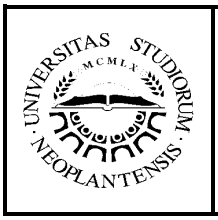


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Михаило Трифковић**
Број индекса: **РА 36-2019**

Тема рада: **Имплементација подршке за проширену реалност у циљу надгледања и интеракције са виртуелним објектима**

Ментор рада: **проф. др Илија Башичевић**

Нови Сад, Јул, 2023

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Завршни (Bachelor) рад
Аутор, **АУ**:		**Михаило Трифковић**
Ментор, **МН**:		**проф. др Илија Башичевић**
Наслов рада, **НР**:		**Имплементација подршке за проширену реалност у циљу надгледања и интеракције са виртуелним објектима**
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		**2023**
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**7/49/0/20/18/0/0**
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		**Проширена реалност, клијент сервер комуникација, интеракција са објектима, приказ података у реалном времену**
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		**У овом раду описана је имплементација једне апликације за проширену реалност. Апликација приказује за шта је све способна проширена и виртуелна реалност, као и ккоје су могућности ових технологија. Описани су уређаји који се користе за покретање ове и сличних апликација. Главни алат коришћен за имплементацију је *Unity* програмско окружење за развој игара.**
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		14.07.2023.
Чланови комисије, **КО**:	Председник:	проф. др Мирослав Поповић
	Члан:	проф. др Иван Каштелан
	Члан, ментор:	проф. др Илија Башичевић

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Bachelor Thesis	
Author, **AU**:		**Mihailo Trifković**	
Mentor, **MN**:		**Ilija Bašičević, PhD**	
Title, **TI**:		**Implementation of support for augmented reality for monitoring and interaction with virtual objects**	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:			
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/49/0/20/18/0/0**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Augmented reality, client-server communication, object interaction, display of data in real time	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		**This paper describes the implementation of an application for augmented reality. The application shows what augmented and virtual reality are capable of, as well as the possibilities of these technologies. The devices used to run this, and similar applications are described. The main tool used for implementation is the Unity game engine.**	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	**Miroslav Popović, PhD**	
	Member:	**Ivan Kaštelan, PhD**	Menthor's sign
	Member, Mentor:	**Ilija Bašičević, PhD**	

Захвалност

Захваљујем се члановима породице и пријатељима на пруженој подршци током студирања.

Захваљујем се ментору проф. др Илији Башичевићу, Саши Јагодину и Михајлу Милотићу, као и колегама из тима на стручној помоћи и саветима током израде рада.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

SDK	*Software developer kit*
VR	*Virtual Reality*
AR	*Augmented reality*
XR	*Extended Reality*
OLED	*Organic light-emitting diode*
USB	*Universal Serial Bus*
6DOF	*6 Degrees of Freedom*
MRTK	*Mixed Reality Toolkit*
JSON	*JavaScript Object Notation*
TCP	*Transmission Control Protocol*
HTTP	*HyperText Transfer Protocol*
URL	*Uniform Resource Locator*
UI	*User Interface*
ADB	*Android Debug Bridge*

1. Увод

Проширена реалност је технологија која се последњих година убрзано развија, а могућности које нам пружа у будућности су заиста узбудљиве. Проширена реалност се развија преко различитих уређаја као што су паметни телефони, наочаре и хедсети. Сензори у овим уређајима прате окружење и омогућавају пројекцију виртуелних елемената на реални свет. Проширена реалност омогућава истраживање и интеракцију са тим виртуелним елементима.

Једна од могућности проширене реалности је унапређена интеракција са виртуелним светом. Користећи руке и глас, корисници ће моћи да комуницирају са виртуелним објектима и добијају реалније искуство.

У будућности можемо очекивати да ће проширена реалност бити уграђена у неке алате и уређаје које свакодневно користимо. Можемо замислити да имамо наочаре које ће нам приказивати информације о предметима које гледамо или на пример приказује путоказе док шетамо, као неки пример навигације. Управо имплементација једне такве апликације је задатак овог рада.

Задатак овог рада је имплементација апликације за проширену реалност која омогућава посматрање неких виртуелних објеката, у овом случају једне виртуелне машине, и интеракцију са тим виртуелним објектима, као и интеракцију са корисничком спрегом преко којег добављамо и приказујемо податке о тој машини. Податке је потребно добављати са сервера и ажурирати их на корисничкој спрези у реалном времену тако да у сваком тренутку знамо тренутно стање и вредности неких параметара те машине. Потребно је омогућити померање машине по просторији као и саме корисничке спреге, ротирање машине и промену величине машине. На

корисничкој спрези је потребно имплементирати дугмад чијим притиском се приказују подаци везани за машину, ресетује позиција машине, као и клизача чијим померањем се машина ротира око своје осе. Поред наведених функционалности треба имплементирати дугме чијим притиском се покреће анимација, која симулира рад машине.

Полазну тачку у раду представља подешавање окружења за имплементацију ове апликације. Главни алат који се користи за израду ове апликације је *Unity* програмско окружење за развој рачунарских игара са неким својим пратећим модулима и радним окружењим који ће бити описани у наставку рада.

Рад се састоји из 6 поглавља.

У другом поглављу су описане теоријске основе на којима је базирана реализација решења. У овом поглављу су описане неке најбитније ствари кључне за разумевање функционалности саме апликације.

У трећем поглављу је описан концепт решења.

Четврто поглавље описује само програмско решење, док пето описује начин тестирања као и неке резултате.

Шесто и последње поглавље представља закључак где су описана нека будућа унапређења апликације и могућности прављења сличних апликација.

2. Теоријске основе

У овом поглављу су описане теоријске основе на којима се заснива имплементација апликације коју описује овај рад.

Описани су главни алати који су се користили за имплементацију, као и нека алтернативна решења за те алате.

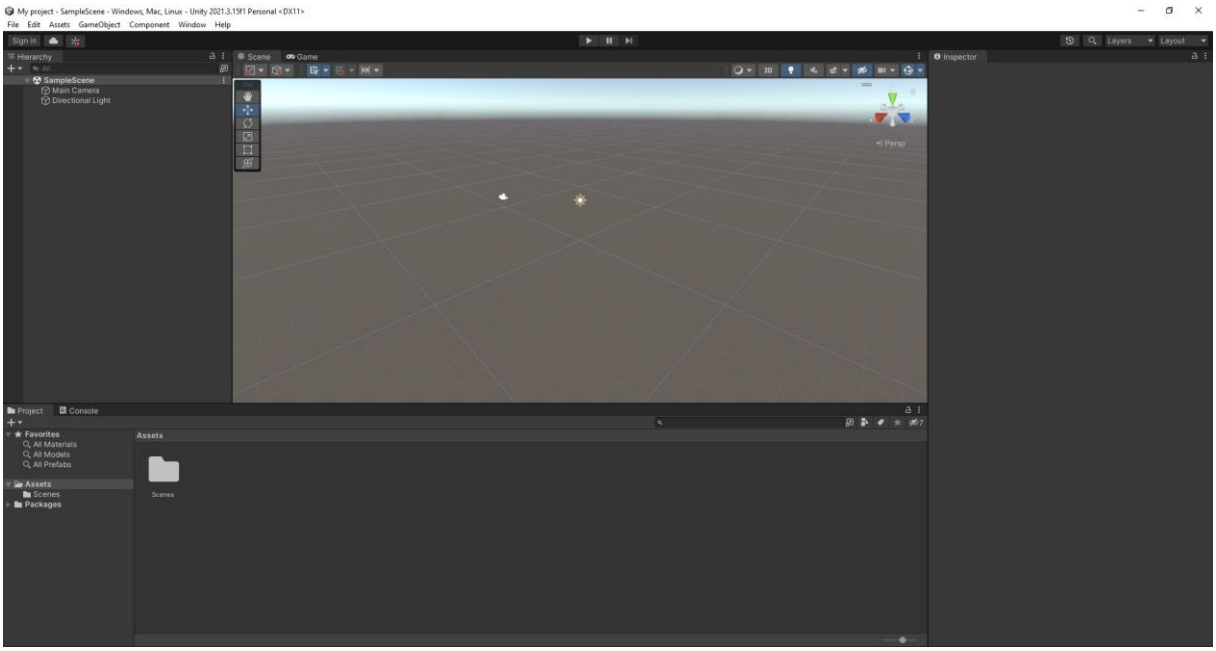
Објашњена су нека теоријска знања везана за проширену, као и виртуелну реалност, неопходна за разумевање саме функционалности ове и сличних апликација.

Такође описане су платформе које подржавају рад са овим технологијама.

2.1 *Unity*

Unity програмско окружење за развој рачунарских игара је моћан и популаран алат за креирање различитих видова интерактивних апликација, игара и симулација. Он представља интегрисану развојну средину која омогућава креирање визуелно привлачних и интерактивних дигиталних садржаја.

Unity је изграђен на бази компонентног модела, што омогућава лаку и флексибилну развојну средину. Користећи *Unity*, програмери могу користити визуелни спрегу за креирање сцена и објеката и лако придруживати скрипте и компоненте да би доделили жељено понашање сваком објекту. То омогућава брз развој игара и апликација.



Слика 2.1 Изглед *Unity* едитора

Unity подржава велики број платформи, укључујући рачунаре, конзоле, мобилне уређаје и виртуелну, као и проширену реалност. Ово значи да један пројекат или једна апликација имплементирана у *Unity-y* може бити компајлиран и покренут на различитим уређајима без великих промена у коду. Овај аспект *Unity-a* значајно олакшава развој и портабилност апликација на различите платформе.

Једна од најистакнутијих карактеристика *Unity-a* је његов визуелни систем за развој игара. Он има широк спектар алатки и ресурса за креирање визуелних ефеката, анимација, осветљења, физике и још много тога. Кроз визуелни интерфејс, програмери могу креирати игрице и апликације које изгледају реално и привлачно, без потребе за писањем комплексног кода за сваки детаљ.

Unity се може сматрати универзалном платформом за развој игара, што значи да иако је његов основни сет алата можда усмерен ка стварању одређених врста садржаја, изграђен је да пружи подршку за различите пројекте, дизајнерске методологије и радне токове производње [1].

Unity такође има велику заједницу програмера која активно дели знање, ресурсе и пакете. Постоји мноштво документације, туторијала и форума који подржавају развој са *Unity* програмским окружењем. Ова заједница омогућава програмерима да уче, деле и унапређују своје вештине у развоју апликација.

Unity користи *Microsoft Visual Studio* као развојно окружење, као и *C#* програмски језик, стога је за имплементацију ове апликације коришћен *C#* програмски језик.

Укратко, *Unity* је платформа за развој рачунарских игара који омогућава креирање различитих видова интерактивних апликација и игара. Са његовим визуелним системом за развој и подршком за велики број платформи, *Unity* представља одличан избор за професионалне програмере, а и за програмере почетнике који желе да стварају сопствене интерактивне игрице и апликације.

2.2 Андроид

Андроид оперативни систем је оперативни систем заснован на Линукс језгру и развијен од стране *Google-a*. Он је највише познат као оперативни систем који покреће мобилне уређаје као што су паметни телефони и таблети, али такође се користи и на другим уређајима као што су телевизори, носиве технологије и аутомобили.

Андроид оперативни систем је пројектован да буде отворен и флексибилан, што значи да је доступан за широк круг произвођача и програмера. Ово је довело до велике разноврсности уређаја који користе Андроид, како у смислу хардверских спецификација тако и у смислу верзија оперативног система. Корисницима се пружају многобројне опције у избору уређаја који најбоље одговарају њиховим потребама и преференцијама.

Unity подржава Андроид оперативни систем и омогућава програмерима да креирају игре и апликације које могу бити покренуте на Андроид уређајима. Програмери могу развијати и тестирати своје апликације у *Unity-y*, а затим и извршити компилацију и изградњу за Андроид платформу.

Unity се интегрише са Андроид *SDK-om* (енг. *Software Developer Kit*) и омогућава развој функционалности које су специфичне за Андроид, као што су апликације базиране на сензорима, коришћење камера, управљање додиром екрана и многе друге. *Unity* такође нуди могућности за извршавање оптимизација и преносивости кода, што олакшава развој апликација за различите верзије и моделе Андроид уређаја.

Коришћењем *Unity-a* у комбинацији са Андроид оперативним системом, програмери имају моћан алат за креирање интерактивних апликација и игрица које могу достићи велики број корисника широм света који користе Андроид уређаје.

Разлог због којег смо од свих платформи фокусирани на Андроид јесте зато што апликација, чијом се имлементацијом бавимо, се прави за *Quest Pro* наочаре на коме се налази Андроид оперативни систем.

2.3 Проширена реалност (AR)

Проширену реалност можемо укратко дефинисати као директни или индиректни поглед на физичко окружење у реалном времену, које је допуњено додавањем виртуелних информација генерисаних на рачунару [2].

Проширена реалност је технологија која спаја реални свет са виртуелним елементима, пружајући корисницима могућност да виде исписане информације, симболе, графике или 3Д објекте у свом реалном окружењу. Она пружа додатну информацију и интеракцију у реалном времену, преко уређаја као што су паметни телефони, наочаре и хедсети.

Основна идеја иза технологије проширене реалности је да прошири перцепцију и искуство света око нас. Користећи сензоре и камере у уређајима, проширена реалност прецизно прати нашу околину, тј. реални свет. Користећи рачунарске алгоритме, виртуелни елементи се пројектују и постављају у наше стварно окружење. То значи да можемо видети исписане информације на зидовима, симболе или 3Д објекте који се појављују испред нас у некој соби у којој се налазимо у датом тренутку.

Проширена реалност има широк спектар примена у различитим областима. На пример, у медицини, лекари могу користити AR за добављање додатних информација о стању пацијента преко сопствених наочара. У образовању, ученици могу искусити неке историјске догађаје кроз виртуелне репрезентације или у случају већ поменуте медицине може се интерактивно изучавати анатомија човековог тела на пример. У трговини, AR се може користити за виртуелну пробу одеће или уређење неке просторије и постављање намештаја у њој и на тај начин се може олакшати доношење одлуке о куповини без одласка у продавницу. Такође у маркетингу се AR може користити за креирање интерактивних рекламних кампања и виртуелних искустава са неким производима.

Генерално гледано, циљ технологије проширене реалности је да наш реални свет постане интерактивнији и информативнији.

На слици 2.2 се може видети пример AR апликације примењене на телефону која показује где се које одредиште налази и колика је удаљеност до њих.

Слика 2.2 Пример АР апликације на телефону

Слика 2.3 АР апликација на хедсету

На слици 2.3 може се видети пример АР апликације на хедсету.

Близак појам проширеној реалности је и виртуелна реалност. Стога потребно је навести неке разлике између те две технологије како не би дошло до забуне која технологија шта користи.

2.3.1 Разлике између проширене и виртуелне реалности

Разлике између проширене реалности (AP) и виртуелне реалности (ВР) су у начину на који интерагујемо са окружењем:

1. Виртуелна реалност: ВР технологија потпуно урања корисника у виртуелни свет и искључује га из реалног окружења. Корисник носи хедсет који приказује цео виртуелни свет пред њим и онемогућен је да види реални свет. Корисник може да истражује и интерактивно комуницира у окружењу које је потпуно виртуелно.
2. Проширена реалност: AP технологија додаје виртуелне елементе у постојеће реално окружење. Корисник не губи поглед на реални свет, већ га додатни виртуелни елементи допуњују. Корисник може видети пројекције информација , објеката или графике који се приказују у реалном окружењу и интерактивно комуницирати са њима.

Још неке од разлика између AP и ВР:

- Виртуелна реалност захтева специјалну опрему у виду хедсета, док је проширена реалност доступна и на уређајима као што су паметни телефони
- Виртуелна реалност се најчешће користи за игрице и свеобухватно искуство, док AP има широку примену у различитим областима као што су образовање, тренинг, медицина, маркетинг и слично.
- У виртуелној реалности, корисник је изолован у виртуелном свету, док у проширеној реалности корисник остаје у контакту са реалним окружењем.

2.4 Подржане платформе од стране AP

AP је подржан на многим платформама. Неке од често коришћених уређаја тј. платформи за приказ проширене реалности ће бити приказани и објашњени у овом поглављу.

2.4.1 Мобилни уређаји

Примена проширене реалности на телефону открива могућности које нам омогућавају да интерагујемо са дигиталним садржајем у реалном окружењу. Телефони са AP функционалностима користе камеру, сензоре и процесоре како би препознали

окружење и на тој основи додали виртуелне елементе који се приказују на екрану телефона.

Неки примери примене АР на телефону:

- Навигација: Апликације за навигацију могу користити АР да прикажу смерове и путање кретања директно на екрану телефона. На пример, док гледамо кроз камеру свог телефона, на екрану ће нам бити приказане стрелице које указују на смер кретања.
- Игре: На мобилном телефону се такође могу покретати и игрице направљене за проширену реалност. *Pokemon GO* је пример једне веома популарне игрице која користи АР за приказивање виртуелних ликова и објеката у нашем реалном окружењу.
- Визуализација производа: Производне компаније и дизајнери могу користити АР за визуализацију производа. На пример, можемо скенирати каталог или физички предмет и користити АР апликацију да видимо како би производ изгледао у нашем окружењу.
- Употреба у медицини: АР се може користити у медицини за сврхе приказивања виртуелних слика или додатних информација о пацијенту током операција.

Један пример изгледа АР апликације на телефону се може видети на Слици 2.2.

2.4.2 *Nreal* наочаре

Nreal наочаре представљају нови производ на тржишту наочара за проширену реалност. Постоје два типа ових наочара, то су *Nreal Air* и *Nreal Light*.

Nreal Air су наочаре за АР чија је главна могућност пројекција виртуелног "биоскопа". Овај термин је искоришћен јер ове наочаре, када су прикључене на компатибилан уређај, праве пројекцију екрана тог уређаја испред нас. Стога се садржај екрана може видети испред нас као да смо у биоскопу. Под компатибилним уређајем се подразумева Андроид мобилне телефоне, као и *iPhone* телефоне. Користе *OLED* панеле за приказ садржаја екрана повезаног уређаја.

Nreal Air наочаре немају камере, стога има ограничене АР функционалности и ограничен је само на тај "биоскопски" доживљај.

Битно је напоменути да постоји ограничен број компатибилних телефона на које ове наочаре могу бити повезане. Такође да би ове наочаре функционисале, потребно је да константно буду повезане на телефон.

Nreal Light наочаре су наочаре које се много више могу назвати АР наочарима у односу на *Nreal Air*. Ове наочаре имају две камере на себи што омогућава да се са њима може детектовати околина. Само постојање камера на наочарима нам даје много већи спектар АР могућности и функционалности. Помоћу њих можемо детектовати објекте око нас и на пример добављати информације о снимљеним објектима. Такође, за разлику од *Nreal Air* наочара, *Nreal Light* не мора стално бити повезан на телефон, већ има батерију на себи.

Слика 2.4 *Nreal Light* и *Nreal Air* наочаре

На следећој табели се могу видети разлике између ових наочара.

	Nreal Air	***Nreal Light***
Батерија	-	3 сата
Видно поље	46°	52°
Резолуција	1920x1080	1920x1080
Фреквенција освежавања	90 Hz	-
Оперативни систем	Андроид	Андроид

Тежина	76г	85г
360 праћење	-	Има
Позиционо праћење	-	Има
Камера	-	Има
Микрофон	-	Има
Гласовне команде	-	Има
USB	Има	Има
Bluetooth	-	Има
Wi-Fi	-	Има

Табела 2.1 Компарација *Nreal Air* и *Nreal Light* наочара

2.4.3 *Meta Quest Pro*

Meta Quest Pro је најновији производ *Meta* компаније за приказ виртуелне реалности. Овај хедсет обезбеђује корисницима потпуно урањање у виртуелну реалност.

Meta Quest Pro има неколико значајних функција и спецификација. Може се похвалити резолуцијом од 1920x1800 по сочиву, подржава фреквенцију освежавања од 90Hz, као и *6DOF* (енг. *6 degrees of freedom*). За управљање се користе два контролера за леву и десну руку. *Meta Quest Pro* подржава детекцију руку па се самим тим, поред контролера, за управљање хедсета могу користити и руке.

Meta Quest Pro на себи има камере које омогућавају да се види реални свет када имамо хедсет на глави. Тај тренутак када излазимо из виртуелног света и улазимо у реални свет и можемо да видимо просторију, иако још имамо хедсет на глави, се назива *pass-through*. *Pass-through* је за разлику од *Oculus Meta Quest 2* у боји, стога релативно аутентично можемо да видимо своју околину. *Pass-through* ће имати битну улогу у имплементацији наше апликације, јер захваљујући њему имамо тај AP моменат у апликацији [3].

Поред камера *Meta Quest Pro* на себи има сензоре и детекторе лица. Они омогућавају да, уколико имамо свог аватара у некој апликацији на хедсету, да тај аватар опонаша наше покрете лица и очију.

Meta Quest Pro користи Андроид оперативни систем. Поред покретања ВР апликација, на хедсету се могу користити и неке стандардне апликације као што су *YouTube, Facebook, Gmail*...

Слика 2.5 *Meta Quest Pro*

2.5 *OpenXR*

OpenXR је бесплатан стандард који је развила *Khronos* група са циљем да омогући примену ВР и АР апликација на више платформи. Пружа нам радни оквир и програмску спрегу за развој апликација које могу радити на више ВР и АР уређаја, без обзира на платформу коју користе.

Сврха *OpenXR* је да реши фрагментацију и неспојивост која је постојала за ВР и АР апликације. Пре постојања *OpenXR-a*, програмери су морали имплементирати посебну верзију апликације са посебном подршком за одређену платформу.

OpenXR пружа сет основних функционалности, укључујући откривање уређаја, праћење, рендеровање, управљање улазима и интеракцију. Дефинише заједнички скуп апстракција и конвенција на које програмери могу да се ослоне, поједностављујући процес развоја и подстичући иновације. Додатно, *OpenXR* подстиче развој проширења које омогућавају произвођачима да унесу нове могућности, одржавајући уједно компатибилност са основном спецификацијом.

Укратко, *OpenXR* је стандардни API који има за циљ да успостави заједничку основу за развој ВР и АР технологија. Сврха му је да пружи уједињен интерфејс програмерима, омогућавајући им да креирају апликације које су компатибилне са различитим уређајима [4].

2.5.1 *OpenXR* и *Unity*

Интеграција *Unity* програмског окружења са *OpenXR-ом* омогућава програмерима да једноставно креирају ВР и АР апликације. *Unity* пружа *OpenXR* додатак који програмери могу увести у своје пројекте. Овај додатак делује као мост између *OpenXR* покретачког програма и *Unity-a*, омогућавајући апликацији да комуницира са различитим АР и ВР уређајима.

OpenXR апстрахује детаље о хардверу и платформи, као што су системи за праћење, улазни уређаји и конфигурације приказа. *Unity,* путем *OpenXR* додатка, омогућава приступ овим апстракцијама, омогућавајући програмерима да се фокусирају на изградњу апликације.

Интеграција *Unity* програмског окружења са *OpenXR-ом* пружа јединствени систем за управљање улазом који омогућава програмерима да обрађују улазне сигнале са различитих контролера, сензора и других уређаја. То поједностављује сваки вид интеракције корисника, као што су знакови руку, притисак дугмади или покрети главе.

АР апликације често захтевају разумевање стварног света ради прецизног смештања виртуелних објеката. *OpenXR* подржава просторно мапирање, што омогућава апликацији да прикупи информације о физичком свету као што су површине, границе и објекти.

OpenXR омогућава приступ подацима о положају и оријентацији корисникове главе, као и положају и кретању контролера или руку. Ове информације су важне за прецизно смештање и манипулацију виртуелним објектима у АР окружењу.

2.6 *MRTK*

MRTK (енг. *Microsoft Mixed Reality Toolkit*) је *framework* који је намењен олакшавању развоја апликација за ВР и АР апликације. Пружа скуп компоненти, скрипти и алатки које омогућавају програмерима да стварају свеобухватна искуства за кориснике, као и њихову примену на разним АР и ВР платформама.

MRTK подржава више платформи, као на пример *HoloLens*, *Windows Mixed Reality* наочаре и *Unity* едитор. То омогућава програмерима да циљају широк спектар уређаја без значајних модификација кода [5].

MRTK подржава јединствен систем улаза који контролише улазне информације са различитих извора као што су знакови руку, глас и контролери. Ово олакшава обраду интеракција корисника чиме се олакшава креирање интерактивних искустава.

MRTK подржава компоненте за просторно мапирање, омогућавајући програмерима да скенирају и разумеју физичко окружење. Ова функционалност

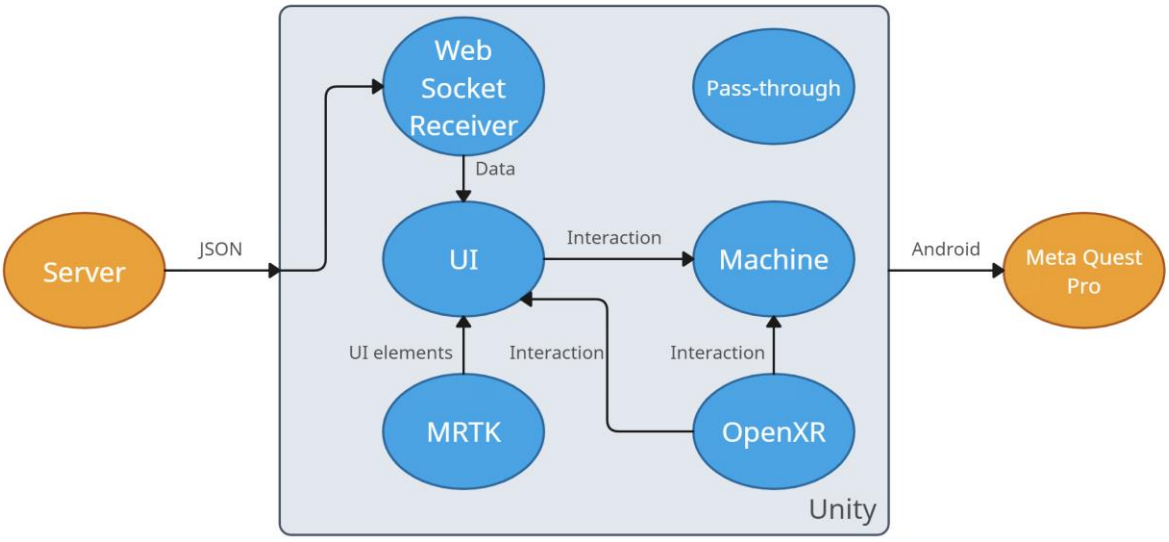
омогућава смештање виртуелних објеката у стварни свет и олакшава интеракцију између виртуелног и физичког света.

MRTK нуди функционалности за манипулацију виртуелним објектима, као што су хватање, ротирање или скалирање објеката. Ове функционалности пружају основу за омогућавање интеракције са виртуелним објектима.

MRTK пружа скуп предефинисаних градивних објеката који служе за имплементацију корисничке спреге, као и обрасце за интеракцију са њим. Предефинисани градивни објекти подразумевају дугмад, клизаче, панеле и блокове који исписују неке информације. Ови објекти олакшавају креирање корисничке спреге у 3Д простору.

3. Концепт решења

У овом поглављу биће описан концепт програмског решења апликације. На слици 3.1 приказан је изглед саме архитектуре апликације. На слици се виде најбитније компоненте апликације заслужне за могућност њене имплементације, као и шта повезује те компоненте и шта се преноси између њих.



Слика 3.1 Архитектура апликације

Архитектуру можемо разделити на 3 засебне целине. Прву представља сервер који нам представља главни извор података везани за наш објекат у сцени, тј. виртуелни модел машине. Друга целина је најкомплекснија и представља све што се ради у склопу *Unity-a*, тј дефинише се изглед апликације и њена функционалност. Трећа целина представља припрему апликације за компајлирање на наш уређај, који је у нашем

случају већ поменути *Meta Quest Pro*, као и припрема самог уређаја да успешно функционише са нашом апликацијом.

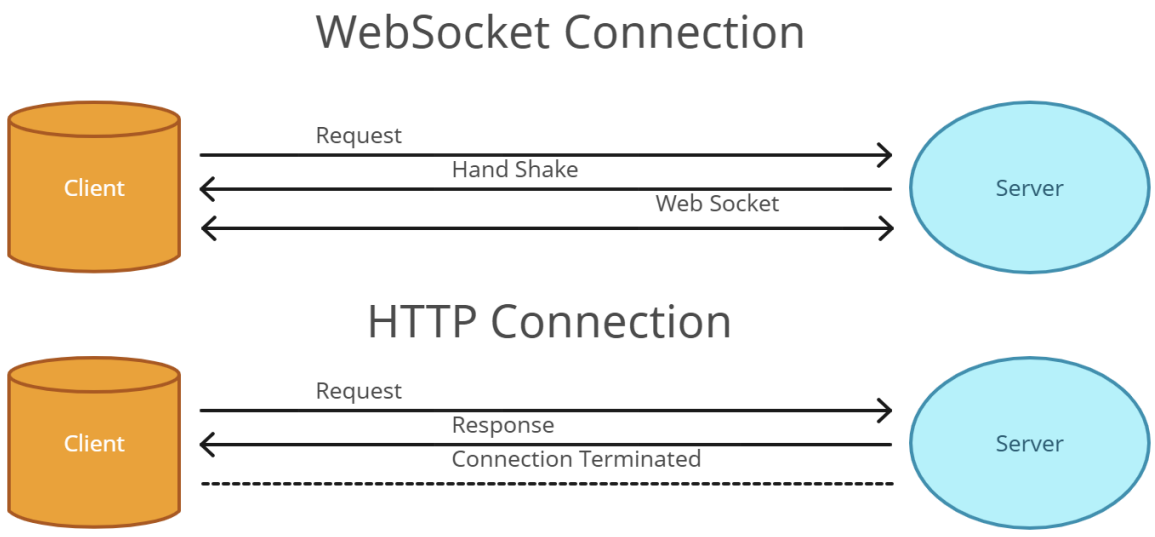
3.1 Сервер

Сервер се односи на неки рачунар или програм који прима и обрађује захтеве других уређаја или програма који се називају клијентима. Он представља централну фигуру за управљање и дистрибуцију података или услуга клијената преко мреже.

Сервер, који се користи у овој апликацији, садржи податке везане за машину чији виртуелни модел приказујемо у сцени и он те податке у форми *JSON* фајла шаље клијенту, што је у нашем случају то *WebSocketReceiver*.

Неки од података које шаље су стање машине, позиција, идентификатор грешке уколико се деси и слично. Сервер је писан у *JAVA Script* програмском језику.

Клијент, што је у нашем случају *WebSocketReceiver*, шаље захтев серверу за успостављање везе. Захтев се шаље преко *WebSocket* протокола. *WebSocket* протокол је протокол који омогућава двосмерну комуникацију преко једне *TCP* везе. *WebSocket* протокол је дизајниран како би избегао неке недостатке које *HTTP* протокол има. Са овим протоколом када се једном успостави веза између клијента и сервера, обе стране могу слати поруке без потребе за поновно слање захтева за успостављање везе.



Слика 3.2 Разлика између *HTTP* и *WebSocket* протокола

3.2 *Unity* подршка

У склопу ове целине из архитектуре апликације налази се комплетна функционалност апликације као и дефиниција њеног изгледа, мислећи при том на корисничку спрегу. У овом поглављу биће објашњена свака битнија функционалност апликације.

3.2.1 *Web Socket Receiver*

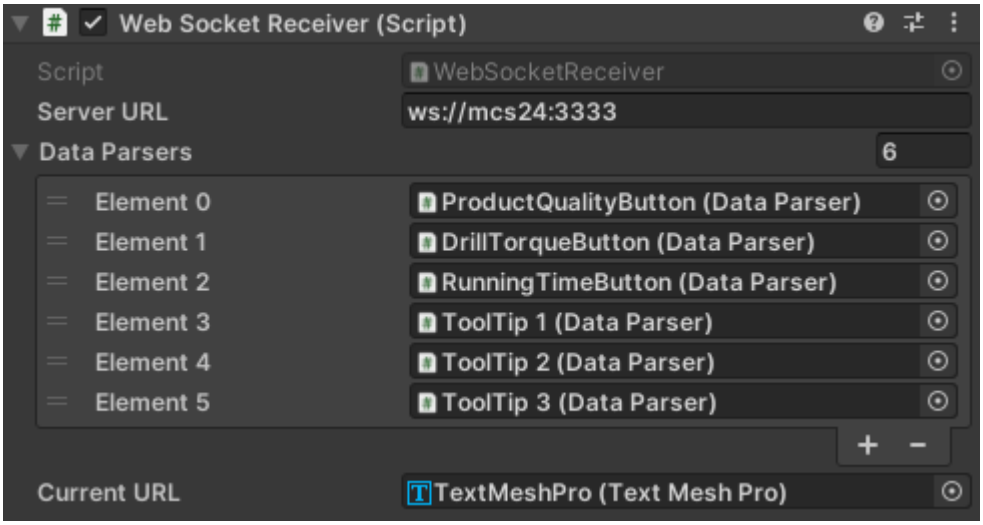
Web Socket Receiver представља клијента који шаље захтев за податке серверу. Он успоставља везу са сервером слањем захтева за успостављање везе преко *Web Socket* протокола на одговарајући *URL* сервера. *Web Socket* протокол држи везу стално успостављену како би се ажурирали подаци у реалном времену када дође до промене неких параметара.

Web Socket Receiver прима податке од сервера у облику *JSON* фајла па је *Web Socket Receiver* задужен да податке из тог *JSON* фајла обради и прилагоди корисничкој спрези тако да може приказати те податке и да их ажурира у реалном времену.

Једна техника коју *Web Socket Receiver* користи је *Data binding*. *Data binding* је техника која омогућава синхронизацију података између различитих компоненти или слојева апликације. Она омогућава слободан проток података између извора и одредишта без потребе за ручним манипулацијама при протоку. У нашем случају извор представља *JSON* фајл, а одредиште корисничка спрега. На овај начин, када дође до промене неких података, аутоматски се те промене одразе на корисничку спрегу тако што ажурира вредности параметара. Такође, уколико корисник мења неке податке на корисничкој спрези, те промене ће стићи до извора података.

Ова двосмерна синхронизација коју омогућава *Data binding* смањује потребу за ручним ажурирањем корисничке спреге или управљањем промена података. То поједностављује процес развоја и побољшава доследност апликације.

Што се тиче имплементације ове технике *Unity-y*, она је урађена на следећи начин. Како свака компонента на корисничкој спрези представља засебан објекат, *Unity* има функционалност да те објекте директно налепи на скрипту као поља скрипте. На слици 3.3 можете видети како то изгледа.



Слика 3.3 *Data binding* у *Unity-y*

Елементи од 0 до 5, као и *Current URL* представљају поља скрипте *Web Socket Receiver* и на та поља су директно превучени објекти са корисничке спреге. Шта представљају ови објекти, биће објашњено у поглављу задужен за *UI*, док ће сама скрипта бити објашњена у поглављу програмског решења.

На овај начин, како се деси промена параметара у *JSON* фајлу, те промене се аутоматски постављају на прослеђене објекте.

3.2.2 *Pass-through*

У AP апликацији *pass-through* се односи на способност уређаја, у нашем случају *Meta Quest Pro* , да прикаже реални свет кориснику из самог хедсета. *Meta Quest Pro* на себи има 4 камере које снимају спољашњи свет.

Стога, како је наш циљ да имплементирамо AP апликацију, морамо оспособити *pass-through* у нашој апликацији. То се ради додавањем већ постојеће скрипте *OVR Passthrough Layer* у сцену.

3.2.3 *MRTK*

Како је улога *MRTK* већ објашњена у теоријским основама, сада ћемо поменути неке његове елементе који се користе у апликацији.

Неке кључне компоненте које се користе из *MRTK* алата:

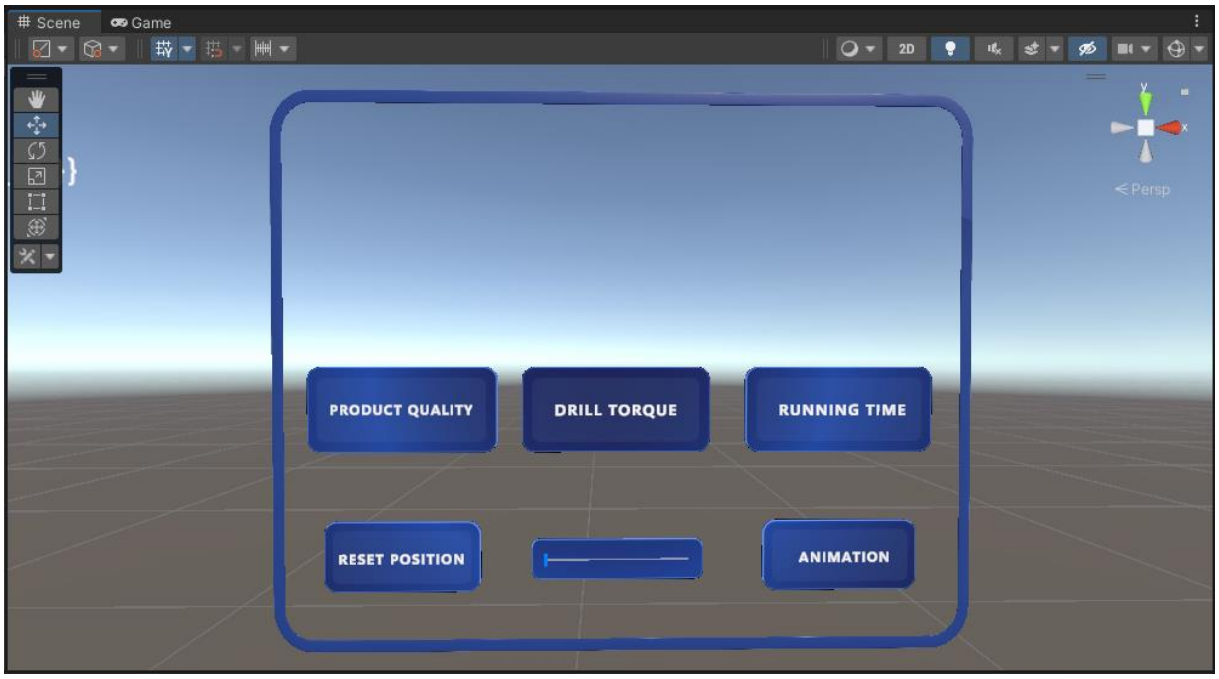
- *MRTK XR (Mixed Reality Toolkit Extended Reality) Rig*. Он је дизајниран да поједностави подешавање и конфигурацију *Unity* пројекта како би се омогућио прави доживљај виртуелне и проширене реалности код корисника. *MRTK XR Rig* делује као централни контролер за управљање

главном камером и руковање различитим захтевима и конфигурацијама различитих *XR* уређаја.

- *PieChart* (кружни дијаграм) омогућава приказивање података у облику кружног дијаграма.
- *Torque* (момент) представља ротациони приказ података, изглед је као да посматрамо кружни лук који се повећава или смањује у зависности од вредности параметара.
- *Tooltip* (описни текст) је компонента која омогућава приказ текстуалних инфорамција или упутства везана за објекте у некој апликацији.

3.2.4 *UI* (корисничка спрега)

Корисничка спрега представља главну компоненту апликације за интеракцију са корисником. Корисничка спрега је изграђен уз помоћ *MRTK* компонената описаних у претходном поглављу. Корисничка спрега је замишљена као панел на којем се налазе пет дугмади и један клизач.

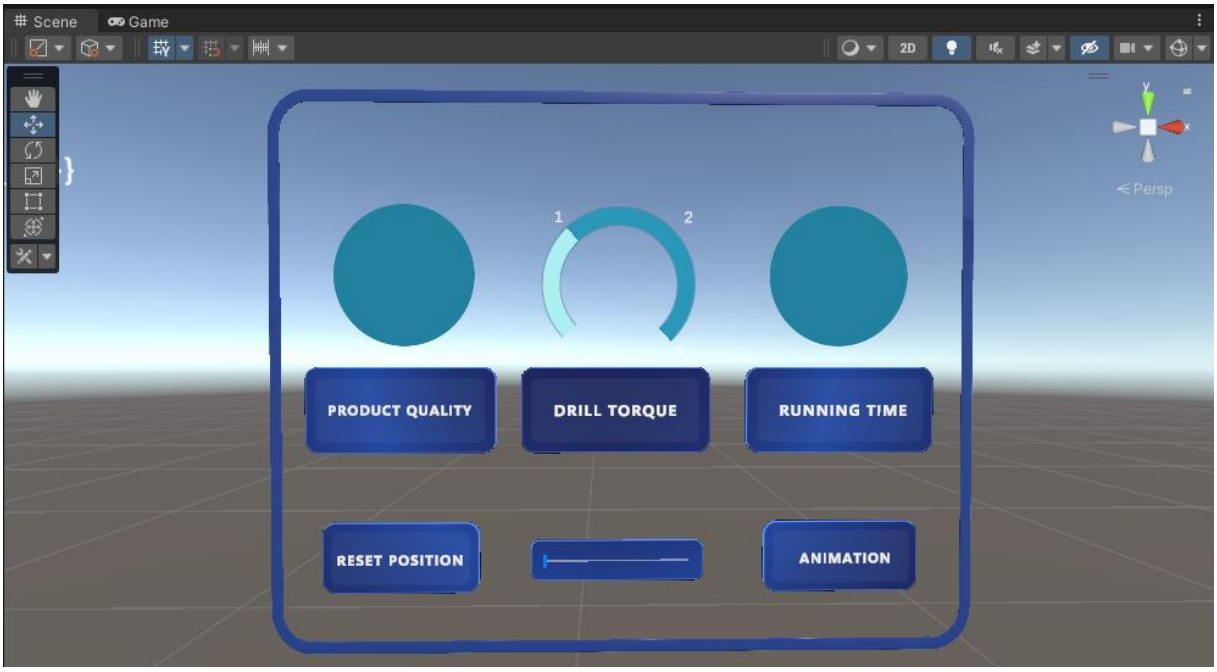


Слика 3.4 Изглед корисничког интерфејса

За сваку компоненту корисничке спреге је имплементирана засебна функционалност па ћемо објаснити посебно функционалност за сваку:

- *PRODUCT QUALITY* је дугме чијим притиском се изнад њега појављује *PieChart* тј. кружни дијаграм чија се вредност мења у реалном времену у зависности од пристиглих података са сервера.

- *DRILL TORQUE* је дугме чијим притиском се изнад њега појављује *Torque* чија се вредност такође мења у реалном времену у зависности од података пристиглих са сервера.
- *RUNNING TIME* има исту функционалност и приказује кружни дијаграм као и прво дугме, само што приказује друге податке.



Слика 3.5 Изглед корисничког интерфејса након притиска прва 3 дугмета

- *RESET POSITION* је дугме чијим притиском ресетујемо позицију машине. Како је могуће померати машину по простору, као и мењати њене димензије, имплементирана је ова функционалност како би се машина вратила у њен иницијални положај.
- *ANIMATION* је дугме чијим притиском се покреће анимација, тј. симулација рада машине
- Клизач има за функционалност да се његовим померањем ротира машина око своје осе

Поред корисничког интерфејса, у сцени се налази и већ поменута машина са којом се такође може интераговати на већ објашњен начин. На машину су увезани *Tooltip-и* тј. описни текст који нам даје неке информације везане за саму машину. На следећој слици можете видети изглед машине са увезаним описним текстом.

Слика 3.6 Изглед машине са описним текстом

Још једна имплементирана функционалност је да се уноси *URL* сервера преко виртуелне тастатуре. Када се покрене апликација, тастатура се може активирати тако што када окренемо дланове према себи, појавиће се дугме чијим притиском се појављује виртуелна тастатура. На дугмету се налази текст тренутног *URL* сервера и после куцања новог, текст на дугмету ће се ажурирати.

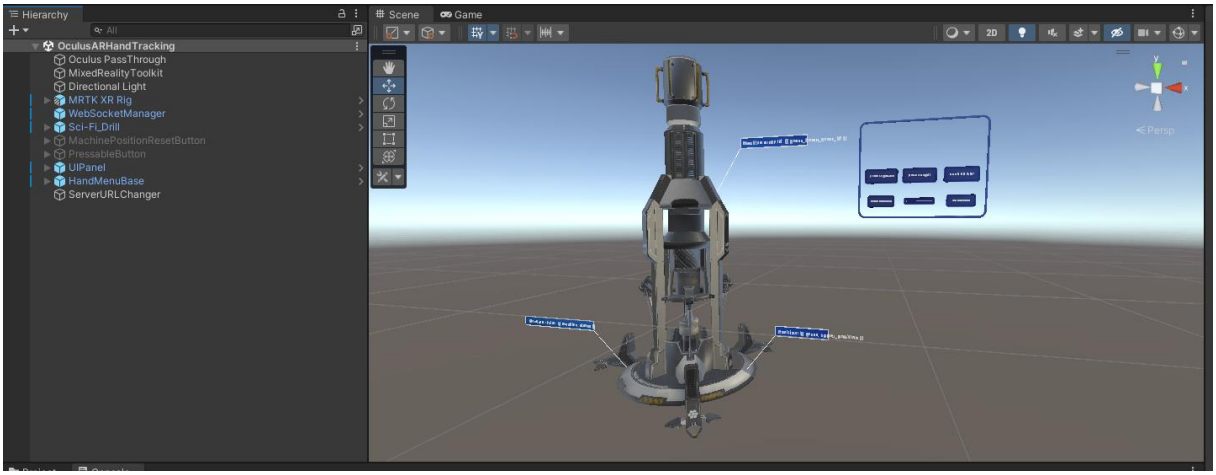
3.2.5 *OpenXR*

Функционалност *OpenXR* је већ објашњена у теоријским основама. Овде ћемо само напоменути да је *OpenXR* додат у пројекат као додатак и заједно са *MRTK* алатом омогућава кориснику да интерагује са објектима. Корисник има могућност да интерагује са њима уз помоћ контролера или уз помоћ руку.

3.2.6 Садржај *Unity* сцене

Unity сцена је простор у којем се може креирати и организовати игрица или апликација. Она представља скуп који садржи све ресурсе, објекте и подешавања потребна за израду пројекта. Комплетна сцена је организована у објекте, где сваки има

неку своју функционалност. На слици 3.7 можете видети како изгледа сцена за нашу апликацију.

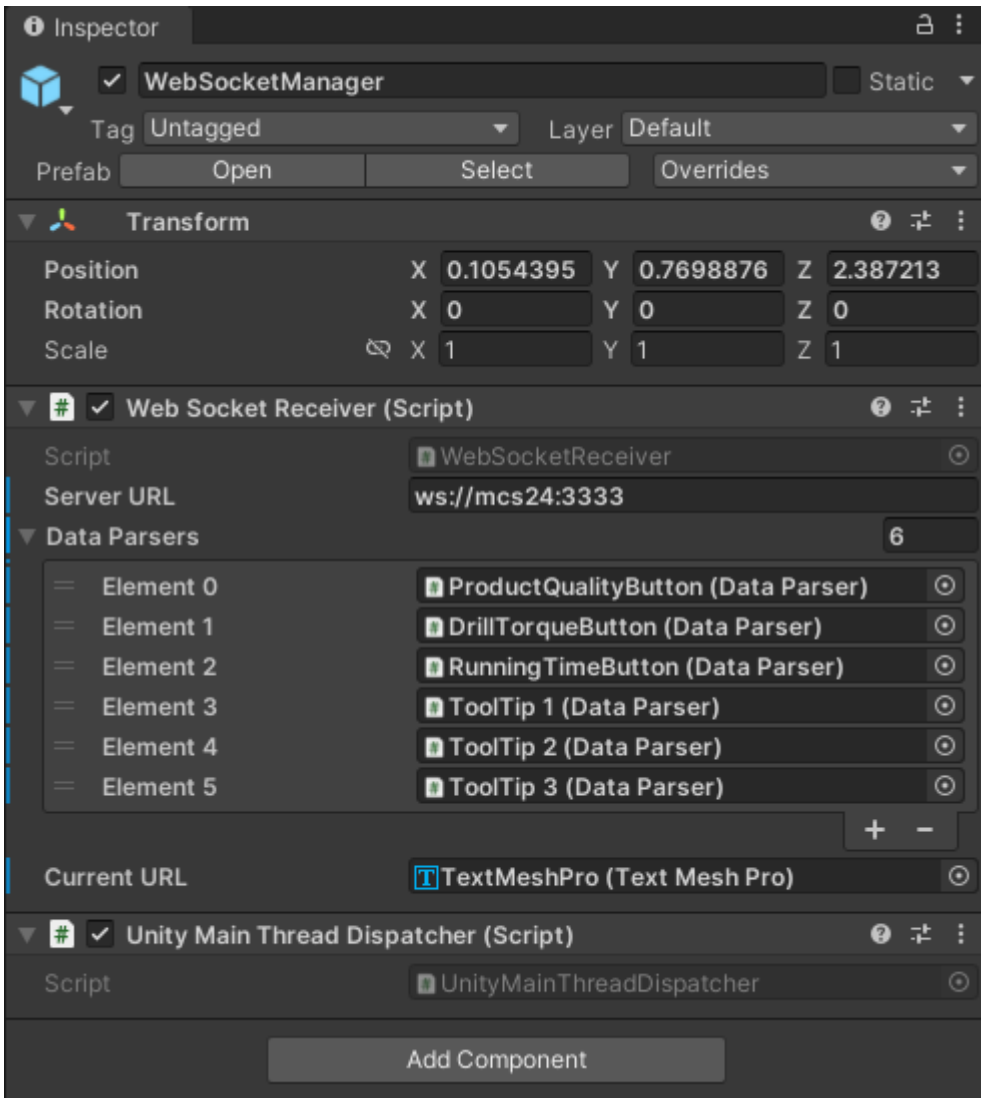


Слика 3.7 Изглед *Unity* сцене

Са леве стране слике могу се видети сви објекти који се налазе у сцени, стога ћемо рећи шта сваки од њих представља:

- *Oculus PassThrough* је објекат који омогућава примену *pass-through* у нашој апликацији тако што је на њега налепљена скрипта *OVR Passthrough Layer*.
- *MixedRealityToolkit* је објекат на који је додат *MRTK* алат па самим тим имамо могућност примену његових компоненти.
- *Directional Light* представља осветљење у сцени.
- *MRTK XR Rig* је централни контролер за управљање главном камером и руковање различитим захтевима и конфигурацијама различитих *XR* уређаја.
- *WebSocketManager* је објекат на који су налепљене скрипте за добављање података са сервера и затим њиховог постављања на корисничку спрегу.
- *Sci-FI_Drill* је објекат који представља машину у сцени.
- *UIPanel* представља корисничку спрегу са свим својим компонентама
- *HandMenuBase* омогућава појављивање дугмета након окретања дланова ка лицу
- *ServerURLChanger* је објекат који омогућава унос и памћење унетог *URL* преко виртуелне тастатуре

На слици 3.8 можете видети како изгледа када се скрипте налепе на објекат.



Слика 3.8 Изглед објекта када су на њега налепљене скрипте

3.3 Спуштање апликације на *Meta Quest Pro*

Након завршене имплементације апликације у *Unity* едитору, сада је потребно компајлирати пројекат за одређену платформу. Како се на *Meta Quest Pro* налази Андроид оперативни систем, потребно је у подешавањима за компајлирање специфицирати Андроид. Након компајлирања апликације, изгенерисана датотека можемо инсталирати на уређају коришћењем *Meta Developer Hub-a* преко терминала користећи *ADB*(енг. *Android Debug Bridge*).

Пре покретања апликације потребно је оспособити детекцију руку на самом уређају како би их могли користити у апликацији. Још једна ствар која се треба подесити на уређају јесте граница до које се можемо кретати у реалном свету, а да смо и даље у апликацији. То се назива *guardian* и он се буквално црта по просторији уз помоћ контролера.

4. Програмско решење

У овом поглављу биће представљена програмска реализација ове апликације кроз опис скрипти и класа које се користе у пројекту.

4.1 Преглед структуре класа

Уколико бисмо пројекат изделили по неким засебним целинама, могли бисмо рећи да се пројекат састоји из 3 целине. То су добављање података са сервера, приказ података на корисничкој спрези и остале функционалности као што су анимације. По томе можемо разделити и класе које користимо.

Целина	**Класа**
Комуникација са сервером	*WebSocketReceiver* *DataParser* *ChangeServerUrl*
Корисничка спрега	*PieChart* *TorqueFill* *UISpawner* *MachineSpinner*
Остало	*DrillAnimator*

Табела 4.1 Преглед структуре класа

4.2 Опис класа

Прелазимо на опис класа и њихових функционалности.

4.2.1 Класе за комуникацију са сервером

Прва класа коју ћемо описати је *WebSocketReceiver*. Ова класа је задужена за комуникацију са сервером.

Поље	**Опис**
protected DataParser [] dataParsers	Компоненте корисничког интерфејса
private TextMeshPro currentURL	Приказује *URL* сервера у сцени
public string serverURL	Сервер *URL*
private WebSocket ws	Протокол

Табела 4.2 Поља класе *WebSocketReceiver*

Метода	**Опис**
public string ServerURL	Поставља нови *URL* сервера ако се променио
void Start ()	Прва се позива
private void SetServerUrl ()	Поставља прослеђени *URL* сервера
private void CreateWebSocket ()	Инстанцира *WebSocket* и повезује се на сервер
void WebSocketEvents ()	Контролише *WebSocket* догађаје (*open, close, error, message*)
private void HandleWebSocketMessage (object sender, MessageEventArgs e)	За сваку компоненту корисничке спреге позива *ParseData* из *DataParser* скрипте
private void OnDestroy ()	Позива *Dispose*
public void Dispose ()	Ослобађа заузете ресурсе за *WebSocket*

Табела 4.3 Методе класе *WebSocketReceiver*

Следећа класа је *DataParser*, она је задужена да расчлањује податке добијене са сервера и да поставља вредности података на кориснички интерфејс.

Поље	**Опис**
protected PieChart [] pieCharts	*pieCharts* из сцене
protected TorqueFill [] torques	*torques* из сцене
Dictionary<string, object> MachineParams	Параметри машине
Dictionary<int, string> oidToFieldName	*oid* подаци из *JSON* фајла
Dictionary<int, string> sidToFieldName	*sid* подаци из *JSON* фајла
private Dictionary<int, string> machineStatusMapping	*machine status* подаци из *JSON* фајла
private Dictionary<int, string> activeSideMapping	*active side* подаци из *JSON* фајла

Табела 4.4 Поља класе *DataParser*

Метода	**Опис**
public void ParseData (string jsonData)	Рашчлањује податке из *JSON* фајла
public void GetParamFromId (int oid, int sid, int value)	На основу прослеђених *oid* и *sid* поставља се *value* на одговарајућу *UI* компоненту
private string GetMachineStatus (int value)	Враћа вредност из речника *machineStatusMapping* са кључем *value*, ако постоји
private string GetActiveSide (int value)	Враћа вредност из речника *activeSideMapping* са кључем *value*, ако постоји

Табела 4.5 Методе класе *DataParser*

Следећа класа је *ChangeServerUrl*. Ова класа нам омогућава да преко тастатуре у сцени унесемо *URL* сервера.

Поља	Опис
private WebSocketReceiver webSocketReceiver	Инстанца *WebSocketReceiver-a*
private SystemKeyboardExample keyboardExample	Инстанца *SystemKeyboardExample* која је већ постојећа
private TextMeshPro currentURL	Текст који показује тренутни *URL*

Табела 4.6 Поља класе *ChangeServerUrl*

Метода	Опис
private void Update ()	Проверава сваки оквир да ли је нови *URL* унесен са тастатуре и ако јесте поставља га за сервер *URL*

Табела 4.7 Методе класе *ChangeServerUrl*

4.2.2 Класе корисничког интерфејса

Прва класа коју описујемо је *PieChart*. Ова класа служи да поставља вредности на *PieChart* објекат из сцене, тј. са корисничке спреге.

Поље	Опис
private List<PieChartItem> pieCharts	Листа *pieCharts* који представљају вредности које се приказују
float totalAmount	Укупан број вредности који се могу приказати на кружном дијаграму
float totalValues	Вредности које се заправо приказују на кружном дијаграму

Табела 4.8 Поља класе *PieChart*

Метода	Опис
public void SetValues (Dictionary<string, object> valuesToSet)	Поставља вредности на корисничку спрегу
private void UpdateFillAmount (KeyValuePair<string, float> kvp)	Ажурира вредност која се приказује на дијаграму

Табела 4.9 Методе класе *PieChart*

Следећа класа је *TorqueFill*. Ова класа служи да поставља вредности на *TorqueFill* објекат из сцене, тј. са корисничке спреге.

Поље	**Опис**
private string torqueName	Име *torque-a*
private Image image	Слика која представља позадину *torque-a*, тј. његов дизајн

Табела 4.10 Поља класе *TorqueFill*

Метода	**Опис**
public void SetValues (Dictionary<string, object> valuesToSet)	Поставља вредност на *torque*

Табела 4.11 Методе класе *PieChart*

Следећа класа је *UISpawner*. Ова класа служи да поставља одговарајуће објекте из сцене активне, тј. видљиве или неактивне, тј. невидљиве.

Поље	**Опис**
private GameObject [] objectsToSpawn	Низ објеката из сцене чијом се активношћу манипулише

Табела 4.12 Поља класе *UISpawner*

Метода	**Опис**
public void ActiveInactive ()	Поставља активност објеката

Табела 4.13 Методе класе *UISpawner*

Прва класа коју ћемо описати је *MachineSpinner*. Ова класа је задужена за функционалност да се машина ротира око своје осе померањем клизача са корисничке спреге. Такође задужена је за ресетовање машине, тј. њене позиције и величине.

Поље	Опис
private Transform machine	Представља елементе објекта, тј. машине као што су позиција, ротација и величина
private Slider slider	Клизач са корисничке спреге
private Vector3 initialPosition	Помоћна променљива за позицију машине
private Quaternion initialRotation	Помоћна променљива за ротацију машине
private Vector3 initialScale	Помоћна променљива за величину машине

Табела 4.14 Поља класе *MachineSpinner*

Метода	Опис
private void Start ()	Прва се позива и иницијализује почетну позицију машине
public void OnSliderChange ()	Ротира машину
public void ResetMachinePosition ()	Ресетује позицију, величину и ротацију машине

Табела 4.15 Методе класе *MachineSpinner*

4.2.3 Остале класе

DrillAnimator класа контролише анимацију, тј. симулацију рада машине.

Поље	Опис
private Animator m_Animator	Инстанца аниматора
private AudioSource m_AudioSource	Инстанца за аудио

Табела 4.16 Поља класе *DrillAnimator*

Метода	Опис
public void OnButtonPressed ()	Приказује анимацију са аудио звуком након притиснутог дугмета

Табела 4.17 Методе класе *DrillAnimator*

5. Тестирање и верификација

У овом поглављу ће бити објашњен процес тестирања и верификације свих функционалности апликације. Како је свака скрипта, тј. свака класа задужена за неку функционалност апликације, биће објашњене по класама. На следећој табели је дат преглед свих функционалности апликације.

Класа	**Опис функционалности**
WebSocketReceiver	Успоставља везу са сервером
DataParser	Рашчлањује податке из *JSON* фајла
ChangeServerUrl	Поставља *URL* сервера преко виртуелне тастатуре
PieChart	Поставља вредност на *PieChart*
TorqueFill	Поставља вредност на *TorqueFill*
UISpawner	Поставља активност објеката
MachineSpinner	Омогућава ротацију машине као и ресетовање позиције
DrillAnimator	Симулира рад машине

Табела 5.1 Преглед функционалности апликације

У наредној табели приказаћемо тестирање свих функционалности као и успешност тестирања.

Опис теста	Очекивани резултати	Успешност теста
Опис: успоставa везе са сервером **Поступак:** након покретања апликације добијамо поруку о успешности повезивања	Добијање *WebSocket connection opened* поруке у логу	Успешан
Опис: добављање *JSON-а* и рашчлањивање података **Поступак:** добијамо поруке о валидности *JSON-а* и постојању података	Нема порука да *JSON* није валидан или да подаци не постоје	Успешан
Опис: постављамо *URL* сервера **Поступак:** укуцамо *URL* преко виртуелне тастатуре	Успешна успустава везе са новим сервером	Успешан
Опис: приказ података на *PieChart-у* **Поступак:** потребно је притиснути дугме *Product Quality* и *Running Time*	Приказ *PieChart-ова* са подацима који се ажурирају у реалном времену	Успешан
Опис: приказ података на *Torque-у* **Поступак:** потребно је притиснути дугме *Torque*	Приказ *Torque-а* са подацима који се ажурирају у реалном времену	Успешан
Опис: појављивање дијаграма након притиснутог дугмета **Поступак:** притиснути нека од 3 претходно поменута дугмета	Успешан приказ дијаграма након притиснутих дугмади	Успешан
Опис: ротација машине **Поступак:** потребно је померати клизач на корисничкој спрези	Успешна ротација машине након померања клизача	Успешан

Опис: ресетовање позиције, ротације и величине	Враћање машине у њен иницијални положај	Успешан
Поступак: потребно је притиснути на *Reset Position* дугме		
Опис: покренути симулацију рада машине	Покретање симулације рада машине заједно са звуком рада	Успешан
Поступак: потребно је притиснути дугме *Animation*		

Табела 5.2 Поступак тестирања

5.1 Комуникација и добављање података од сервера

Како је више пута наведено, класа *WebSocketReceiver* успоставља везу са сервером. *WebSocket* класа у *Unity-y* успоставља везу са сервером тако што се инстанцира њен објекат и њеном конструктору се проследи *URL*. Уколико је прослеђени *URL* празан стринг, у логу ћемо добити следећу поруку:

- *WebSocket server URL is null or empty*

Уколико је са прослеђеним *URL-ом* све у реду позваће се *ConnectAsync* метода из *WebSocket* класе. У зависности од успешности везе, у логу добијамо једну од следећих порука:

- *WebSocket connection opened* – говори да је веза успешно успостављена
- *WebSocket error* – говори да је дошло до грешке приликом успостављања везе
- *WebSocket connection closed* – говори да је веза раскинута

Како постоји функционалност да се *URL* сервера мења преко виртуелне тастатуре, након промене у логу добијамо следећу поруку:

- *ServerURL changed to: serverURL*

Када добијемо ову поруку у логу, тада знамо да је *URL* успешно промењен.

DataParser је класа која рукује са *JSON* фајлом, добијеним од *WebSocketReceiver*. Она рашчлањује податке добијене у *JSON* фајлу и уз помоћ пропратних класа поставља параметре на корисничку спрегу.

Како *WebSocketReceiver* добавља *JSON* са сервера, он и прослеђује добијени *JSON DataParser* класи позивањем *ParseData* методе која за параметар прима сам *JSON* фајл.

Уколико добијени *JSON* није валидан, тј. ако је празан, у логу добијамо одговарајућу поруку:

- *Invalid JSON data*

У супротном, *JSON* је валидан и врши се извлачење података из *JSON* фајла. Параметри се добављају на основу *oid* и *sid*, стога је потребно прво те вредности добавити из *JSON-a*. Успешност добављања *oid* и *sid* параметара закључујемо на основу порука из лога. Уколико неки од тих параметара нису пронађени у *JSON-y* добијамо неку од следећих порука у логу:

- *Oid not found in dictionary*
- *Sid not found in dictionary*

За вредности *oid* и *sid* кључева добавља се одређени параметар, тако за вредности истих кључева добијамо параметре *MachineSide* и *ActiveSide*. *MachineSide* добијамо позивањем *GetMachineStatus* и уколико тај податак није пронађен у *JSON-y*, у логу добијамо поруку:

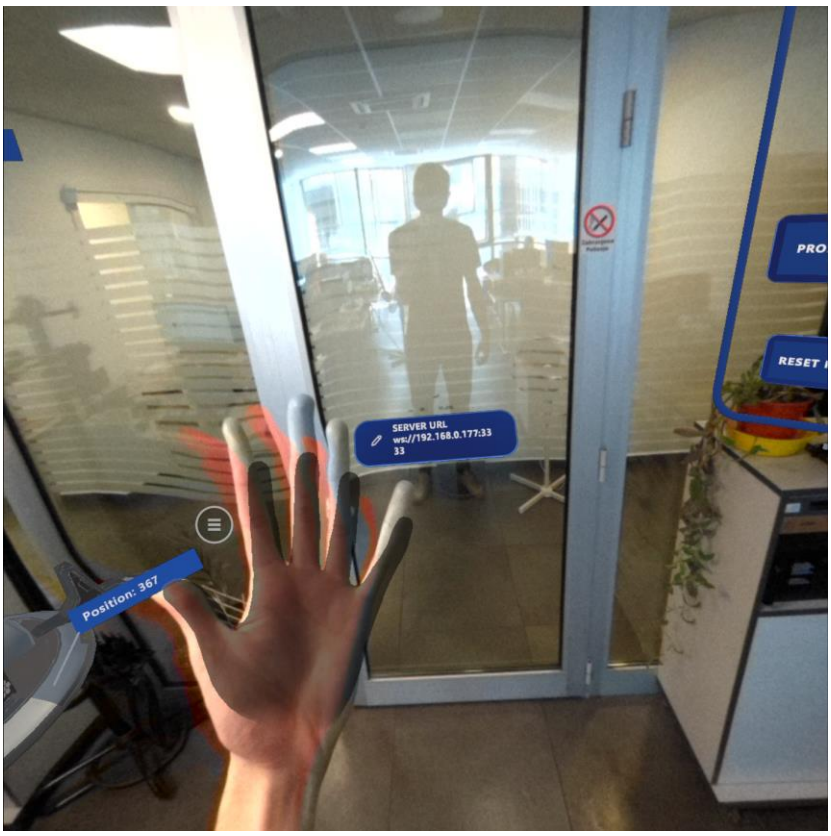
- *Machine status mapping not found for value: value*

ActiveSide добијамо позивањем *GetActiveSide* методе и уколико тај податак није пронађен у *JSON-y*, у логу добијамо поруку:

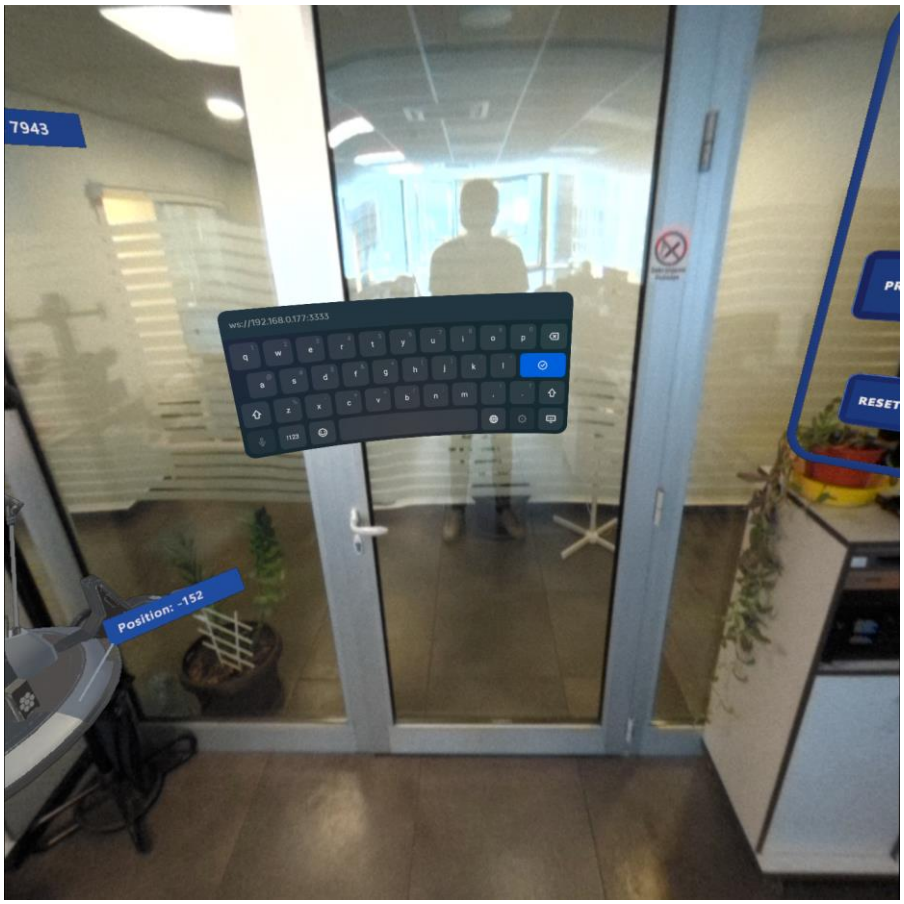
- *Active side mapping not found for value: value*

Помоћу *ChangeServerUrl* класе можемо мењати *URL* користећи виртуелну тастатуру. Након што на тастатури искуцамо нови *URL* сервера и потврдимо, у логу ћемо добити поруку:

- *Server URL changed to {webSocketReceiver.serverURL}* – где *webSocketReceiver.serverURL* представља нови *URL*



Слика 5.1 Дугме за унос *URL* сервера



Слика 5.2 Тастатура за унос *URL* сервера

5.2 Приказ података на корисничкој спрези и интеракција са објектима

Овде ћемо редом проћи кроз класе чија је функционалност приказ добављених података на корисничкој спрези, као и интеракција са објектима из сцене.

Прва је *PieChart* класа. У оквиру ње се налази *SetValues* метода која као аргумент прима речник са параметрима који могу бити приказани на *PieChart-y*. На *ProductQuality* дугмету, на корисничкој спрези, је дефинисано да приказује параметре *no_of_good_prod* и *no_of_bad_prod*. То је дефинисано у инспектору *Unity* едитора на објекту где је налепљена *PieChart* скрипта. *RunningTime* је такође скрипта која приказује податке у виду *PieChart-a*. Он је дефинисан да приказује *machine_uptime* и *machine_downtime* податке. Када притиснемо дугме, изнад се појављује дијаграм који показује вредности дефинисаних параметара. Уколико се подаци ажурирају у реалном времену, то значи да је имплементација функционална.

Слика 5.3 Изглед *UI* након притиска *ProductQuality* и *RunningTime* дугмади

Следећа класа је *TorqueFill* класа. Она такође има *SetValues* методу која прима исти речник као аргумент. На *DrillTorque* је дефинисано да приказује *press_torque* параметре. Када притиснемо дугме, изнад дугмета се појављује *Torque* који приказује дефинисане податке. Такође, ако се подаци ажурирају у реалном времену, то значи да је тражена функционалност успешно имплементирана.

UISpawner има само једну методу и то *ActiveInactive*. Њен задатак је да се након притиска дугмета појаве дијаграми изнад њих. Функционалност се проверава простим притиском на дугме и ако се после тога појаве дијаграми изнад, имплементација је успешна.

Слика 5.4 Изглед *UI* након притиснутог *Torque* дугмета

Следећа класа је *MachineSpinner*. Она обавља две функционалности, а то су ротација машине око своје осе и ресетовање стања машине на њен иницијални положај. У склопу ове класе налазе се *Start* метода, која памти иницијално стање машине, затим *OnSliderChange* метода, која је задужена за ротацију машине после померања клизача и на крају *ResetMachinePosition* метода, која враћа машину у њен иницијални положај. Простим померањем клизача, машина се ротира око своје осе и такође, након померања машине у сцени, притиском на *ResetPosition* дугме, машина се враћа у њен иницијални положај, стога су обе функционалности успешно имплементиране.

Слика 5.5 Позиција машине пре и после притиска на *Reset Position* дугме

Последња класа је *DrillAnimator*. Ова класа служи да покрене анимацију машине. Анимација представља симулацију рада. У оквиру ове класе постоји метода *OnButtonPressed* у којој се позивају *Play* методе из *Animator* и *AudioSource* класе, које пуштају саму анимацију, као и звук анимације. Након притиска дугмета *Animation*, анимација је успешно покренута и можемо закључити да је функционалност успешно имплементирана.

6. Закључак

У овом раду су приказане могућности технологије проширене реалности кроз успешну имплементацију једне АР апликације у Unity-у. Апликација пружа интерактивно корисничко искуство омогућавајући корисницима интеракцију са објектима из сцене, да мењају величину објекта и премештају га кроз своје физичко окружење. Коришћење MRTK алата за дизајн корисничке спреге даље истиче могућности интеграције једноставних и интерактивних интерфејса у АР окружењу.

Ова апликација показује динамичке могућности проширене реалности укључивањем ажурирања података добављених са сервера у реалном времену омогућавајући корисницима да прате реалне параметре виртуелног објекта, у нашем случају виртуелне машине. Разумевање ових података је визуализовано елементима као што су кружни дијаграми, описни текстови и лучни дијаграми. Поред тога, интеграција ажурирања података са сервера и укључивање анимација покренутих интеракцијама корисника додаје још један слој реализма АР искуству.

Ова апликација демонстрира значајан потенцијал АР технологије у различитим областима, као што су производња, одржавање, образовање и многе друге. Омогућавањем корисницима да интерактивно комуницирају са комплексним машинама, можемо олакшати боље разумевање, побољшане сигурносне мере и значајну уштеду што времена, тако и материјала. На пример, радници могу симулирати рад машине на виртуелном моделу саме машине пре рада са стварном.

Као и код неке друге револуционарне технологије, постоје начини за даљи развој ове и сличних апликација. Неке функционалности које би се могле

надоградити на ову апликацију би могле бити функције као што су гласовне команде за интеракцију без употребе руку или коришћење алгоритама машинског учења за предиктивно одржавање, где систем може предвидети потенцијалне кварове или прекиде машине на основу неких података.

Технологија проширене реалности је тек на почетку свог развоја. Како наставља свој развој, можемо очекивати још интерактивније и реалистичније апликације са већим доживљајем саме апликације. Спајање реалног и виртуелног света, као што је приказано у овом раду, обећава узбудљиву будућност када је у питању људска интеракција са овом технологијом.

7. Литература

[1] Benjamin Nicoll and Brendan Keogh, The Unity Game Engine, and the Circuits of Cultural Software (2019), Palgrave Pivot Cham

[2] Julie Carmigniani and Borko Furht, Augmented Reality: An Overview (2011), Springer

[3] Meta Quest Pro: Our Most Advanced New VR Headset, https://www.meta.com/quest/quest-pro/, 01.07.2023.

[4] The OpenXR Specification – Khronos Registry, https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html, 01.07.2023.

[5] Mixed Reality Toolkit 3 Developer Documentation – MRTK3, https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/, 02.07.2023.