Prijenos 3D CAD modela u virtualnu stvarnost

Galić, Noa

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Noa Galić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

izv. prof. dr. sc. Stanko Škec, mag. ing. mech. Noa Galić

Zagreb, 2024.

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

Zahvaljujem se svom mentoru izv. prof. dr. sc. Stanku Škecu na svim savjetima i pomoći prilikom izrade ovoga rada.

Zahvaljujem se također cijeloj svojoj obitelji, bez čije dugogodišnje potpore i brige ovaj rad i završetak ovog studija ne bi bio moguć.

Zahvaljujem se i svim svojim prijateljima, posebno dvojici bez kojih bi ovaj rad vjerojatno bio završen i prije.

Na kraju, posebno se zahvaljujem svojoj dugogodišnjoj curi Martini koja je uvijek bila uz mene kroz sve teške i lijepe trenutke tijekom studija.

Noa Galić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 24 – 06 / 1**	
Ur.broj: **15 – 24 –**	

ZAVRŠNI ZADATAK

Student: **Noa Galić** JMBAG: **0035223867**

Naslov rada na hrvatskom jeziku: **Prijenos 3D CAD modela u virtualnu stvarnost**

Naslov rada na engleskom jeziku: **Import of 3D CAD models in virtual reality**

Opis zadatka:

Pri koncipiranju i oblikovanju konstrukcija upotrebljavaju se različite vrste i formati 3D CAD modela. U kontekstu povećanja upotrebe tehnologija virtualne stvarnosti za pregled konstrukcija, nameće se potreba za čestim prijenosom CAD modela u navedeno okruženje. S ciljem uspješnog prijenosa inženjerskih informacija, 3D CAD modeli zahtijevaju prilagodbe te odabir prikladnih formata. Naime, vizualizacija proizvoda u aplikacijama za virtualnu stvarnost zahtijeva pripremu inženjerskih podataka vezanih uz geometriju, materijale, teksture, relacije itd. Stoga, u okviru ovog završnog zadatka, potrebno je analizirati konverziju, prijenos i interpretaciju 3D CAD modela u različitim okruženjima virtualne stvarnosti.

U radu je potrebno:

- Proučiti i usporediti načine konverzije i uvođenja 3D CAD modela u različita okruženja virtualne stvarnosti temeljem pregleda literature.
- Definirati kriterije za analizu navedenih postupaka prijenosa 3D CAD modela.
- Analizirati načine i ishode konverzije, prijenosa i interpretacije 3D CAD modela upotrebom prethodno kreiranih modela dijelova i sklopova.
- Pripremiti aplikaciju za prijenos 3D CAD modela u okruženje virtualne stvarnosti.
- Predložiti preporuke za pripremu 3D CAD modela za konverziju i prijenos u okruženje virtualne stvarnosti.

Opseg analize i interpretacije rezultata dogovorit će se tijekom izrade rada.

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

Zadatak zadan:
30. 11. 2023.

Datum predaje rada:
1. rok: 22. i 23. 2. 2024.
2. rok (izvanredni): 11. 7. 2024.
3. rok: 19. i 20. 9. 2024.

Predviđeni datumi obrane:
1. rok: 26. 2. – 1. 3. 2024.
2. rok (izvanredni): 15. 7. 2024.
3. rok: 23. 9. – 27. 9. 2024.

Zadatak zadao:
Izv. prof. dr. sc. Stanko Škec

Predsjednik Povjerenstva:
Prof. dr. sc. Damir Godec

SADRŽAJ

Popis slika

Popis tablica

1. UVOD

Tehnologija CAD-a (*Computer Aided Design*) igra ključnu ulogu u modernom strojarstvu, omogućujući precizno oblikovanje, simulaciju i analizu proizvoda prije njihove stvarne izrade. Kako se tehnologija razvija, sve veći naglasak stavlja se na integraciju CAD modela u okruženja VR (*Virtual Reality*), koja pružaju nove mogućnosti za pregled i analizu konstrukcija u realističnom, interaktivnom okruženju. VR omogućuje inženjerima, dizajnerima i klijentima da dožive proizvode na načine koji prije nisu bili mogući tako da rukuju proizvodom u stvarnom vremenu prije nego je uopće izrađen.

Međutim, prijenos CAD modela u VR okruženje nije jednostavan zadatak. Podaci o obliku, materijali, i relacije između komponenti moraju biti precizno prilagođeni kako bi se osigurao uspješan prijenos i vizualizacija modela u VR-u. Neodgovarajući formati datoteka modela, složenost modela ili loše strukturirana hijerarhija modela mogu rezultirati lošom kvalitetom prikaza, smanjenom interaktivnošću s modelom i neuspješnim prijenosom modela u virtualno okruženje.

Cilj ovog rada je analizirati proces prijenosa 3D CAD modela u okruženju virtualne stvarnosti, razviti aplikaciju koja bi prenosila CAD modele u okruženje virtualne stvarnosti te postaviti temelj za izradu aplikacije za konstrukcijske vježbe u VR okruženju. Rad će istražiti načine prijenosa CAD modela u VR okruženje, kako se prenose različiti aspekti modela poput oblika, postaviti kriterije za uspješan prijenos, na temelju kriterija izraditi aplikaciju za prijenos modela i na kraju analizirati ishode prijenosa modela. Također, bit će razmotreni specifični koraci potrebni za prilagodbu CAD modela za VR okruženje. Na temelju provedenih analiza, rad će predložiti smjernice za pripremu CAD modela za učinkovit prijenos u virtualna okruženja.

2. PREGLED LITERATURE

Prije nego što se provede analiza prijenosa CAD modela, potrebno je napraviti pregled trenutnog stanja VR-a u CAD industriji. Treba navesti prednosti rada sa CAD modelima u VR okruženju i koji su trenutni izazovi korištenja CAD modela u VR okruženju. Zatim, da se jasno prikaže problem prijenosa CAD modela, potrebno je definirati koji su modeli uključeni u okviru strojarstva, koja je njihova namjena i zašto je prijenos tih modela važna tema u okviru CAD-a. Onda je potrebno objasniti što se podrazumijeva kod prijenosa CAD modela i definirati osnovne principe, vrste te kriterije prijenosa modela.

2.1. Stanje VR-a u CAD industriji

CAD modeli se koriste za više od samog 3D modeliranja, poput simulacija i planiranja rada. CAD sustavi omogućuju provjeru funkcionalnosti modela, ali nedostaje im interaktivnost koju pruža VR. Tradicionalne CAD tehnologije teško oponašaju stvarni 3D svijet, dok je u VR-u omogućena detaljna simulacija i konceptualizacija CAD modela. VR omogućava inženjerima strojarstva bolje razumijevanje gradiva, osobito kada je potrebna vizualna reprezentacija stvarnog procesa. U zdravstvenom sektoru, tehnologija virtualne stvarnosti koristi se za podršku proceduralnoj obuci, dijagnostici i pružanje virtualne terapije u hitnim situacijama. U zrakoplovnoj industriji, VR se često koristi za niz primjena, uključujući dizajn kabina i obuku pilota. [1]

Neki CAD sustavi današnjice imaju mogućnost integracije s VR sustavima. Primjeri ovoga su Demo3DVR za SolidWorks CAD sustav i NX Virtual Reality za Siemens NX CAD sustav. No, važno je naglasiti kako ovi sustavi nude veoma ograničeno iskustvo pri interakciji s modelom. Korisnik može pomicati model i detaljno ga proučavati, no izvođenje kompleksnih radnji na modelu nije moguće.

Glavni izazov prijenosa CAD modela u VR okruženje je gubitak spojeva i ograničenja modela, ali postoje istraživanja koja pokušavaju riješiti taj problem korištenjem specijaliziranih softverskih sustava. [2] Ako bi se mogao izvesti prijenos spojeva i ograničenja CAD modela u VR okruženje, bilo bi moguće raditi napredne simulacije rada modela u VR okruženju što bi uvelike poboljšalo kvalitetu VR CAD sustava.

2.2. Općenito o CAD modelima

CAD (Computer-Aided Design) modeli predstavljaju digitalne prikaze inženjerskih proizvoda, izrađene pomoću softverskih alata koji omogućuju precizno konstruiranje i analizu proizvoda. U strojarstvu, ovi modeli služe kao temelj za cijeli proces razvoja proizvoda, od koncepta i dizajna, preko simulacija i optimizacija, do same proizvodnje. CAD modeli mogu uključivati sve aspekte potrebne za izradu i montažu proizvoda, kao što su geometrija, topologija, materijali, i međusobni odnosi dijelova.

Najčešći oblik CAD modela su B-Rep (Boundary Representation) modeli. B-Rep modeli detaljno opisuju topologiju objekta kroz njegovu referentnu geometriju, koristeći vrhove, bridove i plohe kako bi precizno definirali oblik modela. CAD sustavi omogućavaju izradu B-Rep modela parametarskim modeliranjem, te se zato CAD modeli još nazivaju i parametarskim modelima.[3] Parametarski modeli omogućuju korisnicima definiranje modela pomoću parametara koji određuju oblik i veličinu geometrijskih elemenata. Ovakvi modeli omogućuju jednostavne modifikacije promjenom samo određenih parametara, što je vrlo korisno u iterativnim procesima konstruiranja.

Osim geometrijskih i topoloških podataka, CAD modeli često uključuju i metapodatke kao što su informacije o materijalima, svojstvima površina, tolerancijama, te spojevima između različitih dijelova modela.

S obzirom na sve veći zahtjev za interaktivnim i realističnim prikazom proizvoda prije njihove izrade, prijenos CAD modela u VR okruženja postaje ključan korak u modernim inženjerskim procesima. Ovaj proces, međutim, nosi svoje izazove, jer podrazumijeva ne samo prijenos geometrije, već i prilagodbu modela kako bi se očuvala njegova funkcionalnost i interaktivnost u virtualnom prostoru. Stoga je važno razumjeti osnovne koncepte i strukture koje stoje iza CAD modela kako bi se osigurala njihova uspješna primjena u VR tehnologiji i drugim naprednim sustavima.

2.3. Prijenos modela

Prije nego što se može govoriti o prijenosu specifičnih aspekata modela poput oblika, potrebno je prvo sagledati kako se modeli prenose u cijelosti. Nakon što je objašnjen prijenos modela u cijelosti, potrebno je proučiti kako se prenosi svaki aspekt modela, od oblika do spojeva i ograničenja.

2.3.1. Definicija prijenosa CAD modela

Proces prijenosa 3D CAD modela definiran je kao prijenos podataka CAD modela iz izvornog CAD sustava u drugi CAD sustav ili 3D program.

3D programima će se u sklopu ovoga rada nazivati svi programi koji ispunjavaju svoje funkcionalnosti radom s 3D grafikom, ali nisu namijenjeni za CAD modeliranje. Primjer ovakvih programa su platforme za izradu video igara (Unity3D, Unreal Engine) i programi za izradu modela i animacija (Blender). Ovi programi su podijeljeni na više kategorija, no u sklopu ovoga rada jedino ih je potrebno jasno razlikovati od CAD sustava.

Tablica 1. Podjela programa za izradu i manipulaciju 3D modela/grafike

Kategorije programa	**Način modeliranja**	**Primjeri**	**Glavne značajke**
CAD Sustavi	B-Rep/NURBS/Parametarsko modeliranje	SolidWorks, CATIA	Parametarsko modeliranje, prikaz tehničkih podataka o modelu
Programi za neparametarsko modeliranje	Modeliranje mnogokutima	Blender, Maya	Modeliranje „slobodnom rukom“, izrada animacija
Platforme za razvoj video igara	Modeliranje predefiniranim mnogokutnim primitivima, češće se koriste unaprijed definirani modeli	Unity, Unreal Engine	Upravljanje modelima pomoću skripti, primjena fizičkih principa na modele u stvarnom vremenu
Programi za fotorealistični prikaz	Koriste unaprijed definirane modele	KeyShot, Lumion	Fotorealistični prikaz modela u stvarnom vremenu

Za razliku od CAD sustava, navedeni programi nisu namijenjeni za rad u strojarstvu i precizno modeliranje sklopova. Neki od ovih programa nemaju opciju 3D modeliranja osim dodavanja unaprijed određenih primitiva, dok programi koji omogućuju 3D modeliranje (Blender) nemaju mogućnost parametarskog modeliranja što uvelike otežava izradu preciznih modela i sklopova. Umjesto toga, navedeni programi omogućuju načine manipulacije 3D modela koji uglavnom nisu mogući u CAD sustavima, kao što je primjena skripti i programske logike na individualne dijelove ili cijele sklopove što omogućuje stvaranje interaktivnih simulacija i preglede modela.

U kontekstu ovoga rada, za uspješni prijenos CAD modela potrebno je prenijeti podatke o geometriji, hijerarhiji i spojevima te ograničenjima unutar modela. Osnovni principi potrebni

za razumijevanje procesa prijenosa CAD modela su prijenos podataka o modelu između sustava i kako se prenosi geometrija, topologija, hijerarhija i spojevi i ograničenja unutar modela. Sve te informacije se između sustava prenose pomoću datoteka koje sadrže modele te je za to potrebno razumjeti kako se podatci prenose između programa.

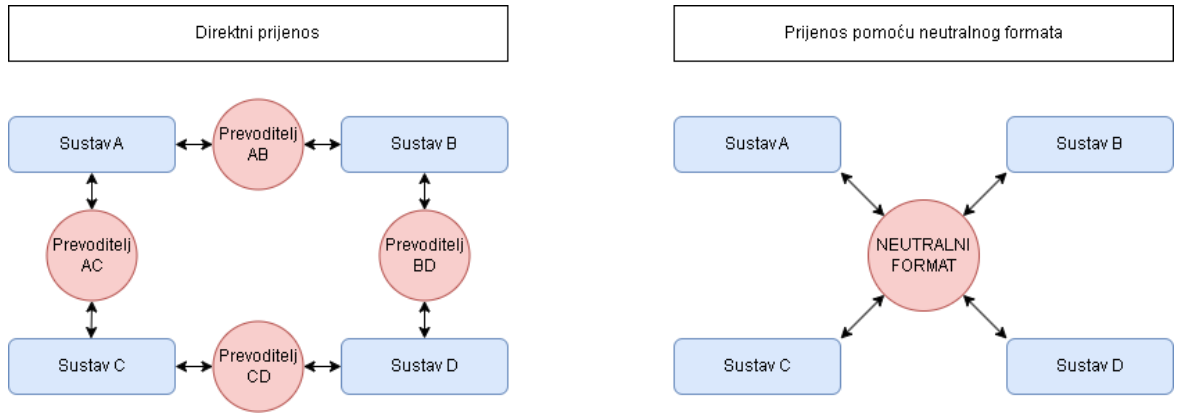
2.3.2. Vrste prijenosa podataka

Postoje dva načina za prijenos podataka o modelu CAD sustava i 3D programa i međusobno između CAD sustava [3]:

1. Prijenos pomoću neutralnog formata
2. Direktni prijenos između sustava

Prijenos pomoću neutralnog formata koristi se standardiziranim formatima kako bi omogućio prijenos podataka o modelu između CAD sustava. Kako bi CAD sustav mogao zapisati i čitati podatke iz nekog formata, potreban mu je prevoditelj. Prevoditeljem se naziva svaki program, integrirani ili zasebni, koji omogućuje CAD sustavu zapisivati ili interpretirati podatke o modelu u specifičnom formatu. Svaki CAD sustav mora imati dva prevoditelja za svaki podržani format: jedan prevoditelj zapisuje podatke o modelu prilikom izvoza modela (*post-processor*), dok drugi interpretira podatke o modelu prilikom uvoza modela (*pre-processor*) [4]. Svi CAD sustavi sadrže integrirane prevoditelje za najraširenije standardizirane formate u industriji, poput STEP i IGES formata.

Direktni prijenos podataka između sustava koristi se prevoditeljima kako bi zapisao ili interpretirao podatke o modelu u izvornom formatu drugoga CAD sustava. Direktni prijenos podataka omogućuje prijenos podataka o modelu koje standardizirani formati nemaju mogućnost prenijeti, poput spojeva unutar nekog sklopa. Primjer direktnog prijenosa je prijenos podataka između CAD programa SolidWorks i CATIA pomoću vlasničkog prevoditelja FormatWorks [5].



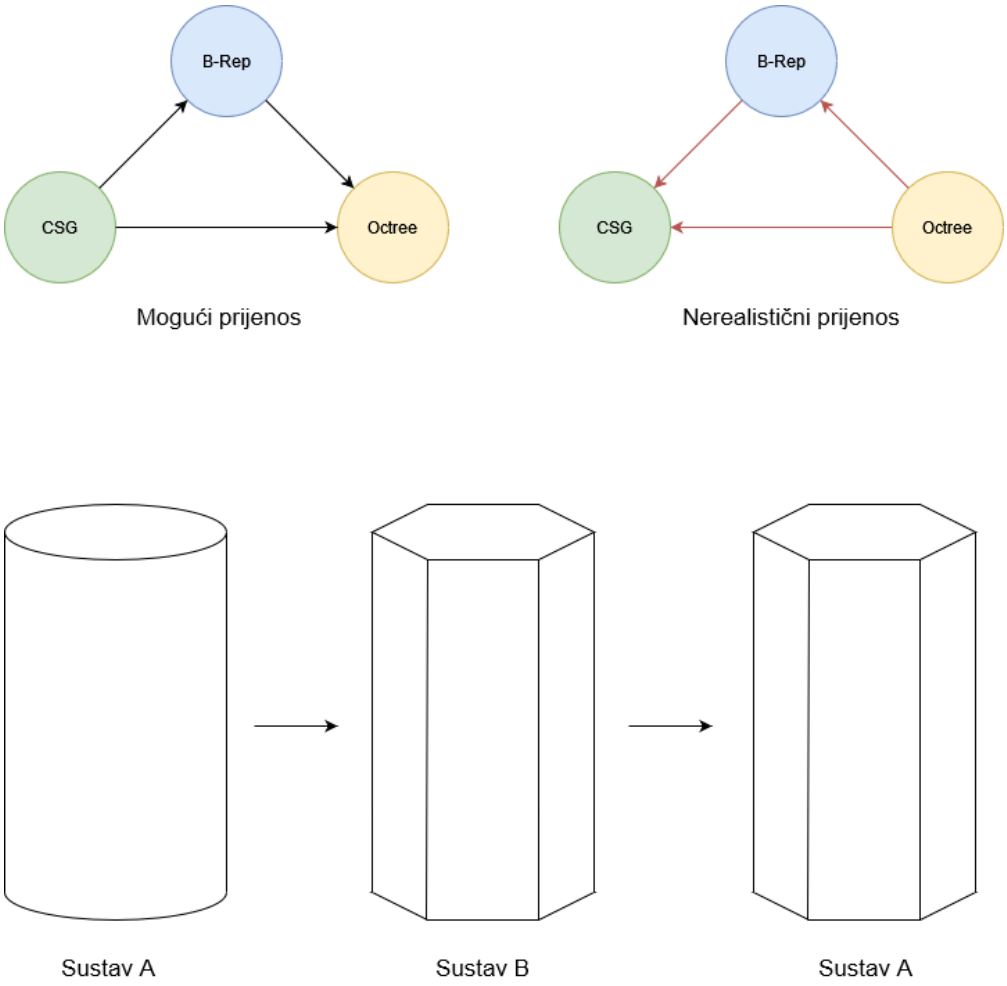
Slika 1. Dijagram načina prijenosa podataka

Kao što je vidljivo iz dijagrama, direktni prijenos podataka zahtjeva prevoditelja za svaki par CAD sustava i njihove izvorne formate. Ako bi još uveli novi potpuno povezani sustav E, bilo bi potrebno razviti čak 8 novih prevoditelja (četiri za interpretaciju formata E u sustavima A-D, i četiri za interpretaciju formata sustava A-D u sustavu E) za prijenos modela. Problem direktnog prijenosa je također zaštićenost vlasničkih formata. Za razvoj prevoditelja nekog vlasničkog formata, potreban je pristup kodu koji izvršava zapisivanje i interpretaciju tog formata. Zbog ekonomskih razloga, tvrtke nisu voljne dati pristup svojem izvornom kodu mogućoj konkurenciji. Zato su postojeći prevoditelji za direktni zapis i interpretaciju vlasničkih formata uglavnom napravljeni za CAD sustave u vlasništvu iste tvrtke (CATIA i SolidWorks) ili za prijenos podataka između najraširenijih CAD sustava (PTC Creo i SolidWorks).

Za direktni prijenos još postoji i mogućnost korištenja vanjskih prevoditelja, neovisnih o tvrtkama. Ovi prevoditelji omogućavaju prijevod izvornih formata u druge izvorne formate te tako miču potrebu za integriranim prevoditeljima. Kako bi uspješno ispunjavali funkciju prijevoda formata, potreban im je pristup izvornom kodu prevoditelja vlasničkih CAD programa te su ti programi često skupi zbog svih licenci koje su potrebne za razvoj različitih prevoditelja. Najpoznatiji ovakav program je CAD exchanger [6]. Također, u sklopu ovoga rada koristit će se vanjski prevoditelj Pixyz, koji omogućuje prijenos podataka o topologiji i hijerarhiji CAD modela u platformu za razvoj video igara Unity. Princip rada i razlog odabira Pixyz-a bit će detaljnije objašnjen u 3. poglavlju. Da bi se opravdao odabir direktnog vanjskog prevoditelja umjesto prijenosa neutralnim formatom za potrebe ovoga rada, potrebno je sagledati princip prijenosa neutralnim formatima te sve nedostatke koje ti formati imaju naspram direktnom prijenosu.

Prijenos pomoću neutralnog formata, kao što je već naznačeno i prikazano u dijagramu, zahtjeva samo dva prevoditelja po standardiziranom formatu. Zbog raširenosti standardiziranih

formata te integriranih prevoditelja tih formata prisutnih u CAD sustavima, prijenos pomoću neutralnog formata omogućuje jednostavni prikaz modela napravljenog u različitim CAD sustavima bez potrebe za prevoditeljem izvornih formata tih sustava. Neutralni formati poput OBJ i STL omogućuju čak i prijenos podataka o obliku u 3D programe, objašnjene u *2.1.1.*. Nedostatci prijenosa pomoću neutralnog formata postaju očiti prilikom prijenosa sklopova neutralnim formatom. Zbog razlike u značajkama koje različiti CAD sustavi nude, standardizirani formati ne zapisuju podatke o značajkama koje nisu univerzalne za svaki CAD sustav. Standardizirani formati ne prenose podatke poput informacija o spojevima u sklopu i materijalima dijelova te ih zbog toga nije preporučljivo koristiti u projektima gdje je zahtjev prijenos navedenih podataka. Također je potrebno naglasiti kako prijenos pomoću neutralnog formata može rezultirati gubitkom informacija o obliku modela. Prijenosom podataka između CAD sustava koji koriste različite tehnike modeliranja automatski se gubi na kvaliteti modela zbog potrebe za aproksimacijom oblika modela. Iako u današnjici ovo nije značajan problem jer većina CAD sustava koriste B-Rep tehniku modeliranja, ovaj problem je još jedan nedostatak korištenja neutralnog formata. [3]



Slika 2. Prikaz mogućih prijenosa između sustava sa različitim tehnikama modeliranja i kvalitativni prikaz gubitka informacija prilikom takvog prijenosa [3]

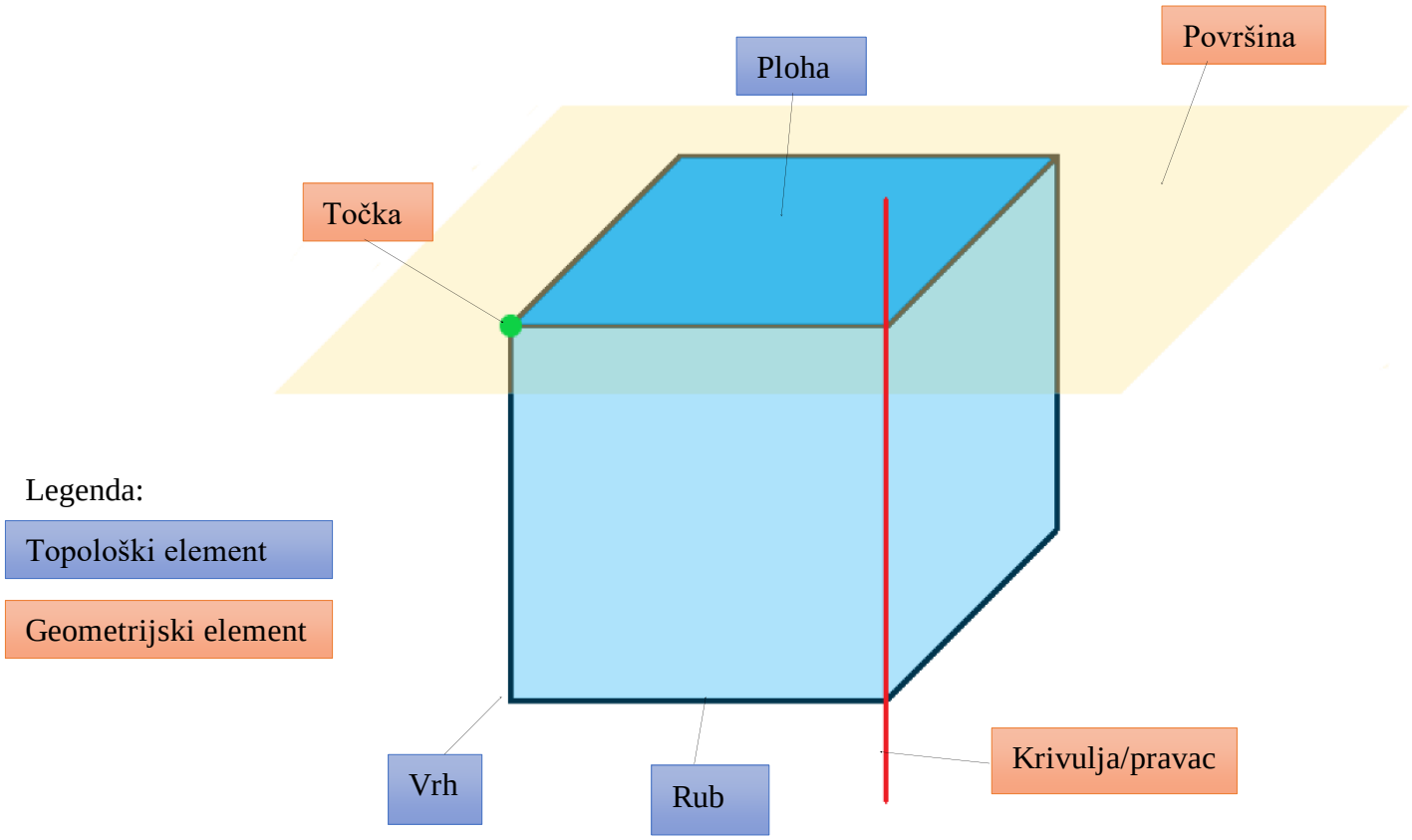
Danas postoje razni alati koji olakšavaju prijenos podataka neutralnim formatom. Modularni FOSS *(Free Open Source Software)* programi poput FreeCAD-a [7] te API-i *(Application Programming Interface)* poput SolidWorks API [8] omogućuju inženjerima pristup internim funkcijama CAD sustava kako bi zasebno pristupili podatcima koje nije moguće prenijeti neutralnim formatima. Iako je ovaj proces teže izvediv od direktnog prijenosa modela, njime se smanjuje potreba za pristupom zaštićenim sredstvima tvrtki za razvoj prevoditelja.

Do sada je opisano kako se podatci o modelu prenose između sustava u cjelini. No za razumijevanje cijeloga procesa prijenosa, potrebno je analizirati način prijenosa svakog aspekta modela koji je naznačen kao potreban za uspješan prijenos u okviru ovoga rada. Najvažniji od tih aspekata je oblik CAD modela, kojeg definiraju geometrija i topologija modela.

2.3.3. Prijenos oblika

Kako bi bilo moguće objasniti proces prijenosa oblika modela, potrebno je objasniti kako CAD sustavi interpretiraju i zapisuju podatke o obliku B-Rep modela.

U CAD sustavima, B-Rep modeli se interpretiraju kao sveobuhvatni opisi geometrije objekata, gdje svaki element modela nosi precizne informacije o obliku, dimenzijama i međusobnim odnosima. Pri interpretaciji B-Rep modela, CAD sustavi obrađuju vrhove, bridove i plohe kako bi stvorili točan prikaz fizičkog objekta. Svaki od ovih elemenata povezan je s matematičkim funkcijama koje opisuju geometriju površina i rubova, omogućujući visoku preciznost u modeliranju i analizi. [3]



Slika 3. Grafički prikaz uloga geometrije i topologije u B-Rep modelu kocke

Slikom 3. može se jednostavno prikazati kako izgleda suradnja topologije i geometrije u modelu. Naime, svaki topološki element ima svoj geometrijski ekvivalent koji koristi kako bi stvorio mrežu na koju se može primijeniti nekakva tekstura. Svaka ploha je dio neke površine, svaki rub je dio neke krivulje ili pravca i svaki vrh je neka točka.

Zahvaljujući ovome zapisu svaki model može se prikazati kao skup parametara koji opisuju geometriju i topologiju tog modela. Ovakav tip modela naziva se parametarski model.

Zbog matematički definiranog oblika modela, moguće je izvesti prijenos B-Rep modela u sustave koji ne koriste B-Rep modeliranje pomoću aproksimacije geometrije i topologije. [3]

Slika 4. Primjer zapisa SolidWorks modela po STEP A214 standardu

Glavna razlika između CAD sustava i 3D programa opisanih u 2.2.1. jest da 3D programi ne koriste parametarske već mnogokutne modele (*Polygonal Model*). Mnogokutni modeli napravljeni su od mnogokutnih površina (*Polygonal Mesh*) te se oblikuju povećavanjem, smanjivanjem, dodavanjem ili brisanjem tih mnogokuta. Dok ovaj pristup nudi veću slobodu pri izradi modela u svrhe umjetnosti ili zabave, glavni problem je nedostatak preciznosti koji definira parametarske modele. Naime, površine mnogokutnih modela definirane su „slobodnom rukom“, uglavnom koristeći interno mjerilo 3D programa koje često ne ovisi o standardnim mjernim jedinicama (nemogućnost uvođenja parametara). Zbog toga se mnogokutni modeli ne koriste u CAD sustavima i parametarski modeli u 3D programima. Jasno je da ova razlika predstavlja problem prilikom prijenosa modela između CAD sustava i 3D programa, specifično prilikom prikaza modela.

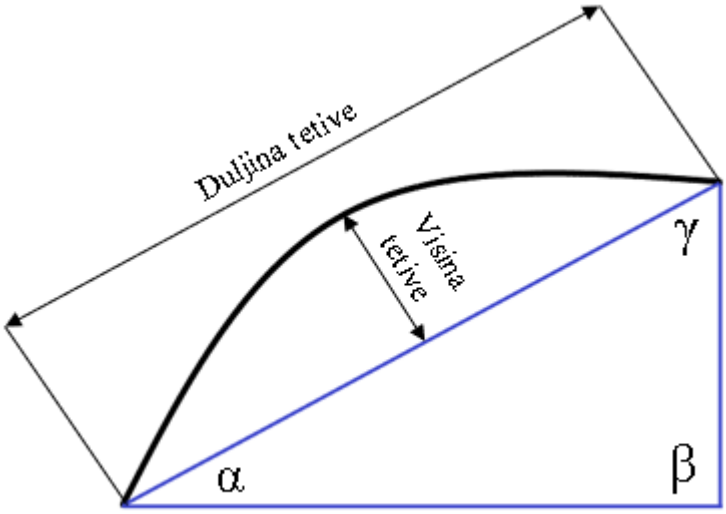
2.3.3.1. Prikaz modela

U računalnoj grafici, modeli se najčešće prikazuju pomoću skupova trokuta. [3] Takav prikaz je generalno najstabilniji izbor za računalni hardware. Postoje i drugi prikazi, poput žičanog (*Wireframe*), no za 3D programe se koristi prikaz pomoću trokuta te će se samo taj prikaz obraditi u ovome radu [3]. Potrebno je još naglasiti da postoje prikazi modela pomoću

površina krivulja, specifično NURBS (*Non-Uniform Rational B-Spline*) površine, no zbog velikog opterećenja računalnog hardware-a prilikom prikaza ovakvih površina u stvarnom vremenu, one se također neće obrađivati u ovome radu [9].

Prikaz mnogokutnih modela jednostavniji je od prikaza parametarskih modela jer su mnogokutni modeli već sastavljeni od mnogokuta koje je moguće implicitno podijeliti na skupove trokuta, ili su modelirani pomoću samih trokuta. Ovaj „*what you see is what you get"* prikaz čini ih idealnim za rad u VR aplikacijama koje nisu direktno integrirane u neki CAD sustav. Razlog tome je to da je većina VR aplikacija razvijena na platformama za razvoj video igara koje su prilagođene za rad s mnogokutnim modelima.

Parametarski modeli, zbog svoje matematičke prirode, zahtijevaju proces zvan triangulacija kako bi se prikazali pomoću skupova trokuta. Triangulacija je metoda raspodjele mnogokutne površine na skup trokuta [10]. Ako je geometrija modela sastavljena isključivo od ravnih, mnogokutnih površina, onda je prikaz veoma izravan, pošto ne postoje krivulje unutar modela. Ravninske plohe, nastale od ravnih mnogokutnih površina, triangulacijom su podijeljene na skup n broja trokuta, gdje je n najmanji broj trokuta koji su potrebni za prikaz mnogokuta. Modeli koji sadrže zakrivljene površine su teži za prikazati, pa se za njih provodi linearna interpolacija zakrivljenih rubova i površina kako bi krivulje bile podijeljene na skup točaka. Zatim se te točke povežu ravnim linijama, i primjenjuje se proces triangulacije dobivenih rubova. Kvalitetu dobivenog prikaza moguće je kontrolirati promjenom tolerancije tetive. Tolerancija tetive definirana je visinom tetive, duljinom tetive, i minimalnom veličinom kutova u trokutu [3]. Što je tolerancija tetive manja, to će kvaliteta prikaza biti bolja te će prikaz biti intenzivniji za hardware računala.



Slika 5. Tolerancija tetive

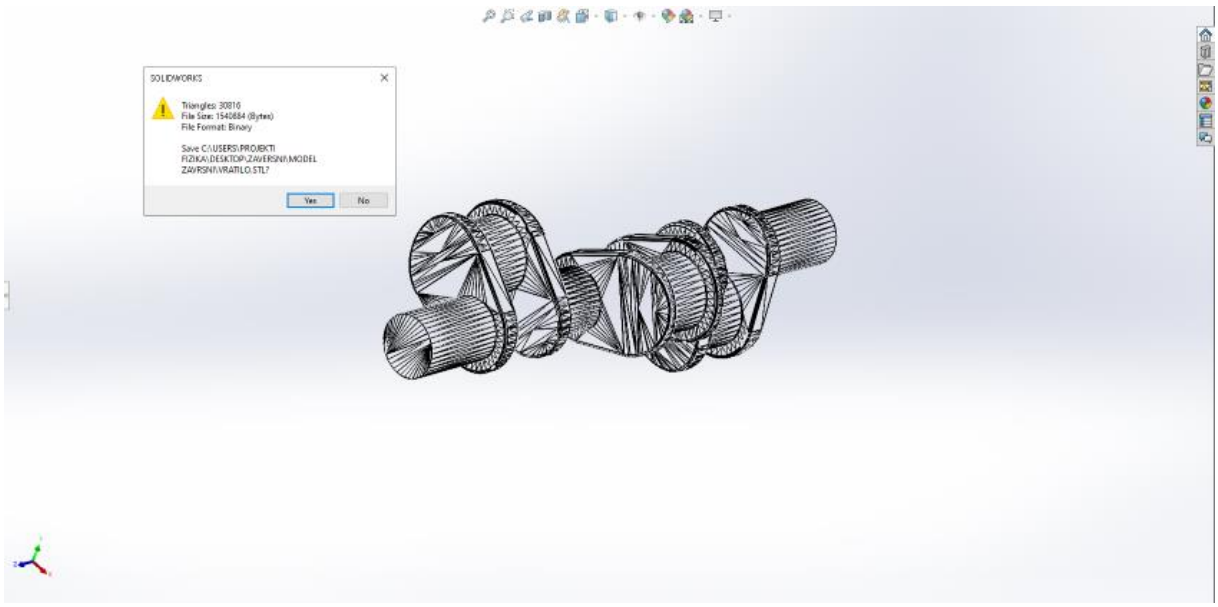
Na slici 5. vidljivo je kako tolerancije tetive utječe na veličinu trokuta generiranog tijekom procesa triangulacije. Visina tetive određuje maksimalnu udaljenost tetive od najudaljenije točke krivulje, duljina tetive određuje maksimalnu duljine te tetive, a minimalni kut trokuta određuje najmanji dozvoljeni kut α, β i γ u trokutu.

Unutar CAD sustava, proces triangulacije moguće je veoma precizno provoditi u stvarnome vremenu, jer su CAD sustavi optimizirani za prikaz parametarskih modela. No, kako 3D programi nisu napravljeni za rad s parametarskim modelima, CAD modeli moraju se prilikom prijenosa u 3D program pretvoriti u mnogokutne modele.

2.3.3.2. Prijenos geometrije i topologije CAD modela u 3D program

Kako bi se oblik parametarskog CAD modela mogao prenijeti u 3D program, potrebno ga je pretvoriti u mnogokutni model. Ovim prijenosom gubi se referentna geometrija modela pošto se topologija pomoću procesa zvanog teselacija redefinira kao skup mnogokuta i ne ovisi o geometriji modela. Teselacija je proces sličan triangulaciji, te je jedina razlika između njih to što teselacija razdjeljuje površinu na skupove mnogokuta umjesto trokuta. Ovaj proces uvelike smanjuje kompleksnost modela, pošto se nezakrivljene mnogokutne površine rascjepkane na trokute pretvaraju u jedan mnogokut. Veličina datoteke teseliranog modela manja je od veličine trianguliranog modela zbog manjeg broja topoloških podataka koje je potrebno spremiti, no za prikaz u stvarnome vremenu teselirani model još uvijek je potrebno triangulirati kako bi se mogao prikazati. Očito je da se ovim procesom gube i svi promjenjivi parametri unutar modela.

3D programi nemaju integrirane prevoditelje za CAD formate, bilo izvorne ili standardne, te je potrebno izvršiti prijenos modela pomoću vanjskog prevoditelja ili je model potrebno pretvoriti u mnogokutni model pomoću CAD sustava. Svi CAD sustavi imaju mogućnost izvoza modela u nekom formatu mnogokutnog modela, najčešće u STL formatu za potrebe 3D printanja. Neki CAD sustavi poput FreeCAD-a omogućuju izvoz modela i u drugim mnogokutnim formatima poput OBJ.



Slika 6. Spremanje SolidWorks modela koljenastog vratila u STL formatu

Potrebno je naglasiti kako ovaj način izvoza modela prenosi isključivo informacije o obliku modela. Kada bi se na ovaj način spremio neki sklop, za svaki zasebni dio sklopa i podsklopova (ako postoje) bila bi napravljena po jedna STL datoteka.

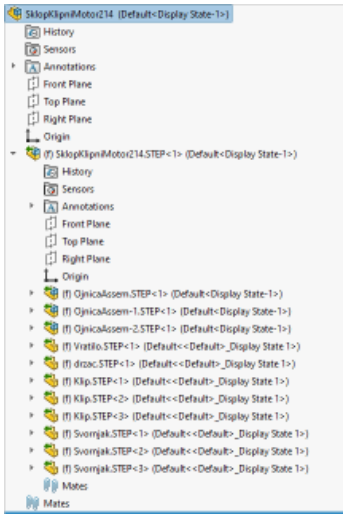
Za prijenos oblika sklopova kao jedinstvene datoteke, potrebno je koristiti vanjski prevoditelj ili 3D program s ugrađenim prevoditeljem za CAD formate poput Rhinocerosa. Primjer vanjskih prevoditelja su dodatci Pixyz za Unity3D Engine i Datasmith za Unreal Engine.

2.3.4. Prijenos hijerarhije dijelova sklopa

CAD modeli koji se sastoje od više dijelova imaju definirane hijerarhije dijelova ili sklopova. Manipulacijom sklopa pomiču se svi dijelovi kojima je sklop nadređen u hijerarhiji. Naknadno, sve funkcije ili spojevi koji utječu na sklop utječu i na dijelove i sklopove od kojih se taj sklop sastoji. Pravilno definirana hijerarhija također uvelike olakšava rad s modelom te navigaciju u kompleksnim modelima.

Prijenos hijerarhije je jednostavan proces zahvaljujući modernim CAD sustavima i standardima. STEP format razlikuje dijelove unutar sklopa te je u zapisu sklopa dovoljno ukazati na to koji sklop je nadređen kojem dijelu ili podsklopu.

3D programi, kao što je već naglašeno, nemaju ugrađene prevoditelje za CAD formate te se za prijenos hijerarhije koriste vanjski prevoditelji.



Slika 7. Prikaz hijerarhije STEP datoteke unutar SolidWorksa

2.3.5. Prijenos spojeva i ograničenja unutar sklopa

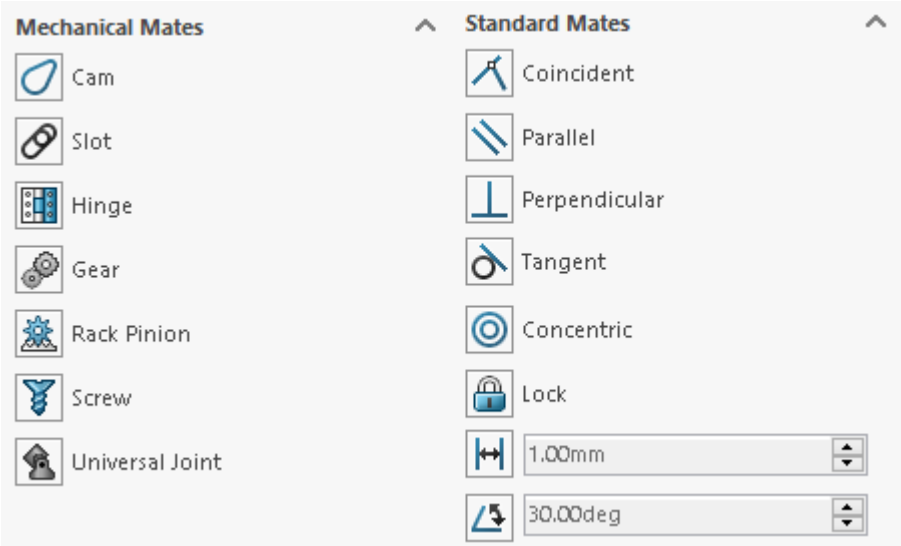
Kako bi dijelovi unutar sklopnog modela u CAD sustavu činili smislenu skupinu te se ponašali željenim načinom, potrebno je na njih primijeniti logiku koja definira njihov položaj i utjecaj u odnosu na druge dijelove sklopa. Da bi se to ostvarilo potrebno je primijeniti ograničenja i spojeve na dijelove.

Ograničenja su odnosi između jednostavnih geometrijskih elemenata (ravnina, pravaca, i točaka) koji definiraju poziciju i orijentaciju topologije dijelova kojima pripadaju. Ograničenja također definiraju stupnjeve slobode dijelova s obzirom na njihov lokalni koordinatni sustav. Primjeri ograničenja su koncentričnost, paralelnost, dodir dva geometrijska elementa, itd. [3]

Spojevi, ili kinematski mehanizmi, u CAD modelima definirani su kao skup ograničenja s logičnom namjenom. Za razliku od ograničenja, spojevi su funkcionalni te predstavljaju spoj strojnih elemenata i koriste za simulaciju rada modela. Primjeri spojeva su rotacijski spojevi, kruti spojevi, opružni spojevi, zupčani spojevi…[3]

Za razliku od oblika i hijerarhije, spojevi i ograničenja unutar modela se ne prenose između CAD sustava bez direktnog prevoditelja jer neutralni formati ne zapisuju informacije o

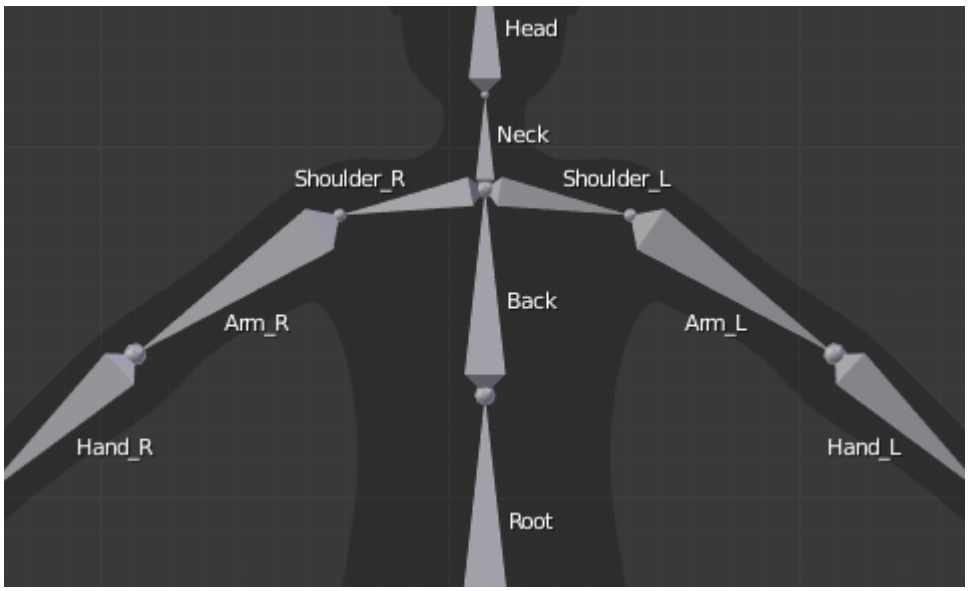
njima. Razlog tome je prevelika varijacija u njihovom zapisu između CAD sustava. Različiti CAD sustavi mogu zapisivati ograničenja na dva načina: pomoću cijelokupnog geometrijskog elementa ili samo pomoću točaka. Zapis pomoću geometrijskog elementa ostvaruje se definiranjem ograničenja između točke, ravnine ili pravca/krivulje geometrije dijela, dok se zapis pomoću točaka ostvaruje definiranjem ograničenja samo između točaka unutar geometrije. Iako je zapis nekih ograničenja, poput udaljenosti geometrijskih elemenata, univerzalan svim CAD sustavima, većina se sklopova rijetko sastoji samo od tih ograničenja. Zbog toga, neutralni formati ne zapisuju i ne prenose informacije o ograničenjima i spojevima između CAD sustava. [11]



Slika 8. Neki spojevi i ograničenja unutar SolidWorks-a. S lijeve strane se nalaze spojevi (*Mechanical Mates*) a s desne ograničenja (*Standard Mates*)

Ovaj problem je potenciran prilikom prijenosa CAD modela u 3D programe. Dok je za prijenos ograničenja i spojeva između dva CAD sustava moguć pomoću direktnih prevoditelja, takvi prevoditelji ne postoje za 3D programe. Neki 3D sustavi, poput Blendera, Autodesk Maya, i ostalih programa za modeliranje mnogokutnih modela, uopće nemaju mogućnost definiranja veza sličnih ograničenjima i spojevima bez veoma kreativne primjene strukturalnih *kosti* (Bones) koje služe za pokretanje modela. *Kosti* su alati koji se koriste specifično u programima za stvaranje mnogokutnih modela i animacija. Jedna kost sastoji se od 3 dijela: glava, rep i tijelo. Glava kosti postavlja se na željeni početni skup vrhova unutar modela, a rep modela na željeni završni skup vrhova. Tijelo kosti automatski povezuje sve međusobno povezane vrhove između glave i repa. Vrhovi modela se tako pomicanjem repa kosti kreću oko glave kosti, sa šest stupnjeva slobode ili unutar ograničenja koje je određeno samoj kosti u cijelosti. Pomoću

njih stvaraju se tzv. *kosturi za animaciju* (Animation Rigs), pomoću kojih je model moguće pokretati kroz 3D prostor. Dok je ovakav pristup idealan za industrije videoigara i računalne animacije, pomoću njega nije jednostavno moguće simulirati ponašanje ograničenja i spojeva unutar CAD modela. Još jedna prepreka je da se, kako je već naglašeno, u mnogokutnom modeliranju koristi pristup slobodne ruke, tj. modeli nisu definirani parametrima već arbitrarnim mjerilom ugrađenom u 3D program. Dok je vjerojatno moguće simulirati spojeve i ograničenja CAD modela kostima, taj proces bi sigurno bio puno kompliciraniji i nepreciznij od korištenja CAD sustava.

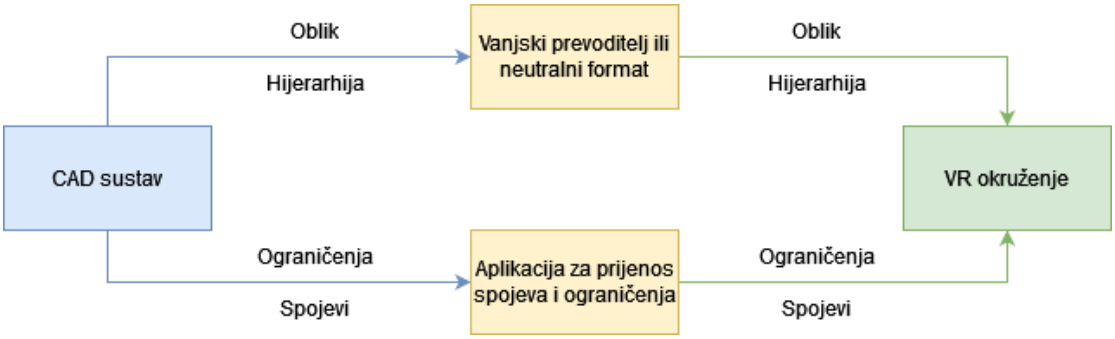


Slika 9. Primjer korištenja kosti za stvaranje „animation rig-a“ modela čovjeka unutar Blender-a[12]

No, specifična potkategorija 3D programa ima mogućnost izravno emulirati spojeve i ograničenja pristupne u CAD modelima. Platforme za razvoj 3D igara, uz to da imaju direktnu integraciju s VR sustavima, mogu simulirati željeno ponašanje modela zahvaljujući primjeni programskog jezika na modele. CAD model, kada je uveden u platformu za razvoj videoigara, nema mogućnost više mijenjati svoj oblik, no na njega se mogu primijeniti skripte napisane u programskom jeziku platforme koje simuliraju njegovo ponašanje u 3D prostoru. Ove skripte imaju mogućnost simuliranja ograničenja definicijom osi translacije i rotacije na kojima se dijelovi smiju kretati, definicijom povezanih dijelova i kretnja oko njih, i tako dalje. No, za prijenos spojeva i ograničenja CAD modela u 3D program prema provedenom pregledu alata ne postoji alat na tržištu, uglavnom zbog istih razloga zašto direktni prijenos između dva CAD sustava bez prevoditelja nije moguć: zapis spojeva i ograničenja previše varira ovisno o sustavu.

3. METODOLOGIJA

U ovome poglavlju će se na temelju pregleda literature definirati kako će se odvijati prijenos modela iz CAD sustava u VR okruženje. Potrebno je definirati kojom će se od dvije metode definirane u 2.2.2. prenositi oblik i hijerarhija modela. Za spojeve i ograničenja modela, potrebno je definirati kako će se ti podatci zapisati izvan CAD sustava te kako će ih se interpretirati u VR okruženju. Na temelju tih zahtjeva, odabrat će se CAD sustav u kojem je model potrebno izraditi, način prijenosa oblika i hijerarhije, alate za izradu aplikacije za prijenos spojeva i ograničenja, te napokon platforma za izradu video igara koja će služiti kao VR okruženje. Zatim će se postaviti zahtjevi modela za prijenos i po njima izabrati CAD modeli na temelju kojih će se analizirati uspješnost prijenosa.



Slika 10. Kvalitativni dijagram prijenosa podataka o modelu prije postavljanja zahtjeva za prijenos

Kako bi se počeli definirati zahtjevi prijenosa, potrebno je prvo proučiti kako se priprema model za prijenos.

3.1. Priprema modela za prijenos

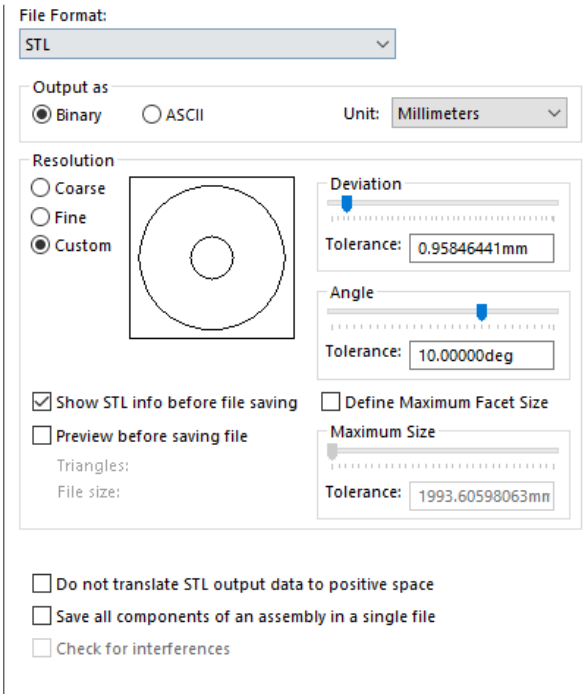
Kako bi se izveo uspješni prijenos modela definiran u 2.2.1., potrebno je koristiti različite alate za prijenos oblika, hijerarhije, spojeva i ograničenja modela. Zato će se u ovome poglavlju na temelju literature postaviti kriteriji za odabir navedenih alata te ostalih radnji koje su potrebne za pripremu modela za prijenos.

3.1.1. Prijenos oblika i hijerarhije

Glavni zahtjev za prijenos oblika definiran u literaturi je precizan prijenos B-Rep modela u mnogokutni model. Ovaj proces moguće je izvesti na dva načina, pomoću neutralnog formata ili vanjskog prevoditelja. Obje metode provode teselaciju modela, ali pošto će se u

okviru ovoga rada koristiti samo jedna od navedenih metoda u nastavku je potrebno definirati koja je metoda prigodnija za ovaj rad.

Kod prijenosa modela neutralnim formatom, većina CAD sustava jedino ima opciju izvoza mnogokutnog modela u STL formatu, koji iako provodi teselaciju i generira mnogokutni model, ne sadržava nikakve druge tehničke podatke osim oblika modela. Prilikom spremanja modela sklopa u STL formatu, generirat će se zasebna datoteka za svaki pojedinačni dio sklopa i podsklopova u tom sklopu. Kako u STL formatu nije zapisano gdje se u sklopu nalazi koji dio, uvozom STL formata u VR okruženje će svi dijelovi biti postavljeni u istu točku i time će se narušiti struktura sklopa. Na slici 11. vidljive su opcije za izvoz modela u STL formatu iz SolidWorks CAD sustava



Slika 11. Korisničke postavke za izvoz modela u STL format u CAD sustavu SolidWorks. Prilikom prijenosa u VR okruženje, potrebno je obratiti posebnu pozornost na opcije *Deviation* i *Angle* za kvalitetu teselacije modela.

Ovakav način prijenosa također stvara problem i za hijerarhiju sklopa. Pošto je svaki dio prenesen pojedinačno, hijerarhija vidljiva u CAD sustavu trebala bi se ponovo ručno replicirati u VR okruženju. Za sklopove s velikom količinom dijelova i podsklopova, ovaj proces zahtijevao bi previše vremena za provedbu. Kako se pod modelima u okviru ovoga rada ne podrazumijevaju samo pojedinačni dijelovi već i sklopovi, prijenos neutralnim formatom ne ispunjava uvjete prijenosa modela. Zbog toga, za prijenos oblika i hijerarhije koristit će se vanjski prevoditelj, što znači da je obavezno model spremiti u izvornom formatu CAD sustava.

3.1.2. Prijenos ograničenja i spojeva

Za razliku od prevoditelja za prijenos oblika i hijerarhije, na tržištu ne postoji već razvijeni prevoditelj za prijenos ograničenja i spojeva unutar CAD modela zbog razloga navedenih u 2.2.4. Važno je također razmotriti činjenicu da nemaju svi 3D programi mogućnost replicirati funkcionalnosti CAD ograničenja i spojeva, već koriste metode poput strukturalnih kostiju opisanih u 2.2.4. koje su prilagođene za izradu animacija a ne funkcionalnih CAD modela. Zato za prijenos ograničenja i spojeva nije potrebno samo definirati kriterije koje mora ispunjavati aplikacija za njihov prijenos, već i za VR okruženje u koje se oni uvode.

Najbolji kandidati za to okruženje su platforme za razvoj video igara. Potrebno je ograničenja i spojeve iz modela zapisati u zasebnu datoteku u formatu koji platforma može interpretirati te unutar platforme razviti skripte koje na dijelove sklopa primjenjuju kod koji se ponaša jednako ili barem slično ograničenjima i spojevima u CAD sustavima. Na primjer, ako CAD sustav koristi koncentrično ograničenje koje omogućava rotaciju oko samo jedne osi, skripta u platformi treba simulirati isto ponašanje, omogućujući rotaciju oko te iste osi.

Glavni zahtjev aplikacije za prijenos je mogućnost prijenosa svih ograničenja i spojeva u modelu neovisno o njihovom broju i funkciji. Analogno tome, sam model nije potrebno posebno pripremiti za ovaj prijenos, već je potrebno odabrati CAD sustav koji dopušta pristup informacijama o spojevima i ograničenjima modela iz vanjske aplikacije.

3.2. Provedba prijenosa ograničenja i spojeva

Za razvoj aplikacije za prijenos ograničenja i spojeva potrebno je definirati tok podataka o ograničenjima i spojevima između CAD sustava i platforme za razvoj videoigara.

3.2.1. Prijenos ograničenja i spojeva iz CAD sustava u alat

Kako bi alat mogao prepoznati i interpretirati podatke o ograničenjima i spojevima u modelu, potrebno je ostvariti komunikaciju s aktivnom instancom CAD sustava s otvorenom datotekom modela. To se može izvesti na dva načina: pristupom izvornom kodu sustava ili korištenjem API-ja nekog CAD sustava kako bi se pomoću koda dobili svi potrebni podatci.

Zatim, kada se ostvarila komunikacija s aktivnom instancom CAD sustava, potrebno je identificirati i izolirati ograničenja i spojeve te dijelove s kojima su povezani unutar modela. Nakon što su ti podatci uspješno izolirani, potrebno ih je zapisati u neutralnom tekstualnom formatu poput .csv kako bi platforma za razvoj video igara mogla pročitati i procesirati podatke.

Kako je naglašeno, postoje dva različita načina kako da se ovaj prijenos ostvari. Od njih, puno jednostavniji je pristup korištenja API-ja CAD sustava. Ovaj pristup ne samo da je lakši za izvesti, već je i sigurniji pošto su sve funkcije unutar API-ja odvojene od funkcija potrebnih za rad sustava te nemaju mogućnost mijenjati i analogno tome naštetiti tim funkcijama. Modifikacijom izvornog koda i time funkcija potrebnih za rad sustava korisnik koji nije već upoznat s kodom nosi rizik da potpuno promjeni ponašanje sustava, ili da sustav prestane funkcionirati zbog brisanja ili mijenjanja važnih dijelova koda.

Prvi dio toka između CAD sustava i alata za prijenos je definiran, te je odlučeno da će se koristiti CAD sustav s mogućnosti korištenja internog API-ja. Potrebno je sada definirati kako će se tekstualni podatci interpretirati i primijeniti na model u platformi za razvoj videoigara u nastavku toka.

3.2.2. Prijenos ograničenja i spojeva iz alata u platformu za razvoj videoigara

U nastavku toka ključni zadatak je interpretirati tekstualnu datoteku koja sadrži podatke o ograničenjima i spojevima dobivenim iz CAD modela, te primijeniti te podatke u platformi za razvoj videoigara.

Za učitavanje tekstualne datoteke može se koristiti standardna biblioteka u programskom jeziku koji se koristi za razvoj unutar platforme. Ova biblioteka će čitati tekstualnu datoteku i pretvoriti podatke u oblik koji je jednostavan za daljnju obradu, poput liste. Zatim će se, na temelju te liste, na dijelove modela redom primjenjivati skripte koje odgovaraju ograničenjima i spojevima po kojima su napisane. Skripta može primjenjivati već postojeću funkciju unutar platforme ili novu funkciju napisanu specifično za taj spoj. Za realizaciju ovoga procesa važno je odabrati odgovarajuće alate za razvoj aplikacije za prijenos te aplikaciju za prijenos ostalih aspekata modela, oblika i hijerarhije.

3.3. Opis programa za prijenos modela

Na temelju zahtjeva definiranih u 3.1. i 3.2., mogu se odabrati programi koji će se koristiti u okviru ovoga rada. Potreban je CAD sustav koji sadrži vlastiti API preko kojeg je moguće zapisivati podatke o spojevima i ograničenjima, vanjski prevoditelj za prijenos oblika i hijerarhije iz CAD sustava u VR okruženje, te platforma za razvoj video igara koja ima mogućnost VR integracije. Za razvoj aplikacije za prijenos spojeva i ograničenja potrebni su

programerski alati s kojima je moguće povezati korisničko sučelje s API-jem za lakše korištenje aplikacije.

Za CAD sustav koristit će se popularni CAD sustav SolidWorks proizvođača Dassault Systems, zahvaljujući mogućnosti korištenja SolidWorks API-ja. Za platformu za razvoj video igara koristit će se program Unity3D proizvođača Unity Technologies. Razlog tome je raširenost i jednostavnost platforme što olakšava suradnju prisutnu u open-source projektima. Također, rad u Unity3D omogućuje pristup Pixyz prevoditelju koji će se koristiti za prijenos oblika i hijerarhije modela. Svo potrebno programiranje bit će napisano u C# programskom jeziku, koristeći alat Visual Studio proizvođača Microsoft. Za izradu aplikacije za prijenos spojeva i ograničenja koristit će se .NET framework s Microsoftovim paketom Windows Forms kako bi se izradila desktop aplikacija.

3.3.1. Solidworks API

SolidWorks API je SolidWorks-ovo sučelje za programiranje aplikacija, specifično automatizacije operacija unutar SolidWorks-a i uređenja korisničkog sučelja sustava. API pruža direktni pristup funkcijama poput stvaranja geometrije, uvoza dijelova u sklop, spajanja dijelova itd. API također pruža pristup informacijama o modelu, među kojima su za ovaj rad najvažnije informacije o spojevima i ograničenjima unutar modela. [13] Na slici 12. moguće je vidjeti nekoliko *sučelja* (interfaces) u SolidWorks.Interop.sldwrks namespaceu uključenog u SolidWorks API.

ILoftFeatureData	Allows access to a loft feature.
ILoop	Obsolete. Superseded by ILoop2.
ILoop2	Allows access to the owning face and to the list of edges and coedges contained in the loop.
IMacroFeatureData	Allows access to the data that defines a macro feature.
IMagneticLine	Allows access to a magnetic line.
IManipulator	Allows access to a manipulator, which is similar to the triad or the drag arrow (also called a handle), in a SOLIDWORKS part or assembly document.
IMassProperty	Obsolete. Superseded by IMassProperty2.
IMassProperty2	Allows access to individual mass properties as found in the Mass Properties dialog box.
IMassPropertyOverrideOptions	Allows access to mass property override options.
IMate	Obsolete. Superseded by IMate2.
IMate2	Allows access to various assembly mate parameters.
IMateEntity	Obsolete. Superseded by IMateEntity2.
IMateEntity2	Allows access to mated entities and the assembly mate definition.

Slika 12. Isječak *interfaces* prisutnih u SolidWorks.Interop.sldworks *namespace*-u [14]

Najvažniji *interface* koji će se koristiti u sklopu ovoga rada bit će „IMate2 interface“, koji omogućuje pristup različitim parametrima spojeva i ograničenja u modelu poput dijelova spoja, vrsti spoja ili ograničenja itd.

SolidWorks API radi na način da iz koda direktno otvara novu ili dohvaća već postojeću instancu SolidWorks CAD sustava te primjenjuje napisane funkcije na trenutno otvorenu datoteku.

3.3.2. Visual Studio, .NET framework i Windows Forms

Za izradu aplikacije za prijenos spojeva i ograničenja potrebno je odrediti u kojem programskom jeziku će ona biti napisana. SolidWorks API ima mogućnost rada u tri programska jezika: Visual Basic, C# i C++. Kako je jedan od zahtjeva za olakšani rad s aplikacijom to da ona ima korisničko sučelje, koristit će se programski jezik C# koji zahvaljujući .NET frameworku služi za razvoj velikog broja Windows aplikacija.

3.3.2.1. Visual Studio

Visual Studio je IDE (*Integrated Development Environment*) software tvrtke Microsoft koji se koristi primarno za izradu računalnih i mobilnih aplikacija. Visual Studio omogućuje automatizaciju procesa kompilacije aplikacije te ekstenzivne alate za testiranje i pronalazak grešaka, što uvelike ubrzava proces izrade. Visual Studio također sadrži direktnu integraciju s .NET framework-om i njegovom SDK-u (*Software Development Kit*) potrebnim za izradu računalnih aplikacija.[15]

3.3.2.2. .NET framework

.NET framework, također u vlasništvu tvrtke Microsoft, je programska infrastruktura koja služi za olakšavanje izrade aplikacija za Windows operativni sustav. .NET omogućava interpretaciju napisanog koda te prevođenje tog koda u strojni kod u stvarnome vremenu pomoću JIT (*Just-In-Time*) kompajlera integriranog u .NET infrastrukturu. Ovo je omogućeno zahvaljujući CLR sastavnici .NET framework-a. CLR (*Common Language Runtime*) je softverski sustav koji automatizira kompilaciju, upravljanje memorijom, sigurnost koda i ostale aspekte .NET aplikacije. .NET također nudi pristup raznim bibliotekama za razvoj računalnih aplikacija, specifično Windows Forms biblioteci za razvoj Windows aplikacija. [16]

Windows Forms je .NET biblioteka za razvoj GUI-a (*Graphical User Interface*) za Windows aplikacije. Zahvaljujući ovoj biblioteci, moguće je definirati korisničko sučelje za

aplikaciju, što uvelike olakšava korištenje aplikacije i njenih funkcija. Forms elementi poput gumba omogućavaju izvedbu specifičnih dijelova koda tek kada ih korisnik odluči koristiti, bez potrebe za poznavanjem točnog imena funkcije koju je potrebno pokrenuti (što je generalno način korištenja programa bez korisničkog sučelja). [17]

3.3.3. Unity Engine i Pixyz Unity Plugin

Za platformu za razvoj videoigara koja ima sposobnost integracija s VR uređajima, postojale su dvije opcije: Unreal Engine i Unity Engine. Za ovaj rad odabrana je platforma Unity Engine zbog dva razloga: mogućnost pisanja skripti u C# programskom jeziku (za razliku od C++ pristupnom u Unreal Enginu) te smanjeno opterećenje računalnog hardwarea prilikom prikaza modela u stvarnom vremenu unutar VR okruženja.

3.3.3.1. Unity Engine

Unity Engine je platforma za razvoj video igara tvrtke Unity Technologies. Sama platforma ima i mogućnost izrade 2D video igara, zvan Unity2D, no ta opcija ne podržava rad s 3D modelima te se neće obrađivati u okviru ovoga rada. Unity ima mnogo različitih verzija te će se za potrebe ovoga rada koristiti verzija 2020.3.25f1. Između novijih verzija ne postoje velike razlike te je jedino važno izabrati verziju koja podržava rad s VR sustavima, što je bilo koja verzija Unityja napravljena poslije ciklusa verzija 2016.X [18].

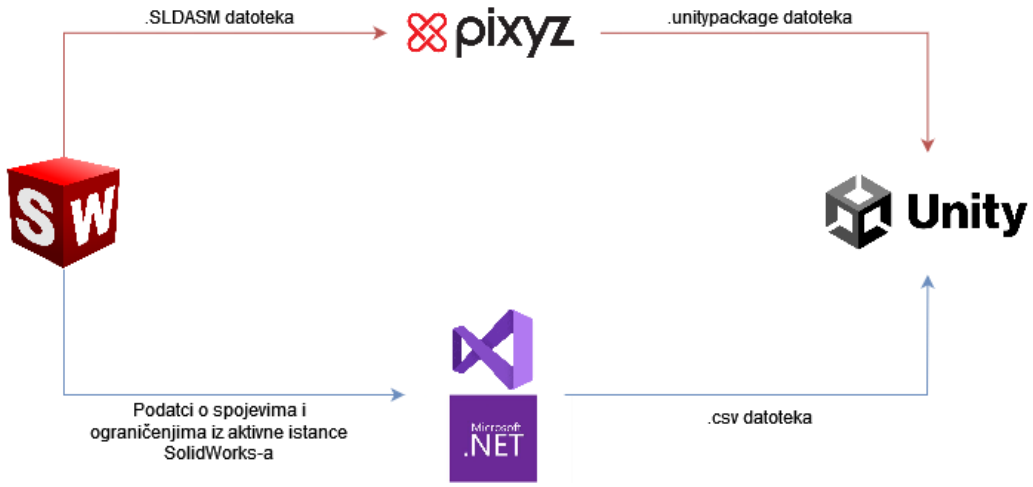
Unity nema mogućnost modifikacije oblika uvedenih modela, te modeli moraju biti mnogokutni kako bi se omogućio rad s njima. Kao što je već napomenuto, glavna značajka Unity platforme je mogućnost definiranja funkcionalnosti modela pomoću programskih skripti. Zahvaljujući tim skriptama, moguće je simulirati ponašanje spojeva i ograničenja u modelu.

3.3.3.2. Pixyz Unity Plugin

Pixyz je vanjski prevoditelj CAD modela koji razvija tvrtka Pixyz Software. Pixyz preko Pixyz Unity Plugina omogućuje integraciju prevoditelja u Unity te omogućuje prijevod izvornih CAD formata u .unitypackage format. Ovaj format unutar Unityja ima oblik mape koja sadrži sve datoteke potrebne za prikaz modela, od generirane mnogokutne mreže, tekstura do skripti koje sadrže informacije o autoru, vremenu izrade i izvoza, itd. Ove informacije nazivaju se *metadata* te su one čisto informativne prirode i ne utječu na ponašanje modela. Pixyz uz oblik

također prenosi i potpunu hijerarhiju dijelova modela, što pokriva dva od tri kriterija definiranih za uspješni prijenos modela.

Pixyz, zahvaljujući licencama raznih proizvođača CAD sustava, interpretira podatke o modelu direktno iz izvornog formata CAD sustava tog modela, koji je u okviru ovoga rada SLDASM format. Kako je objašnjeno u 2.3.2, ovakav prijenos naziva se direktnim prijenosom pomoću vanjskog prevoditelja, te se tim prijenosom prenosi najveći broj podataka između CAD sustava i drugih programa. Nakon što su definirani svi programi koji se koriste u radu, moguće je u kvalitativni tok podataka sa slike 10. uvesti programe kako bi se pokazao tok podataka između programa, prikazano na slici 13.



Slika 13. Dijagram prijenosa modela nakon definiranja programa potrebnih za proces

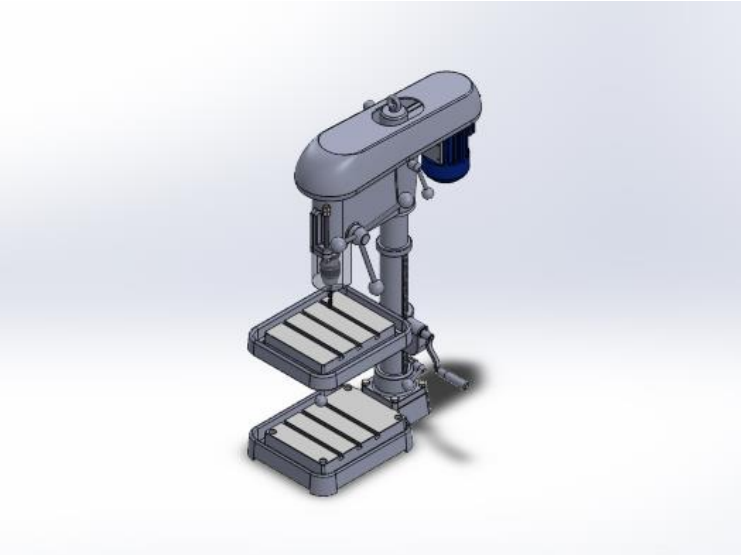
3.4. Opis odabranih modela za prijenos

Za prijenos koristit će se tri CAD modela sklopova izrađenih u SolidWorks-u i preuzetih s web-stranice GrabCad. Modeli su izabrani po količini dijelova te ograničenja i spojeva unutar sklopova. Uz to, glavni zahtjev modela bio je da su spremljeni u .SLDASM formatu, što je izvorni format CAD sustava SolidWorks. Također, modeli nisu modificirani prije prijenosa u Unity te je prije prijenosa utvrđena njihova funkcionalnost.

Svaki model ima barem jedan ključni spoj. Ključni spoj definiran je kao spoj ili ograničenje u modelu koji izvršava smislenu mehaničku funkciju. Na primjer, stupna bušilica ima dva ključna spoja u ručici koja se nalazi blizu podnožja modela. Okretanjem ove ručice u SolidWorks-u, pokreće se zupčani spoj koji zatim pokreće spoj zupčanika i zupčaste letve na koju je spojena podloga za obradak koja se onda translatira po Z osi modela.

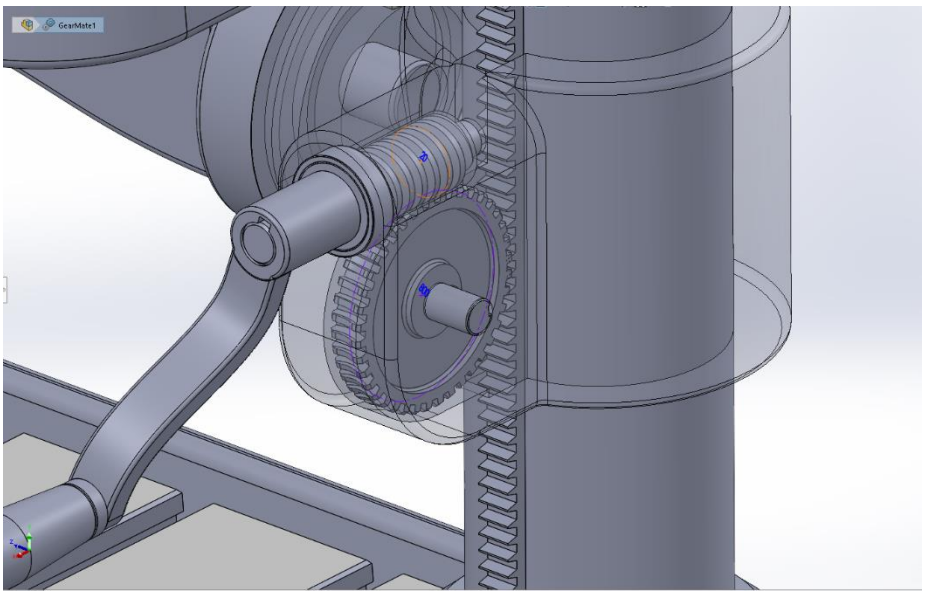
3.4.1. Model stupne bušilice

Najkompleksniji model u okviru ovoga rada bit će model stupne bušilice. Model se sastoji od 154 komponente i 239 različitih ograničenja i spojeva. [19]



Slika 14. Model stupne bušilice

Prvi ključni spoj modela je zupčani spoj vretena spojenog s ručicom i zupčanika spojenog sa zupčastom letvom. Drugi ključni spoj je spoj zupčanika i zupčaste letve. Okretom ručice u smjeru kazaljke na satu, vreteno pomiče zupčanik te podiže podlogu za obradak po zupčastoj letvi. Ovaj ključni spoj prikazan je na slici 15.



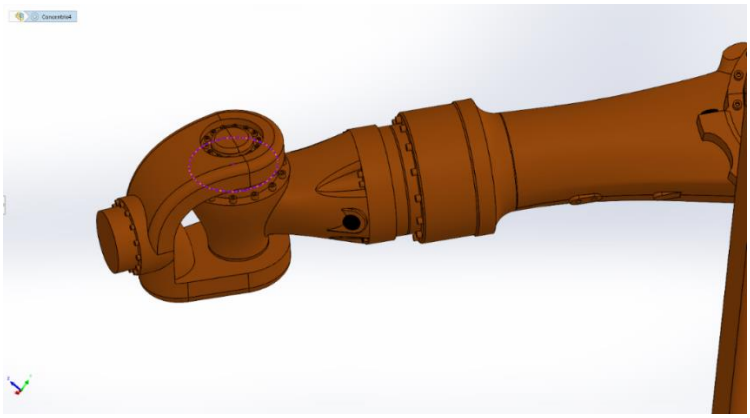
Slika 15. Ključni spoj vretena i zupčanika

3.4.2. Model robotske ruke Kuka KR16

Sljedeći model u okviru ovoga rada bit će model robotske ruke Kuka KR16. Model se sastoji od 12 komponenata i 32 različitih ograničenja i spojeva. [20]

Slika 16. Model robotske ruke Kuka KR16

Model ima 3 ključna spoja. Prvi se nalazi u zglobu prvog i drugog članka robotske ruke, i omogućuje rotaciju robotske ruke od drugog članka na dalje oko y osi. Drugi spoj nalazi se u zglobu drugog i trećeg članka robotske ruke, i omogućuje rotaciju robotske ruke od trećeg članka na dalje oko y osi. Napokon, zadnji funkcionalan spoj nalazi se u zglobu trećeg članka robotske ruke i adapteru za hvataljku na kraju robotske ruke. Adapter se može okretati u rasponu od 90 stupnjeva u obje strane oko osi koja prolazi kroz sredinu adaptera. Ovaj ključni spoj prikazan je na slici 17.



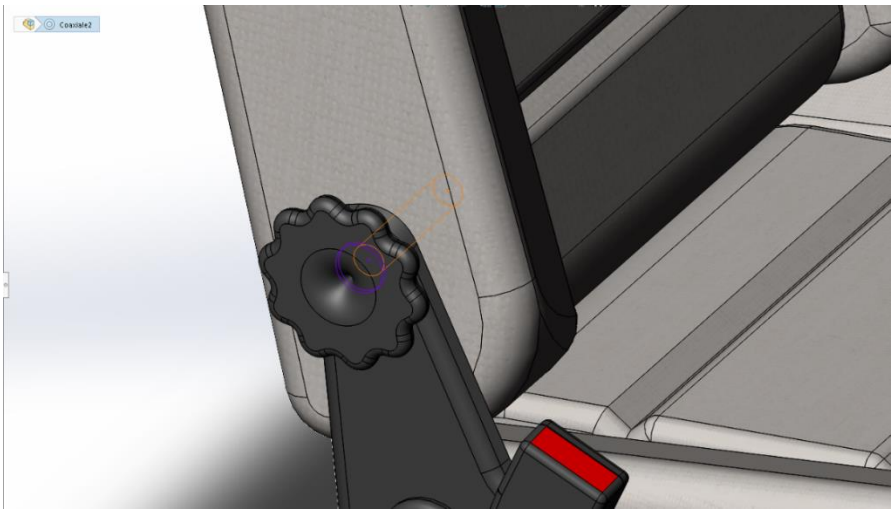
Slika 17. Ključni spoj trećeg zgloba robotske ruke i adaptera za hvataljku

3.4.3. Model automobilskog sjedala

Najjednostavniji model u ovome radu bit će model automobilskog sjedala. Model se sastoji od 7 komponenti i 20 različitih ograničenja i spojeva. [21]

Slika 18. Model automobilskog sjedala

Jedini ključni spoj u modelu je spoj kotačića s desne strane sjedala sa sjedalom. Kotačić se može okretati u oba smjera, te ne postoji ograničenje koliko se puta kotačić može okrenuti i koliko stupnjeva. Ovaj ključni spoj prikazan je na slici 19.



Slika 19. Ključni spoj kotačića i sjedala

4. IZRADA UNITY SKRIPTI I APLIKACIJE

U ovome poglavlju, potrebno je napraviti pregled rada Unity platforme za razvoj videoigara, i na temelju toga definirati kako je potrebno izraditi skripte za prijenos unutar Unityja. Nakon što su definirane skripte, potrebno je definirati rad aplikacije za prijenos spojeva i ograničenja pomoću API-ja.

4.1. Rad s modelima u Unityu

Platforme za razvoj video igara poput Unityja imaju puno drugačiji princip rada s modelima od CAD sustava. Mnogokutni modeli nemaju mogućnost prikaza referentne geometrije modela, pa tako ni Unity nema mogućnost rada s geometrijom. Zato je prije pisanja skripti potrebno prvo napraviti pregled Unity-evog načina rada s modelima.

4.1.1. Modeli unutar Unityja

Kako je već naglašeno, svaki model unutar Unityja je mnogokutni model. Kako mnogokutni modeli stvaraju puno manje opterećenje na rad računala, jasno je da je ovaj pristup idealan za izradu video igara i animacija. No, za CAD modele, nedostatak referentne geometrije i parametara modela stvara izazov prilikom rada s tim modelima u Unityju. Spojevi i ograničenja u sklopnim CAD modelima koriste referentnu geometriju dijelova, tako da je zbog nedostatka iste potrebno sagledati koje su mogućnosti rada s modelima unutar Unityja kako bi se definirao način prijenosa spojeva i ograničenja.

Unutar Unityja modeli se, ako nemaju definirano ograničenje, mogu gibati sa 6 stupnjeva slobode. Gibanje modela može biti po globalnom i lokalnom koordinatnom sustavu. Globalni koordinatni sustav unutar Unity projekta je nepromjenjiv, dok lokalni koordinatni sustav ovisi o tome gdje se u hijerarhiji nalazi dio. Ishodišta lokalnih koordinatnih sustava sklopova i podsklopova definirani su težištem lokalnih koordinatnih sustava svih dijelova i podsklopova kojima su nadređeni, dok su ishodišta koordinatnih sustava dijelova jednaka ishodištima koordinatnih sustava dijelova u CAD sustavu. Ako je pozicija dijela ili podsklopa u sklopu nepromijenjena od trenutka uvoza tog sklopa, onda će sve lokalne koordinate dijela ili podsklopa biti 0. Gibanjem nekog sklopa gibaju se i svi ostali dijelovi tog sklopa te se mijenja samo njihova pozicija u odnosu na globalni koordinatni sustav, ne lokalni.

Unity također ima integrirani sustav gravitacije. Gravitacija u Unityju je temeljni dio fizikalnog sustava koji omogućava simulaciju realističnih kretanja objekata u virtualnom

prostoru. Unutar Unityja, gravitacija djeluje kao sila koja povlači objekte „prema dolje“, odnosno prema smjeru definiranom unutar globalnog koordinatnog sustava. Globalni koordinatni sustav unutar Unityja je nepromjenjiv, pa je i smjer gravitacije uvijek određen prema negativnom smjeru osi Y.

Kako bi gravitacija djelovala na model, potrebnu je na nju primijeniti komponentu *Rigidbody*. Kada se na neki model unutar Unityja primijeni *Rigidbody* komponenta, taj model postaje podložan fizikalnim silama poput gravitacije. Bez te komponente, modeli u Unity-u neće reagirati na gravitacijske sile. Moguće je gravitaciju isključiti na razini pojedinog modela isključivanjem opcije "Use Gravity" unutar *Rigidbody* komponente, čime se omogućava slobodno kretanje objekta bez utjecaja ove sile. *Rigidbody* komponentna također ima mogućnost isključivanja opcije „Is Kinematic“. Isključivanjem ove opcije, model će se isključiti iz fizikalnog sustava unutar Unityja, te na njega neće djelovati nikakve fizikalne sile.

Unityjev fizikalni sustav također pruža opcije za ograničenje gibanja modela unutar tog sustava. Najsvestranija od ovih opcija su zglobovi.

4.1.2. Zglobovi unutar Unityja

Zglobovi omogućuju povezivanje dva modela kako bi simulirali različite vrste mehaničkih veza. Na taj način definiraju kako će se objekti gibati u odnosu jedni na druge. Svaki zglob ima specifične parametre koji kontroliraju gibanje objekata, poput rotacije i translacije, te dopuštene stupnjeve slobode.

Postoji nekoliko vrsta zglobova unutar Unityja, od kojih su najčešći *Hinge Joint* i *Spring Joint*. *Hinge Joint* simulira ponašanje zgloba poput šarke, dopuštajući rotaciju oko jedne osi, dok *Spring Joint* omogućava elastično povezivanje objekata, simulirajući oprugu koja povlači ili gura povezane objekte. Svi zglobovi imaju nekoliko opcija koje korisnik može promijeniti da bolje definira njihovo ponašanje, od kojih je najvažnija opcija *anchor*.

Unutar Unityjevih zglobova, *anchor* je ključni parametar koji definira ishodište koordinatnog sustava unutar kojeg zglob funkcionira. Konkretno, *anchor* predstavlja poziciju u prostoru gdje su dva modela povezana zglobom, a ta se pozicija nalazi u lokalnom koordinatnom sustavu modela koji sadrži komponentu zgloba. Ona određuje osi rotacije ili translacije modeli unutar zgloba, ovisno o vrsti zgloba koji se koristi. Po zadanim postavkama, *anchor* se nalazi u ishodištu lokalnog koordinatnog sustava modela, no korisnik ga može promijeniti. Pravilnim postavljanjem *anchor* parametra, može se precizno definirati kako će se

objekti kretati ili rotirati u odnosu na spoj, što je ključno za postizanje realističnog ponašanja u igri ili simulaciji.

No, „najsvestraniji“ zglob unutar Unityja je konfigurabilan zglob (*Configurable Joint*). Ova vrsta zgloba omogućava potpunu kontrolu nad svim stupnjevima slobode, što znači da se može precizno definirati kako će se model gibati u odnosu na drugi model. *Configurable Joint* pruža opcije za ograničavanje ili dopuštanje gibanja po svim trima osima, kao i za podešavanje rotacijskih stupnjeva slobode.

Ovaj zglob je posebno koristan kada se radi na projektima koji zahtijevaju vrlo specifične fizikalne simulacije, poput CAD modela. Na primjer, moguće je ograničiti kretanje po X osi dok se ostavi sloboda rotacije oko Y i Z osi, čime se omogućuje stvaranje preciznih kinematskih veza. Pomoću konfigurabilnog zgloba, moguće je simulirati gotovo svaki tip mehaničkog spoja ili ograničenja.

Zbog toga, skripte koje će se razvijati u okviru ovoga rada moraju ograničenja i spojeve iz SolidWorksa pretvoriti u konfigurabilne zglobove s konfiguracijom koja odgovara spoju ili ograničenju koje se prenosi.

4.2. Izrada Unity skripti

Unity skripte sastojat će se od dvije glavne skripte: skripta za interpretaciju tekstualne datoteke generirane alatom za prijenos i skripte za primjenu zglobova na dijelove. Također je potrebno napraviti treću pomoćnu skriptu u kojoj će se sadržavati konfiguracije zglobova koji će primjenjivati na modele. Ideja je da se nakon uvoza modela pokrene skripta za interpretaciju datoteke koja će automatski na dijelove koji su dio spoja ili ograničenja primijeniti konfigurabilni zglob s konfiguracijom koja odgovara tom spoju ili ograničenju.

4.2.1. Izrada Unity skripte za automatsku interpretaciju datoteke

Unity skripta za automatsku interpretaciju datoteke mora biti u mogućnosti interpretirati podatke o spojevima i ograničenjima iz CSV datoteke te ih primijeniti na odgovarajuće dijelove sklopa. Najvažniji zahtjev ove skripte je to da se interpretacija i raspodjela podataka mora odviti prije simulacije unutar Unityja. Naime, Unity ima dvije glavne okoline za prikaz modela: *Editor View* i *Game View*. *Editor* je okolina u kojoj je potrebno primijeniti svu logiku ponašanja modela prije nego li se pokrene *Game View* u kojem se na model primjenjuje fizikalni sustav u stvarnom vremenu. Također, sve izmjene koje su primijenjene na model u *Game Viewu* u

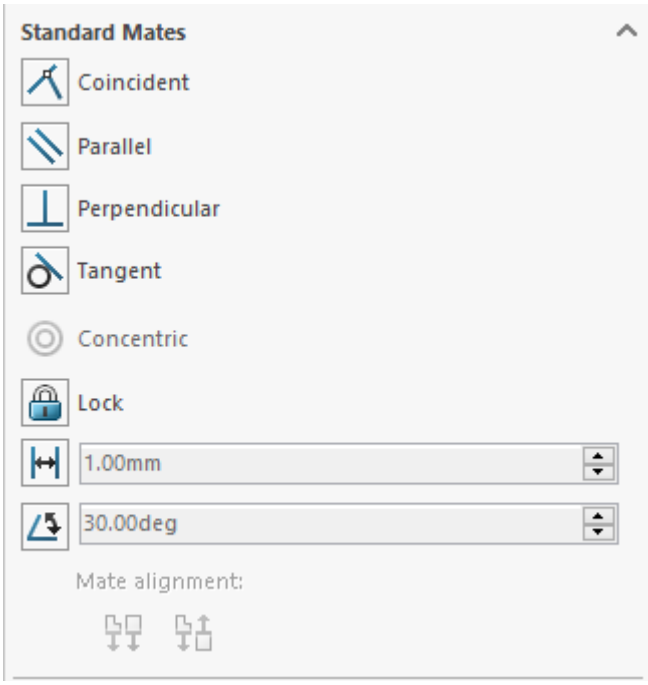
stvarnom vremenu nisu trajno spremljene u projekt, već samo imaju utjecaj na trenutnu instancu okoline (dok se ne zaustavi). Razlog tome je što korisnik ponekad želi provjeriti ponašanje modela u stvarnom vremenu bez da narušava već spremljene postavke u projektu. Zato je važno da skripta bude napisana za *Editor* okolinu, pošto će se primjena ograničenja i spojeva morati odvijati prije pokretanja *Game Viewa* kako bi se mogla trajno spremiti u projekt.

4.2.2. Izrada Unity skripti za spojeve i ograničenja

Kako bi bilo moguće napisati Unity skripte za ograničenja, potrebno je analizirati kako se modeli s tim ograničenjima ponašaju; koje su im kretnje dopuštene i kako utječu na dijelove s kojima su povezani. Na model će se po spoju ili ograničenju primjenjivati po jedan konfigurabilni zglob. Kako Unity nema mogućnost prepoznavanja referentne geometrije u modelima, jer je mnogokutni modeli ni nemaju, potrebno je definirati koja će os u lokalnom koordinatnom sustavu modela biti glavna os primijenjenog zgloba. Glavna os modela definirana je kao os u odnosu na koju je potrebno primijeniti ograničenja gibanja zgloba. Zadana postavka Unityja je da je glavna os modela definirana u smjeru X osi lokalnog koordinatnog sustava modela, no to korisnik može promijeniti po želji. Na primjer, ako se na model u SolidWorksu primijeni koncentrično ograničenje tako da se može rotirati oko i translatirati po Z osi, a lokalna X os tog modela je okomita na Z os, onda je potrebno unutar Unityja odabrati Z os kao glavnu os. Zbog toga, za pravilnu primjenu zglobova bit će potrebno pitati korisnika za glavnu os svakog spoja ili ograničenja.

4.2.2.1. Ograničenja (Standard Mates)

Ograničenja unutar SolidWorksa definirana su već poznatim matematičkim ograničenjima u geometriji.



Slika 20. Ograničenja (Standard Mates) u SolidWorksu

4.1.2.1.1. Koincidentnost (Coincident)

Kada se dva geometrijska oblika dodiruju u nekom geometrijskom elementu, bila to linija, točka ili ravnina, smatra se da su koincidentni. U CAD sustavu, koincidentnost ograničava gibanje modela ovisno o tome koji su geometrijski elementi spojeni. Ako je koincidentnost u točci, onda se model može rotirati po bilo kojoj osi, ali ne i translatirati. Ako je koincidentnost u liniji, ovisno o vrsti linije, model se može samo translatirati po jednoj osi (u slučaju pravca), rotirati oko jedne osi (u slučaju kružnice) ili uopće ne kretati (u slučaju krivulje). Napokon, ako je koincidentnost u ravnini, onda se model može rotirati oko osi na kojoj leži normala te ravnine i slobodno translatirati po toj ravnini.

Kao što je vidljivo, za koincidentnost je potrebno uvažiti sve navedene slučajeve prilikom pisanja skripte. Kako Unity nema mogućnost prepoznavanja referentne geometrije modela, skripta mora sadržavati kod za svaki od navedenih slučaja te, uz odabir glavne osi zgloba, korisniku ponuditi izbor za koji je geometrijski element potrebno primijeniti

ograničenje. Zbog tih ograničenja Unityja potrebno je također naglasiti kako se slučaj koincidentnosti sa zakrivljenom površinom neće razmatrati u okviru ovoga rada.

4.1.2.1.2. Paralelnost (Parallel)

Kada se dvije ravnine, dva pravca ili pravac i ravnina nalaze u istom prostoru i nemaju presječnih točaka, one se smatraju paralelnima. U CAD sustavu, paralelnost ograničava model tako da prilikom gibanja ravnina ili pravac dijela mora biti paralelna s referentnom ravninom ili pravcem. Ovo znači da se model može slobodno translatirati po osi koja je paralelna s ravninom ili pravcem, ali ne može rotirati oko osi koja bi prekinula tu paralelnost.

Kako bi se ovo ponašanje simuliralo u Unityju, potrebno je omogućiti translaciju modela isključivo po onim osima koje su paralelne s referentnim elementom, dok se rotacija mora dopustiti samo oko osi koja je okomita na oba pravca ili ravnine.

4.1.2.1.3. Okomitost (Perpendicular)

Kada su dva pravca ili ravnine međusobno postavljene pod pravim kutom (90°), one se smatraju okomitima. U CAD sustavu, okomitost ograničava kretanje modela tako da se ravnine ili pravci zadržavaju u okomitom položaju. Ovo znači da se model može rotirati ili translatirati samo po osima koje ne narušavaju okomitost između povezanih elemenata.

Kako bi se ovo ponašanje simuliralo u Unityju, potrebno je ograničiti rotaciju modela na način da se zadrži pravi kut između zadanih elemenata, što znači da je rotacija dozvoljena samo oko okomitih osi dijelova ograničenja. Također, translacija će biti moguća samo po glavnoj osi i osi unutar lokalnog koordinatnog sustava koja je okomita na nju.

4.1.2.1.4. Tangencijalnost (Tangent)

Kada se krivulja dodiruje u jednoj točci s ravninom ili pravcem tako da je normala krivulje u toj točci okomita na ravninu ili pravac koju dodiruje, onda se smatra da je ta ravnina ili pravac tangencijalan na tu krivulju. U CAD sustavu, tangencijalnost ograničava slobodnu translaciju modela po osi koja se ne nalazi u ravnini krivulje na koju je tangencijalan, te translaciju (i automatski rotaciju u slučaju crte) u ravnini krivulje dok god je element tangencijalan na nju.

Zbog potrebe za poznavanjem geometrije prilikom korištenja tangencijalnog ograničenja, očito je da je izvedba ovog ograničenja veoma kompliciranа u Unityju zbog nesposobnosti platforme za raspoznavanje geometrije. Kako bi se pokrio svaki mogući slučaj tangencijalnosti, bilo bi potrebno pronaći sve vrhove unutar mnogokutne mreže modela koji sadrži krivulju na koju se primjenjuje tangencijalnost, zatim prepoznati i izolirati vrhove koji tu krivulju definiraju, te ih definirati kao dio navigacijske mreže. Navigacijska mreža je mreža vrhova u modelu koja nastaje međusobnim povezivanjem određenog skupa vrhova, te tako stvara putanju po kojoj se modeli mogu gibati.[22]

Izvedba ove metode je veoma kompleksna i zahtjeva ručnu obradu veliku količine podataka kako bi se pronašli vrhovi koji definiraju krivulju na koju se primjenjuje tangencijalnost. Stoga se u okviru ovoga rada za tangencijalnost neće izraditi skripta koja pokriva slučaj svake krivulje. Umjesto toga, izvest će se jednostavnija implementacija, u kojoj se uvodi pretpostavka da je svaka krivulja na koju je primijenjena tangencijalnost kružnica kroz čiji centar prolazi jedna od osi lokalnog koordinatnog sustava. Zahvaljujući ovoj pretpostavci, moguće je definirati koja će gibanja biti potrebno omogućiti modelu s tangencijalnim ograničenjem. Modeli s tangencijalnim ograničenjem mogu se translatirati po osi koja je okomita na ravninu u kojoj se nalazi kružnica na koju se tangencijalnost primjenjuje, te se tangencijalni elementi mogu rotirati oko te osi na udaljenosti jednakoj radijusu kružnice.

4.1.2.1.5. Koncentričnost (Concentric)

Kada dvije kružnice dijele središte u ravnini na koju su njihove normale okomite, smatra se da su koncentrične. U CAD sustavu, koncentričnost ograničava gibanje modela na translaciju i rotaciju oko osi na kojoj leže koncentrični elementi.

Kako bi se ovo ponašanje simuliralo u Unityju, potrebno je gibanje po lokalnim osima ograničiti na translaciju i rotaciju oko osi na kojoj leže koncentrični elementi.

4.1.2.1.6. Fiksiranost (Lock)

Kada je model ili njegov dio učvršćen u određenom položaju bez mogućnosti translacije ili rotacije, smatra se da je fiksiran. U CAD sustavu, fiksiranost potpuno ograničava sve oblike gibanja modela, držeći ga na točno određenoj poziciji u prostoru.

Kako bi se ovo ponašanje simuliralo u Unityju, potrebno je potpuno onemogućiti sve vrste gibanja modela. Model će se jedino moći pomicati ako se pomiče sklop kojega je on dio.

4.1.2.1.6. Udaljenost

Kada su dva geometrijska elementa odvojena točno definiranom udaljenosti, smatra se da između njih postoji ograničenje udaljenosti. U CAD sustavu, ograničenje udaljenosti određuje koliko se daleko ili blizu dva elementa mogu kretati jedan prema drugome, bez narušavanja zadane udaljenosti.

Za simulaciju ograničenja udaljenosti u Unityju, potrebno je omogućiti translaciju modela isključivo unutar okvira zadane udaljenosti. Skripta će provjeravati udaljenost između povezanih elemenata i automatski zaustaviti translaciju čim se dostigne minimalna ili maksimalna udaljenost. Korisnik će moći definirati udaljenost koju skripta treba održavati.

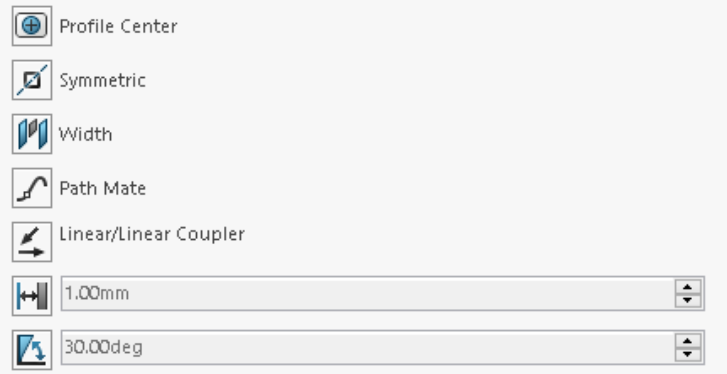
4.1.2.1.7. Kut

Kada su dva geometrijska elementa u odnosu pod definiranim kutom, smatra se da između njih postoji kutno ograničenje. U CAD sustavu, kutno ograničenje drži dva elementa u stalnom kutnom odnosu, dozvoljavajući rotaciju samo oko osi koja ne mijenja taj kut.

Za simulaciju kutnog ograničenja u Unityju, potrebno je ograničiti rotaciju modela na način da se održava zadani kut između elemenata. Skripta će korisniku omogućiti definiranje željenog kuta, te će provjeravati rotaciju kako bi osigurala da se taj kut ne promijeni. Kretanje modela će biti ograničeno na osi koje ne mijenjaju definiran kut.

4.2.2.2. Skupovi ograničenja (Advanced Mates)

Ovo je kategorija ograničenja koja se razlikuju od običnih ograničenja po tome što ne proizlaze iz poznatih geometrijskih definicija. Još uvijek se definiraju preko referentne geometrije i svaki se sastoji od više matematičkih ograničenja, te će se u sklopu ovoga rada tretirati kao ograničenja. Kako su *Limit Distance* i *Limit Angle* ograničenja slična ograničenjima udaljenosti i kuta, tretirat će se kao takve u okviru ovoga rada te će se za njih primjenjivati zglobovi konfiguracije udaljenosti i kuta.



Slika 21. Skupovi ograničenja (Advanced Mates) u SolidWorksu

4.1.2.2.1. Centar profila (Profile Center)

Ograničenje centar profila omogućava postavljanje centra pravokutnog ili kružnog profila jednog modela u odnos na drugi element u sklopu. U CAD sustavu, ovo ograničenje postavlja centar profila tako da bude poravnat s centrom ili osi drugog geometrijskog elementa, ograničavajući gibanje modela kako bi zadržao tu „centriranost".

Kako bi se ovo ponašanje simuliralo u Unityju, potrebno je osigurati da se centar profila modela zadrži u zadanoj poziciji u odnosu na drugi element. Kako jedan od spojenih dijelova mora prenositi gibanje, u Unityju će se zglob definirati tako da se na jedan od dijelova u zglobu prenosi svo gibanje drugog dijela spoja.

4.1.2.2.2. Simetrija (Symmetry)

Ograničenje simetrije koristi se kada je potrebno da dva ili više elemenata budu simetrično postavljeni u odnosu na određenu referentnu ravninu ili crtu. U CAD sustavu, ovo ograničenje osigurava da svi elementi ostanu simetrični u odnosu na zadanu referentnu točku, ravninu ili os, bez obzira na moguće translacije ili rotacije modela.

Za simulaciju simetrije u Unityju, potrebno je osigurati da su svi elementi modela poravnati simetrično u odnosu na referentnu točku, ravninu ili os. Kako je ovo ponašanje vrlo teško replicirati pomoću skripte zbog nemogućnosti korištenja referentne geometrije u Unityju, skripta za simetriju neće biti izrađena, već će korisnik morati sam postaviti simetriju ograničenjem udaljenosti dijelova koji bi trebali biti simetrični, slično ograničenju udaljenosti.

4.1.2.2.3. Širina (Width)

Ograničenje širine koristi se kada dva elementa moraju biti smještena unutar određenih granica širine nekog drugog elementa. U CAD sustavu, ovo ograničenje osigurava da se elementi modela mogu pomicati samo unutar zadanih granica širine, bez mogućnosti izlaska iz tih granica.

Za simulaciju ograničenja širine u Unityju, potrebno je ograničiti translaciju modela na način da se kreće isključivo unutar zadanih granica širine. Kako ovaj skup ograničenja zahtjeva sparivanje više od dva dijela, umjesto pisanje zasebne skripte na dio će biti primijenjena skripta ograničenja udaljenosti te će se korisniku ponuditi odabir udaljenosti.

4.1.2.2.4. Putanja (Path Mate)

Ograničenje putanje koristi se kada je potrebno da se element modela kreće po zadanoj putanji. U CAD sustavu, ovo ograničenje osigurava da se model kreće isključivo duž unaprijed definirane putanje, uz mogućnost rotacije ovisno o obliku putanje.

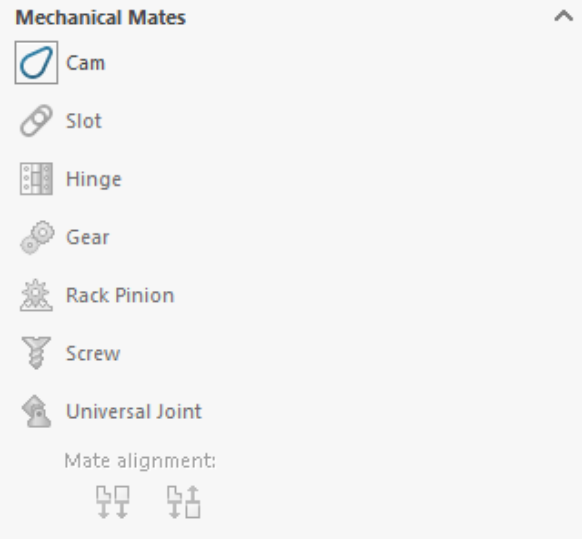
Kako se u CAD sustavima putanje definiraju geometrijski, a Unity nema mogućnost prepoznavanja te geometrije, ovaj skup ograničenja neće se obrađivati unutar ovoga rada.

4.1.2.2.5. Linearni pomak (Linear Coupler)

Kada je potrebno translatirati dva dijela u SolidWorksu različitim brzinama, na dijelove se primjenjuje ograničenje linearnog pomaka. U CAD sustavu, ovo ograničenje omogućava translaciju dva dijela u smjeru jedne osi, ali različitim brzinama. Ovo ograničenje se koristi kada je potrebno simulirati rad kao što je klizanje dijelova unutar vodilica.

U Unityju, primjena linearnog pomaka na model podrazumijeva izračunavanje relativnih pozicija i brzina između objekata te primjenu tih izračuna na modele unutar Unity okruženja. Kako ovo ponašanje nije moguće simulirati pomoću zglobova već je potrebno simulirati pomoću zasebne skripte, linearni pomak se neće obrađivati u sklopu ovoga rada.

4.2.2.3. Mehanički spojevi (Mechanical mates)



Slika 22. Mehanički spojevi (Mechanical Mates) u SolidWorksu

Što se tiče mehaničkih spojeva, u Unity se automatski mogu prenijeti samo dva: spoj šarke i spoj univerzalnog zgloba. Razlog tome je već ugrađena funkcionalnost unutar Unityja za ove spojeve, a to su *Hinge Joint* i *Character Joint*. Ovi zglobovi unutar Unityja simuliraju rad šarke i univerzalnog zgloba, te se zbog toga mogu automatski prenijeti iz CAD sustava u Unity.

Za ostale mehaničke spojeve, automatski prijenos nije izvediv u okviru ovoga rada. Ti spojevi zahtijevaju zapis matematičkih skripti koje ovise o geometriji dijelova, a pošto Unity ne prepoznaje referentnu geometriju modela, za svaki takav spoj bilo bi potrebno iznova modificirati postojeće skripte napisane za te spojeve.

Na primjer, spoj zupčanika unutar SolidWorksa automatski računa prijenosni omjer zupčanika u oba smjera na temelju njihovih diobenih kružnica. Kako bi se prijenosni omjer izračunao u Unityju, korisnik bi trebao prenijeti model u Unity, ručno izmjeriti diobene kružnice zupčanika, izračunati prijenosni omjer zupčanika i napisati skriptu koja pokreće zupčanike u skladu s tim prijenosnim omjerom. Dok je moguće unaprijed pripremiti skriptu za prijenos u koju će se upisati podatci, ili saznati prijenosni omjer iz CAD sustava te ga upisati u Unity, za modele koji imaju veliku količinu zupčanih spojeva bi taj proces bio nepotrebno naporan.

Zbog toga, sve skripte za mehaničke spojeve osim šarke i univerzalnog zgloba bit će ručno dodane po potrebi, pošto postoji previše varijabli za njihovu automatizaciju.

Kako bi uopće bilo moguće rasporediti sva ova ograničenja i spojeve po dijelovima, potrebno je je znati koji dijelovi su u kakvom odnosu. Za to je potrebno razviti aplikaciju koja će iz SolidWorks modela zapisati sva ograničenja i spojeve u modelu i zatim te podatke spremiti u vanjsku datoteku koja se može uvesti i čitati u Unityju.

4.3. Izrada aplikacije

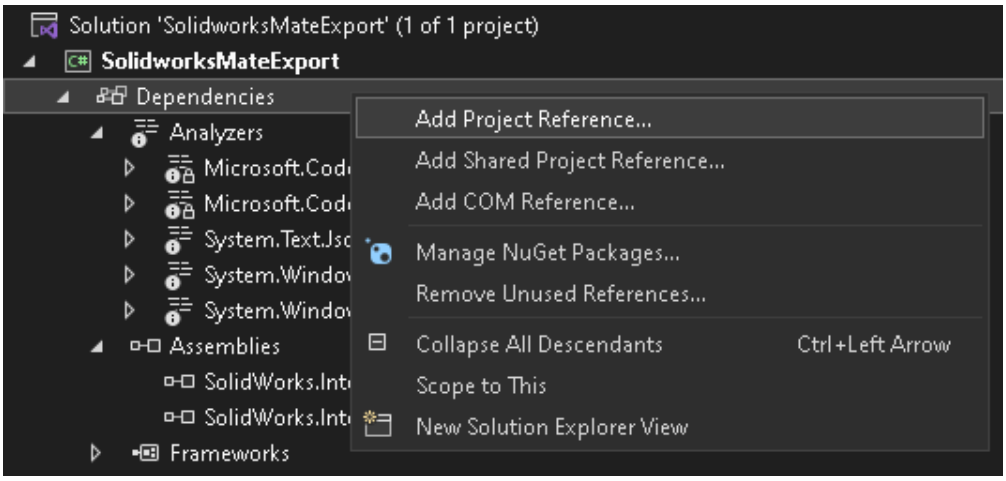
Po toku podataka definiranom u 3.2, moguće je postaviti zahtjeve za izradu aplikacije za prijenos ograničenja i spojeva iz SolidWorksa u Unity.

Glavni zahtjev je zapis imena dva povezana dijela te ograničenje/spoj koji ih povezuje u CSV formatu. Aplikacija mora biti u mogućnosti prepoznati spoj između dva dijela unutar SolidWorks dokumenta, zapisati podatke o spoju (imena dijelova, ime ograničenja/spoja) te ponoviti postupak sve dok se nisu zapisali svi dijelovi koji su povezani ograničenjem ili spojem.

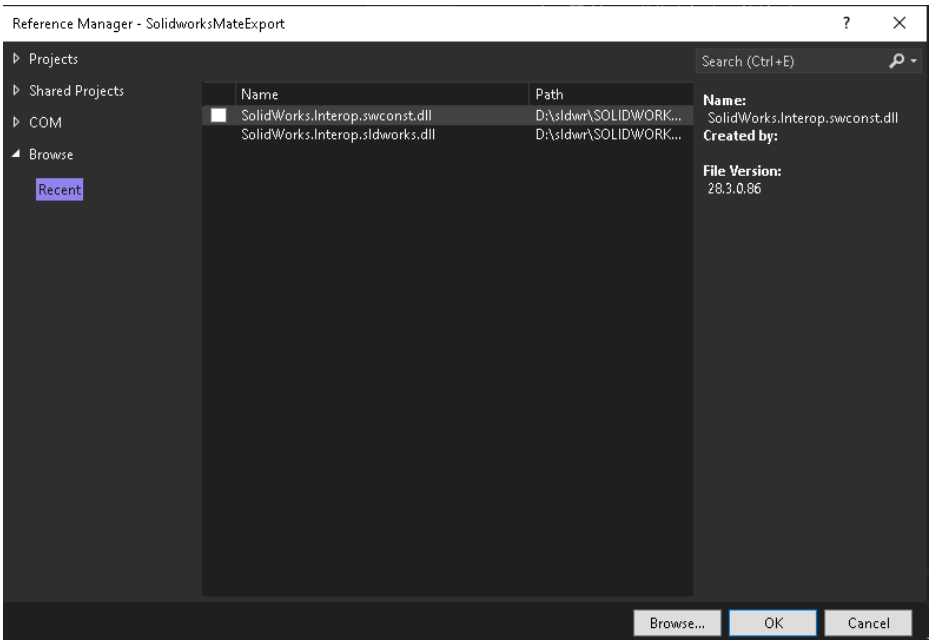
Sporedni zahtjev aplikacije je da može prepoznati postoji li aktivna SolidWorks instanca na računalu te, ako ne postoji, da pokrene novu instancu. Također je potrebno napraviti osnovni GUI (*Graphical User Interface*) kako bi korisniku bilo lakše izvoditi navedene funkcije te kako bi korisnik imao uvid u podatke koji se zapisuju i potvrdi njihovu točnost.

4.3.1. Programiranje aplikacije

Kako bi uopće bilo moguće pristupiti SolidWorks API-ju, potrebno je aplikaciji naznačiti gdje se nalazi SolidWorks API *interop assembly*. *Interop assemblies* su sklopovi u .NET frameworku koji spajaju razvojni kod s gotovim, te tako omogućuju razvojnome kodu da utječe na izvornu aplikaciju interop assemblya, u ovome slučaju SolidWorks [23]. Za potrebe ove aplikacije koristit će se jedan interop assembly, SolidWorks.Interop.sldworks.



Slika 23. Desnim klikom na izbor *Dependencies*, otvara se padajući izbornik te se odabire opcija *Add Project Reference…*



Slika 24. Nakon odabira željenih sklopova, Visual Studio se sada može pozvati na SolidWorks API za sve potrebne funkcije

5. KOD APLIKACIJE I UNITY SKRIPTI

U ovome poglavlju, objasnit će se čemu je namijenjen svaki dio koda za izradu aplikacije i Unity skripta. Također će biti objašnjeno kako djeluju na model i kako su međusobno povezani.

5.1. Kod aplikacije

Aplikacija se sastoji od 4 .cs datoteke. Nastavak .cs samo označava tekstualnu datoteku koja je pisana u C# programskom jeziku. Datoteke su:

- sldwrSingleton.cs – kod za pokretanje ili dohvat SolidWorks CAD alata
- getMates.cs – kod za dohvat i zapis svih ograničenja i spojeva unutar aktivnog SolidWorks modela
- Form1.cs – kod za povezivanje napisanih funkcija s korisničkim sučeljem
- Form1.Designer.cs – automatski generiran kod nastao prilikom uređivanja korisničkog sučelja integriranim dizajnerom za sučelja.

Različite komponente koda međusobno surađuju na sljedeći način: datoteka Form1.Designer.cs oblikuje korisničko sučelje aplikacije. Elementi aplikacije su gumbi „Get SolidWorks Instance“ i „Get mates“ te *textbox* element u kojem je moguće provjeriti rad aplikacije. Ove elemente s funkcijama sadržanima u sldwrSingleton.cs i getMates.cs datotekama povezuju funkcije sadržane u datoteci Form1.cs. Što sve sadržavaju ove funkcije i kako rade bit će objašnjeno u sljedećim poglavljima.

5.1.1. sldwrSingleton.cs

Ova datoteka sadržava klasu `sldwrSingleton`, koja sadrži jednu funkciju: `getApplication()`.

`getApplication()` je asinkrona funkcija koja provjerava postoji li aktivna instanca SolidWorksa, i ako ne postoji otvara novu. Nakon što je otvorena nova instanca, ili pronađena postojeća, funkcija vraća vrijednost `swApp`. `swApp` je definirana kao `SldWorks` tip varijable. `SldWorks` je tip varijable u `SolidWorks.Interop.sldworks` sklopu koja se preko operativnog sustava povezuje na aktivnu instancu SolidWorks CAD sustava.

5.1.2. getMates.cs

Ova datoteka sadržava jednu klasu `getMates` koja sadrži jednu funkciju: `mates(ModelDoc2 model)`.

`mates(ModelDoc2 model)` je asinkrona funkcija koja prima jedan parametar tipa `ModelDoc2`. `ModelDoc2` je tip varijable u `SolidWorks.Interop.sldworks` sklopu koja dohvaća trenutno otvoreni model u dohvaćenoj instanci SolidWorks CAD sustava. Funkcija prvo definira sve potrebne varijable koristeći tipove sadržane u `SolidWorks.Interop.sldworks`. Od tih tipova, najvažniji je tip `Mate2` koji se priključuje varijabli `swMate`. Nakon definiranja svih potrebnih varijabli, funkcija pokreće while petlju koja prolazi sve `Feature`-e prisutne u modelu. `Feature` u SolidWorks modelu označava značajke i komponente modela, koje mogu biti koordinatni sustav, ravnina, dijelovi, podsklopovi, itd. Petlja iterira po `Feature`-ima sve dok ne naiđe na `Feature` imena „`Mate Group`“. Ovaj `Feature` sadrži sve spojeve i ograničenja prisutne u modelu. Taj `Feature` prvo se sprema u varijablu `swMateFeat` i zaustavlja petlju. Zatim se u varijablu `swSubFeat` sprema funkcija interop sklopa `swMateFeat?.GetFirstSubFeature()`, koja dohvaća prvi spoj ili ograničenje u modelu. Zatim se otvara instanca `StreamWritera`, ugrađene .NET klase koja se koristi za zapisivanje teksta u zasebnu datoteku. Nakon otvaranja `StreamWritera`, pokreće se `while` petlja koja ponavlja sljedeći skup koraka: petlja prvo provjeri je li varijable `swSubFeat` prazna. Ako nije, varijabli `swMate` se dodjeljuje `(Mate2)swSubFeat.GetSpecificFeature2()` koja vraća varijablu tipa `IMate2`. Ako vraćeni tip nije `Mate2`, vraća vrijednost `null`. Zatim se pomoću `StreamWritera` zapisuju podatci sljedećim redom: prvo se zapisuje ime prvog dijela ili podsklopa na koji je primijenjen spoj ili ograničenje. Zatim se zapisuje ime drugi dijela ili podsklopa u spoju ili ograničenju. Napokon, zapisuje se ime spoja ili ograničenja i otvara se novi redak. Na kraju petlje varijabli `swSubFeat` dodjeljuje se sljedeći spoj ili ograničenje pozivom na funkciju `swSubFeat.GetNextSubFeature()`. Petlja se zaustavlja kada `swMate` bude vrijednosti `null`, što se dogodi kada je sljedeći `Feature` različit od `Mate2` tipa.

5.1.3. Form1.cs

Ova datoteka sadržava jednu klasu `Form1` koja sadrži tri funkcije: `Form1()`, `button1_Click(object sender, EventArgs e)`, `button2_Click(object sender, EventArgs e)`.

Funkcija `Form1()` inicijalizira instancu aplikacije, i povezuje korisničko sučelje s `Form1` klasom i svim funkcijama sadržanima u njoj.

Funkcije `button1_Click(object sender, EventArgs e)` i `button2_Click(object sender, EventArgs e)` povezuju gumbe korisničkog sučelja „Get Mates“ i „Get SolidWorks Application“ s klasama `getMates` i `sldwrSingleton`.

5.1.4. Korisničko sučelje aplikacije u Visual Studio dizajneru

Korisničko sučelje sastoji se od jednog textbox elementa i dva gumba. Gumb „Get Mates“ povezan je preko klase `Form1` iz datoteke Form1.cs s klasom `getMates` iz datoteke getMates.cs, i pokreće funkciju `mates(ModelDoc2 model)`. Gumb „Get SolidWorks Instance“ povezan je preko klase `Form1` iz datoteke Form1.cs s klasom `sldwrSingleton` iz datoteke sldwrSingleton.cs i pokreće funkciju `getApplication()`. Textbox element čisto je informativnog karaktera, i prikazuje informacije na kojem se spoju ili ograničenju trenutno nalazi petlja. Uz ime spoja ili ograničenja, prikazuje ime povezanih dijelova spoja te vrstu spoja označenu brojem.

```
1  base-2,rotary head-2,Concentric1
2  base-2,rotary head-2,Coincident1
3  rotary head-2,lower arm-2,Coincident2
4  rotary head-2,lower arm-2,Concentric2
5  lower arm-2,upper arm-2,Coincident3
6  lower arm-2,upper arm-2,Coincident4
7  upper arm-2,p2-2,Concentric3
8  upper arm-2,p2-2,Coincident5
9  p2-2,WRIST-2,Coincident6
10 p2-2,WRIST-2,Concentric4
11 rotary head-2,a1servo-3,Coincident8
12 rotary head-2,a1servo-3,Coincident9
13 rotary head-2,a1servo-3,Coincident10
14 rotary head-2,a1servo-4,Coincident11
15 rotary head-2,a1servo-4,Coincident12
16 rotary head-2,a1servo-4,Coincident13
17 a4a5a6 servo-4,a4a5a6 servo-5,Coincident14
18 a4a5a6 servo-5,a4a5a6 servo-6,Coincident15
19 a4a5a6 servo-4,a4a5a6 servo-5,Coincident17
20 upper arm-2,a4a5a6 servo-5,Coincident18
21 upper arm-2,a4a5a6 servo-4,Coincident19
22 upper arm-2,a4a5a6 servo-6,Coincident20
23 upper arm-2,a4a5a6 servo-5,Lock2
24 upper arm-2,a4a5a6 servo-4,Lock3
25 upper arm-2,a4a5a6 servo-6,Lock4
26 upper arm-2,a3servo-2,Coincident22
27 upper arm-2,a3servo-2,Coincident23
28 upper arm-2,a3servo-2,Coincident24
29 upper arm-2,a3servo-2,Parallel1
30 lower arm-2,rotary head-2,LimitAngle2
31 lower arm-2,upper arm-2,LimitAngle3
32 WRIST-2,p2-2,LimitAngle4
33
```

Slika 25. Sadržaj generirane CSV datoteke za robotsku ruku KUKA KR16

5.2. Kod Unity skripti

Za Unity razvijene su 3 skripte koje međusobno surađuju za primjenu spojeva i ograničenja na preneseni model. To su:

- CSVProcessor.cs – skripta koja učitava .csv datoteku i sprema podatke datoteke u listu kako bi je druge skripte mogle interpretirati
- CSVProcessorEditor.cs – skripta koja interpretira generiranu listu podataka i primjenjuje zglobove na model pomoću MateLibrary.cs skripte
- MateLibrary.cs – skripta u kojoj se nalaze konfiguracije zglobova za svaki prethodno definirani spoj ili ograničenje.

Skripte su povezane na sljedeći način: skripta CSVProcessor.cs primjenjuje se na sklop modela (najviša razina hijerarhije). Nakon što je CSVProcessor.cs primijenjen na glavni sklop, u mapu „Assets“ unutar Unity projekta se stavlja mates.csv datoteka generirana aplikacijom za prijenos spojeva i ograničenja. Zatim se pritiskom na gumb „Load CSV“ u kod CSVProcessora učitava ta datoteka, te se njeni podatci zapisuju u listu. Pritiskom na gumb „Apply CSV Data to Children“ lista generirana CSVProcessor skriptom poslana je u skriptu CSVProcessorEditor.cs, koja u suradnji sa skriptom MateLibrary.cs interpretira podatke iz liste i primjenjuje zglobove na dijelove modela.

Potrebno je također naglasiti razlog zašto je proces zapisa podataka iz CSV datoteke u listu i interpretacija te liste razdvojen na dvije skripte. Naime, Unity ima dvije vrste skripti koje se koriste u okviru ovoga rada, podijeljene po okružjima u kojima funkcioniraju: MonoBehaviour i Editor skripte. MonoBehaviour skripte mogu jedino biti priključene na aktivan element u sceni unutar Unityja, dok Editor skripte svoj kod izvršavaju tako da korisnik upravlja njima preko korisničkog sučelja Unity platforme. Razlog zašto se koriste obje vrste skripti je olakšavanje rada sa skriptama korisniku. Editor skripte ne mogu direktno referencirati element u okruženju, već je na korisniku da ručno referencira element na koji želi primijeniti kod skripte. S druge strane, MonoBehaviour skripte ne mogu rad svojih funkcija kontrolirati preko korisničkog sučelja osim ako su povezane na Editor skriptu, već svoje funkcije izvršavaju unutar Unity scene. Na primjer, da je cijeli kod napisan u MonoBehaviour skripti, korisnik ne bi imao mogućnost odabira glavnih osi, pošto je dijaloški okvir u kojemu se pruža taj izbor dio korisničkog sučelja.

Zbog toga, za dohvat glavnog spoja i učitavanje .csv datoteke koristi se MonoBehaviour skripta CSVProcessor.cs, a za interpretaciju i primjenu zglobova se koristi Editor skripta CSVProcessorEditor.cs.

5.2.1. CSVProcessor.cs

Ova skripta sadržava klasu `CSVProcessor`, koja sadrži dvije funkcije: `LoadCSV()` i `GetCSVData()`. Klasa koristi ugrađene UnityEngine i .NET biblioteke za čitanje CSV datoteka.

Funkcija `LoadCSV()` je odgovorna za učitavanje podataka iz CSV datoteke. U početku, postavlja vrijednost varijable `rootObject` tipa `GameObject` koristeći ugrađenu `gameObject` referencu. `gameObject` referenca proizlazi iz ugrađene UnityEngine biblioteke, i referencira se na `GameObject` na koji je skripta postavljena. Zbog toga je potrebno skriptu staviti na sklop modela u Unityju. Definira se i lista `csvData`, koja pohranjuje svaki redak CSV datoteke kao niz stringova. Koristi `StreamReader` klasu kako bi se otvorila i pročitala CSV datoteka koja se nalazi na putanji specificiranoj u varijabli `csvFilePath`. Svaki stupac u retku CSV datoteke se dijeli po zarezima pomoću metode `Split()`, a rezultirajući nizovi stringova se pohranjuju u listu `csvData`.

Funkcija `GetCSVData()` vraća listu `csvData` koja sadrži učitane podatke iz CSV datoteke. Ova funkcija omogućava pristup podacima CSV datoteke ostatku skripti CSVProcessorEditor.cs i MateLibrary.cs.

5.2.2. CSVProcessorEditor.cs

Ova skripta sadržava klasu `CSVProcessorEditor`, koja sadrži dvije funkcije: `OnInspectorGUI()` i `ApplyCSVDataToChildrenWithDialog(CSVProcessor script)`.

Funkcija `OnInspectorGUI()` upravlja korisničkim sučeljem unutar Unity Editora. Nasljeđuje osnovno ponašanje `Inspector` sučelja koristeći metodu `DrawDefaultInspector()`, a zatim dodaje dva nova gumba. Prvi gumb, nazvan „Load CSV“, povezan je s funkcijom `LoadCSV()` iz klase `CSVProcessor`, koja učitava podatke iz CSV datoteke pohranjene u listu. Drugi gumb, „Apply CSV Data to Children“, poziva funkciju `ApplyCSVDataToChildrenWithDialog()`, koja omogućava primjenu podataka iz CSV-a na djecu objekata u Unity sceni. Ova funkcija obrađuje podatke, dohvaća elemente iz scene i na njih primjenjuje zglobove temeljem podataka iz CSV datoteke.

Funkcija `ApplyCSVDataToChildrenWithDialog(CSVProcessor script)` je funkcija preko koje se odvija interpretacija liste CSV podataka. Prima parametar tipa `CSVProcessor`, koji se referira na skriptu koja sadrži tu klasu. Prvo provjerava jesu li svi potrebni uvjeti zadovoljeni, uključujući postojanje `rootObject` objekta i liste podataka. Ako ti uvjeti nisu ispunjeni, funkcija prekida rad. Nakon toga, dohvaća se komponenta `MateLibrary` iz `rootObject` objekta, koja je odgovorna za postavljanje zglobova na dijelove modela u sceni.

Nakon što su provjereni svi uvjeti, funkcija dodaje `Rigidbody` komponentu na `rootObject` s isključenom gravitacijom i kinematikom, čime se osigurava da je glavni sklop modela fiksiran u sceni. Zatim se otvara for petlja koja obrađuje podatke iz liste redak po redak. Svaki redak sadrži informacije o povezanim komponentama te vrsti spoja ili ograničenja koji se treba primijeniti između njih.

Prvi korak u obradi svakog retka jest dohvat prvog i drugog objekta iz scene putem funkcije `GameObject.Find()`, koristeći imena objekata iz liste. Ako bilo koji od tih objekata nema pridruženu `Rigidbody` komponentu potrebnu za funkcionalnost zglobova, funkcija automatski dodaje `Rigidbody` komponentu i postavlja joj potrebne opcije, kao što je isključivanje gravitacije.

Nakon što su komponente i vrsta spoja ili ograničenja definirani, korisniku se otvara dijaloški okvir za odabir osi na kojoj će se primijeniti ograničenje. Korisnik može birati između X, Y ili Z osi, ovisno o glavnoj osi oko koje korisnik želi ograničiti gibanje komponenti. Ako je vrsta ograničenja koincidentnost, korisnik dodatno specificira geometrijsku prirodu spoja putem dijaloga, birajući između točke, crte ili ravnine, kako bi precizno definirao u kojem su odnosu komponente. Pošto funkcija dijaloškog okvira podržava maksimalno tri opcije koje može ponuditi korisniku, pritiskom na opciju „crta“ otvara se novi dijaloški okvir koji nudi korisniku izbor između pravca, krivulje i kružnice.

Na kraju, nakon što je odabrana os, funkcija se poziva na skriptu `MateLibrary` koja na prvu komponentu spoja ili ograničenja primjenjuje odgovarajuću konfiguraciju zgloba.

5.2.3. MateLibrary.cs

Ova skripta sadržava klasu `MateLibrary`, koja sadrži funkciju `SetupJoint()`. Sama skripta nema funkcije kojima korisnik može pristupiti preko korisničkog sučelja, već služi kao biblioteka u kojoj su sadržane konfiguracije zglobova koje je potrebno primijeniti na dijelove modela.

Klasa sadrži četiri važne varijable: `selectedGameObject`, `connectedGameObject`, `primaryAxis` i `configurableJoint`. Varijabla `selectedGameObject` predstavlja prvu komponentu spoja ili ograničenja. Na tu komponentu se dodaje Unity komponenta `ConfigurableJoint`, koja dodaje zglob na komponentu. `connectedGameObject` predstavlja drugu komponentu spoja ili ograničenja. Varijabla `primaryAxis` je tipa `Vector3` i koristi se za definiranje glavne osi spoja.

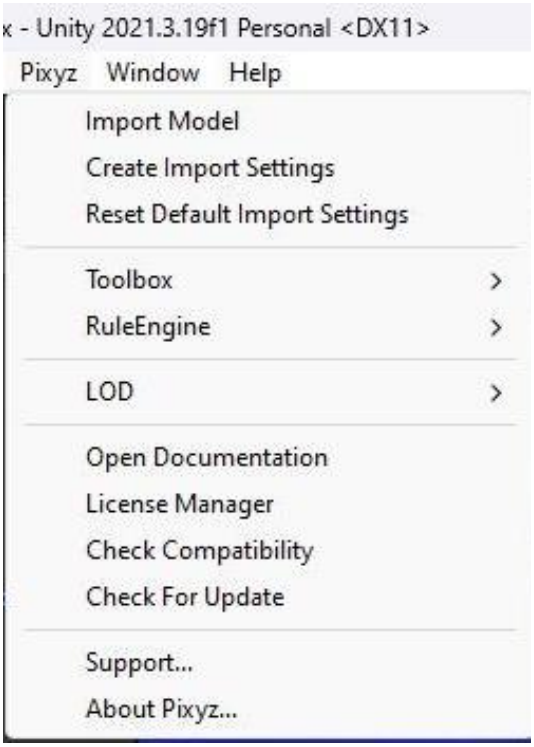
Funkcija `SetupJoint()` ima dva parametra: `selectedMateType` tipa `string`, koji označava vrstu spoja ili ograničenja, i `geometricType` tipa `string`, koji se koristi jedino u slučaju koincidentnosti. Ovisno o vrijednostima ovih parametara, funkcija dodaje `ConfigurableJoint` na `selectedGameObject` i pozivom na pomoćne funkcije na zglob primjenjuje konfiguraciju u skladu sa željenim spojem ili ograničenjem. Funkcija također koristi varijablu `primaryAxis` kako bi definirala glavnu os zgloba.

6. PRIJENOS MODELA I PRIMJENA SKRIPTI

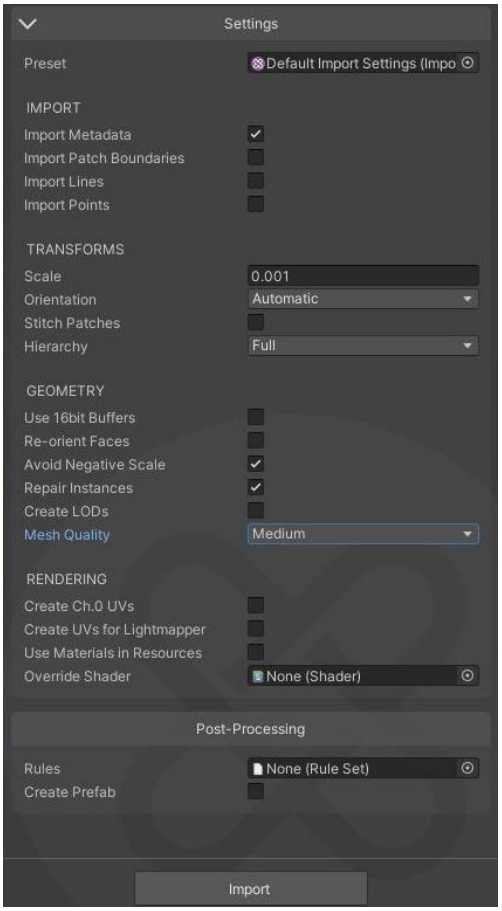
Nakon što su napisane skripte i napravljena aplikacija, može započeti proces prijenosa CAD modela iz sustava SolidWorks u VR okruženje Unity platforme. Potrebno je naglasiti kako su koraci opisani u nastavku jednako primijenjeni na sve modele tako da nije potrebno analizirati korake koji su bili potrebni za svaki model. Cilj prijenosa je da bude primjenjiv na svaki sklopni model bez posebnih modifikacija modela (poput dodavanja novih korisničkih skripti) prilikom prijenosa. Sve ručne modifikacije koje je potrebno odraditi na specifičnom modelu poslije prijenosa bit će naknadno objašnjene.

6.1. Prijenos geometrije i hijerarhije sklopa

Model je potrebno unutar SolidWorksa spremiti u .SLDASM ili .SLDPRT formatu, ovisno o tome radi li se o sklopu ili dijelu. Zatim je unutar Unityja potrebno otvoriti padajući izbornik *Pixyz* te kliknuti na Import Model. Na otvorenom izborniku potrebno je izabrati *Medium* kod izbora *Mesh Quality,* te *Full* kod izbora *Hierarchy*. Opcija *Medium* izabrana je zbog smanjenog opterećenja na hardware računala korisnika. Platforme za izradu video igara veoma su grafički i memorijski zahtjevni programi, te opcija *Medium* pruža dobar omjer kvalitete prikaza i performansi računala. Opcija *Full* kod izbora *Hierarchy* označava da korisnik želi prenijeti potpunu hijerarhiju (sve dijelove, podsklopove i dijelove u tim podsklopovima) prisutnu u CAD modelu. Postoje više opcija u ovome izboru, poput *Merge All* koji sve dijelove u hijerarhiji spoji i pretvori u jedan mnogokutni model. Dok je ova opcija dobra za uvoz npr. statičnih modela u videoigrama, za potrebe CAD modela sklopa ne zadovoljava prijenos hijerarhije pošto ona nakon uvoza ne postoji. U izborniku se također može izabrati opcija *Default Import Settings* kod izbora *Preset*. Ova opcija automatski podešava *Mesh Quality* na *Medium* i *Hierarchy* na *Full*. Sve ostale opcije nemaju veliki utjecaj na prijenos oblika i hijerarhije pa se mogu zanemariti u okviru ovoga rada. Opisani proces prikazan je na slikama 26. i 27.

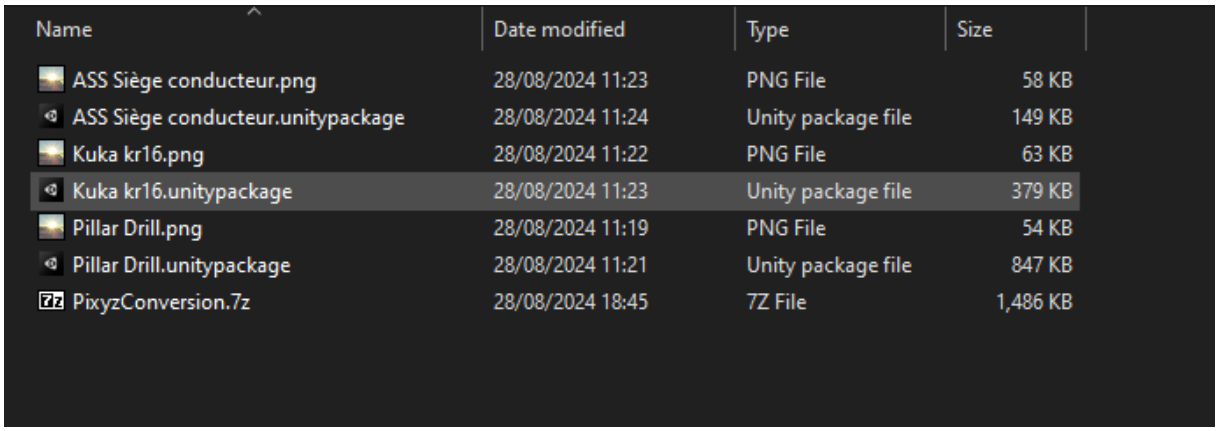


Slika 26. Klikom na *Import model* otvara se sučelje za uvoz sklopa



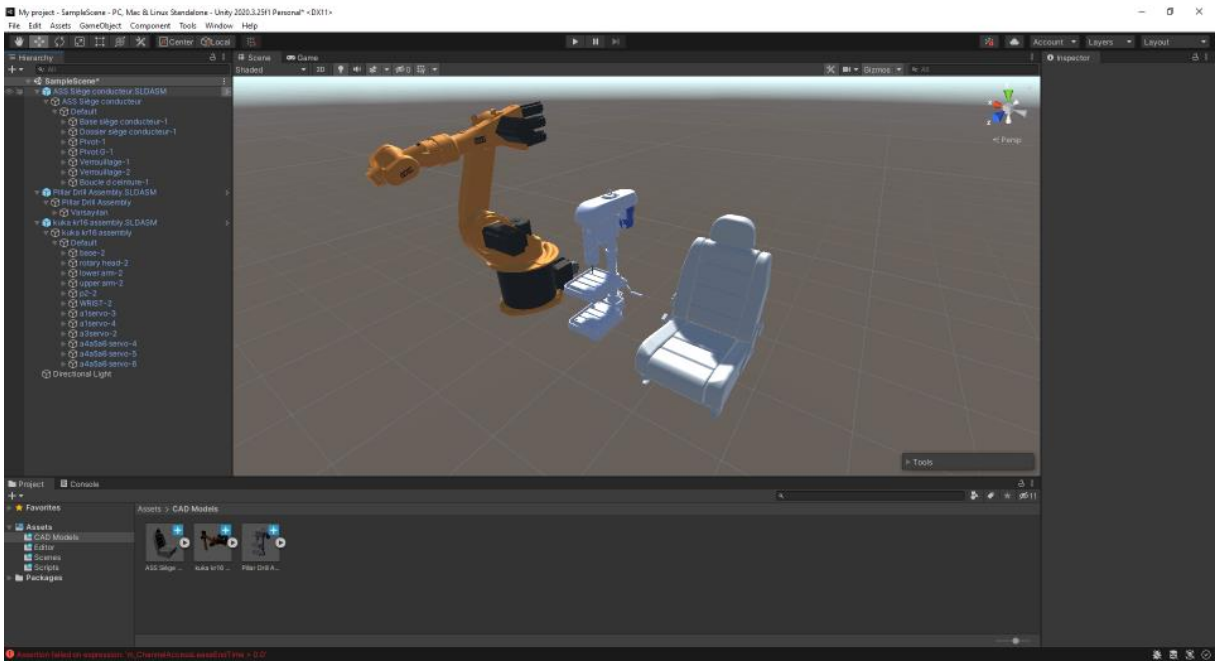
Slika 27. Pod *Preset* se odabire *Default Import Settings*, što generira mnogokutni model srednje (*Medium*) kvalitete

Nakon pritiska na gumb *Import*, Pixyz generira .unitypackage datoteku. Ova datoteka sadrži preneseni mnogokutni model koji sadrži oblik modela, njegovu potpunu hijerarhiju i neke od boja prisutnih u CAD sustavu. Datoteka također sadrži i skripte zvane *Metadata*, koje sadrže informacije o autoru modela, datumu izrade, itd. Generirane .unitypackage datoteke vidljive su na slici 28.



Slika 28. Prikaz .unitypackage datoteka generiranih Pixyz-om

Datoteku .unitypackage moguće je otvoriti unutar Unityja, te je potrebno iz izbornika *Project* mišem povući datoteku u Unity scenu kako bi se vidio prikaz modela. Na slici 29. moguće je vidjeti sva tri modeli prenesena u Unity pomoću Pixyza.



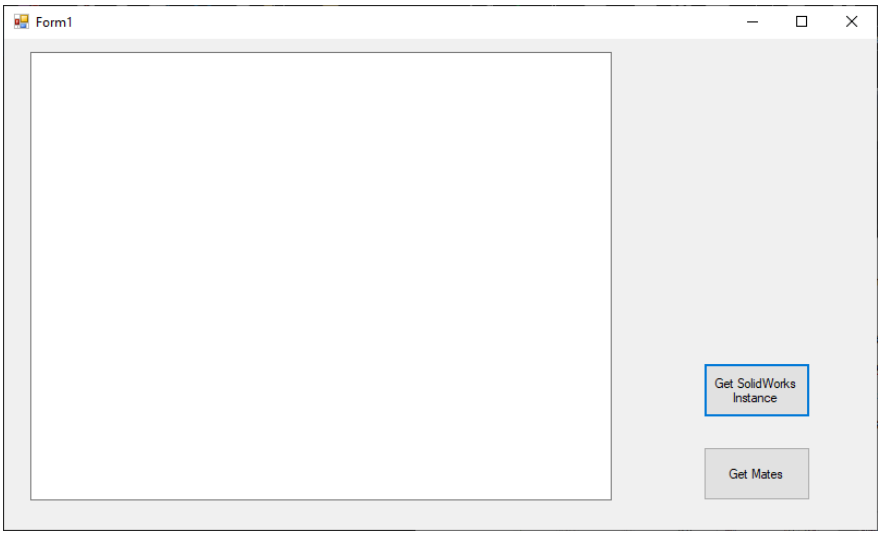
Slika 29. Prikaz modela u Unityju

6.2. Prijenos ograničenja i spojeva i primjena skripti

Nakon što je prenesen oblik modela moguće je i pokrenuti proces prijenosa spojeva i ograničenja. Prvo je potrebno iz aktivne SolidWorks instance zapisati sve spojeve i ograničenja u CSV datoteku pomoću izrađene aplikacije. Zatim je tu CSV datoteku potrebno prebaciti u mapu *Assets/* unutar Unityja te pokrenuti skriptu CSVProcessor.cs.

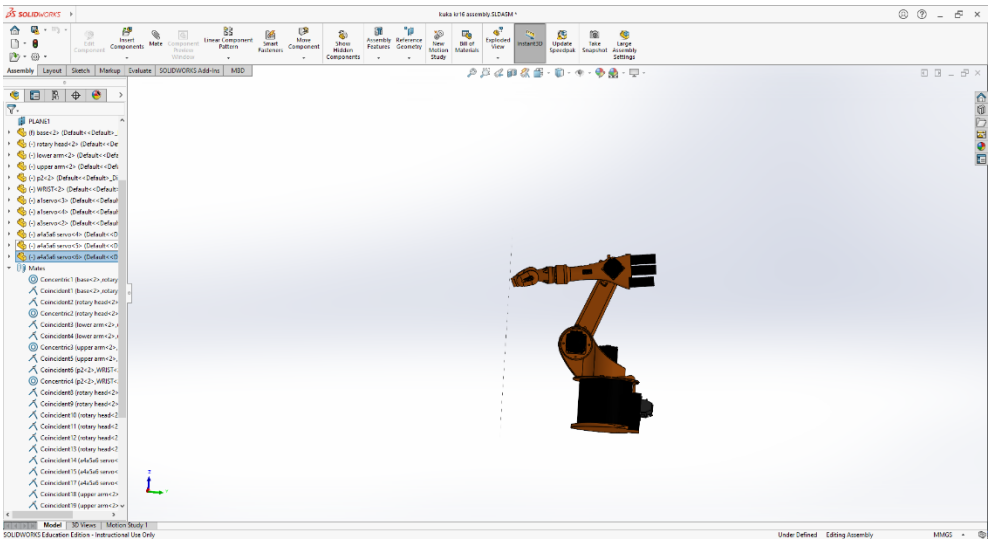
6.2.1. Generacija CSV datoteke

Za generaciju CSV datoteke potrebno je prvo imati aktivnu instancu SolidWorksa na računalu pokrenuti novu preko gumba „Get SolidWorks Instance“.



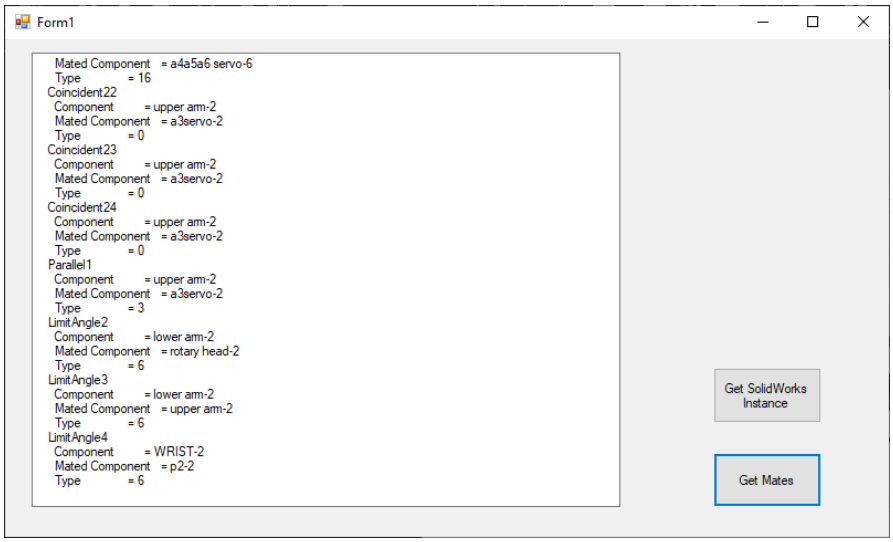
Slika 30. Dohvaćanje SolidWorks instance pomoću .NET aplikacije

Zatim je potrebno otvoriti željenu datoteku modela u toj instanci SolidWorksa, prikazano na slici 31.



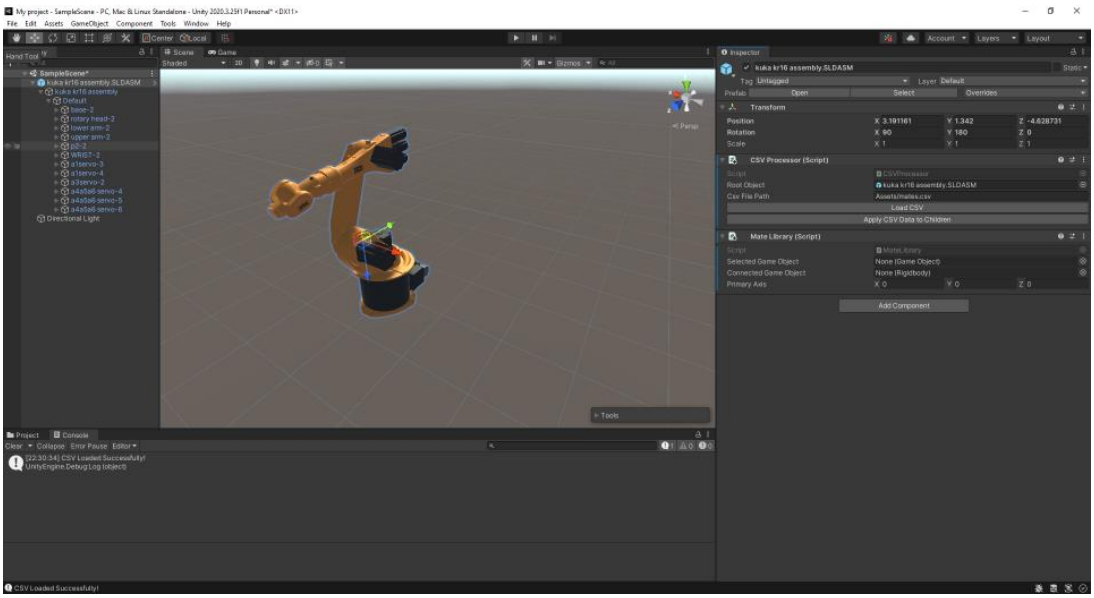
Slika 31. Otvaranje željenog modela unutar te instance

Učitavanjem modela u aktivnu SolidWorks instancu, pritiskom na gumb „Get Mates“ unutar aplikacije počinje generiranje CSV datoteke. Korisnik može pratiti napredak procesa pomoću teksta u *textbox* elementu, te njime potvrditi da se proces dobro izvršava jer se informacije pokazuju u *textbox* elementu jedino nakon što su zapisane u CSV datoteku. Ovaj proces je vidljiv na slici 32.



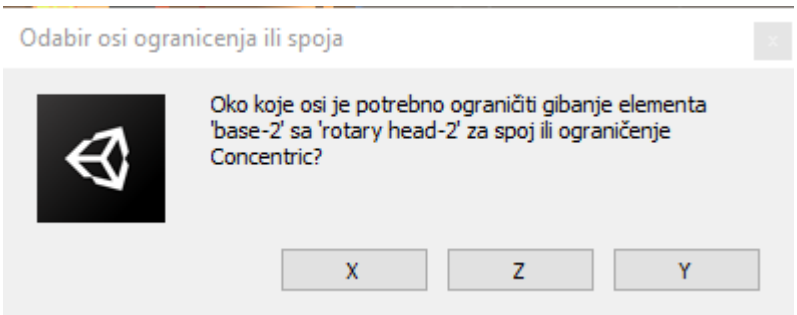
Slika 32. Pronalazak i zapis ograničenja i spojeva pomoću .NET aplikacije

Nakon što je generirana CSV datoteka sa spojevima i ograničenjima, potrebno ju je premjestiti u *Assets/* mapu unutar Unity projekta. Nakon što je datoteka prebačena, na sklop modela primjenjuju se skripte CSVProcessor.cs i MateLibrary.cs. Pritiskom na gumb „Load CSV“ CSV datoteka se učitava u skriptu CSVPRocessor i zapisuje u C# listu, prikazano na slici 33.

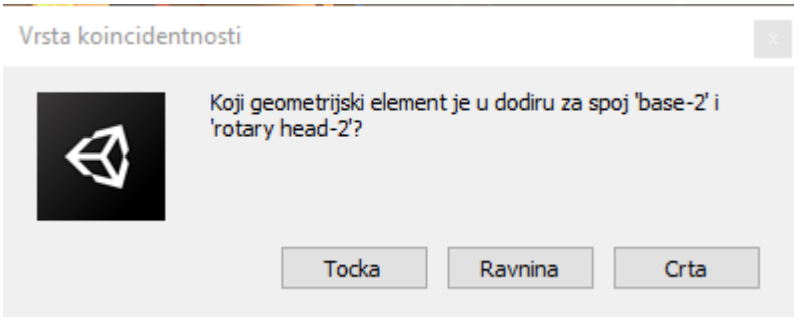


Slika 33. Primjena CSVProcessor.cs i MateLibrary.cs skripti na model i učitavanje CSV datoteke pritiskom na „Load CSV“ gumb

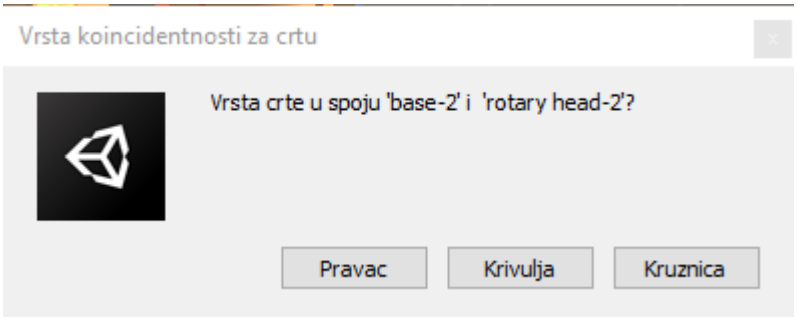
Nakon što je CSV Datoteka pretvorena u listu, korisnik pritiskom na gumb „Apply CSV Data to Children“ pokreće skriptu CSVProcessorEditor.cs koja provodi proces dodjele zglobova na dijelove sklopa. Korisnik u dijaloškim okvirima bira glavnu os ograničenja ili spoja. Dolaskom do koincidentnog ograničenja korisniku se nudi izbor između kojih se geometrijskih elemenata nalazi ograničenje. Dijaloški okviri za proces su prikazani na slikama 34., 35., i 36.



Slika 34. Pritiskom na „Apply CSV Data to Children“ gumb, počinje proces koji će za svaki spoj pitati korisnika da odredi glavnu os spoja



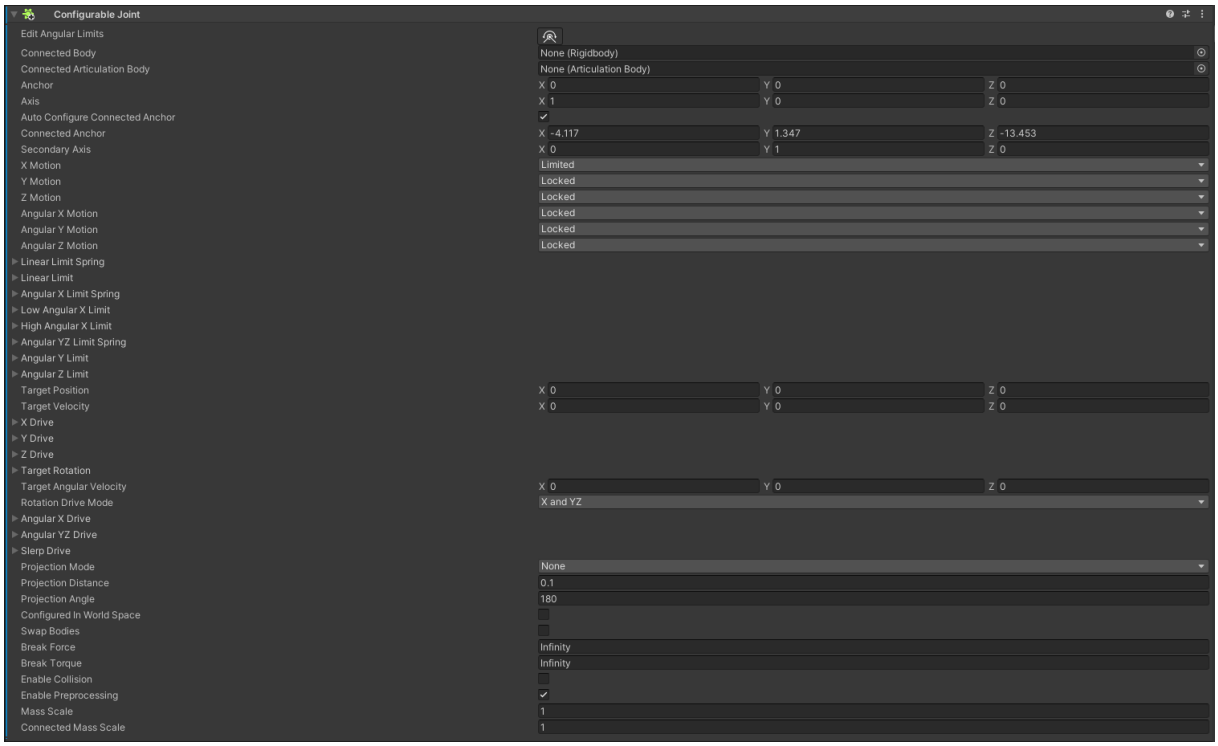
Slika 35. Dolaskom do koincidentnih ograničenja, skripta traži od korisnika u kojem bi geometrijskom elementu trebao biti spoj



Slika 36. U slučaju koincidentnog ograničenja u „crti“, skripta mora prepoznati je li ta crta kružnica, pravac ili krivulja

Nakon što je skripta prošla kroz listu i dodijelila zglobove na model, korisnik može na dijelovima modela vidjeti dodane komponente zglobova, te ih po potrebi uređivati. Prikaz ovih zglobova u Unity sceni vidljiv je na slici 37.

Slika 37. Nakon primjene svih ograničenja i spojeva iz CSV datoteke, na dijelovima sklopa moguće je vidjeti zglobove generirane skriptom

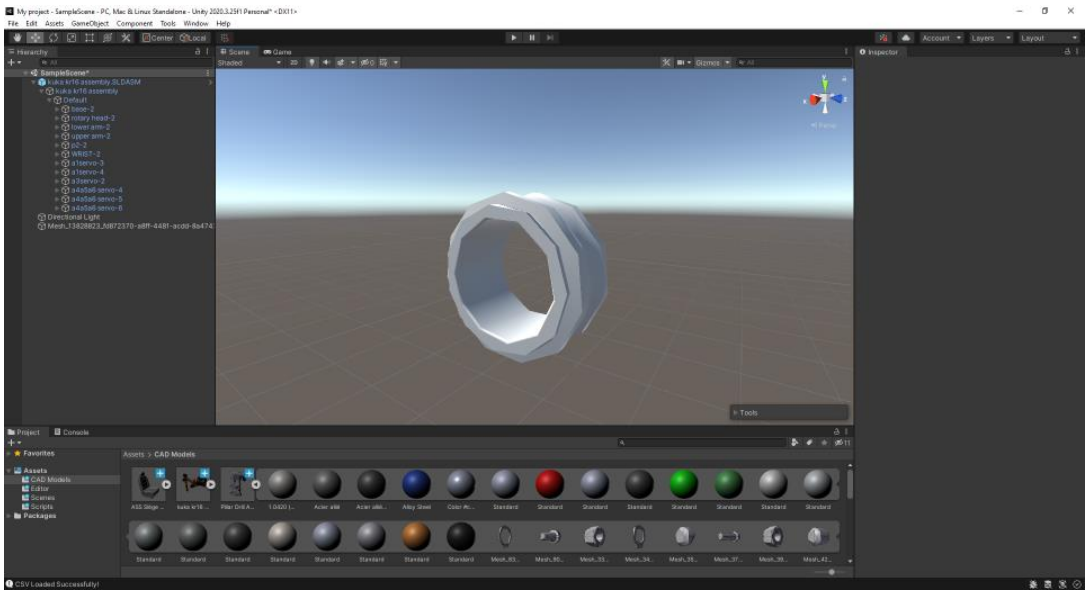


Slika 38. Opcije konfigurabilnog zgloba

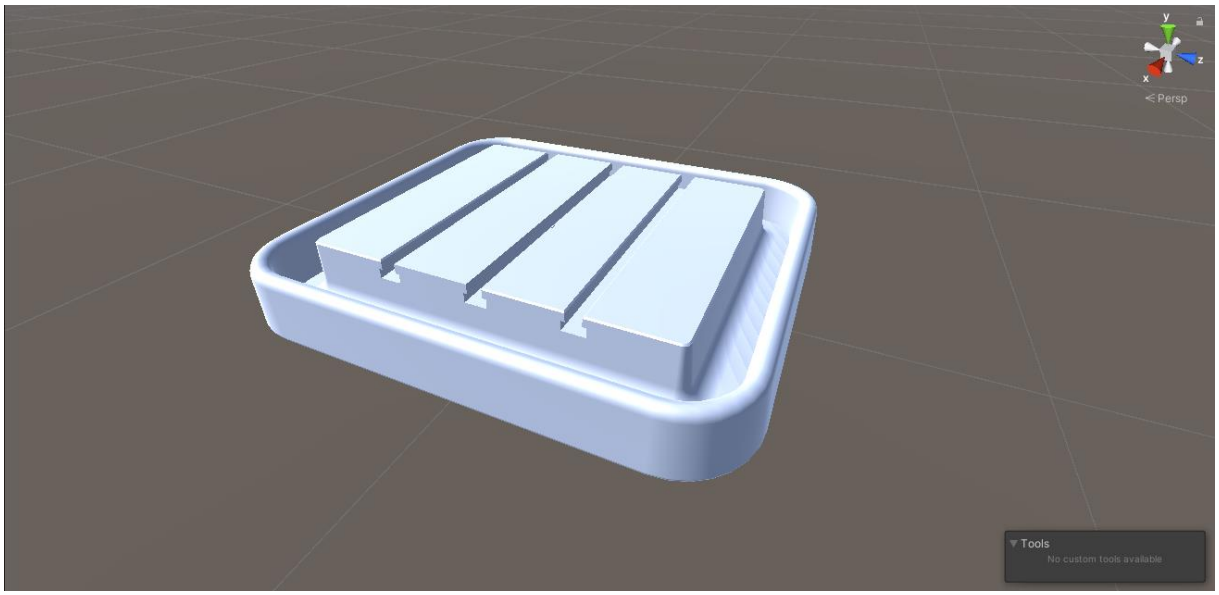
Na slici 38. vidljive su opcije zgloba koji simulira ograničenje udaljenosti, primijenjenog pomoću skripti. Vidljivo je da je rotacija i translacija po svim osima osim glavne osi (X motion) onemogućena, dok je translacija po glavnoj osi ograničena. Iznos ograničenja udaljenosti korisnik može izmijeniti u opciji *Linear Limit*. Još dvije važne opcije u konfigurabilnom zglobu su *Anchor* i *Connected Anchor*. *Anchor* definira ishodište zgloba i po zadanim postavkama se nalazi u ishodištu lokalnog koordinatnog sustava. Zbog toga su sve koordinate *Anchora* nula. *Connected Anchor* je točka u drugoj komponenti zgloba na koju će se primjenjivati svo gibanje (ili nedostatak istog) prve komponente zgloba. Koordinate točke su prikazane u lokalnom koordinatnom sustavu prve komponentne zgloba, i po zadanim postavkama taj vrh se nalazi u ishodištu lokalnog koordinatnog sustava druge komponente. Zbog toga, osim ako se ishodišta lokalnih koordinatnih sustava obje komponentne ne nalaze u istoj točci, koordinate *Connected Anchora* neće biti nula.

6.3. Analiza prijenosa i potrebne daljnje ručne modifikacije na modelima

Oblik sva tri modela zadovoljavajuće se prenio po kriteriju kvalitete prenesenih površina. Niža kvaliteta teseliranih površina mnogokutnog modela jedino je vidljivo lošija uvećanim pogledom na zakrivljene površine malih dimenzija u odnosu na cijeli model, poput matica i vijaka. Razlika kvalitete između modela malih i modela velikih dimenzija vidljiva je na slikama 39. i 40.

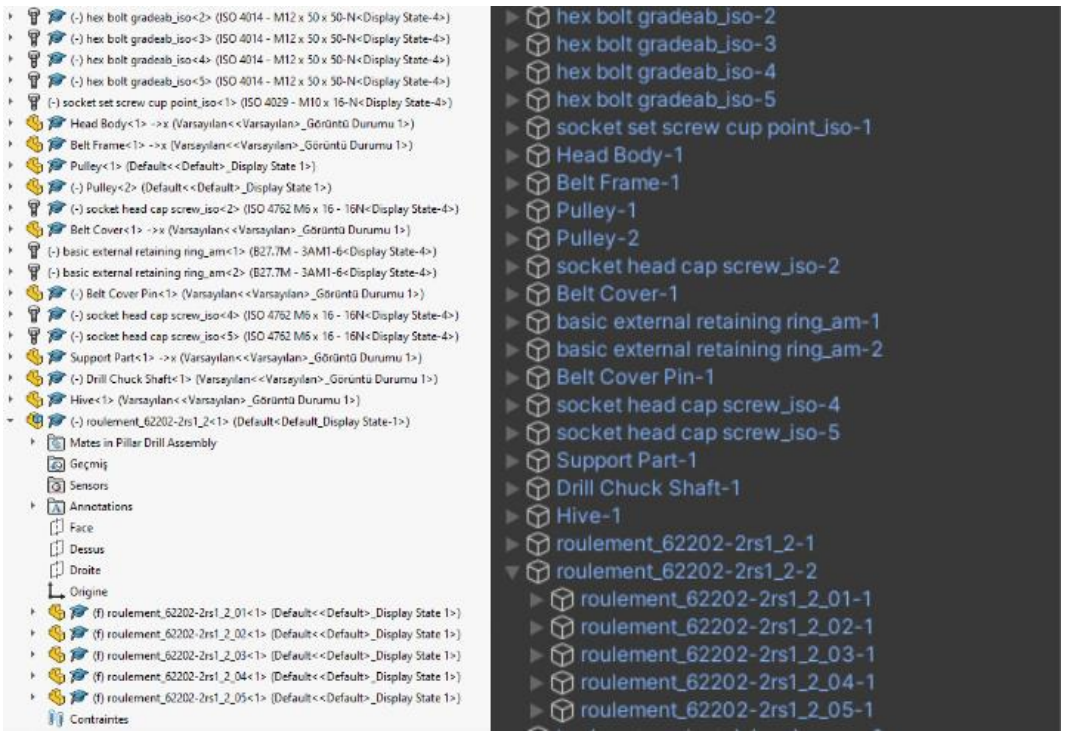


Slika 39. Mnogokutni model matice stupne bušilice u Unityju



Slika 40. Mnogokutni model podloge za obradak stupne bušilice u Unityju

Prijenos hijerarhije je za sva tri modela također je bio zadovoljavajući po kriteriju prijenosa potpune hijerarhije modela iz SolidWorksa. Na slici 41. prikazana je usporedba isječka hijerarhije stupne bušilice u Solidworksu i Unityju.



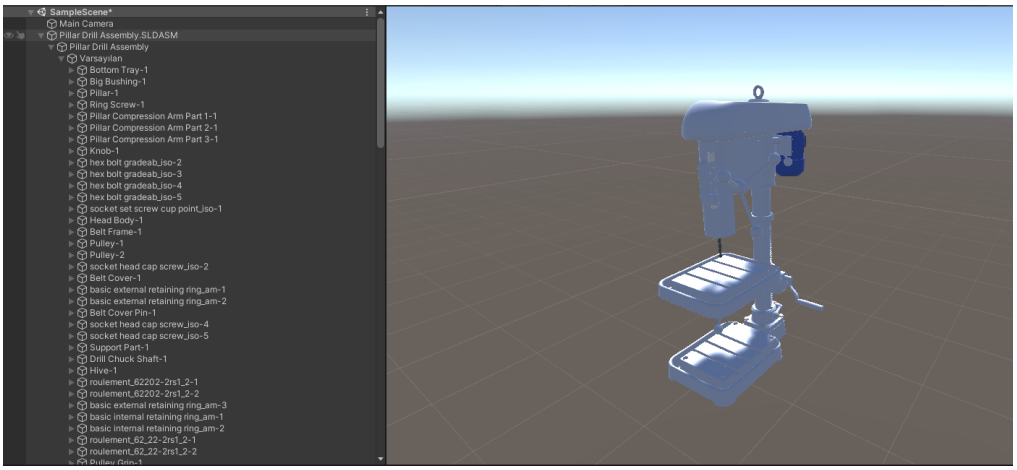
Slika 41. Usporedba hijerarhije modela stupne bušilice u SolidWorksu i Unityju nakon prijenosa pomoću Pixyza

Prijenos spojeva i ograničenja nije bio toliko uspješan. Najveći problem prijenosa je činjenica da Unity ne prepoznaje referentnu geometriju modela te se svi zglobovi referiraju na središnju točku zasebnih dijelova. Ova točka je ishodište dijela definirano još u SolidWorksu te je njezina pozicija u prostoru vjerno prenesena u Unity zahvaljujući Pixyzu. Ne koriste sva ograničenja i spojevi ovu točku kao referencu u SolidWorksu, i zbog toga se većina spojeva neće vjerno prenijeti.

Kako bi se dobio bolji uvid u to kako su se prenijeli ograničenja i spojevi modela, potrebno je svaki analizirati te vidjeti koji su koraci bili potrebni kako bi modeli bili funkcionalni.

6.3.1. *Stupna bušilica*

Prenesen oblik i hijerarhija modela stupne bušilice unutar Unityja prikazan je na slici 42.



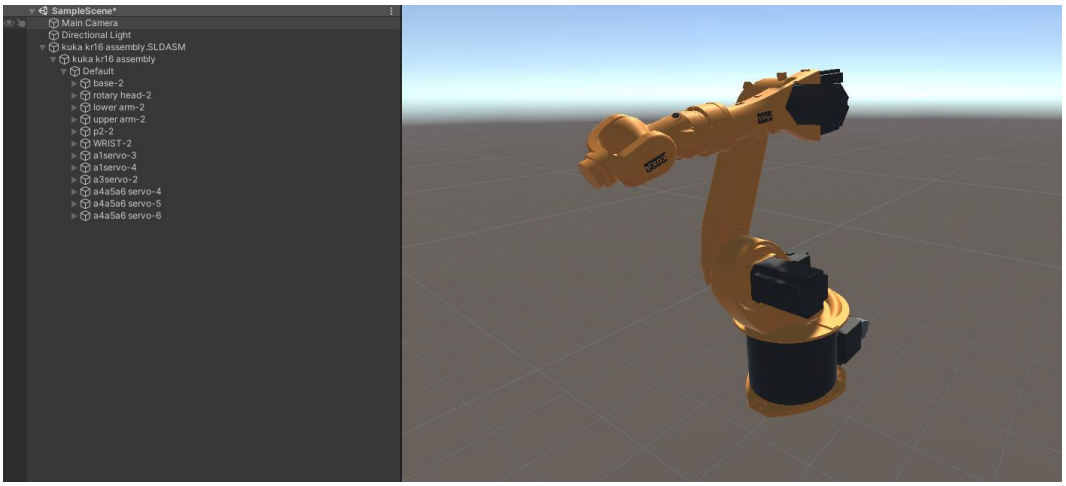
Slika 42. Oblik i hijerarhija stupne bušilice u Unityju

Prijenos ograničenja i spojeva stupne bušilice prošao je najgore od svih modela. Glavni razlog tome je broj spojeva i ograničenja u sklopu. Pošto skripte za prijenos traže od korisnika da odabere glavnu os za svaki spoj, teško je zapamtiti kako bi se svaki od 239 spojeva i ograničenja trebao ponašati. Čak i ako se glavna os svakog spoja dobro odabere, ključni spoj bušilice definiran u 3.4.1. bilo bi potrebno ručno isprogramirati kako bi bio funkcionalan. Čak i sa svim tim koracima, prijenos još uvijek ne bi bio uspješan, ponovno pozivajući se na nemogućnost Unityja da prepoznaje referentnu geometriju.

Zbog navedenih razloga, utvrđeno je da je prijenos modela stupne bušilice neuspješan te da je nepotrebno na njemu provoditi ručne modifikacije. Iako je model moguće dovesti u funkcionalnu stanje unutar Unityja s ručnim modifikacijama, naglasak ovoga rada je na automatizaciji prijenosa te bi taj proces korisniku bio previše naporan i predugo bi trajao da se opravda njegovo provođenje u okviru ovoga rada.

6.3.2. *Kuka KR16 Robotska Ruka*

Prenesen oblik i hijerarhija modela robotske ruke kuka KR16 unutar Unityja prikazan je na slici 43.



Slika 43. Oblik i hijerarhija robotske ruke kuka KR16 u Unityju

Prijenos ograničenja i spojeva robotske ruke Kuka KR16 nakon inicijalnog prijenosa bio je nezadovoljavajući, ali kroz ručne modifikacije prijenos je bio uspješan.

Najveći problem prilikom prijenosa, osim već spomenutog problema geometrije, bio je način na koji skripte primjenjuju zglobove na dijelove. Naime, skripta primjeni zglob samo na prvu komponentu spoja, dok se druga referencira kao povezano tijelo. Razlog tome je problem koji nastaje ako se oba dijela spoja međusobno referenciraju kao povezana tijela, a oba imaju primijenjena ograničenja od zglobova. Gibanjem jedne komponente pomicat će se druga komponenta, no pomicanjem druge komponente aktivirat će se zglob koji referencira prvu komponentu. Time će zglob druge komponente pokušati prenijeti svoju kretnju na prvi dio. Tako dolazi do poremećaja kretnje prvoga dijela i željeno kretanje je obustavljeno.

Kako su spojevi unutar SolidWorksa većinom referencirali zglobove robotske ruke kao prvu komponentu, servomotori koji su bili fiksirani na zglobove pomoću ograničenja *Lock* prilikom prijenosa u Unity su prenijeli fiksirane zglobove namijenjene njima na zglobove robotske ruke. Ručna modifikacija bila je ispravak u obliku prijenosa tih zglobova na servomotore. Time su servomotori bili fiksirani na robotsku ruku i gibali su se zajedno sa njom, a gibanje robotske ruke sa svih 6 stupnjeva slobode više nije bilo onemogućeno.

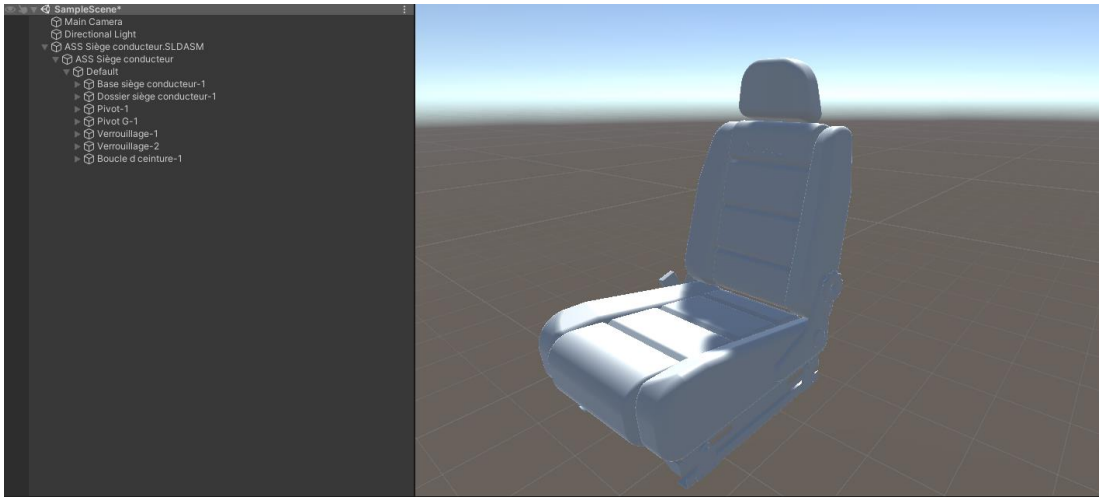
Također, bilo je potrebno ukloniti nekoliko automatski dodanih zglobova. Ovi zglobovi su uglavnom bili nepotrebni unutar Unityja, poput koncentričnog i koincidentnog ograničenja između baze ruke i prvog zgloba. Oba ograničenja su se referencirala na kružnicu u bazi ruke. I jedno i drugo ograničenje je omogućavalo rotaciju oko Z osi modela, ali koncentrično ograničenje je još dopuštalo i translaciju po Z osi. Kako tu translaciju ne dopušta koincidentnost, a oba zgloba imaju iste referentne točke, koncentrično ograničenje je

nepotrebno te je bilo uklonjeno. Unutar SolidWorksa, ovi spojevi su važni za potpunu definiciju modela, no unutar Unityja nisu potrebni zbog nedostatka geometrijskih referenci.

Zaključno je da je model robotske ruke Kuka KR16 uspješno prenesen uz malu količinu ručnih modifikacija modela. Identifikacija problematičnih zglobova i provođenje ručnih modifikacija trajalo je otprilike pola sata.

6.3.3. Automobilsko sjedalo

Prenesen oblik i hijerarhija modela automobilskog sjedala unutar Unityja prikazan je na slici 44.



Slika 44. Oblik i hijerarhija automobilskog sjedala u Unityju

Prijenos ograničenja i spojeva automobilskog sjedala nakon inicijalnog prijenosa bio je najbliži uspješnom od sva tri modela, ali još uvijek je bilo potrebno primijeniti par ručnih modifikacija. Identifikacija problematičnih zglobova i primjena ručnih modifikacija trajala je otprilike 10 minuta.

Većina ovih modifikacija odnosile su se na uklanjanje nepotrebnih zglobova, objašnjenih u analizi prijenosa robotske ruke Kuka KR16 u prošlom poglavlju. Nakon toga, model se mogao smatrati uspješno prenesenim.

Razlog zašto je ovaj model bio najjednostavniji za prijenos je njegova mala količina spojeva te jednostavna izvedba ključnog spoja. Jedini pokretni dio je kotačić za spuštanje sjedala koji se nalazi na desnoj strani modela, te je on ujedno dio jedinog ključnog spoja modela. Ovaj kotačić je na ostatak modela spojen sa samo dva ograničenja: koncentričnost i ograničenje udaljenosti. Ti spojevi su veoma laki za automatski prijenos, pošto ne postoji puno varijacija u njihovim kretnjama za razliku od ograničenja poput koincidentnosti.

7. DISKUSIJA

Osvrtom na izvršeni proces prijenos modela, potrebno je komentirati na nedostatke primijećene prilikom prijenosa i dati preporuke za daljnji rad na ovoj temi. No prije nego se provede osvrt na nedostatke prijenosa, potrebno je naglasiti koje su prednosti korištenja aplikacije za prijenos ograničenja i spojeva naspram njihovog ručnog prijenosa.

7.1. Prednosti aplikacije i skripti naspram ručnog prijenosa

Prijenos opisan u ovome radu koristio je .NET aplikaciju i Unity skripte kako bi olakšao proces prijenosa ograničenja i spojeva CAD modela iz SolidWorksa u Unity. Ako bi se ovaj proces provodio ručno, zahtijevao bi veliku količinu vremena i truda od strane korisnika. Za svako ograničenje i spoj u modelu bi od strane korisnika trebao ručno biti postavljen i konfiguriran zglob. Korisnik bi ručno trebao unositi komponente zgloba i ograničenja gibanja komponenti po osima. Iako bi takav ručni prijenos bio precizniji od automatskog, bio bi puno vremenski zahtjevniji.

Koristeći aplikaciju i Unity skripte, vrijeme prijenosa bez ručnih modifikacija od otvaranja datoteke modela u instanci SolidWorksa do primijene svih zglobova na model unutar Unityja bilo je par minuta po modelu. Uz to, primjena ručnih modifikacija trajala je maksimalno 30 minuta u okviru ovoga rada. Korištenjem ovog većinom automatiziranog prijenosa, potencijalno višesatni ručni prijenos skraćen je na nešto više od pola sata sveukupnog rada oko prijenosa modela. Sve skripte, kao i aplikacija, mogu se naknadno nadograditi uz bolje razumijevanje SolidWorks API-ja i Unity platforme kako bi proces u budućnosti bio što precizniji. No, kako bi bilo lakše preporučiti smjernice za daljnji razvoj aplikacije i skripti, potrebno je sagledati nedostatke prijenosa aplikacijom i skriptama.

7.2. Nedostatci aplikacije za prijenos spojeva i ograničenja

Najveći nedostatak aplikacije je činjenica da može prenositi spojeve i ograničenja samo iz SolidWorks CAD sustava. Aplikacija za generaciju CSV datoteke koristi klase i funkcije SolidWorks API-ja koje su primjenjive samo na SolidWorks. Kako nemaju svi CAD sustavi integrirani API, aplikacija neće moći prenijeti spojeve i ograničenja modela iz tih sustava. Čak i ako CAD sustav ima integrirani API, bila bi potrebna prilagodba koda s funkcijama specifičnima za taj API. Najbolje rješenje za ovaj problem bilo bi iščitavanje podataka direktno

iz izvornog formata CAD sustava, no kako je već napomenuto, taj način zahtjeva pristup izvornom kodu prevoditelja CAD sustava. Taj kod se smatra intelektualnim vlasništvom tvrtke, tako da je bez njihovog dopuštenja nemoguće saznati način zapisa izvornih formata CAD sustava određene tvrtke.

Sljedeći nedostatak aplikacije je nemogućnost zapisa koordinata gdje se u prostoru nalazi spoj ili ograničenje. SolidWorks API sadrži vrijednost klase MateEntity koja može pristupiti koordinatama spoja ili ograničenja u odnosu na ishodište sklopa. Problem ne proizlazi iz zapisa ovih koordinati preko aplikacije, već u njihovoj primjeni unutar Unityja. Kao što je već spomenuto, zglobovi unutar Unityja zapisuju svoje referentne točke u lokalnim koordinatama dijela na koji je zglob primijenjen. Da bi se primijenile koordinate iz CAD sustava na referentnu točku zgloba unutar Unityja, potrebno bi bilo razviti dodatnu skriptu koja bi te koordinate prvo transformirala u Unity koordinate. Zatim bi bilo potrebno te koordinate transformirati u lokalne koordinate specifičnog dijela na koji se zglob postavlja. Ovaj proces zahtijeva napredno razumijevanje Unity platforme te je jedna od preporuka za daljnji razvoj procesa prijenosa.

7.3. Nedostatci Unity skripti

Najveći nedostatak opažen prilikom primjene Unity skripti bio je proces dodjeljivanja zglobova. Primjenom zgloba samo na prvu komponentu spoja ili ograničenja se na tu komponentu mogu prenijeti ograničenja gibanja namijenjena za drugu komponentu. Za modele s malim brojem spojeva i ograničenja poput automobilskog sjedala opisanog u ovome radu ovo ne stvara veliki problem. Potrebno je samo kopirati zglob na drugu komponentu spoja ili ograničenja i obrisati zglob s prve komponente. No, za sklopove s velikim brojem ograničenja i spojeva poput stupne bušilice opisane u ovome radu, ovaj proces bio bi previše vremenski zahtjevan da ga je korisniku isplativo provoditi. Jedno moguće rješenje za ovaj problem je da se prilikom primjene spojeva i ograničenja na sklop u CAD sustavu obrati pozornost na to kojoj komponenti je potrebno ograničiti gibanje, te tu komponentu definirati kao prvu komponentu spoja ili ograničenja. Za rješenje ovog problema unutar same Unity platforme potrebno je napredno znanje njenih funkcionalnosti, te je za daljnji rad ovo jedan od nedostataka koji se treba riješiti za bolju automatizaciju procesa.

Još jedan od nedostataka je upit korisnika za glavnu os svakog spoja ili ograničenja. Glavni razlog zašto je dijaloški okvir za odabir glavne osi implementiran je zbog već spomenute nemogućnosti Unity platforme da prepoznaje referentnu geometriju modela. Moguće rješenje

za ovaj problem je da se modeli u CAD sustavu modeliraju s namjenom prijenosa modela u Unity. Što to podrazumijeva je modeliranje dijelova tako da svi namijenjeni spojevi i ograničenja za taj dio dijele glavnu os (definirana u 4.2.2.) na koju se referiraju. Na primjer, odlučeno je da je glavna os za svaki dio u sklopu os X. Korisnik želi u sklop dodati vijak koji će sam modelirati i glava tog vijka će biti koncentrično spojena sa rupom u sklopu. Vijak je potrebno modelirati tako da je tijelo izvučeno u smjeru X osi, jer tako će se primjenom koncentričnosti na vijak os na kojoj se nalaze koncentrični elementi biti X. U Unity skripti bi se zatim maknuli dijaloški okviri iz koda, i za svaki spoj i ograničenje bi se pretpostavilo da je glavna os X. Vidljivo je da bi ovo rješenje limitiralo izradu modela koji zahtijevaju spojeve i ograničenja prisutne na različitim osima. Također, svaki model bi trebao izrađivati korisnik, pošto modeli preuzeti s interneta ili napravljeni od strane drugih korisnika koji nisu upoznati sa smjernicama neće pratiti iznesene smjernice.

7.4. Preporuka za daljnji rad

Osim već navedenog, za daljnji rad preporučljivo je istražiti kako bi se prijenos modela odvio u druge platforme za razvoj video igara, poput Unreal Enginea. Unreal Engine ima pristup Datasmithu, prevoditelju sličnom Pixyzu, koji je besplatan. U budućnosti bi to bila puno održivija opcija jer ne bi bilo potrebno svake godine obnavljati i plaćati Pixyz licencu. Nedostatci Unreal Enginea u usporedbi s Unityem su veće opterećenje platforme za hardware računala te potrebno znanje C++ programskog jezika. Zbog tih razloga, Unreal Engine nije bio obrađen u okviru ovoga rada.

Također, dobar nastavak ovoga rada bio bi pronalazak metode prijenosa spojeva i ograničenja bez upotrebe API-ja, preferabilno interpretacijom datoteke modela bez njenog otvaranja u CAD sustavu. Izazovi koje donosi ovaj pristup već su naznačeni, ali ako bi se uspio izvesti prijenos spojeva i ograničenja iz CAD sustava na ovaj način uvelike bi se olakšao postupak prijenosa. Razlog tome je to što bi se datoteka mogla direktno analizirati preko skripti, umjesto interpretacije odvojene CSV datoteke.

8. ZAKLJUČAK

Potreba za integracijom CAD sustava s VR tehnologijom raste iz dana u dan. Aplikacija koja sveobuhvatno integrira VR sa CAD sustavima omogućila bi inženjerima i dizajnerima interakciju s proizvodom u virtualnom prostoru koja simulira rad s proizvodom u stvarnome svijetu. Mogućnost rukovanja s proizvodom u virtualnom okruženju prije nego je izrađen smanjio bi broj iteracija i potencijalnih prototipa proizvoda te bi se proces razvoja proizvoda značajno ubrzao. No razvoj takve aplikacije podrazumijeva potrebu za prijenosom CAD modela u VR okruženje, te se ovim radom postavio temelj za definiciju i razvoj procesa tog prijenosa.

Od početka ovoga rada postojao je naglasak na činjenicu da će prijenos modela iz CAD sustava u platformu za razvoj video igara biti veoma težak proces. Pregledom literature definirani su svi izazovi koji su prisutni u prijenosu CAD modela u VR okruženje poput Unityja. Pregledom metodologije definirana su potencijalna rješenja ovih izazova, te su izradom aplikacije i Unity skripti na temelju metodologije ta rješenja implementirana u proces prijenosa. Uspješnim prijenosom dva od tri modela korištena u okviru ovoga rada pokazano je kako je proces moguće olakšati, ako ne i u budućnosti potpuno automatizirati.

VR okruženje koje nudi Unity platforma omogućuje rad s modelima kakav nije moguć u CAD sustavima. Primjenom integriranog fizičkog sustava Unity platforme na modele, korisnici mogu vidjeti kako model reagira na fizičke sile koje su prisutne u stvarnome svijetu. Unity platforma također podržava unos više modela u VR okruženje, te bi korisnici mogli analizirati međusobnu interakciju više proizvoda u istome prostoru. Ovo su samo neke od prednosti korištenja CAD modela u Unity VR okruženju. Uspješnom integracijom CAD modela sa Unityjem otvorila bi se nova dimenzija interakcije korisnika sa CAD modelima.

Ovaj rad pokušao je postaviti temelj za razvoj te aplikacije. Daljnjim radom na ovoj temi postoji velika mogućnost da se potpuna integracija CAD modela u VR okruženje izvrši već u bližoj budućnosti. Unity platforma, kao i ostale platforme za razvoj videoigara, nudi veliki broj alata za modifikaciju virtualnog prostora. Boljim razumijevanjem rada platforme i njezinih funkcionalnosti mogu se značajno nadograditi ili čak iznova napisati skripte koje bi vjernije prenosile podatke o spojevima i ograničenjima u VR okruženje Unityja. Optimizacijom ovog prijenosa otvorio bi se put do izrade aplikacije za analizu CAD modela unutar Unity VR okruženja, olakšavajući tako proces razvoja i interakciju sa CAD modelima.

9. LITERATURA

[1] B. R. Hunde, A. D. Woldeyohannes: Future prospects of computer-aided design (CAD) – A review from the perspective of artificial intelligence (AI), extended reality, and 3D printing, Results in Engineering, Volume 14, 2022

[2] M. Lorenz, M. Spranger, T. Riedel, F. Pürzel, V. Wittstock, P. Klimant: CAD to VR – A Methodology for the Automated Conversion of Kinematic CAD Models to Virtual Reality, Procedia CIRP, Volume 41, 2016,

[3] I. Stroud, H. Nagy: Solid modelling and CAD systems: How to survive a CAD system. Springer Science & Business Media, 2011.

[4] G. H Choi, M. Duhwan, H. Soonhung: Exchange of CAD Part Models Based on the Macro-Parametric Approach; 2002.

[5] https://www.solidworks.com/partner-product/formatworks, 08.10.2023., 19:35

[6] https://cadexchanger.com/, 19.08.2024., 21:11

[7] https://www.freecad.org/, 10.10.2023., 17:15

[8] https://www.solidworks.com/sw/support/api-support.htm, 10.10.2023., 17:20

[9] A. Krishnamurthy, R. Khardekar, S. McMains: "Direct Evaluation of NURBS Curves and Surfaces on the GPU." In Proceedings of the 2007 ACM symposium on Solid and physical modeling, pp. 329-334, 2007.

[10] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf: „3. Polygon Triangulation“, Computational Geometry (2nd ed.), Springer-Verlag, pp. 45–61, 2000.

[11] H. Wallach: Visual Representation of CAD Constraints, Newnham College; 2001.

[12] https://docs.blender.org/manual/en/latest/animation/armatures/bones/editing/naming.html, 06.02.2024. 12:30

[13] https://help.solidworks.com/2023/english/api/sldworksapiprogguide/Welcome.htm, 20.01.2024., 03:10

[14] https://help.solidworks.com/2023/english/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks_namespace.html?id=838af00b66c446a2b8a39169d9ec2436#Pg0, 21.01.2024, 09:15

[15] https://en.wikipedia.org/wiki/Visual_Studio, 21.01.2024., 09:45

[16] https://en.wikipedia.org/wiki/.NET_Framework, 21.01.2024., 10:20

[17] https://en.wikipedia.org/wiki/Windows_Forms, 21.01.2024., 12:10

[18] https://venturebeat.com/pc-gaming/unity-5-6-launches-with-support-for-vulkan-graphics-nintendo-switch-and-more/, 21.01.2024., 14:00

[19] https://grabcad.com/library/pillar-drill-all-parts-1, 05.07.2024., 21:20

[20] https://grabcad.com/library/kuka-kr16-industrial-robotic-arm-1, 05.07.2024., 21:25

[21] https://grabcad.com/library/car-seat-design-1, 05.07.2024., 21:25

[22] https://en.wikipedia.org/wiki/Navigation_mesh, 22.01.2024., 23:40

[23] https://learn.microsoft.com/en-us/dotnet/visual-basic/programming-guide/com-interop/introduction-to-com-interop, 26.01.2024., 20:00

10. PRILOZI

I. CD-R disc

II. Kod datoteka

sldwrSingleton.cs kod

Otvara novu ili dohvaća postojeću instancu SolidWorksa.

```
using System;
using System.Threading.Tasks;
using SolidWorks.Interop.sldworks;

namespace MateTransfer
{
    internal class sldwrSingleton
    {
        private static SldWorks? swApp;

        internal async static Task<SldWorks> getApplication()
        {
            if (swApp == null)
            {
                return await Task.Run(() =>
                {
                    swApp =
Activator.CreateInstance(Type.GetTypeFromProgID("Sldworks.Application")) as SldWorks;
                    swApp.Visible = true;

                    return swApp;
                });

            }
            return swApp;
        }

        internal static void Dispose()
        {
            if (swApp != null)
            {
                swApp.ExitApp();
                swApp = null;
            }
        }
    }
}
```

getMates.cs kod

Zapisuje podatke o ograničenjima i spojevima u CAD modelu u CSV formatu.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Threading.Tasks;
using SolidWorks.Interop.sldworks;

namespace MateTransfer
{
    internal class getMates
    {
        public Form1 form { get; set; }
        public async Task mates(ModelDoc2 model)
        {
            string fileName = @"C:\Users\NoaG\Desktop\zavrsni\mates.csv";
            if (File.Exists(fileName))
            {
                File.Delete(fileName);
            }

            List<string> mates = new List<string>();
            ModelDoc2 swModel;
            AssemblyDoc swAssembly;
            Feature swFeat = default(Feature);
            Feature swMateFeat = null;
            Feature swSubFeat = default(Feature);
            Mate2 swMate;
            MateEntity2[] swMateEnt = new MateEntity2[3];
            var textBox = form.textBox1;

            object[] Components = null;
            int i = 0;

            swModel = model;
            swAssembly = (AssemblyDoc)swModel;
            Components = (Object[])swAssembly.GetComponents(false);
            swFeat = (Feature)swModel.FirstFeature();

            while (swFeat != null)
            {
                if ("MateGroup" == swFeat.GetTypeName())
                {
                    swMateFeat = swFeat;
                    break;
                }
                swFeat = swFeat.GetNextFeature();
            }

            swSubFeat = swMateFeat?.GetFirstSubFeature();

            using (StreamWriter tw = File.CreateText(fileName))
            {
                while (swSubFeat != null)
                {
                    swMate = (Mate2)swSubFeat.GetSpecificFeature2();
                    if (swMate != null)
                    {
                        swMateEnt[0] = swMate.MateEntity(0);
                        swMateEnt[1] = swMate.MateEntity(1);
                        string firstName = swMateEnt[0].ReferenceComponent.Name2;
                        string lastName = swMateEnt[1].ReferenceComponent.Name2;
```

```
                        if (firstName.Contains("/"))
                        {
                            firstName = firstName.Substring(firstName.IndexOf("/") +
1);
                        }
                        else if (lastName.Contains("/"))
                        {
                            lastName = lastName.Substring(lastName.IndexOf("/") + 1);
                        }

                        await tw.WriteAsync(firstName + ",");
                        await tw.WriteAsync(lastName + ",");
                        await tw.WriteLineAsync(swSubFeat.Name);

                        textBox.AppendText("    " + swSubFeat.Name + "\r\n");
                        textBox.AppendText("      Component         = " + firstName +
"\r\n");
                        textBox.AppendText("      Mated Component   = " + lastName +
"\r\n");
                        textBox.AppendText("      Type              = " + swMate.Type
+ "\r\n");
                    }
                    swSubFeat = swSubFeat.GetNextSubFeature();
                }
            }
        }
    }
}
```

Form1.cs kod

Povezuje funkcije getMates.cs i sldwrSingleton.cs datoteka sa korisničkim sučeljem.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SolidWorks.Interop.sldworks;

namespace MateTransfer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        SldWorks swApp;

        private async void button1_Click(object sender, EventArgs e)
```

```
        {
            swApp = await sldwrSingleton.getApplication();
        }

        private async void button2_Click(object sender, EventArgs e)
        {
            getMates getmates = new getMates();
            getmates.form = this;
            await getmates.mates((ModelDoc2)swApp.ActiveDoc);
        }

    }
}
```

CSVProcessor.cs kod

Učitava CSV datoteku generiranu .NET aplikacijom i pretvara je u C# listu.

```
using UnityEngine;
using System.IO;
using System.Collections.Generic;

public class CSVProcessor : MonoBehaviour
{
    public GameObject rootObject;
    public string csvFilePath = "Assets/mates.csv";
    private List<string[]> csvData;

    public void LoadCSV()
    {
        rootObject = gameObject;
        csvData = new List<string[]>();

        try
        {
            using (StreamReader reader = new StreamReader(csvFilePath))
            {
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] values = line.Split(',');
                    csvData.Add(values);
                }
            }

            Debug.Log("CSV Loaded Successfully!");
        }
        catch (System.Exception e)
        {
            Debug.LogError("Failed to load CSV: " + e.Message);
        }
    }
    public List<string[]> GetCSVData()
    {
        return csvData;
    }
}
```

CSVProcessorEditor.cs kod

Interpretira listu generiranu CSVProcessor.cs skriptom i primjenjuje zglobove na komponentne ograničenja i spojeva.

```
using UnityEditor;
using UnityEngine;
using System;
using System.Linq;

[CustomEditor(typeof(CSVProcessor))]
public class CSVProcessorEditor : Editor
{
    public MateLibrary mateLibrary;
    private GameObject firstComp;
    private GameObject secondComp;
    Rigidbody firstCompRigidBody;
    Rigidbody secondCompRigidBody;
    public override void OnInspectorGUI()
    {
        DrawDefaultInspector();

        CSVProcessor script = (CSVProcessor)target;

        if (GUILayout.Button("Load CSV"))
        {
            script.LoadCSV();
        }

        if (GUILayout.Button("Apply CSV Data to Children"))
        {
            ApplyCSVDataToChildrenWithDialog(script);
        }

        void ApplyCSVDataToChildrenWithDialog(CSVProcessor script)
        {
            if (script.rootObject == null)
            {
                Debug.LogError("No root object specified.");
                return;
            }

            var rootObject = script.rootObject;
            var csvData = script.GetCSVData();

            if (csvData == null || csvData.Count == 0)
            {
                Debug.LogError("No data loaded from CSV.");
                return;
            }

            mateLibrary = rootObject.GetComponent<MateLibrary>();

            if (csvData == null || csvData.Count == 0)
            {
                Debug.LogError("No data loaded from CSV.");
                return;
            }
```

```
            if (rootObject == null)
            {
                Debug.LogError("No root object specified.");
                return;
            }

            Rigidbody rootObjectRigidBody = rootObject.AddComponent<Rigidbody>();
            rootObjectRigidBody.useGravity = false;
            rootObjectRigidBody.isKinematic = false;

            for (int i = 0; i < csvData.Count; i++)
            {
                string[] row = csvData[i];

                string firstComponent = row[0];
                string secondComponent = row[1];
                string mateName = row[2];

                string mateNameDenumbered = new String(mateName.Where(c => c != '-' &&
(c < '0' || c > '9')).ToArray());

                firstComp = GameObject.Find(firstComponent);
                secondComp = GameObject.Find(secondComponent);

                if (firstComp.GetComponent<Rigidbody>() == null)
                {
                    firstCompRigidBody = firstComp.AddComponent<Rigidbody>();
                    firstCompRigidBody.useGravity = false;
                }

                if (secondComp.GetComponent<Rigidbody>() == null)
                {
                    secondCompRigidBody = secondComp.AddComponent<Rigidbody>();
                    secondCompRigidBody.useGravity = false;
                }

                mateLibrary.selectedGameObject = firstComp;
                mateLibrary.connectedGameObject = secondCompRigidBody;

                if (firstComponent != null)
                {
                    string coincidentChosen = null;
                    int axisChange = EditorUtility.DisplayDialogComplex("Odabir osi
ogranicenja ili spoja",
                                                                        $"Oko koje osi je
potrebno ograničiti gibanje elementa '{firstComponent}' sa '{secondComponent}' za spoj
ili ograničenje {mateNameDenumbered}?",
                                                                        "X",
                                                                        "Y",
                                                                        "Z");
                    if (mateNameDenumbered == "Coincident")
                    {
                        int coincidentChoice1 =
EditorUtility.DisplayDialogComplex("Vrsta koincidentnosti",
                                                                        $"Koji geometrijski
element je u dodiru za spoj '{firstComponent}' i  '{secondComponent}'?",
                                                                        "Tocka",
                                                                        "Crta",
                                                                        "Ravnina");
                        if (coincidentChoice1 == 0)
                        {
                            coincidentChosen = "tocka";
```

```
                        }
                        else if (coincidentChoice1 == 2)
                        {
                            coincidentChosen = "ravnina";
                        }
                        else if (coincidentChoice1 == 1)
                        {
                            int coincidentChoice2 =
EditorUtility.DisplayDialogComplex("Vrsta koincidentnosti za crtu",
                                                                $"Vrsta crte u
spoju '{firstComponent}' i  '{secondComponent}'?",
                                                                "Pravac",
                                                                "Kruznica",
                                                                "Krivulja");
                            if (coincidentChoice2 == 0)
                            {
                                coincidentChosen = "pravac";
                            }
                            else if (coincidentChoice2 == 2)
                            {
                                coincidentChosen = "krivulja";
                            }
                            else
                            {
                                coincidentChosen = "kruznica";
                            }
                        }
                    }

                    if (axisChange == 0)
                    {
                        mateLibrary.primaryAxis = Vector3.right;
                        mateLibrary.SetupJoint(mateNameDenumbered, coincidentChosen);
                    }
                    else if (axisChange == 1)
                    {
                        mateLibrary.primaryAxis = Vector3.up;
                        mateLibrary.SetupJoint(mateNameDenumbered, coincidentChosen);
                    }
                    else if (axisChange == 2)
                    {
                        mateLibrary.primaryAxis = Vector3.forward;
                        mateLibrary.SetupJoint(mateNameDenumbered, coincidentChosen);
                    }

                }
            }
        }
    }
}
```

MateLibrary.cs kod

Sadrži konfiguracije zglobova koje je potrebno postaviti i primjenjuje ih kada su pozvane od strane CSVProcessorEditor.cs skripte.

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;

public class MateLibrary : MonoBehaviour
{
    public GameObject selectedGameObject;
    public Rigidbody connectedGameObject;

    public Vector3 primaryAxis;

    private ConfigurableJoint configurableJoint;

    public void SetupJoint(string selectedMateType, string geometricType)
    {
        configurableJoint = selectedGameObject.AddComponent<ConfigurableJoint>();
        configurableJoint.connectedBody = connectedGameObject;

        if (primaryAxis != null)
        {
            configurableJoint.axis = primaryAxis;
        }

        switch (selectedMateType)
        {
            case "Concentric":
                SetupConcentricJoint();
                break;
            case "Coincident":
                SetupCoincidentJoint(geometricType);
                break;
            case "Tangent":
                SetupTangentJoint();
                break;
            case "Parallel":
                SetupParallelJoint();
                break;
            case "Perpendicular":
                SetupPerpendicularJoint();
                break;
            case "Lock":
                SetupLockJoint();
                break;
            case "Distance":
                SetupDistanceJoint();
                break;
            case "Angle":
                SetupAngleJoint();
                break;
            case "ProfileCenter":
                SetupProfileCenterJoint();
                break;
        }
    }

    void SetupConcentricJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Free;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
        configurableJoint.xMotion = ConfigurableJointMotion.Free;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;
```

```
    }

    void SetupCoincidentJoint(string geometricType)
    {
        if (geometricType == "pravac")
        {
            configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
            configurableJoint.xMotion = ConfigurableJointMotion.Free;
            configurableJoint.yMotion = ConfigurableJointMotion.Locked;
            configurableJoint.zMotion = ConfigurableJointMotion.Locked;
        }
        else if (geometricType == "tocka")
        {
            configurableJoint.angularXMotion = ConfigurableJointMotion.Free;
            configurableJoint.angularYMotion = ConfigurableJointMotion.Free;
            configurableJoint.angularZMotion = ConfigurableJointMotion.Free;
            configurableJoint.xMotion = ConfigurableJointMotion.Locked;
            configurableJoint.yMotion = ConfigurableJointMotion.Locked;
            configurableJoint.zMotion = ConfigurableJointMotion.Locked;
        }
        else if (geometricType == "kruznica")
        {
            configurableJoint.angularXMotion = ConfigurableJointMotion.Free;
            configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
            configurableJoint.xMotion = ConfigurableJointMotion.Locked;
            configurableJoint.yMotion = ConfigurableJointMotion.Locked;
            configurableJoint.zMotion = ConfigurableJointMotion.Locked;
        }
        else if (geometricType == "krivulja")
        {
            configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
            configurableJoint.xMotion = ConfigurableJointMotion.Locked;
            configurableJoint.yMotion = ConfigurableJointMotion.Locked;
            configurableJoint.zMotion = ConfigurableJointMotion.Locked;
        }
        else if (geometricType == "ravnina")
        {
            configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
            configurableJoint.angularZMotion = ConfigurableJointMotion.Free;
            configurableJoint.xMotion = ConfigurableJointMotion.Free;
            configurableJoint.yMotion = ConfigurableJointMotion.Free;
            configurableJoint.zMotion = ConfigurableJointMotion.Locked;
        }
    }

    void SetupTangentJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Free;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
        configurableJoint.xMotion = ConfigurableJointMotion.Free;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;
    }

    void SetupParallelJoint()
```

```
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Free;
        configurableJoint.xMotion = ConfigurableJointMotion.Free;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;
    }

    void SetupPerpendicularJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Free;
        configurableJoint.xMotion = ConfigurableJointMotion.Free;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;
    }

    void SetupLockJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
        configurableJoint.xMotion = ConfigurableJointMotion.Locked;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;
    }

    void SetupDistanceJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
        configurableJoint.xMotion = ConfigurableJointMotion.Limited;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;

        SoftJointLimit limit = new SoftJointLimit();
        limit.limit = 1.0f;
        configurableJoint.linearLimit = limit;
    }

    void SetupAngleJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Limited;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;

        SoftJointLimit angularLimit = new SoftJointLimit();
        angularLimit.limit = 45.0f;
        configurableJoint.lowAngularXLimit = angularLimit;
        configurableJoint.highAngularXLimit = angularLimit;
    }

    void SetupProfileCenterJoint()
    {
        configurableJoint.angularXMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularYMotion = ConfigurableJointMotion.Locked;
        configurableJoint.angularZMotion = ConfigurableJointMotion.Locked;
        configurableJoint.xMotion = ConfigurableJointMotion.Free;
        configurableJoint.yMotion = ConfigurableJointMotion.Locked;
```

```
        configurableJoint.zMotion = ConfigurableJointMotion.Locked;
    }
}
```