Izrada VR akcijske videoigre preživljavanja u programskom alatu Unity

Rosenthal, Karlo

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Karlo Rosenthal

IZRADA VR AKCIJSKE VIDEOIGRE PREŽIVLJAVANJA U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Karlo Rosenthal

Matični broj: 0016155693

Studij: Informacijski sustavi

IZRADA VR AKCIJSKE VIDEOIGRE PREŽIVLJAVANJA U PROGRAMSKOM ALATU UNITY

ZAVRŠNI RAD

Mentor :

Doc. dr. sc. Mladen Konecki

Varaždin, Rujan 2024.

Karlo Rosenthal

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad prikazuje razvoj igre namijenjene za uređaje virtualne stvarnosti (dalje: VR). Osnovni cilj ovog rada je prikazati postupak razvoja igre od početnih postavki do gotove igre. Kako bi se taj cilj ostvario, korišteni su besplatni, javno dostupni alati i aplikacije. Za sam razvoj igre korišteni su Unity i Visual Studio, dok su za kreiranje sadržaja između ostaloga korišteni Blender i Umjetna inteligencija. Poseban fokus postavljen je na integraciju VR aspekata u 3D igru, kako bi se postiglo ugodno i uživljeno iskustvo korisnika. Daljnja poglavlja opisuju korištene alate te njihovu namjenenu za vrijeme razvoja igre, zatim su objašnjeni dijelovi videoigre te njihova logika. Finalan proizvod je igra koja se može pokrenuti na uređaju Oculus Quest 2 u kojoj korisnik mora čim duže preživjeti valove sve jačih neprijatelja uz pomoć različitih oružja i njihovih nadogradnji.

Ključne riječi: VR, Unity, videoigre, razvoj videoigre, C#, 3D, umjetna inteligencija

Sadržaj

1. Uvod

Razvoj računalnih igara (dalje: gaming industrija) je grana IKT-a koja svake godine postaje sve više relevantna. Relevantnost među korisnicima i velika potražnja za novim i inovativnim idejama pretvara to u konkurentno i profitabilno područje.[1] 2019. godine globalan prihod gaming industrije iznosio je 265.21 milijardu američkih dolara, dok je ta brojka 2023. porasla na 406.2 milijarde američkih dolara. Isti izvor predviđa konstantan rast industrije te je 2029. predviđeno da će globalan prihod gaming industrije u iznosu od 666.69 milijardi američkih dolara.

VR je specifična grana IKT-a. Sposobnost uživljenja korisnika u virtualno generirani svijet ima primjene u svim granama djelovanja tehnologije. Ta osobnost je pogotovo poželjna u gaming industriji. VR tehnologija je trenutno u početcima razvoja. Do prije nekoliko godina razvoj VR igara nije imao adekvatnu programsku potporu te su ograničenja tadašnjih uređaja predstavljali još veći izazov pri razvoju igara. Trenutno popularne platforme za razvoj videoigara poput Unityja i Unreal Enginea dolaze s uključenim paketima za razvoj VR aplikacija. Postavljeni su novi standardi poput OpenXR-a koji omogućuje razvoj VR aplikacija za više različitih uređaja odjednom. Također, popularnost VR uređaja postaje sve veća, što potiče kompanije da ulažu više resursi u sve naprednije uređaje.

Tema ovog završnog rada je izrada VR igre u programskom okruženju Unity kroz programiranje u jeziku c#. Žanr igre je preživljavanje s ciljem ostvarenja čim boljeg rezultata. Kako bi igrač preživio mora koristiti oružja stečena kroz igru u borbi protiv sve jačih neprijatelja. Igra mora biti prilagođena VR uređajima te ne smije izazivati mučninu kod korisnika. Način igranja inspiriran je pretežito igrom Vampire survivors. Poseban značaj ove teme je da ne postoji puno igara slične tematike koje su prilagođene za VR uređaje.

Moja motivacija za izradu ovog završnog rada je pretežito fascinacija virtualnom stvarnošću. Od prvog susreta s VR tehnologijom, počela me zanimati izrada aplikacija za takav oblik uređaja. S pomoću vještina stečenih tijekom polaženja modula Umreženi sustavi i računalne igre, smatram da je izabrana tema, iako izazovna, izvediva. Izrada videoigara također je jedna od grana informacijskih i komunikacijskih tehnologija koja me jako zanima. Predviđanja o konstantom porastu vrijednosti i važnosti tržišta također je velika motivacija prilikom učenja novih vještina unutar takve grane.

2. Metode i tehnike rada

Tema izrade videoigre za VR uređaj bila je dogovorena s mentorom već na početku 5. semestra. Nakon toga započelo je istraživanje žanrova igre sukladnih mogućnosti izrade. Jedna od originalnih ideja bila je igra žanra tower defense u kojoj se neki objekt brani od valova neprijatelja. Nakon predstavljanja ideje mentoru, postignut je konsenzus o promjeni teme na VR akcijsku videoigru preživljavanja inspiriranu igrom Vampire survivors.[2] Vampire survivors je igra s ciljem preživljavanja određenog vremena s minimalističnim načinom igre i roguelite elementima. Istraživanje takvog tipa igre proizvelo je tri ključna elementa. Prvi element je postepeno pojačavanje protivnika. Svake minute izazov postaje sve veći, što znači da igra počinje lagano ali zaokupira korisnika konstantnim otežavanjem. Drugi element je konstantno pojačavanje igrača. Kako bi igrač imao šanse protiv sve jačih protivnika potrebno je kroz savladavanje izazova nagraditi ga pojačanjem. Posljednji ključni element je osjećaj raznovrsnosti. Potrebno je korisniku pružiti više načina igre i pustiti ga da za sebe otkrije koji način mu najviše odgovara. Osim ciljeva za način igre postavljeni su i ciljevi vezani uz prilagodbu korištenja VR uređaja.

Nakon postavljenih ciljeva, istraživanje je prešlo na potrebne alate i literaturu. Sve što je bilo potrebno za izradu projekta pronađeno je putem interneta. Prikupljeni materijali pobliže su objašnjeni u sljedećem poglavlju.

Prva inačica projekta napravljena je u 2D okruženju s dodacima za VR. Doduše, testiranjem igre postalo je evidentno da nativni 2D nije poželjan kao prikaz 3D svijeta. Korisniku bi od previše animacija i premalo dubine ubrzo postalo mučno. Stoga je prema tim saznanjima finalna verzija izrađena s pomoću 3D predložaka. Također je smanjen broj animacija i proširen pogled igrača kako bi se stupanj mučnine smanjio. Za vrijeme izrade rada verzioniranje se pokazalo kao vrlo važan proces, jer se u nekoliko slučajeva dogodila pogreška koja je mogla ugroziti integritet cijelog projekta te utrošiti neizmjeran broj sati rada. Projekt se konstantno razvijao, testirao te prema analizi testiranja mijenjao. Prema tome može se reći da je ovo bio iterativan razvoj projekta.

2.1. Literatura

Za vrijeme izrade ovog rada najviše su korištena dva izvora, [3] dokumentacija aplikacije Unity te [4] knjiga koja objašnjava kako prilagoditi Unity projekt za VR uređaj. S Unity Asset Sotrea preuzeti su model igrača [5], paket čestica eksplozija [6], izgled metka [7], paket čestica vatre [8]. Unutar rada osim prije spomenutih izvora, citirani su izvještaj o stanju gaming industrije [1], opis Steam stranice igre Vampire Survivors [2] te opis proizvoda OpenXR [9]. Također su korišteni i AI jezični modeli ChatGPT [10] i Copilot [11]. Od AI alata još je korišten i online alat za generiranje slika [12].

2.2. Korišteni alati

Ovo poglavlje služi za opis korištenih alata i online aplikacija.

2.2.1. Unity

Unity je jedna od najpopularnijih platformi za izradu videoigara u cijelom svijetu. Pruža širok spektar razvojnih alata te korisnicima omogućuje izradu 2D i 3D igara za mobilne uređaje, računala i konzole. Upravo ta svestranost je potrebna za izradu ovog rada. Kako bi se igra mogla pokrenuti na Meta Quest 2 uređaju ona mora biti formata ".apk", koji je namijenjen za Android uređaje. Unity također dolazi s paketom koji omogućuje podešavanje projekta za rad na VR uređaju.

Unity Asset Store je svojevrsna platforma u sklopu Unityja putem koje korisnici mogu prodavati sredstva za igru koje su sami generirali. Tamo je također moguće pronaći i besplatna sredstva. Koristeći Asset Store u svrhu izrade rada, smanjilo se vrijeme potrebno za izradu, dok je kvaliteta ostala na istoj razini.

2.2.2. Visual Studio

Logika objekata u Unityju programira se jezikom c#. Visual Studio je razvojno okruženje koje ima ugrađene pakete za provjeru i predviđanje koda pisanog u c#. Visual Studio također ima ugrađenu potporu za rad s Unityjom, što olakšava kreiranje i uređivanje koda te ga čini jednostavnim odabirom kao glavnog uređivača koda.

2.2.3. GitHub

GitHub je online alat koji se koristi za verzioniranje projekata. Na jednostavan način moguće je kreirati grane, što omogućuje eksperimentiranje bez posljedica za integritet projekta. Nakon zadovoljavajućeg modificiranja grane, moguće je pripojiti ju s glavnoj grani. Prilikom izrade ovog rada, glavna grana je služila kao sigurnosna kopija. Kako bi se svi ti procesi olakšali, korišten je i GitHub Desktop, aplikacija koja omogućuje pohranu grane na GitHub cloud u samo nekoliko klikova.

2.2.4. Blender

Iako Unity Asset Store nudi mnogo besplatnih sredstava, u nekim slučajevima nije moguće pronaći ono što je potrebno projektu bez plaćanja. U takvim situacijama aplikacija Blender nudi alternativu, ona služi za modeliranje i uređivanje 2D i 3D objekata. Osim samog modeliranja, u sklopu Blendera moguće je i animirati objekte te ih izvesti u obliku prilagođenom Unityju.

2.2.5. Mixamo

Kako bi se objekt unutar Blendera animirao, potrebno je dobro poznavanje kostura 3D objekta. U drugim riječima, potrebno je odrediti točke koje se pomiču te ih animirati tako da kretanje izgleda prirodno. Upravo za taj problem Adobe Mixamo nudi rješenje. Mixamo je besplatan online alat koji nudi već napravljene animacije 3D likova. Mixamo nudi i zbirku različitih modela, ali ono što ga čini stvarno posebnim i odličnim za ovaj projekt je mogućnost učitavanje i animiranje vlastitog modela. Potrebno je samo odabrati i preuzeti željene animacije.

2.2.6. Umjetna inteligencija

Umjetna inteligencija (dalje: AI) postala je neizostavan alat prilikom rada na području informacijskih i komunikacijskih tehnologija. Kao i kod svakog drugog alata moguće je zloupotrijebiti sposobnosti AI. Za potrebe izrade ovog rada, AI je korišten kao alat za pojašnjenje nepoznatih pojmova. U tu svrhu korišteni su ChatGPT [10] i Microsoft Kopilot[11]. Još jedan oblik AI koji je bio korišten je online alat za generiranje slika[12].

2.2.7. Alati za VR testiranje

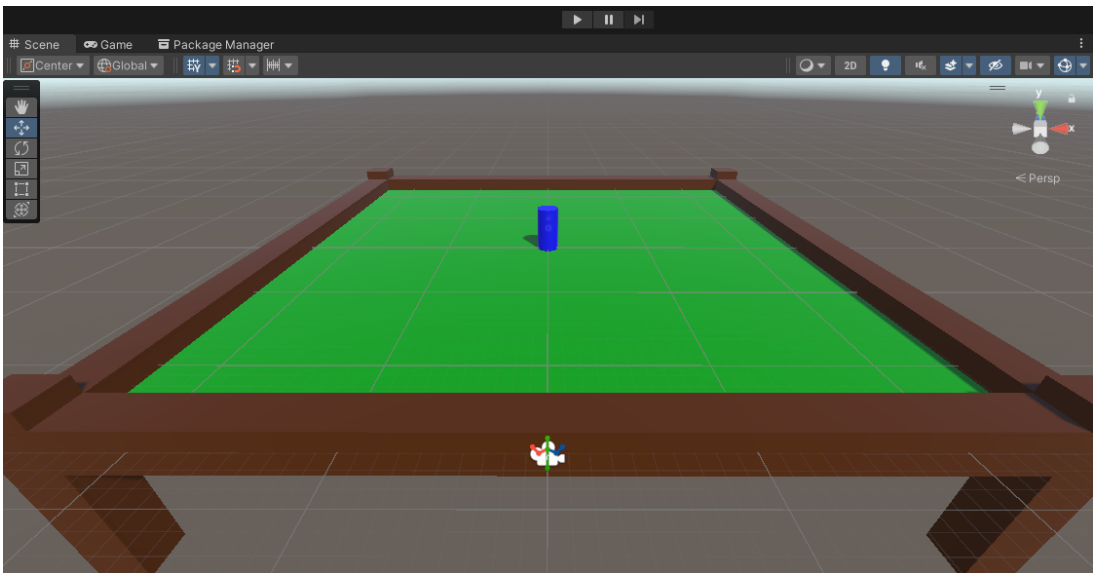
Kako bi se za vrijeme izrade projekta napredak mogao na brz način pratiti s pomoću VR uređaja, korištene su aplikacije Meta Quest Link i Steam VR. S pomoću Meta Quest Link aplikacije moguće je uspostaviti vezu između računala i VR uređaja s pomoću kabla ili bežično. Nakon uspostavljanja veze, putem quest link-a moguće je pristupiti radnoj površini računala kroz sučelje VR uređaja te pokrenuti testiranje igre. Tada se pokreće aplikacija Steam VR, koja emulira igru kao da je pokrenuta direktno s VR uređaja.

3. Razrada teme

3.1. Postavljanje projekta

Prilikom pokretanja Unityja potrebno je napraviti novi projekt te odabrati početne postavke. Unity nudi opciju izrade projekta u VR predlošku, ali uz taj predložak uključeno je puno nepotrebnih paketa i sredstava. Prema tome projekt je kreiran s pomoću 3D predloška.

Prilikom kreiranja 3D predloška, stvorena je kamera i prazan 3D prostor. Kako bi započeli rad na igri potrebni su na osnovni objekti. Za početak je stvorena podloga "Plane" po kojoj će se igrač kretati, zatim su dodani zidovi kako igrač ne bi mogao otići izvan predviđenih granica igre. Zidovima smo dodali komponentu sudarač, s pomoću te komponente Unity otkriva dodir između igrača i zida te igraču ne dopušta daljnje kretanje u tom smjeru. Radi dodatne estetike prostora zidovima su dodane noge te sve zajedno izgleda kao stol. Zatim je dodan novi 3D objekt u obliku cilindra koji je na početku izrade predstavljao igrača. Igraču su dodane komponente sudarač i Rigidbody 3D, koja služi za simulaciju fizike. Objekti koji su tek stvoreni unutar Unityja dolaze bezbojni, kako bi se to promijenilo stvoreni su jednobojni materijali za razlikovanje i uljepšanje izgleda.

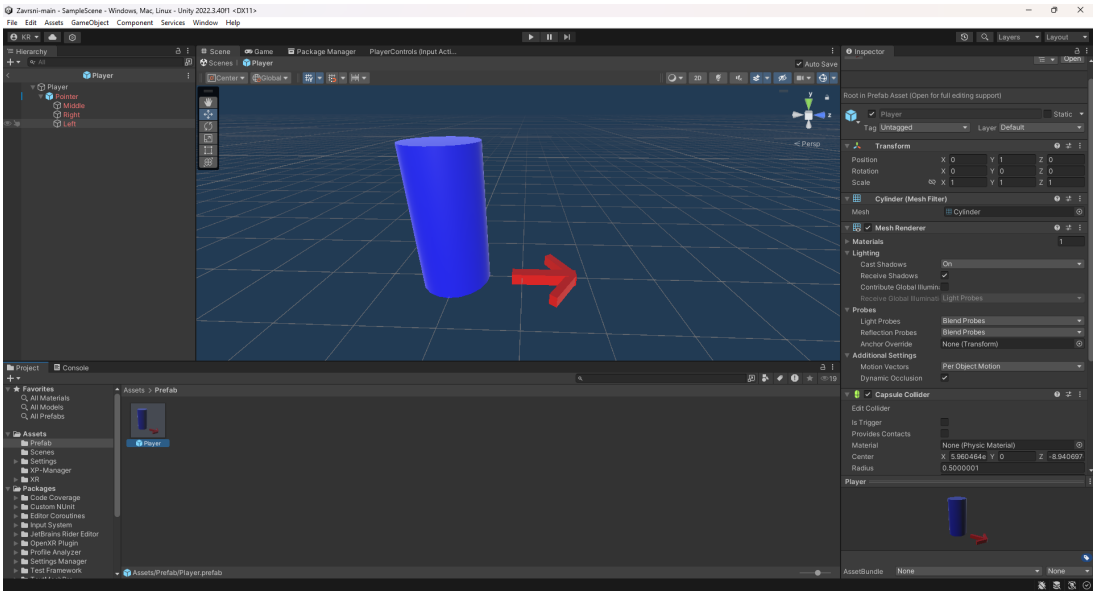


Slika 1: Prikaz osnovnih elemenata Unity Editoru

Sljedeći korak bio je pretvaranje igrača u predložak. Predlošci se u Unityju koriste kako bi spremili neki objekt za kasnije stvaranje [3]. Iako se igrač stvara odmah na početku, pogled predloška omoguće lakše uređivanje objekta. Kako bi se objekt pretvorio u predložak potrebno ga je povući na "Assets" dio Unity uređivača [3]. Već na samom početku razvoja u prikazu sredstava počinjalo je izgledati neuredno. Zato su i stvorene datoteke koje kategoriziraju sredstva. Kao posljednji korak postavljanja projekta, igraču je unutar pogleda predloška dodana strelica koja pokazuje njegovu orijentaciju. Ona će kasnije služiti kako bi korisnik mogao lakše ciljati.

Kada se postavi Unity projekt već postoji definirani način unosa. Zbog toga, nije potrebno svaki puta definirati da se igrač kreče s pomoću „W“, „S“, „A“, „D“. Tako definirani su i

unosi za kontrolere i njihove gljivice. Za potrebe ovog rada, trebalo je razlikovati lijevu i desnu gljivicu, pošto se lijeva koristi za pomicanje, a desna za okretanje. Za to se koristio Unity Input Actions pomoću kojeg su postavljeni unosi koji se očitavaju prilikom određenih događaja. Radi lakšeg testiranja postavljene su i kontrole za računalo, tako da se mogu koristiti strjelice te „W“, „S“, „A“, „D“, kao i gljivice kontrolera prilikom testiranja s VR uređajem.



Slika 2: Prikaz početnih postavki igrača u pogledu predloška

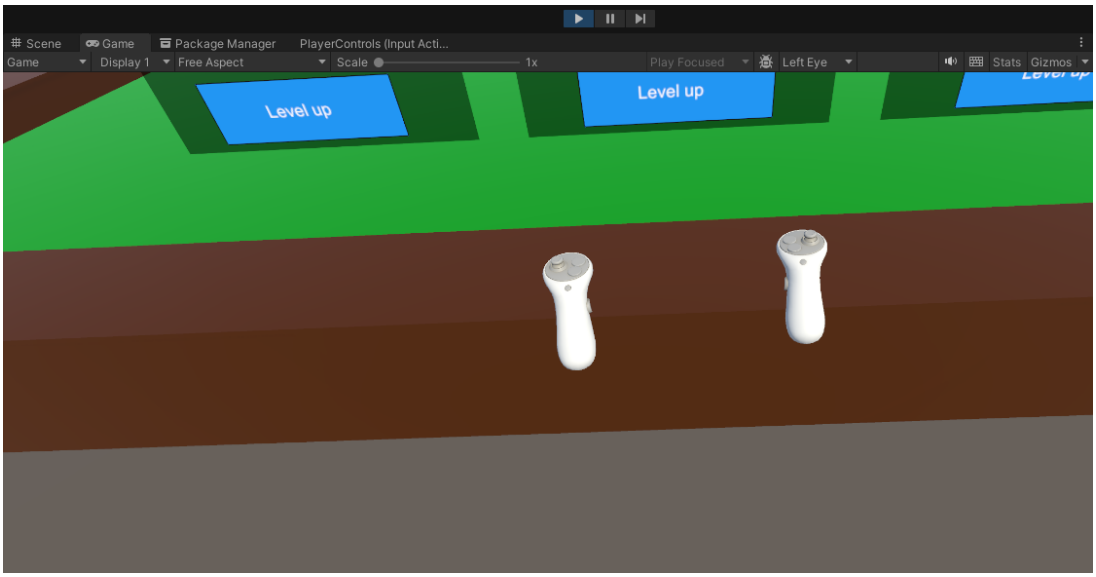
3.2. Virtualna stvarnost

Pošto je razvoj igre započet u 3D predlošku, potrebno je prilagoditi projekt VR uređajima. Za to je korišten paket XR Plugin Management, koji se dodaje s pomoću Package Managera ugrađenog u Unity. Nakon instalacije paketa u postavkama projekta potrebno je unutar XR Plugin Managementa omogućiti OpenXR te kvačicom označiti inicijalizirati XR prilikom pokretanja. OpenXR je besplatan standard koji pruža API-e za razvoj XR aplikacija za razne XR uređaje [9], uključujući i Meta Quest 2. Sljedeći korak je validacija projekta, koja automatski povezuje novo dodane komponente u pozadinske operacije koje Unity izvršava.

Početna verzija 3D projekta bila je zamišljena tako da kamera prati igrača, što bi omogućilo veću igraču površinu. Ali, ne mogućnost okretanja glave se u VR okruženju osjeća neprirodno, što narušava dojam igre. Druga mogućnost po pitanju kamere bila je pogled iz prvog lica. Kretanje u prvom licu jedna je od kontroverznijih tema prilikom izrade VR igara. Novi i neiskusni korisnici mogu se već nakon nekoliko koraka osjećati dezorijentirano, dok iskusniji korisnici također mogu dobiti takav osjećaj nakon nekog vremena [4]. Neke igre upravo zbog toga izbjegavaju kretanje koje imitira hodanje te kao alternativu korisniku omogućuju teleportiranje po malim udaljenostima [4]. Obje inačice kretanja u prvom licu nisu prigodne za igru u kojoj je cilj što duže i konstantno bježati od neprijatelja. U konačnici odlučeno je da će korisnik moći okretati glavu te da će imati pogled iz ptičje perspektive na igru. Cilj takvog pogleda je korisniku dati dojam taktičara koji ima pregled cijelog bojišta, zbog čega lakše donosi odluke.

U sklopu dovedenih odluka trebalo je isto primijeniti na igru. Da bi to bilo moguće bilo je potrebno dodati još jedan paket zvan „XR Interaction Toolkit“. Taj paket sadrži već gotova rješenja prvenstveno za kameru, ali i za ostale VR aspekte. Za početak je trebalo kameru pretvoriti u XR Rig, tako prilikom pokretanja igre u VR okruženju, kamera prati pogled korisnika te se okreće s VR uređajem. XR Rig je na početku prazan te mu je potrebno dodijeliti glavnu kameru, ulaz za poziciju, ulaz za rotaciju te čitač stanja uređaja.

Za dodani ugođaj korisnika u sklopu VR pogleda dodani su i modeli VR kontrolera u igru. Ti modeli prate stvaran položaj kontrolera, kao i njihovu rotaciju. Još jedna zanimljivost kod implementiranih modela jest da prate gumbe i gljivice, što je vidljivo za vrijeme igre. Već napravljeni modeli nalaze se u sklopu paketa „XR Core Utilities“. Postojeći predlošci dodani su XR Rigu kako bi se pomicali s kretanjem uređaja. Za razliku od kamere, predlošci kontrolera dolaze povezani s postavka potrebnima za rad, ali i s dijelovima koji nisu potrebni za ovu igru. Dijelovi koji su uklonjeni bili su vezani uz blisku i udaljenu interakciju s objektima. Pošto korisnik koristi kontrolere za kretanje i rotiranje, nije bilo potrebno koristiti takve komponente. Iznimka je zraka za interakciju na desnom kontroleru, ona se unutar igre koristi za interakciju s korisničkim sučeljima. Doduše, zraka koja je bila dio predloška, bila je povezana s gljivicom kontrolera te bi se pokazivala za vrijeme igre, stoga je ta zraka bila napravljena ponovno. Nova zraka je u hijerarhiji dijete desnog kontrolera, kako bi pratila njegovu poziciju te se pokazuje samo kada je potrebno napraviti odabir na sučelju. Pošto je napravljena nova zraka, ona dolazi bez prijašnjih postavki, jedina postavka koju je trebalo promijeniti je dodati skriptu koja provjerava stanje gumba za interakciju. Ostale postave nije bilo potrebno mijenjati.



Slika 3: Prikaz modela kontrolera iz VR pogleda

3.3. Glavne komponente

Glavne komponente odnose se na vidljive elemente u igri na kojima ili s pomoću kojih se vrši interakcija.

3.3.1. Igrač

Igrač je centralna komponenta svake videoigre, pa tako i ove. Igrač ima već prije objašnjene komponente i skripte za kretanje, rukovođenje događaja i skriptu za nivoe klasa. Kako bi se igrač mogao micati, u ovom slučaju potrebna mu je komponenta „Player Input“. Unutar te komponente dodajemo novo napravljeni sistem unosa. Još jedna komponenta koja je potrebna za kretanje igrača je Animator. S pomoću animatora, objektu se dodaju definirane animacije te okidači za promjene animacija tijekom igre.

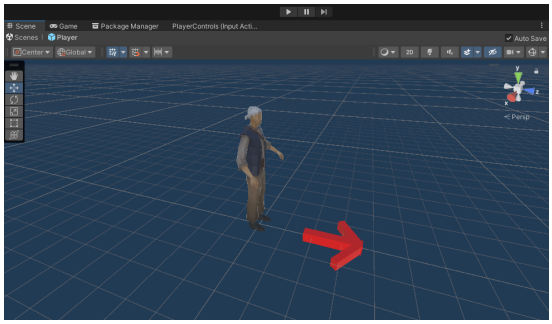
Prva napravljena c# skripta je „Movement“. U toj skripti definirane su brzina kretanja, čitač unosa za micanje te varijabla anim tipa Animator za promjenu animacija. Prilikom pokretanja skripte, anim dohvaća komponentu Animator te tako omogućuje komunikaciju i mijenjanje stanja. Čitač vrijednosti poziva se svaki put kada je zaprimljen unos zadan u komponenti „Player Input“, tada se bilježi vrijednost za koliko i u kojem smjeru se pomiče igrač. Čitač vrijednosti također bilježi i kada je vrijednost pomicanja postavljena na 0, odnosno kada je tipka ili gljivica puštena. Na početku generiranja svakog okvira igre, funkcija Update() poziva funkciju movePlayer(). Funkcija movePlayer() definira da će se igrač pomaknuti za onoliko koliko je posljednje pohranjeno. Unutar te funkcije, također je definirano da u slučaju da se igrač ne pomiče anim postavlja okidač hodanja u animatoru na netočno, prilikom čega igrač ima animaciju stajanja na mjestu. U suprotnom, anim postavlja okidač na točno te je na igraču prikazana animacija hodanja.

Skripta „Player_rotation“ radi na sličnom principu kao i „Movement“. Definirani su čitači vrijednosti, ali velika razlika je u tome što ova skripta radi drugačije ovisno o bool varijabli isPc. Ta se varijabla koristi za svrhe testiranja, kako bi se izbjegle komplikacije prilikom testiranja kako na računalu, tako i na VR uređaju. Prema tome koristi se jedan čitač za unos putem računala i drugi za unos putem kontrolera. Za vrijeme dok je isPc točan, funkcija Update() stvara nevidljivu zraku iz kamere u smjeru miša koji predstavlja čitač za unos računala. U slučaju da ta zraka pogodi točku unutar igre funkcija RotatePlayer() izračunava vektor prema kojem se igrač mora okrenuti kako bi gledao u željenom smjeru. S pomoću funkcije Quaternion.Slerp() taj okret se desi u periodu od 0.15 sekundi što omogućuje prirodnije rotiranje. U slučaju da se samo definira smjer u kojem igrač treba gledati, igrač bi se okrenuo odmah sljedeći okvir te bi se činilo kao da se teleportira. U drugom slučaju, odnosno kada je isPc netočan, više nije potrebno stvarati zrake, već se samo s pomoću čitača prate vrijednosti gljivice desnog kontrolera te se odmah pokreće funkcija RotatePlayer(). U tom slučaju RotatePlayer() izračunava ciljanu poziciju te na isti način okreče igrača prema tom smjeru.

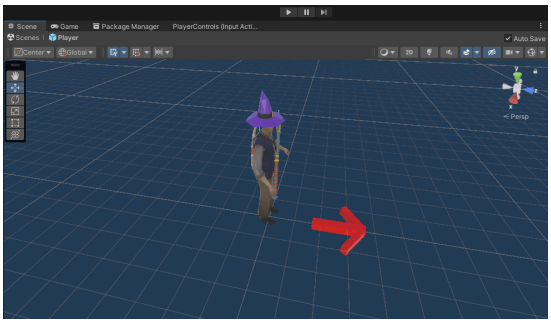
„PlayerStats“ je skripta koja služi kao rukovoditelj događaja koji se dešavaju igraču. Ona sadrži varijable povezane s jačinom igrača poput broja života, maksimalnog borja života te modifikatore vezane uz oružja i brzine kretanja. Još jedna posebnost ove skripte je pohrana

i pozivanje liste oružja. Funkcijom Start() životi se postavljaju na maksimalne živote, tako kroz igru je moguće mijenjati koliko života ima igrač i paziti da ne prelazi granicu maksimalnih života. Funkcija koja je zadužena za takvo dinamično praćenje je TakeDamage(int taken). Ova funkcija sadrži argument „taken“ koji označuje koliko će štete primiti igrač kada ga protivnik napadne. U slučaju da igračevi životi padnu na 0 ili manje, započinje kraj igre te se s pomoću funkcije Destroy(gameObject) uništava objekt igrača te većina sustava prestaje s radom. PlayerStats sadrži još nekoliko funkcija koje će biti objašnjene s povezanim događajima.

Osim skripti, promijenjen je i model igrača. S pomoću Unity Asset Sotrea pronađen je lik avanturista koji po estetici naliči na običnog građanina [5]. Tada je bilo potrebno zamijeniti stari geometrijski oblik za novi model. Nakon zamjene je model animiran koristeći Mixamo te su dodane animacije za mirovanje i hodanje. Po pitanju izgleda igrača ovisno o klasama koje je korisnik odabrao kroz igru, dodaju se prikladna avanturistička oprema. Tako igrač dobiva sjekiru kada je izabran ratnik, luk kada je izabran lovac i šešir kada je izabran čarobnjak. Na taj način se također dobivaju i prikladna oružja.



(a) Model Igrača

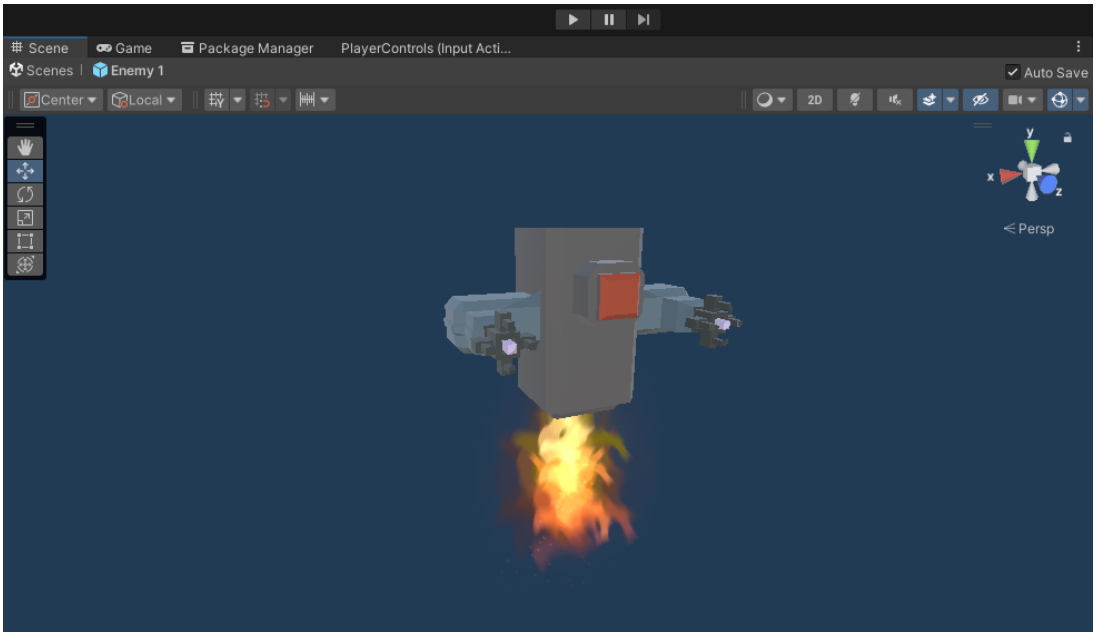


(b) Igrač s oružjima

Slika 4: Prikaz razlika modela igrača

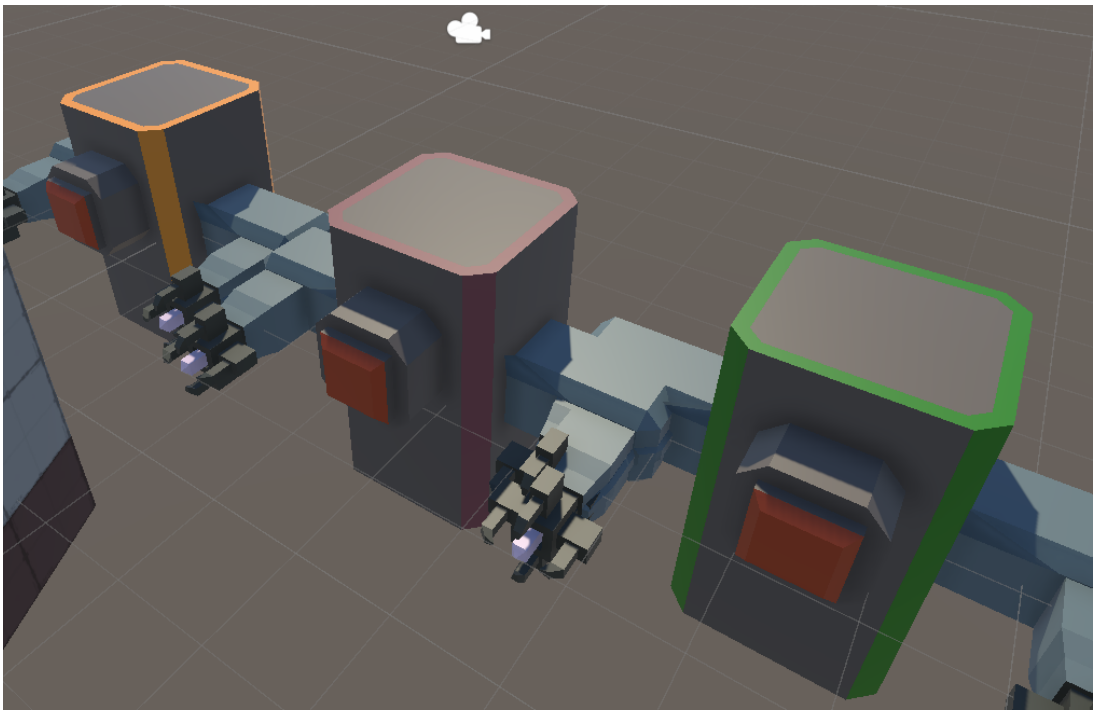
3.3.2. Neprijatelj

U igrama preživljavanja, potreban je izvor opasnosti za igrača u ovom slučaju to je sve snažniji neprijatelj. U osnovi se igrač i neprijatelj ne razlikuju previše, pa tako neprijatelj sadrži varijable za broj života, broj maksimalnih životnih bodova, štetu koju nanosi igraču, brzinu napada, brzinu kretanja te xp vrijednost. Neprijatelj je također u relaciji s XP Managerom kako bi proslijedio svoju xp vrijednost pozadinskom procesu te se čitajući vrijednost proteklog vremena od Timera postepeno pojačava. Po pitanju izgleda neprijatelja, potrebno je spomenuti sustav čestica preuzet s Unity Asset Storea [8]. Sustav čestica u obliku vatre neprijatelju daje smislen dojam letenja.



Slika 5: Prikaz standardnog izgleda neprijatelja unutar Unity-a

Neprijatelji se stvaraju s pomoću pozadinskog procesa stvaranja neprijatelja te kada su stvoreni pokreće se skripta „Enemy“. Jednom kada je skripta pokrenuta, s pomoću funkcije Start(), dohvaća se timer te započinje korutina GenerateStats(). Ta korutina služi za izračun snage protivnika. Prvi faktor koji izračunava je maksimalan broj životnih bodova, generira se nasumičan broj od 1 do 15 te se množi s kvocijentom proteklog vremena i 60. Proteklo vrijeme dobiva se od objekta timer te je u trenutku stvaranja neprijatelja prošlo više od nula vremena, pa se neprijatelji neće stvarati s 0 života te će bez obzira na dobiveni nasumični broj biti dovoljno jednostavni za pobijediti. Zatim se dobiva maksimalan broj životnih bodova tako da se dobiveni proizvod s pomoću funkcije Mathf.CeilToInt() zaokruži na prvi veći cijeli broj. Xp vrijednost također uzima proizvod korišten za izračun maksimalnih životnih bodova te ga dijeli s 3 i zaokružuje na prvi veći cijeli broj. Proteklo vrijeme također je potrebno kako bi se izračunala šteta koju neprijatelj nanosi. Izračun je postavljen tako da se šteta svake minute poveća za 5. Posljednja funkcionalnost korutine GenerateStats() je odabir izgleda protivnika. Kako bi korisnik mogao razlikovati koliko života ima neprijatelj postoje četiri granice određene bojom obruba neprijatelja. U slučaju da neprijatelj ima 7 ili manje života prikazuje se standardan izgled, s 15 ili manje izgled je zelen, 30 ili manje narančast, dok za sve više od 30 ima crveni obrub.



Slika 6: Prikaz neprijatelja u bojama unutar Blender-a

Prilikom pokretanja skripte neprijatelj dohvaća i Transform komponentu igrača. S pomoću dohvaćene komponente u funkciji Update(), neprijatelj traži igračevu poziciju te se okreće i kreće prema njemu, no za to moraju biti ispunjena dva uvjeta. Prvi uvjet je da igrač mora postojati, što ne vrijedi jedino kada igrač izgubi, dok je drugi uvjet da je udaljenost između igrača i neprijatelja veća od 1. U slučaju da je udaljenost veća, protivnik će se pomoću Quaternion.Slerp() funkcije postepeno okretati prema igraču te će se pomoću funkcije Vector3.MoveTowards() kretati prema njemu. Pošto neprijatelj leti, objekt igrača nalazi se na drugačijoj y vrijednosti od neprijatelja, dolazilo bi do greške u kojoj se neprijatelj počeo kretati prema podu. Radi tog detalja, neprijatelju je ograničeno kretanje na y osi samo na vrijednost na kojoj se nalazi od trenutka kada je stvoren. U slučaju da je neprijatelj dostigao igrača te mu se nalazi na udaljenosti manjoj od 1, neprijatelj započinje svoj napad. Prvo se provjerava ako je prošlo dovoljno vremena od zadnjeg napada. Kada neprijatelj ne bi bio ograničen na taj način, napao bi igrača svaki okvir te bi ga ubrzo porazio. Ako je zadovoljen taj uvjet, neprijatelj započinje koriutinu Attack() prilikom koje dohvaća igračevu funkciju TakeDamage(int taken) te kao argument prosljeđuje štetu koju nanosi. U suprotnom slučaju, kada igrač pogodi neprijatelja, pokreće se neprijateljeva slična funkcija TakeDamage(int taken). Velika razlika je da svaki put kada neprijatelj primi štetu je u tome da pomoću korutine GetBody(), prema istoj logici kao i kod stvaranja, neprijatelj mijenja izgled. U slučaju da prilikom primanja štete neprijatelj padne na 0 ili manje života, neprijatelj dohvaća funkciju XP Managera zvanu AddExperience() te prosljeđuje svoju xp vrijednost.

3.4. Pozadinski sustavi

Pozadinski sustavi odnose se na sve što se dešava u pozadini igre te korisnik nema izravnu kontrolu nad njima.

3.4.1. Stvaranje neprijatelja

Stvaranje neprijatelja je proces koji se konstantno odvija za vrijeme igre te se izvodi pomoću skripte „Spawner“. U Unity sceni nalazi se na praznom objektu kojem pozicija nije važna. Skripta sadrži varijablu tipa Enemy koja određuje koji će se protivnik stvoriti, pošto u ovoj igri postoji samo jedan neprijatelj dodijeljena joj je vrijednost neprijateljeva predloška. Pošto Spawner ne vrši direktnu interakciju s igračem, nije mu potrebna varijabla tipa PlayerStats, nego mu je potrebna samo igračeva pozicija.

Iako se neprijatelji konstantno stvaraju za vrijeme igre, Spawner je onemogućen za vrijeme odabira klase. Pošto postoje periodi u kojima se ova komponenta omogućuje i onemogućuje, korištenje funkcije Start(), nije moguće. U takvom slučaju, nakon prvog onemogućenja objekt gubi referencu dobivenu s funkcijom Start() te kada se ponovno omogući, ugrožava integritet cijele igre. Prema tome, je u ovakvim slučajevima korištena funkcija OnEnable(). Tako skripta svaki put kada se objekt u igri omogući, napravi novu referencu s potrebnim objektom.

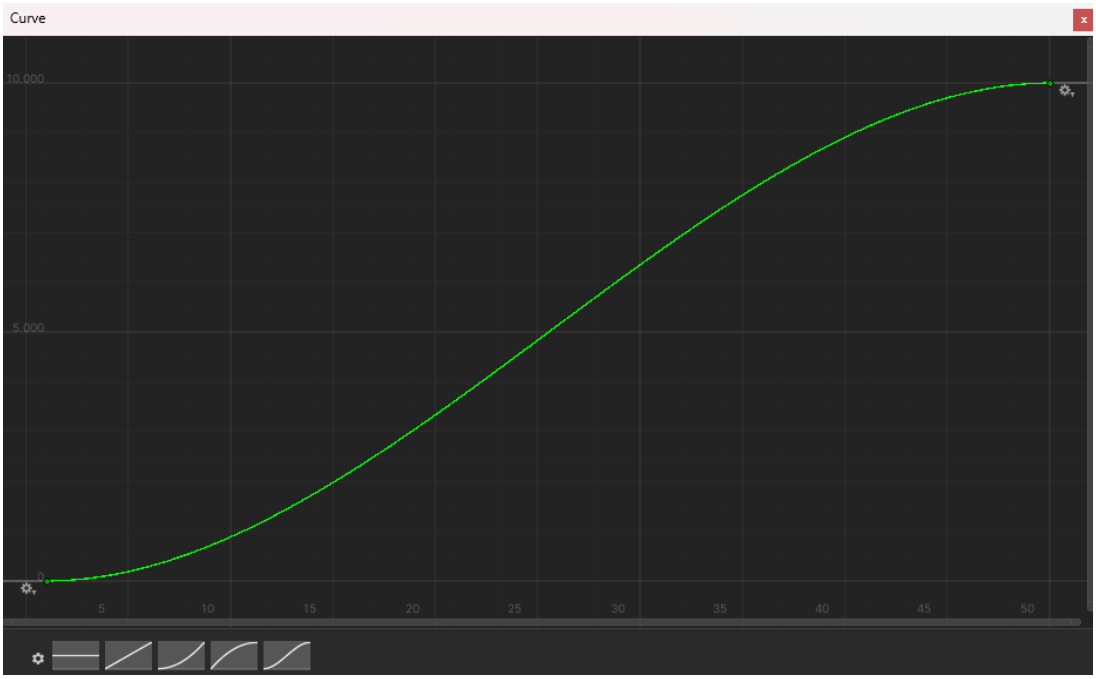
Sporna referenca u skripti Sapwner je traženje objekta s oznakom „Player“ te bilježenje komponente „Transform“ pronađenog objekta. Prilikom omogućenja se pokreće i korutina SpawnEnemy(float interval). Pomoću te korutine, moguće je napraviti interval između stvaranja neprijatelja. Za početak se odabire koji protivnik će biti stvoren te se pokreće još jedna korutina RandomPosition(). RandomPosition() bilježi mjesto na kojem se igrač trenutno nalazi te do tri puta nasumično generira poziciju stvaranja protivnika. Razlog tome je izbjegavanje stvaranja protivnika direktno na igraču. Takvo stvaranje protivnika može izazvati frustraciju te otežati igru. Pošto su programu dana tri pokušaja da se protivnik stvori na distanci, šanse su male, ali je i dalje moguće. Jednom kada je odabrana pozicija, pomoću funkcije Instantiate(), na tom mjestu se stvara neprijatelj te igra čeka toliko dugo koliko je zadan interval prije rekurzivnog pozivanja korutine SpawnEnemy().

3.4.2. Xp Manager

Menadžer iskustva, odnosno XP Manager služi za kalkulaciju potrebnog i zbrajanje prikupljenog iskustva (dalje: xp) te je u ovom projektu to postignuto na unikatan način. Poželjno je da zahtijevani xp za svaki sljedeći level igrača raste donekle inkrementalno. Kada bi takav rast bio stvarno inkrementalan, ubrzo bi došlo do neizmjerno velikih zahtjeva, što može izazvati dosadu kod igrača i takozvani „grind“. Kako bi se postigao pseudoinkrementalni rast korištena je animacijska krivulja. Osim varijabli potrebnih za izračune, korišteni su i objekti za manipulaciju korisničkih sučelja te objekt tipa PlayerStats za komunikaciju s igračem.

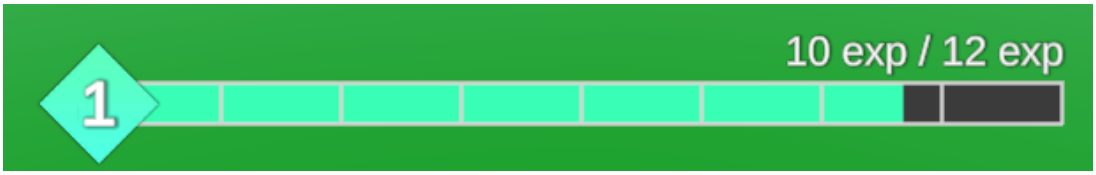
Kako bi se mogao računati potreban xp, potrebno je prvenstveno postaviti animacijsku krivulju. Za vrijeme postavljanja krivulje, moguće je odrediti proizvoljan broj točaka koje krivu-

lja prati, ali za ovaj rad potrebne su samo dvije. Krivulja je prikazana u dvodimenzionalnom prostoru u kojem je apscisa nazvana vrijeme, a ordinata vrijednost. Za potrebe XP Managera, vrijeme predstavlja level, dok vrijednost predstavlja potreban xp. Prva točka postavljena je na koordinatu (1,0), dok je maksimalan level postavljen na (50, 10000), odnosno da bi igrač došao do levela 50 mora skupiti 10000 xpa. Iako se kod prikaza brojka trenutnog xpa postavlja na 0 nakon svakog novog levela u pozadini se sav prikupljen xp zbraja. Što znači da igrač mora sakupiti ukupno 10000 xpa kako bi bio level 50.



Slika 7: Prikaz animacijse krivulje

Kada igrač pobjedi neprijatelja, neprijatelj dohvaća funkciju AddExperience(int amount) u koju prosljeđuje svoj xpWorth. XP Manager dodaje zaprimljeni xp ukupnom zbroju te pokreće dvije funkcije. Prva funkcija je UpdateInterface(), ta se funkcija poziva kada se desi bilo kakva promjena vezana uz xp. Ona direktno djeluje na xp traku, element korisničkog sučelja koji prikazuju stanje xpa korisniku. Postavljaju se brojke u formatu „ x / y“, gdje x predstavlja prikupljeni xp u odnosu na trenutni level, dok y predstavlja xp potreban za novi level. Ispod se nalazi traka koja vizualno prikazuje koliko je igrač blizu novog levela te broj trenutnog levela.



Slika 8: Sučelje prikaza Xpa

Druga funkcija koju pokreće AddExperience() je CheckForLevelUp(). U slučaju da je skupljeno dovoljno xpa za novi level, trenutni level se inkrementira te se šalje igraču. Zatim se pokreće još jedna funkcija UpdateLevel() koja određuje koliko je xpa potrebno za sljedeći level te koliko se oduzima od trenutnog xpa kako bi vizualno krenuli od nule. Za čitanje vrijed-

nosti krivulje koristi se funkcija Evaluate() koja kao argument prima vrijeme te vraća vrijednost krivulje. Nakon obavljenih operacija, pokreće se funkcija UpdateInterface().

3.4.3. Odabir klasa

Odabir klasa je proces koji je povezan sa sučeljem nagrada. Svaki put kada igrač postigne novi level, prikazuje se sučelje s tri ponuđene opcije. Ponuđene klase uključuju klasu ratnik, lovac i čarobnjak. Svaka klasa donosi kozmetičku nadogradnju igraču, pripadajuće oružje i nadogradnju na određene modifikatore. Svi procesi događaju se unutar skripte „Class“, koja se prilikom početka rada referencira na skripte PlayerStats i Movement.

Prilikom odabira klase ratnik, na modelu igrača pojavljuje se sjekira, dodaje se oružje „Sweeper“ te se uvećavaju hpMod i dmgMod. Modifikator hpMod određuje za koliko će se uvećati igračevi maksimalni životni bodovi. Zbog direktnog utjecaja na igrača, prilikom odabira ove klase, pokreće se funkcija skripte PlayerStats, WarriorChanges(). Ta funkcija određuje za koliko će se odjednom uvećati maksimalni životni bodovi igrača te trenutan broj života, na taj način u slučaju da igrač ima maksimalan broj života, obje varijable se uvećavaju bez straha od prijelaza granice maksimuma. Drugi uvećani modifikator je dmgMod koji utječe na količinu štete koju nanose oružja. Prilikom stvaranja oružja, osnovna šteta množi se s modifikatorom te se naknadno zaokružuje na prvi veći cijeli broj prilikom pogađanja neprijatelja.

Klasa lovac igraču dodaje luk na leđa, u listu dodaje oružje „Shooter“ i povećava modifikatore msMod te penMod. Radi ove klase je skripta povezana sa skriptom Movement. MsMod direktno utječe na brzinu kretanja igrača te ju uvećava svaki put kada se odabere klasa lovac. Dok modifikator penMod utječe ne penetraciju oružja, odnosno na broj protivnika koje oružje može udariti prije nego što se razbije.

Posljednja klasa čarobnjak, igraču na glavu stavlja magičan šešir, dodjeljuje mu oružje „Finder“ te pojačava cdMod i psMod. CdMod utječe na to koliko brzo igrač koristi ItemList. PlayerStats ima određen interval pozivanja funkcije Update() za svaki Item u listi. Povećanjem cdMod-a interval potreban za pozivanje funkcije se smanjuje. PsMod utječe na oružja te im uvećava brzinu kretanja.

(a) Ikona ratnika

(b) Ikona lovca

(c) Ikona čarobnjaka

Slika 9: Prikaz ikona klasa generianih s [12]

3.4.4. Najbolji rezultat

Prilikom završetka igre pokreče se sistem bilježenja najboljeg rezultata. Zabilježavanje rezultata omogućeno je s pomoću skripte "SaveSystem" u kojoj je kreira statična klasa SaveSystem, kako bi se izbjeglo stvaranje više inačica te klase. SaveSystem sadrži funkcije SaveScore() i LoadData(). Obje funkcije barataju posebnim oblikom podatka nazvanim "ScoreData". To je klasa koja sadrži javni cijeli broj minute i sekunde te ima funkciju ScoreData(Timer timer) koja kao argument prima objekt tipa Timer iz kojeg iščitava i pohranjuje timer.minute i timer.sekunde. SaveScore() također prima objekt oblika Timer da pohrani primljene podatke. Koristi FileStream kako bi kreirala binarnu datoteku "highscore.txt" na putanji aplikacije te s pomoću binarnog formatera serijalizira zaprimljene podatke. Funkcija LoadData() za početak ispituje ako datoteka "highscore.txt" postoji na traženom mjestu te u slučaju pronalaženja datoteke deserijalizira podatke te ih vraća u obliku ScoreData.

U okruženju skripte "NewGame" nalazi se kod za provjeru i zapis najboljeg rezultata. Kada skripta postane omogućena za početak pronalazi objekt tipa Timer te ga lokalno pohranjuje. Pošto se NewGame pokreće samo na kraju igre, uzima se vrijeme Timera pošto se više neće pokretati. Zatim se pokreće funkcija LoadData() kako bi se otkrilo postoji li datoteka "highscore.txt". U slučaju da ne postoji može se zaključiti da je to bila prva igra pa se trenutno vrijeme uzima kao i najbolje te se s pomoću funkcije SaveScore() pohranjuje u novo kreiranu datoteku "highscore.txt". U suprotnom slučaju, kada je pronađena datoteka, ponovno se pokreće LoadData() te se zapisana vrijednost lokalno pohranjuje i uspoređuje s trenutnom vremenskom vrijednosti. Ako je trenutno vrijeme bolje, prikazat će se na korisničkom sučelju te će se zapisati u datoteku "highscore.txt". U suprotnom, na sučelju se prikazuje zapisano vrijeme.

3.5. Sustav oružja

Logika za svako oružje nalazi se u skripti „Item“, koja također stvara istoimenu apstraktnu klasu. Svako oružje je nasljednik klase Item zbog načina pohrane i pozivanja funkcija. Funkcije koje oružja nasljeđuju su GiveName(), u svrhu razlikovanja oružja prilikom pozivanja te funkciju Update(). Update() je funkcija tipa public virtual void, pošto će svako oružje na drugačiji način modificirati istu funkciju, ona također prima arugmente PlayerStats player i int stacks. Naime, skripta „PlayerStats“ pohranjuje sva oružja koje je igrač osvojio za vrijeme igre u listu tipa ItemList te prilikom poziva svakog Itema prosljeđuje sebe kao argument. ItemList je klasa koja sadrži objekt tipa Item, naziv oružja i cijeli broj „stacks“, s pomoću kojeg se određuje koliko je neko oružje jako. Kada oružje dosegne 5 stack-ova ono dobiva dodatne moći. Klasa ItemList sadrži i funkciju koja omogućuje jednostavno dodavanje novih Itema u listu.

3.5.1. Healing item

Iako je klasificirano kao oružje, „HealingItem“, umjesto da čini štetu neprijatelju, zacjeljuje igrača. Funkcija Update() uređena je na način da svaki put kada se pokrene, igračev se

život uveća za 3 + 2 za svaki proslijeđeni stack. U svrhu zanimljivijeg načina igre te manjeg opterećenja korisnika u vezi odabira klasa, prilikom poziva svakog Itema, pod argument stacks prosljeđuje se igračev level. Također, kako bi se izbjeglo zacjeljivanje iznad maksimalnih životnih bodova, ako bi došlo do toga igračevi životi postavit će se na maksimalne životne bodove.

3.5.2. Sweeper

Prvo oružje namijenjeno za pobjeđivanje neprijatelja je „Sweeper“, koji pripada klasi ratnik. Sweeper stvara objekt sjekire ispred igrača na nekoliko stupnjeva lijevo od smjera u kojem gleda prilikom pozivanja Itema. Pošto se skripta Item niti objekt sjekira ne nalaze unutar hijerarhije igre, nije bilo moguće putem Unity inspektora Sweeperu dodijeliti sjekiru. Prema tome korištena je funkcija Resources.Load() za specifičnu dodjelu objekta koji oružje stvara. Tako funkcioniraju i ostala oružja. Zatim je s pomoću funkcije Instantiate() stvoren objekt sjekira na poziciju igrača uz dodani pomak ulijevo.

Prilikom stvaranja sjekire, dohvaća se PlayerStats, kako bi se očitali trenutni modifikatori. Povećavaju se šteta, maksimalna penetracija te su vrijeme trajanja i brzina rotacije uvećane za psMod. Pokreće se i funkcija Destruction() koristeći Invoke(). Tako sjekiri se zadaje koliko će vremena proći prije nego što se uništi. Za vrijeme svakog okvira unutar funkcije Update(), sjekira se rotira u smjeru kazaljke na satu te prati igračevu poziciju i pomiče se zajedno s njim. U slučaju sudara s drugim objektom, zbog postavke sudarača na isTrigger, neće doći do pomicanja objekata, već će se pozvati funkcija OnTriggerEnter(). Za početak se provjerava ako je sudareni objekt označen s „Enemy“, ako je poziva se neprijateljeva funkcija TakeDamage() te se kao argument prosljeđuje šteta sjekire. Također se inkrementira varijabla koja broji koliko je neprijatelja pogođeno te u slučaju da varijabla dosegne vrijednost maksimalne penetracije, sjekira se ranije uništava.

Pojačanje oružja Sweeper dešava se za svakih 5 dobivenih stack-ova prilikom kojeg se stvara dodatna sjekira. Prva nova sjekira stvara se na suprotnoj strani, dok se na četvrtoj sjekiri, svaka stvara na jednoj strani svijeta. Za svaku sljedeću se stvaraju između postojećih sjekira.

Slika 10: Napad oružjem sweeper

3.5.3. Shooter

Oružje „Shooter“ namijenjeno je za ispucavanje strijela te pripada klasi lovac. Početni napad ispucava strijele u radijusu od 45 stupnjeva u smjeru gledanja igrača. Za svaki prosljeđeni stack se 45 dijeli s brojem stack-ova te se unutar for petlje koja započinje s 0, množi dobiveni broj. Na taj način će jedna strijela uvijek ići direktno u smjeru gledanja. Zatim se svaka neparna strijela šalje na negativnu vrijednost kako bi se ravnomjerno rasporedile.

Objekt koji Shooter stvara je strijela. Početne postavke strijele jednake su kao i kod sjekire, pošto i one prate broj pogođenih neprijatelja. Svaki okvir nakon stvaranja, strijela se nastavlja kretati u smjeru Vector3.forward te tako ide ravno do kad ne istekne zbog funkcije Invoke() ili do kad se ne zabije u maksimalan broj neprijatelja.

Kao pojačanje ovog oružja, za svaki stack iznad 5, dodaje se još jedna strijela među onima ispucanih ispred igrača te dvije strijele u nasumičnim smjerovima. Dodatne strijele odabiru nasumičan smjer koji nije u rasponu originalnih 45 stupnjeva.

Slika 11: Napad oružjem shooter

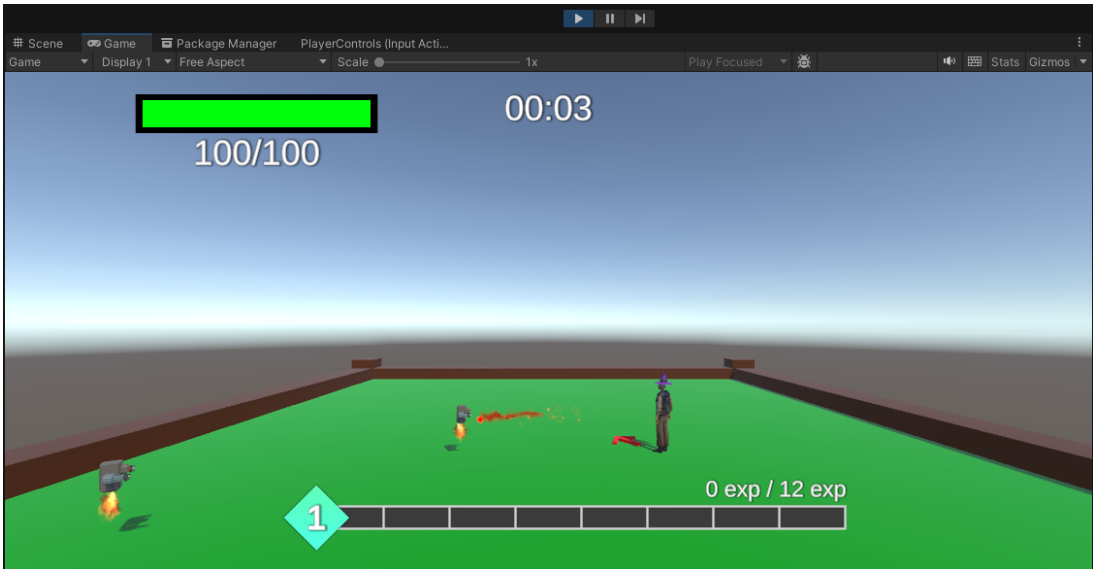
3.5.4. Finder

Posljednje oružje zove se „Finder“. Finder je oružje klase čarobnjak te u slučaju da je igrač manji od levela 5 objekt koji stvara je metak. Metak se stvara na mjestu igrača te prilikom stvaranja kontrolu preuzima skripta „Bullet“. Model metka, uključujući i pripadajući sustav čestica preuzeti su s Unity Asset Storea [7].

Za razliku od prijašnjih objekata koji prate zadanu putanju. Metak koristi dinamičan način kretanja. Prilikom stvaranja metka, osim početnih modifikacija i pozivanja funkcije Invoke() za uništenje objekta, pokreće se korutina FindClosest(). U trenutku kada je FindClosest() pokrenut, u polje allEnemies pohranjuju se svi trenutni objekti tipa „Enemy“. Zatim se za svakog pohranjenog neprijatelja uspoređuje udaljenost do kad se ne pronađe najbliži. Jednom kada

je najbliži pronađen, on postaje meta i dodaje mu se te se bilježi da je taj neprijatelj označen. Tako, ako bi novi metak pronašao istog neprijatelja kao metu, preskočio bi ga i tražio drugog najbližeg. U rijetkom slučaju da metak ne može pronaći niti jednog neprijatelja, metak će se uništiti kako ne bi stvarao probleme bez pronađene reference. Jednom kada je metak pronašao metu svaki okvir se pomiče prema meti te se pomoću funkcije Quaternion.Slerp() postepeno okreće prema traženom protivniku. Prilikom sudara s objektom označenog kao „Enemy“ protivniku se nanosi šteta, te mu se miče označenost, kako bi ga novi metak mogao smatrati metom te se sudareni metak uništava.

Kod ostalih oružja se prilikom dosezanja 5 stack-ova dodaje više objekata. Kod Findera objekt ispucavanja mijenja se iz metka u eksplozivni metak. Eksplozivni metak sadrži sva svojstva običnog metka, uz dodatak stvaranja eksplozije prilikom sudara s protivnikom. Stvorena eksplozija sadrži sudarač, s pomoću kojeg svaki pogođeni neprijatelj prima štetu. Vizualni dio eksplozije je sustav čestica preuzet s Unity Asset Sotrea [6]. Eksplozija traje 0.3 sekunde te svaki protivnik koji se naknadno ušeće u nju također prima štetu.



Slika 12: Napad oružjem finder

3.6. Korisnička sučelja

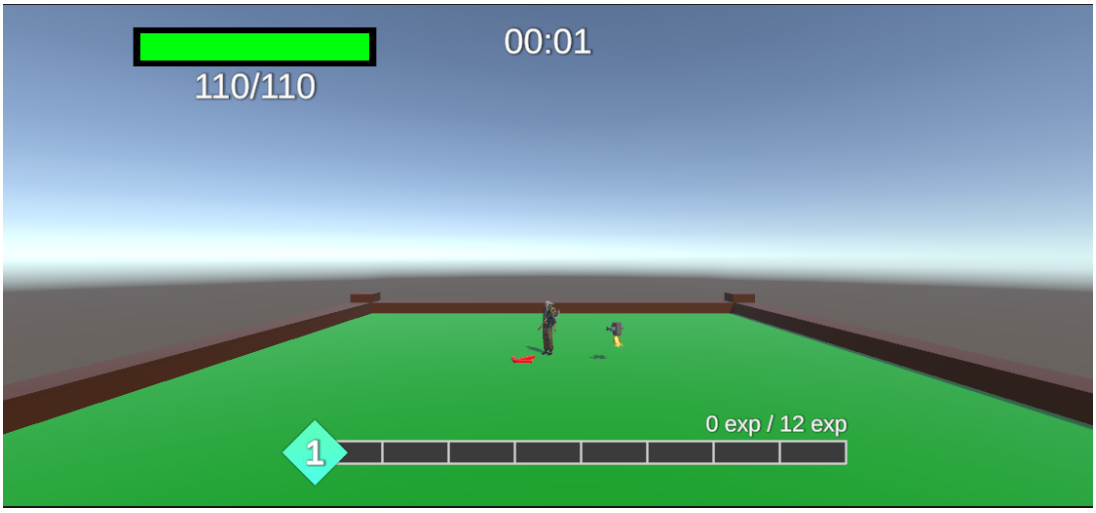
Ovo poglavlje opisuje sva korisnička sučelja vidljiva za vrijeme igre. Sučelja su podijeljena na više kategorija, pošto se neka pojavljuju nakon određenog događaja unutar igre, dok su ostala sučelja za to vrijeme onemogućena. Događaji koji okidaju promjene vezani su uz igrača, pa tako kada igrač dosegne novi level, Overlay sučelje se gase te se pokreće sučelje nagrada. Jednom kada je odabrana nagrada, trenutno sučelje postaje neaktivno te se ponovno omogućuje Overlay sučelje. Drugi okidač je smrt igrača, u kojem slučaju se Overlay sučelje onemogućuje te se prikazuje Završno sučelje.

3.6.1. Overlay sučelje

Overlay sučelje predstavlja korisničko sučelje vidljivo korisniku za vrijeme igre, ono pruža sve potrebne informacije vezane uz stanje igrača. Sučelje je nazvano tako, zato što prati kameru, odnosno korisnikov pogled. Na taj način korisnik ne mora tražiti elemente sučelja po svijetu već mu je sve prikazano. Elementi koji čine ovo sučelje su xp traka, traka životnih bodova te timer.

Xp traka vizualno i brojčano prikazuje koliko je igraču potrebno xpa da postigne novi level. Traka također prikazuje koji je igrač trenutno level. Druga prikazana traka, je traka životnih bodova (dalje: HP traka). HP traka također vizualno i brojčano prikazuje koliko igrač ima maksimalno životnih bodova te koliko života ima trenutno. Ta kalkulacija se izvršava s pomoću skripte "HPManagaer". Skripta je povezana sa slikom oblika fill, što omogućuje dinamičan prikaz omjera života i maksimalnih životnih bodova. Također je povezana i s tekstualnim oblikom prikaza. Kada traka postane aktivna mora pronaći igrača te preuzeti trenutne vrijednosti. Skripta još sadrži i funkciju ChangeBar(), koja kao argumente prima živote i maksimalne životne bodove te na taj način utječe na korisničko sučelje. Ta se funkcija poziva svaki put kad se mijenjaju igračevi životi, što znači da ju poziva igračeva funkcija TakeDamage() te oružje zacjeljivanja.

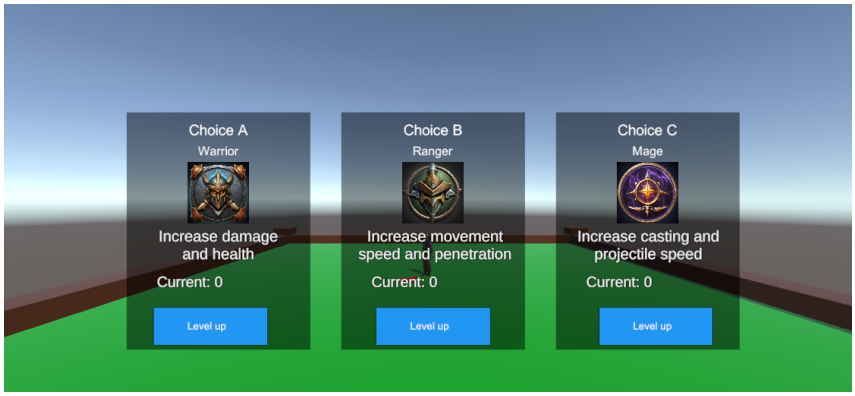
Na vrhu Overlay sučelja nalazi se brojač proteklog vremena Timer. Istoimena skripta, svaki okvir uvećava varijablu elapsedTime za Time.deltaTime, odnosno računa koliko je vremena prošlo za svaki okvir kada je Timer omogućen. Na taj način broji se samo vrijeme provedeno preživljavajući, dok se korištenjem funkcije Time.time, koja računa i vrijeme provedeno na odabir nagrada. Važnost Timera osim prikaza proteklog vremena korisniku, je i pohrana najboljeg vremena koje se odvija u sistemu najboljeg rezultata.



Slika 13: Prikaz overlay sučelja

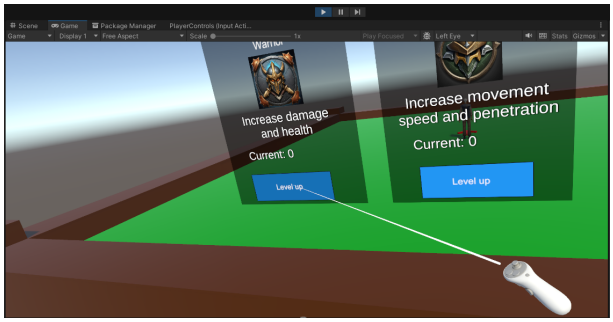
3.6.2. Sučelje nagrada

Sučelje nagrada poziva se kada igrač dosegne novi level. U tom trenu onemogućuju se overlay sučelje i spawner te se prikazuje zraka za odabir koja izvire iz desnog kontrolera. Brzina igre tada također postaje 0 kako u pozadini igrača ne bi napadali protivnici te kako se igrač ne bi mogao micati. Sučelje se sastoji od tri predloška izbora, jednog za svaku klasu. Korisnik ne mora žuriti prilikom odabira klase, zbog toga što se zabilježeno vrijeme preživljavanja ne računa za vrijeme odabira. Isto vrijedi i u slučaju pokušaja inflacije rezultata. Ikone kalas prikazane na sučelju nagrada generirane su pomoću online AI alata [12].



Slika 14: Prikaz sučelja nagrada

Predložak se sastoji od naslova koji označava koji je to odabir, imena klase vezane uz taj odabir, slike klase, opisa na što klasa utječe, trenutan level klase te gumba za odabir. Gumb za odabir glavna je komponenta predloška. Korisnik s pomoću zrake može pritisnuti gumb potezanjem okidača na desnom kontroleru. Prilikom pritiska gumba dešavaju se dva događaja. Prvi događaj je pozivanje funkcije klase koja pripadajućoj klasi povećava level te igraču daje odabrane benefite. U tom trenu pozvana funkcija iz skripte Class uvećava trenutan level klase, ali to nije odmah vidljivo na sučelju zbog drugog događaja. U istom trenu pritiskom na gumb okinuo se događaj koji iz skripte „ChoiceUI“ dohvaća funkciju Resume(). Ta funkcija vraća igru u stanje prije prikaza sučelja nagrada, odnosno sučelje nagrada i zraka za odabir postaju onemogućene, a overlay sučelje i spawner se ponovno uključuju. Vrijeme se vraća na 1 kako bi se igra nastavila normalnom brzinom.



Slika 15: Odabir pomoću zrake

3.6.3. Završno sučelje

Prilikom smrti igrača iz skripte PlayerStats pokreće se funkcija TheEnd(), koja onemogućuje overlay sučelje i spawner. Kao i kod sučelja nagrada, vrijeme igre u pozadini postaje 0 te je korisniku predstavljeno završno sučelje te je omogućena i zraka za odabir. Sučelje prikazuje natpis „Game Over“, preživljeno vrijeme trenutne igre, kao i najduže preživljeno vrijeme. Na dnu sučelja nalazi se gumb za ponovan početak igre.

Prilikom pokretanja završnog sučelja pokreće se i skripta „NewGame“ koja s pomoću procesa „Najbolji rezultat“ prikazuje vrijeme trenutne igre te dohvaća najbolji rezultat. Korisnik pritiskom na gumb okida događaj BeginAgain() u sklopu skripte NewGame. Funkcija s pomoću biblioteke UnityEngine.SceneManagement koristi funkciju SceneManager.LoadScene() kako bi ponovno pokrenula trenutnu scenu i ponovno započela igru.

Slika 16: Prikaz završnog sučelja

4. Zaključak

Za vrijeme izrade ovog završnog rada ispoštovani su ciljevi zadani prije početka implementacije. Uspješno je kreirana igra preživljavanja u kojoj igrač skuplja različita oružja i pojačanja. Konstantno se stvaraju neprijatelji koji s vremenom postaju sve jači. Igrač kroz pobjeđivanje neprijatleja osvaja nove levele te na taj način postaje jači kako bi čim duže preživio. Igra je također uspješno optimizirana za igru pomoću VR uređaja. Za vrijeme igre korisnik ne osjeća mučninu. Različita korisnička sučelja korisniku prikazuju sve potrebne informacije, kako za vrijeme igre, tako i za vrijeme odabria pojačanja. Na kraju igre, bilježi se najbolje vrijeme koje korisnika potiče da ponovno započne igru i postavi još bolje vrijeme. Konačan rezultat je igra koju je moguće igrati na računalu i VR uređaju.

Kroz ovaj rad dokazano je da je razvoj VR igara u današnje doba moguć bez previše kompromisa. Trenutno je potrebno donekle opširno podešavanje projekta, ali osim toga nema velike razlike u odnosu na razvoj 3D igre. Bez obzira na to, još uvijek nije preporučljivo programirati igru za VR bez posjedovanja VR uređaja. Potrebno voditi brigu o pogledu iz VRa, ali pomoću konstantnog testiranja s VR uređajem, vrlo je jednostavno donjeti potrebne promjene.

Popis literature

[1] J. Clement, *Video game market revenue worldwide from 2019 to 2029*, rujan 2024. adresa: https://www.statista.com/forecasts/1344668/revenue-video-game-worldwide.

[2] poncle, *Vampire Survivors*. adresa: https://store.steampowered.com/app/1794680/Vampire_Survivors/.

[3] Unity, *Unity Documentation*. adresa: https://docs.unity.com/.

[4] J. Linowes, *Unity Virtual Reality Projects*. Packt Publishing Ltd., 2018., ISBN: 978-1-78847-880-9.

[5] VertexModeler, *LowPoly Survival Character Rio*, rujan 2024. adresa: https://assetstore.unity.com/publishers/94530.

[6] I. O. Assets, *Simple Particles FX : Toon Effects*, rujan 2024. adresa: https://assetstore.unity.com/packages/vfx/particles/simple-particles-fx-toon-effects-244171.

[7] MTeStudio, *Fire Ice Projectile - Explosion*, rujan 2024. adresa: https://assetstore.unity.com/packages/vfx/particles/fire-ice-projectile-explosion-217688.

[8] D. R. (Johnson), *Fire & Spell Effects*, rujan 2024. adresa: https://assetstore.unity.com/packages/vfx/particles/fire-explosions/fire-spell-effects-36825.

[9] Khronos, *OpenXR*. adresa: https://www.khronos.org/openxr/.

[10] OpenAi, *ChatGPT*. adresa: https://chatgpt.com/.

[11] Microsoft, *Copilot*. adresa: https://copilot.microsoft.com/.

[12] S. Diffusion, *Stable Diffusion Online*. adresa: https://stablediffusionweb.com/.

Popis slika