Primjena algoritama umjetne inteligencije za generiranje igre.

Komljenović, Vlado

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Vlado Komljenović

Primjena algoritama umjetne inteligencije za generiranje igre

Završni rad

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

Rijeka, 17. rujna 2024.

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija
uniri

Rijeka, 15.2.2024.

Zadatak za završni rad

Pristupnik/ica: **Vlado Komljenović**

Naziv završnog rada: **Primjena algoritama umjetne inteligencije za generiranje igre**

Naziv završnog rada na engleskom jeziku: **Video game generation using artificial intelligence algorithms**

Sadržaj zadatka:

Proučiti alate i metode generativne umjetne inteligencije koje se primjenjuju za generiranje elemenata digitalne igre kao što je Ludo.ai, Promethean.ai, ChatGPT i Luma AI..

Osmisliti vlastitu 3D video igru, odabrati odgovarajući game engine i odgovarajuće alate umjetne inteligencije koji se mogu koristiti za generiranje elemenata računalne igre i opisati postupak izrade vlastite igre korištenjem tih alata. Proučiti alate za animaciju likova i druge alate koji pomažu u generiranju elemenata računalne igre. Usporediti rezultate i prokomentirati performanse testiranih alata. Prokomentirate mogućnosti i perspektive korištenja umjetne inteligencije u razvoju računalnih igara.

Mentor/ica
prof. dr. sc. Marina Ivašić-Kos

Voditelj za završne radove
Izv. prof. dr. sc. Miran Pobar

Zadatak preuzet: 15.2.2024.

KVlado
(potpis pristupnika/ice)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR1524020061400006966

Sažetak

Tema ovog završnog rada je proces izrade 3D računalne igre uz primjenu umjetne inteligencije(engl. artificial intelligence, AI). Cilj završnog rada je istražiti različite algoritme umjetne inteligencije za automatizaciju i generiranje sadržaja prilikom razvoja 3D *endless runner* računalne igre te u kolikoj mjeri su oni uspješni. Za razvoj ove igre, korišten je Unity kao glavno razvojno okruženje, dok je Microsoft Visual Studio služio kao alat za pisanje i uređivanje C# koda. AI alati koji se spominju su Ludo.ai, Promethean.ai, ChatGPT i Luma AI. Također, korištena je i online platforma Mixamo, koja služi kao alat za animaciju 3D likova, i *backend* servis LootLocker, koji omogućava programerima jednostavniju implementaciju raznih funkcionalnosti za računalnu igru.

Ključne riječi: 3D, računalna igra, AI, Unity, Visual Studio, C#, Ludo.ai, Promethean.ai, ChatGPT, Luma AI, Mixamo, LootLocker;

Sadržaj

1. Uvod

Cilj ovog završnog rada je istražiti primjenu umjetne inteligencije, tj. koliko umjetna inteligencija ubrzava i automatizira postupak izrade računalne igre. Upravo zbog toga, odlučili smo se za razvoj 3D računalne igre. Odabrana je 3D *endless runner* igra „Ri Run“, slična poznatoj igri „Temple Run“, zbog njene jednostavnosti i mogućnosti upotrebe umjetne inteligencije u stvaranju raznih objekata u igri poput prepreka, okruženja i slično. Posebno će biti naglašen svaki korišteni alat umjetne inteligencije, koji koriste strojno učenje i proceduralne metode za olakšavanje i ubrzanje procesa razvoja igre.

U današnje vrijeme tehnologija napreduje velikom brzinom, razna područja prolaze kroz modernizaciju, a jedno od tih područja je medij zabave, tj. videoigre. Razvoj igara je dinamično okruženje u kojemu se kreatori igara često suočavaju s izazovom kreiranja složenih videoigara u ograničenom vremenu. Videoigra je oblik igre temeljen na suvremenoj tehnologiji[1]. Industrija videoigara je jedna od najvećih na svijetu, a razlog tome je velik broj obožavatelja videoigara. Danas su videoigre puno dostupnije nego u prošlosti, mogu se igrati na računalima, mobitelima i raznim konzolama poput *PlayStation-a* i *Xbox-a.* Veza između videoigara i umjetne inteligencije postoji dugi niz godina, točnije od razvoja prvog programa za igranja šaha 1950. godine[2]. Izazov poraza ljudskih stručnih igrača strateških igri temeljenih na pravilima, poput šaha, značajno je unaprijedilo područje istraživanja umjetne inteligencije. Umjetna inteligencija, često poznata kao strojna inteligencija, oponaša ljudsku inteligenciju koja je osposobljena za razmišljanje poput ljudi. Umjetna inteligencija je tehnologija koja koristi podatke kao znanje kako bi se razvijena inteligencija mogla poboljšavati i učiti iz prethodnih pogrešaka [3]. Umjetna inteligencija može obavljati jednu od sljedeće četiri funkcije: djelovati poput čovjeka, razmišljati kao čovjek, razmišljati racionalno te djelovati racionalno[4]. Novije metode umjetne inteligencije korištene u računalnim igrama, primjerice za poboljšanje grafike, scenarija, uspostavljanja profila igrača, balansiranje složenosti ili dodavanja inteligentnih ponašanja ne igračkim likovima(engl. non-player character, NPC) teže ka tome da olakšaju posao kreatoru igrica. Zahvaljujući benefitima umjetne inteligencije, kreatori igrica više ne moraju tražiti potrebne resurse ili plaćati iste, već sami mogu stvarati resurse za razvoj igrice poput 3D modela, tekstura, materijala, dijaloga, zvukova, animacija i slično. Dakle, algoritmi umjetne inteligencije dovoljno su sofisticirani za:

- dizajniranje složene okoline
- optimizaciju igranja
- generiranje terena, priče i likova
- testiranje igre

U ovom radu fokusirati ćemo se na alate umjetne inteligencije za dizajn računalnih igara kao i na sam postupak. Ovi alati mogu automatizirati ponavljajuće zadatke, omogućujući dizajnerima da se usmjere na ostale grane razvoja. Osim toga, AI alati mogu ponuditi nove

ideje i prijedloge za elemente dizajna igre, poput likova, postavki i priča. Međutim, iako AI tehnologija nudi mnoge prednosti, važno je razumjeti koliko ustvari podržava dizajnere igara u obavljanju zadataka. Kako bismo to razumjeli, koliko AI alati ubrzavaju i olakšavaju proces razvoja videoigre, generirati ćemo 3D računalnu igru uz primjenu AI alata za dizajn.

Motivacija za izradu ovog rada je osobni interes u kreiranju igara i razumijevanja umjetne inteligencije. Razumijevanjem i primjenom AI alata, pratimo korak s vremenom jer se umjetna inteligencija sve više integrira u različite aspekte našeg svakodnevnog života i poslovanja. AI alati nisu samo tehnologija budućnosti, već i sadašnjosti, a njihovo korištenje omogućuje nam da budemo u tijeku s najnovijim inovacijama i promjenama na tržištu.

2. Razvojni alati

U ovome poglavlju detaljno ćemo opisati korištene alate u procesu razvoja 3D *endless runner* „Ri Run.“

2.1. ChatGPT

ChatGPT je *chatbot* i virtualni asistent razvijen od strane OpenAI-a, lansiran 30. studenog 2022. godine. Baziran na velikim jezičnim modelima (engl. large language model, LLM), omogućuje korisnicima da usmjeravaju i oblikuju razgovor prema željenoj dužini, formatu, stilu, razini detalja i jeziku. Sukcesivni korisnički upiti i odgovori uzimaju se u obzir u svakoj fazi razgovora kao kontekst. Do siječnja 2023. postao je najbrže rastuća potrošačka softverska aplikacija u povijesti, stekavši preko 100 milijuna korisnika i doprinoseći rastu trenutne vrijednosti OpenAI-a na 86 milijardi dolara[5]. Pojavom ChatGPT-a dolazi do razvoja ostalih konkurentnih proizvoda, primjerice Microsoft je razvio AI alat „Copilot“ po uzoru na GPT-4. U lipnju 2024. najavljena je suradnja između Apple Inc. i OpenAI-a, u kojoj se ChatGPT integrira u značajku Apple Intelligence u operativnim sustavima Applea. Neki su promatrači izrazili zabrinutost zbog potencijala ChatGPT-a i sličnih programa da zamijene ili oslabe ljudsku inteligenciju, omoguće plagijat ili potaknu širenje dezinformacija[5].

ChatGPT se temelji na specifičnim GPT(engl. Generative Pre-trained Transformer) modelima, naime GPT-4, GPT-4o i GPT-4o mini, koji su fino podešeni za ciljanje konverzacijskih upotreba[5]. Modeli funkcioniraju tako da koriste specijalne algoritme za pronalaženje obrazaca unutar niza podataka. ChatGPT je izvorno koristio model GPT-3 koji je radio na principu neuronskih mreža i treće generacije GPT-a, dok je dijalog optimiziran korištenjem pojačanog učenja s povratnim informacijama od ljudi(engl. Reinforcement Learning with Human Feedback, RLHF). Ovi modeli su izvlačili podatke sa interneta, a na internetu su se tada nalazili podaci pisani od strane ljudi te osim podataka, modeli su izvlačili i ljudske dijaloge kako bi zvučali što više kao ljudi.

Kao i svaki AI alat, ChatGPT također ima svoje prednosti i nedostatke. Neke od glavnih prednosti su:

- efikasnost
- ušteda novca i vremena
- edukacija
- dostupnost
- personalizacija
- razumijevanje materinskog jezika

Nedostatci ChatGPT:

- netočni odgovori
- ograničeno razumijevanje konteksta
- nedostatak kreativnosti
- ažurnost informacija

ChatGPT je jedno od najistaknutijih dostignuća u području AI-a u raznim industrijama pa tako i u industriji videoigara. Upravo zato, odlučili smo koristiti ChatGPT kao jednu vrstu savjetnika, naravno sa dozom opreza. Treba ga koristiti svjesno i sa razumijevanjem prednosti i mana koje donosi.

2.2. Unity

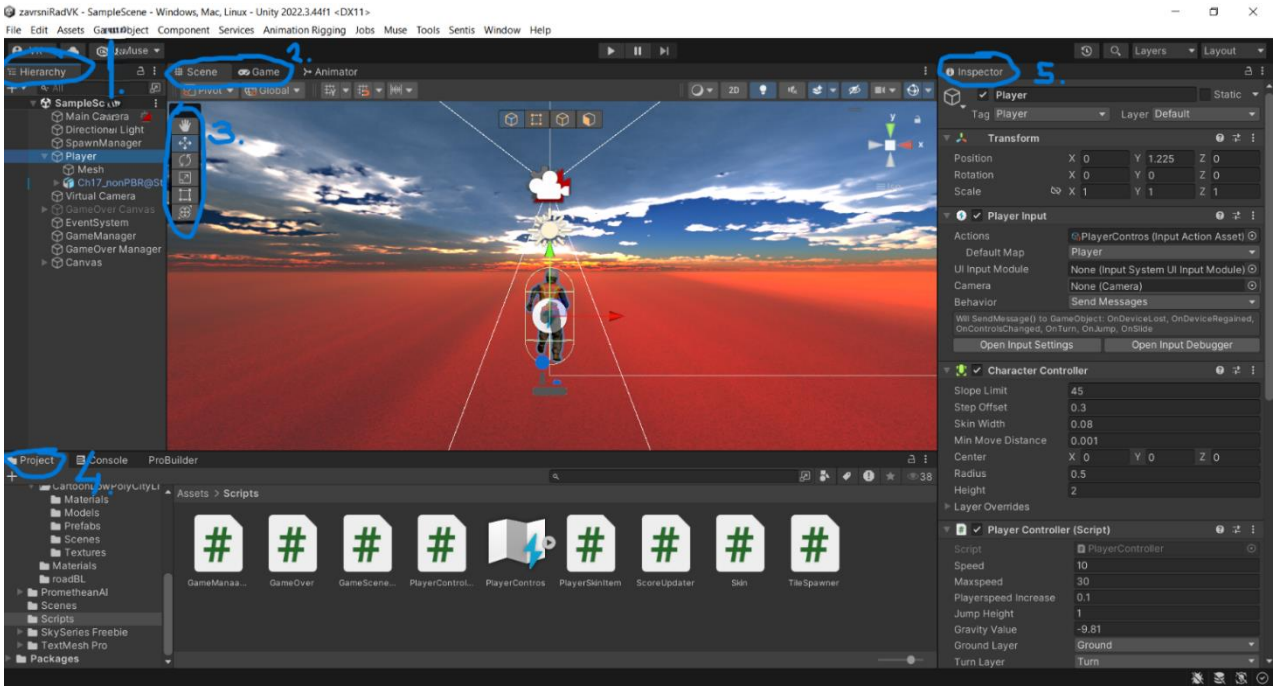
Unity je višestruko – platformski alat za razvoj videoigara, dostupan je u komercijalne svrhe i koristi se za proizvodnju 2D i 3D videoigara. Zbog svoje jednostavnosti, fleksibilnosti i dostupnosti popularan je među kreatorima videoigara. Unity je razvijen od strane Unity Technologies, a to je *cross-platform game engine* koji je prvi put objavljen u lipnju 2005. godine na svjetskoj konferenciji Applce Inc. kao ekskluzivni *game engine* za Mac OS X. Godine 2018. *game engine* je proširen i omogućio je podršku na više od 20 platformi[6]. Osim za proizvodnju 2D i 3D videoigara, *o*vaj *game engine* može se koristiti za razvoj proširene stvarnosti, virtualne stvarnosti, kao i simulacija i drugih primjena. Ovaj *game engine* u 21. stoljeću usvojile su tvrtke izvan video igara, poput filmova, arhitekture, automobilske industrije, građevinarstva i inženjeringa[7]. Unity se sastoji od tri važna sastavna dijela:

- *game engine:* omogućuje kreiranje, testiranje i igranje igara u različitim okruženjima.
- aplikacija: ovdje se sastavlja dizajn ili korisničko sučelje, uz opciju pregleda grafike i funkciju upravljanja igranjem.
- uređivač koda: integrirano razvojno okruženje(engl. Integrated Development Environment, IDE) pruža tekstualni uređivač za pisanje koda. Međutim, često se koristi poseban tekstualni uređivač kako bi se izbjegla zabuna.

2.2.1. Unity editor

Unity aplikacije se izrađuju koristeći Unity Editor, koji je dostupan na glavnim operacijskim sustavima računala, poput Windowsa, MacOS i Linuxa. Unity Editor omogućuje izradu videoigara koristeći scene. Scene mogu sadržavati različite 3D modele, materijale, animacije i logiku videoigre unutar skripte. *GameObjects* su osnovni građevni blokovi u sceni, možemo ih zamisliti kao spremnik sa različitim modelima, funkcijama i resursima. Dodavanjem komponentni unutar *GameObjects* bloka, mijenjamo ponašanje objekta. Primjer komponenti su skripte, *Collider* ili *RigidBody* komponenta. *GameObjects* i njihove komponente možemo kombinirati kako bismo dobili *prefabs,* koji su upotrebljivi između scena te ih se može dodati u scenu pomoću skripti. Kreatori videoigara mogu sami kreirati skripte koristeći C# programski

jezik. Skripte su glavni način dodavanja prilagođene logike videoigri. Unity koristi *open-source* .NET platformu sa skriptnim pozadinskim sustavima koji omogućuju upotrebu C#-a u Unity skriptiranju[10]. Kako bi skripta komunicirala sa Unity sustavom, mora naslijediti ugrađenu Unity-jevu klasu *MonoBehaviour*. Ova klasa pruža različite funkcije i atribute za komunikaciju sa Unity sustavom. Na Slici 1. možemo vidjeti izgled sučelja Unity Editora koje



Slika 1. Sučelje Unity Editora

je podijeljeno na pet glavnih prozora.

1. *Hierarchy* je prozor u kojemu organiziramo *GameObjects*. Prikazuje hijerarhijsku strukturu, objekte možemo grupirati u odnos roditelj-dijete. Na Slici 1. možemo vidjeti objekt roditelja „Player“ i njegove dijete objekt „Mesh.“
2. *Scene view i Game view* su glavni prozori u sučelju Unity Editora. *Scene View* nam omogućava uređivanje i postavljanje objekata na scenu iz različitih perspektiva i kutova. *Game view* je prozor koji nam omogućava pregled u realnom vremenu i testiranje igre.
3. *Toolbar* služi za promjenu točke gledišta te za pokretanje i zaustavljanje *Play* načina rada. Osim toga, sa *Toolbarom* možemo transformirati odabrani *GameObject*, rotiranje, skaliranje i slično.
4. *Project window* je prozor u kojemu možemo pronaći sve datoteke ili resurse dostupne za upotrebe u našem projektu, bez obzira koristimo li ih ili ne. Iz ovog prozora možemo povući resurs direktno u *Scene view* kako bismo odabrani resurs dodali u scenu. Resurs može biti bilo što, modeli, skripte, *prefabovi*, teksture, materijali i slično.
5. *Inspector* je prozor u kojemu pronalazimo i konfiguriramo detaljne informacije o *GameObjects*. Odabirom objekta u *Scene view* ili *Hierarchy prozoru*, vidimo komponente tog objekta u *Inspector* prozoru.

2.2.2 Unity Asset Store i Package Manager

Unity Asset Store i Package Manager važni su aspekti Unity-a. Asset Store je tržište na kojem treće strane mogu prodavati resurse za razvoj aplikacija ili dodatke za sam Unity editor[10]. Primjeri razvojnih resursa uključuju resurse poput 3D modela, zvučnih efekata, skripti ili predložaka, dok dodaci za Unity Editor dodaju dodatke(plug-in) i nove alate u sučelje editora. Unity-ev Package Manager može se koristiti za preuzimanje i instaliranje paketa sličnih onima iz Asset Store-a, kao što su alati za animaciju, korisničko sučelje i slično. Prema zadanim postavkama, Package Manager preuzima pakete iz spremišta kojim upravlja Unity Technologies, ali može pristupiti i spremištima zajednice[10].

2.3. AI alati

2.3.1. Ludo.ai

Ludo.ai je platforma pokretana umjetnom inteligencijom, dizajnirana za kreatore videoigara koji žele pojednostaviti svoje kreativne procese, uštediti vrijeme i povećati produktivnost. Neke od ključnih značajki:

- alat za koncepte igara: automatski generira inovativne ideje i koncepte za igre.
- alat za analizu tržišta: pruža uvid u tržišne trendove i preferencije igrača u stvarnom vremenu.
- generator slika: kreira slike i ikone za videoigre po želji.

Ludo.ai platforma zahtijeva određeno vrijeme za prilagodbu, ali kada se savlada, može uveliko ubrzati razvoj i smanjiti troškove.

2.3.2. Promethean.ai

Promethean.ai je alat koji koristi umjetnu inteligenciju kako bi olakšao proces organiziranja i postavljanja gotovih elemenata u 3D okruženjima, posebno u industriji videoigara, animacije i filmske produkcije. Ovaj alat ne generira nove, već koristi postojeće 3D modele i resurse koje postavlja u scenu na temelju korisničkih uputa. Glavna prednost ovog alat je sposobnost razumijevanja korisnikovih želja. Umjesto ručnog slaganja svakog modela, korisnik može opisati što želi, a Promethean.ai mu zatim uzima već postojeće modele i organizira ih u scenu. Primjerice, ako korisnik želi napraviti tinejdžersku sobu iz osamdesetih, alat uzima dostupne modele i automatski ih organizira na logične pozicije u skladu sa zahtjevom. Kako smo već rekli, Promethean.ai ne kreira nove 3D modele, već se oslanja na resurse koje korisnik već ima. Važno je da korisnik ima gotovu bazu podataka sa svim resursima potrebnim za rad, kako bi ovaj alat funkcionirao.

Promethean.ai alat je je jedan od najpraktičnijih alata za brz i jednostavan način izrade dizajna videoigre, ali zahtjeva već spremne resurse.

2.3.3. Luman AI – Genie

Luma Genie je AI model za pretvaranje teksta u 3D modele. Transformira korisnikove riječi u 3D objekte pogodne za integraciju u videoigru. Primjerice, korisnik upiše riječ „car“ i dobije

gotov 3D model automobila. Alat generira različite formate datoteka, ovisno o potrebama korisnika. Osim toga možemo prilagoditi teksturu generiranog modela i još postoji i mogućnost poboljšanja razlučivosti modela. Najbolje od svega je to što su ove funkcionalnosti potpuno besplatne, za razliku od drugih AI alata. Luma Genie se ističe među alatima za pretvaranje teksta u slike, poput MidJourney ili DALL-E, stvarajući potpuno interaktivne 3D objekte koje možete gledati iz svih kutova. Nevjerojatno učinkovit, generira predmet s materijalima, bojama i svim potrebnim detaljima za 3D printanje ili korištenje u produkcijskim okruženjima za manje od 10 sekundi[11]. Luma Genie je pogodan za korištenje i kreativcima , zbog jednostavnosti korištenja i brzine stvaranja modela.

2.4. LootLocker i Mixamo

LootLocker je „plug and play“ unakrsno-platformski *backend* za developere, koji omogućava izgradnju, lansiranje i upravljanje njihovim najboljim igrama[12]. LootLocker se sastoji od tri glavne skupine značajki:

- sistemi igre: ušteda vremena i mogućnost unapređivanja vlastite videoigre korištenjem ljestvice(engl. leaderboard) i napretka(engl. progression).
- upravljanje sadržajem: sortiranje, uređivanje i upravljanje svime, od vizualnih dodataka do valuta, od korisnički generiranog sadržaja(engl. User -Generated Content, UGC) do preuzetih sadržaja(engl. Downloadable Content, DLC).
- upravljanje igračima: pohrana podataka o igračima na jednom mjestu, pristup profilima i listama prijatelja. Upravljanje prijavama, porukama i darovima kako biste ih zadržali zainteresiranima.

Slika 2. LootLocker logotip

Mixamo je besplatna web aplikacija koju je razvila tvrtka Adobe, a namijenjena je 3D dizajnerima videoigara koji žele brzo kreirati i animirati lika. Glavna značajka ove aplikacije je olakšavanje procesa animiranja 3D modela putem unaprijed napravljenih animacija i likova. Neke od glavnih značajki Mixamo-a su:

- baza likova: baza sa unaprijed kreiranim 3D likovima, primjerice ljudi, roboti i slično.

- postavljanje kostura(engl. rigging): automatsko postavljanje kostura. Korisnik može sam prenijeti vlastiti 3D model, a Mixamo će mu dodati kostur koji je neophodan za animiranje lika.
- baza animacije: baza sa unaprijed kreiranim animacijama koje su lako primjenjive na 3D modele. Animacije pokrivaju širok raspon pokreta. Korisnik može pregledati animaciju na svom modelu prije nego li preuzme animaciju.
- prilagodba animacije: korisnik može mijenjati brzinu animacije, intenzitet pokreta, poziciju lika i slično.
- web platforma: ne zahtijeva instalaciju aplikacije, već je dostupan preko web preglednika

Mixamo se primjenjuje, osim u videoigrama, i za vizualne efekte i film te za virtualnu stvarnost(engl. Virtual Reality, VR) i proširenu stvarnost(engl. Augmented reality, AR). Mixamo je jednostavan i za korištenje zbog čega je odličan izbor za početnike.

Slika 3. Mixamo logotip

3. Izrada videoigre

U ovom poglavlju biti će opisani svi koraci izrade videoigre. Počevši od ChatGPT-a pa sve do ostalih AI alata, objasniti ćemo primjenu tih alata, kako su nam poslužili i za što smo ih koristili. Osim toga, opisati ćemo postavljanje i razvoj igre u Unity sustavu i detaljno ćemo opisati logiku videoigre implementiranu unutar skripti napisanih u programskom jeziku C# koristeći razvojno okruženje Visual Studio.

3.1. Analiza AI alata

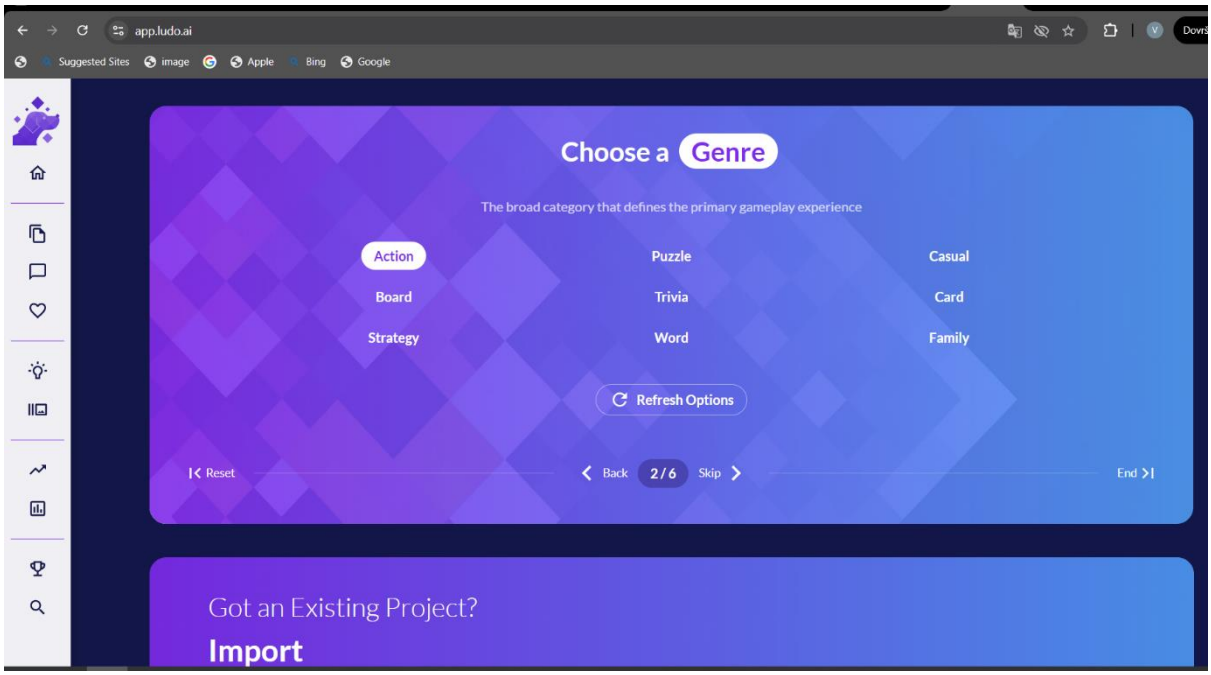
Analizu AI alata za pomoć pri razvoju igrice proveli smo sa AI alatom ChatGPT. Odabrali smo ChatGPT jer sa njim već imamo iskustva i znamo na koji način funkcionira. Pitali smo ga da nam detaljno recenzira AI alate koje smo mu ponudili, a ponuđeni AI alati su bili: Scenario.ai, Promethean AI, Ludo.ai i Layer.ai. Dobili smo sljedeći odgovor od ChatGPT-a:

```
1. Scenario.ai
Scenario.ai je alat koji koristi umjetnu inteligenciju za generiranje prilagođenih
sredstava za igre. Ovaj alat omogućuje korisnicima treniranje vlastitih AI modela na
specifičnim stilovima i zahtjevima njihovih igara. Umjetnici i dizajneri mogu generirati
prilagođene vizualne elemente poput likova, objekata i okruženja u velikim količinama, a da
pritom zadrže visoku kvalitetu. Scenario.ai omogućuje brzu iteraciju i stilizaciju
sredstava prema potrebama igre, čime ubrzava proces kreacije.

2. Promethean AI
Promethean AI koristi umjetnu inteligenciju za proceduralno generiranje 3D okruženja za
videoigre. Umjetnicima omogućuje da opišu željenu scenu, nakon čega AI automatski generira
okruženje u skladu s opisom, bilo da se radi o šumi, uredu ili nekom drugom prostoru. AI
također nudi interaktivne prijedloge i prilagodbe kako bi se osiguralo da krajnja scena
odgovara dizajnerskoj viziji. Ovaj alat je povezan s popularnim alatima za izradu igara kao
što su Unity i Unreal Engine, što olakšava implementaciju generiranih scena u igre.

3. Ludo.ai
Ludo.ai je platforma za generiranje koncepata za igre. Korisnici unose promptove koji
opisuju osnovne ideje za igru, poput teme ili žanra, a alat potom generira ideje za
mehanike, priču i druge elemente igre. Ludo.ai koristi algoritme koji analiziraju postojeće
igre i tržišne trendove, čime pomaže korisnicima da prilagode svoje ideje potrebama
publike. Osim toga, alat omogućuje analizu postojećih igara i nudi prijedloge za
poboljšanja ili dodatne mehanike koje bi igru mogle učiniti zanimljivijom.

4. Layer.ai
Layer.ai koristi proceduralne algoritme za generiranje velikih i detaljnih svjetova u
igrama. Alat je posebno fokusiran na stvaranje terena i krajolika, poput planina, rijeka, i
vegetacije, koristeći simulacije prirodnih zakona za realistično generiranje. Iako
automatski generira scene, korisnici imaju kontrolu nad prilagodbom kako bi osigurali da
svjetovi odgovaraju njihovoj umjetničkoj viziji. Layer.ai također omogućuje simulaciju
prirodnih procesa poput erozije i vremenskih uvjeta, čime stvara dinamične i realistične
svjetove za igre.
```

Kako bismo se uvjerili u točnost informacija, odlučili smo analizirati svaki AI alat samostalno bez umjetne inteligencije. Podaci koje je generirao ChatGPT bili su poprilično točni u ovom slučaju. Za početak, kako još nismo imali ideju koju videoigru bismo razvili, odabrali smo Ludo.ai koji će nam generirati ideju za videoigru.

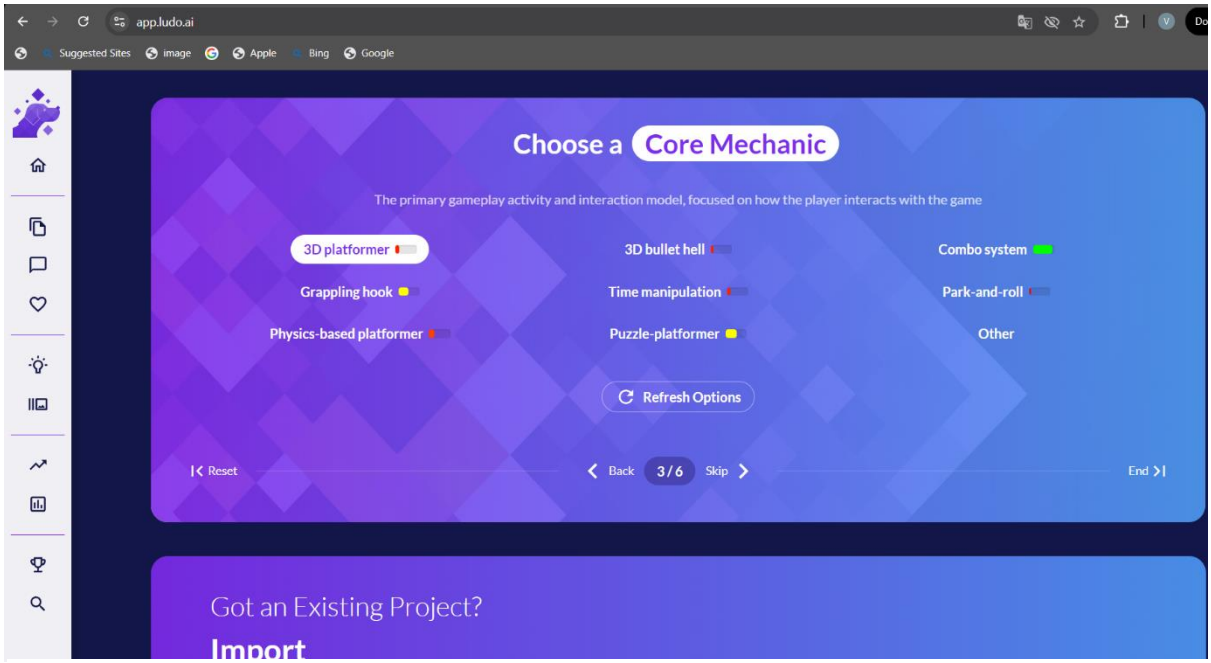
3.2. Ideja za videoigru

Nakon prijave na Ludo.ai, odabiremo „New Game Concept“ nakon kojega slijedi postavljanje koncepta. Ludo.ai nas pita želimo li videoigru za mobitel ili desktop. Odabrali smo desktop. Nakon toga nas pita koji žanr igrice želimo, od ponuđenih žanrova sa Slike 4. odabrali smo akcijski žanr.



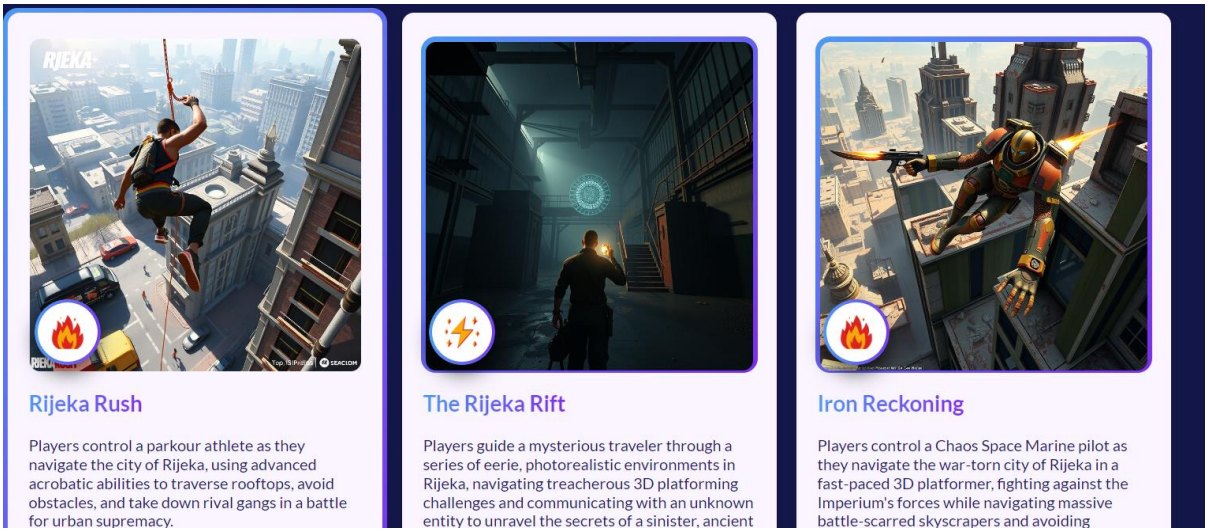
Slika 4. Odabir žanra

Sljedeće pitanje je kakvu mehaniku želimo u igrici, od ponuđenih sa Slike 5. odabrali smo 3D *platformer* mehaniku.



Slika 5. Odabir mehanike

Slično kao na Slici 4. i Slici 5., Ludo.ai nas u sljedećim koracima pita perspektivu igranja, postavljanje svijeta i odabir stila. Odabrali smo treće lice i realističnu 3D igru. Za postavljanje svijeta sami smo odabrali da želimo izgled svijeta kao što izgleda grad Rijeka.



Slika 6. Ludo.ai - ponuđene teme

Ludo.ai nam je kreirao više tema na osnovu naših odabira. Na Slici 6. možemo vidjeti ponuđene teme. Nažalost, ponuđene teme su bile previše komplicirane, tražili smo jednostavniju temu jer je cilj našeg rada istražiti AI alate za generiranje videoigre za što nam ne treba brutalna videoigra, već jednostavna 3D videoigra na kojoj možemo pokazati primjenu AI alata. Ludo.ai ima svoj „Chat Assistant“ pa smo odlučili zatražiti dodatnu pomoć. Pitali smo da nam kreira videoigru u gradu Rijeci sličnu poznatoj igrici „Temple Run“, koja je 3D *endless runner* videoigra pa bi trebalo biti relativno lako kreirati sličnu igru po mehanici. Dobili smo sljedeći odgovor:

```
Želite igru sličnu Temple Runu smještenu u Rijeku. Znači, mogli bismo kreirati koncept koji
bi uključivao:

Naslov: Rijeka Trči

Mjesto radnje: Igra se odvija u povijesnoj i slikovitoj Rijeci. Igrač mora prelaziti kroz
poznate znamenitosti grada kao što su Gradski toranj, Guvernerova palača i Riječka
katedrala, dok izbjegava prepreke i skuplja kovanice.

Radnja: Igrač upravlja lokalnim likom, možda mladim Riječaninom koji je otkrio skrivenu
kartu. Dok trči kroz grad, mora izbjeći zlobnog "Male Učke" (mistično biće iz Rijeke) koji
pokušava doći do blaga.

Igranje:

Igrač upravlja likom koji trči kroz ulice Rijeke, izbjegavajući uske ulice, izbjegavajući
prodajne štande i preskakajući prepreke kao što su kaluže i gradilišta.
Lik može skupiti kovanice, moćne objekte i predmete kao što su tradicionalna riječka
kuhinja (kao što su paprika ili peka) koji mu daju privremeno pojačanje ili sposobnosti.
Kako igrač napreduje, povećava se razina težine, s više prepreka, užim zakretima i bržim
tempom.

Jedinstvene značajke:

Uključite lokalnu riječku kulturu i povijest, kao što je pomorsko nasljeđe grada,
prikazivanjem historijskih brodova ili čamaca u pozadini.
```

```
Koristite autentičnu riječku arhitekturu i znamenitosti kako biste stvorili impresivno
iskustvo.
Razmislite o uključivanju "dnevno-noćnog ciklusa" ili "sezonskih promjena" kako biste igri
dali dinamičan osjećaj.

Vizualni i audio sadržaj:

Uključite žive, šarene grafike koje odražavaju jedinstvenu atmosferu i arhitekturu Rijeke.
Koristite tradicionalnu hrvatsku glazbu ili folklorno inspirirane zvukove kako biste
stvorili impresivni soundtrack.

Ciljna publika:

Casual igrači koji uživaju u beskonačnim trkačima kao što je Temple Run.
Lokalci i turisti koji su zainteresirani za istraživanje riječke kulture i povijesti.
```

Tema koju nam je predložio Ludo.ai nam se svidjela pa smo odlučili kreirati videoigru temeljenu na predloženoj ideji. Ludo.ai nije besplatan alat, međutim novi korisnici dobivaju pet kredita za isprobati alat. Nama je ostalo još malo kredita za isprobati aplikaciju pa smo odlučili generirati sliku grada Rijeke koju bismo pokušali ubaciti u videoigru prilikom razvoja igre. Kako bismo jasnije objasnili kakvu sliku želimo da generira Ludo.ai ubacili smo inicijalnu Sliku 7.

Slika 7. Grad Rijeka – inicijalna slika za Ludo.ai

Na Slici 8. možemo vidjeti kakvu sliku nam je Ludo.ai generirao. Zadovoljni smo sa generiranom slikom, ali ne znamo hoćemo li je moći upotrijebiti na način koji smo zamislili.

Slika 8. Ludo.ai generirana slika grada Rijeke

3.3. Postavljanje videoigre

Kako bi nam bilo jednostavnije snalaziti se tijekom izrade videoigre, pitali smo Ludo.ai da nam detaljno opiše postupak izrade 3D *endless runner* videoigre. Dobili smo sljedeći odgovor:

```
1. Otvaranje novog 3D Unity projekta
Pokreni Unity Hub.
Klikni na New Project.
Odaberi 3D template.
Nazovi projekt (npr. "Endless Runner").
Postavi lokaciju i klikni Create Project.

2. Postavljanje osnovne scene
Kada se projekt učita, vidjet ćeš praznu scenu s Main Camera i Directional Light.
```

Main Camera: Provjeri kameru u Inspector i pozicioniraj je tako da gleda prema igraču (npr. X: 0, Y: 2, Z: -5) i rotiraj je kako bi gledala prema naprijed (X: 10, Y: 0, Z: 0).

3. Dodavanje platforme za kretanje
Desni klik u Hierarchy > 3D Object > Plane kako bi dodao ravnu platformu za igru.
Skaliraj platformu tako da bude dulja (npr. X: 3, Y: 1, Z: 20) kako bi igrač imao dovoljno prostora za trčanje.
Dodavanje materijala: Možeš stvoriti novi materijal (desni klik u Assets > Create > Material) i odabrati boju ili teksturu kako bi platforma izgledala atraktivnije.

4. Dodavanje igrača
Desni klik u Hierarchy > 3D Object > Capsule da stvoriš igrača.
Pomakni kapsulu iznad platforme (npr. X: 0, Y: 1, Z: -9).
Dodaj komponentu Rigidbody (u Inspector prozoru klikni na Add Component > Rigidbody) kako bi omogućio fiziku igraču.

5. Dodavanje prepreka
Desni klik u Hierarchy > 3D Object > Cube da stvoriš prepreku.
Postavi prepreke na različite pozicije ispred igrača na platformi (npr. X: 0, Y: 1, Z: 10).
Skaliraj ih po potrebi (npr. X: 1, Y: 2, Z: 1) kako bi stvarale izazov igraču.

6. Dodavanje animacija i pokreta
Mixamo: Ako želiš dodati unaprijed definirane animacije, možeš posjetiti Mixamo (mixamo.com) i preuzeti gotove animacije trčanja i skakanja.
Uvezi animirani 3D model u Unity (povuci datoteku u Assets).
Zamijeni kapsulu animiranim modelom i poveži animacije s igračem.

7. Dodavanje beskonačnog terena (endless terrain)
Umjesto beskonačne platforme, dupliciraj postojeću platformu i pozicioniraj je na kraju prve (npr. Z: 20). Nastavi duplicirati platforme kako bi igrač imao kontinuirani put.
Na svakom kraju platforme dodaj oznaku (prazan objekt) koja će kasnije poslužiti za spawn novih platformi kada igrač prođe određenu točku.

8. Dodavanje bodovnog sustava (UI)
Desni klik u Hierarchy > UI > Text da kreiraš tekstualni element za prikaz bodova.
U Inspectoru, postavi veličinu, font i poziciju tekstualnog elementa na ekranu (npr. gornji lijevi kut).
Bodovi se mogu dodjeljivati automatski ovisno o udaljenosti koju igrač prijeđe.

9. Testiranje igre
Klikni na Play u gornjem dijelu Unity sučelja kako bi pokrenuo igru.
Testiraj kretanje igrača i detekciju prepreka.
Ako koristiš animirani model, provjeri da li su sve animacije pravilno povezane i da li igrač reagira na prepreke.

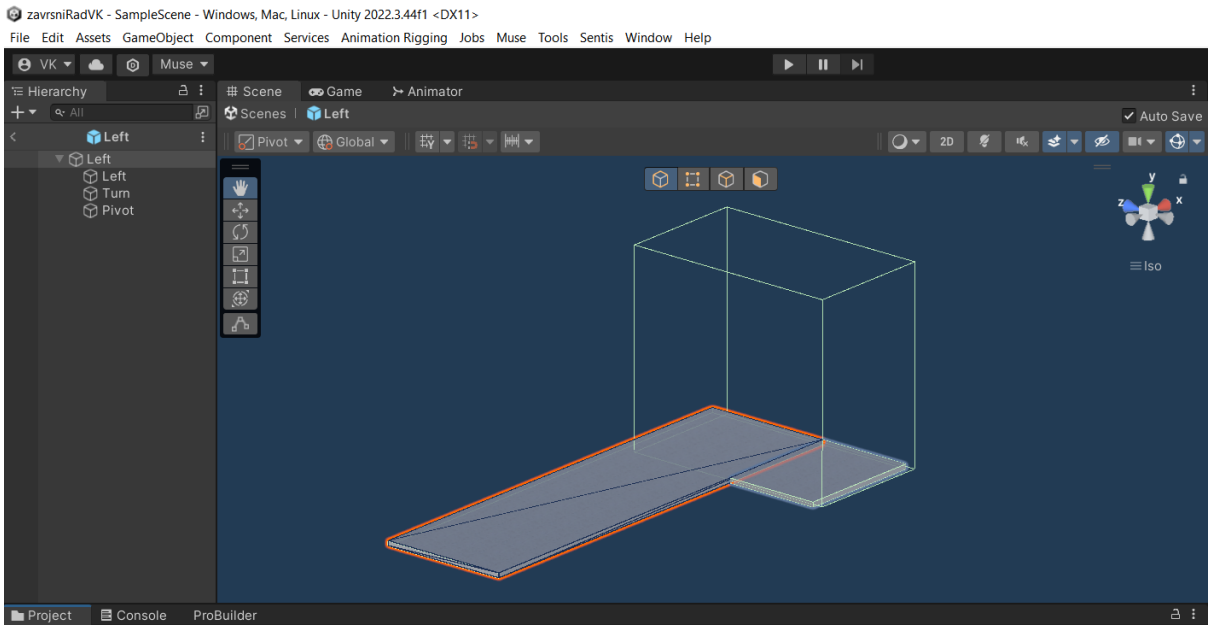
10. Optimizacija i prilagodba
Dodaj više prepreka različitih oblika i veličina kako bi igra postala izazovnija.
Eksperimentiraj s vizualnim efektima i zvukovima kako bi igra bila vizualno atraktivnija.

Odgovor generiran od strane Ludo.ai alata je poprilično dobro generiran pa smo odlučili se voditi njegovim smjernicama, ali smo na ove osnovne korake odlučili dodati svoje elemente kako bi prilagodili igru našim idejama i potrebama. Također nismo pratili korake po redu kako su navedeni.

3.4. Dodavanje staza za kretanje

Unutar Unity Editora otvaramo Package Manager kako bi preuzeli ProBuilder paket. Paket je namijenjen za izgradnju, uređivanje i dodavanje vlastitih geometrijskih oblika. Pomoću njega napravili smo više staza(engl. Tiles) po kojima će igrač trčati. Napravili smo četiri oblika staze: ravna staza, staza sa skretanjem lijevo, staza sa skretanjem desno i staza raskrižje. Na svakoj stazi se nalazi *Box Collider* koji služi kao provjera je li igrač unutar *Box Collider-a* stisnuo tipku za skretanje lijevo ili desno. Ako je pritisnuo, onda može skrenuti. Na Slici 9. možemo vidjeti primjer staze sa skretanjem lijevo. Kako bismo znali imamo li stazu sa skretanjem lijevo ili stazu sa skretanjem desno ili stazu raskrižje moramo definirati logiku u skripti. Kreirali smo novu skriptu „Tile.cs.“ Skripta izgleda ovako:

```
using UnityEngine;


namespace RiRun {

public enum TileType
    {
        STRAIGHT,
        LEFT,
        RIGHT,
        SIDEWAYS
    }

public class Tile : MonoBehaviour
    {
        public TileType type;
        public Transform pivot;
    }

}
```



Slika 9. Izgled staze sa skretanjem lijevo

Skripta koristi *MonoBehaviour* kao bazu, što znači da se može dodati kao komponenta na Unity *GameObjects*. *Enum* možemo gledati kao listu staza, podaci unutar *enum-a* su cijeli brojevi, primjerice „STRAIGHT=0“, „LEFT=1“, „RIGHT=2“, „SIDEWAYS=3.“ Klasu „Tile“ smo napravili kako bi jednostavno mogli pristupiti ovoj skripti iz drugih skripti. „Pivot“ je korišten za ispravno povezivanje staza.

3.4.1. Generiranje staza za kretanje

Skripta koja upravlja generiranjem staza je „TileSpawner.cs.“ Ključne funkcionalnosti skripte su sljedeće:

1. **Inicijalizacija i generiranje početne staze**

```
private void Start()
{
    currentTiles = new List<GameObject>();
    currentObstacles = new List<GameObject>();
    Random.InitState(System.DateTime.Now.Millisecond);

    for (int i = 0; i < tileStartCount; ++i)
    {
        SpawnTile(startingTile.GetComponent<Tile>());
    }
SpawnTile(SelectRandomGameObjectFromList(turnTiles).GetComponent<Tile>());
}
```

- postavljaju se početne staze i prepreke(engl. obstacles). Prepreka je zasad 3D objekt kocke pretvoren u *prefab*. Pomoću `SpawnTile` metode generiraju se nasumice staze.

2. **Generiranje staze**

```
private void SpawnTile(Tile tile, bool spawnObstacle = false)
{
    Quaternion newTileRotation = tile.gameObject.transform.rotation *
Quaternion.LookRotation(currentTileDirection, Vector3.up);
    prevTile = GameObject.Instantiate(tile.gameObject, currentTileLocation,
newTileRotation);
    currentTiles.Add(prevTile);

    if (spawnObstacle) SpawnObstacle();
    if (tile.type == TileType.STRAIGHT)
        currentTileLocation +=
Vector3.Scale(prevTile.GetComponent<Renderer>().bounds.size, currentTileDirection);

    SetTileVisibility(prevTile, false);
}
```

- funkcija `SpawnTile` stvara nove staze na trenutnoj poziciji i rotaciji. Ako je potrebno stvara se prepreka te ako je staza ravna, ažurira se trenutna pozicija na osnovu veličine prethodne staze.

3. Upravljanje vidljivošću staze

```
        private void UpdateTileVisibility()
        {
            foreach (var tile in currentTiles)
            {
                float distanceToPlayer = Vector3.Distance(playerTransform.position,
tile.transform.position);
                if (distanceToPlayer < visibilityDistance)
                {
                    SetTileVisibility(tile, true);
                }
                else
                {
                    SetTileVisibility(tile, false);
                }
            }
        }
```

- funkcija `UpdateTileVisibility` provjerava udaljenost između igrača i svake staze. Staze koje su unutar dometa vidljivosti(`visibilityDistance`) se aktiviraju, dok ostale ostaju deaktivirane.

4. Upravljanje smjerovima i dodavanje novih pločica

```
        public void AddNewDirection(Vector3 direction)
        {
            currentTileDirection = direction;
            DeletePreviousTiles();

            currentTileLocation += tilePlacementScale;

            int currentPathLenght = Random.Range(minimumStraightTiles,
maximumStraightTiles);
            for (int i = 0; i < currentPathLenght; ++i)
            {
                SpawnTile(startingTile.GetComponent<Tile>(), (i == 0) ? false : true);
            }
            SpawnTile(SelectRandomGameObjectFromList(turnTiles).GetComponent<Tile>(),
false);
        }
```

- metoda `AddNewDirection` ažurira trenutni smjer staza, briše prethodne staze i generira novi smjer staza u veličinama između minimalne i maksimalne.

Ova skripta omogućuje kontinuirano generiranje i upravljanje sa stazama. Osim toga, mogu se lako prilagoditi parametri skripte u slučaju potrebnih prilagodbi dizajnu.

3.5. Dodavanje igrača

Kreiramo prazan *GameObject* i nazivamo ga „Player.“ Dodajemo mu *Player Input* komponentu koja služi za upravljanje ulaznim podacima igrača, primjerice kada igrač pritisne tipku *Space, „*Player“ će skočiti. Unutar „Player“ objekta dodajemo 3D kapsulu imena „Mesh.“ Kapsula je dijete od „Player“ objekta. Kako bi igrač mogao skakati, klizati i skretati lijevo desno, moramo definirati logiku koja će omogućiti sve kretnje igrača. Kreirali smo skriptu „PlayerController.cs.“ Njezin najvažniji dio koji upravlja svim ključnim aspektima ponašanja igrača i poziva se svaki okvir(engl. frame) izgleda ovako:

```
private void Update()
 {
     if (!IsGrounded(20f))
     {
         GameOver();
         return;
     }

     score += scoreMultiplier * Time.deltaTime;
     scoreUpdateEvent.Invoke((int)score);
     controller.Move(transform.forward * playerSpeed * Time.deltaTime);

     if (IsGrounded() && playerVelocity.y < 0)
     {
         playerVelocity.y = 0f;
         animator.SetBool("IsJumping", false);
     }

     playerVelocity.y += gravity * Time.deltaTime;
     controller.Move(playerVelocity * Time.deltaTime);

     if (playerSpeed < maxspeed)
     {
         playerSpeed += Time.deltaTime * playerspeedIncrease;
         gravity = gravityValue - playerSpeed;

         if (animator.speed < 1.25f)
         {
             animator.speed += (1 / playerSpeed) * Time.deltaTime;
         }
     }
 }
```

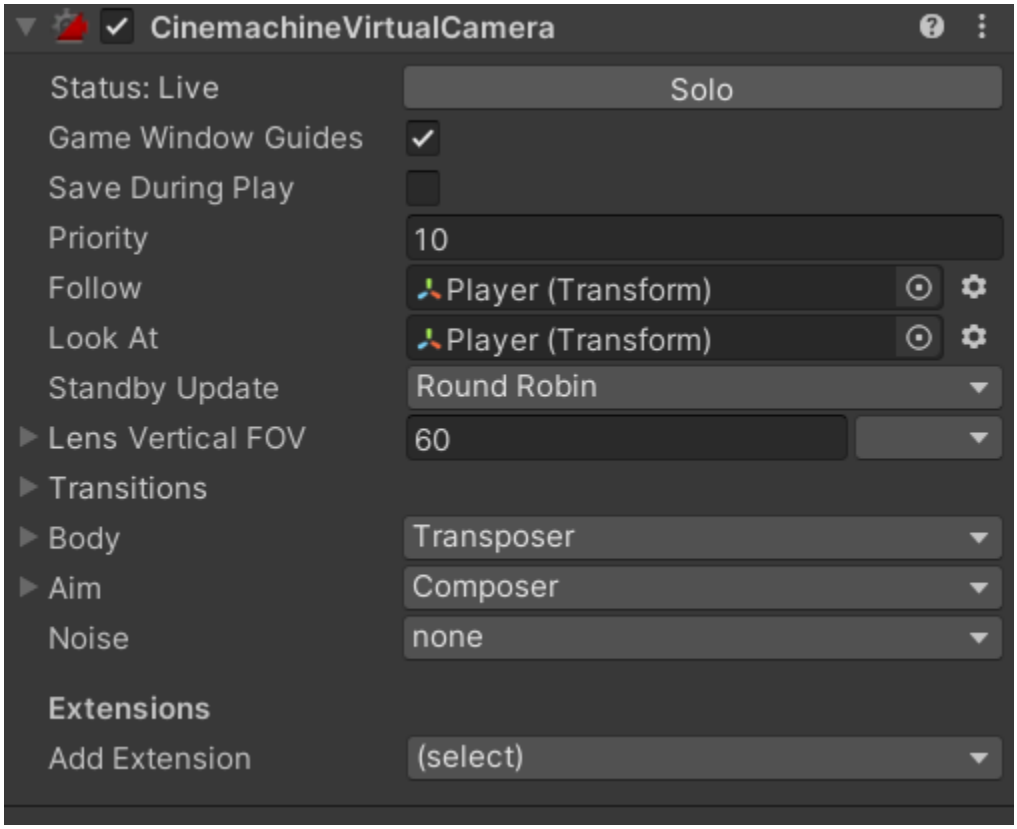
Funkcija `Update` izvršava sljedeće:

- provjerava je li igrač na tlu, tj. stazi pomoću metode `IsGrounded`. Ako se igrač previše udalji od staze(parametar 20f) poziva se metoda `GameOver`.
- upravljanje sa bodovima, bodovi se povećavaju tijekom vremena igranja. Varijabla `scoreMultiplier` određuje koliko se bodova dodaje za svaku sekundu igre.
- Kretnju igrača prema naprijed uz pomoć `CharacterController.Move` metode, koristeći trenutnu brzinu igrača(`playerSpeed`). Važno za napomenuti je to da kada se koristi `Move` metoda, Unity automatski upravlja sudaranjem i fizikom pa nisu potrebne komponente „RigidBody“ i „BoxCollider“ na „Playeru.“

- Upravljanje gravitacijom. Gravitacija se dodaje na vertikalnu brzinu igrača kako bi dobili realističan prikaz gravitacije. Gravitacija se smanjuje dok se horizontalna brzina igrača povećava.
- Povećanje brzine animacije unutar animatora sve dok brzina igrača ne dosegne maksimalnu brzinu.

Ovo nije cijela skripta, već je iz nje izvučeno najbitnije, ostalo su mehanike skakanja, skretanja i klizanja koje se mogu pronaći u Unity dokumentaciji.

Odlučili smo na „Playera“ dodati „Cinemachine“, alat koji upravlja kamerom. Paket smo preuzeli iz Package Managera unutar Unity-a. Ovaj alat nam je omogućio glatke prijelaze između kamera i praćenje objekta, u našem slučaju pratimo i fokusiramo se na „Playera“ što možemo vidjeti i na Slici 10. koja prikazuje postavke naše virtualne kamere.



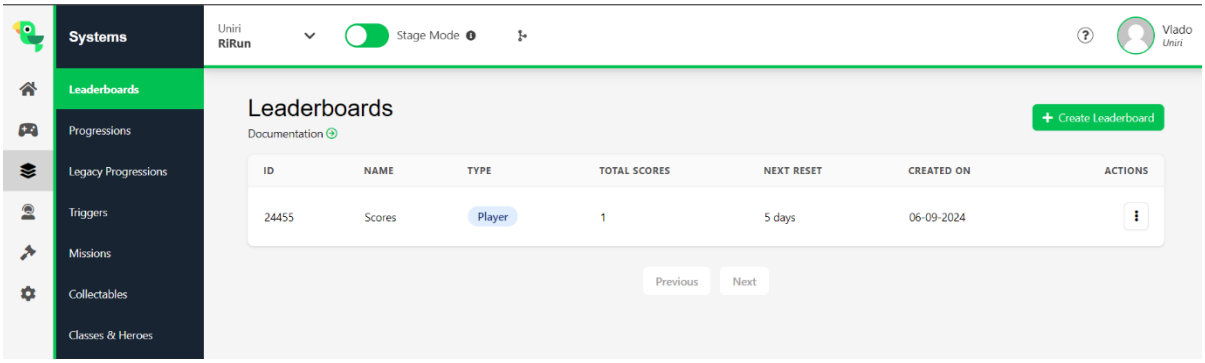
Slika 10. Postavke virtualne kamere

3.6. LootLocker SDK

Odlučili smo se za korištenje LootLocker-a SDK jer nismo htjeli izrađivati vlastitu ljestvicu poretka(engl. leaderboard). Ljestvicu poretka smo željeli implementirati unutar naše videoigre jer inače 3D *endless runner* videoigra bez ljestvice sa najboljim rezultatima nema smisla.

Instalaciju LootLocker SDK u Unity-u smo izvršili unutar Package Manager-a. Najjednostavniji način za instalirati LootLocker SDK je korištenjem Git-a, ali moramo se prije instalacije uvjeriti da imamo instaliran Git na računalu. Nakon što smo se uvjerili da imamo Git na računalu, odabrali smo „Add package from git URL.“ Upisali smo URL kojega smo pronašli u LootLocker dokumenatciji za Unity: https: “//github.com/LootLocker/unity-sdk.git.“ Nakon instaliranja paketa, nastavili smo pratiti jednostavni postupak konfiguracije LootLocker-a, sve što nam je trebalo su „Game API Key“ i „Domain Key“ kako bi povezali Unity projekt sa serverom LootLocker-a.

LootLocker nam je jednostavno kreirao ljestvicu na svome serveru i time nam je uštedio vrijeme i trud jer inače bi morali sami programirati ljestvicu poretka. Na Slici 11. možemo vidjeti sučelje LootLocker-a i ljestvicu poretka „Scores“ koju nam je kreirao.



Slika 11. LootLocker ljestvica

3.6.1. LootLocker programski kod

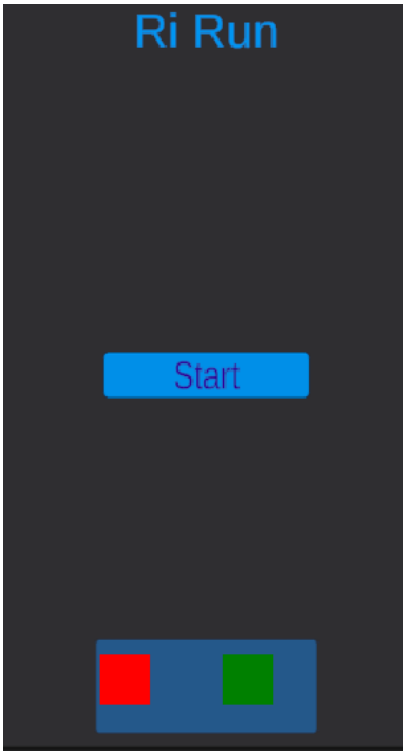
Kako bi povezali „Player-a“ sa serverom LootLocker-a kreirali smo skriptu „GameManager.cs“ koja izgleda ovako:

```
using System.Collections;
using UnityEngine;
using LootLocker.Requests;
using UnityEngine.Events;

public class GameManaager : MonoBehaviour
{
    [SerializeField] private UnityEvent playerConnected;
    private IEnumerator Start()
    {
        bool connected = false; ;
```

```
        LootLockerSDKManager.StartGuestSession((response) =>
        {
            if(!response.success)
            {
                Debug.Log("Error!");
                return;
            }
            Debug.Log("Success!");
            connected = true;
        });
        yield return new WaitUntil(() => connected);
        playerConnected.Invoke();

    }

}
```

Koristeći metodu „LootLockerSDKManager.StartGuestSession“ pokrenuli smo gostujuću sesiju što znači da igrač može igrati bez potrebe za registracijom računa. Nakon što je veza uspostavljena, poziva se UnityEvent `playerConnected.Invoke()` koji poziva sve metode koje su povezane na ovaj događaj.

Osim ljestvice poretka, došli smo na ideju da isprobamo ostale mogućnosti LootLocker-a. Kreirali smo novi „Context“ na serveru kako bi mogli kreirati dva *asseta* koji će predstavljati *skin* od „Playera.“ Igrač će moći odabrati *skin* na početnoj sceni „MainMenu-a“ što možemo vidjeti na Slici 12. Odabirom crvene ili zelene boje, promijeniti će se boja kacige na „Player-u.“ Važno za napomenuti je to da igrač mora imati određeni iznos bodova(engl. experience points, XP) kako bi odabrao crvenu ili zelenu boju. Logika odabira *skina* nalazi se u „PlayerSkinItem.cs“ skripti, a logika dohvaćanja igračevog inventara(engl. inventory) nalazi se u „Skin.cs“ skripti. Obje skripte mogu se naći pod prilozima ovoga rada.



Slika 12. Početni izbornik videoigre

Skripta „GameOver.cs“ obrađuje funkcionalnosti vezane uz završetak videoigre. Izgled sučelja završetka videoigre možemo vidjeti na Slici 13. Najbitniji dijelovi skripte:

1. Zaustavljanje igre, prikazivanje rezultata na zaslonu i dodavanja XP-a.

```
public void StopGame(int score)
 {
     this.score = score;
     scoreText.text = score.ToString();
     GetLeaderboard();
     AddXP(score);
}
```

2. Slanje igračevog rezultata na LootLocker ljestvicu. Pokreće se metoda `SubmitScoreToLeaderboard()`.
3. Dodavanje XP-a igraču pomoću LootLocker progresije. Metoda koja obavlja ovaj zadatak je `AddXP(int score)` koju ćemo obraditi naknadno.
4. Ponovno učitavanje videoigre, tj. ponovno pokretanje koristeći metodu `ReloadScene()`.

Slika 13. Sučelje završetka igre

Metoda za dodavanje bodova igraču:

```
    public void AddXP(int score)
    {
        LootLockerSDKManager.AddPointsToPlayerProgression(progressionKey,
amountOfPoints, response =>
        {
            if (!response.success)
            {
                Debug.Log("Failed ");
            }

            if (response.awarded_tiers.Any())
            {
                foreach (var awardedTier in response.awarded_tiers)
                {
                    Debug.Log($"Reached level {awardedTier.step}!");

                    foreach (var assetReward in awardedTier.rewards.asset_rewards)
                    {
                        Debug.Log($"Rewarded with an asset, id:
{assetReward.asset_id}!");
                    }

                    foreach (var progressionPointsReward in
awardedTier.rewards.progression_points_rewards)
                    {
                        Debug.Log($"Rewarded with {progressionPointsReward.amount}
bonus points in {progressionPointsReward.progression_name} progression!");
                    }

                    foreach (var progressionResetReward in
awardedTier.rewards.progression_reset_rewards)
                    {
                        Debug.Log($"Progression
{progressionResetReward.progression_name} has been reset as a reward!");
                    }
                }
            }
        });
    }
```

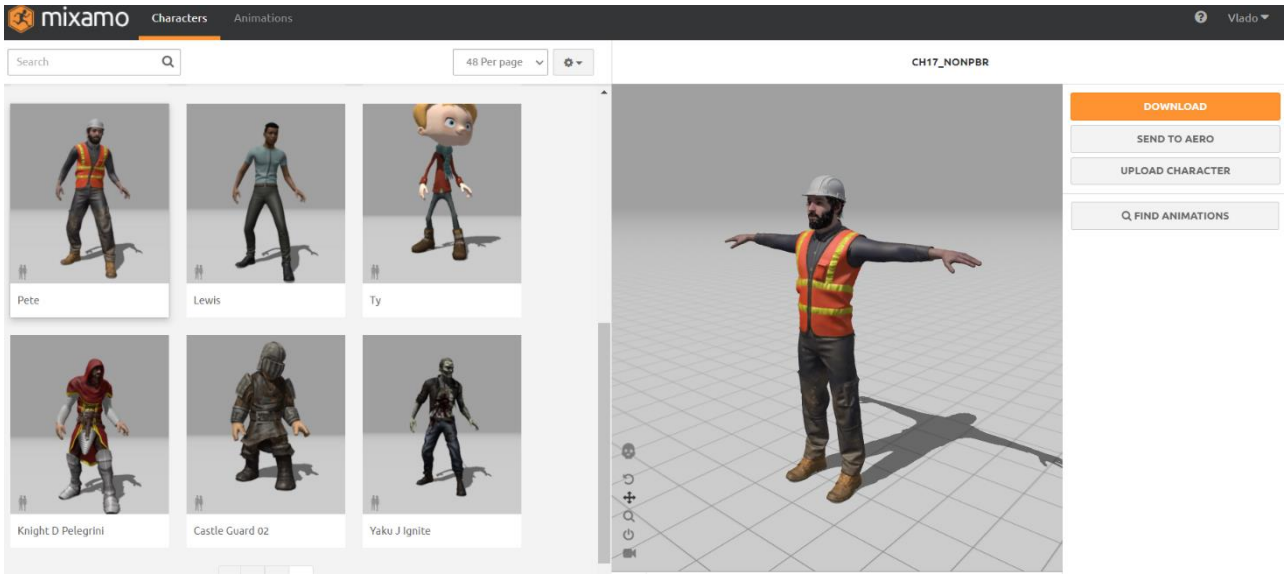
Osim za dodavanja bodova putem LootLocker-a, ova metoda služi i za dodjelu *skina*. „LootLockerSDKManager.AddPointsToPlayerProgression“ je metoda kojom dodajemo bodove igraču, a temelji se na ključu progresije `progressionKey` i unaprijed zadanoj količini bodova `amountOfPoints`. Ako igrač skupi dovoljan broj bodova za prelazak na viši nivo, `response.awarded_tiers` provjerava sadrži li podatke o nivoima te se ispisuje koji nivo je dostignut. Ako je na tom nivou, koji je igrač dosegnuo, predviđeno da igrač dobije nagradu, metoda prolazi kroz tri vrste nagrada:

- `asset reward`: nagrađuje sa *skinom*.
- `progression_points_rewards`: dodatni bodovi.
- `progression_reset_rewards`: resetiranje napretka.

Ukratko, ova metoda dodaje XP igraču, provjerava razinu i ispisuje nagrade.

3.7. Animacija lika

U izradi animacije objekta „Player“ korišten je Mixamo, koji nudi besplatne i već gotove 3D modele i animacije. Na Slici 14. možemo vidjeti sučelje Mixamo-a te 3D model lika kojega smo izabrali. Odabrali smo ovaj model lika jer se u Rijeci trenutno izvršava mnogo građevinskih radova pa smo se odlučili za model građevinara kako bismo se referirali na trenutno stanje u gradu. Preuzeli smo animacije trčanja, skakanja i klizanja za potrebe 3D *endless runner* igrice „Ri Run.“



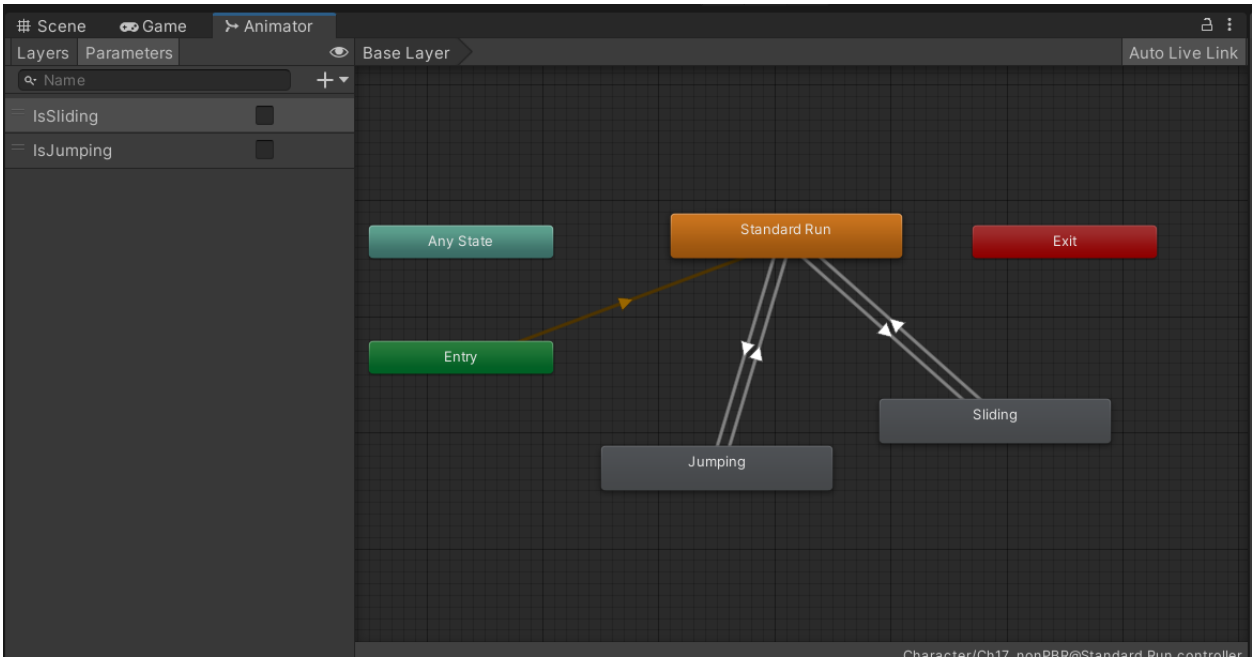
Slika 14. Sučelje Mixamo-a

3.7.1. Unity Muse – iskustvo korištenja

Prilikom traženja kako izraditi animaciju, naišli smo na „Unity Muse.“ To je alat koji koristi umjetnu inteligenciju za razvoj tekstura, materijala i ostalih elemenata igre, uključujući i animaciju. Pokušali smo da nam alat generira animaciju trčanja, klizanja i skakanja, ali nismo bili zadovoljni generiranim sadržajem. Pokušali smo i implementirati animacije u našu videoigru, ali nismo uspjeli postići željene rezultate zbog čega smo se na kraju odlučili na Mixamo. „Unity Muse“ je alat u razvoju, vjerujemo da ima potencijala postati koristan alat za kreiranje videoigara.

3.7.2. Implementacija animacija

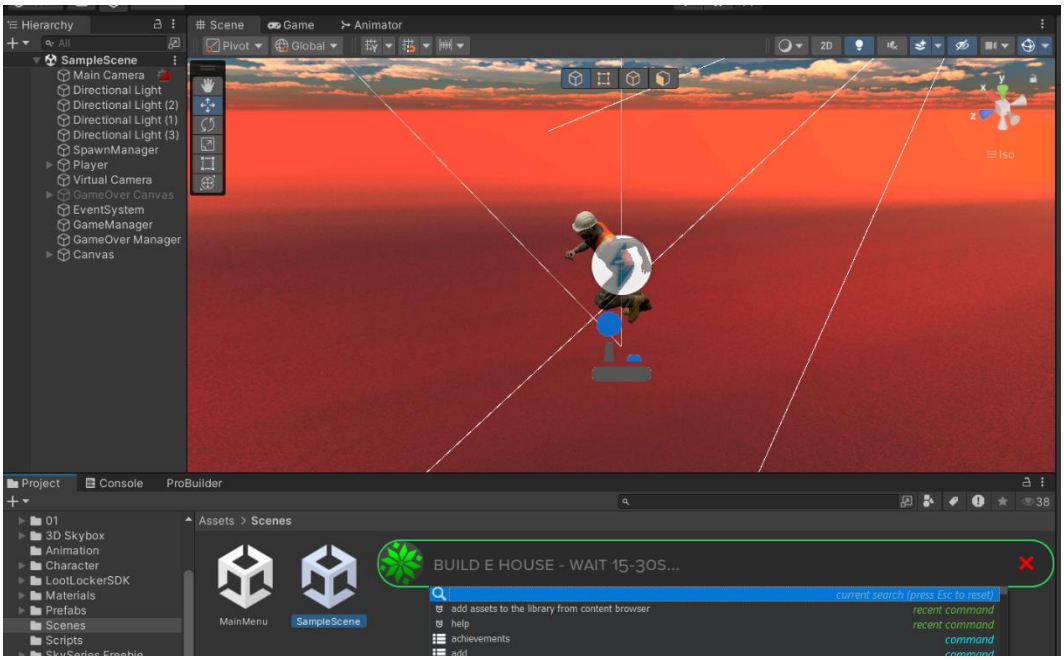
Nakon što smo preuzeli model lika i animacije, slijedi proces njihove implementacije u Unity. Prvo smo uvezli(engl. import) 3D model lika i animacije. Zatim smo povezali animacije unutar „Animator“ komponente što možemo vidjeti na Slici 15. Korištenjem „Animator Controller-a“ kreirali smo stanje(engl. state) za svaku od animacija. Stanja su povezana sa uvjetima *isSliding* i *isJumping*. Uvjeti omogućuju glatke prijelaze između stanja, primjerice kada igrač skoči. Parametri su kontrolirani iz ranije spomenute „PlayerController.cs“ skripte.



Slika 15. Animator

3.8. Promethean AI

Bez imalo radnog iskustva sa alatima umjetne inteligencije odlučili smo isprobati Promethean AI. Preuzeli smo alat koji uključuje Promethean AI i Promethean Browser. Povezali smo Promethean AI sa našim Unity projektom. Isprobali smo besplatnu verziju programa i ništa nam nije radilo kako smo očekivali pa smo uzeli mjesečnu pretplatu u nadi da će program raditi kako smo zamislili. Nažalost to nije bilo tako. Očekivanje da će alat stvarati nove modele iz

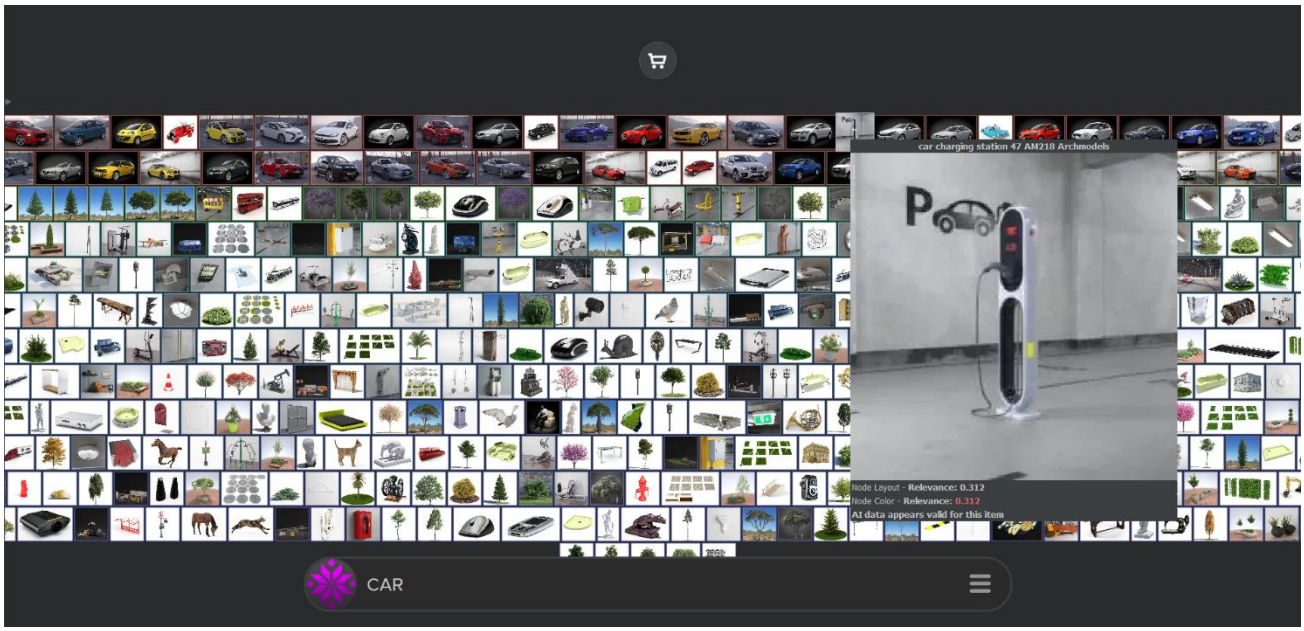


Slika 16. Promethean AI

našeg teksta ili slike koju mu priložimo nije se ostvarilo. Promethean AI je više namijenjen za profesionalne timove, koji već imaju veliku bazu sa resursima potrebnim za razvoj videoigre. Kada korisnik već ima bazu sa modelima, sve što treba je reći što želi, primjerice „Dodaj auto na scenu“ ili „Dodaj animaciju trčanja.“ Na Slici 16. možemo vidjeti naš pokušaj dodavanja kuće na scenu, ali pošto nemamo bazu sa resursima, alat nije dodao kuću na scenu.

3.8.1. Promethean AI Browser

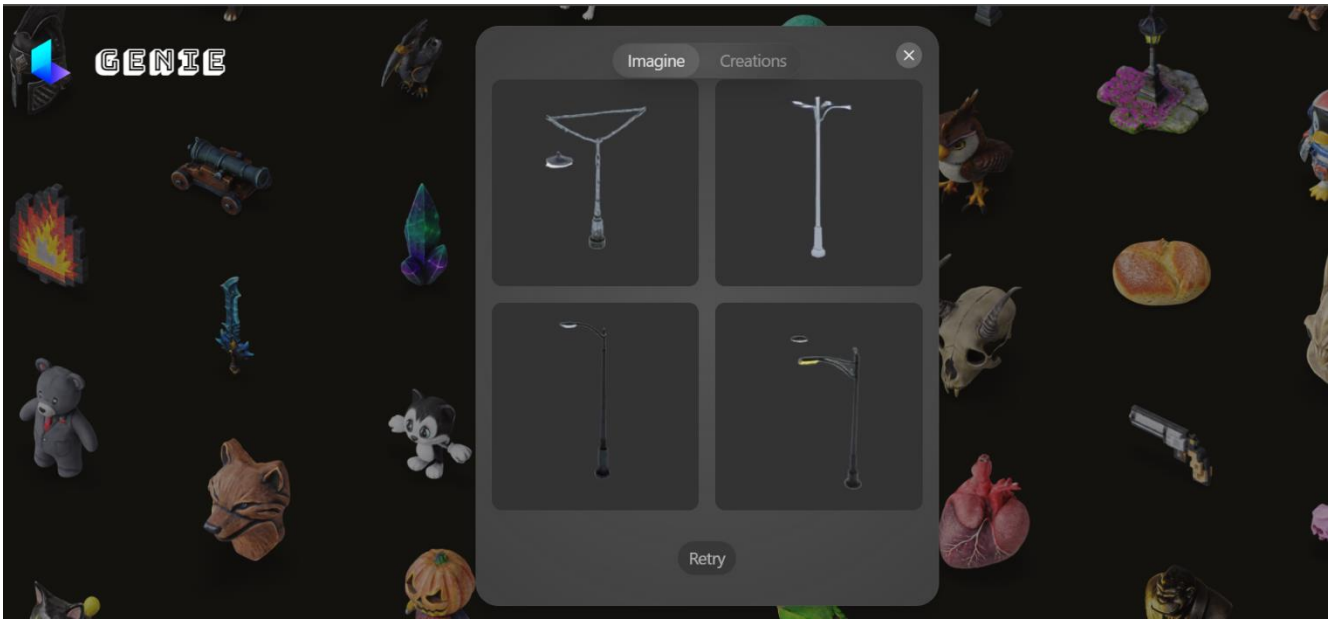
Promethean AI Browser radi na sličan način, samo što Promethean Browser služi za pregledavanje, organiziranje i upravljanje resursima koje korisnik već posjeduje. Probali smo koristiti i ovaj alat, većinom je bilo bezuspješno. Jedino za što smo uspjeli iskoristiti ovaj alat je pronalaženje gotovih 3D modela koje pronalazi na internetu. Na Slici 17. možemo vidjeti kako izgleda sučelje Promethean Browser-a, u kojemu smo tražili da nam nađe 3D modele auta. Uvezli smo par 3D modela na ovaj način, kako bismo mogli usporediti kvalitetu tih modela sa modelima koje je generirao Luman AI – Genie.



Slika 17. Promethean AI Browser

3.9. Luman AI – Genie

Modele smo na kraju kreirali koristeći 3D Luman AI – Genie. Ovaj alat je baziran na webu te ne zahtjeva preuzimanje alata, već samo prijavu korisnika. Alat stvara modele na osnovu tekstualnog opisa korisnika. Većina obrada korisnikovog opisa odvija se na serveru pa nije nužno imati snažno vlastito računalo. Osobno smo kreirali par 3D modela za videoigru kako bi se uvjerili je li alat radi ispravno. Na Slici 18. možemo vidjeti generirane 3D modele za naš tekstualni opis koji glasi „street light.“



Slika 18. Sučelje Luma AI - Genie-a

Odabrali smo treći 3D model jer nam se najviše svidio. Nakon odabira modela otvara nam se prozor kao na Slici 19. U izborniku u donjem desnom kutu na Slici 19. možemo podesiti:

- varijacija novih modela na osnovu već zadanog modela
- željeni format preuzimanja modela
- poboljšanje rezolucije zadanog modela

Preuzeli smo 3D model u .fbx formatu i uvezli smo ga u Unity.

Slika 19. Model generiran od strane Luma AI - Genie

3.10. Dizajn videoigre

U dizajnu igre koristili smo kombinaciju modela generiranih putem umjetne inteligencije i preuzetih modela sa interneta kako bi mogli napraviti usporedbu između generiranih i preuzetih modela. Kako smo već rekli, igrač se kreće po stazi koja se dinamički generira i postoje četiri vrste staza, ravno, lijevo, desno i raskrižje. Svaka od staza je spremljena kao *prefab* pa smo dodavanje 3D modela, na ove četiri vrste staza, dodavali u Unity Prefab Mod-u. Na Slici 20. možemo vidjeti dizajn staze desno. Korišteni su modeli građevina, auta, prepreka i ostali dekorativni modeli. Korištenje umjetne inteligencije omogućilo nam je bržu izradu 3D modela i jednostavnu implementaciju modela koji se uklapaju u gradski dizajn.

Slika 20. Izgled staze desno

3.10.1. Generirani modeli i preuzeti modeli

Generirani modeli pružaju veću fleksibilnost u prilagođavanju materijala, tekstura i oblika jer su kreirani prema određenim opisima korisnika. Omogućuju nam kreiranje specifičnih modela, koje bi teško pronašli na internetu. Za razliku od generiranih modela, preuzeti modeli su već unaprijed definirani, u smislu oblika, materijala i tekstura, nisu nužno prilagođeni za našu igru. Prilikom upotrebe ovih modela, jedino što smo mogli mijenjati na njima su veličina, pozicija i tekstura. Dakle, moramo pažljivo birati preuzete modele kako bi se uklopili u dizajn igre. Osim toga, primijetilo smo vidljivu razliku u kvaliteti rezolucija, između generiranih i preuzetih modela, bar u našem slučaju gdje smo za generiranje objekata koristili Luman AI – Genie. Kvalitetu rezolucije generiranog 3D modela možemo vidjeti na Slici 21., a kvalitetu rezolucije preuzetog 3D modela možemo vidjeti na Slici 22.

Slika 21. Generirani 3D model

Slika 22. Preuzeti 3D model

4. Zaključak

Cilj ovog završnog rada je ispunjen, obrađena je primjena umjetne inteligencije u procesu razvoja 3D *endless runner* igre „RiRun.“ Korištenjem AI alata istražena je mogućnost automatizacije kreiranja sadržaja te isto tako kako AI ubrzava i pojednostavljuje razvoj videoigre. Međutim, u našem slučaju proces nije bio potpuno automatiziran, što znači da smo neke dijelove u razvoju videoigre radili ručno. Da smo na raspolaganju imali vlastitu bazu sa 3D modelima, koristili bi Promethean AI, čime bi većina procesa bila automatizirana.

Umjetna inteligencija ima ogroman potencijal u razvoju videoigara, od kreiranja ideje i koncepta videoigre do kreiranja modela, tekstura i materijala. Usprkos potencijalima, umjetna inteligencija ima prostora za napredak jer trenutno još uvijek zahtijeva ljudsku prilagodbu i kreativnost, posebice kada je riječ o integraciji modela i prilagodbi mehanike igre.

U daljnjem razvoju trebalo bi istražiti dodatne AI alate koji pružaju veću automatizaciju i fleksibilnost. Također, bilo bi dobro izgraditi vlastitu bazu 3D modela kako bi se mogao iskoristiti puni potencijal Promethean AI-a.

Literatura

[1] Ranjan, Aditya, and Priya Chaturvedi. "Digital Technologies and the Intangible Cultural Heritage of the Rural Destination." Disruptive Innovation and Emerging Technologies for Business Excellence in the Service Sector.IGI Global, 2022. 196-218.

[2] Shannon, C. E. (1950). Programming a Computer for Playing Chess. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 41(314), 256–275.

[3] Perez, Javier Andreu, et al. "Artificial intelligence and robotics." arXiv preprint arXiv:1803.10813 147 (2018).

[4] Kühl, Niklas, et al. "Machine learning in artificial intelligence: Towards a common understanding." arXiv preprint arXiv:2004.04686 (2020).

[5] ChatGPT, https://en.wikipedia.org/wiki/ChatGPT (pristupljeno: 16.rujna 2024.).

[6] Unity Game Development Engine: A Technical Survey, https://www.researchgate.net/publication/348917348_Unity_Game_Development_Engine_A_Technical_Survey (pristupljeno: 16.rujna 2024.).

[7] Juliani, Arthur, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. "Unity: A general platform for intelligent agents." arXiv preprint arXiv:1809.02627 (2018).

[8] Unity game engine in visualization, simulation and modelling chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://trepo.tuni.fi/bitstream/handle/10024/133433/NieminenTopi.pdf?sequence=2 (pristupljeno: 20.rujna 2024.).

[9] Explore the Unity Editor, https://learn.unity.com/tutorial/explore-the-unity-editor-1# (pristupljeno: 20.rujna 2024.).

[10] Unity User Manual 2022.3 (LTS), https://docs.unity3d.com/Manual/index.html (pristupljeno: 20.rujna 2024.).

[11] Luma Genie 1.0, https://www.luma-ai.com/luma-genie-1-0/ (pristupljeno: 21.rujna 2024.).

[12] What is LootLocker?, https://docs.lootlocker.com/the-basics/what-is-lootlocker (pristupljeno: 21.rujna 2024.).

Popis slika