SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

RAD S ROBOTSKIM SIMULATORIMA POMOĆU ZASLONA ZA VIRTUALNU STVARNOST

Rijeka, rujan 2017.

Sonia Žuškin

0036475182

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

RAD S ROBOTSKIM SIMULATORIMA POMOĆU ZASLONA ZA VIRTUALNU STVARNOST

Mentor: Doc. dr. sc. Dražen Brščić

Rijeka, rujan 2017.

Sonia Žuškin

0036475182

(original zadatka)

IZJAVA

kojom izjavljujem da sam završni rad s naslovom "Rad s robotskim simulatorima pomoću zaslona za virtualnu stvarnost" izradila samostalno pod voditeljstvom doc. dr. sc. Dražena Brščića. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući navela u završnom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama.

Suglasna sam s objavom završnoga rada na službenim stranicama Fakulteta.

Rijeka, rujan 2017. _____

Sonia Žuškin

ZAHVALA

Zahvaljujem se doc. dr. sc. Draženu Brščiću na mentorstvu i pomoći, nabavki potrebnih resursa za rad te stručnome vođenju kroz proces izrade završnoga rada pod naslovom "Rad s robotskim simulatorima pomoću zaslona za virtualnu stvarnost".

Sonia Žuškin

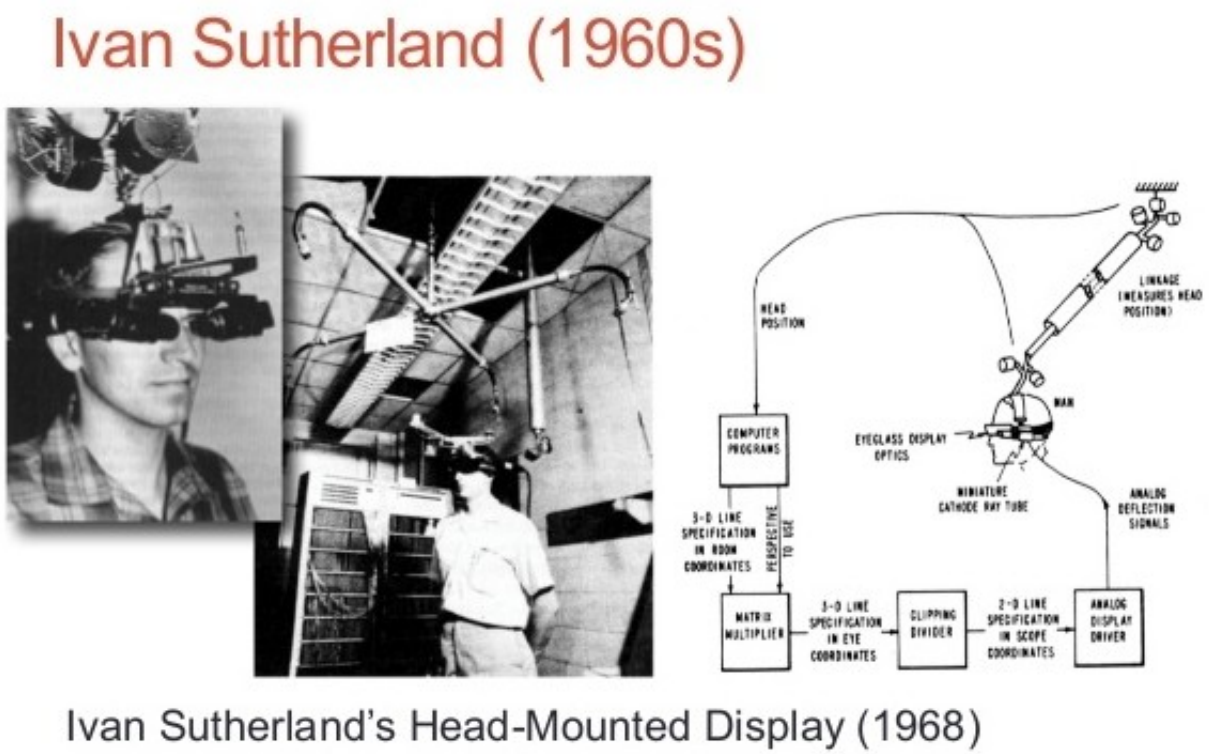
Sadržaj

1. UVOD.....1
2. ROBOTSKI OPERACIJSKI SUSTAV.....3
2.1 Čvorovi.....4
2.2 Teme.....5
2.3 Usluge.....6
2.4 Poruke.....6
2.5 Torbe.....6
2.6 Master.....6
2.7 Parametarski poslužitelj.....7
2.8 Instalacija ROS-a.....7
3. GAZEBO.....9
3.1 Komponente Gazeba.....9
3.1.1. World datoteke.....9
3.1.2 Varijable okruženja.....9
3.1.3 Gazebov poslužitelj.....9
3.1.4 Grafički klijent.....10
3.1.5 Dodaci.....10
3.2 Ovisnosti.....10
3.2.1 SDFormat.....10
3.2.2 Ignition Math.....10
3.2.3 Ignition Transport.....11
3.2.4 Ignition Messages.....11
3.3 Instalacija.....12
4. SIMULACIJA NA ZASLONU ZA VIRTUALNU STVARNOST.....15
4.1 Pokretanje programa.....16
4.2 Performanse računala.....24
4.3 Povezanost simulatora s HTC Vive-om.....27
5. ZAKLJUČAK.....30
6. LITERATURA.....31
7. POPIS SLIKA.....32

1. UVOD

Virtualna stvarnost i robotika spadaju u brže razvijajuće tehnologije suvremenoga svijeta. Sve brži napredak tehnologije i hardvera čine zaslone za virtualnu stvarnost zastupljenijima široj populaciji ljudi. Povijest virtualne stvarnosti proteže se od 18. stoljeća gdje je Charles Wheatstonovo istraživanje [1] demonstriralo da mozak procesuira dvodimenzionalne slike drugačije u svakome oku, u jedan trodimenzionalni objekt. Korisniku je dočarani doživljaj uronjenosti i performanse na način da mu je dano da gleda dvije stereoskopske slike, gdje je svaka slika prikazana jednome oku. Osnovni se principi stereoskopa koriste i danas za popularni Googleov Carboard i nisko budžetni zaslon za virtualnu stvarnost koristeći mobilne telefone.

Ivan Suterland je na sveučilištu Utah, uz pomoć studenta Boba Sproull-a, 1968. godine izumio prvi zaslon za virtualnu stvarnost koji se postavlja na glavu i povezuje na računalo (slika 1.1). Njegov je zaslon bio toliko težak da mu je bila potrebna potpora sa stropa, pri čemu je cijeli sustav dobio ime Damoklov mač (engl. *Sword of Damocles*). Računalna grafika koja je bila generirana za zaslon za glavu bila je dosta primitivna skica sobe i objekta [1].



Slika 1.1: Jedan od prvih zaslona za virtualnu stvarnost za glavu [16]

Povijest se robotike proteže sve do drevne Grčke. Riječ "robot" prvi put se spominje 1921. godine u predstavi autora Karela Capeka. 1940. godine Issac Asimov producirao je seriju kratkih priča o robotima, a njegov je najveći utjecaj na robotiku bilo stvaranje tri poznata zakona robotike. George Devol izumio je 1954. godine prvi pravi moderni robot "Unimate" kojim se digitalno upravljalo i programiralo [2].

Trenutno stanje virtualne stvarnosti i robotike jest da su to relativno nove tehnologije nad kojima se pruža veliki potencijal za daljni razvoj. One su još uvijek skupe pa time i teže dostupne široj populaciji, kao što su to bila i prva osobna računala. Jedno od novijih istraživanja simulatora za virtualnu stvarnost vrši start-up kompanija Neurable gdje pomoću elektroda snimaju aktivnost mozga pomoću elektroencefalografije (EEG). Na taj način pokušavaju pomoću moždanih valova omogućiti korisniku kontrolu nad VR virtualnim igrama [6].

Veliki interes za zaslone za virtualnom stvarnosti imaju velike kompanije poput Valve-a, HTC-a i Google-a koji razvijaju HTC vive i Facebook-a koji je preuzeo Oculus Rift. Zahvaljujući tome ulaganja su se u VR (virtualnu stvarnost) i proširene stvarnosti (engl. augumented reality) utrostručile od 2014., odnosno od kada je Facebook preuzeo Oculus Rift [7]. Trenutni razvoj situacije nesumnjivo dovodi do zaključka da će virtualna stvarnost biti tehnologija budućnosti.

Tema je ovog završnog rada rad s robotskim simulatorima pomoću zaslona za virtualnu stvarnost. Zasloni za virtualnu stvarnost se među ostalim, često koriste u medicini kod pacijenata koji su pretrpjeli teže fizičke ozljede. Postoje istraživanja u kojima je pacijentima prikazana simulacija vlastitog oporavka od teže fizičke ozljede u kojoj je uvjerljivo dočaran osjećaj uronjenosti i performanse što je rezultiralo stvarnim bržim oporavkom. Ova tema kod mene je potakla interes za daljnjim radom i istraživanjem na ovom području.

2. ROBOTSKI OPERACIJSKI SUSTAV

ROS (robotski operacijski sustav) razvojno je okruženje za pisanje programske podrške za robote. Njegov je razvoj započeo 2007. godine Stanfordov laboratorij umjetne inteligencije. Pruža razne knjižice i alate kako bi se programerima olakšalo stvaranje robotskih aplikacija. ROS uključuje hardversku apstrakciju, kontrolu uređaja na niskim razinama, implementaciju često korištenih funkcionalnosti, prenošenje poruka između procesa i upravljanje paketima. Nije napravljen kao razvojno okruženje u stvarnome vremenu, iako ga je moguće integrirati s programskim kodom pisanim za sustave u stvarnome vremenu [3].

ROS nije operacijski sustav poput windows-a, mac-a ili linux-a, već je međusoftver koji povezuje operacijski sustav postavljen na našem računalu i programe koje pišemo.

ROS-ova se arhitektura sastoji od četiri različita elementa: infrastrukture, alata, sposobnosti i ekosistema. Infrastruktura se odnosi na više pokrenutih programa gdje ROS omogućuje tim programima da međusobno komuniciraju. Također sadrži razne drivere za uređaje za komunikaciju sa hardverom. Drugi su element alati. Pod ovim se podrazumijeva simulacija, vizualizacija, grafičko korisničko sučelje i bilježenje podataka. ROS pruža razne sposobnosti poput planiranja, kontrole, percepcije, mapiranja i manipulacije. ROS je veliki ekosistem koji pruža organizaciju paketa, distribuciju računalnog programa, dokumentaciju i tutorijale [16].

Filozofija je ROS-a da se komunikacija izvršava svaki sa svakim (engl. *peer-to-peer*). Individualni programi komuniciraju preko definiranog API-ja (engl. *application proggraming interface*, odnosno ROS poruka, usluga i slično). ROS definira kako se izvršava komunikacija između programa. S obzirom da se komunikacija izvršava više-na-više može biti raspodijeljen na više računala. Primjerice, možemo imati računalo i robot koji komuniciraju putem bežićne komunikacije. Tada nije bitno na kojemu su sustavu pokrenuti čvorovi zato što su API i komunikacija pokrenuti i na računalu i putem bežične komunikacije. ROS je višejezični međusoftver čiji se moduli mogu pisati u bilo kojem programskom jeziku za koji postoji klijentska knjižica. Također, on je besplatan za korištenje.

Catkin je razvojno okruženje ROS-a. To je direktorij gdje se modificira, izgrađuje i vrši instalacija catkin paketa. Catkin se paketi mogu izgraditi kao samostojeći projekti, na isti način kako se izgrađuju normalni cmake projekti, ali catkin pruža koncept radnog prostora, gdje

možemo izgraditi više neovisnih paketa zajedno. Catkin se sastoji od slijedećih direktorija: izvorišnog prostora (engl. *source space*), građevnog prostora (engl. *build space*), razvojnog prostora (engl. *development space*), instalacijskog prostora (engl. *install space*) i rezultatnog prostora (engl. *result space*).

Izvorišni prostor sadrži izvorišni kod catkin paketa. U ovome prostoru možemo klonirati/izvlačiti/odjaviti izvorni kod za pakete koje želimo izgraditi. Svaki direktorij unutar izvorišnog prostora sadrži jedan ili više catkin paketa. Taj bi prostor trebao ostati netaknut od konfiguracije, izgradnje ili instalacije. Sam korijen izvorišnog prostora sadrži simboličku vezu catkinovog CmakeLists.txt datoteke. Ta se datoteka poziva od strane cmake-a tokom konfiguracije catkin projekata u radnome prostoru. Ta se datoteka može stvoriti invokacijom *catkin_init_workspace* naredbom u direktoriju izvorišnog prostora.

Građevni je prostor mjesto gdje se cmake poziva za izgradnju catkin paketa u izvorišnom prostoru. Cmake i catkin skladište cache informacije i druge posredne datoteke u tome direktoriju.

U razvojni se prostor spremaju građevne mete prije njihove instalacije. Način na koji su mete organizirane u razvojnom prostoru je isti kao njihov raspored poslije instalacije. To nam pruža korisno testno i razvojno okruženje koji ne zahtijeva korak instalacije.

Instalacijski je prostor mjesto gdje se instaliraju izgrađene mete. Poziva se *make install* naredbom. [17]

ROS se sastoji od čvorova (engl. *nodes*), tema (engl. t*opics*), usluga (engl. *services*), poruka (engl. m*essages*), torbi (engl. *bags*), glavnoga (engl. *master*) i parametarskog poslužitelja (engl. *parameter server*).

2.1 Čvorovi

Čvorovi su procesi koji izvršavaju kalkulacije. Kombinirani su zajedno u graf i međusobno komuniciraju putem tema, RPC (engl. *remote procedure call*) usluga i parametarskoga poslužitelja. Robotski se kontrolni sistem uobičajeno sastoji od više čvorova. Primjerice, jedan

čvor upravlja robotskim motorima, drugi čvor upravlja lokalizacijom, treći čvor vrši izračun plana puta, četvrti pruža grafički prikaz sistema gdje dalje broj čvorova ovisi o kompleksnosti projekta.

Korištenje čvorova pruža nekoliko benefita cjeloukupnome sistemu. Padovi su sistema izolirani po pojedinome čvoru što uvelike olakšava detekciju greške.

Svi pokrenuti čvorovi imaju izvorno grafičko ime koji su jedinstveno identificirani ostatku sistema. Također su čvorovi raspoređeni po tipu čvorova što olakšava proces referiranja izvršnog čvora na sustav datoteka. Kako bi ROS otkrio tip čvora, on traži izvršne dokumente u paketu sa određenim imenom i odabire prvoga kojeg pronađe. Zbog toga treba obratiti posebnu pozornost prilikom imenovanja izvršnih datoteka u istome paketu [8].

2.2 Teme

Teme su sabirnice putem kojih čvorovi izmjenjuju poruke. Imaju anonimne objavljivače/pretplatnike koji razgranjuju informacije. Općenito, čvorovi nisu svjesni s kime komuniciraju. Umjesto toga čvorovi kojima su potrebni određeni podatci se preplaćuju na njima relevantnu temu. Čvorovi koji proizvode podatke ih objavljuju temi kojoj su potrebni ti podatci. Komunikacija između tema i čvorova je više-na-više (engl. *peer-to-peer*).

Teme su raspodijeljene u različite tipove podataka. Čvorovi mogu primati poruke s pripadajućim tipom. Master ne prisiljava konzistenciju tipova podataka između objavljivača, ali transport poruka između pretplatnika neće biti omogućena ako nisu istoga tipa. Također svi ROS klijenti vrše provjeru podata koristeći MD5. Putem te sigurnosne provjere napravljenog za poruke provjerava se podudarnost poruka. Ova provjera osigurava da su ROS-ovi čvorovi napravljeni od konzistentne baze kodova.

ROS podržava TCP/IP i UDP prijenos podataka. TCPROS je zadani komunikacijski protokol koji se koristi i jedini je prijenos podataka koje knjižice klijenta prema ROS-u trebaju podržavati.

2.3 Usluge

Model objavljivač/pretplatnik fleksibilna je komunikacijska paradigma, ali više-na-više jednosmjerni prijenos nije prikladan za RCP zahtjev/odgovor interakcija, koji su često potrebni u distribuiranim sistemima. Zahtjev/odgovor se obavlja putem usluga, koji su definirani parom poruka: jedna poruka za zahtjev, druga za odgovor. Klijent poziva uslugu slanjem poruke zahtjeva i čekanjem na odgovor. Usluge su definirane .srv datotekama.

Usluge, kao i teme, imaju svoj tip čije su izvorno ime paketa .srv datoteke. Kao i kod tema, njihova se verzija i podudarnost podataka izvršava MD5 sumom .srv datoteka. Čvorovi mogu koristiti pozive usluga ako se tip usluge i MD5 sume podudaraju [9].

2.4 Poruke

Čvorovi komuniciraju međusobno objavljivajući poruke temama. Poruka je jednostavna struktura podataka koja sadrži tipove podataka. Jednostavne tekstualne datoteke za specificiranje strukture podataka poruke su msg datoteke [10].

2.5 Torbe

Torbe su formati datoteke u ROS-u za spremanje podataka ROS-ovih poruka. Obično su stvorene od alata rosbag, čiji se čvor preplaćuje na jedan ili više ROS tema. Zatim spremaju serijalizirane podatke poruka u datoteku kako su generirani. Torbe su efikasne za snimanje i ponovno pokretanje programa. Te su poruke skladištene u istoj reprezentaciji koje su korištene u transportnom sloju ROS-a [11].

2.6 Master

Master pruža uslugu imenovanja i registracije ostatka čvorova u ROS sistemu. On prati usluge, objavljivače i pretplatnike raznih tema. Njegova je uloga da omogući individualnim ROS čvorovima međusobno prepoznavanje. Jednom kada su se čvorovi identificirali, njihova se komunikacija izvršava peer-to-peer.

ROS master pokrećemo komandom roscore, gdje bez njegovog pokretanja ne možemo pokrenuti ostale funkcije ROS-a [12].

2.7 Parametarski poslužitelj

Parametarski je poslužitelj dijeljeni, više-varijabilni rječnik kojemu se pristupa putem mrežnog API-ja. Korišten je od strane čvorova kako bi se spremili i dohvatili potrebni parametri prilikom njihovog izvršavanja. Kako nije napravljen za rad na visokim performansama, najbolje ga je koristiti kod statičnih, nebinarnih podataka poput konfiguracijskih parametara. Napravljen je da je globalno dostupan tako da je alatima lako dostupno konfiguracijsko stanje sistema i da ga modificiraju prema potrebi.

Parametarski je poslužitelj implementiran koristeći XML-RPC i pokreće se unutar ROS master-a [13].

2.8 Instalacija ROS-a

Instalacija je izvršena u linux mint-u mate 18.1 (xenial). Ugrađeni je nvidia-375 upravljački program (engl. *driver*). Korištena verzija ROS-a je kinetic te će sukladno tome biti pisane upute za instalaciju putem terminala.

Kako bi se omogućilo prihvaćanje izvora od OSRF-a (engl. *open source robotic foundation*) potrebno je konfigurirati repozitorije da se dopušta skidanje "restricted", "universe" i "multiverse" repositorija. Zatim postavljamo listu izvora:

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu xenial main" > /etc/apt/sources.list.d/ros-latest.list'

Zatim postavljamo ključeve naredbom:

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

Potrebno je ažuriranje softvera:

sudo apt-get update

Konačna se instalacija ROS-a izvršava slijedećom komandom:

sudo apt-get install ros-kinetic-desktop-full

Kako bi se mogle pokrenuti osnovne komponente ROS-a potrebno je izvršiti inicijalizaciju rosdep-a:

```
sudo rosdep init
rosdep update
```

Također se preporuča postaviti ROS-ove varijable okruženja u bash sesiju:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

3. GAZEBO

Gazebo je dinamički robotski simulator koji simulira robote u trodimenzionalnome okruženju. Također vrši simulaciju dinamičke interakcije između objekata. Prilikom pokretanja Gazeba pokreću se dvije različite izvršne datoteke, gzserver i gzclient.

Gzserver pokreće fizičko ažuriranu petlju i generaciju senzornih podataka. To je jezgra Gazeba koja se može koristiti neovisno o grafičkom korisničkom sučelju.

Gzclient izvršilac se pokreće na QT baziranom korisničkom sučelju. On pruža vizualizaciju simulacije i prikladne kontrole za razne postavke simulacije.

3.1 Komponente Gazeba

Gazebo se sastoji od slijedećih komponenti: world i model datoteka, varijabli okruženja, gazebo servera, grafičkog klijenta i dodataka [14].

3.1.1 World datoteke

Navedene datoteke sadrže sve elemente simulacije, uključujući robote, svijetla, senzore i statičke objekte. Formatiran je koristeći SDF (engl. *simulation description format*) i uobičajeno ima .world ekstenziju. Gzserver učitava tu datoteku kako bi generirao svijet [14].

3.1.2 Varijable okruženja

Gazebo koristi razne varijable okruženja kako bi mogao locirati datoteke i direktorije. Također postavlja komunikaciju između klijenta i poslužitelja. Za većinu slučajeva zadane su vrijednosti već postavljene. Varijablje koje Gazebo koristi su slijedeće: GAZEBO_MODEL_PATH, GAZEBO_RESOURCE_PATH, GAZEBO_MASTER_URI, GAZEBO_PLUGIN_PATH i GAZEBO_MODEL_DATABASE_URI [14]. Prema potrebi zadane vrijednosti možemo postaviti u .bashrc datoteci.

3.1.3 Gazebov poslužitelj

Poslužitelj parsira opisnu datoteku danu u komandnoj liniji gdje zatim simulira svijet koristeći fizičke i senzorne module [14].

3.1.4 Grafički klijent

Grafički klijent povezuje pokrenuti gzserver i vizualizira elemente. On je također alat kojim moduliramo pokrenutu simulaciju [14].

3.1.5 Dodatci

Dodatak je dio koda koji je kompiliran kao dijeljena knjižica i umetnut je u simulaciju. Dodatci imaju direktni pristup sveukupnoj funkcionalnosti Gazeba putem standardnih C++ klasa. Oni su korisni zato što omogućuju developeru da kontrolira gotovo svaki aspekt Gazeba, lako su djeljive rutine i mogu biti umetani i ukonjeni od pokrenutog sistema.

Trenutno postoji šest vrsti dodataka: svijet, model, senzor, sistem, vizualni dodatci i GUI (engl. *graphical user interface*). Svaki je dodatak upravljan drugačijim komponentima Gazeba [15].

3.2 Ovisnosti

Gazebo se sastoji od slijedećih ovisnosti: SDFormata, ignition-math, ignition-transport i ignition-msgs. One su sve održavane od strane Gazebo tima gdje su nove značajke u Gazebu povezane sa novim značajkama u tim knjižicama.

3.2.1 SDFormat

SDFormat je XML format koji opisuje objekte i okruženja za robotske simulatore, vizualizaciju i kontrolu. SDF je dizajniran u svrhu znanstvenih robotskih aplikacija. Gazebo koristi opisni simulacijski formatski (engl. *Simulation Description Format*) protokol za opis svakog aspekta simulacije.

3.2.2 Ignition Math

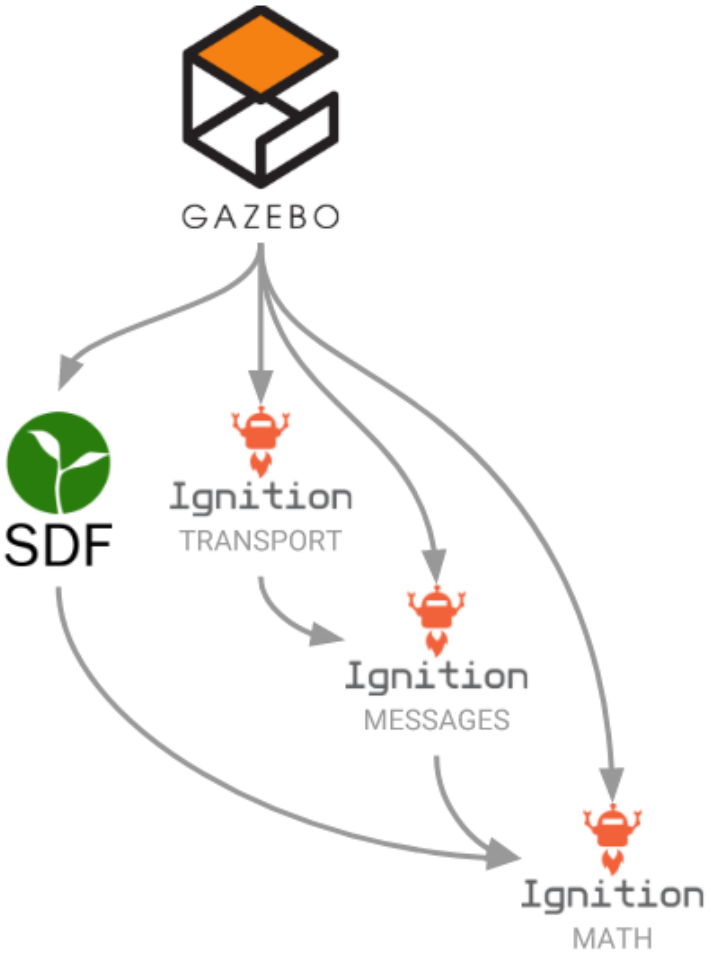
Ignition math je mala, brza matematička knjižica s visokim performansama. Ova knjižica sadrži samoodržavajući set klasa i funkcija prikladne za robotske aplikacije.

3.2.3 Ignition transport

Transportna knjižica kombinira ZeroMQ sa Protobufima kako bi se stvorio brzi i efikasni sustav slanja poruka. Pruženo je asinkrono objavljivanje i potplaćivanje poruka zajedno sa servisnim pozivima i otkrivanjima.

3.2.4 Ignition messages

To je standardni set definicija poruka koje su korištene od strane ignition transporta, i drugih aplikacija kojima su potrebne.
Na slici 3.1 prikazane su knjižice i njihove međuovisnosti potrebne za kompilaciju Gazeba:



Slika 3.1: Knjižice potrebne za kompilaciju Gazeba [5]

3.3 Instalacija

Kako bi omogućili OculusVR ekstenziju za Gazebo, trebamo ukloniti ROS-ove pakete Gazeba kao i druge njegove knjižice povezane s njim. Kako većina ROS-ovih knjižica nisu kompatibilne sa željenom verzijom Gazeba [4], ekstenzija nam za Oculus Rift u protivnome neće biti funkcionalna:

```
sudo apt-get remove '.*gazebo.*' '.*sdformat.*' '.*ignition-math.*' '.*ignition-msgs.*' '.*ignition-transport.*'
```

Instalacija ignition math knjižice izvršava se:

- Vršimo instalaciju potrebnih paketa:

```
sudo apt-get install build-essential \cmake \mercurial
```

- Kloniramo repozitorij:

```
hg clone https://bitbucket.org/ignitionrobotics/ign-math /tmp/ign-math
cd /tmp/ign-math
```

- Kako koristimo Gazebo verzije 8 potrebna nam je ignition math 3 knjižica:

```
hg up ign-math3
```

- Zatim izrađujemo:

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/$USER/local ../
make -j6
sudo make install
```

Izgradnja i instalacija SDFormata od izvora se izvršava slijedećim naredbama:

```
sudo apt-get install build-essential \cmake \mercurial \python \libboost-system-dev \libboost-filesystem-dev \libboost-program-options-dev \libboost-regex-dev \libboost-iostreams-dev \libtinyxml-dev \libxml2-utils \ruby-dev \ruby
hg clone https://bitbucket.org/osrf/sdformat /tmp/sdformat
cd /tmp/sdformat
hg up sdf5
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/$USER/local ../
make -j6
sudo make install
```

Gradnja i instalacija ignition poruka:

```
sudo apt-get install build-essential \cmake \mercurial \libprotoc-dev \libprotobuf-dev \protobuf-compiler
hg clone https://bitbucket.org/ignitionrobotics/ign-msgs /tmp/ign-msgs
cd /tmp/ign-msgs
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/$USER/local ../
make -j6
sudo make install
```

I u konačnici zadnja ignition transport knjižica:

```
sudo apt-get install mercurial cmake pkg-config python ruby-ronn libprotoc-dev libprotobuf-dev protobuf-compiler uuid-dev libzmq3-dev libignition-msgs-dev
hg clone https://bitbucket.org/ignitionrobotics/ign-transport /tmp/ign-transport
cd /tmp/ign-transport
hg up ign-transport3
mkdir build
cd build
```

```
cmake -DCMAKE_INSTALL_PREFIX=/home/$USER/local ../
make -j6
sudo make install
```

Izgradnja i instalacija Gazeba:

```
hg clone https://bitbucket.org/osrf/gazebo /tmp/gazebo
cd /tmp/gazebo
hg up gazebo8
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/$USER/local ../
make -j6
sudo make install
```

Na kraju je potrebno podesiti variable okruženja kako bi nam ROS mogao pronaći izgrađenu verziju Gazeba:

```
echo "export LD_LIBRARY_PATH=<install_path>/local/lib:$LD_LIBRARY_PATH" >> ~/.bashrc
echo "export PATH=<install_path>/local/bin:$PATH" >> ~/.bashrc
echo "export PKG_CONFIG_PATH=<install_path>/local/lib/pkgconfig:$PKG_CONFIG_PATH" >> ~/.bashrc
source ~/.bashrc
```

4. SIMULACIJA NA ZASLONU ZA VIRTUALNU STVARNOST

Kako bi omogućili ekstenziju za Oculus Rift trebamo instalirati OculusVR SDK:

```
sudo apt-get install libusb-dev libudev-dev libxinerama-dev
hg clone https://bitbucket.org/osrf/oculussdk
cd oculussdk
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/usr
make
sudo make install
sudo cp ../LibOVR/90-oculus.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
```

Prije pokretanja Gazeba, trebamo podesiti otvaranje novog prozora koji će biti renderiran od strane Oculus Rift-a. U postavkama zaslona podesimo Oculus Rift. Zatim je potrebno modificirati *~/.gazebo/gui.ini* datoteku:

```
[geometry]
x=0
y=0

[oculus]
x=<ZAMIJENITI_HORIZONTALNOM_REZOLUCIJOM_MONITORA>
y=0
visual=<ZAMIJENITI_VIZUALNIM_LINKOM_PRIKLJUČENIM_ZA_OCULUS>
autolanch=0
```

4.1 Pokretanje programa

Cilj nam je pokrenuti simulator Gazebo preko ROS-a, gdje prvenstveno pokrećemo ROS naredbom *roscore* (slika 4.1) kako bi omogućili komunikaciju između čvorova:

```
zuzili@zuzili-AB350-Gaming ~ $ roscore
... logging to /home/zuzili/.ros/log/7034241c-8728-11e7-b17b-1c1b0d978378/roslaunch-zuzili-AB350-G
aming-6225.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://zuzili-AB350-Gaming:41852/
ros_comm version 1.12.7


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [6239]
ROS_MASTER_URI=http://zuzili-AB350-Gaming:11311/

setting /run_id to 7034241c-8728-11e7-b17b-1c1b0d978378
process[rosout-1]: started with pid [6252]
started core service [/rosout]
```

Slika 4.1: Izvadak od naredbe roscore

Tom se naredbom pokreće ROS master. Ono što je vidljivo na slici je koja je verzija ROS-a korištena i koji je ID procesa. Slijedeće što nam je potrebno je otvoriti novi terminal, pomaknuti se u *oculussdk* direktorij i pokrenuti *oculussdk*:

```
zuzili@zuzili-AB350-Gaming ~ $ cd oculussdk/
zuzili@zuzili-AB350-Gaming ~/oculussdk $ ./oculusd
OVR::DeviceManagerThread - running (ThreadId=0x7ff9422f8700).
OVR::DeviceManager - initialized.
OVR::Linux::HIDDevice - Opened '/dev/hidraw1'
                    Manufacturer:'Oculus VR, Inc.'  Product:'Rift DK2'  Serial#:'H1DE47R6N1E2KD00E
100'
[TrackingManager] Sensor added: PrintedSerial=206ZA509DJGR UUID=H1DE47R6N1E2KD00E100
[TrackingManager] Display added: UUID=H1DE47R6N1E2K Driver=0
Timestamp 0 rollover, was: 4294966140, now: 844
Timestamp 0 rollover, was: 4294966497, now: 1202
Timestamp 0 rollover, was: 4294965932, now: 636
Timestamp 0 rollover, was: 4294966977, now: 1682
Timestamp 0 rollover, was: 4294965578, now: 282
Timestamp 0 rollover, was: 4294966985, now: 1690
Timestamp 0 rollover, was: 4294966937, now: 1641
Timestamp 0 rollover, was: 4294965906, now: 611
Timestamp 0 rollover, was: 4294966541, now: 1245
Timestamp 0 rollover, was: 4294966892, now: 1597
Timestamp 0 rollover, was: 4294965979, now: 683
[TrackingManager] Display removed: UUID=H1DE47R6N1E2K Driver=0
[TrackingManager] Display added: UUID=H1DE47R6N1E2K Driver=0
Timestamp 0 rollover, was: 4294965348, now: 53
Timestamp 0 rollover, was: 4294965579, now: 283
Timestamp 0 rollover, was: 4294965310, now: 15
Timestamp 0 rollover, was: 4294966309, now: 1013
Timestamp 0 rollover, was: 4294966282, now: 986
Timestamp 0 rollover, was: 4294966057, now: 761
```

Slika 4.2: Izvadak naredbe ./oculusd

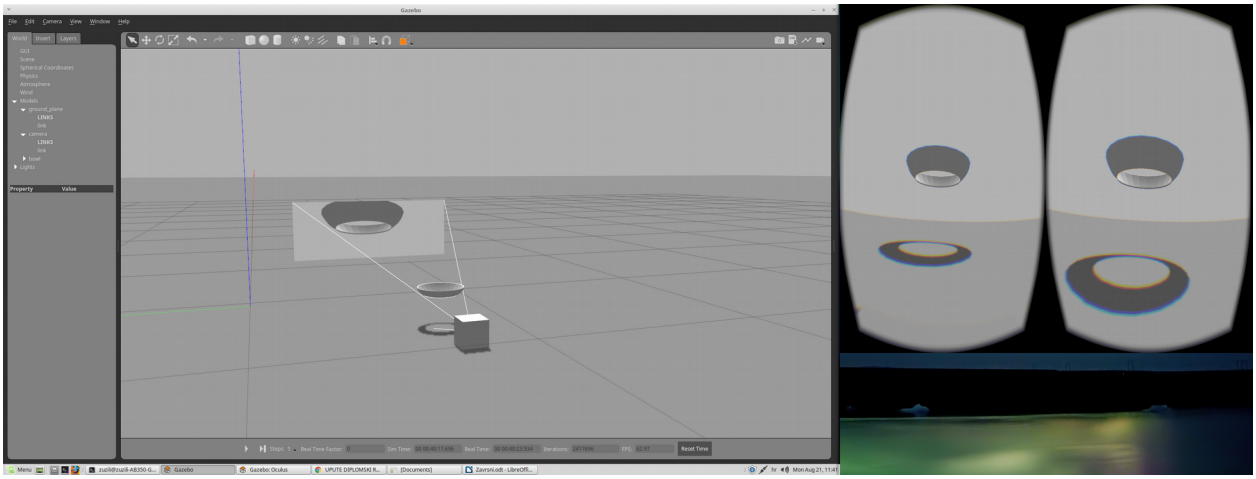
Program prepoznaje Oculus Rift. Na slici 4.2 se također vidi kako program prati ponašanje Oculus Rift-a. Preko tracking manager-a vidimo kada je zaslon priključen, odnosno isključen, za računalo.

Ponovno pokrećemo još jedan terminal te pokrećemo Gazebo naredbom (slika 4.3):

```
zuzili@zuzili-AB350-Gaming ~/oculussdk $ rosrun gazebo_ros gazebo
[ INFO] [1503653876.303571412]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1503653876.304120197]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO] [1503653877.469242431]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[ INFO] [1503653877.523674924]: Physics dynamic reconfigure ready.
```

Slika 4.3: Izvadak naredbe rosrun gazebo_ros gazebo

U konačnici nam je prikaz osnovne simulacije sljedeći (slika 4.4):



Slika 4.4: Prikaz simulacije u Gazebu i Oculus Rift-u

Prilikom pokretanja Gazeba putem ROS-a pokrenuli su se razni čvorovi od strane Gazeba što vidimo na slici 4.6 pokrečući naredbu *rosnode info /gazebo*. Ovim putem vidimo koji su nam objavljeni čvorovi, koji su nam pretplaćeni te koje su usluge pokrenute.

Na slici 4.5 su nam prikazane pokrenute teme.

```
zuzili@zuzili-AB350-Gaming ~/catkin_ws $ rostopic list
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/rosout
/rosout_agg
```

Slika 4.5: Prikaz ROS tema

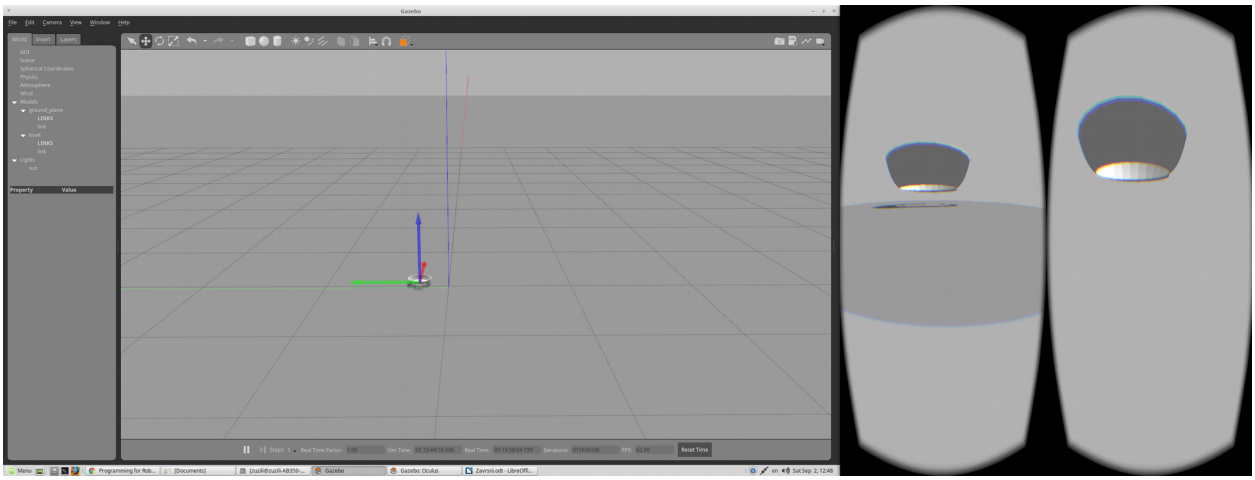
Od strane Gazeba pokrenuto je šest tema. Putem tema izvršava se komunikacija između čvorova.

Na slici 4.6 vidljivo je koje objavljivače i pretplatnike koristi čvor */gazebo*. Objavljuje na šest različitih čvorova, a pretplačeni je na tri čvora. Na slici je vidljivo s kojim uslugama komunicira. Također je vidljivo da se putem ROS-a izvršava komunikacija putem zadanog TCPROS komunikacijskog protokola.

```
zuzili@zuzili-AB350-Gaming ~ $ rosnode info /gazebo
--------------------------------------------------------------------------------
Node [/gazebo]
Publications:
 * /gazebo/link_states [gazebo_msgs/LinkStates]
 * /rosout [rosgraph_msgs/Log]
 * /gazebo/model_states [gazebo_msgs/ModelStates]
 * /clock [rosgraph_msgs/Clock]
 * /gazebo/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 * /gazebo/parameter_updates [dynamic_reconfigure/Config]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]
 * /gazebo/set_model_state [unknown type]
 * /gazebo/set_link_state [unknown type]

Services:
 * /gazebo/apply_joint_effort
 * /gazebo/get_physics_properties
 * /gazebo/set_link_state
 * /gazebo/set_joint_properties
 * /gazebo/set_logger_level
 * /gazebo/reset_world
 * /gazebo/set_model_configuration
 * /gazebo/get_world_properties
 * /gazebo/delete_light
 * /gazebo/set_parameters
 * /gazebo/spawn_sdf_model
 * /gazebo/set_model_state
 * /gazebo/unpause_physics
 * /gazebo/get_loggers
 * /gazebo/get_light_properties
 * /gazebo/clear_body_wrenches
 * /gazebo/pause_physics
 * /gazebo/set_physics_properties
 * /gazebo/set_light_properties
 * /gazebo/reset_simulation
 * /gazebo/delete_model
 * /gazebo/spawn_urdf_model
 * /gazebo/set_link_properties
 * /gazebo/get_joint_properties
 * /gazebo/apply_body_wrench
 * /gazebo/get_link_state
 * /gazebo/get_link_properties
 * /gazebo/get_model_state
 * /gazebo/clear_joint_forces
 * /gazebo/get_model_properties


contacting node http://zuzili-AB350-Gaming:39867/ ...
Pid: 7179
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo
    * direction: outbound
    * transport: INTRAPROCESS
 * topic: /clock
    * to: /gazebo (http://zuzili-AB350-Gaming:39867/)
    * direction: inbound
    * transport: INTRAPROCESS
```

Slika 4.6: Izvadak naredbe rosnode info /gazebo

Izvršit ćemo detaljniju analizu simulacije prikazane na slici 4.7.



Slika 4.7: Pokrenuta simulacija "bowl" u Gazebu i prikaz simulacije u Oculus Rift-u

Naredba *rostopic echo /gazebo/model_states* nam daje informacije o modelu "bowl", njegove pozicije i dimenzijama što je vidljivo na slici 4.8:

```
---
name: ['ground_plane', 'bowl']
pose:
  -
    position:
      x: 0.0
      y: 0.0
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
  -
    position:
      x: 0.0953787951986
      y: 0.243015653225
      z: 0.0174910227309
    orientation:
      x: -6.59982128914e-07
      y: -1.13150290593e-06
      z: -3.59629280895e-08
      w: 0.999999999999
twist:
  -
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  -
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
---
```

Slika 4.8: Informacije o simulaciji "bowl"

Na slici 4.9 izvadak je naredbe *rostopic echo /gazebo/parameter_descriptions*. Vidljiv nam je opis varijabli, njihov tip i pripadnih vrijednosti:

```
  groups:
    -
      name: Default
      state: True
      id: 0
      parent: 0
dflt:
  bools:
    -
      name: auto_disable_bodies
      value: False
  ints:
    -
      name: sor_pgs_precon_iters
      value: 0
    -
      name: sor_pgs_iters
      value: 20
    -
      name: max_contacts
      value: 100
  strs: []
  doubles:
    -
      name: time_step
      value: 0.001
    -
      name: max_update_rate
      value: 60.0
    -
      name: gravity_x
      value: 0.0
    -
      name: gravity_y
      value: 0.0
    -
      name: gravity_z
      value: -9.8
    -
      name: sor_pgs_w
      value: 1.3
    -
      name: sor_pgs_rms_error_tol
      value: -1.0
    -
      name: cfm
      value: 0.0
    -
      name: erp
      value: 0.2
    -
      name: contact_surface_layer
      value: 0.001
    -
      name: contact_max_correcting_vel
      value: 100.0
  groups:
    -
      name: Default
      state: True
      id: 0
      parent: 0
---
```

Slika 4.9: Opis varijabli, tip i vrijednosti za simulaciju "bowl"

Na slici 4.10 i 4.11 prikazan je model *my_verhicle*, kao i pogled simulacije na Oculus Rift-u. Vozilo je kreirano od dvije datoteke. Jedna je datoteka sa ekstenzijom .config, dok druga sa

ekstenzijom .sdf. Obje su datoteke stavljene u jedan direktorij i taj je direktorij premješten u direktorij *.gazebo/models*.

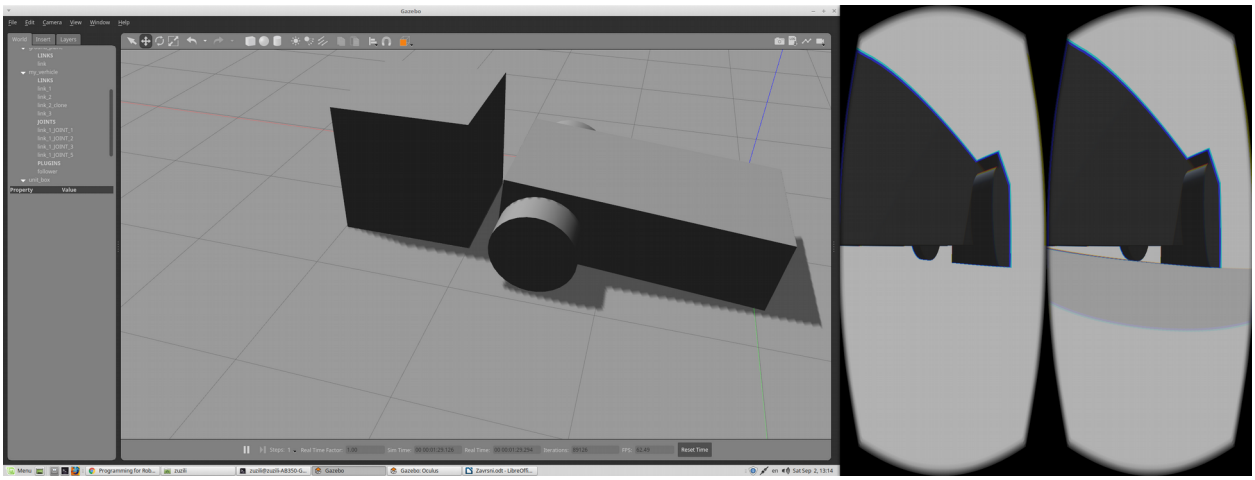
.config datoteka nam služi za lociranje prikladne SDF datoteke i tekstualnog opisa modela koji sadrži .sdf datoteka (model.sdf) [18]. Sadržaj je datoteke .config (model.config) slijedeći:

```
<?xml version="1.0" ?>
<model>
   <name>my_verhicle</name>
   <version>1.0</version>
   <sdf version="1.6">model.sdf</sdf>
   <author>
      <name></name>
      <email></email>
   </author>
   <description></description>
</model>
```

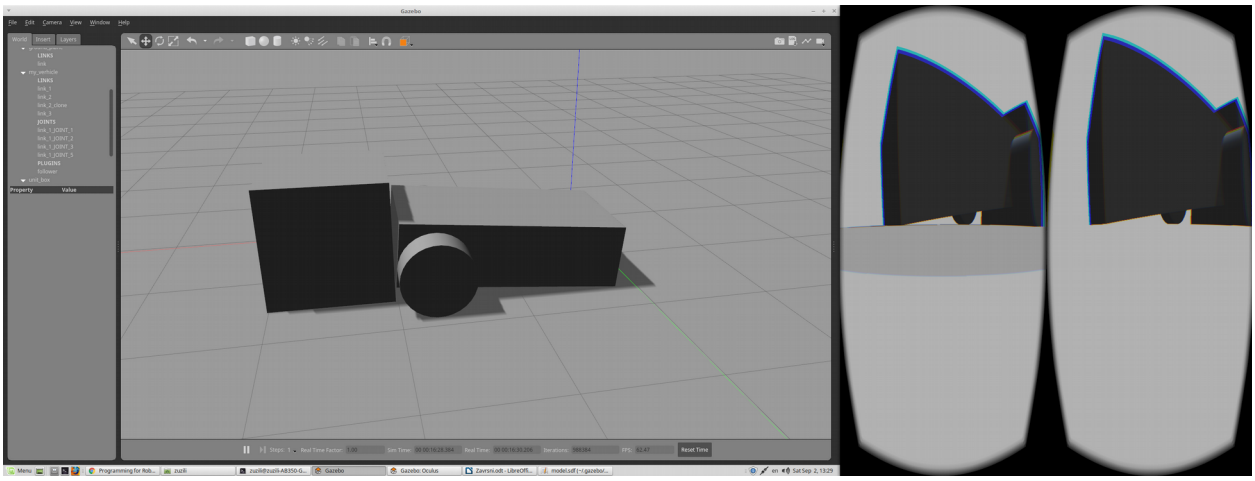
Izdvojit ćemo liniju koda .sdf datoteke (model.sdf):

```
   <plugin name='follower' filename='libFollowerPlugin.so'/>
```

Ova je linija koda vidljiva na kraju datoteke *model.sdf*. Dodatkom pod nazivom "follower" se omogućila kretnja vozila. Navedena se ekstenzija kreira u Gazebo simulatoru. Parametri su se model.sdf datoteke automatski stvorili kreirajući model u simulatoru Gazebo. Povezani (engl. *linked*) su kotači sa kutijom i senzorom koji je postavljen unutar samoga vozila. Na slici 4.10 prikazana je početna simulacija, a na slici 4.11 gotova simulacija.



Slika 4.10: Početak simulacije “my_verhicle”



Slika 4.11: Kraj simulacije “my_verhicle”

Za ovu simulaciju drugačije se ponašaju outputi komande *rostopic echo /gazebo/model_states* (slika 4.12)*:*

```
      z: 0.0
---
name: ['ground_plane', 'my_verhicle', 'unit_box']
pose:
  -
    position:
      x: 0.0
      y: 0.0
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
  -
    position:
      x: 1.14215998241
      y: 1.11696846092
      z: 0.302278264659
    orientation:
      x: -0.000101202383802
      y: -0.0433050850112
      z: -0.00233560943631
      w: 0.999059159559
  -
    position:
      x: 2.49997113734
      y: 1.01115441067
      z: 0.499995109439
    orientation:
      x: -4.89056293579e-06
      y: -4.47410219656e-11
      z: 5.92085187791e-12
      w: 0.999999999988
twist:
  -
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  -
    linear:
      x: 0.0014208053719
      y: -0.00140696123566
      z: 0.00438997357804
    angular:
      x: 0.00398103655244
      y: 0.00411386089542
      z: -2.34904780259e-05
  -
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
---
```


Slika 4.12: Brojčane vrijednosti simulacije "my_verhicle"

Konstantno se mijenjaju vrijednosti parametara x,y i z pozicije te x, y, z i w orijentacije. Također postoje razne varijacije brojeva gledajući okret (engl. *twist*). Gledajući drugi redak linear i angular mijenjaju nam se vrijednosti parametara ovisno o pokrenutoj simulaciji.

4.2 Performanse računala

Završni je rad napravljeni na računalu sa slijedećim specifikacijama:

- procesor: AMD Ryzen 5 1600 šest-jezgreni procesor
- grafička kartica: gigabyte GTX 1060 GDDR5 3GB
- matična kartica: gigabyte B350-Gaming AM4
- RAM: patriot viper4, 2800Mhz 2x8GB

- memorija: siscon power S55 240GB SSD

Putem *xrandr* komande (slika 4.13) vidimo koji su nam hardveri priključeni za računalo. Spojena su nam dva izlaza: HDMI-0 i DVI-D-1. Na HDMI-0 nam je priključeni monitor rezolucije 1080x1920, a na DVI-D-1 nam je spojeni Oculus Rift rezolucije 1920x1080. Frekvencija osvježavanja slike HDMI-0 iznosi 75 Hz, dok DVI-D-1 60 Hz. Zajednička im rezolucija iznosi 2868x1080. Bitna nam je rezolucija monitora. Prisjetimo se podešavanja *~/.gazebo/gui.ini* datoteke. Pod X vrijednost postavljena je 1920.

```
zuzili@zuzili-AB350-Gaming ~ $ xrandr
Screen 0: minimum 8 x 8, current 2868 x 1080, maximum 32767 x 32767
DVI-D-0 disconnected (normal left inverted right x axis y axis)
HDMI-0 connected 948x1080+1920+0 left (normal left inverted right x axis y axis) 71mm x 126mm
   1080x1920     75.00 +  72.00    60.00
   1080x948     119.99*
DP-0 disconnected (normal left inverted right x axis y axis)
DP-1 disconnected (normal left inverted right x axis y axis)
DVI-D-1 connected primary 1920x1080+0+0 (normal left inverted right x axis y axis) 640mm x 360mm
   1920x1080     60.00*+  59.94    50.00    29.97    25.00    24.00    23.97    60.00    50.04
   1360x768      59.80
   1280x1024     60.02
   1280x720      59.94    50.00
   1024x768      60.00
   800x600       60.32
   720x576       50.00
   720x480       59.94
   640x480       59.94    59.93
```

Slika 4.13: Izvadak naredbe xrandr

Putem *top* naredbe na slici 4.14 vidimo koliko nam pokrenuti procesi iskorištavaju procesor. Očigledno Gazebo iskorištava najveći postotak procesora. Znamo da se Gazebo pokreće pomoću naredbi *gzserver* i *gzclient* koji zajedno iskorištavaju veliki udio procesorske snage, odnosno u prosjeku oko 65%. *Oculusd* i *roscore* daleko su manje zahtjevniji procesi gdje *oculusd* iskorištava 2,0% procesora, a *roscore* samo 0,3%.

```
top - 11:04:34 up 1 day, 14:59,  1 user,  load average: 0,85, 1,03, 0,89
Tasks: 303 total,   1 running, 302 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5,6 us,  0,8 sy,  0,0 ni, 93,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  8190284 total,  2381016 free,  3687288 used,  2121980 buff/cache
KiB Swap:  8338428 total,  8338428 free,        0 used.  4035712 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3753 zuzili    20   0 3740836 369076 143984 S  49,5  4,5 631:23.14 gzclient
 3752 zuzili    20   0 5186384 314516 117560 S  16,6  3,8 376:47.67 gzserver
 1426 root      20   0  454936  90116  67964 S   6,0  1,1 200:11.31 Xorg
 3617 zuzili    20   0  241844   7636   5156 S   2,0  0,1  43:12.31 oculusd
 1431 root     -51   0       0      0      0 S   1,3  0,0 203:57.73 irq/64-nvidia
 1722 zuzili    20   0  469420  34248  22164 S   1,0  0,4  33:02.57 marco
 1748 zuzili     9 -11  438652  13636  10336 S   1,0  0,2  31:17.69 pulseaudio
 5322 zuzili    20   0 1565020 502084  80848 S   0,7  6,1  20:04.13 chrome
 2350 zuzili    20   0 1494136 371204 132904 S   0,3  4,5  42:02.41 chrome
 2480 zuzili    20   0  839048  81396  56740 S   0,3  1,0   0:02.82 chrome
 2729 zuzili    20   0  573884  39024  28508 S   0,3  0,5   0:01.80 mate-terminal
 3483 zuzili    20   0  325212  56608   7872 S   0,3  0,7   2:25.80 roscore
17019 root      20   0       0      0      0 S   0,3  0,0   0:00.48 kworker/5:1
27624 root      20   0       0      0      0 S   0,3  0,0   0:01.08 kworker/2:0
29316 root      20   0       0      0      0 S   0,3  0,0   0:00.16 kworker/1:3
29430 zuzili    20   0   43288   4176   3300 R   0,3  0,1   0:00.56 top
```

Slika 4.14: Izvadak naredbe top

Na slici 4.15 i slici 4.16 vidljiva je iskorištenost CPU-a (engl. *central processing unit*) prema dretvama:

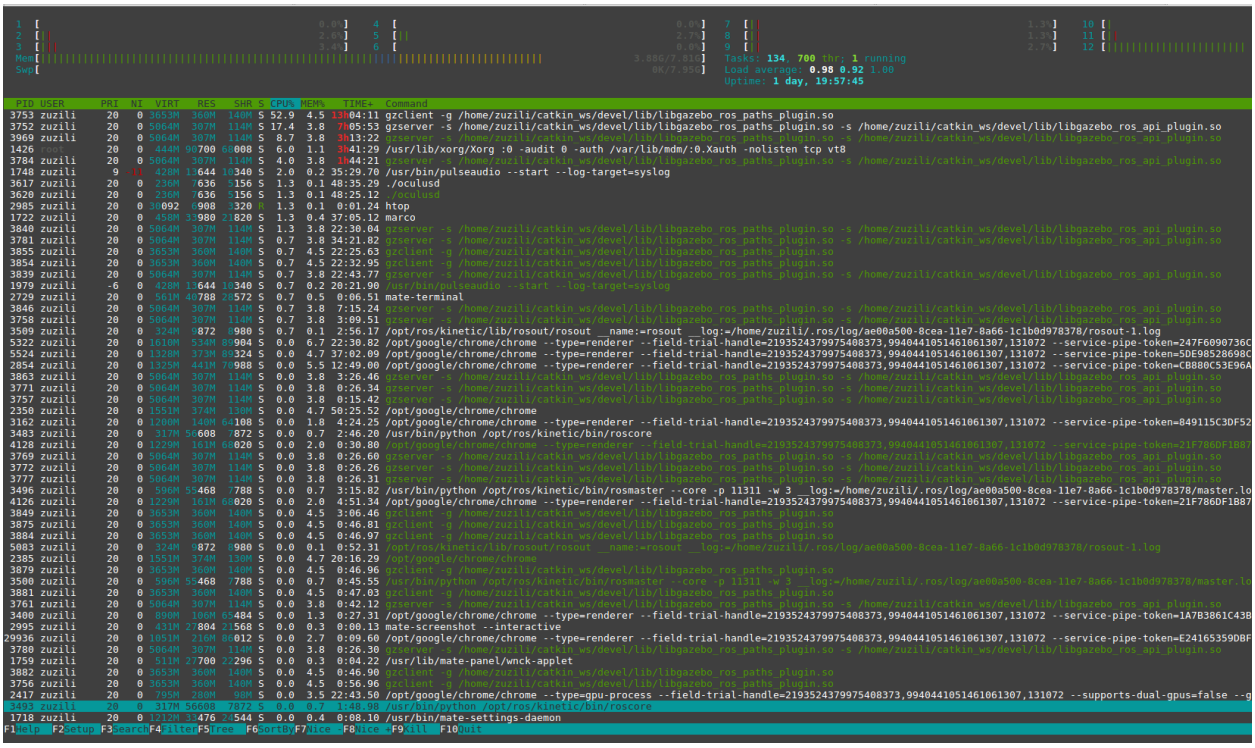
```
nmon-14g-------[H for help]---Hostname=zuzili-AB350-Refresh= 2secs ---19:22.42
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50         |75       100|
  1   1.5   1.5   0.0  97.0|                        >                         |
  2   0.5   2.5   0.0  97.0|S              >                                 |
  3   3.9   3.9   0.0  92.1|US                                   >           |
  4   0.5   0.5   0.0  99.0|                                         >       |
  5   1.0   0.5   0.0  98.5|                           >                     |
  6   0.5   1.0   0.0  98.5|                   >                             |
  7   3.5   1.0   0.0  95.5|U                        >                        |
  8  28.4   0.5   0.0  71.1|UUUUUUUUUUUUUU         >                        |
  9   8.2   1.9   0.0  89.9|UUUU       >                                     |
 10   3.0   1.0   0.0  96.0|U                >                               |
 11   2.5   1.0   0.0  96.5|U                  >                             |
 12  22.4   0.0   0.0  77.6|UUUUUUUUUUU          >                          |
---------------------------+-------------------------------------------------+
Avg   6.4   1.3   0.0  92.3|UUU        >                                     |
---------------------------+-------------------------------------------------+
```

Slika 4.15: Iskorištenost procesora prema dretvama

```
nmon-14g-------[H for help]---Hostname=zuzili-AB350-Refresh= 2secs ---19:23.26
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50         |75       100|
  1   3.5   1.0   0.0  95.5|U                        >                         |
  2   2.0   1.5   0.0  96.5|U              >                                 |
  3  19.1   2.5   0.0  78.4|UUUUUUUUUS                           >           |
  4   3.4   1.0   0.0  95.6|U                                         >       |
  5   8.4   2.0   0.0  89.6|UUUU                           >                 |
  6   1.0   1.5   0.0  97.5|                   >                             |
  7  14.9   1.5   0.0  83.6|UUUUUUU                  >                       |
  8   3.5   0.5   0.0  96.0|U                        >                        |
  9   6.1   1.5   0.5  91.9|UUU        >                                     |
 10  15.6   0.5   0.0  83.9|UUUUUUU            >                             |
 11   5.6   1.5   0.0  92.8|UU                 >                             |
 12   3.0   0.5   0.0  96.4|U                        >                        |
---------------------------+-------------------------------------------------+
Avg   7.2   1.3   0.0  91.5|UUU        >                                     |
---------------------------+-------------------------------------------------+
```

Slika 4.16: Iskorištenost procesora prema dretvama

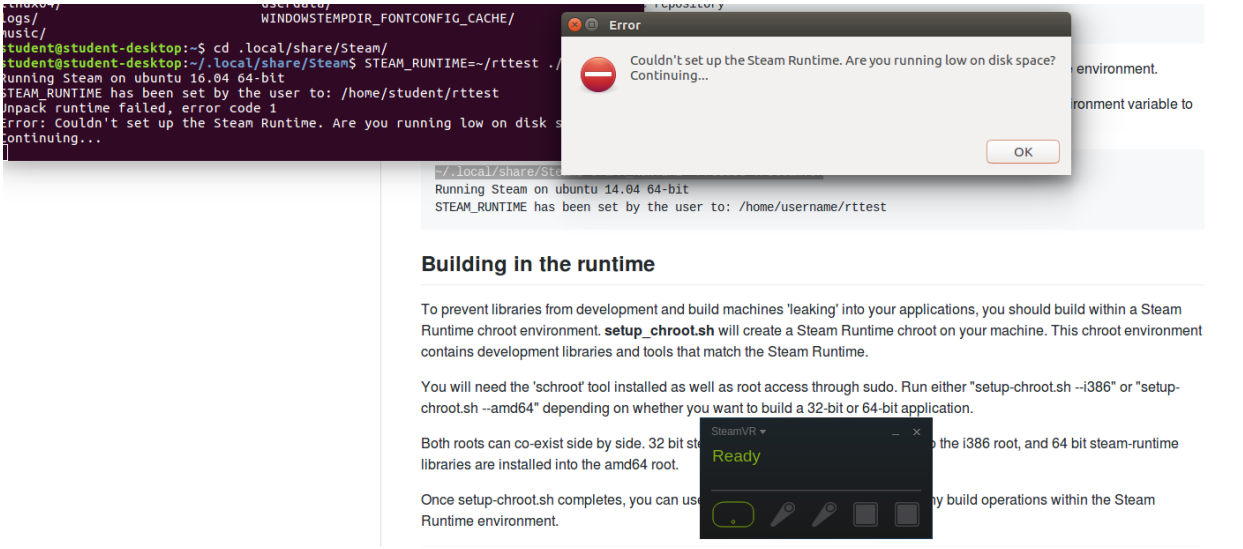
Na slici 4.17 prikazani je detaljniji opis svakoga procesa i njihova iskorištenost prema dretvama (prva, druga i treća linija).



Slika 4.17: Detaljni opis pokrenutih procesa

4.3 Povezanost simulatora s HTC Vive-om

Vive je razvijen od strane HTC-a i Valve-a. Pokreće se putem SteamVR-a, programa koji razvija kompanija Valve. Napravljena je podrška Steam-a za linux, ali pokretanje SteamVR-a u linuxu trenutno je još uvijek u razvojnoj fazi [19]. Stoga povezivanje HTC Vive-a sa simulatorom je prošlo neuspješno kako se Steam-ovo razvojno okruženje, koje nam je potrebno za ovaj rad, nije uspješno pokrenuto (slika 4.18) makar ima dovoljno slobodnog prostora u memoriji.



Slika 4.18: Neuspješno pokretanje Steam razvojnog okruženja

Također vrijedi uočiti razliku između Oculus Rift-a i HTC Vive-a. Naime, putem izvadka naredbe *xrandr* (slika 4.13) vidljivo je kako računalo prepoznaje Oculus Rift. Spojimo li HTC Vive, putem komande *xrandr* uočimo da nije vidljiv kao izlaz (slika 4.19). Jedino monitor prepoznaje kao izlaz (DVI-D-0).

```
student@student-desktop:~$ xrandr
žScreen 0: minimum 8 x 8, current 1920 x 1080, maximum 32767 x 32767
DVI-D-0 connected primary 1920x1080+0+0 (normal left inverted right x axis y axis) 477mm x 268mm
   1920x1080     60.00*+
   1680x1050     59.95
   1440x900      74.98    59.89
   1280x1024     75.02    60.02
   1280x720      60.00
   1024x768      75.03    60.00
   800x600       75.00    60.32
   640x480       75.00    72.81    59.94
HDMI-0 disconnected (normal left inverted right x axis y axis)
DP-0 disconnected (normal left inverted right x axis y axis)
DP-1 disconnected (normal left inverted right x axis y axis)
DVI-D-1 disconnected (normal left inverted right x axis y axis)
```

Slika 4.19: Izvadak naredbe xrandr za HTC Vive

Tokom rada na završnom radu naišla sam na problem gdje računalo nije prepoznalo Oculus Rift. Naime, na računalu su bili dvojno instalirani windows-i i linux. Kako je u jednom trenutku došlo do disfunkcionalnosti linux-a zbog greške u nvidia-inom upravljačkom programu, ponovno je instaliran operacijski sustav linux te izbrisana particija na kojoj su bili postavljeni windows-i. Tada je računalo bilo u mogućnosti prepoznati Oculus Rift kao dodatni izlaz. Kako je HTC Vive bio spojen na računalo na kojemu su dvojno instalirana dva operacijska sustava (windows-i i

linux), postoji mogućnost da je navedeno isti razlog zbog kojega računalo ne prepoznaje HTC Vive kao dodatni izlaz. Unatoč tome sa slike je vidljivo da Steam prepoznaje Vive. Time nije u potpunosti ostvarena svrha ovog rada zato što nam je potrebno da nam računalo prepoznaje zaslon za virtualnu stvarnost kao izlaz.

5. ZAKLJUČAK

Tema je ovog završnoga rada rad s robotskim simulatorima pomoću zaslona za virtualnu stvarnost. Cilj je bio testirati zaslon za virtualnu stvarnost i pokrenuti ga kao sučelje za simulator, što je i realizirano. Također je postignuto povezivanje simulatora s ROS-ovim middleware-om.

Osjećaj uronjenosti i performanse na zaslonu za virtualnu stvarnost pomoću robotskog simulatora Gazeba, dovoljno je dobro postignut. Kroz zaslon Oculus Rift vidljiva je simulacija, no simbioza dvaju slika na kamerama, odnosno spajanje dvaju objekata u jedan, nije u potpunosti postignuta budući da je na drugoj kameri kut gledanja na simulaciji veći nego što je potreban. Izvedeni pokušaj povezivanja zaslona HTC Vive-a na žalost nije bio uspješan.

U konačnici ovaj je rad rezultirao uspješnim povezivanjem robotskog simulatora sa ROS-ovim middleware-om i prikazom simulacije na zaslonu za virtualnu stvarnost Oculus Rift. Može poslužiti kao dobra osnova za daljni rad i testiranje drugih robotskih simulatora u budućnosti.

6. LITERATURA

[1] S interneta, https://www.vrs.org.uk/virtual-reality/history.html, 20. kolovoza 2017.

[2] S interneta, http://www.robotiksistem.com/robotics_history.html, 20. kolovoza 2017.

[3] S interneta, http://wiki.ros.org/ROS/Introduction, 21. kolovoza 2017.

[4] S interneta, http://gazebosim.org/tutorials?tut=install_from_source&cat=install, 21.kolovoza 2017.

[5] S interneta, http://gazebosim.org/tutorials?tut=install_dependencies_from_source, 21. kolovoza 2017.

[6] Metz, R.: "Controlling VR with your mind", s interneta, https://www.technologyreview.com/s/603896/controlling-vr-with-your-mind/, 21. kolovoza 2017.

[7] Wagner, K.: "Two years later: Facebook's Oculus acquisition has changed virtual reality forever", s interneta, https://www.recode.net/2016/3/24/11587234/two-years-later-facebooks-oculus-acquisition-has-changed-virtual, ustupljeno 21. kolovoza 2017.

[8] S interneta, http://wiki.ros.org/Nodes, 22. kolovoza 2017.

[9] S interneta, http://wiki.ros.org/Services, 22. kolovoza 2017.

[10] S interneta, http://wiki.ros.org/Messages, 22. kolovoza 2017.

[11] S interneta, http://wiki.ros.org/Bags, 22. kolovoza 2017.

[12] S interneta, http://wiki.ros.org/Master, 22. kolovoza 2017.

[13] S interneta, http://wiki.ros.org/Parameter%20Server, 22. kolovoza 2017.

[14] S interneta, http://gazebosim.org/tutorials?tut=components&cat=get_started, 24. kolovoza 2017.

[15] S interneta, http://gazebosim.org/tutorials/?tut=plugins_hello_world, 24. kolovoza 2017.

[16] Billinghurst, M.: "Introduction to Virtual Reality" s interneta, https://www.slideshare.net/marknb00/comp-4010-lecture1-introduction-to-virtual-reality, 25. kolovoza 2017.

[17] S interneta, http://wiki.ros.org/catkin/workspaces, 1. rujna 2017.

[18] S interneta, http://gazebosim.org/tutorials?tut=model_contrib, 3. rujna 2017.

[19] S interneta, https://github.com/ValveSoftware/SteamVR-for-Linux, 5. rujna 2017.

7. POPIS SLIKA

Slika 1.1: Jedan od prvih zaslona za virtualnu stvarnost za glavu....1
Slika 3.1: Knjižice potrebne za kompilaciju Gazeba....11
Slika 4.1: Izvadak od naredbe *roscore*....16
Slika 4.2: Izvadak naredbe *./oculusd*....17
Slika 4.3: Izvadak naredbe *rosrun gazebo_ros gazebo*....17
Slika 4.4: Prikaz simulacije u Gazebu i Oculus Rift-u....17
Slika 4.5: Prikaz ROS tema....18
Slika 4.6: Izvadak naredbe *rosnode info /gazebo*....19
Slika 4.7: Pokrenuta simulacija "bowl" u Gazebu i prikaz simulacije u Oculus Rift-u....20
Slika 4.8: Informacije o simulaciji "bowl"....20
Slika 4.9: Opis varijabli, tip i vrijednosti za simulaciju "bowl"....21
Slika 4.10: Početak simulacije "my_verhicle"....23
Slika 4.11: Kraj simulacije "my_verhicle"....23
Slika 4.12: Brojčane vrijednosti simulacije "my_verhicle"....24
Slika 4.13: Izvadak naredbe *xrandr*....25
Slika 4.14: Izvadak naredbe *top*....25
Slika 4.15: Iskorištenost procesora prema dretvama....26
Slika 4.16: Iskorištenost procesora prema dretvama....26
Slika 4.17: Detaljni opis pokrenutih procesa....27
Slika 4.18: Neuspješno pokretanje Steam razvojnog okruženja....28
Slika 4.19: Izvadak naredbe *xrandr* za HTC Vive....28