Studija izvedivosti ozbiljne igre na temu zagađenja oceana

Tehnička dokumentacija

Verzija 1.0

Studentski tim:

Marko Bakić

Marija Dragošević

Matea Gluhak

Iva Ištuk

Maja Jurić

Nastavnici i suradnici:

izv. prof. dr. sc. Jurica Babić

Ana Radović

Sadržaj

Uvod 2
1. Play2Green 3
2. Onečišćenje mora 4
3. Zelene ozbiljne igre 5
4. Opis studijskog slučaja 6
4.1. Prototip programskog rješenja Aqua Saviours 6
4.2. Definicija problema 10
4.3. Unapređenje edukacijskog sadržaja postojećeg prototipa 10
5. Model rješenja 13
5.1. Očekivani rezultati studije izvedivosti 13
5.2. Zadaci 13
6. Implementacija programskog rješenja 15
6.1. Implementacija umjetne inteligencije u Unity razvojnom okruženju 15
6.2. Implementacija proširene stvarnosti u Unity razvojnom okruženju 22
6.3. Implementacija sustava za prikupljanja korisničkih podataka za evaluaciju učinka 27
6.4. Implementacija sustava dijaloga u Unity razvojnom okruženju 33
6.5. Integracija Unity projekta u Flutter aplikaciju 41
7. Evaluacija rezultata studije izvedivosti 49
7.1. Evaluacija prototipa igre Aqua Saviours 49
7.2. Evaluacija implementacije 49
7.3. Ocjena izvedivosti i smjernice za budući rad 55
Zaključak 57
Literatura 58

Uvod

Zelene ozbiljne igre predstavljaju inovativan pristup edukaciji kombinirajući zabavu s edukativnim sadržajem kako bi potaknule svijest o ekološkim izazovima. *Play2Green* je ERASMUS+ projekt koji radi na razvoju zelenih ozbiljnih igara koje se temelje na novim tehnologijama. Play2Green je 2023. godine organizirao hackaton koji se održao u Dubrovniku. Cilj tog hackatona bio je kroz suradnju potaknuti razvoj inovativnih rješenja usmjerenih prema održivosti i zaštiti okoliša. Nagradu za najbolji Figma prototip zelene ozbiljne igre osvojio je tim *Ocean Rescue* s igrom *Aqua Saviours* koja educira korisnike o onečišćenju oceana.

Cilj ovog projekta bio je provesti analizu izvedivosti ključnih dijelova osmišljene ozbiljne igre *Aqua Saviours* za čiji razvoj bi se koristila platforma *Unity.* Jedan od značajnih aspekata analize obuhvaća istraživanje mogućnosti implementacije novih tehnologija, poput umjetne inteligencije i proširene stvarnosti, u *Unity*ju (npr. kako prikazati životinje u proširenoj stvarnosti te kako njihovo kretanje i ponašanje izvesti pomoću umjetne inteligencije). Analiza izvedivosti uključuje i proširenje edukacijskog sadržaja unutar same ozbiljne igre te evaluaciju savladanog znanja putem kvizova i/ili podataka unutar igre. Nadalje, u okviru ove studije izvedivosti, istražena je i mogućnost integracije igara u radni okvir *Flutter*.

Ovaj dokument je rezultat studije izvedivosti za ozbiljnu igru *Aqua Saviours*. U nastavku je prvo predstavljen projekt *Play2Green* te je detaljnije objašnjen koncept zelenih ozbiljnih igara. Zatim je opisan model programskog rješenja koji je osmišljen na hackathonu u Dubrovniku. Također su predloženi planovi za buduće unaprijeđenje. Na kraju, dokumentirani su rezultati studije izvedivosti. Studija izvedivosti provedena je kao 5 odvojenih istraživanja: razvijanje umjetne inteligencija i razvijanje proširene stvarnosti u okruženju za razvoj igara *Unity*, integracija *Unity*ja s radnim okvirom *Flutter*, proširenje edukativnog sadržaja te prikupljanje podataka unutar igre.

1. Play2Green

Play2Green[1] je inovativni projekt u okviru programa ERASMUS+ koji se fokusira na integraciju ozbiljnih igara u "zeleno obrazovanje". Cilj *Play2Green* projekta je podizanje svijesti o okolišu i borbi protiv klimatskih promjena. Taj cilj se pokušava ostvariti tako da se radi na povećanju broja studenata i nastavnika visokog obrazovanja koji su u okviru studija obrađivali zelene teme te tako da se svima omogući pristup resursima za edukaciju o zelenim temama.

Play2Green razvija ozbiljne zelene igre koje koriste napredne tehnologije poput umjetne inteligencije (AI), proširene stvarnosti (AR) i holografske tehnologije tako pružajući studentima i predavačima visokoobrazovnih institucija nove, uzbudljive pristupe edukaciji.

Projekt kombinira fizička i virtualna iskustva učenja, a provodi ih konzorcij različitih institucija visokog obrazovanja i nevladine organizacije. Kroz suradnju s različitim organizacijama i stručnjacima, *Play2Green* potiče razvoj kreativnih i interaktivnih rješenja koja će privući pažnju i educirati širu javnost o problemima okoliša.

Kako bi potaknuo razvoj kreativnih i interaktivnih rješenja, *Play2Green* je 2023. godine organizirao hackaton na kojem se kroz suradnju radilo na osmišljavanju zelene ozbiljne igre koja koristi nove tehnologije.

Projekt *Play2Green* je inspiriran Gretom Thunberg i drugim mladim ekološkim aktivistima diljem Europe te crpi motivaciju za stvaranje platforme koja će podići svijest o ekološkim izazovima među svim studentima sveučilišta, a i šire.

Slika 1. Play2Green logo

[1] Play2Green (Serious Gaming for Universal Access to Green Education) [2022-1-HR01-KA220-HED-000088675] - https://sociallab.fer.hr/play2green

2. Onečišćenje mora

Onečišćenje oceana ozbiljan je ekološki problem koji utječe na ekosustave i morski život [1] . Opisuje prisutnost ili ulazak opasnih materijala, plastike i drugih zagađivača u ocean koji time uzrokuju onečišćenje mora, smanjenje populacije riba i uništavanje staništa. Ova tema relevantna je danas radi sve većeg negativnog utjecaja na okoliš i ljudsku populaciju čiji je život ovisan o resursima iz oceana kao što su hrana, posao, aktivnosti u slobodno vrijeme i slično [2] . Potražnja za resursima raste zajedno s ljudskom populacijom, što je rezultiralo povećanjem otpada, zagađenja i neodrživog ponašanja [3] . Plastični otpad, kanalizacija i drugi zagađivači bacaju se u velikim količinama, posebno u oceane, štetеći ekosustavima i morskom životu na načine koji se ne mogu popraviti [4] .Klimatske promjene, koje uzrokuju kiselost oceana, povećanje razine mora i druge promjene okoliša koje ugrožavaju život u moru, još su jedan faktor koji pogoršava problem. Onečišćenje oceana ima ozbiljne ekonomske i društvene posljedice uz svoje negativne ekološke učinke. Pogođeni su milijuni ljudi koji ovise o ribolovu, turizmu i drugim industrijama povezanim s oceanima [5] . Štoviše, može rezultirati nakupljanjem onečišćujućih tvari u plodovima mora i drugim morskim proizvodima, što predstavlja rizik za ljudsko zdravlje. S obzirom na ozbiljnost i hitnost problema, od velike je važnosti poduzeti mjere za zaustavljanje i smanjenje onečišćenja oceana. To se može učiniti na više načina npr. smanjenjem upotrebe plastike, održivim ribolovom, promicanjem ekološki prihvatljivih putovanja i financiranjem istraživanja i razvoja za stvaranje novih rješenja za smanjenje onečišćenja okoliša. Zaštita morskog života, očuvanje javnog zdravlja i osiguranje održive budućnosti za Zemlju i njezine stanovnike mogući su rješavanjem problema onečišćenja oceana. Postoji mnogo načina na koje se ljudi mogu osvijestiti o zagađenju oceana, uključujući kampanje za podizanje svijesti javnosti, edukativne materijale, sudjelovanje u akcijama čišćenja obala te aktivizam i zagovaranje političkih promjena koje bi mogle smanjiti onečišćenje. Međutim, jedan od inovativnijih načina za osvještavanje ljudi o problemu onečišćenja oceana je ozbiljna edukativna igra.

3. Zelene ozbiljne igre

Ozbiljne igre predstavljaju skupinu igara čiji primarni cilj nije isključivo zabava nego učenje i unaprjeđivanje vještina. Koriste se u raznim područjima, a iako im zabava nije glavni cilj, ne znači da te igre ne bi trebale biti zabavne jer time korisnika potiču da lakše uroni u tematiku koju one pokrivaju [6] .

Ozbiljnim igrama možemo simulirati stvarne ozbiljne tematike, a kako se posljednjih godina poseban naglasak stavlja na zelene teme onda nastaju tzv. zelene ozbiljne igre. Ove ozbiljne igre za cilj imaju edukaciju korisnika o klimatskim promjenama, očuvanju prirode, obnovljivim izvorima energije, održivom razvoju i sličnim temama ključnim za budućnost čovječanstva [7] . Kroz interaktivno iskustvo igre korisnici se suočavaju sa stvarnim izazovima koji nastaju zbog narušavanja eko-sustava. Mogu biti moćan alat za obrazovanje o okolišu jer omogućuju igračima da vide posljedice svojih postupaka u simuliranom okruženju, čineći iskustvo učenja opipljivijim i značajnijim.

Inicijativom sve većeg broja timova i institucija diljem svijeta nastaje značajan broj prototipova zelenih ozbiljnih igara, no i dalje postoje izazovi koje treba savladati. Dizajn i učinkovitost ovih igara ovise o multidisciplinarnoj suradnji i dok je potencijal neosporiv i dalje je potrebno više istraživanja kako bi se optimirao njihov dizajn te procijenio i poboljšao njihov učinak [8] .

4. Opis studijskog slučaja

U sklopu Erasmus+ projekta održan je *Play2Green Hackathon* (17. do 23. rujna 2023.) na kojem su sudjelovali nastavnici i studenti iz 5 visokoobrazovnih institucija iz Hrvatske, Španjolske, Mađarske i Francuske. Hackaton je bio organiziran kao kombinirana mobilnost (virtualna i fizička), a sudionici su bili organizirani u međunarodne timove. Timovi su radili na dizajnu ozbiljnih igara temeljenih na novim tehnologijama koje bi trebale skrenuti pozornost na onečišćenje oceana, zaštitu životinja i srodne zelene teme.

Nagradu za najbolji *Figma* prototip osvojio je tim *Ocean Rescue* sa svojom ozbiljnom igrom *Aqua Saviours*[2].

4.1. Prototip programskog rješenja Aqua Saviours

Aqua Saviours je ozbiljna igra temeljena na mini igrama koja pokušava iskoristiti nove tehnologije, poput proširene stvarnosti (engl. *Augmented Reality, AR*) i umjetne inteligencije (engl. *Artificial Intelligence*, *AI*), za poboljšano iskustvo igranja igre. Sam edukacijski sadržaj zasnovan je na činjenicama koje bi se korisnicima prikazivale interakcijom s likovima u igri.

Korisniku u igri je učenik koji se obrazuje kako bi postao *Aqua Saviour*. Kroz igru susreće razne likove koji ga obrazuju o onečišćenju mora i s time povezanim problemima te mu zadaju izazove kako bi prešao u više razine svog obrazovanja. Na Slika 2 prikazan je početni ekran prototipa igre *Aqua Saviours*. Svaka ikona predstavlja jednu mini-igru, odnosno jedan izazov. Kroz sve mini-igre skupljaju se bodovi i osvajaju nagrade.

[2] Figma prototip igre *Aqua Saviours*: https://www.figma.com/proto/HfD7OjU9Taeaozfkn1pMvE/OceanRescue_P2G?type=design&node-id=317-374&t=Z21LWXcd3PsmcVrE-1&scaling=min-zoom&page-id=56%3A21&starting-point-node-id=338%3A144&show-proto-sidebar=1&mode=design

Slika 2 Početni ekran prototipa igre Aqua Saviours

Ponuđene mini igre su:

1. *Guardians of the Sea*

   U ovoj mini-igri, kako je opisano uputama na zaslonu, potrebno je ukloniti otpad koji se nalazi na životinjama oko korisnika. Igra koristi proširenu stvarnost kako bi prikazala 3d modele životinja u okruženju korisnika te umjetnu inteligenciju za kretanje životinja u prostoru, prikazano slikama 3 i 4.

Slika 3 Početni ekran mini-igre Guardians of the Sea

Slika 4 Guardians of the Sea mini-igra

2. *Trash Ninja*

Trash Ninja mini-igra inspirirana je igrom *Fruit Ninja*[3]. Cilj je pokretima koji nalikuju rezanju otpada ukloniti sav otpad prije nego padne u more (Slika 6). Igra koristi proširenu stvarnost za prikazivanje otpada u prostoru oko korisnika (Slika 5).

Slika 6 Početni ekran mini-igre Trash Ninja

Slika 5 Trash Ninja mini-igra

3. *Deep Abyss Quiz*

Deep Abyss mini-igra je kviz u kojem treba odgovoriti na pitanja o onečišćenju mora i zaštiti morskog života (Slika 7 i Slika 8).

[3] *Fruit Ninja* igra: https://play.google.com/store/apps/details?id=com.halfbrick.fruitninjafree&hl=en_US

Slika 7 Početni ekran igre Deep-Abyss

Slika 8 Deep-Abyss mini-igra

Igrači imaju priliku osvojiti 3d modele predmeta koje mogu prikazati kroz proširenu stvarnost što se može vidjeti na Slika 9.

Slika 9 Prikaz nagrade

4.2. Definicija problema

Ova studija izvedivosti fokusira se na istraživanje i ispitivanje mogućnosti implementacije igre *Aqua Saviours* koristeći platformu *Unity*. To uključuje analizu podrške *Unityja* za nove tehnologije (proširena stvarnost i umjetna inteligencija), razradu edukacijskog sadržaja i njegovog prijenosa korisniku te analizu pristupa evaluaciji usvajanja gradiva. Studija će također proučiti integraciju *Unity* igara s *Flutter* platformom kako bi dobila cjelovit uvid u tehničku izvedivost ove kombinacije tehnologija.

Tijekom studije izvedivosti ispituju se ključni dijelovi predloženog programskog rješenja:

- Implementacija proširene stvarnosti u *Unityju*
- Implementacija umjetne inteligencije u *Unityju*
- Evaluacija i razrada edukacijskog sadržaja
- Evaluacija usvajanja gradiva
- Integracija platformi *Flutter* i *Unity*

4.3. Unapređenje edukacijskog sadržaja postojećeg prototipa

Edukacijski sadržaj ozbiljne igre *Aqua Saviours* prvenstveno se fokusira na očuvanje oceana, pokrivajući teme kao što su onečišćenje oceana, morski ekosustavi, klimatske promjene i ekološke tehnologije. Cilj igre je educirati igrače o važnosti čišćenja oceana, međusobnoj povezanosti morskih vrsta i učincima onečišćenja. Iako *Aqua Saviours* već sadrži edukacijski sadržaj, postoji prostor za njegovo unapređenje kako bi igra postala učinkovitija u podučavanju o zaštiti mora. Za poboljšanje edukacijskog sadržaja igre predložena su sljedeća rješenja:

- **Interaktivne metode učenja**: Integracija više interaktivnih elemenata poput kvizova i zagonetki izravno povezanih s obrazovnim sadržajem. Na taj način, igrači bi se aktivnije uključili u proces učenja. Također, dijalozi između likova u igri *Aqua Saviours* predstavljaju odličan način za edukaciju. Oni omogućavaju interaktivno učenje i dublje razumijevanje ekoloških problema kroz zanimljive i relevantne razgovore. Dijalozi mogu poslužiti kao sredstvo za prenošenje složenih informacija na pristupačan i razumljiv način, doprinoseći tako većoj uključenosti igrača i efikasnijem očuvanju stečenog znanja.

- **Uključivanje studija slučajeva iz stvarnog svijeta**: Uključivanje primjera ekoloških problema iz stvarnog svijeta i njihovih rješenja igračima bi dočaralo praktično razumijevanje problematike onečišćenja mora.
- **Suradnja sa stručnjacima**: Surađivanje s morskim biolozima ili ekolozima osiguralo bi točnost informacija o morskom životu i strategijama njegovog očuvanja.
- **Korištenje AR tehnologije**: Korištenje AR tehnologije, moguće je stvoriti iluziju da su virtualni morski ekosustavi dio naše stvarne okoline, što igračima omogućava da jasnije dožive posljedice onečišćenja i prepoznaju važnost zaštite mora.
- **Pristup obrazovnim resursima**: Omogućavanje pristupa dodatnim obrazovnim resursima, poput knjiga ili članaka o očuvanju oceana, za zainteresirane igrače.
- **Različiti ekosustavi**: Uvođenjem šire raznolikosti morskih ekosustava proširio bi se obrazovni opseg igre.
- **Višejezična podrška**: Uvođenjem podrške igre za više jezika dosegla bi se šira publika i poboljšao bi se obrazovni utjecaj igre na globalnoj razini.
- **Povratne informacije i prilagodba**: Implementacijom sustava za povratne informacije igrača o obrazovnom sadržaju i kontinuiranim prilagođavanjem igre na temelju dobivenih informacija, osiguralo bi se da obrazovni materijal ostane relevantan i privlačan.

Dijalozi u igri Aqua Saviours mogu biti osobito korisni u edukativne svrhe, pružajući platformu za interaktivno učenje koje se može prilagoditi različitim dobnim skupinama. Na primjer, za mlađu publiku, dijalozi o onečišćenju oceana mogu biti pojednostavljeni s likovima i jezikom koji odgovaraju njihovoj razini razumijevanja, dok bi za starije igrače isti koncepti mogli biti prezentirani kroz složenije dijaloge s detaljnijim podacima i izazovnijim pitanjima. Također, upotreba prilagodljivih dijaloga omogućava prikaz različitih perspektiva problematike, te potiče kritičko razmišljanje i dublje razumijevanje teme.
Na donjoj tablici vidljiv je primjer prilagodbe edukacijskog sadržaja o onečišćenju mora za različite dobne skupine.

Dobna skupina	Edukacijski sadržaj
Djeca predškolske dobi	More je puno malih ribica i lijepih stvorenja. Ali kad ljudi ostave smeće kao što su plastične boce u vodi, to može naštetiti ribicama i učiniti more tužnim. Zato uvijek bacamo smeće u kantu i recikliramo!
Djeca školske dobi (7-12 godina)	Znate li da kad plastika završi u oceanu, može ozlijediti kornjače i dupine? Plastika se pretvara u mikroplastiku koju ribe mogu pojesti. Ako smanjimo plastiku koju koristimo, možemo pomoći da naš ocean ostane zdrav i čist!
Adolescenti (13-18 godina)	Ocean je dom mnogim životinjama, ali zagađenje, poput plastike, ugrožava njihov dom. Zagađenje može dovesti do toga da ribe i druge morske životinje pojedu štetne materijale. Razmislite o akcijama poput smanjenja plastike i sudjelovanja u čišćenju plaža kako bismo zaštitili oceane.
Odrasli (18+ godina)	Zagađenje oceana predstavlja kompleksan problem s dugotrajnim posljedicama za naš ekosustav i zdravlje. Plastični otpad se razgrađuje na mikroplastiku koju konzumiraju morski organizmi, utječući na prehrambeni lanac sve do nas. Aktivno sudjelovanje u očuvanju oceana, bilo kroz informiranje o održivim praksama, podršku ekološkim inicijativama ili pritisak na političke čimbenike za promjene, ključno je za zaštitu naše planete.

5. Model rješenja

U ovom poglavlju definiraju se očekivani rezultati studije izvedivosti s obzirom na probleme definirane u prethodnom poglavlju te specifični koraci koji su potrebni za njezino izvršavanje.

5.1. Očekivani rezultati studije izvedivosti

Očekivani rezultati provedbe studije izvedivosti su:

1. Pregled i evaluacija ključnih elemenata relevantnih za uspješnu implementaciju igre
2. Elementi ozbiljne igre *Aqua Saviours* koji će demonstrirati rješenje gore navedenih problema

5.2. Zadaci

- **Implementacija AR interaktivnih životinja:**
    - Istraživanje i odabir odgovarajuće tehnologije za implementaciju proširene stvarnosti unutar igre
    - Evaluacija i dorada prototipa
    - Razvoj modela životinja i njihovih animacija
    - Implementacija interaktivnog prototipa za AR mini igre u Unity-ju
- **Implementacija umjetne inteligencije za ponašanje životinja:**
    - Istraživanje postojećih pristupa implementaciji umjetne inteligencije u video igrama
    - Razvoj prototipa ponašanja životinja unutar igre pomoću odabrane AI tehnike ili skripte
    - Testiranje i optimizacija AI modela kako bi se postiglo realistično ponašanje životinja
- **Evaluacija usvajanja gradiva:**
    - Analiza trenutnog pristupa evaluaciji ishoda učenja unutar igre
    - Razvoj metoda za ocjenjivanje učinkovitosti edukativnih elemenata
    - Prikupljanje podataka o usvojenom znanju korisnika tijekom igranja
    - Implementacija kviza i prikupljanja podataka tijekom igranja u Unity-ju
- **Evaluacija i razgradnja edukacijskog sadržaja:**
    - Evaluacija trenutnog edukativnog sadržaja unutar igre

- Razvoj novih informacija o onečišćenju oceana prilagođenih različitim dobnim skupinama i razinama znanja
- Vizualna optimizacija i poboljšanje prezentacije edukativnog materijala unutar igre radi bolje angažiranosti korisnika

- **Integracija Unity projekta u Flutter aplikaciju:**
  - Istraživanje mogućnosti integracije Unity projekta unutar Flutter aplikacije
  - Razvoj Flutter aplikacije s integracijom Unity projekta mini igara
  - Implementacija obostrane komunikacije između Flutter-a aplikacije i Unity projekta
  - Testiranje integracije kako bi se osigurala stabilnost i učinkovitost

6. Implementacija programskog rješenja

U ovom poglavlju opisana je implementacija elemenata ozbiljne igre koji demonstriraju rješenja problema navedenih u potpoglavlju 5.2. Zadaci.

6.1. Implementacija umjetne inteligencije u Unity razvojnom okruženju

Ovo poglavlje donosi pregled dvaju alata *Emerald AI 3.0* i *ML-Agents* koji se mogu koristiti za implementaciju AI logike u *Unity* platformi. Korištenje je demonstrirano na jednostavnim scenama koje prikazuju AI ponašanje morskih životinja.

6.1.2. Umjetna inteligencija u videoigrama

AI u videoigrama odnosi se na upotrebu tehnika umjetne inteligencije kako bi se kreiralo inteligentno ponašanje likova i neigrivih likova (engl. *Non-playable characters, NPC*) kao npr. donošenje odluka te reagiranje na okruženje igre. Samim igračima koji koriste videoigru pruža privlačno, personalizirano, responzivno i dinamično iskustvo igranja. Neki primjeri su: bolje protivničko ponašanje koje pružaju igraču izazov, prilagođavanje težine igre, preporučivanje sadržaja, generiranje dinamičkog raznolikog sadržaja u okolišu igre, optimizacija performansi igre prilagođavanjem specifičnostima platformi i hardvera, automatizacija testiranja igre, stvaranje realističnih karaktera, poboljšanje grafičkih elemenata i analiza igračkog ponašanja, preferenci i obrazaca.

6.1.2. Usporedba *Emerald AI 3.0.* i *ML-Agents*

Emerald AI 3.0. je alat koji omogućava razvijateljima da brzo i jednostavno kreiraju ponašanja zasnovana na umjetnoj inteligenciji za bilo koje likove unutar igre, bez kodiranja, obzirom da se kreiranje i postavljanje samog lika odvija putem *Emerald* korisničkog sučelja. Sučelje omogućava visok stupanj prilagodljivosti AI ponašanja te pruža upute za postavljanje. Primjeri su predstavljeni u demo scenama koje prikazuju razne situacije kao npr. borba dva AI lika, napad AI lika prema meti koja nema implementirano AI ponašanje, sakupljanje hrane, kretanje na unaprijed zadanu destinaciju i slično. Dokumentacija[4] je opširno i jasno napisana te sadrži sve potrebne upute za instalaciju i korištenje.

[4] Emerald AI 3.0. dokumentacija: https://github.com/Black-Horizon-Studios/Emerald- AI/wiki 3.0. dokumentacija

ML-Agents je *open-source* alat koji omogućava igricama i simulacijama da postanu okruženje za treniranje inteligentnih agenata. Pruža implementaciju *state-of-the-art* algoritama koji omogućavaju razvijateljima jednostavno treniranje inteligentnih agenata za 2D i 3D igre te za igre koje koriste proširenu ili virtualnu stvarnost. Unutar alata najčešće se koristi podržano učenje koje se oslanja na metodu pokušaja i pogreške gdje agent za poželjno ponašanje prima nagradu, dok za nepoželjno prima kaznu. Trenirani agenti mogu se koristiti za kontroliranje ponašanja neigrivih likova, testiranje igrice, ali i razne druge svrhe. *ML-Agents* dodatak je koristan za razvijatelje, ali i AI istraživače koji mogu primjenjivati i kasnije evaluirati u bogatom Unity ekosustavu. Dokumentacija[5] pruža jasne upute za instaliranje, korišenje te demo scene koje prezentiraju mogućnosti alata.

Tablica u nastavku prikazuje uspredbu *Emerald AI 3.0* i *ML-Agents* u nekoliko ključnih apekata kao i subjektivan dojam prilikom korištenja.

Aspekt	**Emerald AI 3.0**	**ML-Agents**
Instalirana verzija	*Emerald AI 3.0*	*Release 20*
Plaćanje	Da, ali je kupljen za potrebe Play2Green projekta.	Besplatan tj. open-source.
Performanse	Alat se dobro ponaša za definirane zadatke.	Alat se dobro ponaša za definirane zadatke.
Integracija u Unity	Dodavanje paketa putem *Package Managera* nakon kupovine u *Asset Storeu* jednostavno i brzo.	Instalacija se provodi kloniranjem projekta s GitHub-a[6]. Potrebno je imati instaliran Python, a poželjno Anacondu za različita okruženja. Instalacija je duga, nailazi se na razne probleme s npr. usklađivanjem verzija,

[5] ML-Agents dokumentacija: https://unity-technologies.github.io/ml-agents/
[6] ML-Agents release 20: https://github.com/Unity-Technologies/ml-agents/releases/tag/release_20

		ali koji su rješivi uz korisne tutoriale[7] i dokumentaciju.Instalacija
Dokumentacija	Opširna, ali neusklađena s trenutnom verzijom (drugačiji izgled sučelja)	Opširna i jasna.
Grafičko korisničko sučelje	Komplicirano i neintuitivno iako postoje napomene i objašnjenja u njemu.	Ne posjeduje klasičan GUI, ali posjeduje jednostavan i intuitivan prikaz *Behaviour Parameters*. Za treniranje modela se koristi terminal.
Korišteni algoritmi	Ne navode.	*Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), Deep Deterministic Policy Gradients (DDPG), Trust Region Policy Optimization (TRPO), SAC-IQ (Implicit Quantile Network).*
Korišteni modeli	Ne navode.	Neuronske mreže koje se koriste za podržano učenje: *Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM)*, vlastite prilagođene arhitekture neuronskih mreža.
Korištenje resursa	Nema treniranja pa nema ni	CPU i GPU prilikom treniranja

[7] Tutorial za instalaciju: https://www.youtube.com/watch?v=Dj-BsYtANE0

	nekih posebnih zahtjeva za korištenjem resursa koji su izvan okvira Unity-ja.	modela, dodatni zahtjevi za memorijom.
Brzina razvoja	Brzi razvoj – odabiranje značajki putem sučelja, nema kodiranja ni debugiranja.	Sporiji razvoj – treniranje modela, pisanje potrebnih skripti, debugiranje koda.
Prednosti alata	Predefinirana ponašanja, jednostavno korištenje, brza instalacija u Unity.	Prilagodljivost, široka primjena, mogućnost samostanog razvoja, kreativnost, jednostavna primjena za predložena ponašanja (demo scene)
Složenost Korištenja	Nakon upoznavanja s dokumentacijom jednostavno, iako se mogu javiti poteškoće ako korisnik nema iskustva s Unity-jem jer se uvelike oslanja na njegovo sučelje i paradigmu korištenja.	Zahtjevnije korištenje, potrebno je trenirati modele i pisati skripte. Potrebno je razumijevanje strojnog i dubokog učenja. Instalacija nije trivijalna.
Ograničenja Alata	Ponašanja predefinirana, oslanja se na animacije modela što mnogi besplatni nemaju te je poželjno koristiti one koji imaju, a to košta.	Potrebno iskustvo i razumijevanje dubokog učenja za optimalne rezultate.

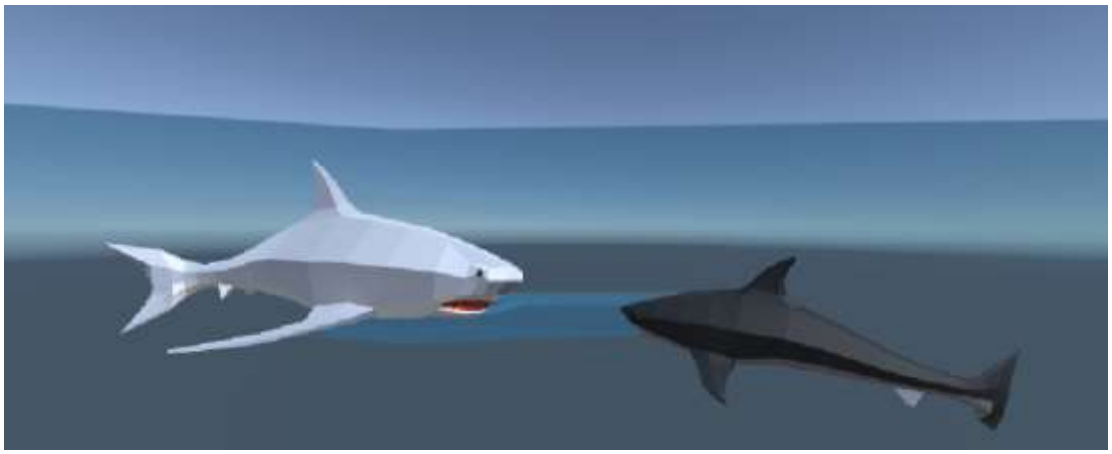
6.1.3. Opis scene s primjenom *Emerald AI 3.0* alata

Slika 10 prikazuje jednostavnu scenu koja je inspirirana predloškom iz *Emerald* dokumentacije. AI vs AI je primjer dva AI lika koja se međusobno bore. Jednostavna scena prikazuje bijelog morskog psa koji se bori protiv tamnog morskog psa. Za početak, kako bi AI

likovi mogli navigirati scenom, potrebno je imati Nav Mesh system baked[8] na ravninu po kojoj se likovi kreću kako bi se izbjegli *errori* u Unity konzoli.

Nakon toga slijedi kreiranje i postavljanje AI lika. Potrebno je slijediti više koraka za uspješno stvaranje AI lika kroz *Emerald AI Setup Manager* kao što je kreiranje *Animator Controllera*, postavljanje animacija za mirno stanje, šetanje, trčanje, borbu i smrt, primjenu sposobnosti na popise sposobnosti, dodjeljivanje *Head Transform-a* na AI i postavljanje, oznaka i sloja umjetne inteligencije. Svi detalji opisani su u dokumentaciji[9] i potrebno je pažljivo slijediti sve korake kako bi dobili očekivani rezultat. Od korisnika se ne očekuje pisanje koda, već se sve namješta iz korisničkog sučelja te je ovaj način pogodan za osobe s malo iskustva s umjetnom inteligencijom općenito, ali zna biti izazovan za osobe koje se prvi put koriste Unity platformom. Stoga je jako bitno da osoba prije korištenja ovog alata bude upoznata s mogućnostima Unity-ja.

Dodatni korišten resurs je dodatak *Low Poly Animated Animals*[10] za likove morskih pasa. S obzirom da Emerald alat zahtjeva velik broj animacija za svakog lika, zgodno bi bilo koristiti dodatke koji pružaju modele životinja s većim brojem animacija od korištenog.



Slika 10 Jednostavna scena s primjenom Emerald AI 3.0 alata

[8] Upute za postavljanje *Nav Mesh Baking*:
https://github.com/Black-Horizon-Studios/Emerald-AI/wiki/Baking-NavMesh
[9] Stvaranje novog AI lika u Emerald alatu:
https://github.com/Black-Horizon-Studios/Emerald-AI/wiki/Creating-a-New-AI
[10] Low Poly Animated Animals:
https://assetstore.unity.com/packages/3d/characters/animals/low-poly-animated-animals-93089

6.1.4. Opis scene s primjenom ML-Agents alata

Korišteni pristup za isprobavanje i primjenu ovog alata je bio kreiranje Unity okoline ispočetka[11] koja demonstrira ponašanje kugle, koja pokušava dostići nepomičnu metu, kocku i pritom nastoji ne pasti s ravnine koja nije omeđena. U ovom primjeru stvoren je agent-kugla da ne padne s ravnine te je taj agent treniran da se kreće prema kocki. Kako bi izvršavanje zadatka bilo u duhu teme, nakon treniranja i postavljanja scene, kugla je zamijenjena morskim psom, a kocka-meta je zamijenjena ribicom. Ideja scene je da morski pas lovi ribu kao što je prikazano na Slika 11. Korišteni dodatni resurs je besplatni Asset Fish-PolyPack[12] iz kojeg je iskorišten model morskog psa i ribice.

Slika 11 Jednostavna scena s primjenom ML-Agents alata

Kako bi se uspješno postavila nova ML okolina za učenje potrebno je slijediti korake opisane u dokumentaciji. Kratak pregled i sažetak glavnih komponenata dan je u nastavku:

1. Kreiranje scene u kojoj će objekti biti stvoreni. U ovom slučaju okolina je 3D objekt Plane po kojem se agent i meta kreću.
2. Implementacija Agent subklase. Ta subklasa definira kod koji agent koristi za promatranje svog okruženja, za izvođenje dodijeljenih radnji i za izračunavanje nagrada koje se koriste za podržano učenje. Mogu se implementirati izborne metode za resetiranje agenta kada je završio ili nije uspio svoj zadatak. Skripta koja primjenjuje navedeno u predstavljenom projektu zove se RollerAgent i implementira

[11]Tutorial za kreiranje nove okoline za ML učenje u Unity-ju: https://unity-technologies.github.io/ml-agents/Learning-Environment-Create-New/

[12] Fish-PolyPack: https://assetstore.unity.com/packages/3d/characters/animals/fish/fish-polypack-202232

metode: OnEpisodeBegin(), CollectObservations(VectorSensor sensor),OnActionReceived(ActionBuffers actionBuffers), Heuristic().

3. Kada su svi GameObjekti i ML-Agent komponente iz skripte na mjestu, moramo sve povezati u Unity Editoru. To ćemo napraviti dodavanjem Agent subklase na GameObject, u ovom slučaju, morskog psa te postavljanjem GameObject-a ribice, kao Targeta. Ostaje nam podešavanje Behavior Parametara u korisničkom sučelju kao što je detaljno opisano u dokumentaciji.
4. Uvijek je dobro na početku testirati okolinu prije treniranja modela tako da manualno kontroliramo agenta koristeći tipkovnicu. Potrebno je postaviti Behavior Type na Heuristic Only u Behavior Parameters GUI odjeljku RollerAgent objekta. Nakon što se klikne Play, ako je sve dobro postavljeno, možemo pomicati agenta strelicama na tipkovnici dok će Target bit nepomičan. Kada ribicu „pojedemo“ resetira se igra.
5. Sada je vrijeme za treniranje okoline, potrebno je imati konfiguracijsku datoteku .yaml i dodati u nju definiciju hiperparametara. Nakon toga u terminalu pokrećemo treniranje agenta naredbom:

   mlagents-learn config/rollerball_config.yaml --run-id=RollerBall
6. Nakon toga u Unity Editoru kliknemo Play i kreće treniranje koje možemo i vizualno promatrati. Agent se pomiče prema meti i scena se nakon uspjeha/neuspjeha ponavlja. Statistike Agenta možemo pratitit putem TensorBoard-a[13]. Kada smo zadovoljni treniranjem nakon određenog broja epoha, možemo ga prekinuti. Tada se u results mapi stvara .ONNX datoteka koju u Behavior Parameters sučelju dovučemo do aributa Model dok Behavior Type mora bit postavljen na Default. Kada pokrenemo opet scenu s Play, vidimo da se Agent ponaša inteligentno. Morski pas pokušava uloviti ribicu samostalno te se prilikom uspješnog lova scena resetira ili ako padne s ravnine. Ako smo dobro istrenirali, očekujemo da će morski pas uvijek uspješno loviti i da neće padati.

[13] Praćenje ML-Agent statistike tijekom učenja: https://unity-technologies.github.io/ml-agents/Using-Tensorboard/

6.1.5. Koji alat koristiti?

Odabir pristupa i alata ovisi uvelike o zadanom problemu, onome što se želi postići, dozvoljenim resursima i iskustvu developera. Kada bismo odluku temeljili na ovim aspektima možemo izvući neke generalne zaključke.

Emerald AI 3.0 je alat pogodan kada želimo jednostavno i intuitivno implementirati predefinirana ponašanje likova, a koja su demonstrirana i dobro dokumentirana npr. *Companion, Tamable, Territorial, Shooting, Fighting* i sl. Obzirom da se sve radi iz korisničkog sučelja, razvijatelji ne moraju imati opsežno znanje u području umjetne inteligencije ni kodiranju, a mogu dobiti realistične, korisne i zabavne rezultate koji će obogatiti njihovu igru i poboljšati korisničko iskustvo. Ako želimo postići dodatnu kreativnost, primjenu naprednih ML tehnika i želimo potpunu slobodu u kreiranju AI ponašanja i funkcionalnosti ovaj alat neće biti pogodan jer to jednostavno ne omogućava. Isto tako, svaka novija verzija s više mogućnosti se mora kupovati.

ML-Agents s druge strane, omogućava veliku fleksibilnost jer podržava napredne tehnike dubokog učenja. Razvojni timovi mogu prilagoditi parametre algoritama i čak implementirati vlastite algoritme ako je potrebno. Ovaj alat je pogodan za složene scenarije gdje razvijatelji žele imati slobodu i ispoljiti svoju kreativnost. Velika prednost je i to što je *open-source*, ima veliku zajednicu te se stalno razvija. Nedostaci ovog pristupa su veći zahtjevi za resursima (CPU, GPU, memorija), ali i zahtjevi za dodatnim vremenom koje se troši na treniranje i isprobavanje modela. Nadalje, razvijatelji moraju kodirati, ali i razumijti koncepte dubokog učenja kako bi ga što uspješnije primijenili i iskoristili potencijale alata što zahtjeva veću razinu stručnosti i utrošenog vremena (*steep learning curve*).

6.2. Implementacija proširene stvarnosti u Unity razvojnom okruženju

Proširena stvarnost je tehnologija koja omogućava stvaranje interaktivnih iskustava spajanjem stvarnog i virtualnog svijeta tako da superponira virtualne 3D objekte na stvarnu okolinu korisnika. Cilj ovog dijela studije izvedivosti bio je proučiti podršku koju *Unity* pruža za razvoj proširene stvarnosti te objasniti ključne korake implementacije jedne scene iz prototipa igre *Aqua Saviours* koja koristi proširenu stvarnost.

6.2.1. Proširena stvarnost u *Unity*-u

Unity nudi mnoge radne okvire i *plug in*-ove za rad s proširenom stvarnosti: *AR Kit* (za razvoj proširene stvarnosti za iOS uređaje), *AR Core* (za razvoj proširene stvarnosti za Android uređaje), *Easy AR* (za lakši i brži razvoj proširene stvarnosti bez potrebe za pisanjem programskog koda)...

Za razvoj proširene stvarnosti u ovom projektu odabran je Unityjev vlastiti radni okvir – *AR Foundation*[14]. *AR Foundation* omogućuje razvoj proširene stvarnosti za Android, iOS i druge uređaje te nudi sve mogućnosti koje su potrebne za razvoj projekta *Aqua Saviours*.

6.2.2. Opis aplikacije koja demonstrira razvoj i korištenje proširene stvarnosti

Za potrebe demonstracije implementacije proširene stvarnosti u razvojnom okruženju *Unity* odabrana je scena koja prikazuje dobivenu nagradu u igri *Aqua Saviours* (Slika 9).

Slika 12 Izbornik na početnom ekranu

Slika 13 Prikaz dobivenih nagrada

[14] *AR Foundation*: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.1/manual/index.html

Na Slika 12 prikazan je izbornik na početnom ekranu demo aplikacije u kojoj se ispitivala izvedivost implementacije proširene stvarnosti u igri u skladu s Figma prototipom[15]. Klikom na gumb *Rewards* otvara se panel u kojem se može izabrati životinja koju želimo prikazati u prostoru koristeći proširenu stvarnost (Slika 13).

Slika 14 Odabrana životinja

Slika 15 Detekcija ravnih površina

Nakon odabira neke od ponuđenih opcija, prelazi se na novu scenu u kojoj se vidi prostor oko korisnika. Na Slika 15 vidi se kako igra točkastim uzorkom prikazuje detektirane ravne površine. Dodirom na ekran na području neke detektirane ravne plohe, stvori se prethodno odabrana životinja (Slika 14). Životinju se zatim dodirom prsta može pomicati, skalirati i rotirati. Klikom na gumb *Lock* dolje desno, pozicija, rotacija i veličina životinje se zaključavaju te ona postaje fiksirana u prostoru.

Klikom na gumb *Unlock* ponovno se može mijenjati pozicija, rotacija i veličina životinje. Klikom na gumb *?* na ekranu se prikazuje zanimljiva činjenica o odabranoj životinji kako je vidljivo na Slika 17. Klikom na veliki okrugli gumb u sredini, okida se slika životinje u prostoru

[15] Figma prototip *Aqua Saviours*: https://www.figma.com/file/HfD7OjU9Taeaozfkn1pMvE/OceanRescue_P2G?type=design&node-id=56-21&mode=design&t=MbXOrP54TaK1mtFE-0

koja se zatim sprema u galeriju uređaja (Slika 16). Povratak na početni ekran moguć je klikom na gumb < dolje lijevo.

Slika 17 Zanimljiva činjenica o životinji

Slika 16 Spremanje slike na uređaj

6.2.3. Razvoj pojedinih komponenti

Za izradu scene koja koristi proširenu stvarnost, potrebno je u stvorenu scenu dodati *AR Session* i *XR Origin*.

Da bi korisnik mogao postaviti životinju u prostor, prvo je potrebno detektirati ravnu površinu. Detekcija ravne površine može se napraviti koristeći gotovu komponentu *AR Plane Manager* koju je potrebno staviti na *XR Origin* objekt. Za dodavanje objekata na detektiranu ravnu površinu koristi se *AR Placement Interactable* komponenta koju treba dodati na *AR Session* objekt. U polje *XR Origin* dodati prethodno stvoreni *XR Origin*. Za određivanje objekta kojeg treba staviti u scenu koristi se vlastita skripta pričvršćena na *Game Manager* objekt

koja u *Placement Prefab* polje *AR Placement Interactable* stavlja životinju[16] koja je odabrana u prethodnoj sceni. Modeli životinja dostupni su u *Unity Asset Store*u. *Za* prenošenje informacija između scena (kao što je ovdje ime odabrane životinje) koristi se statička skripta.

Pomicanje, skaliranje i rotiranje životinje napravljeno je pomoću *Lean Touch*[17] *plugin*a. U tom *plugin*u dostupne su skripte *Lean Drag Translate* (za pomicanje), *Lean Pinch Scale* (za skaliranje) i *Lean Twist Rotate Axis* (za rotiranje) koje se dodaju na stvorenu životinju u prostoru.

Za potrebe stvaranje slike, u sceni, uz glavnu kameru, postoji još jedna kamera kako bi se mogla dobiti slika zaslona bez UI elemenata. Klikom na gumb za slikanje, pokreće se postupak stvaranja i spremanja slike. Za stvaranje slike koja će se kasnije spremiti, prvo je potrebno stvoriti teksturu renderiranja (engl. *Render Texture*) iz onoga što vidi kamera za slikanje, a zatim čitati piksele iz te teksture kako bi se izradila slika. Nakon toga, tekstura se kodira u PNG format pomoću klase *Texture2D* i sprema u varijablu kao niz bajtova. Za spremanje slike u galeriju na Android uređajima ili lokalnu memoriju na računalu (što je bilo potrebno tijekom razvoja aplikacije) prvo se provjerava platforma uređaja, a zatim se koristi odgovarajuća logika za spremanje slike. Ako je uređaj Android, koristi se biblioteka *NativeGallery*, dok se u Windows Editoru slika sprema izravno na uređaj metodom *File.WriteAllBytes*.

Tijekom razvoja, za testiranje igre koristio se *Unity Foundation Remote 2.0*[18] kako bi se izbjegla česta i nepotrebna izrada *build*-ova. *Unity Foundation Remote 2.0* omogućuje simulaciju mobilnih igara koje koriste proširenu stvarnost. Kako bi se koristio *Unity Foundation Remote 2.0,* potrebno je prvo preuzeti asset s *Unity Asset Store*-a te ga zatim podesiti na računalu i mobilnom uređaju[19].

---

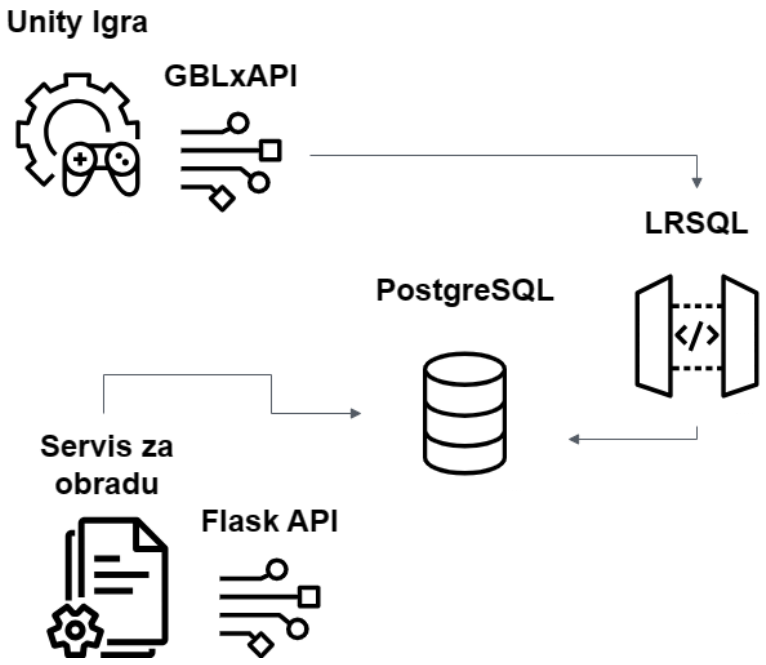
[16] Modeli životinja, *Low Poly Animated Animals*: https://assetstore.unity.com/packages/3d/characters/animals/low-poly-animated-animals-93089
[17] *Lean Touch plugin*: https://assetstore.unity.com/packages/tools/input-management/lean-touch-30111
[18] *Unity Foundation Remote 2.0*: https://assetstore.unity.com/packages/tools/utilities/ar-foundation-remote-2-0-201106
[19] Upute za podešavanje *Unity Foundation Remote 2.0*: https://www.youtube.com/watch?v=fWQwBdxmaVM&ab_channel=samyam

Prikaz nasumične činjenice o odabranoj životinji radi se na način da se iz .json datoteke pročita lista činjenica koja je mapirana na ime životinje te da se generira random broj koji odgovara indeksu činjenice u toj listi koja će se onda prikazati na ekranu.

6.3. Implementacija sustava za prikupljanja korisničkih podataka za evaluaciju učinka

U ovom poglavlju opisat ćemo detaljno sustav za skupljanje i analizu korisničkih podataka na primjeru jednostavne Unity igre kojemu je krajnji cilj sistematično ocijeniti učinak igre.

Prikupljanje podataka bazirano je na *Experience API* (skraćeno xAPI), čije se izjave spremaju na *Learning Record Store* (skraćeno LRS). Dalje, razvijen je servis u Pyhon-u namijenjen analizi prikupljenih podataka koji su dostupni preko *REST API-ja*. Razvoj pojedinih ključnih komponenti sustava, kao i izazovi i ograničenja u implementaciji opisani su u potpoglavljima koja slijede. Arhitektura sustava prikazana je na Slika 18.



Slika 18 Arhitektura sustava za prikupljanje korisničkih podataka

6.3.1. Testna Unity igra

Za potrebe testiranja implementacije prikupljanja i analize podataka napravljen je kviz znanja u Unityju. Kviz testira korisnikovo poznavanje činjenica o moru i morskim životinjama. Pitanja

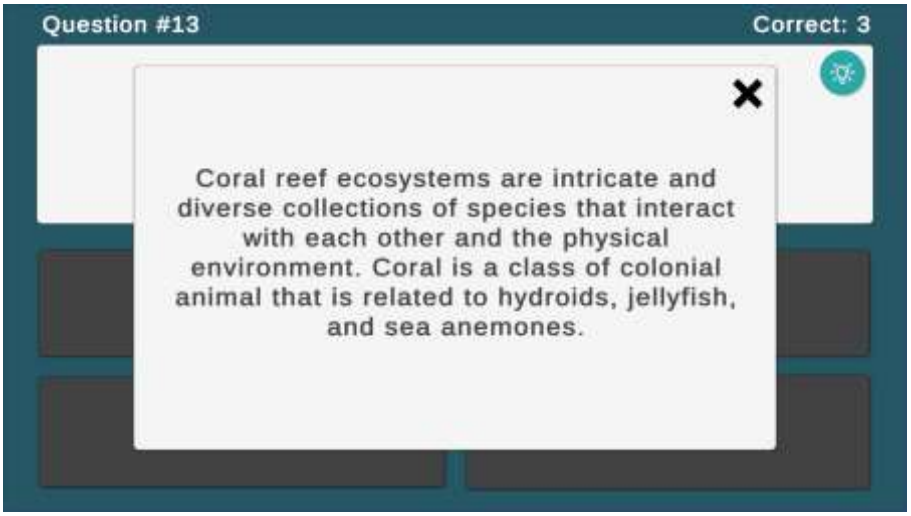
i odgovori te oznaka točnog odgovora definirani su zasebno u JSON[20] datoteci. Moguće je uz svako pitanje definirati kratki odsječak koji bi trebao korisniku pomoći pronaći odgovor. Kako bi se podaci agregirali za različite korisnike prije prvog rješavanja kviza potrebno je unijeti korisničko ime. Kviz se može ponavljati neograničen broj puta. (Slike Slika 19, Slika 20 i Slika

Slika 19 Kviz znanja u Unityju - izbornik

21)

Slika 20 Kviz znanja u Unityju - igra u tijeku

[20] JavaScript Object Notation: https://www.json.org/json-en.html



Slika 21 Kviz znanja u Unityju - hintovi

6.3.2. GBLxAPI

Unity igru proširili smo podrškom za korištenje *xAPI* izjava. *Experience API*[21] (skraćeno xAPI) predstavlja tehničku specifikaciju programskog sučelja koja omogućava prikupljanje podataka o iskustvu neke osobe. Komunikacija je propisana između entiteta klijent (engl. *activity provider*) i krajnje točke (engl. *endpoint*) koja zna čitati xAPI izjave (engl. *statements*). Svaka akcija koju želimo zabilježiti generira izjavu koja se sastoji od *imenice*, *glagola* i *objekta*, a poslužitelj je odgovoran za spremanje iste. Pojednostavljeno, spremamo informacije o tome "Tko je učinio što".

U našem slučaju klijent je već spomenuta kviz igra, a podrška za generiranje izjava omogućena je kroz paket *GBLxAPI*[22]. Izvorni kod paketa potrebno je smjestiti u *Assets* direktorij Unity projekta. *GBL_interface.cs* je skripta koji sadrži template za stvaranje statičkih funkcija za generiranje izjava i dovoljno je njega proširiti s metodama koje generiraju izjave za pojedine parametre.

Svaki glagol i objekt izjave mora biti definiran, a definicije istih nalaze se u *Vocabulary* direktoriju paketa. *Vocabulary* sadrži dvije Excel datoteke, jedna je popunjena s predefiniranim glagolima, a druga je prazna u koju se mogu dodati specifični glagoli i objekti. Tu ipak nailazimo na prve poteškoće jer zbog greške u izvornom kodu paketa korisnički

[21] Experience API: https://xapi.com/ecosystem/
[22] Unity xAPI paket - GBLxAPI: https://github.com/gblxapi/UnityGBLxAPI

definirani glagoli se ne prepoznaju, a sam paket nije ažuriran od 2021. godine. Stoga smo se ograničili na korištenje predefiniranih glagola. Excel tablice potrebno je prevesti u JSON datoteke koristeći *GBL_Json_Parser.py* i premjestiti ih u *Data* direktorij paketa.

Konačno, konfiguraciju klijenta potrebno je napraviti u *GBLConfig.cs* skripti postavljanjem URL-a xAPI krajnje točke te korisničkog imena i lozinke kojom se omogućuje pristup krajnjoj točki.

Imenica i objekt u generiranim izjavama u ovom kontekstu bile su konstantne. Imenica je bio korisnik koji igra igru, a objekt izjave bila je igra. Korišteni su sljedeći glagoli:

- *started* - predstavlja početak igranja jedne iteracije kviza (identifikator igre)
- *earned* - broj točnih odgovora
- *read* - akumulirano vrijeme provedeno čitajući edukativni sadržaj
- *completed* - vrijeme potrebno za prolazak jedne iteracije kviza (u sekundama).

Podaci koji se žele pratiti moguće je proširiti korištenjem drugih glagola, a ako dodamo na to da možemo mijenjati i objekt izjava dobijemo veliki konačni set različitih izjava.

6.3.3. LRSQL

U prethodnom potpoglavlju definirali smo klijent koji generira xAPI izjave. Sada je potrebno dodati i poslužitelja koji će ih konzumirat. *LRS (Learning Record Store)*[23] predstavlja poslužitelj koji zna čitati, spremati i prikazivati akumulirane xAPI izjave. Autori prethodno opisanog GBLxAPI paketa demonstrirali su spremanje i prikaz izjava preko *Learning Lockera*[24] pa je on bio prvi izbor za *LRS* u našem primjeru. *Learning Locker* je *LRS* otvorenog koda koji je moguće podići na lokalno računalo i činio se kao idealan kandidat. Tu se ponovno javljaju poteškoće. Osim što ovaj poslužitelj zahtjeva specifičnu verziju Ubuntua i ostalih alata poput *Node.js* i *MongoDB* čak i kad se uspije podići pogađanjem verzija alata sam API i dalje ne radi zbog greške u izvornom kodu. Na žalost, razlog tomu je najvjerojatnije neodržavanje projekta od 2021. godine.

[23] *Learning Record Store*: https://xapi.com/learning-record-store/
[24] *Learning Locker:* https://learninglocker.atlassian.net/wiki/spaces/DOCS/overview

Drugi *LRS* otvorenog koda koji se mogao lokalno podignuti je *LRSQL*[25]. Ovaj *LRS* napravljen je za platformu *Windows*, a podatke sprema u SQL bazi podataka. Preciznije, korisnik bira između SQLite[26] i PostgreSQL[27] baza podatka. Iako nudi manje mogućnosti od *Learning Lockera* (npr. nema podršku za vizualizaciju prikupljenih podataka) u našem slučaju je dovoljan jer služi samo kao agregator izjava dok se analiza provodi u odvojenom servisu.

Kako bi uspješno povezali *GBLxAPI* i *LRSQL* potrebno je unutar *LRSQL* adminskog sučelja generirati API tajnu i kopirati je u *GBLxAPI* konfiguraciju prikazano na Slika 23. Uspješno spremanje xAPI poruka možemo vidjeti na *LRS Monitor* odjeljku alata. (Slika 22).

Slika 23 Generiranje ključa za pristup xAPIju

Slika 22 Prikaz pristiglih izjava na SQLRS admin stranici

[25] SQL baziran LRS (LRSQL): https://github.com/yetanalytics/lrsql
[26] *SQLite*: https://www.sqlite.org/index.html
[27] *PostgreSQL*: https://www.postgresql.org/

6.3.4. Analiza

Analiza prikupljenih podataka zamišljena je kao odvojen Python servis i za potrebe ovog projekta limitiranih je mogućnosti. Krajnje točke za dohvat rezultata podignute su putem radnog okvira Flask[28]. Servis je povezan na istu bazu podataka kao i LRS i agregira rezultate po korisnicima igre. Jedan od krajnjih točaka servisa radi korelacijsku analizu između bilo koja dva podatka koja su se pratili u igri što se može vidjeti na Slika 24. Dovoljno je poslati GET zahtjev na *http://{host}/correlation/{parametar1}/{parametar2}.* U budućnosti je zamišljeno proširiti skup kranjih točaka na sličan način i omogućiti, osim formatiranog ispisa, dohvat podataka u obliku JSONa kako bi ostale aplikacije mogle prikazivati rezultate.

```
PS C:\Users\Iva> curl http://127.0.0.1:5000/correlation/hint/points
##########################   USER: another   ################################

Correlation statistics for hint and points:

Sample size: 6
hint: mean: 0.0; std: 0.0
points: mean: 6.0; std: 1.632993161855452

Covariance Matrix:
[[0.  0. ]
 [0.  3.2]]

Pearson's coefficient:
statistics: nan; p-value: nan

Spearman coefficient:
statistics: nan; p-value: nan

################################################################################


########################   USER: username   ################################

Correlation statistics for hint and points:

Sample size: 4
hint: mean: 15.657667499999999; std: 16.016188227230245
points: mean: 7.75; std: 1.920286436967152

Covariance Matrix:
[[342.02438044   7.71490917]
 [  7.71490917   4.91666667]]

Pearson's coefficient:
statistics: 0.18813383218423935; p-value: 0.8118661678157686

Spearman coefficient:
statistics: -0.3333333333333334; p-value: 0.6666666666666666

################################################################################
```

Slika 24 Dohvaćanje rezltata korelacijske analize između vremena provedenog na hintovima i uspješnosti rješenosti kviza

[28] Flask - mikro web radni okvir za Python: https://flask.palletsprojects.com/en/3.0.x/

6.4. Implementacija sustava dijaloga u Unity razvojnom okruženju

Dijalozi su najvažniji za prijenos edukacijskog sadržaja u video igrama jer omogućuju interaktivno, angažirajuće i kontekstualizirano učenje. U *Unity*-u, postoji nekoliko načina za implementaciju sustava dijaloga. Za ozbiljnu igru poput *Aqua Saviours* najprikladnije bi bilo korištenje komponente poput *Dialogue 2* ili alata kao što je *Yarn Spinner*.

6.4.1 Implementacija dijaloga pomoću komponente *Dialogue 2*

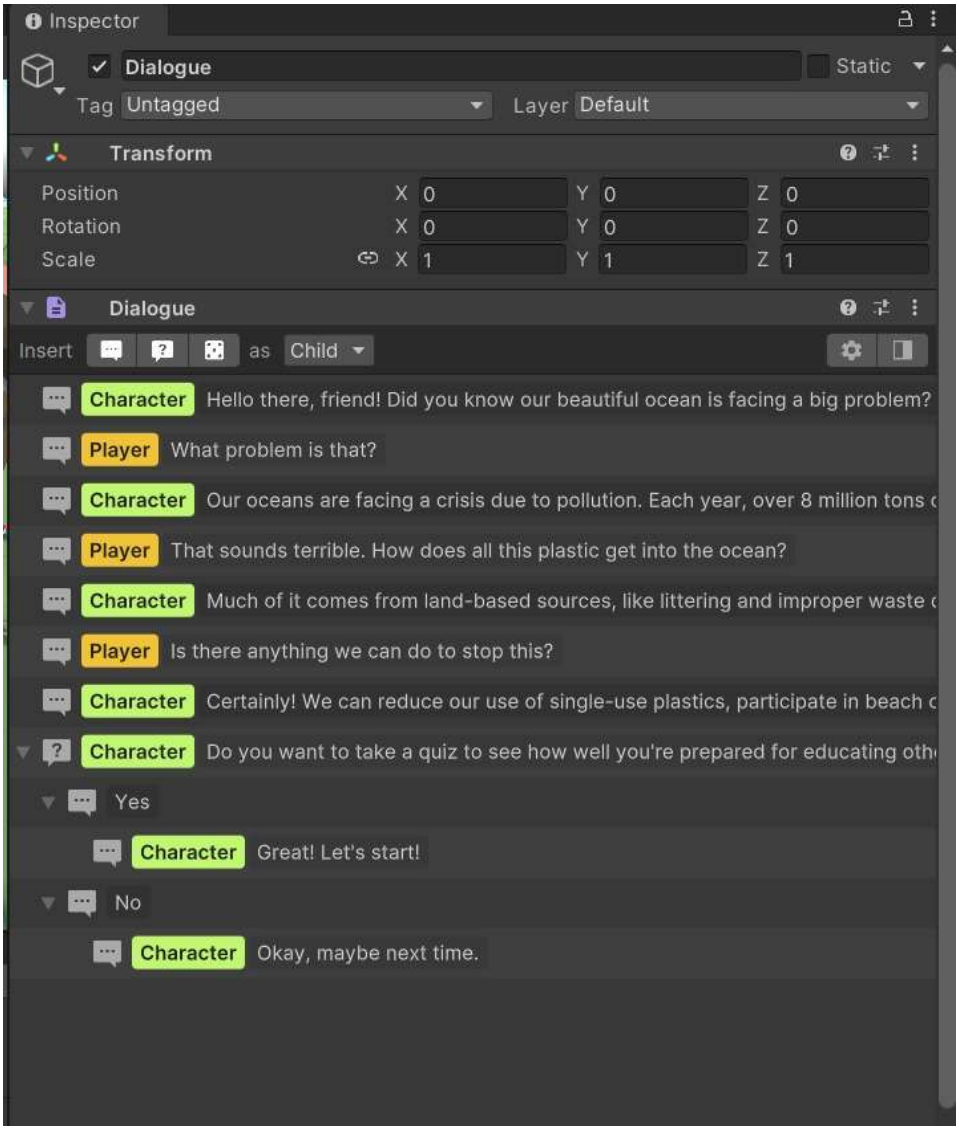
Komponenta *Dialogue 2* tvrtke *Catsoft Works* moćan je alat koji razvojnim programerima s lakoćom omogućuje stvaranje složenih sustava dijaloga. Ova komponenta pruža intuitivni vizualni uređivač koji omogućuje stvaranje razgranatih razgovora i opcija dijaloga s višestrukim izborom. Ova komponenta bi u igri *Aqua Saviours* služila za kreiranje razgovora između igrača i neigrivih likova (engl. *Non-playable characters, NPC*), poput životinja u nevolji i drugih ljudi. Komponenta bi omogućila impresivnije pripovijedanje i pomogla bi igračima da bolje razumiju ciljeve igre.

Neke od korisnih značajki komponente *Dialogue 2* koje bi doprinjele razvoju igre *Aqua Saviours* su sljedeće:

- **Okidači (engl. *triggers*) i radnje (engl. *actions*):** *Dialogue 2* omogućuje stvaranje okidača koji mogu pokrenuti dijaloge na temelju određenih uvjeta ili radnji koje izvodi igrač, primjerice klikom na određenog lika ili pritiskom određene tipke na tipkovnici. Također, ovisno o radnjama igrača moguće je dalje granati dijalog.
- **Višejezična podrška**: *Dialogue 2* može podržati više jezika, što je ključno za dosezanje šire publike i osiguravanje da je obrazovni sadržaj igre dostupan globalnoj bazi igrača. Tekstovi u igri mogu se jednostavno prevoditi pomoću postavki lokalizacije (engl. *localization*) gdje se u radnu okolinu mogu uvesti (engl. *import*) brojni svjetski jezici.
- **Sinkronizacija** (engl. voice over): *Dialogue 2* se može konfigurirati za rad s glasovnim datotekama, dajući razvijateljima mogućnost dodavanja govornog dijaloga, što pridonosi povećanju kvalitete edukacijskog sadržaja.
- **Analitika i povratne informacije**: Moguće je pratiti u kojim dijalozima igrači najviše sudjeluju, što omogućuje uvid u njihove interese i daljnje poboljšanje edukacijskog sadržaja igre.

Glavni koraci kreiranja dijaloga pomoću komponente *Dialogue 2* su sljedeći:

1. **Instalacija komponente *Dialogue 2***: Pretraživanjem komponenti u *Package Manager*-u potrebno je prvo pronaći komponentu *Game Creator* i instalirati ju te zatim instalirati komponentu *Dialogue 2* koja je podkomponenta *Game Creatore*-a.
2. **Stvaranje sustava dijaloga**: Unutar korisničkog sučelja potrebno je kreirati novi objekt tipa *Dialogue*. Na Slika 25 vidljivo je novo sučelje kreiranog objekta *Dialogue*.
3. **Definiranje čvorova dijaloga:** Unutar *Dialogue* objekta potrebno je definirati različite čvorove dijaloga koji predstavljaju linije dijaloga za likove. Uređivanjem svakog čvora može se mijenjati njegov tekstualni sadržaj, lik koji pripada tom čvoru, ekspresije lika koji pripada tom čvoru, ponašanje lika prije i nakon izvršavanja čvora te brojni drugi elementi.
4. **Konfiguriranje grananja dijaloga**: Unutar korisničkog sučelja jednostavno je podesiti grananje dijaloga. Primjerice, ako je kreiran čvor tipa *Choice* i potrebno je na temelju igračeva izbora preusmjeriti dijalog na odgovarajući čvor tipa *Text*, potrebno je čvor tipa *Text* dodati opcijom *Insert as Child* umjesto opcijom *Insert as Sibling*. Na taj način stvara se nova grana dijaloga koji se i još dalje može granati na isti način. Različiti uređeni tipovi čvorova vidljivi su na Slika 25.
5. **Prilagođavanje izgleda dijaloga**: Izgled samog dijaloga može se mijenjati uređivanjem objekta *Dialogue*, kao i uređivanjem samih čvorova dijaloga. Na taj način, izgled dijaloga može se prilagoditi kako bi se dijalog vizualno bolje uklopio u okolinu ostatka igre.
6. **Testiranje dijaloga**: Na kraju, potrebno je pokrenuti igru i dijalog kako bi se osigurala ispravnost sustava dijaloga i načina na koji je prethodno konfiguriran.



Slika 25 Sučelje za kreiranje dijaloga

Slika 26 Primjer dijaloga korištenjem komponente Dialogue 2

Slika 27 Primjer izbora u dijalogu korištenjem Dialogue 2 komponente

Na Slika 26 vidljiv je prikaz dijaloga kreiranog pomoću komponente *Dialogue 2* čiji je okidač pritisak simbola *E* na tipkovnici, dok je na Slika 27 vidljiv prikaz čvora dijaloga u kojemu igrač bira između dvije opcije pritiskom simbola *1* ili *2* te tako pokreće grananje dijaloga.

Dialogue 2 je efikasan alat za stvaranje sustava dijaloga unutar Unityja, pružajući raznovrsne opcije za razgranate dijaloge i interaktivno pripovijedanje. Međutim, ako razvijatelji žele implementirati vlastite skripte za upravljanje dijalozima, kao što je dinamičko učitavanje teksta iz vanjskih datoteka, *Dialogue 2* ne bi bio najprikladniji izbor. Prilagodbe izvornog koda

koje bi bile potrebne za takvu funkcionalnost mogu biti složene, a nedostatak detaljne dokumentacije API-ja može dodatno otežati razvoj.

6.4.2 Implementacija dijaloga pomoću komponente Yarn Spinner

Yarn Spinner je popularna komponenta za osmišljavanje i pisanje dijaloga unutar *Unity* razvojnog okruženja. Ističe se svojim skriptnim jezikom koji je lako čitljiv ljudima te zbog toga pojednostavljuje proces stvaranja složenih interaktivnih dijaloga. Takav jezik, poznat kao *Yarn*, omogućuje razgranate dijaloge u kojima izbori i akcije igrača mogu utjecati na smjer i ishod priče. *Yarn Spinner* je posebno koristan u igrama koje se temelje na naraciji gdje je pripovijedanje najvažnije. Također pojednostavljuje lokalizaciju, olakšavajući prevođenje igara na više jezika, čime se doseže šira publika. Fleksibilnost *Yarn Spinner*-a proširuje se na njegovu sposobnost rukovanja varijablama i uvjetnim izjavama, omogućujući konstruiranje zamršenih scenarija koji se mogu promijeniti na temelju igračevih prethodnih odluka ili radnji tijekom igre. Dodatno, podržava uključivanje vizualnih i audio sredstava, usklađivanje govornog dijaloga i mijenjanje izraza likova s pisanom riječi, obogaćujući kvalitetu igre.
Implementacija sustava dijaloga u *Unity*-u s *Yarn Spinnerom* slijedi sljedeće temeljne korake:

1. **Instalacija *Yarn Spinner*-a**: *Yarn Spinner* potrebno je dodati svom *Unity* projektu putem *Unity Package Manager*-a ili uvozom iz *Asset Store*-a.

2. **Pisanje *Yarn* skripti**: Potrebno je stvoriti *.yarn* datoteku u kojoj će se implementirati dijalog korištenjem *Yarn* sintakse, koja podržava grananje dijaloga i odabire igrača. Primjer takve skripte vidljiv je na Slika 30.

3. **Integracija s radnim okruženjem *Unity***: *Yarn Spinner* potrebno je integrirati s radnim okruženjem *Unity* tako što se postave objekti *DialogueRunner* i *YarnProject* u *Unity* sceni. Ukoliko je potrebno omogućiti pokretanje dijaloga tek kada igrač odradi određenu akciju, potrebno je i implementirati vlastitu skriptu koja će služiti kao posrednik između *Yarn Spinner* objekata i *Unity*-a u svrhu upravljanja objektima u igri. Takva *C#* skripta prikazana je na Slika 31.

4. **Testiranje dijaloga**: Na kraju, potrebno je pokrenuti igru i dijalog kako bi se osigurala ispravnost sustava dijaloga i načina na koji je prethodno konfiguriran.

Slika 28 Dijalog kreiran pomoću komponente Yarn Spinner

Slika 29 Izbor igrača tijekom dijaloga kreiranog pomoću komponente Yarn Spinner

Na Slika 28 vidljiv je prikaz edukativnog dijaloga između igrača i neigrivog lika, kreiranog pomoću komponente *Yarn Spinner*, a na Slika 29 vidljivo je da igrač može donijeti odluku na temelju koje se dijalog dalje grana.

Slika 30 predstavlja jednostavnu *Yarn* skriptu koja sadrži tekst razgovora u kojem lik podiže svijest o zagađenju oceana, a zatim nudi igraču izbor da riješi kviz o toj temi. Ovisno o odgovoru igrača, skripta postavlja varijablu *choseYes* i vodi do različitih grana dijaloga.

Yarn Spinner ističe se svojom jednostavnošću i fleksibilnošću, omogućujući kreiranje sustava dijaloga točno prema potrebama igre. Kao otvoreni softver koji koristi jednostavan tekstualni jezik za skriptiranje, nudi veliku slobodu za prilagodbu i idealan je za igre koje zahtijevaju posebne mehanike dijaloga ili integraciju s drugim sustavima igara.

U usporedbi s *Dialogue 2*, koji nudi više rješenja “izvan kutije” (engl) sa svojim vizualnim uređivačem i ugrađenim funkcijama, *Yarn Spinner* nudi više slobode za prilagodbu i može biti prikladniji za razvoj igara koje zahtijevaju jedinstvenu mehaniku dijaloga i visok stupanj prilagodljivosti.

Slika 30 Primjer Yarn skripte

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Yarn.Unity;
0 references
public class YarnInteractable : MonoBehaviour {
    1 reference
    [SerializeField] private string conversationStartNode;
    4 references
    private DialogueRunner dialogueRunner;
    4 references
    private Light lightIndicatorObject = null;
    2 references
    private bool interactable = true;
    3 references
    private bool isCurrentConversation = false;
    1 reference
    private float defaultIndicatorIntensity;

    0 references
    public void Start() {
        dialogueRunner = FindObjectOfType<Yarn.Unity.DialogueRunner>();
        dialogueRunner.onDialogueComplete.AddListener(EndConversation);
        lightIndicatorObject = GetComponentInChildren<Light>();
        if (lightIndicatorObject != null) {
            defaultIndicatorIntensity = lightIndicatorObject.intensity;
            lightIndicatorObject.intensity = 0;
        }
    }

    0 references
    public void OnMouseDown() {
        if (interactable && !dialogueRunner.IsDialogueRunning) {
            StartConversation();
        }
    }

    1 reference
    private void StartConversation() {
        isCurrentConversation = true;
        dialogueRunner.StartDialogue(conversationStartNode);
    }

    1 reference
    private void EndConversation() {
        if (isCurrentConversation) {
            isCurrentConversation = false;
        }
    }

    0 references
    public void DisableConversation() {
        interactable = false;
    }
}
```

Slika 31 Primjer C# skripte za integraciju Yarn Spinner-a il Unity-a

6.5. Integracija Unity projekta u Flutter aplikaciju

U ovom potpoglavlju dublje ćemo istražiti i opisati ključne korake koji su potrebni kako bi se uspješno integrirao Unity projekt u Flutter aplikaciju. Ovaj proces zahtijeva temeljito razumijevanje oba okvira i pravilno povezivanje njihovih komponenti kako bi se postigla kvalitetna i usklađena integracija. Istražit ćemo tehnološke izazove i navesti korake za uspješnu komunikaciju između Unity okvira i Flutter platforme.

6.5.1. Upute za postavljanje projekta

Potrebne karakteristike:

- Instaliran Flutter [29] i potrebne karakteristike navedene u uputama
- Instaliran Unity Hub, Unity [30] (2019.4.3 ili noviji) i potrebne karakteristike navedene u uputama
- Postojeći Flutter projekt [31]
- Postojeći Unity projekt [32]
- Pripadajući paket *fuw-XXXX.unitypackage* preuzet s *unitypackages* [33] (trenutno *fuw-2022.2.0.unitypackage,* kasnije objašnjeno)
- Pametni telefon
    - opcija emulatora je nepouzdana i ograničena

Za integraciju Unity projekta u Flutter aplikaciju koristi se biblioteka *flutter_unity_widget* [34] koja je pogodna za integraciju za Android, iPad OS, iOS i Web uređaje, dok je potpora za Windows u procesu.

1. korak: dodati *flutter_unity_widget* pod *dependecies* Flutter projekta
    - *flutter_unity_widget*: ^2022.2.0 – za Flutter 3.0.0+
    - *flutter_unity_widget*: ^2022.1.0+7 – za starije verzije Flutter-a
2. korak: postaviti Unity projekt unutar Flutter projekta

[29] Flutter instalacija - https://docs.flutter.dev/get-started/install
[30] Unity instalacija - https://learn.unity.com/tutorial/create-your-first-unity-project#5c7f8528edbc2a002053b410
[31] Flutter novi projekt - https://docs.flutter.dev/get-started/test-drive#create-app
[32] Unity novi projekt - https://learn.unity.com/tutorial/create-your-first-unity-project
[33] Unitypackages - https://github.com/juicycleff/flutter-unity-view-widget/tree/master/unitypackages
[34] flutter_unity_widget - https://pub.dev/packages/flutter_unity_widget

- Napraviti direktorij „unity“ unutar Flutter projekta
- Postaviti Unity projekt unutar „unity“ direktorija (*unity/nazivUnityProjekta*)

3. korak: preuzeti odgovarajuću verziju *fuw-XXXX.unitypackage*
    - Paketi se nalaze na github repozitoriju [35]
    - Odabrati odgovarajuću verziju
        - Ne postoje specifične upute koju točno verziju odabrati, prvo treba provjeriti najnoviju verziju (trenutno *fuw-2022.2.0.unitypackage*)
        - Ako ne funkcionira, postupno provjeravati starije verzije

Sljedeći koraci se odnose na postavljanje unutar Unity projekta:

4. korak: uvesti preuzeti *unitypackage*
    - *Assets > Import Package > Custom Package*
    - Odabrati preuzeti paket i onda Import
5. korak: prilagodba postavki *Player Settings*
    - *File > Build Settings > Player Settings*
    - Te pod *Other settings > Configuration* promijeniti:
        - *Scripting Backend* postaviti na *IL2CPP*
        - *Target Architectures* označiti *ARMv7* i *ARM64*
        - *Active Input Handling* postaviti na *Input Manager (Old)* ili *Both*
            - Moguće naknadne promjene tijekom razvoja pri detekciji dodira korisnika na zaslon

Nakon koraka 5. pojavljuje se *Flutter* opcija u gornjoj navigacijskoj traci *Unity* uređivača. Ta opcija se koristi za izvoz Unity projekta. Projekt je potrebno izvesti svaki put kada se napravi neka promjena na strani *Unity*-a (i prilikom prve integracije projekta).

6. korak: izvoz Unity projekta
    - *Flutter* opcija navigacijske trake
    - Odabrati *Export Android Debug* ili *Export Android Release*
    - Ovim korakom se Unity projekt izvozi u Flutter projekt pod *android/unityLibrary*

[35] Unitypackages - https://github.com/juicycleff/flutter-unity-view-widget/tree/master/unitypackages

Sljedeći koraci se odnose na postavljanje unutar Flutter projekta:

7. korak: postavljanje Android NDKa
    - Potrebno je uskladiti verziju s verzijom iz Unitya
    - Najlakši način je da se putanja postavi na istu putanju koju koristi Unity
    - Unutar Unity uređivača pod *Edit > Preferences > External Tool* pronaći *Android NDK* i kopirati postavljenu putanju
    - Kopiranu putanju je potrebno zaljepiti unutar Flutter projekta pod *andoid/local.properties* kao varijablu *ndk.dir =*
    - Primjer:
      ndk.dir=C:\\Program Files\\Unity\\Hub\\Editor\\2021.3.13f1\\Editor\\Data\\PlaybackEngines\\AndroidPlayer\\NDK
    - Drugi način usklađivanja je da se ručno promijeni potrebna verzija definirana u *android/app/build.gradle*
    - Unutar varijable *andoid { … }* promijeniti *ndkVersion „ xxx “* na odgovarajuću verziju
8. korak: moguća je potreba za postavljanjem ispravne vrijednosti za *minSdkVersion* postavljenu u *android/app/build.gradle*
    - Provjeriti postavljenu vrijednost unutar Unity-a pod *Player Settings > Minimum API Level*
    - Uskladiti verziju unutar Flutter projekta pod varijablom *minSdkVersion*
9. korak: u pripadajućoj datoteci, gdje se integrira Unity projekt uvesti *flutter_unity_widget*
    - Dodati *import 'package:flutter_unity_widget/flutter_unity_widget.dart';*
    - Sada se koristi *UnityWidget* kao običan Flutter Widget

Dodatne opcionalne postavke, te nejasnoće su opisane na stranici *flutter_unity_widget* pod odjeljkom *Setup*[36].

[36] Setup - https://pub.dev/packages/flutter_unity_widget#setup

6.5.2. Upute ostvarivanja komunikacije Flutter <-> Unity

Kod ostvarivanja komunikacije potrebno je unijeti promjene i u Flutter i u Unity projektu.

- Komunikacija Flutter -> Unity
    - Dohvatiti *UnityWidgetController* u *UnityWidget*-u pozivom povratne funkcije *onUnityCreated*
    - Preko *UnityWidgetController*, pomoću njegove funkcije *postMessage* se šalje poruka Unityju
    - *postMessage* prima 3 parametra:
        - *gameObject*
            - naziv Unity GameObjekta
            - GameObjekta mora biti postojeći unutar Unity-ja, te se nazivi moraju podudarati
        - *methodName*
            - naziv Unity funkcije
            - mora postojati pod *MonoBehaviour* klasom GameObjekta, te biti dostupna kao metoda
            - nazivi se moraju točno podudarati
            - primati *string* parametar
        - *message*
            - *String* poruka koja se šalje Unityju
- Komunikacija Unity -> Flutter
    - Odabrati GameObjekt koji pokreće komunikaciju
    - Dodati mu komponentu *Unity Message Manager*
    - Stvoriti novu *MonoBehaviour* potklasu i dodati je pripadajućem objektu kao skriptu
    - Unutar skripte pozvati *GetComponent<UnityMessageManager>()* kako bi se dohvatio *UnityMessageManager*
    - Koristiti *SendMessageToFlutter* za slanje poruka
    - Unutar Flutter koda, poruka se prima preko povratne funkcije *onUnityMessage* od *UnityWidget*-a

6.5.3. Prototip

U ovom dijelu prikazan je prototip implementiran na temelju zahtjeva ovog zadatka. Izrađen je prototip Flutter aplikacije s integriranim Unity projektom.

Flutter aplikacija se sastoji od ekrana:

- *Home screen*
    - sadrži gumb koji otvara ekran *Main menu screen*
- *Main menu screen*
    - prikazuje početni ekran Unity projekta te sadrži ispis rezultata iz Unity igara
- *Game1 screen*
    - sadrži Flutter elemente za kontroliranje igre 1
- *Game2 screen*
    - sadrži Flutter elemente za kontroliranje igre 2

Unity projekt se sastoji od scena:

- *Main menu scene*
    - sadrži gumbove (*Game 1* i *Game 2*) za odabir scena *Game1* i *Game2*
- *Game1 scene*
    - Igra 1
    - gumb za povratak na *Main menu scene*
- *Game2 scene*
    - Igra 2
    - gumb za povratak na *Main menu scene*

Ostvarene komunikacije Flutter <-> Unity:

- Flutter -> Unity
    1. Gumb *Sign in* poziva ekran s integriranim Unity projektom
    2. Igra 1:
        2.1. Gumb *Flutter jump* šalje informaciju skoka karaktera
        2.2. Slider šalje trenutno postavljenu vrijednost rotacije karaktera
    3. Igra 2:
        3.1. Strelice lijevo/desno šalju informaciju o kretanju karaktera lijevo – desno

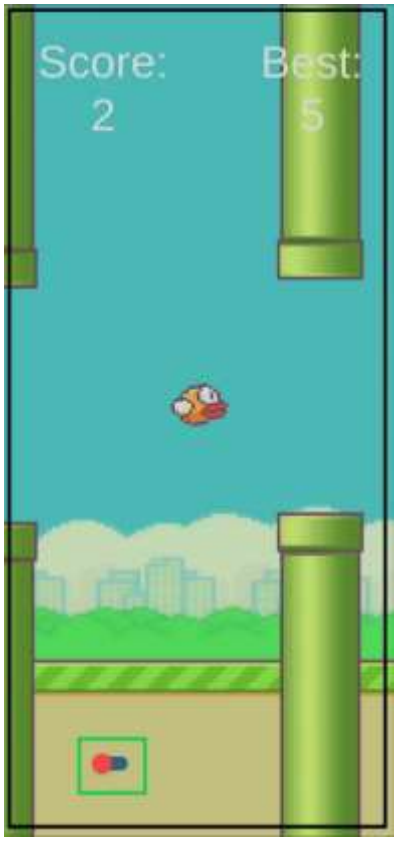
- Unity -> Flutter
    1. Gumb *Game 1*
        - šalje informaciju da se otvara igra 1
    2. Gumb *Game 2*
        - šalje informaciju da se otvara igra 2
    3. Gumbovi za vraćanje iz igre 1/2 na *Main menu screen-u*
        - Šalju informaciju da je zatvorena trenutna igra i da se korisnik vraća na *Main menu screen*
        - Šalju informaciju o postignutom rezultatu korisnika
            - Rezultat se prikazuje u Flutter elementu na *Main menu screen-u*

Na sljedećim slikama prikazan je izrađeni prototip. Na slikama (Slika 33 i Slika 32) su označeni dijelovi aplikacije koji pripadaju Flutter platformi i dijelovi Unity okvira:

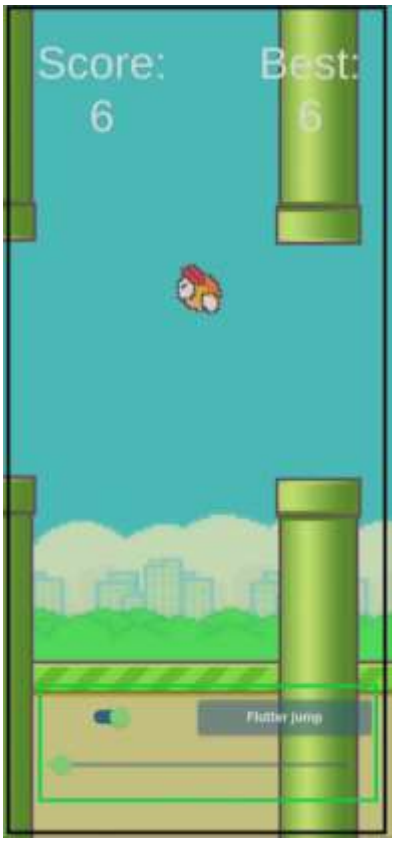
- Flutter – kvadrat obruba zelene boje
- Unity – kvadrat obruba crne boje

Slika 33 Home screen

Slika 32 Main menu



Slika 34 Game1 default



Slika 35 Game1 Flutter UI

Slika 36 Game1 end

- Flutter – *Game1 screen*
    1. Slika 34 *toggle* gumb za prikaz Flutter UI dijela
    2. *Slika 35*
        - *Flutter jump* – Flutter gumb koji služi za skok Unity karaktera
        - Flutter *Slider* – za kontroliranje rotacije Unity karaktera

- Unity – *Game1 scene*
    1. Unity igra 1
    2. *Slika 36* Unity gumb za povratak na *Main menu screen*
        - šalje informaciju Flutter aplikaciji o povratku i ostvarenom rezultatu

Slika 37 Game2 deafult

Slika 38 Game2 Flutter UI

- Flutter – *Game2 screen*
    1. *Slika 37 toggle* gumb za prikaz Flutter UI dijela
    2. Slika 38*Slika 38* Flutter UI za pomicanje Unity karaktera lijevo - desno

- Unity – *Game2 scene*
    1. Unity igra 2

7. Evaluacija rezultata studije izvedivosti

U ovom poglavlju daje se osvrt na sam prototip igre *Aqua Saviour* te komentari nakon provedbe studije izvedivosti. Također se daje ocjena je li igra izvediva te komentari i smjernice za budući rad.

7.1. Evaluacija prototipa igre Aqua Saviours

Evaluacija prototipa igre *Aqua Saviours* otkriva niz pozitivnih aspekata. Tim *Ocean Rescue*, koji je osmislio ovaj prototip, smislio je zanimljivu priču za igru. Također, mnoštvo mini-igara unutar igre pružaju dinamično iskustvo igranja. Nadalje, očit je potencijal u korištenju novih tehnologija jer one dodatno potiču interes korisnika.

Međutim, zapaženi su i određeni nedostaci koji zahtijevaju dodatnu pozornost ako se planira ići u daljnji razvoj igre. Iako je priča zanimljiva, nedostaje joj detalja kako bi korisnik zapravo uronio u svijet koji igra pokušava stvoriti. Nadalje, iako je ovo igra temeljena na novim tehnologijama, te nove tehnologije se u igri jako malo koriste. Prijedlozi za alternativne načine korištenja novih tehnologija su u poglavlju *7.3. Ocjena izvedivosti i smjernice za* budući rad. No, najveći problem predstavlja nedostatak edukacijskog sadržaja – treba ga biti više i treba ga bolje uklopiti u igre kako bi one bile i zabavne i korisne.

7.2. Evaluacija implementacije

U ovom potpoglavlju napravljen je osvrt na samu izvedivost igre *Aqua Saviours* po definiranim zadatcima.

7.2.1. Evaluacija umjetne inteligencije

Razvoj alata za jednostavnu primjenu umjetne inteligencije u Unity okruženju napreduje jako brzo. Pokazali smo da je primjena dvaju izabranih, Emerald AI 3.0 i ML-agents moguća uz neke manje poteškoće prilikom instalacije i integracije što oduzima vrijeme prilikom razvoja igrice. S obzirom na samu prirodu igre Aqua Saviours, čiji je primarni cilj edukacija, primjena AI alata ne bi doprinijela tom aspektu. Zbog nedostatka vremena ne postoji konkretna vizija scenarija u kojem bi implementacija AI agenata učinila igricu zabavnijom, edukativnijom i korisnijom.Postoje neke mogućnosti koje bi se mogle razrađivati u budućem radu kao npr. kreiranje životinje pomagača koja demonstrira sakupljanje otpada po podmorju prije nego i

sami krenemo sakupljati otpad u igrici. Model za takvu akciju je istreniran kao što je opisano u poglavlju 6.1.4. te se može iskoristiti. Osmišljavanje scenarija, likova, integracija i treniranje oduzima dosta vremena. Smatramo da bi to bio nepotreban dodatak i da nema smisla gubiti vrijeme koje bi se moglo korisnije utrošiti na edukativni razvoj. Mogli bi se dodati neigrivi likovi za obogaćenje podmorja kako bi igrica bila što realističnija, ali s obzirom da postoje već gotovi modeli životinja s ugrađenim ponašanjima, (Low Poly Animated Animals) nema smisla pisati skripte i primjenjivati modele ispočetka. Iako se danas umjetna inteligencija pokušava inkorporirati u gotovo sve, ravnoteža između cijene i koristi u ovom slučaju nije povoljna. Dakako, ako se pokaže prilikom razvoja potreba, postoje pokazni video materijali i opširna dokumentacija s kojima je to moguće.

7.2.2. Evaluacija proširene stvarnosti

Uspješna implementacija odabrane scene za demonstraciju proširene stvarnosti u *Unityju* pokazuje da je implementacija takve scene uistinu moguća. *Unity* se ističe kao platforma s dobrom podrškom za rad s proširenom stvarnosti. Nudi mnoštvo resursa (forumi, videozapisi, objave na blogovima...) što olakšava rješavanje problema na koje se naiđe tijekom razvoja. Unity podržava i mnoge druge mogućnosti za rad s proširenom stvarnosti koje prototip igre *Aqua Saviours* ne iskorištava, npr. prepoznavanje slike ili prepoznavanje lica. Također, u *Unity Asset Store*u dostupno je mnoštvo 3D modela koji se mogu iskoristiti za rad s proširenom stvarnosti.

7.2.3. Evaluacija sustava za prikupljanja korisničkih podataka za evaluaciju učinka

Istraživanje i izrada prototipa za prikupljanje korisničkih podataka pokazala je kako je moguće relativno lako proširiti bilo koju Unity igru s podrškom za prikupljanje razne vrste korisničkih interakcija. Razlog tomu je što se u posljednje vrijeme razvijaju standardi namijenjeni baš u ovu svrhu (xAPI) i usporedno s tim niču nova programska rješenja otvorenog koda izgrađena baš oko ovih standarda. No, moramo spomenuti da postoje ograničenja. Naime, iako postoji veliki broj *Learning Record Store-ova* manji broj njih je otvorenog koda i obično je ograničenih mogućnosti za razliku od njihovog plaćenog pandana. Uz to mnogi od njih se ne održavaju aktivno posljednjih godina.

U ovom radu je pokazano da je unatoč ograničenjima moguće uspješno podignuti sustav za prikupljanje korisničkih podataka. Ovi podaci zasigurno mogu pomoći istraživačima i programerima dobiti dragocjene uvide u učinkovitost igre Aqua Saviours pošto u inicijalnom prototipu nije predložena nikakva vrsta evaluacije učinka igre na poticanje ekološki osviještenog ponašanja kod korisnika.

Bitno je naglasiti da je ključni dio ovakve vrste evaluacije modeliranje metoda analize i odabir korisničkih interakcija koji će se prikupljati za analizu kako bi se dobili što je više moguće objektivni rezultati. Sam odabir parametara koji se prate ovisi o ozbiljnoj igri, a metoda analize ovisi o odabranim parametrima.

7.2.4. Evaluacija sustava dijaloga u *Unity* razvojnom okruženju

Izrada sustava dijaloga u *Unity* razvojnom okruženju pokazala je da korištenje alata poput *Dialogue 2* i *Yarn Spinner* može značajno unaprijediti interaktivnost u igri Aqua Saviours. Oba alata omogućuju razvoj kompleksnih dijaloga koji su ključni za prenošenje edukativnog sadržaja igračima. Također, i *Dialogue 2* i *Yarn Spinner* su se pokazali jednostavni za implementaciju dijaloga te oboje omogućuju i integraciju s ostalim elementima igre kao što su zvučni i vizualni prikazi. Međutim, korištenjem *Dialogue 2* alata neke mogućnosti, poput automatskog dodanja teksta dijaloga iz datoteke, ostaju ograničene jer takav API ne postoji unutar alata. Pisanjem skripti, kao u slučaju korištenja alata *Yarn Spinner*, to bi bilo mogće. Uzimajući u obzir potrebe igre *Aqua Saviours* oba alata su se pokazala kao efikasna. Za daljnji razvoj igre *Aqua Saviours*, važno je razmotriti kako integrirati ove sustave dijaloga na način koji će najbolje podržati edukativne ciljeve igre. To uključuje ne samo tehničku implementaciju, već i kreativno osmišljavanje dijaloga koji će informirati, motivirati i angažirati igrače.

Izbor između *Dialogue 2* i *Yarn Spinnera* ovisi o specifičnim potrebama i ciljevima igre *Aqua Saviours*. Ako je fokus na jednostavnijoj implementaciji i integraciji s raznim elementima igre, *Dialogue 2* može biti prikladniji izbor. Međutim, za igre koje zahtijevaju složeniju narativnu strukturu i fleksibilnost u kreiranju dijaloga, *Yarn Spinner* pruža potrebnu funkcionalnost i prilagodljivost.

7.2.5. Evaluacija integracije *Unity* projekta u *Flutter* aplikaciju

Integracija *Unity* projekta u *Flutter* aplikaciju uspješno je obavljena te je izrađen prototip integracije kako bi se na praktičnom primjeru prikazao razvoj takvog projekta. U nastavku su navedene prednosti i mane uočene tijekom razvoja.

Prednosti integracije *Unity* projekta u *Flutter* aplikaciju

- Višeplatformski razvoj
    - brži i jeftiniji razvoj, jednostavnije održavanje u vidu višeplatformske potpore, širi doseg publike i slično
- Proširenje Flutter aplikacije s Unity mogućnostima
    - fleksibilnost Flutterovog korisničkog sučelja uz Unity snažne grafičke i 3D mogućnosti
- Aktivne zajednice obiju platformi
    - brojni resursi, upute, edukativni sadržaj i podrška prilikom razvoja i rješavanja problema
- Proširen spektar dodataka
- Proširenje postojećih Flutter aplikacija s postojećim ili novim Unity projektima
    - Unity dio aplikacije ne mora nužno biti bit same aplikacije nego može služiti kao i dodatak koju aplikacije pruža.
    - Unity projekt koji se integrira može biti jedan od već postojećih
- Fleksibilnost UI-a
- Pogodno za razvoj igre unutar aplikacije
    - Sama smisao ovakve integracije omogućuje ostvarivanja koncepta razvoja aplikacije koja unutar svojih funkcionalnosti nudi igranje jedne ili više igara
- Brži razvojni ciklus
    - raspodjela razvojnog tima na razvoj Flutter aplikacije i Unity projekta
    - paralelno razvijanje
    - Razvoj kompleksnih 3D scena ili igara brže je razviti u Unity-ju zatim integrirati u aplikaciji
- Pogodno za razvoj edukativnih aplikacija

- aplikacije koja zahtijevaju interaktivne i vizualno bogate elemente te napredno korisničko sučelje, navigaciju i slično
- aplikacije raznovrsnih mogućnosti i načina prenošenja edukativnih sadržaja, od kojih su neki pomoću Unity grafičkih scena i/ili igara

Mane integracije *Unity* projekta u *Flutter* aplikaciju

- Posjedovanje znanja obiju tehnologija
  - bez iskustva korištenja i razvoja unutar obje tehnologije ovakav projekt postaje gotovo nemoguć
- Unošenje promjena kod integrirane aplikacije
  - kod donošenja promjena, testiranje i razvoj traju duže nego u zasebnim projektima
  - potrebno je izvesti Unity projekt, a zatim pokrenuti Flutter aplikaciju (kod promjena koje su dio integracije)
- Pronalaženje grešaka u kodu
  - greške unutar jednog ili oba projekta
  - teži pronalazak izvora greške
  - sintaksne greške su teže za pronaći, iako su vrlo jednostavne
- Mogući pad performansi
  - radi integracije dva različita okvira i načina njegovih funkcionalnosti
  - ovisno o kompliciranosti projekata
- Složenost održavanja
  - ponekad je održavanje dosta složenije u usporedbi s projektom koji je razvijen na temelju jedne od tehnologija
  - održavanje jednog projekta može utjecati na interakciju s drugim projektom, iako se logički potrebne promjene održavanja ne odnose na drugi projekt
- Izazovi integracije
  - Integriranje Unity projekta nije jednostavan proces i mogu se pojaviti neočekivani problemi
  - Neki od problema nisu dobro dokumentirani
- Korištenje emulatora pri razvoju

    - Korištenje emulatora se ne preporučuje jer se često javljaju greške
    - Podrška *flutter_unity_widget-a* je ograničena za emulator
    - Kod korištenja emulatora, podržanost ovisi o procesoru računala na kojem se emulator pokreće
- Potreba za fizičkim uređajem za testiranje tijekom cijelog razvoja
- Podržane Unity verzije
    - *flutter_unity_widget* podržava Unity 2019.4.3 ili noviji
- Ograničenja obiju tehnologija
    - Moraju se poštivati zahtjevi i ograničenja obiju tehnologija prilikom razvoja
    - Ograničenja koja su proizašla iz korištenja nekih od biblioteka jedne tehnologije mogu utjecat na drugu
    - Ograničenja tehnologija za uređaj na koji se aplikacija instalira (kao što je minimalni API)

Integracija *Unity* projekta u *Flutter* aplikaciju donosi niz prednosti koje se očituju u višeplatformskom razvoju, proširenju Flutter aplikacije s Unity mogućnostima fleksibilnosti UIja i slično. Osim toga, omogućuje brži razvojni ciklus, proširen spektar dodataka te pogoduje razvoju igara i edukativnih aplikacija unutar aplikacije.

S druge strane, postoji nekoliko izazova i mana koje proizlaze iz poznavanja obe tehnologije, unošenja promjena u integriranu aplikaciju, potencijalnog pada performansi, složenosti održavanja te specifičnih izazova vezanih uz integraciju Unity projekta.

Unatoč tim izazovima, razvoj takvog projekta može biti izuzetno koristan i učinkovit, pod uvjetom da se pažljivo procijene potrebe projekta i da se uloži trud u prevladavanje specifičnih tehničkih izazova. Upravljanje ovim izazovima zahtijeva stručnost i pažljivo planiranje kako bi se iskoristile prednosti obje tehnologije i pružila korisnicima visokokvalitetna i funkcionalna integrirana aplikacija.

7.3. Ocjena izvedivosti i smjernice za budući rad

Detaljnim objašnjenjem implementacije pojedinih dijelova igre *Aqua Saviours* u poglavlju *6* Implementacija programskog rješenja, pokazano je da: da, igra je izvediva, ali pitanje je li igra izvediva drugačije je od pitanja treba li igra biti izvedena.

Već pri evaluaciji prototipa jasno je da u igri ima mjesta za napredak, postoje dijelovi na kojima bi trebalo poraditi prije nego li se krene u razvoj igre. Najveći problem predstavlja manjak edukacijskog sadržaja. Dakle, svakako postoji prostor za njegovo unaprjđenje kako bi igra postala učinkovitija u podučavanju o zaštiti mora. Suradnja sa stručnjacima poput morskih biologa ili ekologa mogla bi pridonijeti autentičnosti igre pružanjem točnih informacija o morskom životu i strategijama očuvanja. Ova vrsta partnerstva osigurala bi da igra odražava stvarne izazove i rješenja u području ekologije mora. Integracija primjera situacija iz stvarnog svijeta pružila bi igračima konkretna iskustva ekoloških problema i njihovih rješenja omogućujući im praktično razumijevanje izazova povezanih s onečišćenjem mora. Nadalje, uvođenje različitih morskih ekosustava unutar igre proširilo bi edukativni opseg što bi pomoglo igračima da bolje razumiju složenost i važnost očuvanja različitih morskih okoliša.

Što se tiče korištenja novih tehnologija, vrlo je ograničeno. U prototipu igre, proširena stvarnost služi više kao „dekoracija“, nego što je ona ključni dio iskustva igranja igre. Kad bi se stvarno krenulo u razvoj s igrom Aqua Saviors, trebalo bi identificirati u kojim trenutcima u igri proširena stvarnost može donijeti dodatnu vrijednost. To može uključivati mogućnosti poput obogaćivanja okoline 3D modelima, dodavanja interaktivnih elemenata ili čak stvaranja mini-igara koje bi igrači trebali riješiti korištenjem AR tehnologije. Također bi se mogle iskoristiti i druge mogućnosti proširene stvarnosti poput detekcije slika ili detekcije lica i pokreta. Stoga, važno je razmotriti preinake ili razvoj potpuno novih igara koje bi stvarno iskoristile prednosti proširene stvarnosti.

Nadalje, integracija umjetne inteligencije ne bi imala veliki učinak za poboljšanje igre Aqua Saviours ili igračkog iskustva jer je primjena modela jako kompleksna, a obujam igre ne zahtjeva korištenje naprednih tehnika umjetne inteligencije za stvaranje neigrivih likova, praćenja igrača i analiziranje. Ovom studijom pokazalo se da je primjena izvediva, ali kao budući rad ostaje pitanje kako primijeniti umjetnu inteligenciju da ona bude i korisna. Bilo bi potrebno osmisliti drugačiji pristup igri kako umjetna inteligencija ne bi sama sebi postala

svrhom i kako bi stvorili intrizičnu vrijednost igre njezinom primjenom. Kada bi se pratile sve ove smjernice, igra *Aqua Saviours* mogla bi postati dobar alat za edukaciju o onečišćenju oceana.

Zaključak

Prototip igre *Aqua Saviours* razvijen je u sklopu hackathona održanog 2023. godine u Dubrovniku kojeg je organizirao *Play2Green*. Ozbiljna igra *Aqua Saviours* usmjerena je na podizanje svijesti o onečišćenju oceana te pokušava iskoristiti nove tehnologije kako bi to učinkovito učinila. Korištenjem proširene stvarnosti, umjetne inteligencije te praćenja napretka korisnika, *Aqua Saviours* korisnicima pruža interaktivno iskustvo učenja.

U provedenoj studiji izvedivosti istraženi su ključni aspekti implementacije ozbiljne igre *Aqua Saviours* unutar *Unity* platforme. Studija izvedivosti obuhvatila je istraživanje proširene stvarnosti, umjetne inteligencije, praćenja napretka korisnika, integracije s radnim okvirom *Flutter* te evaluaciju edukativnog sadržaja. Dokumentiran je način implementacije svake komponente te prednosti i nedostaci prototipa.

Iako igra nastoji ukomponirati nove tehnologije i prenijeti edukativni sadržaj, to čini vrlo površno. Nedostaje joj edukativnog sadržaja te treba na bolji način iskoristiti nove tehnologije za prenošenje znanja. Osmišljavanje inovativnijih načina korištenja novih tehnologija i dodavanje dodatnog edukativnog sadržaja zato predstavlja ključne korake za daljnji razvoj igre. Ova poboljšanja ne bi samo unaprijedila korisničko iskustvo već bi i dodatno povećala edukativnu moć igre.

Literatura

[1] Abbing, M.R., *Plastic Soup, An Atlas of Ocean Pollution*, Island Press, 2019.

[2] Brannon, B., *Discover Ocean Pollution*, Benchmark Education Company, 2005.

[3] Sindermann, C.J., *Pollution, Effects on Living Resources and Humans*, CRC Marine Science, 1995.

[4] Botte, S.E.; Arias A.H., *Coastal and deep ocean pollution*, CRC Press, 2020.

[5] Clark, R.B., *Marine pollution*, Oxford University Press, 2001.

[6] Grendel Games, *What are serious games?*, Grendel Games, poveznica: https://grendelgames.com/what-are-serious-games/, pristupljeno: 8. siječnja 2024.

[7] Boncu, S.; Candel, O.S.; Popa, N.L. *Gameful Green: A Systematic Review on the Use of Serious Computer Games and Gamified Mobile Apps to Foster Pro-Environmental Information, Attitudes and Behaviors,* Sustainability 2022. https://doi.org/10.3390/su141610400

[8] Babić, J.; Podobnik, V.; Složić, I.; Zekić, I.; Gourves-Hayward, A.; Sable, C.; Papp, I.; Zichar, M.; Hernandez F. C.; Peñaranda F. F.; Kešelj, A.; Zakarija, I.; Žubrinić, K.*, A report on the potential of green serious games in higher education,* 2023, https://urn.nsk.hr/urn:nbn:hr:168:305313

[9] *Learning Record Store*, xAPI, poveznica: https://xapi.com/learning-record-store/?utm_source=google&utm_medium=natural_search, pristupljeno: 1. studenog 2023.

[10] Torrance M., *What is xAPI?*, TD, 2021, svibanj. poveznica: https://www.leadinglearning.com/episode-319-megan-torrance-xapi/, pristupljeno: 15. studenog 2023.