vypadat, a co by měla obsahovat. Po fázi navrhování byl vývoj soustředěn na realizaci dle návrhů a blokových schémat. Prvním objektem realizace byla tvorba 3D modelu automobilu, jako hlavního herního prvku, který je ve hře používán. Model 3D automobilu byl na vývoj časově náročný, jelikož se jednalo o modelování podle blueprintu.

Blueprint je předloha pochopitelná jako jednoduchý obrázek s hlavními obrysy auta. K modelování automobilu podle blueprintu jsou zapotřebí celkem 4 předlohy, a to z boku, shora, zepředu a zezadu. Princip modelování spočívá v tvorbě polygonů, a následném pozicování jejich bodů přesně podle předlohy. Při modelování pomocí blueprintu je třeba dbát na přesnost už v počátečních fázích, to znamená, že blueprinty je nutné mít přesně rozmístěné tak, aby na sebe navazovaly ve všech pohledech modelování. Nepřesnost v tomto kroku by byla zdrojem dalších, důležitých nepřesností v modelování.



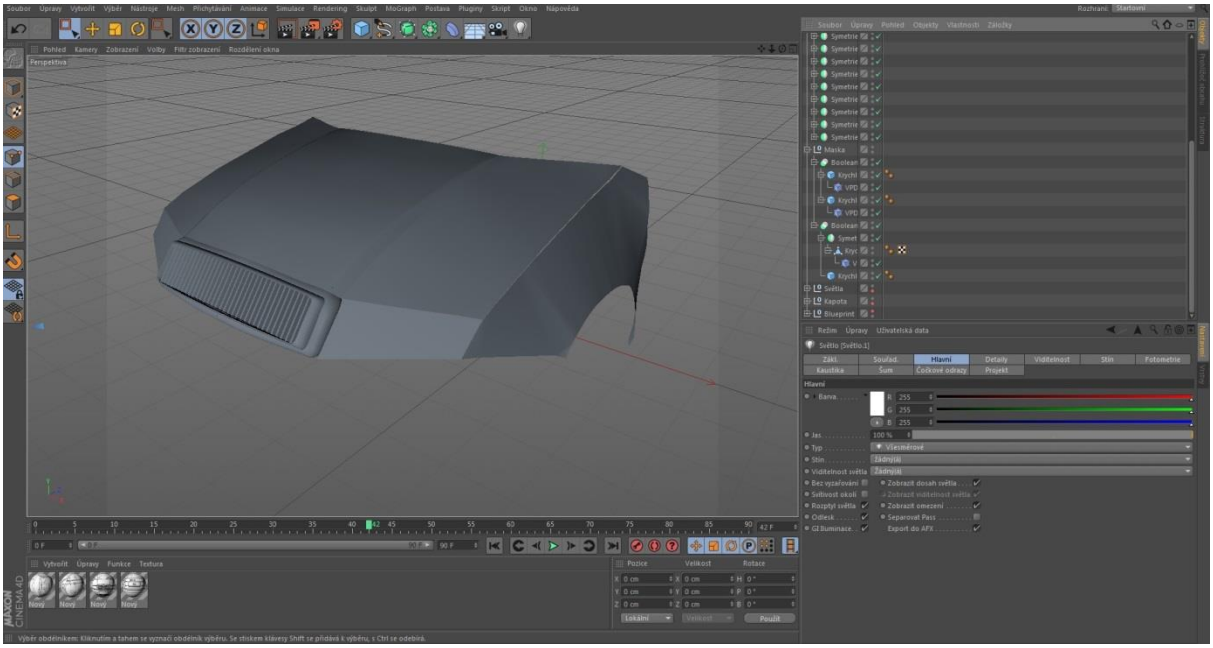
Obrázek 7 - předloha – blueprint (carblueprints.info)

Výše uvedený blueprint bylo nutné rozdělit na jednotlivé pohledy, ty byly exportovány jako obrázkové soubory, a dále použity jako materiály pro pomocné roviny při modelování.



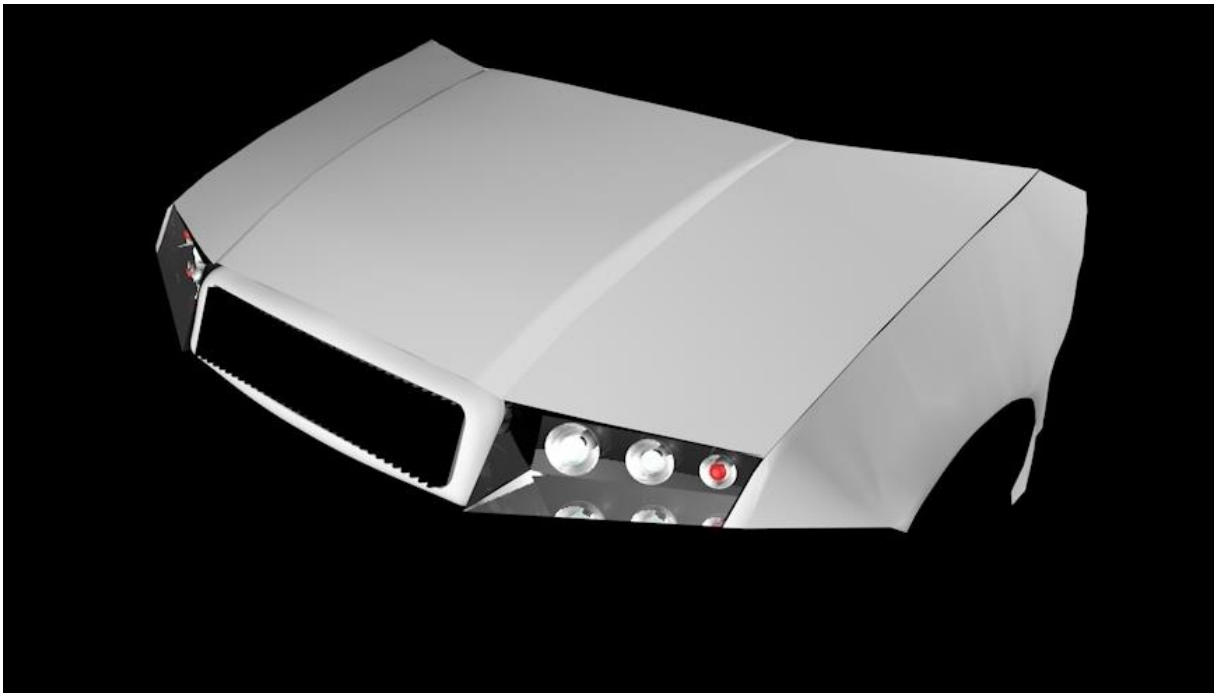
Obrázek 8 - tvorba kapoty podle blueprintu

Na snímku obrazovky je uživatelské rozhraní 3D grafického programu Cinema4D, v kterém byly modely pro vývoj hry vytvářeny. V tomto uživatelském rozhraní se nachází 4 pohledy na 3D objekt. Levý horní pohled je 3D pohled, pravý horní pohled je pohled shora, levý spodní pohled je pohled z boku a pravý spodní pohled je pohled zepředu. Modelování tedy spočívá v posouvání bodů polygonů (šedá plochá část), pokud uživatel pohne s bodem například s bodem v pohledu ze předu, je tato změna pozice vidět ve všech ostatních pohledech. Takto se pozicují všechny body polygonu, dokud nesedí přesně na blueprintu ve všech pohledech.



Obrázek 9 - maska, kapota, světla, blatníky

Na tomto snímku obrazovky je ukázka již hotové kapoty, masky, světel a předních blatníků. Vždy byla tvořena pouze polovina součásti, druhá polovina byla symetricky zrcadlena pomocí nástroje symetrie. Na součásti byla aplikována funkce Phong vyhlazení. Tato funkce zajišťuje hladkost vymodelovaného objektu.



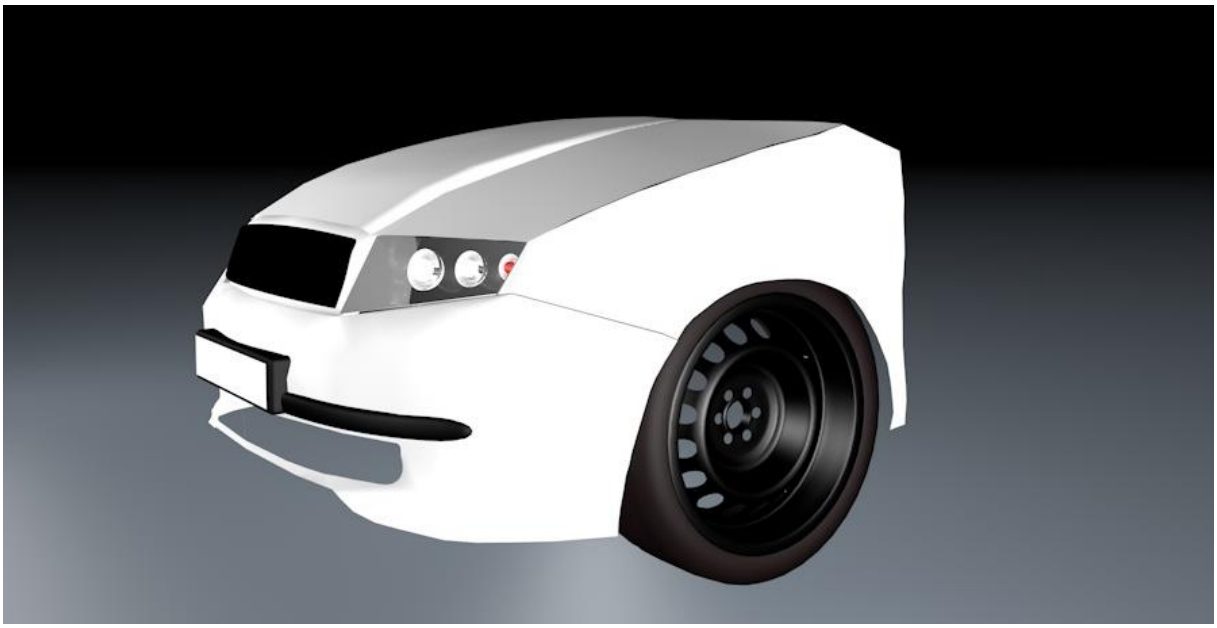
Obrázek 10 - maska, kapota, světla, blatníky

Na tomto snímku je ukázka detailně propracovaných světel, které byly vytvořeny válci, polokoulemi, koulemi z primitiv. Dále jsou na všech objektech aplikovány materiály.

Obrázek 11 - přední nárazník

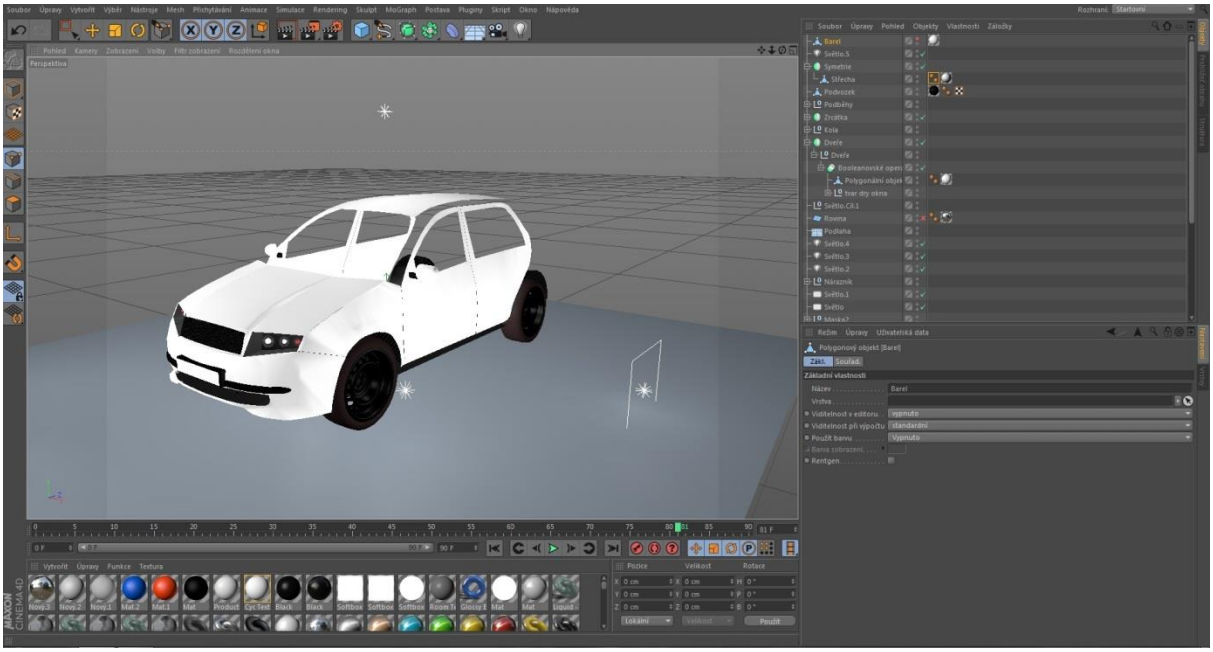
Na tomto snímku obrazovky je již hotový přední nárazník včetně lišt a podložkou pod státní poznávací značku. Nárazník byl tvořen stejně jako předchozí objekty polygonem, a následnou symetrií na zrcadlovou stranu. Bylo zde použito rovněž vyhlazení Phong a materiál bílé barvy. Spodní výřez nárazníku byl vytvořen pomocí booleanovské operace, která odečítá navzájem protínající objekty. Pro odečtení byl vytvořen kvádr se zakulacenými hranami. Lišty byly vytvořeny pomocí zaobleného, ohnutého kvádru podle již vytvořeného nárazníku, lišta byla nejprve vytvořena jedna, a poté přenesena symetrií na zrcadlovou druhou stranu. Podložka poznávací značky byla vytvořena z kvádru, na kterém je další součást, rovina, která představuje již vlastní poznávací značku.

Obrázek 12 - model ocelových kol



Obrázek 13 - model ocelových kol

Dalším krokem při modelování byla kola. Model kol byl inspirován běžně používanými ocelovými koly na reálných vozidlech. Kola byly vytvořeny nakreslením profilu poloviny kola, To znamená jako by se kolo rozřízlo napůl přes širší kruhovou část a z té části byla odříznuta další polovina. Tento profil byl pomocí funkce rotace rotován o 360 stupňů tak, že vzniklo kolo bez otvorů. Otvory byly vytvořeny z válců, které byly shromážděny do pole. Toto pole bylo dále odečteno od tvaru kola přes booleanovskou operaci tak, že vznikly otvory. Tyto funkce byly na kolo použity celkem 3 krát, a to pro středový otvor a otvory na šrouby. Pneumatika je tvořena primitivem anuloid. Toto primitivum bylo přizpůsobeno tak, aby bylo podstatně hranatější než po vložení do modelu. Anuloid byl dále upraven pro rozměry kola. Celá kola byla symetricky zrcadlena, a duplikována i pro zadní nápravu.



Obrázek 14 - dveře, střecha

Na tomto snímku obrazovky jsou již hotové dveře, střecha a zrcátka s podvozkem. Zrcátka byla tvořena dvěmi zaoblenými krychlemi, jedna jako držák zrcátka, druhá jako vlastní zrcátko. Dalším použitým primitivem byl podvozek, který je vytvarovaný z roviny. Dveře a střecha byly tvořeny polygony podle blueprintu přesně tak, jak tomu bylo v předchozích krokách.

Obrázek 15 - model vozidla

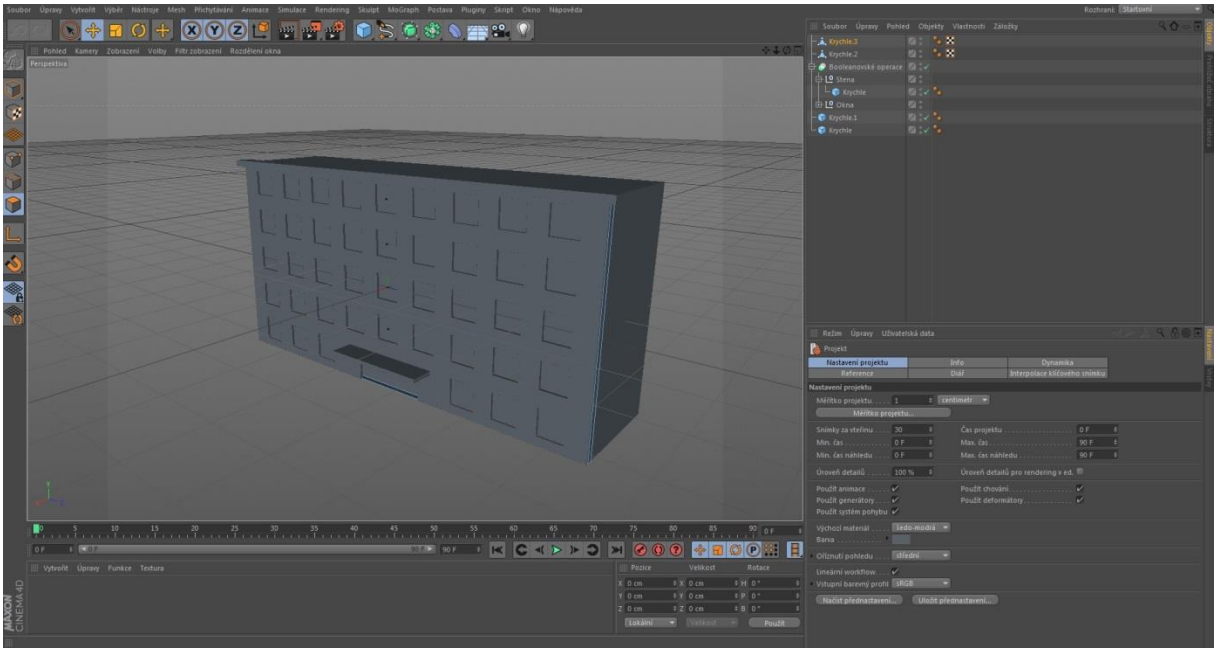
Na tomto snímku obrazovky je téměř hotový 3D model pro hru. Na skla byla použita primitiva rovina s černým lesklým materiálem. Zadní blatník, nárazník a páté dveře byly tvořeny polygony a následným symetrickým zrcadlením na druhou stranu. Zadní světla jsou z upravené zaoblené krychle.

Obrázek 16 - hotový model vozidla autoškoly

Na tomto snímku je již hotový model automobilu i s použitými materiály a označením autoškoly na střeše. Označení je tvořeno z primitiva kapsle o třech segmentech po obvodu. Na tento díl byl aplikován nápis autoškola vytvořený v grafickém programu Adobe Photoshop CS6.

6.2 Okolní prostředí

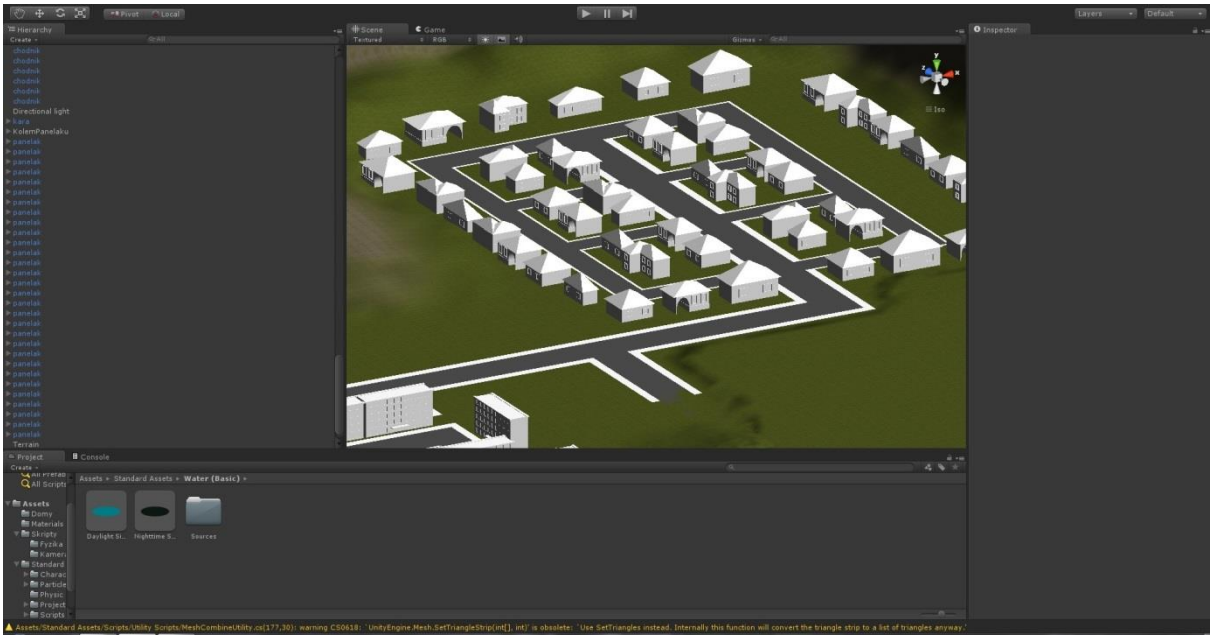
Pro okolní prostředí bylo vytvořeno dohromady 6 druhů domů. Jedná se o čtyři druhy modelů rodinných domů a dva druhy panelových domů. Domy byly tvořeny z primitiv (krychle) s jednoduchými okny.



Obrázek 17 - model panelového domu

Terén byl vytvořen v editoru terénu přímo v enginu Unity. Pro tvarování terénu se používá štětec s příslušnými nástroji (zvýšení, snížení, kreslení textury…). Mapa hry byla rozdělena na dvě části, a to městskou a vesnickou. Zázemí autoškoly se nachází mezi těmito částmi, přibližně uprostřed mapy.

Obrázek 18 - městská část mapy



Obrázek 19 - vesnická část mapy

Obrázek 20 - kompletní mapa hry

Na tomto snímku obrazovky je kompletní mapa simulátoru autoškoly. V horní části se nachází městská část s panelovým sídlištěm, ve spodní části je vesnická část s pískovým koupalištěm a parkovištěm. Uprostřed se nachází budova autoškoly.

6.3 Pohled hráče

Pohled hráče je statický, kamera je naprogramována tak, že v závislosti na rychlosti se vzdaluje od jedoucího vozidla.Hra se může pozastavit stisknutím tlačítka P (navrácení do hry také P) nebo přesměrování do menu (tlačítko ESC).

Obrázek 21 - pohled hráče ve hře

6.4 Optimalizace hry

Aby hra neměla příliš velkou odezvu a nepozastavovala se, byla použita mlha. Má hned více výhod, tou nejhlavnější je, že díky mlze nemusí hra vykreslovat objekty které jsou vzdáleny, tyto objekty se objeví až když je k nim hráč blíže. Další výhodou je estetika, mlha dodává lepší a reálnější vzhled. Místo stínů reálného času zde byly použity zapečené stíny, což zajistí nenutnost vykreslování pod různými úhly pohledu. Tím pádem je ušetřena paměť a stíny jsou statické.

Obrázek 22 - ukázka mlhy a vykreslování

6.5 Skriptování

Skripty se skládají dohromady z patnácti zdrojových kódů naskriptovaných objektově orientovaným skriptovacím jazykem JavaScript. Hlavní skript *Auto.js* je skript pro fyziku a ovládání vozidla. Vozidlo se v unity rozpohybuje vložením kolizních objektů na kola, přiřazením proměnných a použitím funkce pro vytvoření motoru. Aby vozidlo bylo stabilní, jsou použity funkce pro vyvážení. Vozidlo se vyvažuje posouváním těžiště v objektu v závislosti na nastavené hmotnosti, která je v případě projektu nastavena na 1 500 kg.

```
#pragma strict
var LPkolo : WheelCollider;
var PPkolo : WheelCollider;
var LZkolo : WheelCollider;
var PZkolo : WheelCollider;
var rychlost : float;
var maximalka : float = 100;
var otacky : float = 10;
function Start (){
rigidbody.centerOfMass.y = 0.5;

}
function FixedUpdate (){

LZkolo.motorTorque = otacky * Input.GetAxis("Vertical");
PZkolo.motorTorque = otacky * Input.GetAxis("Vertical");
LPkolo.steerAngle = 25 * Input.GetAxis("Horizontal");
PPkolo.steerAngle = 25 * Input.GetAxis("Horizontal");
}


function Update ()
{
rychlost = rigidbody.velocity.magnitude;
if (rychlost > 20){
LPkolo.steerAngle = 2 * Input.GetAxis("Horizontal");
PPkolo.steerAngle = 2 * Input.GetAxis("Horizontal");
}
if(Input.GetKeyDown("p"))
{
if(Time.timeScale == 1.0)
{
Time.timeScale = 0.000001;
}

else{
Time.timeScale = 1.0;
}
}
if(Input.GetKeyDown("escape"))
{
Application.LoadLevel("Rozhrani");
}
else{
}
}
```

Obrázek 23 - ukázka skriptu ovládání vozidla a fyziky

Nejpoužívanější funkcí v projektu je *OnTriggerEnter*. Tato sleduje, zda se nachází hráč v definované zóně. Tato funkce funguje tak, že se přiřadí kolizní objekt, který je neviditelný a jde projet (funkce *is Trigger*). Pokud hráč projede, nebo se zastaví v této zóně, má výstup funkce logickou 1. Ve funkci je dále skript co se stane v případě logické 1 nebo logické 0. Logická 0 je permanentně nastavená tak, že při běžné jízdě hráči nic nezobrazuje. Pokud hráč do zóny vjede, bude mu vytisknuta hláška na displej v závislosti, u které značky právě stojí. Touto funkcí je realizováno také uživatelské rozhraní. V ukázce skriptu je nejprve definování proměnných. Proměnná *najezd* je číselná (logická 0 nebo 1), k proměnné *collider* se přiřazuje kolizní objekt, ke kterému se skript vztahuje. Dále je skript pro vytisknutí textu „Dejte přednost vozidlům na hlavní silnici!“. Tento skript je v podmínce if, pro kterou jsou další podmínky v dolní části. Zde se zjišťuje funkcí *OnTriggerEnter* a *OnTriggerExit*, zda se hráč nachází v zóně. Pokud ano, výstup je *true* a hláška se vytiskne. Pokud ne, výstup je *false* a text se nezobrazuje (výchozí stav).

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Dejte přednost vozidlům\nna hlavní silnici!</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Obrázek 24 - ukázka funkce

6.6 Herní scény

Celý projekt se skládá z pěti scén (scéna = základní jednotka projektu, projekt se skládá ze scén). Hned po zapnutí je spuštěna scéna *Logo*. Ve scéně *Logo* se zobrazí logo herního simulátoru, a přes funkci *Application.LoadLevel ("Rozhrani");* se automaticky načte uživatelské rozhraní - herní menu. Dále scény *Jizda2* (vlastní hra), *Instrukce* (návod na ovládání hry) a *Zakonceni* (logo školy s automatickým odkazem pro ukončení hry).

6.7 Uživatelské rozhraní

Uživatelské rozhraní se skládá z místnosti, prostorového textu, který je možné rozbořit a ze tří vjezdů (Nová hra, Instrukce, Konec). Vjezdy jsou tvořeny funkcemi OnTriggerEnter. Pokud hráč vjede do brány, vykoná se funkce podle vjezdu do kterého vjel. Pokud hráč vjede do brány Nová hra, bude přesměrován přes funkci *Application.LoadLevel ("Jizda2");* do scény, kde je přímo hra. Pokud hráč vjede do brány Instrukce, bude přesměrován opět přes funkci *Application.LoadLevel ("Instrukce");* do téměř stejné scény, kde jsou napsaná tlačítka pro ovládání hry. V této scéně lze opět vybrat bránu dle výběru. Poslední možností je brána Konec, ke které je připojena funkce *Application.LoadLevel ("Zakonceni");*. Hráč bude po najetí do zóny přesměrován přibližně na sekundu na logo školy, poté se hra zcela vypne pomocí funkce *Application.Quit ();*.

Obrázek 25 - ukázka uživatelského rozhraní

7 Uživatelská dokumentace

7.1 Popis

Hra neobsahuje instalátor. Stačí spustit spouštěcí soubor hry. Po spuštění se objeví okno pro nastavení rozlišení obrazovky a kvalita hry. Při potvrzení bude uživatel přesměrován přes herní logo do uživatelského rozhraní.

V uživatelském rozhraní (herní menu) se uživatel pohybuje buď pomocí šipek, nebo kláves W, A, S, D (W – dopředu, A – doleva, S – brzda/dozadu, D - doprava). Za uvítacím textem se nacházejí brány – vstupy do dalších možností.

Pokud uživatel vjede do brány Nová hra, bude přesměrován do hry, kde se může opět pohybovat šipkami nebo W, A, S, D. Pauza se vyvolá tlačítkem P a odvolá opět stisknutím P. Pokud bude chtít hráč odejít ze hry, stiskne tlačítko ESC. Hra funguje na principu volné jízdy. Uživatel se tedy může pohybovat kdekoliv po městě, ale bude upozorňován na dopravní značky, a jak se má chovat.

Pokud uživatel vjede do brány Instrukce, bude přesměrován do scény, kde jsou vypsány tlačítka a jejich funkce – ovládání hry.

Hra se ukončí vjezdem do brány Konec.

7.2 Ovládání

Tlačítko	Akce
↑, W	Pohyb dopředu
↓, S	Brzda, Pohyb dozadu
←, A	Zatáčení vlevo
→, D	Zatáčení vpravo
P	Pauza, konec pazy
ESC	Herní menu

Tabulka 7 – ovládání hry

8 Závěr

V tomto projektu dlouhodobé maturitní práce bylo tématem vytvoření didaktické počítačové hry v platformě Unity3D. Celý proces od počátečních fází šel téměř podle plánů. Komplikace nastaly při tvorbě 3D modelu automobilu. Po importování do vývojového prostředí Unity bylo skrze tento objekt vidět, jakoby součást vůbec neexistovala. Tento problém byl vyřešen ve spolupráci s panem Mgr. Janem Vrzalem. Problém spočíval ve špatně otočených normálech polygonů, které model tvořily. Hra je didaktická, zaměřená na naučení dopravních situací na křižovatkách bez semaforů, s hlavními a vedlejšími silnicemi. Styl hry spočívá ve volné jízdě, při které je hráč upozorňován na pravidla silničního provozu na křižovatkách. V budoucnosti by hra mohla být rozšířena o nové funkce, zejména o testovací funkce hráče. Tyto funkce by nahradily doposud existující informační funkce u křižovatek a byly by nahrazeny funkcemi s otestováním zkušeností (např. Co je to za značku?).

8.1.1 Seznam použitých zdrojů

8.1.2 Literární zdroje

ŽÁRA, Jiří, Jiří SOCHOR, Petr FELKE a Bedřich BENEŠ. *Moderní počítačová grafika*. 2. vyd. Brno: Computer press, 2005, 608 s. ISBN 80-251-0454-0.

ODELL, Den. *JavaScript: průvodce programováním ajaxových aplikací*. Vyd. 1. Brno: Computer Press, 2010, 368 s. ISBN 978-80-251-2733-9.

NASH, Trey. *C# 2010: rychlý průvodce novinkami a nejlepšími postupy*. Vyd. 1. Brno: Computer Press, 2010, 624 s. ISBN 978-80-251-3034-6.

8.1.3 Internetové zdroje

Game engine [online]. 2013 [cit. 2014-03-08]. Dostupné z: http://en.wikipedia.org/wiki/Game_engine

What is a Game Engine? [online]. 2008 [cit. 2014-03-08]. Dostupné z: http://www.gamecareerguide.com/features/529/what_is_a_game_.php

API versus SDK [online]. 2009 [cit. 2014-03-09]. Dostupné z: http://stackoverflow.com/questions/834763/api-vs-sdk

Historie vývoje počítačových her [online]. 2014 [cit. 2014-03-09]. Dostupné z: http://www.root.cz/clanky/historie-vyvoje-pocitacovych-her-117-cast-vznik-hernich-enginu/

PhysX [online]. 2008 [cit. 2014-03-09]. Dostupné z: http://www.porse.cz/PhysX.html

3D formats [online]. 2013 [cit. 2014-03-15]. Dostupné z: http://docs.unity3d.com/Documentation/Manual/3D-formats.html

License Comparisons [online]. 2014 [cit. 2014-03-15]. Dostupné z: http://unity3d.com/unity/licenses

Release Archive [online]. 2014 [cit. 2014-03-15]. Dostupné z: https://unity3d.com/unity/whats-new/archive

Unity 3.5 [online]. 2014 [cit. 2014-03-16]. Dostupné z: https://unity3d.com/unity/whats-new/unity-3.5

Unity 4.0 [online]. 2014 [cit. 2014-03-16]. Dostupné z: https://unity3d.com/unity/whats-new/unity-4.0

8.1.4 Seznam obrázků

8.1.5 Seznam tabulek

8.1.6 Seznam příloh

Příloha 1 – náhled plakátu projektu

```
#pragma strict
var LPkolo : WheelCollider;
var PPkolo : WheelCollider;
var LZkolo : WheelCollider;
var PZkolo : WheelCollider;
var rychlost : float;
var maximalka : float = 100;
var otacky : float = 10;
function Start (){
rigidbody.centerOfMass.y = 0.5;

}
function FixedUpdate (){

LZkolo.motorTorque = otacky * Input.GetAxis("Vertical");
PZkolo.motorTorque = otacky * Input.GetAxis("Vertical");
LPkolo.steerAngle = 25 * Input.GetAxis("Horizontal");
PPkolo.steerAngle = 25 * Input.GetAxis("Horizontal");
}


function Update ()
{
rychlost = rigidbody.velocity.magnitude;
if (rychlost > 20){
LPkolo.steerAngle = 2 * Input.GetAxis("Horizontal");
PPkolo.steerAngle = 2 * Input.GetAxis("Horizontal");
}
if(Input.GetKeyDown("p"))
{
if(Time.timeScale == 1.0)
{
Time.timeScale = 0.000001;
}

else{
Time.timeScale = 1.0;
}
}
if(Input.GetKeyDown("escape"))
{
Application.LoadLevel("Rozhrani");
}
else{
}
}
```

Příloha 2 – skript fyziky a ovládání vozidla

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Jste na účelové komunikaci\ndejte přednost vozidlům\nna hlavní silnici</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 3 – skript pro upozornění na výjezd z účelové komunikace

```
#pragma strict

function Start () {

}
Application.LoadLevel("Rozhrani");
function Update () {
}

```

Příloha 4 – skript pro odkázání do scény „Rozhrani“

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Dejte přednost vozidlům\nna hlavní silnici!</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 5 – skript pro dodržení přednosti na vedlejší silnici

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>STOP!\nDáváte přednost všem na hlavní silnici!</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 6 – skript pro upozornění na značku stop

```
#pragma strict
var auto : Transform;
var VzdalenostKamery : float = 6.4;
var VyskaKamery : float = 1.4;
var priblizeni : float = 0.5;
var VychoziPohled : float = 60;
private var rotationVector : Vector3;

var kamerka : Transform;
function Start () {
}
    //if (auto.rychlost2<-100){
    //kamerka.y +180;

function LateUpdate () {
}
function FixedUpdate (){
var localVilocity = auto.InverseTransformDirection(auto.rigidbody.velocity);
if (localVilocity.z<-0.1){
rotationVector.y = auto.eulerAngles.y + 180;
}
else {
rotationVector.y = auto.eulerAngles.y;
}
var zrychleni = auto.rigidbody.velocity.magnitude;
camera.fieldOfView = VychoziPohled + zrychleni*priblizeni;
}
```

Příloha 7 – skript pro pohyb kamery v závislosti na voze

```
#pragma strict
function OnTriggerEnter ()
    {
    Application.Quit();
    }
```

Příloha 8 – skript pro ukončení hry

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Jste na hlavní silnici.\nMáte přednost.</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 9 – skript pro upozornění hlavní silnice

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Zde dejte přednost vozidlům\npřijíždějícím z pravé strany!</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 10 – skript pro upozornění přednosti z pravé strany

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Slepá ulička, \nna konci to otočte</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 11 – skript pro upozornění na slepou ulici

```
#pragma strict
var najezd : boolean;
var collider : BoxCollider;
function OnGUI()
{
if (najezd)
{
GUI.BeginGroup(Rect(Screen.width/2, Screen.height/2-170, 200, 150));
GUI.Label(Rect(0,0,200,150), "<color=yellow><size=20>Přikázaný směr jízdy\nvpravo, ihned odbočte</size></color>");
GUI.EndGroup();
}
}
function OnTriggerEnter(col : Collider)
{
najezd = true;
}

function OnTriggerExit(col : Collider)
{
najezd = false;
}
```

Příloha 12 – skript pro upozornění na přikázaný směr doprava

```
#pragma strict
function OnTriggerEnter ()
    {
    Application.LoadLevel ("Jizda2");
    }
```

Příloha 13 – skript pro spuštění scény Jizda2

```
#pragma strict
function OnTriggerEnter ()
    {
    Application.LoadLevel ("Instrukce");
    }
```

Příloha 14 – skript pro spuštění scény „Instrukce“

```
#pragma strict
function OnTriggerEnter ()
    {
    Application.LoadLevel ("Zakonceni");
    }
```

Příloha 15 – skript pro spuštění scény „Zakonceni“

```
#pragma strict

function Start () {

}

function Update () {
Application.Quit();
}
if(Input.GetKeyDown("escape"))
{
Application.Quit();
}
```

Příloha 16 – skript scény „Zakonceni“