ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA DOPRAVNÍ

Bc. Klára Pudová

Automatická tvorba provozu ve scénářích pro vozidlové simulátory

Automated creation of traffic in scenarios for driving simulators

Diplomová práce

2019

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K616.....Ústav dopravních prostředků

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Klára Pudová

Kód studijního programu a studijní obor studenta:

N 3710 – IS – Inteligentní dopravní systémy

Název tématu (česky): **Automatická tvorba provozu ve scénářích pro vozidlové simulátory**

Název tématu (anglicky): Automated creation of traffic in scenarios for driving simulators

Zásady pro vypracování

Při zpracování diplomové práce se řiďte osnovou uvedenou v následujících bodech:

- Prozkoumejte kinematické chování vozidla během jízdy
- Navrhněte experiment a vlastní měřící zařízení, pomocí kterého získáte informace o chování vozidel během jízdy na pozemních komunikacích
- Popište metody tvorby provozu do vozidlových simulátorů
- Vytvořte metodu/nástroj pro automatické generování provozu ve scénářích vytvořených z 3D modelů segmentů komunikací pro vozidlový simulátor FD, a to na základě naměřených/reálných dat
- Ověřte funkčnost vyvinutého nástroje v nově vymodelovaném scénáři a zhodnoťte jeho využitelnost a efektivnost

Rozsah grafických prací: Dle pokynů vedoucího

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury: ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. Moderní počítačová grafika. Praha: Computer Press, 1998. ISBN 80-7226-049-9.

ORLICKÝ, Adam. Automatická tvorba silniční infrastruktury ve 3D pro vozidlové simulátory. Praha, 2016. Diplomová práce. ČVUT Fakulta Dopravní.

Vedoucí diplomové práce: **doc. Ing. Stanislav Novotný, Ph.D.**

Ing. Adam Orlický

Datum zadání diplomové práce: **22. června 2018**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **28. května 2019**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

L. S.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní

doc. Ing. Petr Bouchner, Ph.D.
vedoucí
Ústavu dopravních prostředků

doc. Ing. Pavel Hrubeš, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Klára Pudová
jméno a podpis studenta

V Praze dne ..... 22. června 2018

Poděkování

Na tomto místě bych ráda poděkovala všem, kteří mi poskytli podklady pro vypracování této práce. Zvláště pak děkuji oběma vedoucím mé diplomové práce, a to doc. Ing. Stanislavu Novotnému, Ph.D. a Ing. Adamu Orlickému, za odborné vedení a konzultování mé práce a za rady, které mi poskytovali. Dále bych chtěla poděkovat DiS. Janu Válkovi za rady a pomoc při psaní diplomové práce.

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na ČVUT v Praze Fakultě dopravní.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 28. května 2019 Bc. Klára Pudová

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

Ústav dopravních prostředků

Automatická tvorba provozu ve scénářích pro vozidlové simulátory

Diplomová práce

květen 2019

Bc. Klára Pudová

Abstrakt

Předmětem diplomové práce „Automatická tvorba provozu ve scénářích pro vozidlové simulátory“ je vytvořit metodu automatického vytváření všech souborů potřebných k definici animace pro snadné a rychlé generování provozu do scénářů pro vozidlové simulátory.

Teoretická část se zabývá vozidlovými simulátory a virtuálním prostředím, metodami tvorby provozu do simulátorů a popisem chování vozidla během jízdy, který je důležitý pro správnou definici pohybu vozidla během animace.

V praktické části je vytvořen nástroj (plugin), který umožňuje automaticky přidávat animace jízdy vozidla do scénářů pro vozidlové simulátory. Údaje o rychlosti potřebné pro definici animace byly získány pomocí experimentu s vlastním měřicím zařízením. V práci je popsán algoritmus pluginu a vysvětlen princip fungování. V samotném závěru práce je proveden experiment, který testuje práci s pluginem a kvalitu výsledné animace vytvořené novou metodou.

Klíčová slova: 3D modelování, Rhinoceros, RhinoScript, vozidlové simulátory, provoz, animace, měření rychlosti.

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Transportation Sciences

Department of Vehicle Technology

Automated creation of traffic in scenarios for driving simulators

Diploma thesis

May 2019

Bc. Klára Pudová

Abstract

The subject of the diploma thesis „Automated creation of traffic in scenarios for driving simulators“ is to create a method for automatic creation of all files needed for the definition of animation for easy and fast generation of traffic into scenarios for driving simulators.

The theoretical part is dealing with vehicle simulators and virtual environment, methods of creating traffic for the simulators and the description of the vehicle behavior during the ride, which is important for the correct definition of vehicle movement during animation.

In the practical part is created a tool (plugin), which allows to automatically add vehicle ride animations into scenarios for driving simulators. The velocity data needed to define the animation were obtained by an experiment with its own measuring device. The algorithm of the plugin is described and the principle of operation is explained. At the end of the thesis, an experiment is performed that tests the work with the plugin and the quality of the resulting animation created by the new method.

Key words: 3D modelling, Rhinoceros, RhinoScript, driving simulators, traffic, animation, speed measurement.

Obsah

1 Seznam použitých zkratek

2D	Dvojrozměrný
3D	Trojrozměrný
ČSN	Česká technická norma
ČVUT	České vysoké učení technické
DP	Diplomová práce
EEG	Elektroencefalogram (biosignál mozku)
HTML	Hyper text markup language
IDE	Integrated development environment
IDM	Intelligent Driver Model
LCD	Liquid crystal display
NURBS	Non-uniform rational basis spline
SD	Secure Digital
TIN	Triangulated irregular network
USB	Universal serial bus

2 Úvod

Virtuální realita je v dnešní době stále více diskutovaným odvětvím, které se pomalu, ale jistě dostává do popředí v mnoha oblastech a stává se běžnou součástí našich životů. Jedná se o různé způsoby využití dnes již pokročilých a velmi realistických virtuálních prostředí, a to jak v zábavním průmyslu, tak i v různých vědních odvětvích.

Virtuální prostředí s 3D modely je možné vytvářet a upravovat v nejrůznějších grafických, navrhovacích, animačních či 3D modelovacích programech přímo v počítači, pro koncové uživatele ho pak lze přenést do zařízení k tomu přímo určených, například do simulátorů či 3D brýlí pro virtuální realitu. Pomocí nich lze pak virtuální realitu využít například pro bezpečné trénování či osvojení si dovedností, u kterých by každá drobná chyba mohla mít v reálu fatální následky nebo způsobit vysoké škody.

Brýle pro virtuální realitu se stávají čím dál oblíbenějším prostředkem k využití nekonečných možností virtuálního prostředí v různých odvětvích – například mnozí zaměstnavatelé je používají ke školení svých zaměstnanců, které by v ostrém provozu bylo velmi nákladné či těžko proveditelné. Školená osoba si tak může vyzkoušet různé technologické postupy, aniž by mohla napáchat nějakou škodu, a získává tak zručnost a sebejistotu ještě před samotným nástupem do praxe. Pomocí virtuální reality lze dobře školit například také chirurgy (kteří si před každou náročnější operací mohou plánovaný výkon vyzkoušet nanečisto ve virtuální realitě a lépe se tak na něj připravit), vojáky (pro které je jednodušší a bezpečnější možné rizikové scénáře připravit ve virtuální realitě, aby poté při reálném cvičení nebyli zaskočeni), či piloty (kteří mohou mít přezkoušení na trenažéru či ve virtuální realitě dokonce povinné v rámci získání nebo udržení pilotní licence).

Pro brýle pro virtuální realitu je navržen také nespočet her, které zažívají v současnosti obrovský boom. Lidé si tak po nasazení těchto brýlí mohou vyzkoušet, jak by obstáli v boji s nejrůznějšími netvory, kolik dokáží chytit létajících žetonů, případně si mohou natrénovat nové tenisové údery či taneční kroky. Tyto aktivity se staly mezi lidmi velmi oblíbenými, a jistě mají do budoucna obrovský potenciál.

Velké využití má virtuální realita i v dopravním inženýrství, a to jednak pro trénování řidičů, strojvedoucích či pilotů, ale také pro výzkum – zkoumají se nejrůznější scénáře, se kterými může například účastník silničního provozu přijít do styku. Účastníci těchto testování se tak po nasazení virtuálních brýlí ocitnou třeba na přechodu pro chodce, u železničního přejezdu či v kabině řidiče nákladního vozidla. Takto se testuje, jak se jednotlivé osoby zachovají v daných situacích, případně jaký vliv na ně tyto situace mají. Z výsledků takovýchto testů pak lze například vyvodit, jak vylepšit řízení křižovatky, jak upravit povolenou

rychlost na testovaném úseku, či jak upravit palubní desku nebo vybavení kokpitu ve vozidle tak, aby byl pro řidiče přívětivější.

Pro podobné testy lze kromě brýlí pro virtuální realitu využít i vozidlové simulátory, kdy testovaná osoba sedí přímo v reálném vozidle či jeho části, a virtuální realita je mu promítána na displej nebo plátno před ním, případně i okolo něj. Při takovýchto experimentech pak můžeme kromě chování a osobních pocitů testovaného získávat i další údaje, jako jsou například technická data z vozidla. Můžeme přitom provádět nejrůznější experimenty včetně těch, které by v reálném prostředí nebyly možné, a to za bezpečných podmínek pro řidiče i jeho okolí.

Jelikož jsou tyto experimenty velkým přínosem pro výzkum v oblasti dopravního inženýrství, je třeba hledat nové metody pro zjednodušení vytváření virtuálních scénářů pro tyto experimenty – 3D modelování je totiž velmi pracné a časově náročné. I proto jsem v rámci své bakalářské práce vytvořila metodu KS_Software, pomocí níž lze tyto scénáře vytvářet rychleji a efektivněji – pomocí skládání předpřipravených segmentů komunikací za sebe. Tato metoda šetří při vytváření scénářů mnoho času, a na Fakultě dopravní se aktivně využívá. I proto jsem se rozhodla na svoji práci navázat a obohatit tuto metodu o nové prvky, aby příprava nových scénářů pro experimenty byla ještě snazší.

Pomocí KS_Software lze připravit kompletní statickou virtuální scénu. Pro realisticky působící prostředí je ale nutné do scénáře zahrnout také animace – především provoz, tj. animace vozidel projíždějících po silnicích kolem vozidla, v němž sedí testovaná osoba. Ta se pak ve virtuálním prostředí cítí o něco lépe – koneckonců, na reálných silnicích také málokdy jedeme úplně sami, většinou po cestě potkáváme mnoho jiných vozidel.

Právě za účelem přidání animace do virtuálních scénářů jsem se rozhodla vytvořit nové funkce – pomocí nich bude možné do scénáře přidat jedoucí vozidlo, a to takové, které si uživatel zvolí. Vytvořená animační složka bude obsahovat jak soubory potřebné k samotné animaci, tak i model a texturu zvoleného vozidla, přičemž vozidla se budou pohybovat po křivce, kterou si uživatel zvolí, a se zrychlením, jakého je zvolené vozidlo schopno dosáhnout. Rychlost jízdy vozidla bude vycházet z maximální povolené rychlosti, přičemž vozidlo bude do směrových oblouků zpomalovat na rychlost, kterou pro každý poloměr oblouku určím na základě reálných dat získaných z experimentu.

TEORETICKÁ ČÁST

3 Vozidlové simulátory

Vozidlový simulátor je nástroj pro zkoumání chování řidičů za různých podmínek, a to v prostředí bezpečnějším, než je reálný provoz – ve virtuálním prostředí. Je to systém sestávající z vozidla (nebo jeho části), a virtuálního prostředí promítaného na monitory (např. LCD obrazovky) či promítací plátna obklopující vozidlo. Cílem je, aby se řidič ve vozidle cítil jako v reálném prostředí. To může být kromě co nejrealističtějšího virtuálního prostředí umocněno například zvukovými efekty či pohybovou plošinou, která simuluje naklánění a dynamiku pohybu vozidla (např. při akceleraci či při jízdě do oblouku).

Obrázek 1: Vozidlový simulátor na Fakultě dopravní ČVUT v Praze

Vozidlové simulátory můžeme z hlediska konstrukce dělit na tzv. lehké a plnohodnotné. Lehké simulátory sestávají pouze z kokpitu vozidla či jeho části – výhodou je, že je možné je snadno přestavět pro potřeby konkrétního experimentu, který se na nich provádí, nebo je dodatečně dovybavit různými přídavnými zařízeními. Nevýhodou je hlavně to, že výsledky z experimentů na takovýchto simulátorech mohou být ovlivněny tím, že řidič se v nich nemusí cítit jako v reálném vozidle. Oproti tomu u plnohodnotných simulátorů je využito celé vozidlo, což zaručuje přesnější výsledky – řidič se cítí více jako při jízdě v reálném provozu, a to i díky tomu, že virtuální scéna je v tomto případě často promítána na plátna umístěná jak před vozidlem, tak i po stranách, a často dokonce i zezadu, díky čemuž řidič vidí scénu i ve zpětném

zrcátku. Nevýhodou však je hlavně to, že simulátor nelze snadno někam přemístit nebo ho přestavět pro potřeby konkrétního experimentu.

Hlavním přínosem vozidlových simulátorů je možnost zkoumat chování řidiče či měřit úroveň jeho pozornosti, a to i v situacích, které by mohly být při měření v reálném provozu pro řidiče či jeho okolí nebezpečné. Na vozidlových simulátorech lze ale také například trénovat řidiče na nejrůznější rizikové situace a zvýšit tak jejich odolnost vůči stresu či poklesu pozornosti při řízení.

3.1 Experimenty na vozidlových simulátorech

Vozidlové simulátory slouží zejména pro výzkumy zkoumající chování řidiče v různých situacích – za standartních podmínek i při zátěži (například pod vlivem alkoholu či omamných látek, při únavě). Měří se třeba úroveň pozornosti řidiče nebo příčiny jeho agresivního chování. Díky těmto experimentům je možné zvyšovat spolehlivost a bezpečnost dopravy – např. optimalizací kokpitu vozidel, využitím asistenčních a varovných systémů, intuitivnější obsluhou vozidla či zvýšením komfortu řidiče.

Při experimentech prováděných na vozidlových simulátorech získáváme data, která se následně vyhodnocují – ta mohou být subjektivní nebo objektivní. Subjektivní data jsou ta, která získáváme dotazováním řidiče v průběhu experimentu či po něm. Objektivní data získáváme buď přímo z vozidla (jedná se např. o rychlost, trajektorii jízdy, polohu pedálů, úhel natočení volantu, zařazený rychlostní stupeň, otáčky motoru), nebo od řidiče (měří se například jeho reakční čas, pohyby hlavy, EEG či pohyb očí – fokusu pomocí EyeTrackingu).

Obrázek 2: Měření pohybu očí během jízdy pomocí EyeTrackingu (místo pohledu řidiče je označeno červeným křížem)

3.2 Součásti vozidlového simulátoru

Každý vozidlový simulátor se skládá z projekce a místa pro řidiče, ze kterého ovládá vozidlo – to může být řešeno jako kokpit vozidla, jeho část nebo jednoduše pouze pomocí sedačky. Softwarové vybavení simulátoru tvoří především matematický model chování vozidla, software obstarávající vizualizaci, záznam jízdy řidiče a nástroje k vyhodnocení jeho výkonu, řízení běhu scén a generátor událostí, případně řízení pohybu plošiny. Nedílnou součástí systému simulátoru je virtuální scéna a virtuální scénář tvořící virtuální prostředí, v němž se řidič pohybuje – to je mu promítáno na obrazovky či plátna okolo vozidla.

Virtuální scéna je virtuální prostředí vytvořené pomocí počítačové grafiky, které se promítá okolo vozidla, a ve kterém se řidič pohybuje. Je nezbytné, aby toto prostředí co nejvíce připomínalo prostředí reálné a řidič se tak při jízdě cítil jako v reálném provozu – aby výsledky z měření řidičova chování byly co nejméně ovlivněny.

Obrázek 3: Ukázka virtuální scény

Virtuální scénář je pak soubor událostí, které se dějí ve virtuálním prostředí během simulace, a na které musí řidič reagovat. Jedná se například o různé překážky na trase či definice dynamických objektů, které oživují jinak statickou virtuální scénu. Tyto události jsou do scénářů přidávány pomocí animací. Příkladem jsou třeba pohybující se vozidla či chodci, blikající billboardy a semafory, pohyblivé mraky na obloze nebo srnka, která vyběhne z lesa na silnici před vozidlem řidiče.

3.3 Virtuální prostředí

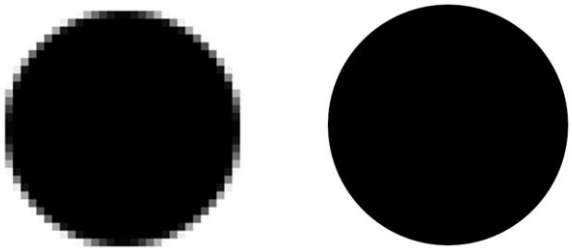
Kvalita virtuálního prostředí pro interaktivní vozidlové simulátory má značný vliv na objektivitu výsledků získaných z prováděných experimentů. Jelikož experimenty na vozidlových simulátorech bývají dost specifické a zaměřené na nějaký konkrétní problém, je nutné pro každý experiment vytvořit nový scénář. Modelování virtuálního prostředí pro konkrétní scénář vyžaduje dlouhé přípravy.

Modely ve scénářích pro vozidlové simulátory musí splňovat dvě základní podmínky – měly by vypadat co nejreálněji, aby řidiči navodily iluzi reálného světa, ale zároveň musí být co nejjednodušší, aby se dostatečně rychle vykreslovaly v reálném čase během experimentu. Je tedy nutné najít kompromis pro zajištění kvalitního scénáře.

3.3.1 Počítačová grafika

Počítačová grafika využívá počítač k tvorbě umělých grafických objektů a úpravě objektů nasnímaných z reálného světa. Dělí se na 2D a 3D grafiku. 2D grafika se zabývá prací s obrázky a fotografiemi, zatímco 3D grafika řeší reprezentaci objektů v prostoru a jejich zobrazení v počítači.

2D grafika může být rastrová nebo vektorová. V rastrové grafice obraz tvoří pravidelnou síť pixelů – každý pixel přitom nese informaci o barvě, jasu, průhlednosti atd. Vektorová grafika oproti tomu popisuje obraz pomocí přesných geometrických dat, díky čemuž mohou mít jednotlivé tvary ostré hrany i při jejich přiblížení. Rozdíl mezi rastrovou a vektorovou grafikou je vidět na obrázku č. 4.



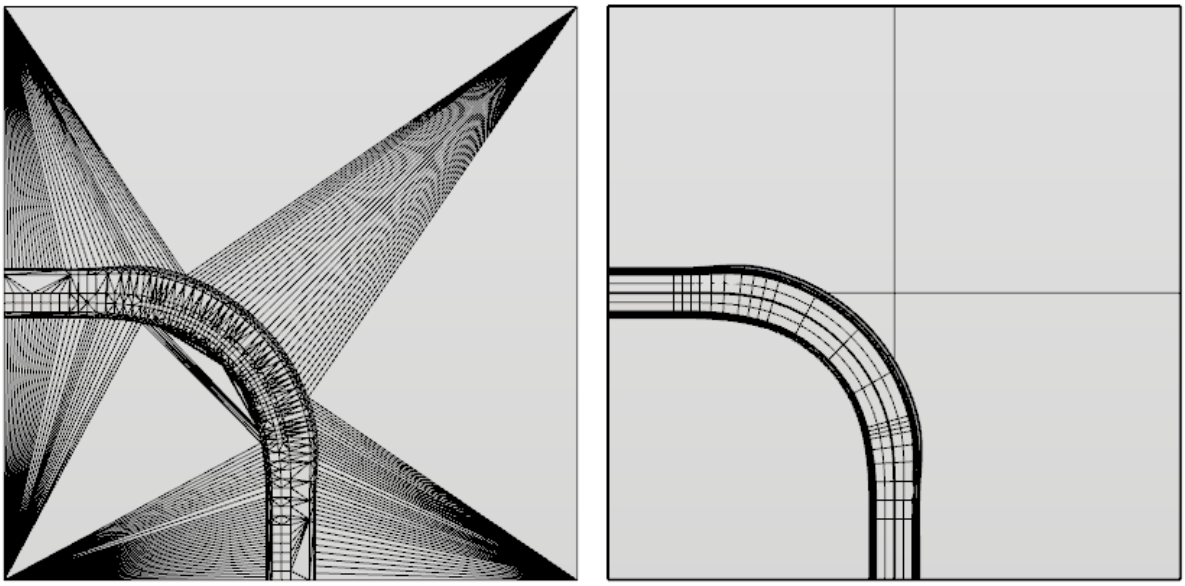
Obrázek 4: Kruh znázorněný pomocí rastrové (vlevo) a vektorové grafiky (vpravo)

V 3D grafice existují tři metody reprezentace objektů – objemová (těleso je popsáno pomocí jeho objemu, a to sítí prostorových buněk), procedurální (definice 3D modelu tělesa na základně jeho charakteristik pomocí algoritmů) a hraniční (popisován je povrch těles pomocí množiny hraničních geometrických prvků – bodů, úseček atd.).

Nejvíce využívaný způsob popisu 3D objektů je popis pomocí hraniční reprezentace, která je pro uživatele nejpřirozenější. Pracuje s ní i většina 3D modelovacích programů.

Existuje mnoho metod pro tvorbu objektů pomocí hraniční reprezentace, nejrozšířenějšími metodami jsou TIN a NURBS. TIN, neboli Triangulated Irregular Network, je metoda, při které je objekt popsán pomocí nepravidelné sítě trojúhelníků, přičemž přesnost popisu objektu je dána počtem použitých trojúhelníků – čím hustší je trojúhelníková síť, tím je model hladší a přesnější, ale jeho vykreslení je zároveň náročnější na paměť počítače. U metody NURBS, tedy Non-Uniform Rational B-Spline, je objekt popsán pomocí sítě matematických křivek – výsledný model je díky tomu velmi přesný, a přitom o sobě nenese příliš velké množství informací. Rozšířením metody NURBS vznikla metoda T-Spline s jednodušší definicí modelu, a to díky možné existenci spojů ve tvaru písmene T, což umožňuje redukci křivek potřebných k popisu objektu.

Rozdíl mezi metodami TIN a NURBS je vidět na obrázku č. 5.

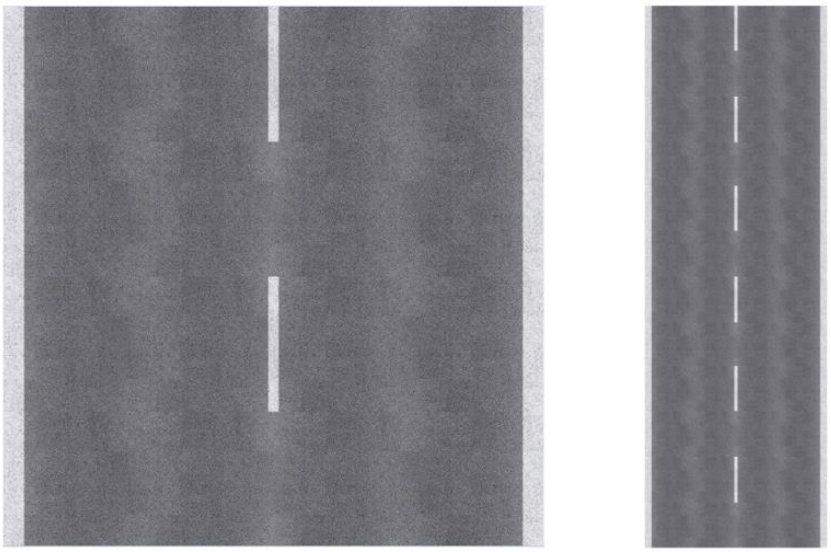


Obrázek 5: Segment s obloukem R30 znázorněný metodou TIN (vlevo) a NURBS (vpravo)

Při tvorbě scénářů využíváme oba druhy počítačové grafiky, 2D grafiku i 3D grafiku. Pomocí 2D grafiky vytváříme textury, které následně nanášíme na objekty vytvořené 3D grafikou (pomocí 3D modelování). Potažením modelu texturou (obrázkem, fotografií) dodáme modelu optické vlastnosti a barvu a vytvoříme tak iluzi reálného objektu. Vhodným mapováním pak docílíme použití textury ve správném měřítku a s odpovídajícím poměrem stran.

Textury by měly být vytvářeny tak, aby byly využitelné pro větší množství modelů – např. pro model velkého stromu použijeme celou texturu, pro malý strom využijeme část stejné textury. Objekty budou vypadat rozdílně, ale použijeme menší počet textur.

Pro nanesení textury na objekty, které mají velké stejně vypadající plochy, se používají tzv. bezešvé textury. Jedná se o textury, které vykreslují pouze malou část povrchu, ale jejich okraje jsou upraveny tak, aby na sebe dokonale navazovaly a opakováním této textury se obrázkem pokryla celá plocha. Příkladem jsou velké travnaté plochy nebo povrch silnice, který je vidět na obrázku č. 6 níže.



Obrázek 6: Textura povrchu silnice (vlevo) a její použití (vpravo)

U objektů, kterým je třeba definovat průhlednost, se využívají textury s alfa kanálem. Pro tyto případy je třeba zkombinovat dvě textury – Diffuse Map, která definuje barvu objektu, a Transparency Map definující průhledné oblasti (černobílá textura, kde černé plochy budou průhledné, bílé plochy ponesou barvu z Diffuse Map). Příklad takovéto dvojice textur je vidět na obrázku č. 7.

Obrázek 7: Textura smrku s alfa kanálem

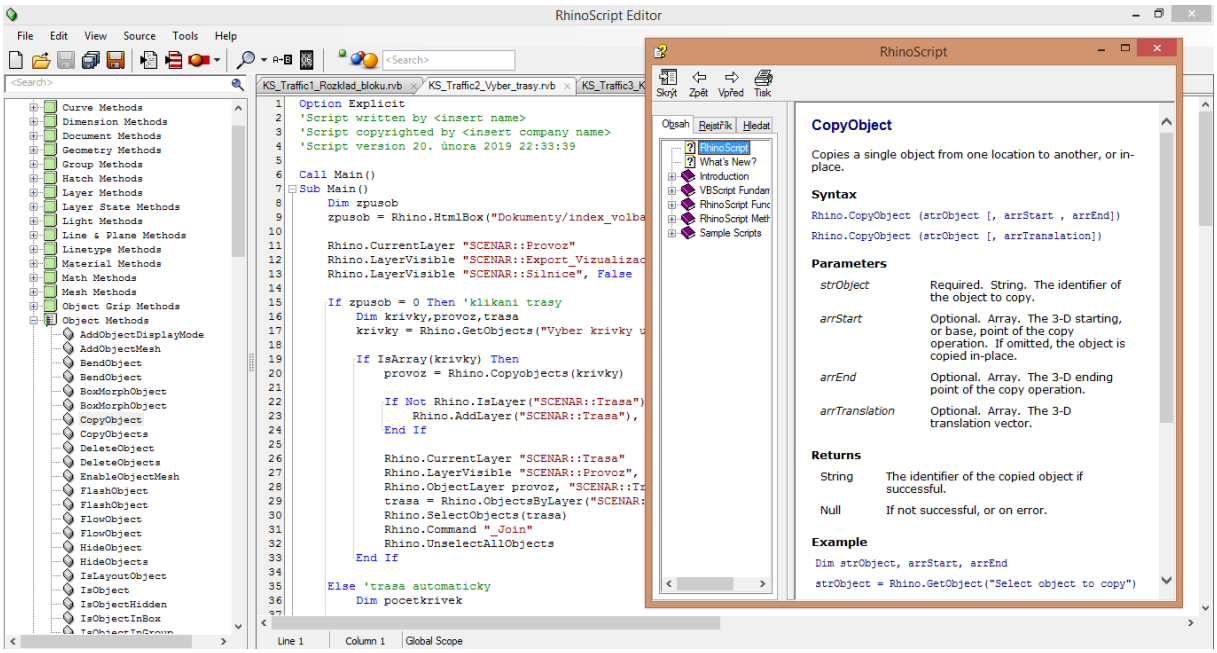
3.3.2 3D modelování

Pomocí 3D modelování vytváříme a tvarujeme modely – objekty do virtuálního prostředí. Tyto modely jsou pro virtuální scénáře nejčastěji definovány pomocí metody TIN (nepravidelnou trojúhelníkovou sítí), přičemž pro dostatečně rychlé vykreslování objektů v reálném čase během simulace by měly být nízkopolygonální (lowpoly), to znamená být tvořeny co nejmenším počtem trojúhelníků a jednoduchou texturou.

Příklady 3D modelovacích programů jsou Rhinoceros, AutoCAD 3DS Max, Sketchup, Blender a další. Pro tuto práci byl využit program Rhinoceros, který umožňuje vytvářet velmi přesné modely. Základní reprezentací, ve které jsou objekty v tomto programu vytvářeny, je NURBS, v níž je možné přesně vytvořit požadovaný tvar tělesa. Všechny modely je pak možné převést na trojúhelníkovou reprezentaci TIN, která je vhodná pro využití ve scénářích pro vozidlové simulátory – při tomto převodu můžeme nastavit hustotu trojúhelníkové sítě a optimalizovat tak počet trojúhelníků v modelu.

Většinu 3D modelovacích programů lze rozšířit o nové funkce pomocí skriptování – s využitím jednoduchého programovacího jazyka je možné vylepšovat původní program a usnadnit si tak práci při modelování zautomatizováním některých kroků.

Pomocí příkazů vytvořených ve skriptu lze spustit většinu funkcí, které 3D modelovací program nabízí, a s využitím cyklů, podmínek a dalších funkcí je možné vytvářet nové užitečné funkce programu.



Obrázek 8: RhinoScript Editor pro vytváření skriptů

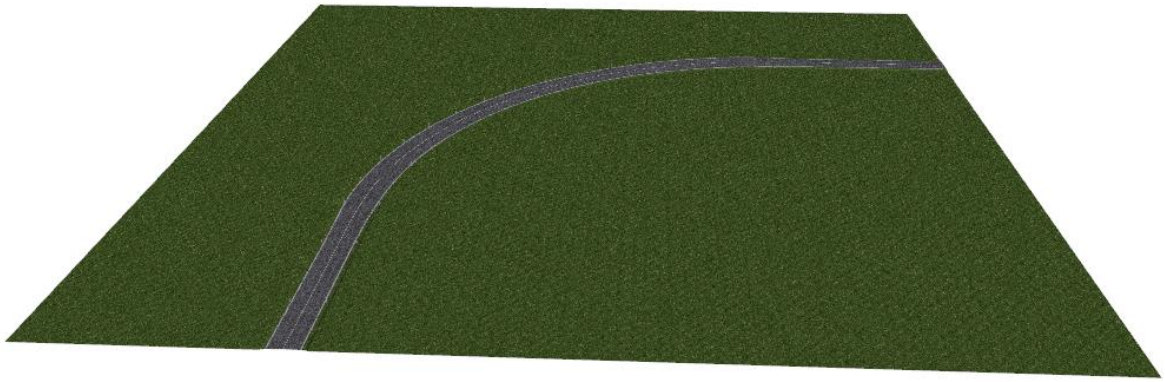
Pro rozšiřování 3D modelovacích programů se nejčastěji využívají skriptovací jazyky jako Python, Ruby, VBScript, Coffee a MAXScript.

Program Rhinoceros umožňuje použití jazyků Python a VBScript. VBScript je rozšířenější a využívá zjednodušenou syntaxi programovacího jazyka Visual Basic a v programu Rhinoceros je nazvaný RhinoScript. Pro tvorbu skriptů zde existuje speciální editor nazvaný RhinoScript Editor, který je uživatelsky přívětivý a obsahuje zároveň knihovnu podporovaných příkazů s ukázkami jejich využití. Díky tomu je pochopení syntaktických pravidel skriptovacího jazyka jednodušší a práce s ním efektivnější.

4 Plugin pro generování segmentů komunikací

Tato diplomová práce navazuje na moji bakalářskou práci s názvem „Příprava počítačových 3D modelů segmentů komunikací pro vozidlové simulátory“ [3]. V rámci ní jsem vytvořila plugin s názvem KS_Software pro 3D modelovací program Rhinoceros, díky němuž lze rychle a efektivně vytvářet scénáře pro vozidlové simulátory, a to na základě spojování předem připravených segmentů komunikací. Plugin byl vytvořen pomocí skriptu ve skriptovacím jazyce RhinoScript.

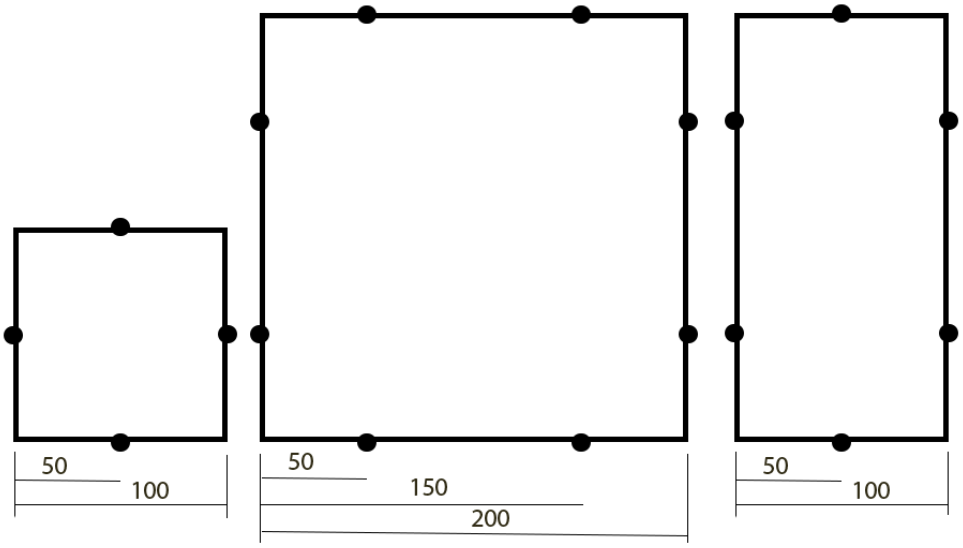
Pro tento plugin jsem do databáze připravila několik druhů základních segmentů, které obsahují část silniční komunikace a travnaté plochy kolem ní. V databázi jsou segmenty s přímou silniční komunikací (s plnou i přerušovanou čarou), segmenty s obloukem o různých poloměrech od R30 až po R220, segmenty s různými druhy křižovatek či s nájezdem na polní cestu a přechodové segmenty mezi extravilánem a intravilánem[1]. Silniční komunikace na segmentech jsem modelovala s využitím pluginu RoadCreator for Rhino, který vytvořil Ing. Adam Orlický v rámci své diplomové práce s názvem „Automatická tvorba silniční infrastruktury ve 3D pro vozidlové simulátory“ [2], silnice tedy odpovídají české technické normě ČSN 73 6101 – Projektování silnic a dálnic a technickým podmínkám.



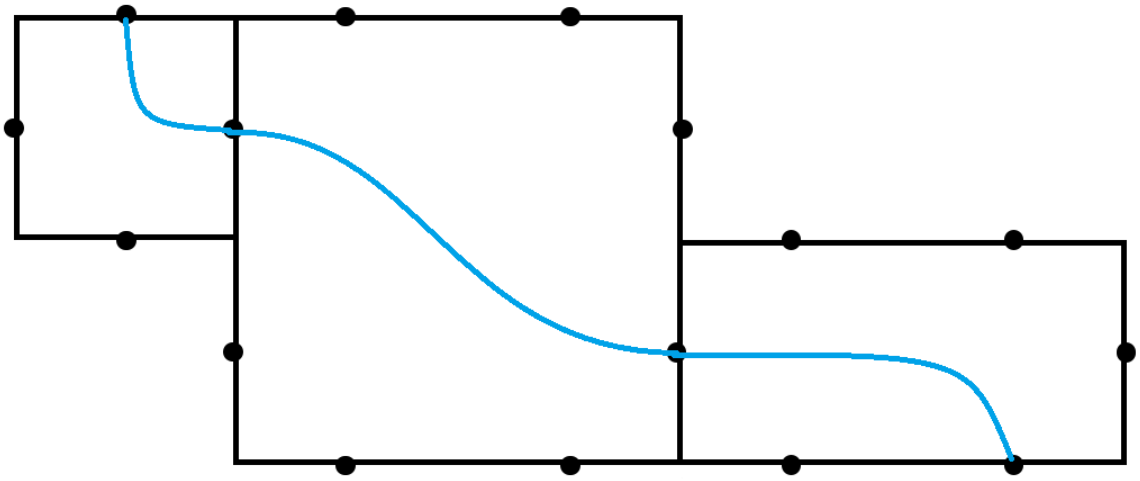
Obrázek 9: Ukázka jednoho ze základních segmentů – segment s obloukem o poloměru R100

Všechny segmenty mají přesně definované rozměry (násobky stovek metrů) a vstupy a výstupy silnice (50 metrů od okraje segmentu). Uživatel vytvářející scénář pomocí této metody pak tyto segmenty postupně generuje a skládá je za sebe na principu puzzle, proto musí být zajištěná dokonalá návaznost všech těchto segmentů.

[1] Intravilán – prostředí s městskou zástavbou, extravilán – prostředí mimo městskou zástavbu

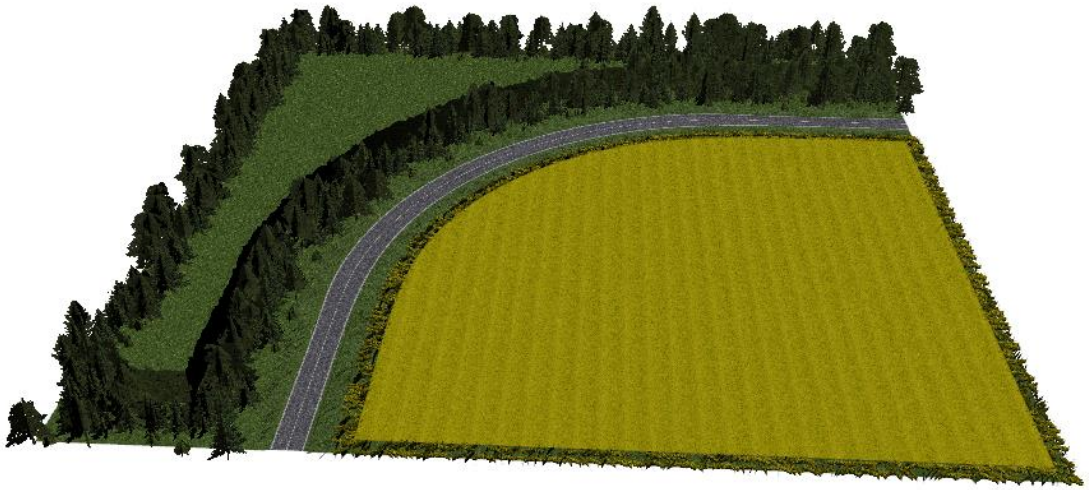


Obrázek 10: Rozměry jednotlivých segmentů a umístění vstupů a výstupů silniční komunikace



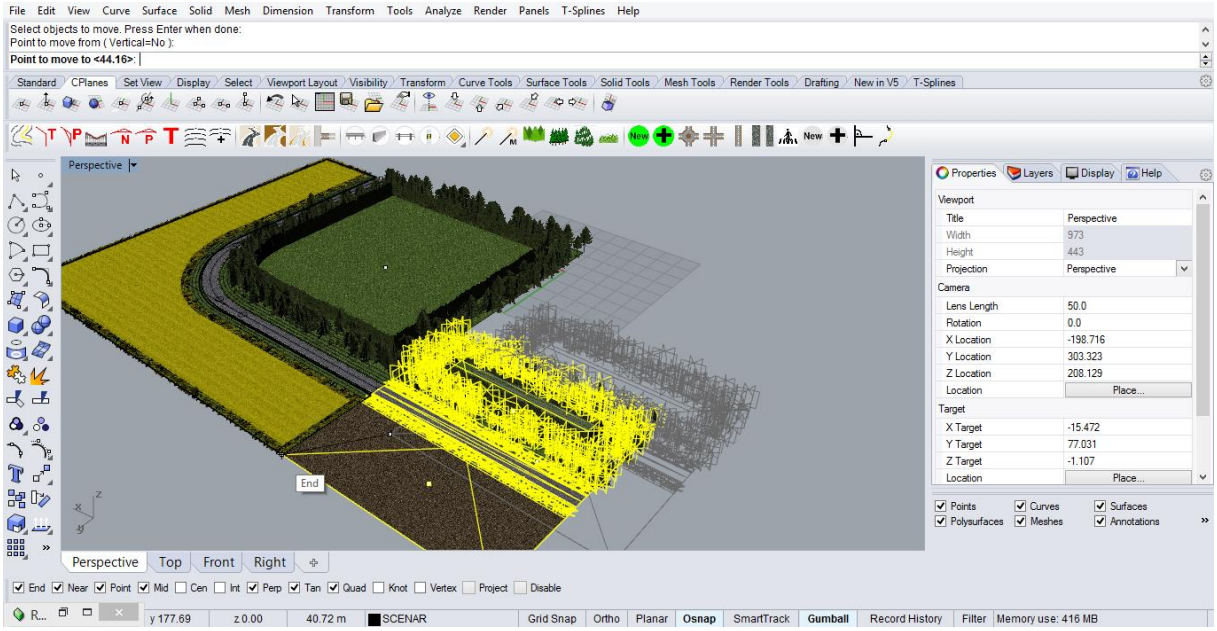
Obrázek 11: Princip napojování segmentů

V rámci generování segmentů má navíc uživatel možnost na každý segment přidat na plochy kolem silnice některou z forem vegetace, které jsou rovněž předpřipravené v databázi. Má na výběr z několika různých druhů lesů a polí. V případě lesů se jednotlivé stromy generují ve třech řadách a za specifických podmínek, díky kterým je každý les jedinečný a co nejvíce realistický – každý strom se natáčí v náhodném úhlu, mění svoji velikost v náhodném poměru a vzdálenost mezi jednotlivými stromy na křivce je rovněž náhodná. Pro navození iluze neprůhledného lesa je za touto skupinou stromů generována kulisa lesa, která snižuje potřebný počet modelů stromů.



Obrázek 12: Příklad hotového segmentu (vlevo jehličnatý les, vpravo řepkové pole)

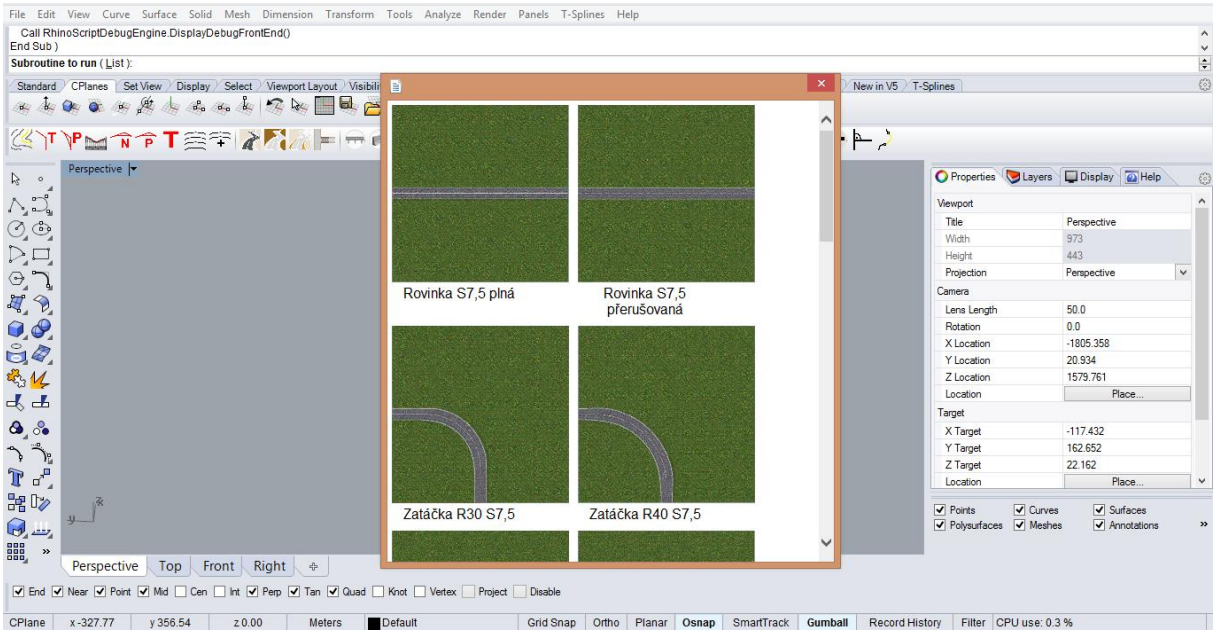
Na konci přípravy každého segmentu jsou všechny modely tvořící daný segment uloženy do bloku (tj. sloučeny do jednoho objektu), aby se dalo s jednotlivými segmenty snadněji posouvat a skládat je tak za sebe.



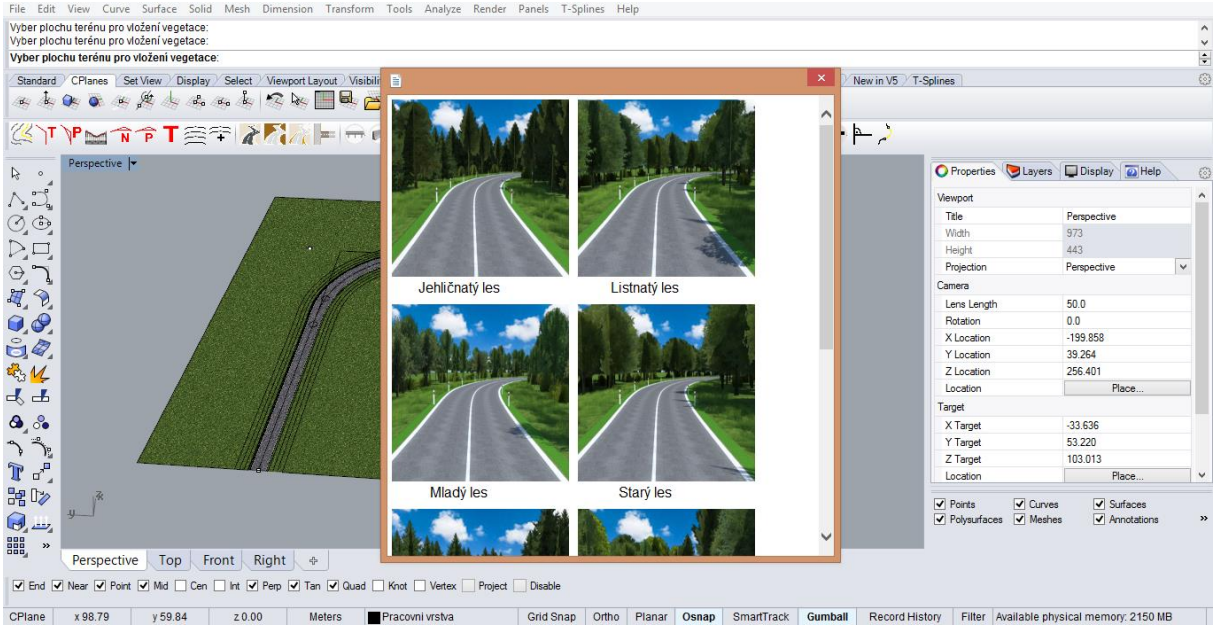
Obrázek 13: Vytváření scény skládáním segmentů komunikací za sebe [3]

I přes relativně malý počet modelů v databázi (16 typů segmentů a 8 typů vegetace) lze vzájemnými kombinacemi dosáhnout velkého množství různých variant, je tedy možné vytvářet stále unikátní scénáře.

Celý plugin je navržen tak, aby byl snadno využitelný i pro uživatele bez větších znalostí 3D modelování – uživatel pouze vybírá z předpřipravených segmentů pomocí HTML rozhraní, ve kterém vidí náhledy jednotlivých segmentů s popisky, podobně jako při následném vybírání vegetace pro každou volnou plochu na segmentu.

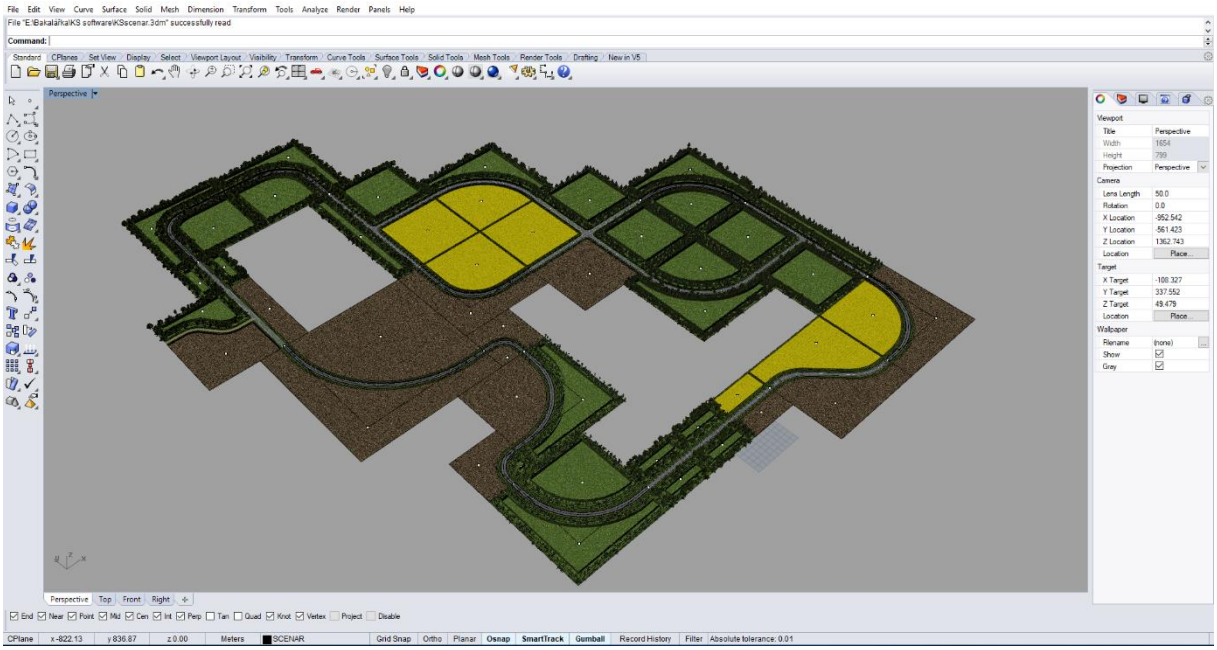


Obrázek 14: HTML rozhraní pro výběr segmentu [3]



Obrázek 15: HTML rozhraní pro výběr typu vegetace [3]

Tato metoda velmi usnadňuje vytváření virtuálních scénářů pro experimenty prováděné na vozidlovém simulátoru, a to hlavně díky velké úspoře času, který by byl potřeba na 3D modelování. S využitím pluginu je samotná scéna, jejíž vytvoření by běžně zabralo dny až týdny, hotová v řádu minut, a uživatel tak může ušetřený čas věnovat například vytváření dalších důležitých objektů potřebných pro daný experiment (např. dopravní značky, budovy). Úspora času při práci na scénářích s využitím KS_Software se potvrdila provedením experimentu v rámci mé bakalářské práce [3].



Obrázek 16: Příklad hotové scény vytvořené ze segmentů (délka okruhu: cca 1.5 km, doba přípravy: 45 minut) [3]

Z důvodu značné efektivity této metody jsem se rozhodla navázat na tuto práci a ještě více zvýšit její využitelnost, a sice přidáním možnosti generovat provoz do scénářů vytvořených touto metodou. Přidáním provozu se z původní statické scény stane dynamická, která se více přiblíží iluzi reálného prostředí a bude ji možné využívat pro velké množství nových typů experimentů.

5 Metody tvorby provozu do vozidlových simulátorů

Aby se řidič jedoucí v simulátoru cítil více jako při jízdě v reálném prostředí, je vhodné virtuální prostředí doplnit různými animacemi, které danou scénu oživí – jedná se například o pohyb mraků, animované billboardy, pohyb chodců, ale také o pohyb ostatních vozidel objevujících se ve scénáři. Takováto vozidla tvoří provoz ve virtuálním scénáři.

5.1 Typy simulace provozu

Provoz je pro simulace ve vozidlovém simulátoru nejdůležitější prvek. Můžeme ho podle závislosti animovaných vozidel na jízdě řidiče v simulátoru rozdělit do tří kategorií, na provoz neinteraktivní, interaktivní a autonomní.

5.1.1 Provoz neinteraktivní

Jedná se o simulaci, při které mají jednotlivá vozidla přesně určenou svoji trasu a parametry jízdy, které se v čase nemění a nereagují na stav okolí.

Pro tuto simulaci je pro každé vozidlo známa křivka, po které pojede. Tato křivka je rozdělena body, a v každém tomto bodě jsou určeny konkrétní parametry popisující pohyb a aktuální rychlost vozidla:

a) Čas – čas v milisekundách, ve kterém se vozidlo v daném bodě nachází,
b) Souřadnice polohy – souřadnice v ose x, y a z určující polohu vozidla,
c) Souřadnice natočení – úhel natočení v osách x, y a z určující klopení, klonění a stáčení vozidla.

Při neinteraktivní simulaci provozu nejsou vozidla schopna reagovat na jízdu řidiče v simulátoru, používá se tedy spíše k oživení okolí, pro animaci vozidel jedoucích např. po jiné silnici než řidič simulátoru – tedy vozidel, se kterými řidič nepřijde do kontaktu.

5.1.2 Provoz interaktivní

Při interaktivní simulaci se pohyb vozidel řídí pomocí složitějších algoritmů, které reagují na podněty z okolí a přizpůsobují jim svoji jízdu a chování. Vozidla mají opět danou svoji trasu po křivkách, po kterých se mohou pohybovat, ale svoji jízdu mohou přizpůsobovat jízdě řidiče simulátoru – to znamená, že pokud řidič v simulátoru zpomalí, animované vozidlo jedoucí za ním do něj nenarazí, ani neprojede skrz něj, ale zpomalí taky.

5.1.3 Provoz autonomní

V autonomním režimu simulace již vozidla mají svoji vlastní umělou inteligenci, pomocí níž se rozhodují, kudy a jak pojedou. Jejich trasa není dána křivkami, ale vozidla si ji určují sama, např. tak, že rozpoznávají silnici podle jejího materiálu či barvy a jedou v daném jízdním pruhu. Když dorazí na křižovatku, sama se rozhodnou, jakým směrem se vydají – pomocí

samoučících se algoritmů. Na jízdu řidiče v simulátoru jsou schopny reagovat stejně jako při provozu interaktivním.

5.2 Generování provozu do segmentů

V této práci jsem vytvořila novou metodu generování provozu (vycházející z neinteraktivního typu provozu), a to do scénářů vytvořených ze segmentů komunikací. Při takovémto typu scénářů je možné na každý segment předpřipravit křivky určující trasu, po které se bude moci provoz pohybovat. Po napojení segmentů lze i tyto křivky spojit do celé trasy, a to buď ručně, nebo na základě algoritmu vyhledávání cesty mezi počáteční a koncovou křivkou.

Výsledná trasa je poté rozdělena body a pro každý tento bod jsou určeny parametry, které definují chování vozidla v daném bodě (čas, souřadnice polohy a natočení). Všechny tyto údaje jsou pak exportovány do animačního souboru, s využitím kterého již lze spustit animaci přímo ve vozidlovém simulátoru.

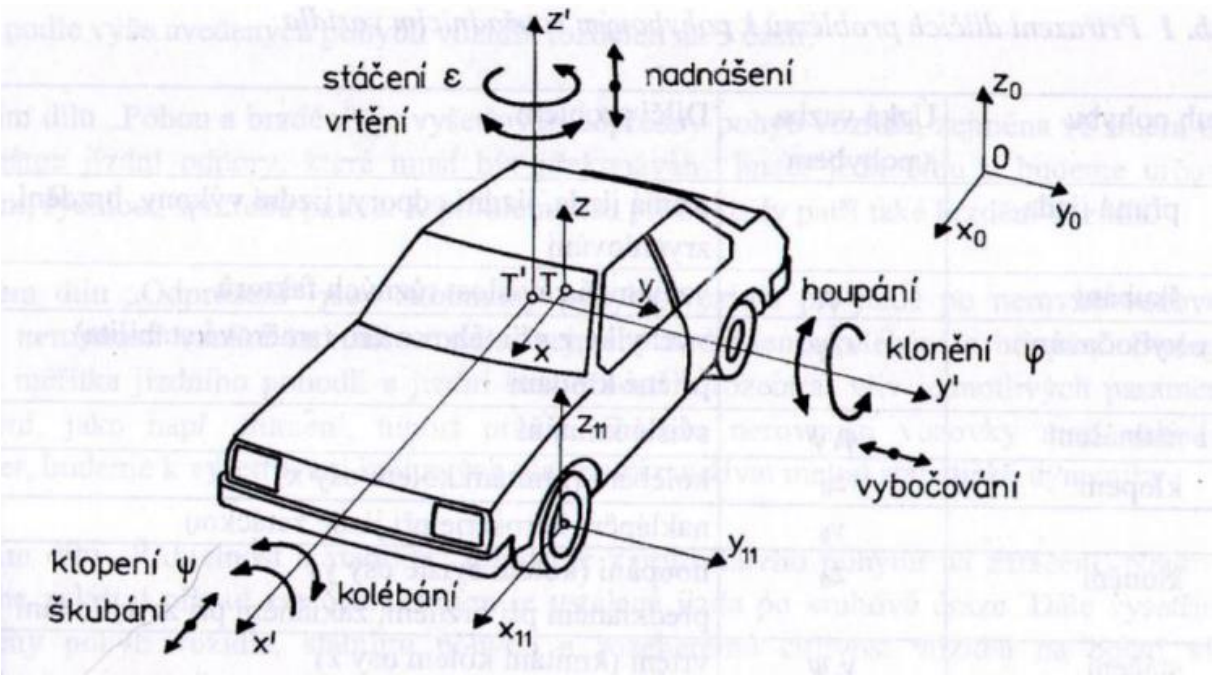
Tato metoda je vytvořena pro vozidlový simulátor na Fakultě dopravní ČVUT, který využívá svůj vlastní 3D grafický engine (tj. vizualizační software) s touto konkrétní definicí potřebných animačních souborů.

Chování provozu generovaného touto metodou vychází z reálných dat. Jednak jsou to reálné parametry vozidel, které má uživatel tvořící provoz na výběr (hmotnost, součinitel odporu vzduchu, výkon motoru, čelní plocha vozidla), a jednak jsou to také reálné hodnoty rychlosti při průjezdu směrovými oblouky o různém poloměru.

6 Chování vozidla během jízdy

6.1 Popis pohybu vozidla

Pohyb vozidla v průběhu času můžeme v prostoru popsat jeho polohou a natočením vůči osám x, y a z, viz obrázek č. 17 níže. Tento popis pomocí souřadnic můžeme využít i při simulaci pohybu vozidla při generování provozu do vozidlových simulátorů.



Obrázek 17: Souřadnice určující polohu a natočení vozidla [4]

6.2 Dynamika jízdy vozidla

To, jak se vozidlo chová během své jízdy, lze popsat pomocí dynamiky jízdy vozidla. Dynamiku jízdy vozidla můžeme rozdělit na:

a) podélnou dynamiku (zkoumající rozjezd a brzdění),
b) příčnou dynamiku (zkoumající průjezd zatáčkou),
c) svislou dynamiku (zkoumající pérování vozidla). [4]

6.2.1 Podélná dynamika

V rámci podélné dynamiky mají na jízdu vozidla vliv jízdní odpory. Celkový jízdní odpor F_k je určen součtem všech dílčích odporů, na něm je pak závislý potřebný výkon hnacího mechanismu vozidla P_k, který je nutný na překonání odporů:

$$F_k = O_f + O_v + O_s + O_z \tag{1}$$

$$P_k = F_k * v \tag{2}$$

$$P_k = (G * \cos \propto * f + \frac{1}{2} * S_x * c_x * \rho * v^2 + G * \sin \propto + \vartheta * m * a) * v \tag{3}$$

F_k … celkový jízdní odpor, tj. potřebná hnací síla na kolech vozidla [N]

P_k … výkon na kolech vozidla [W]

v … rychlost vozidla [m/s].

Jednotlivé jízdní odpory jsou: odpor valivý O_f, odpor vzduchu O_v, odpor stoupání O_s a odpor zrychlení O_z. Jsou to síly, které působí proti pohybu vozidla. Lze je určit pomocí následujících vztahů:

a) **odpor valivý – O_f**, vzniká deformací pneumatiky a vozovky

$$O_f = F_N * f \tag{4}$$

F_N … normálová síla, $F_N = G * \cos\alpha = m * g * \cos\alpha$

m … hmotnost vozidla [kg]

g … tíhové zrychlení [m/s^2], $g = 9{,}81\ m/s^2$

α … úhel stoupání [°], při $\alpha = 0$ je $F_N = G$

f … součinitel valivého odporu kola [-], pro pneu-asfalt je $f = 0{,}01$

b) **odpor vzduchu – O_v**, vzniká prouděním vzduchu kolem vozidla během jízdy (část vzduchu proudí kolem horní části karoserie a část prochází mezi spodní částí karoserie a povrchem vozovky, odpor vzniká kvůli víření za vozidlem)

$$O_v = \frac{1}{2} * S_x * c_x * \rho * v^2 \tag{5}$$

S_x … čelní plocha vozidla [m^2]

c_x … součinitel vzdušného odporu [-], pro osobní automobily je $c_x = 0{,}3 - 0{,}4$

ρ … hustota vzduchu [kg/m^2]

v … rychlost vozidla [m/s]

c) **odpor stoupání – O_s**, vzniká při jízdě do svahu, působí v těžišti vozidla (pro jízdu do svahu platí v rovnici znaménko plus, pro jízdu ze svahu znaménko minus – sinová složka tíhy vozidla pak není odporem, ale naopak vozidlo pohání)

$$O_s = \pm\, G * \sin\alpha \tag{6}$$

G … tíhová síla [N], $G = m * g$

α … úhel stoupání [°], tj. úhel, který svírá rovina vozovky s vodorovnou rovinou

d) **odpor zrychlení – O_z**, vzniká působením setrvačné síly proti směru zrychlení při zrychlování vozidla, skládá se z odporu zrychlení posuvné části a odporu zrychlení otáčejících se částí

$$O_z = \vartheta * m * a \tag{7}$$

ϑ … součinitel rotačních částí, $\vartheta = 1 + \frac{J_c}{m}\left(\frac{2}{D_k}\right)^2$

J_c … celkový moment setrvačnosti rotačních částí [kg*m^2]

D_k … průměr kola [m]

a … zrychlení [m/s^2].

Akcelerace a decelerace:

Aktuální zrychlení (akcelerace) vozidla závisí zejména na jeho technických parametrech (P_k – výkon motoru, S_x – čelní plocha vozidla, c_x – součinitel vzdušného odporu, m – hmotnost vozidla; údaje lze většinou zjistit od výrobce vozidla), aktuální rychlosti a míře sešlápnutí plynového pedálu.

Při běžné jízdě řidič málokdy využívá maximální možnou akceleraci, ale zrychluje s tzv. komfortní akcelerací. Jedná se o maximální zrychlení, při kterém je jízda pro posádku vozidla pohodlná. Hodnota tohoto zrychlení je definována například v modelu Intelligent Driver Model (IDM), a její typická hodnota pro osobní automobil je podle [1] stanovena takto:

- Komfortní zrychlení – akcelerace $a_0 = 1{,}0\ m/s^2$,
- Komfortní zpomalení – decelerace $b_0 = 1{,}5\ m/s^2$.

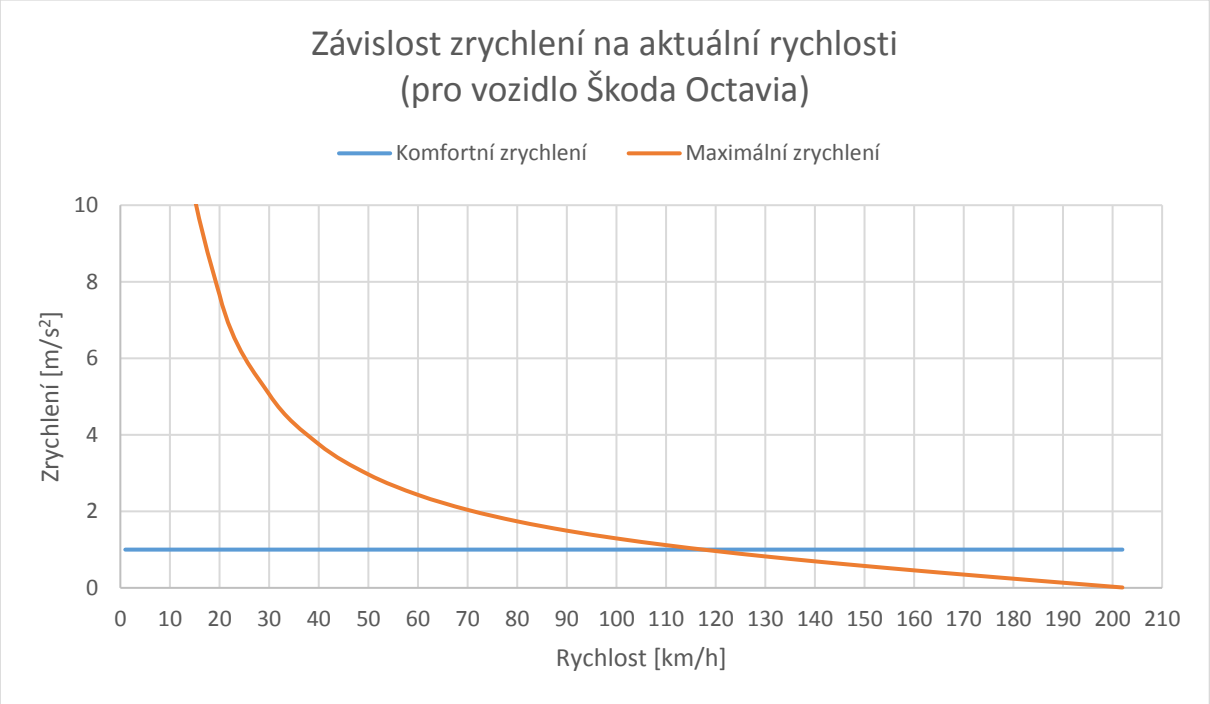
Maximální zrychlení vozidla je však limitováno jeho technickými parametry, a při dosažení určité rychlosti může být zrychlení nižší, než je hodnota komfortní akcelerace – to znamená, že vozidlo již není schopno zrychlovat s maximální komfortní hodnotou akcelerace, ale zrychluje pomaleji. Hodnotu maximálního zrychlení konkrétního vozidla je možné na základě předchozích vztahů vypočítat pro danou hodnotu aktuální rychlosti.

Pro realistickou simulaci pohybu vozidla je třeba určit hodnotu jeho zrychlení pro každou hodnotu aktuální rychlosti – zejména je třeba určit bod, od kterého je hodnota maximálního zrychlení nižší, než je hodnota zrychlení komfortního. Hodnota rychlosti, od které již není možné zrychlovat s komfortním zrychlením, se liší u každého typu vozidla v závislosti na jeho technických parametrech (výkon motoru P [W], hmotnost vozidla m [kg], čelní plocha vozidla S_x [m^2], součinitel vzdušného odporu c_x [-]).

Na obrázku č. 18 níže je vidět příklad grafického znázornění závislosti zrychlení vozidla na jeho aktuální rychlosti. Ukázkový graf je vykreslen pro vozidlo Škoda Octavia, které dosahuje maximální rychlosti 203 km/h. Modrou barvou je znázorněno komfortní zrychlení, jehož průběh je konstantní (1 m/s^2), oranžová křivka znázorňuje průběh teoretického maximálního zrychlení vozidla při dané rychlosti, které je vypočteno z technických parametrů daného vozidla. Ty jsou popsány v tabulce č. 1.

Tabulka 1: Parametry vozidla Škoda Octavia [21]

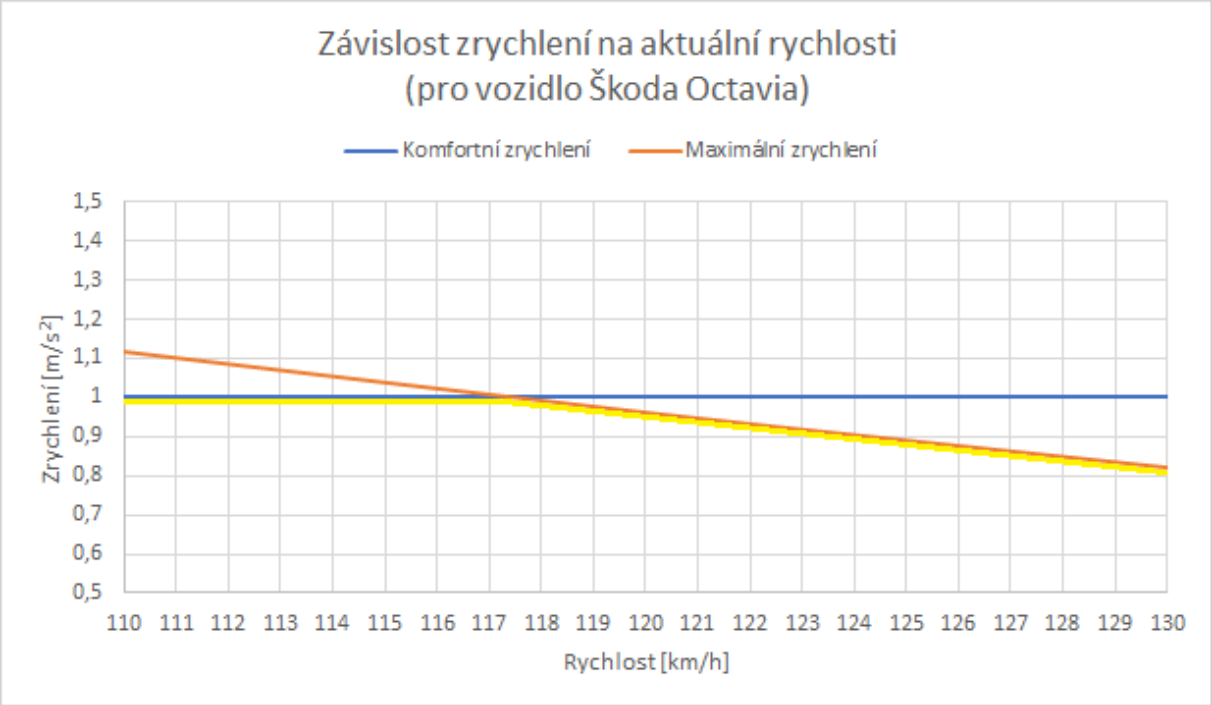
Škoda Octavia	
P [W]	85 000
m [kg]	1 775
S_x [m^2]	2,17
c_x [-]	0,28



Obrázek 18: Grafické znázornění závislosti zrychlení na aktuální rychlosti vozidla

Od bodu, kde se protíná komfortní zrychlení s maximálním, již není možné zrychlovat s komfortním zrychlením – z grafu vyplývá, že k tomu v případě tohoto vozidla dochází cca při rychlosti 117 km/h. Při této a vyšší rychlosti tedy vozidlo zrychluje již pouze se zrychlením maximálním (méně než 1 m/s^2).

Detail průběhu zrychlení v okolí tohoto bodu je vidět na obrázku č. 19. Výsledné zrychlení, se kterým se vozidlo při jednotlivých hodnotách aktuální rychlosti pohybuje, je zde zvýrazněno žlutou barvou.



Obrázek 19: Detail okolí bodu, od kterého je maximální zrychlení nižší než komfortní

6.2.2 Příčná dynamika

Příčná dynamika popisuje průjezd vozidla směrovým obloukem, při čemž na vozidlo působí:

a) odstředivá/dostředivá síla – F_o

$$F_o = m * a_o = m * \frac{v^2}{R} \tag{8}$$

R … poloměr křivosti oblouku [m]

b) odstředivé/dostředivé zrychlení – a_o

$$a_o = v * \dot{\varepsilon} \tag{9}$$

$\dot{\varepsilon}$ … úhlová rychlost stáčení vozidla [1/s]

c) boční síla větru – N

$$N = \frac{1}{2} * \rho * S_y * c_y * v^2 \tag{10}$$

S_y … boční plocha vozidla [m^2]

c_y … boční součinitel vzdušného odporu [-].

6.2.3 Svislá dynamika

Svislá dynamika se zabývá silami působícími ve svislém směru – tj. gravitační síla:

$$G = m * g. \tag{11}$$

Jízdou pneumatiky po nerovném povrchu vozovky dochází ke kmitání vozidla. Díky odpružení pomocí pérování mohou kola vozidla lépe kopírovat podložku a dochází ke zmírnění otřesů s ohledem na pohodlí a bezpečnost posádky a namáhání součástí vozidla.

Kmitání vozidla při jízdě po nerovném povrchu je třeba tlumit, a zajistit tak dobrý kontakt pneumatiky s vozovkou a dostatečnou přítlačnou sílu kola na vozovku. K tlumení kmitů slouží tlumiče, které mění frekvenci kmitání.

6.3 Chování vozidla při animaci

Pro definici pohybu vozidla při animaci jsou důležité zejména souřadnice určující polohu a natočení vozidla v prostoru, rychlost jízdy vozidla a hodnoty zrychlení a zpomalení při různých situacích.

PRAKTICKÁ ČÁST

7 Měření rychlosti při průjezdu směrovými oblouky

V pluginu, který jsem vytvořila v rámci této diplomové práce, vytvářím nový způsob generování provozu do vozidlových simulátorů ve scénářích vytvořených pomocí KS_Software, tj. scénářích poskládaných z předpřipravených segmentů komunikací. Tyto segmenty obsahují několik typů silničních komunikací – přímou, oblouky o různých poloměrech, křižovatky a jiné. Pro realistické generování provozu na takto vytvořeném scénáři je tedy nutné, aby byla pro každý úsek definována ideální průjezdná rychlost.

7.1 Arduino

Za tímto účelem jsem se rozhodla provést experiment, pomocí něhož tuto rychlost pro každý segment určím. K tomu jsem s využitím Arduina sestavila vlastní měřicí zařízení.

„Arduino je otevřená (open source) elektronická platforma, založená na uživatelsky jednoduchém hardware a software.“ [7]

Arduino je nástroj pro tvorbu jednoduchých počítačů, který lze rozšiřovat o různé prvky (tj. senzory, pomocí kterých může Arduino komunikovat s vnějším světem a reagovat na něj), nahrávat do něj programy a prakticky tak zkoušet sestavovat vlastní elektronická zařízení na míru pro daný konkrétní projekt.

Základní vlastností této platformy je uživatelská přívětivost a jednoduchost při práci s ní. Pořizovací náklady základního dílu se pohybují v řádech stovek Kč (Arduino Uno na serveru https://www.hwkitchen.cz/arduino-originaly/ stojí 573 Kč), jednotlivé příslušenství jako jsou např. senzory, propojovací kabely či LED diody pořídíme rovněž za desítky až stovky korun, celé zařízení je tak dostupné i pro běžné uživatele. Software na tvorbu programů pro Arduino je zdarma. Velkou výhodou je rovněž obrovské množství návodů a tutoriálů, které jsou k Arduinu k dispozici díky velké celosvětové komunitě Arduino nadšenců (tipy ostatních uživatelů najdeme např. na https://forum.arduino.cc/, česká alternativa je https://www.arduino-forum.cz/). Je to proto vhodný nástroj pro získání či rozšíření základních znalostí elektrotechniky a programování.

7.1.1 Arduino – hardware

Arduino je jednoduchá počítačová deska, která pomocí různých snímačů a senzorů přijímá informace z okolí – vstupy a na základě nich ovládá výstupy (vstupem může být například stisknutí tlačítka nebo informace ze snímače intenzity osvětlení, výstupem třeba rozsvícení LED, spuštění motoru či zápis informace do souboru).

Samotnou základní desku Arduino tvoří mikrokontrolér, krystal, napájecí zdroj 5V a převodník pro komunikaci s počítačem. Pro pokročilejší funkčnost lze dokoupit tzv. Arduino Shield, tj. rozšiřující desku s požadovanými funkcemi pro konkrétní projekt. Pro propojení desky s počítačem slouží standardní USB kabel (A plug/B plug). [12]

Obrázek 20: Arduino UNO a kabel USB [24]

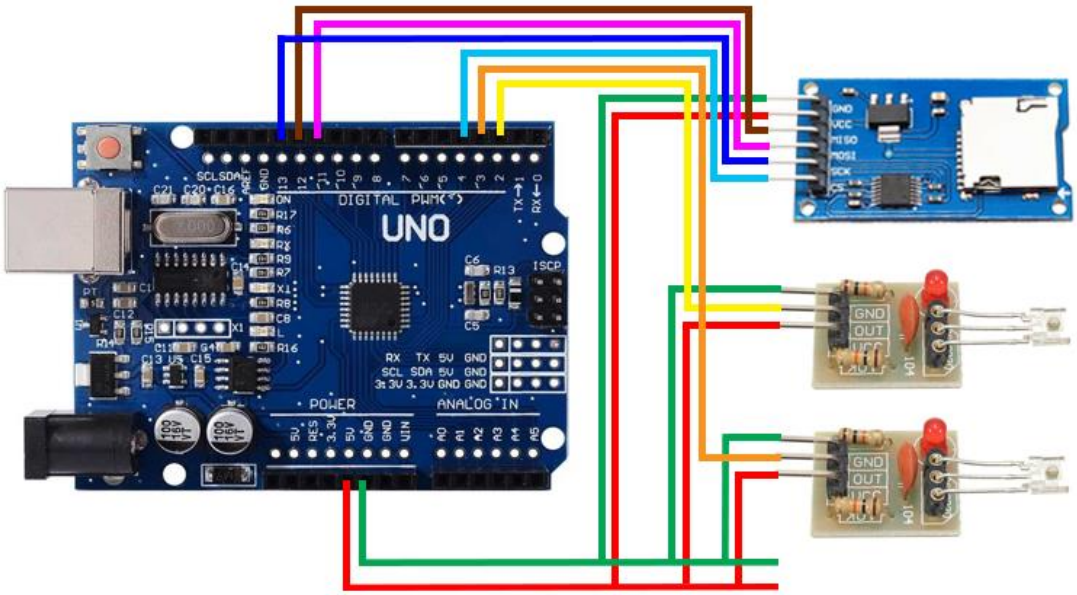
7.1.2 Arduino – software

Nedílnou součástí elektronické platformy Arduino je vývojové prostředí pro tvorbu software, pomocí kterého vytvoříme program pro Arduino mikrokontrolér, aby nám Arduino deska vykonávala to, co potřebujeme. Vývojové prostředí se jmenuje Arduino Software (IDE) a je založené na prostředí Processing, programovací jazyk se jmenuje Arduino a je založený na jazyce Wiring. Vývojové prostředí je možné stáhnout z oficiálních webových stránek Arduina – https://www.arduino.cc/en/Main/Software.

Obrázek 21: Ukázka vývojového prostředí Arduino Software se vzorovým programem Blink

7.2 Vytvoření měřicího zařízení

Samotné vyrobené měřicí zařízení se skládá z desky Arduino Uno, microSD Card modulu a 2 modulů laserového snímače. Zapojení jednotlivých prvků je vidět na obrázku č. 22.



Obrázek 22: Schéma zapojení měřicího zařízení

Pro celé měřicí zařízení byl pomocí 3D tiskárny vyroben plastový box pro jeho ochranu před vnějšími vlivy a pro lepší manipulaci s ním. Tento box má z jedné strany dutinu s laserovým snímačem, do které by měl při měření dopadat jeden z laserových paprsků; druhý snímač je vyveden na delším kabelu do oddělené trubice s dutinou, do které by měl dopadat druhý laserový paprsek. Celé měřicí zařízení je vidět na obrázku č. 23.

Obrázek 23: Vytvořené měřicí zařízení včetně dvou laserových vysílačů

7.2.1 Pseudokód

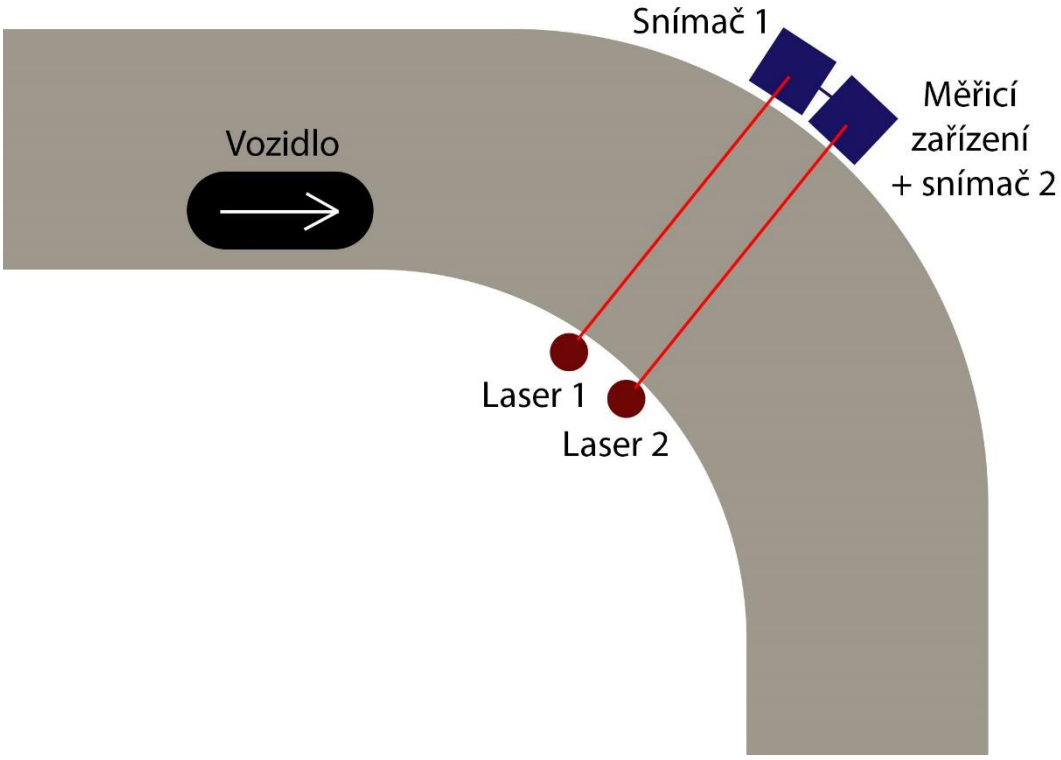
Program nahraný v Arduinu pro požadované měření jsem přepsala formou pseudokódu[2].

```
1   nahraj knihovnu pro práci s SD kartou
2   PinLaser1 = 2, PinLaser2 = 3 // Zapojení laserů - pin 2 a 3
3   chipSelect = 4 // Zapojení SD karty - pin 4
4   // Setup
5   začni komunikaci s SD kartou (9600 bitů za sekundu)
6   označ pin 2 a 3 jako vstup
7   nastav počáteční hodnotu TakeTime1, TakeTime2 = 1
8   // Začátek cyklu
9       time = počet milisekund od spuštění programu
10      LaserState1 = přečti hodnotu na vstupu PinLaser1 // 0 nebo 1
11      LaserState2 = přečti hodnotu na vstupu PinLaser2 // 0 nebo 1
12      If LaserState1 = 0 and TakeTime2 = 1 Then // při přerušení 1. laserového paprsku
13          If TakeTime1 = 1 Then
14              cas1 = time // ulož aktuální hodnotu time do proměnné cas1
15              TakeTime1 = 0
16          End If
17          If LaserState2 = 0 Then // při přerušení 2. laserového paprsku
18              If TakeTime2 = 1 Then
19                  cas2 = time // ulož aktuální hodnotu time do proměnné cas2
20                  TakeTime2 = 0
21              End If
22          End If
23      End If
24      If LaserState1 = 1 and LaserState2 = 1 and TakeTime2 = 0 Then
25          cas = cas2 - cas1
26          pokud ještě neexistuje, vytvoř soubor "mereni.txt" na SD kartě
27          zapiš do souboru hodnotu cas
28          TakeTime1, TakeTime2 = 1 //nastav proměnné zpět na výchozí hodnotu
29      End If
30      If LaserState1 = 1 and TakeTime1 = 0 Then
31          TakeTime1 = 1
32      End If
33      Delay(1) // mezera mezi cykly pro stabilitu programu
34  // Konec cyklu
```

[2] Pseudokód je neformální přepis kódu, který zjednodušeně popisuje algoritmus funkce. K jeho čtení není potřeba znalost daného programovacího jazyka a jeho syntaktických pravidel.

7.3 Metodika měření

Princip měření jsem zvolila následující: v měřeném místě jsem umístila k silniční komunikaci dva lasery v dané vzdálenosti od sebe (příklad uveden v tabulce č. 2 v kapitole 7.4.1 – Příklad výpočtu průměrné rychlosti), které vysílaly laserový paprsek skrz jízdní pruhy, přičemž na druhé straně silniční komunikace bylo samotné měřicí zařízení vyrobené z Arduina – to snímalo oba svítící laserové paprsky. Při průjezdu vozidla přes měřený úsek se přerušil nejprve první laserový paprsek, v tu chvíli si měřicí zařízení zapsalo čas, poté se jízdou vozidla přerušil i druhý laserový paprsek, a opět se zapsal čas. Rozdíl těchto dvou časových hodnot pak dal čas průjezdu jednotlivého vozidla. Schéma měření je vidět na obrázku č. 24.



Obrázek 24: Schéma měření rychlosti ve směrovém oblouku

Všechny získané hodnoty se automaticky zapisovaly do textového souboru na SD kartu. Z takto získaných hodnot času průjezdu ve dvou bodech a známé vzdálenosti těchto bodů jsem poté vypočítala průměrnou rychlost průjezdu vozidel daným úsekem.

Můj měřicí nástroj umožňoval rozeznávat vozidla jedoucí v druhém směru – ta byla z měření vyjmuta. Stejně tak jsem při následném vyhodnocování odstranila všechna extrémní data, která mohla nastat ve specifických situacích (např. když projížděla vozidla naráz v obou směrech) a která by tak mohla ovlivnit výsledky měření.

Pro samotné měření jsem vybrala oblouky na komunikacích ležících v extravilánu (kde je předpokládaná výchozí rychlost 90 km/h), a to takové, aby se jejich tvar co nejvíce blížil obloukům na segmentech komunikací v databázi KS_Software. Jedná se tedy o oblouky, které mají podobný poloměr jako oblouky v databázi a jsou přibližně pravoúhlé. Příklad jednoho z měřených oblouků je vidět na obrázku č. 25.

Obrázek 25: Oblouk s poloměrem R50 - příklad porovnání reálného měřeného oblouku (vlevo) a příslušného segmentu s obloukem (vpravo)

Tyto oblouky jsem vyhledala pomocí leteckých snímků, které jsou součástí digitálních map na serveru www.mapy.cz [17]. Snímek každého vybraného oblouku jsem poté převedla do programu Autodesk Civil 3D, kde jsem pomocí vložení oblouku mezi úsečky na daném místě měřila poloměry zvolených oblouků v mapě.

7.4 Průběh měření

Samotné měření probíhalo tak, že jsem u každého vybraného směrového oblouku umístila vyrobené měřicí zařízení způsobem, který je popsán na obrázku č. 24 výše. Příklad měření na jednom z oblouků je vidět na obrázku č. 26 a 27.

Obrázek 26: Měřený oblouk o poloměru R30 na silnici II/150

Obrázek 27: Příklad umístění měřicího zařízení - jeden z laserů upevněných u měřeného úseku komunikace

Takto nastavené měřicí zařízení snímalo všechna projíždějící vozidla a do textového souboru na SD kartě se zapisovaly časy jejich průjezdu. Zároveň byla změřena vzdálenost mezi snímači, a z těchto hodnot byla posléze vypočtena rychlost pro všechna projíždějící vozidla podle vztahu:

$$v = \frac{s}{t} \tag{12}$$

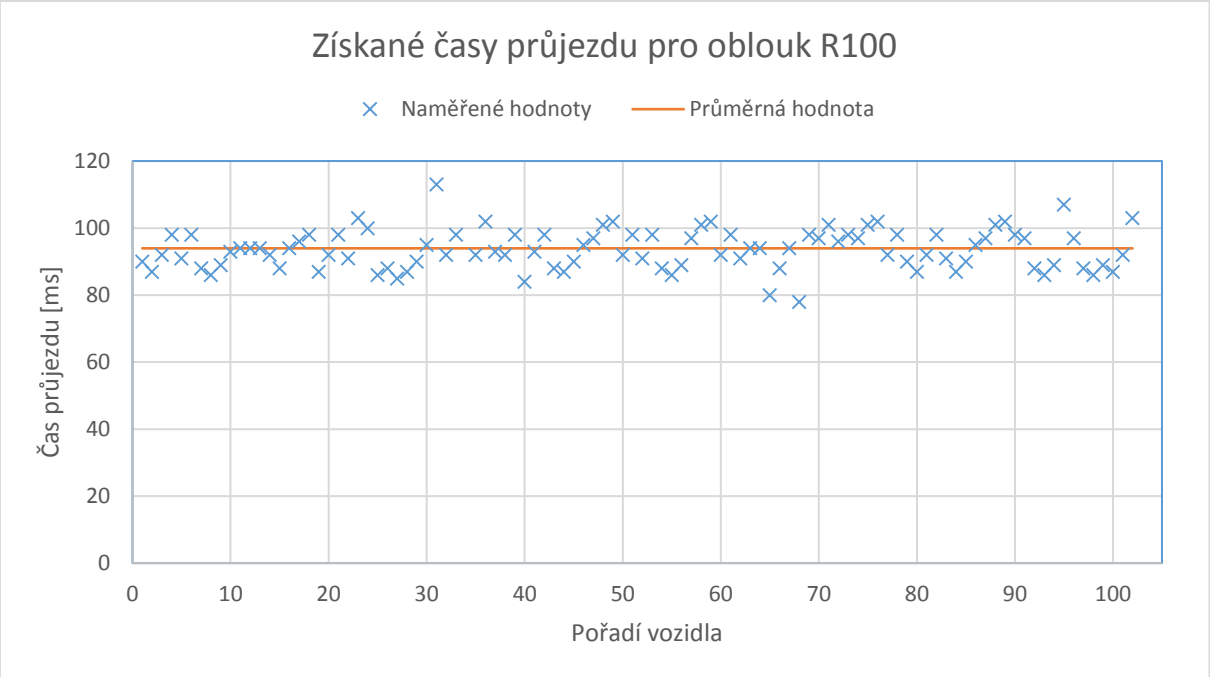
v … rychlost vozidla při průjezdu [m/s],

s … vzdálenost mezi snímači [m],

t … doba průjezdu změřená měřicím zařízením [s].

7.4.1 Příklad výpočtu průměrné rychlosti

Jako příklad výpočtu uvádím měřený oblouk s poloměrem R100. Při průjezdu vozidla úsekem ve směru, v němž je prováděno měření, se do souboru mereni.txt zapisuje čas průjezdu vozidla úsekem mezi snímači (v milisekundách), v případě průjezdu vozidla opačným směrem (při přerušení laserových paprsků v opačném pořadí) se do souboru zapisuje nula. Hodnoty získané z měření na oblouku R100 a zapsané na SD kartu jsou graficky znázorněny na obrázku č. 28.



Obrázek 28: Grafické znázornění časů získaných z měření na oblouku R100

Ze získaných časů průjezdu (po odstranění nulových a extrémních hodnot) jsem vypočítala aritmetický průměr a získala tak průměrnou dobu průjezdu, a pomocí vztahu (12) jsem pak dopočítala průměrnou rychlost pro daný oblouk, viz tabulka č. 2.

Tabulka 2: Výpočet průměrné rychlosti pro oblouk R100

Výpočet pro R100	
vzdálenost snímačů [m]	1,2
průměrná doba průjezdu [ms]	93,51961
průměrná doba průjezdu [s]	0,09352
průměrná rychlost [m/s]	12,83153
průměrná rychlost [km/h]	46,19352

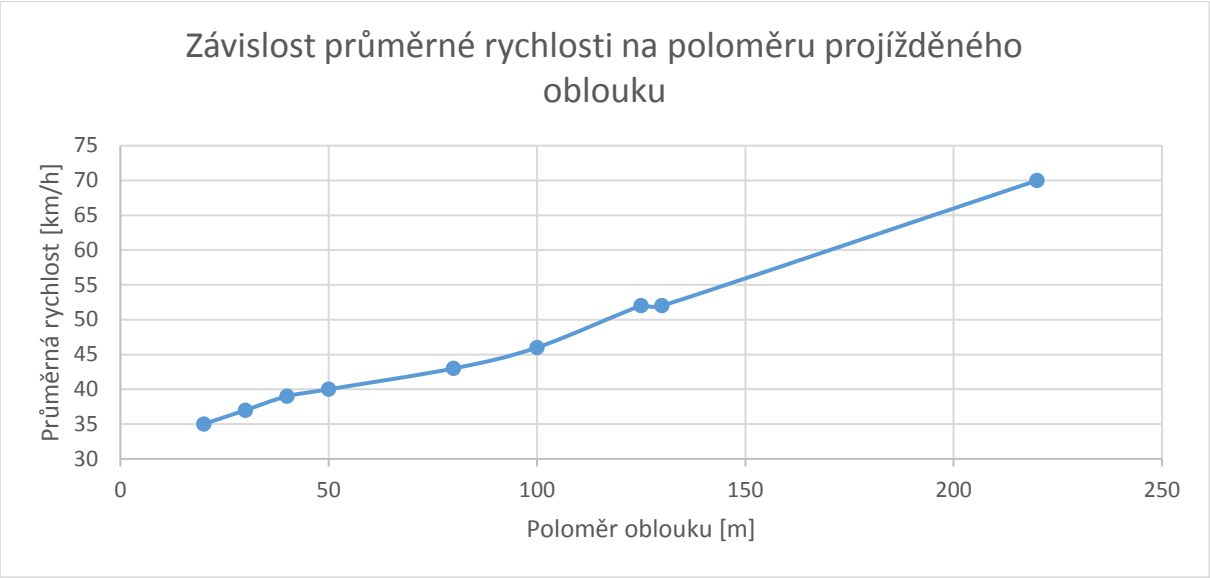
Na základě tohoto výpočtu byla průměrná rychlost pro směrový oblouk s poloměrem R100 stanovena na 46 km/h. Stejným způsobem byly určeny hodnoty rychlosti i pro ostatní měřené oblouky.

7.5 Získané výsledky měření

Z výsledků měření rychlosti na jednotlivých obloucích jsem určila průměrnou rychlost pro každý měřený poloměr směrového oblouku. Výsledné rychlosti jsou vidět v tabulce č. 3 a v grafickém zpracování na obrázku č. 29. Tyto rychlosti jsem poté použila v pluginu pro generování provozu KS_Traffic, který je popsán níže v kapitole 8.

Tabulka 3: Průměrné rychlosti při průjezdu měřenými oblouky

Poloměr	**Průměrná rychlost [km/h]**
R20	35
R30	37
R40	39
R50	40
R80	43
R100	46
R125	52
R130	52
R220	70



Obrázek 29: Grafické znázornění závislosti průměrné rychlosti na poloměru projížděného oblouku

8 Plugin pro generování provozu

Veškeré nové funkce, které jsem v rámci této diplomové práce vytvořila, jsou pluginy[3] pro 3D modelovací program. Plugin pro tuto práci byl vytvořen pro 3D modelovací program Rhinoceros ve skriptovacím jazyce RhinoScript.

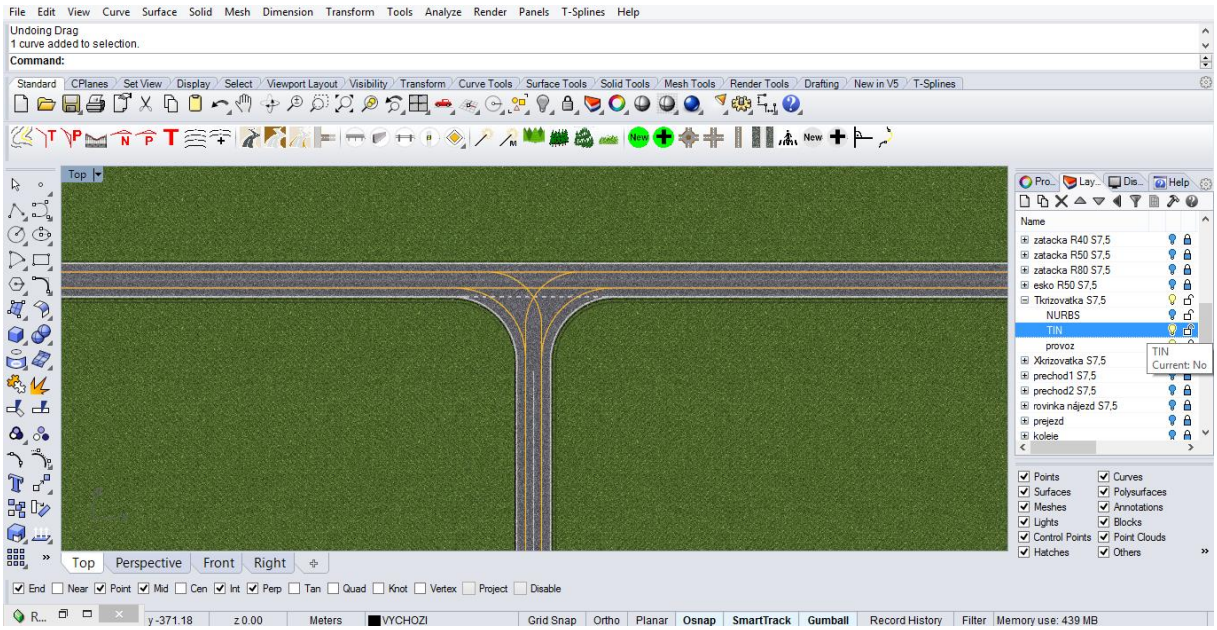
Plugin pro generování provozu, který jsem vytvořila v rámci této diplomové práce, navazuje na metodu tvorby virtuální scény KS_Software popsanou v mé bakalářské práci [3]. Tato metoda je blíže popsána v kapitole 4 – Plugin pro generování segmentů komunikací. Nově vytvořený plugin jsem nazvala KS_Traffic. Tento plugin se skládá ze čtyř dílčích funkcí, a to: příprava scény, výběr trasy, vykreslení rychlostní křivky a vytvoření složky s animačními soubory.

8.1 Funkce nového pluginu

Pomocí pluginu KS_Software je možné velice rychle vytvořit kompletní statickou virtuální scénu, kterou můžeme využít pro různé experimenty ve vozidlovém simulátoru. Pro zvýšení realističnosti experimentů je však vhodné, aby se statická scéna doplnila o animace, zejména o animaci jízdy ostatních vozidel, díky kterým se řidič simulátoru bude cítit více jako v reálném prostředí. Proto jsem se rozhodla možnosti této metody rozšířit o funkci automatického generování provozu do těchto scénářů, a to opět tak, aby bylo provoz možné vygenerovat v co nejkratším možném čase a bez nutných větších znalostí 3D modelování a exportování souborů do simulátoru.

Původní plugin jsem proto rozšířila o křivky provozu, které jsem doplnila pod každý segment v databázi modelů. Jedná se o křivky kopírující všechny jízdní pruhy na silnicích v daném směru. Tyto křivky definují trasu, po které mohou jezdit vozidla během animace. Všechny křivky na sebe navazují a po jejich spojení vznikne výsledná trasa jízdy animovaného vozidla.

[3] Plugin neboli zásuvný modul je přídavná funkce programu, která rozšiřuje jeho možnosti a urychluje práci v něm. Lze jej vytvořit např. pomocí skriptování.



Obrázek 30: Příklad připravených křivek provozu na dlaždici s křižovatkou ve tvaru T

Databázi jsem navíc rozšířila o nové segmenty – segment s železničním přejezdem a segment s železniční tratí (pro rozmanitější možnosti tvorby scénářů), dále pak segment s obloukem o poloměru R20 s velkým převýšením a segment s 12% stoupáním (pro otestování správného naklánění vozidla ve všech směrech). Přehled všech segmentů v databázi je součástí přílohy č. 1.

8.2 Příprava scény

Prvním krokem k vytvoření virtuálního scénáře je sestavení virtuálního prostředí ze segmentů pomocí pluginu KS_Software, postup je blíže popsán v kapitole 4. Takto získáme model prostředí, který se skládá ze segmentů – ty jsou uloženy v blocích ve vrstvě Scénář. Pro umožnění další práce s modely je tedy nejprve nutné bloky rozložit na jednotlivé objekty, a ty rozmístit do podvrstev, podle toho, jak je budeme později potřebovat. K tomuto účelu slouží nová funkce Rozklad bloku.

Tato funkce nejprve vybere všechny bloky ve vrstvě Scénář, a přidá k ní nové podvrstvy Export_vizualizace, Silnice a Provoz. Poté rozloží bloky na jednotlivé objekty, a vybere z nich všechny křivky provozu – ty přesune do vrstvy Provoz, dále všechny plochy silnic – ty přesune do vrstvy Silnice, a všechny ostatní objekty přesune do vrstvy Export_vizualizace. Poté ještě pro pozdější rychlejší export modelu do grafického enginu tato funkce sloučí všechny stromy a keře obsažené ve scéně, které mají stejný materiál.

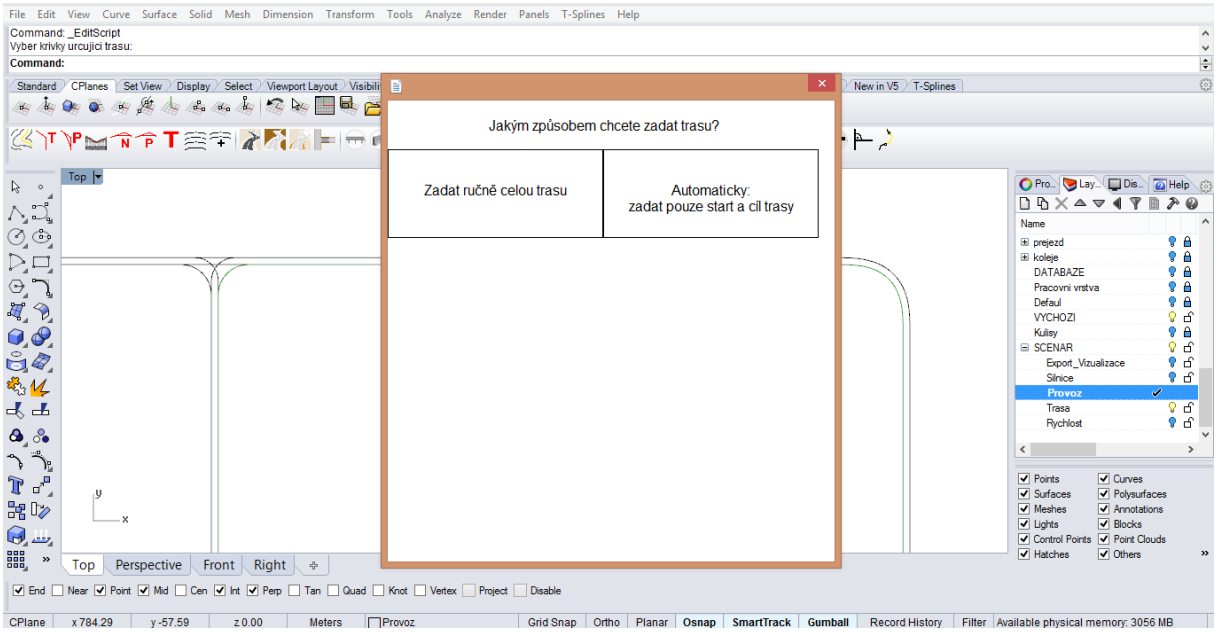
8.2.1 Pseudokód

```
1   scenar = vyber všechny bloky ve vrstvě SCÉNÁŘ
2   přidej podvrstvy Export_vizualizace, Silnice, Provoz
3   For each blok In scenar
4       blokobjects = rozlož blok na objekty
5       // nejprve přesuň stromy pro urychlení běhu programu
6       přesuň blokobjects do pracovní vrstvy
7       stromy = vyber objekty s názvem Strom
8       If Not Null stromy Then
9           Přesuň stromy do vrstvy Export_vizualizace
10      End If
11      blokobjects = vyber všechny objekty v pracovní vrstvě // vše kromě stromů
12      For Each object In blokobjects
13          If jméno object = provoz_krivka Then
14              přesuň object do vrstvy Provoz
15          Else If jméno object = silnice Then
16              přesuň object do vrstvy Silnice
17          Else
18              přesuň object do vrstvy Export_vizualizace
19          End If
20      Next
21  Next
22  sluč všechny stromy a keře se stejným materiálem
```

8.3 Výběr trasy

Jakmile je výše uvedeným způsobem virtuální prostředí připravené – vymodelované a rozřazené do daných vrstev, může uživatel vybrat trasu, po které budou během animace jezdit vozidla. Trasu uživatel vybírá spojováním jednotlivých křivek provozu, které se nacházejí pod každou dlaždicí a kopírují všechny jízdní pruhy. Pro tento účel slouží další nová funkce s názvem Výběr trasy.

Po spuštění této funkce má uživatel na výběr, jakým způsobem bude chtít trasu vybrat – buď ručně, tj. postupným vybíráním jednotlivých částí trasy, nebo automaticky, kdy si zvolí pouze požadovaný počátek a konec trasy, a ta se poté propojí automaticky. Tento výběr uživatel provede pomocí HTML rozhraní.



Obrázek 31: HTML rozhraní pro výběr způsobu zadání trasy

8.3.1 Výběr trasy ručně

Při zvolení první možnosti, tedy zadání trasy ručně, je uživatel vyzván, aby postupně vybral všechny křivky určující danou trasu. Program poté vybrané křivky sloučí a zkopíruje je do nové podvrstvy vrstvy Scénář s názvem Trasa.

Tento způsob výběru je vhodný např. v případech, kdy uživatel potřebuje vybrat konkrétní trasu, která projíždí danými oblastmi, různě se větví a nebylo by tedy možné ji vybrat automaticky. Dává tak uživateli možnost vybrat si svoji trasu nezávisle na algoritmu zvoleném pro automatický výběr trasy.

8.3.2 Výběr trasy automaticky

V případě, kdy uživatel potřebuje určit pouze místa, odkud a kam má animované vozidlo jet, je vhodné zvolit druhou možnost, tedy automatický výběr trasy. V tomto případě je uživatel vyzván, aby vybral počáteční a koncovou křivku trasy, a program automaticky mezi těmito křivkami vytvoří trasu. Jelikož mezi segmenty komunikací jsou i segmenty s křižovatkou, bylo nutné vyřešit rozhodování algoritmu v případě větvení trasy.

Theseova metoda

Algoritmus, který jsem použila pro automatické vytvoření trasy, je inspirovaný Theseovou metodou, popsanou v Teorii grafů [5]. Tato metoda vychází z legendy Labyrint z řecké mytologie, ve které je v labyrintu chodeb a komnat uvězněný Minotaurus (napůl člověk, napůl býk), kterému se každých 9 let musí obětovat 7 mladých aténských mužů a dívek. Theseus se proto rozhodne s tímto skoncovat tak, že Minotaura zabije – musí se proto dostat přes labyrint k Minotaurovi, tam ho zabít a poté se dostat zase ven, aby mohl ostatním

Atéňanům říct, že jsou zachráněni. Od Ariadné dostane kouzelný meč k zabití Minotaura, klubíčko nití z ovčí vlny pro nalezení cesty zpět, a nádobu s barvou k označení chodeb, které již prošel v obou směrech.

Labyrint představuje v tomto případě neorientovaný graf, jehož vrcholy jsou znázorněny jako komnaty a hrany jako chodby labyrintu.

Theseus tedy vyrazí z výchozí komnaty jednou z chodeb a rozmotává při tom klubko niti. Při příchodu do další komnaty mohou nastat tyto možnosti:

a) vstoupí do další komnaty, která je slepá – vrátí se do předchozí komnaty a označí prošlou chodbu barvou,
b) vstoupí do další komnaty, ze které vedou další chodby – pokud existuje alespoň jedna, která ještě není označená, vydá se jí a odmotává při tom nit; pokud jsou již všechny další chodby označeny, vrátí se chodbou, kterou přišel a označí ji barvou, zároveň při tom namotává nit zpět,
c) vstoupí do cílové komnaty, kde je Minotaurus – zabije ho a po cestě určené odmotanou nití se vrátí do výchozí komnaty k Ariadné.

Použitá metoda pro vytvoření trasy

Na podobném principu jako Theseova metoda funguje i můj algoritmus pro automatické vytvoření trasy, kdy je nutné najít cestu mezi počáteční a koncovou křivkou.

Algoritmus nejprve testuje pro zvolenou počáteční křivku, zda má nějaké navazující křivky. Pokud navazuje jedna křivka, uloží ji do databáze, a provede stejné testování pro tuto další křivku. Pokud je navazujících křivek více, zvolí program jednu z nich, po které se vydá – jestliže touto cestou dojde k nesprávnému konci (na poslední křivku již nenavazuje žádná křivka a zároveň se nejedná o zvolenou koncovou křivku), vrátí se program k poslednímu větvení a tentokrát zvolí jinou křivku, a pokračuje v testování. Tímto způsobem zkouší všechny možné cesty, dokud nedojde k uživatelem zvolené koncové křivce.

Pro snazší orientaci jsem zavedla pole IDcesty, které obsahuje informace o tom, kterou cestu program využil při posledním větvení, aby se při dosažení nesprávného konce trasy mohl vrátit k předchozímu větvení, a tam použít jinou křivku než minule. Funkce pole IDcesty je nastíněna v tabulce č. 4 (křivky jsou číslovány od nuly).

Tabulka 4: Funkce pole IDcesty

Informace v poli IDcesty	**Význam**
IDcesty(0)	Trasa obsahuje jedno větvení, ve kterém byla použita první křivka.
IDcesty(0,0)	Trasa obsahuje dvě větvení, v prvním i v druhém byla použita první křivka.
IDcesty(0,1)	První křivka ve druhém větvení nevedla ke správné koncové křivce. Trasa obsahuje stále dvě větvení, v prvním byla použita první křivka, v druhém byla použita druhá křivka.
IDcesty(0,2)	Druhá křivka ve druhém větvení nevedla ke správné koncové křivce. Trasa obsahuje stále dvě větvení, v prvním byla použita první křivka, v druhém byla použita třetí křivka.
IDcesty(1)	Po druhém větvení následoval při využití všech křivek nesprávný konec, trasa tedy nyní obsahuje opět jedno větvení, ve kterém byla použita druhá křivka.

Podrobný algoritmus nalezení trasy je vysvětlen níže v pseudokódu. Po nalezení výsledné trasy jsou opět křivky sloučeny a zkopírovány do nové podvrstvy vrstvy Scénář s názvem Trasa.

8.3.3 Pseudokód

```
1   zpusob = uživatel vybere způsob výběru trasy z nabídky přes HTML rozhraní
2   If zpusob = 0 Then // klikání trasy
3     krivky = uživatel vybere křivky určující trasu
4     provoz = zkopíruj objekty krivky
5     vytvoř novou vrstvu Trasa
6     přesuň provoz do vrstvy Trasa
7     sluč všechny objekty ve vrstvě Trasa
8   Else // trasa automaticky
9     krivky = vyber všechny objekty ve vrstvě Provoz
10    startcurve = uživatel vybere počáteční křivku trasy
11    endcurve = uživatel vybere koncovou křivku trasy
12    nextcurve = první nextcurve je startcurve
13    vetveni = první objekt v poli vetveni je startcurve // vetveni = pole prvních křivek uložených
      po každém větvení
14    IDcesty = první hodnota v poli IDcesty je 0
```

```
15  Do While Not nextcurve = endcurve //opakuj pro každou křivku nextcurve dokud nedojdeš ke křivce endcurve
16      endpoint = koncový bod nextcurve
17      q = 0
18      i = -1
19      // testuj, kolik má křivka nextcurve navazujících křivek
20      Do While i < počet křivek // opakuj pro každou křivku v poli krivky
21          i = i + 1
22          // pokud na křivku nenavazuje žádná křivka - špatný konec trasy, jinak ulož křivku do dalsi
23          If i = počet křivek Then
24              If q = 0 Then // špatný konec trasy - vrať se k poslednímu větvení a vydej se jinou cestou
25                  zvyš poslední index pole IDcesty o 1
26                  vymaž poslední objekt z pole vetveni
27                  nextcurve = aktuální poslední objekt v poli vetveni
28              End If
29          Else // otestuj, zda křivka i navazuje na nextcurve
30              startpoint = počáteční bod křivky i
31              If vzdálenost mezi endpoint, startpoint < 1 Then // byla nalezena navazující křivka
32                  přidej křivku i do pole dalsi // dalsi = pole všech navazujících křivek pro nextcurve
33                  q = q + 1
34              End If
35          End If
36      Loop
37      If Not q = 0 Then // pokud existují navazující křivky
38          If q = 1 Then // pokud existuje jedna navazující křivka
39              nextcurve = křivka uložená v poli dalsi
40              přidej nextcurve do pole provozkrivka // provozkrivka = pole všech uložených křivek trasy, které nenásledují po větvení
41              vymaž objekty v poli dalsi
42          Else // pokud existuje více navazujících křivek
43              If poslední index pole IDcesty = počet větvení Then
44                  // setkáváme se s tímto větvením poprvé - použijeme první navazující křivku
45              Else // již jsme se s tímto větvením setkali - použijeme jinou navazující křivku než minule
46                  přidej další index do pole IDcesty a přiřaď mu 0
47              End If
48              If hodnota posledního indexu IDcesty ≤ q - 1 Then // pokud je více navazujících křivek, než je hodnota posledního indexu pole IDcesty, přidej další křivku
49                  nextcurve = křivka uložená v poli dalsi v pořadí odpovídajícím hodnotě posledního indexu pole IDcesty
50                  přidej nextcurve do pole vetveni
51                  vymaž objekty v poli dalsi
52              Else // pokud již není více navazujících křivek, než je hodnota posledního indexu pole IDcesty (byly otestovány všechny navazující křivky), vrať se o jedno větvení zpět
```

```
53                  uber jeden index z pole IDcesty
54                  zvyš aktuální poslední index pole IDcesty o 1
55                  vymaž poslední objekt z pole vetveni
56                  nextcurve = aktuální poslední objekt v poli vetveni
57              End If
58          End If
59      End If
60    Loop
61    odstraň duplicitní křivky v poli provozkrivka
62    vytvoř novou vrstvu Trasa
63    sluč všechny objekty v poli provozkrivka a v poli vetveni a zkopíruj je do vrstvy Trasa
64    vymaž přebytečné křivky
65  End If
```

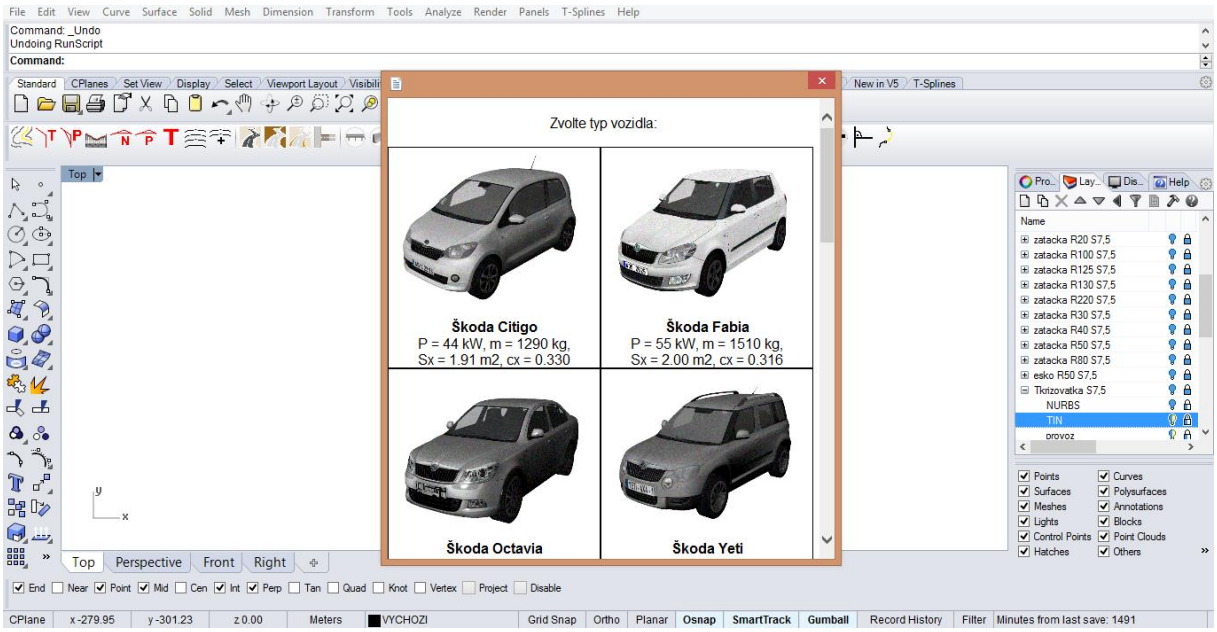
8.4 Křivka rychlosti

Po nalezení trasy provozu může uživatel přistoupit k vykreslení grafu průběhu rychlosti, ze kterého se budou později získávat informace o rychlosti jízdy vozidla v každém bodě trasy. K tomu slouží další část nového pluginu s názvem Křivka rychlosti.

Pro možnost zvolení konkrétního animovaného vozidla jsem vytvořila databázi vozidel, které jsem umístila do systémové složky programu Rhinoceros – tato složka obsahuje pro každý typ vozidla soubor s jeho modelem ve formátu pbtz a textury pro jednotlivé barevné variace, připravené pro export do simulátoru. Tyto soubory budou poté na základě uživatelova výběru kopírovány do výstupní složky.

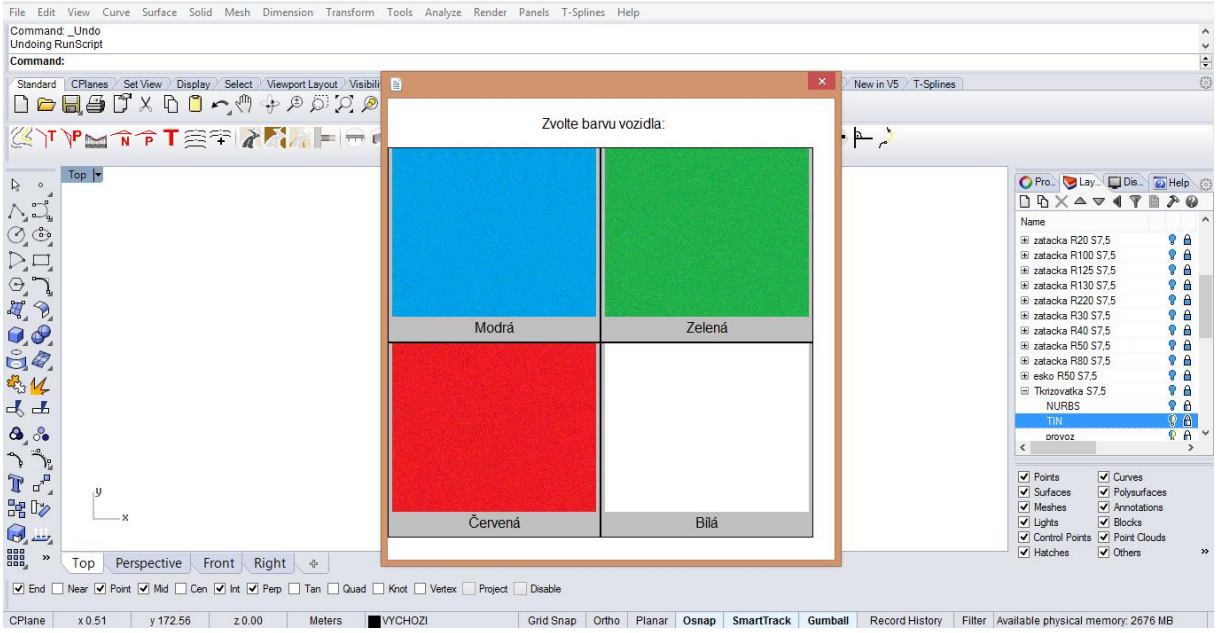
8.4.1 Zadání vstupních hodnot

Před samotným vykreslením grafu je uživatel vyzván, aby si zvolil typ vozidla, pro které bude chtít vytvořit animaci – učiní tak opět přes HTML rozhraní, ve kterém má na výběr ze 7 osobních automobilů, 2 dodávek, 2 kamionů a autobusu; případně si může zvolit vlastní vozidlo a zadat pro něj parametry ručně. Pro každé vozidlo jsem našla či dopočítala jeho technické parametry, které budou nutné pro další výpočty – jedná se o výkon motoru P [W], celkovou hmotnost vozidla m [kg], čelní plochu vozidla S_x [m^2] a součinitel vzdušného odporu c_x [-]. Tabulka získaných údajů pro jednotlivá vozidla je součástí přílohy č. 2 – Technické parametry vozidel.



Obrázek 32: HTML rozhraní pro výběr typu vozidla

Po zvolení typu vozidla si může uživatel ještě zvolit jeho barvu – nabídka barev se zobrazí opět prostřednictvím HTML rozhraní, a to vždy na základě zvoleného vozidla podle toho, jaké modely jsou v databázi k dispozici.



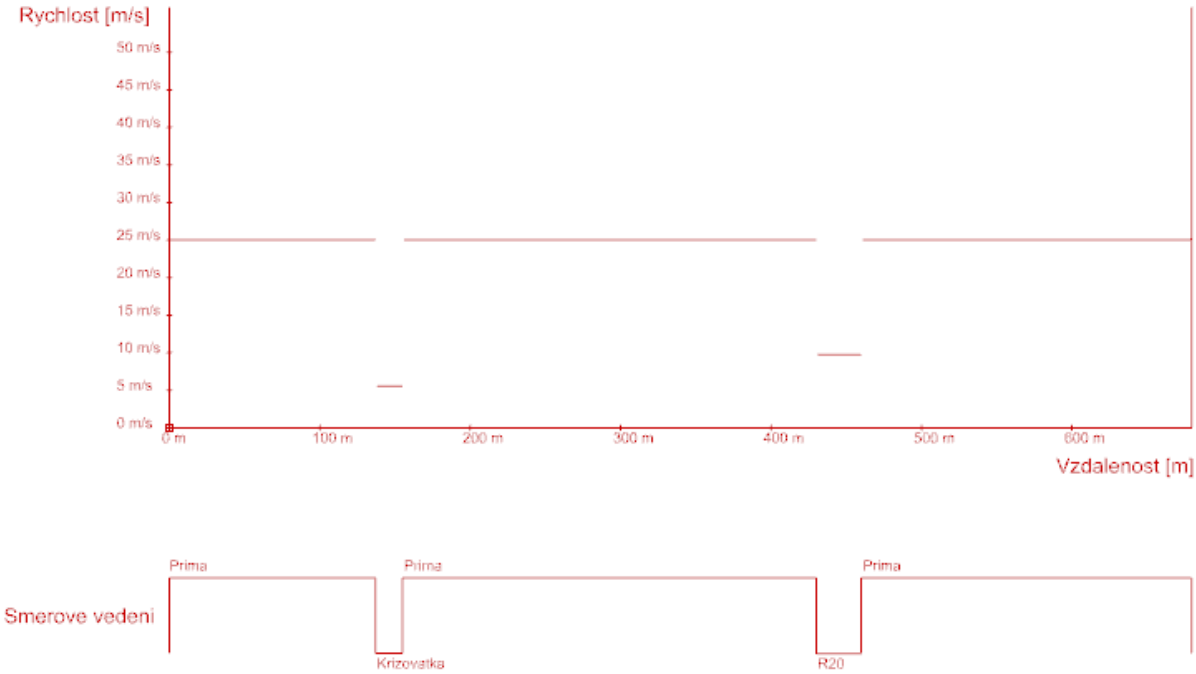
Obrázek 33: HTML rozhraní pro výběr barvy vozidla (nabídka pro vůz Škoda Citigo)

Dále uživatel zadá hodnoty komfortního zrychlení a zpomalení, které chce pro danou animaci použít – výchozí hodnoty jsou nastaveny na 1 m/s^2 pro zrychlení a 1.5 m/s^2 pro zpomalení.

8.4.2 Vykreslení grafu průběhu rychlosti

Graf průběhu rychlosti má na x-ové ose vzdálenost získanou z údajů o délce vybrané trasy provozu, a na y-ové ose je rychlost, jakou vozidlo po trase pojede. Pod grafem se vykresluje také směrové vedení trasy, které obsahuje informace o tom, ve kterých úsecích vozidlo projíždí oblouky (včetně informace o poloměru oblouku), a ve kterých jede přímo. Od toho se také odvíjí rychlost, s jakou vozidlo daný úsek projíždí – pro přímé úseky je nastavena rychlost na 90 km/h, pro křižovatky na 20 km/h, pro oblouky je rychlost zvolená na základě výsledků měření popsaného v kapitole 7 – Měření rychlosti při průjezdu směrovými oblouky.

Rychlostní průběh je vykreslován tak, že nejprve je pro každý úsek vykreslena úsečka odpovídající hodnotě rychlosti na daném úseku. V této fázi se graf ještě uživateli nevykresluje, na obrázku č. 34 je ukázán pouze pro lepší představu funkčnosti metody.



Obrázek 34: Rychlostní průběh – úsečky udávající rychlost v daných úsecích

Poté jsou jednotlivé úseky propojeny parabolou znázorňující zrychlení a zpomalení vozidla. Tato parabola je vykreslována zvlášť pomocí nové funkce.

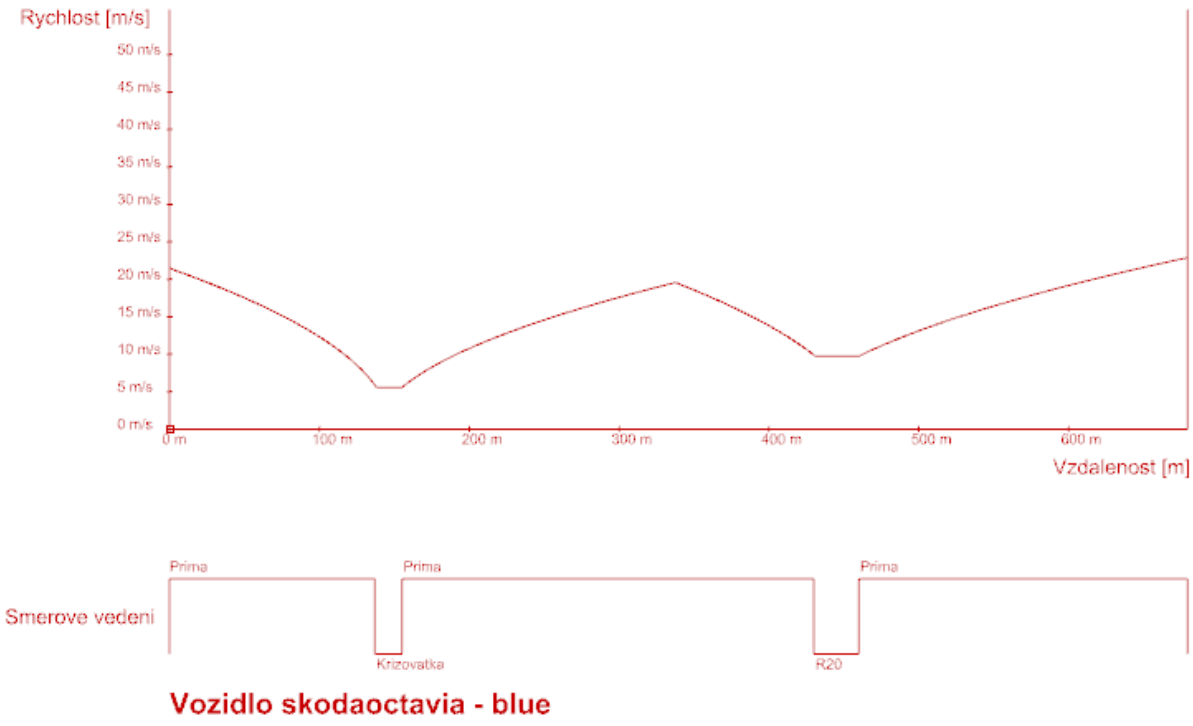
Pro zpomalení vozidla – deceleraci se křivka vykresluje pouze na základě známé počáteční a koncové rychlosti, na kterou je třeba zpomalit, a to s daným komfortním zpomalením.

Pro zrychlení vozidla – akceleraci se při vykreslování křivky kromě komfortního zrychlení a počáteční a koncové rychlosti berou v potaz i technické parametry vozidla, které si uživatel pro animaci zvolil. Hlavním důvodem je to, že u většiny vozidel výkon motoru stačí

pro udržování komfortního zrychlení pouze do určité rychlosti, poté začne vozidlo zrychlovat pomaleji – na tomto principu je křivka zrychlení vykreslována také v rámci této funkce. Dokud je výkon motoru dostatečný, vozidlo zrychluje s komfortním zrychlením, poté je křivka nahrazena rychlostní křivkou s maximálním zrychlením, jaké je s ohledem na výkon motoru v daném okamžiku dosažitelné. Výpočet těchto hodnot je proveden s využitím vzorců popsaných v kapitole 6.2.1 – Podélná dynamika, a je podrobněji popsán níže v pseudokódu.

Po vložení propojovacích křivek jsou původní úsečky oříznuty, odstraněny všechny přebytečné části křivek a vzniklá křivka je sloučena do výsledné rychlostní křivky. Graf je pro lepší orientaci pojmenován podle vybraného typu a barvy vozidla.

Celý graf je poté uložen v nové podvrstvě vrstvy Scénář s názvem Rychlost.



Obrázek 35: Výsledná rychlostní křivka

Funkce pro vykreslení rychlostní křivky není nijak závislá na předchozích funkcích, lze ji proto využít i v obecných scénářích, které nejsou připraveny pomocí blokové metody KS_Software – např. v případě, že je trasa provozu dokreslena ručně.

8.4.3 Pseudokód

```
1   // vykreslení křivky zrychlení a zpomalení - pomocí funkcí
2   // funkce pro deceleraci - Krivkarychlosti (nezávislá na výkonu motoru)
3       (vstupní hodnoty: a0 - zrychlení [m/s2], v0 - počáteční rychlost [m/s], v1 - požadovaná
        rychlost [m/s])
4       time = (v1-v0) / a0 // celkový čas nutný ke změně rychlosti
5       i = 0
6       distance = 0
7       Do While i < 21 // rozdělení křivky zrychlení body na 20 částí
8           speed = v0 + a0 * time * (i/20)
9           dis = speed * time * (1/20) // přírůstek vzdálenosti pro 1/20 křivky
10          distance = distance + dis // aktuální celková vzdálenost
11          přidej bod o souřadnicích (distance - v0 * time * (1/20), speed - v0, 0)
12      Loop
13      Krivkarychlosti = polož vzniklé body křivkou
14  // funkce pro akceleraci - Krivkarychlosti1 (závislá na výkonu motoru)
15      (vstupní hodnoty: a0 - zrychlení [m/s2], v0 - počáteční rychlost [m/s], v1 - požadovaná
        rychlost [m/s], P - výkon vozidla [W], m - hmotnost vozidla [kg], Sx - čelní plocha vozidla
        [m2], cx - součinitel vzdušného odporu [-])
16      ro = 1.225 // hodnota hustoty vzduchu [kg/m2]
17      f = 0.01 // hodnota součinitele valivého odporu pro asfalt
18      eta = 0.9 // hodnota účinnosti motoru
19      time = (v1-v0) / a0 // celkový čas nutný ke změně rychlosti
20      i = 0
21      distance = 0
22      v = v0
23      Do While i < 21 // rozdělení křivky zrychlení body na 20 částí
24          Of = v * m * 9.81 * f // odpor valivý
25          Ov = 0.5 * Sx * cx * ro * v3 // odpor vzduchu
26          Oz = P * eta - Of - Ov // odpor zrychlení
27          a = Oz / (v * m) // max. zrychlení
28          If a > a0 Then // výkon motoru je dostatečný pro toto zrychlení
29              speed = v0 + a0 * time * (i/20)
30          Else // výkon motoru není dostatečný - je dosáhnuto maximálního zrychlení
31              time = (v1-v0) / a
32              speed = v0 + a * time * (i/20)
33          End If
34          dis = speed * time * (1/20) // přírůstek vzdálenosti pro 1/20 křivky
35          distance = distance + dis // aktuální celková vzdálenost
36          přidej bod o souřadnicích (distance - v0 * time * (1/20), speed - v0, 0)
37          v = speed
38          i = i + 1
39      Loop
40      Krivkarychlosti1 = polož vzniklé body křivkou
```

```
41  // vykreslení grafu rychlostní křivky
42  vozidlo, barva = uživatel vybere typ a barvu vozidla z nabídky přes HTML rozhraní
43  vytvoř novou vrstvu Rychlost
44  krivka = uživatel vybere křivku trasy provozu
45  points = rozděl křivku krivka body po 1 metru
46  pocatek = uživatel vybere bod, kam chce vykreslit graf rychlosti
47  akcelerace, decelerace = uživatel zadá hodnotu komfortního zrychlení a zpomalení v m/s2
48  vykresli osy grafu: osa x = vzdálenost [m] (délka podle délky trasy), osa y = rychlost [m/s] (0-
    50)
49  přidej popis os grafu: název osy, hodnoty (staničení, rychlost)
50  For Each point In points
51      radius = změř poloměr křivky krivka v bodě point
52      podle hodnoty radius vykresli pod graf směrové vedení trasy
53  Next
54  k jednotlivým úsekům přidej popis úseku (přímá, oblouk o daném poloměru)
55  rychlosti = nad každý úsek (přímá, oblouk) vykresli v grafu rychlosti úsečku ve výšce
    odpovídající hodnotě rychlosti pro daný úsek
56  For Each rychlost In rychlosti
57      If výška úsečky rychlosti(i) > výška úsečky rychlosti(i+1) Then // zpomalení
58          vykresli spojovací křivku pomocí funkce Krivkarychlosti a přesuň ji na konec úsečky
            rychlosti(i)
59      Else // zrychlení
60          vykresli spojovací křivku pomocí funkce Krivkarychlosti1 (s parametry podle zvoleného
            vozidla) a přesuň ji na konec úsečky rychlosti(i)
61      End If
62  Next
63  ořež protínající se křivky a smaž všechny přebytečné části
64  sluč zbylé křivky ve výslednou křivku rychlosti
65  pojmenuj výslednou křivku podle zvoleného typu a barvy vozidla
```

8.5 Animační soubory

Nejdůležitější částí celého pluginu je jeho poslední část, ve které je vygenerována složka s animačním souborem a dalšími soubory nutnými k úspěšnému exportu do simulátoru. S využitím souborů v této složce již lze vytvořenou animaci v simulátoru spustit.

Animační soubor, který je hlavním výstupem této práce, obsahuje všechny potřebné údaje definující animaci jízdy vozidla. Každý řádek tohoto souboru popisuje, v jakém bodě se vozidlo v daném čase nachází a s jakým natočením; s využitím těchto údajů pak vzniká samotná animace pohybu vozidla. Všechny potřebné údaje jsou generovány ve sloupcích.

Aby byl výsledný soubor přehlednější, vytvořila jsem zvláštní funkci s názvem Decimal, která zajišťuje, že všechna čísla obsažená v animačním souboru budou mít stejnou délku (stejný počet desetinných míst) – tzn. pokud má číslo více desetinných míst, je zaokrouhleno,

pokud má naopak míst méně, je doplněno do potřebné délky případným přidáním desetinné čárky a příslušného počtu nul.

8.5.1 Vytvoření animační složky

Při spuštění této funkce si uživatel nejprve zvolí adresář, ve kterém chce vytvořit složku animation – do ní jsou poté ukládány všechny potřebné soubory. Do jedné animační složky je možné v rámci jednoho scénáře uložit více animačních souborů pro různá vozidla, ty se ukládají postupně pod názvem auto1.anim, auto2.anim, apod.

Dále si uživatel zvolí počáteční čas animace – jedno vozidlo může vyjet v čase 0, další např. o 10 sekund později. Uživatel si zvolí také to, zda se má animace vozidla po jejím skončení opakovat, či má proběhnout pouze jednou.

Po vybrání příslušné trasy a rychlostní křivky provozu je křivka trasy rozdělena body – v případě přímé trasy jsou body přidávány po 5 metrech, v obloucích jsou pro zajištění plynulé animace přidávány častěji. Pro každý tento bod jsou poté získávány všechny potřebné údaje, které jsou zapisovány do animačního souboru.

8.5.2 Údaje v animačním souboru

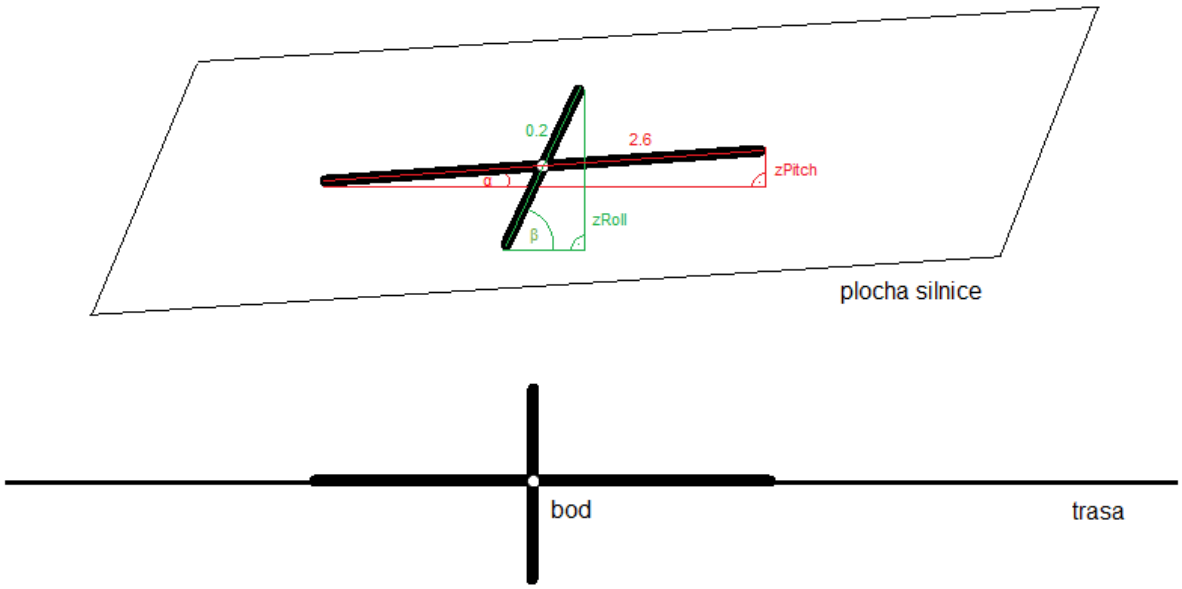
Každý řádek animačního souboru obsahuje všechny potřebné údaje pro jeden bod, který je součástí trasy provozu. Typický řádek animačního souboru vypadá například takto:

48634 24.9681 0.2357 -1877.1776 -2.2923 205.8525 0.0167

Prvním zapisovaným údajem je čas (48 634 ms) – ten se získává z počátečního času, který zadal uživatel, a poté se připočítává přírůstek času pro každý bod vydělením rychlosti v tomto bodě (získané z rychlostní křivky) a vzdálenosti od předchozího bodu.

Další tři zapisované údaje (24.9681, 0.2357, -1877.1776) tvoří souřadnice bodu, v němž se v daném čase vozidlo nachází (x, y, z) – získávají se promítnutím bodu na křivce trasy ve svislém směru na plochu silnice, pod kterou trasa prochází (zajistí se tak správná výška bodu odpovídající výšce povrchu vozovky).

Dalšími údaji pak jsou úhly natočení vozidla v daném bodě. Jedná se o úhel Yaw, tj. natočení vozidla v daném směru jízdy, které se získává ze směrového úhlu křivky trasy (-2.2923°); dále úhel Pitch, tj. náklon vozidla při jízdě do stoupání (205.8525°), a úhel Roll, tj. příčný náklon vozidla (0.0167°) – ty jsou získávány s využitím pomocného kříže, který je umístěn na každý bod ve směru křivky trasy a promítnut ve svislém směru na plochu silnice. Výsledný úhel je vypočítaný pomocí funkce sinus, viz obrázek č. 36.



Obrázek 36: Princip výpočtu úhlu Pitch a Roll

Všechny souřadnice a úhly zapsané v animačním souboru jsou upraveny pomocí výše popsané funkce Decimal, aby byl výsledný soubor čitelnější. Výsledný soubor pak tvoří přehledná matice všech výše zmíněných hodnot pro všechny body trasy.

8.5.3 Další soubory v animační složce

Kromě animačního souboru je v animační složce vytvořen také soubor animation.ini, který obsahuje inicializační údaje pro všechny animace ve virtuálním scénáři. Jedná se o údaje o animaci mraků a údaje o animaci jednotlivých vozidel – např. rozměry vozidla, pořadí animace a další.

Mimo to je do animační složky zkopírován ze systémové složky programu Rhinoceros také soubor mraky.anim (animační soubor definující pohyb mraků na obloze, používá se ve většině scénářů na Fakultě dopravní), a také uživatelem zvolený model vozidla (který je přejmenován podle názvu animačního souboru – např. auto1.pbtz) a textura vozidla ve zvolené barvě. Tyto soubory poté stačí přesunout do složky pro export do simulátoru, a animace je připravena pro spuštění.

8.5.4 Pseudokód

```
1   // funkce Decimal (pro jednotnou délku zapisovaných čísel - větší přehlednost výstupního souboru)
2       (vstupní hodnoty: číslo, počet desetinných míst)
3       roundcislo = zaokrouhli číslo na daný počet desetinných míst
4       textcislo = převeď číslo roundcislo na text
5       rozděl textcislo v místě desetinné čárky
6       If počet částí = 2 Then // jedná se o desetinné číslo
```

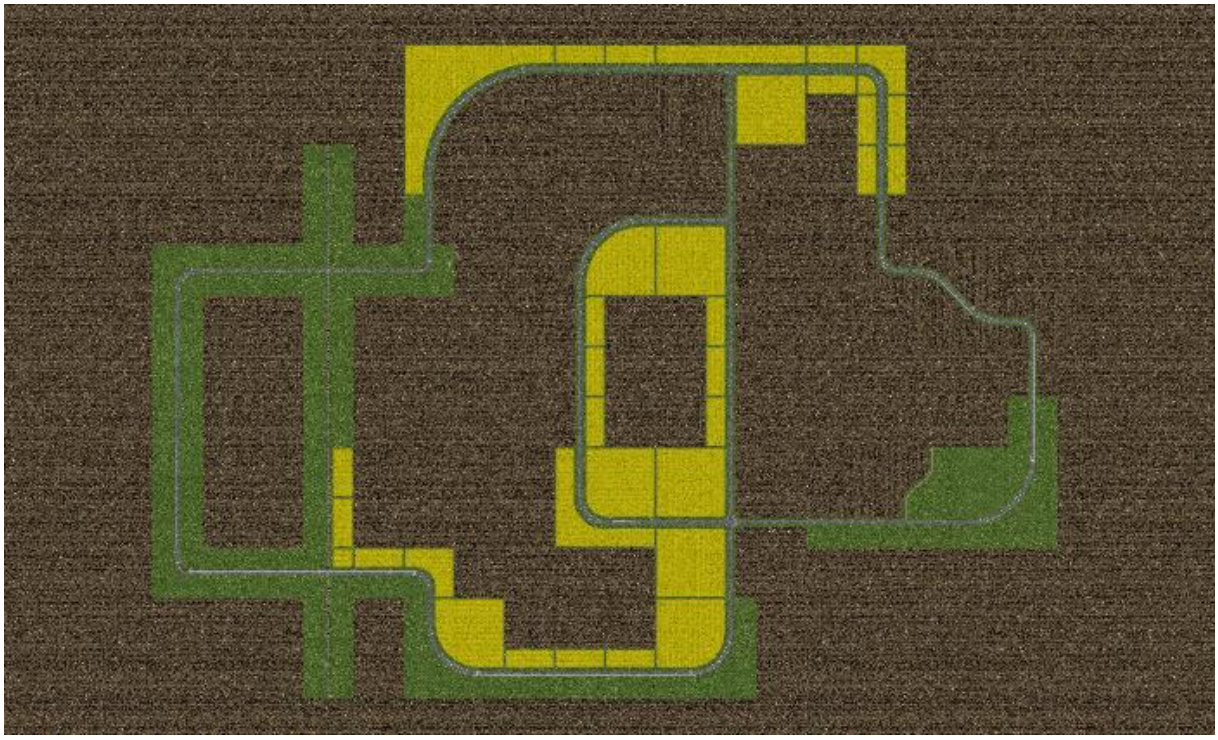
```
7           doplňuj 0 k části za desetinnou čárku, dokud nedojdeš k požadovanému počtu
            desetinných míst
8           ***Decimal*** = první část čísla + desetinná čárka + druhá část čísla + přidané nuly
9       Else // jedná se o celé číslo
10          doplňuj 0, dokud nedojdeš k požadovanému počtu desetinných míst
11          ***Decimal*** = číslo + desetinná čárka + přidané nuly
12      End If
13  // vytvoření složky animation a souborů v ní
14  folder = uživatel vybere adresář pro uložení složky animation
15  newfolder = vytvoř složku animation v adresáři folder
16  vytvoř v newfolder soubor auto(i).anim (i = pořadí vytvářeného souboru) // např. pokud již ve
    složce existuje soubor auto1.anim, vytvoří se auto2.anim
17  cas = uživatel zadá počáteční čas pro spuštění animace
18  repeat = uživatel zvolí, zda se má animace opakovat
19  krivka = uživatel vybere křivku trasy provozu
20  krivkapoly = rozděl křivku krivka body (na přímé po 5 metrech, v obloucích hustěji)
21  points = body na křivce krivkapoly
22  krivkarychlost = uživatel vybere křivku rychlosti
23  plochysilnic = vyber všechny objekty ve vrstvě Silnice
24  vykresli pomocný kříž pro určení náklonu vozidla (Pitchline: ±1.3 v ose x, Rollline: ±0.1 v ose y)
25  For Each point In points
26      Yaw = směrový úhel křivky v bodě point
27      prompoint = promítni bod point na všechny plochysilnic, pokud se promítne, ulož danou
        plochu
28      distance = vzdálenost mezi bodem prompoint a minulým bodem
29      speed = rychlost podle grafu rychlosti - hodnota v bodě určeném pomocí distance
30      time = čas v minulém bodě + distance/speed
31      zkopíruj Pitchline a Rollline na bod point, orotuj je podle úhlu Yaw
32      promítni počáteční a koncový bod Pitchline a Rollline na plochysilnic
33      vypočítej úhel natočení Pitch a Roll
34      zapiš nový řádek do souboru auto(i).anim: "time, souřadnice x, y, z bodu prompoint, úhly
        Roll, Yaw, Pitch" - hodnoty uprav pomocí funkce ***Decimal***
35  Next
36  vytvoř v newfolder soubor animation.ini
37  zapiš do něj potřebné údaje pro mraky, auto(i)
38  zkopíruj do složky newfolder soubor mraky.anim, model zvoleného vozidla (soubor pbtz),
    texturu pro zvolený typ a barvu vozidla (soubor dds)
```

9 Experiment

Při využití nového pluginu KS_Traffic pro automatické generování provozu vzniká složka animation, která obsahuje veškeré potřebné soubory pro zahájení animace ve vozidlovém simulátoru. Po spuštění grafického enginu se spustí animace provozu ve vymodelovaném scénáři, a to podle toho, jak si ji uživatel během generování animačních souborů nastavil.

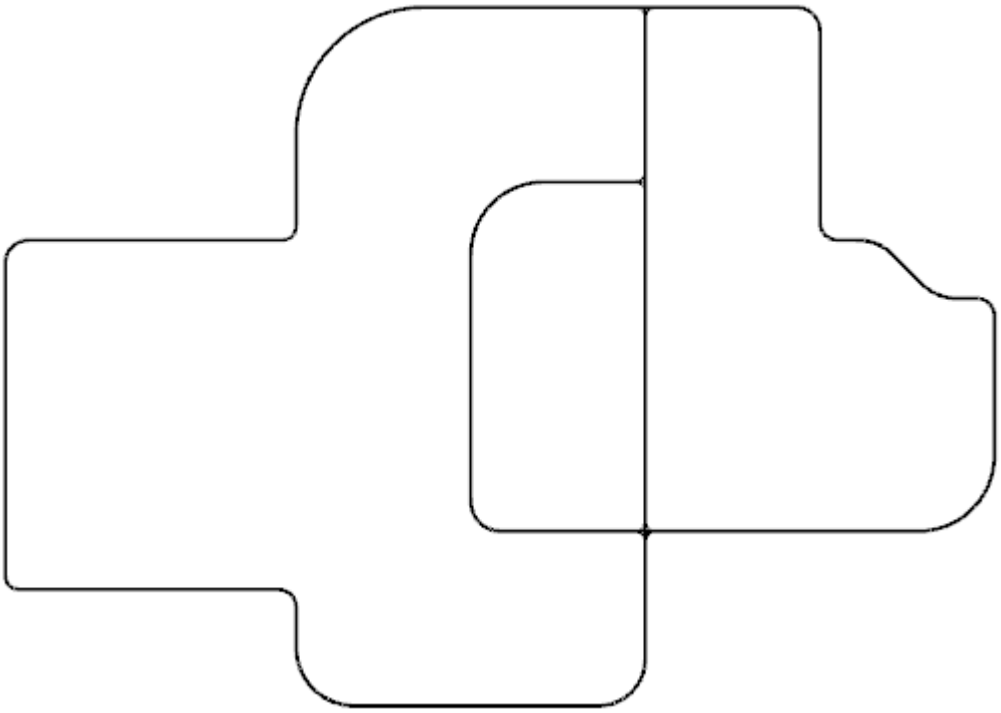
Pro otestování funkčnosti nové metody jsem pomocí KS_Software vytvořila novou virtuální scénu, ve které jsem využila všechny typy segmentů v databázi. Pro tuto scénu jsem poté pomocí jednotlivých funkcí nového pluginu KS_Traffic rozložila všechny bloky do vrstev, vybrala trasu pro provoz, vytvořila pro ni rychlostní křivku a vygenerovala složku s animačními soubory. Ty jsem poté spolu s modelem vytvořené virtuální scény nahrála do grafického enginu a takto vytvořený scénář s animací provozu jsem otestovala ve vozidlovém simulátoru na Fakultě dopravní.

Výsledky testování nové metody jsou vidět na obrázcích č. 37 - 45.

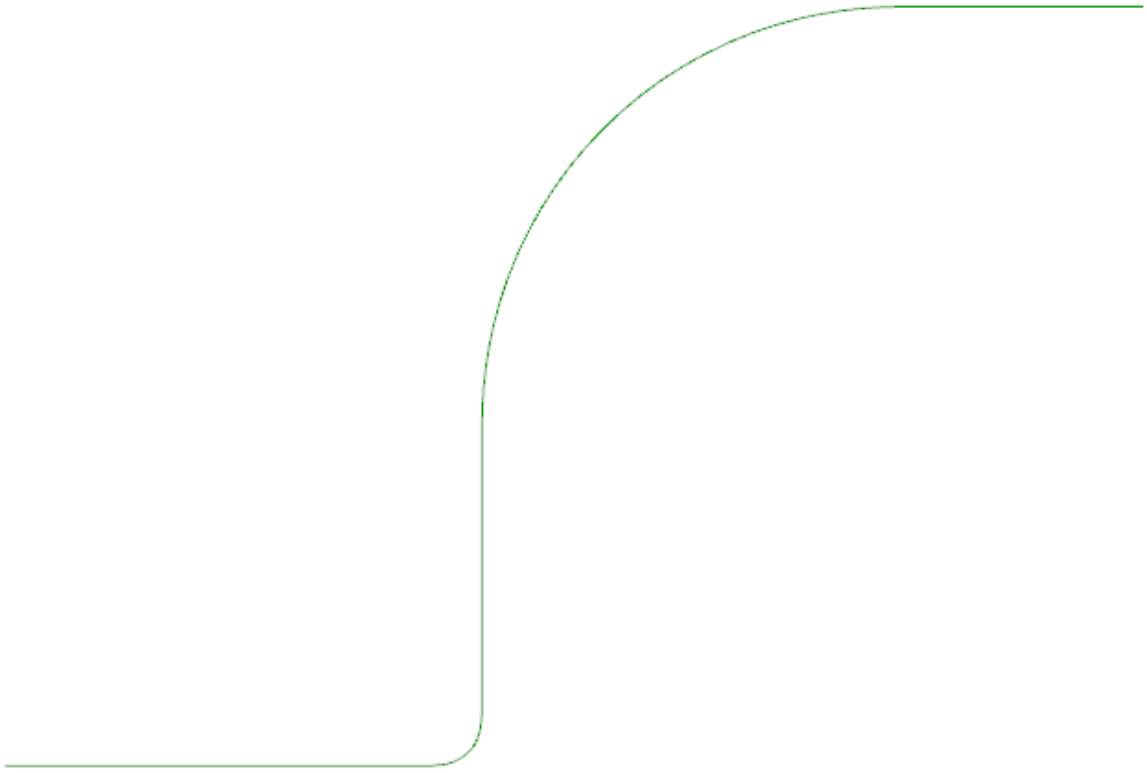


Obrázek 37: Scéna vytvořená pro testování pomocí KS_Software

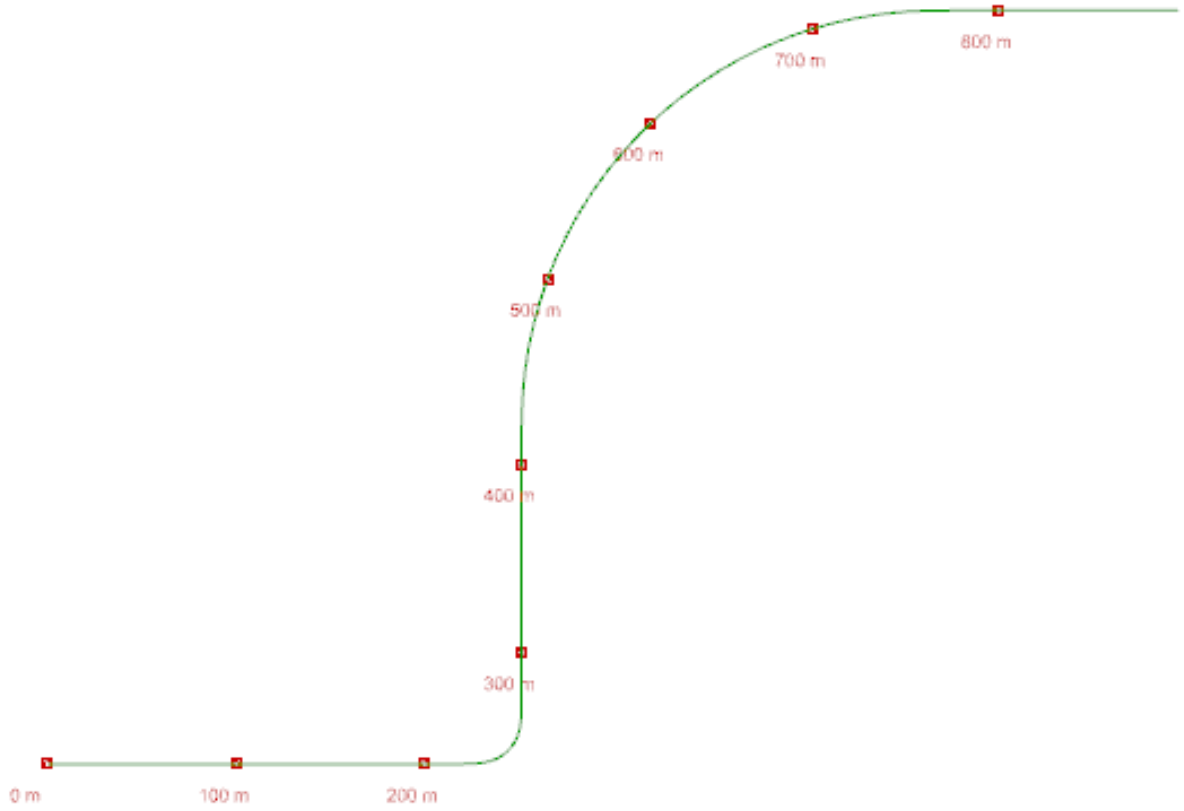
Scéna vytvořená pro experiment se skládá celkem z 57 segmentů, z toho je 26 přímých, 2 přímé do stoupání, 15 oblouků, 3 křižovatky, 2 železniční přejezdy a 9 železničních tratí. Pro dotvoření krajiny leží celá scéna na velké ploše s texturou orby.



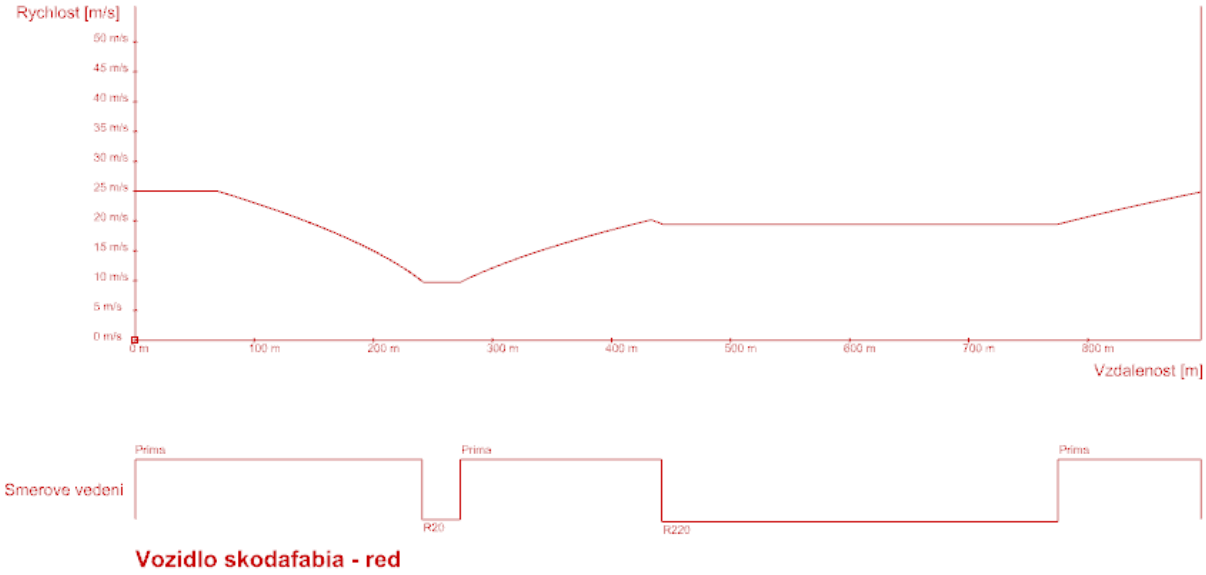
Obrázek 38: Možné trasy ve scéně, které jsou na výběr



Obrázek 39: Vybraná trasa



Obrázek 40: Vybraná trasa se staničením



Obrázek 41: Rychlostní křivka pro vybranou trasu

Obrázek 42: Testovací jízda v grafickém enginu

Do grafického enginu lze prvním spuštěním funkce pluginu pro vytvoření animačních souborů vložit jízdu jednoho vozidla, a jeho opětovným použitím můžeme snadno přidat další vozidla – uživatel si opět může zvolit, jaké vozidlo pojede jako další v pořadí a jaká bude časová mezera mezi jednotlivými vozidly.

Obrázek 43: Testovací jízda v grafickém enginu (jedno vozidlo)

Obrázek 44: Testovací jízda v grafickém enginu (2 vozidla) – testování náklonu vozidel

Každá animace může mít jiné parametry – jiný typ nebo barva vozidla, jiná rychlostní křivka, jiný čas začátku a podmínky animace, ale například i jiná trasa, po které animované vozidlo pojede.

Obrázek 45: Testovací jízda v grafickém enginu (2 vozidla s jinou trasou)

Práce s novou metodou je efektivní a intuitivní, lze pomocí ní snadno vytvořit kompletní virtuální scénář ve velmi krátké době. Výsledná animace provozu působí realisticky, a to zejména díky reálným datům, na nichž je založena – rychlosti získané měřením v reálném provozu, skutečné výkony vybraných vozidel a z nich vycházející hodnoty maximálního zrychlení, dále pak zvýšená hustota bodů v obloucích na trase a přesně určené úhly natočení v každém tomto bodě.

9.1 Přínosy nové metody

Nová metoda nazvaná KS_Traffic velice usnadňuje přidávání animace provozu do scénářů pro různé experimenty na vozidlovém simulátoru, a spolu s pluginem KS_Software pro tvorbu virtuální scény ze segmentů komunikací tvoří velmi efektivní kompletní nástroj pro snadnou a rychlou tvorbu scénářů pro nové experimenty na vozidlovém simulátoru na Fakultě dopravní.

Hlavní výhodou nové metody je především úspora času stráveného přípravou křivek pro trasu provozu, vykreslováním rychlostní křivky a tvorbou animačních souborů. Práce, která by jinak trvala i několik dnů, se tak dá zvládnout efektivně v řádu několika minut. Jedná se tedy o výraznou pomoc při tvorbě nových scénářů.

Dalším plusem je také fakt, že práce s nově vytvořeným pluginem je snadná a uživatelsky přívětivá, může ji tak zvládnout i člověk bez větších znalostí 3D modelování a práce s grafickými enginy.

10 Závěr

Cílem této diplomové práce bylo vytvořit novou metodu pro automatické generování provozu do scénářů pro vozidlové simulátory, a to zejména do scénářů vytvořených ze segmentů, tj. pomocí metody KS_Software vytvořené v rámci mé bakalářské práce, na kterou tato diplomová práce navazuje.

V úvodních kapitolách byla objasněna problematika vozidlových simulátorů, popsán jejich účel a funkčnost. Jsou to nenahraditelné nástroje pro trénování řidičů a jejich testování v bezpečných podmínkách virtuálního prostředí, díky čemuž je možné na nich zkoumat reakce řidičů na nejrůznější situace a scénáře, které by v reálném provozu byly příliš nebezpečné, drahé nebo těžko proveditelné. Byly popsány základní komponenty simulátorů a vysvětleny principy tvorby virtuálního prostředí do nich. To se vytváří s využitím 2D i 3D počítačové grafiky, a to vždy na míru danému experimentu, který se na simulátoru bude provádět.

Dále byl popsán plugin KS_Software, který slouží k efektivnímu generování virtuální scény, a to pomocí výběru z databáze předem připravených segmentů komunikací, které na sebe perfektně navazují a ze kterých lze poskládat kompletní scénu. Tento nástroj, který jsem vytvořila v rámci své bakalářské práce, nyní velmi usnadňuje práci na nových scénářích pro některé experimenty, proto jsem se ve své diplomové práci rozhodla jej rozšířit o nové funkce, konkrétně o funkci přidání animace provozu. Provoz se totiž vkládá do drtivé většiny scénářů, které se vytváří pro experimenty na vozidlovém simulátoru, a jeho vytvoření je přitom velmi časově náročné. Proto nalezení nové metody, která tento proces urychlí a zjednoduší, velmi usnadní práci na nových scénářích vytvářených na Fakultě dopravní. Za tímto účelem jsem nejprve nastudovala možné současné metody tvorby provozu do vozidlových simulátorů, a na základě nich jsem popsala svoji novou metodu, kterou jsem následně aplikovala do nově vytvořeného pluginu.

V další části této diplomové práce jsem se věnovala popisu chování vozidla během jízdy. Popsala jsem zákonitosti z pohledu podélné, příčné i svislé dynamiky, a zhodnotila jsem, které z těchto parametrů následně využiji při definici animace jízdy vozidla. Jsou to zejména souřadnice určující polohu a natočení vozidla v prostoru, rychlost jízdy vozidla a hodnoty jeho komfortního a maximálního zrychlení.

Poté jsem popsala praktické měření rychlosti průjezdu vozidel směrovými oblouky, jejíž hodnoty jsem následně využila v novém pluginu. Toto měření jsem prováděla s využitím vlastního měřicího zařízení, které jsem pro tento účel vytvořila pomocí Arduina. Princip měření jsem zvolila takový, že v měřeném místě jsem pomocí dvou laserů vytvořila skrz jízdní pruhy světelnou závoru, a pomocí snímačů laserových paprsků umístěných na protější straně silnice jsem měřila čas, za který se jízdou měřeného vozidla postupně přeruší první i druhý laserový

paprsek. Z rozdílu časů přerušení těchto paprsků a známé vzdálenosti snímačů jsem poté dopočítala průměrnou rychlost průjezdu vozidel všemi měřenými oblouky o různém poloměru, od R20 až po R220.

V dalších kapitolách bylo popsáno vytvoření pluginu pro 3D modelovací program Rhinoceros, který umožňuje generování provozu do scénářů pro vozidlové simulátory, zejména těch vytvořených segmentovou metodou KS_Software – tu jsem navíc doplnila o nové segmenty a o křivky provozu, které kopírují všechny možné trasy (jízdní pruhy) na segmentech v databázi.

Tento nový plugin jsem nazvala KS_Traffic. Pomocí něj lze připravit původní scénu, tj. rozložit původní bloky na jednotlivé objekty a rozřadit je do podvrstev. Dále je možné ze všech nabízených tras vybrat tu, kterou chceme použít pro generování provozu, a to buď ručně, tj. výběrem jednotlivých křivek a jejich spojením, nebo automaticky – uživatel si může vybrat pouze start a cíl trasy a algoritmus inspirovaný Theseovou metodou popsanou v Teorii grafů výslednou trasu sám najde a propojí.

Poté můžeme pro vybranou trasu vykreslit křivku rychlosti, a to pomocí další funkce nového pluginu. Ta na trase vytvoří staničení, uživatel si vybere typ a barvu vozidla a poté se vykreslí samotná rychlostní křivka. Jsou v ní zohledněny nejen rychlosti změřené během experimentu popsaného výše, ale i parabolické křivky zrychlení, jehož hodnota vychází z komfortního zrychlení, případně z maximální možné hodnoty zrychlení pro vybraný typ vozidla (podle výkonu jeho motoru) a danou aktuální rychlost. Způsob výpočtu tohoto zrychlení je rovněž v práci popsán.

Pomocí poslední části pluginu je možné pro vybranou trasu a rychlostní křivku vygenerovat animační složku, která obsahuje veškeré potřebné soubory nutné ke spuštění animace provozu v simulátoru – animační soubory s daty popisujícími pohyb vozidla, ale i model a textura vybraného vozidla. Po vložení těchto souborů do grafického enginu, který se používá na Fakultě dopravní, je již možné vytvořenou animaci provozu spustit.

Všechny funkce nového pluginu KS_Traffic jsem vytvořila pomocí editoru RhinoScript Editor pro 3D modelovací program Rhinoceros, a to pomocí skriptovacího jazyka VBScript. Všechny skripty jsou v této diplomové práci přepsány formou pseudokódu, kde jsem srozumitelně popsala jejich algoritmus, aniž by bylo třeba znát syntaxi daného jazyka. Jednotlivé kódy jsem dostatečně okomentovala, aby se v nich vyznal každý, kdo se k nim v budoucnu dostane a chtěl by je dále rozvíjet, či je použít pro svou další práci.

Celý plugin je navržen tak, aby byl uživatelsky přívětivý a aby práce s ním byla snadná a nevyžadovala žádné větší znalosti. Toho je docíleno například i propojením

s HTML rozhraním, pomocí něhož si uživatel vybírá například způsob výběru trasy či typ a barvu vozidla, pro něž chce vykreslit rychlostní křivku.

Výsledkem práce je funkční plugin, který umožňuje bez větších znalostí 3D modelování a práce s grafickými enginy a animacemi vytvořit veškeré animační soubory, které jsou třeba pro přidání animace jízdy vozidla do virtuálního scénáře. Díky mnoha možnostem přizpůsobení podoby výsledné animace podle konkrétních požadavků daného experimentu je metoda velice komplexní a široce využitelná. Vzniklý plugin KS_Traffic tvoří spolu s pluginem KS_Software kompletní nástroj pro rychlou a efektivní tvorbu virtuálních scénářů do vozidlových simulátorů. Největší výhodou této metody je úspora času. Porovnání časové náročnosti tvorby virtuálních scénářů je znázorněno v tabulce č. 5.

Tabulka 5: Čas nutný k vytvoření virtuálního scénáře

Metoda tvorby scénáře	**Tvorba scény**	**Tvorba animací a provozu**	**Příprava experimentu**
Bez využití pluginů	1 – 2 týdny	1 – 2 dny	týdny
KS_Software	hodina	1 – 2 dny	dny
KS_Software + KS_Traffic	hodina	hodina	hodiny

Čas nutný k vytvoření virtuálního scénáře se odvíjí od jeho rozsahu a složitosti, ale lze s jistotou říci, že s využitím pluginu KS_Software a KS_Traffic lze scénáře vytvářet řádově mnohem rychleji a efektivněji. Úspora času stráveného vytvářením provozu do simulátoru je proto hlavním přínosem této diplomové práce.

Vytvořený plugin se jistě stane do budoucna cenným nástrojem pro generování provozu pro vozidlový simulátor na Fakultě dopravní.

11 Použité zdroje

11.1 Literatura

[1] APELTAUER, Tomáš. *Dopravní inženýrství - Modul 2: Modelování dopravního proudu.* Brno: VUT, Fakulta stavební, 2007.

[2] ORLICKÝ, Adam. *Automatická tvorba silniční infrastruktury ve 3D pro vozidlové simulátory*. Praha, 2016. Diplomová práce. Fakulta dopravní, ČVUT. Vedoucí práce Doc. Ing. Stanislav Novotný, Ph.D.

[3] PUDOVÁ, Klára. *Tvorba počítačových 3D modelů segmentů komunikací pro vozidlové simulátory.* Praha, 2017. Bakalářská práce. Fakulta dopravní, ČVUT. Vedoucí práce Doc. Ing. Stanislav Novotný, Ph.D, Ing. Adam Orlický.

[4] VLK, František. *Dynamika motorových vozidel: jízdní odpory : hnací charakteristika : brzdění : odpruženost : řiditelnost, ovladatelnost : stabilita*. Brno: VLK, 2000. ISBN 80-238-5273-6.

[5] VOLEK, Josef a Bohdan LINDA. *Teorie grafů - aplikace v dopravě a veřejné správě*. Pardubice: Univerzita Pardubice, 2012. ISBN 978-80-7395-225-9.

[6] ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. *Moderní počítačová grafika*. Praha: Computer Press, 1998. ISBN 80-7226-049-9.

11.2 Internetové zdroje

[7] ARDUINO.CZ: Průvodce světem Arduina [online]. 2014 [cit. 2019-03-11]. Dostupné z: https://arduino.cz/

[8] Arduino-forum.cz: České fórum pro všechny nadšence do Arduina a dalších technologií. [online]. 2019 [cit. 2019-03-11]. Dostupné z: https://www.arduino-forum.cz/

[9] Arduino Forum [online]. 2019 [cit. 2019-03-11]. Dostupné z: https://forum.arduino.cc/

[10] Arduino originály. In: *HW Kitchen* [online]. 2019 [cit. 2019-03-11]. Dostupné z: https://www.hwkitchen.cz/arduino-originaly/

[11] Citroen C1: Technické parametry. *C-Car s.r.o*. [online]. [cit. 2019-03-18]. Dostupné z: http://www.c-car.cz/foto_vozy/technickepdf_417.pdf

[12] Co je to Arduino?. In: *ARDUINO.CZ* [online]. 2014 [cit. 2019-03-11]. Dostupné z: https://arduino.cz/co-je-to-arduino/

[13] Download the Arduino IDE. In: *Arduino* [online]. 2019 [cit. 2019-03-11]. Dostupné z: https://www.arduino.cc/en/Main/Software

[14] Fiat Ducato Combinato L1H1 30 2.3 MultiJet 130: Technická data. *AUTONOTO* [online]. [cit. 2019-03-18]. Dostupné z: https://autonoto.cz/katalog/detail/fiat-ducato-combinato-l1h1-30-2-3-multijet-130

[15] Ford Transit Kombi 300M FWD 2.2 TDCi 100hp Ambiente: Technická data. *AUTONOTO* [online]. [cit. 2019-03-18]. Dostupné z: https://autonoto.cz/katalog/detail/ford-transit-kombi-300m-fwd-2-2-tdci-100hp-ambiente

[16] LILLING, Ondřej. BMW M5: Raketová věda. *Auto.cz* [online]. [cit. 2019-03-18]. Dostupné z: https://www.auto.cz/bmw-m5-raketova-veda-123471

[17] *Mapy.cz: OpenStreetMap* [online]. Seznam.cz [cit. 2019-05-02]. Dostupné z: https://mapy.cz/

[18] MÉGANE IV: Kompletní technická data a ceny ve Francii. *RNews.cz* [online]. [cit. 2019-03-18]. Dostupné z: http://rnews.cz/megane-iv-kompletni-technicka-data-a-ceny-ve-francii/

[19] Škoda Citigo: katalog. *ŠKODA: Autocentrála s.r.o.* [online]. [cit. 2019-03-18]. Dostupné z: http://www.autocentrala.cz/img/modely/citigo/skoda_citigo_katalog.pdf

[20] Škoda Fabia: katalog. *CB Auto* [online]. [cit. 2019-03-18]. Dostupné z: https://www.cb-auto.cz/pdf/cbauto/57d6b36315f60_fabia-katalog2.pdf

[21] Škoda Octavia: katalog. *Autobible.cz* [online]. [cit. 2019-03-18]. Dostupné z: https://autobible.euro.cz/wp-content/uploads/2017/01/Technick%C3%A9-%C3%BAdaje-%C5%A0koda-Octavia-2017-z%C3%A1%C5%BEehov%C3%A9-motory.pdf

[22] Škoda Yeti: katalog. *ŠKODA: Storyboard* [online]. [cit. 2019-03-18]. Dostupné z: https://cdn.skoda-storyboard.com/2016/12/TD_YETI_cz.pdf

[23] The Citaro city buses: Technical information. *Mercedes-Benz* [online]. [cit. 2019-03-18]. Dostupné z: https://www.mercedes-benz-bus.com/cs_CZ/models/citaro.html

[24] Začínáme s Arduinem ve Windows. In: *ARDUINO.CZ* [online]. 12. 8. 2014 [cit. 2019-03-11]. Dostupné z: https://arduino.cz/zaciname-s-arduinem-ve-windows/

12 Seznam obrázků

13 Seznam tabulek

14 Seznam příloh

Obsah CD

Měřicí zařízení

- detektor.ino (software nahraný v Arduinu)

Plugin

Skripty pro jednotlivé funkce:
- KS_Software.rvb
- KS_Traffic1_Rozklad_bloku.rvb
- KS_Traffic2_Vyber_trasy.rvb
- KS_Traffic3_Krivka_rychlosti.rvb
- KS_Traffic4_Animacni_soubory.rvb

Soubory pro rozhraní HTML:
- index_venkov_dlazdice.html
- index_venkov_priroda.html
- index_volba_barva_vozidlo.html (pro všechny typy vozidel v databázi)
- index_volba_trasy.html
- index_volba_vozidla.html
- obrázky segmentů, vegetace, vozidel, barev vozidel

Soubory v databázi vozidel:
- model vozidla: vozidlo.3dm, vozidlo.pbt, vozidlo.pbtz (pro všechny typy a barvy vozidel)
- textura vozidla: vozidlo.dds (pro všechny typy a barvy vozidel)

Výchozí soubor pro použití skriptů v programu Rhinoceros:
- KS_vychozi.3dm

Scénář vytvořený pro experiment:
- KS_Traffic.3dm
- Složka se všemi soubory potřebnými pro spuštění scénáře v grafickém enginu (spouštěcí soubor: AutoSimt.exe)

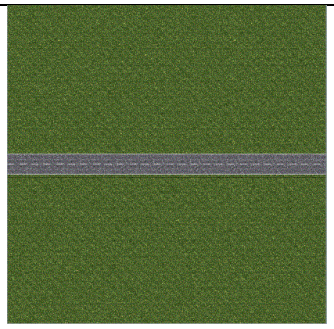
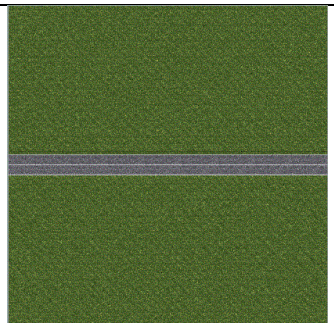
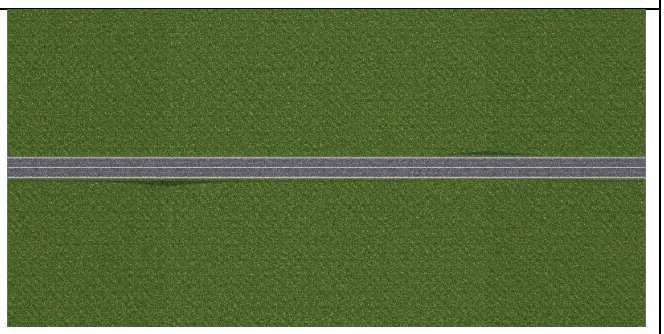
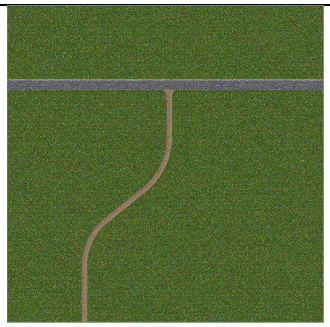
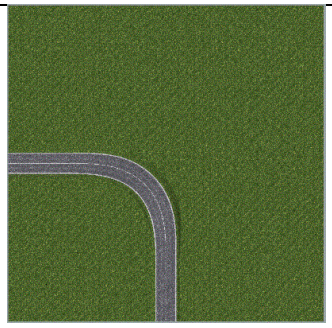
Textury

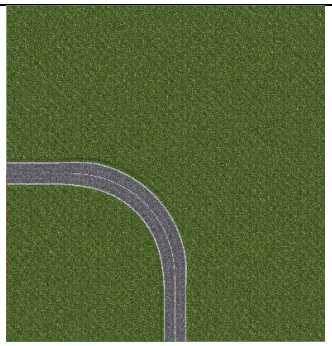
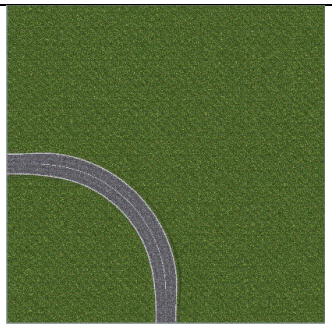
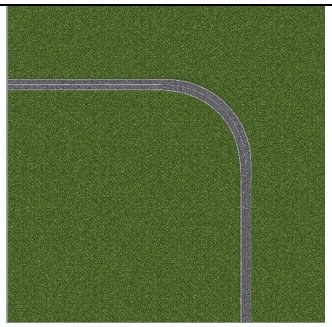
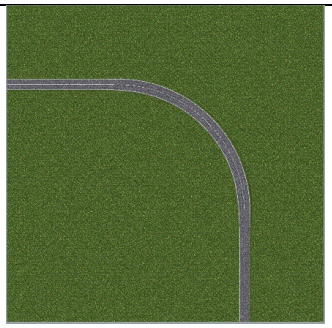
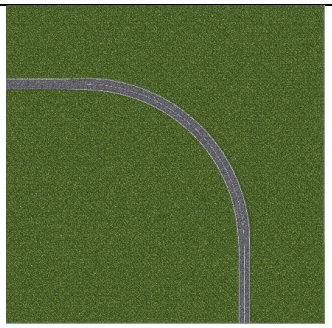
Diplomová práce

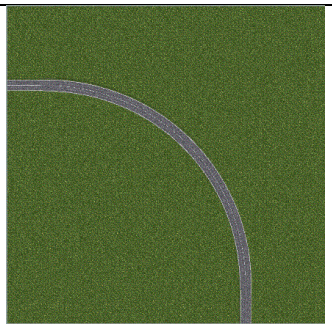
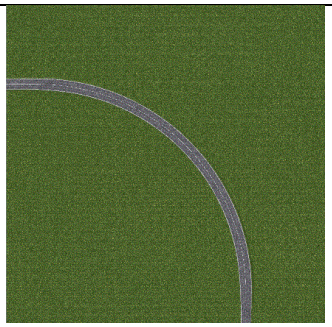
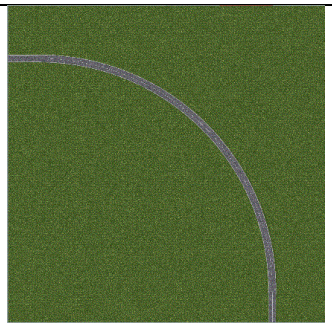
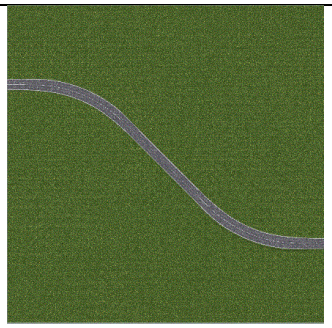
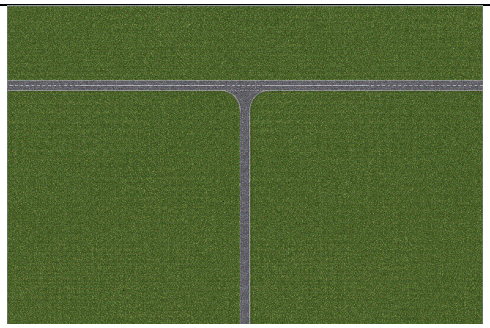
- KS_Traffic.pdf

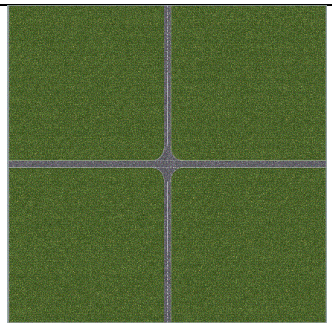
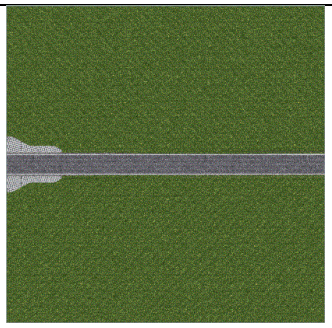
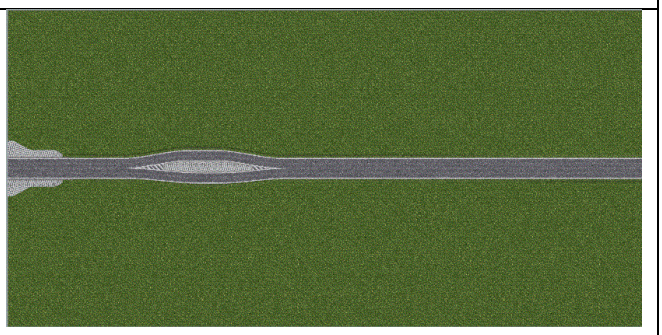
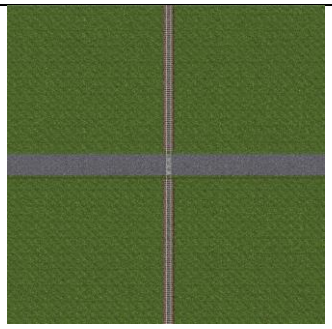
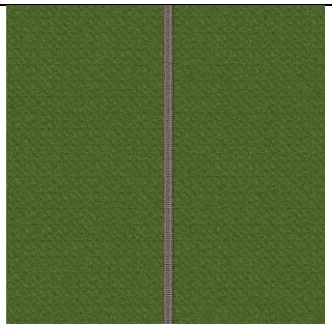
Veškeré obrázky, tabulky a přílohy, které nemají uvedený zdroj, byly vytvořeny autorkou práce.

Příloha 1: Připravené segmenty v databázi

Typ segmentu	Náhled
Přímá přerušovaná	
Přímá plná	
Přímá s 12% stoupáním	
Přímá s nájezdem na pole	
Oblouk R20 (s velkým převýšením)	

Oblouk R30	
Oblouk R40	
Oblouk R50	
Oblouk R80	
Oblouk R100	

Oblouk R125	
Oblouk R130	
Oblouk R220	
Oblouk R50	
T-křižovatka	

X-křižovatka	
Přechod 1	
Přechod 2	
Železniční přejezd	
Železniční trať	

Příloha 2: Technické parametry vozidel

	hmotnost [kg]	c_x [-]	max. rychlost [m/s]	P [W]	P na překonání [W]	odpor valivý	odpor vzdušný	S_x [m^2]
Škoda Citigo	1290	0,33	44,44	44000	39600	5624,40	33975,60	1,91
Škoda Fabia	1510	0,32	47,78	55000	49500	7077,37	42422,63	2,01
Škoda Octavia	1775	0,28	56,39	85000	76500	9818,86	66681,14	2,17
Škoda Yeti	1900	0,36	51,94	92000	82800	9681,93	73118,08	2,34
BMW M5	1855	0,31	84,72	441000	396900	15417,37	381482,63	3,30
Citroen C1	1190	0,34	43,61	50000	45000	5091,12	39908,88	2,33
Renault Megane	1914	0,32	53,06	84000	75600	9961,89	65638,11	2,24
Autobus	19500	0,55	27,78	220000	198000	53137,50	144862,50	7,89
Fiat Ducato	1950	0,35	43,06	96000	86400	8236,31	78163,69	4,62
Ford Transit	1916	0,35	41,11	74000	66600	7727,23	58872,77	3,92
Kamion malý	12000	0,67	25,00	250000	225000	29430,00	195570,00	7,64
Kamion velký	30000	0,86	25,00	350000	315000	73575,00	241425,00	7,64