Pretvaranje teksta u sliku u proširenoj stvarnosti

Rücker, Kristian

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:473812

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-22**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PRETVARANJE TEKSTA U SLIKU U PROŠIRENOJ STVARNOSTI

Završni rad

Kristian Rücker

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 05.07.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Kristian Rücker
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4131, 28.07.2017.
OIB Pristupnika:	36840122741
Mentor:	izv. prof. dr. sc. Časlav Livada
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Pretvaranje teksta u sliku u proširenoj stvarnosti
Znanstvena grana rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak završnog rad:	U radu je potrebno napraviti mobilnu aplikaciju koja će koristiti Stable Diffusion biblioteku za dodavanje slika opisanih tekstom u proširenu stvarnosti. Primjer: https://twitter.com/bjoernkarmann/status/1590648681169985537
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	05.07.2023.
Datum potvrde ocjene od strane Odbora:	12.07.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.07.2023.

Ime i prezime studenta:	Kristian Rücker
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4131, 28.07.2017.
Turnitin podudaranje [%]:	14

Ovom izjavom izjavljujem da je rad pod nazivom**: Pretvaranje teksta u sliku u proširenoj stvarnosti**

izrađen pod vodstvom mentora izv. prof. dr. sc. Časlav Livada

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD *1*

1.1. Zadatak završnog rada **2**

2. PREGLED PODRUČJA RADA ZA STABLE DIFFUSION U PROŠIRENOJ STVARNOSTI *3*

3. STABLE DIFFUSION U PROŠIRENOJ STVARNOSTI *5*

3.1. Stable diffusion **5**

3.1.1. Primjena Stable Diffusion-a 6

3.1.2. Duboko učenje 6

3.1.3. Model pretvaranja teksta u sliku 7

3.1.4. Umjetna neuronska mreža 7

3.1.5. Problemi Stable Diffusion-a 9

3.2. Proširena stvarnost **9**

3.2.1. Razlike između proširene i virtualne stvarnosti 9

3.2.2. Primjena proširene stvarnosti 11

4. IZRADA APLIKACIJE *16*

4.1. Pretvorba govora u tekst **19**

4.2. Snimanje AR zaslona **22**

4.3. Stable Diffusion u AR aplikaciji **23**

5. OPIS I TESTIRANJE RADA MOBILNE APLIKACIJE *36*

6. ZAKLJUČAK *41*

LITERATURA *42*

SAŽETAK *45*

ABSTRACT ..... *Error! Bookmark not defined.*

ŽIVOTOPIS ..... *Error! Bookmark not defined.*

1. UVOD

Napredovanjem moderne tehnologije razvijaju se kvalitetnije opcije snimanja sadržaja u proširenoj stvarnosti (engl. *Augmented Reality* – AR). U posljednjih nekoliko godina, AR postaje sve popularniji u poslovnom svijetu. Neke od danas najpopularnijih mobilnih aplikacija uključuju AR, kao npr. *Snapchat* filteri i *Pokémon Go*. Razvojem moderne tehnologije razvijaju se i mogućnosti primjene umjetne inteligencije, koja u današnje vrijeme predstavlja jako veliki napredak u znanosti i tehnologiji. Putem nje dolazi se do lakših i alternativnih rješenja u određenim područjima ljudskog života. Tijekom zadnjih godina, između ostalog, umjetna inteligencija koristi se i za izradu modela za pretvaranje teksta u sliku. Takav model koristan je alat, ali kao takav ne razumije jezik ljudi, pa to ipak zahtjeva čovjekovo upravlja njime. Primjer takvog modela je StaD (engl. *Stable Diffusion*).

Tema ovog rada je pretvaranje teksta u sliku u proširenoj stvarnosti. Pretvaranje je prikazano pomoću stable diffusion biblioteke u AR-u unutar mobilne aplikacije. Za pretvaranje teksta u sliku unutar StaD modela, koriste se određeni tekstualni upiti putem kojih se želi postići što točnija vizualizacija željenog sadržaja.

Motivacija za izradu ovoga rada je ta što u trenutku pisanja ovog rada ne postoje mobilne aplikacije za snimanje sadržaja u proširenoj stvarnosti, odnosno ne postoji niti jedna mobilna aplikacija koja tijekom snimanja okoline omogućuje dodavanje željenog nestvarnog sadržaja unutar videozapisa. Za izradu mobilne aplikacije, odabran je za korištenje StaD jer je za razliku od prijašnjih modela pretvorbe teksta u sliku, kao što su DALL-E [1] i Midjourney [2], javno dostupan i to ne samo putem oblaka.

Ovaj rad sastoji se od pet poglavlja. U prvom poglavlju dan je uvod u temu završnog rada. U drugom poglavlju opisuju se osnovni principi StaD modela i njegove funkcionalnosti. Također u drugom poglavlju nalazi se opis AR-a. U trećem poglavlju opisan je StaD model u AR-u. U četvrtom poglavlju opisan je postupak izrade aplikacije. U petom poglavlju opisan je i testiran rad aplikacije. Posljednje poglavlje sadrži zaključke proizašle iz cijelog rada.

1.1. Zadatak završnog rada

U radu je potrebno napraviti mobilnu aplikaciju koja će koristiti Stable Diffusion biblioteku za dodavanje slika opisanih tekstom u proširenu stvarnost.

2. PREGLED PODRUČJA RADA ZA STABLE DIFFUSION U PROŠIRENOJ STVARNOSTI

Generiranje slika pomoću umjetne inteligencije relativno je novo tehnološko postignuće u području računalnog vida i umjetne inteligencije, čijim korištenjem omogućeno je da se računalnim putem stvaraju realistične slike temeljene na uzorcima i informacijama naučenim iz velikih skupova podataka. Kroz duboko učenje i neuronske mreže, generativni modeli mogu naučiti kako stvarati slike koje izgledaju autentično i mogu se koristiti u različitim područjima, kao što su dizajn, umjetnost, filmska industrija i mnogim drugim. Za generiranje slika najčešće se koristi javno dostupan model Stable Diffusion. Iako postoje izazovi i ograničenja pri njihovoj izradi, generiranje slika pomoću umjetne inteligencije pokazuje veliki potencijal za inovacije i kreativnost u budućnosti. Neki od izazova s kojima se korisnici Stable Diffusion-a susreću su tehničke naravi, kao što su potrebna količina resursa na računalu ili brzina generiranja slike. Ti problemi obrađeni su u znanstvenom radu D. Bolya i J. Hoffman "Token Merging for Fast Stable Diffusion" [3].

Brzim napretkom umjetne inteligencije, sve bolja kvaliteta sadržaja generiranog umjetnom inteligencijom otvorila je nove mogućnosti primjene takvog sadržaja u AR-u. Spajanje sustava proširene stvarnosti i modela StaD, je koncept koji je zaživio u prošlih godinu dana. Ideja tog koncepta objašnjena je u radu: "Exploring the Design Space of Employing AI-Generated Content for Augmented Reality Display" [4].

Ovaj rad istražuje načine na koje se AI-generirani sadržaj može koristiti u kontekstu AR-a kroz primjenu modela Stable Diffusion. Kombiniranjem AR-a i StaD-a, istraživači su demonstrirali kako AI-generirani sadržaj može biti integriran u stvarni svijet na način koji je dosad bio nedostižan.

Primjena AI-generiranog sadržaja u AR-u otvara vrata za brojne mogućnosti, npr. korisnici mogu doživjeti obogaćene virtualne objekte u svojoj stvarnoj okolini poput umjetničkih instalacija, virtualnih likova ili interaktivnih igara. Također, AI može pomoći u automatskom generiranju sadržaja za AR aplikacije, što olakšava dizajniranje sadržaja na način da više nije neophodno ručno oblikovanje svakog detalja.

Ova nova kombinacija AR-a i AI-generiranog sadržaja ima potencijal za široku primjenu u različitim područjima znanosti i tehnike. Može se koristiti u marketingu i oglašavanju kako bi se

stvorili privlačni vizuali i dojmljive interakcije s proizvodima. U edukaciji, AI-generirani sadržaj u AR-u može poboljšati iskustvo učenja i omogućiti studentima da interaktivno istražuju složene koncepte. Također, ova tehnologija može imati primjene u medicini, arhitekturi, turizmu i mnogim drugim područjima.

3. STABLE DIFFUSION U PROŠIRENOJ STVARNOSTI

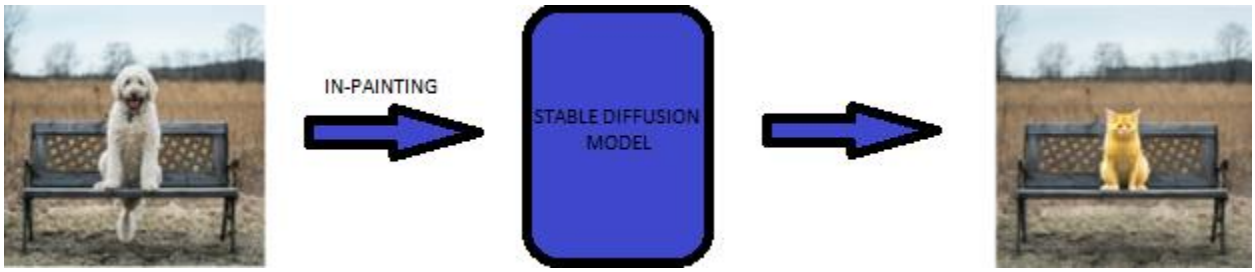
3.1. Stable diffusion

Stable diffusion ili StaD je model pretvaranja teksta u sliku temeljen na dubokom učenju. Napravljen je 2022. godine i razvila ga je start-up firma Stability AI u suradnji s nizom akademskih istraživača i neprofitnih organizacija [5]. Prvenstveno se koristi za generiranje detaljno definiranih slika putem tekstualnih opisa ili upita. Također, StaD se primjenjuje i na drugim zadacima kao što su dopunjavanje nedostajućih i oštećenih dijelova slike, kojeg mogu koristiti u radu restauratori umjetničkih djela, kao i mijenjanje postojećih elemenata u slici (engl. *inpainting*), izdvajanje dijelova slike iz njihove pozadine (engl. *outpainting*) ili pretvaranje iz slike u sliku. Na slici 3.1 prikazan je primjer StaD generirane slike na temelju sljedećeg upita: *"A Hyperrealistic photography of ancient Tokyo/London/Paris architectural ruins in a flooded apocalypse landscape of dead skyscrapers, lens flares, cinematic, hdri, matte painting, concept art, celestial, soft render, highly detailed, cgsociety, octane render, trending on artstation, architectural HD, HQ, 4k, 8k*".

Slika 3.1. Prikaz Stable Diffusion generirane slike [5]

3.1.1. Primjena Stable Diffusion-a

StaD se koristi u raznim poslovnim područjima [4]. U internet trgovini, umjesto snimanja proizvoda iz različitih kutova s različitim pozadinama, pomoću StaD-a dovoljno je nekoliko slika kako bi se proizvod *inpainting*-om stavio ispred različitih pozadina i unutar različitih konteksta. Mnoge aplikacije za uređivanje slika koriste *inpainting* kao jednu od svojih temeljnih značajki za prikaz slika. Na slici 3.2 prikazan je primjer StaD *inpainting* generirane slike na temelju sljedećeg upita: *"Face of a yellow cat, sitting on a park bench"*.



Slika 3.2. Prikaz Stable Diffusion *inpainting* generirane slike [6]

U modi StaD se može koristiti za promjenu odjeće koju modeli nose. U web dizajnu pomoću StaD-a moguće je kombiniranjem različitih elemenata i variranjem boja ostvariti zanimljiva rješenja u izgledu web-stranica. U industriji baziranoj na izradi i prodaji video igara (engl. *gaming* industriji), StaD se upotrebljava za stvaranje realističnih pozadinskih slika kako bi dešavanja pri igri izgledala što vjernije stvarnosti (engl. *asset*). Umjetniku bi trebalo nekoliko tjedana, ako ne i mjeseci, da stvori *asset*-e slične kvalitete.

3.1.2. Duboko učenje

Tehnologija strojnog učenja pokreće mnoge segmente modernog društva, od internet pretraživanja, filtriranja sadržaja na društvenim mrežama do preporuka na stranicama internet trgovine. Sve je prisutnija u potrošačkim proizvodima kao što su kamere i pametni telefoni. Sustavi strojnog učenja koriste se za prepoznavanje objekata na slikama, prevođenje govora u tekst, spajanje vijesti, objava ili prilagođavanje proizvoda interesima korisnika i odabir relevantnih rezultata pretraživanja. [7]

Duboko učenje je potpodručje strojnog učenja koje se fokusira na osposobljavanje umjetnih neuronskih mreža za obavljanje složenih zadataka kao što su prepoznavanje slike i govora, obrada prirodnog jezika i donošenje odluka. Pojam „duboko" u dubokom učenju odnosi se na broj slojeva u neuronskoj mreži. Što mreža ima više slojeva, to su funkcije koje bi mogla

obavljati složenije primjenjujući algoritme koji simuliraju rad ljudskog mozga i time omogućujući računalima da uče iz velikih količina podataka, te donose predviđanja ili odluke na temelju tih podataka. Neki od popularnih algoritama dubokog učenja uključuju neuronskih mreža za pretprocesiranje nestrukturiranih podataka, tj. konvolucijske neuronske mreže (engl. *convolutional neural network* – CNN), neuronske mreže koje u sebi sadrže petlju, tj. rekurentne neuronske mreže (engl. *recurrent neural network* – RNN) i neuronske mreže koje koriste nenadzirani model strojnog učenja za dobivanje rezultata, tj. mreže dubokog uvjerenja (engl. *deep belief network* – DBN). Duboko učenje revolucioniralo je polje umjetne inteligencije, dovodeći do otkrića u računalnom vidu, prepoznavanju govora i drugim područjima. Ova otkrića primijenjena su u zdravstvu, financijama, transportu itd., za rješavanje složenih problema i za poboljšanje učinkovitosti sustava. [7]

3.1.3. Model pretvaranja teksta u sliku

Model pretvaranja teksta u sliku model je strojnog učenja koji kao ulaz uzima opis prirodnog jezika i kao izlaz generira sliku koja odgovara tom opisu. Takvi su se modeli počeli razvijati sredinom 2010-ih godina kao rezultat napretka na području dubokih neuronskih mreža. U 2023. godini rezultati najsuvremenijih modela pretvaranja teksta u sliku, poput DALL-E 2, Imagen-a i StaD-a, počeli su se približavati kvaliteti pravih fotografija i umjetničkih slika koje su izradili ljudi. [8]

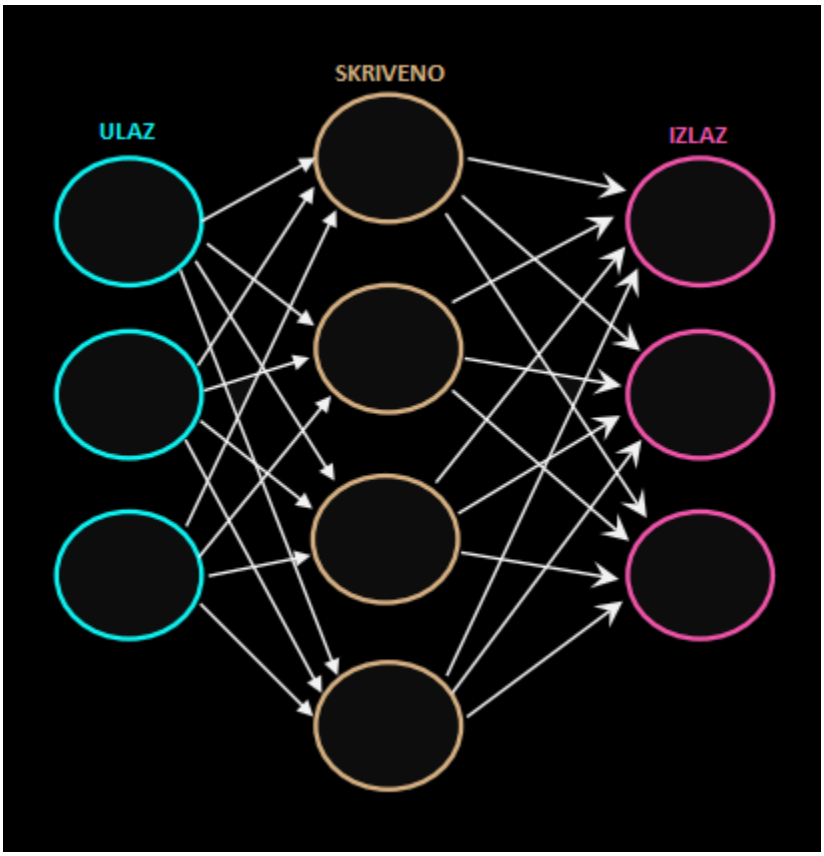
Modeli pretvaranja teksta u sliku općenito kombiniraju jezični model koji transformira ulazni tekst u njegovu reprezentaciju i model generativne slike koji proizvodi sliku uvjetovanu tom reprezentacijom. Najučinkovitiji modeli općenito su uvježbani na golemim količinama slikovnih i tekstualnih podataka sakupljenih s interneta.

3.1.4. Umjetna neuronska mreža

Umjetne neuronske mreže (engl. *artificial neural network* – ANN), koje se obično jednostavno nazivaju neuronske mreže (engl. *neural network* – NN), računalni su sustavi inspirirani biološkim neuronskim mrežama koje čine mozgove živih bića [9]. ANN se sastoje od umjetnih neurona koji su konceptualno izvedeni iz bioloških neurona. Svaka veza, poput sinapsi u biološkom mozgu, može prenijeti signal drugim neuronima. Umjetni neuron prima signale, zatim ih obrađuje i može ih tako prerađene prenijeti neuronima povezanima s njim. Signal na

vezi je realan broj, a izlaz iz svakog neurona izračunava se nekom nelinearnom funkcijom zbroja njegovih ulaza. Te veze nazivaju se rubovi. Neuroni i rubovi obično imaju težinu koja se prilagođava ovisno o informacijama koje mreža posjeduje. Težina povećava ili smanjuje snagu signala na spoju.

Po uzoru na ljudski mozak, neuronska mreža sastoji se od nekoliko tisuća ili čak milijuna jednostavnih čvorova za obradu koji su međusobno gusto povezani. Većina današnjih neuronskih mreža organizirana je u slojeve čvorova u kojima se podaci kreću kroz njih u samo jednom smjeru (engl. *feed-forward)* [9]. Pojedinačni čvor može biti povezan s nekoliko čvorova u sloju ispod sebe, od kojih prima podatke, i nekoliko čvorova u sloju iznad sebe, kojima šalje podatke. Različiti slojevi mogu izvoditi različite transformacije na svojim ulazima. Signali putuju od prvog, ulaznog, sloja do posljednjeg, izlaznog, sloja. Svaki umjetni neuron ima ulaze i proizvodi jedan izlaz koji se može poslati na više drugih neurona. Ulazi mogu biti vanjski podatci, kao što su slike ili dokumenti, ili izlazi iz drugih neurona. Izlazi konačnih izlaznih neurona neuralne mreže ispunjavaju zadatak, kao što je prepoznavanje objekta na slici. Na slici 3.3 prikazana je struktura umjetne neuronske mreže.



Slika 3.3. Prikaz strukture umjetne neuronske mreže [10]

3.1.5. Problemi Stable Diffusion-a

Prilikom primjene StaD-a dolazi do raznih problema, a neki od njih su: manjak GPU memorije, proces generiranja velikih slika, deformirani oblici, brzi inženjering, problemi s autorskim pravima [6].

Najveća prepreka korištenju StaD-a u današnje vrijeme je nedostatak dovoljne memorije i GPU memorije. StaD ja namijenjen da radi na GPU karticama većim od 8 GB virtualne memorije. Zbog toga nastaje problem kada StaD treba generirati velike slike. Ovo je naravno povezano s problemima VRAM-a ili GPU memorije. Danas je gotovo nemoguće stvoriti 4K slike ili video pomoću StaD-a, što bi se trebalo popraviti u budućnosti. Sljedeći problem koji se javlja je drugačije definiranje osnovnih pojmova od onih kakvi su u prirodnom jeziku, te se zbog toga javljaju deformirani oblici tijekom postupka generiranja slike. Npr. ako se od modela traži da generira sliku ljudske šake, mogli biste vidjeti šaku sa šest prstiju ili četiri prsta. No ipak, u većini slučajeva neće doći do takvih deformacija. Dobra strategija za ublažavanje ovog problema je fino podešavanje modela. Brzi inženjering je koncept u umjetnoj inteligenciji kôd kojeg se predefinira koje riječi treba koristi za tekstualan upit. Na primjer, ako je cilj da slika izgleda vrlo stvarno, možda će se StaD modelu morati dodati ključne riječi kao što su *hyper-realistic*. Slike generirane StaD-om mogu imati problema s autorskim pravima, npr. u slučaju da se traži slika automobila koja će se koristiti kao asset ili u reklami, StaD može generirati sliku automobila s neželjenim Fordovim ili Teslinim logotipom zaštićenim autorskim pravima.

3.2. Proširena stvarnost

3.2.1. Razlike između proširene i virtualne stvarnosti

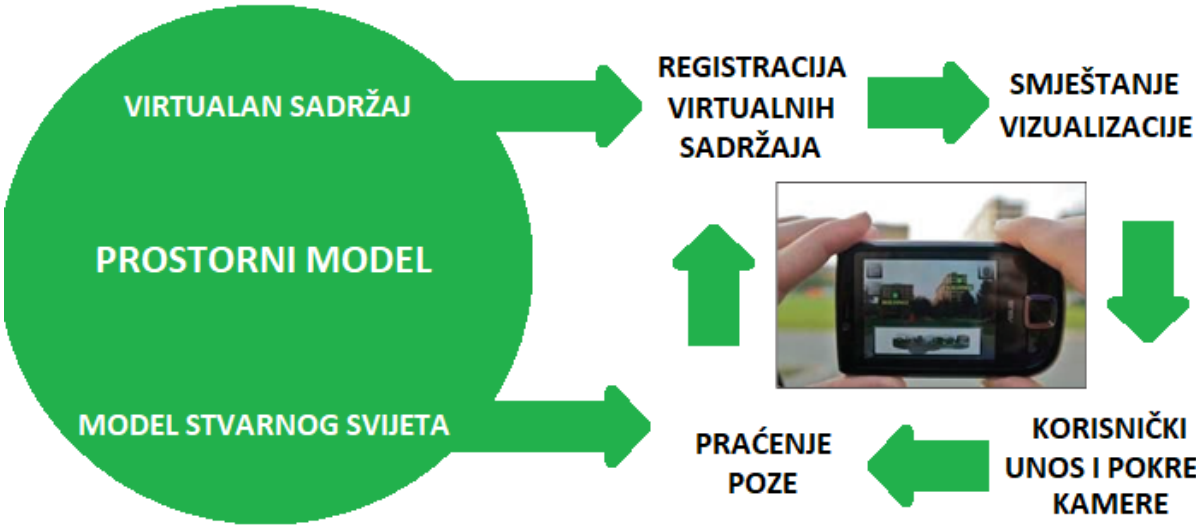
Kako mogućnosti računalne grafike napreduju, slike koje se putem nje generiraju često se ne mogu razlikovati od onih iz stvarnog svijeta, stoga virtualna stvarnost (engl. *Virtual Reality* – VR). postaje sve korištenija i popularnija. Međutim, računalno generirane slike prikazane npr. u igrama i filmovima, odvojene su od našeg fizičkog okruženja, što istovremeno ima i svoje prednosti i mane.

Dok VR smješta korisnika u potpuno računalno generirano okruženje, proširena stvarnost (engl. *Augmented Reality* – AR) prezentira informacije iz izravnog fizičkog okruženja korisnika [10]. AR nadilazi mobilno računarstvo jer prostorno i kognitivno spaja virtualni i stvarni svijet. Uz AR, digitalne informacije postaju dio stvarnog svijeta, barem u percepciji korisnika.

Najšire prihvaćenu definiciju AR-a predložio je američki znanstvenik Robert Azuma u svom znanstvenom radu "A Survey of Augmented Reality" [11] objavljenom 1997. godine. Po njemu, AR je sustav koji mora sadržavati sljedeće tri karakteristike:

- mora moći kombinirati virtualne elemente s onim iz stvarnog svijeta
- mora moći biti interaktivan u stvarnom vremenu
- mora omogućavati prikaz (registraciju) u trodimenzionalnom (3D) prostoru

Kompletan AR sustav zahtijeva četiri komponente: komponentu praćenja, komponentu registracije, komponentu vizualizacije i komponentu prostornog modela koji pohranjuje informacije o stvarnom i o virtualnom svijetu. Model iz stvarnog svijeta mora služiti kao referenca za komponentu praćenja, koja mora odrediti lokaciju korisnika u stvarnom svijetu. Model virtualnog svijeta sastoji se od sadržaja koji se koristi za proširenje. Oba dijela prostornog modela, iz stvarnog i iz virtualnog svijeta, moraju biti registrirana u istom koordinatnom sustavu kako bi se vizualizirali kao virtualni 3D modeli na nekoj stvarnoj lokaciji iz različitih kutova gledanja i u različitim položajima. Na slici 3.4 prikazan je način rada AR modela u kojemu AR koristi povratnu petlju između ljudskog korisnika i računalnog sustava. Korisnik promatra AR zaslon i kontrolira točku gledišta. Sustav prati korisnikovo gledište, registrira pozu u stvarnom svijetu s virtualnim sadržajem i predstavlja situirane vizualizacije.



Slika 3.4. Prikaz načina rada AR modela [12]

Proširena stvarnost obećava stvaranje izravnih, automatskih i djelotvornih veza između fizičkog svijeta i elektroničkih informacija [12] i omogućuje jednostavno i neposredno korisničko sučelje za elektronički poboljšani fizički svijet.

3.2.2. Primjena proširene stvarnosti

AR se već koristi u građevinarstvu, u medicini, tijekom kvalifikacijskih obuka, u *gaming* industriji, za navigaciju, za prikaz informacija od osobnog interesa, za oglašavanje i trgovinu i na televiziji [12].

Industrijski pogoni postaju sve složeniji, što bitno utječe na njihovo planiranje i rad. Arhitektonske strukture, infrastruktura i strojevi planiraju se pomoću softvera za računalno potpomognuto projektiranje (engl. *computer-aided design* – CAD), ali obično se mnoge izmjene rade tijekom stvarne izgradnje i instalacije. Te se izmjene obično ne vraćaju u CAD modele. Osim toga, može postojati velik broj naslijeđenih struktura koje prethode uvođenju CAD-a za planiranje, kao i potreba za čestim promjenama instalacija. Primjer korištenja AR tehnologije u građevinarstvu prikazan je na slici 3.5.

Slika 3.5. Prikaz korištenja AR tehnologije u građevinarstvu [13]

Korištenje rendgenskog snimanja revolucioniralo je dijagnostiku dopuštajući liječnicima da imaju uvid u zbivanja u unutrašnjost pacijenta bez izvođenja operacija. Međutim, konvencionalni rendgenski uređaji i uređaji za kompjutoriziranu tomografiju odvajaju unutarnji od vanjskog pogleda na pacijenta. AR integrira ove poglede, omogućujući liječniku da vidi kompletnu sliku pacijenta. Primjer korištenja AR tehnologije u medicini prikazan je na slici 3.6.

Slika 3.6. Prikaz korištenja AR tehnologije u medicini [14]

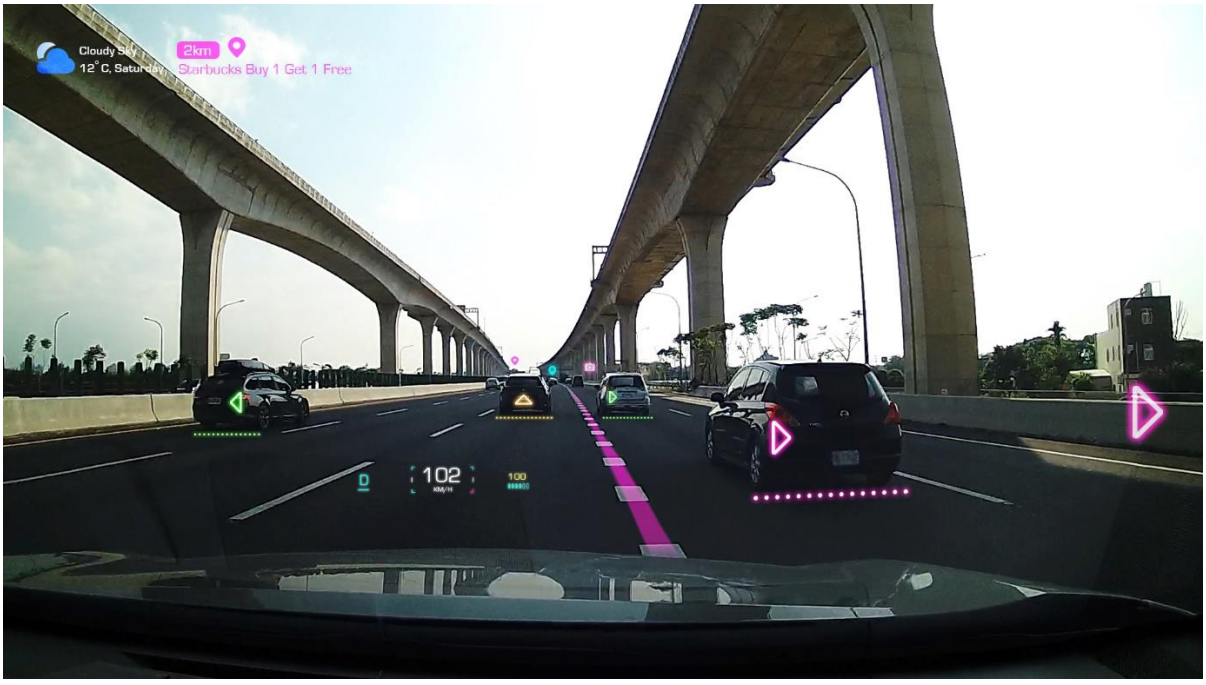
Razumijevanje kako objekti funkcioniraju i učenje kako ih sastaviti, rastaviti ili popraviti važan je izazov u mnogim profesijama. Inženjeri održavanja često odvajaju puno vremena za proučavanje priručnika i dokumentacije, jer je često nemoguće u potpunosti zapamtiti sve postupke i detalje složenih konstrukcija. AR može postaviti upute izravno u njihovo vidno polje, što obuku čini učinkovitijom. Primjer korištenja AR tehnologije pri postupku učenja prikazan je na slici 3.7.

Slika 3.7. Prikaz korištenja AR tehnologije kao postupak učenja u težim područjima inženjerstva [15]

Jedna od prvih komercijalnih AR igara bila je *The Eye of Judgment*, interaktivna kartaška igra za Sony PlayStation 3. Igra se isporučuje s kamerom iznad glave, koja hvata karte za igru i poziva odgovarajuće subjekte za borbu. Također, jedna od najuspješnijih AR igrica je *Pokémon*

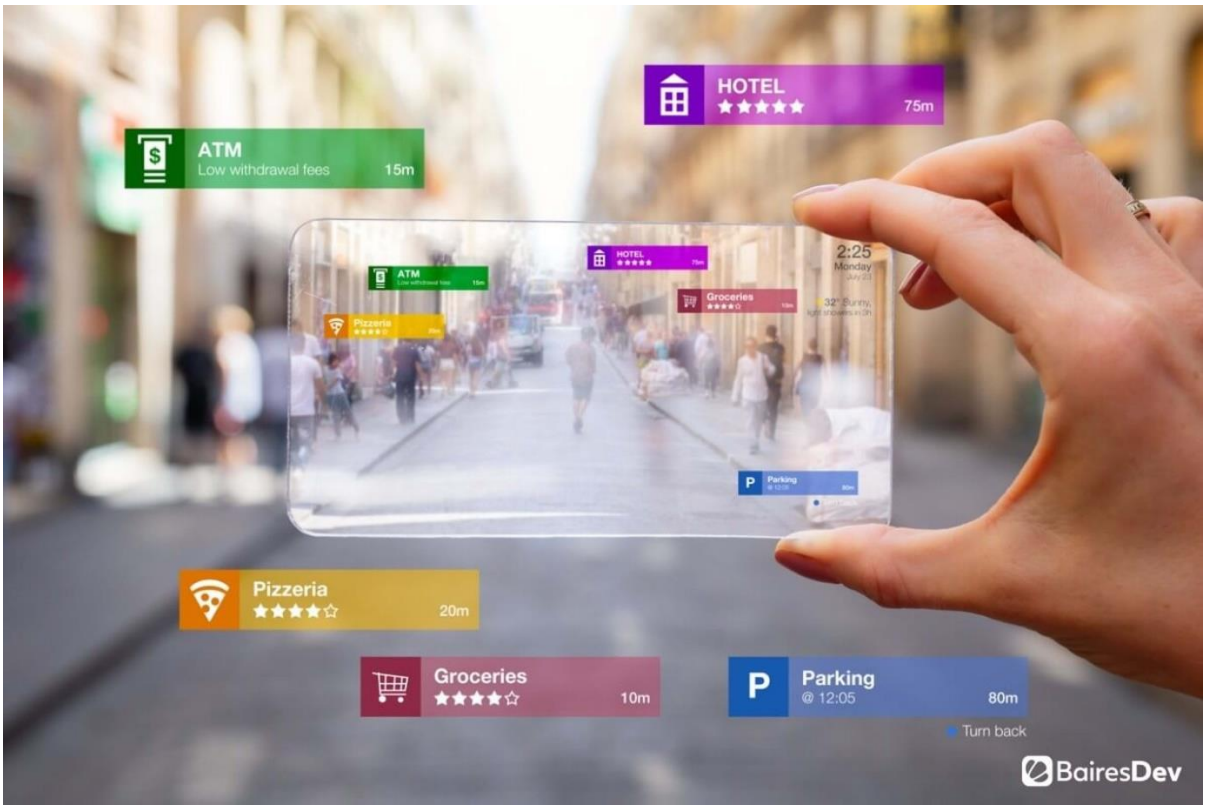
Go, bazirana na lovu na pokemone pomoću *Google Maps* navigacije. Pokemoni su vidljivi u okolini na kameri korisnika. Primjer korištenja AR tehnologije u gaming industriji prikazan je na slici 3.8.

Slika 3.8. Prikaz korištenja AR tehnologije unutar mobilne igrice *Pokémon Go* [16]



Slika 3.9. Prikaz korištenja AR tehnologije za *heads-up* navigaciju [17]

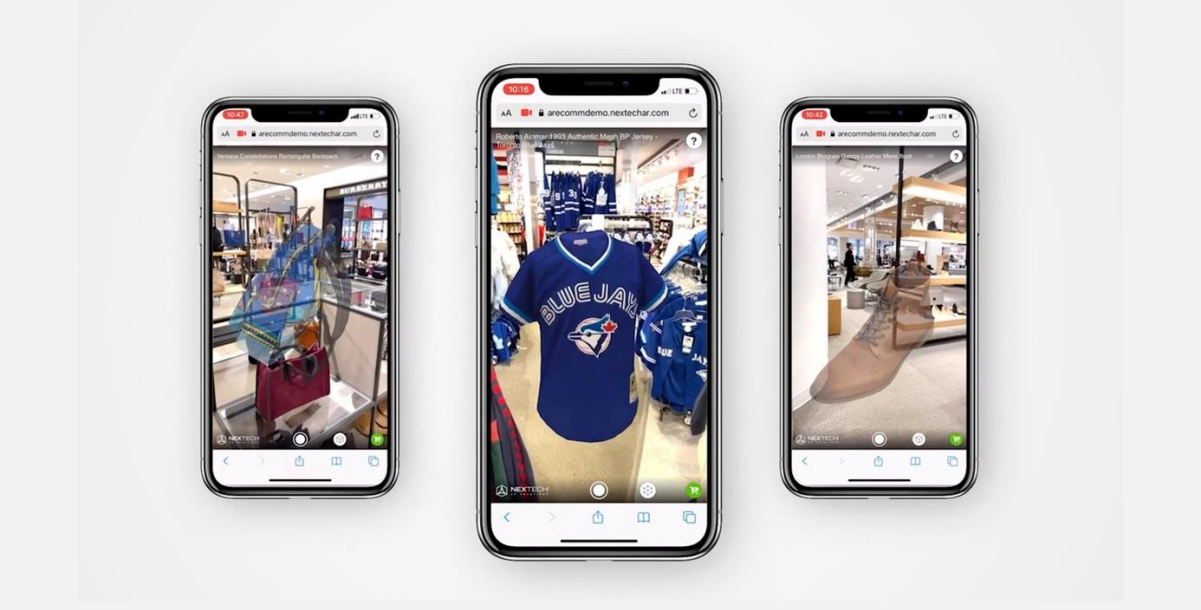
Na slici 3.9 prikazan je primjer korištenja AR tehnologije za *heads-up* navigaciju. Takozvana *heads-up* navigacija, koja ne odvraća pozornost operatera vozila koja se kreću velikim brzinama od njihove okoline, prvi put je razmatrana u kontekstu primjene od strane pilota vojnih zrakoplova. *Heads-up* zasloni su prozirni zasloni koji se mogu montirati na vizir pilotske kacige. Ovi uređaji su uglavnom namijenjeni za prikaz informacija, poput trenutne brzine ili pogonskog momenta zrakoplova, ali se također mogu koristiti za prikaz oblika putem AR-a.



Slika 3.10. Prikaz korištenja AR tehnologije za prikaz informacija od osobnog interesa [18]

Na slici 3.10 prikazan je primjer korištenja AR tehnologije za prikaz informacija od osobnog interesa. Neke mobilne AR aplikacije koristeći kamera uređaj namijenjene su pružanju informacija povezanih s mjestima interesa u korisnikovom okruženju. Mjesta interesa navedena su u geokoordinatama i identificirana putem GPS senzora mobilnog uređaja ili identificirana prepoznavanjem slike. Primjer takvih aplikacija su Layar [19], Wikitudes [20] i Junaio [21].

Sposobnost AR-a da predstavi izmjenjive 3D prikaze proizvoda potencijalnom kupcu već se primjenjuje u oglašavanju i trgovini i može dovesti do istinski interaktivnih iskustava pri kupnji. Primjer korištenja AR tehnologije za oglašavanje prikazan je na slici 3.11.



Slika 3.11. Prikaz korištenja AR tehnologije za oglašavanje proizvoda [22]

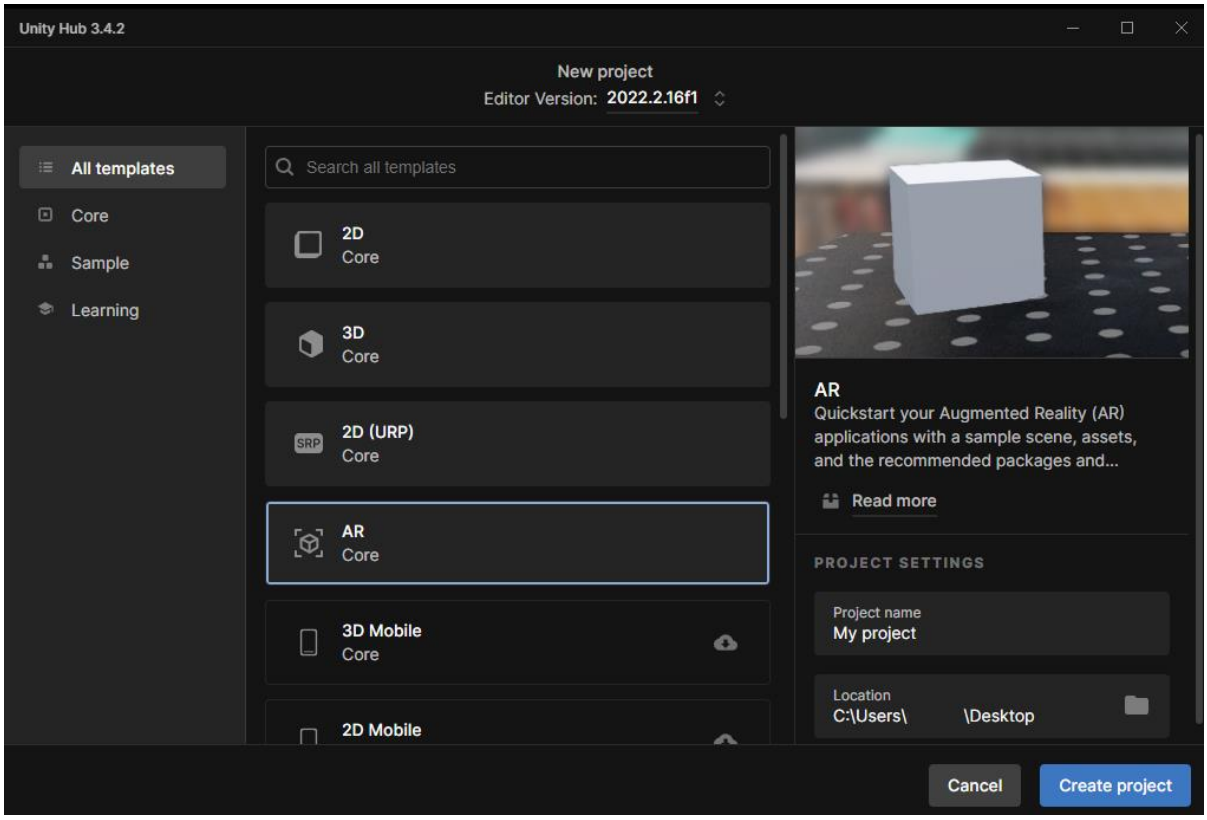
Mnogi su se vjerojatno po prvi put susreli s AR-om u vidu dodataka na snimkama kamera uživo, koje koristi TV. Prvi i najistaknutiji primjer ovog koncepta je virtualna prva i deseta linija u američkom nogometu, koja označava jarde potrebne za prvi *down*, koji se superponira izravno na TV zaslon utakmice. Isti koncept označavanja TV snimaka virtualnim slojevima uspješno je primijenjen na mnoge druge sportove, uključujući nogomet, bejzbol, hokej na ledu i automobilske utrke. Primjer korištenja AR tehnologije u sportu prikazan je na slici 3.12.

Slika 3.12. Prikaz korištenja AR tehnologije u nogometu [23]

4. IZRADA APLIKACIJE

Cilj ovog rada je napraviti mobilnu aplikaciju koja će koristiti StaD biblioteku za dodavanje slika opisanih tekstom u proširenu stvarnost. Zadatak rada odrađen je u programu Unity [24]. Unity je jedan od najpopularnijih i najmoćnijih višeplatformskih okruženja za izradu video igara, simulacija i drugih interaktivnih iskustava. Razvio ga je Unity Technologies i izdao u lipnju 2005. godine. Unity koristi vizualno programiranje i podržava C# kao glavni jezik programiranja. Za pisanje skripta C# programskog jezika u završnom radu koristi se Visual Studio 2022 [25]. Visual Studio 2022 je integrirano razvojno okruženje (engl. *Integrated Development Environment* – IDE) tvrtke Microsoft. Za potrebe pokretanja i korištenja mogućnosti StaD-a u ovome radu koristi se mrežno korisničko sučelje (engl. *User Interface* – UI) Stable Diffusion Automatic 1111 [26]. Automatic 1111 popularan je UI alat otvorenog kôda napravljen da pomogne u primjeni naprednih značajki.

Završni rad napravljen je koristeći AR predložak projekta, koji pruža početnu točku za razvoj proširene stvarnosti unutar programa Unity. Predložak automatski instalira potrebne pakete za AR razvoj aplikacije, te konfigurira scenu hijerarhija za implementaciju AR-a. Primjer kreiranja AR predloška projekta prikazan je na slici 4.1.



Slika 4.1. Prikaz kreiranja AR predloška projekta unutar programa Unity

Ovaj predložak projekta koristi AR *Foundation* [27], paket koji se koristi za višeplatformski AR razvoj. Taj paket predstavlja sučelje koje Unity programeri mogu koristiti, ali sam ne implementira nikakve AR značajke. AR *Foundation* je skup *MonoBehaviour*-a i API-ja za rad s uređajima koji podržavaju sljedeće koncepte:

- Praćenje uređaja
- Praćenje ravnine
- Oblaci točaka
- Referentne točke
- Procjena svjetla
- Okolišne sonde
- Praćenje lica
- Praćenje mreže
- Praćenje slike
- *Raycast*

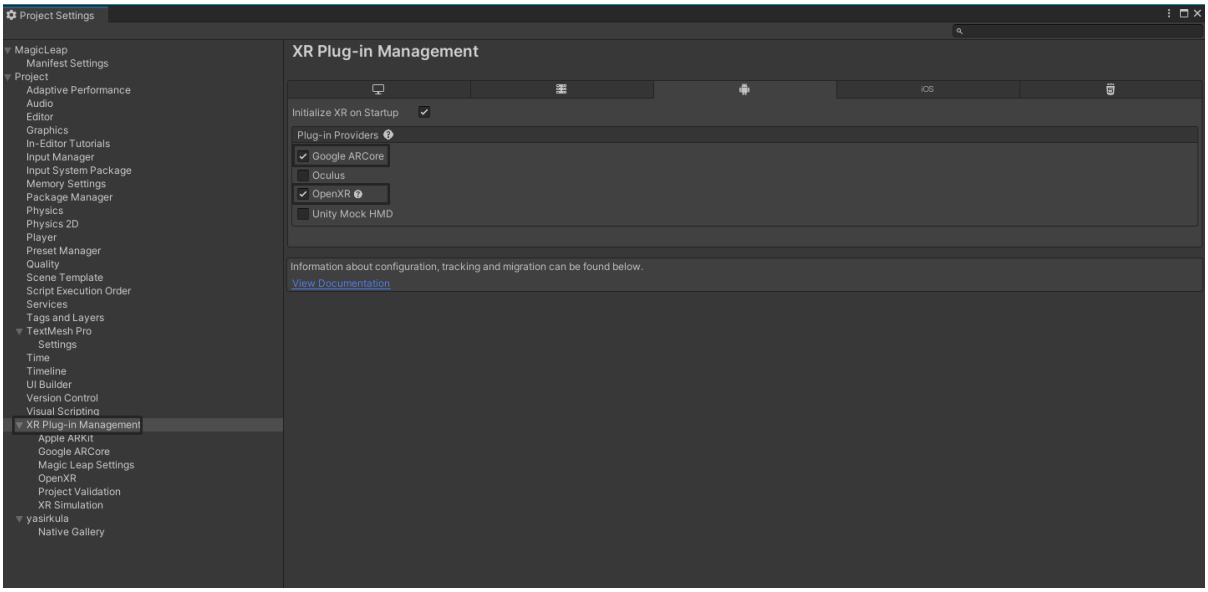
Za praćenje AR prostora u aplikaciji ovoga rada koristi se *Raycast* i praćenje ravnine. Kako bi se AR *Foundation* koristio na ciljnom uređaju, također su potrebni zasebni paketi za ciljne platforme koje službeno podržavaju Unity:

- Dodatak *ARCore XR* na Androidu
- *ARKit XR* dodatak na iOS-u
- *Magic Leap XR* dodatak na *Magic Leap*
- *Windows XR* dodatak na *HoloLens*

Na slici 4.2. prikazani su svi AR paketi koji su potrebni za funkcionalnosti ove aplikacije.

Slika 4.2. Prikaz kreiranja AR predloška projekta unutar programa Unity

Na slici 4.3 prikazan je prozor za upravljanje AR dodacima, te AR dodaci odabrani za ispunjavanje zadaće aplikacije napravljene u ovome radu.

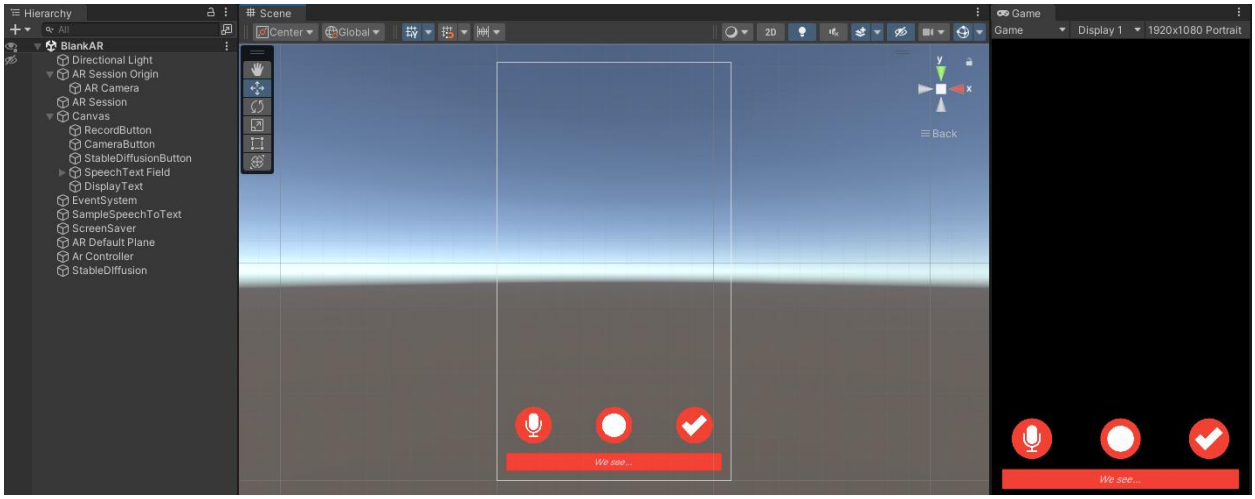


Slika 4.3. Prikaz prozora za upravljanje AR dodacima, te AR dodaci odabrani za ispunjavanje zadaće aplikacije napravljene u ovome radu

Mobilna aplikacija opisana u radu ima sljedeće funkcionalnosti:

- Pretvorba govora u tekst
- Spremanje zaslona
- Primjena StaD-a na temelju upita

Na slici 4.4 prikazan je dizajn aplikacije opisane u radu.



Slika 4.4. Prikaz dizajna aplikacije opisane u ovome radu

Držanje lijevog gumba omogućuje snimanje govora i pretvorbe snimljenog sadržaja u tekst koji se zapisuje u tekstualno polje na dnu ekrana. Pritiskom na srednji gumb pokreće se snimanje

zaslona, te se snimljena slika sprema u galeriju. Pritiskom na desni gumb AR ravnina se postavlja u prostor, te će se na nju postaviti StaD generirana slika na temelju upita zapisanog u tekstualnom polju na dnu ekrana, te na temelju snimljenog dijela zaslona na kojeg se AR ravnina postavila u prostor.

4.1. Pretvorba govora u tekst

U aplikaciji za lakši unos opisa slike generirane StaD-om, uz tekstualni unos, moguć je i unos pomoću govora. Snimanje govora omogućeno je držanjem gumba za pretvorbu teksta u govor. Gumbu je dodijeljena skripta *SpeechButton,* koja služi za prepoznavanje stanja držanja gumba odrađenog pomoću *IPointerDownHandler i IPointerUpHandler* sučelja, te pomoću *PointerEventData* klase. Na slici 4.5 prikazan je kôd za prepoznavanje stanja držanja gumba.

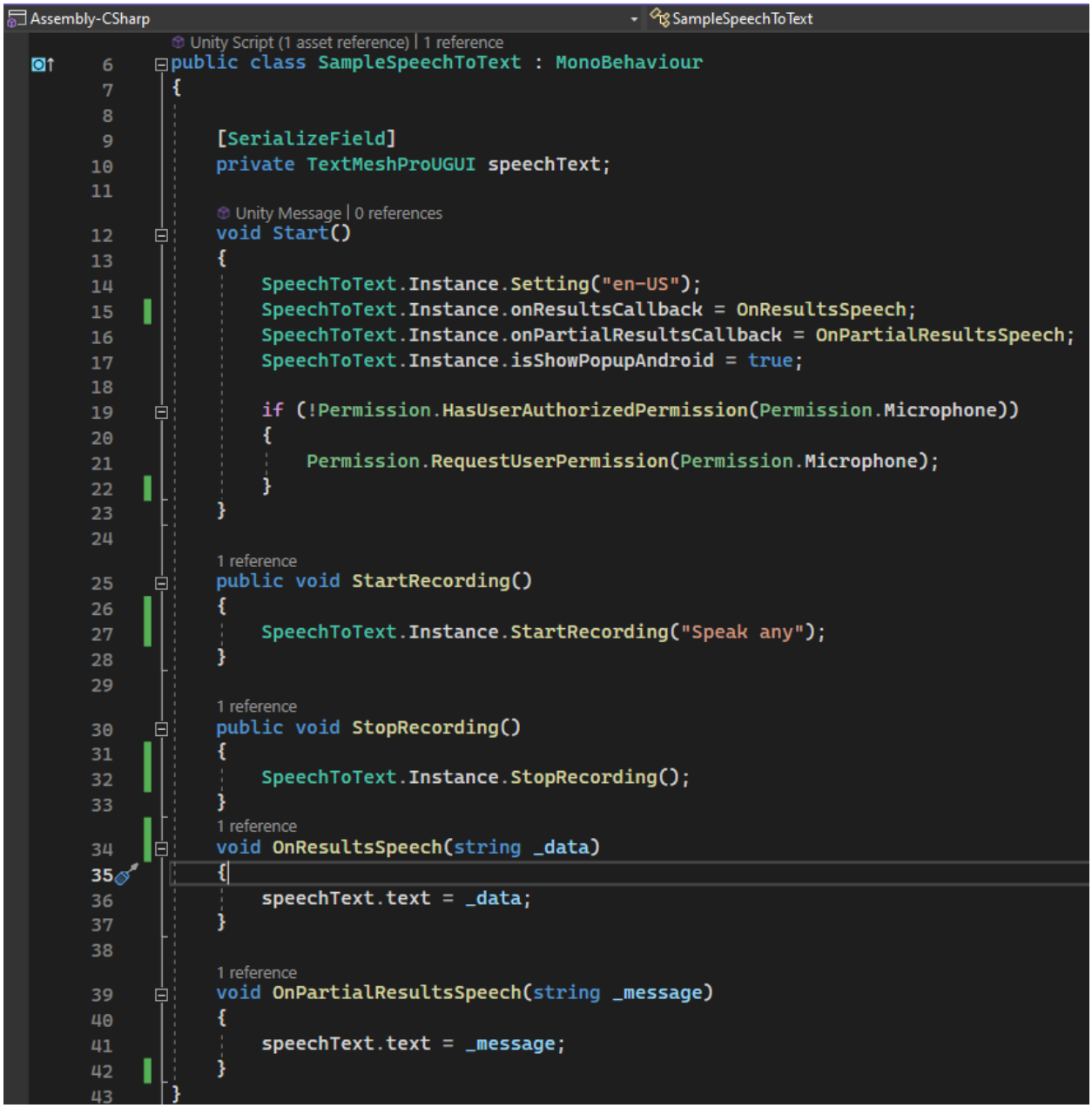
```
Assembly-CSharp                    SpeechButton                    sample
1   using UnityEngine;
2   using UnityEngine.EventSystems;
3
    Unity Script (1 asset reference) | 0 references
4   public class SpeechButton : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
5   {
6       [SerializeField]
7       private SampleSpeechToText sample;
8
        0 references
9       public void OnPointerDown(PointerEventData eventData)
10      {
11          sample.StartRecording();
12      }
13
        0 references
14      public void OnPointerUp(PointerEventData eventData)
15      {
16          sample.StopRecording();
17      }
18  }
```

Slika 4.5. Prikaz kôda za prepoznavanje stanja držanja gumba

Na stisak gumba poziva se *OnPointerDown* metoda, koja poziva *StartRecording* metodu iz skripte *SampleSpeechToText*. Pri prestanku držanja gumba, poziva se *OnPointerUp* metoda koja poziva *StopRecording* metodu iz skripte *SampleSpeechToText*.

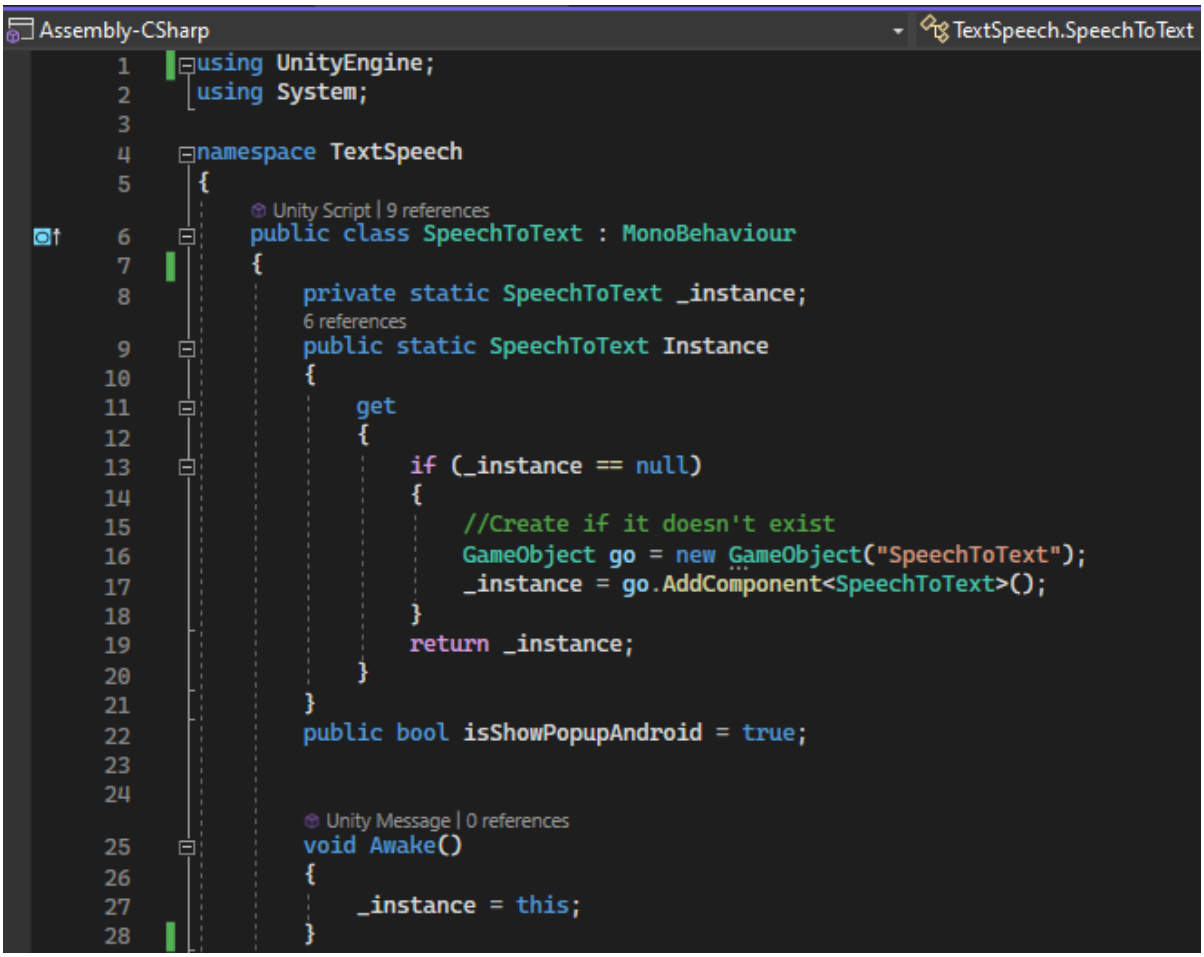
U skripti *SampleSpeechToText* definira se singleton objekt *Instance* klase *SpeechToText*. Pri pokretanju aplikacije objektu *Instance*, podešava se jezik prepoznavanja govora, koji se pretvara u tekst na engleskom jeziku, postavljaju se metode za primanje potpunih i djelomičnih rezultata, te se omogućuje skočni prozor za android mobilni uređaj. Također se traži autorizacija korisnika, kako bi aplikacija imala pristup mikrofonu. Metode *StartRecording* i *StopRecording* pozivaju

istoimene metode iz klase *SpeechToText*. Metode *OnResultSpeech* i *OnPartialResultsSpeech* zapisuju dobivene rezultate pretvorbe govora u tekst u polje na dnu ekrana. Na slici 4.6 prikazan je kôd skripte *SampleSpeechToText*.

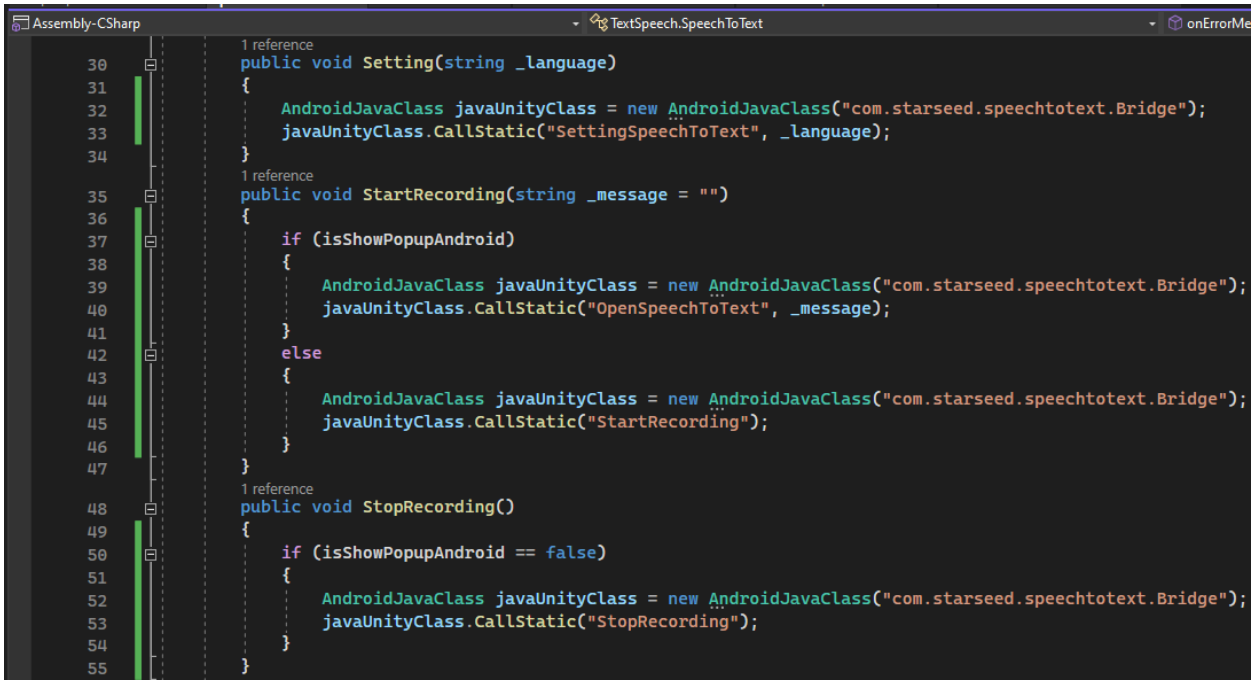


Slika 4.6. Prikaz kôda skripte *SampleSpeechToText*

U skripti *SpeechToText* inicijalizira se singleton objekt *Instance* klase *SpeechToText*, te se kreira novi *GameObject* kojemu je dodjeljuje skripta *SpeechToText*, ukoliko on već ne postoji. Također, omogućuje se skočni prozor za android mobilni uređaj. Do inicijaliziranja instance dolazi pri paljenu aplikacije. Na slici 4.7 prikazan je kôd inicijalizacije *Instance* objekta klase *SpeechToText*.



Slika 4.7. Prikaz kôda inicijalizacije *Instance* objekta klase *SpeechToText*



Slika 4.8. Prikaz kôda u kojemu su definirane metode *Setting*, *StartRecording* i *StopRecording*

Na slici 4.8 prikazan je kôd u kojemu su definirane metode *Setting*, *StartRecording* i *StopRecording*. U metodi *Setting* inicijalizira se novi objekt klase *AndroidJavaClass* koji za parametar prima *SpeechAndTextUnityIosAndroid* [28], dodatak za pretvaranje govora u tekst, te mu se predaje postavka jezika. U metodi *StartRecording* inicijalizira se novi objekt klase *AndroidJavaClass* koji za parametar prima *SpeechAndTextUnityIosAndroid*, te mu se predaje snimka govora. U metodi *StopRecording* inicijalizira se novi objekt klase *AndroidJavaClass* koji za parametar prima *SpeechAndTextUnityIosAndroid*, te snimka govora završava.

Kada *SpeechAndTextUnityIosAndroid* pretvori govor u tekst, naredbom *Action* pozivaju se metode *onResults* i *onPartialResults*, te se u njih zapisuje taj tekst. Metoda *onPartialResults* ažurira tekst za vrijeme slanja glasovne poruke, dok metoda *onResults* ažurira tekst tek nakon završetka slanja glasovne poruke. Na slici 4.9 prikazan je kôd u kojemu su definirane metode *onResults* i *onPartialResults*.

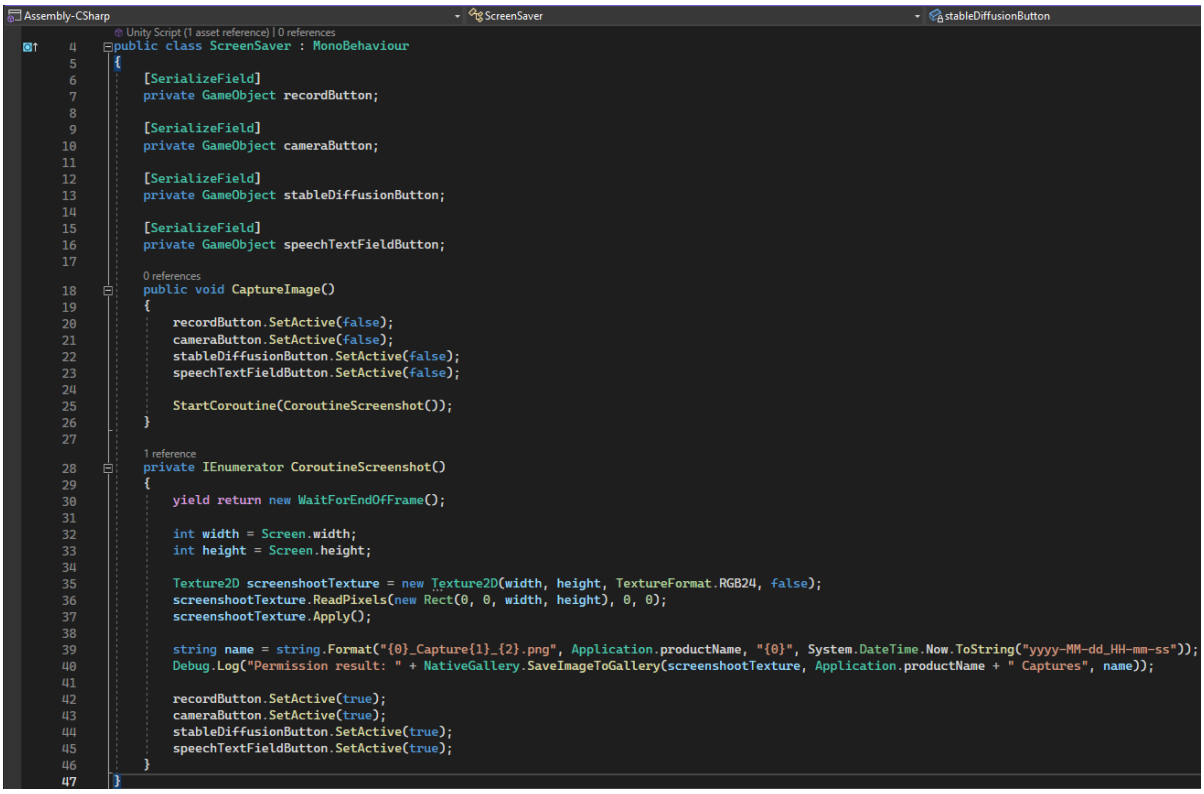
```
132         public Action<string> onResultsCallback;
133         public Action<string> onPartialResultsCallback;
134
135         /** Called when recognition results are ready. */
            0 references
136         public void onResults(string _results)
137         {
138             if (onResultsCallback != null)
139                 onResultsCallback(_results);
140         }
141
142         /** Called when partial recognition results are available. */
            0 references
143         public void onPartialResults(string _params)
144         {
145             if (onPartialResultsCallback != null)
146                 onPartialResultsCallback(_params);
147         }
```

Slika 4.9. Prikaz kôda u kojemu su definirane metode *onResults* i *onPartialResults*

4.2. Snimanje AR zaslona

Aplikacija također omogućava snimanje zaslona pritiskom na srednji gumb. Snimljena slika zaslona sprema se u galeriju pomoću *UnityNativeGallery* [29] dodatka za spremanje slike u galeriju mobitela. Za dodavanje *UnityNativeGallery* u projekt koristi se *Package Manager*. Pritiskom na srednji gumb poziva se metoda *CaptureImage*, koja deaktivira sve vidljive *GameObject*-e unutar *Canvas*-a kako ne bi završili na slici zaslona, te poziva korutinu

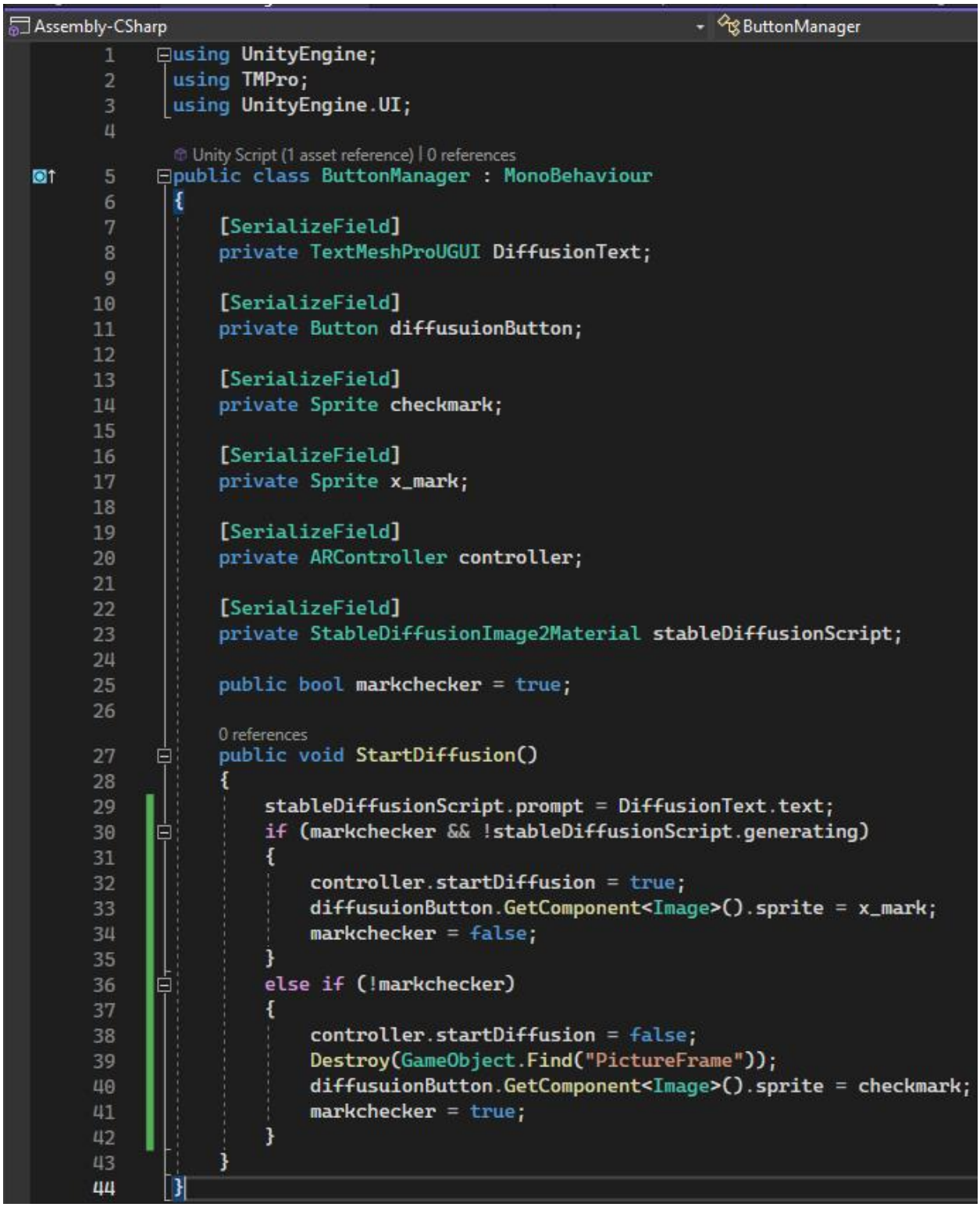
CoroutineScreenshoot. U korutini *CoroutineScreenshoot* inicijalizira se, te definira objekt *screenshootTexture* klase *Texture2D*. Sliku zaslona zapisuje se u objekt *screenshootTexture* te se sprema u galeriju mobitela pomoću dodatka *UnityNativeGallery*. Na kraju korutine *CoroutineScreenshoot* ponovo dolazi do aktivacije svih vidljivih *GameObject*-a unutar *Canvas*-a, kako bi se aplikaciji ponovo omogućile sve funkcionalnosti. Na slici 4.10 prikazan je kôd za spremanje slike zaslona u galeriju android mobitela pomoću dodatka *UnityNativeGallery*.



Slika 4.10. Prikaz kôda za spremanje slike zaslona u galeriju android mobitela pomoću dodatka *UnityNativeGallery*.

4.3. Stable Diffusion u AR aplikaciji

U aplikaciji pritiskom na desni gumb, AR ravnina postavlja se u AR prostor. Gumbu je dodijeljena skripta *ButtonManager*. Na prvi pritisak gumba postavljanjem *bool* tipa varijable *startDiffusion* iz skripte *ARController* na vrijednost *true*, omogućava se postavljanje AR ravnine unutar AR prostora. Pritiskom gumba, također se mijenja sličica kvačice s gumba u sličicu oznake slova x. U slučaju da već postoji slika koja se generira, pri pritisku na gumb onemogućuje se postavljanja AR ravnine u AR prostor. U slučaju da u AR prostoru već postoji AR ravnina, ona se uklanja. Također, gumbu se ponovo mijenja sličica na sličicu kvačice. Na slici 4.11 prikazan je kôd za upravljanje StaD gumbom.



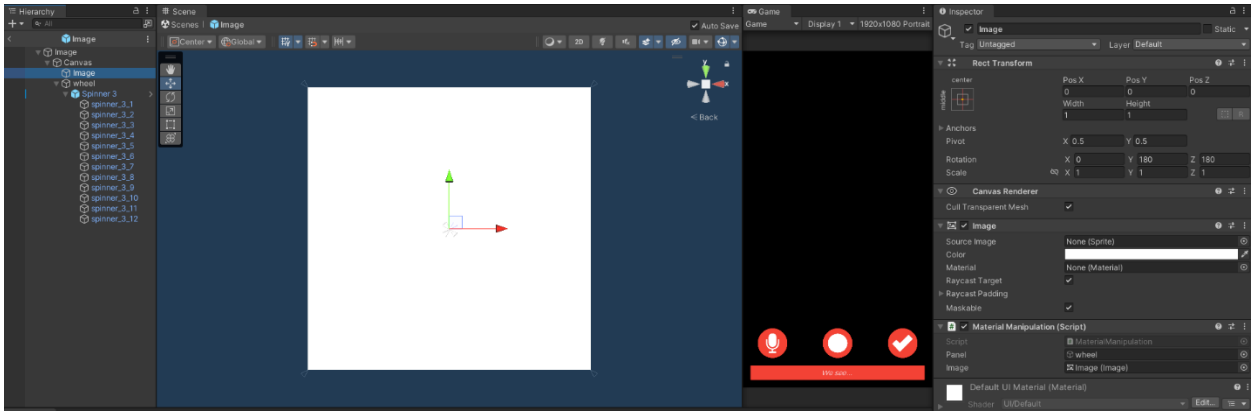
Slika 4.11. Prikaz kôda za upravljanje Stable Diffusion gumbom

Ako je *bool* tip varijable *startDiffusion* postavljen na vrijednost *true*, u skripti *ARController* metoda *Update* koja se poziva na svaku izmjenu sličice, poziva metodu *GeneratePrefab*, te postavlja vrijednost varijable *startDiffusion* na vrijednost *false*. Metoda *GeneratePrefab* dohvaća položaj i rotaciju kamere, te ju sprema u objekt *cam*. Pomoću objekta *cam* instancira se AR

ravnina imena *PictureFrame* na sredini ekrana na statičnoj udaljenosti on kamera iznosa 1. Na slici 4.12 prikazan je kôd za dodavanje AR ravnine u AR prostor, a na slici 4.13 prikazan je početni izgled AR ravnine.

```
Assembly-CSharp                                    ARController
using UnityEngine;

Unity Script (1 asset reference) | 1 reference
public class ARController : MonoBehaviour
{
    [SerializeField]
    private GameObject MyObject;

    [SerializeField]
    private StableDiffusionImage2Material stableDiffusionScript;

    [SerializeField]
    private GameObject ARImage;

    public bool startDiffusion = false;
    Unity Message | 0 references
    private void Update()
    {
        if(startDiffusion)
        {
            GeneratePrefab();
            startDiffusion = false;
        }
    }

    1 reference
    private void GeneratePrefab()
    {
        Transform cam = Camera.main.transform;
        ARImage = Instantiate(MyObject, cam.position + cam.forward * 1, cam.rotation);
        ARImage.transform.name = "PictureFrame";
    }
}
```

Slika 4.12. Prikaz kôda za dodavanje AR ravnine u AR prostor



Slika 4.13 Prikaz početnog izgleda AR ravnine

Pri dodavanju AR ravnine u prostor, poziva se metoda *Start* u skripti *MaterialManipulation*. U metodi *Start* objekt *stablediff* klase StableDiffusionImage2Material [30], koja služi za pokretanje generiranja slike na temelju danog upita, u hijerarhiji *GameObject*-a, s *GameObject*-a

StableDiffusion poziva skriptu *StableDiffusionImage2Material*. Kreira se objekt *texture* klase Texture2D, koji se definira pozivom metode *CaptureScreenshot*. U metodi *CaptureScreenshot* kreira se novi objekt *renderTexture* klase *RenderTexture* s dimenzijama rezolucije ekrana mobilnog uređaja. Teksturu kamere postavlja se na objekt *renderTexture*. Objekt *renderTexture* postavlja se kao aktivan RenderTexture i izrađuje se novi objekt *croppedTexture* klase Texture2D s dimenzijama željenog područja izrezivanja. Postavljaju se kordinate koje označavaju pomak između početka ekrana i željenog područja izrezivanja. U sljedećem koraku čitaju se pikseli *croppedTexture-a* sa željenog mjesta izrezivanja, te se potvrđuje njegova trenutna tekstura. Aktivnu RenderTexture i teksturu kamere postavlja se na *null* objekt. U zadnjem koraku, *renderTexture* se uništava, te se vraća tekstura *croppedTexture*. U metodi *Start*, u sljedećem koraku atribut *inputTexture2D* objekta *stablediff* postavlja se na objekt *texture*. U sljedećem koraku pokreće se metoda Generate() objekta *stablediff*. Također, kreira se objekt *sprite* klase *Sprite* koji se radi na temelju objekta *texture*. Objekt *sprite* postavlja se kao slika na AR ravninu. Ako je generiranje slike upravo završilo, atribut *generatingFinished* objekta *stablediff* postavlja se na vrijednost *true*. U metodi *Update*, ako je *generatingFinished* jednak *true*, onda se kreira objekt *sprite* klase *Sprite* koji se radi na temelju atributa *texture* objekta *stablediff*. Objekt *sprite* postavlja se kao slika na AR ravninu. Atribut *generatingFinished* postavlja se na vrijednost *false*. Na slici 4.14 prikazan je kôd skripte *MaterialManipulation*.

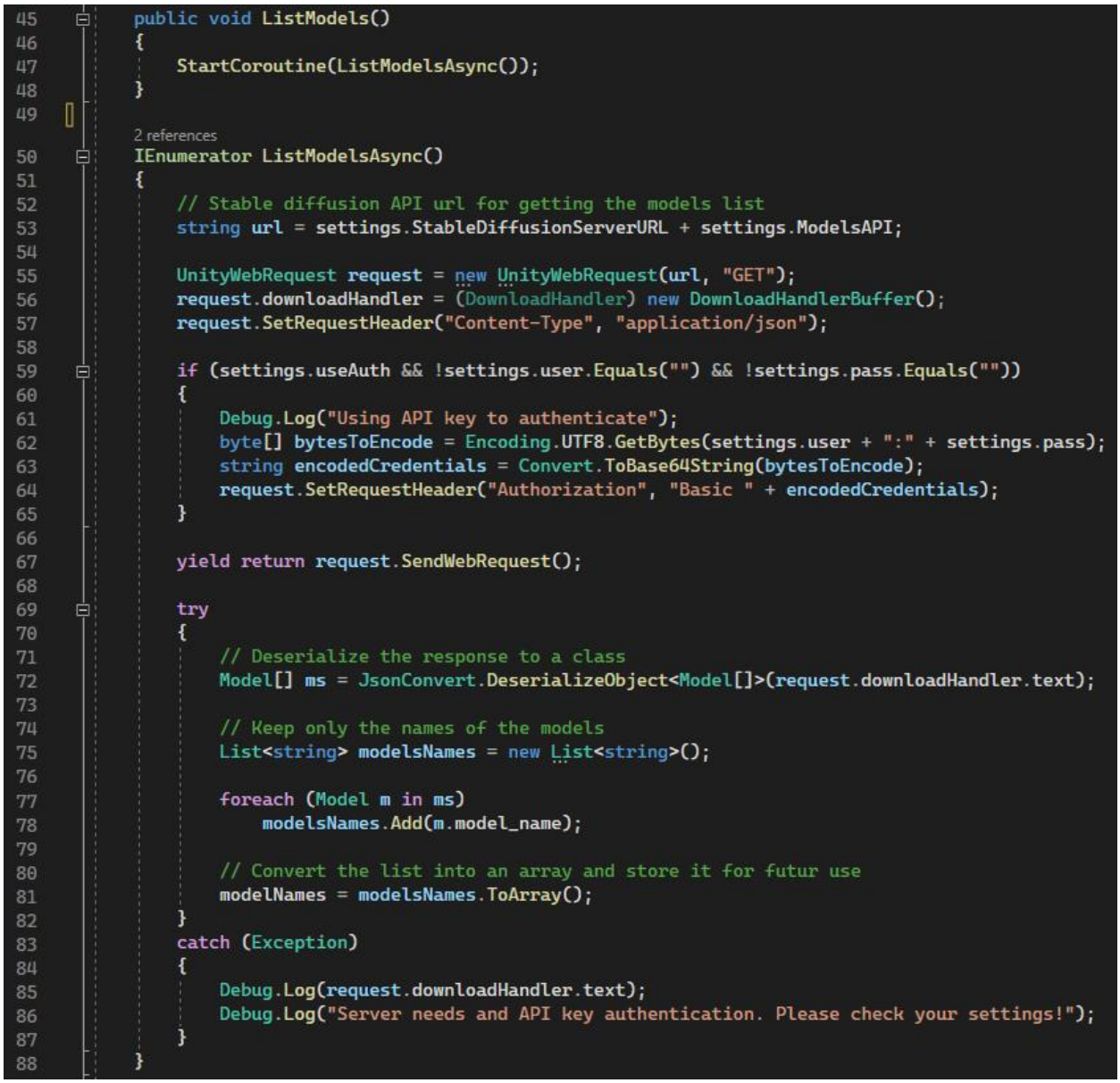
```
Assembly-CSharp                    MaterialManipulation                    Start()

Unity Script (1 asset reference) | 0 references
public class MaterialManipulation : MonoBehaviour
{
    StableDiffusionImage2Material stablediff;
    public Image image;

    Unity Message | 0 references
    private void Start()
    {
        stablediff = GameObject.Find("StableDiffusion").GetComponent<StableDiffusionImage2Material>();
        Texture2D texture= CaptureScreenshot();
        stablediff.inputTexture2D = texture;
        stablediff.Generate();
        Sprite sprite = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height), Vector2.one * 0.5f);
        image.sprite = sprite;
        image.enabled = true;
    }
    Unity Message | 0 references
    void Update()
    {
        if (stablediff.generatingFinished == true)
        {
            Debug.Log("generiranje Sprite");
            Sprite sprite = Sprite.Create(stablediff.texture, new Rect(0, 0, stablediff.texture.width, stablediff.texture.height), Vector2.one * 0.5f);
            image.sprite = sprite;
            stablediff.generatingFinished = false;
        }
    }
    1 reference
    public Texture2D CaptureScreenshot()
    {
        RenderTexture renderTexture = new RenderTexture(Screen.width, Screen.height, 24);
        Camera.main.targetTexture = renderTexture;
        Camera.main.Render();
        RenderTexture.active = renderTexture;
        Texture2D croppedTexture = new Texture2D((int)(Screen.width * 0.4861111111f), (int)(Screen.height * 0.2f), TextureFormat.RGB24, false);
        int startX = (int)(Screen.width * 0.26388888888f);
        int startY = (int)(Screen.height * 0.4f);
        croppedTexture.ReadPixels(new Rect(startX, startY, croppedTexture.width, croppedTexture.height), 0, 0);
        croppedTexture.Apply();
        RenderTexture.active = null;
        Camera.main.targetTexture = null;
        Destroy(renderTexture);
        return croppedTexture;
    }
}
```

Slika 4.14. Prikaz kôda skripte *MaterialManipulation*

Pritiskom na desni gumb, pokreće se metoda *Generate()* iz skripte *StableDiffusionImage2Material*. Metoda *Generate()* šalje *HttpWebRequest* korisničkom sučelju Stable Diffusion Automatic 1111, koje generira sliku. Modeli generiranja slike definiraju se skriptom *StableDiffusionConfiguration*. Mogući modeli za generiranje slike su:

- *Euler a*
- *Euler*
- *LMS, Heun*
- *DPM2*
- *DPM2 a*
- *DPM++ 2S a*
- *DPM++ 2M*
- *DPM++ SDE*
- *DPM fast*
- *DPM adaptive*
- *LMS Karras*
- *DPM2 Karras*
- *DPM2 a Karras*
- *DPM++ 2S a Karras*
- *DPM++ 2M Karras*
- *DPM++ SDE Karras*
- *DDIM, PLMS*

Za generiranje slika StaD-om koristi se algoritam *Euler a*. Dok se slika generira, skripta *StableDiffusionGenerator* dohvaća status generiranja. Slika koja se generira definirana je klasom *SDParamsInImg2Img* iz skripte *SDSettings*. Na slici 4.15 prikazan je kôd za dohvaćanje modela na korisničkom sučelju *Automatic StaD 1111*.

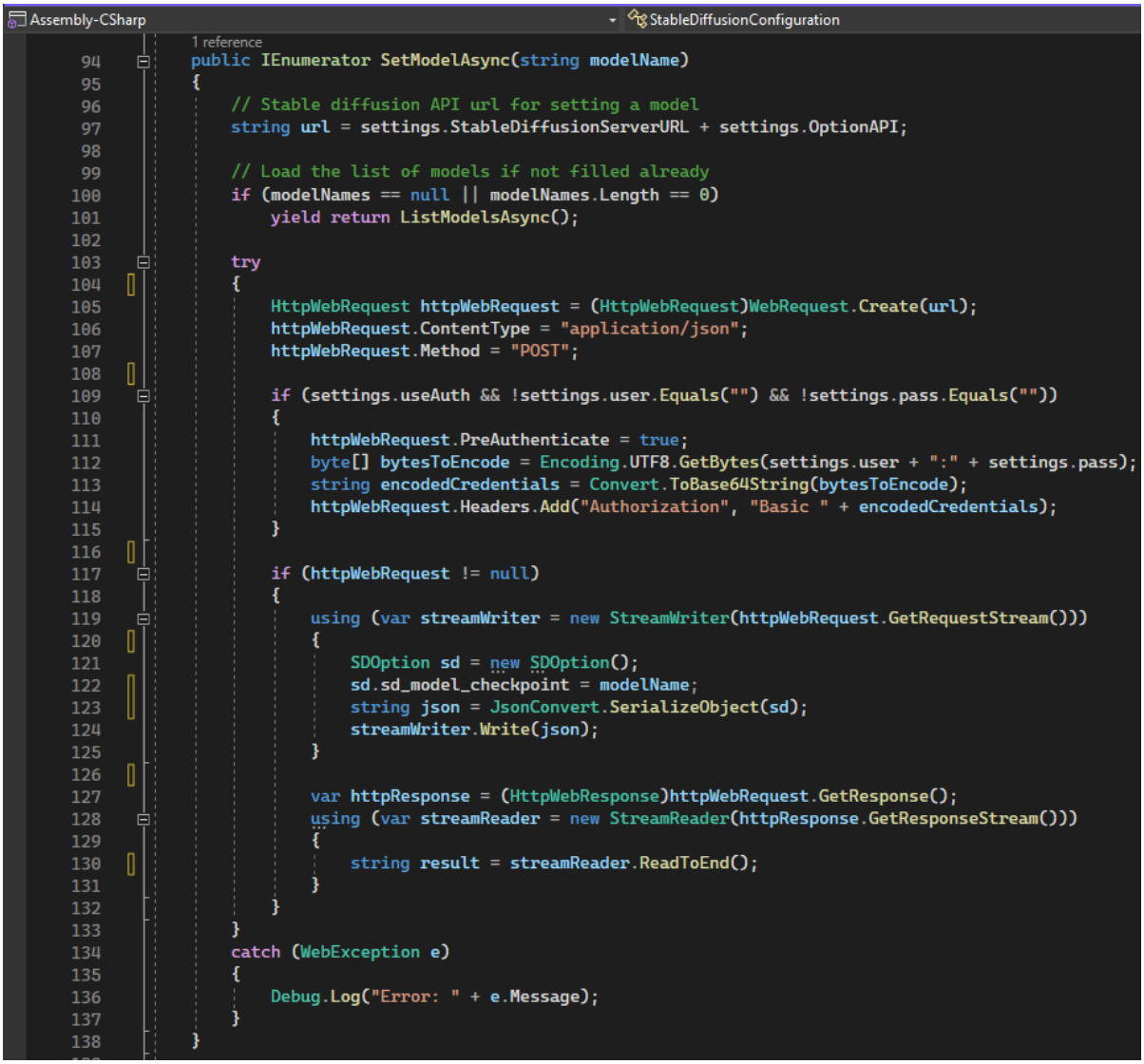


```
public void ListModels()
{
    StartCoroutine(ListModelsAsync());
}

2 references
IEnumerator ListModelsAsync()
{
    // Stable diffusion API url for getting the models list
    string url = settings.StableDiffusionServerURL + settings.ModelsAPI;

    UnityWebRequest request = new UnityWebRequest(url, "GET");
    request.downloadHandler = (DownloadHandler) new DownloadHandlerBuffer();
    request.SetRequestHeader("Content-Type", "application/json");

    if (settings.useAuth && !settings.user.Equals("") && !settings.pass.Equals(""))
    {
        Debug.Log("Using API key to authenticate");
        byte[] bytesToEncode = Encoding.UTF8.GetBytes(settings.user + ":" + settings.pass);
        string encodedCredentials = Convert.ToBase64String(bytesToEncode);
        request.SetRequestHeader("Authorization", "Basic " + encodedCredentials);
    }

    yield return request.SendWebRequest();

    try
    {
        // Deserialize the response to a class
        Model[] ms = JsonConvert.DeserializeObject<Model[]>(request.downloadHandler.text);

        // Keep only the names of the models
        List<string> modelsNames = new List<string>();

        foreach (Model m in ms)
            modelsNames.Add(m.model_name);

        // Convert the list into an array and store it for futur use
        modelNames = modelsNames.ToArray();
    }
    catch (Exception)
    {
        Debug.Log(request.downloadHandler.text);
        Debug.Log("Server needs and API key authentication. Please check your settings!");
    }
}
```

Slika 4.15. Prikaz kôda za dohvaćanje modela na korisničkom sučelju StaD Automatic 1111

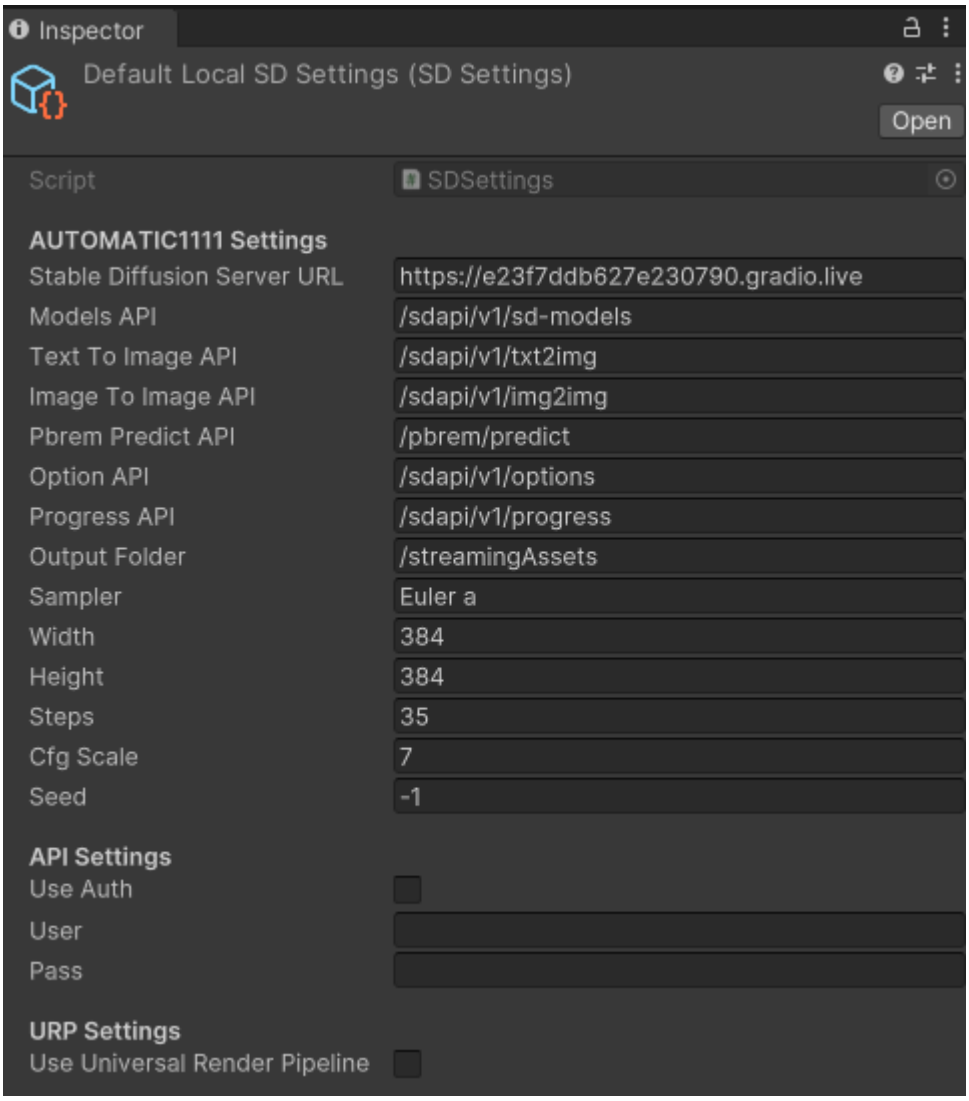
Metoda *ListModels* poziva korutinu *ListModelAsync* koja, šaljući zahtjev API-ju (engl. *Application Programming Interface*), dohvaća listu svih modela. Kôdiraju se podaci potrebni za pristup, te se šalju API-ju kako bi se pristup ostvario. Dohvaćaju se svi modeli, te se zapisuju u listu, koja se zapisuje u polje elemenata radi lakšeg pristupa i odabira željenog modela. Metoda Set*ModelsAsync* dohvaća listu svih postojećih modela pozivom na metodu *ListModelAsync*. Kôdiraju se podaci potrebni za pristup, te se šalju API-ju kako bi se pristup ostvario. Kada je pristup ostvaren, stvara se novi objekt tipa *SDOption*, dodjeljuje mu se ime modela, te se objekt serijalizira u JSON (engl. *JavaScript Object Notation*) tip elementa koji se šalje serveru. Također, šalje se zahtjev serveru, te se čeka njegovo odgovor kako bi se osiguralo da je

funkcionalnost metode završena. Na slici 4.16 prikazan je kôd za postavljanje modela na korisničkom sučelju StaD Automatic 1111.



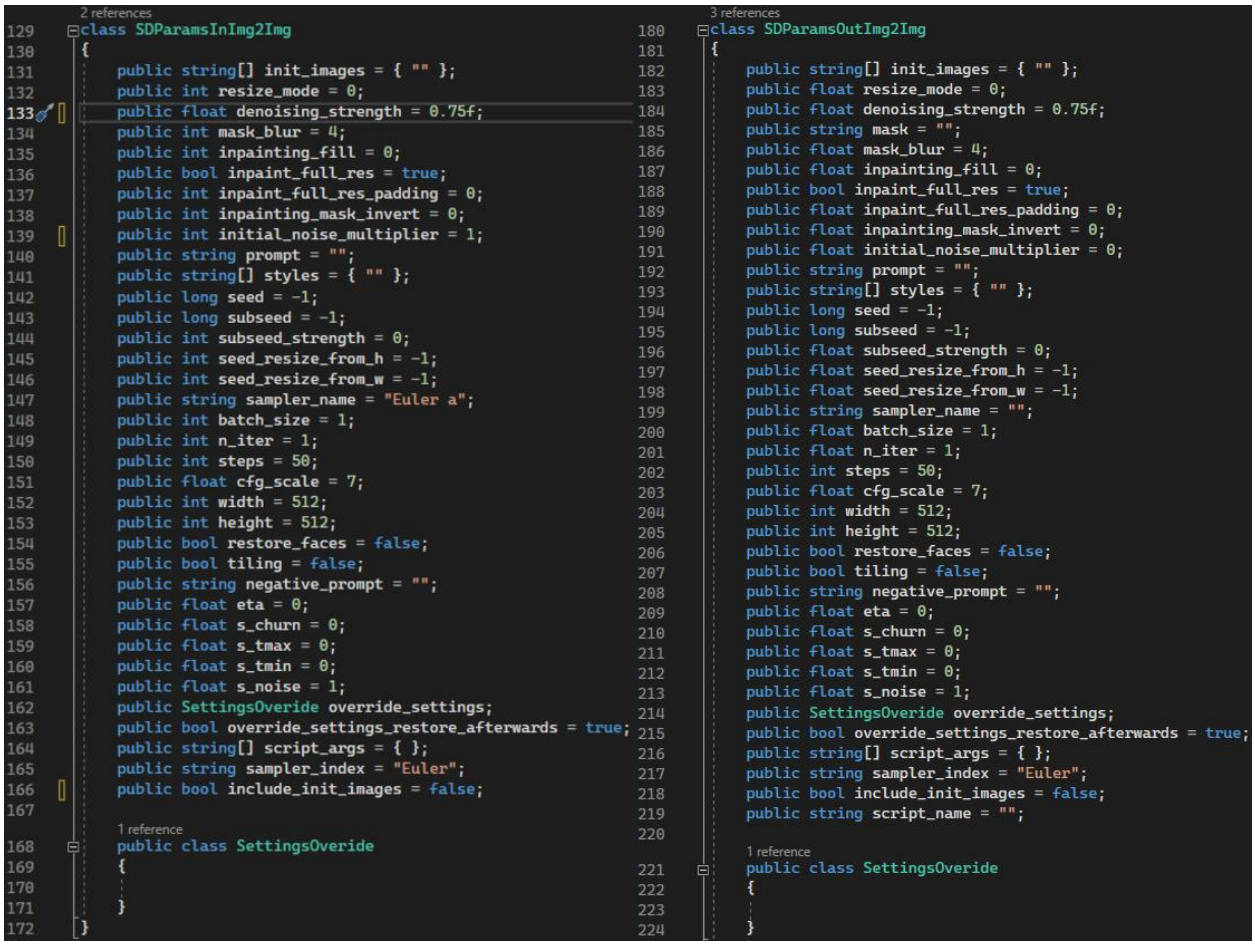
Slika 4.16. Prikaz kôda za postavljanje modela na korisničkom sučelju StaD Automatic 1111

Skriptom *SDSettings* stvara se *ScriptableObject* koji definira postavke za Stable Diffusion Automatic 1111, te postavke za API i URP (engl. *Universal Render Pipeline*). Unutar *ScriptableObject-a* definira se web adresa servera varijablom *StableDiffusionServerURL*. U slučaju promjene servera, dolazi i do promijene web adrese za server. Također, definiraju se ekstenzije na link servera za sve API-je korištene u ovoj aplikaciji, metoda generiranja i zadana rezolucija generirane slike. Na slici 4.17 prikazan je *ScriptableObject* za postavljanje postavki za Stable Diffusion Automatic 1111, API i URP.



Slika 4.17. Prikaz *ScriptableObject*-a za postavljanje postavki za Stable Diffusion Automatic 1111, API i URP

U skripti *SDSettings* klasa *SDParamsInImg2Img* predstavlja strukturu podataka za jednostavnu serijalizaciju parametara za slanje serveru prilikom generiranja slike metodom Img2Img, dok klasa *SDParamsInImg2Img* predstavlja strukturu podataka za jednostavnu deserijalizaciju podataka koje server vraća nakon generiranja slike metodom Img2Img. Na slici 4.18 prikazan je kôd za *SDParamsInImg2Img* i *SDParamsOutImg2Img* klase u skripti *SDSettings*.

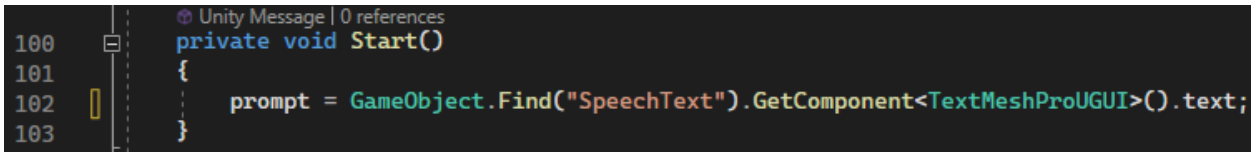


```
2 references
class SDParamsInImg2Img
{
    public string[] init_images = { "" };
    public int resize_mode = 0;
    public float denoising_strength = 0.75f;
    public int mask_blur = 4;
    public int inpainting_fill = 0;
    public bool inpaint_full_res = true;
    public int inpaint_full_res_padding = 0;
    public int inpainting_mask_invert = 0;
    public int initial_noise_multiplier = 1;
    public string prompt = "";
    public string[] styles = { "" };
    public long seed = -1;
    public long subseed = -1;
    public int subseed_strength = 0;
    public int seed_resize_from_h = -1;
    public int seed_resize_from_w = -1;
    public string sampler_name = "Euler a";
    public int batch_size = 1;
    public int n_iter = 1;
    public int steps = 50;
    public float cfg_scale = 7;
    public int width = 512;
    public int height = 512;
    public bool restore_faces = false;
    public bool tiling = false;
    public string negative_prompt = "";
    public float eta = 0;
    public float s_churn = 0;
    public float s_tmax = 0;
    public float s_tmin = 0;
    public float s_noise = 1;
    public SettingsOveride override_settings;
    public bool override_settings_restore_afterwards = true;
    public string[] script_args = { };
    public string sampler_index = "Euler";
    public bool include_init_images = false;

    1 reference
    public class SettingsOveride
    {

    }
}
```

```
3 references
class SDParamsOutImg2Img
{
    public string[] init_images = { "" };
    public float resize_mode = 0;
    public float denoising_strength = 0.75f;
    public string mask = "";
    public float mask_blur = 4;
    public float inpainting_fill = 0;
    public bool inpaint_full_res = true;
    public float inpaint_full_res_padding = 0;
    public float inpainting_mask_invert = 0;
    public float initial_noise_multiplier = 0;
    public string prompt = "";
    public string[] styles = { "" };
    public long seed = -1;
    public long subseed = -1;
    public float subseed_strength = 0;
    public float seed_resize_from_h = -1;
    public float seed_resize_from_w = -1;
    public string sampler_name = "";
    public float batch_size = 1;
    public float n_iter = 1;
    public int steps = 50;
    public float cfg_scale = 7;
    public int width = 512;
    public int height = 512;
    public bool restore_faces = false;
    public bool tiling = false;
    public string negative_prompt = "";
    public float eta = 0;
    public float s_churn = 0;
    public float s_tmax = 0;
    public float s_tmin = 0;
    public float s_noise = 1;
    public SettingsOveride override_settings;
    public bool override_settings_restore_afterwards = true;
    public string[] script_args = { };
    public string sampler_index = "Euler";
    public bool include_init_images = false;
    public string script_name = "";

    1 reference
    public class SettingsOveride
    {

    }
}
```

Slika 4.18. Prikaz kôda za *SDParamsInImg2Img* i *SDParamsOutImg2Img* klase u skripti *SDSettings*

Upit ili varijabla *prompt* iz klasa *SDParamsInImg2Img* i *SDParamsOutImg2Img*, na temelju kojega se slika generira, dohvaća se metodom *Start* u skripti *StableDiffusionImage2Material*. Na slici 4.19 prikazan je kôd za dohvaćanje upita.



```
Unity Message | 0 references
private void Start()
{
    prompt = GameObject.Find("SpeechText").GetComponent<TextMeshProUGUI>().text;
}
```

Slika 4.19. Prikaz kôda za dohvaćanje upita

Za spremanje generiranih slika na mobilnom uređaju, dohvaća se lokacija *persistentDataPath*. Na većini mobilnih uređaja, *persistentDataPath* pokazuje na direktorij */storage/emulated/<userid>/Android/data/<packagename>/files*. U tom direktoriju dohvaća se direktorij */StreamingAssets*, te se u njemu dohvaća direktorij */SDMaterials*. Ako oni već ne postoje, onda se kreiraju. Na slici 4.19 prikazan je kôd za postavljanje direktorija u koji se generirana slika sprema.

```
1 reference
void SetupFolders()
{
    if (sdc == null)
        sdc = GameObject.FindObjectOfType<StableDiffusionConfiguration>();

    try
    {
        string root = Application.persistentDataPath + sdc.settings.OutputFolder;
        if (root == "" || !Directory.Exists(root))
            root = Application.persistentDataPath;
        string mat = Path.Combine(root, "SDMaterials");
        filename = Path.Combine(mat, guid + ".png");

        if (!Directory.Exists(root))
            Directory.CreateDirectory(root);
        if (!Directory.Exists(mat))
            Directory.CreateDirectory(mat);

        if (File.Exists(filename))
            File.Delete(filename);
    }
    catch (Exception e)
    {
        Debug.LogError(e.Message + "\n\n" + e.StackTrace);
    }
}
```

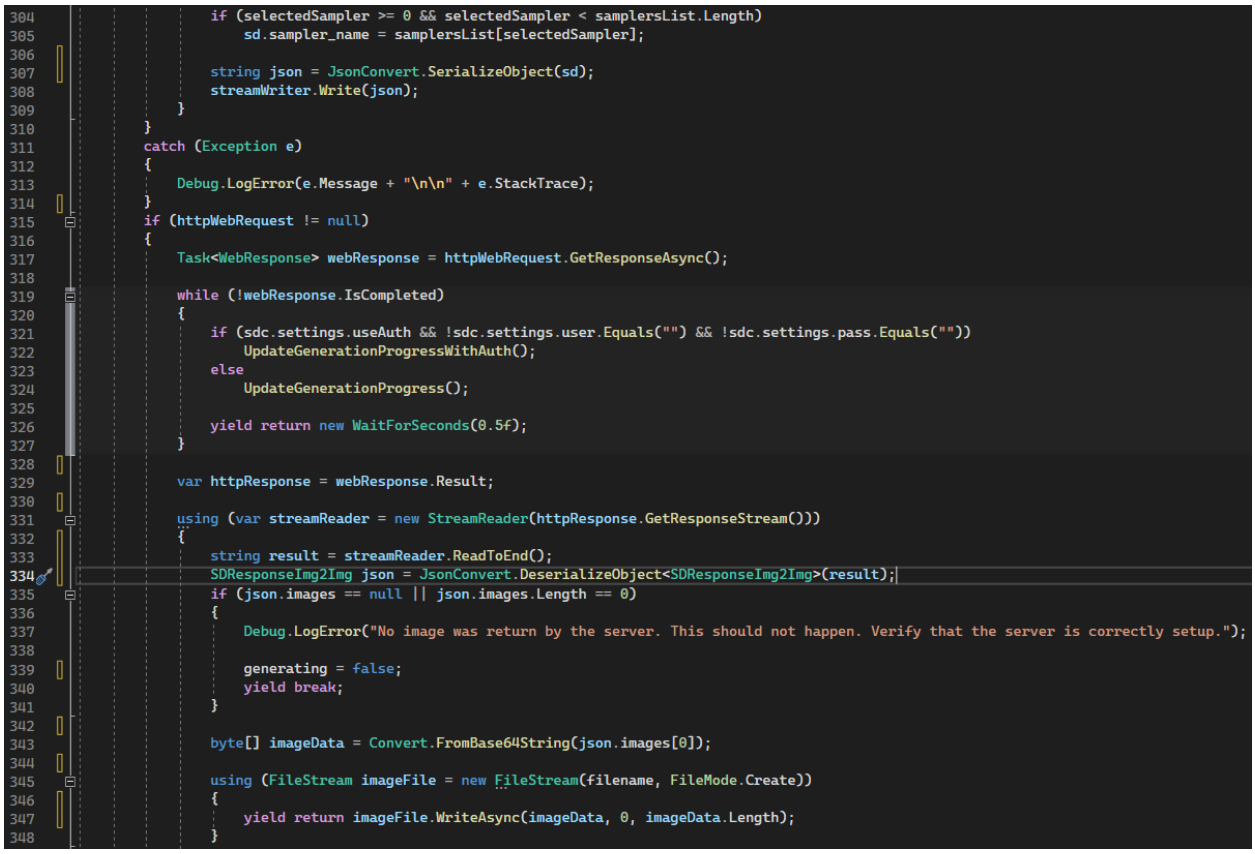
Slika 4.19. Prikaz kôda za postavljanje direktorija u koji se generirana slika sprema

Pritiskom na desni gumb, pokreće se metoda *Generate()* iz skripte *StableDiffusionImage2Material*. Ako postoji upit, te ako se trenutna slika ne generira, metoda *Generate()* pokreće korutinu *GenerateAsync* u kojoj se poziva metoda SetupFolders, te se postavljaju direktoriji za spremanje slike koja će se generirati. Korutina *GenerateAsync* također poziva prije spomenutu metodu *SetModelAsync*, kako bi se odabrao model za generiranje slike. Sljedeći korak je slanje *HttpWebReques-at* serveru kako bi se s njime uspostavio kontakt. Također, korutina *GenerateAsync* šalje potrebne informacije serveru za generiranje slike koristeći objekt *sd* već prije spomenute klase, *SDParamsInImg2Img*. Informacija koje će definirati novonastalu generiranu sliku su: upit, negativan upit, broj koraka, mogućnost popločavanja slike, broj sjemena te rezolucija definirana visinom i širinom. Na slici 4.20 prikazan je dio kôda za slanje zahtjeva serveru za generiranje slike putem *StaD Automatic 1111*.

```
1 reference
public void Generate()
{
    if (!generating && !string.IsNullOrEmpty(prompt))
        StartCoroutine(GenerateAsync());
}

1 reference
IEnumerator GenerateAsync()
{
    generating = true;
    SetupFolders();
    yield return sdc.SetModelAsync(modelsList[selectedModel]);

    HttpWebRequest httpWebRequest = null;

    try
    {
        httpWebRequest = (HttpWebRequest)WebRequest.Create(sdc.settings.StableDiffusionServerURL + sdc.settings.ImageToImageAPI);
        httpWebRequest.ContentType = "application/json";
        httpWebRequest.Method = "POST";

        if (sdc.settings.useAuth && !sdc.settings.user.Equals("") && !sdc.settings.pass.Equals(""))
        {
            httpWebRequest.PreAuthenticate = true;
            byte[] bytesToEncode = Encoding.UTF8.GetBytes(sdc.settings.user + ":" + sdc.settings.pass);
            string encodedCredentials = Convert.ToBase64String(bytesToEncode);
            httpWebRequest.Headers.Add("Authorization", "Basic " + encodedCredentials);
        }

        using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
        {
            byte[] inputImgBytes = inputTexture2D.EncodeToPNG();
            string inputImgString = Convert.ToBase64String(inputImgBytes);

            SDParamsInImg2Img sd = new SDParamsInImg2Img();
            sd.init_images = new string[] { inputImgString };
            sd.prompt = prompt;
            sd.negative_prompt = negativePrompt;
            sd.steps = steps;
            sd.cfg_scale = cfgScale;
            sd.width = width;
            sd.height = height;
            sd.seed = seed;
            sd.tiling = false;
            sd.denoising_strength = denoising_strength;
```

Slika 4.20. Prikaz dijela kôda za slanje zahtjeva serveru za generiranje slike putem *StaD Automatic 1111*

Sljedeći korak je serijaliziranje ulaznih parametara, a ono se postiže pretvaranjem *sd* objekta u niz znakova. Niz znakova koji opisuju objekt *sd*, šalju se serveru. Ako je zahtjev za generiranje slike uspješno poslan, te dok traje generiranje slike, od servera se traži povratna informacija o stanju generiranja. Kada je generiranje gotovo, dobiveni rezultati se deserijaliziraju. Nakon deserijalizacije rezultata, provjerava se je li server uistinu poslao sliku ili je došlo do greške. Ako je došlo do greške pri generiranju slike, ostatak metode se neće izvršiti i korutina *GenerateAsync* završava. Ako nije došlo do greške, slika se dekodira iz Base64 tipa podataka u polje bajtova, te se to polje bajtova zapisuje u izlaznu datoteku. Na slici 4.21 prikazan je ostatak kôda za slanje zahtjeva serveru i dio kôda za primanje slike od servera za generiranje slike putem *StaD Automatic 1111*.



```
                if (selectedSampler >= 0 && selectedSampler < samplersList.Length)
                    sd.sampler_name = samplersList[selectedSampler];

                string json = JsonConvert.SerializeObject(sd);
                streamWriter.Write(json);
            }
        }
        catch (Exception e)
        {
            Debug.LogError(e.Message + "\n\n" + e.StackTrace);
        }
        if (httpWebRequest != null)
        {
            Task<WebResponse> webResponse = httpWebRequest.GetResponseAsync();

            while (!webResponse.IsCompleted)
            {
                if (sdc.settings.useAuth && !sdc.settings.user.Equals("") && !sdc.settings.pass.Equals(""))
                    UpdateGenerationProgressWithAuth();
                else
                    UpdateGenerationProgress();

                yield return new WaitForSeconds(0.5f);
            }

            var httpResponse = webResponse.Result;

            using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
            {
                string result = streamReader.ReadToEnd();
                SDResponseImg2Img json = JsonConvert.DeserializeObject<SDResponseImg2Img>(result);
                if (json.images == null || json.images.Length == 0)
                {
                    Debug.LogError("No image was return by the server. This should not happen. Verify that the server is correctly setup.");

                    generating = false;
                    yield break;
                }

                byte[] imageData = Convert.FromBase64String(json.images[0]);

                using (FileStream imageFile = new FileStream(filename, FileMode.Create))
                {
                    yield return imageFile.WriteAsync(imageData, 0, imageData.Length);
                }
```

Slika 4.21. Prikaz ostatka kôda za slanje zahtjeva serveru i dio kôda za primanje slike od servera za generiranje slike putem *StaD Automatic 1111*

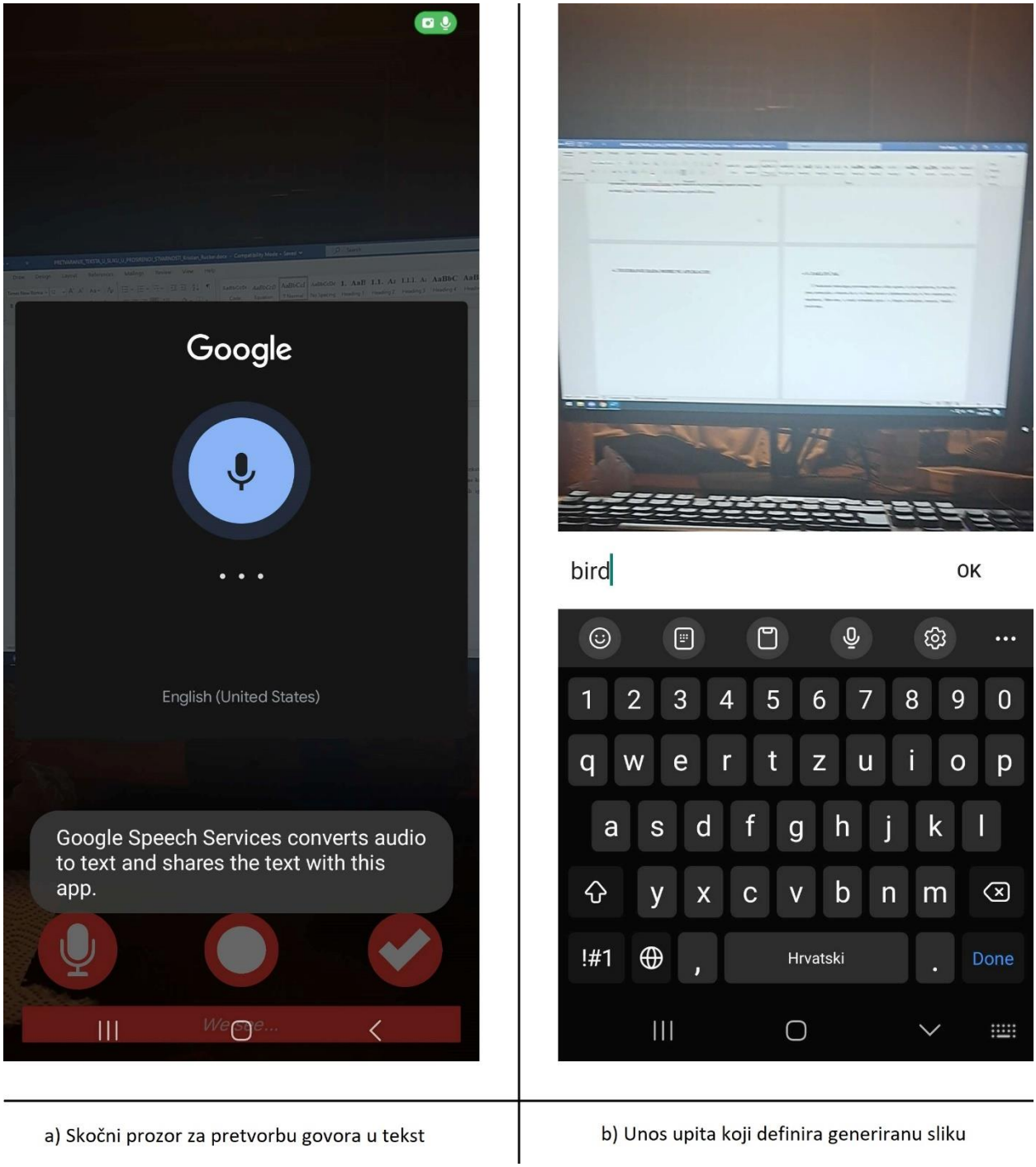
Nakon zapisivanja polja bajtova u izlaznu datoteku, u sljedećem se koraku iz te datoteke čitaju i zapisuju infromacije generirane slike u objekt *texture* klase *Texture2D*. U sljedećem koraku potvrđuju se promjene napravljene nad objektom u kojemu su upisane informacije generirane slike. Potvrđuje se da je generiranje slike gotovo, kako bi se generiranje slike moglo započeti opet. Još preostaje provjeriti informacije o generiranju je li se sve odradilo bez greške. Kreira se objekt prije spomenute klase, *SDParamsOutImg2Img*, nazvan *info*, u kojem se zapisuju sve informacije o postupku generanja slike i u njemu bi jedino odstupanje od predanih parametara trebao biti parametar sjemena. Taj parametar je trebao odabrati sam postupak generiranja, nakon čega se zapisuje u varijablu *generatedSeed*. Na slici 4.22 prikazan je ostatak kôda za primanje slike od servera za generiranje slike putem *StaD Automatic 1111*.

```
350                 try
351                 {
352                     if (File.Exists(filename))
353                     {
354                         texture = new Texture2D(512, 512);
355                         texture.LoadImage(imageData);
356                         texture.Apply();
357 
358                         LoadIntoMaterial(texture);
359                         generatingFinished = true;
360                     }
361 
362                     if (json.info != "")
363                     {
364                         SDParamsOutImg2Img info = JsonConvert.DeserializeObject<SDParamsOutImg2Img>(json.info);
365 
366                         generatedSeed = info.seed;
367                     }
368                 }
369                 catch (Exception e)
370                 {
371                     Debug.LogError(e.Message + "\n\n" + e.StackTrace);
372                 }
373             }
374         }
375         generating = false;
376         yield return null;
377     }
```

Slika 4.22. Prikaz ostatka kôda za primanje slike od servera za generiranje slike putem *StaD Automatic 1111*

5. OPIS I TESTIRANJE RADA MOBILNE APLIKACIJE

Pri pokretanju, aplikacija opisana u ovome radu od strane korisnika traži odobrenje za pristup kameri i mikrofonu. Nakon što se aplikaciji odobrio pristup kameri i mikrofonu, aplikacija prikazuje prostor koji se snima, te omogućava korisnički unos pomoću tri gumba i tekstualnog polja. Na slici 5.1 prikazan je korisnički unos pomoću lijevog gumba i tekstualnog polja.

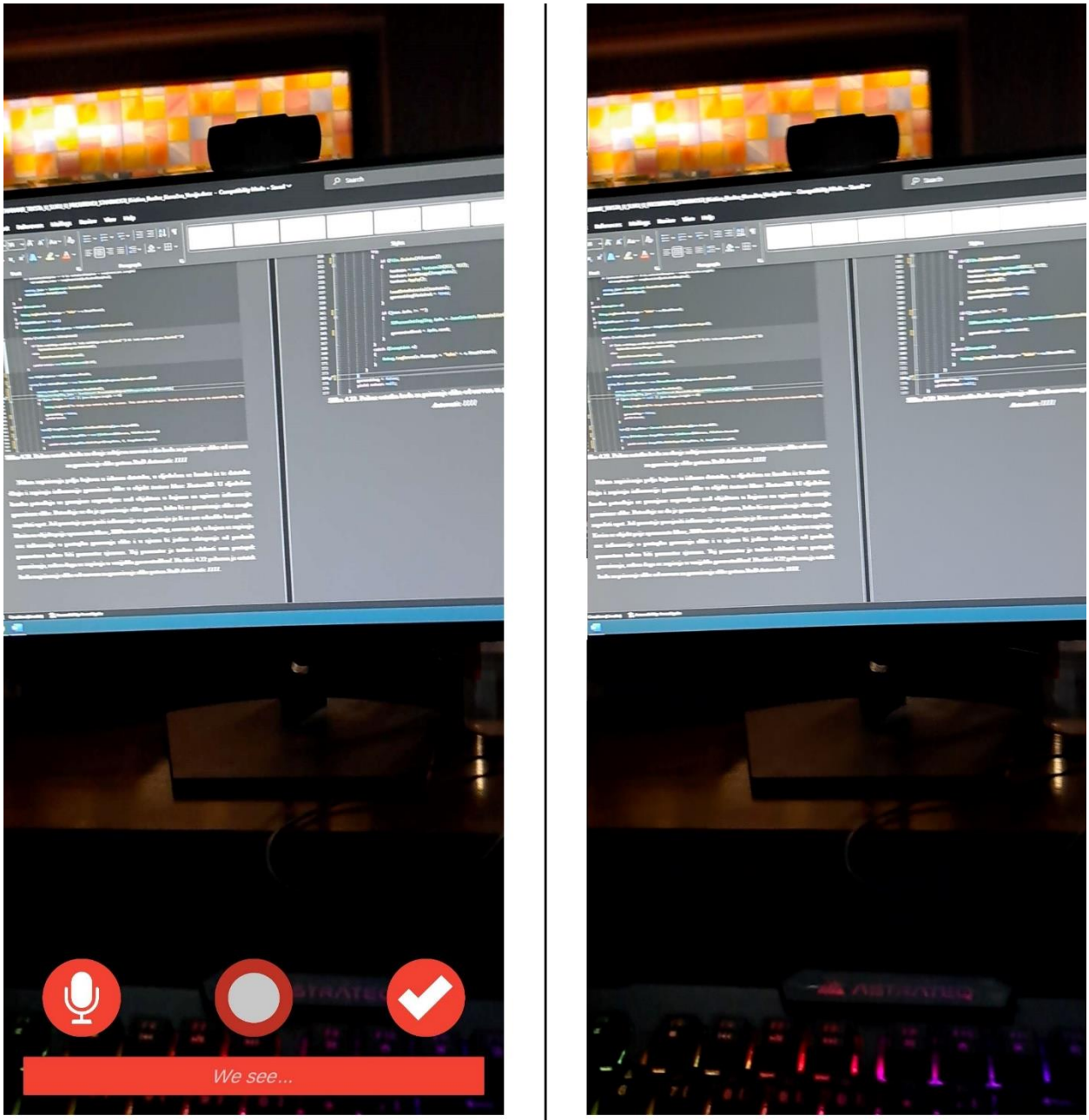


a) Skočni prozor za pretvorbu govora u tekst

b) Unos upita koji definira generiranu sliku

Slika 5.1. Prikaz korisničkog unosa pomoću lijevog gumba koji otvara a) skočni prozor za pretvorbu govora u tekst, i tekstualnog polja za b) unos upita koji definira generiranu sliku

Držanje lijevog gumba omogućuje snimanje govora i pretvorbe snimljenog sadržaja u tekst, koji se zapisuje u tekstualno polje na dnu ekrana. Pritiskom na tekstualno polje, ako upit već postoji, taj upit moguće je doraditi ili bolje definirati. Također ga je moguće i obrisati. U slučaju da upit ne postoji, moguće je unijeti i potpuno novi upit. Na slici 5.2 prikazan je korisnički unos pomoću srednjeg gumba.

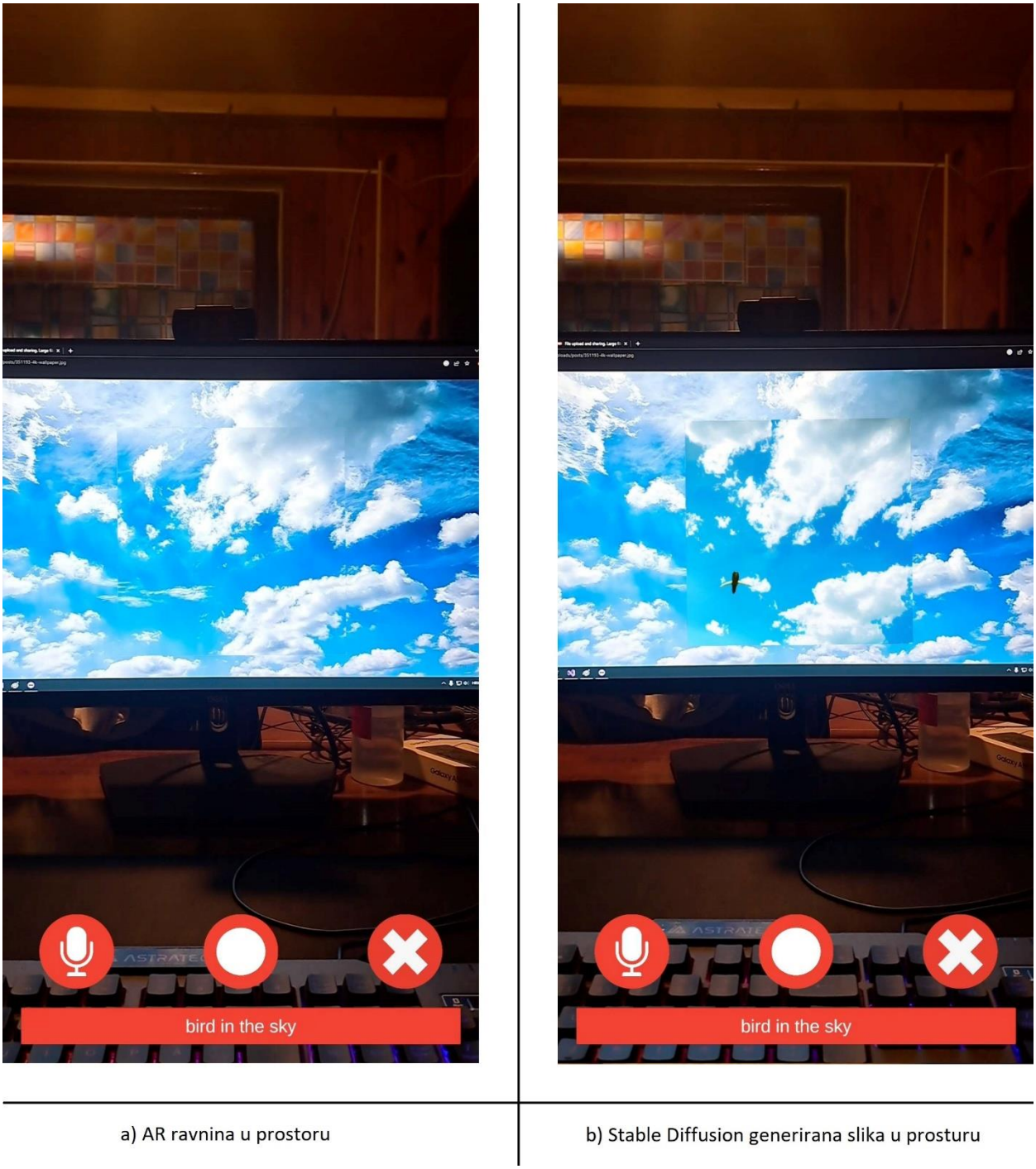


a) Pritisak gumba za snimanje zaslona

b) Snimljen zaslon pritiskom na gumb za snimanje zaslona

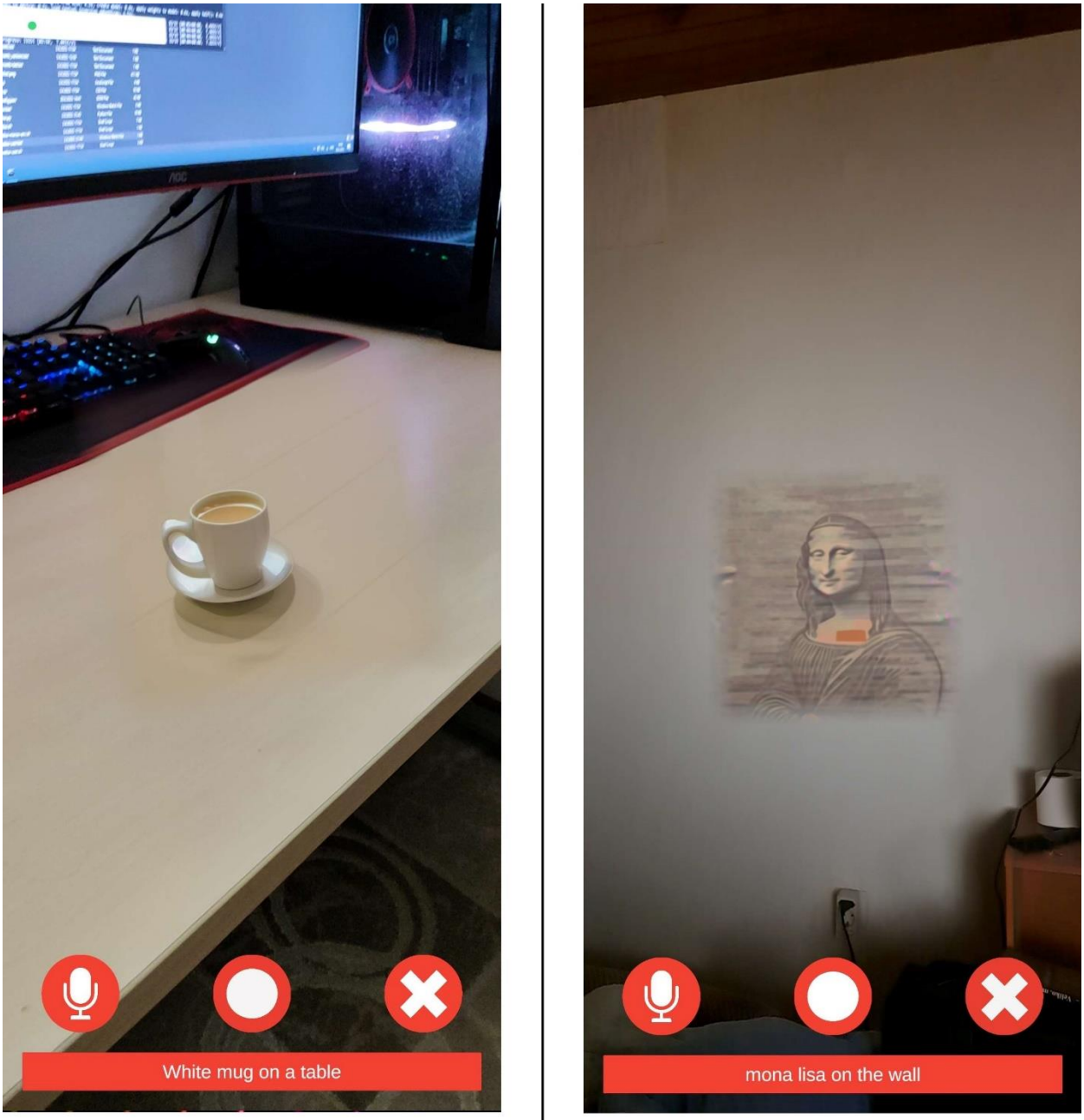
Slika 5.2. Prikaz korisničkog unosa pomoću srednjeg gumba, Na a) pritisak gumba za snimanje zaslona, zaslon se snima te se korisniku u galeriju slika sprema b) snimljen zaslon

Pritiskom na srednji gumb pokreće se snimanje zaslona, te se snimljena slika sprema u galeriju slika. Pritiskom na desni gumb, AR ravnina postavlja se u prostor. Na AR ravninu postaviti će se StaD generirana slika na temelju upita zapisanog u tekstualnom polju na dnu ekrana, te na temelju snimljenog dijela zaslona na kojemu je AR ravnina postavljena u prostor. Na slici 5.3 prikazan je korisnički unos pomoću desnog gumba.



Slika 5.3. Prikaz korisničkog unos pomoću desnog gumba, postavljanje a) AR ravnine u prostor, na koju alat b) Stable Diffusion generira sliku u prostoru

Na slici 5.4 prikazane su Stable Diffusion generirane slike na temelju upita a)“*White mug on a table“* i b)“*Mona Lisa on the wall“*.

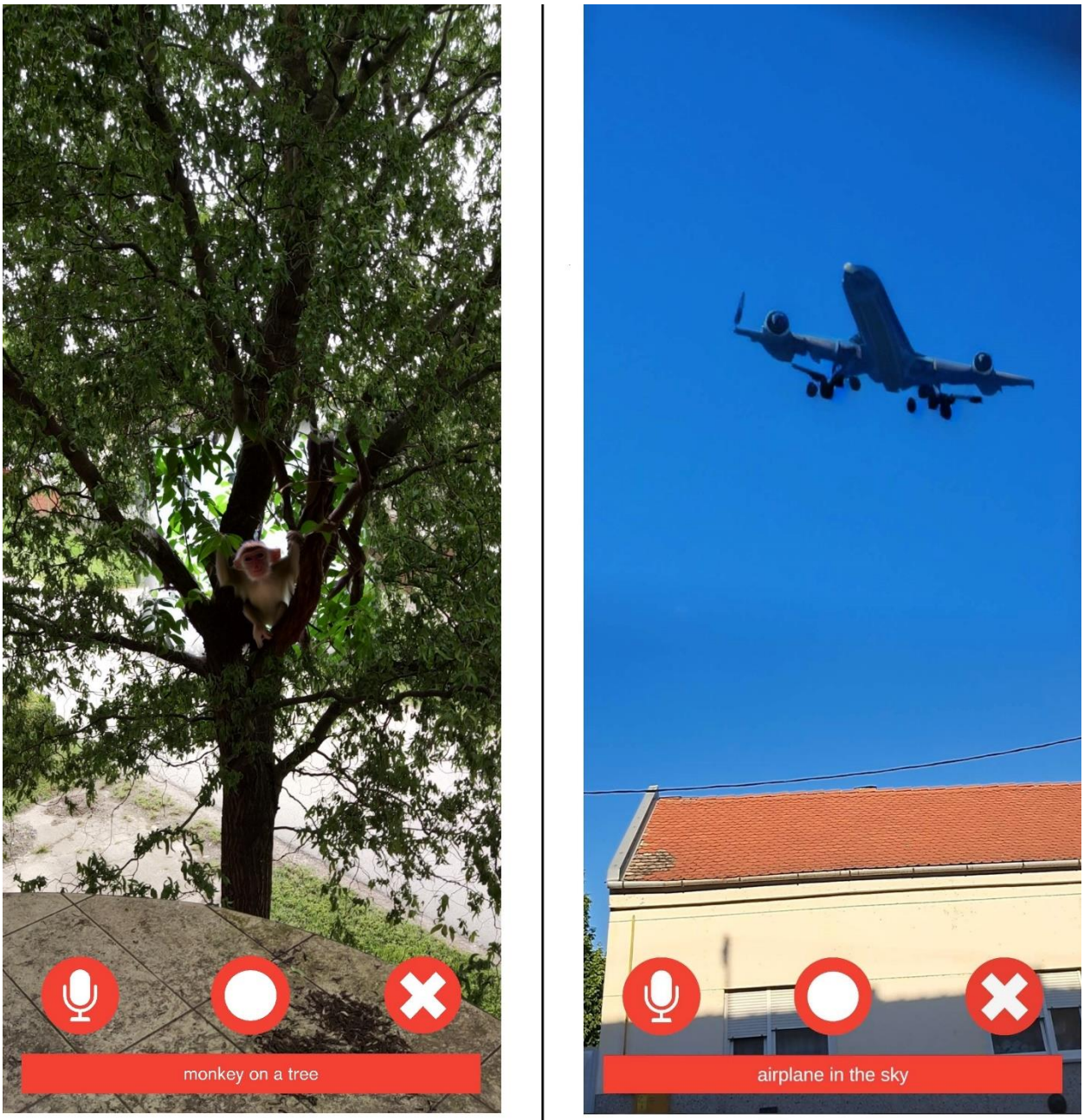


a) Stable Diffusion generira slika na temelju prompta "White mug on a table"

b) Stable Diffusion generira slika na temelju prompta "Mona Lisa on the wall"

Slika 5.4. Prikaz Stable Diffusion generiranih slika na temelju upita a)“*White mug on a table“* i b)“*Mona Lisa on the wall“*

Na slici 5.5 prikazane su Stable Diffusion generirane slike na temelju upita a)“*White mug on a table“* i b)“*Mona Lisa on the wall“*.



a) Stable Diffusion generira slika na temelju prompta "monkey on a tree"

b) Stable Diffusion generira slika na temelju prompta "airplane in the sky"

Slika 5.5. Prikaz Stable Diffusion generiranih slika na temelju upita a)“*monkey on a tree*“ i b)“*airplane in the sky*“

6. ZAKLJUČAK

U ovom radu analizirano je kako radi mobilna aplikacija koja koristi Stable Diffusion biblioteku za dodavanje slika opisanih tekstom u proširenu stvarnost.

Rješenje zadatka ovoga rada izvršeno je unutar programa Unity. Za pisanje skripta C# programskog jezika u završnom radu koristi se Visual Studio 2022. Za potrebe pokretanja i korištenja mogućnosti StaD-a koristi se UI Stable Diffusion Automatic 1111. U programu Unity napravljen je dizajn aplikacije.

Aplikacijom se upravlja pomoću tri gumba i tekstualnog polja. Držanje lijevog gumba omogućuje snimanje govora i pretvorbe snimljenog sadržaja u tekst, koji se zapisuje u tekstualno polje na dnu ekrana. Pritiskom na srednji gumb pokreće se snimanje zaslona, te se snimljena slika sprema u galeriju. Pritiskom na desni gumb, AR ravnina postavlja se u prostor, te će se na nju postaviti StaD generirana slika na temelju upita zapisanog u tekstualnom polju na dnu ekrana, te na temelju snimljenog dijela zaslona na kojeg se AR ravnina postavila u prostor.

Stable Diffusion, kao model pretvaranja teksta testiran u ovome radu, ima svoje probleme tijekom generiranja slika, od krivo spojenih dijelova tijela do abnormalnog broja udova ili prstiju. AR ravnina se ponekad u prostoru izgubi ili dislocira.

U budućnosti, AR sustav i modeli pretvaranja teksta u sliku sigurno će nastaviti napredovati. Postoji veliki potencijal za daljnji razvoj, s obzirom na njihovu sve veću uporabu u djelatnostima koje se bave animacijom, u umjetnosti, filmovima, u izradi računalnih igrica i u drugim područjima znanosti, tehnike i poslovanja.

LITERATURA

[1] DALL-E-2, dostupno na: https://openai.com/dall-e-2 [28/06/2023]

[2] Midjourny, dostupno na: https://www.midjourney.com/home/?callbackUrl=%2Fapp%2F [28/06/2023]

[3] D. Bolya, J. Hoffman, "Token Merging for Fast Stable Diffusion", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2023, pp. 4598-4602, dostupno na: https://openaccess.thecvf.com/content/CVPR2023W/ECV/html/Bolya_Token_Merging_for_Fast_Stable_Diffusion_CVPRW_2023_paper.html [10/04/2023]

[4] Y. Hu, M. Yuan, K. Xian, D. S. Elvitigala, A. Quigley, "Exploring the Design Space of Employing AI-Generated Content for Augmented Reality Display", 2023, dostupno na: https://arxiv.org/abs/2303.16593 [10/04/2023]

[5] STABILITY AI LTD, "Stable Diffusion Launch Announcement", *talk at University of Louisville*, 2023, dostupno na: https://stability.ai/blog/stablediffusion2-1-release7-dec-2022 [10/04/2023]

[6] R. Ramesh, "Stable Diffusion: Common Pitfalls, Use Cases, and How To Handle Real-world Deployments at Massive Scale", 2023, dostupno na: https://blog.segmind.com/stable-diffusion-deployment [10/04/2023]

[7] D. Schmalstieg, T. Hollerer, "Deep Learning", 2015, pp. 1-31, dostupno na: https://www.researchgate.net/publication/277411157_Deep_Learning [10/04/2023]

[8] J. Vincent, " All these images were generated by Google's latest text-to-image AI", *The Verge. Vox Media*, 2022, dostupno na: https://www.theverge.com/2022/5/24/23139297/google-imagen-text-to-image-ai-system-examples-paper [10/04/2023]

[9] L. Hardesty, "Explained: Neural Networks", *MIT News Office*, 2017, dostupno na: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 [10/04/2023]

[10] V. Nandhini, "Blue Brain Using Wetware Technology and Fuzzy Logic", *Department of Computer Applications, S.A.Engineering College*, 2015, pp. 87-90, dostupno na: https://irdp.info/journals/j1/SA_special_issue/ESIJ_SETSRW-MCA-SAEC22.pdf [10/04/2023]

[11] R. T. Azuma, " A Survey of Augmented Reality", *Hughes Research Laboratories*, 1997, dostupno na: https://www.cs.unc.edu/~azuma/ARpresence.pdf [28/06/2023]

[12] D. Schmalstieg, T. Hollerer, “Fast Augmented Reality: Principles and Practice“, 2016, pp. 1-31, dostupno na: https://sites.cs.ucsb.edu/~holl/pubs/Schmalstieg-2016-AW.pdf [10/04/2023]

[13] “AR Construction – Benefits and Limitations“, 2023, dostupno na: https://www.inaugment.com/ar-construction [10/04/2023]

[14] “The Future of Augmented Reality in Advanced Medicine“, 2017, dostupno na: https://www.oneyoungworld.com/blog/future-augmented-reality-advanced-medicine [10/04/2023]

[15] “10 Ways How Augmented Reality Can Help Engineering Students“ , dostupno na: https://www.ixrlabs.com/blog/ar-help-engineering-students [10/04/2023]

[16] Mobilna igrica Pokémon GO, dostupno na: https://pokemongolive.com [10/04/2023]

[17] “Augmented Reality Head-Up Display (AR HUD) Become the Trend of Head-Up Display“, 2022, dostupno na: https://www.fic.com.tw/hud-trend [10/04/2023]

[18] M. Warren, “AR for Navigation – What You Should Know“ , dostupno na: https://www.bairesdev.com/author/michael-warren [10/04/2023]

[19] Mobilna aplikacija Layer , dostupno na: https://www.layar.com [28/06/2023]

[20] Mobilna aplikacija Wikitude, dostupno na: https://www.wikitude.com [28/06/2023]

[21] Mobilna aplikacija Junaio, dostupno na: https://www.appsdrop.com/junaio [28/06/2023]

[22] Nextech, “10 Ways Augmented Reality Can Impact Your Marketing“, 2020, dostupno na: https://www.nextechar.com/blog/10-ways-augmented-reality-can-impact-your-marketing [10/04/2023]

[23] “Examples of Augmented Reality (AR) Experiences in Sports“, 2020, dostupno na: https://www.immersiv.io/blog/ar-digital-experiences-sports [10/04/2023]

[24] Višeplatformska igraća mašina Unity, dostupna na: https://unity.com [10/04/2023]

[25] Integrirano razvojno okruženje Visual Studio 2022, dostupno na: https://visualstudio.microsoft.com/vs [10/04/2023]

[26] Mrežno korisničko sučelje Stable Diffusion Automatic 1111, dostupno na: https://github.com/AUTOMATIC1111/stable-diffusion-webui [10/04/2023]

[27] Unity Manual, dostupna na: https://docs.unity3d.com/Manual/index.html [10/04/2023]

[28] Dodak za pretvorbu govora u tekst, J. To, “Speech and Text Unity Ios Android“, dostupna na: https://github.com/j1mmyto9/speech-and-text-unity-ios-android [10/04/2023]

[29] Dodak za spremanje slike u galeriju, J. To, “Unity Native Gallery“, dostupna na: https://github.com/yasirkula/UnityNativeGallery [10/04/2023]

[30] Stable Diffusion dodak za Unity, Dobrado76, “Stable Diffusion Unity Integration“, dostupna na: https://github.com/dobrado76/Stable-Diffusion-Unity-Integration [10/04/2023]

SAŽETAK

Razvojem znanosti i tehnike, u raznim područjima ljudske djelatnosti dolazi do sve veće upotrebe umjetne inteligencije, pa tako i sustava poput AR-a (engl. *Augmented Reality*) i modela StaD (engl. *Stable Diffusion*). Zadatak ovog rada bio je primjena StaD modela za izradu aplikacije za dodavanje slika opisanih tekstom u AR. Zadatak je izvršen putem Unity alata za izradu računalnih igara. Za pisanje skripta C# programskog jezika, korišten je Visual Studio 2022. Za pokretanje i primjenu mogućnosti koje nudi StaD, korišten je UI (engl. *User Interface*) Stable Diffusion Automatic 1111.

Ključne riječi: Augmented Reality, Proširena Stvarnost, Stable Diffusion, Unity

ABSTRACT

With the development of science and technology, in various areas of human activity there is an increasing use of artificial intelligence, including systems such as AR (Augmented Reality) and the StaD (Stable Diffusion) model. The task of this work was to apply the StaD model to create an application for adding images described by text in AR. The task was performed using the Unity tool for creating computer games. Visual Studio 2022 was used to write C# programming language scripts. UI (User Interface) Stable Diffusion Automatic 1111 was used to launch and apply the features offered by StaD.

Key words: Augmented Reality, Stable Diffusion, Unity

ŽIVOTOPIS

Autor ovog predloška Kristian Rücker rođen je u Zagrebu, 9. svibnja 1998. godine, gdje je završio osnovnu školu, te klasičnu gimnaziju. Upisao je Fakultet računarstva, elektrotehnike i informacijskih tehnologija u Osijeku 2017. godine.

Student vrlo dobro poznaje engleski jezik.

Područje interesa: primjene AR-a, programiranje za gaming industriju.