Korištenje virtualne realnosti za simulaciju pokreta ruke u računalnoj igri Jenga

Mršić, Lorena

Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij Informacijsko komunikacijski sustavi

Lorena Mršić

Korištenje virtualne realnosti za simulaciju pokreta ruke u računalnoj igri Jenga

Diplomski rad

Mentor: izv. prof. dr. sc. Marina Ivašić-Kos

Rijeka, rujan 2021.

Sadržaj:

1. Zadatak završnog rada

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 13.2.2021.

Zadatak za diplomski rad

Pristupnik: **Lorena Mršić**

Naziv diplomskog rada: **Korištenje virtualne realnosti za simulaciju pokreta ruke u računalnoj igri Jenga**

Naziv diplomskog rada na eng. jeziku: **Using virtual reality to simulate hand movements in the computer game Jenga**

Sadržaj zadatka:

Opisati osnovne koncepte virtualne stvarnosti te kratak razvoj i karakteristike uređaja koji se koriste za generiranje i prikaz 3D modela. Istražiti tehnologiju i alate koji se koriste za izradu VR igre te ukratko opisati ključne karakteristike alata koji će se koristiti u praktičnom dijelu rada.

Osmisliti igru proizvoljnog žanra za jednog igrača i izraditi je u VR okruženju. Opisati potrebne prilagodbe u Unity engine-u za izradu 3D VR igre te način upravljanja igrom i kontroliranja pokreta u igri.

Opisati dizajn i mehanike igre te objasniti ključne korake razvoja VR igre kao što je izrada 3D modela, teksturiranje površina, izrada objekata, upravljanje kretanjem, kreiranje UI izbornika, računanje napretka u igri i slično.

Mentor:

Izv. prof. dr. sc. Marina Ivašić-Kos

Voditeljica za diplomske radove:

Izv. prof. dr. sc. Ana Meštrović

Zadatak preuzet: 25.02.2021

(potpis pristupnika)

2. Sažetak

Tema ovog rada bila je proučavanje rada uređaja za praćenje pokreta, upravljanje dodirom gestama u AR/VR aplikacijama, te prikazivanje rezultata praćenja pokreta ruku u obliku aplikacije koristeći se Unity Engine-om.

Na početku je objašnjeno što je virtualna stvarnost, što je praćenje pokreta, koja je njena svrha i u kojim područjima se koristi te zašto je bitan njen razvoj u budućnosti. Zatim se opisuju uređaji koji se koriste za praćenje pokreta kao što je Leap Motion koji se koristi u ovom radu. Nakon toga se opisuje aplikacija Unity Engine u kojoj će se odvijati izrada aplikacije, te tijek izrade. Pri izradi igre koristio se C# programski jezik.

Cilj je bio uzeti igru iz stvarnog života koja se igra rukama, te tu igru transformirati u digitalnu inačicu kojom se također upravlja pokretima ruku ali koristeći uređaj za praćenje pokreta.

Ključne riječi: praćenje pokreta, VR, AR, Leap Motion, Unity Engine, C#

3. Uvod

Virtualna stvarnost je produkt računala tj. simulacija okruženja koje može biti ili iz stvarnog svijeta ili iz izmišljenog [28]. Ona omogućava korisniku da fizički bude prisutan na mjestu i području na kojem želi biti, odnosno na mjestu koje odredi putem računala. Okruženje virtualne stvarnosti je uglavnom trodimenzionalno iskustvo koje se prikazuje na zaslonu putem posebnih ili stereoskopskih prikaza, a poboljšano je informacijama iz senzora poput zvukova koji dopiru iz slušalica ili zvučnika ili korištenjem uređaja koji pružaju povratne informacije o sili ili taktilne informacije. Primjena takvih uređaja i virtualne stvarnosti moguća je u mnogim područjima kao što su medicina, vojska, policija, „gaming“ industrija.

Tehnologija je transformirala percepciju ljudi o svijetu stvarajući novo okruženje kojim će se moći kretati, testove biologije ili kemije koje će moći polagati, operacije koje će moći izvoditi bez ugrožavanja ljudskih života, pružajući mogućnost da posjete određene dijelove zemlje koje možda nikad ne bi mogli posjetiti tijekom svog života, bilo šta što čovjek može zamisliti, virtualna stvarnost može omogućiti. Virtualna stvarnost nadišla je društvene i zemljopisne barijere otkad njezine aplikacije ne koriste samo znanstvenici, čemu je pridonijela činjenica da je njeno korištenje znatno jeftinije nego što je bilo prije. Zbog povećanja virtualnih zajednica, konfiguriranje virtualnih sustava je optimizirano, aplikacije se proizvode uz pomoć stranica poput InstaVR koje ubrzavaju samu izradu i distribuciju, te pomažu u analizi, dok je za korištenje istih aplikacija dovoljno računalo, mobitel ili igrača konzola, uz naravno uređaj virtualne stvarnosti [25, 27]. Prolaskom vremena te razvojem tehnologije, virtualna stvarnost svoju je primjenu pronašla u mnogim aspektima svakodnevnog života, kao što su recimo pružanje usluga internetskog obrazovanja [24].

Virtualna stvarnost, kao i Internet, stvorena je za određenu svrhu, ali moderna tehnologija učinila je da je njena primjena vidljiva u raznim područjima, a neka od njih su znanost, zrakoplovstvo, vojska, kina, škole i sl. [24, 26]. Zavladala je svijetom te trenutno predstavlja jednu od najbrže rastućih grana tehnologije.

4. Virtualna Stvarnost

4.1. Razvoj virtualne stvarnosti

Virtualna stvarnost predstavlja jednu od najpropulzivnijih tehnologija 21. stoljeća, no postoji veliki jaz između onoga što je virtualna stvarnost prezentirala u prošlosti, a što je ona danas, što u vidu tehnologije, ali i percepcije koju je stvarala [1].

Prvi korak prema virtualnoj stvarnosti napravio je Sir Charles Wheatstone 1838. godine kada je svijetu predstavio koncept stereoskopije (eng. Stereopsis). Prepoznao je da svako oko prikazuje različite slike iz vodoravnog položaja, a slika svakog oka se razlikuje od drugog. Ljudski mozak kombinira dvije različite fotografije ako je objekt snimljen iz različitih položaja. Iluzija dubine stvorena je iz ravne slike koju je moguće razlikovati samo u vodoravnom nesrazmjeru [1].

Prva ideja o uređaju virtualne stvarnosti se nalazi u knjizi Stanley G. Weinbaum-a "Pygmalion's Spectacles" - Pygmaniolove naočale. Spomenuti autor u predmetnoj knjizi navodi kako one mogu pružiti jedinstveno iskustvo korisnicima gdje svijet mogu doživjeti kroz holografski pogled, uz različite mirise, dodir i okus [1]. Ta ideja je zaživjela 1960. godine kada je Morton Heilig stvorio Telesphere masku (Slika 1). Ova maska - zaslon za glavu - pružila je širok vid slike sa stereoskopskim 3D slušalicama za stereo zvuk, ali bez praćenja pokreta. Ovo je bio prvi primjer zaslona za glavu. Dvije godine nakon izumio je i mehanički uređaj "Sensorama" (Slika 2), stroj nalik mini kinu u kojem su korisnici mogli iskusiti sva osjetila, ne samo zvuk i prizore koristeći se kombinacijom 3D zaslona, stereo zvučnika, mirisa, vibracija ispod sjedala i atmosferskih efekata poput vjetra tijekom prikazivanja filmova.

Slika 1 Telesphere maska

Slika 2 Sensorama

Sljedeći veliki izum dogodio se 1968. godine kada su Ivan Sutherland i Bob Sproull stvorili Damoklov mač (eng. “Sword of Damocles”), koji je predstavljao prvi uspješan pokušaj povezivanja mehanizma virtualne stvarnosti s računalom, a ne s kamerom. Damoklov mač smatra se prvim sustavom virtualne stvarnosti montiranim na glavu. Dizajniran je na način da ako korisnik pomakne glavu može vidjeti virtualni oblik žičanog okvira (Slika 3). Nažalost zbog svoje veličine, te neprikladnosti za širu upotrebu, ovaj sustav nikad nije u potpunosti razvijen, već je ostao u eksperimentalno-razvojnoj fazi.

Slika 3 Damoklov Mač

Virtualna je stvarnost inicijalno najveću primjenu pronašla u vojsci, odnosno ratnom zrakoplovstvu. Thomas A. Furness 1982. razvija prvu kacigu za obuku vojnih pilota primjenom VR zaslona za glavu. Njegov izum nazvan je "Super kokpit" (eng. "Super Cockpit"), a omogućavala je pilotima da razumiju radarske slike, infracrvene i 3D karte koje su računalno generirane. Zaslon za glavu je bio pričvršćen na senzore za praćenje glave koji su pratili pokrete oka pilota kako bi se pozicija oka podudarala sa slikama koje generiraju računala (Slika 4).

Slika 4 Super kokpit

"Virtualna stvarnost" kao termin kojeg danas poznajemo prvi je put upotrijebljen 1984. godine, i to od strane tvrtke VPL Research, koja je tada bila poznata po svojim VR naočalama i DataGlove rukavicama, te je ujedno i prva tvrtka koja ih je plasirala na tržište široj javnosti. Uređaji koje su prodavali bili su rijetki i skupi, pa su tako na primjer, jedne VR naočale koštale 9400$. Još jedan od uređaja koje vrijedi spomenuti su Power Glove budžetna verzija VPL-ovih rukavica, te su proizvedene kao dodatak kontrolera za Nintendo Entertainment System [2]. Kako bi cijena bila pristupačnija rukavice su izostavile velik dio tehničke sofisticiranosti i osjetljivosti na kretanje koje su imale DataGlove rukavice.

S obzirom na tadašnje cijene uređaja za virtualnu stvarnosti njihova šoroka primjena je bila gotovo nemoguća sve do početka 90-ih godina kada je Virtuality Group lansirala "Virtuality". To je bio stroj za arkadne igre, gdje su igrači morali nositi VR naočale kako bi iskusili 3D karakteristike igre. Ovo je prvi put u povijesti da se neki VR uređaj namijenjen razonodi počeo masovno proizvoditi.

Nakon toga dolazimo do već poznatijih virtualnih uređaja, kao što su Nintendo-ve naočale i kontroleri Virtual Boy i Google-ov "Street View" i naočale "Ocolus Rift".

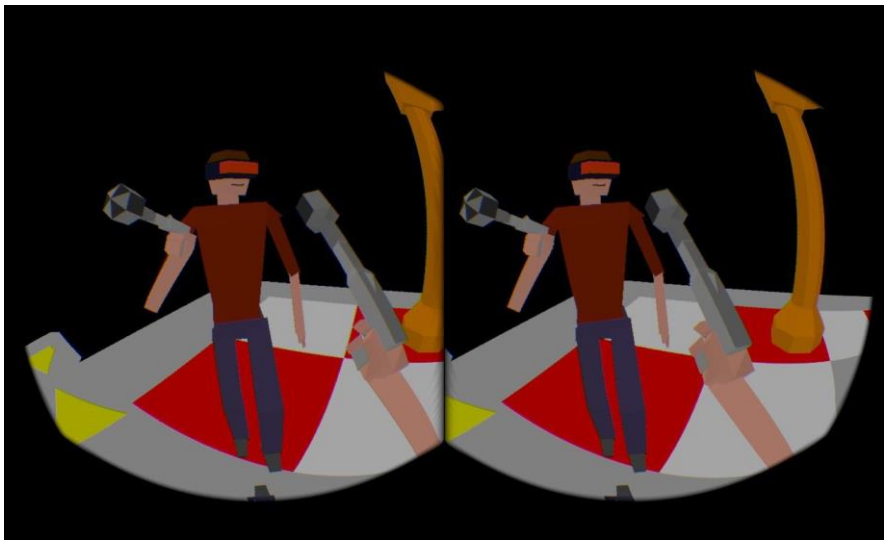
4.2. Računalne igre virtualne stvarnosti

Jedne od prvih igara virtualne stvarnosti bile su proizvedene upravo za korištenje prethodno navedene Power Glove rukavice [4]. Te igre se zovu "Super Glove Ball" i "Bad Street Brawler". U "Super Glove Ball"-u igrač kontrolira virtualnu rukavicu na ekranu. Pojavljuju se energetske kuglice koje lete po sobi, odbijajući se od zidova. Kad lopta udari u

pločicu na zidu, ta pločica se uništi. Igrač rukavicom baca i udara loptu po sobi, te ne može dopustiti da lopta ode iza rukavice jer tada gubi (Slika 5) [3].

Slika 5 Super Glove Ball

Početkom 90-ih Virtuality nam donosi niz igara virtualne stvarnosti. Jedna od njih je "Dactyl Nightmare", igra koja sadrži jednu mapu za više igrača sa nekoliko razina i platformi, gdje su igrači opremljeni oružjem za lansiranje granata, te osim što igraju jedni protiv drugih u određenom trenutku ih napada i pterodaktil (Slika 6)[5,8]. Osim Nightmare-a vrijedi spomenuti robotsku pucačinu "Grid Busters" i "Legends Quest" avanturu.



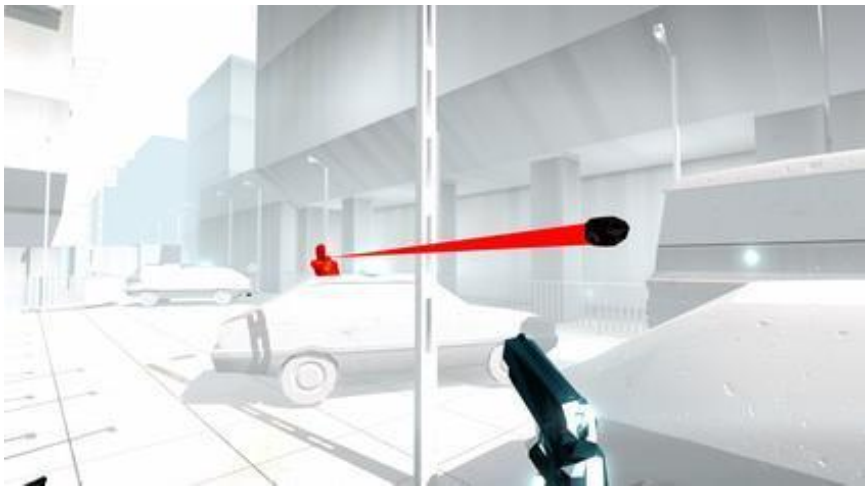
Slika 6 Dactyl Nightmare

Unatoč velikom naporu da se poveća tržište industrije igara koje se temelje na virtualnoj stvarnosti, troškovi proizvodnje takve tehnologije ostali su i dalje preveliki u odnosu na tržište koje nije prihvaćalo tako visoke cijene tehnologije, te je cjenovna nepristupačnost rezultirala zatišjem u razvoju tehnologije virtualne stvarnosti.

Ponovna aktivacija tržišta virtualne stvarnosti dogodila se 2016. godine dolaskom Oculus Rift naočala, koje su prvi put najavljene 2013. godine kao "jeftina" VR opcija za video

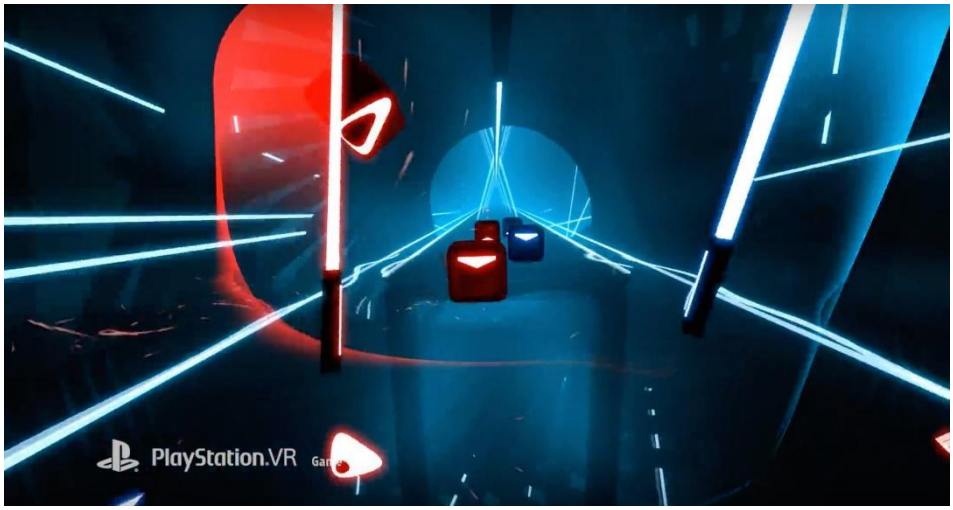
igre. Prva igra koja je dobila svoju VR verziju isključivo za korištenje Oculus Rift-om je Doom 3. Nakon njega došlo je i do razvoja drugih VR igara i hardvera (Playstation VR, HTC Vive).

Unatoč jeftinijim hardverima za VR, sama tehnologija i njezina primjena u industriji video igara nije doživjela značajniji porast, prvenstveno kao posljedica nedovoljnog razvoja igara koje bi tu tehnologiju primjenjivale. Kao posljedica manjka video igara na tržištu koje aktivno implementiraju sustav virtualne stvarnosti, došlo je i stagnacije potražnje za opremom za virtualnu stvarnost [4]. Tek su dvije video igre uspjele potaknuti ozbiljniji porast zanimanja za tehnologiju virtualne stvarnosti, a to su „Superhot“ i „Beat Saber“. Superhot“ je pucačina u prvom licu koja slijedi tradicionalnu mehaniku igranja pucačina (Slika 7). Igrač pokušava ubiti neprijatelje raznim oružjima, te vrijeme unutar igre napreduje samo onda kada se igrač kreće, ako on stoji, vrijeme je usporeno što omogućuje igraču da procijeni situaciju i prikladno reagira [6].



Slika 7 Superhot

„Beat Saber“ je VR ritmička igra koja se događa u nad realističnom neonskom okruženju i sadrži uređaj za rezanje (Saber) blokova koji predstavljaju glazbene ritmove koji lete prema vama [7]. Igra sadrži razne pjesme sa više razina težine, te postoji mogućnost da muzički blokovi ne dolaze samo od naprijed, već sa svih strana. (Slika 8)



Slika 8 Beat Saber

4.3. Vrste stvarnosti

Naziv koji se često koristi kada se govori o tehnologijama virtualne stvarnosti je proširena ili produžena stvarnost ili XR (eng. Extended Reality, hrvatski naziv za to još uvijek nije jasno definiran) [33, 36]. XR označava spoj virtualne tehnologije i dodatnih uređaja i stvarnosti, objedinjuje sve vrste stvarnosti koje postoje, te X u XR zamjenjuje jednu od četiri tehnologije stvarnosti:

- Proširena stvarnost (eng. Augmented Reality)
- Miješana stvarnost (eng. Mixed Reality)
- Potpomognuta stvarnost (eng. Assisted Reality)
- Virtualna stvarnost (eng. Virtual Reality)

4.3.1. Proširena stvarnost (Augmented Reality)

Proširena stvarnost kao što i sam naziv govori, proširuje stvarnost oko nas, ali putem zaslona nekog uređaja i to najčešće mobitela. Proširena stvarnost koristi postojeće okruženje u stvarnom svijetu i stavlja virtualne informacije na nju kako bi poboljšala iskustvo. Na zaslonu se pojavljuju virtualno stvoreni predmeti koji ne postoje u stvarnosti, te idealan primjer proširene stvarnosti poznat diljem svijeta je igra Pokemon GO (Slika 9).

Slika 9 Pokemon GO AR

4.3.2. Miješana stvarnost (Mixed Reality)

Miješana stvarnost je spoj virtualne stvarnosti i proširene stvarnosti te samim time omogućuje interakciju s objektima, odnosno elementima koji se pojavljuju [35, 36]. Izgledom elementi nas podsjećaju na hologrami, ali kao u filmovima, možemo njima upravljati. Jedan od najpoznatijih uređaja koji nam omogućuju to su HoloLens (Slika 10.). HoloLens su par pametnih naočala koje je razvio i proizveo Microsoft, te su to prve naočale – zaslon za glavu

sa miješanom stvarnošću. Koristeći se senzorima, naprednom optikom i holografskom obradom, hologrami vidljivi kroz naočale mogu se koristiti za prikaz informacija ili simulaciju virtualnog svijeta. HoloLens ima puno optičkih senzora koji detektiraju poziciju osobe u prostoru, kameru koja je okrenuta prema dolje kako bi vidjela pokrete ruku, nekoliko mikrofona, posebnu procesorsku jedinicu za holografsku obradu, senzor svjetla, te zvučnike koji simuliraju zvuk s bilo kojeg mjesta u prostoriji. Sve to je potrebno kako bi naočale omogućile što bolju orijentaciju u prostoru, pratile objekte kojima smo okruženi (zidove, predmete) te uklopile holograme u taj isti prostor. HoloLens mogu ubrzati učenje zaposlenika, poboljšati liječenje, pomoći u učenju te mnoge druge stvari, te su zato popularne u mnogim velikim firmama, a samo neke od njih su Audi, L'Oreal, Renault, Mercedes, Jagiellonian Sveučilište u Krakowu i MediView. [29, 30, 31]

Slika 10 HoloLens

4.3.3. Potpomognuta stvarnost (Assisted Reality)

Iako je često zanemarena zbog svoje sličnosti s proširenom stvarnošću, potpomognuta stvarnost odnosi se na bilo koju tehnologiju koja omogućuje osobi da vidi zaslon u svom vidnom polju. Razlikuje se od proširene stvarnosti po tome što se informacije na ekranu ne preklapaju sa fizičkim okruženjem. Nju najviše koriste građevinari, tehničari na terenu, mehaničari i druge profesije sličnih zanimanja [33, 34]. Uređaji koji to omogućuju su HTM-1 i HTM-1Z1 proizvedeni od strane Realwear-a, te najčešće dolaze uz kacigu koja se prikvači za uređaj. (Slika 11)

Oba uređaja su dizajnirana za uporabu u mokrim, prašnjavim, vrućim te općenito opasnim okruženjima, a uz to pružaju i sigurnost na radu zbog kacige. HTM-1Z1 je dizajniran specifično za rad u industrijama nafte, plina i energije. [32]

Slika 11 Realwear HTM-1

4.3.4. Virtualna stvarnost (Virtual Reality)

Virtualna stvarnost označava korištenje računalne tehnologije za stvaranje u potpunosti simuliranog okruženja. Koristeći uređaje virtualne stvarnosti korisnik je „uronjen“ u virtualni svijet te može komunicirati s objektima u tom svijetu, drugim riječima virtualni svijet postaje stvarnost. U virtualnoj stvarnosti računalo koristi slične senzore i matematiku kao i za proširenu stvarnost, no umjesto lociranja predmeta u stvarnom okruženju i postavljanje virtualnih, okruženje je u potpunosti virtualno, te se računa pozicija glave odnosno smjer kretanja. Ako se glava korisnika okrene, grafika unutar virtualnog svijeta reagira u skladu s pokretom. [36, 37]

4.4. Praćenje pokreta

Sustavi za praćenje obično se sastoje od uređaja koji može stvoriti signal i senzora koji ga može očitati. Oni mogu poprimiti različite oblike, uključujući optičke, elektromagnetske i zvučne signale te mehaničke sustave.

U optičkim sustavima emitiranu svjetlost kamere hvataju u različitim formatima. Elektromagnetsko praćenje može se koristiti tamo gdje male naelektrizirane zavojnice utječu na druge elektromagnetske senzore, a kako njihovo magnetsko polje utječe na druge, može ih postaviti u prostor. Akustični sustavi koriste ultrazvučne valove za prepoznavanje položaja i orijentacije ciljnih objekata. Mehaničko praćenje može koristiti zglobne ruke, udove, džojstike ili senzore povezane sa slušalicama ili unutar njih, slično kao što je inercijalno praćenje na telefonima često omogućeno akcelerometrima (brzinomjerima) i žiroskopima. Akcelerometar mjeri brzinu promjene u kretanju osobe. Sićušni senzori prevode sile u čitljive električne signale. Međutim, akcelerometar ne može otkriti položaj telefona u odnosu na sve ostalo pa se u tu svrhu koristi žiroskop. Žiroskop može mjeriti rotacijske pokrete u odnosu na neutralan položaj. Svijet virtualne stvarnosti koristi nekoliko ovih metoda za svoj hardver [12,15].

4.4.1. Uređaji za praćenje pokreta

Telefoni su jedna od najosnovnijih opcija pregledavanja VR sadržaja. Oni pružaju jeftino, ali ipak interaktivno rješenje. Telefon je stavljen unutar slušalica, a zaslon je podijeljen kako bi dao izlaz za svako oko, dajući iluziju dubine. Akcelerometar i žiroskopski sustavi na pametnim telefonima i slušalicama daju aplikacijama virtualne stvarnosti istinski osjećaj pokreta.

Prije GPS sustava i radara brodovi su za navigaciju koristili svjetionike kako bi relativno procijenili položaj. Na isti način funkcioniraju HTC Vive, Vive Cosmos i ostali proizvodi koji su bazirani na Valve-ovom „Lighthouse Base“ tehnologiji (Slika 12). „Lighthouse Base“ tehnologija je precizna verzija stvarnih svjetionika koja emitira nevidljivu svjetlost koja preplavljuje sobu. Svjetlost dolazi iz stacioniranih LED dioda i dva para lasera, koje se vrte na 60HZ-a, odnosno 60 puta svake sekunde LED diode bljeskaju, a zatim jedan od dva rotirajuća lasera emitira svjetlost po prostoriji. U međuvremenu prijemnik (zaslon za glavu ili kontroler) koji je prekriven foto senzorima detektira bljeskove i laserske zrake. U trenutku kad prepozna bljesak, zaslon za glavu ili kontroler počne odbrojavati sve dok ne prepozna koji od njegovih foto senzora je „pogođen“ svjetlosti lasera. Odnos između toga koji je foto senzor pogođen i u kojem trenutku se koristi kako bi se izračunao njegov točni položaj u odnosu na Lighthouse senzore postavljene po prostoriji. Istovremeno se laserom pogađa više foto senzora te oni onda stvaraju „pozu“ odnosno 3D oblik koji govori gdje se nalazi zaslon za glavu u prostoriji i u kojem smjeru je okrenut [12,15].

Slika 12 HTC Lighthouse senzori

Oculus Rift i Rift S, kao i ostali proizvodi poput Nintendo Wii kontrolera, koriste sličan princip prepoznavanja poza, ali implementirani na malo drugačiji način. Oculus Rift koristi konstelacije infracrvenih LED dioda ugrađenih u zaslon i kontrolere. Naknadno su nadodana i dva radna senzora dizajniranja za prepoznavanje specifičnog sjaja LED dioda i pretvaranje njihovog položaja u podatke. U naslonu za glavu se nalaze magnetometar, žiroskop i akcelerometar. U kombinaciji omogućuju precizno praćenje kroz sve tri dimenzije 3D svijeta.

Oculus je osim Rift-a i S-a proizveo i bežični zaslon za glavu pod nazivom Quest koji se koristi drugačijom metodom praćenja. On skenira prostoriju u potrazi za bilo kojim predmetima koji se nalaze oko osobe (zavjese, prozori, stolovi itd.) i stvara 3D kartu za „igranje“. Kombinira podatke žiroskopa i akcelerometra s zemljovidom koji daju zaslonima za glavu položaj brzinom od 1000Hz, odnosno jednom svake milisekunde. Oculus zasloni za glavu se zatim koriste i Guardian sustavom. Guardian sustav prikazuje zidne i podne oznake u aplikaciji kada se korisnici približe granicama područja za igru koje su definirali [18]. Kada se korisnik približi rubu granice, prikazuje se prozirna mreža kao sloj koji je postavljen preko igre [12, 15].

Mechatech je razvio sustav praćenja zasnovan na izravnom mjerenju ljudskog tijela. AgileVR proizvod nosi se samo preko koljena i vrši izravna mjerenja. Izravno fizičko praćenje ljudskog tijela ima jednu veliku prednost od sustava koji se temelje na kameri, a ta je da mu ne treba kamera, ne može se dogoditi da senzor ne može popratiti pokret jer su u AgileVR-u senzori izgrađeni iznutra prema van. Također, podaci o pozi se ne trebaju izračunavati, već se mogu čitati izravno s uređaja u stvarnom vremenu, što smanjuje zakašnjelu reakciju odnosno odaziv, a to je omogućeno od strane visoko preciznih rotacijskih senzora. Svaki modul koristi bežičnu Bluetooth 5.0 komunikaciju za prijenos podataka i bežično ažuriranje softvera. AgileVR ima policentrični zglob koji oponaša bio mehaničko ponašanje ljudskog koljena, nalik na medicinsku potporu za koljeno [13,14].

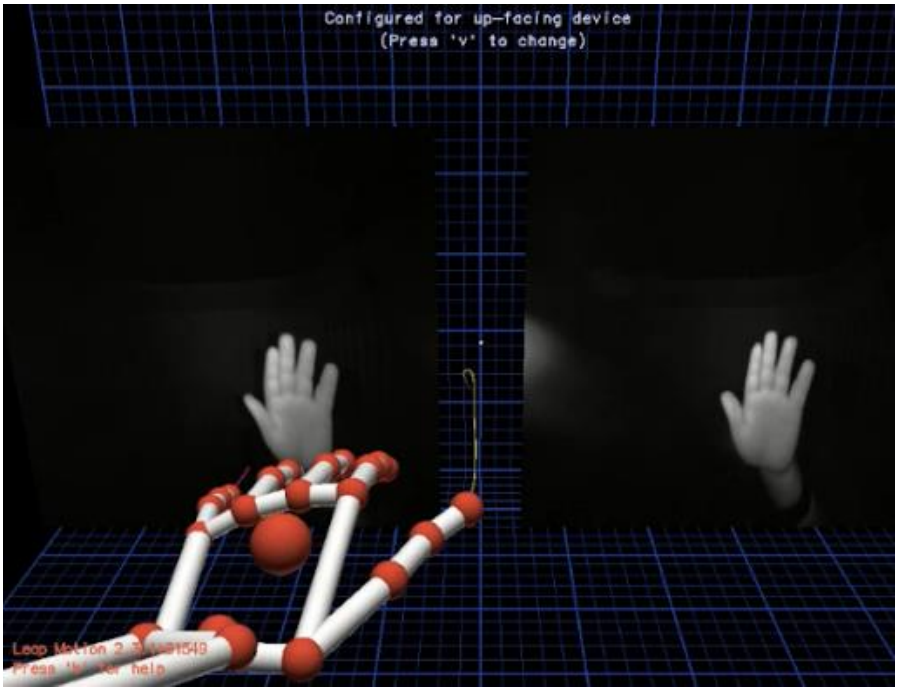
4.4.2. Leap Motion

U ovom diplomskom radu za praćenje pokreta će nam služiti Leap Motion kontroler (Slika 13). To je optički modul za praćenje ruku koji bilježi pokrete ruku koristeći dvije kamere i infracrvene LED diode [17]. One prate infracrvenu svjetlost na valnoj duljini od 850

nanometara, koja je izvan spektra vidljive svjetlosti. LED diode pulsiraju sinkronizirano s brzinom kadra fotoaparata, što omogućuje nižu potrošnju energije i povećani intenzitet.

Slika 13 Leap Motion kontroler

Širokokutne leće koriste se za stvaranje velike zone interakcije unutar koje se mogu prepoznati ruke korisnika (Slika 14). Ta zona se proteže od 10 do otprilike 60 cm, u vidnom polju od 140x120 stupnjeva. Kontroler čita podatke senzora te ih sprema u vlastitu lokalnu memoriju i izvodi sve potrebne prilagodbe, te se ti podaci zatim prenose putem USB-a u Ultraleap-ov softver za praćenje ruku. Finalni rezultat je siva stereo slika infracrvenog svjetlosnog spektra, te najčešće jedini predmeti koji su vidljivi su oni izravno osvjetljeni LED diodama uređaja [16].



Slika 14 Prepoznavanje ruku Leap Motion kontrolerom

5. Unity Engine

U početku, tvrtke koje su razvijale video igre morale su razviti i vlastiti game engine za razvoj igre što je bilo izuzetno skupo i zahtjevno, te je uzimalo puno vremena.

Nakon nekoliko desetljeća Unity Technologies 2005. godine razvija jedan od najpopularnijih game engine-a današnjice: Unity game engine. Objavio ga je prilikom svjetske konferencije za developere Apple-a kao službeni game engine Mac OS X-a, no uskoro je postao dostupan i za Windows. Unity game engine danas podržava više od 25 platformi (neke od njih su PlayStation VR, Xbox One, Linux, Android...) .

Cilj Unity-a je bio pojednostaviti razvoj digitalnih igara, skratiti vrijeme razvoja, smanjiti trošak, ali i omogućiti besplatno korištenje enginea za razvoj igara. Unity game engine je baziran na C# programskom jeziku.

Iako se Unity najviše koristi za izradu video igara, to nisu jedine industrije koje su ga prihvatile. Unity se koristi u izradi filmova, arhitekturi, inženjerstvu, građevini te automobilskoj industriji jer se može koristiti za stvaranje 2D i 3D simulacija, te simulacija virtualne stvarnosti i proširene stvarnosti [9,10,11].

6. Razvoj računalne igre Jenga

6.1. Ideja

Ideja ovog diplomskog rada jest izrada računalne igrice "Jenga" koja se koristi virtualnom stvarnošću, a korištenjem alata Unity i *Leap Motion* kontrolora. *Jenga* je društvena igra u kojoj je cilj izgraditi toranj od drvenih blokova. Ona se sastoji od točnije 54 bloka, te oni za početak igre moraju biti složeni u čvrsti pravokutni toranj od 18 razina, pri čemu su na svakoj razini složena tri bloka. Blokovi unutar svake razine su orijentirani u istom smjeru, dok su razine naizmjence različito orijentirane što je vidljivo na slici 15.

Igru započinje osoba koja je izgradila toranj, te igrači naizmjence uklanjaju jedan blok s bilo koje razine ispod najvišeg reda i postavljaju ga vodoravno na vrh tornja, okomito odnosno u suprotnom smjeru na sve blokove na koje se postavlja. Svaki igrač može koristiti samo jednu ruku da dodirne toranj ili pomakne blok u bilo kojem trenutku, ali može mijenjati ruke kad god to želi. Nakon što najgornji novonastali red sadrži 3 bloka, on je završen te se sljedeći blok postavlja okomito na njega.

Što se više blokova izvlači, to toranj postaje nestabilniji, a sama igra završava kad se struktura raspadne. Igrač u toku čijeg se poteza toranj raspao, gubi igru.

Jengu je stvorila Leslie Scott, suosnivačica tvrtke Oxford Games Ltd, na temelju igre koja se razvila u njenoj obitelji ranih 1970-ih. Naziv Jenga potječe od „Kujenga“ , koja u svahili jeziku znači graditi. [43]

Slika 15 Jenga

Sukladno s time igra će se sastojati od:

Tabela 1 Dijelovi igre

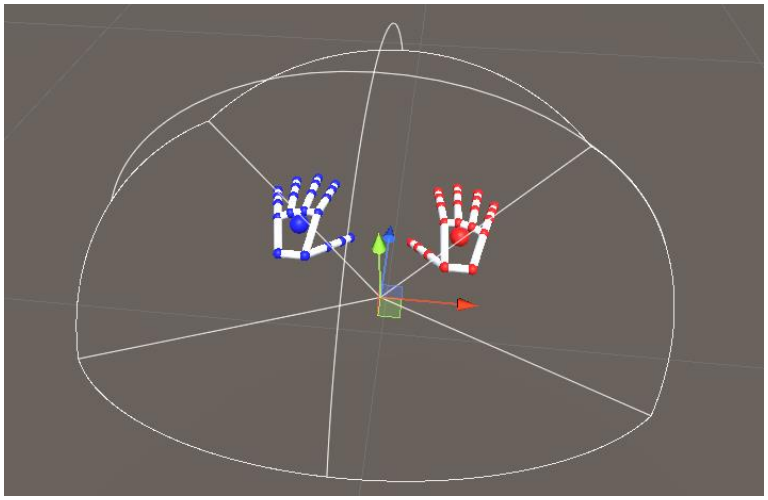
OBJEKTI	FUNKCIJE	OPIS RADNJI
Igrač – VR ruke	• Upravljanje igrom • Uklanjanje drvenih blokova • Navođenje po meni-u	Igrač treba pokretima svojih ruku iznad Leap Motion-a upravljati samom igrom. Ukloniti blok spajajući kažiprst i palac te povlačeći prema sebi ili u stranu pritom pazeći da se ne sruši toranj, a nakon toga postaviti taj isti blok na vrh tornja razdvajanjem kažiprsta i palca.
Teren	• Podloga • Blokovi • Toranj • Soba	Teren se sastoji od sobe u kojoj se nalazi podloga – daska na kojoj se nalaze blokovi koji čine predmet igre odnosno toranj.
Korisničko sučelje	• UI gumbi	Koriste se za rotaciju i pomicanje kamere ukoliko je potrebno te resetiranje tornja.

Nakon što je plan izrade i dijelova napravljen, može se krenuti s izradom igre.

6.2. Igrač – VR ruke

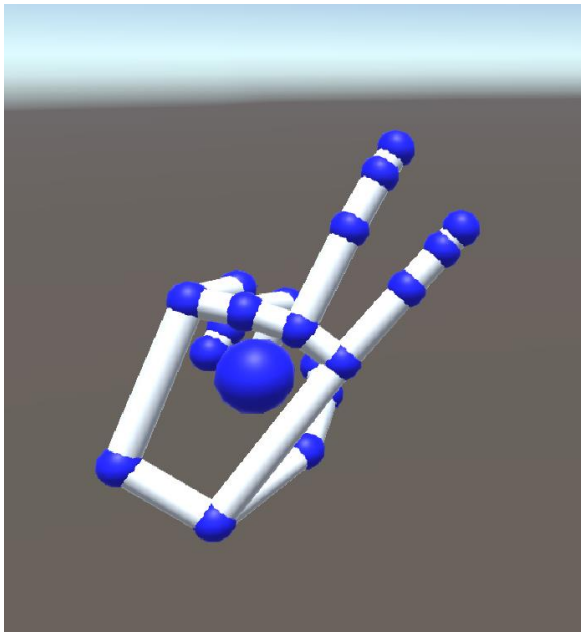
Redoslijed izrade računalne igre započinje stvaranjem projekta i implementacijom Leap Motion kontrolera. Za to je potrebno preuzeti Leap Motion Unity Module, Unity Core Assets paket (U vrijeme preuzimanja verzija paketa je 4.7.1.) i Leap Motion Interaction Engine koji sadrži svu pripadajuću fiziku potrebno za interakciju ruka s objektima unutar računalne igre. Osim fizike, paket sadrži i model ruku koje dobivaju podatke iz Leap Motion kontrolera koji su potrebni za izračunavanje prethodno spomenute fizike kako bi hvatanje, bacanje i micanje objekata bilo moguće.

U projekt unutar datoteke Assets uvezemo prethodno skinuti Leap Motion Core Assets paket. Unutar paketa nalazi se Leap Hands Demo scena koju uvozimo u naš projekt. Iz nje su nam potrebne samo dvije stvari, a to su LeapHandController i HandModels. HandModels kao što i samo ime kaže je model ruku koji se sastoji od lijeve i desne ruke (Slika 16), dok LeapHandController upravlja podacima dobivenim iz Leap Motion kontrolera, obrađuje ih, te ih koristi za upravljanje modelom ruku. Ako pokrenemo projekt možemo vidjeti da praćenje ruku funkcionira te da je uspješno implementirano u naš projekt (Slika 17) [22]. Prema zadanim postavkama boje ruku se mijenjaju svaki put kada se identificira nova ruka jer njegovo praćenje nije u mogućnosti odrediti vidi li istu ruku koju je vidio ranije, odnosno kada ruka izađe iz vidokruga kamere i opet se vrati, ruke promijene boju. Radi jednostavnosti igre modificirali smo zadanu skriptu da koristi samo plavu boju za desnu i lijevu ruku.



Slika 16 HandModels Unity

Prilikom dodavanja ruku u projekt one su postavljenje u istu razinu s kamerom unutar glavnog Leap Rig objekta. Hijerarhija objekata unutar leap motion VR konfiguracije je objašnjena u jednom od sljedećih poglavlja.

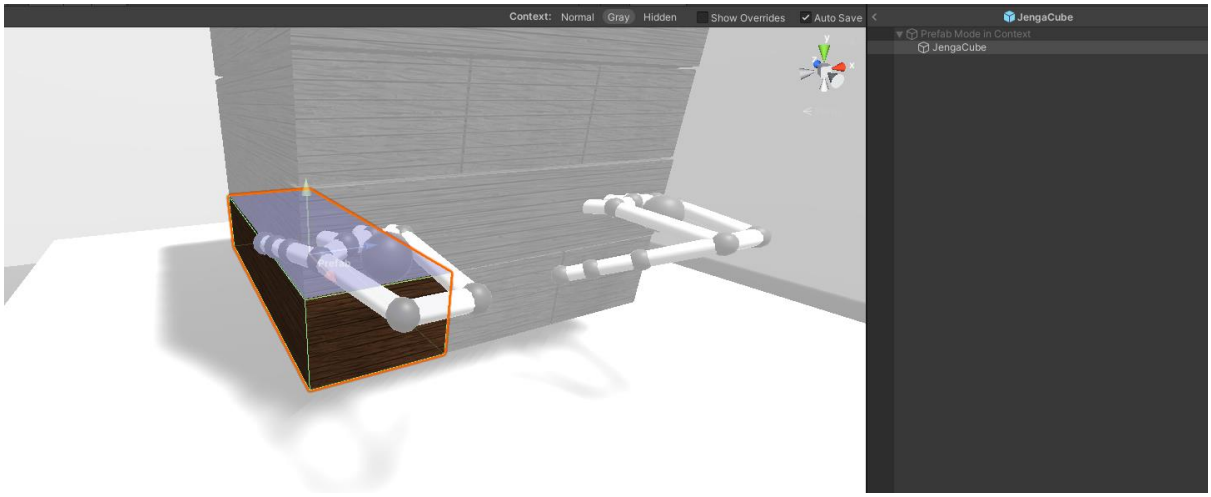


Slika 17 Uspješno praćenje pokreta ruke

6.3. Izrada objekata

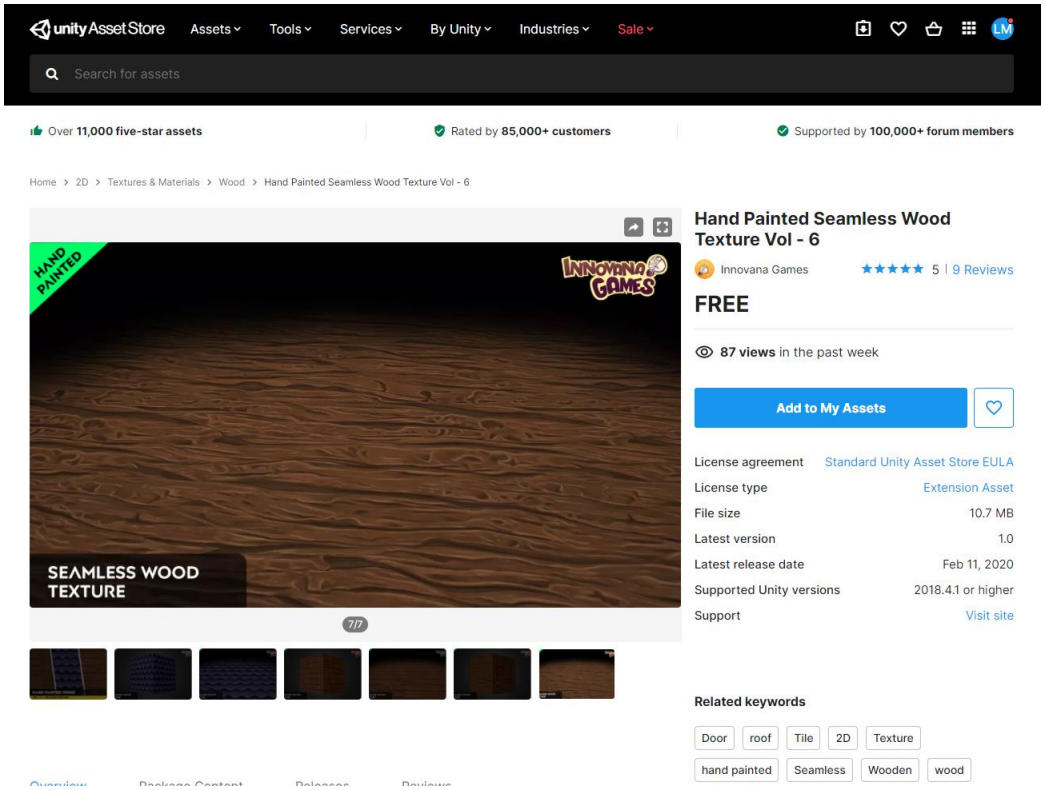
Redoslijed izrade računalne igre uključuje izradu terena koji čine soba, podloga odnosno daska kojem se nalazi toranj napravljen od blokova.

Dodajemo 3D plane objekt i nazivamo ga Floor, te 3D cube objekt kao zid kako bi napravili sobu. S obzirom da se toranj sadrži od više redova, u kojem svaki red sadrži 3 bloka, napravljen je prefab JengaBlock (Slika 18). Prefabe koristimo za generiranje nepokretnih objekata koji se ponavljaju. Svaka promjena koja se napravi na Prefab-u odmah se primjenjuje na sve ostale objekte prefaba generirane u projektu.



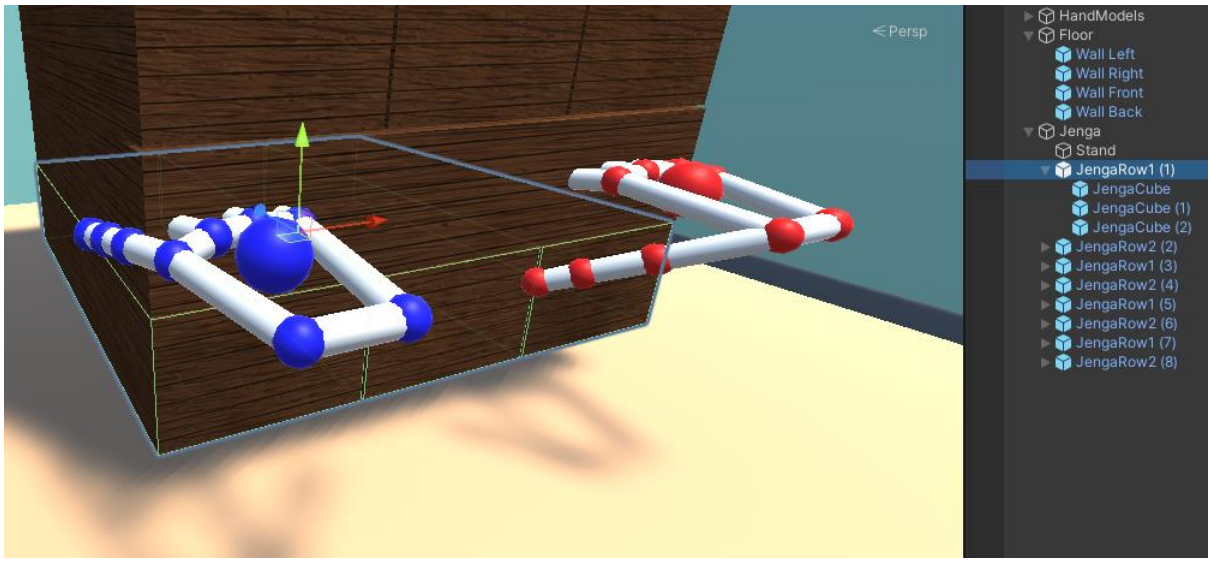
Slika 18 JengaBlock prefab

Za boju materijal blokova korišten je besplatni materijal preuzet sa Unity Asset Store-a (Slika 19) [20], dok su za sve ostale objekte materijali izrađeni unutar projekta. Unity Asset Store je trgovina na kojoj se može besplatno ili uz određenu naknadu preuzeti 2D i 3D objekte, zvukove, predloške skripti za kontrolu igrača, materijale, animacije i slično. Nakon što se u trgovini pronađe nešto što se želi uključiti u projekt, dodaje se u projekt klikom na „Add to My Assets“. Asset Store je službena trgovina Unity-a, no postoje i druge stranice poput Free3D [38] i RenderHub [39] koji služe istoj svrsi. [9]



Slika 19 Materijal blokova

U Jengi su redovi poslagani na dva načina, odnosno u dva smjera, te se tako slažu naizmjenično. Zbog toga su 3 komada poslagana te grupirana u parent Empty object koji je nazvan JengaRow1 za prvi kat, i još 3 komada za drugi kat nazvan JengaRow2 (Slika 20). S obzirom da će se ti katovi ponavljati (11 katova radi jednostavnosti) također radimo prefabe. Empty object-i se koriste kako bi projekt bio organiziran, strukturiran odgovarajućom hijerarhijom objekata i kako bi se bilo lakše snaći u njemu.



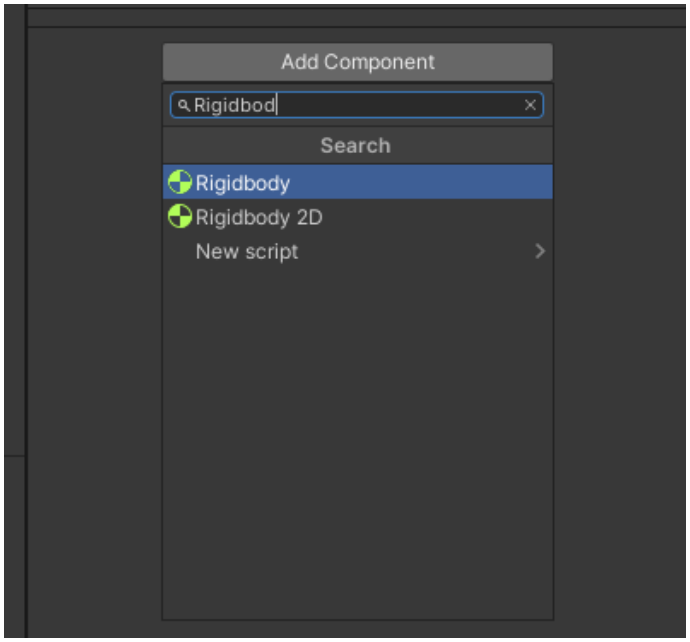
Slika 20 JengaRow prefarbi

6.4. Interakcija s objektima

U trenutnoj fazi izrade imamo sve potrebne objekte, no ako pokrenemo igru vidimo kako naše ruke prolaze kroz stvorene blokove. Razlog tome je što naši blokovi nemaju nikakvu fiziku – Rigidbody, niti collider, te nisu definirana pravila interakcije između ruku i objekata koje želimo pomicati.

6.4.1. Rigidbody – čvrsto tijelo

U fizici Rigidbody je definiran kao čvrsto tijelo u kojem je deformacija jednaka nuli, te se smatra da kruto tijelo ima kontinuirano raspodjelu mase. U vidu izrade igara unutar Unity-a Rigidbody je komponenta koja omogućuje GameObject-u da reagira na fiziku u stvarnom vremenu. To uključuje reakcije na sile i gravitaciju, masu, otpor i zamah. Rigidbody se može dodati na GameObject dodavanjem komponente i upisom Rigidbody u pretragu te klikom na željenu verziju (3D ili 2D) (Slika 21).



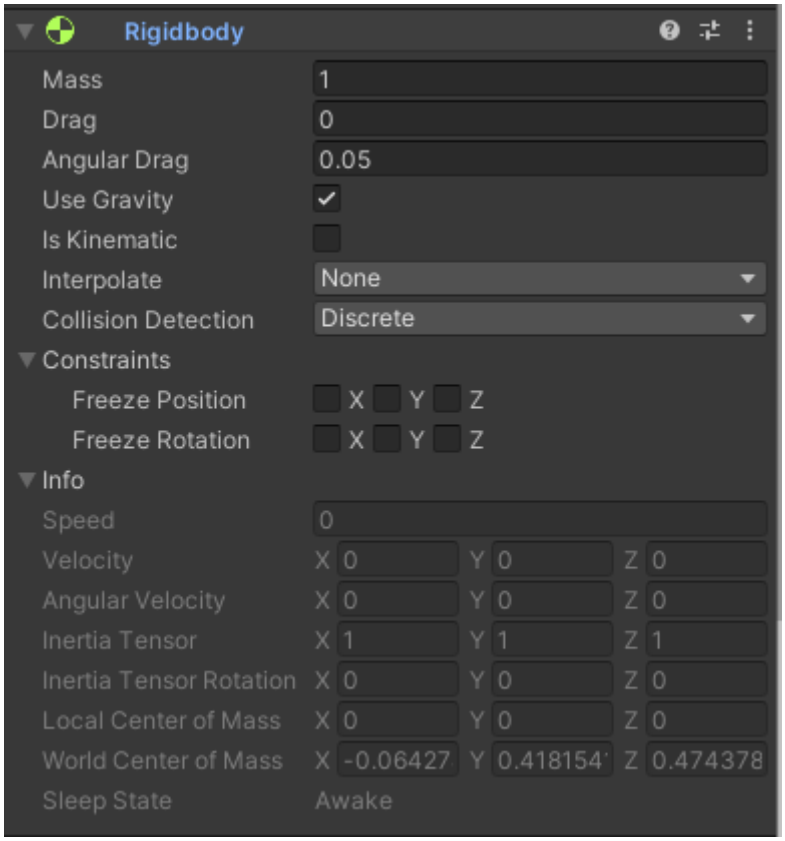
Slika 21 Odabir rigidbody-a

Nakon odabira Rigidbody-a prikazuju se određene postavke (Slika 22):

- Mass – masa predmeta. Predmeti veće mase pri sudaru guraju objekte manje mase, no masa ne utječe na brzinu pada predmeta, na to utječe gravitacija i otpor – trenje.
- Drag – odnosi se trenje/otpor, kojim se onda usporava objekt. Što je veće trenje, to se objekt više usporava.
- Angular Drag – odnosi se na trenje predmeta pod određenim kutom. Angular drag se koristi za usporavanje rotacije objekta, što je veće trenje to je rotacija sporija.
- Use Gravity – kontrolira da li gravitacija utječe na ovaj predmet. Ukoliko je ova postavka stavljena na *false* predmet se ponaša kao u svemiru, odnosno lebdi.
- Is Kinematic – upravlja utjecajem fizike na Rigidbody. Ako je isKinematic omogućen, sile, sudari ili spajanja neće utjecati na Rigidbody, te će tijelo biti u potpunosti pod

kontrolom animacije ili skripte za promjenu odnosno transformaciju pozicije. To je pogodno za izradu likova koji se pokreću animacijom, ali se u određenom trenutku mogu brzo pretvoriti u ragdoll postavljanjem isKinematic na *false*. Ragdolls su vrste animiranih objekata čije kosti potpuno preuzima sila fizike. Najčešće se koriste kada je lik poražen ili iscrpljen, pa se želi vizualno postići efekt utjecaja fizičke sile na lika.

- Interpolate – Interpolacija omogućuje ublaživanje učinka pokretanja fizike prilikom korištenja fiksne brzine kadrova. Interpolacija je prema zadanim postavkama isključena. Interpolacija se najčešće koristi kada se radi o Rigidbody-u igrača odnosno glavnog lika, na kojem je kamera fokusirana, te izvođenje fizike ponekad može utjecati na takozvano „sjeckane slike“ kada fizika i grafika nisu sinkronizirane. Preporučuje se uključivanje interpolacije samo na glavnom liku.
- Collision Detection – Odnosi se na način otkrivanja dodira ili sudara Rigidbody-a. Koristi se za sprječavanje prolaska objekata koji se brzo kreću kroz druge objekte bez otkrivanja dodira. Otkrivanje dodira moguće je postaviti na četiri razine: Discrete (zadana postavka), Continuous, Continuous Dynamic i Continuous Speculative. Za najbolje rezultate preporučeno je postaviti Continuous Dynamic na objekt koji se brzo kreće, te Continuous na objekte kroz koje će on prolaziti to jest s kojima će se sudariti. Otkrivanje dodirivanja je prema zadanim postavkama uvijek omogućeno. Trenutak u kojem želimo isključiti otkrivanje dodirivanja je kada imamo ragdoll na kojem je kinematika uključena. U ovom radu otkrivanje dodira je za Jenga blokove postavljeno na Continuous iz razloga što želimo prepoznati i najmanji dodir odmah kada se on dogodi u slučaju da se ruke brzo pomiču.
- Constraints – Kontrolira koji su stupnjevi slobode dozvoljeni za simulaciju ovog Rigidbody-a. Prema zadanim postavkama dopuštene su sve rotacije i kretanje duž svih osi, no u nekim slučajevima potrebno je ograničiti Rigidbody da se kreće ili rotira samo po nekim osima, na primjer kada se radi na razvoju 2D igre. U ovom projektu ograničenja nisu potrebna iz razloga što se Jenga blokovi mogu kretati i rotirati u bilo kojem smjeru prilikom uzimanje ili spuštanja.



Slika 22 Rigidbody

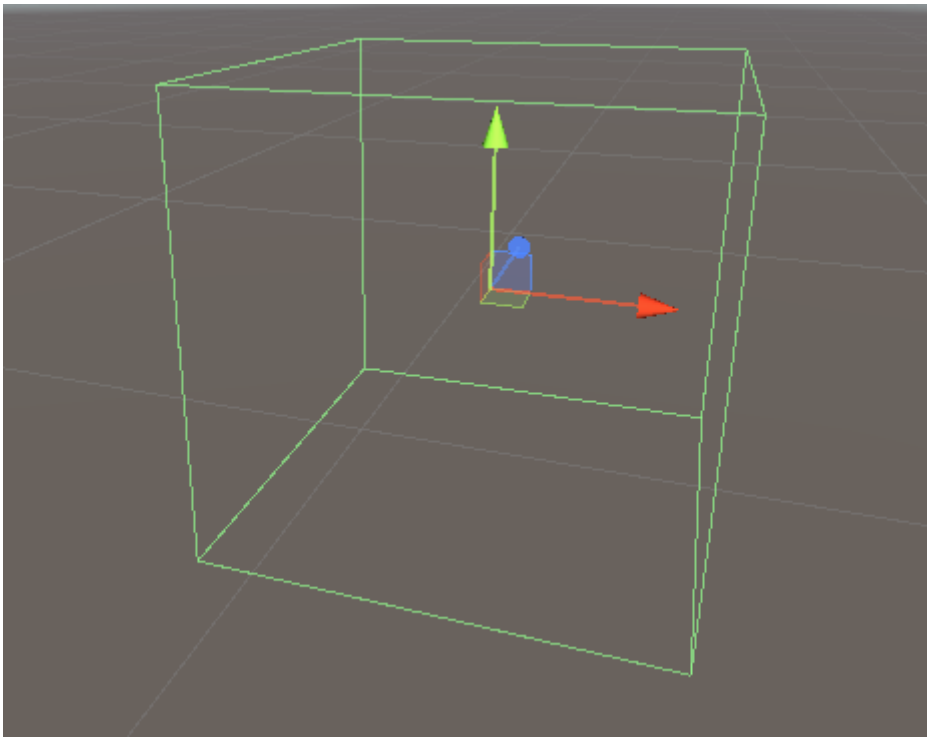
Primjereno mjesto za mijenjanje postavki fizike i primjene sile Rigidbody-a je unutar funkcije FixedUpdate, a ne Update. Razlog tome je što se ažuriranje fizike provodi u izmjerenim vremenskim koracima koji se ne podudaraju s ažuriranjem okvira odnosno Update funkcijom. FixedUpdate se poziva neposredno prije svakog ažuriranja fizike. [40, 41]

6.4.2. Colliders – otkrivanje detekcije

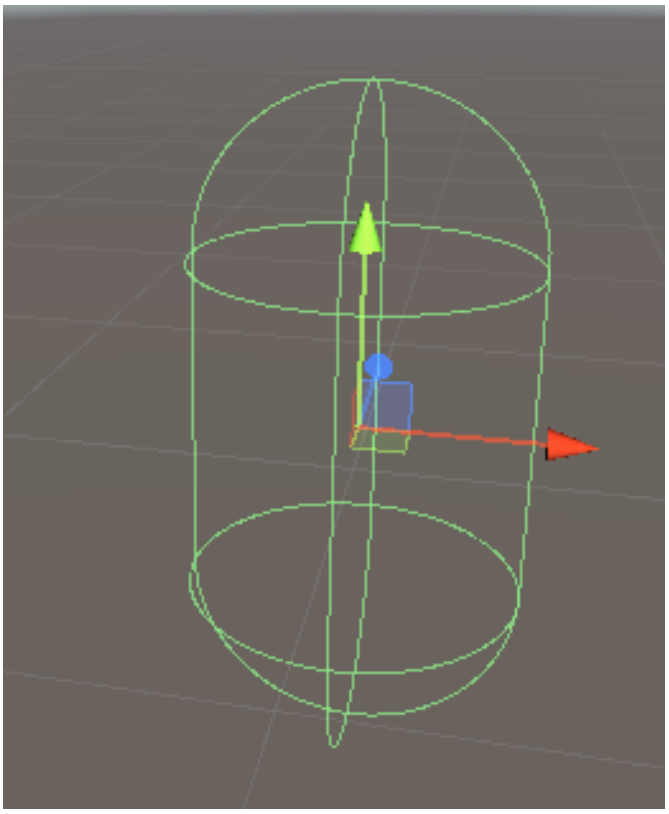
Prethodno je spomenuto i otkrivanje detekcije koje je jedan od ključnih komponenti GameObject-a svake igre. Kako se kroz objekt ne bi moglo prolaziti, kao na primjer zid, odnosno da bi objekt mogao biti detektiran on na sebi mora imati komponentu Collider. U tražilicu upišemo *Collider* te nam se prikažu vrste collider-a (Slika 27). Vrste 3D collider-a:

- Box Collider
- Capsule Collider
- Sphere Collider
- Terrain Collider
- Mesh Collider
- Wheel Collider

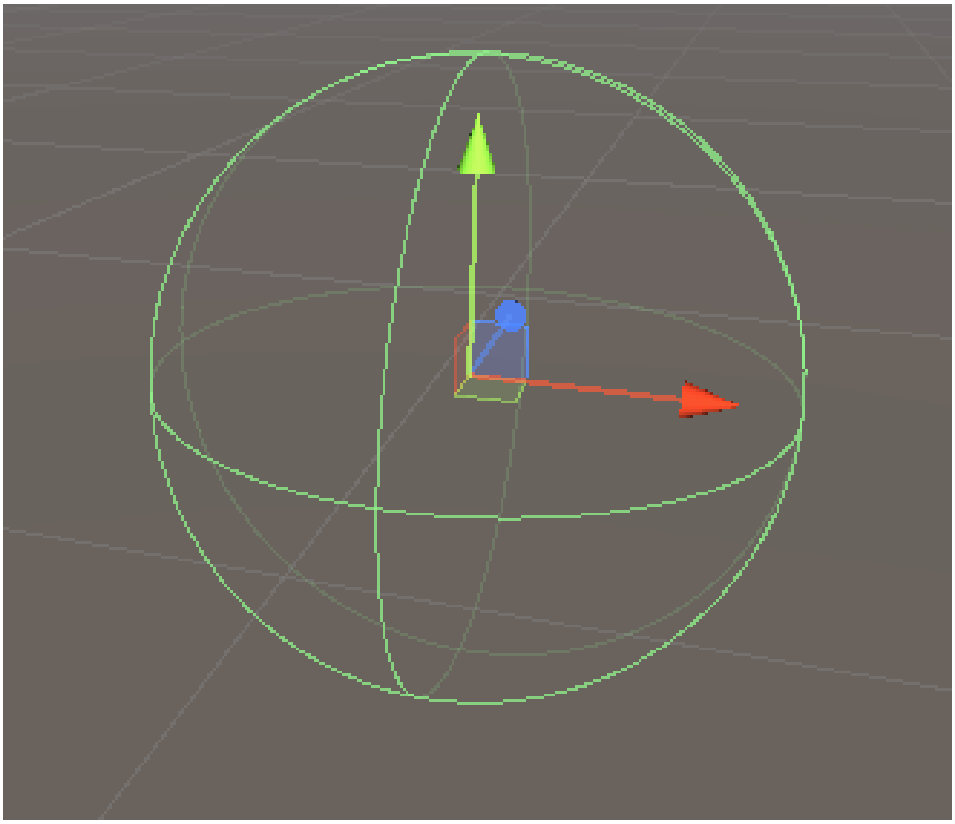
Box, Capsule, Sphere i Mesh Collider vidljivi su na slikama 23, 24, 25 i 26.



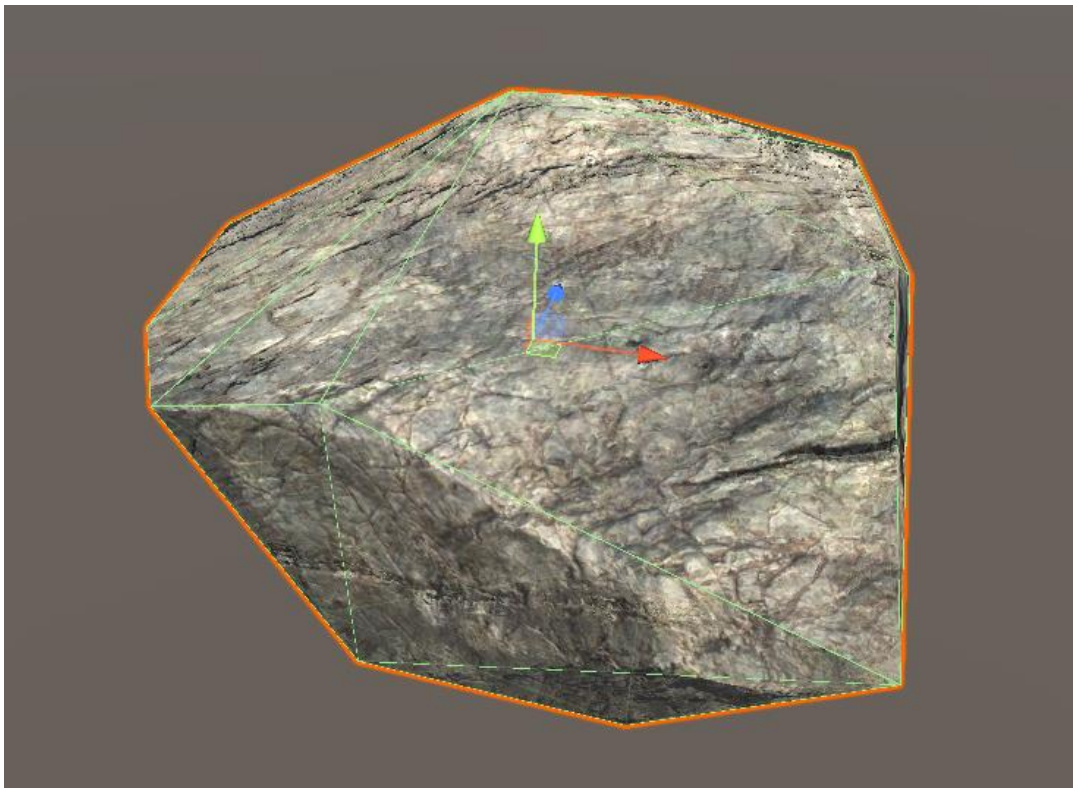
Slika 23 Box Collider



Slika 24 Capsule Collider



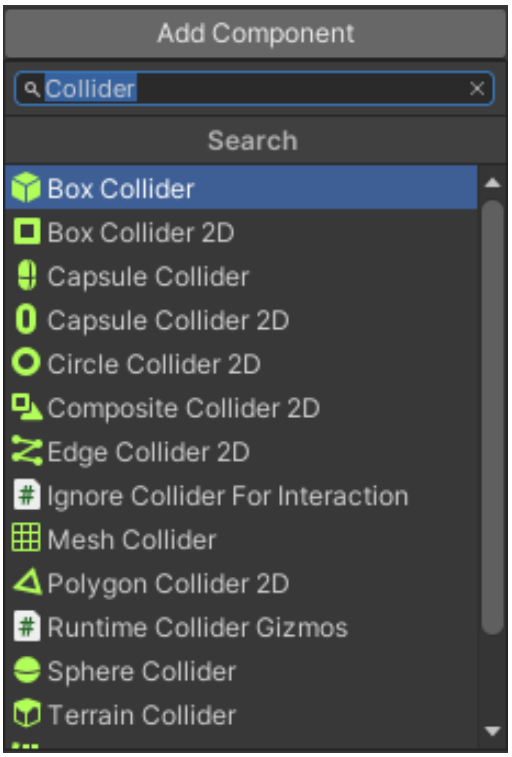
Slika 25 Sphere Collider



Slika 26 Mesh Collider

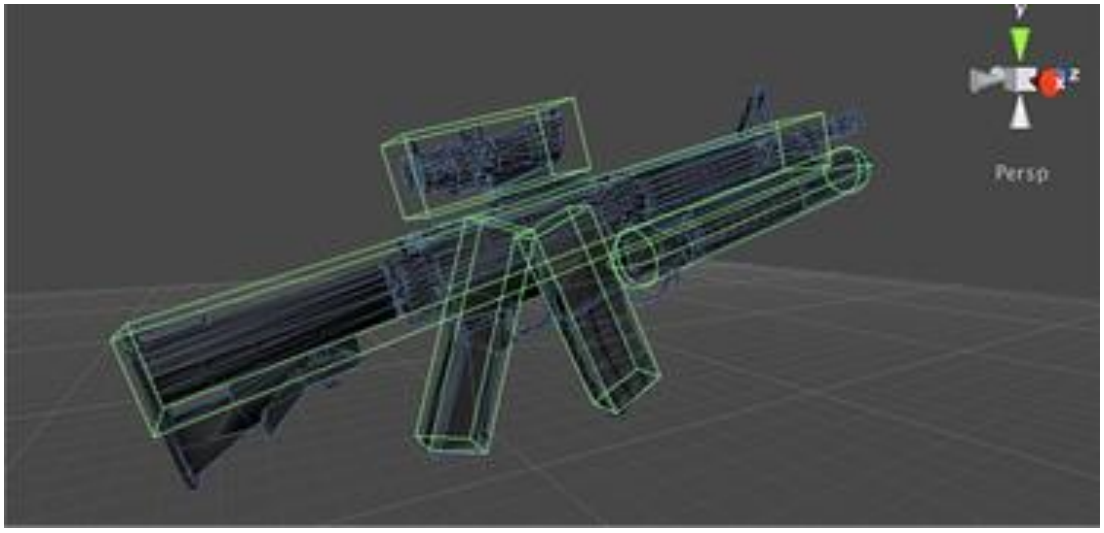
Vrste 2D collider-a:

- Box Collider 2D
- Capsule Collider 2D
- Circle Collider 2D
- Composite Collider 2D
- Edge Collider 2D
- Polygon Collider 2D
- Tilemap Collider 2D



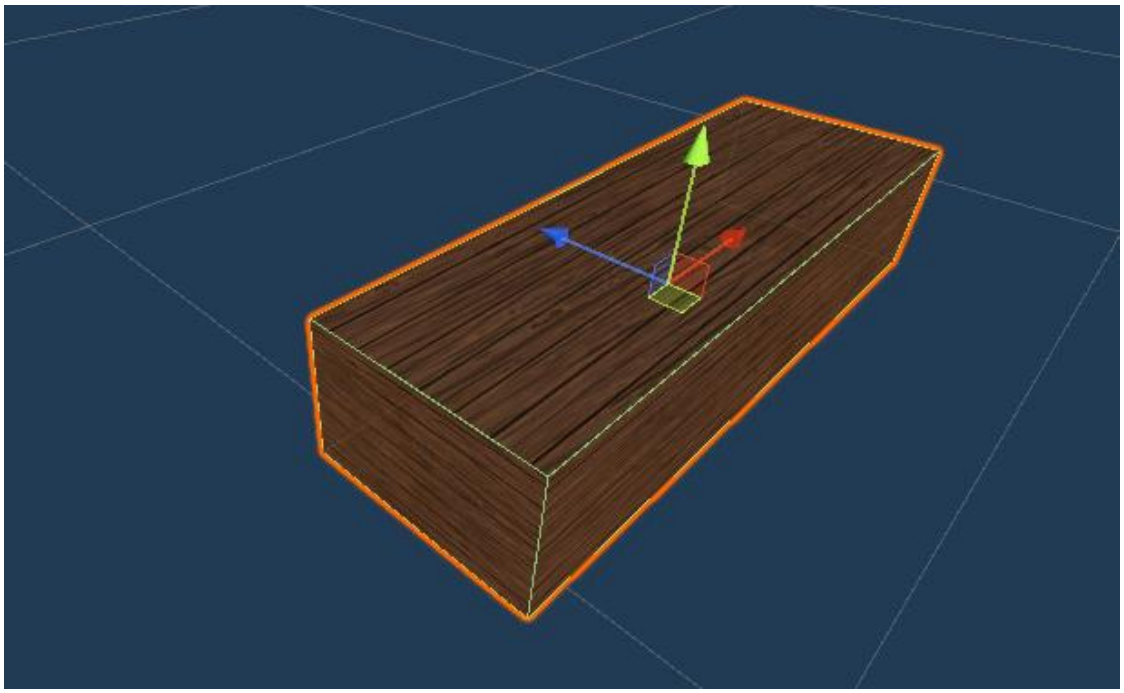
Slika 27 Odabir vrste collider-a

Odabir vrste collider-a ovisi o samom obliku objekta na koji dodajemo collider i vrsti igre. U nekim slučajevima ako se radi o kompleksnim objektima kao na primjer 2D liku potrebno je iskombinirati više collider-a kako bi se dobio točan oblik glave, tijela i nogu lika, ili za pušku kao što je vidljivo na slici 28.



Slika 28 Kombinacija collider-a

Kada se na objekt doda željeni collider, na samom objektu se pojave zelene linije koje obilježavaju granicu dodira (Slika 29). Te linije definiraju stvarni oblik predmeta koji se može spojiti odnosno ukoliko je kolizija ili dodirivanje uključeno, drugi predmet se može spojiti s trenutnim samo do te linije, ne može proći dalje od nje u predmet [42]. Za Jenga blokove korišten je Box Collider zato što su blokovi u obliku kvadra, te se Box Collider koristi za bilo koje objekte čiji oblik sadrži pravokutnike ili kvadrate, odnosno objekte u obliku kvadra ili kocke.



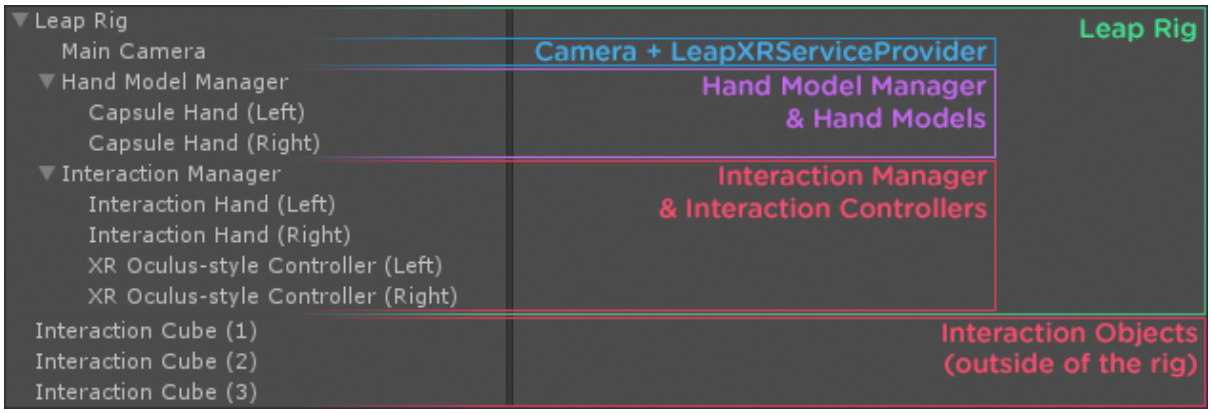
Slika 29 Jenga Box Collider

6.4.3. Interaction Engine

Za interakciju između ruku i objekata u našem radu, potrebno je uvesti prethodno preuzeti paket Leap Motion Interaction Engine. Interaction Engine omogućuje korisnicima

interakciju s objektima unutar XR aplikacije. XR – Proširena stvarnost (eng. Extended Reality) se odnosi na sva stvarna ili virtualna okruženja koju generira računalna grafika ili nosivi uređaji. X u XR-u je jednostavno varijabla koja može značiti bilo koje slovo. Neovisno o vrsti objekta – GameObjects - koja se nalazi u aplikaciji, korisnik mora moći lebdjeti u blizini, dodirnuti ih ili ih uhvatiti na neki način.

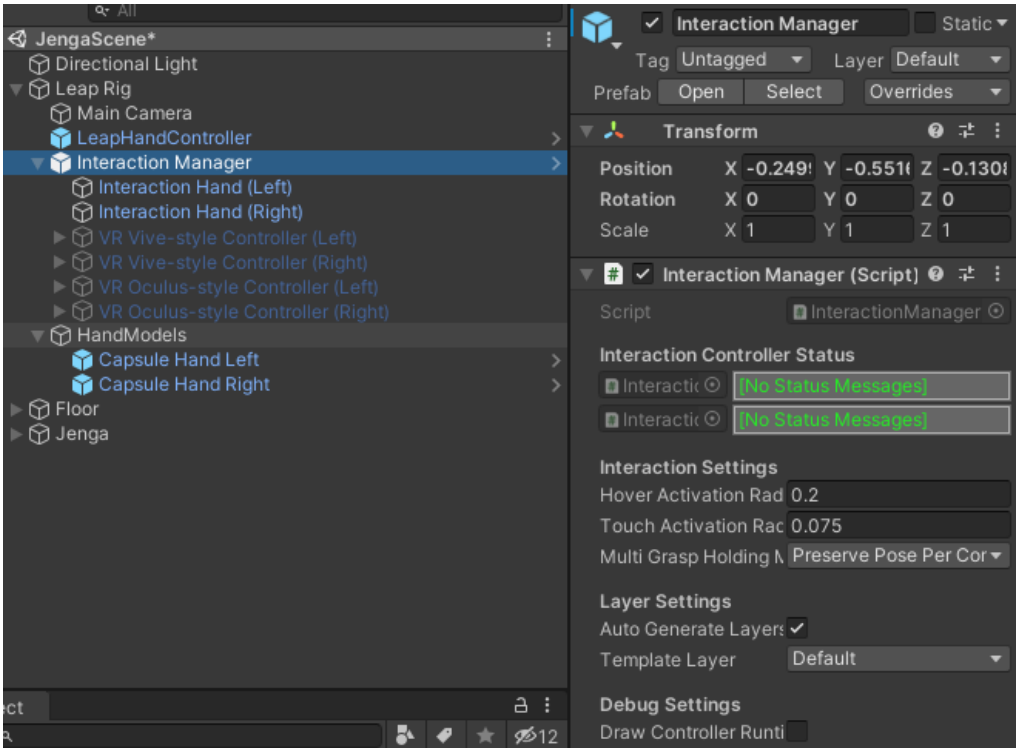
Osnovne komponente potrebne za interakciju između VR ruku odnosno pokreta detektiranih Leap Motion-om i objekata unutar igre su objekti interakcije, upravitelj interakcija te je unaprijed definirana hijerarhija komponenti od strane Leap motion-a (Slika 30).



Slika 30 Leap Motion definirana hijerarhija

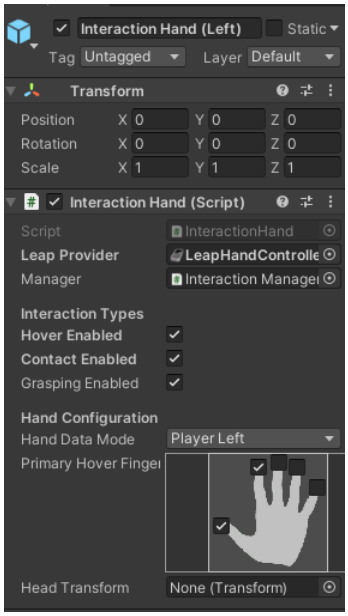
Objekti interakcije, je GameObjekti koji na sebi imaju kao komponentu skriptu InteractionBehaviour u kojem je definirano ponašanje objekta. Objekt interakcije mora imati Rigidbody i najmanje jedan Collider. Objekti interakcije mogu biti smješteni bilo gdje unutar scene, dok god postoji aktivan upravitelj interakcija.

Unutar Prefabs foldera nalazi se Upravitelj Interakcija (eng. Interaction Manager) koji povlačimo u naš projekt. Upravitelj interakcija prima FixedUpdate od Unity-a i obrađuje svu unutarnju logiku koja omogućuje interakcije, uključujući ažuriranje podataka ruku – kontrolera i podataka objekta interakcije. Svaki Kontrolor Interakcije (eng. Interaction Controller) vrši sve stvarne interakcije s objektima, bilo da ih podiže, dodiruje, udara ili je jednostavno u njihovoj blizini, u ovom slučaju to se odnosi na komponente Interaction Hand (Left and Right) i XR kontrolere vidljive na slici 31. U hijerarhiji, kontrolori interakcije uvijek trebaju biti ispod Upravitelja interakcija, dok bi Upravitelj interakcija uvijek trebao biti brat/sestra objekta kamere kako kontrolori ne bi naslijedili neobične brzine ukoliko se igrača oprema pomakne „oko“ [21].



Slika 31 *InteractionManager i hijerarhija kontrolera*

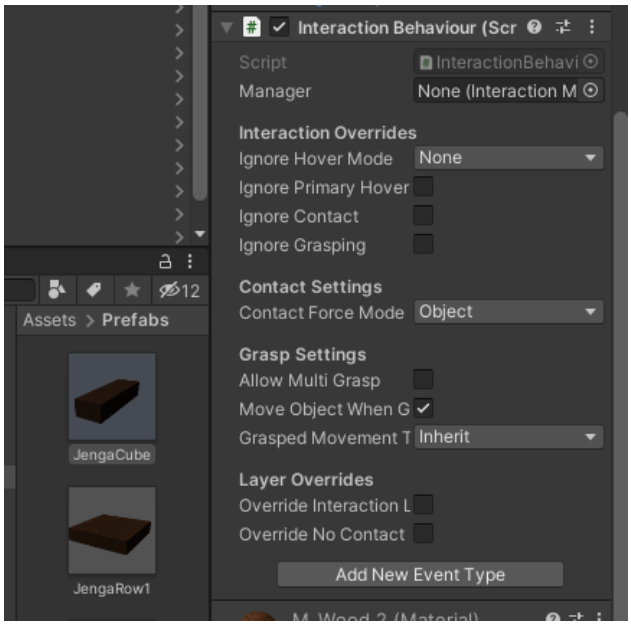
Kod kontrolora ruku potrebno je označiti omogućeno „lebdenje“ (eng. Hover Enabled), omogućen kontakt (eng. Contact Enabled) i omogućeno hvatanje (eng. Grasping Enabled), te konfigurirati ruke tako da su nam primarni prsti koji se koriste za sve tipove interakcije palac i kažiprst (Slika 32).



Slika 32 Interaction Hand (Left)

Objekti interakcije su GameObjekti s priloženom InteractionBehaviour skriptom, te moraju imati Rigidbody i barem jedan Collider, u Jengi to su blokovi. Objekti interakcije mogu

biti bilo gdje u sceni, sve dok je aktiviran Upravitelj interakcija. Na prethodno stvoreni prefab JengaBlock dodajemo komponentu InteractionBehaviour (Slika 33). Za interakciju koja je potrebna u Jengi nije potrebno dodatno podešavanje postavki od onih standardnih.



Slika 33 InteractionBehaviour

Kada se objektu doda komponenta odnosno skripta InteractionBehaviour automatski se dogodi nekoliko stvari:

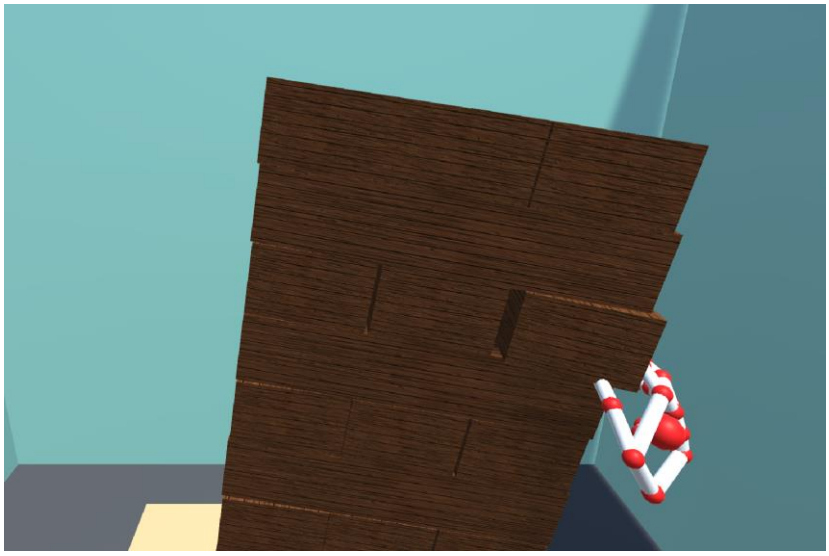
- Ukoliko prethodno nije imao, objekt će dobiti komponentu Rigidbody s omogućenom gravitacijom, što ga čini fizički simuliranim objektom
- Ako objekt nema Collider pasti će kroz pod, no prilikom kreiranja 3D objekta svaki oblik dobije pripadajući Collider
- Pod pretpostavkom da projekt ima Upravitelja interakcija, objekt će se moći podići, pomaknuti i udariti objektom ruka ili XR kontrolerom

Osim postavljanja gravitacije objekta, potrebno je podesiti i gravitaciju unutar same aplikacije. Pri radu sa fizičkim objektima preporučuje se podešavanje gravitaciju unutar Uređivanje → Postavke projekta → Fizika na -4.905 radi boljeg osjećaja prilikom korištenja aplikacije. Osim gravitacije, prilikom izrade XR aplikacija potrebno je podesiti i vremenski korak. Unity-ov fizički mehanizam ima „fiksni vremenski korak“ a taj vremenski korak nije uvijek sinkroniziran s brzinom sličica grafike. Vrlo je važno da se vremenski korak fizike podesi na isti kao i brzina prikazivanja. Ukoliko se za prikazivanje koriste uređaji Oculus ili Vive vremenski korak je potrebno podesiti na 0,0111111 što je jednako 90 sličica u sekundi, te se ono postavlja u Uređivanje → Postavke projekta → Vrijeme [21].

Isprobali smo i testirali tri vrste materijala fizike: sklisko, ljepljivo i osrednje. Svaki materijal fizike ima podesivo dinamičko i statičko trenje i poskakivanje (eng. Bouncines), kombinaciju trenja i kombinaciju poskakivanja. Dinamičko trenje je trenje koje se koristi kada

se objekt već kreće. Obično je vrijednost od 0 do 1, gdje je nula poput leda, dok će 1 vrlo brzo umirit objekt osim ako ga neka velika sila ili gravitacija ne gurnu. Statičko trenje je trenje koje se koristi kada objekt miruje na površini. Obično je vrijednost od 0 do 1, gdje je nula, kao i kod dinamičkog trenja, poput leda, dok će vrijednost 1 jako otežati pokretanje objekta. Poskakivanje je samo po sebi razumljivo, ono određuje koliko je poskakivanje objekta, 0 objekt neće odskočiti, 1 objekt će odskočiti bez ikakvog gubitka energije. Kombinacije trenja i poskakivanja određuju kako se kombinira trenje/poskakivanje dvaju udarajućih objekata, te njihove vrijednosti mogu biti minimalne, maksimalne, prosječne i pomnožene [23].

Sa skliskom fizikom blokovi bi lagano kliznuli van već pri prvom dodiru sa objektom, dok je kod ljepljive fizike situacija bila obrnuta, zbog povećanog dinamičkog i statičkog trenja blok bi bilo toliko teško pomaknuti da bi on povukao za sobom i sve ostale blokove te srušio toranj (Slika 34). Najveću uspješnost prilikom testiranja imala je osrednja fizika, zlatna sredina između skliskog i ljepljivog. Dinamičko i statičko trenje postavljeno je na 0.5, poskakivanje na 0, a kombinacije trenja i poskakivanja su obje postavljene na minimum.

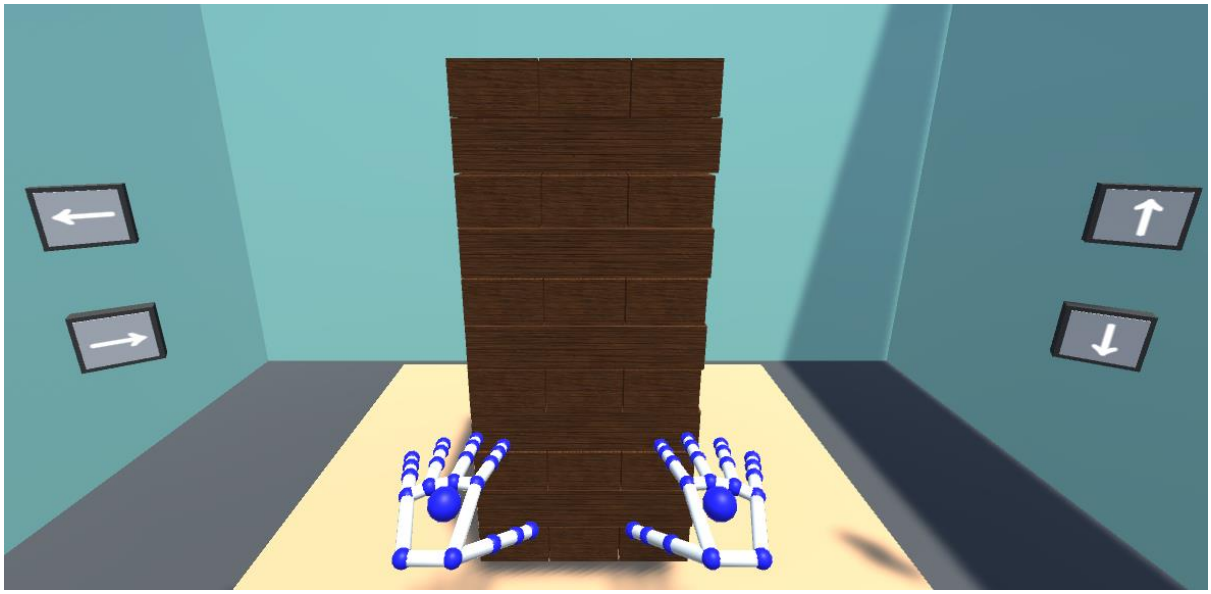


Slika 34 Ljepljivi materijal fizike

1.1. Kretanje unutar igre

U većini igara virtualne stvarnosti koriste se XR kontroleri zajedno sa zaslonima za glavu koji onda omogućuju kretanje i rotaciju unutar igre regulirana gumbima i okretajima glave. U nedostatku opreme, odnosno zaslona za glavu, opremljeni smo sa statičnim Leap Motion kontrolerom kojemu je za kretanje potrebno korištenje gumbića ili gesti. Zbog prostora s kojim raspolažemo odnosno pozicije tornja i ruku, kako ne bi došlo do „šamaranja“ tornja prilikom gestikulacija, odlučeno je koristiti se gumbima.

Stvorena su dva gumba sa svake strane, jedni za rotaciju oko tornja, drugi za pomicanje pozicije gore i dolje (Slika 35). Oni su u hijerarhiji smješteni odmah ispod glavne kamere iz razloga što je pokretnom kamere potrebno pomicanje i samih gumbića.



Slika 35 Gumbi za rotaciju i promjenu pozicije

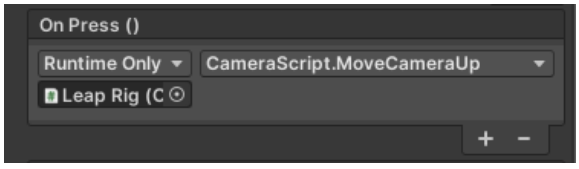
Osim promjene pozicije kamere potrebno je pomaknuti i cijelu Leap Motion konstrukciju (modele ruku, kontroler, upravitelj interakcija). Na objekt Leap Rig dodajemo skriptu pod nazivom „Camera Script“ u kojoj definiramo 4 funkcije:

- MoveCameraUp()
- MoveCameraDown()
- RotateLeft()
- RotateRight()

Za prve dvije funkcije koristimo funkciju Translate() koji pomiče transformaciju u određenom smjeru određenu udaljenost. Pokret se primjenjuje u odnosu na lokalni koordinatni sustav drugog parametra, a u ovom slučaju to je Leap Rig na kojem se nalazi skripta. Za druge dvije funkcije odnosno rotaciju kamere potrebno je dodati novi prazan objekt i pozicionirati ga u središte Jenge. Zatim koristimo funkciju RotateAround koja rotira Leap Rig okolo Jenge u traženom smjeru određenom brzinom odnosno određenu udaljenost.

```
public void MoveCameraUp() {
    transform.Translate(Vector3.up * moveSpeed * Time.deltaTime, transform);
}

public void MoveCameraDown() {
    transform.Translate(Vector3.down * moveSpeed * Time.deltaTime, transform);}

public void RotateLeft() {
    transform.RotateAround(Jenga.transform.position, Vector3.up, rotateSpeed *
Time.deltaTime);
}

public void RotateRight() {
    transform.RotateAround(Jenga.transform.position, Vector3.up, -
rotateSpeed * Time.deltaTime);
}
```

Nakon toga potrebno je povezati funkcije s gumbovima. Svaki gumb sadrži Leap Motion-ovu „Interaction Button“ skriptu u kojoj postavljamo što će se dogoditi prilikom pritiska gumba. Stišćemo + za dodavanje nove akcije te povlačimo Leap Rig koji sadrži skriptu u kojoj se nalaze naše akcije, te za gumb prema gore odabiremo odgovarajuću funkciju. Proces je isti za sve gumbe (Slika 36).



Slika 36 OnPress akcije gumba

1.1.1. Bodovanje

Iako u stvarnoj Jengi ne postoji bodovanje, odlučili smo implementirati bodovanje u našu verziju Jenge. Ideja bodovanja je vrlo jednostavna, kada igrač uspješno spusti blok na vrh tornja, dobije 1 bod te je na potezu sljedeći igrač.

Razvijeno je i istraženo nekoliko načina na koji se to može postići:

- Pratiti koji se blokovi nalaze u najgornjem redu, napraviti funkciju koja sluša kada se gornja ploha jednog od tih blokova počne dodirivati s drugim blokom. U tom trenutku potrebno je dodati bod, označiti novi blok kao blok koji se nalazi u najgornjem redu, te maknuti taj parametar sa prošlog bloka. Ta ideja je naišla na dosta problema s obzirom da je Jenga blok prefab, te svaki od blokova imaju iste komponente, uključujući i skriptu. Zbog toga je jako teško razlikovati odnosno pronaći najgornji red. Iz istog razloga jako je teško slušati dodirivanje s gornjim dijelom bloka, jer se to slušanje odvija na svakom bloku.

- Izračunati udaljenost između početno postavljenog bloka i pozicije na koju je on pušten. Za to je potrebno izračunati početnu poziciju svakog pojedinog bloka te ju spremiti u varijablu. Nakon toga potrebno je prepoznati koji je blok uhvaćen, te trenutak u kojem je on ispušten. U trenutku kada je ispušten izračunati poziciju i usporediti sa početnom pozicijom. Ukoliko je udaljenost veća od određenog iznosa dodati bod. Ovaj način je pokazao da se ide u boljem smjeru i ostvario bolje rezultate, no ne dovoljno dobre. Pokazalo se da izračun pozicije jenga bloka u prostoru nije dovoljno točno, varira i mijenja se, te su zbog toga rezultati nestabilni. Ponekad se doda bod, ponekad ne.
- Izmjena prethodnog načina, i finalan način kojim je implementirano bodovanje je sljedeći: Izračunati udaljenost između početno postavljenog bloka i pozicije na koju je on pušten, ali izračunati početnu poziciju na način da se početna pozicija izračuna kao udaljenost jenge od određenog statičnog elementa. Na taj način pozicija ne varira, i rezultati su puno bolji.

Sukladno s odabranim načinom primjene bodovanja prvo je potrebno saznati početne pozicije svakog pojedinačnog jenga bloka na početku igre. Za statični element u odnosu na koji ćemo izračunavat udaljenost odabran je pod, te je na njega postavljen tag „Floor“ kako bi ga se unutar skripte moglo dohvatiti, te to radimo u skripti *JengaPointsScript* koja je komponenta Jenga bloka:

```
        Vector3 direction = ground.transform.position - transform.position;
        jengaStartHeight = direction.y;
```

Sljedeći korak je prepoznavanje dohvaćenog bloka. Stvaramo novu skriptu koja će nam pomoći sa prepoznavanjem interakcije s blokovima, dodajemo ju kao komponentu jenga blokova i nazivamo ju *JengaInteractionPointsScript*. Kako bi to implementirali potrebno je koristiti biblioteku Leap Unity-a, konkretno *Leap.Unity.Interaction*. Ta biblioteka nam omogućuje da dohvatimo komponentu koja na sebi ima *InteractionBehaviour* skriptu, odnosno pristupimo podacima koji se izračunavaju uz pomoć nje. Podsjetimo se, svaki objekt s kojim naše ruke dolaze u doticaj moraju na sebi imati navedenu skriptu kako bi se njima moglo upravljati odnosno utjecati na njih. Svaki naš Jenga blok na sebi ima *InteractionBehaviour* skriptu. U Start funkciji dohvaćamo Jenga blok, InteractionBehaviour komponentu bloka koju spremamo pod nazivom *_intObj* iz razloga što je tako definirana u ostalim Leap Motion skriptama, te *JengaPointsScript* skriptu kojoj ćemo proslijediti određene parametre kako bi mogla odrediti udaljenost.

```
void Start() {
  _intObj = GetComponent<InteractionBehaviour>();

  jengaPointsScript = _intObj.GetComponent<JengaPointsScript>();

  jenga = this.gameObject;
}
```

Kako je cijelo vrijeme potrebno osluškivati u kojem trenutku je blok uhvaćen i u kojem je trenutku ispušten, tu funkcionalnost potrebno je implementirati unutar Update funkcije. InteractionBehaviour komponenta, odnosno _intObj na sebi ima parametar pod nazivom isGrasped, koji je tipa bool, te kao što sam naziv govori pokazuje ako je predmet na kojem se nalazi skripta dohvaćen odnosno primljen rukama, ako je, vrijednost je true. No nama ne treba trenutak u kojem je on dohvaćen, nego trenutak u kojem je taj predmet ispušten. Prva pomisao bi možda bila provjeriti ako je isGrasped false te onda proslijediti poziciju, no ta pomisao bi bila loša. Vrijednost isGrasped je false kada god naše ruke ne hvataju blok, a skripta se nalazi na svakom pojedinom jenga bloku, te bi ova provjera rezultirala konstantnim dodavanje bodova kad god je isGrasped false. Rješenje za taj problem je u trenutku kada je blok uhvaćen postaviti novu pomoćnu varijablu *hasPickedUpJenga* na true. Ta varijabla će biti istinita samo na GameObject-u na kojem je _intObj.isGrapsed istinit, odnosno na Jenga bloku koji smo uhvatili našim rukama.

```
if (_intObj.isGrasped) {
    hasPickedUpJenga = true;
}
```

Ako je blok koji smo uhvatili pušten, javljamo *JengaPointScript* da je pušten, prosljeđujemo poziciju te postavljamo *hasPickedUpJenga* na false kako bi omogućili detekciju odnosno bodovanje sljedećeg hvatanja bloka.

```
if (!_intObj.isGrasped && hasPickedUpJenga) {
    Debug.Log("Picked up Jenga has been released.");
    jengaPointsScript.releasedObject = true;
    jengaPointsScript.jengaDropPosition = jenga.transform.position;
    hasPickedUpJenga = false;
}
```

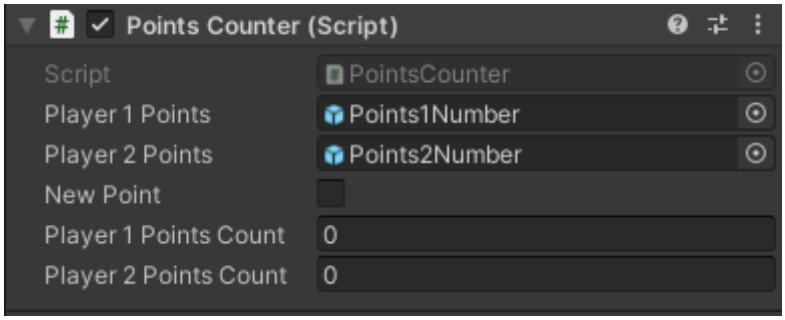
Sljedeći korak je izračunati razliku udaljenosti. Prvi izračun odnosi se na udaljenost nove pozicije jenga bloka od poda. Nakon toga izračunavamo razliku između početne visine bloka i visine na kojoj je blok ispušten. Prilikom određivanja dovoljne udaljenosti za dodavanje boda potrebno je uzeti u obzir predzadnji red jer on ima najmanju udaljenost od pozicija koje se nalaze iznad zadnjeg reda. Optimalna udaljenost, odnosno optimalna razlika između početne i ispuštene visine se tijekom testiranja pokazala kao udaljenost od 0.17. Ako je razlika veća od 0.17 u skripti koja će biti zadužena za prikazivanje bodova na ekranu postavljamo vrijednost pomoćne varijable *newPoint* na true, te releasedObject postavljamo na false.

Jenga blok je postavljanjem na vrh dobio i novu „početnu poziciju“. Iz tog razloga potrebno je ažurirati njegovu postojeću visinu sa novom visinom.

```
if (releasedObject) {
   Vector3 direction = ground.transform.position - jengaDropPosition;
   jengaDropHeight = direction.y;

   differenceBetweenHeight = jengaStartHeight - jengaDropHeight;

   if (differenceBetweenHeight > 0.17 ) {
      Debug.Log("Jenga put on top.");
      pointsCounter.newPoint = true;
      releasedObject = false;

      // Get new height - new position of jenga
      Vector3 newDirection = ground.transform.position - transform.position;
      jengaStartHeight = newDirection.y;
   }
}
```

Unutar novo napravljene skripte *PointsCounter* odvijat će se sva logika za dodavanje bodova i prikazivanje bodova na ekranu.

U projektu stvaramo prefab *PointsCounter* koji se sastoji od naziva igrača i broja bodova. Povlačimo dva objekta u scenu te ih smještamo kao dijete kamere kako bi njihova pozicija bila fiksna, odnosno da bi igrač mogao vidjeti rezultat svo vrijeme ne ovisno o okretanju kamere. Brojače postavljamo u poseban prazan GameObject odnosno direktorij pod nazivom PointsCounters na koji dodajemo skriptu PointsCounter. Kako bi dodali bodove pojedinom brojaču potrebno je dohvatiti njihovu komponentu koja prikazuje broj bodova. Stvaramo dvije public varijable za brojače te njihovu početnu vrijednost postavljamo na 0, te na PointsCounters povlačimo pripadajuće brojače. (Slika 37)



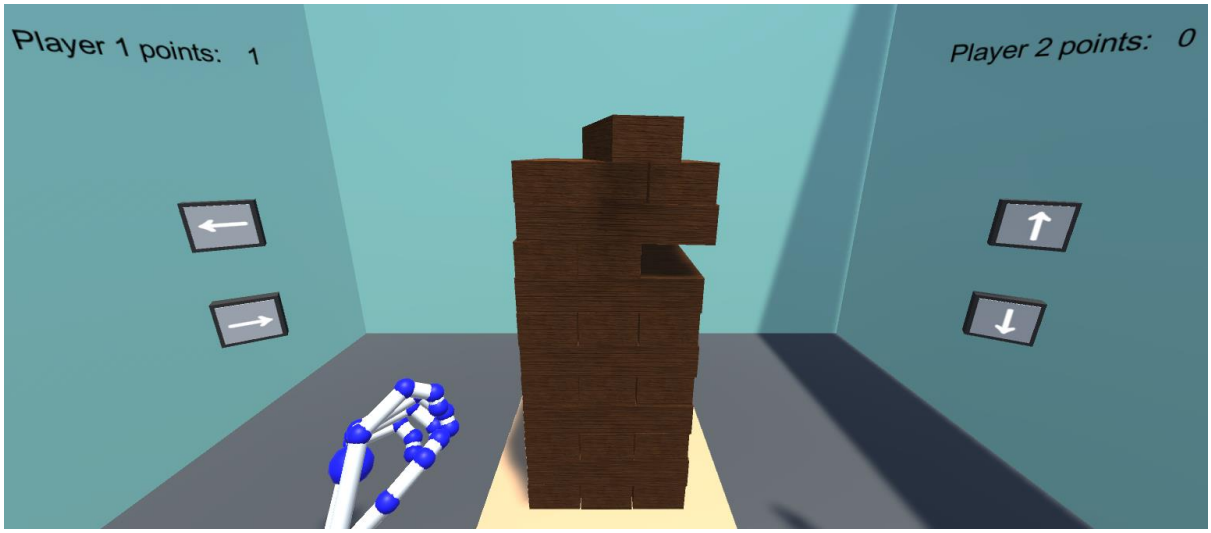
Slika 37 Početne postavke brojača

Zatim u Update funkciji provjeravamo ako je došlo do dodavanje novog boda, te ako je kome se pridodaje. Dodavanje bodova ćemo regulirati uz pomoć pomoćne varijable *player1GetsAPoint* čija je početna vrijednost true. Bodovanje kreće od igrača koji prvi igra, odnosno bod se prvo dodjeljuje njemu, iz tog razloga smo vrijednost pomoćne varijable postavili na true. Ako igrač 1 dobiva bod, povećavamo njegov brojač za 1, dohvaćamo text komponentu koja se nalazi na njegovom objektu i prikazujemo novu vrijednost odnosno novo stanje bodova.

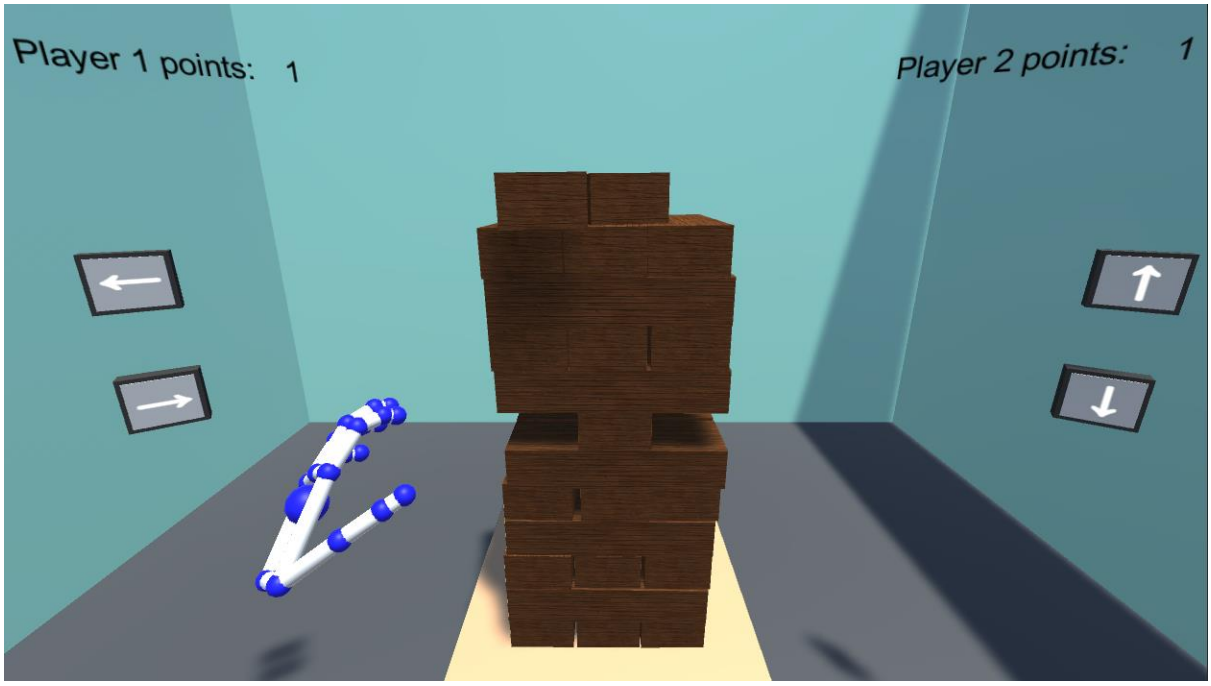
Kako bi omogućili da sljedeći bod dobije drugi igrač postavljamo *player1GetsAPoint* na false. U trenutku kada se ostvari sljedeći bod, iz razloga što smo prethodnu varijablu postavili na false, povećava se brojač drugog igrača za 1, te se identično dohvaća njegova komponenta i prikazuje novo stanje bodova. Jedina razlika je što u ovom trenu *player1GetsAPoint* postavlja na true kako bi sljedeći bod mogao ići prvom igraču. U oba slučaja potrebno je *newPoint* postaviti na false kako bi se izbjeglo beskonačno dodavanje bodova.

```
    void Update()
    {
        if (newPoint) {
            if (player1GetsAPoint) {
                player1PointsCount++;
                player1Points.GetComponent<TextMesh>().text = player1PointsCou
nt.ToString();
                player1GetsAPoint = false;
                // player2GetsAPoint = true;
                newPoint = false;
            } else {
                player2PointsCount++;
                player2Points.GetComponent<TextMesh>().text = player2PointsCou
nt.ToString();
                // player2GetsAPoint = false;
                player1GetsAPoint = true;
                newPoint = false;
            }
        }
    }
```

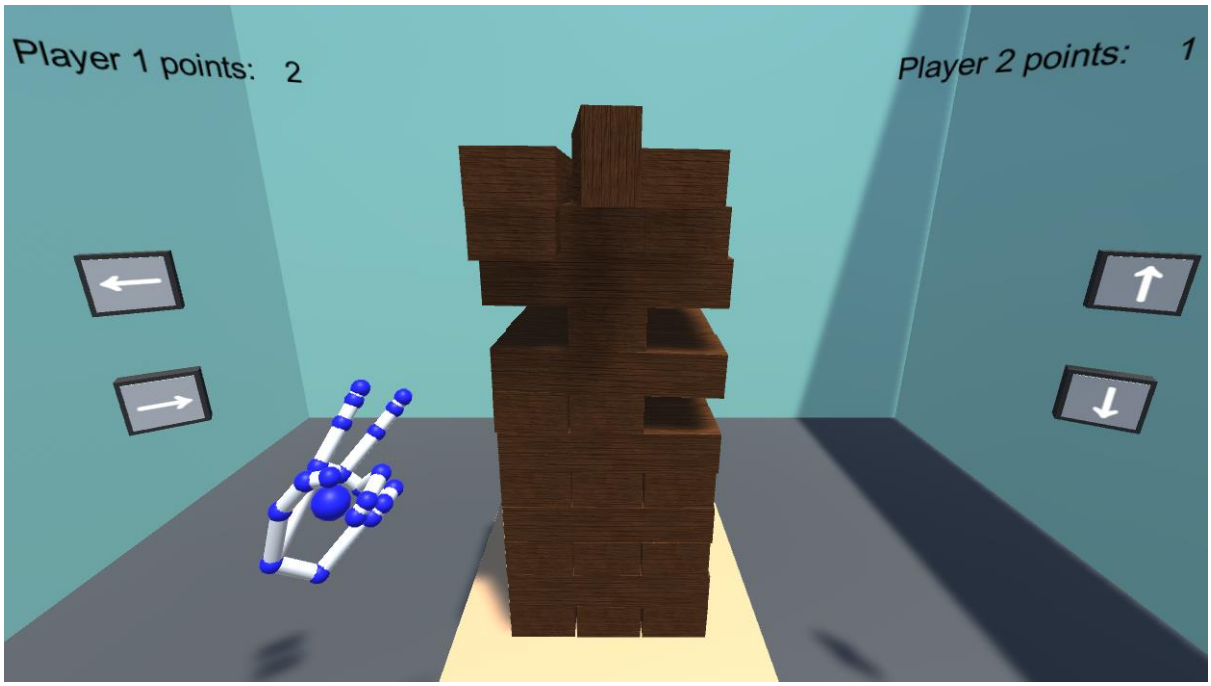
Implementacija bodovanja je završena, te je rezultat vidljiv na slikama 38, 39 i 40. Broj bodova na rezultatima nije veći iz razloga što je igra koristeći Leap Motion izuzetno teška.



Slika 38 Bodovanje



Slika 39 Bodovanje 2



Slika 40 Bodovanje 3

1.1.2. Kraj igre

U trenutku kada jedan ili više Jenga blokova dotaknu pod, igra je gotova, a igrač koji je zadnji igrao odnosno kojemu se toranj srušio je izgubio.

Svaki Jenga Blok ima svoj Collider odnosno Rigidbody, te jedna od funkcija koje se koriste za upravljanje i kontroliranje događaja objekata je funkcija OnCollisionEnter. Ona se poziva kada jedan objekt počne dodirivati drugi objekt, što u našem slučaju predstavlja situaciju kada blok dotakne pod.

```
    private void OnCollisionEnter(Collision collision) {
        if (collision.gameObject.tag == "JengaBrick") {
            Debug.Log("Brick touched the floor.");

            // Remove instructions if still on the screen
            InstructionsUI.gameObject.SetActive(false);

            // Show UI "Game Over" Label and Restart Button
            GameOverUI.gameObject.SetActive(true);
        }
    }

    public void RestartGame() {
        hasRestarted = true;
        SceneManager.LoadScene("JengaScene");
    }
```

U trenutku kad blok dotakne pod, na ekranu se prikazuje poruka koja obavještava igrača kako je igra završila, te da pritiskom na „R“ gumb (koji se pojavljuje usporedno s obavijesti) može ponovno pokrenuti igru (Slika 41). Pritiskom na navedeni gumb, toranj se resetira te igra počinje ispočetka.

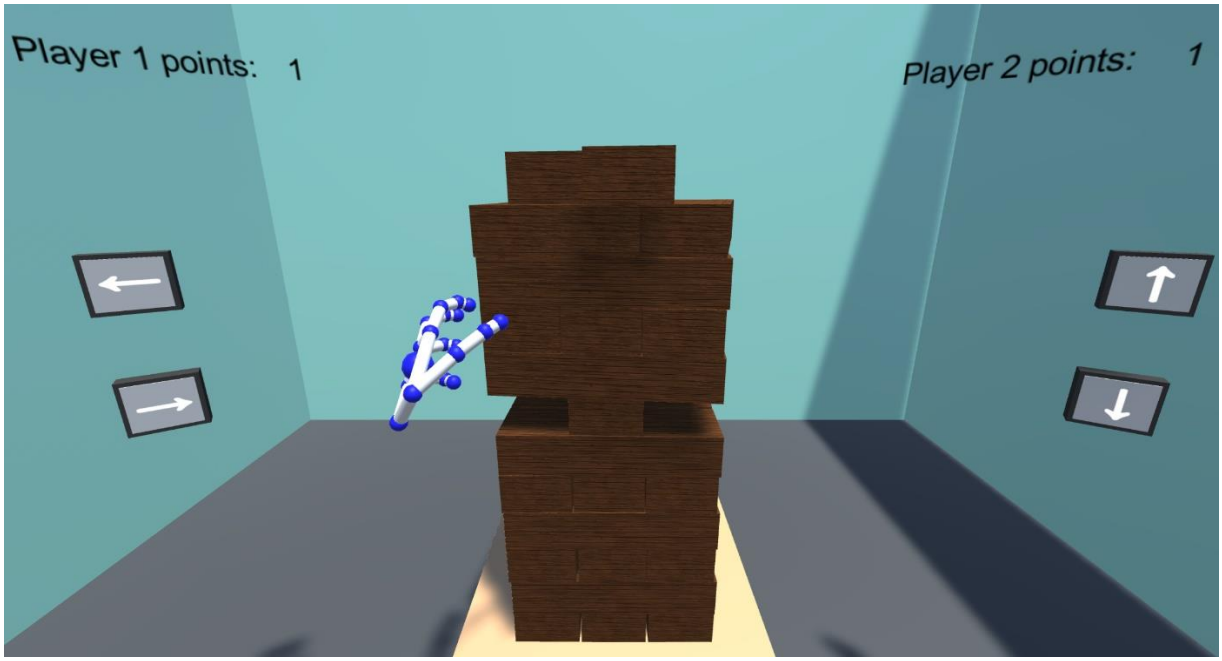
Osim poruke koja se ispisuje bila je potrebna i mala nadogradnja sustava za dodavanje bodova, točnije bilo je potrebno onemogućiti daljnje dodavanje bodova na način da se u provjeri ako je *newPoint* istinit provjerava i ako nije kraj igre odnosno *!gameOver*. Prethodno je potrebno postaviti *gameOver* varijablu na true u trenutku kada blok dodirne pod.

```
        if (collision.gameObject.tag == "Floor") {
            Debug.Log("Brick touched the floor. ");

            pointsCounter.gameOver = true;
            pointsCounter.newPoint = false;
        }
```

Slika 41 Game Over Screen

Uz objašnjenje kraja igre također su napravljene i početne instrukcije prilikom pokretanja igre. Umjesto klasičnog izbornika s postavkama na početnom ekranu nalaze se kratke upute za igranje te igra kreće. Upute se na ekranu nalaze prvih 8 sekundi nakon čega nestaju, i ne pojavljuju se prilikom ponovnog pokretanja igre gumbom. Računalna igra Jenga je gotova (Slike 42 i 43).

Slika 42 Instrukcije



Slika 43 Finalni izgled

2. Zaključak

Cilj ovog diplomskog rada bio je istražiti područje virtualne stvarnosti, koje se u informatičkim znanostima i tehnologiji u posljednjem desetljeću, pogotovo u industriji video igara i obrazovanja, nameće kao sve popularnija tehnologija. Kao područje interesa za preddiplomski rad, kojem je tema bila izrada košarkaške simulacije u Unity-u, virtualna stvarnost i izrada računalne igre primjenom predmetne tehnologije nametnula se kao prirodan slijed razvoja vlastitih interesa.

Jenga kao računalna igra ima puni potencijal za implementaciju tehnologije virtualne stvarnosti, no zbog osobnog manjka sve potrebne opreme, igra koja je predstavljena u ovome radu obilježava statičnost. U svojem potpunom razvoju, iskustvo igranja Jenge u virtualnoj stvarnosti upotpunjavalo bi korištenje drugih uređaja VR-a, poput kontrolera i zaslona za glavu koja bi igraču omogućila da jednostavnim pokretima glave istražuje prostor igrice oko sebe. Pomoću kontrolera, igrač bi se kretao kroz samu igru bez korištenja gumba te jednostavno rotirao kameru, a uklanjanje samih blokova bilo bi jednostavnije i prirodnije. Međutim, s obzirom na navedeni manjak opreme, iste su se funkcije u igru ugradile pomoću programiranja posebnih gumba koje igraču omogućuju da rotiraju kameru. Ipak, unatoč tome, radom je prikazano koliko potencijala virtualna stvarnost ima u industriji video igara te koliko ista može povećati iskustvo za igrača, a samo iskustvo predstavlja najvažniju značajku tog proizvoda.

3. Literatura

[1] https://www.knowledgenile.com/blogs/virtual-reality-history-complete-timeline-explained/ Pronađeno 3.7.2021

[2] https://en.wikipedia.org/wiki/Power_Glove Pronađeno 4.7.2021

[3] https://en.wikipedia.org/wiki/Super_Glove_Ball Pronađeno 4.7.2021

[4] https://en.wikipedia.org/wiki/Virtual_reality_game Pronađeno 6.7.2021

[5] https://en.wikipedia.org/wiki/Virtuality_(product) Pronađeno 9.7.2021

[6] https://en.wikipedia.org/wiki/Superhot Pronađeno 12.7.2021

[7] https://beatsaber.com/ Pronađeno 12.7.2021

[8] https://www.youtube.com/watch?v=dji9YiPZ4AM Pronađeno 9.7.2021

[9] Animacija sportskih akcija u računalnoj igri, Mršić Lorena, Lipanj 2019. Pronađeno 13.6.2021

[10] https://en.wikipedia.org/wiki/Unity_(game_engine) Pronađeno 11.6.2019

[11] http://gamedev.machina.hr/unity-3d-game-engine-tvornica-igara-indie-studija/ Pronađeno 11.6.2019

[12] VR sustavi za praćenje - https://www.mechatech.co.uk/journal/how-do-common-virtual-reality-tracking-systems-work Pronađeno 15.7.2021

[13] https://www.kickstarter.com/projects/mechatech/agilevr-immersive-locomotion-controller-for-virtual-reality Pronađeno 15.7.2021

[14] https://www.vrfitnessinsider.com/agilevr-virutal-reality-exoskeleton-kickstarter/ Pronađeno 15.7.2021

[15] https://en.wikipedia.org/wiki/VR_positional_tracking Pronađeno 15.7.2021

[16] https://www.ultraleap.com/company/news/blog/how-hand-tracking-works/ Pronađeno 15.7.2021

[17] https://www.ultraleap.com/product/leap-motion-controller/#whatsincluded Pronađeno 15.7.2021

[18] https://developer.oculus.com/documentation/native/pc/dg-guardian-system/ Pronađeno 15.7.2021

[19] https://developer.leapmotion.com/unity Pronađeno 21.3.2021

[20] https://assetstore.unity.com/packages/2d/textures-materials/wood/hand-painted-seamless-wood-texture-vol-6-162145 Pronađeno 24.7.2021

[21] https://leapmotion.github.io/UnityModules/interaction-engine.html Pronađeno 26.7.2021

[22] https://leapmotion.github.io/UnityModules/core.html Pronađeno 26.7.2021.

[23] https://docs.unity3d.com/Manual/class-PhysicMaterial.html Pronađeno 5.8.2021.

[24] https://www.classvr.com/ Pronađeno 23.9.2021

[25] https://www.instavr.co/articles/general/why-making-vr-is-now-easier-than-ever-before Pronađeno 23.9.2021

[26] https://amt-lab.org/blog/2021/8/how-vr-and-ar-are-changing-the-world-of-immersive-theater Pronađeno 23.9.2021

[27] https://www.coolblue.be/en/advice/what-do-i-need-for-virtual-reality.html Pronađeno 23.9.2021

[28] https://en.wikipedia.org/wiki/Virtual_reality Pronađeno 23.9.2021

[29] 7 Things about Microsoft Hololens, Information Technology, Grand Valley State University https://www.gvsu.edu/cms4/asset/7E70FBB5-0BBC-EF4C-A56CBB9121AECA7F/7_things_about_microsoft_hololens.pdf Pronađeno 24.9.2021

[30] https://www.microsoft.com/en-us/hololens Pronađeno 24.9.2021

[31] https://engineering.case.edu/HoloAnatomy-honors Pronađeno 24.9.2021

[32] https://www.realwear.com/products/ Pronađeno 24.9.2021

[33] https://xrgo.io/en/x-reality-introduction/ Pronađeno 24.9.2021

[34] https://www.ptc.com/en/blogs/service/what-is-assisted-reality Pronađeno 24.9.2021

[35] https://www.intel.com/content/www/us/en/tech-tips-and-tricks/virtual-reality-vs-augmented-reality.html Pronađeno 24.9.2021

[36] https://ec.europa.eu/croatia/content/what-is-AR-what-VR-and-how-technology-helps-us-to-experience-reality_hr Pronađeno 24.9.2021

[37] https://www.marxentlabs.com/what-is-virtual-reality/ Pronađeno 24.9.2021

[38] https://free3d.com/ Pronađeno 23.6.2019

[39] https://www.renderhub.com/ Pronađeno 23.6.2019

[40] https://www.tutorialspoint.com/unity/unity_rigidbodies_and_physics.htm Pronađeno 26.9.2021

[41] https://docs.unity3d.com/ScriptReference/Rigidbody.html Pronađeno 26.9.2021

[42] https://www.tutorialspoint.com/unity/unity_understanding_collisions.htm Pronađeno 26.9.2021

[43] https://en.wikipedia.org/wiki/Jenga Pronađeno 28.9.2021

4. Popis slika