Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Diplomski rad br. 565

Računalna grafika i animacija

Fizikalno temeljen model vozila

Autor:

Petar Mrazović

Mentor:

Prof. dr. sc. Željka Mihajlović

Zagreb, lipanj 2013.

Sadržaj

1 Uvod **3**

1.1 Simulatori motornih vozila 3

1.2 Fizikalni pogoni u računalnim simulacijama 5

2 Klasična mehanika **6**

2.1 Newtonovi aksiomi 7

2.1.1 Aksiom tromosti 7

2.1.2 Temeljni aksiom gibanja 8

2.1.3 Aksiom djelovanja i protudjelovanja 8

2.2 Vrste sila u klasičnoj mehanici 9

2.2.1 Gravitacijska sila 9

2.2.2 Sila trenja 11

2.2.3 Elastična sila 14

2.2.4 Centripetalna sila 18

3 Fizikalni model vozila **20**

3.1 Dijagram sila 20

3.2 Snaga i okretni moment motora 22

3.3 Stupnjevi prijenosa i okretni moment kotača 23

3.4 Sile otpora 27

3.4.1 Aerodinamička sila otpora 27

3.5 Izračun ubrzanja i brzine automobila 28

3.6 Kočenje 30

3.7 Proklizavanje kotača 31

3.8 Gibanje u zavojima 31

4 Programska implementacija **33**

4.1 XNA okruženje 33

4.1.1 Struktura XNA programskog koda 33

4.2 BEPUphysics fizikalni pogon 34

4.3 Simulator automobila 35

4.3.1 3D modeli 35

4.3.2 Generiranje terena 37
4.3.3 Kamere 38
4.3.4 Automobil 39
4.3.5 Grafičko korisničko sučelje 44
4.4 Parametri simulacije 45

5 Zaključak **46**

6 Sažetak **52**

7 Summary **53**

1 Uvod

Ubrzanim razvojem novih tehnologija konstantno se traže novi načini kojima bi se utjecalo na proširivanje ljudskog doživljaja interakcije s računalnim svijetom. Posebno atraktivne postaju tehnologije iz relativno novog područja vizualne računarske znanosti, računalne grafike, koja pronalazi sve širu primjenu u raznim sferama ljudskog života. Računalna grafika počinje se razvijati 60-tih godina prošlog stoljeća s ciljem stvaranja slika ili uklapanja i mijenjanja slikovnih i prostornih podataka koji su uzeti iz stvarnosti pomoću računala. Ostvariti računalnu predodžbu vizualne stvarnosti u ono vrijeme se činilo vrlo teškim i neizvedivim zadatkom, no današnji filmovi, video igre ili razne realizacije proširene stvarnosti nam ukazuju na brzinu kojom računalna grafika napreduje.

Tehnologije računalne grafike visoko su primjenjive u širokom rasponu ljudskih djelatnosti. Razvoj korisničkih sučelja, grafičko programiranje, interaktivno crtanje, grafičko projektiranje, simulacija i animacija samo su neke od disciplina kojima se bavi računalna grafika. Posebno zanimljiv i čest zadatak računalne grafike je ostvariti interaktivno virtualno okruženje generiranjem računalne predodžbe vizualne stvarnosti. Ovim zadatkom ćemo se i mi baviti u diplomskom radu *Fizikalno temeljen model vozila*. Pokušat ćemo tehnike računalne grafike iskoristiti za znanstveno-inžinjersku vizualizaciju, ali i zabavu, te implementirati simulator vozila upoznavajući se pritom s apstraktnim matematičko-fizikalnim modelom vremenski promjenjivih pojava s kojima se susrećemo upravljajući motornim vozilom.

1.1 Simulatori motornih vozila

Većina današnjih simulatora motornih vozila razvijaju se kao računalne igre koje pružaju igračima vizualni doživljaj perspektive pilota ili vozača stvarnog motornog vozila. Glavni izazov ovakvih simulatora je svladati upravljanje vozilom iz perspektive vozača, a često se nameću dodatni ciljevi kao što su utrke ili borbe sa suparničkim vozilima.

Veliko tržište igara-simulatora podijeljeno je na kompleksne realistične simulacije, te manje zahtjevne zabavne arkadne igre.

Realistične simulacije

Realistične simulacije nastoje uvjerljivo reproducirati stvarno upravljanje vozilom.

Ovakve igre često nastaju u suradnji s proizvođačima motornih vozila kako bi se iskoristili stvarni rezultati mjerenja dinamike vozila. Profesionalni vozači i piloti obično su također uključeni u proces razvoja ovakvih simulatora. Korisnici realističnih simulatora suočeni su s većinom izazova kao i profesionalni vozači što čine upravljanje vozilima iznimno zahtjevnim. Korisnici simulatora trkaćih automobila će primjerice morati voditi računa o tehnici i preciznosti pokreta u zavojima, broju okretaja motora i preciznom mjenjanju brzina, stanju guma ili razini goriva. Ponekad ovakav realizam čini igre izrazito neigrivim, pa one često nude mogućnost uključivanja pomoći kao što su kontorla proklizavanja, ABS, asistiranje pri zakretanju volana, otpornost na oštećenja, itd.. Primjeri ovakvih simulatora su igre Gran Tourismo, Colin McRae Rally, Test Drive i Formula One.

Arkadne simulacije

Arkadne simulacije stavljaju zabavu i brzo stjecanje iskustva ispred strogih zakona fizike. Ključno obilježje arkadnih simulatora je opušteno fizikalno ponašanje vozila, što nije slučaj kod prethodno opisanih relističnih simulatora. Korisnici su motivirani da isprobaju življu vožnju ne brinući se pritom o preciznosti pokreta prilikom upravljanja vozilom. Sudari s drugim vozilima i objektima neizostavni su element svake arkadne simulacije, te su često prenaglašeno modelirani što čini simulatore zabavnijima. Neki od poznatijih arkadnih simulatora motornih vozila su primjerice Need for Speed, Carmageddon, Midnight Club, Burnout, itd..

Slika 1.1: Realistični i arkadni simulatori (*Gran Tourismo, Need for Speed*)

Ovaj diplomski rad pratit će proces razvoja i implementacije vlastitog simulatora vozila, preciznije automobila, koji predstavlja praktični rezultat rada. Simulator implementiran ovim radom moći ćemo svrstati u obje prethodno opisane kategorije. Naime,

implementirat ćemo mogućnost promjena parametara simulacije tijekom izvođenja programa što će uvelike utjecati na fizikalno ponašanje vozila, pa tako i na sam karakter simulatora. Primjerice, moći ćemo mijenjati parametre šasije, motora, amortizera, kotača i podloge, što će učiniti naš simulator iznimno zabavnim, ali i primjenjivim u praktičnim analizama ponašanja automobila.

Da bi implementirali vlastiti simulator, moramo moći razumjeti fizikalno ponašanje vozila, te osmisliti fizikalni model koji će ga opisivati. Upravo će ovo biti središnja tema diplomskog rada. Na samom početku bavit ćemo se jednostavnim fizikalnim modelom gibanja automobila koji ćemo i implementirati koristeći programski jezik C# i XNA radni okvir. U nastavku ćemo zatim unaprijediti fizikalno ponašanje vozila koristeći jedan od javno dostupnih pogona fizikalnog ponašanja.

1.2 Fizikalni pogoni u računalnim simulacijama

Grafički fizikalni pogoni uključuju uvođenje zakona fizike u grafičke simulacije kako bi pojave učinili stvarnijima za promatrača. Tipično je simulacijska fizika samo približna stvarnim zakonima fizike, budući da koristi brojna pojednostavljenja, te naravno diskretne vrijednosti prilikom izračuna. Simulacijska fizika najčešće se sastoji od dvije glavne komponente:

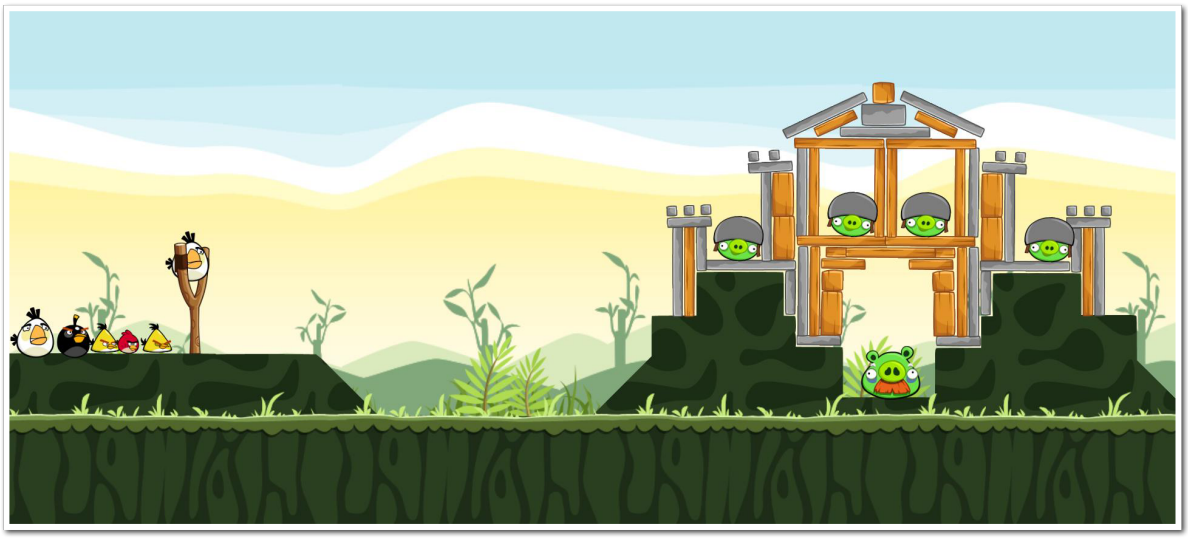
Simulacija dinamike (eng. *dynamics simulation*)

Programski kod koji simulira zakone klasične Newtonove fizike u jasno definiranom okruženju.

Otkrivanje kolizije (eng. *collision detection*)

Programski kod koji rješava probleme preklapanja dva ili više fizičkih objekata.

U sljedećem poglavlju upoznat ćemo se s osnovnim zakonima Newtonove fizike, te ih zatim iskoristiti kako bi opisali jednostavno gibanje automobila koje ćemo i programski implementirati. Kasnije ćemo se susreti i s gotovim fizikalnim pogonima, te detaljnije proučiti njihov dizajn. Simulacijska fizika sastavni je dio dizajna većine modernih video igara, budući da nam njeno uvođenje donosi brojne prednosti, od povećanja igrivosti do stvarnijeg izgleda gibanja. Ponekad su kreativna ideja i fizikalni pogon sve što je potrebno za uspjeh računalne igre. Najpoznatiji primjer takve igre je popularni *Angry Birds*.



Slika 1.2: Primjer jednostavnog fizikalnog pogona (*Angry Birds*)

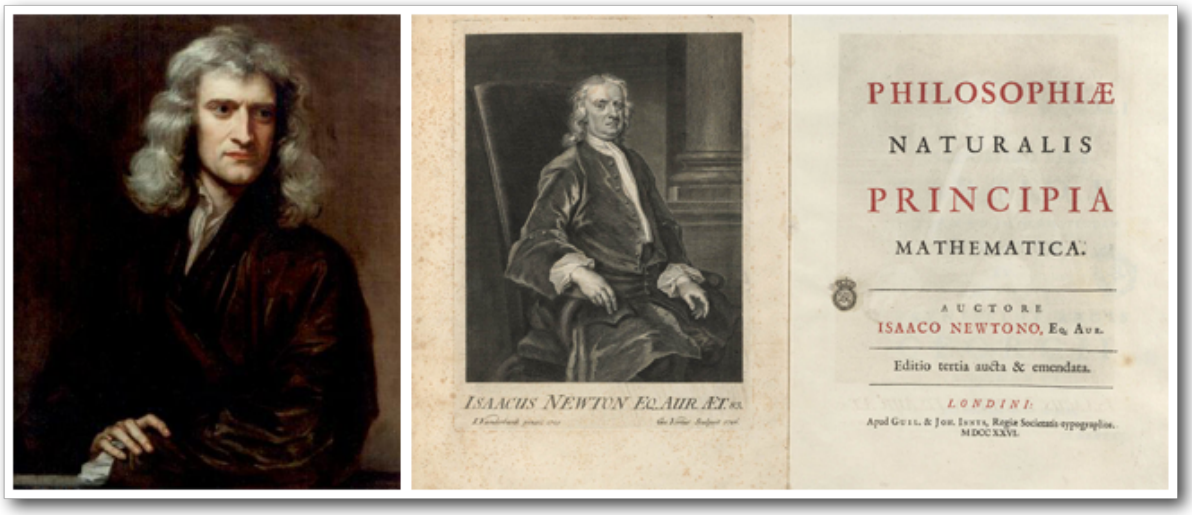
2 Klasična mehanika

U ovom poglavlju prisjetit ćemo se klasične mehanike, tj. Newtonove fizike. Kao što možda već znamo, riječ je o području fizike koje proučava gibanje tijela pod djelovanjem sila. Klasična mehanika predstavlja temelj svakog fizikalnog pogona u računalnim simulacijama i igrama.

Osnove moderne mehanike postavio je Issac Newton u 17. stoljeću, preciznije 1687. godine u Londonu kada je objavio svoje najveće znanstveno djelo *Philosophiae naturalis principia mathematica*. Budući da se u ono vrijeme filozofijom prirode nazivalo ono što se danas zove teorijskom fizikom, naslov glasovitog Newtonova djela mogao bi se prevesti kao *Matematička načela teorijske fizike*. Ovdje je po prvi puta uveden diferencijalni račun, objavljen je zakon gravitacije, te su postavljeni temelji klasične mehanike - Newtonovi aksiomi.

Međutim, na prijelazu iz 19. u 20. stoljeće rađaju se nove ideje, temelji nečeg što će se kasnije nazvati kvantna mehanika, i teorija relativnosti. Zahvaljujući napredovanju znanja i tehnologije danas znamo da je klasična Newtonova mehanika samo aproksimacija veće istine o materiji, prostoru i vremenu koja nam još nije u potpunosti poznata. Suvremena kvantna fizika nas uči da Newtonovi aksiomi nisu primjenjivi na sve sastavne elemente svemira. Ponašavnje mikroskopskih objekata ne pokorava se zakonima klasične mehanike, već zadovoljavaju neke druge jednadžbe čija rješenja ne daju točan položaj

objekta već samo njihovu vjerojatnost. Također u analizi gibanja masivnih objekata kao što su zvijezde i galaksije treba računati s učincima zakrivljenosti prostor-vremena, kao što je pokazano u Einsteinovoj općoj teoriji relativnosti. No, iako je klasična mehanika samo aproksimativna, ona je u većini slučaja dovoljna za opisivanje svakodnevih problema gibanja. Zakoni klasične fizike bit će više nego dovoljni za modeliranje realističnih računalnih simulacija gibanja, pa je stoga korisno još jednom prisjetiti se temelja klasične mehanike - Newtonovih aksioma gibanja [1, 2, 3].



Slika 2.1: Isaac Newton i njegovo glasovito djelo *Philosophiae naturalis principia mathematica*

2.1 Newtonovi aksiomi

Isaac Newton je uveo silu kao uzrok promjene stanja gibanja čestice i cijelu mehaniku je sažeo u tri aksioma koji govore što se događa sa česticom kada na nju djeluju ili ne djeluju sile.

2.1.1 Aksiom tromosti

Prvi Newtonov aksiom gibanja temelji se na eksperimentalnom zapažanju Galilea Galileija. U ono vrijeme bilo je uvriježeno naučavanje da je potrebna vanjska sila na objekt da bi se on kretao. Galileo je promatrajući loptu koja se kotrlja na glatkoj kosoj podlozi došao do zaključka da je potrebna vanjska sila da bi se objekt ubrzao, ali i da se u nedostatku vanjske sile objekt nastavlja gibati jednoliko. Newton formalizira njegova zapažanja kao prvi aksiom gibanja - aksiom tromosti ili inercije [3, 4]

Svako tijelo ostaje u stanju mirovanja ili jednolikog gibanja po pravcu, sve dok ga vanjske sile ne prisile da to stanje ne promijeni.

$$\vec{F} = 0 \Longrightarrow \vec{v} = const. \tag{2.1}$$

2.1.2 Temeljni aksiom gibanja

Drugi Newtonov aksiom gibanja temeljni je zakon klasične mehanike i glasi:

Akceleracija tijela posljedica je djelovanja sile na tijelo određene mase. Akceleracija tijela ima smjer sile i razmjerna je sili, a obrnuto razmjerna masi tijela.

$$\vec{a} = \frac{\vec{F}_r}{m} \tag{2.2}$$

Gornja formulacija često se navodi u literaturi kao jednostavna formulacija temeljnog zakona gibanja budući da se odnosi na svakodnevne slučajeve kada se tijelu ne mijenja masa tijekom promjene brzine, što je moguće samo za brzine puno manje od brzine svjetlosti. Općenitija formulacija temeljnog zakona gibanja bliža je Newtonovom izvornom tekstu i glasi:

Vremenska promjena količine gibanja ($\vec{p} = m\vec{v}$) čestice (tijela), jednaka je zbroju svih sila, $\vec{F}$, koje djeluju na česticu (tijelo).

$$\vec{F} = \frac{\mathrm{d}\vec{p}}{\mathrm{d}t} \tag{2.3}$$

Za slučaj da je masa tijela konstantna, lako se vidi iz definicije količine gibanja i iz pravila deriviranja da ova opća formulacija prelazi u prethodni jednostavniji oblik:

$$\vec{F} = \frac{\mathrm{d}\vec{p}}{\mathrm{d}t} = \frac{\mathrm{d}(m\vec{v})}{\mathrm{d}t} = m\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = m\vec{a} \tag{2.4}$$

Sila vodi na promjenu količine gibanja čestice. Ako je sila jednaka nuli, tada nema ni promjene količine gibanja, tj. količina gibanja je konstantna veličina. U tom slučaju kažemo da vrijedi zakon očuvanja količine gibanja [3, 4].

2.1.3 Aksiom djelovanja i protudjelovanja

Treći Newtonov aksiom gibanja opisuje međudjelovanje dvaju tijela. Ako na primjer dvije osobe sjede na stolicama s kotačima, te jedna osoba pokuša odgurnuti drugu osobu, obje stolice će se početi kretati u suprotnim smjerovima. Ovo ponašanje Newton je opisao kao treći zakon gibanja koji glasi [3, 4]

Ako čestica (tijelo) A djeluje na česticu (tijelo) B silom $\vec{F}_{AB}$, tada i čestica (tijelo) B djeluje na česticu (tijelo) A silom $\vec{F}_{BA}$, istog iznosa, a suprotnog smjera.

$$\vec{F}_{AB} = -\vec{F}_{BA} \tag{2.5}$$

2.2 Vrste sila u klasičnoj mehanici

U ovom poglavlju kroz primjere ćemo se upoznati s nekim osnovnim vrstama sila u klasičnoj mehanici: gravitacijskom, elastičnom, centripetalnom te silom trenja. Budući da je njima moguće opisati većinu važnijih fizikalnih ponašanja u računalnim igrama i simulacijama, one su temelj svakog modernog fizikalnog pogona. Vidjet ćemo i kako programski implementirati ove sile na primjeru vozila čije fizikalno ponašanje modeliramo u ovom diplomskom radu.

2.2.1 Gravitacijska sila

Vjerovatno je svima poznata slavna legenda o Newtonovom proučavanju pada jabuke sa stabla prilikom kojeg je došao do spoznaje da ista ona sila koja privlači jabuku tlu, održava Mjesec u njegovoj putanji oko Zemlje i planete u njihovim putanjama oko Sunca. Newton je svoje otkriće zabilježio u prije spomenutom dijelu *Principia*, a danas ga nazivamo općim zakonom gravitacije koji glasi [1, 3]

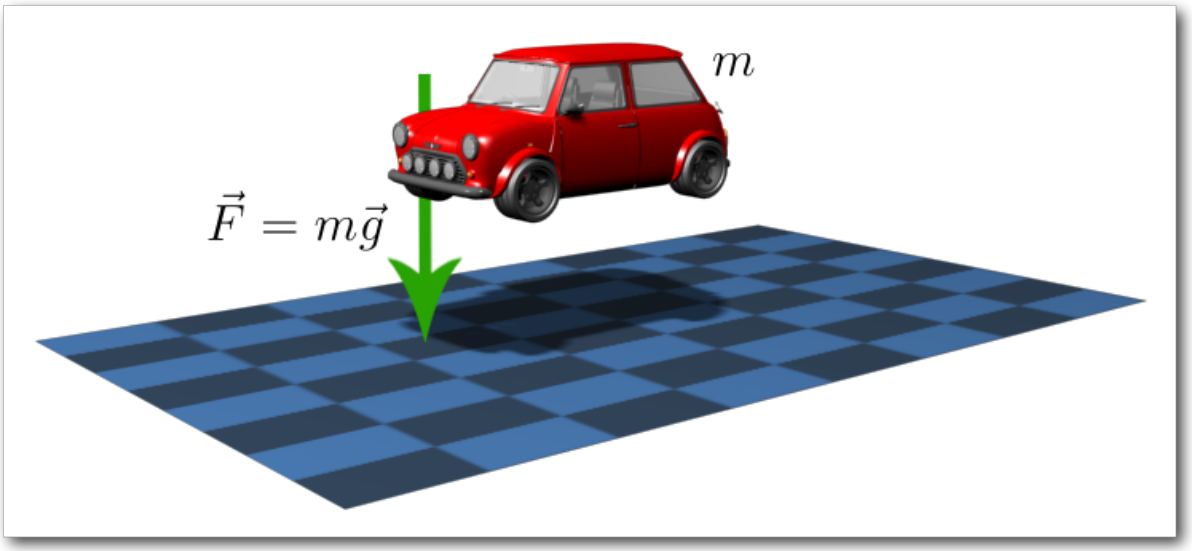
Između svaka dva tijela u Svemiru javlja se privlačna sila koja je proporcionalna umnošku njihovih masa, a obrnuto proporcionalna kvadratu njihove udaljenosti:

$$\vec{F} = -G\frac{m_1 m_2}{r^2}\hat{r} \tag{2.6}$$

gdje je $G = (66,73 \pm 0,03)10^{-12} m^3/(kgs^2)$ tzv. univerzalna gravitacijska konstantna čija je vrijednost eksperimentalno utvrđena.

Gravitacijska sila je razmjerno slaba, no sila teže na Zemlji je zamjetna zbog njene velike mase. Težina tijela mase m na površini zemlje bit će $T = G\frac{mM}{R^2}$, gdje je M masa Zemlje, a R njen polumjer. Veličina $g = \frac{GM}{R^2}$ naziva se gravitacijsko ubrzanje, a njen iznos na zemlji varira od $g = 9,781 m/s^2$ na ekvatoru do $g = 9,833 m/s^2$ na polovima [3, 5].

Sila gravitacije neizostavni je dio svakog modela fizikalnog ponašanja u računalnim simulacijama, te ju je vrlo jednostavno programski implementirati što ćemo vidjeti i u sljedećem primjeru.



Slika 2.2: Primjer sile teže na vozilo

Na slici 2.2 prikazan je grafički prikaz jednostavnog problema kojeg ćemo programski riješiti. Modelirat ćemo gibanje pod utjecajem sile teže koja djeluje na automobil neke zadane mase m. Budući da znamo da je slobodni pad zapravo jednoliko ubrzano gibanje gdje je akceleracija gravitacijska konstanta $g = 9,81 m/s^2$, iskoristit ćemo kinematički opis jednolikog ubrzanog gibanja:

$$\begin{aligned} a &= const \\ v &= v_0 + at \\ x &= x_0 + v_0 t + \frac{1}{2} a t^2 \end{aligned} \tag{2.7}$$

Ovakav matematički opis jednolikog ubrzanog gibanja je vrlo jednostavno preslikati u programski kôd, no pritom valja uvesti vektorske veličine kako bi modelirali gibanje u trodimenzionalnom prostoru. U nastavku je dan primjer implementacije jednolikog ubrzanog gibanja pod utjecajem sile gravitacije u programskom jeziku `C#` i `XNA` radnom okviru s kojim ćemo se kasnije bolje upoznati.

```
// konstanta gravitacije
const Vector3 GRAVITY = new Vector3(0, 0, -9.81f);
```

```
// trenutna brzina gibanja tijela
Vector3 velocity = new Vector3();
// polozaj tijela
Vector3 position = new Vector3();
// akceleracija jednolikog ubrzanog gibanja
Vector3 acceleration = GRAVITY;
// trenutak pocetka gibanja
float startTimestamp = 0.0f;
// vremenska razlika
float deltaT = 0.0f;
...
public static void Update(GameTime gameTime)
{
  ...
  // vremenska razlika od trenutka pocetka gibanja
  deltaT = startTimestamp - (float)gameTime.ElapsedGameTime.
     TotalSeconds
  position += velocity * deltaT + 0.5f * acceleration * deltaT *
     deltaT;
  velocity += acceleration * deltaT;
  ...
}
```

Programski odsječak 1: Implementacija slobodnog pada.

2.2.2 Sila trenja

Vrlo često se susrećemo sa situacijama u kojima je tijelo prisiljeno se gibati duž neke određene površine. U takvim slučajevima tijelo je podvrgnuto određenim uvjetima gibanja. Uslijed djelovanja sile teže, čestica će djelovati silom na podlogu kojom se giba, pa će u skladu s trećim Newtonovim aksiomom i podloga djelovati na tijelo silom iste jakosti, ali suprotnog smjera, $\vec{N}$. Osim sile reakcije na podlogu, postoji još jedna sila koja je posljedica postavljenih uvjeta gibanja - trenje. Uslijed privlačnog djelovanja tijela s molekulama podloge po kojoj se giba, javljaju se sile koje se odupiru gibanju i nastoje zaustaviti tijelo. Sila trenja, $\vec{F}_r$, suprotna je smjeru gibanja tijela ($\vec{e}_v$), te se opisuje preko koeficijenta trenja μ koji se eksperimentalno određuje [3]

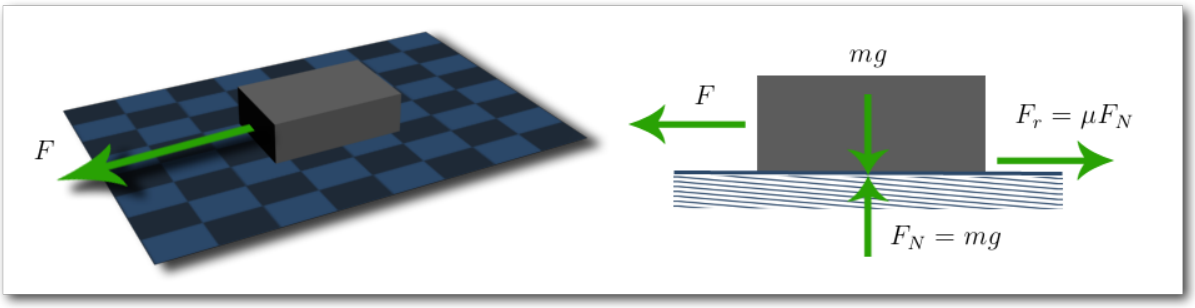
$$\vec{F}_r = -\mu N \vec{e}_v \tag{2.8}$$

Ukoliko se tijelo pokreće iz stanja mirovanja, spomenuti koeficijent trenja se preciznije naziva statički koeficijent trenja, μ_s, a ako je tijelo već u stanju gibanja on se naziva dinamičkim koeficijentom gibanja, μ_d, te općenito vrijedi $\mu_s > \mu_d$. S ovim efektom smo se sigurno nebrojeno puta sreli prilikom guranja tereta po ravnoj podlozi. Primijetili smo da jednom kada se teret pokrene postaje nam lakše gurati ga po podlozi. Prilikom programskog modeliranja sile trenja važno je uzeti u obzir odgovarajuće koeficijente trenja, pa je u sljedećoj tablici dan primjer koeficijenata trenja za neke materijale [1].

Tablica 2.1: Koeficijenti trenja za neke materijale

Materijali	μ_s	μ_d
željezo - željezo	$0,7 - 0,74$	$0,57 - 0,6$
aluminij - željezo	$0,61$	$0,47$
bakar - željezo	$0,53$	$0,36$
teflon - teflon	$0,04$	$0,04$
staklo - staklo	$0,94$	$0,4$
drvo - drvo	$0,25 - 0,5$	$0,2 - 0,3$
guma - beton	$1,0$	$0,8$
guma - vlažan beton	$0,7$	$0,5$
led - led	$0,1$	$0,03$

Trenje je iznimno važan fenomen i za računalne igre i simulacije, te je obično sastavni dio svakog fizikalnog pogona. Jasno je da na primjer igra koja simulira neko klizanje po površini mora uključiti učinke trenja. Trenje između ping-pong loptice i reketa uzrokuje spin i definira putanju kretanja loptice. Trenje između gume i ceste je ono što pokreće automobil i drži ga na cesti u zavojima. U nastavku ćemo za jednostavni primjer implementirati učinak sile trenja. Promatrat ćemo kutiju koja se giba po ravnoj površini kako je prikazano na slici 2.3. Sila trenja se pritom opire gibanju i nastoji zaustaviti kutiju [1].



Slika 2.3: Primjer djelovanja sile trenja

Budući da trenje usporava kutiju u gibanju, najjednostavnije je implementirati smanjenje brzine kretanja kutije koje ovisi o njenoj masi i koeficijentu trenja između nje i podloge:

```
speed = speed - mu * m * g * deltaT;
```

U nastavku se opet nalazi kompletnija programska implementacija programskim jezikom `C#` i `XNA` radnim okvirom.

```
// vektor smjera kretanja kutije
Vector3 direction = new Vector3();
// brzina kretanja kutije
float speed = 0.0f;
// polozaj tijela
Vector3 position = new Vector();
// koeficijent trenja
float mu = 0.8f;
// masa kutije
float m = 500.0f;
// gravitacijska konstanta
float g = 9.81f;
// trenutak pocetka gibanja
float startTimestamp = 0.0f;
// vremenska razlika
float deltaT = 0.0f;
...
public static void Update(GameTime gameTime)
{
  ...
  // vremenska razlika od trenutka pocetka gibanja
```

```
    deltaT = startTimestamp - (float)gameTime.ElapsedGameTime.
        TotalSeconds
    speed = speed - mu * m * g * deltaT;
    position += direction * speed;
    ...
}
```

Programski odsječak 2: Implementacija sile trenja.

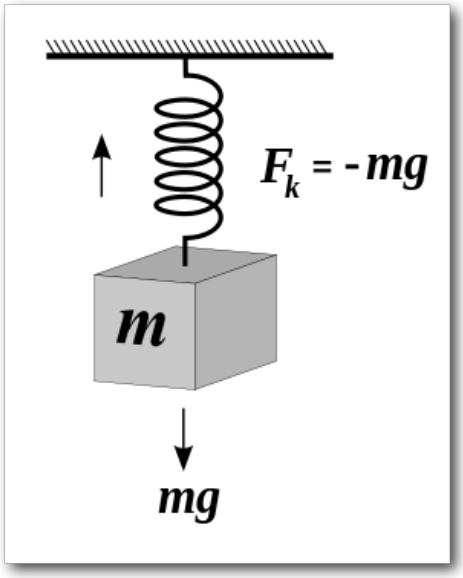
2.2.3 Elastična sila

Sila gravitacije i trenja neizostavni su fizikalni fenomeni svakog računalnog fizikalnog pogona. Manje uobičajena je elastična sila opruga kojom ćemo se baviti u ovom poglavlju, no i ona pronalazi zanimljive primjene koje često izgledaju vrlo atraktivno. Elastične sile opruga susrest ćemo i na nekoliko mjesta u praktičnom dijelu diplomskog rada. Jedno od zanimljivijih primjena bit će prateća kamera koja je oprugom vezana za objekt kojeg snima. U ovom poglavlju pokazat ćemo kako implementirati ovakvu kameru i objasniti koje su njene prednosti, no prije svega upoznat ćemo se s Hookeovim zakonom i fizikalnim osnovama elastičnih opruga.

Robert Hooke, britanski fizičar i matematičar, 1676. godine formulirao je osnovni zakon teorije elastičnosti kojeg danas nazivamo njemu u čast Hookeovim zakonom. Zakon kaže da je deformacija tijela proporcionalna primijenjenoj sili pod uvjetom da se ne pređe granica elastičnosti tijela. Kada se vanjska sila ukloni tijelo će se vratiti u svoj prvobitni oblik. Ako se primjerice tijelo na elastičnoj opruzi pomakne iz ravnotežnog položaja, djelovat će povratna sila, tj. elastična sila opruge, koja će nastojati vratiti tijelo u ravnotežni položaj [7]. Iznos te sile proporcionalan je pomaku tijela iz ravnotežnog položaja. Dakle, ako je Δx pomak iz ravnotežnog položaja, iznos povratne sile bit će:

$$F_k = -k\Delta x \tag{2.9}$$

gdje je k konstanta opruge koja ovisi o njenim dimenzijama, obliku i materijalu od kojega je izrađena.



Slika 2.4: Opruga u ravnotežnom položaju

Vrlo često dodajemo i silu prigušenja:

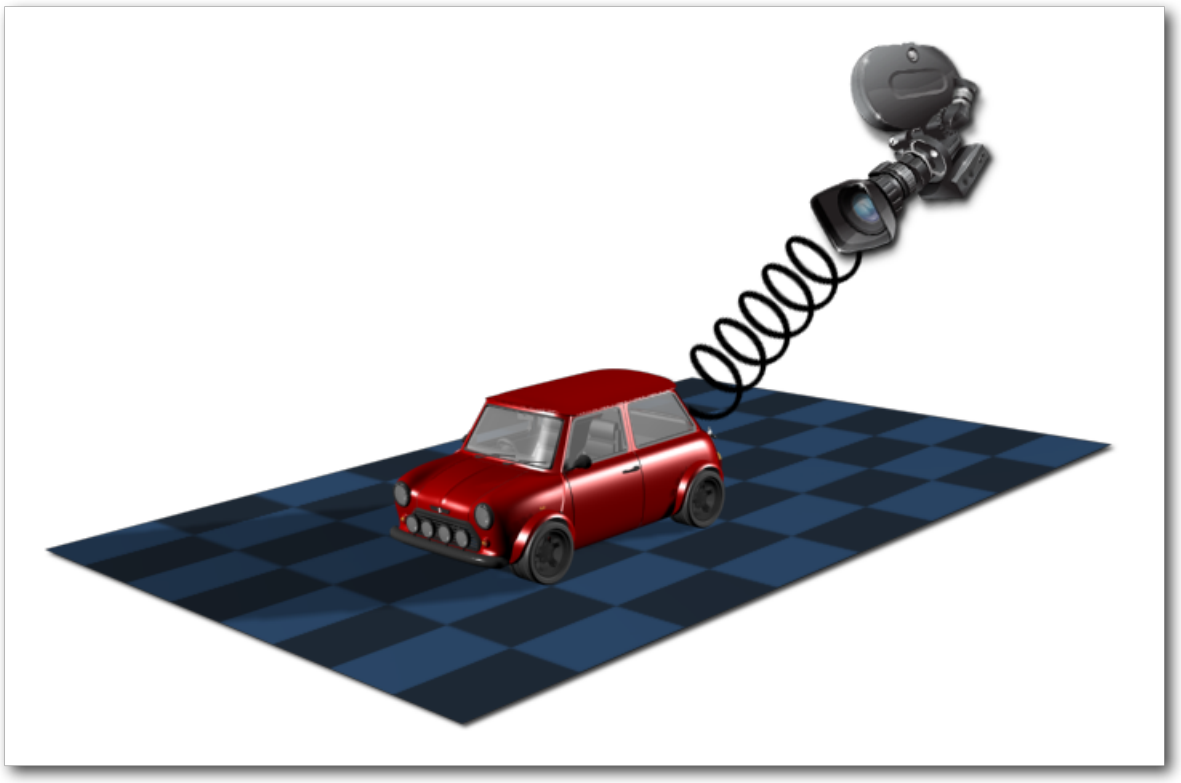
$$\vec{F}_b = -b\vec{v} \tag{2.10}$$

gdje je b konstantna prigušenja, a $\vec{v}$ brzina tereta (u smjeru opruge). Tada je ukupna sila na tijelo jednaka:

$$\vec{F}_u = mg - k_s\Delta\vec{x} - b\vec{v} \tag{2.11}$$

Hookeov zakon elastičnosti široko je primjenjiv u računalnim simulacijama, pa ćemo tako često susresti fizikalne modele opruga na najrazličitijim mjestima. Primjerice česta je primjena sustava opruga i masa u simulaciji tkanina, gumenih predmeta, kose i sl.. U nastavku ćemo se pozabaviti s jednom praktičnom i vrlo atraktivnom primjenom opruge koja je korištena u praktičnom dijelu diplomskog rada. Riječ je o već prije spomenutoj pratećoj kameri koja je oprugom vezana na objekt kojeg snima (slika 2.5). Ovakva kamera često se primjenjuje u simulacijama upravljanja vozilima, gdje je potrebno pratiti položaj vozila prilikom njihovog kretanja. Želimo postići da povećanjem brzine vozila dobivamo širi kut pogleda, budući da je pri većim brzinama teže pratiti stazu. Pri manjim brzinama želimo da kamera bude bliže vozilu, tj. da je kut pogleda ograničen na samo vozilo. Očito je da je primjena opruge u ovakvom primjeru najpraktičnije rješenje: prilikom većih brzina, tj. jačih sila, opruga će se jače rastegnuti, što znači da će kamera biti dalje

od objekta.



Slika 2.5: Prateća kamera povezana oprugom na objekt kojeg snima

Na samom početku kreirat ćemo klasu `ChaseCamera` koja će implementirati prateću kameru. Budući da kamera u svakom trenutku mora pratiti definirani objekt dodat ćemo atribute koji opisuju položaj i orijentaciju objekta (`ChasePosition`, `ChaseDirection`, `Up`). Također definirat ćemo i atribute koji opisuju ravnotežni i trenutni položaj kamere, te trenutnu brzinu kretanja kamere (`desiredPositionOffset`, `desiredPosition`, `dosition`, `velocity`).

```
Vector3 chasePosition;
Vector3 chaseDirection;
Vector3 up = Vector3.Up;
Vector3 desiredPositionOffset = new Vector3(0, 0.002f, 0.002f);
Vector3 desiredPosition;
Vector3 position;
```

Da bi modelirali fizikalno ponašanje opruge, potrebno je definirati koeficijent elastičnosti (tvrdoća opruge), prigušenje opruge, te masu kamere (`stiffness`, `damping`, `mass`).

```
float stiffness = 1800.0f;
private float damping = 600.0f;
private float mass = 50.0f;
```

Položaj kamere računat ćemo u metodi `Update()` gdje ćemo programski opisati fizikalno ponašanje opruge koristeći Hookeov zakon. Najprije računamo pomak iz ravnotežnog položaja Δx:

```
Vector3 stretch = position - desiredPosition;
```

Ukupnu silu na kameru ćemo zatim izračunati pomoću formule 2.11, ali ćemo pritom zanemarit silu teže:

```
Vector3 force = -stiffness * stretch - damping * velocity;
```

Sada kada nam je poznata ukupna sila na kameru te njena masa, prisjetit ćemo se drugog Newtonovog zakona i formule 2.2. Izračunat ćemo akceleraciju kao omjer sile i mase na koju ta sila djeluje. Iz akceleracije i vremenskog pomaka dalje je jednostavno izračunati novi položaj kamere.

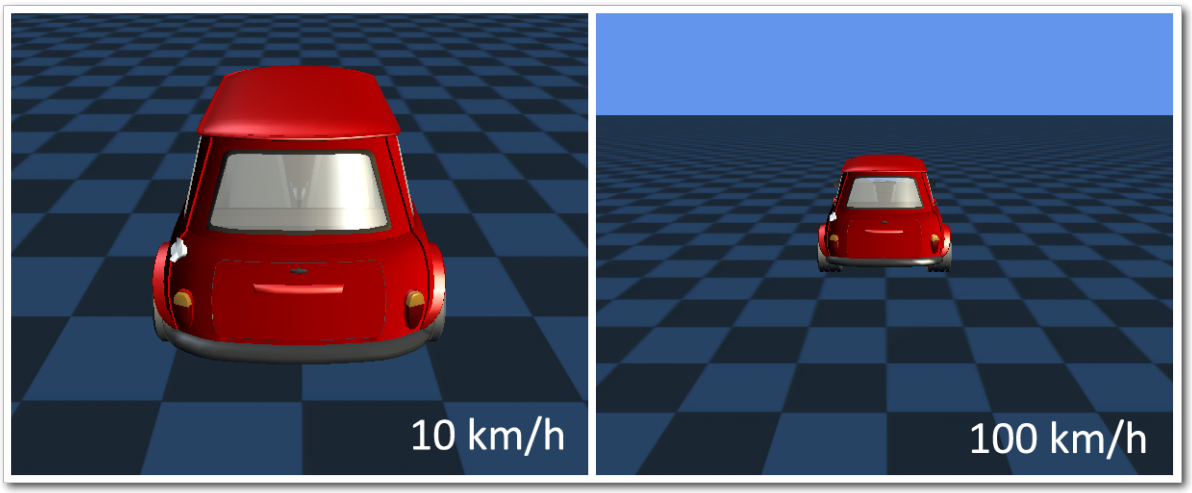
```
Vector3 acceleration = force / mass;
velocity += acceleration * elapsed;
position += velocity * elapsed;
```

```
public class ChaseCamera
{
    ....

    public void Update(GameTime gameTime)
    {
        if (gameTime == null)
            throw new ArgumentNullException("gameTime");

  // vremenski pomak
        float elapsed = (float)gameTime.ElapsedGameTime.
            TotalSeconds;

  // pomak iz ravnoteznog stanja
```

```
        Vector3 stretch = position - desiredPosition;
    // povratna sila
        Vector3 force = -stiffness * stretch - damping * velocity;

    // akceleracija (II. Newtonov aksiom)
        Vector3 acceleration = force / mass;
        velocity += acceleration * elapsed;

    // novi polozaj kamere
        position += velocity * elapsed;
    }

    ....
}
```

Programski odsječak 3: Implementacija fizikalnog ponašanja opruge.

Na sljedećoj slici prikazan je rezultat implementacije oprugom vezane kamere.



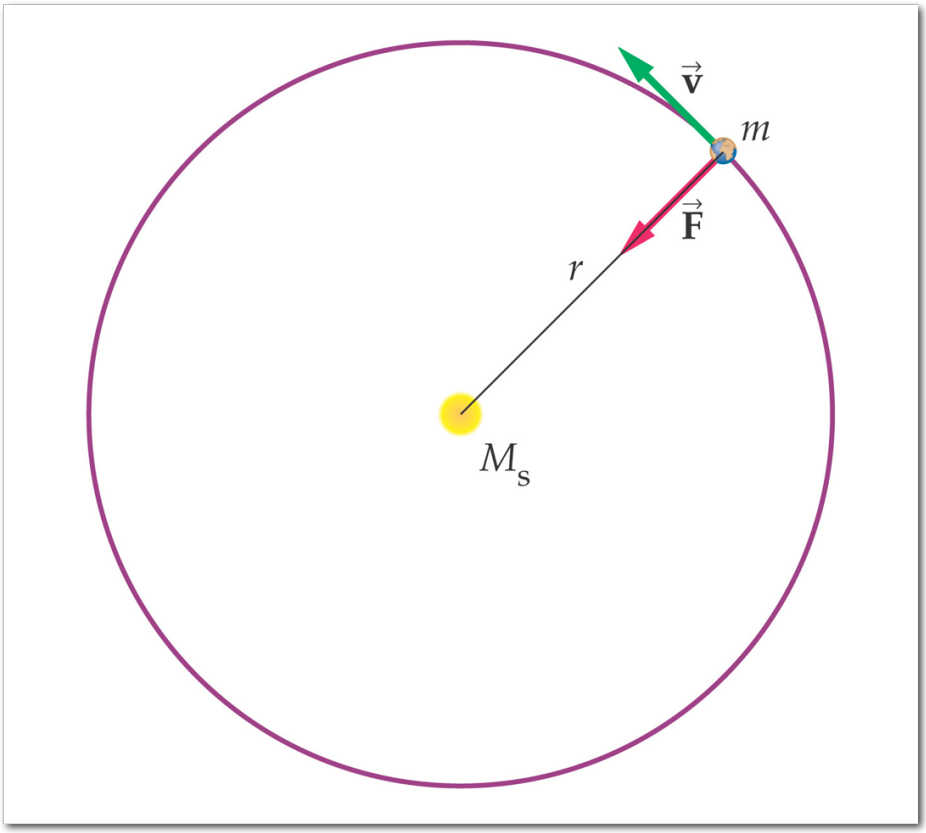
Slika 2.6: Rezultat implementacije oprugom vezane kamere

2.2.4 Centripetalna sila

Na kraju ovog poglavlja još ćemo spomenuti centripetalnu silu koja se javlja prilikom kružnih gibanja te nastoji zadržati tijelo na kružnoj putanji. Drugim riječima, centripetalna sila je ukupna sila koja je potrebna da bi se tijelo mase m gibalo brzinom v po kružnoj putanji polumjera r:

$$F_c = \frac{mv^2}{r} \tag{2.12}$$

Centripetalna sila uvijek je usmjerena prema središtu kružnice i mijenja smjer vektora brzine $\vec{v}$. Važno je razumjeti da centripetalna sila nije nova sila u prirodi, već ona označava rezultantnu silu koja je usmjerena prema središtu kružne putanje. Ona se može sastojati od sile napetosti, sile trenja, gravitacijske sile i dr.. Primjerice prilikom gibanja automobila po kružnom zavoju centripetalnu silu daje trenje između kotača i ceste. Pri gibanju satelita oko Zemlje, centripetalnu silu daje sila gravitacije, a pri gibanju elektrona oko jezgre, centripetalnu silu daje Coulombova sila između elektrona i jezgre [8].



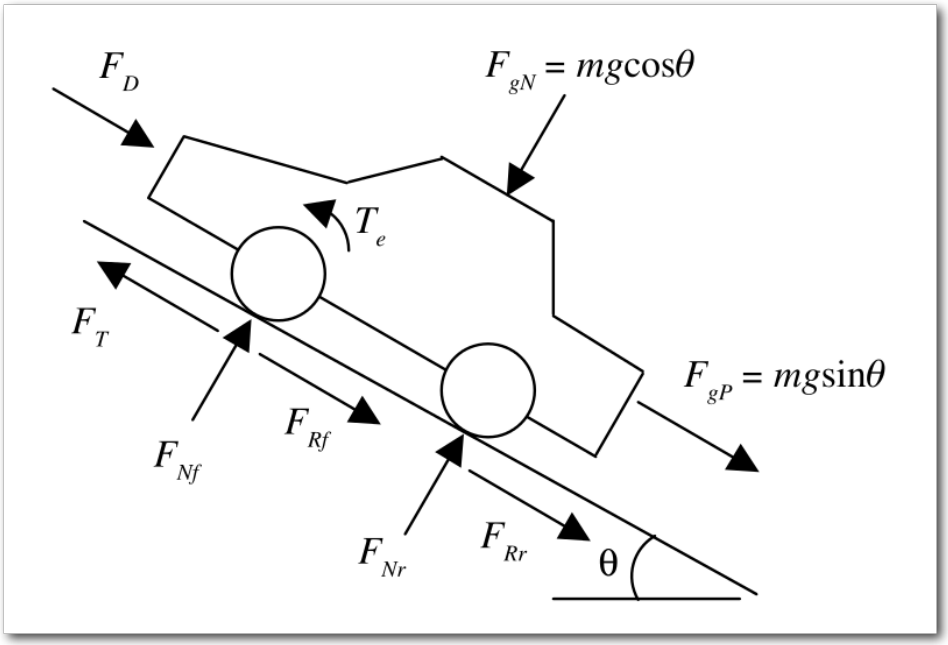
Slika 2.7: Sila gravitacije kao centripetalna sila

3 Fizikalni model vozila

U ovom poglavlju teorijski ćemo opisati fizikalni model automobila koristeći neke teorijske osnove klasične mehanike i kinematike iz prethodnog poglavlja, te uz malo novih znanja o prijenosu snage s motora na kotače automobila. Krenut ćemo od osnova jednostavne pravocrtne vožnje, te istražiti zanimljive teme kao što su sile koje djeluju na automobil prilikom pravocrtnog gibanja, zakretni moment motora, te izmjena stupnjeva prijenosa snage. Također proučit ćemo sile koje se opiru kretanju automobila, a pritom se i susresti s nekim osnovama aerodinamike u automobilskoj industriji. Na samom kraju poglavlja istražit ćemo kako modelirati brzinu i akceleraciju automobila, ponašanje automobila u zavojima, te proklizavanje kotača.

3.1 Dijagram sila

Na slici 3.1 nalazi se shematski prikaz sila koje djeluju na automobil koji se pravocrtno giba po kosini stupnja nagnutosti θ. Automobil je pod utjecajem sile teže, sile trenja klizanja i kotrljanja, sile otpora zraka, te okretnog momenta koji proizvodi motor i prenosi ga na kotače. Proći ćemo kroz sve ove sile kako bi bolje razumjeli prikazani dijagram sila.



Slika 3.1: Dijagram sila na automobil u stanju pravocrtnog gibanja

Sila teže, F_g, privlači automobil površini Zemlje, a na prikazanom dijagramu rastav-

ljena je na dvije komponente: $F_{gN} = mg\cos\theta$ koja je okomita na kosinu, te $F_{gP} = mg\sin\theta$ koja je paralelna s kosinom. Okomita komponenta uravnotežena je protusilom F_N kojom podloga djeluje na kotače automobila (Newtonov aksiom djelovanja i protudjelovanja). F_N jednaka je sumi sila kojom podloga djeluje na svaki pojedini kotač automobila.

$$F_N = F_{Nf} + F_{Nr} = mg\cos\theta \tag{3.1}$$

Motor proizvodi okretni moment koji se prenosi na kotače i rotira ih. Trenje između guma i podloge opire se rotaciji što rezultira silom F_T suprotnom smjeru rotacije kotača. Sila F_T jednaka je okretnom momentu prenesenom na kotače, T_w, podijeljenom s polumjerom kotača, r_w. Vidjet ćemo kasnije da okretni moment koji proizvodi motor u pravilu nije jednak okretnom momentu prenesenom na kotače [10, 11].

$$F_T = \frac{T_w}{r_w} \tag{3.2}$$

Gibanju automobila opire se aerodinamička sila otpora koju možemo modelirati funkcijom gustoće zraka, ρ, najveće površine presjeka automobila okomitog na pravac gibanja, A, kvadrata brzine, v, te koeficijenta otpora zraka, C_D.

$$F_D = \frac{1}{2}C_D\rho v^2 A \tag{3.3}$$

Koeficijent otpora zraka, C_D, je bezdimenzijska veličina koja ovisi o obliku i veličini tijela, položaju u struji zraka i glatkoći površine. Kasnije u ovom poglavlju više ćemo se baviti aerodinamičkom silom otpora [1, 9].

Na kraju nam je još preostala sila trenja kotrljanja koja se javlja kada je ostvaren kontakt u jednoj točki, odnosno liniji. Ukupna sila trenja kotrljanja jednaka je umnošku sile F_N i koeficijenta trenja kotrljanja, μ_r.

$$F_R = \mu_r F_N = \mu_r mg\cos\theta \tag{3.4}$$

Iznos ukupne sila koja djeluje na automobil, F_{total}, jednaka je sumi iznosa prethodno opisanih sila, ali pritom moramo paziti na njihov smjer.

$$F_{total} = \frac{T_w}{r_w} - \mu_r mg\sin\theta - \frac{1}{2}C_D\rho v^2 A \tag{3.5}$$

Iz ukupne sile koja djeluje na automobil lako ćemo izračunati njegovu akceleraciju.

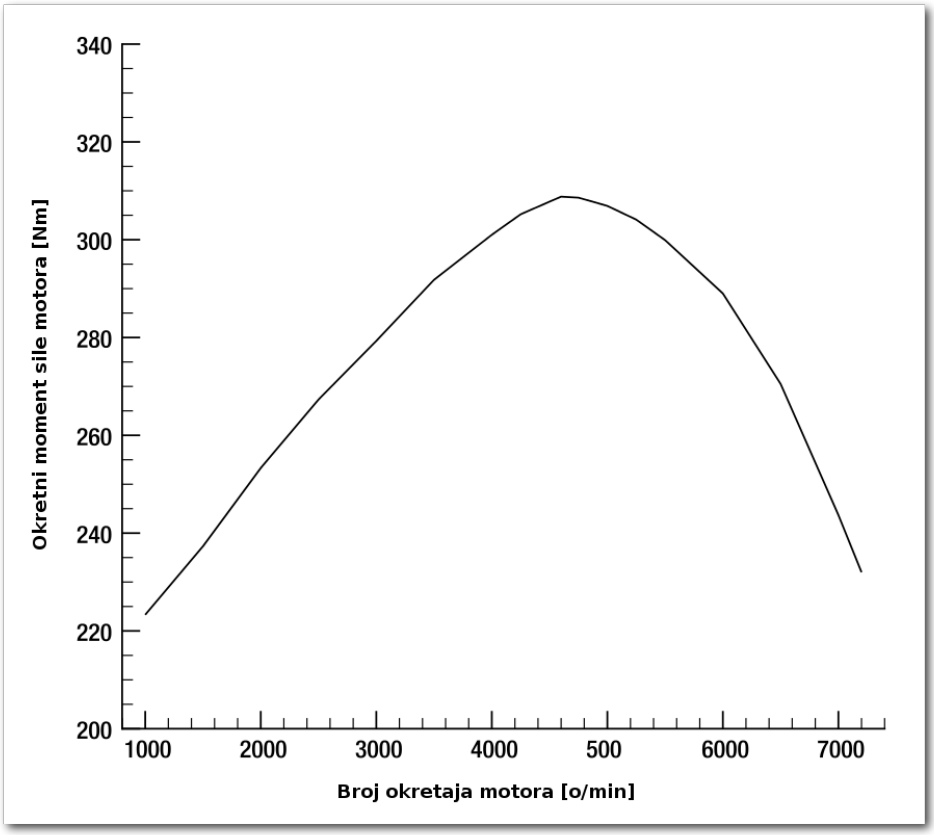
$$a = \frac{F_{total}}{m} = \frac{T_w}{r_w m} - \mu_r g \sin\theta - \frac{1}{2}\frac{C_D \rho v^2 A}{m} \tag{3.6}$$

3.2 Snaga i okretni moment motora

Kao što smo već ranije spomenuli motor pokreće automobil proizvodeći okretni moment koji rotira kotače. Okretni moment automobila zapravo je funkcija broja okretaja motora.

$$T_e = T_e(\Omega_e) \tag{3.7}$$

Zanimljivo je pogledati krivulju funkcije okretnog momenta u ovisnosti o broju okretaja. Tipična karakteristika okretnog momenta sile motora je da se ne povećava uvijek povećanjem broja okretaja motora. Vidimo na slici 3.2 gdje je prikazan primjer krivulje okretnog momenta da se on povećava povećanjem broja okretaja i dostiže svoj maksimum od 309 Nm na 4600 o/min kada počinje opadati [10].

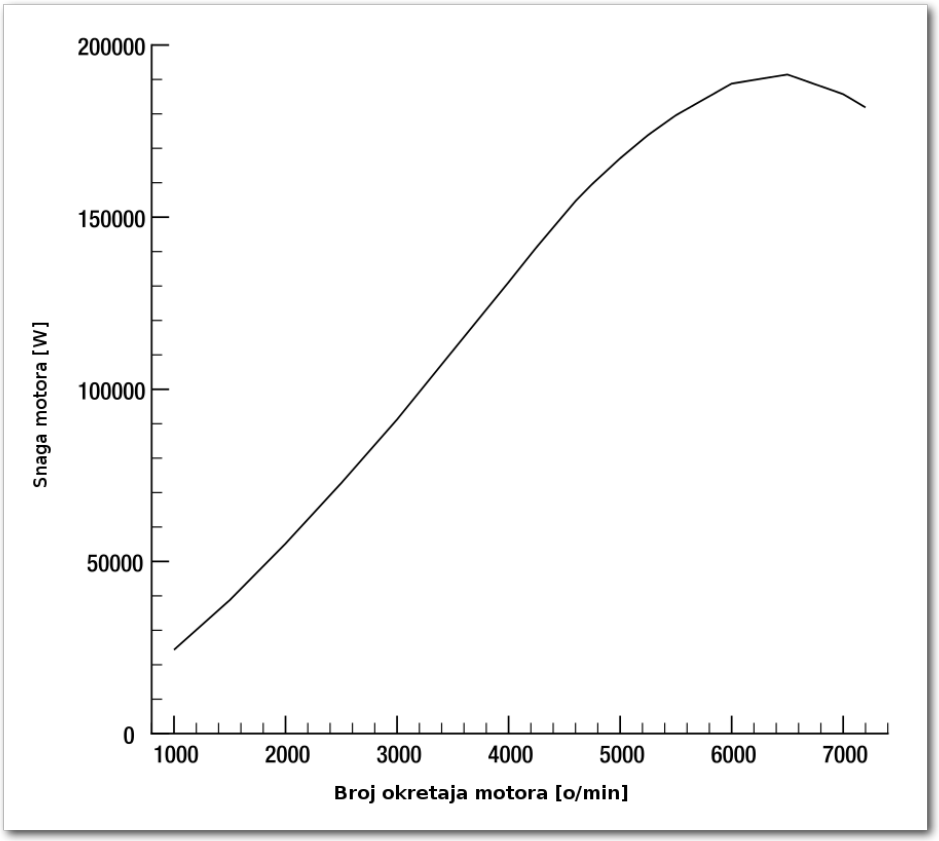


Slika 3.2: Primjer krivulje okretnog momenta motora

Ako želimo izračunati trenutnu snagu motora računat ćemo umnožak okretnog mo-

menta motora, T_e, i kutne brzine motora, ω_e.

$$P_e = T_e\omega_e = T_e\frac{2\pi\Omega_e}{60} \tag{3.8}$$



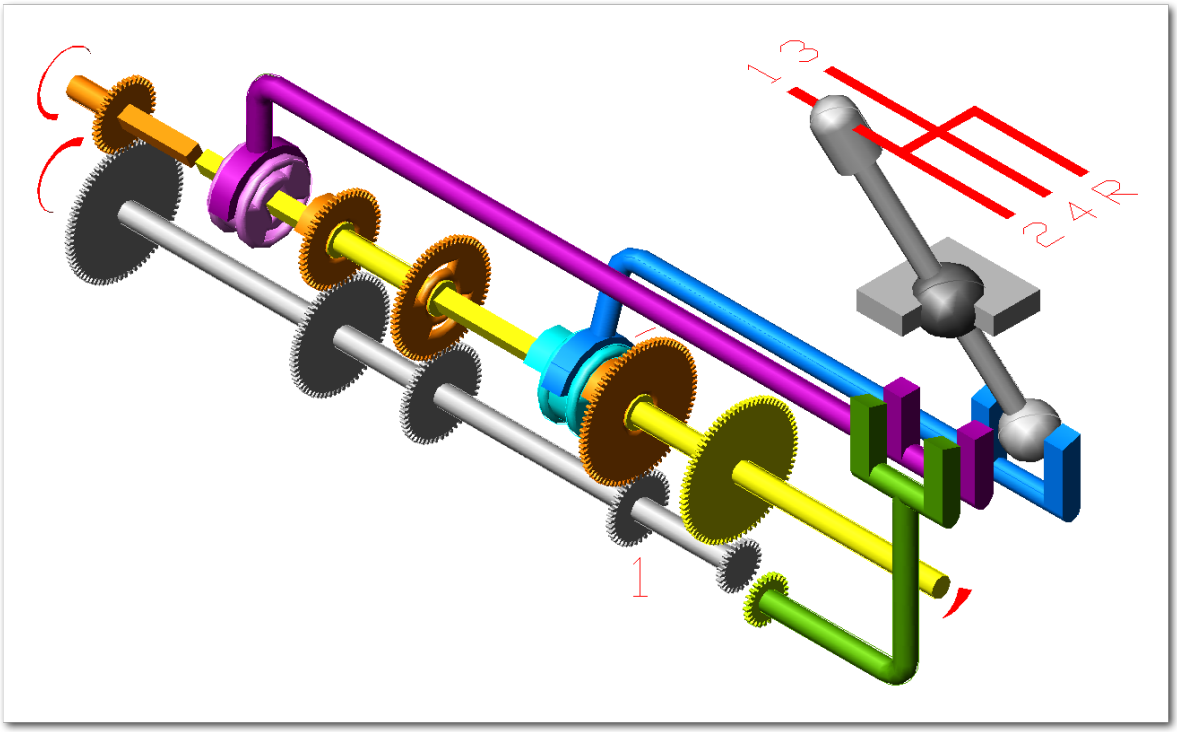
Slika 3.3: Primjer krivulje funkcije motora

3.3 Stupnjevi prijenosa i okretni moment kotača

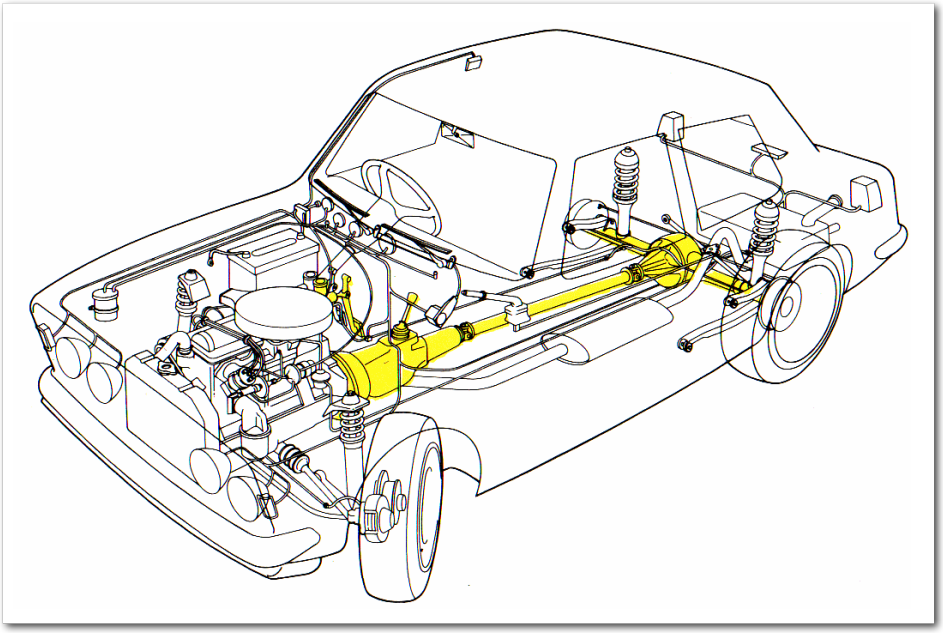
Okretni moment sile motora koji se prenosi na kotače automobila određuje njegovu brzinu. No okretni moment koji se prenosi na kotače u pravilu nije jednak okretnom momentu motora, jer prolazi kroz sustav za prijenos snage. Snagu motora treba prilagođavati uvjetima vožnje i zato u prijenosu snage postoji mjenjač. Što je veći broj okretaja motora u odnosu na broj okretaja kotača, to je veća snaga na raspolaganju za pogon kotača, ali se u jednakom omjeru smanjuje brzina vožnje. Mijenjanjem stupnjeva prijenosa mjenjača mijenjamo prijenosne odnose između motora i kotača [12].

Uključivanjem mjenjača u neki od stupnjeva prijenosa zahvaćaju se zupčanici koji između motora i kotača uspostavljaju najpovoljniji omjer okretaja. Većina današnjih

automobila ima mjenjače s pet ili šest stupnjeva prijenosa za vožnju naprijed i jednim za vožnju natrag. U praznom hodu zupčanici u mjenjaču nisu zahvaćeni, a tada motor može raditi a da se automobil ne pomiče (slika 3.4) [12].

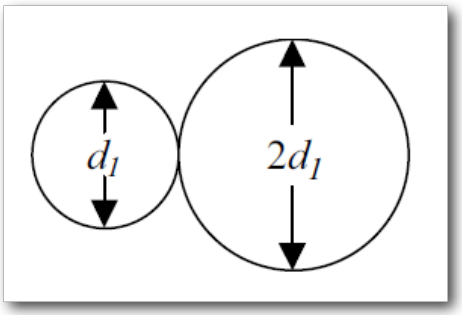


Slika 3.4: Shema sustava za prijenos stupnja brzine



Slika 3.5: Sustav za prijenos snage

Promjenom stupnja prijenosa mijenja se kutna brzina i okretni moment koji se prenosi s motora na kotače. Recimo da imamo samo dva zupčanika (slika 3.6) gdje drugi ima dvostruko veći promjer od prvog, tada će svakim okretom drugog zupčanika prvi napraviti dva. To znači da će drugi zupčanik imati dvostruko manju kutnu brzinu od prvog, ali dvostruko veći okretni moment sile [10].



Slika 3.6: Sustav prijenosa okretnog momenta sile s dva zupčanika

Da bi izračunali ukupni okretni moment sile koji se prenosi s motora na kotače, T_w, potreban nam je tzv. omjer stupnja prijenosa koji je jednak omjeru promjera pripadajućih zupčanika. Završnim omjerom stupnja prijenosa nazivamo omjer promjera zupčanika u najvećem stupnju prijenosa. Okretni moment sile T_w bit će jednak umnošku okretnog momenta sile motora, omjera trenutnog stupnja prijenosa, te omjera završnog stupnja prijenosa.

$$T_w = T_e g_k G \tag{3.9}$$

Prethodni izraz za moment sile koji se prenosi na kotače uvrstit ćemo u (3.6) i dobiti formulu za izračun akceleracije u zadanom stupnju prijenosa.

$$a = \frac{T_e g_k G}{r_w m} - \mu_r g \sin\theta - \frac{1}{2}\frac{C_D \rho v^2 A}{m} \tag{3.10}$$

Ako na trenutak zanemarimo proklizavanje kotača, translacijsku brzinu automobila možemo poistovjetiti s brzinom rotacije kotača. Kutna brzina kotača mijenja se u ovisnosti o broju okretaja motora, Ω_e:

$$\omega_w = \frac{2\pi\Omega_e}{60 g_k G} \tag{3.11}$$

pa će brzina rotacije kotača, tj. translacijska brzina automobila biti jednaka

$$v = r_w \omega_w = \frac{r_w 2\pi \Omega_e}{60 g_k G} \tag{3.12}$$

gdje je r_w polumjer kotača.

Iz prethodno izvedenih izraza vidimo da se za dani broj okretaja motora povećanjem omjera stupnja prijenosa povećava i akceleracija, ali smanjuje translacijska brzina automobila. U tablici 3.1 dan je primjer omjera stupnjeva prijenosa sportskog automobila [10].

Tablica 3.1: Omjeri stupnjeva prijenosa sportskog automobila (Porsche Boxster S, 2004.)

Stupanj prijenosa	Omjer stupnja prijenosa
Prvi	$3,82$
Drugi	$2,20$
Treći	$1,52$
Četvrti	$1,22$
Peti	$1,02$
Šesti	$0,84$

Ako nismo vozači, možemo se zapitati zašto ne upravljamo automobilom samo u prvoj brzini, budući da nam ona daje najveće ubrzanje. Moramo se prisjetiti da je brzina rotacije kotača funkcija omjera stupnja prijenosa i broja okretaja motora, te da postoji ograničenje u broju okretaja motora. U slučaju prelaska graničnog iznosa (eng. *redline*) broja okretaja, motor trpi veliku štetu. Uz pomoć podataka iz tablice 3.1 i izraza (3.12) moguće je izračunati maksimalne brzine za pojedini stupanj prijenosa (tablica 3.2) [10].

Tablica 3.2: Maksimalne brzine u pojedinim stupnjevima prijenosa sportskog automobila (Porsche Boxster S, 2004.)

Stupanj prijenosa	Maksimalna brzina $[km/h]$
Prvi	$65,8$
Drugi	$114,3$
Treći	$165,4$
Četvrti	$206,0$
Peti	$246,4$
Šesti	$299,3$

3.4 Sile otpora

Važno je razumjeti da su vrijednosti maksimalnih brzina u tablici 3.2 samo teorijske vrijednosti. Prema specifikaciji proizvođača automobila one su podosta manje, jer su uzete u obzir sile otpora koje se opiru gibanju automobila, a od kojih najveći utjecaj imaju sila trenja i aerodinamička sila otpora. S trenjem smo se već susreli u poglavlju 2.2.2 Sila trenja, pa ćemo u ovom poglavlju reći nešto više o aerodinamičkoj sili otpora.

3.4.1 Aerodinamička sila otpora

Aerodinamička sila otpora djeluje suprotno od smjera kretanja nekog krutog tijela i ovisi o nizu sljedećih faktora:

- gustoća zraka kroz koji se tijelo giba
- oblik tijela, njegova veličina, položaj u struji zraka i glatkoća površine
- površina presjeka tijela okomita na pravac gibanja
- brzina gibanja tijela kroz zrak.

Jednadžbu za izračun aerodinamičke sile otpora već smo susreli u poglavlju 3.1 Dijagram sila:

$$F_D = \frac{1}{2} C_D \rho v^2 A \tag{3.13}$$

gdje ρ gustoća zraka, A površine presjeka tijela okomita na pravac gibanja, v brzina tijela, te C_D koeficijenta otpora zraka.

Koeficijent otpora zraka, C_D, standardna je mjera uspjeha aerodinamičnosti. Ovaj koeficijent uspoređuje silu otpora zraka sa silom koja bi bila potrebna da se zaustavi struja zraka ispred automobila. Što je veći koeficijent otpora zraka to je veća sila otpora zraka koju motor automobila mora svladati. Koeficijent otpora zraka je bezdimenzijska veličina koja ovisi o obliku i veličini tijela, položaju u struji zraka i glatkoći površine, a najčešće se određuje praktičnim istraživanjima u zračnim tunelima. U sljedećoj tablici dani su primjeri vrijednosti koeficijenta otpora zraka za neke vrste vozila [9, 10, 11].

Tablica 3.3: Vrijednosti koeficijenta otpora zraka za neke vrste vozila

Vrsta vozila	Koeficijent otpora zraka
Sportski automobil	$0,27 - 0,38$
Sedan	$0,34 - 0,5$
Kamion	$0,6 - 1,0$
Kamp prikolica	$0,6 - 1,2$
Motocikl	$0,5 - 1,0$

Za izračun aerodinamičke sile otpora potrebna nam je i površina okomitog presjeka prednjeg (udarnog) kraja tijela. Budući da automobil nije tijelo homogenog oblika, ovu veličinu aproksimirat ćemo za potrebe našeg fizikalnog modela:

$$A = 0,85 * sirina\_automobila * visina\_automobila \tag{3.14}$$

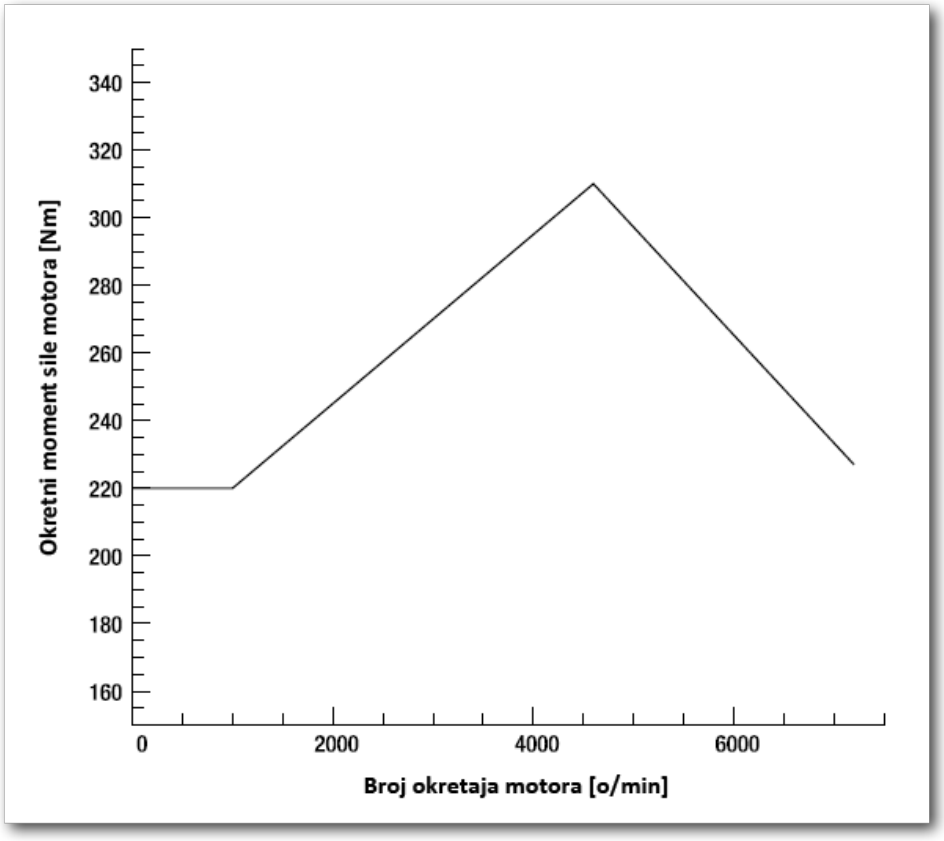
3.5 Izračun ubrzanja i brzine automobila

Simulatori upravljanja vozilom moraju biti sposobni u svakom trenutku izračunati akceleraciju i brzinu vozila. Izraz za izračun akceleracije već je dan u (3.6), gdje su nam poznate sve vrijednosti osim okretnog moment kotača, T_w. U (3.9) smo pokazali da okretni moment kotača ovisi o okretnom momentu motora, te trenutnom i završnom omjeru stupnja prijenosa. Omjeri stupnjeva prijenosa obično su nam poznati iz tehničkih specifikacija automobila, a okretni moment motora najčešće je prezentiran kao krivulja koju za naše potrebe treba aproksimirati matematičkim funkcijama. Tako primjerice krivulju okretnog

momenta motora sa slike 3.2 možemo pojednostaviti i opisati jednadžbama pravaca [10].

$$T_e = b\Omega_e + d \tag{3.15}$$

$$T_e(\Omega_e) = \begin{cases} 220 & \text{za } \Omega_e \leq 1000 \\ 0,025\Omega_e + 195 & \text{za } 1000 < \Omega_e < 4600 \\ -0,032\Omega_e + 457,2 & \text{za } \Omega_e \leq 4600 \end{cases} \tag{3.16}$$



Slika 3.7: Krivulja okretnog momenta motora aproksimirana pravcima

Budući da želimo akceleraciju automobila izraziti kao funkciju trenutne brzine automobila, morat ćemo izraziti okretni moment kotača u ovisnosti o trenutnoj brzini automobila. Prisjetimo se izraza (3.12) i zaključka da uz pretpostavku da nema proklizavanja kotača, brzinu automobila možemo izraziti kao umnožak kutne brzine i polumjera kotača. Uvrštavanjem izraza (3.15) i (3.12) u (3.6) dobivamo formulu za izračun akceleracije u ovisnosti o trenutnoj brzini automobila.

$$a = \frac{60 g_k^2 G^2 b v}{2\pi m r_w^2} + \frac{g_k G d}{m r_w} - \frac{1}{2}\frac{C_D \rho v^2 A}{m} - \mu_r g \cos\theta - g \sin\theta \tag{3.17}$$

Gornji izraz za akceleraciju zapravo je kvadratna jednadžba gdje je brzina v nepoznanica:

$$a = \alpha v^2 + \beta v + \gamma \tag{3.18}$$

$$\alpha = -\frac{1}{2}\frac{C_D \rho A}{m} \tag{3.19}$$

$$\beta = \frac{60 g_k^2 G^2 b}{2\pi m r_w^2} \tag{3.20}$$

$$\gamma = \frac{g_k G d}{m r_w} - \mu_r g \cos\theta - g \sin\theta \tag{3.21}$$

Koristeći kvadratnu jednadžbu (3.17) i izraz za izračun brzine rotacije kotača (3.12), u svakom trenutku možemo izračunati brzinu i akceleraciju automobila uz veliku pretpostavku da nema proklizavanja kotača.

3.6 Kočenje

Do sada smo se bavili samo ubrzanjem automobila, no svaki napredniji simulator modelirat će i usporavanje, tj. kočenje. Postoje dvije vrste kočenja koje je moguće modelirati - kočenje motorom, te kočenje nožnom kočnicom. Prva vrsta kočenja javlja se zbog same prirode motora, tj. načina gibanja cilindara motora. Rezultantni okretni moment pri kočenju motorom, T_{eb}, matematički je modeliran kao umnožak broja okretaja u sekundi i tzv. koeficijenta kočenja motorom, μ_{eb}, do kojeg je ponekad vrlo teško doći, pa ga često aproksimiramo vrijednošću $\mu_{eb} = 0,75$ [10].

$$T_{eb} = \mu_{eb}\frac{\Omega_e}{60} \tag{3.22}$$

Drugi način usporavanja automobila je nožnom kočnicom, prilikom čega disk pločice pritišću diskove kotača i time ih sprječavaju u rotaciji. Ovu vrstu kočenja vrlo je jednostavno matematički modelirati ukoliko nam je poznata vrijednost tzv. puta kočenja. Primjerice ako je automobilu koji se giba brzinom 25 m/s potrebno 40 m da se potpuno zaustavi, deceleracija se može izračunati kao

$$a_b = -\frac{v_0^2}{2x} = -\frac{25^2}{2 * 40} = 7,8125\ m/s^2 \tag{3.23}$$

3.7 Proklizavanje kotača

Do sada smo modelirali fizikalno ponašanje kotača automobila ne uzimajući u obzir proklizavanje kotača. Već znamo da se na kotače prenosi okretni moment sile motora, a zahvaljujući sili trenja koja djeluje između guma kotača i podloge, kotači su sposobni pokretati automobil. Sila trenja koju smo opisali dijagramom sila u poglavlju 3.1 Dijagram sila predstavlja najveću moguću silu kotača pri kojoj ne dolazi do proklizavanja.

$$F_T = \mu_k F_N = \mu_k mg \cos\theta \tag{3.24}$$

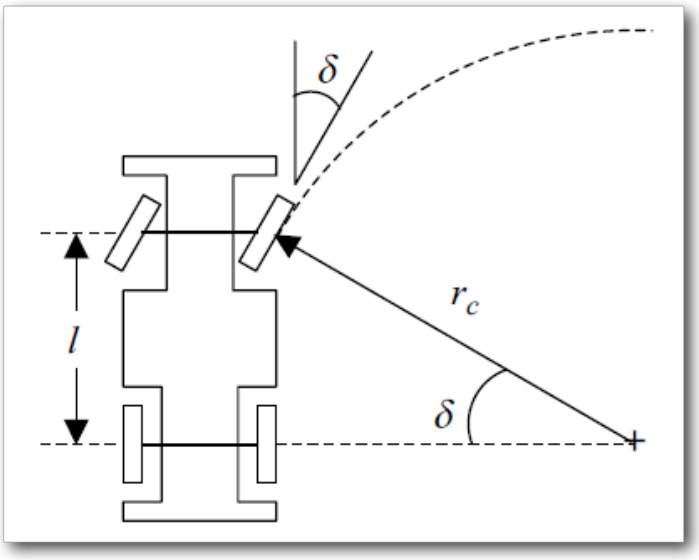
Ovu silu nazivamo silom proklizavanja, a njena vrijednost utvrđuje se tzv. testom proklizavanja gdje se automobil ubrzava po kružnoj stazi sve do njegovog izletanja, kada centripetalna sila postaje jača od sile proklizavanja. Rezultat testa proklizavanja je akceleracija pri kojoj kotači ne proklizavaju.

Prilikom programske implementacije proklizavanja najprije ćemo računati silu kojom kotači djeluju na podlogu, a zatim ćemo ju usporediti sa silom proklizavanja. U slučaju da je sila kotača manja od sile proklizavanja, u jednadžbama gibanja koristit ćemo silu kotača, a ukoliko je sila kotača veća od sile proklizavanja, koristit ćemo silu proklizavanja [10].

$$F_T = (\mu_k mg \cos\theta > \frac{T_e g_k G}{r_w}) \ ? \ \frac{T_e g_k G}{r_w} \ : \ \mu_k mg \cos\theta \tag{3.25}$$

3.8 Gibanje u zavojima

U ovom potpoglavlju bavit ćemo se matematičkim modeliranjem skretanja automobila, pri čemu ćemo zanemariti centripetalnu silu i proklizavanje kotača. Na slici 3.8 prikazana je shema automobila u zavoju. Prednji kotači zakrenuti su za kut δ tako da automobil skreće udesno. U slučaju da se automobil giba konstantnom brzinom kretat će se po kružnoj putanji radijusa r_c. Središte kružne putanje je sjecište okomica na prednje i zadnje kotače. Udaljenost prednjih i zadnjih kotača, označena na slici kao l, naziva se međuosovinski razmak.



Slika 3.8: Shema automobila u zavoju

Radijus r_c moguće je pronaći koristeći trigonometrijske odnose u trokutu:

$$r_c = \frac{l}{\sin \delta} \tag{3.26}$$

Kutnu brzinu gibanja automobila izrazit ćemo koristeći gornji izraz za radijus r_c:

$$\omega_t = \frac{v}{r_c} = \frac{v \sin \delta}{l} \tag{3.27}$$

Jednadžbe (3.26) i (3.27) su sve što nam je potrebno za modeliranje skretanja automobila pri malim brzinama. Prilikom programske implementacije najprije ćemo iz kuta zakrenutosti kotača, δ, i međuosovinskog razmaka, l, izračunati radijus kružne putanje, r_c. Zatim ćemo kutnu brzinu izračunati koristeći radijus, r_c, i trenutnu tangentnu brzinu, v. Iz kutne brzine na kraju računamo vrijeme potrebno da automobil opiše određeni kut skretanja [10, 11].

4 Programska implementacija

Ovo poglavlje donosi nam detaljniji opis praktičnog dijela diplomskog rada, tj. programske implementacije simulatora upravljanja automobilom. Objasnit ćemo kako preslikati fizikalni model automobila iz prethodnog poglavlja u programski kod koristeći programski jezik `C#`, gotovi fizikalni pogon *BEPUphysics*, te *XNA* biblioteku za razvoj grafičkih aplikacija i video igara. Na samom početku napravit ćemo kratki uvod u XNA radni okvir, a zatim ćemo se upoznati s BEPUphysics fizikalnim pogonom te objasniti kako uz pomoć njega modelirati vozilo.

4.1 XNA okruženje

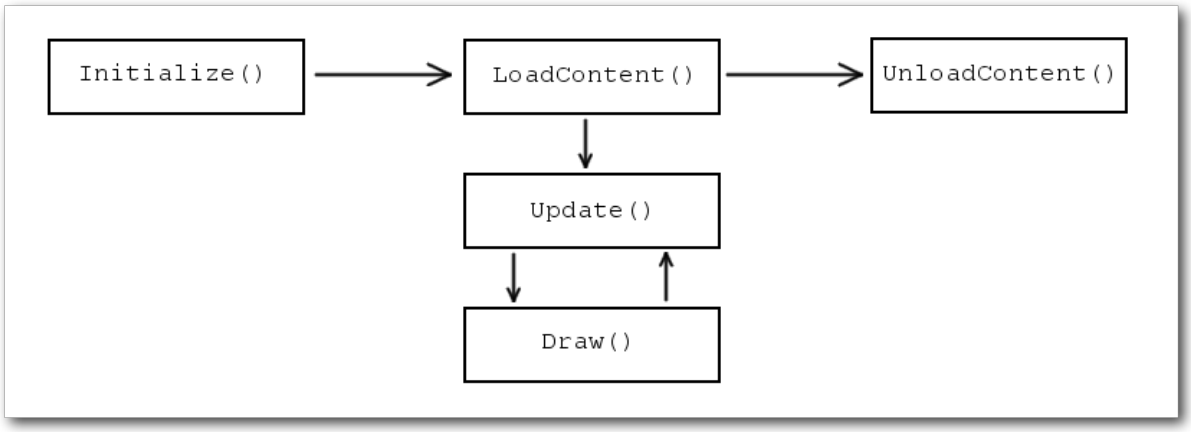
XNA je Microsoftova razvojna platforma kojoj je glavni cilj pružiti korisnicima jednostavan i elegantan pristup razvoju video igara. Svoju popularnost duguje visokoj razini apstrakcije ostvarenoj kroz vješto osmišljene biblioteke namijenjene rješavanju raznih kompleksnih zadataka. Specifičnost XNA radnog okvira leži u mogućnosti izvršavanja istog koda uz minimalne preinake na nekoliko različitih platformi kao što su Windows računala, Xbox 360, Zune, te Windows Phone 7.

Naziv *XNA* izvorno dolazi od ***X****Box* ***N****ew* ***A****rchitecture development*, no nakon izlaska XBox platforme 2005. godine autori su se odlučili našaliti igrom riječi te ovaj radni okvir dobiva puni naziv ***X****NA is* ***N****ot an* ***A****cronym*. Ovaj skup alata nastao je 2004. godine, a službeno je objavljen 2 godine kasnije u Kaliforniji. Službeno razvojno okruženje za razvoj XNA aplikacija naziva se *XNA Game Studio* (trenutne verzije 4.0) i oslanja se na poznato razvojno okruženje Visual Studio [13, 14, 15].

4.1.1 Struktura XNA programskog koda

Da bi ostvarili ciklički program kao što su računalne igre potrebno je implementirati 5 osnovnih metoda klase `Game` radnog okvira XNA (slika 4.1). Prva metoda je `Initialize()` koja je zadužena za inicijalizaciju raznih postavki aplikacije kao što su na primjer veličina i pozicija prozora te pozadinske boje. Metoda `Initialize()` izvršava se pri samom pokretanju programa, a nakon nje izvršava se metoda `LoadContent()`. Ova metoda zadužena je za učitavanje svih resursa potrebnih za pravilno izvođenje naše aplikacije. To mogu biti 3D modeli, teksture, zvukovi, video isječci, fontovi i sl. Po završetku rada programa

poziva se metoda `UnloadContent()` koja „odbacuje“ učitane resurse kako bi se rasteretio grafički uređaj sustava. Nakon što je program učitao potrebne resurse ciklički se izvršavaju dvije najvažnije metode klase `Game`: `Update()` i `Draw()`, čiji su pozivi sinkronizirani s otkucajem sata procesora i ovise o postavkama programa. `Update()` metoda služi za obradu logike i manipulaciju podacima potrebnima za ostvarenje raznih funkcionalnosti aplikacije. U `Update()` metodi se tako uobičajeno nalazi programski kod koji obavlja akcije pokrenute preko kontrola od strane igrača, tj. korisnika aplikacije. Metoda `Draw()` služi za iscrtavanje svih komponenti igre kao što su 3D modeli, teksture, efekti, i sl [13, 15, 16].



Slika 4.1: Struktura XNA programskog koda

4.2 BEPUphysics fizikalni pogon

Kao što smo već prije naučili, fizikalni pogoni su računalni programi koji omogućavaju približnu simulaciju određenih fizikalnih pojava u domeni računalne grafike, a najčešće računalnih igara. U većini današnjih igara brzina simulacije puno je važnija od preciznosti izračuna, pa fizikalni pogoni pružaju samo perceptivno točnu aproksimaciju stvarne fizike pojava oko nas. Fizikalni pogoni obično se sastoje od dva glavna dijela: otkrivanja sudara (detekcija kolizije) i simulacije dinamike. Napredniji fizikalni pogoni uz dinamiku krutih tijela implemetiraju i dinamiku mekih tijela i fluida.

U vrijeme pisanja diplomskog rada bio je dostupan čitav niz fizikalnih pogona za XNA okruženje, pa je kratak pregled dan u sljedećoj tablici.

Tablica 4.1: Fizikalni pogoni u XNA okruženju

Open Source fizikalni pogoni		
Područje primjene	Naziv pogona	Platforme
2D	Box2D	Windows, XBOX 360, Windows Phone 7
	Farseer	Windows, XBOX 360, Windows Phone 7
3D	Henge	Windows, XBOX 360, Windows Phone 7
	BEPUphysics	**Windows, XBOX 360, Windows Phone 7**
	Jitter	Windows, XBOX 360, Windows Phone 7
	BulletSharp	Windows
Komercijalni fizikalni pogoni		
Područje primjene	Naziv pogona	Platforme
3D	Matali	Windows, XBOX 360, Windows Phone 7
	DigitalRun	Windows, XBOX 360, Windows Phone 7

Budući da je nama potreban 3D fizikalni pogon, te bi pritom voljeli imati uvid u izvorni kod, izbor nam se uvelike sužava. Odabrat ćemo tako BEPUphysics pogon koji pruža najkvalitetniju dokumentaciju, te okuplja velik broj korisnika u aktivnu zajednicu. BEPUphysics pogon odlikuje prilagodljivo kontinuirano otkrivanje sudara, generiranje terena, te statičkih i dinamičkih objekata iz učitanih modela, implementacija vozila, uključivanje višedretvenog izračuna, itd.. Najvažnije značajke ovog fizikalnog pogona upoznat ćemo u sljedećem poglavlju kroz korake implementacije simulatora automobila.

4.3 Simulator automobila

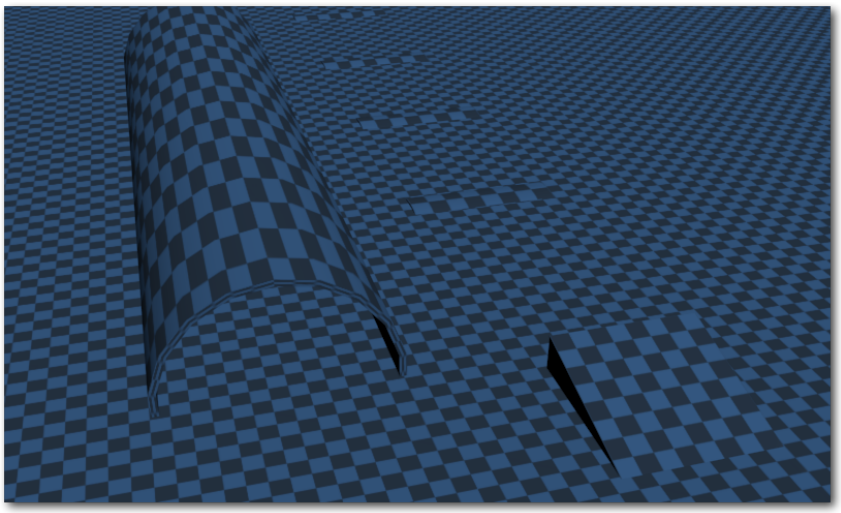
4.3.1 3D modeli

Većina fizikalnih pogona, pa tako i BEPUphysics, prilikom modeliranja fizikalnog ponašanja dijele automobil u dvije komponente: kotače i šasiju. Dakle, automobil je u fizikalnim pogonima predstavljen krutim tijelom (šasijom) na koje se oprugama vežu kružnice ili cilindri (kotači). Svaki kotač pokreće motor koji mu prenosi okretni moment kao što je objašnjeno u poglavlju 3.3 Stupnjevi prijenosa i okretni moment kotača. To znači da ćemo morati razdvojiti 3D model automobila na kotače i šasiju, te ih odvojeno učitati u program. Koristit ćemo 3D model automobila Austin Mini Cooper iz 1969. godine, kojeg

je moguće besplatno preuzeti s web stranice *artist-3d.com*, te koristiti u nekomercijalne svrhe (slika 4.2).

Slika 4.2: 3D modeli šasije i kotača

Osim 3D modela kotača i šasije, bit će nam potreban i poligon po kojem ćemo upravljati automobilom. Poligon ćemo sami modelirati koristeći studentsku verziju alata za 3D modeliranje i animaciju Autodesk 3D Studio Max 2013. Poligon za vožnju vrlo je jednostavno modelirati, pa nam neće trebati neko preveliko predznanje u 3D dizajniranju. Jednostavno ćemo kreirati ravnu površinu, na koju ćemo zatim dodati zapreke koristeći niz standardnih primitiva, kao što su cilindri, piramide i kvadri (slika 4.3).



Slika 4.3: 3D model poligona

4.3.2 Generiranje terena

Prethodno modelirani poligon za vožnju želimo dodati u simulator kao statički teren koji će biti u koliziji s automobilom kojim ćemo upravljati. BEPUphysics pogon implementira niz klasa koje nam omogućavaju jednostavno dodavanje objekata u prostor (eng. *space*). `Entity` klasa primjerice omogućava dodavanje standardnih fizičkih objekata kao što su kvadri, cilindri, kugle, kapsule i drugi koji se mogu naći u prostoru imena `BEPUphysics.Entities`. BEPUphysics razlikuje dvije osnovne vrste entiteta u prostoru: dinamičke i kinematičke. Dinamički entiteti se mogu sudarati, padati, skakati ili klizati, kao što je i za očekivati. Kinematički entiteti se također mogu gibati kao i dinamički, ali njihova je masa beskonačna. To znači da se njihova brzina neće mijenjati pod utjecajem sudara s drugim entitetima. Kinematički entiteti su idealni za opisivanje statičkih objekata kao što je teren, ili u našem slučaju poligon za vožnju. U slučaju da želimo stvoriti dinamički entitet pozvat ćemo konstruktor s masom kao jednim od parametara, a u slučaju dodavanja kinematičkih entiteta, masu ćemo izostaviti.

```
Box staticBox = new Box(Vector3.Zero, 30, 1, 30);
space.Add(staticBox);
Box dynamicBox = new Box(new Vector3(0, 4, 0), 1, 1, 1, 1);
space.Add(dynamicBox)
```

Jedna od velikih prednosti BEPUphysics pogona je mogućnost stvaranja entiteta na temelju učitanih modela. To znači da možemo učitati model našeg poligona te iz njega kreirati statički entitet koji će nam predstavljati teren. No, postoji i druga mogućnost koja se češće koristi prilikom generiranja terena. BEPUphysics implementira klasu `StaticMesh` koja predstavlja statičku mrežu trokuta (eng. *triangle mesh*). Da bi kreirali statičku mrežu potrebno je izvući podatke o vrhovima učitanog modela, a za to nam služi pomoćna metoda `TriangleMesh.GetVerticesAndIndicesFromModel`. Programski odsječak koji implementira generiranje terena dan je u nastavku.

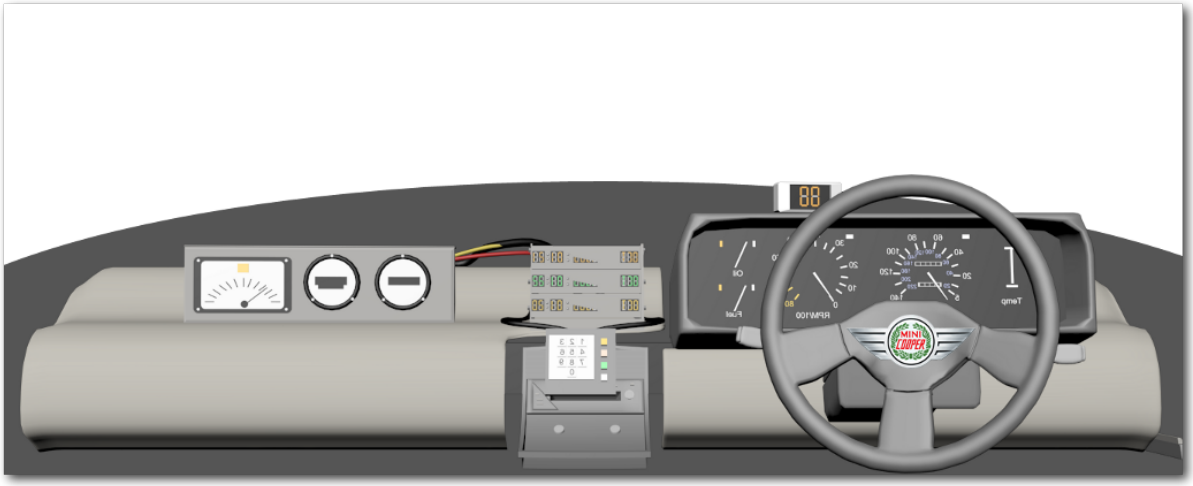
```
protected override void LoadContent()
{
    ...
    groundModel = Content.Load<Model>("Models/Ground");
    Vector3[] staticTriangleVertices;
    int[] staticTriangleIndices;
```

```
    TriangleMesh.GetVerticesAndIndicesFromModel(groundModel, out
        staticTriangleVertices, out staticTriangleIndices);
    var staticMesh = new StaticMesh(staticTriangleVertices,
        staticTriangleIndices, new AffineTransform(new Vector3(1, 1,
         1), Quaternion.Identity, new Vector3(0, 0, 0)));
    space.Add(staticMesh);
    ...
}
```

Programski odsječak 4: Učitavanje terena.

4.3.3 Kamere

Pratećom kamerom smo se već bavili u poglavlju 2.2.3 Elastična sila, gdje smo pokazali kako programski implementirati Hookeov zakov na primjeru oprugom vezane kamere. U ovom poglavlju implementirat ćemo još jednu vrstu kamere, onu iz perspektive vozača. Iskoristit ćemo prethodno implementiranu prateću kameru te ju smjestiti u središte automobila. Pritom ćemo isključiti iscrtavanje 3D modela automobila i elastično ponašanje kamere, te dodati teksture volana i kontrolne ploče (slika 4.4). Teksture ćemo iscrtavati kao 2D grafičke objekte (eng. *sprite*), s time da ćemo prilikom skretanja automobila rotirati teksturu volana.



Slika 4.4: Teksture volana i kontrolne ploče

```
if (currentKeyboardState.IsKeyDown(Keys.A))
{
    steeringWheelRotation -= 0.08f;
```

```
    if (steeringWheelRotation < -1.5f)
    steeringWheelRotation = -1.5f;
}
else if (currentKeyboardState.IsKeyDown(Keys.D))
{
    steeringWheelRotation += 0.08f;
    if (steeringWheelRotation > 1.5f)
        steeringWheelRotation = 1.5f;
}
else
{
    if (steeringWheelRotation < 0)
    {
        steeringWheelRotation += 0.08f;
    }
    if (steeringWheelRotation > 0)
    {
        steeringWheelRotation -= 0.08f;
    }
}
```

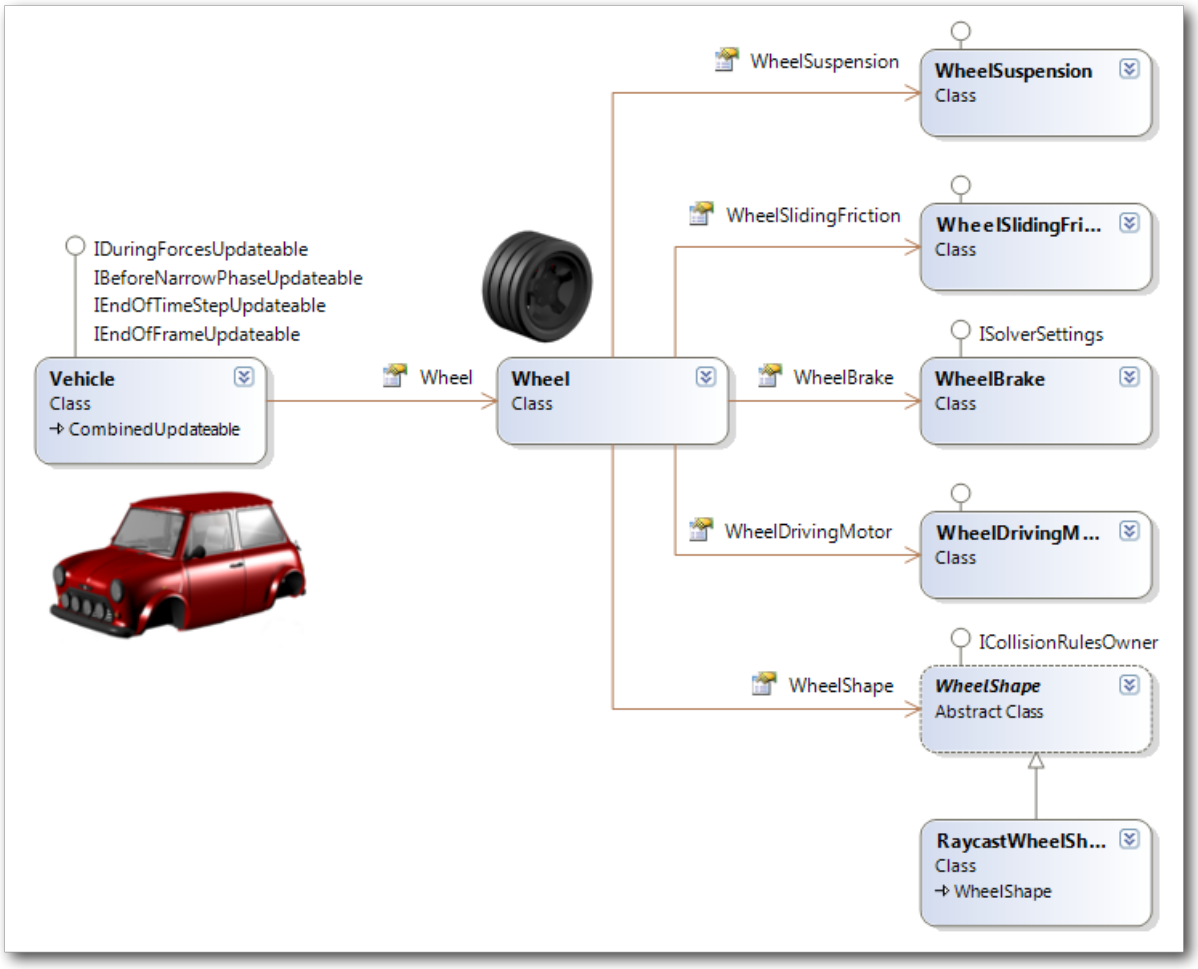
Programski odsječak 5: Rotacija teksture volana

Rotacija teksture volana samo je vizualni efekt koji nije potpuno precizan i u skladu s rotacijom kotača (programski odsječak 5). Volan ćemo zakretati za kut koji je u ovisnosti samo o vremenskoj duljini interakcije s tipkovnicom. Primjerice ukoliko držimo tipku za skretanje jednu sekundu kut zakretanja volana će biti veći nego u slučaju da smo držali tipku pola sekunde. Ukoliko nema interakcije s tipkama za skretanje automobila, volan se vraća u početni položaj.

4.3.4 Automobil

U ovom poglavlju dolazimo do najzanimljivijeg dijela implementacije - automobila. U poglavlju 4.3.1 3D modeli napomenuli smo da većina fizikalnih pogona modelira automobil u dva dijela. Prvi dio je šasija, tj. tijelo automobila, a drugi dio je lista kotača. Odgovarajuće 3D modele smo već učitali, ali s njima ne možemo apsolutno ništa bez pripadajuće fizikalne reprezentacije. Ne možemo znati niti točan položaj za iscrtavanje, jer će njega

izračunavati upravo fizikalni pogon. Važno je shvatiti da fizikalni izračun nema nikakve veze s 3D prikazom objekata nad kojima se vrši izračun. Fizikalni pogon nam daje samo informacije o položaju i orijentaciji objekata pomoću kojih prikazujemo učitane modele na sceni. Da bi bolje razumjeli kako je građen fizikalni model vozila poslužit ćemo se pojednostavljenim dijagramom klasa u nastavku.



Slika 4.5: Dijagram klasa Vehicle i Wheel

Atribut `body` klase `Vehicle` predstavlja šasiju, tj. tijelo vozila. Riječ je zapravo o instanci klase `Entity` s kojom smo se već susreli ranije u poglavlju. Klasa `Vehicle` također okuplja kotače vozila. Svaki kotač određen je oblikom, pozicijom na koju se veže na vozilo, oprugom (amortizerom), motorom koji ga pokreće, kočnicama i trenjem koje djeluje između njega i podloge. Kao što vidimo na slici 4.5 ove komponente su ostvarene u zasebnim klasama, a implementacija im se temelji na fizikalnom modelu opisanom u poglavlju 3. Fizikalni model vozila.

Svaki kotač je fizički objekt kao i tijelo automobila. On ima određeni oblik i dimenzije

pomoću kojih se izračunava kolizija s podlogom. BEPUphysics nam nudi dvije mogućnosti fizičke prezentacije kotača, pa tako kotače možemo modelirati kružnicama, ili ako želimo precizniju simulaciju, cilindrima. Klase `RaycastWheelShape` i `CylinderCastWheelShape` modeliraju oblike kotača, te su izvedene iz klase `WheelShape`.

Klasa `WheelDrivingMotor` modelira motor koji prenosi okretni moment sile na kotače. Prilikom stvaranja motora definiramo maksimalnu brzinu koju motor može razviti, te koeficijent trenja između podloge i gume. Na taj način se uzima u obzir ograničenja brzine i količina sile koju motor gubi prilikom predaje okretnog moment sile kotačima. Metoda `ApplyImpulse` izračunava ukupan impuls sile koji nastaje kada koristimo kontrole za upravljanje vozilom, te ih zatim primjenjuje na kotače.

Amortizeri kotača ostvareni su klasom `WheelSuspension` koja implementira fizikalno ponašanje amortizera koristeći opruge i Hookeov zakon. S implementacijom Hookeovog zakona već smo se susreli ranije, pa znamo da za svaki amortizer moramo definirati koeficijent, prigušenje i duljinu opruge.

Klasa `WheelBrake` i modelira sustav kočenja vozila, a temelji se na implementaciji sile trenja. Prilikom stvaranja instance klase `WheelBrake` definiramo statički i dinamički koeficijent sile trenja prilikom kočenja. Očito je da će ovi koeficijenti biti puno većeg iznosa nego koeficijenti trenja kotrljanja ili klizanja prilikom vožnje. Na temelju definiranih koeficijenata modelira se sila trenja kočenja koja će prilikom kočenja zamijeniti standardnu silu trenja kotrljanja. Osim sile trenja kočenja i sile trenja kontrljanja, BEPUphysics implementira i silu trenja klizanja. Primjerice ako postavimo mali koeficijent trenja klizanja, primjetit ćemo da automobil proklizava u zavojima.

Da bi kreirali novi automobil implementirat ćemo klasu `MiniCooperInput` koja će se brinuti za stvaranje automobila, ali i za promjenu parametara simulacije. Prilikom stvaranja automobila nećemo se brinuti o preciznosti simulacije, jer ćemo u sljedećem poglavlju implementirati grafičko sučelje koje će omogućiti izmjenu parametara simulacije tijekom izvođenja programa. Prilikom stvaranja automobila, šasiju ćemo modelirati spajajući dinamičke entiteteu u cjelinu (eng. *compound body*) koja odgovara učitanom modelu šasije.

```
var bodies = new List<CompoundShapeEntry>()
{
    new CompoundShapeEntry(new BoxShape(2.1f, .8f, 3.5f), new
        Vector3(0, 0, 0), 50),
```

```
    new CompoundShapeEntry(new BoxShape(2.1f, .3f, 2.7f), new
        Vector3(0, .8f / 2 + .3f / 2, 0.3f), 1),
    new CompoundShapeEntry(new BoxShape(2.1f, .5f, 2.0f), new
        Vector3(0, .8f / 2 + .3f / 2 + .5f /2, 0.5f), 1)
};
var body = new CompoundBody(bodies, 52);
body.CollisionInformation.LocalPosition = new Vector3(0, 0.5f, 0);
body.Position = position;
Vehicle = new Vehicle(body);
```

Programski odsječak 6: Modeliranje tijela automobila

Na tijelo automobila vezat ćemo kotače. Za svaki kotač definirat ćemo odgovarajući oblik, amortizer, motor koji ga pokreće, kočnicu, te silu trnja klizanja po podlozi. Ove parametre ćemo mijenjati tijekom izvođenja programa te promatrati njihov utjecaj na simulaciju fizikalnog ponašanja.

```
Vehicle.AddWheel(new Wheel(
     new RaycastWheelShape(.600f, wheelGraphicRotation),
     new WheelSuspension(2000, 80f, Vector3.Down, .81f, new Vector3
         (-1.0f, 0, 1.25f)),
     new WheelDrivingMotor(2.5f, 30000, 10000),
     new WheelBrake(1.5f, 2, .02f),
     new WheelSlidingFriction(4, 5)));
```

Programski odsječak 7: Primjer dodavanja zadnjeg lijevog kotača automobila

Da bi uopće mogli upravljati kreiranim automobilom implementirat ćemo metodu `Update()` koja će rukovati akcijama korisnika, tj. ostvariti interakciju s automobilom putem tipkovnice. Prilikom pritiskanja tipki W i S postavljat ćemo ciljnu brzinu koju želimo da automobil postigne u smjeru naprijed ili nazad (programski odsječak 8).

```
if (keyboardInput.IsKeyDown(Keys.A))
{
    steered = true;
    angle = Math.Max(Vehicle.Wheels[1].Shape.SteeringAngle -
        TurnSpeed * dt, -MaximumTurnAngle);
    Vehicle.Wheels[1].Shape.SteeringAngle = angle;
```

```
    Vehicle.Wheels[3].Shape.SteeringAngle = angle;
}
```

Programski odsječak 8: Pokretanje automobila u naprijed

Skretanje ćemo aktivirati korištenjem tipki A i D, na način da ćemo mijenjati kut zakretanja kotača. U koliko nema interakcije tipkama A i D, kotače ćemo vraćati u početni položaj (programski odsječak 9).

```
float angle;
bool steered = false;
if (keyboardInput.IsKeyDown(Keys.A))
{
    steered = true;
    angle = Math.Max(Vehicle.Wheels[1].Shape.SteeringAngle -
        TurnSpeed * dt, -MaximumTurnAngle);
    Vehicle.Wheels[1].Shape.SteeringAngle = angle;
    Vehicle.Wheels[3].Shape.SteeringAngle = angle;
}
if (keyboardInput.IsKeyDown(Keys.D))
{
    steered = true;
    angle = Math.Min(Vehicle.Wheels[1].Shape.SteeringAngle +
        TurnSpeed * dt, MaximumTurnAngle);
    Vehicle.Wheels[1].Shape.SteeringAngle = angle;
    Vehicle.Wheels[3].Shape.SteeringAngle = angle;
}
if (!steered)
{
    if (Vehicle.Wheels[1].Shape.SteeringAngle > 0)
    {
        angle = Math.Max(Vehicle.Wheels[1].Shape.SteeringAngle -
            TurnSpeed * dt, 0);
        Vehicle.Wheels[1].Shape.SteeringAngle = angle;
        Vehicle.Wheels[3].Shape.SteeringAngle = angle;
    }
    else
    {
        angle = Math.Min(Vehicle.Wheels[1].Shape.SteeringAngle +
            TurnSpeed * dt, 0);
```

```
            Vehicle.Wheels[1].Shape.SteeringAngle = angle;
            Vehicle.Wheels[3].Shape.SteeringAngle = angle;
30      }
}
```

Programski odsječak 9: Skretanje automobila

4.3.5 Grafičko korisničko sučelje

Budući da želimo simulator učinit zabavnijim, ali i primjenjivim u praktičnim analizama ponašanja automobila, omogućit ćemo korisnicima *runtime* izmjenu parametara simulacije putem grafičkog korisničkog sučelja. Korisničko sučelje implementirat ćemo korištenjem 2D tekstura (eng. *sprites*), koje ćemo organizirati u nekoliko panela: panel s parametrima šasije, panel s parametrima kotača, te paneli s korisničkim uputama i informacijama o projektu.

Slika 4.6: Primjer panela grafičkog korisničkog sučelja

U panele ćemo upisivati trenutne parametre simulacije. Da bi ih mogli promijeniti trebat će nam gumbi koji nažalost nisu implementirani XNA bibliotekom, pa ćemo morati sami kreirati klasu `Button` i implementirati željeno ponašanje. Gumb će se sastojati od dvije teksture - aktive u boji i neaktivne sive (slika 4.7). Kada se kursor miša nalazi iznad guba ili ako je gumb aktivan iscrtavat će se tekstura u boji, a u drugim slučajevima iscrtavat će se siva tekstura. Ukoliko je registriran klik miša iznad teksture gumba izvršit

će se odgovarajuća akcija. Prilikom izvršavanja akcije gumba koristit ćemo prethodno kreiranu klasu `MiniCooperInput` preko koje ćemo doći do tijela ili kotača automobila, te promijeniti odgovarajuće parametre.

U nastavku je prikazan primjer stvaranja gumba pozivom konstruktora klase `Button` koji prima parametre aktivnu i neaktivnu teksturu, 2D poziciju iscrtavanja, te način rada koji ovisi o tome treba li gumb nakon klika ostati aktivan ili neaktivan.

```
Texture2D buttonTexture;
Texture2D butttonTextureActive;

buttonTexture = Content.Load<Texture2D>("Sprites/button_body");
butttonTextureActive = Content.Load<Texture2D>("Sprites/
    button_body_active");
Button bodyOptionsButton = new Button(buttonTexture,
    butttonTextureActive, new Vector2(35, 90), spriteBatch, true);
```

Programski odsječak 10: Konstruktor klase Button

Slika 4.7: Primjer aktivnih i neaktivnih tekstura gumba

4.4 Parametri simulacije

5 Zaključak

Diplomskim radom *Fizikalni model vozila* upoznali smo se s implementacijom fizikalnih pogona koji su postali neizostavni dio svake moderne računalne igre ili simulacije. Simulacijska fizika samo je približna stvarnim zakonima fizike, budući da koristi brojna pojednostavljenja, te naravno diskretne vrijednosti prilikom izračuna. Možemo reći da nam fizikalni pogoni pružaju perceptivno točnu aproksimaciju stvarne fizike pojava oko nas, što je za neše potrebe sasvim dovoljno.

U poglavlju 2. Klasična mehanika upoznali smo se s temeljima svakog fizikalnog pogona. Objasnili smo značaj Newtonovih zakona, a posebno temeljnog zakona gibanja. Poznavajući rezultantnu silu na neko tijelo te njegovu masu, fizikalni pogoni su sposobni modelirati gibanje računajući odgovarajuću akceleraciju tijela. Da bi došli do rezultantne sile, modeliraju se osnovne vrste sila kao što su gravitacijska, elastična, centripetalna i sila trenja, kojih smo se prisjetili u ovom radu.

Znanja iz klasične mehanike iskoristili smo u fizikalnom modeliranju vozila. Naučili smo koje sve sile djeluju na automobil u vožnji, te kako se snaga motora prenosi na kotače kroz sustav za prijenos snage. Naš fizikalni model pokazao se praktičnim i široko primjenjivim, te su tako neki dijelovi opisanog fizikalnog modela dostupni u velikom broju gotovih fizikalnih pogona. S druge strane, neki se segmenti, kao što je npr. izmjena stupnja prijenosa snage, ne mogu pronaći u fizikalnim pogonima koji su većinom namijenjeni razvoju arkadnih simulacija. Implementacija izmjene stupnja prijenosa učinila bi igru izrazito zahtjevnom i neigrivom, pa se ona često izostavlja iz fizikalnih pogona.

Diplomski rad je pratio i proces razvoja i implementacije vlastitog simulatora automobila, koji predstavlja praktični rezultat rada. Simulator implementiran ovim radom možemo svrstati i u akradne i u realistične simulacije. Naime, implementirali smo mogućnost promjene parametara simulacije tijekom izvođenja programa što uvelike utječe na fizikalno ponašanje vozila, pa tako i na sam karakter simulatora. XNA Game Studio se prilikom implementacije pokazao vrlo praktičnim razvojnim okruženjem koje pruža čitav niz gotovih metoda za grafičke transforamcije, što nam je uvelike olakšalo programsku implementaciju. Valja primijetiti da implementirani simulator nije samo računalna igra ili arkadna simulacija, već može poslužiti i kao sustav mjerenja utjecaja pojedinih parametara vožnje, te na taj način pronaći širu primjenu.

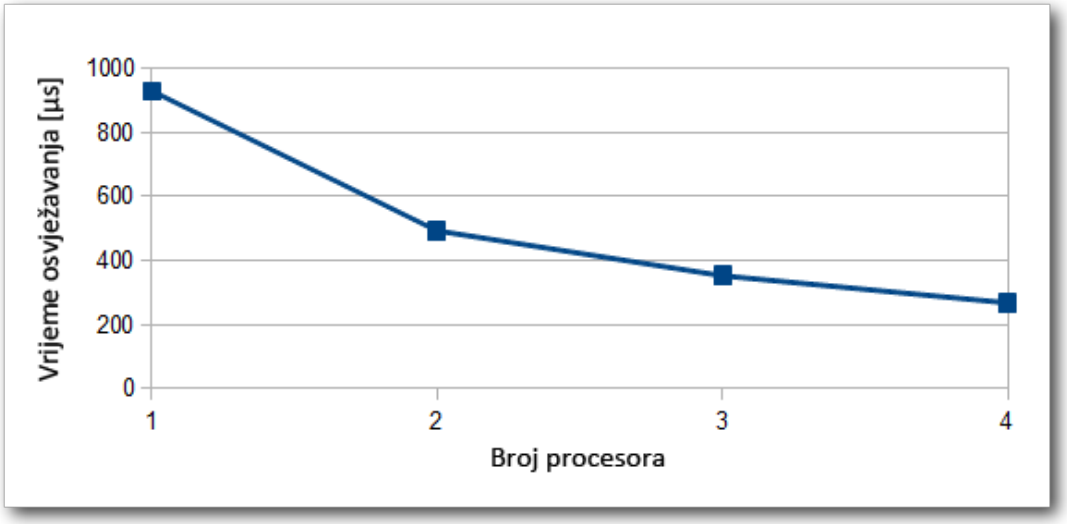
Prilikom simulacije fizikalnih pojava moramo voditi računa o jediničnim mjerama fi-

zikalnog pogona. Tako primjerice BEPUphysics pogon, koji smo koristili u praktičnom dijelu diplomskog rada, koristi jedan metar kao jediničnu mjeru u trodimenzionalnom prostoru. To znači da prilikom modeliranja vozila moramo uzeti u obzir ovu normu ukoliko ne želimo preračunavati jedinice veličina koje definiraju fizikalna ponašanja. Iz tog razloga modelirali smo šasiju automobila veličine $3,5$ jediničnih mjera, tj. $3,5$ metra. Razmjerno veličini šasije automobila definirali smo i dimenzije ostalih dijelova automobila - amortizere i kotače. BEPUphysics također definira kilograme kao zadanu jedinicu mase, stoga smo masu automobila definirali upravo u kilogramima. Mijenjajući parametre pojedinih komponenata automobila, došli smo do zaključka da u slučaju regularnog ponašanja simuliranog automobila, vrijednosti parametara odgovaraju stvarnim vrijednostima dobivenim mjerenjima. Primjerice postavljanje stvarnih vrijednosti mase tipičnog automobila ($\approx 1000\ kg$), tvrdoće opruge ($\approx 20000\ N/m$) te prigušenja opruge ($\approx 2000\ N/(M/s)$) rezultirat će vrlo realističnim ponašanjem automobila. Ovime dolazimo do zaključka da je fizikalni izračun BEPUphysics pogona prilično precizan i daje vrlo realistične simulacije.

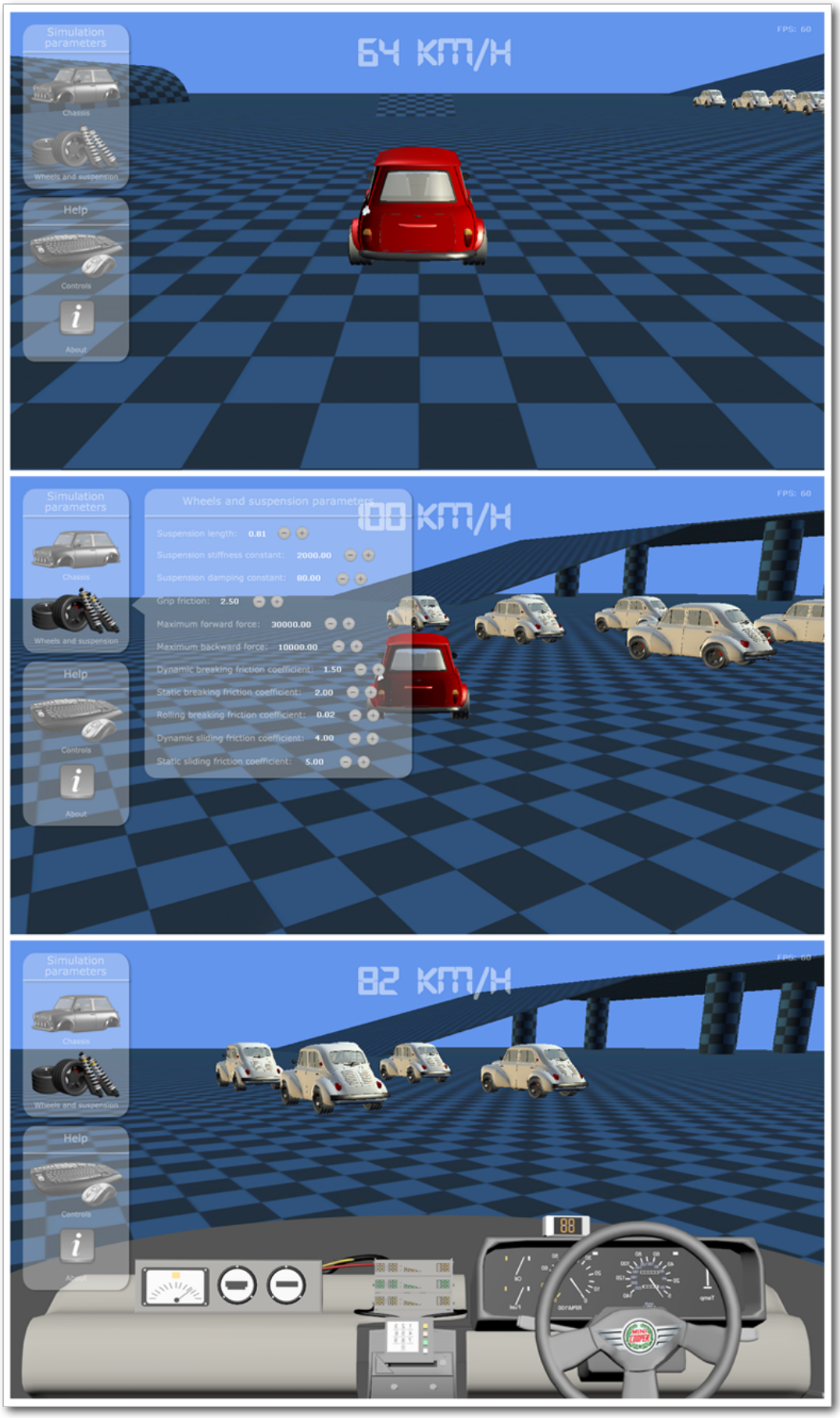
Da bi postigli stvarni doživljaj vožnje automobila valjalo bi implementriati podršku za različite ulazne uređaje kao što su volan te papučice gasa i kočnice. Ovim ulaznim uređajima postigli bi finije doziranje impulsa sile koji predajemo automobilu, te kuta zakretanja kotača. No, velik problem ovakvoj integraciji je XNA okruženje koje zasad pruža podršku samo za *XBox Game Pad* ulazni uređaj. Podršku za standardne igraće ulazne uređaje trebalo bi implementirati vlastitim kodom što predstavlja velik i složen posao. Osim podrškom za različite ulazne uređaje, naš simulator mogao bi se poboljšati i boljim modeliranjem sudara, primjerice deformacijom materijala. Nažalost, BEPUphysics ne pruža podršku za deformaciju tijela, niti fiziku mekih tijela. No moguće je podijeliti mrežu modela šasije na više dijelova, koji bi se prilikom jačih sudara odvajali od automobila.

Na samom kraju prokomentirat ćemo performanse BEPUphysics fizikalnog pogona. Većina fizikalnog izračuna odvija se na centralnoj procesorskoj jedinici, ali u budućim verzijama pogona spominje se i mogućnost ubrzavanja korištenjem grafičkog procesora. Već smo ranije spomenuli da se trenutna verzija pogona (v1.2.0) odlikuje mogućnošću višeprocesorskog izračuna čime se uvelike ubrzava simulacija. Tako će za osvježavanje informacija o simulaciji 10 000 jednostavnih primitiva biti potrebno oko 900 μs prilikom izvođenja na jednom procesoru, dok će u slučaju višeprocesorkog izračuna biti potrebno

oko 350 μs (mjerenje je izvršeno na procesoru Intel® Core™2 Quad Processor Q6600, 2.40 GHz). Ovakve performanse ubrzanja izračuna uvelike su utjecale i na naš odabir fizikalnog pogona, te možemo slobodno reći da smo izabrali trenutno najnapredniji fizikalni pogon dostupan u XNA okruženju.



Slika 5.1: Rezultati mjerenja višeprocesorskog izračuna



Slika 5.2: Rezultat praktičnog dijela diplomskog rada

Literatura

[1] Palmer, G., Physics for Game Programmers: Basic Newtonian Mechanics, SAD: Apress, 2005.

[2] Bourg, D.M., Physics for Game Developers: Basic Concepts, SAD: O'Reilly, 2002.

[3] Glumac, Z., Klasična mehanika: Newtonovi aksiomi gibanja, konzervativnost, rad, energija momenti, Osijek: 2013.

[4] Wikipedia: Newtonovi zakoni gibanja, 31. ožujak 2013., *Newtonovi zakoni gibanja*, `http://hr.wikipedia.org/wiki/Newtonovi_zakoni_gibanja`, 1. lipnja 2013.

[5] Furić, M., Gravitacijska sila, Predavanja iz predmeta Opća fizika 1, Prirodoslovno-matematički fakultet, 2012.

[6] Kolonić, F.; Sumina, D., Trenje, Predavanja iz predmeta Upravljanje elektromotornim pogonima, Fakultet elektrotehnike i računarstva, 2013.

[7] Wikipedia: Hookeov zakon, 31. ožujak 2013., *Hookeov zakon*, `http://hr.wikipedia.org/wiki/Hookeov_zakon`, 5. lipnja 2013

[8] Bistričić L., Centripetalna sila, Predavanja iz predmeta Fizika 1, Fakultet elektrotehnike i računarstva, 2013.

[9] Wikipedia: Aerodinamička sila otpora, 11. ožujak 2013., *Aerodinamička sila otpora*, `http://hr.wikipedia.org/wiki/Aerodinami%C4%8Dka_sila_otpora`, 5. lipnja 2013.

[10] Palmer, G., Physics for Game Programmers: Cars and Motorcycles, SAD: Apress, 2005.

[11] Bourg, D.M., Physics for Game Developers: Cars, SAD: O'Reilly, 2002.

[12] Prometna zona: Prijenos snage, 2007., *Prijenos snage od motora na kotače*, `http://www.prometna-zona.com/autodijelovi-035_prijenos_snage.php`, 6. lipnja 2013.

[13] Mrazović P., Postupci animacije ljudskih likova, Završni rad preddiplomskog studija, Fakultet elektrotehnike i računarstva, 2011.

[14] Reed, A., Learning XNA 4.0, SAD: O'Reilly, 2011.

[15] Lobao, A.S.; Evangelista, B.; Leal de Farias, J.A.; Grootjans, R.; Beginning XNA 3.0 Game Programming: From Novice to Professional, SAD: Apress, 2009.

[16] Nitschke, B., Professional XNA Game Programming (For Xbox 360 and Windows): Writing a Racing Game, SAD: O'Reilly, 2002.

Sažetak

Diplomski rad bavi se modeliranjem fizikalnog ponašanja vozila. Poseban naglasak je dan fizikalnim pogonima i njihovim primjenama u modernim realističnim i arkadnim simulatorima. Rad nas upoznaje s temeljima fizikalnih pogona i daje uvid u osnove klasične mehanike kao što su Newtonovi zakoni gibanja i dinamika čvrstog tijela. Koristeći znanja klasične mehanike ostvaren je fizikalni model vozila, koji je i programski implementiran u jeziku C# i XNA okruženju za razvoj grafičkih aplikacija. Rad tako prati i proces razvoja praktičnog dijela diplomskog rada, programske implementacije interaktivnog simulatora upravljanja automobilom.

KLJUČNE RIJEČI:

fizikalni model vozila, fizikalni pogon, klasična mehanika, Newtonovi aksiomi gibanja, XNA, BEPUphysics

Summary

Paper describes the process of modeling physical behavior of vehicles. Special emphasis is given to physical engines and their applications in modern realistic and arcade simulations. The paper introduces the basis of physical engines and provides insight into fundamentals of classical mechanics, such as Newton's laws of motions and rigid body dynamics. The working knowledge of the classical mechanics was demonstrated in realization of physical model of vehicles, which was implemented in C# programming language and XNA framework for video game development. The paper also follows development of its practical result, interactive car simulator.

KEYWORDS:

physical model of vehicles, physical engine, classical mechanics, Newton's laws of motions, XNA, BEPUphysics