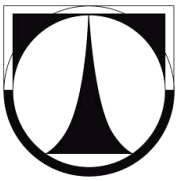


TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra aplikované kybernetiky
Studijní rok: 2011/2012

Diplomová práce

Internetová herní aplikace pro více hráčů

Bc. Dan Kryštof

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Katedra částí a mechanismů strojů **Studijní rok: 2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno a příjmení: **Dan Kryštof**
Studijní obor: **N2301 / Strojní inženýrství**
Obor: **3902T021 Automatizované systémy řízení ve strojírenství**
Zaměření: **Inovační inženýrství**
Ve smyslu zákona č. 111/1998 Sb. o vysokých školách se Vám určuje diplomová práce na téma:

Internetová herní aplikace pro více hráčů

Zásady pro vypracování:
(Uveďte hlavní cíle diplomové práce doporučené metody pro vypracování.)

1) Navrhněte strukturu herní internetové aplikace pro více hráčů (Massive Multiplayer Online Game) simulující automobilové závody). Do návrhu zahrňte možnost změny vzhledu vozidla podle požadavků konkrétního hráče a automatické ukládání nejlepších výsledků závodů. Při návrhu využijte volně dostupného vývojového herního rozhraní Unity3D a databázového systému MySQL.
2) Sestavte 3D model vozidla. Vytvořte základní textury pro výběr vzhledu vozidla.
3) Naprogramujte herní aplikaci v jednom ze skriptovacích jazyků (JavaScript,C#,Boo) podporovaných rozhraním Unity3D.
4) Nainstalujte herní systém na některém ze serverů katedry.
5) Ověřte funkčnost aplikace v reálném provozu.

Forma zpracování diplomové práce:

- průvodní zpráva cca 50 stran;
- Poster ve formátu A3, shrnující základní body práce.

Seznam literatury

[1] anon: *3D Platfomer Tutoriál-Building a 3D platform Game in Unity 2.0*

[2] NARAMORE, E., GERNER, J. SCOUARNEC, Y. BORONCZYK, T : PHP6, MySQL, OCH, Apache. Praha: Computer Press, 2009

[3] SHARP, J.: *Microsoft C# 2010-Krok za krokem*. Praha: Computer Press, 2010

[4] ZAKAS, N.: *JavaScript pro webové vývojáře*. Praha: Computer Press, 2010

[5] ŽÁRA, J. a kol.: Počítačová grafiky-principy a algoritmy. Praha: Grada, 1992

[6] ŽÁRA, J. - BENEŠ. B. - FELKEL, P.:*Moderní počítačová grafika*. Praha: Computer Press, 1998

Vedoucí diplomové práce: Ing. Michal Moučka, Ph.D.
Konzultant diplomové práce:

L. S.

Ing. Michal Moučka Ph.D.
vedoucí katedry

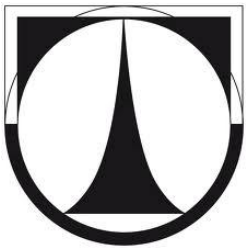
Doc. Ing. Miroslav Malý, CSc.
děkan

V Liberci dne 1. 1. 2012

Platnost zadání diplomové práce je 15 měsíců od výše uvedeného data (v uvedené lhůtě je třeba podat přihlášku ke SZZ).
Termíny odevzdání diplomové práce jsou určeny pro každý studijní rok a jsou uvedeny v harmonogramu výuky.

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta Strojní



Studijní program: N2301 – Strojní inženýrství

Studijní obor: 3902T021 Automatizované systémy řízení ve strojírenství

Zaměření: Automatizace inženýrských prací

Vedoucí diplomové práce: Ing. Michal Moučka Ph.D.

Diplomová práce

Internetová herní aplikace pro více hráčů

Bc. Dan Kryštof

Rozsah práce a příloh:
Počet stran: ……..82
Počet obrázků: …….28
Počet příloh: ……..4

V Liberci 14. 5. 2012

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č.121/200 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 20. 5. 2012

Podpis: …….

Poděkování

Na tomto místě bych chtěl především poděkovat vedoucímu diplomové práce Ing. Michalu Moučkovi, Ph.D. z katedry aplikované kybernetiky TU v Liberci za cenné připomínky, rady a čas, který mi věnoval při psaní této práce.

Dále bych chtěl poděkovat své rodině a přátelům za velkou podporu v průběhu celého studia.

Anotace

Tato práce se zabývá vývojem počítačové online hry. V prvních kapitolách vysvětluje elementární pojmy související s 3D počítačovou grafikou.

V další části práce je rozebrán pojem game engine a opět jsou uvedeny 2 praktické příklady současných enginů. Podrobněji je popsáno použité prostředí Unity3D a jeho nejpoužívanější nástroje.

Ke konci práce je demonstrována praktická implementace hry, herních pravidel, popis některých zajímavých funkcionalit a ukázka několika skriptů.

Klíčová slova

Cinema4D
Diplomová práce
JavaScript
Networking
Počítačová hra
Unity3D
3D grafika

Annotation

This thesis describes the development of computer games online. The first chapter explains the basic concepts of 3D computer graphics.
The next section explains the concept of game engine and provides 2 practical examples of current engines. More Unity3D environment is described and its most commonly used tools.
At the end of the work is demonstrated by a practical implementation of the game, game rules, a description of some interesting functionalities and several example scripts.

Keywords

Cinema4D
thesis
JavaScript
Networking
Computer game
Unity3D
3D graphics

Obsah

SEZNAM OBRÁZKŮ

Úvod

Člověk je od přírody tvor hravý. Většinu věcí vytvořených pro zlepšení své životní úrovně časem použije i pro svoji zábavu.

Ne jinak tomu bylo a je s výpočetní technikou. Původně ryze vědecké odvětví, kde se počítače využívaly jako nástroj pro rychlejší a přesnější zpracování dat, se postupem času stalo mnohem dostupnější a rozšířilo se tak více mezi obyčejné lidi. Počítače tak začaly plnit účel, ke kterému ani nebyly původně určeny. Jako zdroj interaktivní zábavy či odreagování od problémů všedních dní.

Počítačové hry se staly obrovským fenoménem a stejně jako filmy se i počítačové hry staly napevno součástí našeho života. Stejně jako bylo před dvaceti lety samozřejmé, že si kluci hráli s autíčky a holky s panenkami, tak samozřejmě si dnes mladí lidé čím dál častěji hrají ve virtuálním počítačovém světě, kde mohou prožívat neskutečná dobrodružství.

Herní průmysl a trh počítačových her tvoří obrovský podíl v zábavním průmyslu a stále roste. Po 41 letech, kdy vyšla první komerční počítačová hra, má toto odvětví roční obrat okolo 25 miliard dolarů. V roce 2010 prodaly herní společnosti 257 milionů videoher a počítačových her. Nezapočítávají se pirátské kopie, které by toto číslo o mnoho navýšily.

Původně velmi jednoduché a málo pixelové hry nahradily hyper-realisté plně interaktivní trojrozměrná prostředí obsahující složité herní mechanismy. Spousta současných her už vypadá jako filmy. Vlastně mnohdy lépe než filmy.
Díky poslední generaci ovládacích periferiím založené na principu snímaní pohybu hráče (např. kinect od microsoftu) si divák pro zobrazený předmět může téměř sáhnout a zážitek ze hry se stává přitažlivější a dostupnější.

Nádherná prostorová grafika však není zadarmo. Čím je hra novější nebo čím reálnější nabízí efekty, tím potřebuje výkonnější a dražší hardware.

Tento obrovský vývoj technologií vytváří obrovský tlak na výrobce počítačových komponentů, kteří musí vyvíjet sofistikovanější, výkonnější, výpočetní a grafické procesory pro počítače a konzole. Se značnou nadsázkou se dá říci, že počítačové hry byly jedním z katalyzátorů, který způsobil masivní vývoj počítačového HW v posledních 20 letech.

Počítačové hry neslouží pouze k zábavě, ale mohou nás i něco naučit. Čím dál tím více vědeckých studií naznačuje, že hraní zlepšuje kreativitu, schopnost rozhodování a vnímání. Konkrétní výhody jsou široké od zlepšení koordinace mezi očima a rukama u chirurgů až po větší kreativitu dětí. Lidé, kteří hráli akční počítačové hry, se rozhodují o 25 procent rychleji než jiní, aniž by se zhoršila správnost či přesnost jejich rozhodnutí.
Zkušení hráči jsou schopni věnovat pozornost více než šesti věcem najednou oproti „nehráčům“, kteří zvládají jen čtyři.[1]

Je třeba říct, že počítačové hry mohou být nápomocny v širokém spektru vědních oborů pedagogie, umění, architektury, práva, inženýrství, aplikované vědy nebo půl tuctu dalších věcí. Například počítačové hry jsou v současnosti nejrozšířenější a nejčastější aplikací metod umělé inteligence.

Když jsem zvažoval téma svojí diplomové práce, chtěl jsem takové, které by mě bavilo a zároveň by bylo jejím výstupem něco zajímavého a zábavného pro širší okruh lidí. To mě přivedlo k myšlence vytvořit hru, pokud možno co nejméně náročnou na HW, aby si ji mohlo zahrát co nejvíce lidí, avšak s grafikou na dostatečně vysoké úrovni.

Na otázku, který žánr použít, se mi závodní simulátory zdály jako dobrá volba pro vytvoření zábavné a chytlavé hry.
Hlavním znakem simulátorových her je virtuální svět co nejvíce podobný tomu skutečnému. Simulátory se snaží napodobit podmínky, které panují v realitě- fyzikální a přírodní zákony, které se pak odráží ve virtuálním světě podobně jako v reálném světě.

Na začátku je potřeba si ujasnit, co vše je a co není v mých silách. Nebylo by moc rozumné pokusit se v rámci diplomové práce vytvořit kompletní herní engine se všemi funkcemi. Alespoň ne jedním člověkem. Proto jsem se rozhodl použít pro vývoj hry některý z dostupných současných enginů.

Po konzultaci s vedoucím práce jsem zadání rozšířil o možnost hraní více lidí po síti, i když jsem tento styl hraní původně v úmyslu neměl.
Hraní po síti bylo přidáno nejen pro jeho zajímavou programovou stránku, ale také představuje mnohem větší výzvu pro hráče. Hráč je vytrvale přítomen ve virtuálním světě spolu s dalšími lidmi. Díky tomu hra nabídne svým uživatelům stále nové neopakovatelné situace.

Celkový vývoj počítačové hry zahrnuje řešení několika okruhů problémů. Prvním je, jak získat zdrojová data a modely. Za druhé, jak reprezentovat obraz. Za třetí, jak ukládat příslušné datové soubory a jaká bude jejich velikost a čtvrtý problém řeší, jakým způsobem a na jakém zařízení výslednou hru zobrazit.

Součástí vývoje je grafický návrh hry, tzn. návrh jména, hlavních a vedlejších menu apod. Dále návrh prostorů, kde se bude hra odehrávat, tedy jakou grafickou podobu bude mít terén a jak bude vypadat každá dílčí část hry.

Cíle této diplomové práce bych tak stručně shrnul do těchto bodů:

- tvorba modelů v libovolném 3D grafickém softwaru;
- volba vhodného grafického enginu pro aplikaci;
- vytvoření virtuálního světa hry;
- vytvoření fyzikálního modelu automobilu co nejvíce se podobající reálnému vozu;
- navrhnutí kompletního uživatelského prostředí aplikace;
- naprogramování rozhranní server/klient obsluhující veškeré události související s hrou po síti;
- vytvoření databázového systému spravující informace o online hráčích;
- optimalizace pro co nejlepší plynulost.

Členění práce

V první kapitole jsou vysvětleny základní pojmy související s počítačovou grafikou. Následuje popis současného stavu grafických editorů a postup výběru softwaru použitého pro tvorbu modelů hry.

Ve třetí kapitole je ukázán základní přehled nejpoužívanějších herních enginů. Podrobněji je popsáno použití Unity3D a jeho vývojové prostředí. To zahrnuje popis nejdůležitějších částí, dostupné technologické možnosti a stručný přehled jednotlivých nástrojů.

Tato kapitola také obsahuje úvod do networkingu v Unity a zmiňuje řešení dále vyjmenovaných problémů. Jak vytvořit vlastní server a jak se k němu připojit, jak spravovat seznam připojených uživatelů, jak navrhnout a vytvořit grafické rozhraní v unity a zasílání zpráv mezi klienty.

Čtvrtá kapitola popisuje zvolené programové řešení a postup při vytváření návrhu designu počítačové hry. Dále kapitola obsahuje praktické implementace předchozích částí v konečné hře. Jsou popsány použité herní mechanismy, příklady kódů a schéma některých praktických řešení.

Pátá kapitola pojednává o použitém databázovém systému MySql a jazyku PHP. Konec práce zahrnuje návod ke hře a popis testování hry.

1 Počítačová grafika

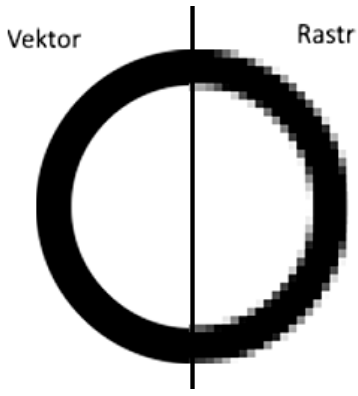
Počítačová grafika je z technického hlediska obor informatiky, který používá počítače k tvorbě virtuálních grafických objektů a dále také na úpravu zobrazitelných a prostorových informací, nasnímaných z reálného světa. [2]

Počítačová grafika se dnes v různých formách používá prakticky všude kolem nás, a ani si toho nemusíme být vědomi. Ať už ve formě technických výkresů, digitálních fotografií nebo rozpohybována ve filmech až po zcela interaktivní 3D aplikace.

1.1 Vektorová vs. Rastrová grafika

Počítačovou grafiku lze z principu rozdělit na 2 hlavní oblasti - 2D a 3D. 2D grafiku lze ještě dále dělit na vektorovou a rastrovou.

U vektorové grafiky je obraz rozdělen na matematicky definované části - křivky, lomené čáry a jimi ohraničené stejnorodé plochy.
Každé z těchto částí obrazu se přiřadí kódovaná informace o tom, jak má být zobrazena, jak silnou čarou, jakou barvou popř. jakým vzorkem má být uzavřená plocha vybarvena. Jelikož jsou jednotlivé objekty definovány pomocí analytické matematiky, jsou nezávislé na rozlišení a lze tak měnit jejich velikost bez sebemenší ztráty kvality obrazu.



Obrázek 1 - Porovnání vektorové a rastrové grafiky

Naproti tomu rastrová grafika (jinak také bitmapová) má základ v pravidelné síti pixelů. Každý pixel nese specifické informace např. o jasu, barvě, průhlednosti bodu. Rozlišení rastrové sítě vůči zobrazovaným datům je dopředu dané a nelze je jednoduše měnit bez snížení kvality zobrazení.

Jelikož se tato práce zaměřuje převážně na 3D grafiku, nebude téma 2D grafiky nějak zvlášť rozepisováno a soustředí se pouze na 3D.

1. 2 Grafika 3D

Jaký je zásadní rozdíl mezi 2D a 3D grafikou snad podvědomě chápe každý. Co ale nemusí být jasné, jak taková práce ve 3D funguje. Trojrozměrná grafika pracuje s trojrozměrnými objekty (mají výšku, délku a hloubku), čímž oproti 2D grafice získává možnost ukázat model ze všech stran. Pokud bude srovnána 3D a 2D grafika, tak 3D bude principielně více příbuzná 2D vektorové grafice. Trojrozměrné objekty jsou neztrátové a po změně jejich velikosti nebo při přiblížení nedochází ke ztrátě kvality výsledného obrazu.

3D grafiku lze potkat téměř všude a je jen málo oborů, které ji částečně nevyužívají. Hlavní odvětví, kde došlo k jejímu masivnějšímu rozšíření, jsou zmíněna v další části.

1. 2. 1 Filmový průmysl

První použití 3D grafiky ve filmu bylo již v roce 1971 a nyní je možná jednodušší hledat filmy, které by ji v současnosti nevyužívaly. Autoři mohou vytvořit věci, které by se reálně dělaly buď obtížně, nebo vůbec (např. filmové zpomalení času, vesmírné lodě apod.). Jako příklad může skvěle posloužit film Avatar, který kromě svého 3D provedení diváky oslnil i 3D zobrazením (nezaměňovat 3D zobrazení s 3D grafikou). Tento film byl prakticky kompletně tvořen ve studiu.

1. 2. 2 Reklama

Pokud si člověk vybírá z katalogu nějakou elektroniku, pravděpodobně uvidí nádherně vypadající a lesknoucí se výrobek. Když si ho pak ale koupí, dostane jen obyčejnou černou škatuli.

Právě v reklamě se hojně využívá 3D grafika pro vizualizaci výsledných produktů, aby pro zákazníka vypadaly co možná nejlépe. Vypiplané stíny, materiály a další vlastnosti umí výrobek lépe prodat.

1. 2. 3 Herní průmysl

Po filmovém průmyslu se jedná o další odvětví, které táhne vývoj 3D grafiky poslední dvě dekády. Dnešní 3D tituly překypují nejnovějšími technologiemi. Kromě samozřejmosti, jako je výrazné navýšení počtu polygonů nebo rozlišení textur, jsou velmi pokročilé metody způsobu nasvícení scény. Nechybí perfektní dynamický cyklus dne a noci, HDR, dynamické stíny a objekty s vysokým počtem polygonů apod.
Je jen otázkou několika nejbližších let, kdy ani hráči nebudou vědět, zda je realita okolo nebo uvnitř monitoru.

1. 2. 4 Strojírenství

3D grafika značnou měrou zasahuje i do dalších odvětví, jako je strojírenství, elektrotechnika, architektura a zdravotnictví.

3D modelační nástroje umožňují ve strojírenství navrhovat, vizualizovat a simulovat výrobky na přesném 3D modelu ještě předtím, než dojde na jejich vlastní výrobu. Vytváření takovýchto modelů pomáhá firmám navrhovat dokonalejší výrobky, snižovat náklady na vývoj a rychleji uvádět výrobky na trh.

Vývoj za pomoci 3D digitální vizualizace se postupně prosazuje v řadě průmyslových oborů, protože nahrazují postup návrhu, konstrukce a testování s reálnými prototypy výrobků nebo technologií výroby, které jsou velmi nákladné, nepružné a tudíž neefektivní. Virtuální modely mohou být snadno měněny a nové prototypy mohou být vytvořeny a matematicky testovány v řádech hodin namísto původních měsíců.

Obrázek 2 - Model motoru vytvořený v PTC Creo (převzato z [7]).

V technické praxi se určitě nejvíce používají produkty Autodesk AutoCAD, ArchiCad, SolidWorks, Catia a mnohé další.

Používání 3D grafiky ve strojírenství s sebou přináší následující výhody:

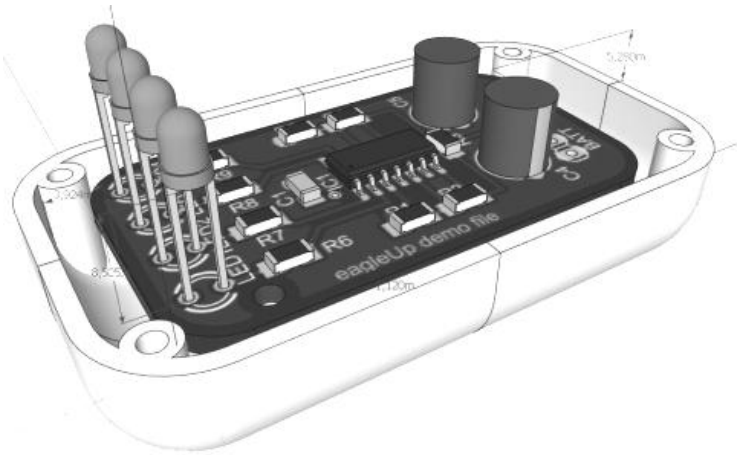
- snížení časové náročnosti vývoje nových výrobků a technologií;
- omezení zmetkovitosti;
- variantní řešení;
- minimalizace nebo úplné odstranění nákladů při provádění reálných prototypů;
- úspora materiálu a energie apod.

Výstup takové strojírenské aplikace může být ve formě statických renderů či 3D animací (video sekvence znázorňující objekt při různých stavech). Takto vytvořené video podstatně rozšiřuje možnosti prezentace vyvíjených produktů.

Další možností výstupu je vytvoření multimediální prezentace (např. FLASH, DHTML, PPS), díky níž lze vytvořit interakci budoucího výrobku s divákem.
Značně se urychlila a zpřesnila tvorba výkresové dokumentace oproti 2D CAD nástrojů.

1. 2. 5 Elektrotechnika

Současné softwary zvládají nejen nakreslit schéma a s ním spojené desky plošných spojů, ale navíc i vytvořit 3D vizualizaci osazení desky a příslušenství.



Obrázek 3 - Prostorová vizualizace tištěného spoje (převzato z [8]).

1. 3 Základní pojmy 3D grafiky

V této kapitole jsou vysvětleny základní pojmy týkající se 3D grafiky, mezi které patří body a hrany, polygony, objekty, normál mapy apod.

1. 3. 1 Body a hrany

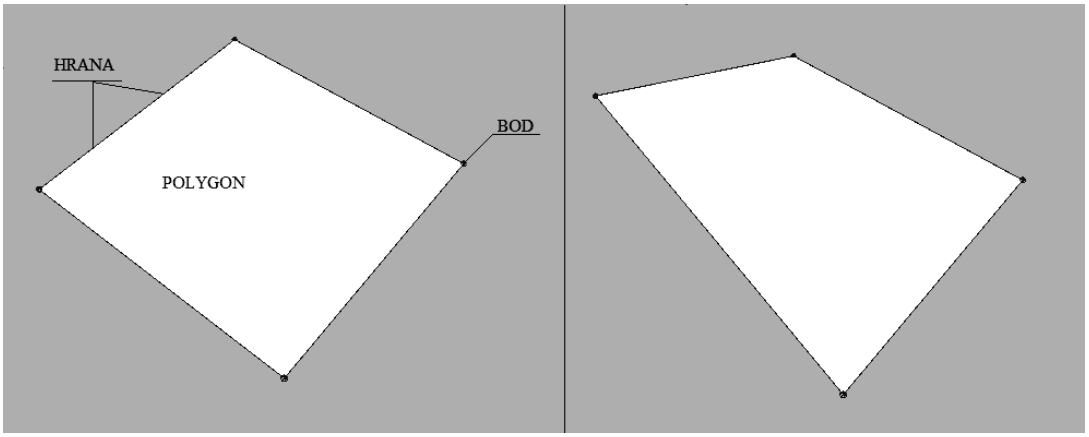
Jsou to základní elementy při prostorovém modelování. Body (jinak zvané vertexy) mají jednoznačně danou svou pozici v prostoru. Při propojení dvou vertexů přímkou vznikne hrana.

1. 3. 2 Polygon

Po spojení tří hran vzniká trojúhelník, ten tvoří jednu plošku z povrchu modelu neboli polygon. I přesto, že dnešní standardy pro zobrazování dat podporují i čtyřúhelníky a mnohoúhelníky, tak jsou při výpočtu výsledného obrazu zpravidla převedeny na dílčí trojúhelníky, tzn., že čtverec bude tvořen 2 polygony.

Důvod pro použití trojúhelníků je nasnadě, neboť 3 body vždy určují rovinu a je tak snadné určit tzv. **normálu povrchu** na rozdíl od jiných mnohoúhelníků.
Každý polygon má vlastně dvě normály, které míří od sebe. Druhá je často označována jako zadní stěna (back face) polygonu. To, která strana bude přední a která zadní je velmi důležitý parametr, neboť některé grafické enginy vykreslují pouze přední stěny a zadní ignorují.

Většina modelačních nástrojů nabízí samostatný režim úprav bodů, hran a polygonů. Ať už se upravuje jakýkoliv z nich, upravují se zároveň parametry obou zbylých.



Obrázek 4 - Dílčí části polygonu a jejich vzájemná propojenost při editaci

1. 3. 3 Objekt

Skupině polygonů spojených sdílenými body se říká objekt (*mesh*). Oproti 2D grafice se objekt zobrazuje ve všech třech osách- výška, šířka a hloubka (x, y, z). Ve scéně se tak nepohybuje jen horizontálně a vertikálně, ale lze se pohybovat k sobě/od sebe a se scénou libovolně otáčet a naklápět ji.

Počítač sám zobrazit „prostorový“ obraz neumí, proto se 3D objekt převede matematickým výpočtem na 2D obraz a ten se zobrazí na monitoru. Tento proces převedení 3D modelu na 2D obraz se nazývá rendering. Rendering si lze představit jako vyfocení scény fotoaparátem, kdy je výsledkem bitmapový obrázek.

Obecně platí, čím více mají objekty polygonů, tím jsou detailnější ale také mnohem náročnější na výpočet.

1. 3. 4 Nízkopolygonové vs. Vysokopolygonové objekty

Objekty lze podle počtu polygonů rozdělit na nízkopolygonové (lowpoly) a vysokopolygonové (highpoly) objekty. Obě tyto varianty mají své výhody a nevýhody a jiné využití.

Čím více polygonů se použije, tím detailnějšího zobrazení modelu lze docílit. Nevýhodou těchto modelů je dlouhý renderovací čas a s tím související náročnost na HW.

Přesně opačně je to u lowpoly modelů, které mají minimum detailů a minimální zátěž na výpočet.

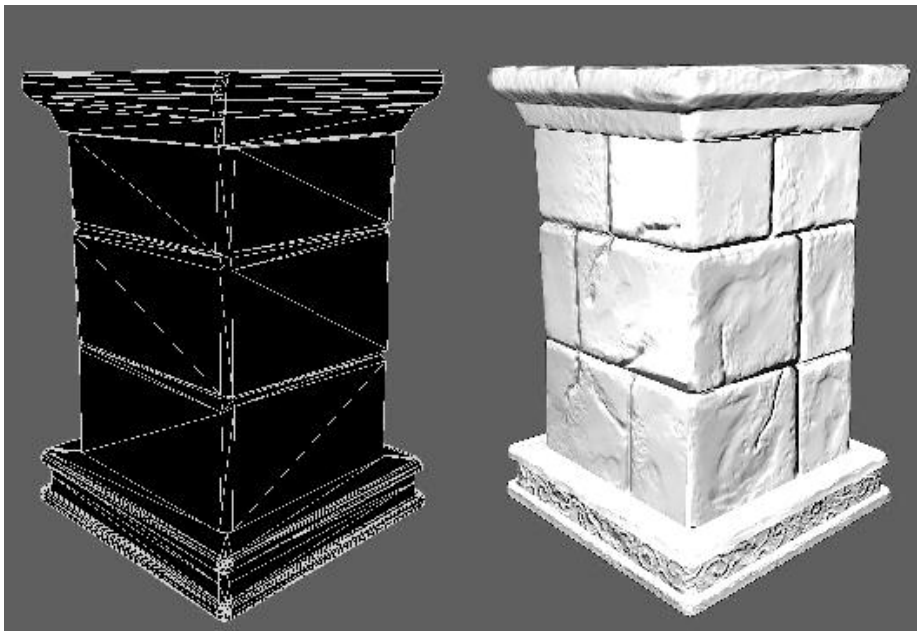
Optimální cesta je někde uprostřed. Z high polygonových objektů lze různými postupy vytvořit displacement mapu nebo normálovou mapu.
Obě textury na lowpoly modelu vytváří falešný dojem detailnosti, aniž by se přidávaly nové polygony.

1. 3. 5 Normálová textura

Normálová mapa představuje speciální texturu sejmutou z detailního modelu, která je většinou tvořená RGB spektrem. Toto spektrum definuje hodnotu normálového vektoru (směr vizuální deformace), kdy Z (modrá) je ve směru normály, Y (zelená) je nahoru a X (červená) vpravo.

Matné materiály jsou např. charakteristické tím, že nejsou dokonale hladké a světlo se na jejich povrchu rozptyluje všemi směry, zatímco na ideálně hladkém povrchu se jen odráží

v závislosti na normálovém vektoru. Na základě těchto vektorů získává model rozdílné osvětlení. Pomocí normálové mapy lze zvětšit počet těchto vektorů na dvojnásobek.
Získá se tak mnohem více detailů i na relativně jednoduchém modelu a to především v kombinaci s vhodnými osvětlovacími technikami, jak je vidět na Obrázku 5.



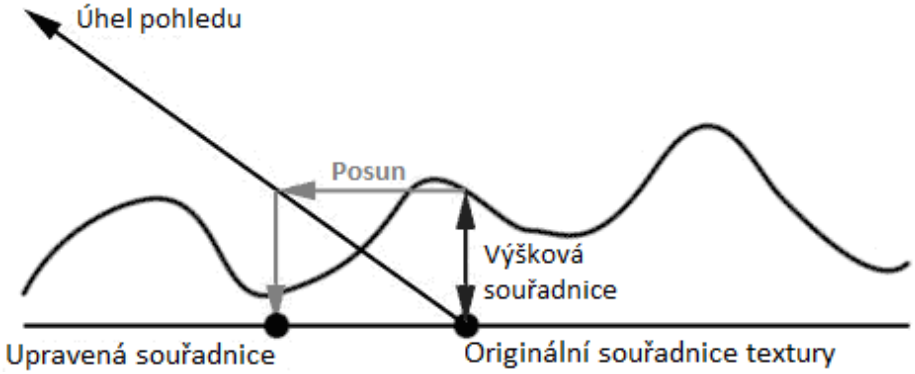
Obrázek 5 - Low poly model s normál mapou

1. 3. 6 Výšková mapa

Výšková mapa (displacement map) je další z pokročilých metod používaných v 3D grafice pro vytvoření detailnějšího povrchu na jednoduchých modelech.

Nové detaily v tomto případě vytváří 8bitová textura, která určuje členitost povrchu. Jak tedy tato technologie pracuje, popisuje Obrázek 6. Díky výškové mapě upraví souřadnice textury tak, aby pozorovatel nabyl dojmu, že se dívá na zakřivený povrch.

Původní textura je upravena výškovou mapou. Algoritmus vzdálí vystouplá místa textury od oka pozorovatele, zatímco propadliny k němu naopak přiblíží. Je tedy jasné, že posuny a vlastně "deformace" textury vycházejí z hodnot ve výškové mapě a úhlu pohledu. [2]



Obrázek 6 - Princip výškové mapy

Ve zkratce lze popsat rozdíl mezi Normál a Displacement mapou. Normál mapa zakřivení povrchu pouze simuluje, tudíž silueta objektu a jeho vrhané stíny jsou stále jen obrazem původního objektu. Tento nedostatek řeší displacement mapa, která je schopná členitost povrchu přímo vytvořit a měnit tak i siluetu modelu.

1. 3. 7 Scéna

Je pracovní prostor, ve kterém se vše odehrává. Ve scéně jsou uloženy všechny vytvořené objekty, materiály a světla.
Základní kostru scény tvoří:

1. kamera, která promítá scénu na svou obrazovku;
2. světlo (jedno nebo více, může být zhmotněno nějakým objektem) osvětlující objekty scény;
3. veškeré objekty, které tvoří scénu.

Pohyb ve scéně probíhá buď podle nějakého globálního souřadného systému scény, nebo podle souřadného systému jednotlivých objektů, které se stávají v ten okamžik globálními.

1. 3. 8 Textura

Textura je 2D obrázek, který „obaluje“ celkový povrch modelu, čímž výrazně ovlivní jeho konečný vzhled. Za texturu lze použít jakýkoliv obrázek nebo fotografii od betonové podlahy, vegetaci až po vlastní obličej. Pokud se přeci jen nenalezne vhodná textura, není nic jednoduššího, než si vytvořit zcela novou v oblíbeném grafickém editoru.

Z hlediska způsobu vytvoření textury lze rozlišit 2 typy:

Procedurální textury jsou vyjádřeny pomocí nějakého matematického algoritmu, z čehož vyplývá, že nepotřebují žádný zdrojový obrázek. Další výhoda spočívá v jejich nezávislosti na rozlišení. Procedurální textura se přizpůsobí velikosti renderovaného modelu. Protože je textura generována pomocí matematické funkce, je možno ji libovolně skládat vedle sebe. Nelze poznat, kde začíná a kde končí. Tímto způsobem lze vykreslit velkou plochu jen za pomocí malého obrázku. Určitou nevýhodou použití tohoto typu je „homogenní“ vzhled povrchu.

Bitmapové textury jsou v tomto ohledu mnohem flexibilnější. Je u nich omezující faktor, který vyplývá z velikosti dostupné paměti, ale dovolují přidávat detaily tam, kde jsou zrovna potřeba.
Bitmapové textury používají vlastní předpřipravený rastrový obrázek. Pro nejlepší výsledek je potřeba mít nejdetailnější texturu. Opět platí, že čím větší obrázek, tím větší jsou nároky na HW, proto je vhodné najít hranici, kdy použití vetší textury má kýžený efekt a kdy už nemá.

1. 3. 9 Materiál

V počítačové grafice materiály většinou tvoří tři složky, které se vhodně prolínají podle pravidel, které platí i v přírodě. Mezi nejpoužívanější patří následující tři složky.

Barevný kanál (Diffuse) je vlastní barva materiálu a určuje kolik světla má materiál odrazit nebo pohltit. Do tohoto kanálu se načítá připravený rastrový obrázek.

Průhlednost (Transparency) často též Alfa kanál udává hodnotu průhlednosti materiálu.

Odrazová složka (Reflectivity) říká, jak moc bude povrch ovlivňován okolními odrazy, tzn., jaká část světla se odrazí a jaká bude tvořit odrazy. Tato složka se používá k vytvoření iluze lesklého materiálu.

Odrazový kanál v sobě obsahuje informace o "jasnosti" jednotlivých pixelů. Textura tohoto kanálu může být černobílá (čím světlejší bod, tím větší jasnost, černá znamená žádnou změnu), anebo barevná, přičemž barva napovídá o vlastnosti povrchu. Do modra zbarvená odrazová složka (tedy její jednotlivé pixely) přidá materiálu kovový nádech.

Základní složky materiálů mohou být dále rozšířeny o další vrstvy.

2 Současné grafické softwary

Každá hra potřebuje na počátku velkou spoustu dat ve formě textur, 3D modelů, animací a mnoha dalších. Většina herních enginů přímo neumožňuje vytvářet složité modely nebo textury, ale využívá konverzí z jiných formátů. Obvykle se pro vytváření modelů a textur využívají nástroje třetích stran.

Dnes je na trhu poměrně široký výběr grafických programů a situace je mnohem lepší, než tomu bylo v minulosti, a tak si přijde na své téměř každý. Existují softwary pro různá odvětví a cílové skupiny, pro které jsou určeny. Lze nalézt produkty pro vytváření strojírenských součástek až po aplikace schopné vytvořit fotorealistické animované scény.
V oblasti vizualizační grafiky jsou nejoblíbenější Blender, 3DSMax (Autodesk), Cinema4D (Maxon), Maya. Každý tento software poskytuje určité nástroje a různou finanční politiku a je na koncovém zákazníku, zda mu bude vyhovovat či nikoliv.

Na použitý grafický software jsou stanoveny níže popsané požadavky:

- nejkomplexnější modelační nástroje, protože je potřeba vytvořit velké množství různorodých modelů;
- zvládat editovat/vytvářet UV mapy modelů;
- obsahovat alespoň základní grafický editor pro tvorbu a úpravu textur;
- možnost exportu do nejpoužívanějších 3D formátů pro externí aplikaci.

První místo, kam každý grafik při výběru směřuje, jsou oficiální stránky vydavatele. Nalezne zde základní informace o programu, přehled o tom, co daný SW zvládá, minimální HW nároky, ale hlavně lze zpravidla stáhnout demo softwaru zdarma. Ovšem tyto verze jsou nějakým způsobem omezeny. Buď časem použití, nebo ukládáním projektu apod. Přesto se jedná o skvělý způsob, jak daný program otestovat na vlastní kůži. Takový test je rozhodně k nezaplacení.

Mezi hlavní favority patří: 3DSMax (Autodesk), Cinema4D (Maxon).

Z vlastní zkušenosti mohu říci, že 3DS Max i Cinema4D jsou si prakticky rovni. Jde jen o to, na který SW si uživatel zvykne. Na poli 3D scény je Maxko o něco rozšířenější, musí se ale vzít v úvahu, že má delší historii než C4D, tudíž bude logicky používanější.

Pro mne hroznou nevýhodou 3DSmaxka je jeho prodejní politika. 3DsMax se může zakoupit pouze jako celek. Nelze si zvolit jen určité části, ale musí se koupit všechno.
Oproti tomu má C4D otevřenou modulární architekturu, takže jde z jednotlivých modulů vybrat a koupit jen potřebné.

Seznam modulů, které vydává firma Maxon, jsou popsány níže.

- **Advanced Render** je modul pro pokročilé nastavení renderingu (HDRI, Caustic, Ambient Oclussion, simulace mraků).
- **BodyPaint 3D** označuje modul pro pokročilé textůrování a malování přímo na 3D objekty.
- **Dynamics** modul pro simulaci dynamiky měkkých a tvrdých těles a fyzikálních zákonů.
- **Hair** modul slouží pro tvorbu vlasů, chlupů, srsti a trávy.
- **MOCCA** modul znázorňuje animaci postav a možnost animace oděvů.
- **MoGraph** je modul pro tvorbu televizní grafiky a reklam.
- **PyroCluster** znázorňuje modul pro snadnou tvorbu ohně a kouře.
- **Thinking Particles** je modul pro pokročilou tvorbu částicových systémů. [3]

Dalším negativem MaxStudia je lokalizace. Za dlouhou dobu jeho vývoje nikdy nebyl lokalizován do češtiny. Oproti tomu Cinema je vždy po vydání dostupná i v české lokalizaci. Čeština není rozhodně ten nejdůležitější parametr při výběru, ale určitě se pracuje s pokročilejšími nástroji lepé a rychleji pokud jsou v mateřském jazyku.

Z předešlých odstavců je patrné, kterým směrem se ubírala volba výběru softwaru.
V konečném hodnocení vyšla lépe Cinema4D. Poskytuje vynikající systém modelování, díky kterému lze projektu vtisknou použitelnou moderní grafickou podobu, má velmi intuitivní ovládání a širokou českou uživatelkou podporu.

2. 1 Cinema4D - základy

Program vytvořila německá společnost MAXON Computer, která má již 25letou tradici v oblasti vývoje profesionálních nástrojů pro tvorbu digitálního obsahu ve 3D. Cinema4D nabízí integrované prostředí pro modelování, texturování, vizualizaci a animaci ve 3D.

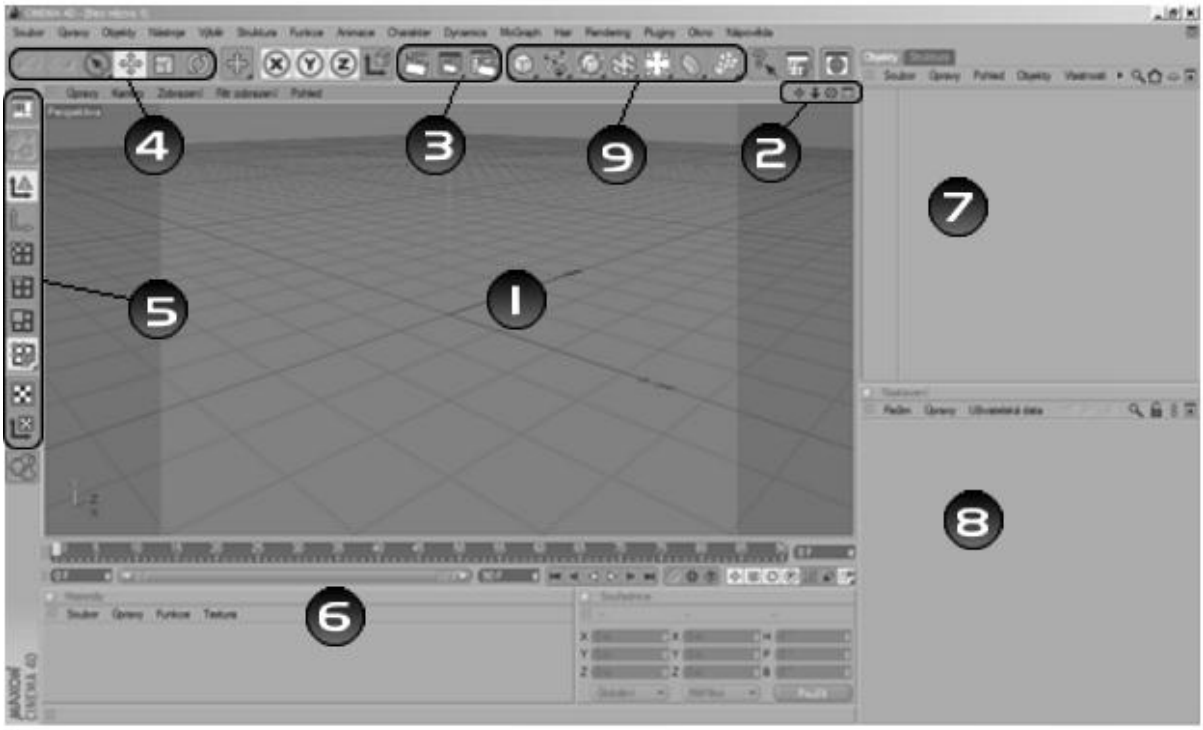
Program vyniká rychlou učící křivkou. Již během několika dní jsou i začínající uživatelé schopni vytvořit netriviální model, pokrýt jej realistickými texturami, nasvítit a rozhýbat v krátké animaci.

(Poznámka: Uhodli- byste, co znamená 4D v názvu? Jedná se o čas, který v době jejího vzniku značil schopnost vytvářet i animace, což tehdy nebyl každý modelační nástroj schopen.)

Pro tvorbu všech modelů byla použita verze Cinema4D Release 11. 5, lokalizovaná v češtině. Rozdíly s poslední verzí 13 jsou sice již markantnější, než tomu bylo s verzí 12, zásadní změny se však odehrály v nástrojích, které pro tuto práci nejsou až tak podstatné.

V Cinemě je velké množství panelů se základními a nejdůležitějšími nástroji. Bohužel zde není prostor na popisování a vysvětlování všech pokročilých možností C4D . Na druhou stranu je důležité zde zmínit základní postupy, které přiblíží způsob, jakým je vytvořena většina 3D modelů.

Znázornění základního prostředí po spuštění C4D:



Obrázek 7 - Základní prostředí C4D

1) Scéna: Horní lišta obsahuje několik důležitých příkazů, potřebných při pohybu v 3D prostoru scény:

Kamery: Na výběr jsou různé možnosti, jak sledovat pracovní scénu. Standardní zobrazovací režimy jsou levý/pravý, přední/zadní a perspektiva, ve které je možné jako jediné scénu natáčet, ve zbylých lze pouze přibližovat a posouvat pohled.

2) Změna pohledu: V pravém horním okraji scény se nacházejí důležité ovládací prvky, tlačítka *Posun* pohledu, *Přiblížení/Oddálení* (ZOOM) pohledu a *Rotace* pohledu. Podržením LMB na ikoně a následným pohybem myši se upravuje aktuální pohled na scénu.

Změna pohledu je možná pomocí zkratek, ty jsou dostupné pro většinu popisovaných příkazů. Po delší spolupráci rychle přejdou do krve a práce je s nimi mnohem rychlejší a proto jsou následně uvedeny:

L-Alt+ RMB[*]- podržením zkratky a posouváním myši se přibližuje/oddaluje

L-Alt+ roller - podržením zkratky a posouváním myši se posouvá pohled;

L-Alt+ LMB[**]- podržením zkratky a posouváním myši se otáčí pohled.

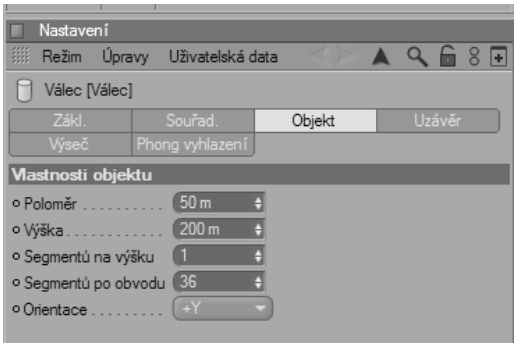
Poslední ikonka se používá k Přepínání oken.

Mezi pohledy se lze přepínat zkratkami *F1*- perspektiva, *F2*- vrchní, *F3*- zprava, *F4*- přední, *F5*- zobrazení skupiny všech 4 oken.

3) Primitiva: Tento panel slouží pro vkládání nových objektů do scény (Primitiva, Křivky, NURBS, Světla,…). Po výběru a vložení kterékoliv primitivy lze editovat její parametry. Například u válce jde měnit jeho poloměr, výška a počet segmentů po obvodu. Přímo ovlivňují „hladkost“ a úroveň detailů válce.

[*] RMB – right mouse button neboli pravé tlačítko myši

[**] LMB – left mouse button neboli levé tlačítko myši



Obrázek 8 - Panel nastavení primitiva (válce)

Pokud zcela vyhovuje základní nastavení objektu a je potřeba ho dále editovat na úrovni polygonů, převede se na editovatelný (klávesa c) nebo příslušnou ikonou v panelu číslo 5.

4) Nástroje pro výběr, posun a rotaci objektu. Kombinací kláves Shift+ LMB je možné přidávat další entity k aktuálnímu výběru a pomocí *Ctrl* pak odebírat. Poslední ikony reprezentují posun (*E*), velikost (*T*) a rotaci (*R*) modelů.

5) Pracovní schéma C4D: Výběr režimu úprav (polygony, hrany, body) označeného objektu. Dále nastavení pracovního prostředí, které umožňuje vybrat z několika přednastavených schémat pracovních prostředí pro specifickou práci a několika vlastních uživatelských. Pro potřeby práce postačí zmínit pouze standardní a BodyPaint, ve kterém byly vytvořeny všechny UVmapy a textury.

6) Manažer materiálů

7) Manažer objektů

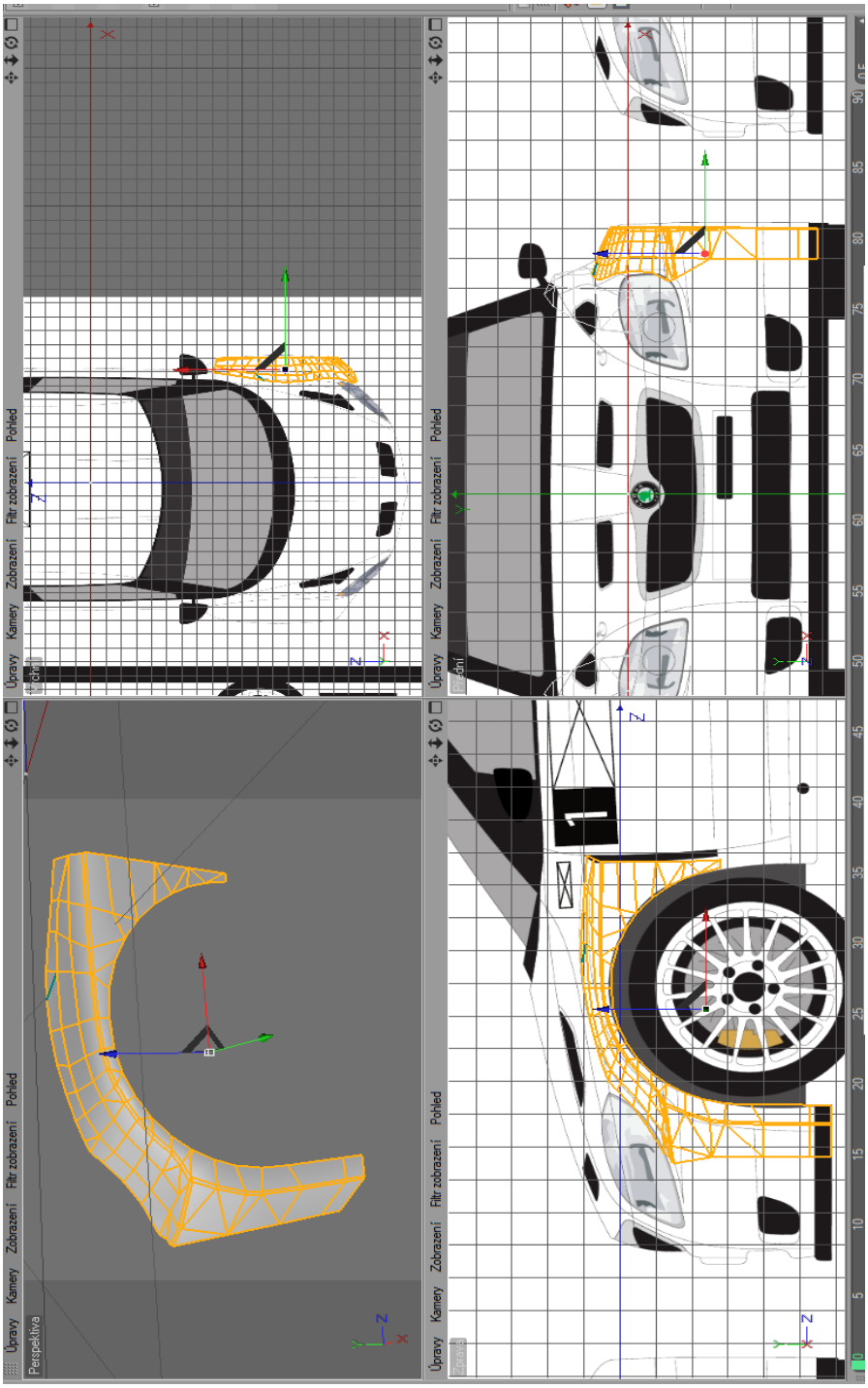
8) Nastavení aktuálně zvoleného (Nástrojů, Objektů, Materiálů aj.): Tento panel při modelování a používání programu je jeden z nejdůležitějších. C4D obsahuje nepřeberné množství objektů, funkcí či nástrojů a v tomto panelu se jednoduše zobrazí nastavení a možnosti aktuálně zvoleného. Nelze tedy popisovat nastavení všech možných možností i nemožností.

9) Renderování a nastavení renderingu

2. 2 Postup tvorby modelu

V následujících řádcích bude shrnut postup, který je použit při tvorbě modelu automobilu škoda Fabia. Popsanou techniku je však možné použít i pro tvorbu jiných modelů bez ohledu na jejich tvar nebo účel.

Společným rysem pro veškerou tvorbu je snaha o nejmenší počet použitých polygonů kvůli nejnižší grafické i výpočetní zátěži.



Obrázek 9 - Ukázka použití podkladů při modelování

Aby se 3D modely nejvíce blížily svým skutečným předlohám, používají se často při modelování podklady. Podkladem může být fotografie, ale mnohem lepší je opatřit si technické výkresy zachycující všechny 3 pohledy (anglicky Blueprint). Nákresy se načtou do pozadí odpovídajícího pohledu editoru, v C4D pomocí nabídky: *úpravy->konfigurovat->pozadí*. Jejich velikost a umístění je možné doladit ve stejném panelu.

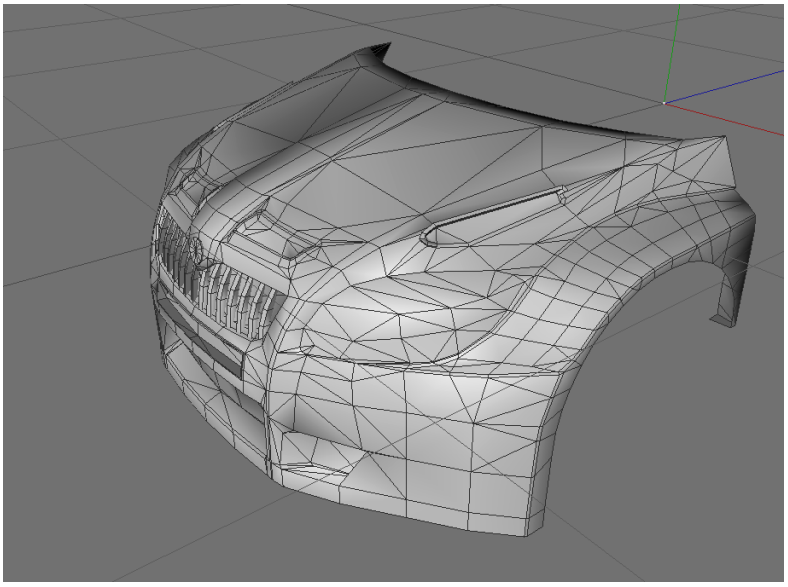
Postup tvorby modelu vychází z klasického přístupu, kdy z jednoduchých tvarů se postupně přechází k těm složitějším. Hlavní a nejsložitější částí modelu automobilu bylo vytvoření karoserie.

Obrázek 7 ukazuje již pokročilejší část tvorby blatníku. Na scénu je vložen objekt kruh a převeden na polygony. Body kruhu jsou přesunuty podle předlohy nejprve v pravém a posléze v předním pohledu. Při modelování je snahou nejlepší opsání objektů, aby se nejvíce shodovalo s jeho načtenou předlohou.

Do fáze, která je na obrázku, zbývá použít kombinaci nástrojů vytažení, nůž a zkosení hran. Tyto nástroje společně s posuvem/změnou velikosti jsou během celého procesu modelování používány nejčastěji.

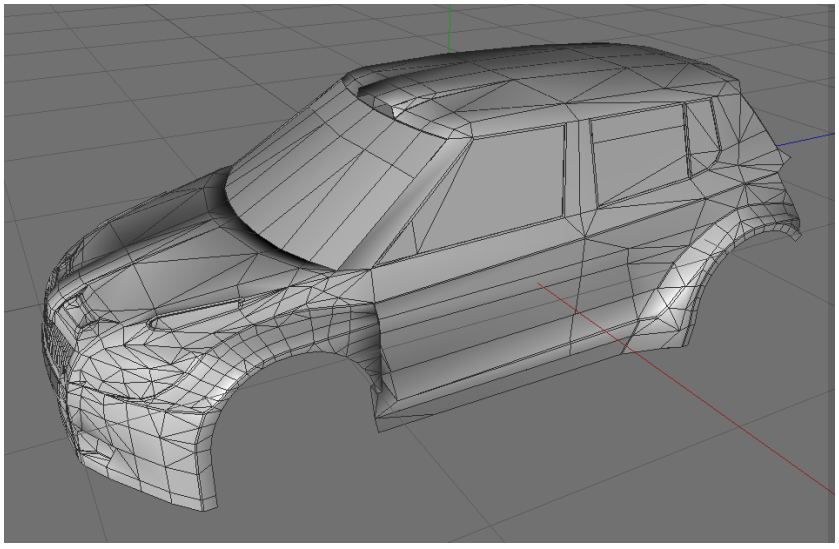
Při vytváření symetrických modelů (což je i tento případ) není nutné modelovat obě poloviny, ale stačí vytvořit pouze jednu část. Pro dopočítání symetrické strany obsahuje C4D hned několik možností - např. symetrie nebo zrcadlení. Nástroj symetrie v reálném čase vytváří symetrické polygony podle zvolené roviny, tzn., že okamžitě reaguje na provedené změny.

Po několika aplikování postupu *vytažení->rozřezání->posuv* na správné místo se model dostane do stavu na Obrázku číslo 10. Model je také zařazen pod objekt symetrie.



Obrázek 10 - Model s použitou symetrií

Při dalším postupu byly polygony zakončení předního blatníku vytaženy směrem dozadu a vznikl tak základ dveří. Hlavní část zadního blatníku opět tvoří převážně polygonový kruh. Vytažením horní hrany bočních polygonů vznikly okna a střecha. Po několika dalších úpravách pomocí nože a posuvu model dostává podobu zobrazenou na dalším obrázku.

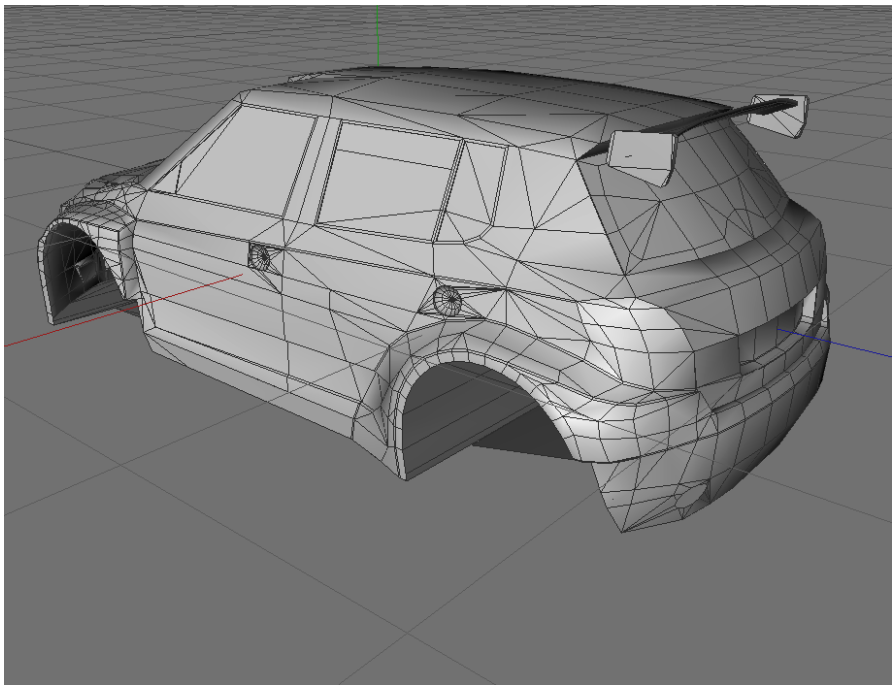


Obrázek 11 - Pokročilejší stav modelu

Poslední krok, který ještě zbývá, je vytvoření zadní části a doladění některých detailů. Detaily by měly být konzistentní ve všech částech karoserie, proto tvorba modelů je pracnější a náročnější na představivost a čas.

Vytvoření zadní části již není složité a jedná se opět o opakování popsané metody *vytažení->rozřezání->posuv polygonu.*

Za zmínění stojí postup vytvoření prolisu klik dveří. Pro jejich vytvoření je třeba mít 2 objekty - objekt dveří a objekt koule. Oba objekty se zařadí pod objekt Boleanovské operace. Tento nástroj pracuje v několika režimech s rozdílnými výsledky. Písmenka A, B označují objekty vložené pod Boleanovskou operaci, tudíž záleží na jejich pořadí. A spojit s B, B odečíst od A, A průsečík B, A bez B. Určitě vznikne jiný model, když se bude odečítat A od B nebo B od A. Jednotlivé názvy režimů již dostatečně popisují jejich použití. Konkrétní režim pro vytvoření klik byl B odečíst od A, kde A jsou dveře a B koule.



Obrázek 12 - Hotový model

Vnější plášť automobilu je dokončený, zbývá jen optimalizace a zarovnání normál. Příkaz Optimalizovat odstraní nepoužité a duplicitní body a polygony modelu.

Příkaz Zarovnání normál nesouvisí přímo s konstrukcí modelu, ale ovlivní směr normál polygonů. Většina enginů (pokud není řečeno jinak) nezobrazuje zadní stranu polygonu, a proto velmi záleží na směru normál polygonů. Pojem normála polygonu byl podrobně již vysvětlen v úvodní kapitole.

I když se tato práce nezdá být přílišná náročná, opak je pravdou. Předem je potřeba mít jasnou představu o tom, jak by měl konečný model vypadat. Ať už se vytváří objekt, který má reálný nebo zcela smyšlený základ.
Tato práce je převážně o kreativním myšlení, citu pro detail a značné trpělivosti.

2. 3 UV mapování, Textury

Po dokončení jsou modely připravené na další fázi a tím je texturování.

Aby počítač věděl, jak má texturu na modelu správně zobrazit, musí být schopen přiřadit prostorové souřadnice povrchu objektu plošným souřadnicím textury. Tento proces si lze zjednodušeně představit jako slepování papírové vystřihovánky, kde jednotlivé díly modelu se vystřihnou a poslepují dohromady, s tím rozdílem, že zde tento proces probíhá v opačném směru. 3D model se rozloží na jednotlivé polygony, které se umisťují na vhodné místo na texturu.

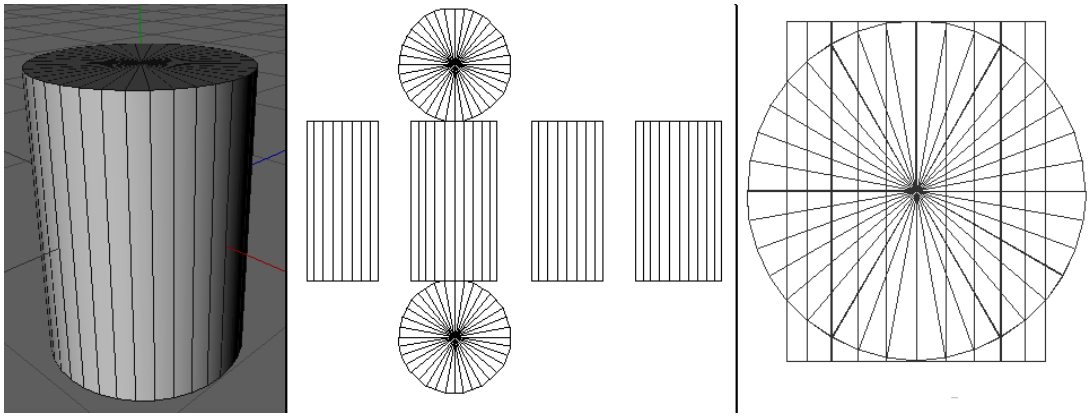
Pro všechny body modelu přibudou další 2 souřadnice. Kromě základních prostorových souřadnic X, Y, Z jsou zde ještě UV souřadnice (podle nich se tento proces nazývá UVmapování), které určují umístění bodu na daném místě textury.

Jelikož bylo potřeba otexturovat kolem 200 modelů, jednalo se o jednu z nejnáročnějších fází celého projektu. Nejprve byly pořízeny fotografie objektů v okolí, které bylo možné použít jako zdroj pro textury.

Takto pořízené fotografie byly importovány do modulu Cinemi – BodyPaintu. U objektů, u kterých nebylo možné získat počáteční data vyfotografováním, se našla vhodná náhrada z množství obrázků dostupných na internetu.

2. 3. 1 Modul BodyPaint

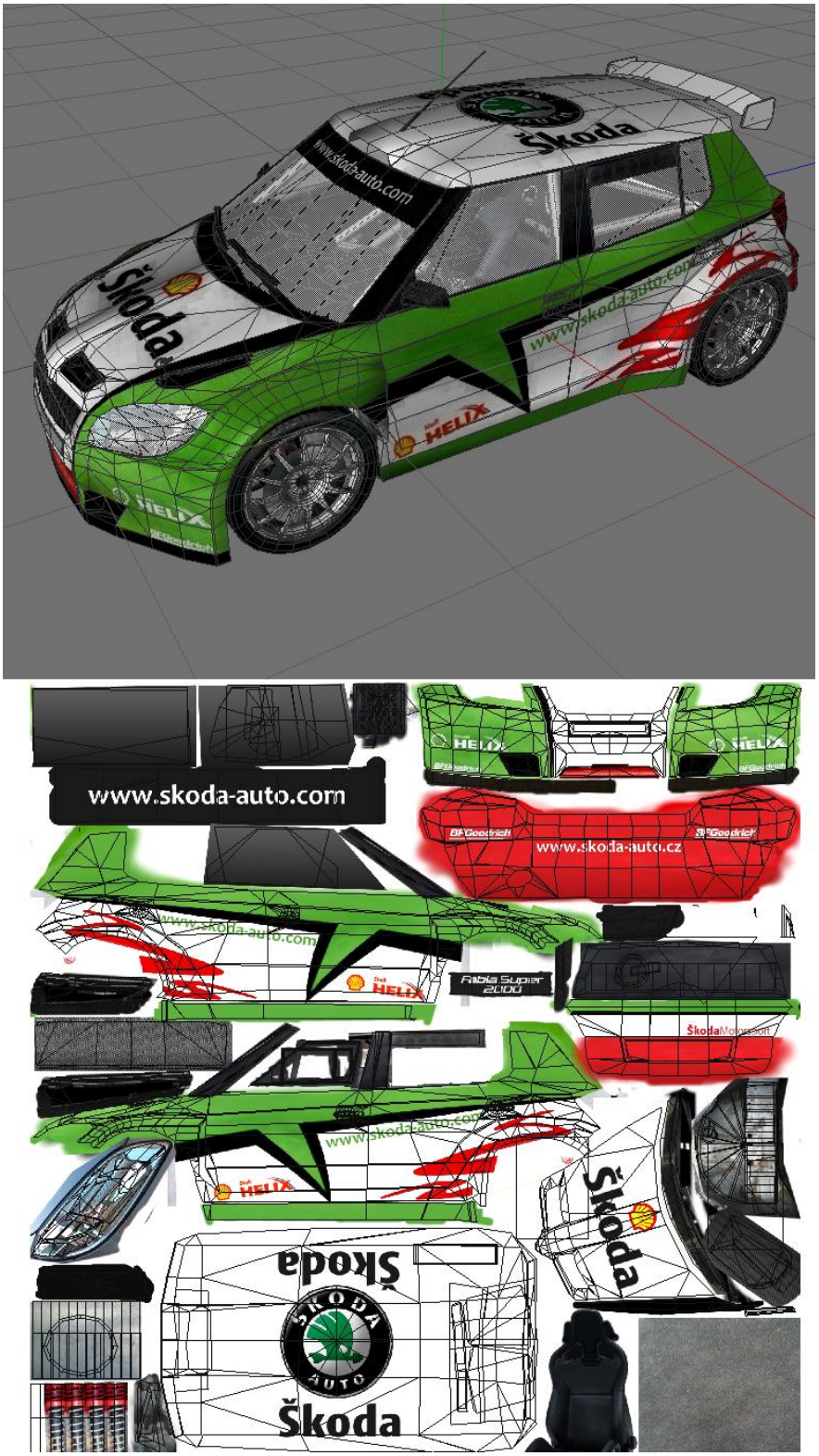
Pro vytvoření veškerých UVmap se používá integrované prostředí BodyPaint. Většina nástrojů je shodná s běžnými kreslícími aplikacemi s tím rozdílem, že se kromě vytváření textury pracuje s umístěním polygonů. Pro prvotní rozvinutí modelu do plochy se nabízí několik režimů: sférické, kubické, čelní, box a stažené. Je dobré zvolit nejvhodnější režim pro daný objekt, neboť polygony by měly být uspořádány na 2D plochu tak, aby nejlépe vystihly původní prostorový tvar a zároveň maximálně využily prostor textury.



Obrázek 13- Ukázka 2 způsobů počátečního UV rozvinutí válce

2. 3. 2 Problém s texturami

Snaha přiblížit se současným AAA titulům způsobila v konečném měřítku menší problémy. Původním záměrem byl pro každý model vytvořit značný počet detailů, aby se nejvíce blížil realitě. To zahrnovalo vytvoření color textur a normal textur o vysokém rozlišení. Jedna textura může zabírat na disku několik desítek MB, což při počtu kolem 200 modelů by byl značný problém. Jelikož je primárním cílem plynulost a rychlost načtení dat, původní záměr byl zamítnut. Vytváření textur o menším rozlišení, kde velikost textury nepřesáhla 1MB, se ukázalo jako lepší volba. Záměrem byla zajímavost a funkčnost textur ve všech velikostech, a aby byly od sebe navzájem dostatečně odlišitelné, ale ne příliš složité.



Obrázek 14 - Model s aplikovanou texturou a UVmapou

3 Herní enginy

V počítačovém světě představuje herní engine znovu použitelnou část hry. Poskytuje základní i pokročilejší funkcionalitu, která je víceméně společná všem hrám. Některé enginy se vyznačují opětovnou použitelností, zatímco jiné jsou ušité na míru konkrétní hře. Rozdíl mezi tím, co je ještě engine, a co už je hra, je poté velmi nejasný.

3. 1 Unity3D vs. UDK

Na videích s posledními herními enginy jako je Unity3D 3.5 nebo UDK 3 (Unreal Development Kit) si lze udělat představu, jak neskutečné obrazy se s jejich pomocí můžou vytvořit (vše v reálném čase).

Většina lidí znalých tématu považuje Unreal Engine za nejlepší herní editor, který vyšel. Engine se snadno naučí a je preferován mnoha profesionálními herními studii. Většina Unreal her, které se dočkají zveřejnění, se obvykle stávají nezapomenutelnými tituly.

Epic v průběhu let poskytl svoje prostředí pro více než 50 titulů, včetně: Splinter Cell a Rainbow Six 3 (Ubi Soft), Deus Ex, Harry Potter (Electronic Arts), Americká armáda (US Department of Defense), Gears of War. Pokud vývojářský tým použije toto prostředí pro svou aplikaci, má jistotu, že používá ten samý nástroj, který využívají teamy pro vývoj AAA titulů současnosti.

Může se nyní zdát jasné, který engine je v současnosti nejlepší?! Nicméně v posledních letech se začíná prosazovat další kvalitní engine Unity3D. Unity je poměrně nový. Již za krátkou dobu získal ohromnou popularitu a vývojářskou základnu. Je ještě snazší na použití než UDK, má velmi slušnou fyziku a poměrně jednoduché rozhraní a podporuje mnoho různých programovacích jazyků. Jakýkoliv grafický handicap oproti UDK umí Unity bohatě vynahradit. Unity3D neomezuje vývojáře v žádném směru, bez ohledu na to, co za žánr hry se rozhodne vytvořit.

Unity se zdá ideální volbou pro výrobu nezávislých her, stejně jako špičkových komerčních titulů.

S UDK je možné dosáhnout více realistickou grafiku a poskytuje jeden z nejlepších fyzikálních enginů. Z hlediska použitelnosti nabízí Unity mnohem více. Engine Unity je mnohem lépe zdokumentován a navzdory popularitě UDK je pravděpodobné, že Unity dosáhne popularity v příštích deseti až dvaceti letech.

Také je důležité neopomenout nejdůležitější parametr, který ovlivňuje konečnou kvalitu hry. Velmi záleží na dovednostech vývojáře. Lze vytvořit úžasné aplikace s oběma Enginy. Hlavní je využít své znalosti, dovednosti a odhodlání do dalšího vývoje.

3. 2 Unity3D

I když je Unity primárně herní engine, je natolik obecný, že vývojáře nenutí vytvářet pouze hry. Není problém vytvořit různé interaktivní aplikace, webové prezentace, vizualizace výrobků apod. Prázdné scény nejsou nijak specifické a lze je snadno nastavit pro daný záměr.

Dalším pozitivem této technologie je její nezávislost na cílové platformě. Vývojář může udělat hru pouze jednou a zveřejnit jí ihned na všech platformách. Tento způsob začal neskutečně ovlivňovat herní scénu a to, jak někteří nezávislí vývojáři vytvářejí své hry.

V současné době Unity podporuje kromě PC i handheld zařízení společnosti Apple Iphone a IPad a mobilní zařízení s OS Google Android. Unity dále podporuje konzole Xbox360, Nintendo Wii a nedávno vyšla podpora na Playstation 3.
Společně s multiplatformou vyniká možnost publikovat hry přímo na internetu
v internetovém prohlížeči. (Jedná se o první AAA herní engine v historii, který toho byl schopen.)

Již delší dobu zažívají velikou oblibu online hry hrané v internetovém prohlížeči. Tyto hry jsou velmi oblíbené díky možnosti hrát je všude, kde je připojení na internet a v podstatě na čemkoli od osobního počítače přes tablety až po chytré telefony. Tyto hry již nejsou jednoduché hříčky ve flashy jako ještě před několika lety, ale některé se vyrovnají klasickým hrám na PC a na konzolích.

Ke spuštění takové hry stačí mít nainstalovaný tzv. Unity WebPlayer (plugin do prohlížeče), který obstará celý chod 3D hry v okně browseru. Žádná administrátorská práva pro instalaci nejsou potřeba. Navíc od poslední verze Unity3.5 přichází Google Chrome s podporou Web Playeru automaticky, bez nutnosti jakékoliv dodatečné instalace.

Další obrovskou výhodou je podpora hry více hráčů (tzv. multiplayer mód), což byl jeden z cílů této práce.

Dnes má Unity technologies více než 100 000 registrovaných uživatelů po celém světě včetně předních společností jako je Cartoon Network, Electronic Arts a Nasa, velkých i

malých, nezávislých studií, studentů a fandů. Unity se stala velmi oblíbeným nástrojem pro rozvoj interaktivního 3D obsahu Webu, mobilů a konzolí.[4]

3. 2. 1 Licence

Unity je prodávána v komerční distribuci UnityPro nebo je dostupná zdarma pod licencí Unity free. Pro vývojáře to znamená, že můžou využívat většiny výhod komerčních distribucí pro komerční i nekomerční účely.

Pokud vývojář chce vytvářet hry na nejvyšší úrovni, určitě se bez placené verze neobejde. Unity Pro obsahuje nejmodernější vizuální technologie jako Blur a Motion Blur effect, HDR rendering plus mnohé další nástroje zlepšující celkový vzhled a výkon aplikace.
Pro účely této práce bohatě postačily funkce poskytující free verze.

3. 2. 2 Editor

Editor má tři různé možnosti rozvržení, kde se každá liší pouze uspořádáním oken. Pro vývoj bylo využito nastavení „2 by 3“. Dominantním prvkem jsou zde 2 okna s 3D perspektivním náhledem scény.

Do horního okna *Scene* se umísťují všechny vkládané objekty a celkově se zde tvoří výsledná scéna.

V dolním okně *Game* se zobrazuje pohled z hlavní kamery na výslednou scénu. Tlačítka play v toolbar přepne *Game* okno do tzv. play modu, který spustí prohlíženou scénu bez nutnosti kompilace. Po spuštění play modu se okamžitě zobrazí okno debuggru, kde se v reálném čase zobrazují chyby nebo varování. Během testování je možné hru pozastavit a projít chybová hlášení, měnit nastavení proměnných nebo umístění objektů.

Během playmodu je možné určité objekty ve scéně libovolně upravovat, aniž by jejich pozice, rotace nebo odstranění byly uloženy do výsledné scény. Dají se nezávazně graficky testovat různé nápady a návrhy, které nakonec není třeba ukládat ani vracet zpět.

Hierarchy zahrnuje všechny objekty použité pouze v aktuální scéně. Některé z nich jsou přímo instance souborů jako 3D modely a jiné jsou instance prefab. Zařazením objektu pod jiný objekt se docílí nejen přehlednosti, ale také vnořený objekt bude od svého rodiče přebírat transformační nastavení – pozici, rotaci a zvětšení.

Okno *Projekt* obsahuje veškeré objekty, skripty, prefab v celém projektu.

Pro přidání souboru do projektu, stačí obyčejné přetažení libovolného souboru do projektového okna nebo se může použít volby z menu: *Assets/Import new asset*. Díky složkám lze vytvořit přehlednou stromovou strukturu všech součástí projektu.

Okno *Inspektoru* zobrazuje podrobné informace pro zvolený prvek včetně všech připojených komponent. V tomto okně se dá nastavit pozice, velikosti a rotace objektu.

Nejdůležitější funkce tohoto okna je správa komponent. Nový komponent nebo Script se aplikuje na objekt přetažením v hierarchy, nebo na jeho grafickou reprezentaci v okně Scene. [4]

3. 2. 3 Multiplatformní specifikace

Dříve byl velký problém, pokud měl autor za cíl publikovat aplikaci na několika platformách. Byla nutná konverze kódu pro daná specifika a nároky platformy, což nebylo zrovna snadné a efektivní.

Unity 3 umožňuje zaměřit se na všechny platformy v rámci jednoho vývoje.
Unity je schopná zajistit, aby libovolný kód pracoval v různých zařízeních. Provádí to extrahováním většiny platformních rozdílů. Je-li potřeba upřesnit nastavení, stačí použít příkaz *# ifdef*, což bude značit specializovaný kód pro danou platformu.

Jelikož různé platformy mají různé výpočetní schopnosti, je možné určit např. komprese textur a nastavení rozlišení pro každou platformu zvlášť. [4]

3. 2. 4 Formáty

Unity podporuje širokou škálu aplikací a formátů pro import skriptů, 3D modelů a textur:

Plná podpora: Maya (.mb/.ma), 3D Studio Max (.Max), Gepard 3D (.Jas), Cinema4D (.C4d), Blender, Carrara, COLLADA, Lightwave, Autodesk FBX (.DAE) a XSI 5. x.

Částečná podpora: SketchUp Pro, Wings 3D, 3D Studio (0,3 ds), Wavefront (obj.), AutoCAD DXF (DXF.).

Pro účely práce je nejčastěji využíván formát C4D, který unity nativně podporuje. Dále export přes formát FBX6. Formát C4D je výhodnější kvůli okamžitému promítnutí změn přímo do enginu. Změna provedená v C4D se po uložení okamžitě promítne do Unity. Bez nutnosti aplikaci vypínat.

Tento přístup značně urychlil vývoj, jelikož nebylo nutné exportovat soubor při každé sebemenší změně. Celý proces znovunačtení je spolehlivý a velmi rychlý. To samé platí i pro manipulaci s texturami. Při uložení je textura automaticky komprimována a upravena i v enginu.

3. 2. 5 Materiály

Pro nejvěrohodnější reprezentaci svého reálného obrazu musí mít 3D objekty vhodné materiály.

Unity má velké množství vestavěných materiálů od jednoduchého diffuse až po velmi složité materiály simulující kapaliny. Jejich variabilita a vzájemná kombinovatelnost dává velký prostor pro fantazii a rozvoj vlastní kreativity.
Diffuse shader je nejzákladnější a nejpoužívanější. Jeho typickým použitím je neodrazový materiál jako dřevo nebo plast. Průhledné shadery jsou určeny pro průhledné nebo poloprůhledné objekty. Tyto shadery využívají alfa kanálu, který ovlivňuje konečnou úroveň průhlednosti objektu. Tento shader perfektně funguje pro skleněné plochy.
Reflexní shadery jsou určeny např. pro chrom, kapalné povrchy nebo monitory. Počet odrazů se počítá z alfa kanálu základní textury. [4]

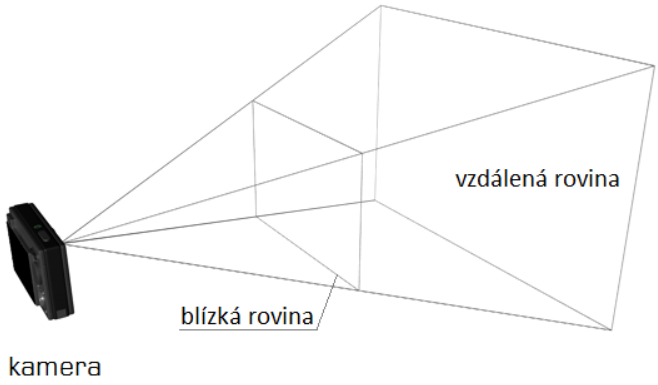
Většina materiálů použitých v aplikaci jsou Unity poskytované základní shadery poupravené, aby splňovaly aktuální nároky na daný materiál. Nejčastěji je použito diffuse kanálu v kombinaci s normál mapou. Při použití takového materiálu s vhodným nasvícením lze dosáhnout dostatečné úrovně detailů.

3. 2. 6 Kompatibilita staršího HW

Mnoho potenciálních uživatelů může používat starší grafický hardware a ovladače. Hry jsou rok od roku náročnější a mnoho hráčů nesplňuje ani minimální podmínky pro spuštění některých her. Unity má solidní podporu téměř všech Hardwarových / Softwarových kombinací jak v DirectX tak openGL, nabízí i řadu technologií pro zlepšení efektivity vykreslování.

Snad nejvýznamnější zvýšení výkonu je použití Occlusion Culling. Cílem Occlusion Culling je rychle určit části prostoru, které nejsou ze současné kamery viditelné. Tím se vyřadí všechny objekty v nich obsažené.

Například se nebude zatěžovat výpočtem stovky kamenů, když jsou schované za zdí. Unity také používá tzv. Frustrum ořezávání, viz Obrázek 15.



Obrázek 15 - Princip rovin kamery pro Frustrum ořezávání

V zorném poli jsou 2 roviny (blízká a vzdálená), které jsou kolmé ke směru umístění kamery. Úhel pohledu kamery tvoří jakýsi komolý jehlan. Ten vymezuje vykreslování objektů na objekty, které jsou viditelné v zobrazení úhlu kamery. Mimo úhel jsou objekty, které se nezpracovávají.

3. 2. 7 Prefab

Jedním z nejpoužívanějších prvků při vývoji je prefab. Prefab značí game objekt, který lze vytvořit pouze v Unity (projektovém okně).

Soubor typu prefab je realizací určitého způsobu uložení programového kódu a zdrojů do společného, komprimovaného souboru. Zdrojem může být téměř cokoliv včetně modelů, obrázků a zvuků. Jeho vlastnosti jsou napříč aplikací stejné a přístupné z kteréhokoliv místa s možností mnohonásobného použití v prostředí Unity (s přihlédnutím na hardwarové vybavení). Proto mají hlavní využití při standardizaci. Tyto objekty jsou navrženy tak, aby co nejvíce zpřehlednily a usnadnily použití a úpravu u velkého počtu instancí. Všechny vytvořené instance ve scéně jsou spojeny s původní Prefab a jsou v podstatě jejími klony. Pokud se provede nějaká změna v rodičovském objektu, je aplikována i na všechny potomky, bez ohledu na to, kolik instancí v projektu existuje. [4]

3. 2. 8 Asset Management

Při vývoji se dříve či později dostane každý projekt do bodu, kdy jeho velikost nabude takových rozměrů, že není snadné najít vše, co se aktuálně hledá. Pro udržení vysoké

přehlednosti projektu existuje v Unity Asset Management. Asset Management je projektový vyhledávač a nesnadněji ho lze přirovnat k průzkumníku Windows. Objekty je možné hledat klasicky podle jména nebo podle typu. Seznam nalezených objektů se zobrazí v hierarchy a navíc zůstanou tyto objekty vybarveny v *scene view*.

3. 3 Programovací jazyky

Unity používá pro kompilaci Mono platformu, což je multiplatformní Open Source verze NET. Jednou z výhod .NET/mono je funkčnost s jakýmkoliv jazykem, který může sestavit CIL.

Unity podporuje celkem tři skriptovací jazyky: C#, JavaScript a Boo, což je obdoba jazyka python. Všechny tyto jazyky podporují .NET knihovny, které umožňují vytvářet databáze, vytváření sítí, XML atd.

Editor obsahuje velké množství vizuálních pomocníků, takže např. veřejné proměnné definované ve skriptu jsou zobrazeny v okně inspektora. Tímto způsobem je jednoduše možné upravit proměnné pomocí táhnutí myši na posuvníku.

Obrázek 16 - Ukázka vizuálního pomocníka veřejné proměnné

Tímto vizuálním způsobem se lze odkazovat na nepřeberné množství parametrů a typů objektů, díky čemuž je unity velmi flexibilní a snadno použitelná.

CIL (Common Intermediate Language) je v informatice nejnižší člověkem čitelný programovací jazyk. Při kompilování .NET programovacích jazyků je zdrojový kód přeložen do CIL kódu (nepoužívá se platformě nebo výpočetně specifický objektový kód). CIL je procesorově a zároveň platformě nezávislý soubor instrukcí, který může být realizován v jakémkoli prostředí podporující CIL.

Pro programování je možné použít plně integrované vývojové prostředí MonoDevelop. Samozřejmostí je podpora všech zmiňovaných jazyků a platforem. Pomocí MonoDevelop lze ladit skripty jako v každém jiném softwaru. Je schopen vytvářet body přerušení,

krokování řádek po řádku a kontrolu hodnot. Pokud někomu MonoDevelop nevyhovuje, má možnost použít Visual Studio, se kterým je editor unity také plně synchronizováno.

Poslední ze škály programovacích nástrojů je profiler. Ten zobrazuje a zaznamenává údaje o aktuálním výkonu HW v reálném čase. Okno Profiler zobrazí data v časové ose, takže není obtížné vyhledat snímky nebo oblasti s nejvyšší zátěží.
Uvedené statistiky zahrnují CPU, grafický HW a paměť. V rámci těchto hlavních kategorií je uveden každý proces, který využívá těchto zdrojů. Například CPU zpracovává následující procesy: vykreslování, skripty, fyzika a další.
Profiler zobrazí podrobné informace o vybraném rámu, což velmi urychluje finální optimalizaci výkonu, neboť se primárně zaměří na ty části skriptu, které spotřebovávají nejvíce času. Porovnáním výsledků profilování před a po zjištění změny kódu se zjistí, zda provedená změna měla pozitivní/negativní vliv na výkon.[4]

3. 3. 1 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk. JavaScript se často zaměňuje s Javou. Java v názvu jazyka je však pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe.

Nejběžněji ho lze potkat jako interpretovaný programovací jazyk pro WWW stránky, často zapisovaný přímo do HTML kódu. Velkou výhodou je jeho jednoduchost.

Program v JavaScriptu se obvykle překládá na straně klienta, tzn. až po stažení WWW stránky do prohlížeče, na rozdíl od ostatních jiných interpretovaných programovacích jazyků (např. PHP a ASP), které většinou zpracovává server.

Tento přístup má své nesporné výhody. Např. reaguje na uživatelem iniciované akce bez jakéhokoli síťového přenosu. Pokud uživatel něco napíše do textového pole, není nutné, aby byl tento formulář posílán serveru, zkontrolován a poslán zase zpět. Vstup je ověřen přímo klientovou aplikací.
Za určitých podmínek je však možné JavaScript použít i na straně serveru.

Z klientského překladu plynou i jistá bezpečností omezení. JavaScript nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele.

JavaScript je závislý na prohlížeči (uživatel ho může vypnout). V různých verzích prohlížečů nemusí skript vždy korektně fungovat.

Další vlastností je case sensitiv, záleží na velikosti písmen v zápisu.

JavaScript patří do skupiny interpretovaných jazyků, tudíž se nemusí kompilovat. Interpretace kódu má proti překladu do strojového kódu a přímému spuštění výhody:

- **Rychlost vývoje**: Zejména u větších projektů je při interpretaci rychlejší cyklus *editace – spuštění – oprava chyb*.
- **Laditelnou:** Stav interpretu je jednoznačně definován, jeho činnost přesně odpovídá zdrojovému kódu. Nebývají použity optimalizace, které mohou původní strukturu kódu zakrýt nebo pozměnit.
- **Kompatibilita**: Stejný program může být bez problémů spuštěn na mnoha cílových architekturách, jazyk obvykle zakrývá rozdíly mezi různými operačními systémy.
- **Správa paměti**: Při běhu interpretovaného programu nemusí být proměnné vázány na fyzickou adresu v operační paměti, proto je možné přemapování proměnných, což může zabránit fragmentaci paměti. [2]

3. 4 Unity a Networking

Unity již ve free verzí obsahuje mnoho síťových funkcí. Patří mezi ně WWW rozhraní, možnost zahrnout JavaScript nebo AJAX, podpora NET knihovny. WWW rozhraní Unity umožňuje přístup na webové stránky a webové služby. To se provádí při jednom volání funkce, která spustí stahování z určeného webu. WWW funkce může být použita například při načítání textur nebo high skóre.
Je-li aplikace exportována do formátu Webplayer, pak prohlížeč může komunikovat bez problémů s JavaScriptem aplikace. To znamená, že webové stránky mohou volat funkce uvnitř obsahu WebPlayeru a WebPlayer naopak může volat funkce webových stránek.

3. 4. 1 Server

Serverem je obecně označován počítač, který poskytuje nějaké služby nebo počítačový program, který tyto služby realizuje.

Server poskytuje služby klientům, proto se tento model označuje *klient-server*. Služby mohou být nabízeny v rámci jednoho počítače (lokálně) nebo více počítačům pomocí počítačové sítě. Takovou typickou síťovou serverovou službou je uchování a nabízení webových stránek a nebo DNS, e-mail atd.

Hlavní rozdíl mezi osobním počítačem a serverem je vybavenost programy (software). Současné operační systémy jsou obvykle univerzální a mohou sloužit jako osobní počítač i jako server. Rozdíl je v jejich nastavení, kdy u osobních počítačů je preferována interaktivita (počítač rychle reaguje na požadavky uživatele) a u serverů se klade důraz na škálovatelnost (schopnost dosažení co nejvyššího výkonu). To může zahrnovat velmi rychlé připojení k síti a vysokou I / O propustnost. [2]

Druhy serveru:

- webový server – především v síti Internet poskytuje WWW stránky;
- souborový server – slouží např. v podnikové síti jako centrální úložiště dat (dokumentů);
- databázový server – slouží jako úložiště strukturovaných dat (databází);
- aplikační server – počítač specializovaný na provoz nějaké aplikace;
- herní server – nabízí hraní her s více hráči (multiplayer).

3. 4. 2 Klient

Jedná se o aplikace nebo systém, který přistupuje ke vzdálené službě na jiném počítačovém systému, označovaném jako server. Nejčastěji se tak děje prostřednictvím počítačové sítě. [2]

3. 4. 3 Unity Master Server

Master server je samostatná služba, která se používá pro vytvoření vlastních serverů a jejich zpřístupnění pro široké publikum. Master server může povolit komunikaci přes bránu firewall nebo domácí síť pomocí techniky zvané NAT punchthrough.
Každá jednotlivá instance hry je zaregistrována do tabulky. Pokud tabulka neexistuje, je dynamicky vytvořena. Když se host připojí, pošle zprávu se svým identifikátorem obsahující informace o něm. IP a port si master server uloží do databáze hostů pro pozdější komunikaci.

Tabulka nebo seznam hostitelů je stažena z databáze a odeslána klientovi.

Pokud master server zjistí, že klient žádá hostitele o data pro herní server a server má stejnou IP adresu, pak používá soukromou adresu serveru namísto vnější. To je řešení případů, kdy klient a server jsou na stejné lokální síti, pomocí stejného routeru s NAT adresou.

I když budou mít stejnou vnější adresu, mohou se k sobě připojit a komunikovat spolu.
Unity Technologies uvolnilo kompletní skript master serveru zdarma, takže ho každý může libovolně používat.

3. 4. 4 Network view

Jedná se o základní prvek při stavbě online her v Unity. Network view plní úkoly: synchronizaci dat mezi uživateli, sledování a zobrazování ID scény.
Synchronizovat lze vždy jednu komponentu - souřadnici v prostoru, animace, Rigid body nebo skript.
Synchronizace dat může probíhat ve 3 režimech: *vypnuto, pouze důležitá komprimovaná data, všechny data*. Při režimu důležitých komprimovaných dat se posílají pouze v případě, že došlo k nějaké změně, pokud k ní nedojde, nic se neodešle.

Pokud je synchronizovaným objektem skript, musí být proměnné výslovně serializovány ve skriptu. To se provede pomocí funkce OnSerializeNetworkView. Pokud nejsou přímo odesílána data, ale je Network view využíván pro nepřímou komunikaci pomocí RPC funkce, je možné synchronizaci úplně vypnout.

Při tomto režimu jsou pakety vždy obdrženy v pořadí, v jakém byly odeslány.

Pokud jsou během přenosu některé pakety ztraceny, dojde k jejich opětovnému zaslání. Všechny pozdější pakety jsou ve frontě, dokud nebude přijat paket dřívější.[2]

3. 4. 5 RPC-Remote Procedure Call

PRC dovoluje volání funkce na vzdáleném PC. V principu se příliš neliší od volání normální funkce, ale i přes to má určité rozdíly. Počet parametrů posílaných RPC funkcí je prakticky neomezený (stejně tak je libovolný i datový typ). Musí se ale brát v úvahu, že čím víc parametrů se odesílá, tím náročnější bude přenos. Dalším rozdílem je, že musí být zadáno, koho RPC volá. Může to být výzva pro všechny připojené stroje, pouze server nebo konkrétní uživatel.
RPC volání bývá ponecháno ve vyrovnávací paměti serveru, což umožňuje snadný způsob synchronizace nově připojených klientů. Například po připojení host pošle všem RPC informaci o svém připojení. Server si ho zaregistruje a z vyrovnávací paměti opět pomocí RPC mu pošle všechny předchozí výzvy. Nový uživatel je rázem na stejné úrovni jako ostatní hráči. [2]

3. 4. 6 OnSerializeNetworkView

Tato funkce se používá pro synchronizaci proměnné ve skriptu. Provede se zcela automaticky po připojení nového klienta nebo při její změně.
Pokud jsou proměnné synchronizovány mezi hráči, musí se určit, kdo ji momentálně odesílá a kdo ji přijímá.

3. 5 Fyzikální a kolizní systém

Ve většině her se vyskytuje řada entit, které mezi sebou nějakým způsobem interagují. Tyto entity představují vozidla, postavy, projektily atd.
Detekce kolizí je jeden ze základních kamenů většiny her. Bez ní by objekty procházely jeden druhým a bylo by nemožné se pohybovat v herním světě. Zatímco bez kolizního systému se žádná hra nemůže obejít, tak simulace fyziky už není nezbytnou součástí.

Proč tedy vlastně používat fyziku? Lze jednoduše odpovědět jedním slovem: předvídatelnost. V animovaném světě je všechno pevně dáno. Oproti tomu fyzikální simulace je nepředvídatelná. Navíc ani simulace s naprosto shodným vstupem nedopadne pokaždé stejně kvůli numerickým nepřesnostem výpočtu. Fyzikální zákony jsou sice fixní, ale ve hře je možné řadu věcí ovlivnit. Například se může měnit velikost gravitace apod. Při použití fyzikální simulace se hra chová mnohem věrohodněji a blíže realitě.

Simulací fyziky ve hrách se téměř vždy myslí dynamika pevných těles (Rigid Body Dynamics). Pravdou je, že výpočet fyziky a detekce kolizí spolu velmi úzce souvisí. Nalezené kolize se musí vyřešit jako součást simulace fyziky.

Dnes už prakticky žádná firma nepíše vlastní fyzikální systém. Převážná většina enginů používá pro detekce kolizí a simulace fyziky hotová řešení třetích stran, které nabízejí detekci kolizí i simulaci fyziky v jednou uceleném balení. Ne jinak tomu je v Unity, která využívá NVIDIA PhysX fyzikální engine.

3. 5. 1 Fyzikální engine PhysX

Tento engine původně vyvinula firma AGEIA, která vyráběla akcelerátory fyziky (PPU - Physics Processing Unit) pro svůj fyzikální engine PhysX. V únoru roku 2008 ji pak NVIDIA koupila. NVIDIA se rozhodla dále speciální akcelerátory nevyvíjet a soustředila se na podporu tohoto enginu ve svých nových grafických kartách. PhysX je možné akcelerovat pomocí CPU i pomocí GPU.

Pro spuštění PhysX je potřeba pouze grafika Nvidia obsahující CUDA jádro. V případě použití grafiky z dílen AMD jsou všechny důležité problémy u PhysX počítány procesorem. Software pod tímto enginem běží i na počítačích bez GeForce. Hardwarová akcelerace se používá pouze pro rozšířené funkce, jako jsou částicové efekty (tekutiny, dynamické kouře, trosky a střepiny z výbuchu) a efekty plátna (oblečení a vlasy, vlajky). Tyto efekty se bez hardwarové akcelerace emulují v CPU. PhysX tímto dává výhodu vlastníkům NVIDIA karet. [2]

Inicializace fyziky v Unity probíhá vytvořením prvku rigidbody, čímž těleso získá své základní vlastnosti. Vedle fyzického tvaru, který je dán tvarem modelu, je to hmotnost, pružnost a drsnost (vliv tření), gravitace atd.

Jak bylo popsáno výše veškerou alfou a omegou při pohybu v herním světě je výpočet kolizí mezi objekty. Aby na sebe dva Rigidbodies působily, je potřeba připojit kolizní systém.

V unity se dá použít velká řada kolizních efektorů, zde stačí zmínit pouze ty nejpoužívanější - Box collider, Sphere collider, capsule collider, mesh collider, wheel collider.
U *box collideru* je testovaný objekt uzavřen do krabice, u které je testována kolize. Funguje skvěle pro dveře, stěny, krabice apod. Jedná se o jednu z rychlých a vývojáři velmi oblíbených metod.

Další dva *sphere* a *capsule collider* jsou si velmi podobné. Objekt je obklopen sférickým kolizním systémem. Liší se v parametru výšky u capsule collideru, který koule nemá.

Mesh Collider analyzuje objekt a vytvoří Collider na základě jeho sítě. Tento způsob detekce kolizí je přesnější než pomocí primitiv hlavně u složitých objektů.

3. 5. 2 Wheel Collider

Pro účely práce nejzajímavější je nástroj *Wheel Collider*. Jednou z podstatných součástí simulace fyziky vozidla jsou bezesporu jeho kola. Musí být schopny přenášet výkon na vozovku, v závislosti na rychlosti ovlivňovat přilnavost a s tím také související tření při brzdění. Toho je dosaženo pomocí wheel Collideru. Jedná se o speciální fyzikální člen vyvinutý pro pozemní vozidla. Zahrnuje detekci kolizí, fyziku kola, výpočet skluzu na bázi tření pneumatiky s vozovkou atd.

Během vývoje bylo třeba nalézt vlastní nejvhodnější nastavení hodnot tohoto colideru. Základní parametry, které je možné ovlivnit, jsou dále uvedeny. *Střed (Center)* značí střed kola v jeho lokálních souřadnicích. *Radius* je poloměr simulovaného kola. *Velikost odpružení (Suspension Distance)* neboli maximální vzdálenosti y- ové osy tlumičů, prakticky určuje, jak vysoko bude vůz nad kolem resp. Vozovkou. Celková výška nad vozovkou bude součet Suspension Distance + radius. *Pérování (Suspension spring)* v sobě zahrnuje několik parametrů, které nastavují jak rychlé a jakou silou bude probíhat tlumení. Parametry jsou následující:

Odpružení (Spring) definuje sílu potřebnou k maximálnímu stlačení „pružiny" tlumičů. U tohoto parametru platí závislost: čím větší bude jeho hodnota, tím snazší bude dosáhnout mezní hodnotu.

Tlumení (Damper) určuje rychlost tlumení. Čím vyšší hodnota, tím pomalejší je tlumení.

Předpětí (Target Position): Parametr předpětí ukazuje, jak moc je pružina na počátku předepnuta. Pokud je na 0, pružina je volná a pokud na 1, je pružina již plně stlačená.

Hmotnost (Mass) je hmotnost jednoho kola. *Přímá a boční přilnavost (Forward/Sideways Friction)* definuje velikost čelní a boční třecí síly kola.

3. 5. 3 Křivky Tření

Křivky tření představují nejsložitější část nastavení Wheel collideru. Chování tření pneumatik v Unity popisuje křivka, která je určena několika parametry a je potřeba najít vyvážený poměr mezi nimi.

Existují hned dvě křivky, definující síly tření při pohybu dopředu a do strany. V obou směrech se nejprve stanoví velikost skluzu Sk (na základě rychlosti pneumatiky a jejího otáčení). Pak je tato hodnota skluzu použita pro zjištění síly, kterou pneumatika působí na kontaktním místě. Kontaktní místo se stanoví tažením paprsku ze středu kola dolů po ose Y.

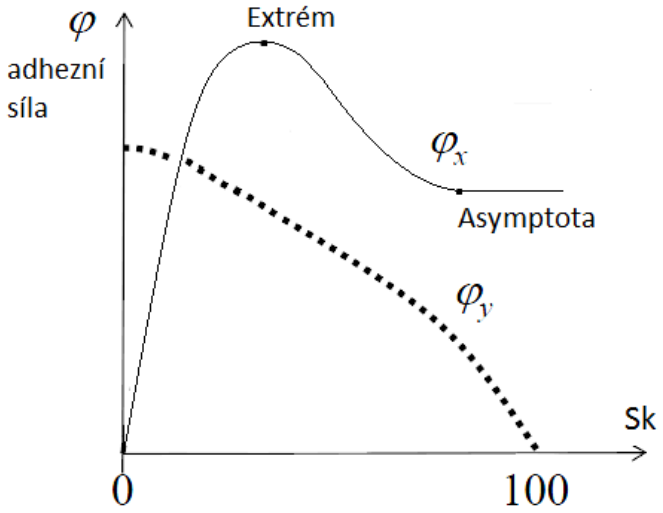
Velikost skluzu: $S_K = \frac{v - v_K}{v} \cdot 100$

v.... rychlost jízdy

v_K.. rychlost otáčení kola

V reálném světě se pneumatika chová tak, že při nízké rychlosti působí vysoké síly, protože guma kompenzuje skluz. Později, když skluz nabude vyšších hodnot, jsou přilnavostní síly sníženy a pneumatika začne klouzat.

Proto křivky tření mají následující tvar:



Obrázek 17 - Křivky tření prvku wheel collider

Křivka je aproximována ve dvou částech. První část vede z (0, 0) do Extrému, kde je tangenta křivky nulová. Druhá část se pohybuje od Extrému až do asymptoty, kde tečna je opět nulová. První oblast tak určuje fázi, než pneumatika proklouzne a druhá, kdy začne prokluzovat.

Souřadnice bodů Extrému a asymptoty lze měnit, čímž se změní tvar křivky a dosáhne se různého chování pneumatiky. Při vhodném nastavení tvaru křivky se může simulovat jízda na ledu, písku, mokré vozovce apod.

4 Praktická implementace

Hry jsou multimediální systémy, skládající se z mnoha subsystémů a sladit všechny prvky do jednoho funkčního prvku byla opravdu výzva.

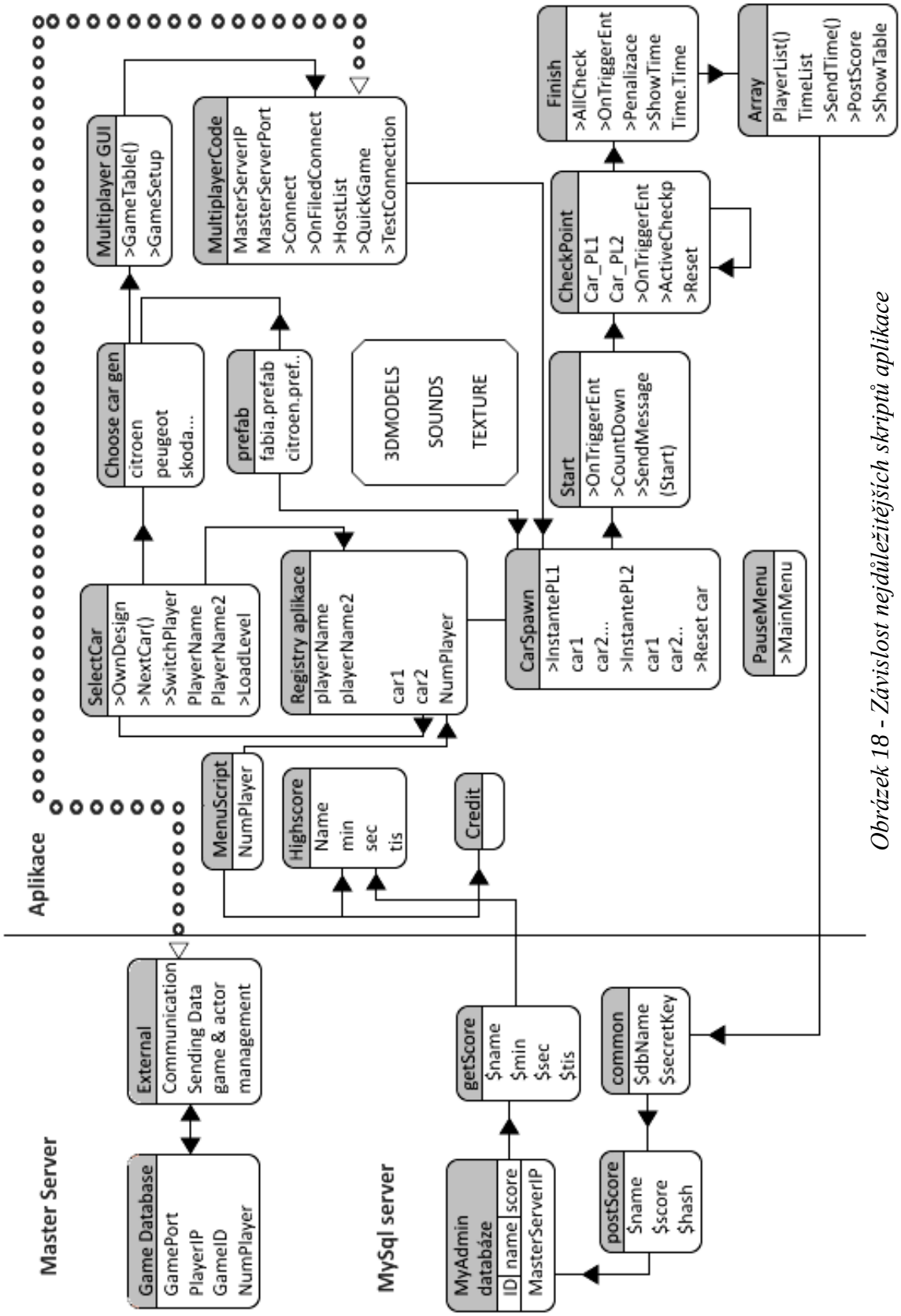
4. 1 Návrh aplikace

Přehled nejdůležitějších prvků aplikace a jejich vzájemná propojení je zjednodušeně popsáno na Obrázku 18. Z obrázku je patrné, že hru tvoří 3 větší celky - Master Server, MySql server a Aplikace.

Kromě skriptu pro deformaci vozu a konstrukce master serveru jsou všechny použité skripty a modely můj vlastní výtvor.
Co všechno jsem prakticky implementoval do hry?

- Veškerou herní logiku;
- fyziku vozidel a detekci kolizí;
- možnost načtení vlastního designu na libovolný vůz pro oba hráče;
- systém výběru vozů založený na prefab;
- online multiplayer – klientskou část;
- detekci vstupních zařízení. Hra nativně podporuje klávesnici, myš a možnost přidání dalších zařízení.
- Herní svět zahrnující základní objekty atd

V následujících kapitolách jsou rozebrány vybrané skripty/subsystémy podrobně. Popis probíhá především z hlediska funkčnosti a nezabíhá příliš do detailů implementace.



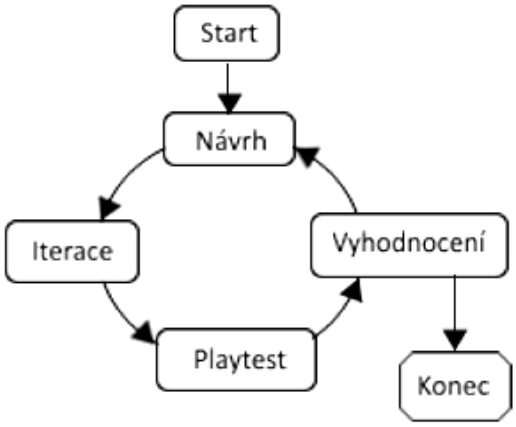
Obrázek 18 - Závislost nejdůležitějších skriptů aplikace

4. 2 Game Design

Prvním krokem při přenesení hry z myšlenky do skutečného světa je jasná představa o tom, jak má hra vypadat.

4. 2. 1 Iterační postup vývoje

Stejně jako u běžného programování se nejprve navrhne základní schéma (principy) hry. Principy se implementují a posléze se zkouší, zda všechno správně funguje. Po zahrání hry přichází rozhodnutí, co je dobré a co potřebuje změnit. Po dalším kole testování přichází rozhodnutí, je-li již aplikace hotova, nebo se musí udělat nějaké změny. Pokud je hra dostatečně dobrá, pak již není co řešit. Jestli během testování vyšly najevo nějaké chyby v herním mechanismu, vrátí se vývoj zpět k části návrhu. Najde se způsob k vyřešení problémů, implementují se tyto změny a pak se hra zhodnotí znovu. Tento postup se opakuje tak dlouho, dokud není hra hotová.



Obrázek 19 - Schéma iteračního postupu vývoje

Předešlý postup je založen na experimentální vědecké metodě:

1. Učinit pozorování. („Má zkušenost s hraním her mi ukázala, že určité typy mechanismů jsou zábavné.“)
2. Učinit hypotézu. („Napíši vlastní soubor pravidel, o kterých si myslím, že učiní hru zábavnou.“)
3. Provést experiment k dokázání či vyvrácení hypotézy. (Hraní hry)
4. Vyložit výsledky experimentu, zformovat nový soubor pozorování. Vrátit se k prvnímu kroku.

Obecně platí, že čím vícekrát se vývoj iteruje, tím lepší finální hra bude. [5]

4. 2. 2 Základní koncept

Vývoj počítačové hry se obecně skládá z 3 částí:

- **Návrh prostředí,** ve kterém se hra bude odehrávat.
- **Stanovení pravidel**: Jak hra začne? Pravidla pro rozvoj hraní. Když hra začne, co může hráč dělat a co se stane, když to udělá?
- **Pravidla pro ukončení hry:** Co, pokud vůbec, způsobí konec hry? Pokud má hra cíl (výhru nebo prohru), jakým způsobem je určen?

  Cíl hry (zisk) je jádrem veškerého herního snažení. Hra se hraje pro dosažení zisku. Ten může nabývat různých forem, každopádně dva různé zisky musí být poměřitelné (musí existovat uspořádání na množině zisků).

První fází vývoje bylo vytvoření herního světa, ve kterém se budou závody odehrávat. Na rozdíl od většiny současných závodních her, které jsou striktně koridové, byl zvolen „sandboxový“ styl prostředí. Sandbox nebo také otevřený svět značí druh herního level designu, kde se hráč může pohybovat volně po virtuálním světě a není omezován umělými překážkami.

Při vytváření tak velkého prostředí se ve většině her vezme nějaká část reálného nebo smyšleného světa. Tato část světa se matematicky modeluje, aby ji mohl zobrazit počítač. Tento matematický model je typicky výrazně zjednodušený oproti realitě (i té smyšlené), protože zjevně nemá smysl modelovat svět na úrovni atomů. Matematický model je simulací reálného nebo smyšleného světa. Toto zjednodušení je v počítačových hrách nesmírně důležité, protože i velmi zjednodušený model může být pro člověka nerozeznatelný od reality, pokud se použije chytře. [2]

Jako zdroj dat pro terrain objekt hry je použit Google Earth. Jednak je dostupný zadarmo a přesnost zobrazení výškových dat je více než dostačující. Vytvořený terén o velikosti ± 25 km^2 tak svými vlastnostmi napodobuje skutečný terén s odpovídajícími parametry. Jedná se o zjednodušené okolí Železnobrodska.
Aby herní svět nebyl prázdný, je potřeba ho osadit nějakými objekty. Samozřejmě se terén nejprve naplní připravenými modely ze C4D. Jak bylo již zmíněno, Unity přímo podporuje formát C4D, takže je manipulace s nimi velmi snadná.

Prostředí s domy a doplňky již vypadá mnohem lépe, stále však chybí známky nějaké vegetace, které by terén udělaly mnohem uvěřitelnější. Pro tyto účely Unity obsahuje nástroje tree creator a grass.

Tree creator umožňuje vytvářet a upravovat procedurální* stromy. Parametrů, které lze ovlivnit, je nepřeberné množství od velikosti a tloušťky kmenu až po počet generovaných listů na jednotlivých úrovních větví.

Výsledné stromy mohou být použity jako normální GameObjects nebo integrovány přímo do terénu. Stromům je možné přidat animace větru, pod kterým se budou ohýbat, takže budou vypadat opravdu jako živé.

Grass je speciální nástroj pro vytváření travního porostu, skal, nebo jiných doplňků terénu ve vysokém počtu. Pro použití trávy dokonce není nutné vytvořit model jako u tree creatoru, stačí jen textura s alpha mapou.

4. 2. 3 Hráči

Kolik hráčů hraje hru? Musí to být přesné číslo (pouze 4 hráči) nebo proměnné číslo (2 až 5 hráčů)?

Jelikož hra je koncipovaná jako online multiplayerova, může ji najednou hrát libovolný počet hráčů od 1 až po 20. Maximální počet na jednu hru je stanoven na 20 kvůli optimálnímu zatížení serveru.

Mohou hráči přijít nebo odejít v průběhu hry? Ano, hráči se mohou libovolně přidávat a odpojovat od hry. Určitou výjimku však tvoří hráč, který hru založí, tzv. hostitel. Jelikož on hru na serveru zakládal a zaregistroval, nastavil její parametry, tak při jeho odpojení dojde ke zrušení hry i všem ostatním hráčům.

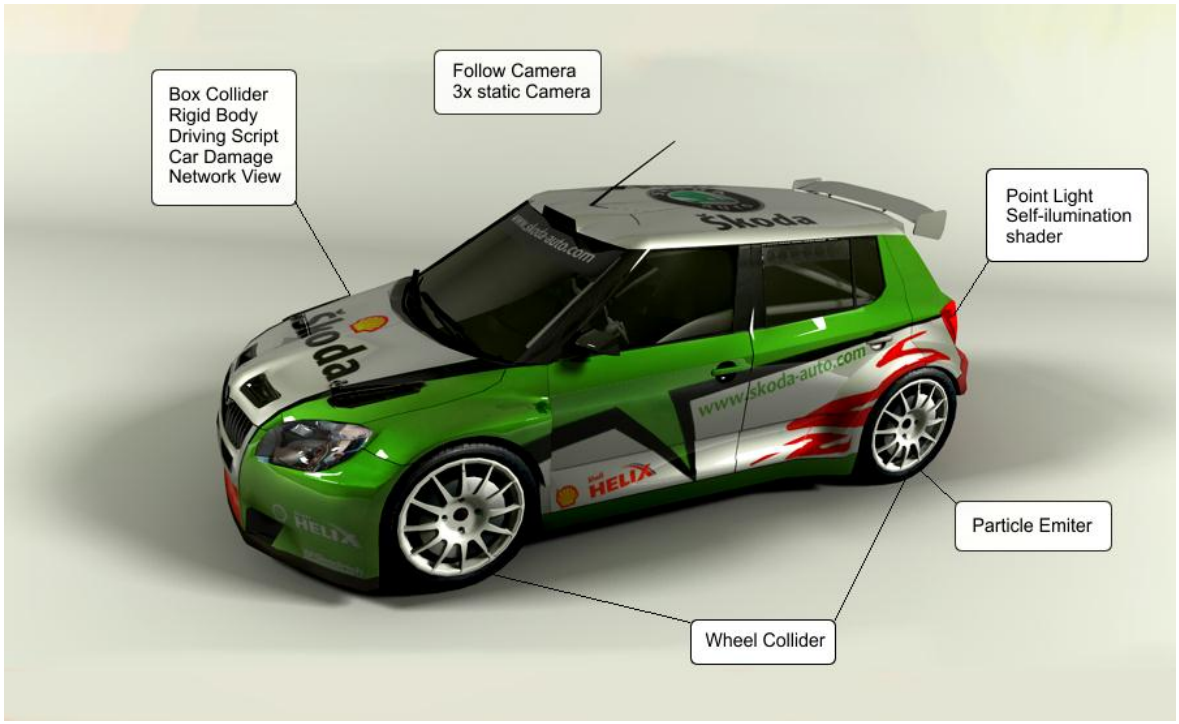
4. 3 Prefab vozidel

Prefab všech vozidel vždy obsahuje samotný model, který je pro každý automobil unikátní a dále pak 3 základní fyzikální prvky: rigid body, box collider pro karoserii vozu a wheel collider použitý na pneumatiky. Při definici parametrů všech těchto složek (ale nejen jich)

* procedurální- v závislosti na nastavení proměnné mění svůj stav

byl záměr nastavit je tak, aby co nejvíce odpovídaly reálnému vozu. Například hmotnost je nastavena na 1 200 kg, což odpovídá závodnímu speciálu.

Do objektu je potřeba umístit ještě driving skript zachytávající události, kterými jsou akce stisknutí klávesnice a vše ostatní, co je nutné pro ovládání hry. Tím se pak budou řídit parametry jednotlivých složek.



Obrázek 20 - Schéma prefab vozu

4. 3. 1 Driving script

Tento skript je klíčovým prkem celého vozu. Driving skript obsahuje mnoho proměnných a vysvětlovat funkci všech by bylo velmi složité. Proto zde jsou zmíněny jen nejhlavnější parametry, které zásadně ovlivňují jízdní model: maximální rychlost, maximální točivý moment motoru, počet převodových stupňů převodovky, maximální brzdná síla, souřadnice těžiště apod. Po spuštění hry však kromě těchto statických proměnných spravuje i dynamické proměnné. Příkladem takové dynamicky se měnící proměnné je například aktuální rychlost, aktuální zařazený převodový stupeň, aktuální přilnavost jednotlivých kol.

Příklad driving skript kódu:

```
if((Input.GetAxis("Vertical") < 0 )&&(currentSpeed>0)){
SwitchBackLight(1);}
```

Jednoduchá ukázka toho, jak funguje zachytávání událostí v Unity. Pokud je stisknuta klávesa *nahoru/dolů* a zároveň je nenulová rychlost, pak se rozsvítí brzdová světla.

```
if (isBraking ==false) {
       if ((currentSpeed < maxSpeed) && (currentSpeed > (maxBackwardSpeed*-1))){
           FrontLeftWheel.motorTorque  = EngineTorque * Input.GetAxis("Vertical");
           FrontRightWheel.motorTorque = EngineTorque * Input.GetAxis("Vertical");

       }
       else {
             FrontLeftWheel.motorTorque =  0; //pokud není nic stisknuto je výkon 0
             FrontRightWheel.motorTorque =  0;
           }
 }
else { SetBrakeEffects(true);
       FrontLeftWheel.brakeTorque = maxBrakeTorque;
       FrontRightWheel.brakeTorque = maxBrakeTorque;
       FrontLeftWheel.motorTorque =  0;
       FrontRightWheel.motorTorque =  0;
       if(currentSpeed<0){     SwitchBackLight(2);}
  else{ SwitchBackLight(0);}
}
```

Předešlý skript ukazuje poněkud složitější konstrukci obsluhující dávkování výkonu na hnaná kola. Pokud uživatel stiskne klávesu pro *akceleraci/reverzaci* a nedosáhl stále maximální možné rychlosti, pak začne motor posílat výkon na kola. V případě hráčovy nečinnosti je výkon 0 a vůz se bude pohybovat maximálně svou setrvačnou hmotou.

Poslední podmínka ověřuje, zda hráč brzdí/couvá, pokud je stisknuta klávesa *brzdění*, velikost momentu posílaného na kola se změní a bude opačný (brzdící). Pokud rychlost klesne do záporných hodnot, automobil začne couvat a rozsvítí se reverzní světla.

Kombinace driving skriptu a fyzikálních komponent rozhýbou automobil, ale stále se není jak na scénu (automobil) dívat. Proto je dalším prvkem v prefab kamera, která všechno snímá. Ve skutečnosti jich je ve scéně více - celkem 4. První snímá vůz zezadu, druhá snímá interiér vozu, třetí kapotu a čtvrtá snímá prostor před vozem.

Nastavení různých zorných úhlů pro splitscreen, přepínání kamer a spravování jsou jedny z mála eventualit, o které se stará **camera skript**.
Přepínání mezi jednotlivými kamerami je standardně definováno na klávese *c*.

Všechny kamery mají klasické parametry jako v každém jiném 3D programu a kdo již měl možnost s nějakou takovou pracovat, tak mu parametry rozhodně nebudou cizí.
U kamery je možné definovat poměr stran neboli její šířku a výšku, úhel pohledu, bližší a vzdálenou ořezávací rovinu, pozici a rotaci, přiblížení. To vše lze upravovat i pomocí JavaScriptu.

4. 4 Implementace Networkingu

4. 4. 1 Konstrukce klient/server

Pro nastavení a navázání spojení se serverem klient používá 2 skripty: Menu_GUI a Menu_multiplayerCode.
Oba tyto skripty spolu úzce souvisí. První Menu_GUI vytváří uživatelské rozhraní, které graficky reprezentuje stav dějů probíhajících při komunikaci mezi serverem a aplikaci. Tu pak řídí a usměrňuje druhý skript Menu_multiplayerCode.
Nejprve se hra připojí na katedrový server, kde z PHP databáze zjistí aktuální IP Master serveru:

```
var myMasterServerIP;
var serverUrl="http://..... /PHP/getServer.php";
 hs_get = WWW(serverUrl);
    yield hs_get;
    if(hs_get.error) {
        print("There was an error getting IPserver: " + hs_get.error);
    } else {   var server=hs_get.text;
               myMasterServerIP=server;
           }
       Network.natFacilitatorIP = myMasterServerIP;
```

Následující funkce je volána, když se někdo úspěšně připojí k serveru. Pokud se tak stane, server si uloží základní informaci o hráči jako IP a port hráče.

```
function StartHost(players : int, port : int){
       if(players<=1){
              players=1;
       }

       Network.InitializeServer(players, port);
}
function OnConnectedToServer(){
       Network.isMessageQueueRunning = false;
```

```
        PlayerPrefs.SetString("connectIP", Network.connections[0].ipAddress);
        PlayerPrefs.SetInt("connectPort", Network.connections[0].port);
}
```

Po úspěšném připojení se klient dotazuje v pravidelných časových intervalech na stav serveru. Tyto dotazovací zprávy musí být na serveru vyřízeny co nejrychleji, aby se zbytečně nezatěžoval server. Jejich posílání je závislé na stavu klientské aplikace potažmo stavu připojení.

Pro jednoznačnou identifikaci klienta je použit systém přidělování číselných identifikátorů. Na serveru je zaručeno, že dvě klientské aplikace nedostanou stejný identifikátor.
Pro získání identifikátoru klient volá metodu `CreateSortedArray`. Ta vrátí platný identifikátor a založí na serveru objekty patřící danému klientovi. Při veškeré další komunikaci klient používá tento identifikátor.

```
public var sortedHostList : Array;
function CreateSortedArray(){
        sortedHostList = new Array();

        var i : int=0;
        var data : HostData[] = hostData;
        for (var element in data)
        {
                AddToArray(i);
                i++;
        }
}
function AddToArray(nr : int){
        sortedHostList.Add (nr);
        SortLastItem();  }
```

Při každém volání metody serveru se u platného identifikátoru aktualizuje časová značka posledního použití. Server v pravidelných intervalech kontroluje tyto značky a vyřazuje staré identifikátory.

Čas, po kterém server vyřadí neaktivní identifikátor klienta, je načítán z proměnné:

```
        public var CONNECT_TIMEOUT : float = 0.75;
        if(lastPlayNowConnectionTime+CONNECT_TIMEOUT<=Time.time){
                        Debug.Log("Interrupted by timer:");
                                        ……
        FailedConnRetry(NetworkConnectionError.ConnectionFailed);
                                        }
```

Klient po přihlášení na server hlídá dlouhou síťovou neaktivitu a případně volá na serveru metodu pro prodloužení času platnosti jeho identifikátoru.

Pokud je klient síťovou částí prohlášen za neaktivního nebo pokud klient volá metodu pro odhlášení, server nejprve zkontroluje, zda je v aréně a odhlásí ho a následně smaže jeho klientská data.

```
function OnPlayerDisconnected(player: NetworkPlayer) {
	Network.RemoveRPCs(player, 0);
	Network.DestroyPlayerObjects(player);

	//Odebrání hráče ze server listu
	for(var entry : FPSPlayerNode in  playerList){
		if(entry.networkPlayer==player){
			playerList.Remove(entry);
			break;
		}
	}
}
```

4. 4. 2 Vytvoření instance vozidla

Při navázání spojení klient pokračuje podle zvolené možnosti a bude připojen k rozehrané hře nebo založí novou a stane se tak hostitelským PC. Při obou možnostech hra pokračuje načtením herního světa.

Zde se nejprve načte kód, který popisuje typ zvolené logiky hry, počet hráčů na mapě a druh jejich vozu. Z těchto parametrů je skript CarSpawn schopen vytvořit instance všech vozů.

```
function Awake()
	{	x=PlayerPrefs.GetInt("Car_1");

	switch (x){
		case 1:
			Network.Instantiate((prefab1), pos, transform.rotation,0);
		break;

		case 2: Network.Instantiate((prefab2),  .....
```

Funkce Awake se používá pro naplnění proměnných ještě před spuštěním hry a je volána vždy pouze jednou. Na dalším řádku příkaz PlayerPrefs nahlédne do registrů a zjistí číslo vybraného vozu z proměnné car_1.

Ta se nastaví při výběru vozu pomocí funkce `PlayerPrefs.SetInt("Car_1",5);`, kde 5 značí číslo aktuálně vybraného vozu. Díky této proměnné vybere z následující switche správnou možnost a vytvoří instanci (prefab) vozu.
Parametry a návratové typy všech metod při výběru vozů jsou pouze celočíselné proměnné, které se dále nikde nevyužívají. Je to proto, aby byla možná jednodušší implementace nových vozidel do hry.

Následně se inicializují startovní pozice, checkpointy a proměnné herní logiky a také defaultní pozice a cíl kamery.

Při vytváření online instancí vozů vznikl problém, který dlouho zdržoval další vývoj hry. Každý vytvoření prefab obsahuje vždy kompletní vůz včetně kamer a ovládání. Z toho plyne, že při připojení více hráčů do hry jsou vytvořeny u každého vozu vždy nové instance ovládání a nová kamera.

V důsledku toho se všem připojeným hráčům zobrazoval pohled pouze hostitelského hráče (jelikož jeho instance byla první) a všichni hráči ovládali zároveň všechny vozy. Po prostudování uživatelské příručky se ukázalo jako ideální řešení použití funkce *networkView*.

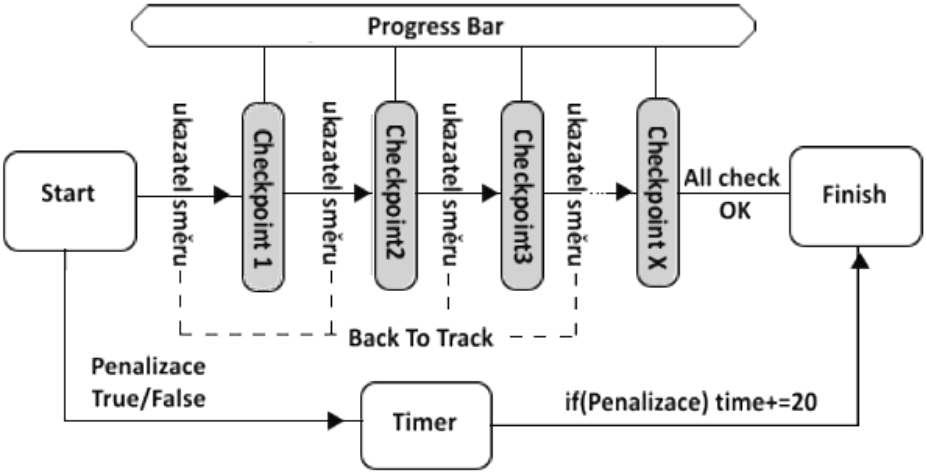
```
function Start(){

    if (networkView.isMine) {
       enabled=true;    }
       else{
           Destroy (gameObject);
   }}
```

Networkview rozezná, zda vytvořený objekt, proměnná či skript patří lokální instanci hry nebo zda přichází z vnějšku. V uvedeném praktickém příkladu, pokud nepatří instance lokálnímu hráči, jednoduše daný objekt smaže.

4. 4. 3 Implementace herních pravidel

Ať už se jedná o lokální nebo síťovou hru, bylo třeba hráče motivovat k tomu, aby hru hráli. Zároveň je ale třeba usměrnit jejich cestu k cíli podle určitých pravidel.
Jako motivační faktor zde slouží ukládání nejlepších časů do online tabulky, více bude popsáno dále.
Jako omezení je zde implementováno několik jednoduchých pravidel, která jsou pro všechny hráče stejná a mají nenásilnou formou donutit hráče dodržovat vytýčenou trať.

Herní plocha obsahuje pět interaktivních typů objektů: Start, Checkpointy, Ukazatele směru, Cíl a Zakázanou oblast.



Obrázek 21 - Schéma propojení herních prvků

4. 4. 4 Start hry

Po přihlášení a načtení vozu hráči ihned nezačne ubíhat čas jako v jiných závodních hrách. Pokud chce závodit, musí nejprve dorazit na startovní pozici. Teprve poté se spustí startovní odpočet a po jeho uplynutí se začne počítat čas. Pokud hráč během této startovní procedury opustí start, je to považováno za předčasný start a v cíli je potrestán 20 sekundovou penalizací.

Nyní praktická část. Základní problém, který se zde řešil, byla detekce jednotlivých hráčů, dále eventuality související s měřením času.

Pro detekci (ne)přítomnosti objektu ve vytýčeném prostoru Unity používá funkci *OnTriggerEnter* s kombinací *collider* se zapnutým parametrem *Is Trigger*.

```
function OnTriggerEnter (other : Collider){
if((other.attachedRigidbody.name == "car") && pripraven==true){
        car1=true;
endTime = Time.time + 6;
        pripraven=false;

}    }
```

Pokud objekt, který bude obsahovat collider a zároveň se jeho nositel bude jmenovat car, vstoupí do této zóny, vykoná se kód v podmínce. Zde se zavolá funkce *Start* () s odpočtem času a znemožní se opětovné spuštění odpočtu.

Pro řešení veškerých časových eventualit Unity používá třídu Time.time. V horním skriptu se díky ní nastaví hodnota odpočtu času na 6 sekund.

Třída Time.time je vždy při volání vykonání akčního řetězu snížena o jednu sekundu a její smazání se provede až při dosažení nulové časové hodnoty. Teprve pak se aplikují objekty umístěné v řetězu za ním.

Pro jeho odpočet se použije funkce *Update* (), která je automaticky volána opakovaně:

```
function Update()
        {if(car1==true){
        timeLeft= endTime - Time.time; }
```

Obdobně se funkce použije i pro měření času s tím rozdílem, že se Time bude přičítat.

4. 4. 5 Kontrolní body

Hráči se potřebují dostat z jednoho konce do druhého. Aby byli donuceni dodržovat trať, je cestou rozmístěno několik kontrolních bodů (checkpointy), které musí v přesném pořadí projet něž dorazí do cíle. Pokud nějaký bod minou, čas se jim vůbec nezapočítá.

Co se týká praktické implementace, skript z velké části kopíruje startovní skript s rozdílem v podmínce:

```
NextChekpoint.GetComponent("checkpoint_skript").enabled=true;
NextChekpoint.collider.active = true;
```

kde *NextChekpoint* je na začátku definovaný jako další kontrolní bod. Z této skutečnosti plyne, pokud hráč neprojede jeden kontrolní bod, nespustí ani následující.
Vzdálenost k dalšímu kontrolnímu místu hráči zobrazuje horní ukazatel postupu (progress bar). Vzdálenost se stanoví pomocí příkazu:

```
var dist = Vector3.Distance(transform.position, Race_Car.transform.position);
```

Transform.position jsou souřadnice checkpointu a Race_Cartransf. jsou souřadnice hráče.

Pro přepočet vzdálenosti na procentuální postup progress báru jsem sestrojil vlastní konstrukci, která vychází ze vzdálenosti automobilu od dalšího stanoviště neboli od celkové vzdálenosti mezi sousedními stanovišti a počty kontrolních bodů.
Skript progress baru :

```
Totaldist=Vector3.Distance(transform.position, NextChekpoint.transform.position);
```

```
function OnGUI(){
    if(activeC){
      var dist = Vector3.Distance(transform.position, Race_Car.transform.position);

       playerProgress  = dist/ Totaldist * 100;
              var ProgressPosition : float=(Screen.height-offset_YD)-
              (actualP*(((Screen.height)-80)/NumP));

              GUI.DrawTexture(Rect(175,Screen.height-offset_YD,10,-
                        Screen.height+55), progressBarEmpty);
              GUI.DrawTexture(Rect(175,ProgressPosition,10,-(((Screen.height-80)/NumP)-
       (Screen.height/NumP)*playerProgress/100)), progressBarFull);

GUI.DrawTexture(Rect(175,Screen.height-offset_YD,10,-((Screen.height/2)-80)/NumP),
progressBarFull2);
}
else{
       GUI.DrawTexture(Rect(175,Screen.height-offset_YD,10,-(actualP*(Screen.height-
       80)/NumP)), progressBarFull2);
                     }
              }
```

Total distance počítá vzdálenost mezi aktuálním a dalším (NextCheckpoint) kontrolním stanovištěm, proměnná dist značí vzdálenost od posledního checkpointu. Tyto 2 hodnoty se navzájem podělí, vynásobí 100, čímž se získá procentuální postup. Kolik to bude zobrazených pixelů se dopočítá z rozlišení monitoru a počtu checkpointu. Poslední řádky skriptu vybarvují již ukončené úseky.

4. 4. 6 GUI ukazatele směru

Během cesty je hráč navigován pomocí ukazatelů, které mu jednak ukazují směr další cesty a zároveň ho upozorňují na obtížnost následujícího úseku. Obtížnost je vyjádřená pomocí čísel 1-9, kde 1 znamená velmi lehkou (přímou zatáčku), 5 značí pravoúhlou zatáčku a 9 bude ostrá zatáčka o 180˚.

Ukazatele se opět spouští projetím collideru, zároveň jsou tyto objekty využívány jako záchranné body při vracení se na trať.

Pokud hráč zmáčkne tlačítko pro návrat na trať, hra najde nejbližší předešlý ukazatel v hráčově okolí a přemístí ho na tuto pozici. Současně s tím vynuluje jeho rychlost a veškeré hmotové charakteristiky.

Prakticky to probíhá následovně:

```
if (Input.GetKeyDown ("h")){
        FindClosestArrow();
              }

function FindClosestArrow () : GameObject {
    var gos : GameObject[];
    gos = GameObject.FindGameObjectsWithTag("SaveP");
    var closestAR : GameObject;
    var distance = Mathf.Infinity;
    var position = transform.position;
    for (var go : GameObject in gos)  {
        var diff = (go.transform.position - position);
        var curDistance = diff.sqrMagnitude;
        if (curDistance < distance) {
            closestAR = go;
            distance = curDistance;
        }
    }
   closetSP  = closestAR.transform;
       resetMM();
       transform.position=closetSP.position;  //změní pozici i rotaci automobilu na
       transform.rotation=closetSP.rotation;  // nejbližší save point
    return closest;   }
```

Nejprve se načtou všechny objekty ve vrstvě „SaveP“ do matice gos. V dalším kroku skript porovnává vzdálenost všech prvků od vozidla. Pokud najde bližší vzdálenost, než byl předešlý bod, uloží ho do proměnné closetAR. Po zkontrolování všech prvků matice nastaví pozici nejbližšího proměnné closetSP. Ta se již použije jako nové souřadnice vozu, automobil se tak vrátí na trať ve správném směru.

4. 4. 7 Zakázaná oblast

Aby si hráč při závodu příliš nezkracoval trať a nebyl tak zvýhodněn oproti ostatním spoluhráčům, jsou na některých místech umístěny tzv. zakázané zóny. Pokud do nich hráč vjede, je automaticky vrácen zpátky na trať.

Prakticky se jedná opět o boxcollider s parametrem Is Trigger, který po aktivaci pošle zprávu funkci FindClosestArrow, z předešlé kapitoly, a opět se vykoná její skript.

```
function OnTriggerEnter (other : Collider){
       if(other.attachedRigidbody.name == "car"){
              Race_Car=GameObject.Find("car");
```

```
Race_Car.SendMessage("FindClosestSpawnEnemy");
}
```

4. 4. 8 Cíl hry

V případě, že hráč úspěšně projede všemi kontrolními body, poslední z nich mu „zpřístupní“ cílovou čáru. Po dojetí se zobrazí jeho konečné umístění a dosažený čas. Dosažený čas značně souvisí se startem hry. Jestliže během startovací procedury hráč předčasně odstartoval, navýší se mu konečný čas o 20 sekund.

K tomuto tématu je nutné zmínit algoritmus na seřazení časů do výsledkové tabulky. I když Unity obsahuje funkci na seřazení matic, tak její praktická implementace nevykazovala výsledky, které byly očekávány. Proto se přistoupilo ke kroku implementovat vlastní alogoritmus pro seřazení tabulky podle časů.

Jelikož se probraly různé metody v předmětu Algoritmů a datových struktur, bylo možné si volit z několika variant.

Po otestování několika různých alternativ se nejlépe osvědčil Algoritmus **Selection sort.** Tento systém určování pořadí není nejelegantnější. Dalo by se jistě použít sofistikovanější určování. **Selection sort** systém je jednoduchý a dá se snadno implementovat, a proto je použit.

Princip tohoto algoritmu je následovný:

1. Najde prvek s nejmenší hodnotou v posloupnosti dat.
2. Zamění se s prvkem na první pozici.
3. Na první pozici se nyní nachází správný prvek, zbytek posloupnosti se uspořádá opakováním těchto kroků pro zbylých n-1 prvků, dokud je n > 1.

Kódově pak algoritmus vypadá:

```
function SelectSort()
	{	for (i=0; i<numPL; i++)
		{	var minim:int = i;
			for (j=i+1; j< numPL; j++)
			if (values[j] < values[minim])
			minim = j;
			var temp = values[i];
			values[i]=values[minim];
			values[minim]=temp;
```

```
                var temp = namePL[i];
                namePL[i]=namePL[minim];
                namePL[minim]=temp;          }
```

Při seřazování časů (matice values) se zároveň přehazují jména hráčů v druhé matici (matice namePL).

Sám o sobě algoritmus již bude fungovat. Aby měl co seřazovat, bylo nutné do skriptu přidat prvek, který bude vyměňovat data mezi tabulkami/hráči. Podmínkou pro takovou výměnu je přidat vlastnost NetworkView pod objekt. Dále již lze používat funkce *@RPC*, které zprostředkovávají samotnou výměnu dat. Praktický příklad takové funkce je následující:

```
@RPC
function setonline(TimePLA:float,ID:int,nameP:String){
        values[ID] = TimePLA;
        namePL[ID]=nameP;
        for(i=0;i<10;i++){
                listPL2[i]=listPL[i];
                }
        }
```

Kromě zápisu @RPC funkce nevypadá nikterak zvláštně.
Mnohem zajímavější je syntaxe volání této funkce:

```
networkView.RPC("setonline",RPCMode.All,TimePLA,ID,myPlayerName);
```

kde parametr RPCMode.All značí, komu (všem připojeným uživatelům) a jaké parametry budou funkcí odeslány.

Další součástí tohoto kódu je posílání jména a hesla do databáze PHP.
Pro větší bezpečnost databáze (a autenticitě dat) se veškerá data při odeslání tzv. zahashují.

Zahashování znamená jednoznačné zakódování uživatelského jména a času na číslo, popř. na řetězec, pomocí nějaké funkce (Autorizačního klíče). Přitom se předpokládá, že dobrá hashovací funkce bude přiřazovat výsledky rovnoměrně do oboru hodnot, i když vstupní hodnoty rovnoměrně zadávány nebudou. Tedy pro velmi podobné vstupy funkce vrátí velmi nepodobné hodnoty.
Autorizační klíč se mění při každém přístupu do databáze. Kombinace odesílaných dat a vygenerovaného klíče je zahashována do dlouhého těžko zapamatovatelného řetězce. Kódu se předává PHP skript v adrese jako proměnná. [4]
Jako hashovací funkci jsem použil funkci md5, která je standardní součástí PHP.

4. 5 Publishing

Pro vytvoření spustitelné verze výsledného projektu je nutné si nastavit určité vlastnosti, které se nachází: *File->Build Settings*. V tomto menu se vybírají scény, které budou exportovány do výsledné aplikace. Při větším počtu scén je nutné definovat jejich pořadí. Vždy platí pravidlo, že 0-tá scéna se spustí jako první. Poslední parametr, který je třeba nastavit, je platforma, pro kterou bude aplikace exportována.

4. 5. 1 Windows verze

Při zvolení exportu do Windows verze je vytvořena klasická *.exe aplikace, kterou podporují všechny verze Windows od XP dále. Unity společně s exe souborem vytvoří složku obsahující všechny potřebné knihovny pro spuštění projektu. Výsledkem je projekt, který lze spustit na většině PC bez jakékoliv instalace nebo případné potřeby dalších práv na účtech, na kterých je aplikace spouštěna.

Za zmínku také stojí možnost zapnutí tzv. Display resolution dialog, který umožňuje při startu aplikace nastavit rozlišení a namapovat všechny klávesy ovládání.

4. 5. 2 Webová verze

Webová verze oproti Windows verzi vygeneruje 2 soubory, klasické html a k tomu datové *.unity3d. Pro spuštění Webové verze v prohlížeči ji stačí upnout na libovolný server a spustit pomocí linku na soubor Html.

Za malou nevýhodu, která vlastně ani nevýhodou není, by mohla být považována již zmíněná nutnost mít nainstalovaný Unity WebPlayer. Pokud doplněk není v PC nainstalovaný, při prvním spuštění hmtl soubor odkáže na web www.unity3d.com, kde je volně ke stažení. WebPlayer má velikost do 1MB. Pro jeho instalaci nejsou vyžadována administrátorská práva.
Další mírné omezení vyplývá ze samotného konceptu spuštění aplikace bez instalace. U webové verze není možné volně přistupovat na disk – číst a zapisovat soubory.

4. 5. 3 Ostatní verze

Ostatní verze (Andoir, iOS i konzolích Xbox 360, PS3 a Wii) jsou již zahrnuty pouze v placené verzi unity. Každá z verzí má určitá omezení případně nějaké další možnosti. Před zvolením výsledné platformy je dobré podívat se do dokumentace, jaké limity je nutné dodržovat.

5 Databázový systém

5. 1 Definice PHP

PHP je skriptovací programovací jazyk navržen pro programování dynamických internetových stránek a webových aplikací.

PHP je technologie běžící na serveru. Typický PHP skript obsahuje jednak kusy normálního HTML kódu, a jednak kusy programového kódu. Když webový server obdrží požadavek na zpracování takového skriptu, vezme:

- kusy HTML kódu;
- části PHP programového kódu provede;
- výsledek zkombinuje a odešle prohlížeči.

Tato filozofie fungování je nesmírně mocná. Server totiž může provést jednu nebo desítky operací a výsledek vždy pošle do prohlížeče jako obyčejnou HTML stránku.

PHP je v současné době dostupný na většině webových serverů a funguje na většině operačních systémů. Nejběžnější je spojení PHP s databázovým systémem MySQL a webovým serverem Apache.

PHP má syntaxi velmi podobnou jazyku C a je většině vývojářů dost blízký.

5. 1. 1 Použití PHP

Hlavní účel, pro který jsem PHP použil ve své aplikaci, je vytvoření jednoduché a snadno přístupné databáze. Dalším důvodem použití PHP je komunikace s externí aplikací Unity3D respektive se skripty psané v JavaScriptu. Od nich pak přijímá data, která zapisuje do MySQL databáze a naopak posílá výsledky zpět do aplikace při výpisu nejlepších časů.

Příklad PHP skriptu ukládající časy do tabulky:

```
        $db = mysql_connect('mysql.tym.cz', 'tym_adanys', '670466') or die('Could not
connect: ' . mysql_error());
       mysql_select_db('scores') or die('Could not select database');

       $name = mysql_real_escape_string($_GET['name'], $db);
       $score = mysql_real_escape_string($_GET['score'], $db);
       $hash = $_GET['hash'];
```

```
$secretKey="XXXX";
$real_hash = md5($name . $score . $secretKey);
if($real_hash == $hash) {
    $query = "insert into scores values (NULL, '$name', '$score');";
    $result = mysql_query($query) or die('Query failed: ' . mysql_error());
}
```

5. 2 MySQL

Mysql je multiplatformní relační databázový systém běžící jako server. Podporuje širokou paletu operačních systémů jako Linux, Mac OS, Windows ale i Novell NetWare a další. Systém MySQL je šířen jako Open source a je tak dostupný zdarma všem uživatelům.

MySQL má široký záběr a vysoký podíl při řešení mnoha různorodých úloh pracujících s databázemi. Popularita MySQL tkví v jeho komplexním a přesto jednoduchém použití spolu s intuitivním editačním rozhraním, kde nejpopulárnější je phpMyAdmin.

Pro práci s databázovými tabulkami je užitečné (ne-li přímo nutné) mít alespoň jednu položku (sloupec), jejíž hodnota bude jednoznačně identifikovat záznam v tabulce. U každého sloupce je možné definovat jeho jméno a datový typ jednotlivých polí záznamu.

Návrh databáze se odvíjel od vzniklých potřeb pro mou aplikaci. Databáze je dimenzovaná pro uchování maximálně 25 časů, čímž databáze spadá mezi malé a středně velké. Tabulky v databázi jsou vytvořené pro data typu float (časy hráčů) a také pro string (jména hráčů).

Vzhledem k potřebnému využití databáze aplikací by bylo vytváření složitějších struktur zbytečné. Samozřejmě by bylo možné ještě rozšíření o pokročilejší systém loginů uživatelů, ale to je úkol, jehož realizace je nad moje programátorské i HW možnosti, neboť udržovat obsáhlé informace vyžaduje značné kapacity.

5. 2. 1 Jazyk SQL

Práce s databázemi, tabulkami a daty probíhá pomocí příkazů, respektive dotazů na databázi v jazyku SQL(Structured Query Language).

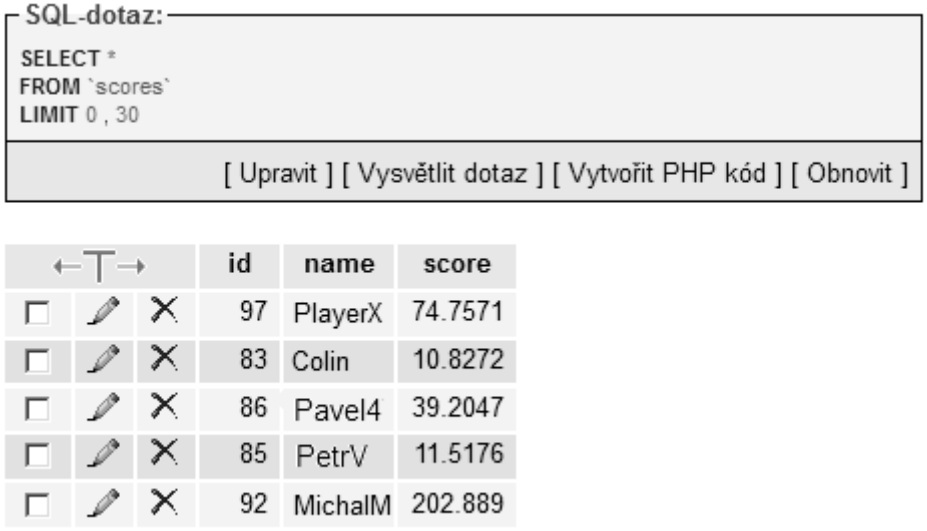
SQL patří mezi tzv. **deklarativní programovací jazyky**, což v praxi znamená, že kód jazyka SQL se nepíše v žádném samostatném programu (jako tomu bylo např. u jazyka C nebo Pascal), ale vkládá se do jiného programovacího jazyka, který je již procedurální. Se samotným jazykem SQL se pracuje pouze v případě, že se terminálem připojí na SQL server a na příkazový řádek jsou zadávány přímo příkazy jazyka SQL. [2]

Následně co unity zašifruje a odešle data, jsou načtena metodou post a přiřazena do proměnných. Dále se vykoná PHP skript, který má za úkol vytvořit dotaz pro MySQL.

Na základě nastavených omezení MySQL buď zařadí odeslaná data na vhodné místo v databázi, nebo je zcela zahodí.

Při načítaní se postupuje obdobným způsobem pouze s malými rozdíly. V databázové struktuře se naleznou relevantní data, které jsou odeslány skriptu getScore. Ten data převede do vhodného tvaru a odešle do Unity.

Práce s databázovými strukturami není v této aplikaci příliš složitá, a proto je použití PHP a MySQL dostačující.



Obrázek 22 - Ukazka MySQL databáze hráčů

6 Návod ke Hře

6. 1 Hlavní Menu

Při spuštění se uživatel dostane do hlavního menu (Obrázek 23), kde si vybírá ze 4 možností:

Single player neboli hra 1 hráče. Lokální hra proti naprogramovaným NPC. Cílem hry je dosažení nejlepšího lokálního času.

Multiplayer hra pro více hráčů. Tato položka dále zahrnuje další 2 možnosti: online a splitscreen.

V online hře se hráč připojí k hernímu serveru a hraje s živými soupeři. Jeho konečné časy se ukládají do online databáze a lze je srovnávat s ostatními hráči. Tato hra vyžaduje online připojení.

*Splitscreen D*va hráči hrají proti sobě na stejném PC. Hra je na obrazovce rozdělena na 2 oblasti, kde každý hráč má jednu polovinu (Obrázek 24). Výsledné časy se opět srovnávají pouze lokálně mezi oběma hráči a NPC.[*] Protože je zde grafika přepočítávána dvakrát, je možné, že tento režim poběží na slabších konfiguracích pomaleji. Potom se doporučuje snížit grafické nastavení hry.

Highscore obsahuje žebříčky nejlepších časů hráčů z online her. Pro získání tabulky je potřebné být připojen k síti.

C*redit* přehled autorů a lidí spolupracujících při vývoji hry.

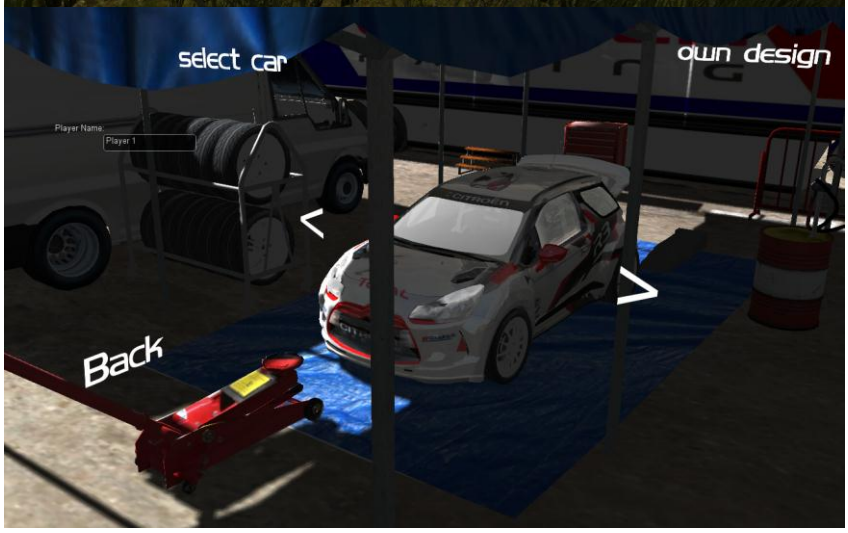
6. 2 Menu výběr vozu

Nejprve by měl hráč vyplnit své jméno. Pod tímto jménem budou ukládány veškeré dosažené výsledky při aktuální hře.

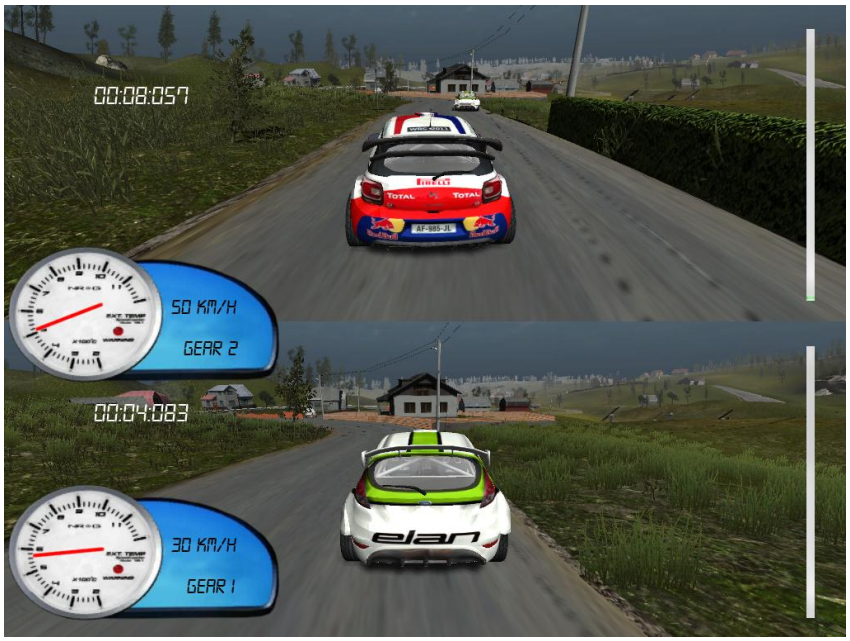
Jednotlivé vozy se přepínají pomocí ikon < > .Vůz se vybere kliknutím na ikonu *select car*. V případě režimu single player dojde již ke spuštění hry. Pokud je vybrán režim splitcreen, vybírá si vůz hráč 2, je též nutné vyplnit i jeho jméno.

[*] **NPC** (z anglického **non-player character**, příp. non-playable character) postava neovládaná hráčem ale počítačem. Muže být zcela naskriptovaná nebo být ovládána jednoduchou umělou inteligencí.

Obrázek 23 - Hlavní menu hry



Obrázek 24 - Menu výběr vozu



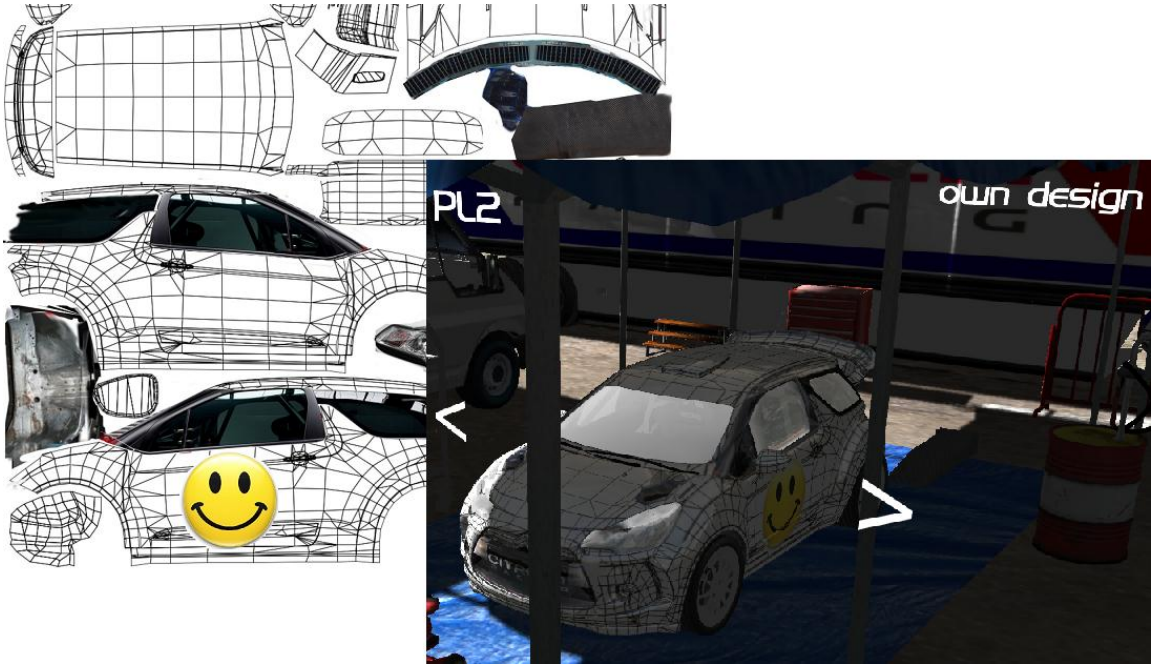
Obrázek 25 – Splitscreen hra

6. 2. 1 Vlastní design vozu

Jednou z důležitých vlastností aplikace, která ji odlišuje od podobných her žánru, je možnost vytvořit si vlastní design libovolného vozu.

Pro vytvoření takového designu je potřeba mít jednoduchý grafický editor např. malování. Po otevření složky „data“ stačí nalézt bitmapový obrázek s názvem odpovídacího vozu a koncovkou body. Např. pokud bude hráč vytvářet design pro Citroen DS, otevře obrázek citroenDS_body a upraví ho podle své fantazie. Jediné omezení při vytváření vlastní textury je nutnost dodržet hranice jednotlivých ploch, aby odpovídaly originálnímu obrázku, jinak dojde ke zkreslení textury. Načtení vytvořené textury probíhá pomocí tlačítka *own desing* v menu volby vozu.

Popsaný postup platí pouze pro exe verzi hry. Jelikož webová verze není umístěna na lokálním disku, nemůže z uvedeného umístění načítat zdrojová data.



Obrázek 26 - Ukázka vytvoření vlastního designu vozu

6. 3 Menu nastavení online hry

Centrální tabulka zobrazuje seznam aktuálně spuštěných her. Uživatel si zde může vybrat, ke které hře se připojí. Aplikace pravidelně stahuje ze serveru informace o vytvořených arénách a zobrazuje jejich detaily. Pro okamžitou aktualizaci seznamu se použije tlačítko *refresh list*. Pro rychlé připojení k náhodné hře slouží tlačítko *quick game*.

Je také umožněno založení nové arény z menu přes příslušný formulář. Je potřeba vyplnit počet hráčů (maximum 20) a nastavit port, na který se hra má připojit.

6. 4 Nastavení hry

Menu nastavení hry se zobrazuje před spuštěním hry. To je rozděleno do 2 sekcí:

6. 4. 1 Nastavení Ovládání

Tato obrazovka umožňuje shlédnout nebo změnit nastavení kláves ovládání. Každá akce hry má možnost přiřazení alternativní klávesy. Klávesy se mohou i kombinovat např. použít klávesu *a* pro plyn a klávesu *z* pro brzdu a směrové šipky pro změnu směru).

Akce	Klávesa hráč 1	Klávesa hráč 2	Popis
Akcelerace	↑	W	Zvýšení rychlosti
Brzdy	↓	S	Snížení rychlosti, Reverzace
Vlevo	←	A	Změna směru
Vpravo	→	D	Změna směru
Ruční brzda	space	L Shift	Ruční brzda zablokuje zadní kola, což způsobí smyk
Vrácení na trať	H	Q	Při vyjetí z tratě nebo uvíznutí mimo trať se vrátí vůz zpátky na trať
Resetování vozu	R	R	V single playeru a online hraní hráče vrátí na start a vynuluje jeho čas. Při splitscreenu dojde k resetování obou hráčů.
Kamery	C	E	Přepínání kamer
Pauza	**Esc**	**Esc**	Hra se v ten okamžik pozastaví a objeví se okno menu.

6. 4. 2 Nastavení grafiky

Screen resolutions umožňuje vybrat rozlišení, ve kterém bude hra zobrazena. Od minimálního rozlišení 640x480 až po maximální 1920x1080, záleží na grafické kartě a možnostech monitoru. Čím vyšší rozlišení se nastaví, tím lepší bude kvalita grafiky za cenu vyššího výkonu. Proto nejvyšší možnosti jsou doporučeny jen na silných počítačích.
Graphic quality nastavuje velikost detailů ve hře. Od rychlé po fantastickou. Jednotlivá nastavení ovlivňují parametry jako viditelnost, detaily vozů, osvětlení a speciální efekty. Opět platí, že při zvolené nejlepší grafice může klesnou počet FPS (frame per second- počet snímků za vteřinu), a proto je doporučována pouze u kvalitních strojů.

Windowed Zapnutí nebo vypnutí tohoto parametru ovlivňuje, zda hra bude spuštěna v plném nebo v menším okně.

6. 5 Herní GUI

Herní prostředí obsahuje tyto elementy:

Čas *hry*: Tento údaj zobrazuje čas uplynulý od startu (bez penalizace). Počítání času se ukončí projetím cíle. Před tím hráč musí projet všechny kontrolní body ve stanoveném pořadí.

V*zdálenost k dalšímu checkpointu* se ukazuje na přehledném ukazateli, v jaké části tratě se hráč aktuálně nachází a kolik mu chybí k dalšímu kontrolnímu bodu. Barvy ukazatele značí: Zelená probíhající průjezd mezi 2 kontrolními body, červená značí úspěšné projetí kontrolního bodu a ukončený úsek trati.

Ukazatele směru naznačují směr další cesty a zároveň upozorňují na obtížnost následujícího úseku. Obtížnost je vyjádřená pomocí čísel 1-9, kde 1 znamená velmi lehkou (přímou zatáčku), 5 značí pravoúhlou zatáčku a 9 bude ostrá zatáčka o 180˚.

Multifunkční tachometr zobrazuje rychlost automobilu, aktuální zařazený rychlostní stupeň a otáčky motoru.

Obrázek 27 - Herní GUI

6. 6 Minimální požadavky

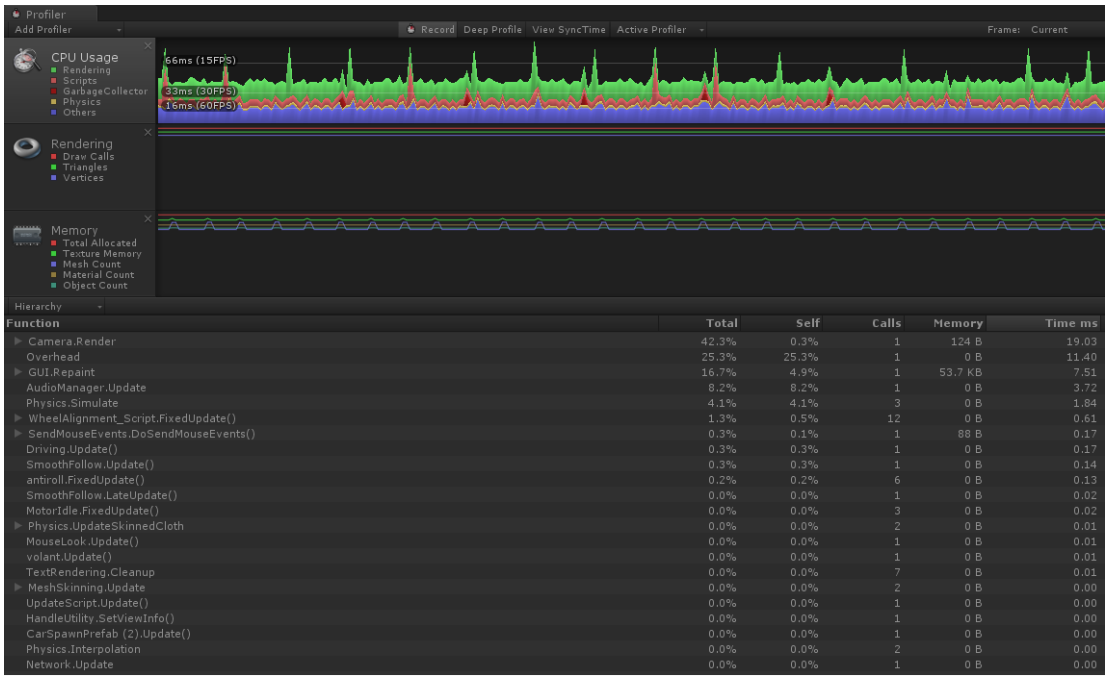
Při vytváření této aplikace bylo původním záměrem splnit současné hardwarové možnosti většiny současných uživatelů. Vzhledem k tomu, že dnešní Low Endová PC sestava má parametry procesor 1.6GHz, 64MB grafickou kartu a 512MB RAM, stanovil jsem ji jako minimální požadavek pro běh aplikace. Pro bezproblémový běh hry v plných detailech rozhodně doporučuji mnohem silnější stroj.

Velikost aplikace: Hotová webová aplikace přeložená do finálního formátu má velikost 64MB.

Výsledná exe aplikace obsahující kompletní hru včetně textur a modelů má velikost cca 160MB, což je více než bylo původním záměrem, avšak při současných rychlostech internetového připojení to není nepřekonatelný problém. Pro optimální běh hry se doporučuje stáhnout si verzi exe. Jednak zaručuje mnohem lepší plynulost a možnost vytvoření vlastního designu, dále nastavení kvality grafiky, rozlišení kláves, což WEBová verze neumožňuje.

6. 6. 1 Test výkonu

Tato kapitola zobrazuje proběhlý test výkonu v Unity integrovaného testeru náročnosti, kde lze vyčíst zatížení jak Grafické karty tak procesoru. Pro největší názornost testu byly nastaveny úrovně detailů na režim Beautiful. Výsledky měření jsou zobrazeny v grafu:



Obrázek 28 - Test výkonu pomocí profileru

6. 6. 2 Testování uživateli

Testování probíhalo po celou dobu implementace. Po dokončení celého softwaru byla aplikace důkladně testována mými přáteli a známými.

Otázka 1

Jak na vás působí celkový vzhled aplikace a grafika hry? Je menu dostatečně přehledné?

Odpověď A

Veškerý interface aplikace je z uživatelského pohledu vzhledově přívětivý. Velmi se mi zamlouvala 3D reprezentace většiny menu. Určitě bych uvítal integraci nastavení hry do hlavní menu.

Co se týká grafické stránky rozhodně si nemohu stěžovat. Modely vozů jsou velmi dobře zpracovány a odpovídají reálným předlohám. Herní mapa má obdivuhodné rozměry a přesto je dostatečně rozmanitá a členitá. Vegetace vypadá na současné poměry velmi dobře a většina modelů rozhodně nezaostává.

Odpověď B

Při prvním spuštění hry jsem byl mile překvapen z vlastního vzhledu hry. Velmi se mi líbilo moderní prostorové menu hry a krásné interaktivní menu výběru aut.

Po grafické stránce patří hra k tomu lepšímu, co se dnes dá vidět na free scéně. Kvalita modelů je na velmi dobré úrovni, herní prostředí je na závodní hru velmi rozsáhlé a některé lokace vypadají opravdu hezky.

Otázka 2

Který herní režim se vám nejvíce líbil a proč? Přidali byste nějaký?

Odpověď A

Měl jsem dostatek času vyzkoušet všechny herní režimy a nejvíce sympatický mi přišel splitscreen. Zahrát si naživo se soupeřem na jednom počítači bude mít vždy nezaměnitelné kouzlo.

Přidal bych herní režim typu šampionát, kde by se jelo několik tratí za sebou a záleželo by, jak by kdo dojel v každé z nich. Pro tento typ hry by bylo potřeba vytvořit více tratí.

Odpověď B

Hru jsem pořádně otestoval a pokud mám vybrat nejoblíbenější režim, tak to bude splitcreen. Tento režim mám strašně rád, pamatuju si ho už z her z roku 1998. Pokud ho hra nabízí, neváhám a s kamarády si ho vždycky zahraji.

Otázka 3

Jak na vás působí fyzika vozů implementovaná ve hře?

Odpověď A

Nejedná se sice o žádný super reálný simulátor vozu jako u některých AAA titulů, ale svůj účel rozhodně skvěle plní.

Odpověď B

Chování vozidel a jejich ovládání ve mně rozhodně zanechalo pozitivní dojmy. Fyzika mi přijde vyvážená a hráči nehází zbytečně klacky pod nohy. Určitě to není 100% reálné, ale ovládání je rozhodně hratelné a zábavné.

Otázka 4

Ve srovnání s jinými free aplikacemi, má aplikace nějaké zásadní nedostatky?

Odpověď A

Z hlediska funkčnosti je hra propracována do nejmenších detailů a po čas testování nenastal žádný problém.

Odpověď B

Až na drobné chyby v terénu, na které jsem přišel, a občasné cukání při nastavených vysokých detailech jsem na žádné zásadní chyby nepřišel.

Otázka 5

Jak se vám zamlouvá možnost vytvořit si vlastní design vozidla?

Odpověď A

Tuto možnost nenabízí některé prodávané tituly, takže mi to přijde jako dobrý nápad. Vlastní design jsem si zkusil vytvořit a funguje dobře.

Odpověď B

Hru, kde je možné kreativně měnit herní obsah, mám velmi rád. Proto mi přidání této možnosti přišlo jako dobrá volba. Podobnou možnost si pamatuji z jedné starší placené hry a hrozně mě bavilo vymýšlet si vlastní vzhled vozu, aby vypadal ku mému obrazu.

Otázka 6

Jak byste hru ohodnotili v rozmezí 1-10, kde 10 je nejlepší?

Odpověď A

Veškeré funkce aplikace jsou z uživatelského pohledu funkční. S ohledem na to , že Aplikace je volně šířitelná a je vzhledově na velmi dobré úrovni. Hra je zajímavá a bezesporu „chytlavá“. Dal bych hodnocení 7.

Odpověď B

Hra je zábavná a vizuálně příjemná. Zajímavá je možnost hry dvou hráčů současně a tvorba vlastního designu. Moje hodnocení je 9.

Závěr

Cílem mé diplomové práce bylo vytvořit závodní síťovou hru s možností jednoduché implementace vlastního designu vozu. Návrh hry byl ponechán zcela na mé fantazii a úsudku, přičemž mým osobním cílem bylo dosáhnout maximální možné funkčnosti a co nejlepšího grafického zpracování.

Vývoj hry pro mne byl v některých chvílích velmi obtížný. Nejkrušnější byly samotné začátky, kdy jsem měl minimální znalosti o networkingu a jádru enginu Unity. To obnášelo napsat velké množství kódů bez jakéhokoliv viditelného výsledku. Jak se postupně začalo herní prostředí plnit dalšími a dalšími funkcemi a modely, tak se situace o dost zlepšila a vývoj mě začal bavit.

Prostředí Unity se ukázalo být velmi přívětivé, intuitivní a mocným nástrojem při implementaci mých programátorských myšlenek.
Během vývoje jsem se dobře seznámil s různými vlastnostmi prostředí Unity, hlavně pak s použitým jazykem JavaScrip, který jsem před tímto projektem vůbec neznal.

Aplikace funguje bezproblémově na všech zamýšlených platformách. Během testování se objevily drobné problémy s plynulostí hry ve webové verzi, ale ty byly většinou zapříčiněny rychlostí připojení uživatele, tudíž mnou neovlivnitelné.
Pokud se uživatel setká s problémem při spuštění této aplikace přes web browser, je pravděpodobnější, že nemá správně nainstalovaný WEBPlayer na svém stroji (nebo jej nemá nainstalovaný vůbec), než-li programátorská chyba.

Během práce na tomto projektu se mi podařilo nasbírat značné množství zkušeností převážně z prostředí Unity3D a vývoje her obecně. Řadu vzniklých problémů jsem řešil metodou pokus omyl, ale o to více mě obohatilo jejich případné řešení. Celý projekt se také nečekaně rozrostl. Aplikace v některých ohledech dokonce překračuje mé původní plány, např. funkční animované řízení vozidla, respawn vozu na trať, deformace automobilů apod. V tuto chvíli celý projekt obsahuje zhruba 100 000 řádek kódů, přes 150 modelů a několik desítek textur.

Vše se samozřejmě nepovedlo perfektně. Například některé mé zdrojové kódy mají k přehlednosti a čistému návrhu daleko.

Se zkušenostmi, které mám nyní, bych některé věcí řešil jinak. Také je možné, že vzhledem k rozsahu se v budoucnu mohou vyskytnout nějaké chyby, které bude třeba řešit.

Zároveň prosím mějte na paměti, že každý člověk má svůj originální názor a s řešením některých problémů možná nebudete souhlasit. Veškeré postupy použité pro vytvoření této aplikace jsou takové, aby co nejvíce vyhovovaly mým požadavkům a možnostem a nikdo není nucen je 100% akceptovat.
Výsledný program ovšem splňuje předem stanovené cíle a svou formou jej lze zařadit mezi současné webové hry, které uživatele zabaví a snad i zaujmou svým grafickým ztvárněním. Což považuji za velký úspěch.

To ovšem neznamená, že nemá v určitých ohledech i své rezervy, tudíž možnost pro další rozšíření. Např. automobily se vzájemně od sebe liší pouze svým „fyzickým“ vzhledem, nikoli však již jízdním modelem. Těmito atributy jízdního modelu se rozumí maximální rychlost, zrychlení, stabilita atd. Jistě by nebyl problém změnit jejich hodnotu v driving skriptu. Tento přístup ovšem padá na množství a kvalitě parametrů pro všechny vozy. Optimální nastavení vozu mi trvalo nalézt 14 dní, takže vyvážené nastavení pro všech 11 vozů by trvalo značný čas, který jsem spíše investoval do nových funkcí hry.

Kapitolou samo o sobě bylo vytvoření modelů pro celou aplikaci. Prostředí C4D se velmi osvědčilo a snadno se s ním pracuje. Přesto při velkém množství modelů, které bylo potřeba vytvořit pro celou hru, to byl na jednoho vývojáře velmi těžký úkol.

Tento text se v žádném případě nedá považovat za kompletní návod, jak vytvořit vlastní hru v prostředí Unity3D. Představuje skutečně jenom velmi stručné nastínění jednotlivých subsystémů herního enginu Unity s popisem typických úkonů použitý při vývoji takové nezávislé počítačové hry.

Použitá literatura

[1] *Hraní počítačových her zlepšuje kreativitu.* [online]. [cit. 2012-05-08]. Dostupné z: http://www.novinky.cz/internet-a-pc/.

[2] *Wikipedie.* [online]. [cit. 2012-05-07]. Dostupné z: http://cs.wikipedia.org/wiki/

[3] *Cinema4D návody.[*online]. [cit. 2012-05-07]. Dostupné z: www.3dsoftware.cz

[4] *Unity documentation*. [online]. [cit. 2012-05-08]. Dostupné z: http://unity3d.com/support/documentation/

[5] ROCH, M. a SCHREIBER, I. *Škola game designu.* [online]. [cit. 2012-05-08]. Dostupné z:http://gamedesign.cz/

[6] ŠKULTÉTY, R. *JavaScript, Programujeme internetové aplikace.* Computer Press, 03/2004. ISBN 80-251-0144-4.

[7] RANDALL, H. *KeyShot2 for PTC Creo.* [online]. [cit. 2012-05-18]. Dostupné z: http://www.vizworld.com/2011/06/

[8] *EagleUp*. [online]. [cit. 2012-05-18]. Dostupné z: http://mcu.cz/news.php?extend.3014.3

Seznam příloh

Příloha 1 - Obrázek ze hry

Příloha 2 - Obrázek ze hry

Příloha 3 – Obrázek ze hry

Příloha 4 - Změna kamery