Simulacija robota koristeći V-REP

Krstičević, Ivan

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

SIMLACIJA ROBOTA KORISTEĆI V-REP

Ivan Krstičević

Split, rujan 2019.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

SIMULACIJA ROBOTA KORISTEĆI V-REP

Ivan Krstičević

SAŽETAK

Opis V-REP-a i njegovih osnovnih funkcionalnosti i izgradnja robota i scene u kojoj robot izbjegava objekte i slijedi liniju.

Ključne riječi: Robotika, simulacija

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 34 stranice, 13 grafičkih prikaza i 6 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Saša Mladenović,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Dr. sc. Saša Mladenović,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dr. sc. Goran Zaharija, *poslijedoktorand Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dr. sc. Divna Krpan, *viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: rujan 2019

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of informatics
Ruđera Boškovića 33, 21000 Split, Croatia

ROBOT SIMULATION USING V-REP

Ivan Krstičević

ABSTRACT

Description of V-REP and its basic functionalities and constructing a robot and a scene in which the robot avoids objects and follows a line.

Key words: Robotics, simulation

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 34 pages, 13 figures and 6 references

Original language: Croatian

Mentor: **Saša Mladenović, Ph.D.** *Associate Professor of Faculty of Science, University of Split*

Reviewers: **Saša Mladenović, Ph.D.** *Associate Professor of Faculty of Science, University of Split*

Goran Zaharija, Ph.D. *Postdoctoral Researcher of Faculty of Science, University of Split*

Divna Krpan, Ph.D. *Senior Lecturer of Faculty of Science, University of Split*

Thesis accepted: September 2019

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom SIMULACIJA ROBOTA KORISTEĆI V-REP izradio samostalno pod voditeljstvom Dr. sc. Saše Mladenovića**,** U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u diplomskom radu na uobičajen, standardan način citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Student

Ivan Krstičević

Sadržaj

Uvod

Simulacija je približno oponašanje operacije ili procesa sustava;[1] čin simuliranja prvo zahtjeva da je model razvijen. Taj model je dobro definiran opis simulirane teme, i predstavlja ključne karakteristike, kao što su ponašanje, funkcije i apstraktna ili fizička svojstva. Model predstavlja sami sustav, dok simulacija predstavlja njegovo djelovanje kroz vrijeme.

Simulacija se koristi u mnogim kontekstima, kao što je simulacija tehnologije za optimizaciju performanse, sigurnost, testiranje, trening, obrazovanje, i video igre. Često se koriste računalni eksperimenti za proučavanje simulacijskih modela. Simulacija se također koristi za znanstveno modeliranje prirodnih sustava ili ljudskih sustava za dobivanje uvida u njihovo djelovanje,[2] kao u ekonomiji. Simulacija može biti korištena da pokaže eventualne stvarne efekte ili alternativne uvjete i sljedove akcije. Simulacija se također koristi kada stvarni sustav ne može biti iskorišten, zato što nije trenutno pristupan, ili zato što je opasno ili neprihvatljivo pristupiti mu, ili zato što je dizajniran ali ne još sagrađen, ili možda jednostavno ne postoji.[3]

Ključni problemi simulacije uključuju pridobivanje pravilnih izvora informacije o relevantnim sekcijama ključnih karakteristika i ponašanja, korištenje pojednostavljujućih aproksimacija i pretpostavki unutar simulacije, i vjerodostojnost ishoda simulacije. Procedure i protokoli za verifikaciju i potvrđivanje se i dalje razvijaju u simulacijskoj tehnologiji i praksi, osobito u polju računalne simulacije.

Simulator robotike se koristi za stvaranje aplikacija za specifičnog (ili ne) robota bez da budu ovisne o 'stvarnom' robotu. U nekim slučajevima, ove aplikacije mogu biti prebačene na stvarnog robota (ili ponovno sagrađene) bez izmjena. Simulacije robotike dopuštaju reproduciranje situacija koje ne mogu biti 'stvorene' u stvarnom svijetu zbog cijene, vremena, ili 'jedinstvenosti' resursa. Simulator također i dozvoljava brzo stvaranje prototipova. Mnogi simulatori robota imaju i motor za fiziku (engl. *Physics engine*) za simuliranje dinamike robota.

1. Alati za simulaciju

Stvaranje potpunog virtualnog modela robota ili sustava simuliranjem komponenti i kontrolnih programa može utjecati na opću učinkovitost projekta. Ovisno o razini detalja i preciznosti simulacijskog okruženja, postoje različita područja koja mogu biti analizirana, od kojih sva utječu na cijenu i životni ciklus razvoja do određene mjere.

Prednosti simulacije

- Smanjuje troškove uključene u proizvodnji robota;
- Dijagnozira izvorni kod koji kontrolira neki osobiti resurs ili skup resursa;
- Simulira različite alternative bez da uključuje fizičke cijene;
- Roboti ili njihove komponente mogu biti testirani prije implementacije;
- Simulacija može biti odrađena u stadijima, što je prednost u kompleksnim projektima;
- Demonstracija sustava da bi se odredilo je li izvediv ili ne;
- Kompatibilnost sa širokim rasponom programskih jezika;
- Kraće vrijeme izvršavanja projekta.

Nedostatci simulacije

- Aplikacija može simulirati samo ono za što je programirana da simulira – ona neće simulirati vanjske ili unutrašnje faktore koje previdimo u fazi razvoja;
- Robot se može naći u puno više različitih scenarija u stvarnom svijetu nego što ih može biti simulirano;

Nove verzije simulacijskih softvera nude sve više svojstava koja čine simuliranje lakšim i vrlo bliskim stvarnom životu. Većina simulacijskih alata su kompatibilni sa programskim jezicima poput *C/C++, Perl, Python, Java, LabVIEW, URBI* ili *MATLAB*, no imaju veoma različite skupove svojstava, ovisno o njihovoj svrsi ili području na koje su usredotočeni.

1.1. Virtual Robotics Toolkit

Virtual Robotics Toolkit od *Cogmation Robotics*-a je simulator za *LEGO Mindstorms* ili *VEX* robote, ovisno o odabranoj verziji. Proizvod je fokusiran na STEM obrazovanje i također je koristan za timove koji se žele pripremiti za robotička natjecanja. Podržava uvođenje 3 modela iz *LEGO Digital Designer*-a ili drugih sličnih alata dok programiranje virtualne inteligentne cigle funkcionira isto kao i u stvarnom životu.

1.2. Visual Components

Visual Components[5] dolazi iz Finske i napredan je dizajnerski i simulacijski program za produkcijske linije. Čitavi procesi proizvodnje mogu biti simulirani i analizirani, uključujući robotičku opremu, tok materijala, djelovanje ljudskih operatera i više. Predvodeći proizvod serije *3DAutomate* čak podržava simulacije čitave tvornice. Druga svojstva uključuju „*off-line*“ programiranje (u kontekstu robotike, „*off-line*“ programiranje je programiranje koda robota bez da se radi na fizičkom robotu), otvorene API-e, i opširne biblioteke sa preko 1800 3D modela industrijskih robota, mašinerije, postrojenja, alate i ostali hardware koji se može naći u tvornici.

1.3. RoboDK

RoboDK je „*off-line*“ alat za programiranje za industrijske robote koji dozvoljava skriptiranje koristeći *Python* ili stvarajući programe vizualno zahvaljujući njegovom integriranom 3D simulacijskom okruženju. Svi programi se automatski pretvaraju u jezike specifične za robote prije nego što se učitaju u fizičkog robota. Biblioteka softvera nudi 3D modele za preko 200 industrijskih robota i alata iz *ABB, KUKA, Yaskawa* za spomenuti samo par njih.

RoboDK nudi brojne razvojne pogodnosti – može generirati obavijesti kad se detektiraju mogući sudari robota, grafički predstavlja robotov radni prostor, i omogućuje korisniku da ima pregled čitave tehnologije i programira u skladu s tim.

1.4. Robot Virtual Worlds

Robot Virtual Worlds je napredni simulacijski softver građen oko moćnog *ROBOTIC IDE*-a. Korisnici mogu programirati virtualne *LEGO Mindstorms NXT, EV3, VEX* ili *TETRIX* robote, ili koristeći *ROBOTC* ili vizualno pomoću *Graphical Natural Language library* ekstenzije, i promatrati njihovo ponašanje u 3D simulacijskom okruženju koje precizno renderira te robote i njihove interakcije. *Robot Virtual Worlds* je primarno dizajniran kao obrazovni alat, međutim dobro e prilagođen za sve razine stručnosti – početnici mogu učiti kako programirati ove robote, učitelji i učenici ga mogu koristiti za domaće uratke ili za laboratorijske vježbe, dok napredni korisnici mogu rafinirati kod ili detektirati greške u svom programiranju. Nekoliko softverskih ekstenzija komplimentiraju skup odlika još i dalje. Na primjer, *Vitual Brick Emulator* nudi korisnicima slično iskustvo kao da programiraju stvarni *LEGO Mindstorms brick* sa *NXT-G* ili *LabVIEW*-om. Postoje i ekstenzije za stvaranje ručno rađenih razina, uvođenje 3D modela ili mjerenje udaljenosti i putanje kutova oko virtualnih okruženja.

Robot Virtual Worlds može se pokrenuti na Windowsu i dostupan je u nekoliko specifičnih izdanja. Besplatna verzija je dostupna za preuzimanje , a licencirana verzija počinje sa 49 dolara. Ima i nekoliko aplikacija dostupnih za iPad u kojima korisnik može programirati *VEX* robote ili igrati igre sa simuliranom ponašanju robota koji ovisi o programiranju korisnika.

1.5. Microsoft Robotics Developer Studio

Microsoft nudi razvijačima robota potpun alat koji može biti korišten da programira i stvara 3D simulacije robota i okruženja. *Microsoft Robotics Developer Studio 4* podržava veće robotske platforme kao što su *LEGO Mindstorms, VEX* ili različiti hardware poput *HiTechnic* senzora i mnoge druge. Softver nudi različite metode i tehnologije za brzo pravljenje prototipova i uključuje veliku količinu funkcionalnih biblioteka.

Nažalost, od 22.9.2014 Microsoft je suspendirao svoj istraživački odjel za robotiku, ostavljajući *Microsoft Robotics Developer Studio 4* kao posljednju izdanu verziju softvera. Naravno, to znači da je podrška prilično ograničena i može se naći samo među online zajednicama.

1.6. LabVIEW

Razvijen od strane *National Instruments*-a, *LabVIEW* je više-platformsko (engl.*cross-platform*) okružje za razvoj i dizajn građen oko istoimenog grafičkog programskog jezika. Prva verzija proizvoda izdana je 1986 i trenutno se ekstenzivno koristi u obrazovanju, inženjerstvu i istraživačkim okruženjima. Ovo je kompleksni ekosustav dobro prilagođen za kontroliranje, simulaciju, automaciju, pridobivanje podataka, analizu, mjerenje i mnoge druge svrhe. Velike biblioteke modela su dostupne za simulaciju i širok izbor hardverskih komponenti i većina standardnih sučelja u današnjoj upotrebi su jako dobro podržana. Sam *LabView* nije program otvorenog koda (engl. *open source*), no postoje bezbrojne druge ekstenzije otvorenog koda za laku integraciju sa ostalim sustavima i softverima.

1.7. V-REP

V-REP (Virtual Robot Experimentation Platform) je 3D simulator kompatibilan s Windows-om, Mac-om i Linux-om i dostupan je ili sa besplatnom edukacijskom licencom ili sa plaćenom licencom za komercijalne svrhe.

Softver dopušta modeliranje čitavog sustava ili samo određenih komponenti poput senzora, mehanizama, opreme robota i tako dalje. Kontrolni program komponente može biti pričvršćen na objekt ili scenu s objektima za modeliranje na način sličan stvarnosti. Platforma može biti korištena za kontroliranje hardverskog dijela, razvijanje algoritama, stvaranje tvornice autonomnih simulacija, ili za edukacijske demonstracije.

V-REP je odabir simulatora za projekt iz nekoliko razloga. Kompatibilan je s mnogim operacijskim sustavima i programskim jezicima, besplatan je, ima izrazito opširnu dokumentaciju unutar svog sučelja do te mjere da online podrška gotovo nije ni potrebna. Ima mogućnost uvođenja 3D objekata, i pri uvođenju ih može preraditi na razne načine, od smanjivanja broja trokuta radi performanse, do automatske podjele objekta na osnovnije komponente. To je osobito važno zbog same srži V-REP-a, a to je podjela na objekte i hijerarhija objekata unutar modela. Svaki smisleni robot je sigurno dovoljno kompleksan da formira roditelj-dijete odnos između sebe i još nekoliko objekata, koji isto tako mogu imati druge objekte pod sobom u hijerarhiji modela. Svaki objekt ima razna obilježja i može imati

svoju skriptu. V-REP može integrirati razne programske jezike, no unaprijed zadan mu je njegov lua, koji ima stotine funkcija od kojih je svaka vrlo detaljno dokumentirana. V-REP je sposoban raditi sa izuzetno kompleksnim scenarijima, kao na primjer igra šaha između dva robota, ili simulacija *'Asti'* robota, koji hoda na dvije noge.

2. Tehnička implementacija

Cilj ovog projekta je napraviti robota koji se ne zabija u prepreke i slijedi liniju. Koristeći V-REP, naglasak neće biti na konkretno sklopovlje (na primjer koja je marka vizualnog senzora), nego na funkcionalnost. V-REP će pretpostaviti da su podatci koje korisnik unese mogući, a ako nisu to je korisnikova krivica. Kada bi V-REP omogućio samo rad s konkretnim sklopovljem, to bi riješilo ovaj problem, ali bi isto tako učinilo simulaciju puno manje fleksibilnom.

2.1. Početna scena

Početna V-REP scena sastoji se od poda, svjetala i kamere. Svi objekti u V-REP-u se mogu izmijeniti, pa tako i unaprijed zadani objekti koji čine scenu. Može se čak izmijeniti i glavna skripta V-REP-a, no to nije preporučeno. V-REP radi u trodimenzionalnom prostoru, dakle ima koordinate za X,Y i Z.

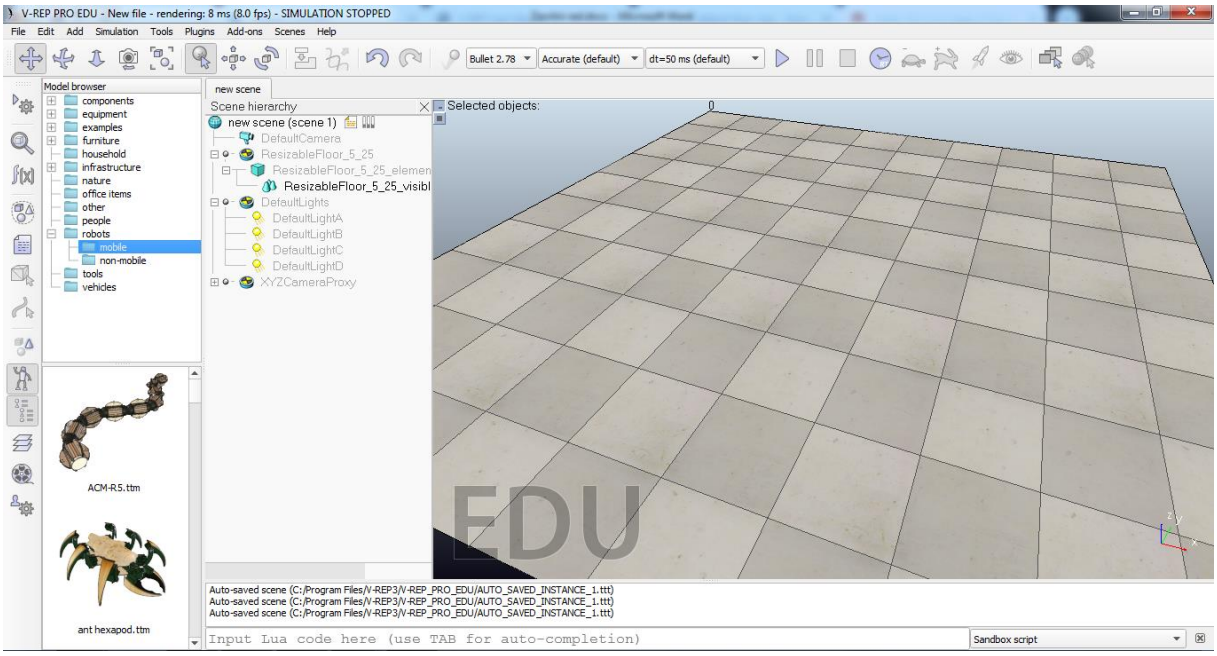
Pod čini 100 kvadrata dimenzija 0,5m x 0,5m. Veličinu poda možemo promijeniti klikom na njega u listi objekata scene, prilikom čega nam iskaču dva klizača, za duljinu i širinu poda. Dimenzije poda mogu samo biti višekratnici broja 5 od 5 do 25, što povećava broj kvadrata poda, dakle dimenzije kvadrata uvijek ostaju iste i mogu biti korišteni kao mjera. Pod predvodi prazan objekt (engl. *dummy*), koji je roditelj kvadratu koji čini pod, a taj kvadrat je roditelj ranije spomenutim manjim kvadratima s fiksiranim dimenzijama. Svaki drugi kvadrat je tamnije nijanse radi preglednosti i to se postiže preko ovog objekta. Iako je pod u svakom slučaju ograničen, objekti ne mogu pasti sa njega ako mu prekorače rub, to jest funkcionira i kao unaprijed zadani zid. Ako objekt stavimo na negativnu Z koordinatu, on će biti ispod poda, ali osim toga nema načina da neki konkretan fizički objekt nestane ispod poda.

Imamo 4 različita svjetla koja se nalaze pod praznim objektom. Svako svjetlo ima različite koordinate u modelu simulacije, i različite kutove pod kojima sja na scenu.

U početnoj sceni imamo i unaprijed zadanu kameru *'Default Camera'*, koja je zadužena za naš pregled simulacije. Ako držimo klik miša i vučemo sa njim po ekranu, mijenjamo X,Y i Z koordinate, a ako držimo srednju tipku miša i vučemo, mijenjamo kutove kamere. Pomicanje kotačića miša služi za uvećavanje pogleda. Osim nje postoji i prazan objekt s 6 djece kamera, koje su zadužene za ostale načine pregleda simulacije.

Prazni objekti sami po sebi nemaju oblik, dimenzije, ni ikakva posebna svojstva, no jako je zgodno koristiti ih za svrstati više objekata s istom svrhom pod jednu cjelinu, a možemo i modificirati prazni objekt da ima nekakvu skriptu u kojoj može doseći sve svoje objekte-djecu.

2.2. Sučelje



Slika 1- Početna scena

Početni meni sastoji se od *File, Edit, Add, Simulation, Tools, Plugins, Add-ons, Scenes* i *Help* kartica.

File kartica (engl. tab) koristimo za upravljanje scenama. Možemo imati više scena otvorenih odjednom i kopirati objekte iz jedne scene u drugu. Preko file kartice otvaramo novu scenu, učitavamo postojeću scenu ili sačuvamo trenutnu scenu.

Add kartica koristimo kad god dodajemo neki osnovni objekt u scenu, uključuju geometrijska tijela, putove, senzore, svijetla i slično.

Simulation kartica služi za podešavanja općih pravila simulacija, kao što su koji motor za fiziku koristimo, usporavanje i ubrzavanje simulacije i slično.

Tools kartica služi za pregled raznih aspekata simulacije, kao što su liste svih objekata na sceni, sve skripte modela, slojevi vidljivosti i slično.

Plugins kartica služi za uvoz objekata (ili čitavih robota) u našu simulaciju.

Add-ons služi za korisnički generirane skripte za proširivanje funkcionalnosti V-REP-a.

Scenes kartica služi za odabir scene među trenutno otvorenim scenama.

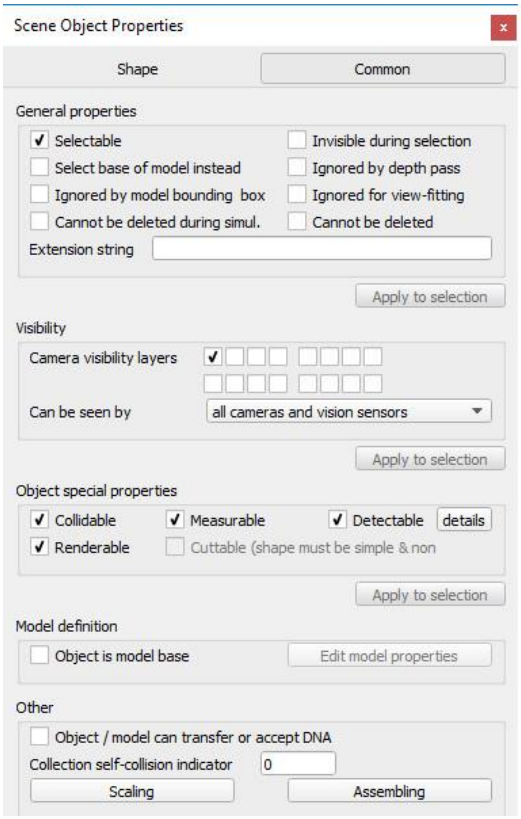
Help kartica nas može povesti na V-REP stranicu za pomoć, gdje možemo naći dokumentaciju svih elemenata i funkcija V-REP-a.

Ispod izbornika (engl. menu) imamo razne gumbove za manipuliranje kamere, objekata i općenito simulacije. Prvih pet gumbova služi za različite načine korištenja kamere. Nakon toga imamo 3 gumba koji služe za manipulaciju objekta, to jest mijenjanje njegove pozicije ili orijentacije. U istom redu su i gumbovi za korak unatrag (engl. undo) i korak unaprijed (engl. redo). Nakon toga imamo gumbove i padajuće izbornike (engl. dropdown) za manipulaciju simulacije. Prvi padajući izbornik bira naš motor za fiziku, drugi bira omjer preciznosti simulacije i brzine izvršavanja simulacije, a treći duljinu vremenskog koraka simulacije. Zadane opcije su *Bullet 2.78* motor za fiziku, precizno izvršavanje simulacije i vremenski korak od 50 milisekundi. Izvršavanje simulacije možemo učiniti manje preciznim ako nam program nije dovoljno brz. Ostali gumbovi u tom redu su za manipulaciju simulacije, to jest započinjanje, pauziranje, zaustavljanje, ubrzavanje i usporavanje simulacije.

Imamo i gumbove sučelja sa lijeve strane. Prvi je za postavke sučelja, drugi je za mijenjanje svojstava odabranog objekta iz scene, treći je za razne izračune kao što su udaljenost između dva objekta, detekcija kolizije i slično, četvrti je lista kolekcija. Kolekcije u V-REP-u su bilo kakva lista objekata. Korisnik ih sam stvara i popunjava. Na primjer možemo izračunati udaljenost između nekog specifičnog objekta i najbližeg objekta iz neke kolekcije. Peti gumb

pokazuje sve skripte u odabranoj sceni. Šesti gumb nam omogućuje izmjenjivanje odabranog oblika, ako je oblik odabran. Ako je put odabran, sedmi gumb nam omogućuje uređivanje puta. Osmi gumb otvara prozor u kojem piše koliko postoji svake vrste objekta i omogućava odabir za sve objekte pojedine vrste. Deveti gumb uključuje preglednik modela. To je malo sučelje s lijeva koje pokazuje sve modele u V-REP-u. Modele možemo dovući na scenu. Modeli su većinom objekti koje ne bismo lako morali napraviti sami, ili bismo morali uvesti da bi ih imali u sceni, tipa stolice, stabla i slično. Neki modeli su već isprogramirani roboti napravljeni po uzoru onih koji postoje u stvarnom svijetu, kao što su *KUKA* roboti i slično. Podijeljeni su po mnogim kategorijama. Deseti gumb je hijerarhija scene. Hijerarhija scene nam prikazuje sve objekte, njihove skripte, djecu i slično. U radu s V-REP-om sigurno se najviše koristi baš hijerarhija scene jer iz nje možemo pristupiti direktno svakom objektu. Jedanaesti gumb je gumb za odabir slojeva. Slojevi određuju koji su objekti vidljivi kameri a koji ne. Po zadanom prvih 8 slojeva je vidljivo a drugih 8 slojeva nije vidljivo. Ako je objektu dodijeljen vidljiv sloj, on je vidljiv kameri, a ako mu nijedan dodijeljeni sloj nije među vidljivima, on nije vidljiv kameri. Predzadnji gumb služi za snimanje videa, a zadnji gumb otvara sučelje sa postavkama korisnika.

2.3. Osnovni objekti



Slika 2 - Osnovni objekti

Postoje svojstva koja su zajednička svim objektima u modelu. Koji god objekt odaberemo, u prvoj kartici imamo specifičnosti za njega, a u drugoj kartici imamo „*Common*“ svojstva[6], to jest njegova svojstva kao općeg objekta.

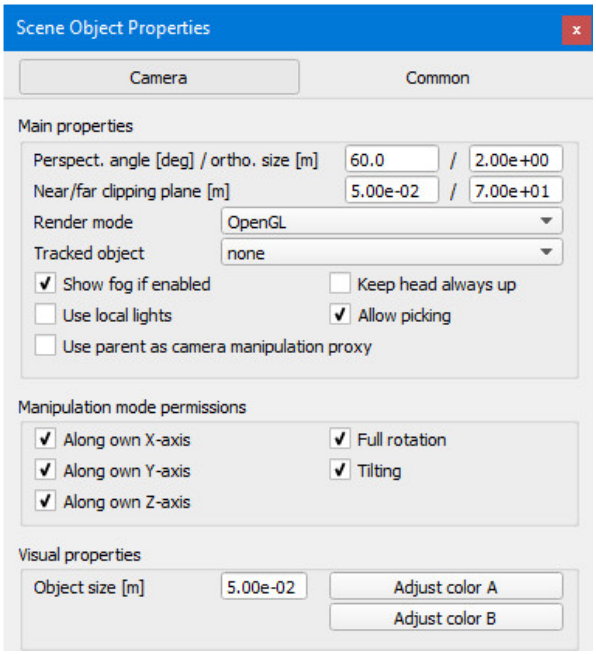
- *Selectable*: označava može li objekt biti odabran u sceni. Objekti uvijek mogu biti odabrani u hijerarhiji scene.
- *Invisible during selection*: kad je odabrano, objekt će biti nevidljiv tijekom odabira, odnosno vidjeti ćemo kroz objekt.
- *Ignored by depth pass*: kad je odabrano, objekt će biti ignoriran tijekom dubinskog renderiranja. To služi za točno pozicioniranje crvene sfere za kretanje kamere.
- *Select base of model instead*: ako je odabrano, onda biranje objekta u sceni će odabrati njegov prvi roditeljski objekt označen kao baza modela umjesto njega. Ovo svojstvo je zgodno za zaštitu modela od neprikladne manipulacije, dopuštajući mu da bude manipuliran kao jedan entitet zajedno sa ostalim objektima.

- *Ignored by model bounding box*: kada je odabrano, i objekt je dio modela, onda kutija koja obavija model pri odabiru neće obuhvaćati taj model. Ovo je korisno za nevidljive objekte od kojih bi kutija za obavijanje modela postala prevelika. Ovo svojstvo nema nikakav funkcionalan učinak.
- *Ignored for view-fitting*: objekti sa ovim artiklom odabranim neće biti uzeti u obzir kada namještamo scenu na pogled dok nije odabran nijedan objekt. Obično će podovi i slično biti tako označeni.
- *Cannot be deleted during simul.*: kada je odabrano, objekt će ignorirati brisanje operacije dok se simulacija izvodi (brisanje je i dalje moguće ako je izvršeno kodom).
- *Cannot be deleted*: kada je odabrano, objekt će ignorirati operaciju brisanja (brisanje je i dalje moguće ako je izvršeno kodom).
- *Extension string*: Tekst koji opisuje dodatna svojstva objekta, uglavnom korišteno za ekstenzije.
- *Camera visibility layers*: svaki objekt u V-REP-u može imati dodijeljen jedan ili nekoliko slojeva vidljivosti. Ako ima barem jedan sloj vidljivosti koji se podudara sa odabranim slojevima u dijalogu odabranih slojeva, onda će objekt biti vidljiv kada je promatran iz kamere. Po zadanom, obliku je pridružen prvi sloj, zglobu (engl. joint) drugi sloj, praznom objektu treći sloj i tako dalje.
- *Can be seen by*: dopušta odabir pojedinačne (ili kolekcije) kamere ili vidnog senzora koji će jedini biti u mogućnosti vidjeti odabrani objekt. Po zadanom je dozvoljeno svim kamerama i senzorima da vide objekt.
- *Collidable*: odabirom omogućujemo ili onemogućujemo sposobnost detekcije kolizije za odabrani objekt koji se može sudariti.
- *Measurable*: odabirom omogućujemo ili onemogućujemo sposobnost računanja minimalne udaljenosti za odabrani mjerljivi objekt.
- *Detectable*: odabirom omogućujemo ili onemogućujemo sposobnost senzora približnosti da detektiraju odabrani objekt koji je moguće detektirati. Klikom na detalje mogu se editirati detektabilni detalji.
- *Renderable*: odabirom omogućujemo ili onemogućujemo sposobnost vidnih senzora da detektiraju odabrani renderabilni objekt.
- *Cuttable*: odabirom omogućujemo ili onemogućujemo sposobnost glodalice da razreže odabrani objekt koji može biti razrezan.

- *Object is model base*: indicira bi li se objekt trebao ponašati kao baza modela. Objekt označen kao baza modela ima posebna svojstva (na primjer, sačuvanje ili kopiranje objekta će isto tako automatski sačuvati i kopirati i njegovu djecu i djecu njegove djece i tako dalje.). Dodatno, kad je takav objekt odabran, obuhvaćajuća kutija za odabir obuhvaća čitav model.
- *Edit model properties*: dopušta otvaranje dijaloga modela.
- *Object / model can transfer or accept DNA*: kada je ovo svojstvo omogućeno za objekt ili model, onda će dijeliti isti identifikator sa svim svojim kopijama. Objekt ili model može prebacivati svoj DNA (kopirati sam sebe) sa svom svojom braćom (objektima / modelima sa istim identifikatorom), preko tipke za transferiranje DNA. Ako imaju 100 istih robota u sceni koje treba modificirati na sličan način, koristeći ovo svojstvo može se modificirati jedan, odabrati, i onda kliknuti gumb za prebacivanje DNA.
- *Collection self-collision indicator*: kada se izvodi izračun kolizije (ili minimalne distance) između dvije identične kolekcije, V-REP će inače provjeriti sve artikle kolekcije protiv drugih artikla u toj kolekciji. U nekim situacijama, poput kinematičkog lanca, nije poželjno provjeravati uzastopne poveznice, pošto bi se mogli konstantno sudarati na sučelju. U tom slučaju, odobravanjem ovog svojstva, dva artikla iste kolekcije neće biti provjereni jedan protiv drugog ako je razlika indikatora točno 1.
- *Scaling*: Objekti ili modeli mogu skalirati na fleksibilan način u V-REP-u. Veličina objekta ili modela, i sva povezana svojstva prikladno skaliraju (na primjer rasponi zglobova, postavke brzine, masa, itd.) tako da skalirani objekt ili model može normalno nastaviti sa djelovanjem (ali na drugačijoj skali)
- *Assembling*: otvara dijalog koji dozvoljava određivanje kako će gumb za sastavljanje/rastavljanje u alatnoj traci podnijeti objekt tijekom sastavljanja

Osim zajedničkih svojstava objekata, postoje neke vrste objekata koje su osnovne za svaki projekt, te se zato isplati upoznati sa njihovim specifičnim svojstvima.

2.4. Kamera

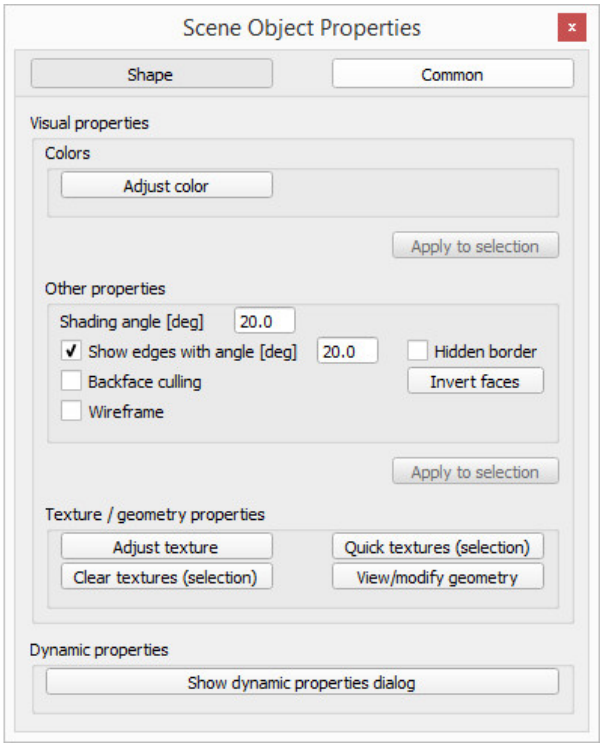


Slika 3 - Specifična svojstva kamera

- *Perspective angle*: kut projekcije perspektive kamere. Učinak ovog parametra je vidljiv samo ako je trenutni pogled u načinu rada za projekciju perspektive.
- *Orthographic size*: veličina ortografske projekcije (ili veličina pogleda) kamere. Učinak ovog parametra je vidljiv samo ako je trenutni pogled u načinu rada za ortografsku projekciju.
- *Near / far clipping plane*: udaljenost od točke ishodišta kamere na kojoj kamera počinje/prestaje vidjeti. Ako bliski / daleki objekti ne bi trebali biti prikazani, ova vrijednost se povećava / smanjuje. Ovi parametri su direktno povezani sa preciznošću renderiranja, pogotovo kada je asocirani pogled u načinu rada za projekciju perspektive. Tada uvijek treba paziti da ne bude prevelik razmak između udaljenosti odsijecanja bliskih i dalekih ravnina, jer inače može doći do z-svađe (kada površine ili pikseli koji su udaljeniji izgledaju kao da se preklapaju sa površinama / pikselima koji su bliže kameri).
- *Render mode*: način renderiranja kamere. Treba uzeti u obzir da POV-Ray renderiranje je puno sporije i renderiranje možda potraje nekoliko sekundi, a možda čak i minuta.

- *Tracked object*: objekt kojeg će kamera pratiti. Praćenje objekta neće utjecati na položaj kamere, nego samo na njenu orijentaciju, koja će automatski biti prilagođena da slijedi praćeni objekt.
- *Show fog if enabled*: ako nije dozvoljeno, kamera neće vidjeti nikakvu maglu ako je magla omogućena.
- *Keep head always up*: ako je omogućeno, kamera će uvijek pokušati držati svoju glavu gore, odnosno držati kamerinu y-os u vertikalnoj ravnini.
- *Use local lights*: ako je omogućeno, onda samo lokalna svijetla koja su roditelj ovoj kameri će biti aktivirana kada se prikazuje sadržaj slike kamere. Svijetla mogu biti učinjena lokalnima u postavkama svjetla.
- *Allow picking*: dopušta omogućavanje / onemogućavanje odabira objekta.
- *Use parent as camera manipulation proxy*: ako je omogućeno, gumbovi alatne trake neće direktno utjecati na poziciju kamere, nego će umjesto toga utjecati na roditeljski objekt kamere. Ova opcija je korisna ako je nekoliko kamera međusobno povezano (na primjer zadana scena u V-REP-u ima 3 kamere ortogonalno poredane jedna prema drugoj. Ako je jedna pomaknuta, druge dvije slijede).
- *Along own X- / Y- / Z-axis*: dopušta određivanje dozvoljenih promjena pokreta kamere.
- *Full rotation*: ako je odabrano, kameru ne ograničavaju rotacije po kutovima.
- *Tilting*: ako je odabrano, onda kamera može biti nagnuta sa odgovarajućim gumbom alatne trake.
- *Object size*: veličina kamere. Ovaj parametar ima samo vizualni učinak i nema funkcionalno značenje.
- *Adjust color A / B*: dopušta podešavanje boje futrole kamere.

2.5. Oblici



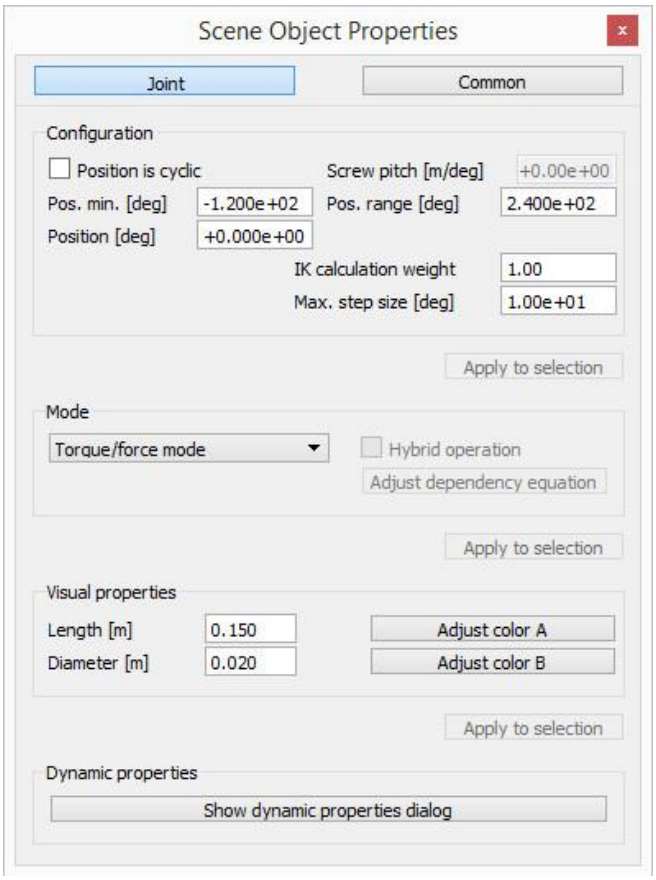
Slika 4 - Specifična svojstva oblika

- *Adjust color*: dopušta izmjenjivanje boje oblika.
- *Shading angle*: kut osjenjivanja je kut iz kojeg su razaznate različite plohe. Ovo samo utječe na vizualni prikaz oblika. Mali kut će učiniti prikaz objekta oštrijim, sa mnogim rubovima, a veliki kut će učiniti izgled oblika glatkim i sa manje rubova.
- *Show edges*: prikazuje rubove crnom bojom. Prikazani rubovi će ovisiti o određenom kutu. Ako je 'Hidden border' odabrano, onda rubovi koji ne dijele više od jednog trokuta će biti skriveni.
- *Backface culling*: svaki trokut koji sastavlja oblik ima unutrašnju i vanjsku plohu. Kada je ova opcija odobrena, onda unutrašnje plohe neće biti prikazane. Ovo je koristan parametar za zatvorene oblike i za prozirne oblike.
- *Invert faces*: Ovo će obrnuti sve trokute. Unutrašnje plohe postaju vanjske i vanjske plohe postaju unutrašnje. Konveksni oblici će postati nekonveksni, osim čistih oblika.
- *Wireframe*: Ako je odabrano, onda će se objekt uvijek prikazivati kao žičani okvir dok je promatran kroz kameru.
- *Adjust texture*: otvara dijalog teksture za odabrani oblik. Kada je oblik asociran sa teksturom, biti će prikazan na teksturiran način.

- *Quick textires (selection)*: primjenjuje kubičnu mapiranu teksturu na sve odabrane oblike. Ovo je osobito korisno za glatke teksture da objekte učini realnijima.
- *Clear textures (selection)*: uklanja teksture sa svim odabranih oblika.
- *View/modify geometry*: otvara dijalog geometrije oblika za odabrani oblik. On dopušta prilagodbu raznih parametara oblika.
- *Show dynamic properties dialog*: otvara dijalog dinamičnih svojstava oblika. Dijalog prilagođava dinamična svojstva oblika.

Neki od ovih parametara samo su dostupni za jednostavne oblike. Kada je složeni oblik odabran, onda se njegovi atributi mogu izmijeniti prelaskom u način rada za izmjenu oblika za složene oblike. Moguće je i rastaviti ga na jednostavne komponente i zasebno izmijeniti svaki od njih.

2.6. Zglob

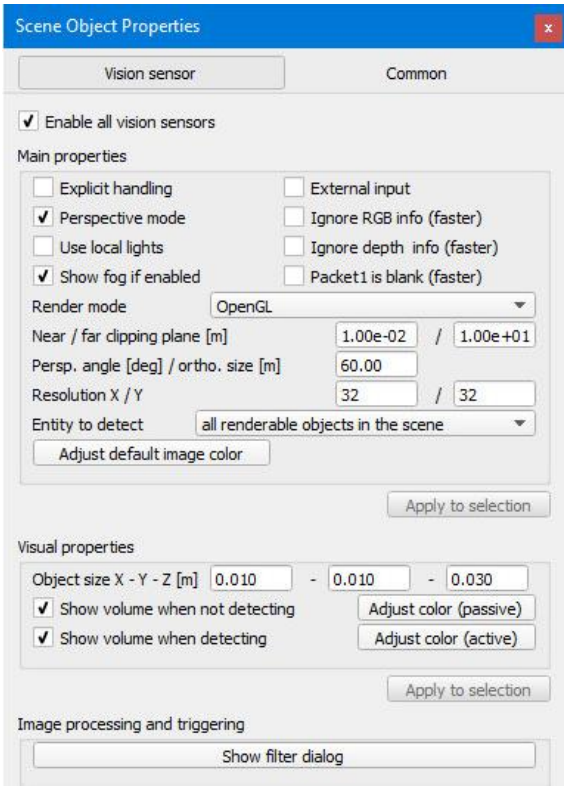


Slika 5 - Specifična svojstva zglobova

- *Position is cyclic*: određuje je li zglob cikličan (varira između -180 i 180 stupnjeva bez ograničenja). Samo revolucionarni zglob može biti cikličan.
- *Screw pitch*: visina navoja zgloba. Ovo svojstvo postane dostupno samo ako je zglob revolucionaran ili vijak, kad je „*Position is cyclic*“ nepotvrđen.
- *Position minimum*: minimalna dozvoljena vrijednost za ne-ciklički revolucionarni zglob, ili vijak, ili prizmatički zglob.
- *Position range*: varijacija raspona ne-cikličkog relovucionarnog zgloba, vijka ili prizmatičnog zgloba. Pozicija takvog zgloba je ograničena između vrijednosti za „*Position minimum*“ i zbroja vrijednosti „*Position minimum*“ i vrijednosti „*Position range*“.
- *Position*: unutarnji položaj revolucionarnog zgloba, prizmatičnog zgloba ili vijka.
- *IK calculation weight*: težina zgloba tijekom inverznih kinematičnih izračuna. Ova opcija omogućuje favoriziranje određenih zglobova tijekom razrješavanja inverzne kinematike. Zglob sa manjom težinom će imati relativno manje varijacije u poziciji u usporedbi s ostalim zglobovima.
- *Maximum step size*: maksimalna varijacija u poziciji dopuštena tijekom jednog kinematičkog izračuna. Manji korak obično rezultira duljim računanjem, ali može biti stabilniji. U dijalogu za inverznu kinematiku, moguće je ovu vrijednost zanemariti odobravanjem svojstva „*Ignore max. step sizes*“.
- *Mode*: način rada zgloba. Zglob može biti u pasivnom načinu rada, inverzno kinematičkom načinu rada, ili u okretajnom / silnom načinu rada.
- *Hybrid operation*: kada je zglob u pasivnom načinu rada, inverzno kinematičkom načinu rada ili zavisnom načinu rada, on može opcionalno operirati još i hibridno: hibridno djelovanje omogućuje zglobu da djeluje na normalan način, ali dodatno, prije dinamičkim kalkulacija, trenutna pozicija zgloba biti će kopirana na traženu poziciju zgloba, i onda, tijekom dinamičkih kalkulacija, zglob će biti rukovan kao motor (ako i samo ako je to dinamički omogućeno).
- *Adjust dependency equation*: ako su zglobovi u zavisnom načinu rada, onda linearna jednadžba može biti određena koja povezuje zglob sa drugim zglobom. Vrijednosti u ovoj sekciji dijaloga su sve određene metrima i radijanima.
- *Length*: duljina zgloba. Nema funkcionalno značenje.
- *Diameter*: dijametar zgloba. Nema funkcionalno značenje.

- *Adjustcolor A / B*: boja A je boja fiksiranog dijela zgloba, boja B je boja pokretnog dijela zgloba.
- *Show dynamic properties dialog*: prikazuje dijalog za dinamična svojstva zgloba. Dijalog omogućava prilagodbu dinamičnih svojstava zgloba.

2.7. Vizualni senzor



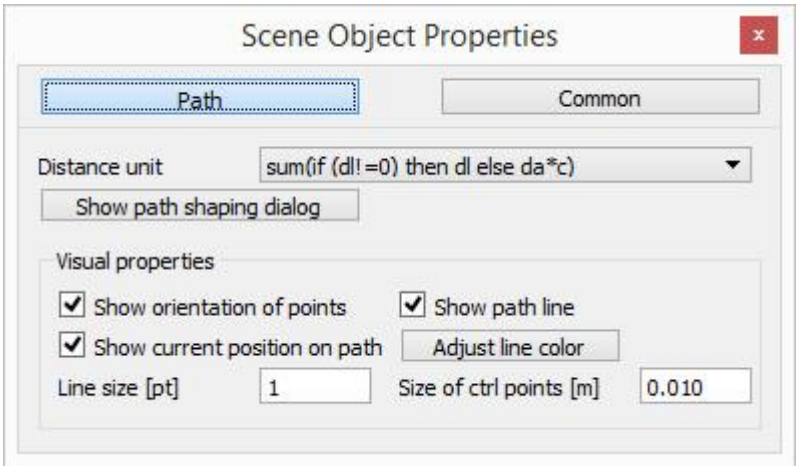
Slika 6Specifična svojstva vizualnih senzora

- *Enable all vision sensors*: upravlja funkcionalnošću svih vizualnih senzora.
- *Explicit handling*: indicira bi li senzor bio izričito obrađen. Ako je omogućeno, senzor neće biti obrađen lua metodom *sim.handleVisionSensor(sim.handle_all_except_explicit)*, nego *sim.handleVisionSensor(sim.handle_all)* ili *sim.handleVisionSensor(visionSensorHandle)*. Ovo je korisno ako korisnik želi rukovati senzorom u dječjoj skripti radije nego u glavnoj skripti. Ako ova opcija nije omogućena, senzor može biti rukovan i u glavnoj i u dječjoj skripti.

- External input: kada je odabrano, senzorove uobičajene operacije biti će izmijenjene tako da može koristiti i filtrirati vanjske slike umjesto onih iz simulacije.
- *Perspective mode*: dopušta odabir između perspektivne projekcije i ortogonalne projekcije senzora.
- *Ignore RGB info (faster):* ako je odabrano, RGB informacije senzora (boja) će biti ignorirane tako da može operirati brže. Opcija je osobito korisna ako je samo bitna informacija dubine senzora.
- *Ignoredepth info (faster):* ako je odabrano, informacija o dubini senzora će biti ignorirana tako da može operirati brže. Ova opcija je osobito korisna ako ionako nije bitna informacija o dubini senzora.
- *Packet1 isblank (faster):* ako je odabrano, V-REP neće automatski izvlačiti određene informacije od dobivenih slika, tako da može operirati brže. Ova opcija je osobito korisna ako nisu važni prvi paket ili sporedne vrijednosti vraćene API funkcijama *sim.readVisionSensor* ili *sim.handleVisionSensor*.
- *Use local lights*: ako je omogućeno, samo lokalna svijetla koja su roditelj ovog vizualnog senzora će biti aktivirana pri prikazu slike ovog senzora. Svijetla se mogu učiniti lokalnima u postavkama svijetla.
- *Show fog if enagled*: ako je onemogućeno, onda ovaj vizualni senzor neće vidjeti maglu ako je magla omogućena.
- *Render mode*: ima 6 različitih načina renderiranja slike. Svaki način rada ima različite interakcije sa kanalima boja ili sjenama slike i slično.
- *Near / far clipping plane*: minimalna / maksimalna udaljenost sa koje senzor može detektirati.
- *Perspective angle*: maksimalni otvarajući kut detekcijskog opsega kada je senzor u perspektivnom načinu rada.
- *Orthographicsize*: maksimalna veličina (uz x ili y) detekcijskog volumena kada senzor nije u perspektivnom načinu rada.
- *Resolution X / Y:* željena x- / y-rezolucija slike obuhvaćene vizualnim senzorom. Rezolucija mora biti pažljivo odabrana s obzirom na aplikaciju. Viša rezolucija će rezultirati sporijim radom. Sa starijim modelima grafičkih kartica, stvarna rezolucija može biti drugačija od onog što je ovdje indicirano (stari modeli grafičkih kartica samo podržavaju rezoluciju 2^n, gdje je n broj iz skupa N0).

- *Entity to detect*: dopušta specificiranje entiteta koji bi trebali biti renderirani. Zadana vrijednost je detektiranje svih objekata scene.
- *Adjust default image color*: dopušta specificiranje boje koja bi trebala biti korištena u područjima gdje ništa nije bilo renderirano. Zadana vrijednost je boja magle okruženja.
- *Object size X / Y / Z*: veličina tijela vizualnog senzora. Ovo svojstvo nema funkcionalni učinak na senzor.
- *Show volume when not detecting*: ako je odabrano, volumen detekcije će biti prikazan kada senzor nije okinut, to jest ne detektira ništa.
- *Show volume when detecting*: ako je odabrano, volumen detekcije je pokazan kada je senzor okinut, to jest nešto detektira.
- *Object colors*: dopušta podešavanje različitih boja vizualnog senzora.
- *Show filter dialog*: prikazuje filter dijalog vizualnog senzora. Dijalog dopušta specificiranje filtera koji se primjenjuju na slike.

2.8. Put



Slika 7 - Specifična svojstva putova

Distance unit: određuje način računanja krivulje između dvije točke na putu.

Show path shaping dialog: prikazuje dijalog za oblikovanje puta. U njemu možemo dodavati nove točke, mijenjati im koordinate i krivulju između točaka i slično.

Show orientation of points: prikazuje x-, y- i z-os svaki Bezierove točke.

Show path line: prikazuje liniju koja povezuje sve Bezierove točke.

Show current position on path: prikazuje crvenu sferu koja ističe trenutnu poziciju na putu.

Adjust line color: Dopušta podešavanje boje puta (tj. Boju linije). Ovo je različito od boje dobivene koristeći dijalog oblikovanja puta.

Line site: širina linije koja povezuje sve Bezierove točke.

Size of ctrl points: veličina kocaka koje predstavljaju kontrolne točke. Kontrolne točke su samo vidljive kad je put odabran ili kada je u načinu rada za izmjenjivanje puta.

2.9. Implementacija

Robot je programibilni stroj koji je sposoban automatski izvoditi seriju kompleksnih naredbi.[4]

Robot može biti uvezen u V-REP, ili može biti jedan od već postojećih modela u V-REP-u. Za učenje V-REP-a najbolje je napraviti vlastitog robota. Idealno bi bilo napraviti jednostavnog robota koji se kreće na kotačima i uz to ima još neke funkcionalnosti.

Za ovaj rad napravljen je robot koji izbjegava sudaranje sa preprekama i slijedi liniju.

Za napraviti robota vlastitog robota potrebno je dodati nekoliko oblika koje se kasnije doda u isti model, što ih čini dijelom jedne cjeline. U model se dodaje primitivna sfera čiji proizvoljnog dijametra. Dodana sfera po zadanom se nalazi u prvom sloju vidljivosti. V-REP ima 16 slojeva vidljivosti, od kojih je prvih 8 omogućeno a drugih 8 onemogućeno. To znači da ako se objekt nalazi u jednom od vidljivih slojeva on će biti vidljiv, a u protivnom neće biti prikazan na kamerama i vizualnim senzorima. Sfera je po zadanom i dinamična i respondibilna, što možemo vidjeti u dijalogu za dinamična svojstva. To znači da će robotovo tijelo padati i moći reagirati (sudariti se) sa ostalim respondibilnim tijelima.

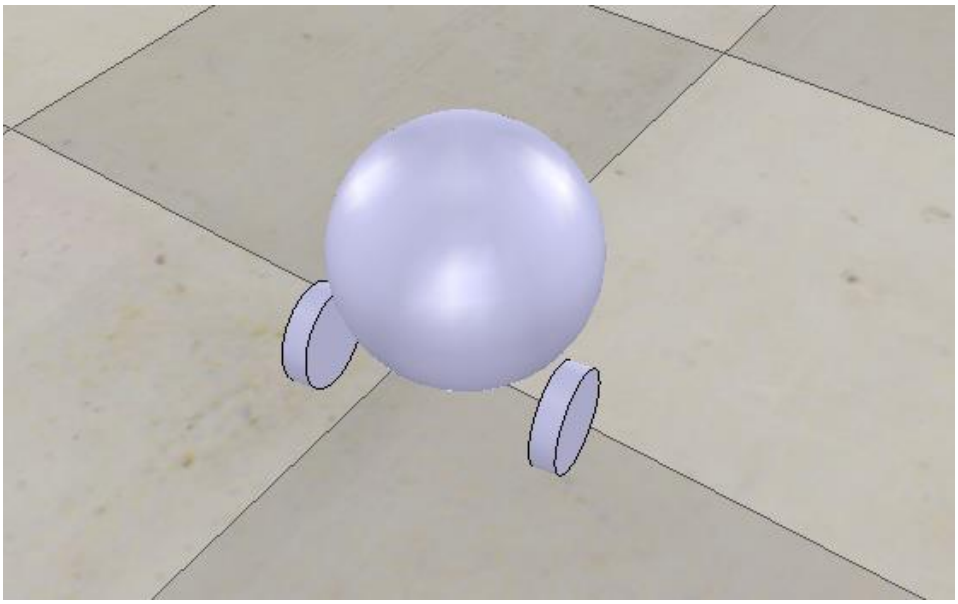
Poželjno je da robotovo tijelo bude respondibilno i na druge kalkulacijske modele. Iz tog razloga, u općim svojstvima objekta treba omogućiti *'Collidable','Measurable','Renderable'* i *'Detectable'*. U svojstvima oblika moguće je promijeniti izgled oblika.

Sada kada postoji oblik, on može biti transferiran i rotiran u prostoru. Pomicanje objekta kroz prostor je obavezno za svaki projekt. Dok je odabrana sfera, to se postiže klikom na gumbove na alatnoj traci *'Object/item shift'* ili *'Object/item rotate'*.

Klikom na *'Object/item shift'* moguće je vući objekt po sceni mišem. Isto tako otvara se dijalog pozicije. Ima 4 kartice. *'Mouse translation'* kartica se odnosi na povlačenje objekta mišem i može biti izmijenjeno da funkcionira drugačije. Sva pomicanja mogu biti relativna na svijet ili roditelja, a neka i relativna uz samog sebe. Promjenom translacijskog koraka mijenja se koliko detaljno miš pomiče objekt; što je korak manji to se manje objekt pomakne kretnjom miša, i to je pomicanje preciznije. Mogu biti odabrane osi po kojima se objekt miče. Zadane osi su x i y. Mogu biti najviše dvije odabrane osi i najmanje jedna. Držeći „*control*“ tipku objekt se okreće po osi ortogonalnom trenutnom izboru (dakle moraju biti dvije odabrane osi), a držeći „*shift*“ tipku smanjuju se koraci čineći translaciju preciznijom. *„Position“* kartica služi za direktan unos koordinata u odnosu na svijet ili roditeljski objekt. Uneseni broj za neku os postaje nova koordinata te osi. „*Translation*“ kartica služi za pomak na osi u odnosu na svijet, roditelja ili sebe. Uneseni broj pomiče koordinatu za taj broj. „*Pos. Scaling*“ kartica množi koordinatu sa unesenim brojem.

Klikom na „*Object/item rotate*“moguće je rotirati objekt. Isto tako otvara se dijalog orijentacije. Ima 3 kartice, *'Mouse rotation', 'Orientation'* i *'Rotation'*. *'Mouse rotation'* kartica se odnosi na povlačenje objekta mišem i može biti izmijenjeno. Orijentiranje mišem može biti relativno na svijet, roditelja ili sebe. Može biti podešen korak rotacije radi preciznijeg rotiranja. Može biti promijenjena os oko koje se objekt rotira. Zadana os oko koje se rotira je Z. Držanjem tipke „*control*“ rotacija se obavlja oko ortogonalne osi, a držanjem tipke „*shift*“ rotacija se obavlja manjim koracima. *'Orientation'* kartica direktno unosi orijentaciju za alfa, beta i gama. Unos je obavljen u stupnjevima. '*Rotation*' kartica okreće objekt za uneseni broj.

Sfera može poslužiti kao trup robota, no trebaju mu kotači. Ne moraju svi objekti biti u istoj sceni pri izgradnji modela; kotači mogu biti napravljeni u zasebnoj sceni tako da ne ometaju rad trenutne scene. Često je praktično raditi sa više različitih scena odjednom radi usredotočivanja na određene elemente. Nova scena može biti stvorena odabirom 'File' kartice iz izbornika. Za kotače su prikladni primitivni cilindri, koji naravno bi trebali biti manji od sfere. Kao i za svaki fizički objekt, i za kotače treba omogućiti svojstva *'Collidable', 'Measurable', 'Renderable'* i *'Detectable'*.



Slika 8 - Osnovno tijelo robota

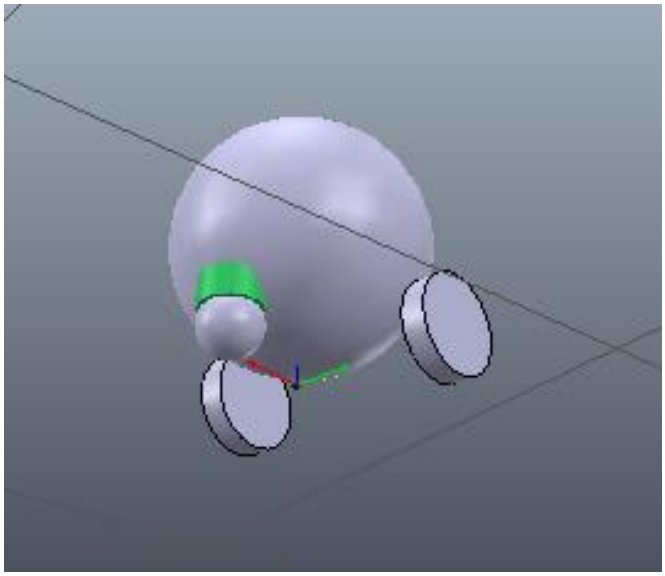
Oba kotača trebaju zglobove koji će ih okretati. V-REP ima 3 vrste zglobova, revolucionarni, prizmatički, i sferični zglob. Pošto je cilj novih zglobova da okreću kotač, revolucionarni zglobovi su najprikladniji. Zglob bi trebao biti usred kotača, tako da je praktično da imaju iste koordinate. Moguće je kopirati koordinate iz svojstava kotača, no V-REP ima praktičniji način. Dovoljno je prvo odabrati kotač, zatim zglob, i otvoriti dijalog pozicije. U *'Position'* kartici klikom na *'Apply to selection'* tipku zglob će poprimiti koordinate kotača. Redoslijed je bitan pri ovome, jer isto tako je moguće kotaču dati koordinate zgloba. Isto može biti učinjeno i sa orijentacijom zgloba u dijalogu orijentacije.

Dvostrukim klikom na ikonu zgloba otvara se dijalog sa njegovim svojstvima. U njegovim specifičnim svojstvima potrebno je pristupiti dinamičnim svojstvima, što se radi klikom na tipku *'Show dynamic properties dialog'*. Tu je nužno promijeniti par svojstava da bi se omogućilo okretanje zgloba,a okretanjem zgloba se postiže i okretanje kotača na vozilu. Potrebno je omogućiti svojstva *'Motor enabled'* i *'Lock motor when target velocity is zero'*. Prvo svojstvo dopušta zglobu da se rotira, a drugo zaustavlja zglob kad mu je željena brzina jednaka nuli. Sa svim ovim izmjenama, kotači mogu biti pričvršćeni na robota.

Ovakav robot ima dva kotača i može se voziti, no nestabilan je i lako se prevrće. Treba još nekakav oslonac, ili još dva stražnja kotača da ga balansiraju ili nekakvu treću kontaktnu točku sa podom koja bi ga držala uspravnim.

U ovom projektu dodana je još jedna manja sfera. Kotači su smješteni ispred polovice tijela robota, a sfera iza. Dodana sfera mora opet omogućiti svoja svojstva *'Collidable'*,

'Measurable', 'Renderable' i 'Detectable'. No, kada bi bio simuliran robot sa sferom kao trećom nogom, on bi je vukao po podu za sobom, i to bi stvaralo trenje, kako u stvarnom modelu tako i u V-REP simulaciji. U ovakvom slučaju nužno je izmijeniti svojstva materijala koja se nalaze u dinamičnim svojstvima oblika na materijal bez trenja. Za povezati novu sferu sa trupom robota u projektu je korišten senzor sile. No na ovakav način i dalje dolazi do problema sa novom sferom. Pošto su obje sfere stvarni fizički objekti, one se odguruju jedna od druge i ne mogu se spojiti ili ignorirati jedna drugu same od sebe. Jedan od načina za riješiti ovakav problem bi bilo informirati V-REP da ova dva objekta, iako oboje imaju omogućenu koliziju, međusobno se ne sudaraju. U dinamičnim svojstvima oblika, obje sfere (tijelo i klizač) imaju svoje lokalne maske. Ako na primjer maska jedne sfere bude 00001111 a druge 11110000, V-REP će smatrati da ta dva objekta međusobno ne reagiraju.



Slika 9 - Nadograđeno tijelo robota

Robot sa ovakvim postavkama je stabilan, no i dalje se miče iako su mu zglobovi kotača zaključani. Ovo je radi različitih masa objekata koje sastavljaju model robota. Korištenje različitih motora za fiziku će isto tako davati različite rezultate, no robot se uvijek i dalje miče, samo na različite načine. Stabilnost dinamičkih simulacija usko je povezana sa masama i inercijama uključenih pokretnih objekata. U ovom slučaju konkretno problem je što sfera koja čini tijelo je daleko teža nego kotači i stražnja sfera. Jedno rješenje ovog problema bi bilo odabrati ta tri objekta, otvoriti dijalog za dinamička svojstva oblika i povećati masu 8 puta. V-REP već ima opciju za udvostručenje mase pa je dovoljno to odabrati tri puta za svaki objekt. Isto vrijedi za inerciju ta tri objekta.

Tijelo robota je baza svih objekata koji čine mode robota. Kada bi u trenutnom okružju se koristilo neku od kalkulacijskih modula V-REP-a, V-REP ne bi prepoznao da je kotač dio robota i slično. Pri mjerenju minimalne udaljenosti, on bi mogao izmjeriti i udaljenost između robota i kotača, što je suvišno. Iz tog razloga i iz sličnih interakcija sa svim ostalim kalkulacijama V-REP-a, korisno je smjestiti robota u kolekciju. Klikom na gumb za kolekcije u alatnoj traci otvara se dijalog kolekcija. Gumbom *'Add new collection'* dodaje se nova kolekcija, no ona je prazna. Klikom na kolekciju i robota u hijerarhiji scene, i zatim klikom na gumb *'Add'* dodaje se robot u kolekciju. Pošto je robot roditelj svih svojih dijelova, to u kolekciju isto tako dodaje i sve njegove dijelove. Na ovaj način puno je lakše koristiti V-REP-ove kalkulacije.

Za korištenje V-REP-ovih kalkulacija klikće se na odgovarajući gumb na alatnoj traci. Kad je otvoren dijalog kalkulacija, za izračunati udaljenost između robota i najbližeg objekta robotu koristi se kartica *'Distance calc.'*. U tom sučelju klikom na gumb *'Add new distance object'* otvara se prozor u kojem se traže dva objekta ili kolekcije između kojih će se mjeriti udaljenost. Za prvu opciju se uzme kolekcija koja sadrži robota, a za drugu opciju se uzme 'sve osim prvog objekta/kolekcije'. Na ovaj način V-REP mjeri udaljenost između robota i bilo kojeg drugog objekta. Sada postoje dva problema u simulaciji: prvi je prikaz te distance. Iako V-REP stvarno računa udaljenost u svakom koraku simulacije, on je mora nekako i prikazati. Drugi je što nema niti jedan drugi objekt na sceni s kojim bi se mjerila udaljenost od robota.

Za prikaz bilo kakve varijable kroz vrijeme korisni su graf objekti. Nakon dodavanja grafa, pričvrsti se za robota (učini se djetetom). Dvostrukim klikom na ikonu grafa otvaraju se svojstva grafa. Kroz svojstva grafa moguće je potpuno promijeniti način na koji graf funkcionira. On po zadanom prikazuje vrijednost varijable kroz vrijeme, no podešavanjem njegovih svojstava možemo ga natjerati da svoj ispis vrši na sceni umjesto na zasebnom prozoru, ili da crta po sceni i slično. Kad se stvori novi graf, obično je poželjno onemogućiti *'Display XYZ-planes'*. Ovo svojstvo renderira graf na sceni. Graf najčešće služi tome da prikazuje podatke u zasebnom prozoru, a kad bi već i bio na sceni trebao bi imati i nekakvu funkcionalnost, kao na primjer da prikazuje crtu od robota do najbližeg objekta i ispisuje udaljenost na njoj. Zadani graf usred scene je samo smetnja. Za dodavanje novih mjerenja grafa, u svojstvima grafa treba kliknuti gumb *'Add new dana stream to record'*. Za mjerenje pozicije robota, odabire se *'Object: absolute x-position'* kao odabrani protok podataka, a kao mjereni objekt se uzima robot. Odobravanjem ovih izbora dodao se novi artikl u listi protoka

podataka. Isto treba napraviti i za robotove apsolutne y i z vrijednosti. Još jedan protok podataka će biti nužan za praćenje najmanje udaljenosti između robota i najbližeg objekta na sceni. Za njega treba odabrati *'Distance: segment length'* kao vrstu protoka podataka i onda udaljenost najbližeg objekta od robota koja je ranije definirana u kalkulacijama kao objekt koji se mjeri. Trenutno postoje 4 varijable za mjerenje u grafu: apsolutne x, y i z pozicije i minimalna udaljenost objekta od robota. To je sve sasvim uredu, no ovo je dobra prilika za pokazati sposobnosti funkcionalnosti grafa u V-REP-u koristeći 3D krivulju, to jest stvarajući krivulju na sceni koja prati poziciju robota kroz vrijeme. Na taj način postići će se učinak kao da robot ostavlja trag iza sebe, što je osobito korisno za funkcionalnost praćenja linije robota. U tu svrhu, sada je nužno za x, y i z apsolutne pozicije grafa onemogućiti svojstvo *'Visible'*. Time se samo minimalna udaljenost od objekata na sceni prikazuje na grafu. Klikom na *'Edit 3D curve'* otvara se dijalog za 3D krivulje. U novoj 3D krivulji, za njenu X vrijednost koristi se apsolutna X vrijednost robota, za Y vrijednost apsolutna Y vrijednost robota i za Z vrijednost apsolutna Z vrijednost robota. Da bi se krivulja ispravno prikazivala u svijetu, nužno je odabrati svojstvo *'Relative to world'*. Svojstvo *'Relative to graph'* bi značilo da krivulja i dalje ovisi o grafu i njegovoj poziciji u sceni, a na ovaj način krivulja prati robota u sceni. Ovim promjenama kretnje robota su ispravno prikazane u sceni.

Robot još uvijek nije kompletan model. U svojstvima robota nužno je odabrati *'Object is model base'* i *'Object/model can transfer or accept DNA'*.

U ovoj točki projekta, robot je spreman za implementaciju izbjegavanja kolizije s objektima. Za to će biti nužna V-REP lua skripta, senzor blizine i naravno, objekti za izbjegavanje. Za to je dovoljno postavljati primitivne cilindre u krug oko robota, jedino što treba biti uzeto u obzir je da su cilindri dovoljno veliki da ih senzor blizine zapravo detektira. Objekti mogu biti i bilo koji drugi oblici. Kako su to fizički objekti, treba im omogućiti svojstva *'Collidable', 'Measurable', 'Renderable'* i *'Detectable'*. Za dva zgloba, senzor približnosti i graf nužno je odobriti svojstva *'Don't show as inside model selection'*. Ovim promjenama klikom na bilo koji dio robota ne odabire se specifična komponenta, nego se uvijek odabere čitav robot. Za dva zgloba i senzor blizine treba promijeniti sloj vidljivosti. Prvih 8 je vidljivo a drugih 8 nije, a njihov sloj je po zadanoj vrijednosti 2. Promjenom sloja vidljivosti zglobova na 10 oni će postati nevidljivi. Za svu djecu robota potrebno je omogućiti svojstvo *'Select base of model instead'*.

Moguće je dodati vizualni senzor kao kameru koju koristi robot. Na taj način biti će jasno vidljivo što se nalazi ispred njega, te je puno lakše podešavati vrijednosti da robot reagira kako je namijenjeno. Vizualni senzor ima svoje specifične vrijednosti koje je nužno podesiti, kao što je koliko daleko vidi i koja je rezolucija kamere. *'Far clipping pane'* je koliko daleko senzor vidi, dovoljno je postaviti na 1 metar. Ako je ovo svojstvo preveliko u odnosu na *'Near clipping plane'*, to jest koliko mu blizu objekt može biti a da ga on vidi, može doći do z-svađe, dakle nije svejedno kolike su vrijednosti ovih svojstava i ne bi trebale biti veće nego što je nužno. Moguće je izmijeniti i rezoluciju. U slučajevima poput ovoga gdje zapravo treba vizualno prikazati sliku, korisno je imati solidnu rezoluciju tipa 256x256 piksela. Ima i drugačijih slučajeva kao što je praćenje linije u kojem slučaju je čak najbolje imati sitnu rezoluciju tipa 1x1, jer tako znamo da robot ili detektira liniju ili nije. Vizualni senzor ima razne filtere koje može primijeniti na sliku koji mogu biti dodani u filter dijalogu klikom na gumb *'Show filter dialog'*. Vizualni senzor ima unaprijed zadana 2 filtera. Redoslijed filtera može davati različite rezultate. Da bi rezultati ovog senzora bili vidljivi, potrebno je dodati plutajući pogled i asocirati senzor sa njim. Dok je odabran vizualni senzor, to se asociranje se postiže desnim klikom na plutajući pogled i odabirom opcije *'Associate view with selected vision senzor'*. Nakon svih ovih izmjena moguće je u plutajućem pogledu vidjeti što se nalazi ispred robota. Robot sad može vidjeti što je ispred njega i osjetiti je li mu išta preblizu. Cilj je da reagira na blizinu i povuče se u stranu ako je nešto ispred njega. Za ovo će biti potrebna lua skripta.

Klikom na robota i dodavanjem skripte iz alatne trake stvara se skripta za robota. Postoje dvije osnovne funkcije koje će svaki robot u V-REP-u koristiti: *sysCall_init()* i *sysCall_actuation()*. Način na koji funkcioniraju može biti modificiran u glavnoj skripti V-REP-a, no to nije uopće preporučeno osim ako za to postoji jako dobar razlog. *sysCall_init()* je inicijalizacija robota i izvršava se kad je skripta započeta. U to funkciji se obično deklariraju sve varijable koje će se koristiti. Za svaki objekt kojim će skripta manipulirati nužno je dodijeliti upravljač (engl. Handler). U kontekstu V-REP-a, upravljač je objekt u lua skripti koji predstavlja objekt iz simulacije. Svi upravljači i zadane brzine kotača će biti u funkciji za inicijaliziranje robota. *sysCall_actuation()* je kod koji manipulira robotom i izvršava se u svakom koraku simulacije. Trenutna skripta samo treba imati reakciju na očitavanja senzora blizine. Funkcija *sim.readProximitySensor()* će pročitati informacije senzora blizine. Ako je išta očitao, robot treba reagirati i kretati se unazad iduće 4 sekunde. Nakon toga samo treba podesiti brzinu kotača. Dok se robot kreće unaprijed, oba kotača se

kreću istom brzinom. Dok se robot kreće unatrag, oba kotača se kreću smanjenom, ali različitom brzinom. Na taj način robot se kreće u stranu dok se povlači, i ubuduće može izbjeći objekt kojeg je očitao.

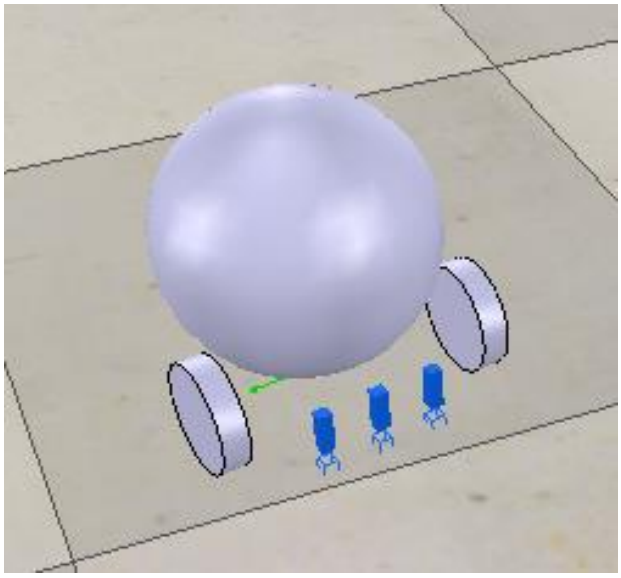
```
function sysCall_actuation()
    result=sim.readProximitySensor(noseSensor) -- Ocitavanje proximity senzora
    -- Ako je nesto detektirano, robot se krece unazad
    if (result>0) then backUntilTime=sim.getSimulationTime()+4 end

    if (backUntilTime<sim.getSimulationTime()) then
        -- Kretanje naprijed
        sim.setJointTargetVelocity(leftMotor,speed)
        sim.setJointTargetVelocity(rightMotor,speed)
    else
        -- Kretanje natrag, smanjena brzina
        sim.setJointTargetVelocity(leftMotor,-speed/2)
        sim.setJointTargetVelocity(rightMotor,-speed/8)
    end
end
```

Slika 10 - lua kod za izbjegavanje objekata na sceni

Robot sada uspješno izbjegava prepreke. Sve što je preostalo je praćenje linije. Treba dodati robotu senzore za liniju, i naravno treba dodati liniju koju će pratiti. Konačna scena će biti takva da će robot biti okrenut 180 stupnjeva i udaljen od linije, i morati će izbjegavati objekte dok se na takav način ne orijentira prema liniji i počne je pratiti.

Prvo je nužno dodati 3 vizualna senzora ortografskog tipa koji će biti upereni prema podu. Njihova rezolucija će biti 1x1 piksel. Na taj način senzor ili očitava liniju ili nije. Isto kao što je senzor blizine tek korišten u kodu, tako će i vizualni senzori biti pozvani u kodu i njihove vrijednosti očitane.



Slika 11 - Robot sa vizualnim senzorima

Nakon vizualnih senzora simulacija treba put za slijediti. Dodavanje puta će staviti dvije točke na scenu, ali to samo po sebi neće biti dovoljno. Osim pomicanja puta u stranu, biti će potrebno koristiti dijalog za oblikovanje puta. V-REP-ov put je ubiti Bezierova krivulja. To znači da je put sačinjen od samo nekoliko točaka, a krivulje između njih se izračunaju koristeći varijable koje mogu biti ručno uređene između svake dvije točke. To znači korisnik može podešavati kolika je krivulja između neke dvije točke, a uvijek mora biti krivulja jer robot neće dobro reagirati na linije pod 90 stupnjeva i slično. Kada je napravljen proizvoljan put, korisno je samo ga malo povisiti po z koordinati. Na taj način put i pod nisu na jednakoj Z koordinati i neće doći do z-svađe. Nakon toga potrebno je prilagoditi kod.

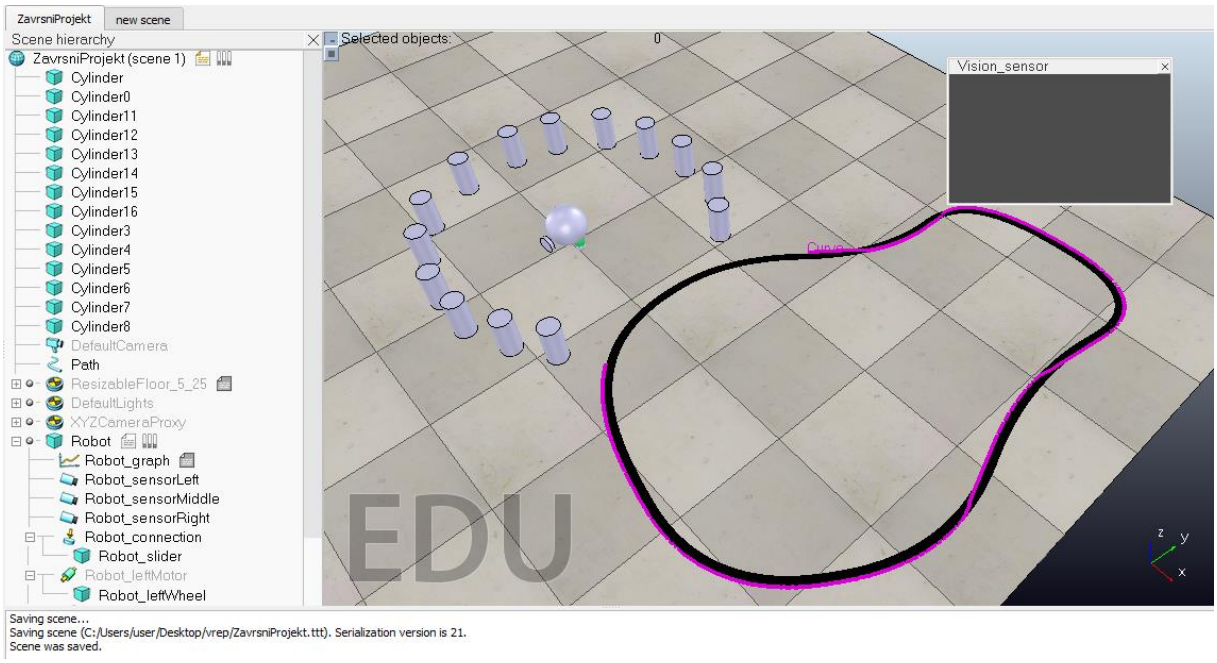
```
-- citanje vizualnih senzora:
sensorReading={false,false,false}
for i=1,3,1 do
    result,data=sim.readVisionSensor(floorSensorHandles[i])
    if (result>=0) then
        sensorReading[i]=(data[11]<0.3) -- data[11] je prosjecna intenzivnost slike
    end
    print(sensorReading[i])
end

-- kod za slijed linije:
rightV=speed
leftV=speed
if sensorReading[1] and not sensorReading[3] then
    leftV=0.03*speed
end
if sensorReading[3] then
    rightV=0.03*speed
end
```

Slika 12 - lua kod za pratnju linije

Za čitanje podataka vizualnih senzora koristi se funkcija *sim.readVisionSensor()*. Ona vraća niz podataka, i sprema ih u dvije varijable, „*result*“ i „*dana*“. „*Result*“ govori je li čitanje uspješno. Ako je čitanje uspješno, provjerava se prosječna intenzivnost očitane slike. Ovaj podatak daje omjer crnog i bijelog tona slike, a put je crn. Ako je prosječna intenzivnost lijevog senzora dovoljno mala, robot se kreće ulijevo. Ako je prosječna intenzivnost desnog senzora dovoljno mala, robot se kreće udesno. Jednom od smjerova treba dati prednost u slučaju da je robot ugazio na liniju i lijevo i desno. Problem može biti riješen i na neki drugi način ali ovo je jako jednostavno i kratko rješenje.

3. Vrednovanje



Slika 13 - Konačna scena

Konačna scena projekta sastoji se od robota okruženog objektima koje mora izbjegavati. Izbjegavanje u ovom scenariju znači da će se vraćati unazad, i pritom kretati u stranu. Na taj način robot se treba dovoljno okrenuti da uspješno iziđe iz kruga objekata i pređe na put na podu. Nadalje samo slijedi put i time je njegov zadatak izvršen. Na vizualnom senzoru se prikazuje što je direktno ispred robota. Robot ispod sebe ostavlja trodimenzionalnu liniju koja označava njegovu putanju. Na ovaj način je lakše vidjeti grešku slijeda puta. Pri pokretanju simulacije prikazuje se i prozor s grafom koji prikazuje udaljenost robota od najbližeg objekta na sceni kroz vrijeme. Prikazuje se i klizač (engl. *slider*) koji može podesiti brzinu robota. Pošto robot uspješno izvršava sve svoje zadatke, implementacija modela je uspješna.

Zaključak

Prije same realizacije robota, koja je prije svega vremenski i financijski zahtjevna, poželjno je provjeriti idejni projekt. Za provjeru idejnog projekta mogu se koristiti klasične metode temeljene na fizici i matematici, ali i metode koje svoj pristup temelje na učenju iz iskustva. Za učenje iz iskustva potrebno je robota koristiti u različitim okruženjima i u različitim situacijama. Kada bi se odlučili za navedeni pristup u realnom svijetu testiranje bi bilo dugo i skupo, te se virtualna okruženja nameću kao logičan odabir.

Ako je donesen zaključak da je simulacija nužna, a pri svakom većem projektu će biti, treba odabrati i koje okruženje će biti korišteno za simulaciju. Mnoga različita okruženja za simulaciju imaju specifične prednosti i mane. V-REP je odabran za ovaj projekt zato što je idealan za opće učenje robotike, za razliku od tipa sučelja koje bi se specijaliziralo za specifične sklopove.

V-REP-ova mana bi bila u tome da nema osobito ugodno korisničko sučelje. Inicijalno je teško započeti rad sa njim zbog jako opširnog broja opcija, no to se ispostavi kao prednost jednom kad je savladano. Pošto V-REP ne radi nužno sa specifičnim stvarnim komponentama, on će pretpostaviti da što korisnik unese je moguće. Ako korisnik odredi da se robot kreće pri ogromnim brzinama, V-REP će to prihvatiti bez obzira na to koliko je to zapravo realno. Time V-REP se čini nešto manje pouzdanim od okruženja za simulaciju koja rade sa specifičnim komponentima sa unaprijed poznatim svojstvima i mogućnostima, no to ga isto tako čini jako fleksibilnim, jer za te aspekte je ponekad čak poželjno da budu zanemareni, kao u industriji ako pretpostavljamo da će problem potencijalno biti riješen naknadno, ili u obrazovanju gdje je svrha stjecanje znanja, a ne neka osobita preciznost. No, ta mana je jednostavno u tome što je korisniku omogućeno da napravi grešku. Ako korisnik dobro definira svoj model, to neće biti problem.

V-REP je odličan alat za simuliranje robotike u obrazovanju zbog stvari na koje se odlučio fokusirati. Neki drugi simulatori mogu samo raditi sa određenim robotima, ili su prilagođeni industriji i previše se fokusiraju na specifičnosti komponenti, ali V-REP je idealan za učenje općih osnovnih koncepata robotike bez da se previše fokusira na specifične koncepte.

Literatura

[1] J. Banks; J. Carson; B. Nelson; D. Nicol (2001). Discrete-Event System Simulation. Prentice Hall. p. 3.

[2] Simulation article in Encyclopediaof Computer Science

[3] Sokolowski, J.A.; Banks, C.M. (2009). Principles of Modeling and Simulation. Hoboken, NJ: Wiley. p. 6.

[4] Definition of 'robot'. Oxford English Dictionary

[5] https://www.virtualroboticstoolkit.com/

[6] V-REP User Manual

Sažetak

Ovaj rad opisuje V-REP okružje za simulaciju rada s robotom, uspoređuje ga s ostalim dostupnim simulatorima i prolazi kroz izgradnju modela robota i dodavanje svih njegovih funkcionalnosti i svojstava. Opisane su i iskorištene sve najosnovnije komponente robota i njihova svojstva. Dobiveni robot je sagrađen od skupa osnovnih komponenti. Ima sposobnost zaobilaženja objekata i sposobnost praćenja linije. Osim toga, bilježi podatak o udaljenosti od najbližeg objekta kroz vrijeme i ostavlja liniju za sobom koja pokazuje njegovu putanju. Prednost V-REP-a je njegov objektno orijentiran pristup koji čini simulaciju jednostavnijom, i njegova općenitost zbog koje nije bitno je li riječ o nekom vrlo određenom sklopovlju ili o nečemu što je korisnik upravo izmislio. Rezultat toga je da V-REP ostavlja mogućnost za stvaranje nerealnih scenarija, no korisnik pažljivim simuliranjem može lako zaobići taj problem. To je istovremeno i prednost jer V-REP time zaobilazi ograničavanje na samo par unaprijed stvorenih komponenti.

Summary

This thesis describes the V-REP robot simulation enviroment, compares it to other available simulators and describes the process of constructing a robot and adding onto it functionalities and properties. All of the most basic components of robots were described and used. The resulting robot is built from a set of basic components. It has the ability to avoid obstacles and to follow a line. Besides that, it records the data about the distance between itself and the nearest object through time. It also leaves a line behind it which follows its path. The advantage of V-REP is its object oriented approach which makes simulation simpler, and its generality because of which it does not matter wheter it is about a highly specific component or about something the user just made up. This results in V-REP having the possibility of creating unrealistic scenarios, however with careful simulation by the user this problem should be easily avoided. At the same time this is an advantage since V-REP overcomes the limitation to just a few default components.