Primjena virtualne stvarnosti u mehatroničkim i robotskim sustavima

Vlahović, Filip

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Filip Vlahović

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Prof. dr. sc. Željko Šitum, dipl. ing. Filip Vlahović

Zagreb, 2020.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Ovaj rad izrađen je na Katedri za strojarsku automatiku, Zavoda za robotiku i automatizaciju proizvodnih sustava pod vodstvom prof. dr. sc. Željka Šituma.

Zahvaljujem mentoru prof. dr.sc. Željku Šitumu na potpori tijekom izrade aplikacije za prikaz robotskog manipulatora u virtualnoj stvarnosti, korisnim savjetima, uloženom vremenu, primjedbama i korekcijama koje su pridonijele kvaliteti ovog rada.

Zahvaljujem se tvrtki INETEC d.o.o. na pozajmljenoj opremi, stručnim savjetima te svoj pomoći bez koje ne bi bilo moguće izraditi ovaj rad. Posebno se zahvaljujem voditeljima odjela za razvoj softvera Davoru Brunoviću i odjela za robotiku Sandri Perici Cvjetko na ustupljenoj pomoći i nabavi svih potrebnih resursa za izradu ovog rada.

Na kraju bih zahvalio svojoj obitelji i svojoj djevojci koji su mi bili podrška tijekom ovog studija i hvala svima koji su mi na bilo kakav način pomogli tijekom studiranja.

Filip Vlahović

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602 - 04 / 20 - 6 / 3	
Ur. broj: 15 - 1703 - 20 -	

DIPLOMSKI ZADATAK

Student: **FILIP VLAHOVIĆ** Mat. br.: 0035197955

Naslov rada na hrvatskom jeziku: **Primjena virtualne stvarnosti u mehatroničkim i robotskim sustavima**

Naslov rada na engleskom jeziku: **Application of virtual reality in mechatronic and robotic systems**

Opis zadatka:

Virtualna stvarnost (eng. Virtual reality, VR) je uporaba računalnog modeliranja i simulacija koja omogućava korisniku da stupi u interakciju s umjetnom 3D okolinom. Razvojem računalne tehnologije i stavljanjem novih zahtjeva za bržu i jeftiniju izradu prototipova, sustavi za virtualnu stvarnost sve više se počinju koristiti u razvoju, konstrukciji, montaži i testiranju raznih visokotehnoloških sustava. Korištenjem ovih rješenja skraćuje se vrijeme razvoja proizvoda, smanjuje se broj grešaka u samoj konstrukciji, a izradom detaljnih simulacija na vrijeme se mogu otkriti i ukloniti neke greške pri radu sustava prije nego su oni uopće pušteni u pogon. Također, ovi sustavi se mogu koristiti za trening i edukaciju zaposlenika kako bi što brže obavili zahtjevne operacije (npr. obuka pilota i astronauta, servisera u nuklearnim elektranama, kirurga u operacijskim salama i dr.), što doprinosi većoj kvaliteti proizvoda uz smanjenje troškova razvoja i izrade proizvoda.

U radu je potrebno:
- dati osnovna obilježja virtualne stvarnosti i navesti područja primjene ove tehnologije,
- stvoriti virtualno okruženje i u njega smjestiti robotski manipulator te napraviti značajke sustava koje omogućuju rukovanje dijelovima manipulatora (pregled, rastavljanje manipulatora, itd.),
- izraditi simulaciju u kojoj je moguće primjenom virtualne stvarnosti 'uroniti' korisnika u računalno generiranu okolinu te pokretati određene stupnjeve slobode gibanja manipulatora,
- dodati mogućnost podešavanja parametara rada manipulatora (npr. s promjenom tlaka napajanja se mijenja brzina gibanja i iznos mase tereta koji se može prenositi).

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan: 24. rujna 2020.

Rok predaje rada: 26. studenog 2020.

Predviđeni datum obrane: 30. studenog do 4. prosinca 2020.

Zadatak zadao:

prof. dr. sc. Željko Šitum

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TEHNIČKE DOKUMENTACIJE

SAŽETAK

U ovom radu opisana su područja virtualne stvarnosti s naglaskom na izradu simulacija pneumatski pogonjenih mehatroničkih sustava te na izradu virtualnog okruženja u kojem je moguće detaljno pregledati sklop manipulatora kao i sve njegove dijelove zasebno. Za prikaz virtualnog okruženja korišten je Steam Valve Indeks VR set. U radu je detaljno opisan postupak postavljanja programskog paketa Unity kako bi se mogao koristiti za izradu VR aplikacija, pretvorba standardnih formata koje možemo dobiti iz programskih paketa koji se koriste za konstruiranje pomoću računala u formate koje podržava Unity, izrada skripti potrebnih za rukovanje 3D modelima kao i za pokretanje osi manipulatora. Analizirani su i rezultati izrađene aplikacije te su navedeni nedostaci i prednosti izrade simulacija mehatroničkih sustava putem virtualne stvarnosti.

Ključne riječi: mehatronika, mehatronički sustavi, virtualna stvarnost, Steam Valve Indeks, Unity, C#, robotski manipulator, pneumatika

SUMMARY

This work describes the areas of virtual reality with an emphasis on the creation of simulations of pneumatically driven mechatronic systems and the creation of a virtual environment in which it is possible to examine the manipulator assembly and all its parts separately. The Steam Valve Index VR set was used to display the virtual environment. The paper describes in detail the process of setting up the Unity software package so that it can be used to create VR applications, converting standard formats that can be obtained from software packages used for computer construction to formats supported by Unity, creating scripts needed to handle 3D models and to move the manipulator axis. The results of the developed application are also analyzed, and the disadvantages and advantages of creating simulations of mechatronic systems via virtual reality are stated.

Key words: mechatronics, mechatronic systems, virtual reality, Steam Valve Index, Unity, C #, robotic manipulator, pneumatics

1. UVOD

Čovjek po svojoj prirodi teži novim saznanjima, otkrivanju, istraživanju te dijeljenju informacija od samog pamtivjeka razvoja ljudskog roda. Od samih početaka glavna potreba je želja za napretkom i prenošenjem informacija koristeći snažnu ljudsku sposobnost mašte i promišljanja ulazeći u najmanje detalje, što na kraju rezultira radoznalošću i željom za potpunim uranjanjem u imaginarni svijet i do stvaranja tehnologije virtualne stvarnosti.

Sama tehnologija se svakodnevno razvija sve brže i brže, a virtualna stvarnost predstavlja jedan aspekt računalne tehnologije koja omogućuje stvaranje simuliranog okruženja koje može simulirati fizičku prisutnost u mjestima u stvarnom svijetu ili u zamišljenim svjetovima.

Umjetna inteligencija, proširena ili virtualna stvarnost, *big data* i Internet stvari tehnologije su koje nam kreiraju svakodnevnicu i asistiraju nam pri nekim, nama beznačajnim stvarima.

Virtualna stvarnost je uporaba računalne tehnologije za stvaranje učinka interaktivnog trodimenzionalnog svijeta u kojem objekti imaju osjećaj prostorne prisutnosti.[1]

Tehnologija virtualne i proširene stvarnosti nije baš najnovija i još uvijek nije postala nešto što je svima poznato i svakodnevno se koristi, kao što su to primjerice mobilni telefoni koji su nam stalno u ruci. Prvi relativno funkcionalni VR sustav izumljen je 1990. godine, ali unatoč tome VR i AR tehnologije još uvijek se najčešće vežu uz igrice i zabavu.[2]

Virtualna stvarnost (eng. *Virtual reality,* VR) je upotreba računalnog modeliranja i simulacija koje omogućavaju korisniku da stupi u interakciju s umjetnom 3D okolinom. Primjena virtualne stvarnosti uranja korisnika u računalno generiranu okolinu koja simulira stvarnost kroz upotrebu interaktivnih uređaja koji šalju i primaju informacije i nose se kao naočale, slušalice, rukavice ili odijela.

U uobičajenom VR formatu, korisnik nosi kacigu sa stereoskopskim zaslonom na kojem vidi animirane slike simulirane okoline. Iluzija „uronjenosti“ dolazi od senzora pokreta koji čitaju korisnikove pokrete i prilagođavaju pogled na zaslon u skladu s tim, obično u stvarnom vremenu. Zato korisnik može proći kroz nekoliko simuliranih soba i iskusiti promjenu

gledišta i perspektiva koje su uvjerljivo povezane s njegovim vlastitim pokretima glave i koracima.

Iako se virtualna stvarnost najčešće veže uz igranje videoigara ili gledanje filmova, ona osim zabave ima i neka vrlo važna područja primjene. Ponajviše su to vježbe za aktivnosti u stvarnom životu. Privlačnost simulacija je u tome što se pomoću njih može pružiti obuka, jednaka ili gotovo jednaka praksama sa stvarnim sustavima, ali o manjem financijskom trošku i s većom razinom sigurnosti. Ovo je posebno slučaj s vojnom obukom, a prva značajna primjena komercijalnih simulatora bile su pilotske vježbe tijekom drugog svjetskog rata.

Virtualna stvarnost postala je i dijelom kirurgije kroz upotrebu robotskih uređaja koje se kontrolira na daljinu putem posredovanih povratnih osjetilnih informacija za obavljanje zadataka.[3]

Virtualna stvarnost svoju primjenu u mehatronici nalazi u izradi simulacija konstruiranih mehatroničkih sustava. Kroz izrađene simulacije mogu se eliminirati neki nedostaci koji bi bili uočeni tek pri izradi prototipa, može se provjeriti funkcionalnost mehatroničkog sustava, a može se čak i optimirati same parametre rada kao što su tlak i protok u pojedinim granama sustava.

Za prikaz primjene virtualne stvarnosti u mehatronici koristit će se automatizirani pneumatski manipulator izrađen u sklopu završnog rada s naslovom „Projektiranje, izrada i upravljanje pneumatskog manipulatora“.

U okviru gore navedenog završnog rada izrađen je automatizirani pneumatski manipulator pogonjen stlačenim zrakom i izrađen je u svrhu edukacije studenata. Manipulator ima dva stupnja slobode gibanja i služi za prepoznavanje, prihvaćanje i premještanje obradaka u proizvodnji. Ostvarivanje stupnjeva slobode gibanja, prihvaćanje obradaka i konstantan dotok materijala osigurani su pneumatskim aktuatorima. Dovod zraka u pneumatske aktuatore regulira ventilski otok koji je preko relejnog modula upravljan upravljačkim uređajem Arduino Mega2560. Prepoznavanje prisutnosti obradaka izvodi se induktivnim senzorom koji preko tiskane pločice šalje signale Arduinu, temeljem kojih je ostvareno daljnje upravljanje cijelog sustava.

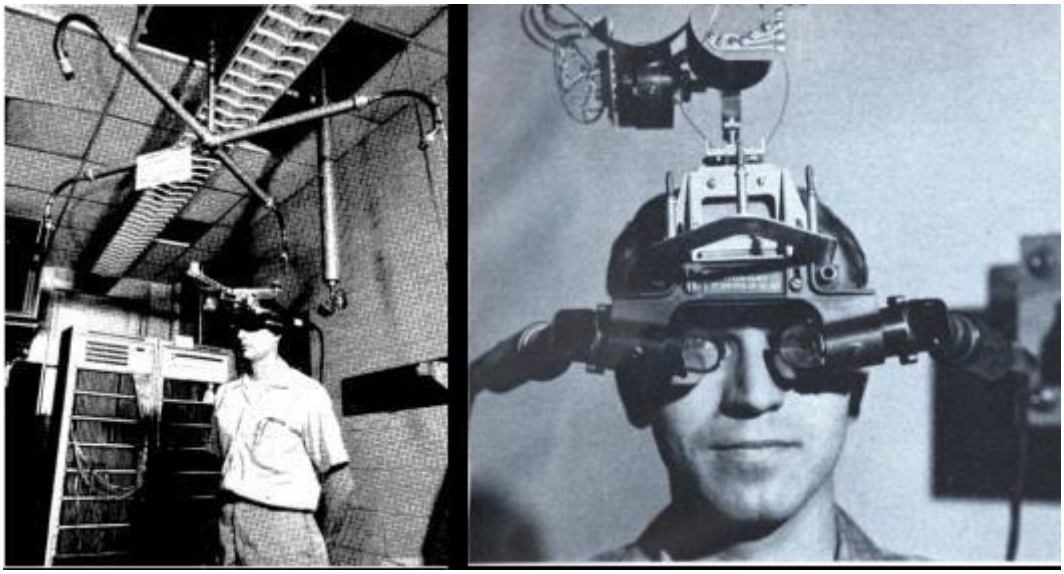
2. POVIJEST VIRTUALNE STVARNOSTI

Točno podrijetlo virtualne stvarnosti teško je precizno odrediti, dijelom zbog toga koliko je bilo teško formulirati definiciju koncepta alternativnog postojanja. Razvoj perspektive u renesansnoj Europi stvorio je uvjerljive prikaze prostora koji nisu postojali u onome što se nazivalo umnožavanje umjetnih svjetova. Ostali elementi virtualne stvarnosti pojavili su se već 1860-ih.

Antono Artaud zauzeo je stav da se iluzija ne razlikuje od stvarnosti, zagovarajući da gledatelji u predstavi trebaju zanemariti nevjericu i dramu koja se odvija na sceni smatrati stvarnom. Prve naznake modernijeg koncepta virtualne stvarnosti došle su iz znanstvene fantastike.[4]

Morton Heilig pedesetih je godina 20. stoljeća pisao o "Kazalištu iskustava" koje je moglo učinkovito obuhvatiti sva osjetila, privlačeći tako gledatelja na aktivnost na ekranu. Izradio je prototip svoje vizije nazvane Sensorama 1962. godine, zajedno s pet kratkometražnih filmova koji će se prikazati u njoj dok je sudjelovao u više osjetila (vid, zvuk, miris i dodir). Sensorama je bio mehanički uređaj. Heilig je također razvio ono što je nazivao "Telesferska maska" koju je patentirao 1960. godine. U prijavi patenta uređaj je opisan kao teleskopski televizijski uređaj za individualnu upotrebu. Gledatelju se pruža potpuni osjećaj stvarnosti, tj. pomicanje trodimenzionalnih slika koje mogu biti u boji, sa 100% perifernim vidom, stereo zvukom, mirisima i vjetrom.

Ivan Sutherland je 1968. godine uz pomoć svojih učenika, uključujući Boba Sproulla, stvorio je ono što se smatralo prvim sustavom prikaza na glavi koji se koristi u imerzivnim simulacijskim aplikacijama. Bio je primitivan i u pogledu korisničkog sučelja i vizualnog realizma, a zaslon koji je korisnik trebao nositi bio je toliko težak da ga je trebalo objesiti sa stropa. Grafika koja je sadržavala virtualno okruženje bila je jednostavna soba s modelima žičanog okvira. Zastrašujući izgled uređaja nadahnuo je njegovo ime te je nazvan Damoklov mač.[4]



Slika 1. Damoklov mač

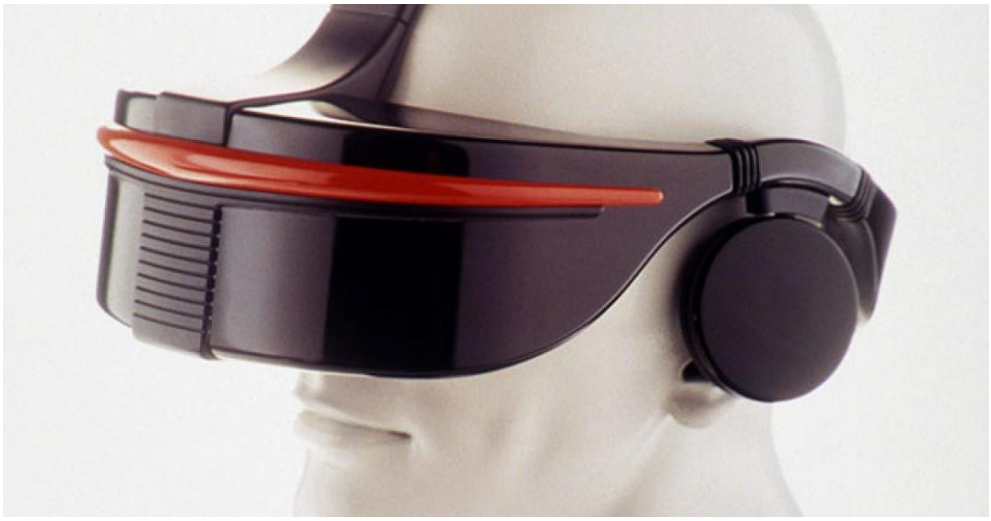
Industrija virtualne stvarnosti od 1970. do 1990. uglavnom je izrađivala VR uređaje za medicinu, simulaciju leta, dizajn automobilske industrije i u vojne svrhe.

David Em postao je prvi umjetnik koji je izradio virtualni svijet kojim se može navigirati. To je razvio u NASA-inom Laboratoriju za mlazni pogon.

Eric Howlett razvio je 1979. godine optički sustav nazvan LEEP (*Large Expanse Extra Perspective*). Ovaj sustav stvorio je stereoskopsku sliku s vidnim poljem dovoljno širokim da stvori uvjerljiv osjećaj prostora. Korisnici sustava bili su impresionirani osjećajem dubine vidnog polja u sceni i odgovarajućim realizmom. Izvorni LEEP rekonstruiran je za NASA-in Ames Research Center 1985. godine kako bi im poslužio za njihovu prvu instalaciju virtualne stvarnosti, VIEW (*Virtual Interactive Enviroment Workstation*), a rekonstrukciju je izveo Scott Fisher. LEEP sustav je postavio temelje za većinu modernih sustava za virtualnu stvarnost.[4]

Slika 2. NASA VIEW VR sustav

Devedesetih godina prošlog stoljeća dolazi do komercijalizacije setova za virtualnu stvarnost te su tada počela predviđanja da će do 1994. godine postojati VR sustav prihvatljive cijene. Sega je 1991. godine napravila Sega VR sustav za igranje arkadnih igara i Mega Drive konzolu. Sega je tu upotrijebila LCD zaslone, stereo slušalice i inercijske senzore koji su omogućili praćenje i reagiranje na pokrete glave korisnika.[4]



Slika 3. Sega VR

Iste je godine Virtuality pokrenuo prvi umreženi višenamjenski sustav za više igrača koji je najavljen i pušten u prodaju u mnogim zemljama, a uključivao je posebnu VR arkadnu igru Embarcadero Center. Cijeli sustav zajedno sa igrom koštao je i do 73 000 američkih dolara.

Palmer Luckey je 2010. godine konstruirao prvi prototip Oculus Rifta. Prototip je izrađen na ljusci drugog seta za virtualnu stvarnost i mogao je pratiti samo rotacijske pokrete, no mogao se pohvaliti vidnim poljem od 90° koje u to doba nije bilo viđeno na potrošačkom tržištu. Problemi s izobličenjem prikaza koji proizlaze iz leće ispravljeni su softverom koji je napisao John Camrack za verziju Doom 3. Facebook je 2014. godine kupio Oculus VR.

Valve je 2013. godine otkrio i podijelio s ostatkom svijeta otkriće zaslona koji omogućuje prikaz VR sadržaja bez zastajkivanja i razmazivanja. Ovaj tip zaslona je usvojio i Oculus te ga od tada koristi u svojim VR sustavima.[4]

Slika 4. Oculus Rift

HTC i Valve su 2015. godine izbacili HTC Vive VR sustav koji je sadržavao tehnologiju praćenja nazvanu Lighthouse, koja je koristila zidne bazne stanice za praćenje pozicije pomoću infracrvene svjetlosti. VR sustav korišten za izradu ovog projekta nasljednik je HTC Vive sustava.[4]

Slika 5. HTC Vive

3. VRSTE VIRTUALNOG PRIKAZA

3.1. Proširena stvarnost AR

AR tehnologija omogućuje prikaz elemenata koji ne postoje u stvarnom svijetu putem aplikacija kroz zaslon nekog uređaja, najčešće mobilnog telefona. Ti elementi proširuju stvarnost oko nas, ali samo ako je gledamo kroz zaslon. Aplikacija omogućuje gledanje bez mogućnosti mijenjanja elemenata koje vidimo, tj. bez mogućnosti ikakve interakcije s njima.

Osim mobilnog uređaja postoje i AR naočale kroz čije leće vidimo stvarnost oko sebe, ali i dodatne elemente kojima se dopunjuje slika. Google i društvena mreža Snapchat razvili su jedne od prvih takvih naočala, no ta digitalna inovacija nije doživjela široku primjenu. Trenutno na tržištu, tehnološki i kvalitetom, prednjači Microsoft sa svojim Hololens AR setom.

AR tehnologiju uz korištenje naočala ili čak leća koje se stavljaju u oči često viđamo u akcijskim filmovima, a jedna od najpoznatijih igrica vođena tom tehnologijom bila je Pokemon GO.[5]

Slika 6. Microsoft Hololens 2

3.2. Virtualna stvarnost VR

Za VR tehnologiju koriste se neprozirne naočale kroz koje se ne vidi ništa. Jedino što se vidi je virtualno stvoreni svijet. U tom trenutku taj svijet postaje virtualna stvarnost u kojoj fizička interakcija s vidljivim elementima nije moguća.

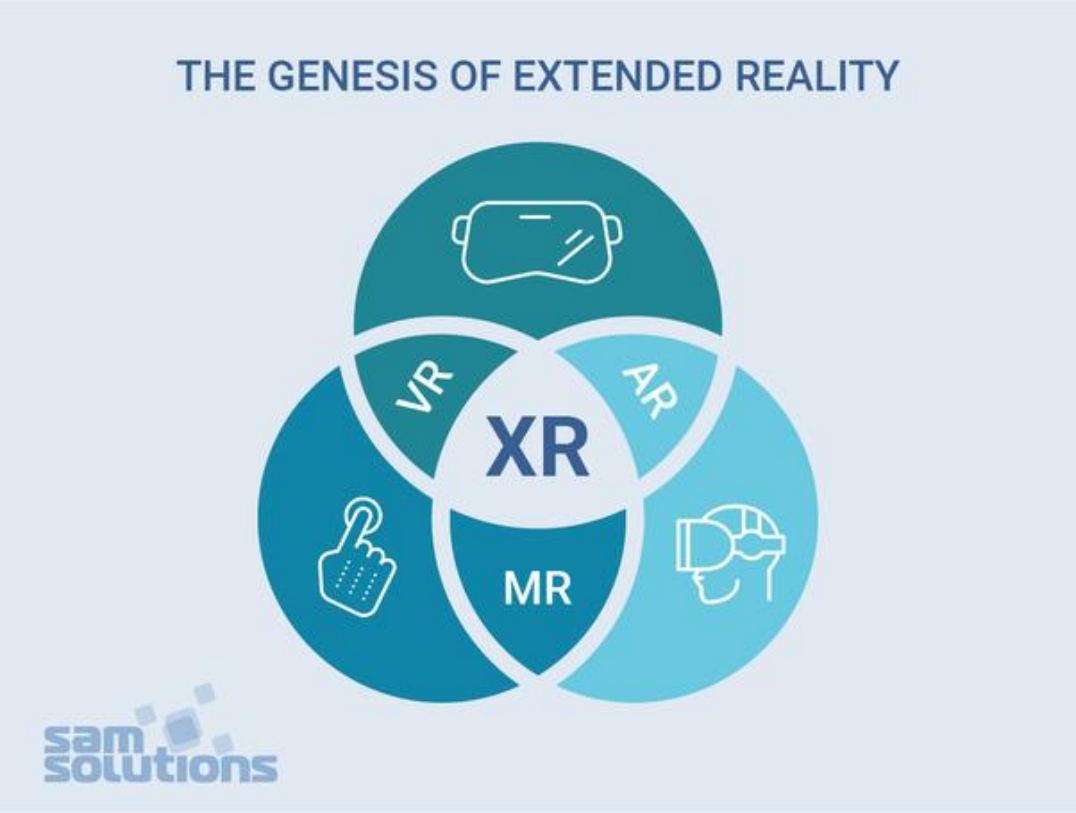
Postoje veoma jednostavni VR sustavi za koje je potreban samo kartonski okvir sa dvije leće u koji se umetne mobilni telefon koji zatim služi kao mozak i mišići tog VR sustava. Drugi sustavi imaju sofisticirane dodatke koji omogućuju dublje uranjanje u virtualni svijet. Ti sustavi koriste rukavice, slušalice, bežične kontrolere i senzore kako bi omogućili što realnije virtualno iskustvo. Neki virtualni sustavi mogu uključivati i pokretnu traku za kretanje koja simulira hodanje ili trčanje. Kada korisnik trči brže u stvarnom svijetu, njihov avatar može se kretati istom brzinom u virtualnom svijetu. Kada se korisnik prestane kretati zaustavit će se i lik u igri.[5]

Slika 7. Sustav za virtualnu stvarnost

3.3. Miješana MR i produžena XR stvarnost

Miješana stvarnost spaja virtualnu i proširenu stvarnost jer omogućuje i interakciju s elementima koji se pojavljuju. Pomoću te tehnologije može se mijenjati elemente koji se nalaze u prikazima, graditi ih, stvarati nove, a sve to u stvarnom vremenu.

Produžena stvarnost je relativno nov pojam u tehnološkom rječniku pa još ne postoji ustaljeni hrvatski prijevod, osobito zbog preklapanja prijevoda za pojmove *augmented* i *extended*. Međutim XR tehnologija jednostavno znači spoj stvarne i virtualne tehnologije te svih dodatnih elemenata koji se koriste uz njih. Drugim riječina XR tehnologija objedinjuje sva tri pojma (AR, VR i MR) – baš kao što je pojam „računalna tehnologija“ krovni termin za sve ostale informacijske tehnologije i dodatke koje koristimo ili trebamo kada koristimo računalo.[5]



Slika 8. Produžena stvarnost

4. INSTALACIJA I POSTAVLJANJE HARDVERA I SOFTVERA ZA IZRADU VR APLIKACIJE

4.1. Steam Valve Index

Steam Valve Indeks je set za virtualnu stvarnost koji se sastoji od naočala, dva kontrolera i dvije bazne stanice. Konstruiran je i proizveden od strane tvrtke Valve. Zaslon koji ovaj set koristi ima razlučivost 1440x1600 piksela po svakom oku što znači da je ukupna rezolucija 2880x1600 piksela. Brzina osvježavanja slike zaslona može biti i do 144 Hz, a vidno polje mu se prostire na 130° što su trenutno najbolje specifikacije na tržištu.

Na naočale su ugrađene i slušalice punog opsega zvuka koje koriste BMR pogonske jedinice za stvaranje točnih i impresivnih zvukova. Osim slušalica naočale imaju i dvije ugrađene kamere koje razvojni inženjeri mogu koristiti kako god žele. Jedna od ugrađenih primjena je prikaz okruženja u kojem se korisnik nalazi u trenutku kada se on približi rubu virtualnog prostora mapiranog za kretanje.

Steam Valve indeks koristi jednu žičanu vezu koja se dijeli na video i audio preko *Display porta,* na USB i na napajanje.

Slika 9. Steam Valve Index naočale

VR set je namjenjen za korištenje s Valve kontrolerima poznatim i kao *Knuckles Controllers*, ali je također kompatibilan s kontrolerima za HTC Vive i HTC Vive Pro. Kontroleri imaju kontrolnu palicu, *trackpad*, dva gumba, gumb za izbornik i 87 senzora koji omogućuju kontrolerima da prate položaj ruke, položaj prsta, pokrete i pritiske tipke kako bi stvorili što vjerniji prikaz korisnikovih ruku u virtualnoj stvarnosti.[6]

Slika 10. Steam Valve Index kontroleri

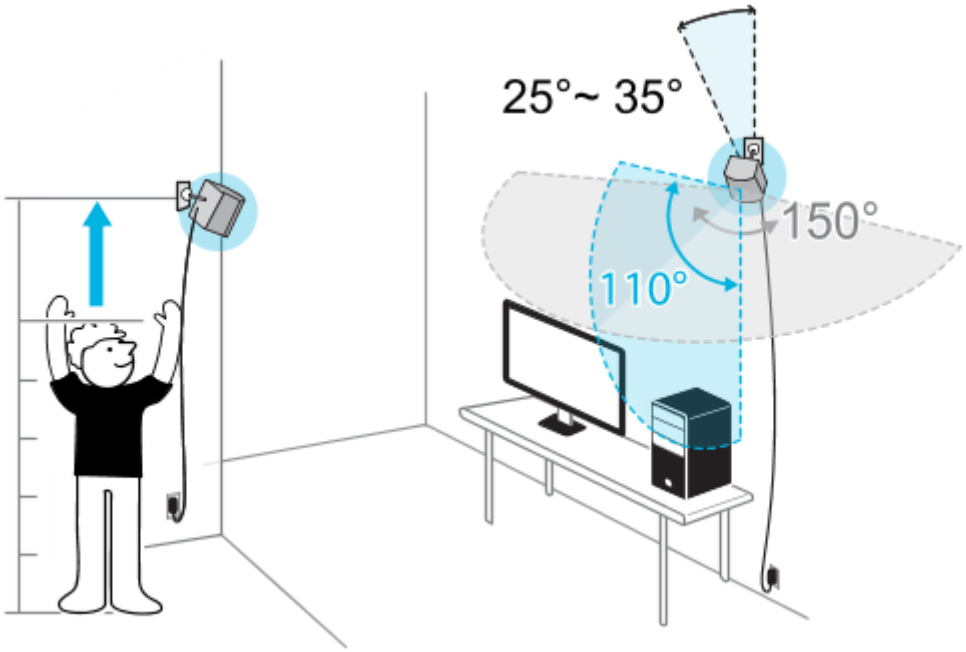
Za praćenje položaja naočala i kontrolera Indeks koristi tehnologiju *Lighthouse tracking 2.0* koja u odnosu na svog prethodnika *Lighthouse tracking 1.0* ima 20% veće vidno polje, koristi samo jedan motor umjesto 2 i ima veću brzinu praćenja za 30%. Dvije bazne stanice pokrivaju područje veličine i do 10x10 metara te se ne trebaju međusobno „vidjeti“ u prostoru. Fiksni laseri frekvencijom 100 Hz osvjetljavaju prostoriju kako bi pratili svjetlosne senzore na naočalama i kontrolerima što osigurava razlučivost pokreta manju od milimetra u svim položajima i radnjama. Komunikacija između baznih stanica i kontrolera te naočala je bežična što uvelike povećava funkcionalnost cijelog sustava.[7]

Slika 11. Steam Valve Index bazna stanica

4.1.1. Postavljanje Steam Valve Indexa

Ovaj VR set je iznimno zahtjevan što se tiče performansi računala za njegovo pokretanje pa je tako bilo potrebno nabaviti dovoljno dobru grafičku karticu, što je u ovom slučaju Nvidia GeForce GTX 1060 sa 6Gb RAM-a.

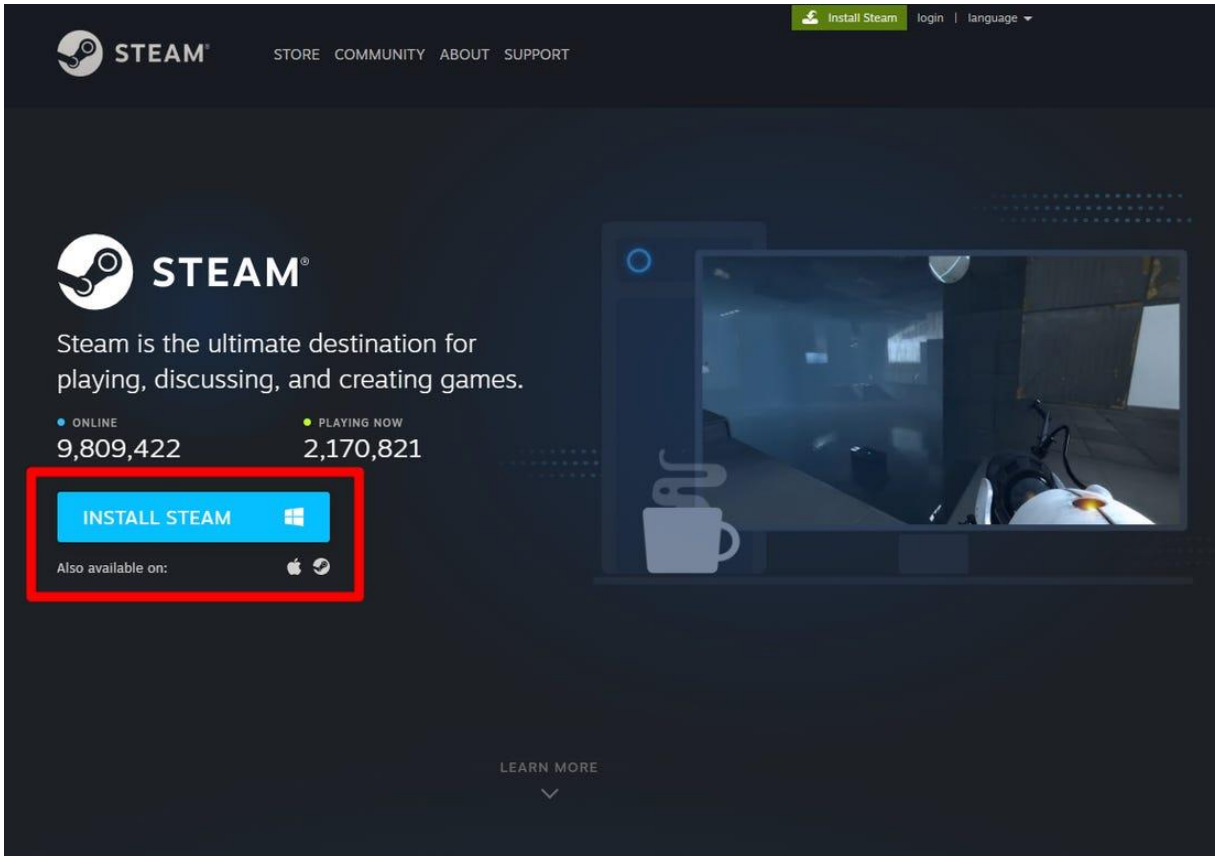
Steam Valve Index se u prostoriju u kojoj će se koristiti postavlja na način da se najprije u računalo ukopčaju *DisplayPort* (DP) i USB 3.0. ulazi za VR naočale. Nakon toga je potrebno po sobi rasporediti dvije bazne stanice koje služe za detekciju položaja naočala i kontrolera. Bazne stanice se postavljaju na način da se nalaze na povišenim pozicijama u odnosu na korisnika i gledaju prema njemu pod kutem od 25° do 35° i da korisnika gledaju na način da u svakom trenutku barem jedna bazna stanica može detektirati kontrolere i naočale.



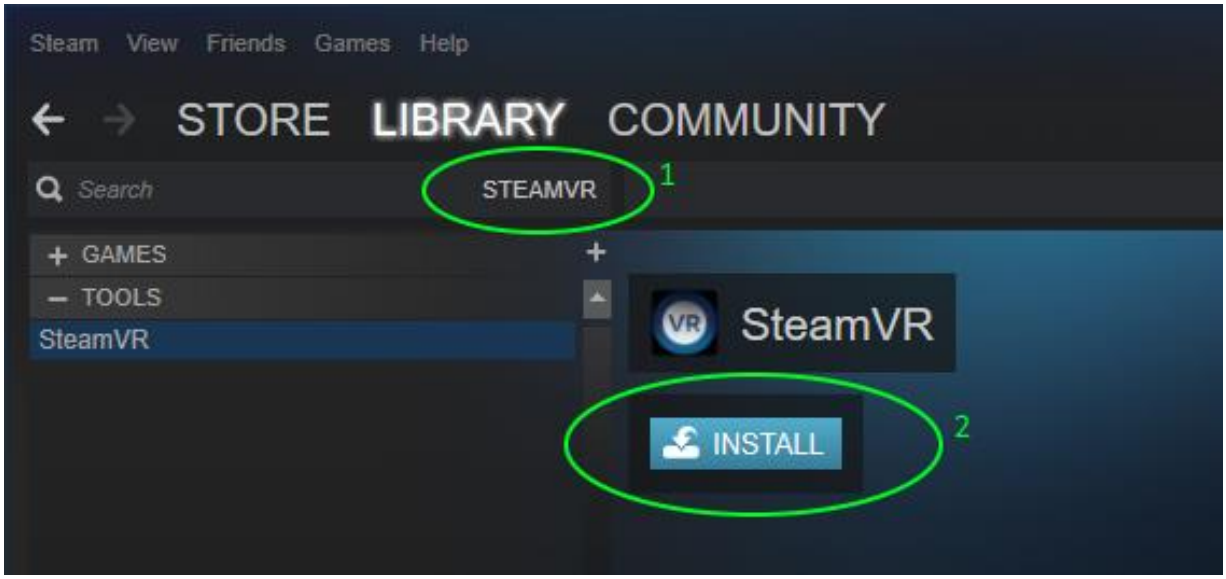
Slika 12. Postavljanje baznih stanica

4.1.2. Instalacija potrebnog softvera i inicijalna kalibracija Steam Valve Indexa

Kako bi VR set mogao ispravno funkcionirati na računalo je potrebno instalirati aplikacije Steam i SteamVR nakon čega je Steam Valve Indeks spreman za inicijalnu kalibraciju.

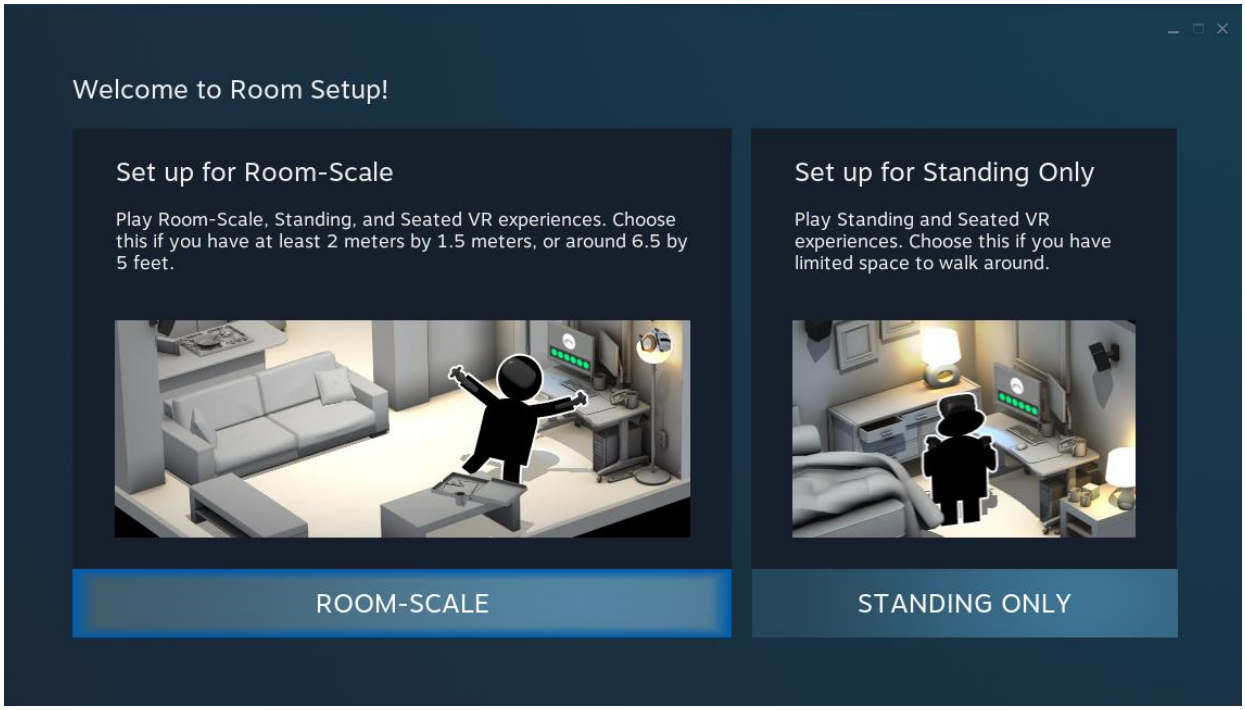


Slika 13. Instalacija Steam aplikacije



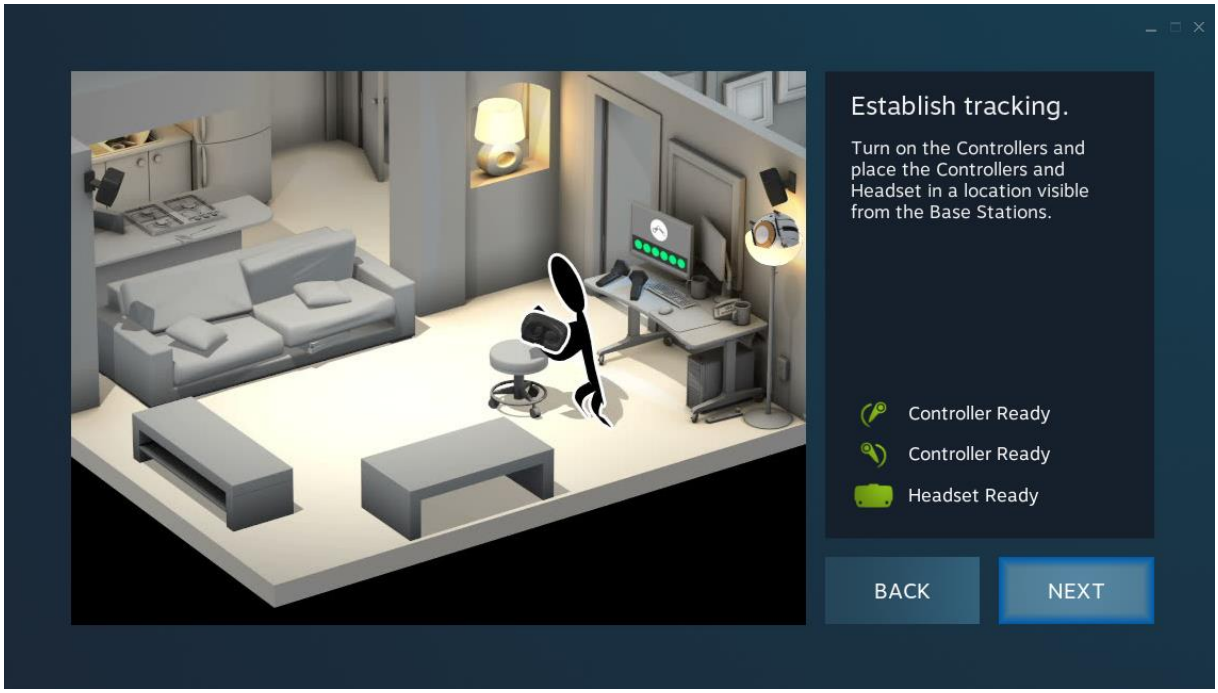
Slika 14. Instalacija SteamVR aplikacije

Kalibracija počinje tako da se pokrenu Steam, a zatim i SteamVR aplikacije i iz padajućeg izbornika SteamVR aplikacije se odabere opcija *Room Setup*. Na prvom koraku kalibracije moguće je odabrati želimo li VR set koristiti na način da stojimo na mjestu ili da omogućimo korisniku da sam neometano šeće po virtualnom prostoru što dodatno povećava sam dojam uronjenosti u virtualni svijet. U ovom radu odabran je način rada u kojem se korisnik proizvoljno može gibati kroz prostor što znači da treba kliknuti opciju *Room Scale*.



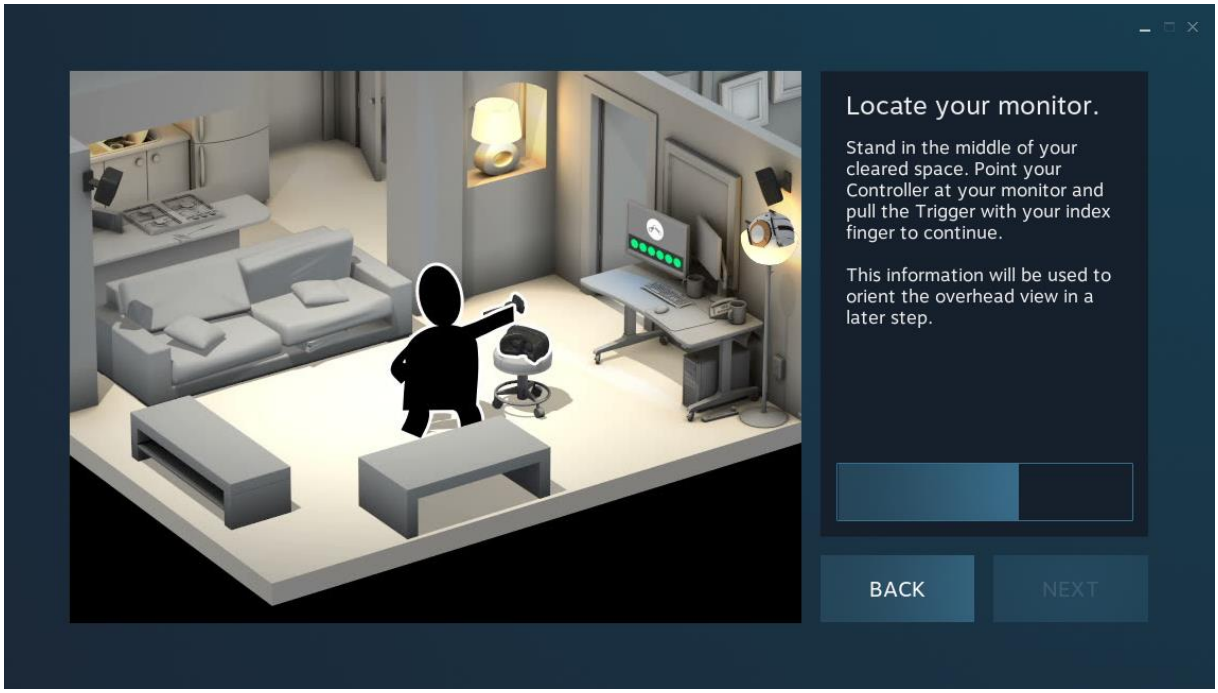
Slika 15. Odabir načina rada Steam Valve Indexa

Nakon odabranog načina rada potrebno je omogućiti praćenje VR naočala i kontrolera baznim stanicama. Nakon što su oba kontrolera i VR naočale spremni, možemo krenuti na sljedeći korak u kalibraciji.



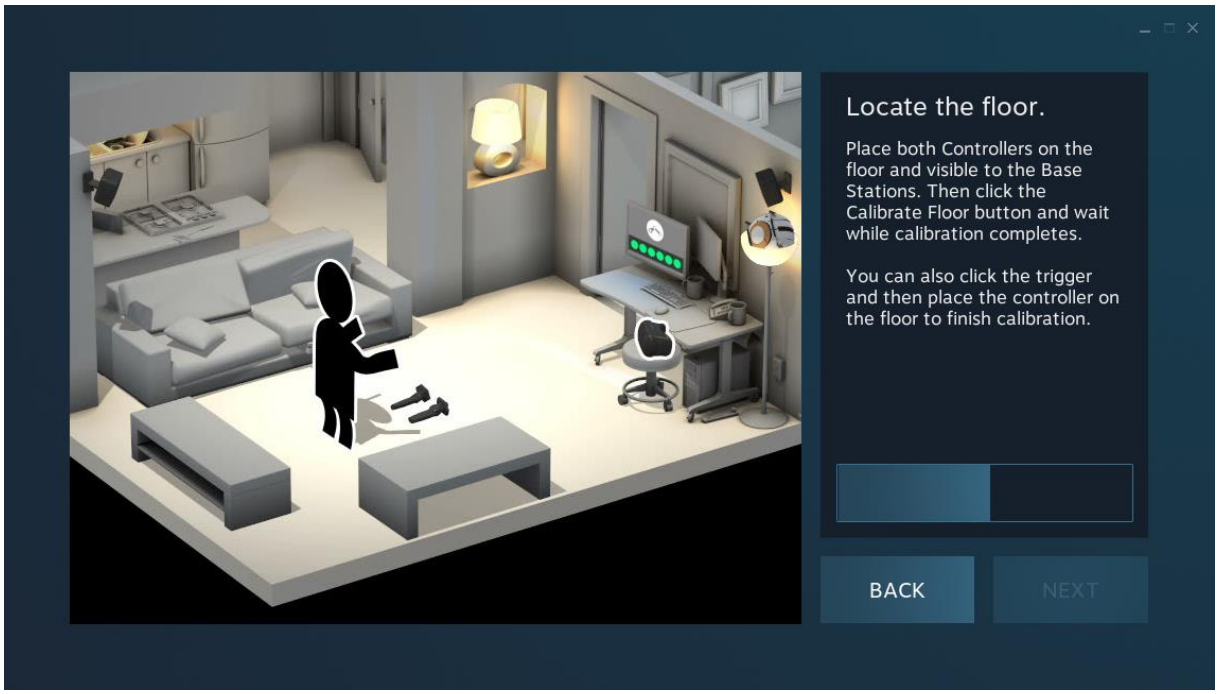
Slika 16. Detekcija VR naočala i kontrolera

Nakon što smo osigurali praćenje oba kontrolera i VR naočala od strane baznih stanica potrebno je locirati monitor, što se radi tako da se uzme jedan od kontrolera, uperi se prema monitoru i pritisne se dugme koje se nalazi ispod kažiprsta, takozvani *Trigger*.



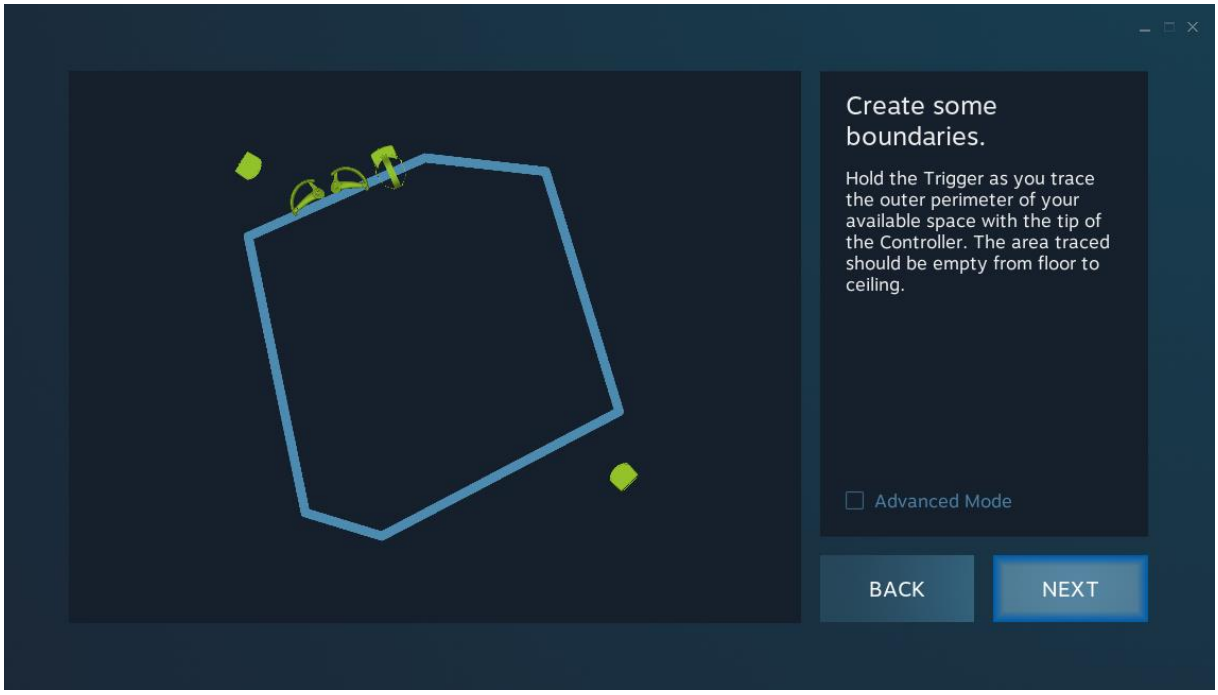
Slika 17. Lociranje monitora

Sljedeće na redu je kalibracija poda što VR sustavu omogućuje postavljanje poda na razinu koja odgovara korisnikovoj visini. Za ovaj korak potrebno je postaviti kontrolere na pod i VR naočale na glavu koji VR sustavu zatim služe kao dvije kontrolne točke na temelju kojih se radi kalibracija poda.



Slika 18. Kalibracija poda

Na samom kraju kalibracije potrebno je definirati virtualni prostor kojim će se korisnik moći kretati dok je u virtualnom svijetu. Svrha ovog koraka je postavljanje granica na način da se unutar definiranog prostora u kojem će se korisnik kretati ne nalaze nikakve prepreke koje bi ga mogle ozlijediti. Prilaskom rubu definiranog virtualnog prostora VR naočale će prikazati virtualnu barijeru koja služi za signalizaciju prilaska rubu dozvoljene zone kretanja kako bi se izbjegle ozljede.



Slika 19. Kalibracija virtualnog prostora

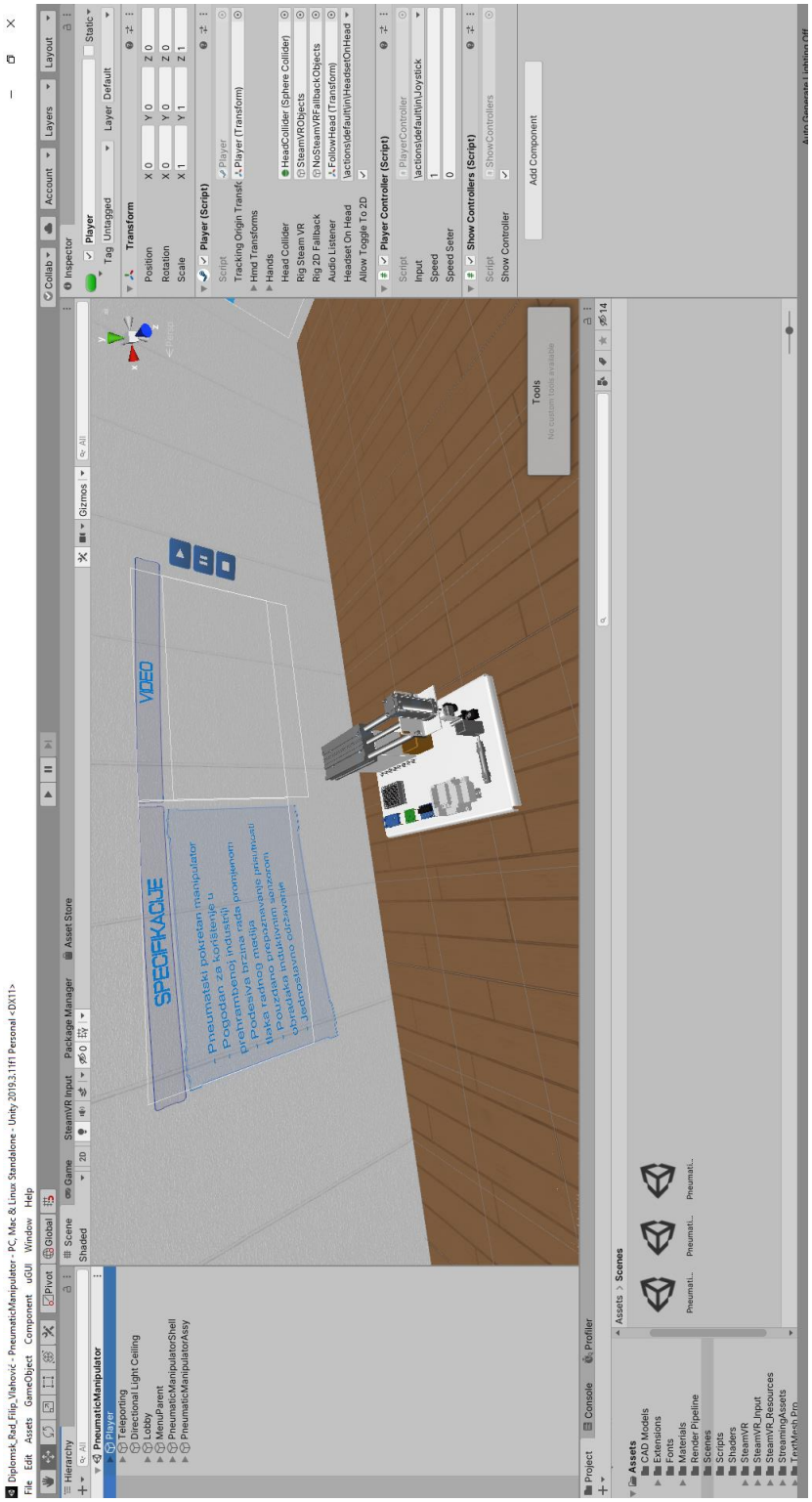
4.2. Unity

Kako je upotreba virtualne stvarnosti u mehatronici još u fazi razvoja ne postoji poseban softver za izradu takvih simulacija pa se u tu svrhu koristi Unity, softver za izradu igara na platformama kao što su Android, iOS, Windows, Linux i mnoge druge. Unity se može koristiti za stvaranje trodimenzionalnih igara, dvodimenzionalnih igara, igara s virtualnom stvarnošću, proširenom stvarnošću, kao i simulacija i drugih aplikacija. Unity su prihvatile i neke industrije izvan *gaming* industrije poput filmske i automobilske industrije, arhitekture, inženjerstva i građevine.

Unity omogućuje specificirane kompresije tekstura, mipmapa i postavke razlučivosti za svaku platformu koju mehanizam igre podržava. Pruža i podršku za mapiranje udaraca, mapiranje refleksije, dinamičke sjene pomoću mapa sjena, renderiranje tekstura i naknadnu obradu prikaza na zaslonu (eng. *post-processing*). [8]

Jedini 3D formati koje Unity prihvaća su .3ds, .dxf, .obj, .fbx i .dae, a 3D model pneumatskog manipulatora izrađen je u SolidWorksu pa je potrebno koristiti neku vrstu konverzije da bi mogli ubaciti 3D model manipulatora u Unity. Uz Unity postoje još neki razvojni softveri za izradu igara koji mogu poslužiti za izradu VR aplikacije, a neki od njih su Unreal Engine,

CryEngine i Godot. Unity je odabran jer ima najdetaljniju dokumentaciju kao i najviše korisnika što znači da je lakše doći do potrebnih informacija i rješenja određenih problema.

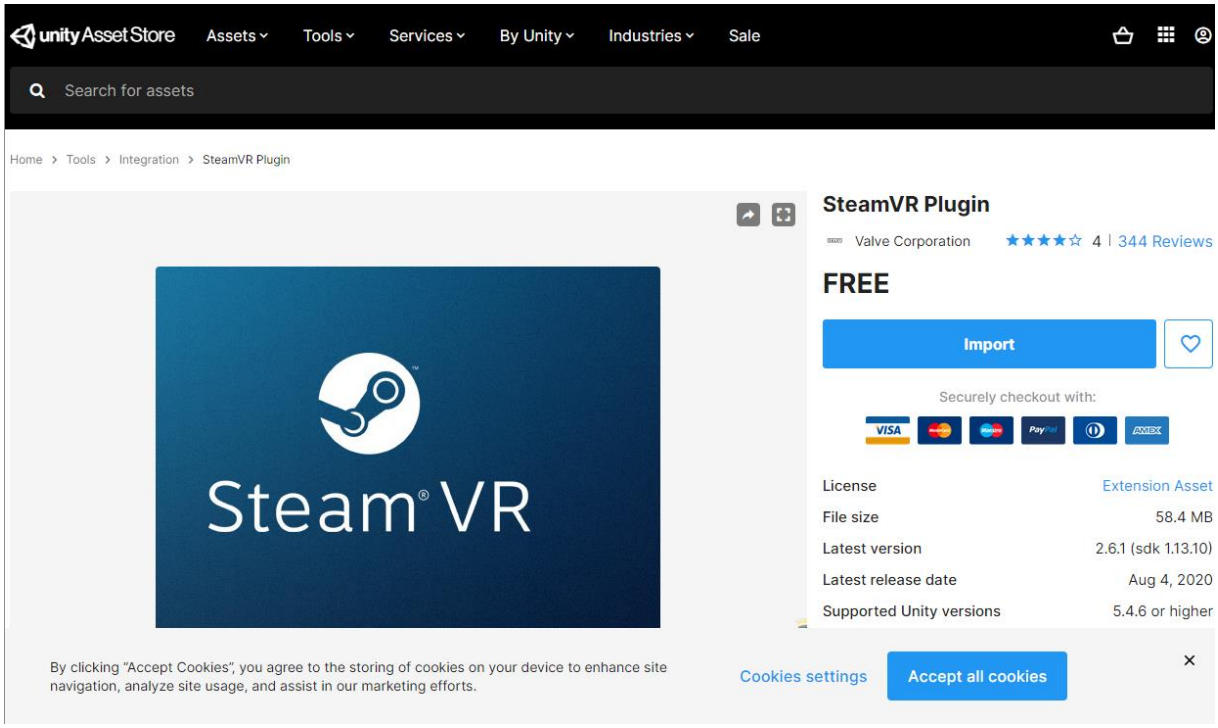


Slika 20. Unity sučelje

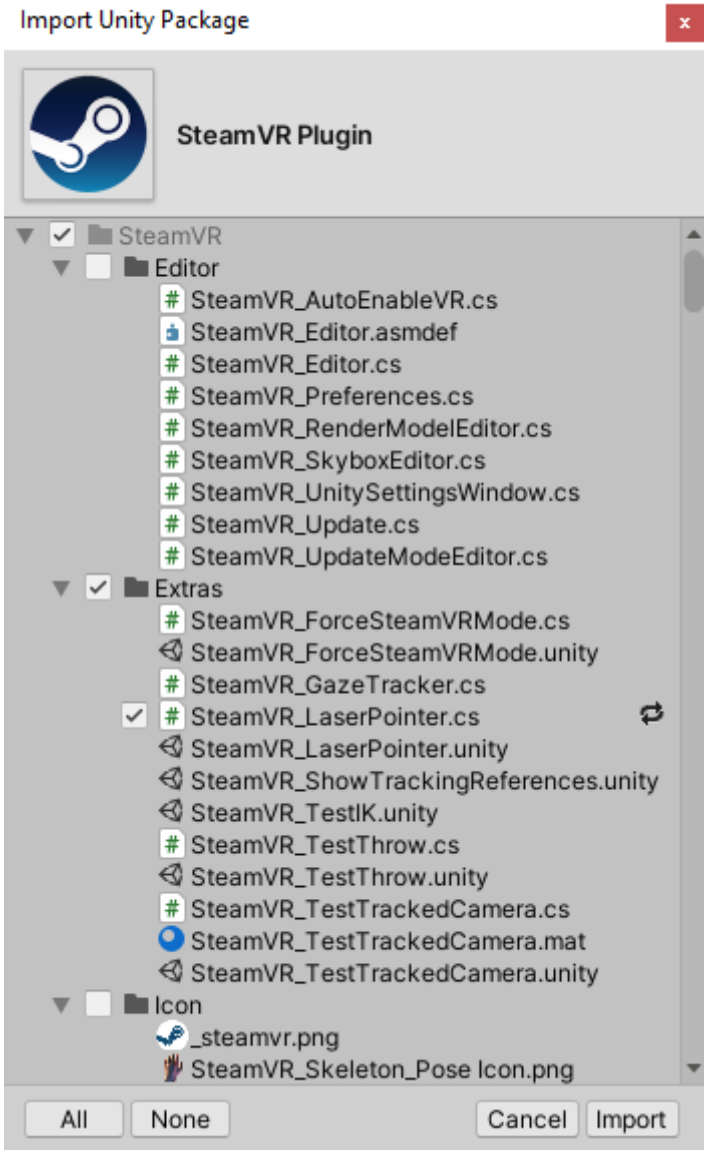
4.2.1. Postavljanje Unity softvera za izradu VR aplikacija

Nakon instalacije Unity-a potrebno je otvoriti novi projekt u kojem zatim treba podesiti pojedine opcije kako bi bilo moguće izraditi VR aplikacije.

Prvo u Unity treba dodati SteamVR paket za razvoj softvera. SteamVR se brine za tri glavne stvari koje su bitne programerima: učitavanje 3D modela za VR kontrolere, rukovanje unosa s tih kontrolera i procjenom izgleda ruke pri korištenju kontrolera. Pored upravljanja tim stvarima ovaj paket brine i o interakcijskom sustavu koji pomaže pri izgradnji temelja svake VR aplikacije, a pruža i konkretne primjere interakcije s virtualnim svijetom i aplikacijskim virtualnim sučeljima. SteamVR paket za razvoj softvera može se besplatno preuzeti kroz Unity-jevo *Asset Store* sučelje koje se otvara iz padajućeg izbornika *Window* ili kombinacijom tipki Ctrl+9.[9]

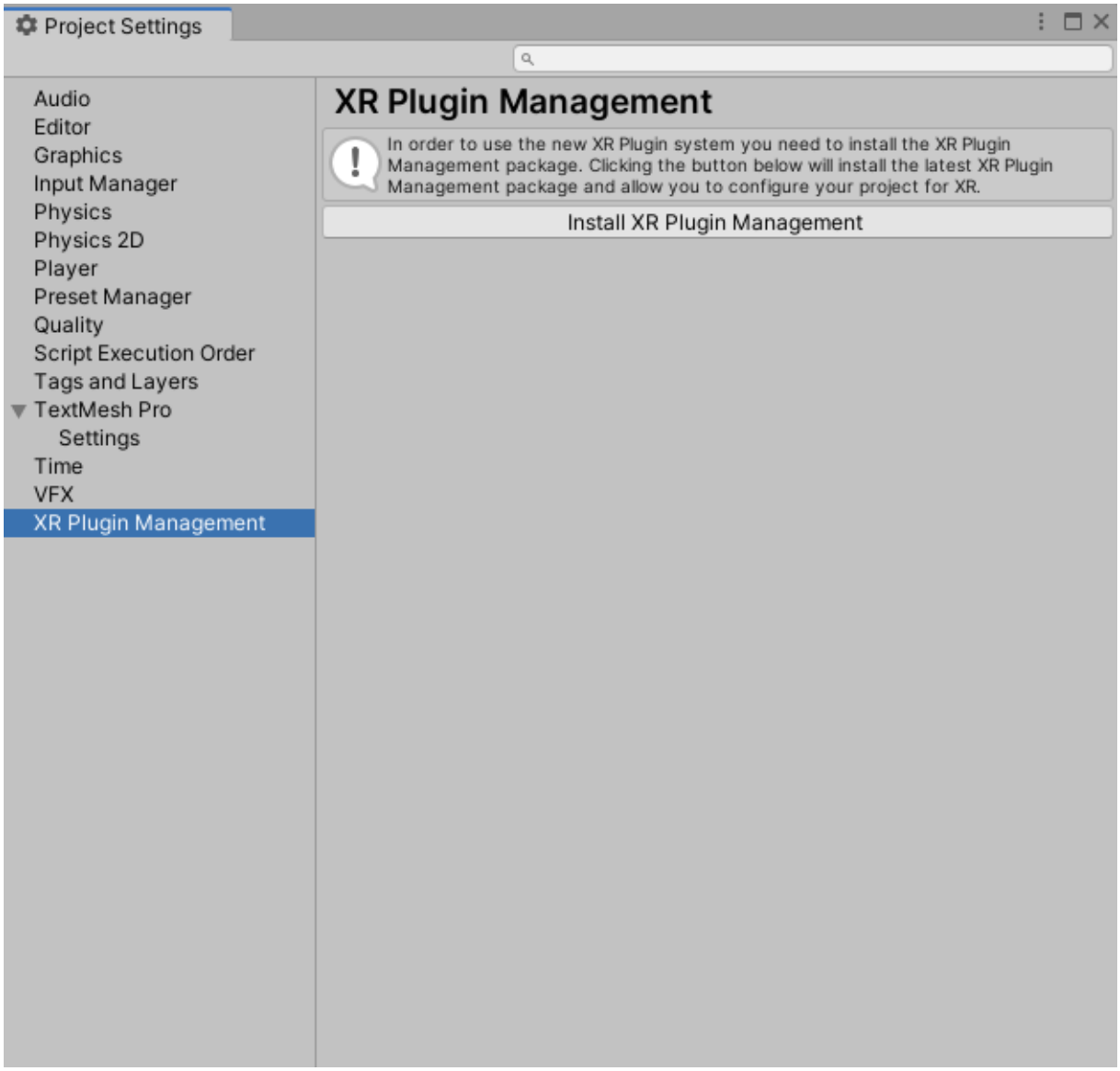


Slika 21. Preuzimanje SteamVR-a



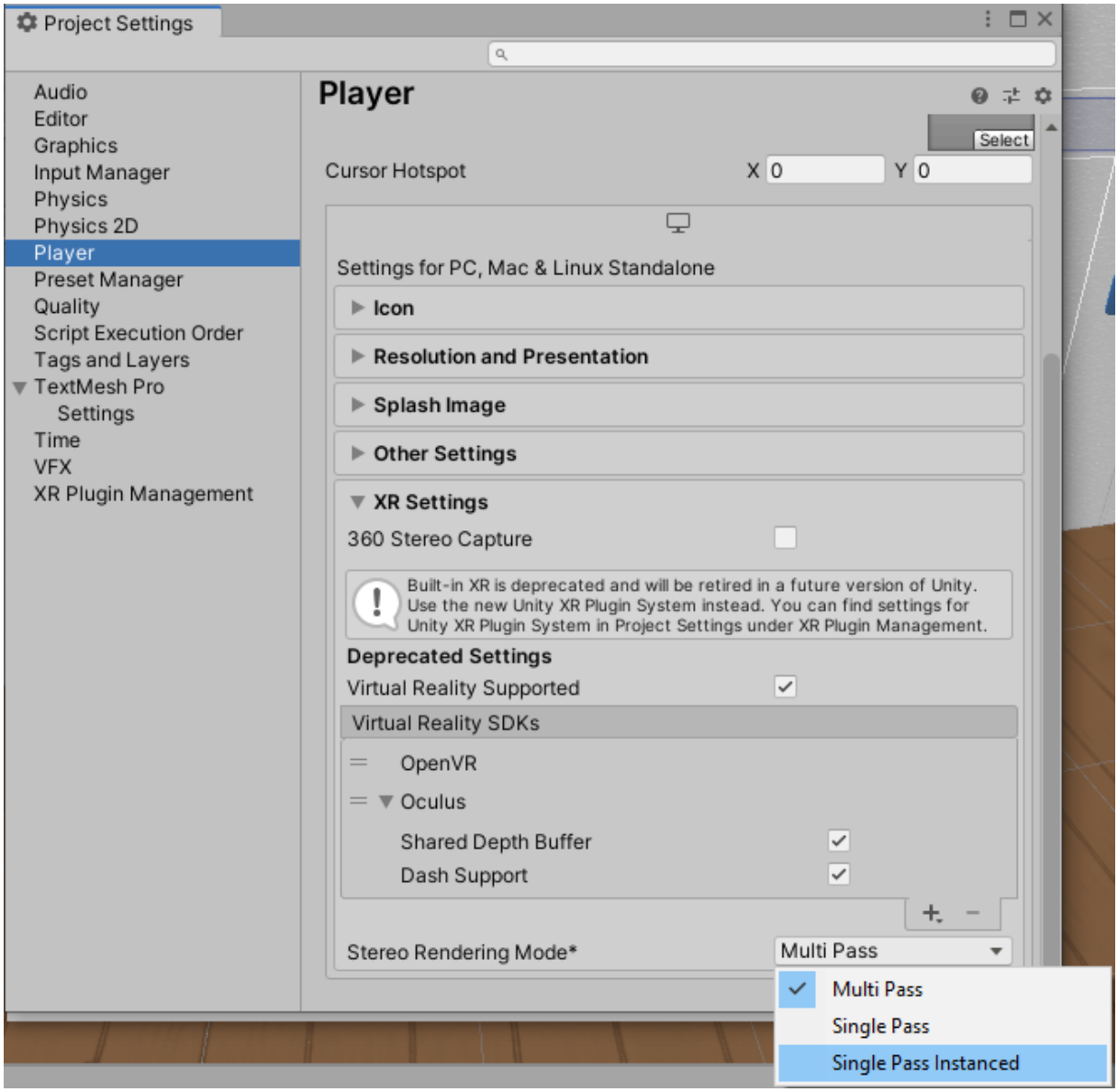
Slika 22. Ubacivanje SteamVR-a u Unity projekt

Nakon ubacivanja SteamVR-a u Unity projekt potrebno je instalirati XR Plugin Management, sučelje koje upravlja dodacima za produženu stvarnost.



Slika 23. Instalacija XR Plugin Management sučelja

Kako bi postigli bolje performanse same aplikacije prije početka izrade potrebno je još podesiti opciju *Stereo Rendering Mode*. Za izradu Windows aplikacija treba ju prebaciti sa *Multi pass* na *Signle Pass Instanced*. Razlog tome je to što odabirom *Single pass Instanced* opcije tjeramo aplikaciju da prikaze za lijevo i desno oko renderira u isto vrijeme što uvelike pomaže pri radu s kompleksnom geometrijom kao što su precizni 3D modeli dobiveni iz softverskih paketa za konstruiranje pomoću računala.[10]

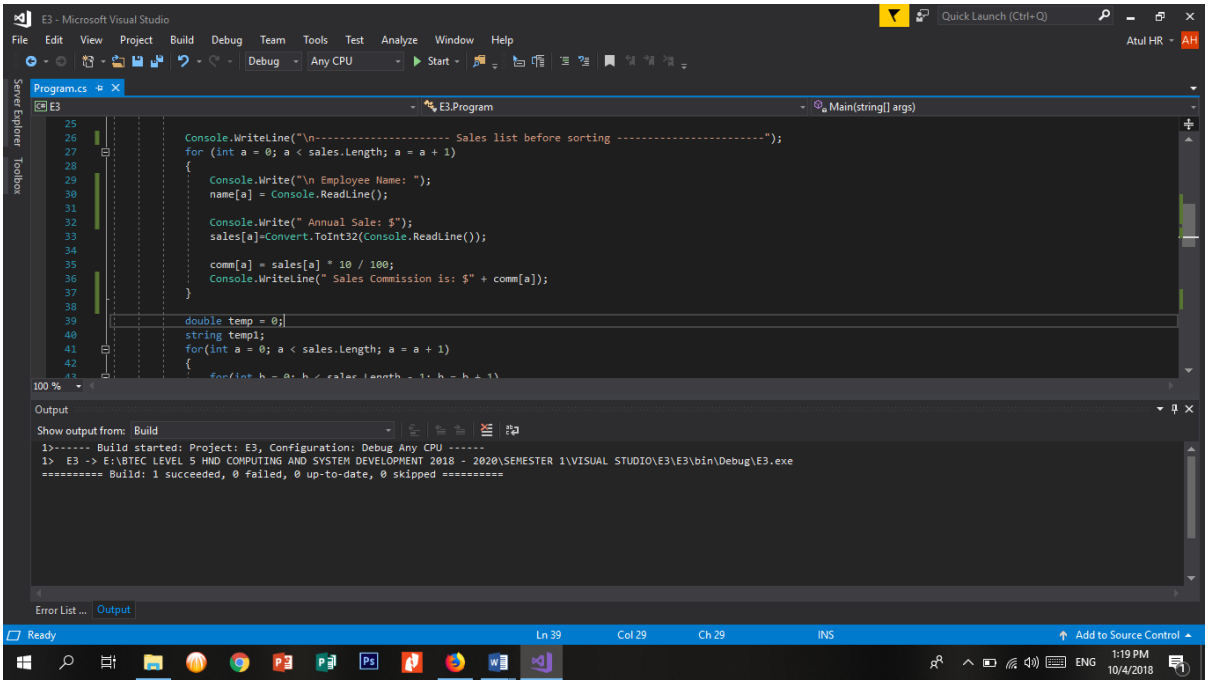


Slika 24. Postavljanje *Stereo Rendering Mode* postavke

4.3. C# i Visual Studio

Programski jezik koji se koristi u Unityu za izradu skripti i shadera je C#. Izradio ga je Anders Hjelsberg sa svojim razvojnim timom kojeg trenutno vodi Mads Torgersen. Zadnja verzija na tržištu je verzija 8.0 koja je izašla na tržište 2019. godine zajedno sa Visual Studio 2019. okruženjem za razvoj softvera. C# je objektno orjentirani programski jezik s više paradigmi i opće je namjene. Razvio ga je oko 2000. godine Microsoft kao dio svoje .NET inicijative, a kasnije su ga kao međunarodni standard odobrili ECMA i ISO.

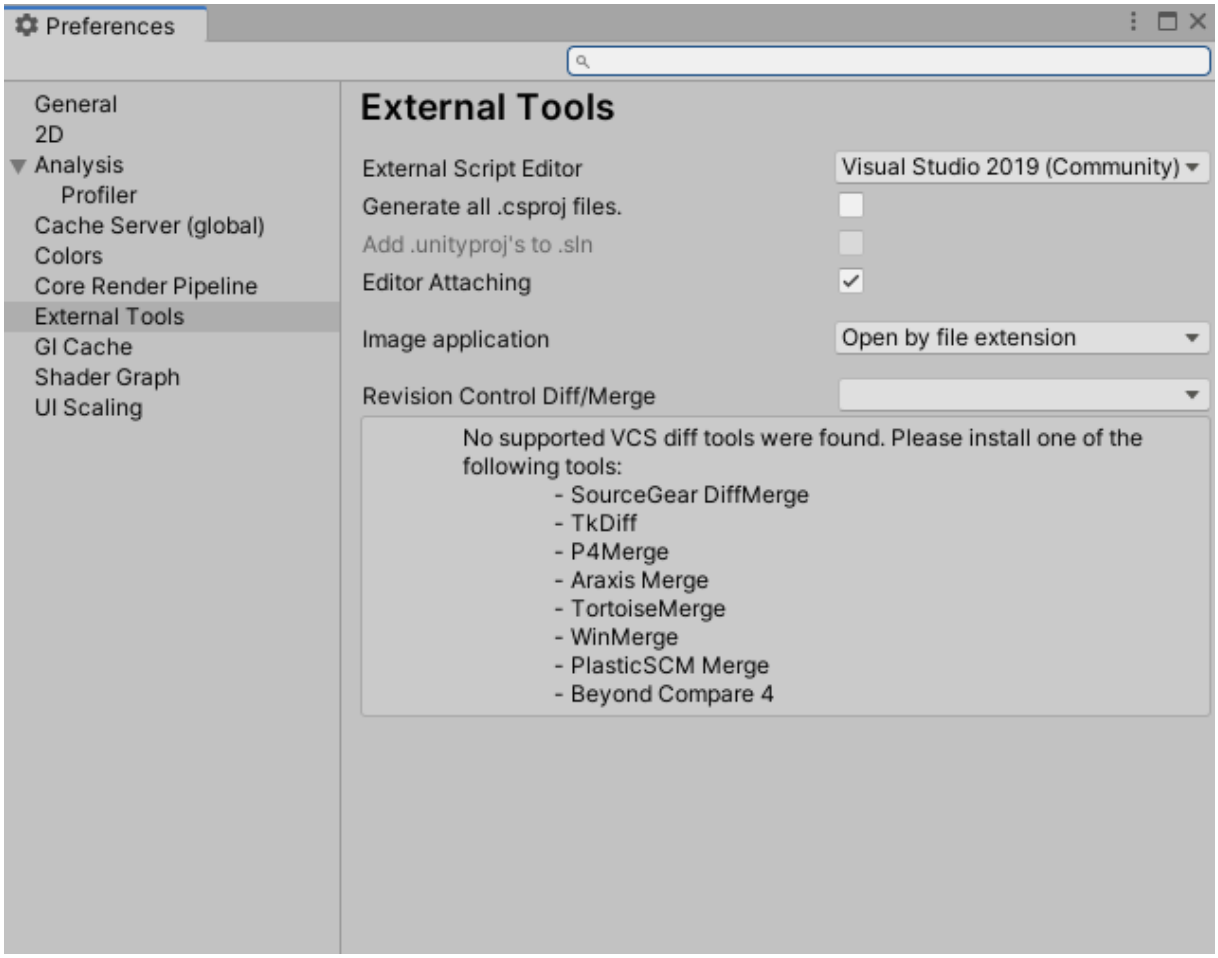
Za izradu svih skripti korišten je C#, a skripte su pisane u Visual Studiu. Prije nego što je C# postao primarni jezik za pisanje skripti u Unityu na njegovom mjestu je bio programski jezik Boo koji je uklonjen izlaskom Unitya 5 i verzijom JavaScripta pod nazivom UnityScript koji su zastarjeli 2017. godine. [11]



Slika 25. Visual Studio

Iznimno važan dio Visual Studia je IntelliSense. IntelliSense je opći pojam za razne značajke za uređivanje koda, uključujući: dovršavanje koda, informacije o parametrima, brze informacije i popise klasa, objekata, varijabli, tipova podataka i td. Značajke IntelliSense-a pokreću servisi pojedinih programskih jezika. IntelliSense omogućuje dovršavanje koda na temelju semantike jezika i analize postojećeg izvornog koda. Ako postoji mogućnost dovršavanja koda prijedlozi će se pojaviti tokom pisanja dijelova koda. Filtrira se tako da uključuje samo članove koji sadrže već upisane znakove. Pritiskom na tipke tab ili enter odabrani unos se upisuje u kod i na taj način olakšava programiranje.[12]

Intellisense nije odmah uključen za rad s Unity projektima pa ga je potrebno uključiti što se može iz *Preferences* prozora kojeg se otvara iz padajućeg izbornika *Edit*. Odabirom opcije *External Tools* otvara se prozor u kojem je pod *External Script Editor* potrebno odabrati Visual Studio, a da bi se moglo debugirati kod potrebno je još omogućiti opciju *Editor Attaching*.



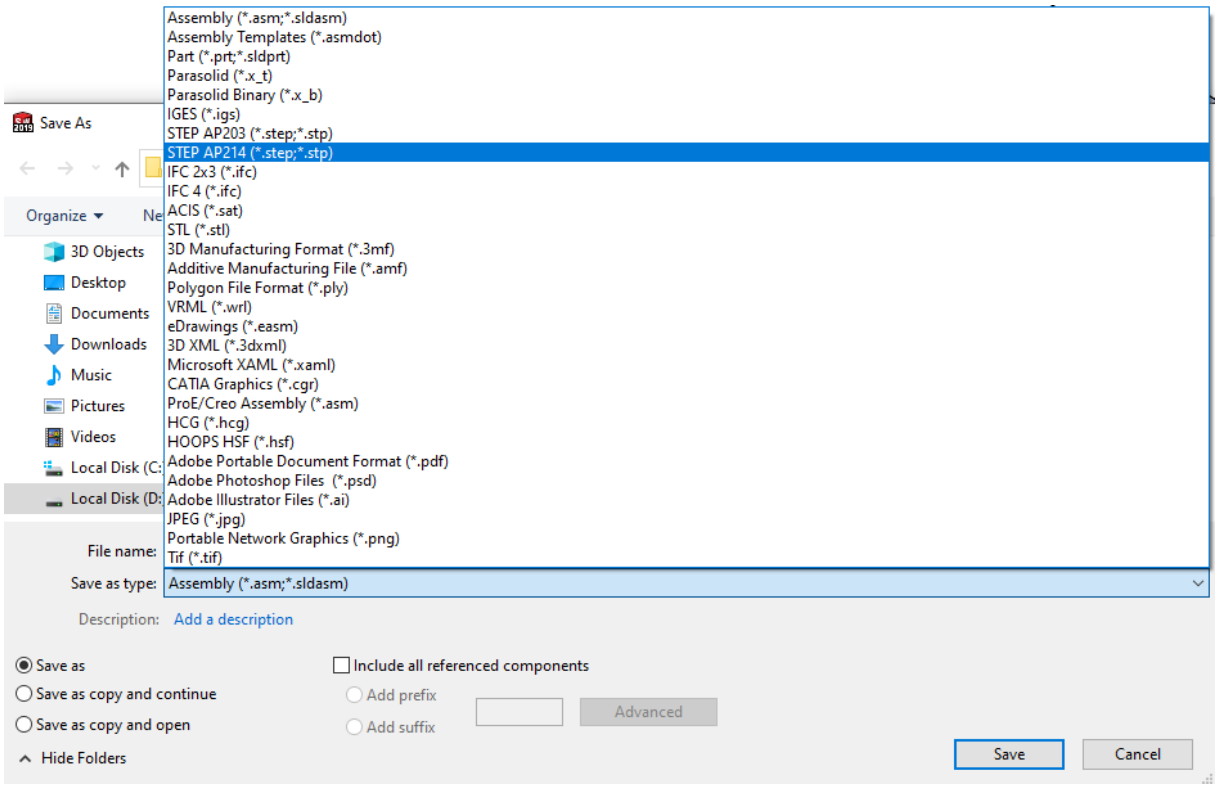
Slika 26. UključivanjeVisual Studio IntelliSensea

4.4. Konverzija i ubacivanje 3D modela iz softvera za konstuiranje pomoću računala u Unity

Za modeliranje geometrije u CAD-u koristio se programski paket SolidWorks. SolidWorks je programsko rješenje koje objavljuje Dassault Systèmes za konstruiranje pomoću računala (CAD), proizvodnju pomoću računala (CAM) i druge razne analize koje se koriste pri konstruiranju. Ovaj CAD softver se najčešće izvodi na operacijskom sustavu Microsoft Windows. Iako je moguće pokrenuti SolidWorks na MacOS-u, SolidWorks to ne podržava. Prema izvođaču, više od 2 000 000 inženjera i dizajnera u više od 165 000 tvrtki koristilo je SolidWorks 2013. godine što ga čini iznimno rasprostranjenim i korištenim od strane inženjera i dizajnera u svijetu.[13]

4.4.1. Pretvorba .SLDASM formata u .STEP format

Kako je 3D model pneumatskog manipulatora izrađen u svrhu konstruiranja i izrade samog manipulatora, a ne u svrhu izrade ove simulacije potrebno je neki od izlaznih formata koje nudi SolidWorks pretvoriti u neki od ulaznih formata koje podržava Unity. Za pretvorbu .STEP formata dobivenog eksportiranjem .SLDASM iz SolidWorksa u .obj format koji podržava Unity koristit će se Autodesk Inventor. Za spremanje .SLDASM formata kao .STEP format u SolidWorks-u je potrebno iz padajućeg izbornika odabrati opciju *Save As* te u prozoru koji se zatim otvori kao izlazni format odarati .STEP



Slika 27. Pretvorba .SLDASM formata u .STEP format

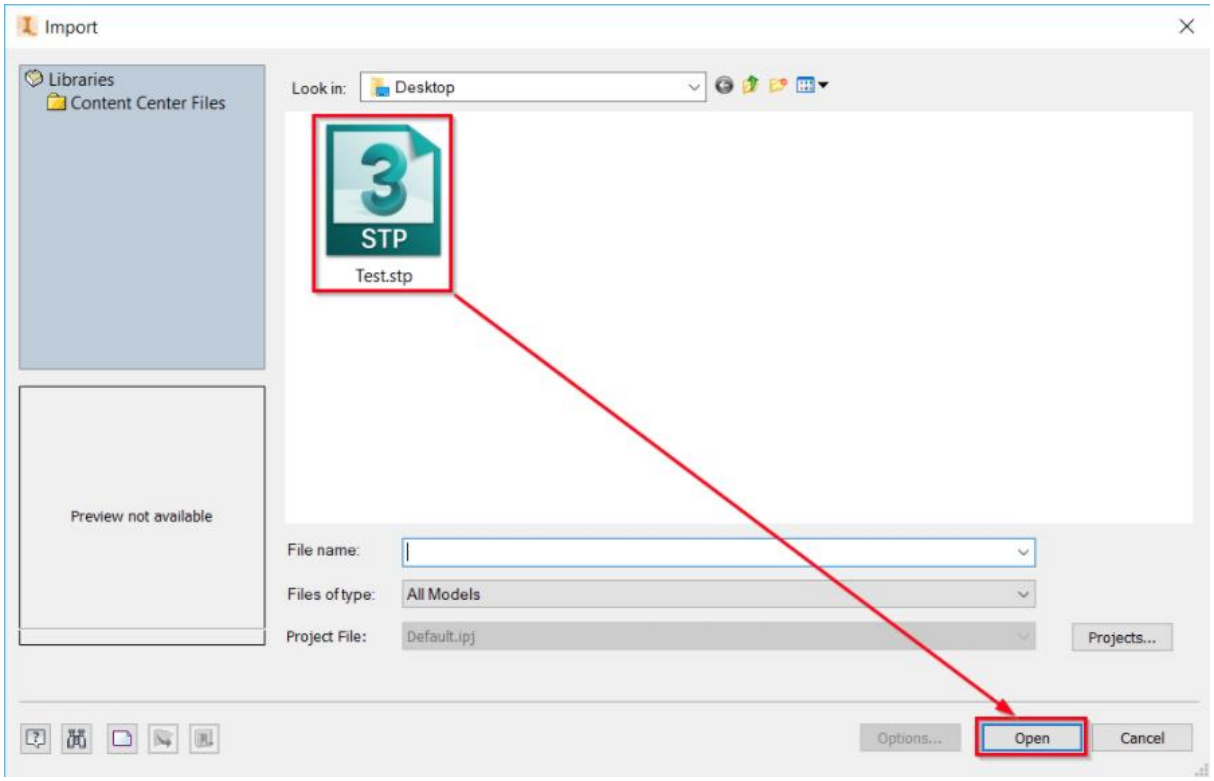
4.4.2. Pretvorba .STEP formata u .obj format

Autodesk Inventor je, kao i SolidWorks, softverski paket za konstruiranje pomoću računala (CAD), izradu simulacija, vizualizaciju i izradu dokumentacije. Inventor je odabran jer je jedan od rijetkih softverskih paketa korištenih za konstruiranje pomoću računala koji može konvertirati neki od svojih ulaznih formata u .obj format ili čak otvarati .obj datoteke. Pretvorba se radi tako da se u Inventor najprije učita .STEP datoteka dobivena iz SolidWorksa, a zatim se otvoreni CAD model spremi kao .obj datoteka.

Da bi se .STEP datoteka učitala u Inventor prvo je potrebno odabrati opciju *Import*.

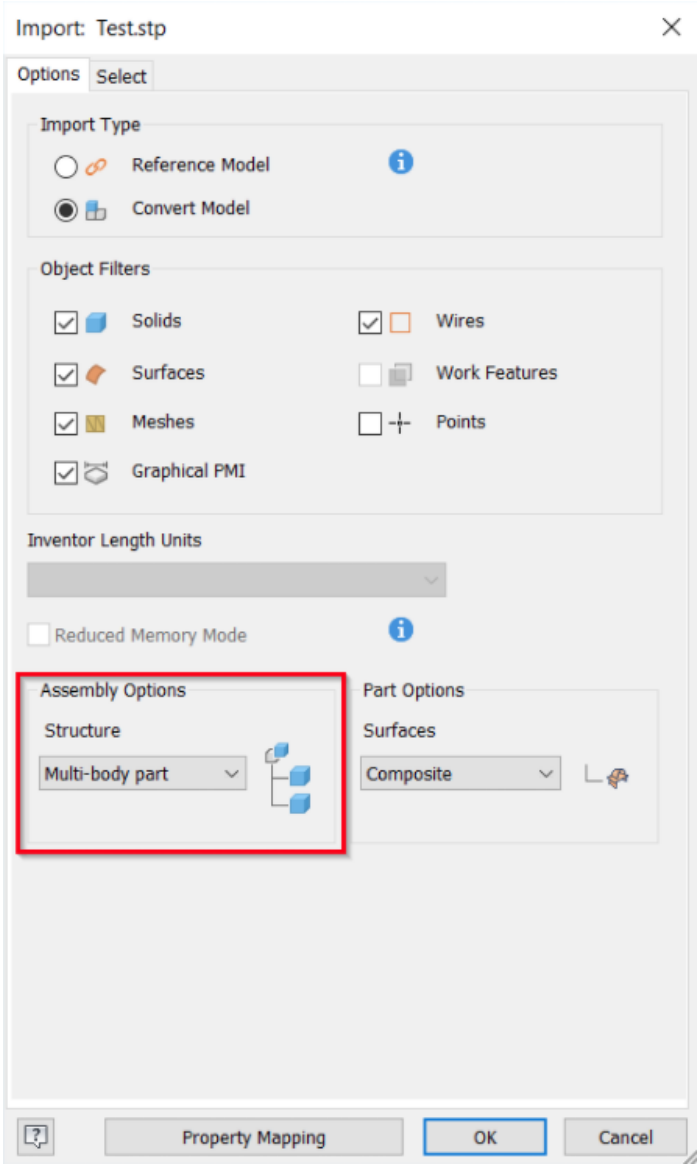
Slika 28. Odabir opcije Import

Nakon odabira opcije I*mport* otvara se prozor u kojem je potrebno navigirati do željene datoteke, odabrati ju te stisnuti gumb *Open*.



Slika 29. Otvaranje .STEP datoteke

Za dovršavanje uvoza .STEP datoteke u sljedećem prozoru koji se automatski otvara pritiskom na gumb *Open* potrebno je pod opcijom *Structure* odabrati *Multi-Body Part.*

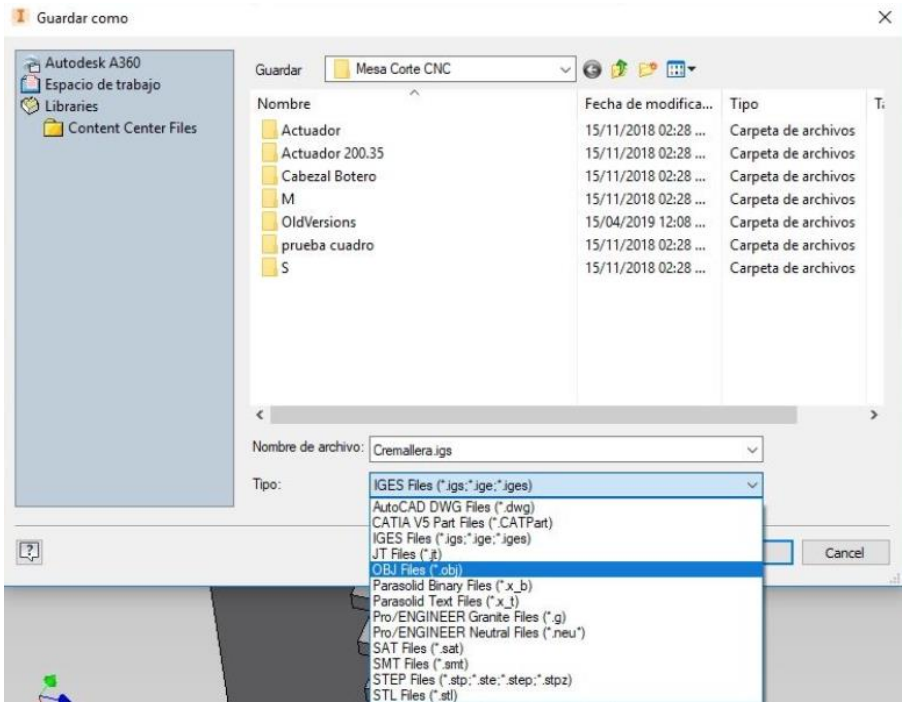


Slika 30. Postavke uvoza .STEP datoteke

Ovdje počinje pretvorba u .obj format koji će zatim biti moguće uvesti u Unity. Da bi se otvoreni sklop spremio kao .obj file prvo je potrebno iz padajućeg izbornika odabrati *File* i zatim opciju *Export*.

Slika 31. Odabir opcije *Export*

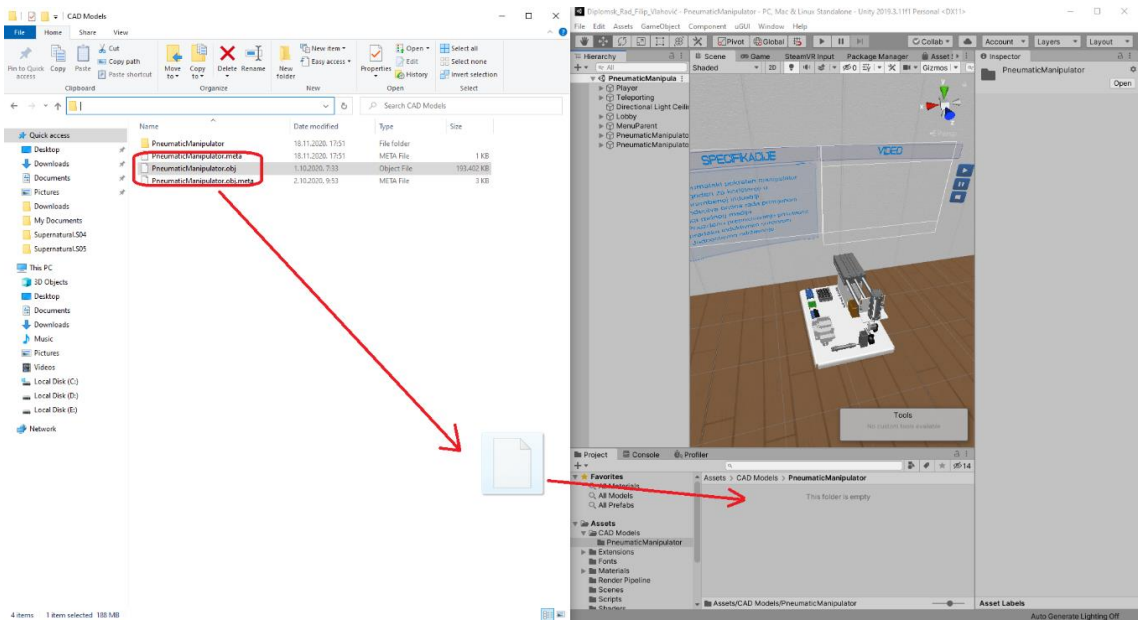
Po odabiru opcije *Export* otvara se novi prozor u kojem je potrebno kao izlazni format odabrati .obj i spremiti datoteku.



Slika 32. Spremanje .obj datoteke

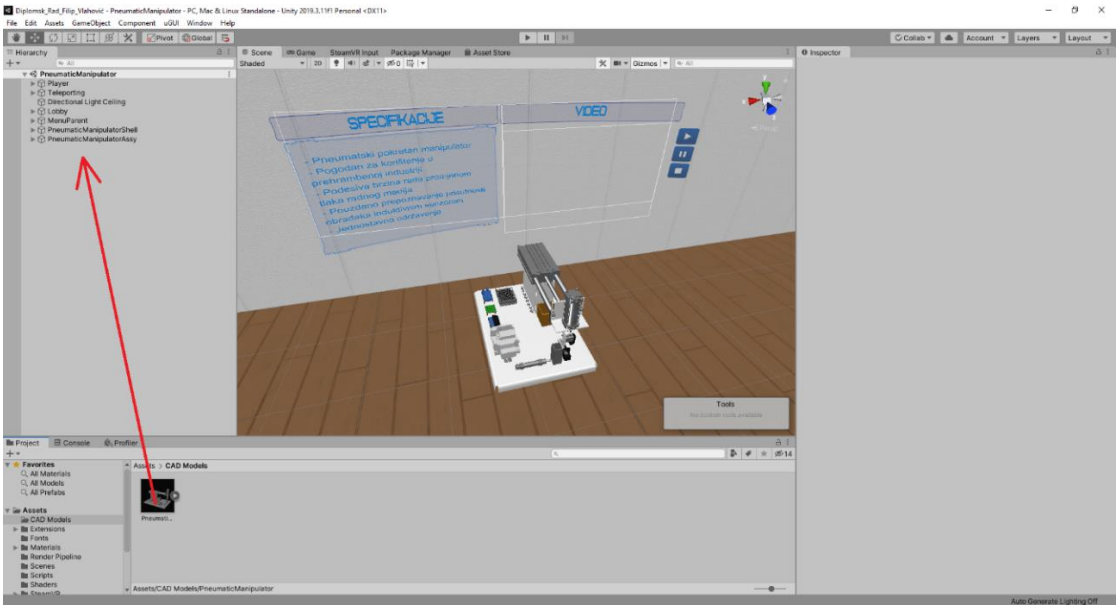
4.4.3. Uvoz 3D geometrije u Unity

Nakon pretvorbe u prihvatljiv .obj format potrebno je sklop uvesti u Unity projekt, što je iznimno jednostavno i radi se takozvanom *drag & drop* radnjom. Otvore se Unity projekt u koji želimo uvesti 3D geometriju i mapa u kojoj se nalazi ta 3D geometrija te se mapa samo „prenese i pusti“ u Unity projekt.



Slika 33. Uvoz 3D geometrije u Unity

Po završetku konverzije i uvoza 3D geometrije u Unity za prikaz 3D geometrije u sceni još je samo podrebno napraviti *drag & drop* geometrije u hijerarhijsko stablo scene.

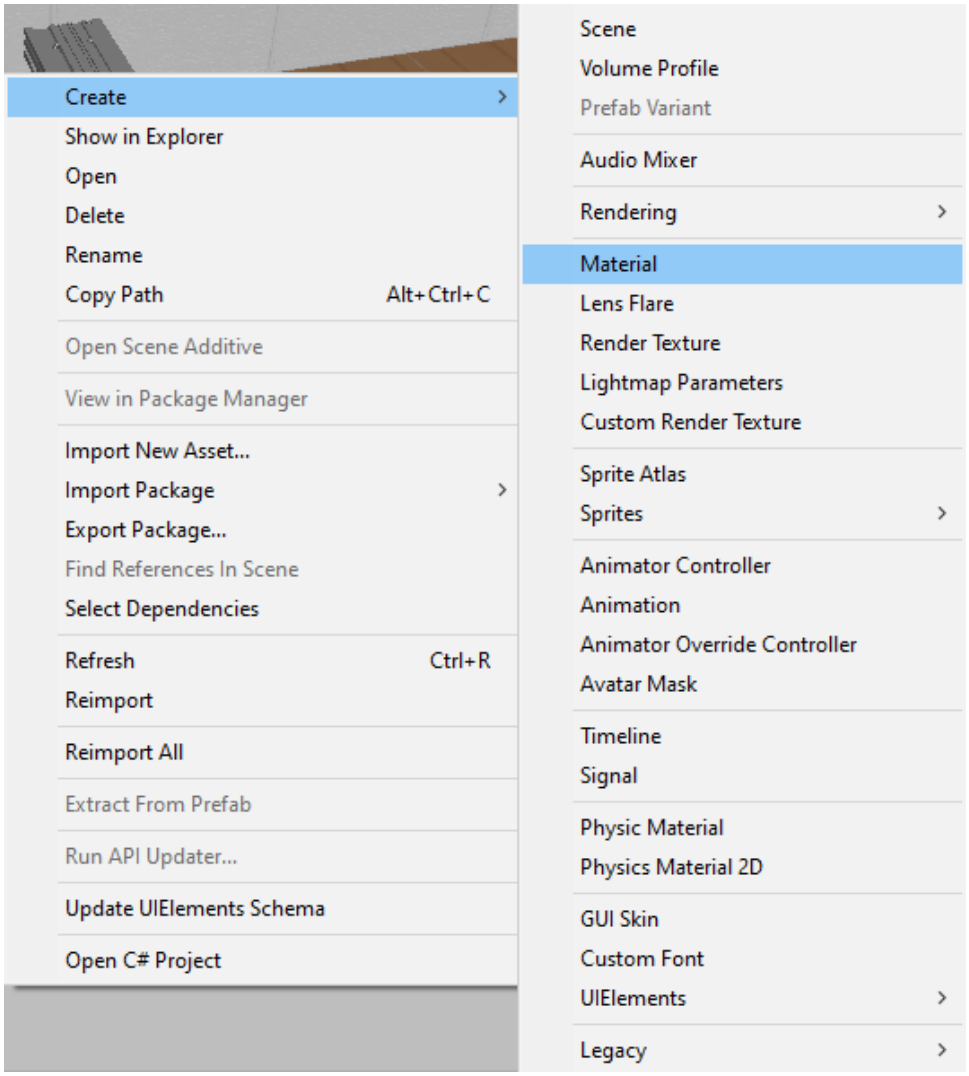


Slika 34. Ubacivanje geometrije u hijerarhijsko stablo

4.4.4. Izrada i dodjeljivanje materijala pojedinim 3D modelima unutar sklopa manipulatora

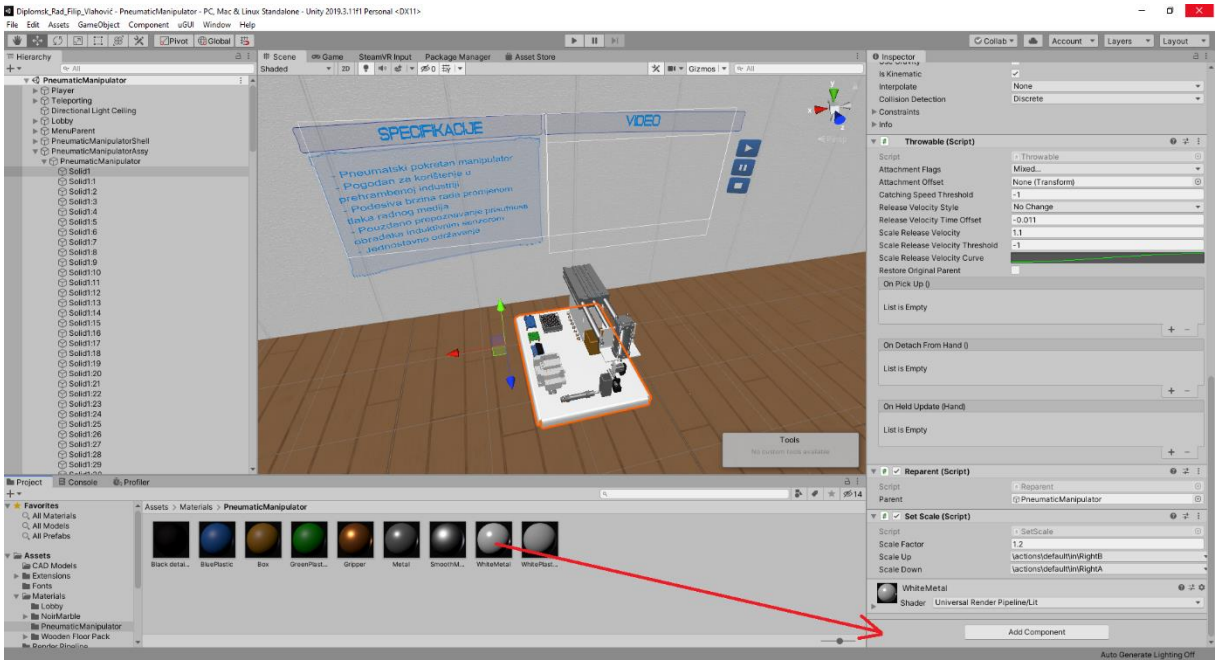
Sva uvezena geometrija je bijela i ne posjeduje nikakve teksture koje .obj format sprema i razlikuje. Razlog tome je što Unity ne prepoznaje boje i teksture pohranjene u .obj format. Teksture i boje se određenoj geometriji dodaju na način da se kreiraju materijali kojima se dodijele boja, tekstura, kao i druge određene mape koje utječu na izgled i teksturu same 3D geometrije.

Novi materijal se izrađuje pritiskom desne tipke miša na prazan prostor *Project* dijela Unity sučelja nakon čega se otvara izbornik iz kojeg se odabere *Create* i zatim *Material*.



Slika 35. Izrada materijala

Izrađeni materijal se pojedinom 3D modelu dodjeljuje *drag & drop* funkcijom na način da se odvuče na samo dno *Inspector*-a.



Slika 36. Dodjeljivanje materijala 3D modelu

Postupak dodjeljivanja i izrade materijala ponovljen je za sve ostale dijelove koji se nalaze u sklopu pneumatskog manipulatora kako bi se dobio vjeran prikaz manipulatora u virtualnoj stvarnosti.

5. IZRADA RADNE OKOLINE

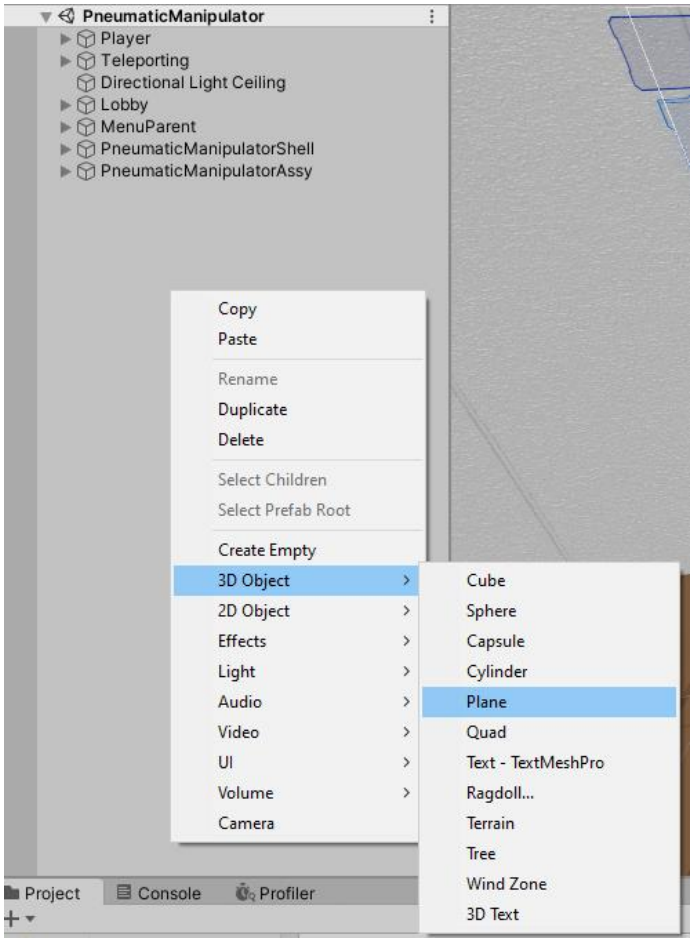
Kako se rad temelji na tri različita modula od kojih svaki koristi istu radnu okolinu u ovom dijelu će se opisati izrada radne okoline i izrada dva sustava kretanja kroz virtualni prostor koji će se onda koristiti u sva tri modula rada.

5.1. Radna okolina

Radna okolina sastoji se od šest ravnina spojenih na način da tvore jednu prostoriju. Podu, stropu, kao i zidovima dodijeljeni su materijali na način opisan u poglavlju 4.4.4. Izrada i dodjeljivanje materijala pojedinim 3D modelima unutar sklopa manipulatora. Materijali su dodijeljeni kako bi se stekao bolji dojam uronjenosti u virtualni svijet i kako bi se postigao bolji izgled same aplikacije. Za osvjetljenje radne okoline korišteno je devet svjetala nazvanih *Directional Light*.

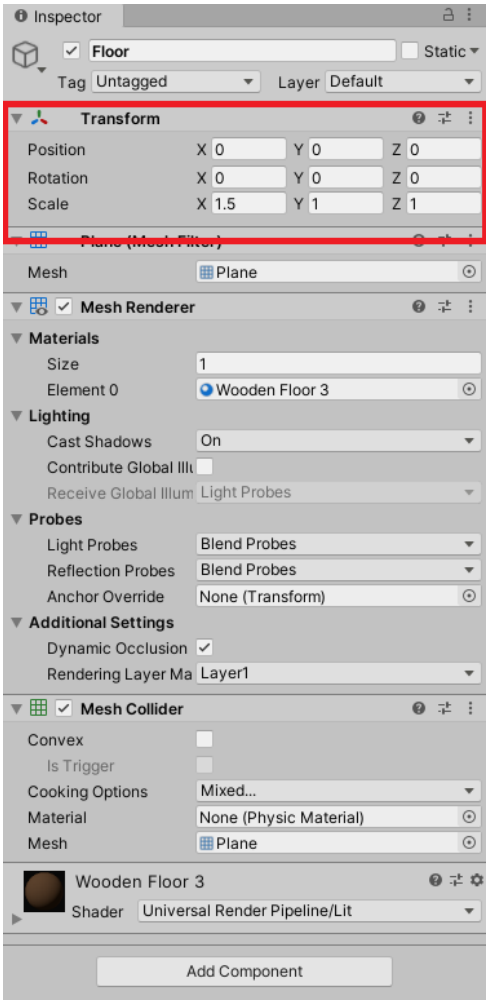
5.1.1. Izrada prostorije

Pod, zidovi i strop izrađeni su od dvodimenzionalnih ravnina. Ravnine se rade na način da se na prazan prostor hijerarhije scene klikne desni klik miša, odabere se 3D *Object* i zatim *Plane*.



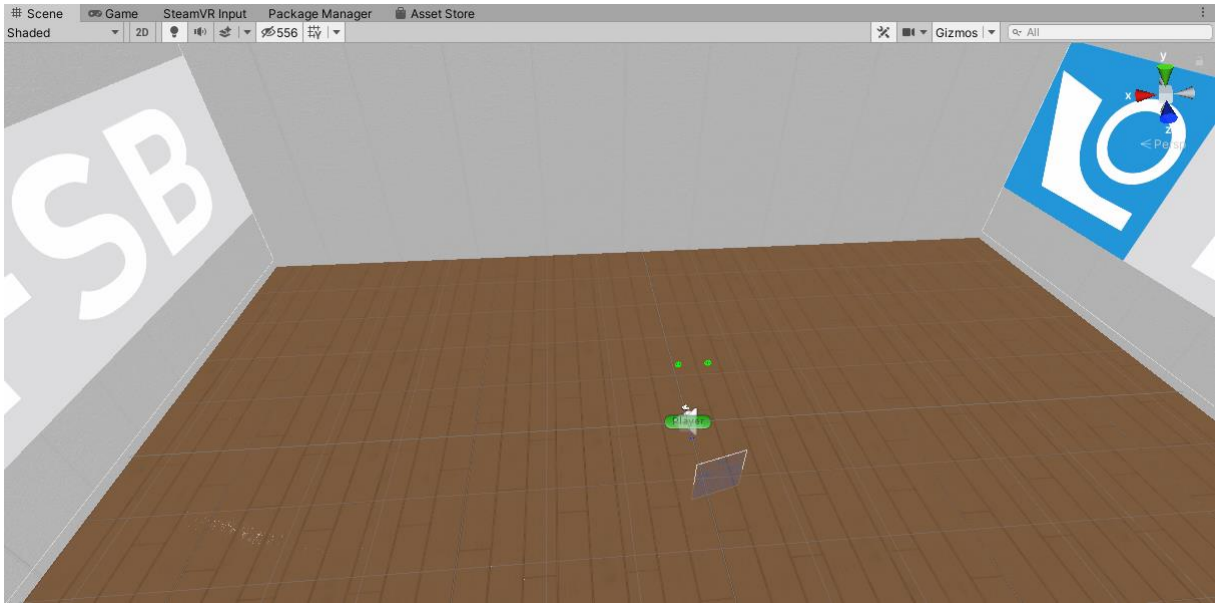
Slika 37. Izrada *Plane* značajke

Nakon izrade *Plane* značajki potrebno im je promijeniti rotacije i dimenzije kako bi tvorile sobu, što se radi u *Inspector* dijelu Unity sučelja na način da se prvo označi *Plane* kojem je potrebno promijeniti dimenzije i zatim se u *Transform* dijelu *Inspector*-a mijenjaju rotacije, položaj i veličina pojedinog *Plane*-a.



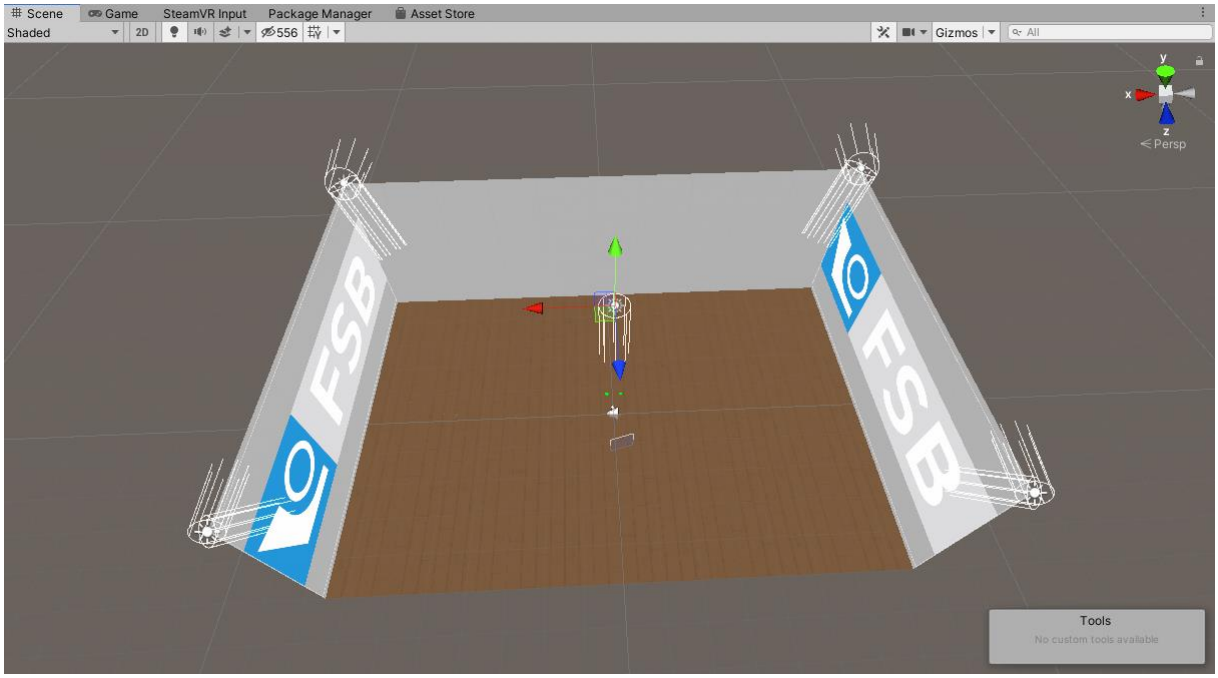
Slika 38. Podešavanje parametara *Plane* značajke

Po završetku podešavanja parametara dobije se oblik sobe kojoj se zatim dodaju materijali kako bi se dobio što vjerniji prikaz željene okoline.



Slika 39. Prikaz gotove sobe

Za osvjetljenje su korištena svjetla tipa *Directional Light* koja su postavljena na svaki kut sobe i to na način da su u svakom kutu sobe po dva svjetla koja gledaju jedno pod 45° prema gore, a drugo pod 45° prema dolje što daje osam svjetla, no korišteno je još jedno *Directional Light* svjetlo postavljeno na sredinu scene na način da gleda ravno prema dolje.



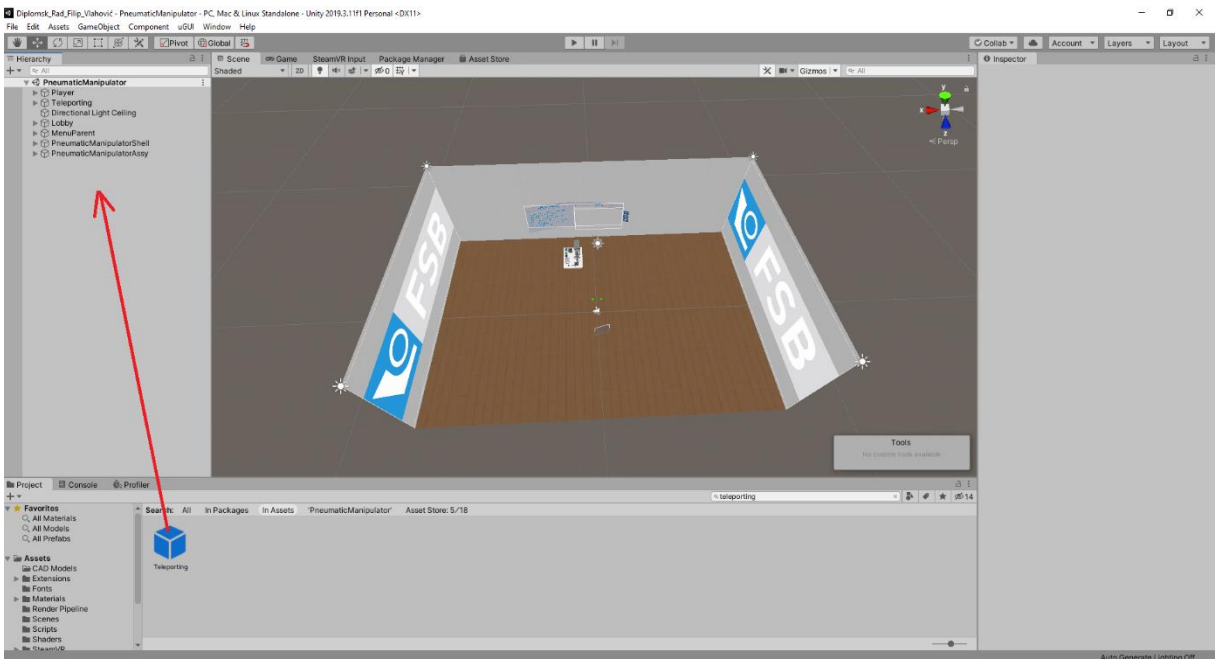
Slika 40. Postavljena svjetla

5.2. Sustavi kretanja kroz virtualni prostor

U aplikaciji postoje dva sustava kretanja kroz virtualni prostor. Jedan je teleportacijom na unaprijed predviđena mjesta, ravnine ili objekte, a drugi je slobodnim gibanjem kroz cijeli prostor. Dva sustava postoje kako bi se novim korisnicima omogućilo nesmetano korištenje aplikacije, jer korištenje VR sustava ponekad može izazvati bolest kretanja koja dovodi do vrtoglavice, a ponekad i povraćanja.

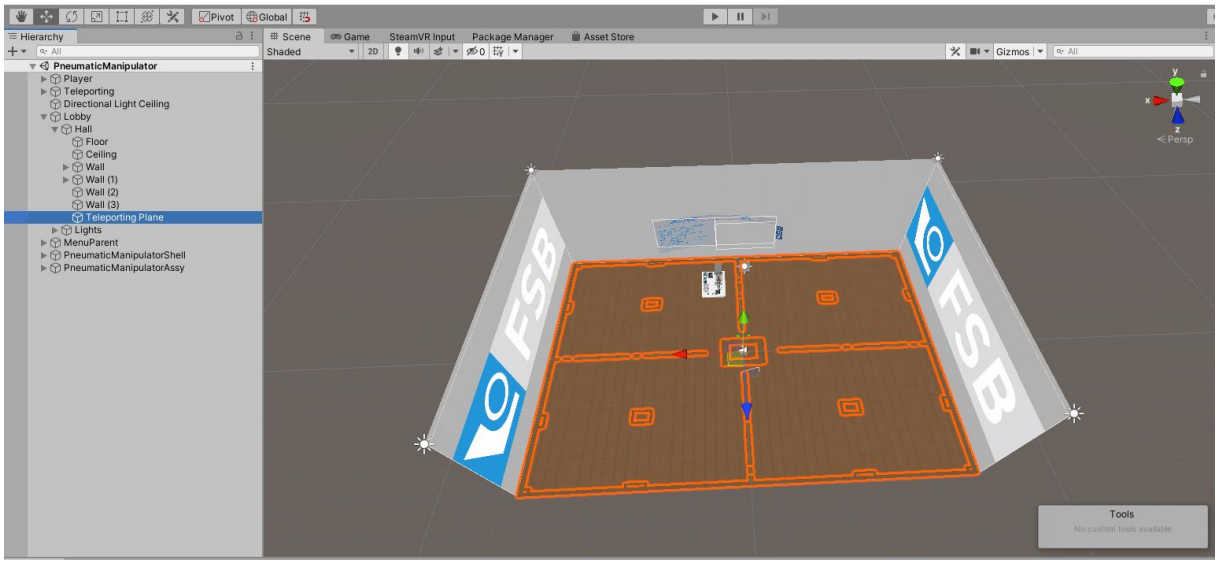
5.2.1. Teleportacija

Za teleportaciju se koristi unaprijed izrađena klasa koja se nalaze u SteamVR paketu za razvoj softvera, a obuhvaća pokazivač kojim se odabire željeno mjeto teleportacije, popratne zvukove teleportacije, značajke za označavanje površina, mjesta ili objekata unutar scene na koja se moguće teleportirati kao i sve popratne skripte koje upravljaju samom teleportacijom. Da bi bilo moguće implementirati sustav teleportacije najprije treba napraviti *drag & drop Teleporting* klase u hijerarhiju scene u koji želimo implementirati teleportaciju.[14]

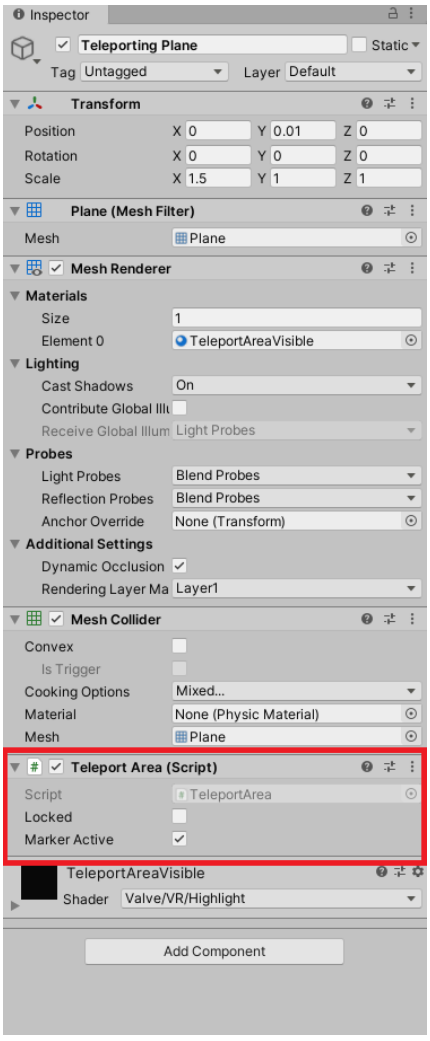


Slika 41. Dodavanje *Teleporting* klase u hijerarhiju scene

Još ostaje samo označiti mjesta na koja će biti moguće teleportirati se za što je odabrana cijela površina poda radne okoline. Da bi se pod radne okoline pretvorio u zonu teleportacije potrebno je sastavu radne okoline dodati još jednu ravninu koja će se zvati *Teleporting Plane*. Nova ravnina treba biti istih dimenzija kao i pod, treba biti u potpunosti transparentna i potrebno joj je kroz *Inspector* dodati skriptu *Teleport Area*.



Slika 42. Dodavanje *Teleporting Plane* ravnine

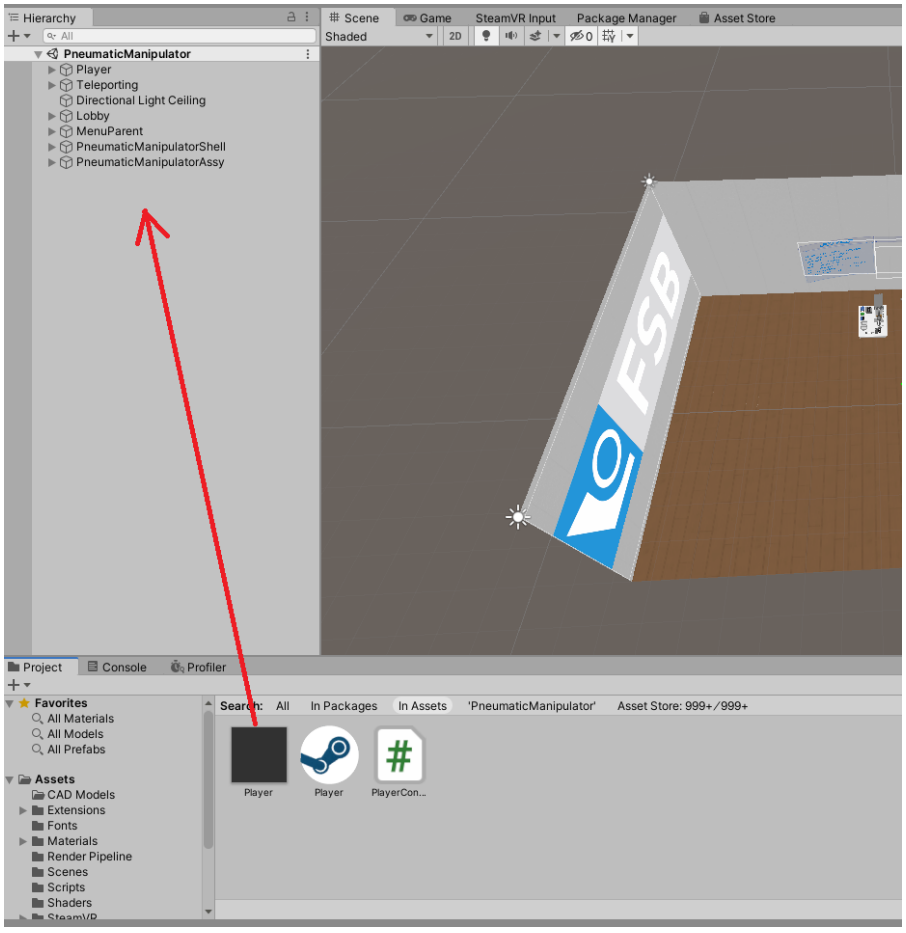


Slika 43. Dodavanje *Teleporting Area* skripte

5.2.2. Slobodno gibanje

Iako je potrebno neko vrijeme da se korisnik navikne na slobodno gibanje kroz virtualnu stvarnost, ono pruža bolji osjećaj uronjenosti u virtualni svijet kao i fleksibilnije kretanje njime. Korištenjem slobodnog gibanja lakše se pozicionirati na željeni dio koji se pregledava, a samim time moguće ga je i pogledati iz bilo koje perspektive što je velika prednost u odnosu na teleportaciju.

Za izradu sustava za slobodno kretanje prvo je potrebno u hijerarhiju scene ubaciti klasu *Player* koja u sebi sadrži potrebne skripte i komponente koje omogućuju praćenje kontrolera, VR naočala, kao i svih operacija koje se rade s kontrolerima. *Player* klasa također ima ugrađenu kameru i idealna je za izradu VR aplikacija koje koriste Steam Valve Index. *Player* klasa radi na način da je moguće imati samo jednu klasu po sceni.[14]



Slika 44. Ubacivanje *Player* klase u hijerarhiju scene

Ubacivanjem *Player* klase u scenu dobiven je statičan lik koji se ne može gibati ni u jednom smjeru, već je moguće samo okretati glavu s VR naočalama kako bi se okrenula kamera i tako pregledavati virtualni svijet te pomicati ruke unutar virtualnog svijeta pomicanjem kontrolera. Da bi bilo moguće slobodno se kretati kroz virtualni svijet potrebno je napraviti skriptu koja će to omogućiti. Skripta je nazvana *PlayerController* i slijedi u izlistu 1.

```
using UnityEngine;
using Valve.VR;
using Valve.VR.InteractionSystem;

public class PlayerController : MonoBehaviour
{
    public SteamVR_Action_Vector2 input;
    public float speed;
    public float speedSeter;

    void Start()
    {
        //Postavljanje inicijalne brzine kretanja
        speedSeter = 1;
    }

    void Update()
    {
        //Postavljanje granica brzine kretanja
        speedSeter = Mathf.Clamp(speedSeter, 1, 15);
        speed = speedSeter;
        if (input.axis.magnitude > 0.1f)
        {
            //Pokretanje lika pomicanjem lijeve kontrolne palice
            Vector3 direction = Player.instance.hmdTransform.TransformDirection
(new Vector3(input.axis.x, 0, input.axis.y));
            transform.position += speed * Time.deltaTime * direction;
        }
    }
}
```

Izlist 1. *PlayerController*

Dodavanjem ove skripte *Player* klasi postiže se željeni učinak slobodnog kretanja kroz virtualnu stvarnost i prvi korak uranjanja korisnika u virtualni svijet.

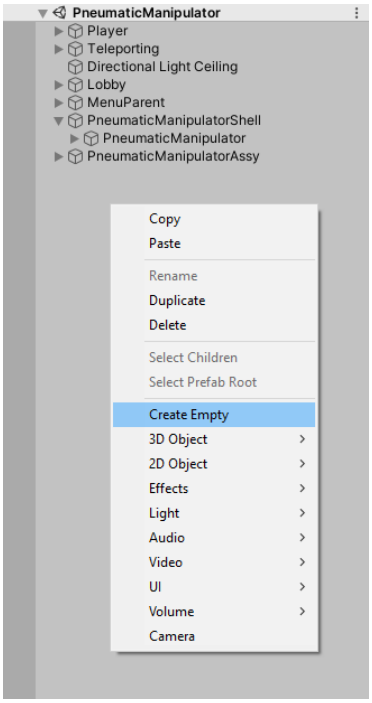
6. MODUL ZA MANIPULACIJU I PREGLED SKLOPA MANIPULATORA

Funkcija ovog modula je prikaz manipulatora u dva pogleda. Pogled u obliku sklopa koji se može pregledati kao cjelina (*Shell view*) i pogled u kojem se svaki dio sklopa može zasebno rastaviti i povećati (*Assembly view*). Ova dva prikaza omogućuju konstruktoru da vidi kako će manipulator izgledati u stvarnosti, da si predoči njegovu stvarnu veličinu pa čak i da detaljnim pregledom uoči nedostatke, ako postoje. U oba pogleda moguće je povećati i smanjiti manipulator i mijenjati svjetlinu prikaza kako bi bilo moguće prilagoditi prikaz vlastitim zahtjevima, a u *Assembly view* modu moguće je zasebno povećati i smanjiti svaki odabrani dio.

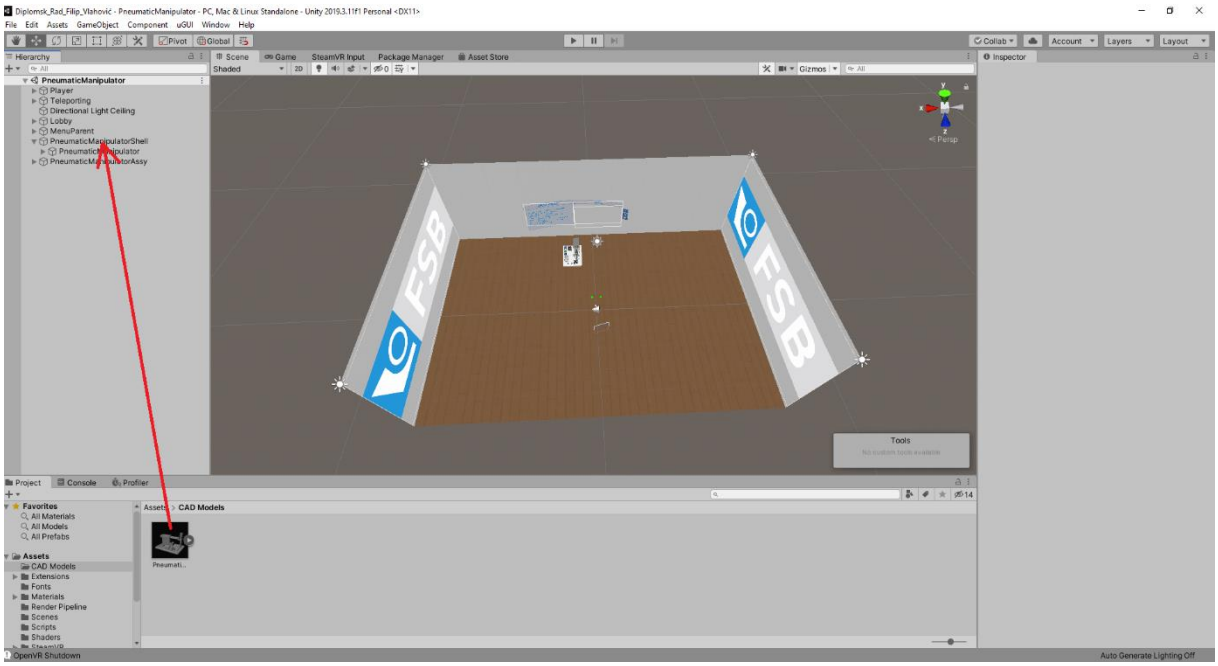
6.1. *Shell view* modul

Shell view modul služi za prikaz manipulatora kao jednog objekta koji se zatim može pregledavati, okretati te povećavati i smanjivati po želji. Ovaj prikaz služi za pregled gotovog rješenja u cjelosti.

Izrada se sastoji od nekoliko dijelova. Prvo je *drag & drop* tehnikom potrebno odvući 3D model manipulatora u hijerarhiju scene i postaviti ga kao *Child* prethodno izrađenog, praznog objekta. Prazni objekt se izrađuje tako da se na prazan dio hijerarhije klikne desni klik miša i iz izbornika koji se tada pojavi odabere se *Create Empty*.



Slika 45. Izrada praznog objekta



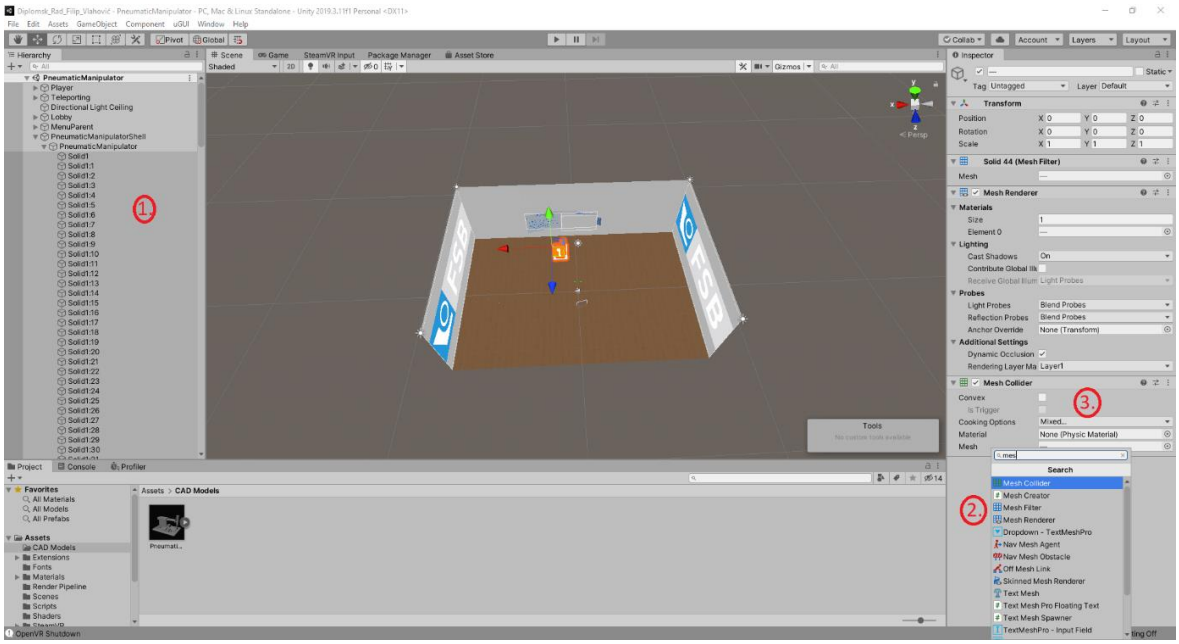
Slika 46. Postavljanje 3D modela manipulatora kao *Child* praznog objekta

Dovršavanjem ovih radnji u sceni će se pojaviti manipulator s kojim se za sada ništa ne može napraviti. Kako bi se omogućila interakcija s manipulatorom potrebno mu je dodijeliti *Collider*-e koji se koriste za detekciju kolizije, kao i u druge svrhe kao što su dodjela ili

oduzimanje ponašanja u skladu s fizikalnim pojavama i interakciju s drugim objektima u sceni.

Korišteni su *Mesh Collider*-i jer su najprecizniji i u potpunosti prate geometriju pojedinog 3D modela. *Mesh Collider*-i koji su označeni kao *Convex* mogu se sudarati s drugim *Mesh Collider*-ima. [15]

Collideri će se dodijeliti na način da se najprije odaberu svi dijelovi manipulatora i zatim im se kroz *Inspector* preko *Add Component* značajke dodaju *Mesh Collider*-i, a zatim se na isti način doda *Mesh Collider Parent*-u u koji je smješten 3D model pneumatskog manipulatora.

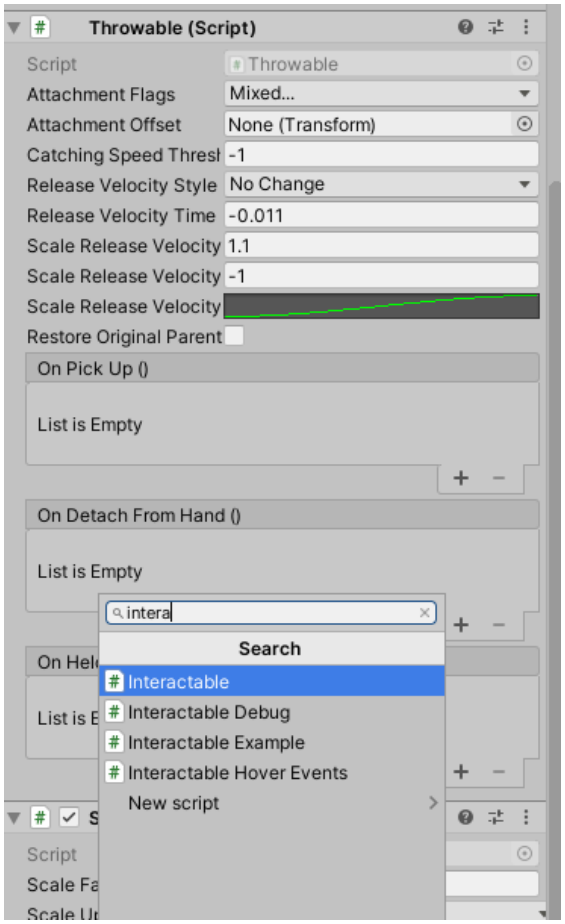


Slika 47. Dodjela *Mesh Collidera*

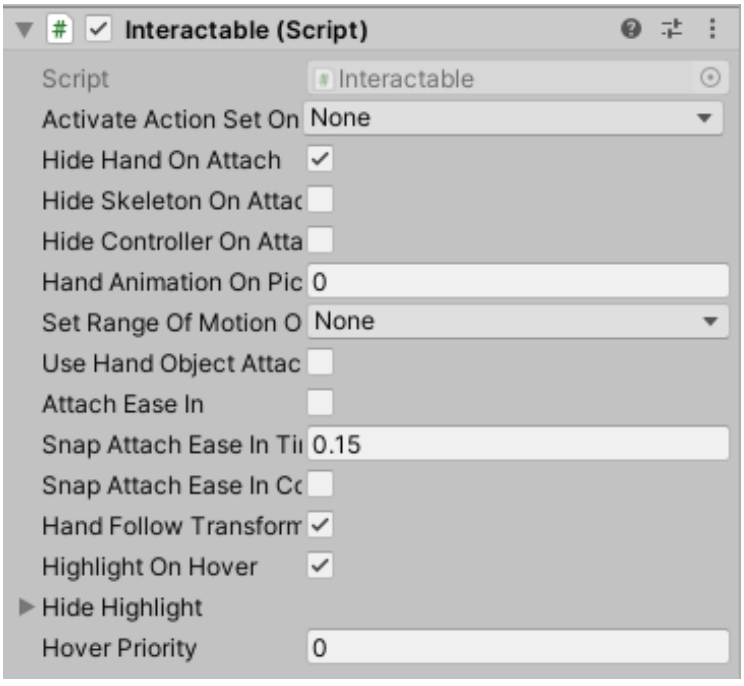
Time je omogućena interakcija između ruku i manipulatora. Ostaje još napraviti koordiniranu interakciju i omogućiti manipulaciju modelom pneumatskog manipulatora.

Za to će *Parent*-u u koji je smješten 3D model biti potrebno dodati skripte *Interactable*, *Throwable* i *SetScale*, a dodavanje skripte *Throwable* izazvat će automatsko dodavanje klase *Rigidbody*.

Skripta *Interactable* omogućuje primanje objekata u ruku, njihovo okretanje te povećavanje i smanjivanje ukoliko ih se uhvati s obje ruke te se ruke raširi ili skupi. *Interactable* se dodaje na isti način kao i *Mesh Collider*, preko *Inspectora* kroz *Add Component* sučelje. U dodijeljenoj skripti potrebno je podesiti određene parametre. Treba isključiti *Use Hand Object Attach* i *Hide Controller On Attach* kako bi dobili željeni prikaz ruke pri hvatanju pojedinih objekata.

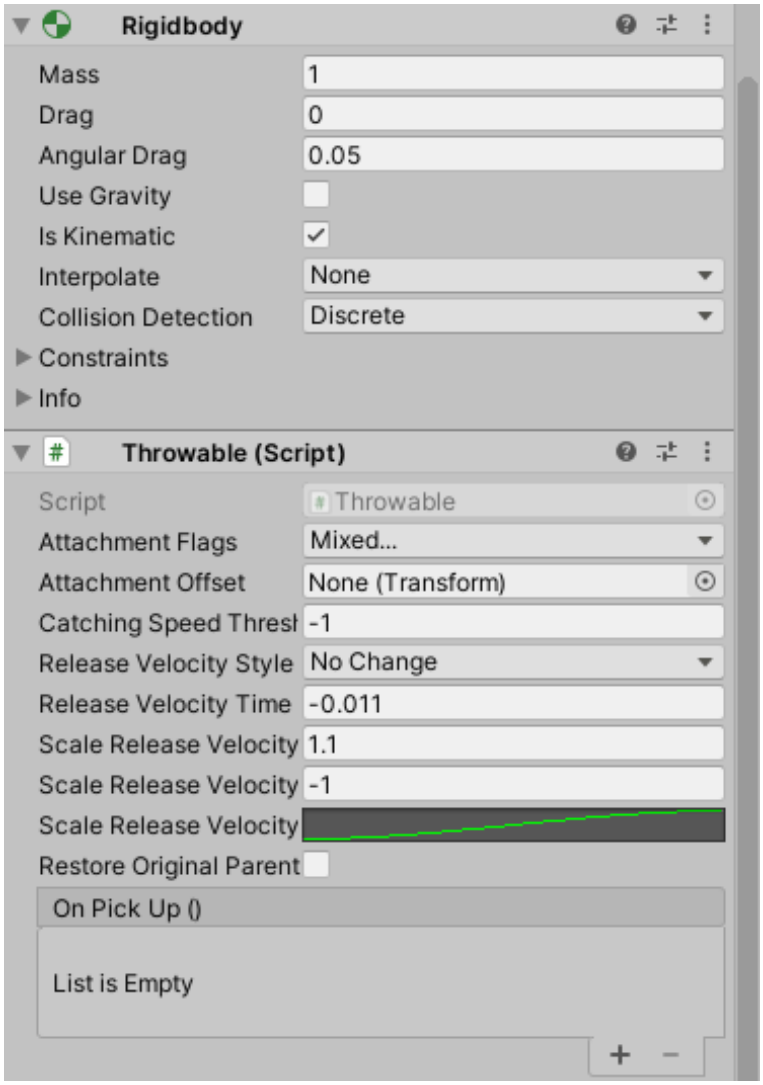


Slika 48. Dodjela *Interactable* skripte



Slika 49. Podešavanje parametara *Interactable skripte*

Da bi uhvaćeni dijelovi nakon puštanja ostali tamo gdje ih se ispustilo iz ruku potrebno je *Parent*-u dodati skriptu *Throwable* koja određuje kako će se objekti ponašati nakon ispuštanja iz ruku. Uz skriptu *Throwable* automatski se dodaje i klasa *Rigidbody* koja se brine za kontrolu pozicije objekta kroz simulacije fizikalnih pojava. Dodavanje *Rigidbody* klase nekom objektu i određivanje njegovog kretanja prepušta se Unity-u. *Throwable,* a i *Rigidbody* dodaju se na isti način kao i *Interactable*, a također je potrebno podesiti par opcija. U klasi *Rigidbody* potrebno je ugasiti opciju *Use Graviti* i uključiti opciju *Is Kinematic* kako se objekt ne bi kretao sukladno fizikalnim pojavama kao što je gravitacija, jer po ispuštanju iz ruke objekt mora ostati na mjestu gdje je ispušten. U skripti *Throwable* pod *Release Velocity Style* potrebno je odabrati opciju *No Change* kako objekti po ispuštanju ne bi dobili neku brzinu ili ubrzanje i poletjeli.



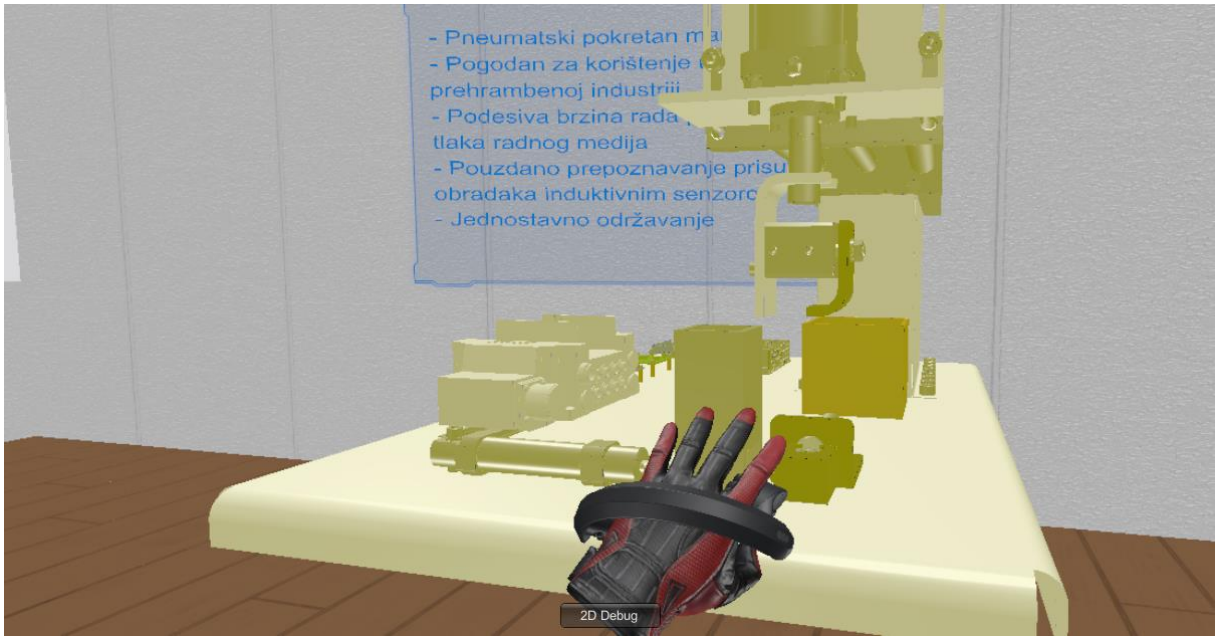
Slika 50. Podešavanje opcija *Throwable* i *Interactable* komponenti

Sada je omogućeno dohvaćanje, okretanje, detaljan pregled te povećavanje i smanjivanje sklopa kao cjeline.

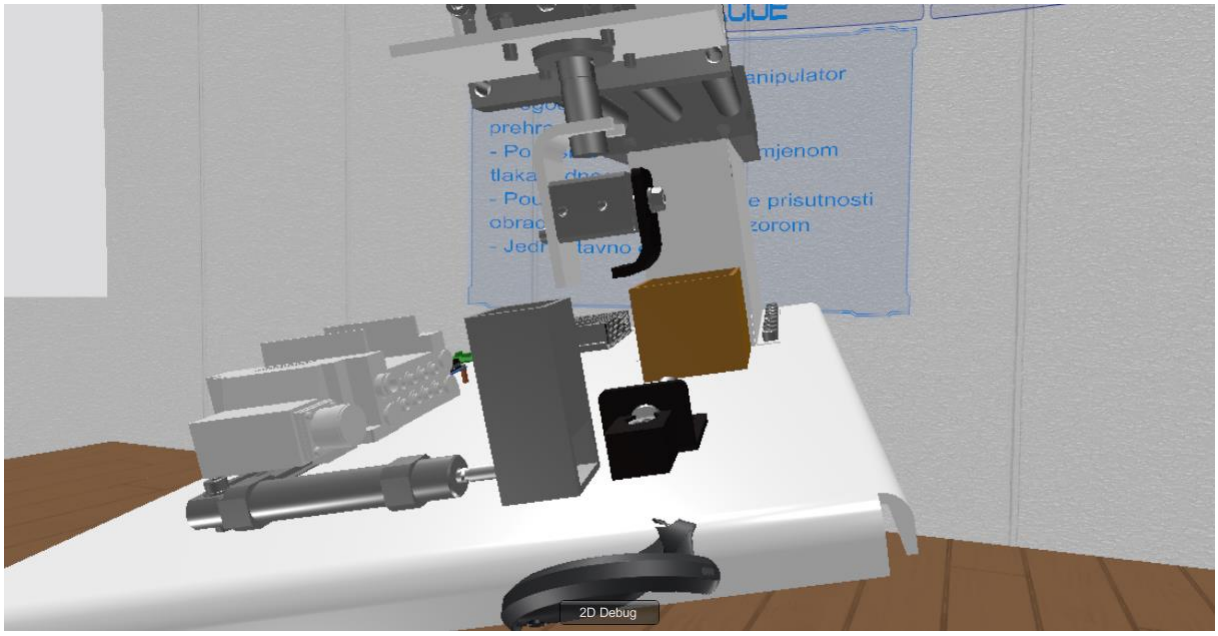
Prethodno implementirana opcija za povećavanje i smanjivanje objekta radi samo kad se on uhvati s dvije ruke i „razvuče“ ili „skupi“. Da bi se tome dodala fleksibilnost i neka količina preciznosti izrađena je skripta *SetScale* i dodana *Parent*-u u koji je smješten 3D model manipulatora. Svrha ove skripte je da proizvoljno pritiskom gumba na kontroleru u nekom omjeru povećava i smanjuje sklop manipulatora. Skripta slijedi u izlistu2.

```
using UnityEngine;
using Valve.VR;

public class SetScale : MonoBehaviour
{   //Postavljane faktora povećanja i smanjenja
    public float scaleFactor = 1.2f;
    public SteamVR_Action_Boolean scaleUp = null;
    public SteamVR_Action_Boolean scaleDown = null;

    void Start()
    {

    }

    void Update()
    {
        //Smanjenje objekta pritiskom na željeni gumd
        if (scaleUp.stateDown)
        {
            transform.localScale = transform.localScale * scaleFactor;
        }

        //Povećanje objekta pritiskom na željeni gumb
        if (scaleDown.stateDown)
        {
            transform.localScale = transform.localScale / scaleFactor;
        }
    }
}
```

Izlist 2. *SetScale*



Slika 51. Osjenčani sklop koji se može primiti zatvaranjem ruke

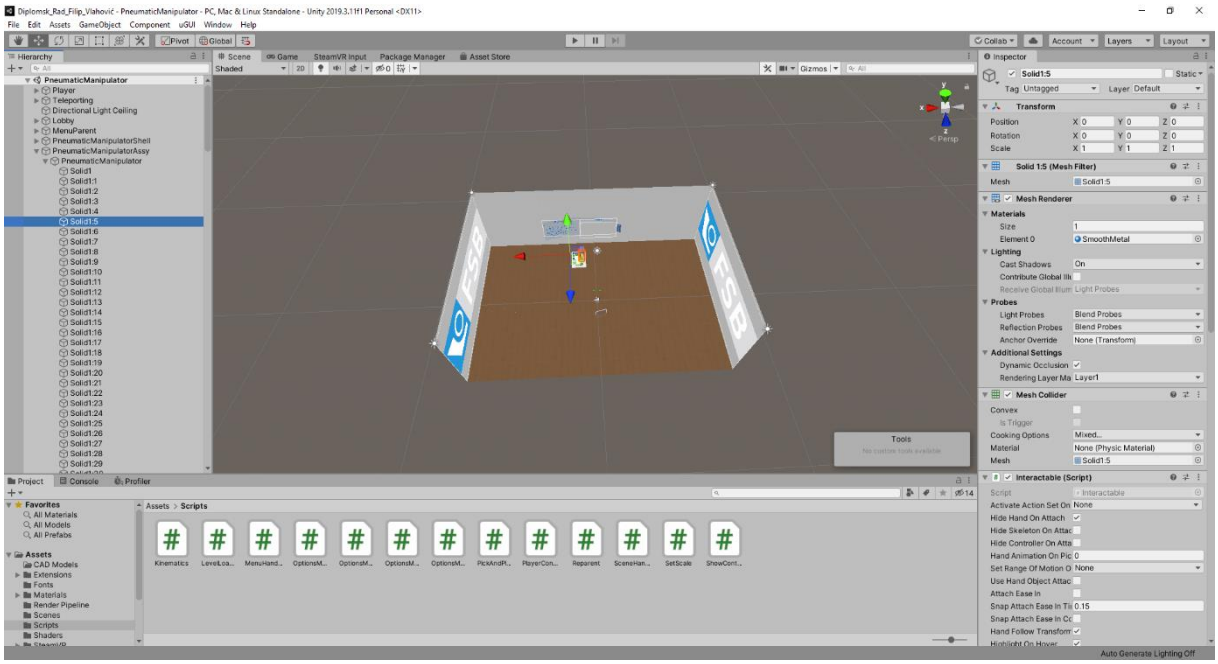


Slika 52. Uhvaćen sklop

6.2. Assemby view modul

Assembly view služi za rastavljanje sklopa manipulatora na sastavne dijelove i pregled svakog dijela pojedinačno. Koristi sve iste skripte i komponente kao *Shell view*, no oni su drugačije raspoređeni.

Umjesto dodavanja skripti *Interactable, SetScale* i *Throwable* te klase *Rigidbody Parent*-u u koji je smješten 3D model manipulatora, oni se dodaju svakom dijelu manipulatora zasebno kako bi se svakom dijelu omogućio zaseban odabir, pregled i manipulacija. Sve postavke su jednake kao i u 6.1. *Shell view*, no komponente su dodijeljene na drugačiji način koji je prikazan na slici 51.

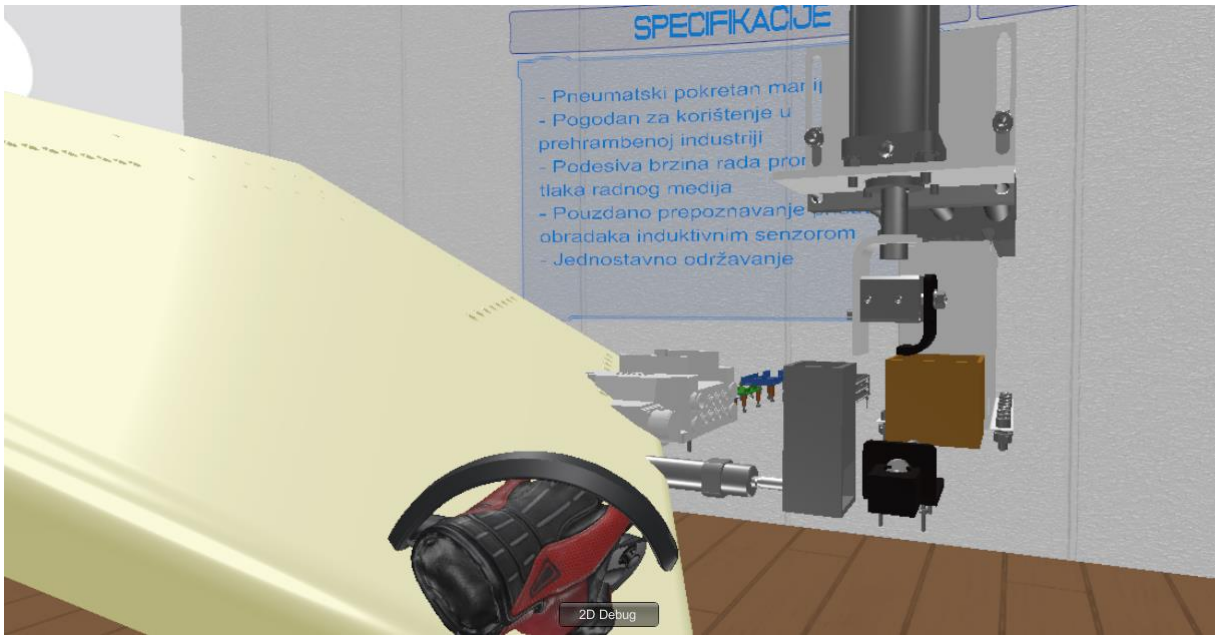


Slika 53. Dodavanje komponenti svakom dijelu zasebno

Prilikom rastavljanja dijelova naišlo se na dosta velik problem. Svaki objekt koji bi se uhvatio u ruku i pregledao je nakon ispuštanja „ispao“ iz svog *Parent*-a što je uzrokovalo problem svim skriptama i klasama koje operacije nad tim objektima rade na način da ih gledaju kao cjelinu dok su unutar svog *Parent*-a. Da bi se to popravilo izrađena je skripta *Reparent* koja svaki ispušteni objekt vraća nazad u sklop *Parent*-a. Skripta je vrlo jednostavna, ali i efikasna.

```
using UnityEngine;

public class Reparent : MonoBehaviour
{
    public GameObject Parent;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
    // Vraćanje 3D modela u Parent sklopa
        this.transform.parent = Parent.transform;
    }
}
```

Izlist 3. ***Reparent***



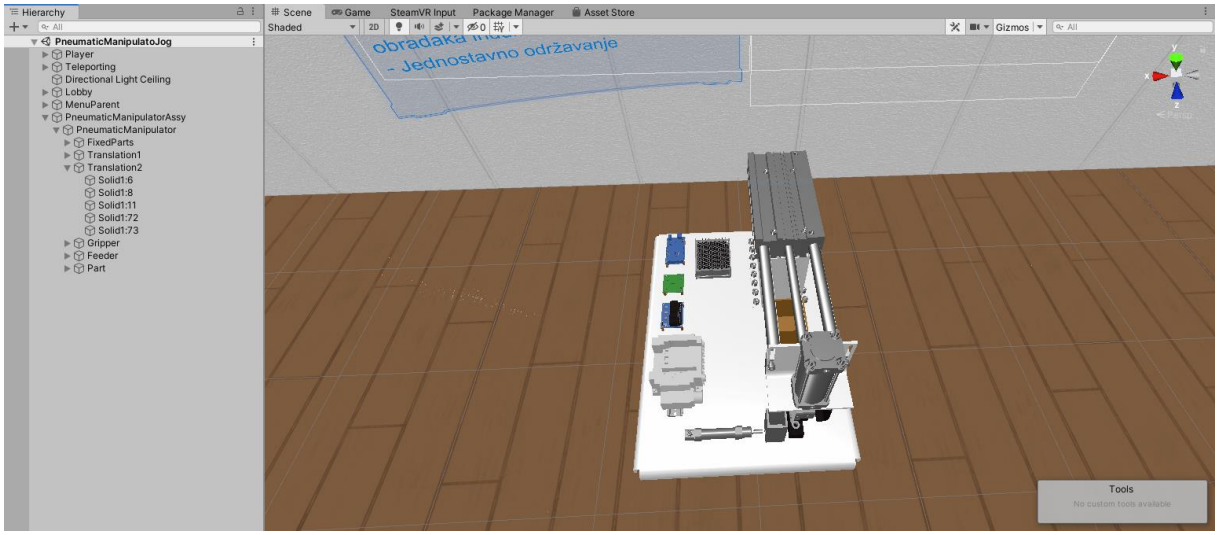
Slika 54. Osjenčani dio koji se može primiti zatvaranjem ruke

Slika 55. Uhvaćen dio

7. MODUL ZA UPRAVLJANJE MANIPULATOROM

Kako simulacija ne bi bila statična i kako bi joj se dodijelio jedan dodatni aspekt koji korisnika još više uranja u virtualni svijet i približava simulaciju stvarnosti izrađen je ovaj modul. Njegova svrha je omogućiti pokretanje pojedinih osi manipulatora kako bi se stekao dojam kako će manipulator raditi kad bude gotov.

Kako bi bilo moguće pokretati određene dijelove ili grupe dijelova najprije ih je potrebno grupirati na način da oni tvore smislene cjeline koje će se zatim pokretati. Cjeline se grupiraju tako da se najprije izrade prazni objekti i dodaju im se željena imena što je pokazano u poglavlju 6.1. *Shell view* modul. U izrađene objekte zatim se funkcijom *drag & drop* dodaju željeni dijelovi i tvore smislene cjeline.



Slika 56. Grupiranje dijelova u cjeline

Cijelim radom manipulatora upravlja skripta *Kinematics* u kojoj su definirana gibanja pojedinog cilindra, kao i granice u kojima se oni mogu gibati. Pritiskom određenog gumba na kontrolerima dolazi do pozivanja pojedine metode unutar skripte te se izvršavanjem koda iz metode manipulator pomiče konstantnom preddefiniranom brzinom. *Kinematics* skripta dodaje se u klasu *Player* kroz *Inspector* preko *Add Component* sučelja, a slijedi u izlistu 4.

```
using System;
using Valve.VR;
using UnityEngine;

public class Kinematics : MonoBehaviour
{
    //Inicijalizacija objekata za kontrolu ulaza pomoću kontrolera
    public SteamVR_Action_Vector2 horizontalTranslation;
    public SteamVR_Action_Vector2 verticalTranslation;
    public SteamVR_Action_Boolean openGripper;
    public SteamVR_Action_Boolean closeGripper;
    public SteamVR_Action_Boolean expandFeeder;
    public SteamVR_Action_Boolean retractFeeder;

    private float translationSpeed = 100f;
    private GameObject parent;// = GameObject.Find("PneumaticManipulator");
    private GameObject translation1;// = GameObject.Find("Translation1");
    private GameObject translation2;// = GameObject.Find("Translation2");
    private GameObject gripper;// = GameObject.Find("Gripper");
    private GameObject feeder;// = GameObject.Find("Feeder");
    private float translationFactor;

    //Pokretanje translacije jedan u desno
    public void Translation1Right()
    {

        translation2.transform.parent = translation1.transform;
        gripper.transform.parent = translation1.transform;

        Vector3 translation1CurrentPosition = translation1.transform.localPosition
;
        translation1CurrentPosition.z += translationFactor * translationSpeed * Ti
me.deltaTime;
        Vector3 translation1NewPosition = new Vector3(translation1CurrentPosition.
x, translation1CurrentPosition.y, Mathf.Clamp(translation1CurrentPosition.z, 230f,
 476.4721f));
        translation1.transform.localPosition = translation1NewPosition;

        translation2.transform.parent = parent.transform;
        gripper.transform.parent = parent.transform;
    }
    //Pokretanje translacije jedan u lijevo
    public void Translation1Left()
    {
        translation2.transform.parent = translation1.transform;
        gripper.transform.parent = translation1.transform;

        Vector3 translation1CurrentPosition = translation1.transform.localPosition
;
        translation1CurrentPosition.z += (-
translationFactor) * translationSpeed * Time.deltaTime;
        Vector3 translation1NewPosition = new Vector3(translation1CurrentPosition.
x, translation1CurrentPosition.y, Mathf.Clamp(translation1CurrentPosition.z, 230f,
 476.4721f));
        translation1.transform.localPosition = translation1NewPosition;

        translation2.transform.parent = parent.transform;
        gripper.transform.parent = parent.transform;
```

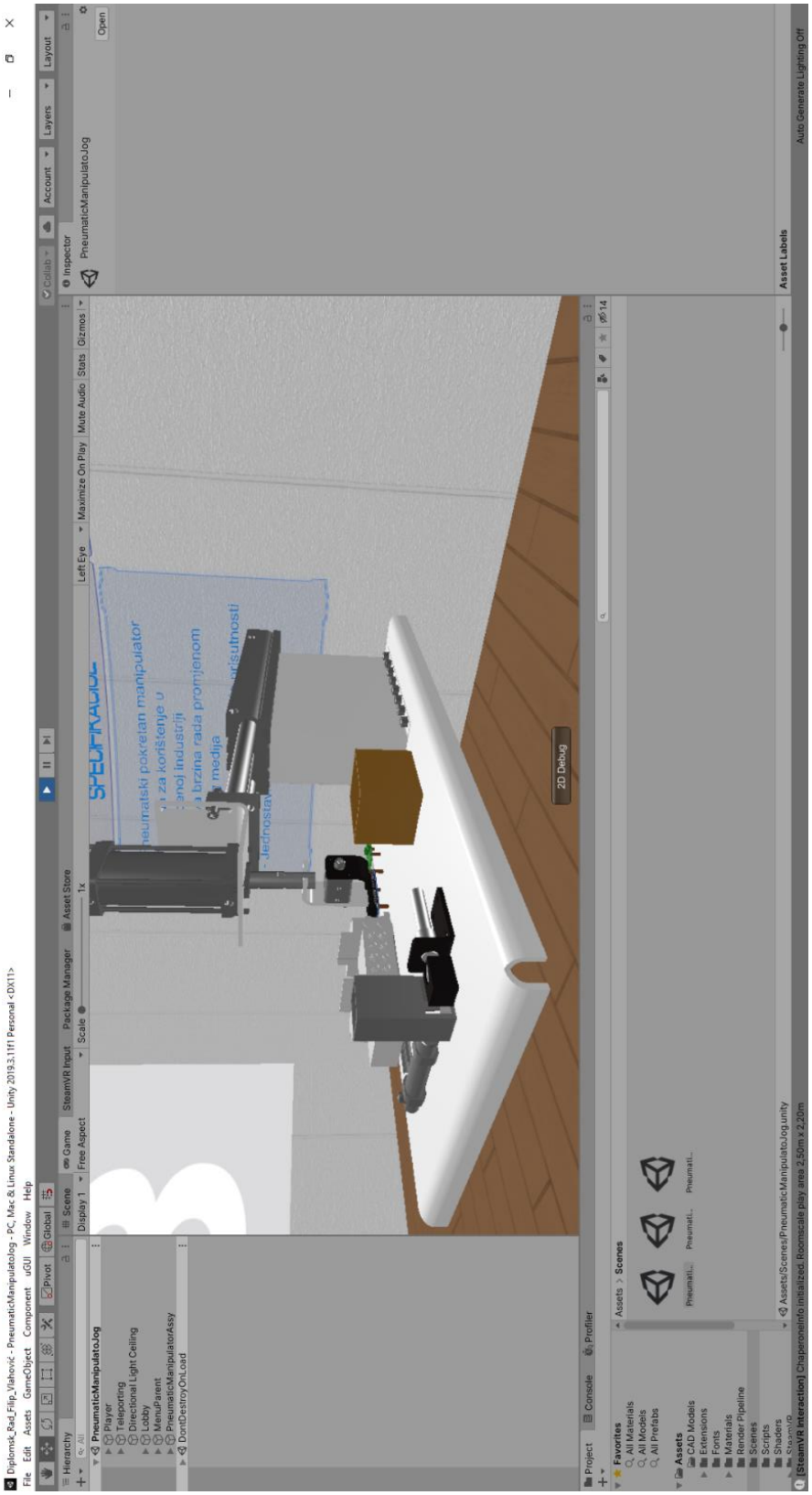
```
    }
    //Pokretanje translacije dva prema gore
    public void Translation2Up()
    {
        gripper.transform.parent = translation2.transform;

        Vector3 translation2CurrentPosition = translation2.transform.localPosition
;
        translation2CurrentPosition.y += translationFactor * translationSpeed * Ti
me.deltaTime;
        Vector3 translation2NewPosition = new Vector3(translation2CurrentPosition.
x, Mathf.Clamp(translation2CurrentPosition.y, -175f, -
75f), translation2CurrentPosition.z);
        translation2.transform.localPosition = translation2NewPosition;

        gripper.transform.parent = parent.transform;
    }
    //Pokretanje translacije dva prema dolje
    public void Translation2Down()
    {
        gripper.transform.parent = translation2.transform;

        Vector3 translation2CurrentPosition = translation2.transform.localPosition
;
        translation2CurrentPosition.y += (-
translationFactor) * translationSpeed * Time.deltaTime;
        Vector3 translation2NewPosition = new Vector3(translation2CurrentPosition.
x, Mathf.Clamp(translation2CurrentPosition.y, -175f, -
75f), translation2CurrentPosition.z);
        translation2.transform.localPosition = translation2NewPosition;

        gripper.transform.parent = parent.transform;
    }
    //Zatvaranje prihvatnice
    public void GripperIn()
    {
        float gripperPosX = gripper.transform.localPosition.x;

        while (gripperPosX < -3388f)
        {
            Vector3 gripperCurrentPosition = gripper.transform.localPosition;
            gripperCurrentPosition.x += translationFactor * translationSpeed * Tim
e.deltaTime;
            Vector3 gripperNewPosition = new Vector3(gripperCurrentPosition.x, gri
pperCurrentPosition.y, gripperCurrentPosition.z);
            gripper.transform.localPosition = gripperNewPosition;
            gripperPosX = gripperNewPosition.x;
        }
    }
    //Otvaranje prihvatnice
    public void GripperOut()
    {
        float gripperPosX = gripper.transform.localPosition.x;

        while (gripperPosX > -3411f)
        {
            Vector3 gripperCurrentPosition = gripper.transform.localPosition;
```

```
            gripperCurrentPosition.x += (-
translationFactor) * translationSpeed * Time.deltaTime;
            Vector3 gripperNewPosition = new Vector3(gripperCurrentPosition.x, gri
pperCurrentPosition.y, gripperCurrentPosition.z);
            gripper.transform.localPosition = gripperNewPosition;
            gripperPosX = gripperNewPosition.x;
        }
    }
    //Izbacivanje cilindra za dodavanje obradaka
    public void FeederIn()
    {
        float feederPosX = feeder.transform.localPosition.x;

        while (feederPosX < -2513.85f)
        {
            Vector3 feederCurrentPosition = feeder.transform.localPosition;
            feederCurrentPosition.x += translationFactor * translationSpeed * Time
.deltaTime;
            Vector3 feederNewPosition = new Vector3(feederCurrentPosition.x, feede
rCurrentPosition.y, feederCurrentPosition.z);
            feeder.transform.localPosition = feederNewPosition;
            feederPosX = feederNewPosition.x;
        }
    }
    //Uvlačenje cilindra za dodavanje obradaka
    public void FeederOut()
    {
        float feederPosX = feeder.transform.localPosition.x;

        while (feederPosX > -2549f)
        {
            Vector3 feederCurrentPosition = feeder.transform.localPosition;
            feederCurrentPosition.x += (-
translationFactor) * translationSpeed * Time.deltaTime;
            Vector3 feederNewPosition = new Vector3(feederCurrentPosition.x, feede
rCurrentPosition.y, feederCurrentPosition.z);
            feeder.transform.localPosition = feederNewPosition;
            feederPosX = feederNewPosition.x;
        }
    }

    void Start()
    {
        parent = GameObject.Find("PneumaticManipulator");
        translation1 = GameObject.Find("Translation1");
        translation2 = GameObject.Find("Translation2");
        gripper = GameObject.Find("Gripper");
        feeder = GameObject.Find("Feeder");
        translationFactor = 0.5f;
    }

    void Update()
    {
        //Pozivanje pojedinih metoda na pritisak određenog gumba ili Touch pada na
 kontroleru
        if (horizontalTranslation.axis.y > 0.1)
        {
            Translation1Left();
```

```
        }

        if (horizontalTranslation.axis.y < -0.1)
        {
            Translation1Right();
        }

        if (verticalTranslation.axis.y > 0.1)
        {
            Translation2Up();
        }

        if (verticalTranslation.axis.y < -0.1)
        {
            Translation2Down();
        }

        if (openGripper.state)
        {
            GripperIn();
        }

        if (closeGripper.state)
        {
            GripperOut();
        }

        if (expandFeeder.state)
        {
            FeederOut();
        }

        if (retractFeeder.state)
        {
            FeederIn();
        }
    }
}
```

Izlist 4. ***Kinematics***

Kinematics skripta je tip skripte koji se dodaje samo jednom u scenu jer bi njezino višestruko dodavanje moglo izazvati neželjene posljedice u vidu nekontroliranog gibanja manipulatora.



Slika 57. Pomicanje manipulatora

8. MODUL ZA PROMJENU PARAMETARA RADA

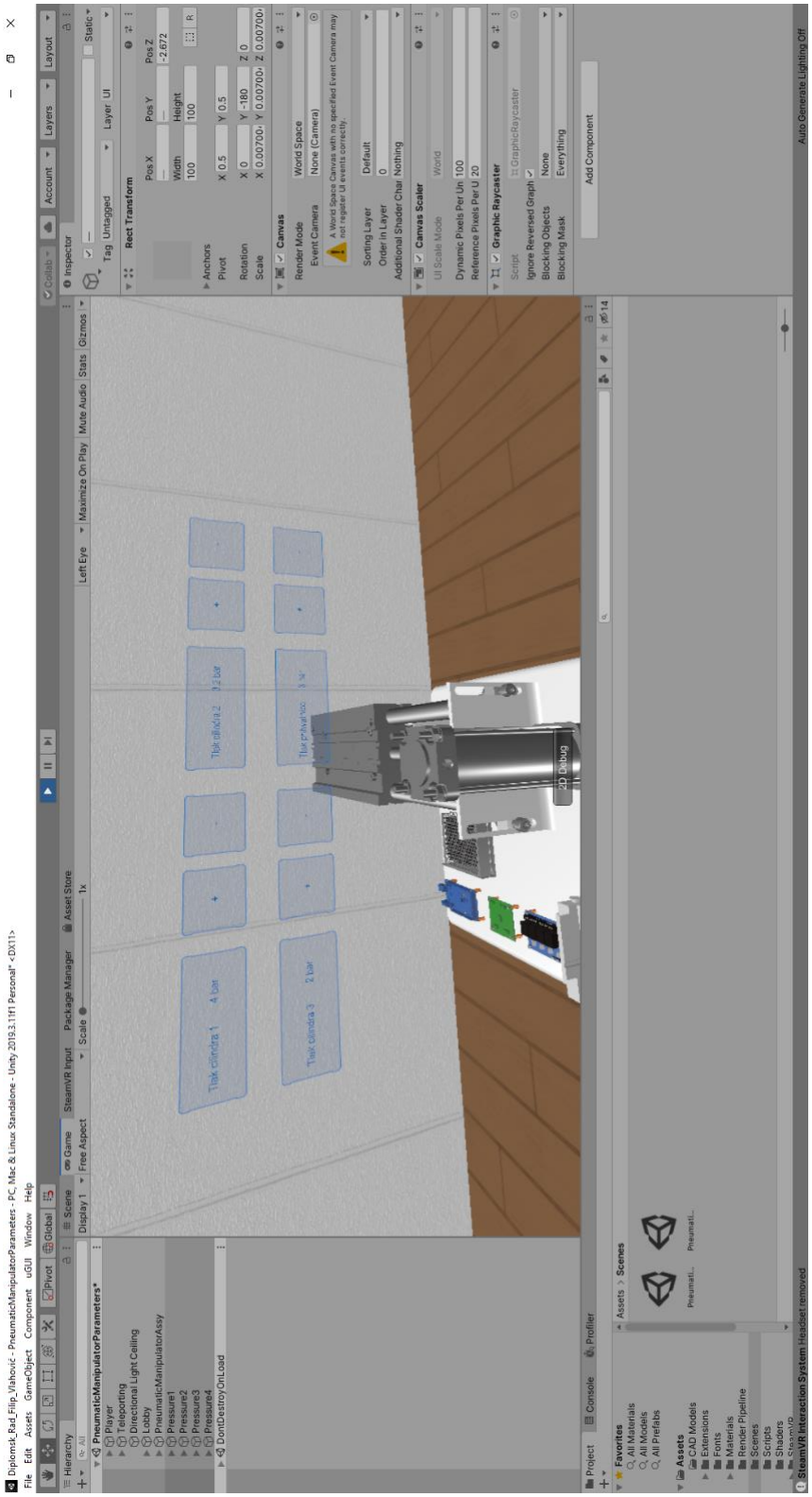
U svrhu optimizacije parametara rada manipulatora izrađen je ovaj modul. Niti jedan mehatronički sustav nije savršen čim se izradi. Potrebno je uložiti dosta truda kako bi se optimiralo parametre rada tako da sustav bude što efikasniji. Modul za promjenu parametara omogućuje inženjerima postavljanje nekih početnih parametara od kojih će se lakše krenuti u krajnje optimiranje rada gotovog sustava.

U ovom modulu moguće je mijenjati tlak rada pojedinog cilindra što određuje brzinu uvlačenja i izvlačenja cilindara. S obzirom na to da ovdje simulirani pneumatski manipulator nema senzore položaja cilindra potrebno je ručno definirati uključivanje i isključivanje releja tijekom ciklusa rada manipulatora. Promjenom tlaka zraka dolazi do promjene brzine uvlačenja i izvlačenja cilindara na temelju čega se sustav može optimirati na način da obavi što više ciklusa u što kraćem vremenu.

Za pokretanje cilindara koristi se skripta *Kinematics* dana u izlistu 4., dok je jedino što se mijenja brzina kretanja cilindra.

Povećanje i smanjenje tlaka moguće je za 0.1 bar po pritisku na određeni gumb. Kada su dobiveni zadovoljavajući tlakovi oni se uz podešavanje uključivanja i isključivanja releja podešavaju na sustavu.

S obzirom na to da tlak i izvlačenje prvog cilindra ne ovise o ostalim cilindrima njegovo gornje ograničenje tlaka je maksimalno dopušteno od strane proizvođača, dok s druge strane tlak cilindra koji dodaje obratke ovisi o masi obratka, jer bi previsok tlak mogao prejako izbaciti obradak i ozlijediti nekoga, a prenizak tlak bi mogao dovesti do toga da obradak ne dođe direktno ispod prihvatnice i uzrokovati koliziju s prihvatnicom. Tlak prihvatnice također ovisi o masi obratka koju je potrebno podignuti.



Slika 58. Promjena tlaka cilindara

9. ZAKLJUČAK

Virtualna stvarnost relativno je nova tehnologija koja će svoj vrhunac tek dostići. Prije samo nekoliko desetaka godina stvari koje danas možemo raditi pomoću virtualne stvarnosti bile su samo daleka budućnost i teško je bilo zamisliti da će biti dostupne širokim masama. Koristile su se zajedno s imenima kao što je NASA.

Izrađena aplikacija vjerno prikazuje rad manipulatora, kao i sve njegove sastavne dijelove. Imerzivnost tehnologije virtualne stvarnosti omogućuje pregled manipulatora iz jedne sasvim nove perspektive. Moguće je vidjeti gotov proizvod u gotovo realističnom stanju prije samog sklapanja pa čak i prije izrade prototipa.

Razvojem tehnologije i stavljanjem novih zahtjeva za bržu i jeftiniju izradu prototipova pred inženjere, sustavi za virtualnu stvarnost sve više se počinju koristiti u razvoju, konstrukciji, montaži i testiranju raznih mehatroničkih sustava. Primjene virtualne stvarnosti su mnogobrojne, a neke još nisu ni otkrivene. Korištenjem ovih visoko tehnoloških sustava skraćuje se vrijeme razvoja, smanjuje se broj grešaka u samoj konstrukciji, a izradom detaljnih simulacija na vrijeme se mogu otkriti i ukloniti neke greške pri radu sustava prije nego su oni uopće pušteni u pogon. Primjena VR sustava tu ne prestaje. Mogu se koristiti i za trening i edukaciju novih zaposlenika kako bi što brže savladali prepreke koje ih čekaju na novom radnom mjestu i kako bi im se što realnije prikazalo što ih čeka. Sve ovo smanjuje troškove razvoja i izrade nekog proizvoda, a isto tako i doprinosi većoj kvaliteti.

S obzirom na to da je tehnologija virtualne stvarnosti još uvelike u fazi razvoja i ne postoje programski paketi razvijeni isključivo za izradu simulacija mehaničkih i mehatroničkih sustava potrebno je uložiti puno vremena i truda kako bi se korištenjem alternativnih programskih paketa namijenjenih za razvoj i izradu igara napravile simulacije mehatroničkih sustava. Sve fizikalne pojave, kao i pomicanje osi manipulatora potrebno je ručno programirati što omogućuje precizno definiranje željenih zahtjeva, ali isto tako uvelike povećava kompleksnost cijelog zadatka.

LITERATURA

[1] URL izvor: https://www.nas.nasa.gov/Software/VWT/vr.html, 12.11.2020.

[2] URL izvor: https://ec.europa.eu/croatia/virtual_reality_reality_we_live_in_hr, 08.09.2020.

[3] URL izvor: https://geek.hr/pojmovnik/sto-virtualna-stvarnost/, 08.09.2020.

[4] URL izvor: https://en.wikipedia.org/wiki/Virtual_reality#History, 08.09.2020.

[5] URL izvor: https://ec.europa.eu/croatia/content/what-is-AR-what-VR-and-how-technology-helps-us-to-experience-reality_hr, 09.09.2020.

[6] URL izvor: https://en.wikipedia.org/wiki/Valve_Index, 11.9.2020.

[7] URL izvor: https://www.valvesoftware.com/en/index/base-stations, 11.9.2020.

[8] URL izvor: https://en.wikipedia.org/wiki/Unity_(game_engine), 10.09.2020.

[9] URL izvor: https://valvesoftware.github.io/steamvr_unity_plugin/, 12.11.2020.

[10] URL izvor: https://docs.unity3d.com/Manual/SinglePassStereoRendering.html, 12.11.2020.

[11] URL izvor: https://en.wikipedia.org/wiki/C_Sharp_(programming_language), 11.9.2020

[12] URL izvor: https://code.visualstudio.com/docs/editor/intellisense, 14.11.2020.

[13] URL izvor: https://hr.wikipedia.org/wiki/SolidWorks, 15.11.2020

[14] URL izvor: https://valvesoftware.github.io/steamvr_unity_plugin/articles/Interaction-System.html, 16.11.2020.

[15] URL izvor: https://docs.unity3d.com/Manual/class-MeshCollider.html, 18.11.2020.

PRILOZI

I. CD-R disc