

Generiranje računalnih igara korištenjem umjetne inteligencije

Grgić, Ferdo

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:211:295510>

Rights / Prava: [Attribution 3.0 Unported](#)/[Imenovanje 3.0](#)

Download date / Datum preuzimanja: **2024-08-04**



Repository / Repozitorij:

[Faculty of Organization and Informatics - Digital Repository](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Ferdo Grgić

**GENERIRANJE RAČUNALNIH IGARA
KORIŠTENJEM UMJENTNE
INTELIGENCIJE**

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ferdo Grgić

Matični broj: 45142/16–I

Studij: Poslovni sustavi

**GENERIRANJE RAČUNALNIH IGARA KORIŠTENJEM UMJETNE
INTELIGENCIJE**

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Markus Schatten

Varaždin, lipanj 2023.

Ferdo Grgić

Izjava o izvornosti

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

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Cilj ovoga rada je provjeriti razinu razvijenosti AI alata i njihovu praktičnu upotrebljivost kroz izradu jednostavne računalne video igre. U projektu se većinom vodi prijedlozima Chat GPT-a (engl. Chat Generative Pre-trained Transformer), većinom mu dopuštajući da sam odabire dodatne AI alate koji će se koristiti za izradu 2D grafike, UI elemenata, zvukova u igri itd. Teza autora je da neće biti moguće napraviti cijeli projekt samo uz pomoć AI alata. Zanima nas u kolikoj mjeri su AI alati praktični za korištenje a koliko se umjetna inteligencija mora potpomagati, nadopunjavati i ispravljati njene eventualne greške. Ovaj rad ujedno služi i autoru osobno za trening i uvod korištenja AI alata u poslu, što je vještina za koju se može pretpostaviti da će postati sve bitnija i korisnija za programere i developere u profesionalnoj karijeri.

Ključne riječi: računalne igre; umjetna inteligencija; Unity; DALL-E; generiranje igara; proceduralno generiranje grafike; Bing Image create, Soundraw.

Sadržaj

Sadržaj.....	iii
1. Uvod.....	1
2. Metode i tehnike rada	3
2.1. Unity	3
2.2. Chat GPT.....	4
2.3. DALL - E	4
2.4. Bing Image creator.....	5
2.5. Soundraw.....	5
3. Izrada video igre	6
3.1. Osnovne smjernice i organizacija projekta	6
3.2. Postavljanje scene u Unity-ju	8
3.3. Izrada grafike	9
3.4. Programiranje i izrada igre	16
4. Pozadinska glazba.....	26
5. Završne radnje.....	27
6. Zaključak	28
Popis slika	30
7. Prilog: Izvorni kod	31
7.1. Skripta: GameManager	31
7.2. Skripta: BackgroundMovement	33
7.3. Skripta: PlayerMovement	34
7.4. Skripta: RunningAnimation.....	35
7.5. Skripta: Obstacle.....	35
7.6. Skripta: Spawner.....	36
7.7. Skripta: Wrapper	37

1. Uvod

Tema ovog rada je izrada računalne igre korištenjem raznih alata umjetne inteligencije. Od sada na dalje u radu će se umjesto termina „umjetna inteligencija“ koristiti kraticu AI (artificial intelligence). Svrha ovog rada, s druge strane, je ustvrditi da li je to uopće moguće kao takvo. Ispitati će se da li je moguće da AI odradi tako kompleksan zadatak koji je u teoriji sposoban izraditi, bar gledajući sve njegove dijelove zasebno. Također je od interesa, ako to nije moguće, u kolikom opsegu će autor kao korisnik morati intervenirati u „rad“ AI-ja.

Od kad su AI alati poput Chat GPT-a izašli na tržište, uzrokovali su popriličnu reakciju preko medija, do znanstvenika iz raznih domena znanosti koji se ili zalažu za razvoj AI-a ili upozoravaju o mogućim problemima koje on može stvoriti. Jedna skupina korisnika koji su se našli zahvaćeni tim fenomenom su programeri ili software developeri. Bilo da su vidjeli korisnost AI-a i potencijal u njegovom korištenju u svom poslovanju u obliku alata, ili su se osjetili ugroženi i boje se da bi u nekoj skorijoj budućnosti mogli biti zamijenjeni od iste. Bilo kako bilo, većini je očito da AI donosi promjenu, i da je mudro biti na pravoj strani te promjene kada njen efekt zaista stupi na snagu. U slučaju AI alata to bi značilo da je mudro naučiti ih koristiti na vrijeme. Oni potencijalno donose višestruko povećanje produktivnosti za jednog developera.

Autor je izvanredni student, i već tri godine radi puno radno vrijeme u području informacijskih tehnologija. Dvije godine je radio kako sistemski i mrežni administrator a nakon toga se prebacio u domenu embedded developmenta. Tvrtka u kojoj radi se bavi izradom i prodajom vlastitih proizvoda, mrežnih komponenti koje se koriste u proizvodnim pogonima i automatizaciji istih. Zbog toga se na poslu koristi sa puno različitih tehnologija, programskih domena i programskih jezika.

Kolege na poslu su počeli koristiti Chat GPT, neki su se čak pretplatili na plaćenu noviju verziju alata: Chat GPT-4. Neki u uredu koriste GitHub Copilot, AI alat integriran u GitHub koji radi na predikciji koda koji korisnik piše na osnovi analize tisuća linija koda svih korisnika, ali i specijalne analize vašega koda. Nisu svi alati jednako primjereni za sve vrste korištenja i pogotovo za sve tehnologije i programske jezike koji se koriste, ali oni korisnici koji su našli slučaj upotrebe za koji im je AI alat stvarno koristan govore o velikom povećanju produktivnosti. Takvi pojedinci su rekli da im je AI alat jako olakšao posao tako što umjesto njih odrađuje neke jednostavnije poslove koji se ponavljaju.

Zbog svega ovoga mišljenje je autora da je ovaj završni rad odlična prilika da se na jednom primjeru stvarnog projekta testira koliko su ti alati moćni i sposobni. Autorov hobi, i jedan od razloga zašto se počeo baviti programiranjem, je izrada video igara. Već je imao nekoliko manjih projekata u kojima je izradio neke video igre koje nije nigdje objavljivao nego su većinom služile za učenje. Najviše iskustva u alatima za izradu video igara ima u programskom alatu Unity, a kako je u njemu moguće izraditi puno vrsta i oblika video igara, to ga čini dobrim glavnim alatom za korištenje u ovom projektu.

Cilj je da se uz korištenje AI alata za generiranje programskog koda, slika i zvuka, u programskom alatu Unity napravi funkcionalna video igra. Rad je dokumentarnog karaktera, više služi da popratimo izradu računalne igre od strane AI-ja. Nije fokus na samoj izradi igre nego na praćenju tog procesa AI-ja. Zanima nas oboje sama tehnička kvaliteta izrade (kvaliteta koda, slika, zvukova) kao i kvaliteta prijedloga i odabira koji se odnose na planiranje projekta, odabiri alata, strukturiranje datoteka i projekta itd.

2. Metode i tehnike rada

Za izradu video igre izabran je programski alat Unity koji će služiti za izradu projekta, pisanje programskih skripti i logike igre, dodavanje 2D slika (game sprites), izrada animacija i dodavanje zvukova. Autor je izabrao Unity jer u njemu ima iskustva i jer je vrlo svestran i moćan alat za ovu primjenu.

Ostatak izbora i odluka vezanih uz projekt namjerava se prepustiti Chat GPT-u. Autor će pokušati što manje intervenirati u rad AI-ja, u smislu da će pratiti i poštovati odluke i savjete Chat GPT-a. Kako je već malo koristio taj alat na poslu, jasno je da to neće uvijek biti moguće i da će nekad morati intervenirati. Alat nekad generira loš, nefunkcionalan ili ne svrsishodan računalni kod, tako je za očekivati da će vjerojatno biti i grešaka vezanih uz vođenje projekta. Sve takve greške i autorove intervencije će se evidentirati u radu.

U nastavku ovog poglavlja retroaktivno će biti napisano nešto više o AI alatima koji su se koristili, a koje je preporučio Chat GPT. On će se koristiti kao glavni jezični model s kojim će autor komunicirati u vezi projekta.

2.1. Unity

Unity je jedan od najpopularnijih alata za razvoj računalnih igara. S integriranim razvojnim okruženjem (IDE) omogućava izradu raznih vrsta video igara, ali i animiranih filmova i slično. Podržava različite platforme poput Windowsa, macOS-a, Androida, iOS-a i VR/AR uređaja, čime se osigurava široka dostupnost igara. Unity pruža mnoge alate za izradu 2D i 3D svjetova, nudeći alate za obradu grafike i animacija. Simulacija ponašanja objekata, uključujući fiziku, omogućuje realistične interakcije. Programiranje se vrši na C# jeziku, s podrškom za JavaScript i Boo, pružajući fleksibilnost programerima. [1]

Trgovina sredstava (eng. *Asset Store*) je vrijedan izvor gotovih resursa poput modela i zvukova koji ubrzavaju razvojni proces. Svestranost Unityja čini ga idealnim alatom za razvoj igara za različite platforme. S njegovim širokim spektrom mogućnosti, idealan je za ovaj projekt.

2.2. Chat GPT

ChatGPT je prototip AI chatbota koji se temelji na dijalogu i sposoban je razumjeti prirodni ljudski jezik i generirati impresivno detaljan pisani tekst poput ljudskog. To je najnovija evolucija GPT (eng. *Generative Pre-Trained Transformer*) obitelji AI-jeva za generiranje teksta. [2]

Trenutno javno dostupna verzija temeljena je na GPT-3.5 arhitekturi. Namijenjen je razumijevanju i generiranju tekstualnih odgovora sličnih ljudskom govoru, što ga čini sposobnim za sudjelovanje u konverzacijskim interakcijama s korisnicima. [2]

Chat GPT je treniran na velikom skupu internetskih tekstualnih podataka kako bi stekao široko razumijevanje jezika, činjenica i sposobnosti rasuđivanja. Model koristi duboko učenje uz pomoć transformatorske arhitekture, koja mu omogućuje da uhvati kontekstualne ovisnosti i generira koherentne i kontekstualno relevantne odgovore. Može razumjeti upite, odgovarati na pitanja, pružati objašnjenja i sudjelovati u kreativnim zadacima pisanja. Chat GPT ima razne primjene, uključujući pomoć u pronalaženju informacija, generiranje tekstualnog sadržaja, pružanje prijedloga ili preporuka, pomoć u rješavanju problema i djelovanje kao interaktivni konverzacijski agent. Važno je napomenuti da iako Chat GPT može proizvesti impresivne odgovore, ponekad može generirati netočne ili besmislene informacije. Korisnici bi trebali primijeniti kritičko razmišljanje i procijeniti generirani sadržaj prema točnosti i pouzdanosti. [3]

2.3. DALL - E

DALL-E je model umjetne inteligencije koji kombinira duboko učenje i generativne neuronske mreže kako bi stvorio slike na temelju opisa ili uzoraka. Razvijen od strane OpenAI-a, DALL-E ima sposobnost generiranja fotorealističnih slika koje su potpuno izmišljene i nisu ograničene stvarnim fotografijama ili konkretnim slikovnim podacima. [4]

Ovaj AI model može shvatiti prirodni jezik opisa i interpretirati ga u slikovnom kontekstu kako bi generirao odgovarajuću sliku. Primjerice, ako mu se pruži opis poput "guska s tijelom dinosaura", DALL-E će pokušati stvoriti sliku koja odgovara tom opisu. Također je moguće kontrolirati određene aspekte slike, kao što su boje ili položaj objekata, dodavajući dodatne informacije u opis. DALL-E koristi duboko učenje kako bi naučio složene obrasce i stvorio slikovnu reprezentaciju koja zadovoljava zadane opise. Model je treniran na velikom broju slika s pripadajućim opisima kako bi stekao sposobnost generiranja kvalitetnih slika. [3]

Ovaj napredni umjetnički model ima široki potencijal za različite primjene, uključujući kreiranje vizualnih sadržaja za oglašavanje, video igre, animacije, dizajn i još mnogo toga. Prethodno su modeli DALL-E i njegova naprednija inačica bile besplatne za korištenje i isprobavanje, no od rujna 2023 više nisu.

2.4. Bing Image creator

Bing Image creator je rezultat nove suradnje Microsofta i OpenAI-ja. Microsoft je od ove godine u svoju tražilicu Bing integrirao alate tvrtke OpenAI, i to u obliku chatbot-a koristeći njihov Chat GPT model, i u obliku alata za generiranje slika, koristeći naprednu verziju DALL-E modela. [5]

DALL-E AI model koji je korišten u implementaciji u Bing tražilicu je napredniji od DALL-E 2 koji se mogao besplatno isprobati preko stranice OpenAI-ja. Za korisnike to znači više stvari. On je brži u generaciji slika, ima bolje razumijevanje tekstualnih poticaja (eng. *prompts*), precizniji je u generaciji slika i stvara manje bezličnih i beskorisnih slika. Također stvara čišće slike sa manje vizualnih artefakata. Uza sve to, Bing Image creator je i potpuno besplatan, za razliku od OpenAI-jevog servisa. [5]

Za korištenje se samo potrebno ulogirati u Microsoftov servis. Nije čak potrebno ni koristiti njihov pretraživač Edge, već servis radi na svim većim Internet pretraživačima.

2.5. Soundraw

Soundraw je AI generator glazbe koji pomaže kreatorima da proizvode prilagođenu glazbu za svoje projekte. To je AI alat za stvaranje glazbe od nule ili prilagodbu postojećih pjesama prema potrebama projekta. Korisnici mogu birati iz različitih žanrova, raspoloženja i duljina kako bi generirali prilagođenu glazbu. Zahvaljujući Soundraw-u, kreatori mogu proizvoditi glazbu bez brige o autorskim pravima ili dodatnom utrošku vremena u potrazi za savršenom pjesmom. [6]

Usluga također nudi osobni plan s godišnjom pretplatom od 16,99 dolara mjesečno i mjesečni plan s pretplatom od 19,99 dolara mjesečno. Planovi uključuju neograničenu generaciju pjesama, označavanje omiljenih pjesama i preuzimanje do 50 pjesama dnevno. [6]

Osim toga, uključuje licencu bez autorskih naknada (eng. *royalty-free*) za sav sadržaj koji se može koristiti za YouTube, društvene medije, televiziju, filmove, web oglase, korporativne videozapise, live prijenose, podcaste, radio emisije i reklame, vođene meditacije, audio knjige, streaming glazbe, igre, aplikacije, NFT, pozadinsku glazbu za trgovine i događaje. Licenca je trajna, a korisnici zadržavaju prava za upotrebu pjesama čak i nakon otkazivanja pretplate. [6]

3. Izrada video igre

Projekt je započeo razgovorom sa Chat GPT-om. Na početku su mu postavljena generalna pitanja o ovoj temi rada da vidim koliko je uopće upoznat sa izradom igrica, Unity-jem i drugim AI alatima. Taj kratki prvotni razgovor ostavio je dobar dojam i autoru je dao pouzdanja jer je Chat GPT generalno dobro informiran o svemu što ga je pitao. U nastavku je rad podijeljen na poglavlja po temama izrade video igre.

3.1. Osnovne smjernice i organizacija projekta

Sljedeći cilj je bio dobiti naj grublje smjernice projekta, pa je Chat GPT upitan kakvu video igru je „njemu“ naj jednostavnije napraviti na što smo dobili generički, negativan odgovor:

Kao AI modelu, nemam fizičko tijelo ili mogućnost izravne interakcije s igrama. Stoga, nemam preferenciju ili sposobnost za izradu određene vrste igre. Moja svrha je pružanje informacija, savjeta i podrške u vezi s različitim aspektima razvoja igara, bez obzira na njihovu vrstu ili žanr. Ako imate pitanja o određenim elementima ili tehnikama u izradi igara, slobodno ih postavite, i bit ću ovdje da vam pomognem.

[3]

Bilo je jasno da se treba promijeniti strukturu i pristup pitanja. Chat GPT nije zamišljen da donosi nikakve odluke, zamišljen je da daje savijete i odgovara na direktno postavljena pitanja. Autor je sada postavio pitanje „koje su vrste video igara naj jednostavnije za napraviti u Unity-ju?“ na što je dobio korisniji odgovor:

U Unityju postoji niz jednostavnih vrsta igara koje možete napraviti, posebno ako ste početnik u razvoju igara. Evo nekoliko primjera:

1. Endless Runner (Beskonačna trkačka igra): Ova vrsta igre uključuje kretanje lika koji neprestano trči naprijed, a igrač mora izbjegavati prepreke i prikupljati bodove. Ova vrsta igre može biti jednostavna za implementaciju koristeći osnovne kontrole i skriptiranje.

2. Puzzles (Slagalice): Možete stvoriti različite vrste slagalica, poput slagalica s pomaknutim blokovima, slagalica boja, slagalica sa spojnica i slično. Ove igre se fokusiraju na rješavanje logičkih problema i mogu biti relativno jednostavne za implementaciju.

3. Memory Games (Igre pamćenja): Igre pamćenja zahtijevaju da igrač pamti i upamti određene uzorke, brojeve ili slike, a zatim ih reprodukcira

ili uparuje. Ove igre mogu biti jednostavne za izradu koristeći osnovne animacije, prikazivanje i provjeru podudaranja.

4. Clicker Games (Igre klikanja): Clicker igre temelje se na jednostavnom klikanju na ekran kako bi se postigao napredak ili stekli bodovi. Ove igre mogu biti brzo implementirane koristeći jednostavne događaje klikanja i brojanje bodova.

5. Platformers (Platformne igre): Platformne igre uključuju skakanje i trčanje lika po platformama, izbjegavanje prepreka i prikupljanje predmeta. Iako ova vrsta igre može biti malo složenija za implementaciju od prethodnih primjera, možete početi s osnovnim kontrolama i postupno dodavati složenosti.

Ove su vrste igara dobar početak za početnike, jer za njih ne trebate složene mehanike ili grafičke elemente. Možete početi s malim projektom i postupno ga proširivati kako stječete više iskustva.

[3]

Autor je razmišljao o tome da napravi platformnu igru (eng. *platformer*), ali se ipak odlučio da napravi igru beskonačnog trčanja (eng. *endless runner*). Razlog tome je što je u slučaju „platformne igre“ izrada razina (eng. *levels*) jedan od najbitnijih aspekata igre, a nakon kratkog ispitivanja Chat GPT-a je vidio da nema načina da mu on pomogne u izradi istih dalje od idejnih savjeta ili tehničkih savjeta u Unity-ju. Također me nije znao ni predložiti neki AI alat koji bi mi pomogao u tom zadatku. To znači da bi jedan od bitnijih aspekata igre radio autor, a to nije cilj ovog rada, nego je cilj da u što većem udjelu sami AI alati odrade izradu igre.

Nakon je vrsta igre koja će se raditi odabrana, autor je pitao AI kako da započne raditi „igru beskonačnog trčanja“:

Izrada endless runner igre u Unityju može biti zabavan i uzbudljiv proces. Evo nekoliko koraka kako započeti:

1. Postavljanje projekta: Pokrenite Unity Hub i stvorite novi projekt. Odaberite 2D projekt i odredite ime i lokaciju projekta.

2. Importirajte grafike i resurse: Preuzmite ili izradite 2D grafike koje će se koristiti u vašoj endless runner igri. To može uključivati grafike za likove, pozadinu, platforme, prepreke i ostale elemente. Importirajte te resurse u Unity tako da ih povučete i ispustite u odgovarajući folder unutar Unity projekta.

3. Postavljanje scene: Kreirajte novu scenu u Unityju. Dodajte pozadinu, platforme i ostale vizualne elemente koji čine vašu scenu. Postavite početnu poziciju i veličinu igračkog lika.

4. Dizajnirajte kretanje lika: Stvorite skriptu koja će upravljati kretanjem lika. Možete koristiti Rigidbody2D komponentu za simuliranje fizike kretanja. Definirajte pravila kretanja, kao što su skakanje i izbjegavanje prepreka.

5. Dodajte generiranje platformi: Implementirajte sustav generiranja platformi koje neprestano dolaze prema liku. Možete koristiti kod za stvaranje novih platforma i postavljanje njihove pozicije i oblika. Osigurajte da platforme budu generirane na vrijeme kako bi se igraču pružio neprekidan niz izazova.

6. Postavite kretanje pozadine: Dodajte kretanje pozadine kako bi se stvorio osjećaj trčanja. To može uključivati pomicanje pozadinske slike ili paralaksni efekt koji stvara iluziju dubine.

7. Implementirajte logiku sudara: Stvorite sustav provjere sudara kako biste detektirali kada lik udari u prepreke ili padne s platforme. Na temelju toga možete definirati ponašanje igre, poput završetka igre, prikupljanja bodova ili ponovnog pokretanja.

8. Dodajte zvuk i vizualne efekte: Učinite svoju igru živopisnijom dodavanjem zvučnih efekata i vizualnih efekata. To može uključivati zvukove skakanja, padanja, prikupljanja predmeta i eksplozija. Također možete koristiti efekte poput bljeska ili čestica kako biste naglasili važne događaje u igri.

9. Testirajte i prilagodite: Redovito testirajte svoju igru kako biste provjerili igrivost, izazovnost i eventualne probleme.

Ako primijetite nedostatke ili potrebne izmjene, prilagodite svoju igru da biste je poboljšali.

Ovo su samo osnovni koraci za izradu endless runner igre. Tijekom razvoja možete dodavati složenosti, dodatne mehanike i poboljšanja kako biste postigli željeni rezultat. Slijedite dokumentaciju Unityja, istražujte online resurse i eksperimentirajte s različitim idejama kako biste izgradili jedinstvenu i zabavnu endless runner igru. Sretno s razvojem!

[3]

Kako ima iskustva sa izradom igara u Unity-ju, autor može primijetiti da je ovo sasvim realan hodogram izrade takve igre, tako da kreće po redu, kako je AI predložio. Prva točka na listi je inicijalno postavljanje projekta u Unity-ju.

3.2. Postavljanje scene u Unity-ju

Kako bih započeli sastavljati svoju igricu, prvo u programskom alatu Unity treba postaviti scenu. Autor je upitao Chat GPT kako da to napravi. Ovo je odgovor koji je dobio:

1. Otvorite Unity i stvorite novi projekt ako već niste.
2. U Unityjevom uređivaču, navigirajte do "Project" prozora. Desnom tipkom miša kliknite u "Assets" mapu (ili željenu mapu) i odaberite "Create" > "Scene" kako biste stvorili novu scenu za svoju beskonačnu trkačku igru.
3. Dvaput kliknite na novo stvorenu datoteku scene kako biste je otvorili u Unityjevom uređivaču.
4. Započnite postavljanje kamere: U Hierarchy prozoru desnom tipkom miša kliknite i odaberite "3D Object" > "Camera" kako biste stvorili kameru. Prilagodite njenu poziciju, rotaciju i postavke u Inspector prozoru kako biste dobili željeni pogled za svoju igru. Pazite da je kamera postavljena da prati lik igrača.
5. Stvorite lik igrača: Desnom tipkom miša kliknite u Hierarchy prozoru i odaberite "3D Object" ili "2D Object" ovisno o stilu vaše igre. Možete odabrati osnovne oblike poput kocke (Cube) ili kapsule (Capsule) za predstavljanje lika na početku. Prilagodite njegov izgled, veličinu i

poziciju prema potrebi. Dodajte komponentu Rigidbody kako biste omogućili kretanje temeljeno na fizici.

6. Postavite okolinu: Stvorite ili uvezite svoje pozadinske i platformne elemente. Možete prevlačiti slike ili 3D modele iz mape "Assets" u scenu. Pozicionirajte ih prema željenom rasporedu za vašu beskonačnu trkačku igru. Razmislite o dodavanju dodatnih objekata poput prepreka, sakupljivih predmeta ili dekorativnih elemenata.

7. Dodajte skripte i komponente: Dodajte skripte na lik igrača i druge objekte u igri kako biste upravljali njihovim ponašanjem. Na primjer, možda će vam trebati skripte za obradu korisničkog unosa, kretanje lika, detekciju sudara i brojanje bodova. Možete stvoriti nove skripte ili koristiti već postojeće kako biste implementirali željenu funkcionalnost.

8. Testirajte i iterirajte: Pustite scenu u Unityjevom uređivaču kako biste testirali početnu postavku. Prilagodite kameru, kretanje igrača, platforme i druge elemente prema potrebi. Iterirajte dizajn i igru sve dok ne budete zadovoljni rezultatima.

[3]

Autor je započeo po natuknicama koje sam dobio. Kod četvrte natuknice se može primijetiti da se opis koji smo dobili malo razlikuje od sučelja u Unity, ali tome je vjerojatno razlog to što se autorova verzija programa malo razlikuje od one sa kojom je Chat GPT upoznat, vjerojatno je riječ o malo novijoj ili starijoj verziji. Bez obzira na to, bez većih problema se može pronaći ono što je opisano.

3.3. Izrada grafike

Spriteovi (od eng. *Sprite*, duh) su dvodimenzionalne slike ili grafike koje se koriste u računalnim igrama ili animaciji kako bi se prikazali likovi, objekti ili dijelovi okoline. Spriteovi su često mali dijelovi većih slika koje se koriste za animaciju ili interakciju u igri. [7]

Autor je pitao Chat GPT da mu predloži neke AI alate kojima bi mogao generirati 2D grafiku (*sprites*) koju će koristiti u izradi video igre:

Postoji nekoliko AI alata koji mogu biti korisni za generiranje 2D spriteova za upotrebu u izradi igara u Unityju. Evo nekoliko popularnih alata koji možete istražiti:

1. DALL-E: DALL-E je AI model razvijen od strane OpenAI-a koji može generirati slike na temelju teksta. Iako je njegova primarna svrha generiranje slika u stilu umjetničkih djela, može biti koristan za stvaranje jedinstvenih 2D spriteova na temelju opisa ili konceptualnih ideja.

2. GANPaint Studio: GANPaint Studio je alat koji koristi generativne suparničke mreže (GAN) kako bi omogućio korisnicima da mijenjaju, dodaju ili

uklanjaju elemente iz postojećih slika. Može biti koristan za uređivanje i prilagođavanje postojećih spriteova.

3. RunwayML: RunwayML je platforma koja pruža pristup različitim AI modelima za umjetnost i kreativnost. Može se koristiti za generiranje i manipulaciju 2D spriteova te eksperimentiranje s različitim stilovima i efektima.

4. ArtBreeder: ArtBreeder je online platforma koja kombinira različite slike kako bi generirala nove vizualne rezultate. Može se koristiti za eksperimentiranje s različitim stilovima, varijacijama i kombinacijama kako bi se stvorili jedinstveni 2D spriteovi.

5. Pix2Pix: Pix2Pix je AI model koji se može koristiti za mapiranje između slika, što je korisno za generiranje novih spriteova na temelju postojećih crteža ili skica. Može biti koristan za brzo pretvaranje grubih skica u detaljnije i obojene spriteove.

Svaki od ovih alata ima svoje prednosti i specifičnosti. Preporučujem istraživanje i eksperimentiranje s različitim alatima kako biste pronašli onaj koji najbolje odgovara vašim potrebama i estetici vaše igre.

[3]

Nakon prvotnog istraživanja na internetu o svakom alatu koji je Chat GPT predložio, autor je ocijenio da mu u ovu svrhu naj bolje odgovaraju alati DALL-E i RunwayML. Problem je u tome što oba alata nisu potpuno besplatni, ali oba nude nekakav probni period (oko 100 generacija, stvaranja). Drugi je problem što DALL-E od ranije u 2023. više ne nudi ni taj probni period novo napravljenim računima. Od ove godine se plaća. I treći je problem taj što je autor već imao napravljen račun za DALL-E i već je iskoristio dio besplatnih bodova za generaciju. Svejedno, autor je krenuo generirati grafička sredstva (eng. *game assets*) za korištenje u svojoj igri. U ovakvom projektu trebaju sljedeća sredstva: 1. slike za pozadinu 2. sprite za platformu, tj. „pod“ po kojoj će igrač trčati, 3. tekstura poda ispod platforme, 4. razne prepreke koje će igrač preskakivati, 5. spriteovi za igračevog lika i njegove animacije.

Najveći problem je stvarala generacija ili stvaranje spriteova za lika igrača i dovoljno njegovih sličica da bi se mogla složiti bar nekakva animacija trčanja. Samo za to autoru je trebalo više sati generiranja i svi besplatni bodovi za DALL-E i RunwayML. Ono što smo mogli saznati iz ovog iskustva je da AI alati za generiranje slika imaju velikih problema sa generiranjem ove vrste sličica koje bi se mogle koristiti kao *game assets*, pogotovo imaju problema sa generacijom više uzastopnih povezanih sličica. Očito je da su ovi AI alati trenirani za generaciju imitacije umjetničkih djela i fotografija, i zamišljeni su da se koriste za zabavu, ili maksimalno za davanje inspiracija u obliku konceptne umjetnosti za realne projekte. Nisu zamišljeni za generaciju ovako specifičnog slučaja upotrebe. Ipak uspjeli smo dobiti relativno iskoristivu sliku koristeći sljedeći poticaj ((eng. *prompt*) je kratki tekstualni ili vizualni ulaz koji se koristi kako bi se potaknula generacija određenih sadržaja od strane AI modela):

“game assets, 2D sprites, scifi style, pixel art, main character, running pose, male, helmet on, cyberpunk style”



Slika 1. „Spriteovi lika igrača“ [8]

Ovu sliku autor je u „Photoshopu“ izrezao i obradio pojedinačne sličice i pretvorio ih u spriteove koje se mogu koristiti u izradi igre. Na isti način su generirana ostala grafička sredstva za igru.

Jedina je razlika što nadalje, zbog toga što više nisu besplatni, autor više nije koristio DALL-E i RunwayML nego je pronašao besplatnu alternativu: Bing Image create. Ovaj alat je integriran u Bingovu Internet tražilicu i pogonjen je DALL-E-om. Potpuno je besplatan i mogli bismo čak reći da je s njim autor imao najviše uspjeha, iako valja napomenuti da je generacija lika igrača ipak bio najkompliciraniji zadatak. U nastavku će biti stavljen po jedan primjer slika koje su generirane i poticaje koji su se za njih koristili u AI alatu. U samom projektu nalazi se više varijacija slika koje su generirane.



Slika 2. „Pozadinske slike“ [9]

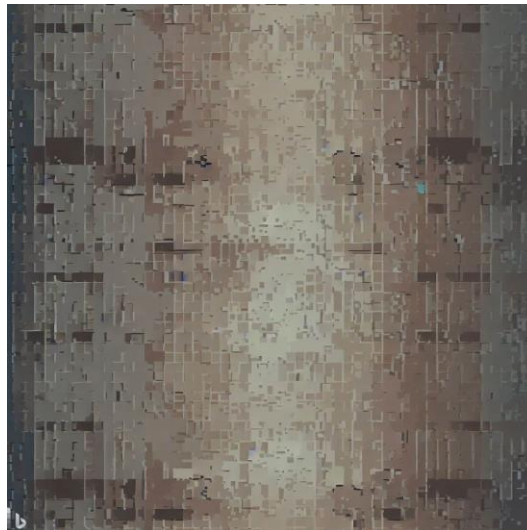
Bing Image creator prompt za pozadinske slike (22 komada):

„2D game background, front view, 2D perspective old downtown building, evening setting, grafitti, in style of TMNT, in style of a sidescroller game“



Slika 3. „Platforme za pod“ [9]

Bing Image creator prompt za platforme za pod (2 komada): „2D video game sidescroller platform sprite, concrete texture, city, urban, TMNT style, sidescroller game style, blank white background“



Slika 4. „Tekstura poda“ [9]

Bing Image creator prompt za teksturu poda (1 komad):

„video game sprite asset, urban concrete texture, in style of 32 bit, symmetrical texture“



Slika 5. „Automobil“ [9]

Bing Image creator prompt za automobil (5 komada):

„2D video game sidescroller car sprite, cheap car, city, urban, TMNT style, sidescroller game style, blank white background“



Slika 6. „kanta za smeće“ [9]

Bing Image creator prompt za kantu za smeće (4 komada):

„2D video game sidescroller garbage container sprite, trash container, urban, TMNT style, sidescroller game style, blank white background“



Slika 7. „Kutije“ [9]

Bing Image creator prompt za Kutije (2 komada):

„2D video game sidescroller wooden boxes sprite, urban, TMNT style, sidescroller game style, blank white background“



Slika 8. „Hrpa smeća“ [9]

Bing Image creator prompt za hrpu smeća (4 komada):

„2D video game sidescroller trash heap sprite, trash heap on the ground, urban, TMNT style, sidescroller game style, blank white background“



Slika 9. „Ograda“ [9]

Bing Image creator prompt za ogradu (2 komada):

„2D video game sidescroller metal fence sprite, simple fence, urban, TMNT style, sidescroller game style, blank white background“

Sa ovim sredstvima za izradu igre imamo sve što nam treba od grafičkih komponenti. Valja napomenuti da oblik u kojem smo dobili slike nije pogodan za korištenje u izradi video igre, nego su sve slike zahtijevale daljinu obradu u alatima za obradu slike.

Za taj zadatak korišten je besplatni alat Gimp. Svaku sliku trebalo je izrezati tako da dobijemo samo motiv koji nas zanima, izbrisati pozadine kako bi dobili samo sprite koji možemo koristiti u izradi igre, i to spremi u .png formatu (originalno AI generira .jpeg format i to se ne može drugačije podesiti) koji se može koristiti u Unity alatu.

Ovaj postupak daljnje obrade slika postavio je presedan za način rada koji je kasnije korišten do okončanja projekta. Vidimo da AI alati mogu na jedan način pomoći korisniku, ali ne daju gotov proizvod i korisnik ipak mora dalje preraditi izlaz (eng. *output*) AI alata.

3.4. Programiranje i izrada igre

Nakon što smo dobili sve grafičke komponente igre započinjemo sa njenom izradom u Unity-ju. Prethodno smo već postavili scenu po natuknicama Chat GPT-a. Sada u scenu počinjemo dodavati vizualne elemente. Autor ih je prvo samo grubo namjestio, bez funkcionalnosti, da se podsjeti kako funkcionira Unity.

U ovom koraku autor je u svojim prethodnim projektima morao gledati video vodiče na YouTube, ali sada možemo primijetiti da je za ovu svrhu odličan Chat GPT. Autor mu je postavljao kratka, jednostavna pitanja poput „kako da na objekt dodam *Rigidbody* komponentu“ ili „kako da od *parent* objekta napravim *prefab* objekt“, na što je dobio jednostavne i deskriptivne opise. Predstavlja se teza da je Chat GPT naj bolji u ovoj ulozi pametne tražilice, osjetno ubrzavajući proces učenja i traženja informacija.

Kada smo u Unity-ju posložili scenu i napravili početne *prefab* objekte (skraćenica od eng. *prefabricated object*) bilo je vrijeme za izradu prvih C# skripti. Prvo nam je trebala skripta za igrača. *Prefab* je u kontekstu razvoja video igara ili 3D modeliranja, objekt ili skup objekata koji je unaprijed kreiran i spremljen kao predložak za ponovnu upotrebu.

Odmah možemo zapaziti potencijalni problem kojeg ćemo imati sa Chat GPT u vezi ove funkcionalnosti. U većini 2D igara lik igrača se može kretati u X i Y osima, ili nekoj kombinaciji istih. U „*infinite runner*“ stilu igre, igrač se zapravo ne miče, on samo skače. Iluzija pokreta se stvara micanjem pozadine iza lika i njegovim animacijama, dok on zapravo stoji na mjestu.

poticaje za kod autor je Chat GPT pitao na engleskom, jer je znao da će on neke riječi pretvoriti u varijable, a one nam moraju biti iskoristive u Unity-ju. Također možete primijetiti da poticaji ne moraju biti gramatički točni. Autor ih nekad namjerno oblikuje na način da AI iz njih lakše generira kod, kao u stilu pseudo koda.

Pitamo Chat GPT da nam napiše skriptu na sljedeći način:

„Možeš li mi napisati c# skriptu za kretanje igrača u beskonačnom trkaču gdje igrač može samo skakati, a brzina igrača će biti u kretanju pozadine“

Dobili smo većinom iskoristiv kod, međutim bilo je nekih detalja koje je trebalo ispraviti. Prva verzija koju smo dobili koristila je tip varijabla Vector3 i 3D koordinate, pa smo pitali AI da nam preradi kod u onaj za 2D prostor, koristeći Vector2 i Rigidbody2D tipove varijabli. Slijedi kod koji smo dobili:

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public float jumpForce = 5f;
    public float forwardSpeed = 5f;
    private Rigidbody2D rb;
    private bool isJumping = false;

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    private void Update()
    {
        // Check for jump input
        if (Input.GetKeyDown(KeyCode.Space) && !isJumping)
        {
            isJumping = true;
        }
    }
}
```



```

        Jump();
    }
}

private void FixedUpdate()
{
    // Move the player forward based on the background movement
    float movement = forwardSpeed * Time.fixedDeltaTime;
    rb.velocity = new Vector2(movement, rb.velocity.y);
}

private void Jump()
{
    // Apply upward force to make the player jump
    rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
}

private void OnCollisionEnter2D(Collision2D collision)
{
    // Check for collision with the ground or platforms to allow
the player to jump again
    if (collision.gameObject.CompareTag("Floor"))
    {
        isJumping = false;
    }
}
}

```

[3]

Nakon što smo implementirali skriptu na objekt u igri - igrača, krećemo dalje. Sljedeća skripta koju generiramo bila je ona za kretanje pozadinskih slika i platformi poda.

Poticaj koji se koristio je: „Možeš li mi napisati skriptu za pozadinu i pod da ubrzavaju do zadane vrijednosti kroz vrijeme.“

```

using UnityEngine;

public class BackgroundMovement : MonoBehaviour
{
    public float accelerationRate = 0.5f;
    public float targetSpeed = 10f;

    private float currentSpeed = 0f;

    private void Update()
    {
        // Accelerate the background and floor movement
        currentSpeed = Mathf.MoveTowards(currentSpeed, targetSpeed,
accelerationRate * Time.deltaTime);

        // Move the background and floor based on the current speed
        transform.Translate(Vector3.back * currentSpeed *
Time.deltaTime);
    }
}

```

[3]

Ovo nije bilo točno što želimo. Ponovno koristi 3D prostor i pozadina se kreće u natrag. Nakon nekoliko pod pitanja i odgovora autor je sam modificirao skriptu na način da koristi jednostavniji izračun brzine i da se pozadina kreće sa desna na lijevo i dobili smo sljedeći kod:

```
using UnityEngine;

public class BackgroundMovement : MonoBehaviour
{
    private float currentSpeed;

    private void Update()
    {
        // Accelerate the background and floor movement
        currentSpeed = GameManager.Instance.gameSpeed;
        // Move the background and floor based on the current speed
        transform.Translate(Vector3.left * currentSpeed *
Time.deltaTime);
    }
}
```

[3]

Zatim trebamo skriptu kojom bi dobili efekt beskonačne pozadine i platforme poda. Autor je ovo zamislio na način da se oni „omotavaju“ oko kamere, tj. kada izađu iz kadra na lijevu stranu ekrana, ponovno se stvore negdje izvan pogleda na desnoj strani ekrana, pa će tako opet doći u kadar i proći kraj igrača. Al je generirao većinom gotovu skriptu, samo je trebalo promijeniti imena varijabli:

```
using UnityEngine;

public class Wrapper : MonoBehaviour
{
    public float checkX = -26f; // The target x value to check against
    public float newX = 17f; // The new x value to move the prefab to

    private void Update()
    {
        // Check if the prefab has reached the target x value
        if (transform.position.x <= checkX)
        {
            // Move the prefab to the new x value
            Vector2 newPosition = new Vector2(newX,
transform.position.y);
            transform.position = newPosition;
        }
    }
}
```

[3]

Sljedeće što je trebalo napraviti bio je sustav animiranja spriteova lika igrača. Autor je prvo pitao Al kako da to napravim na što je dobio natuknice sa uputama. Nakon što je pokušao pratiti uputstva koja sam dobio, vidio je da ona jednostavno nisu ispravna i ne može se odraditi to što mu je Al predložio da radi. Nakon što je autor Al-ju rekao da se ne može napraviti to što

ga traži, i rekao mu u čemu je problem, Al se ispričao i dao seriju drugih, različitih natuknica sa uputama koje su sadržavale i skriptu:

I apologize for the confusion. In Unity, you cannot directly drag and drop sprites into the Animation window to create an animation clip. Instead, you need to create a sprite sheet or individual animation frames and use them to create the animation. Here's an updated step-by-step guide:

1. Import the sprites for your running animation into your Unity project. Make sure the sprites are organized in sequential order, representing each frame of the animation.

2. Create a new folder in the Unity Project window to organize your animation assets.

3. In the newly created folder, select the sprites that make up your running animation frames. Right-click on the selection and choose "Create > Sprite Renderer". This will create a new Sprite Renderer component with the selected sprites as frames.

4. Select the newly created Sprite Renderer component, and in the Inspector window, set the "Sprite Mode" to "Multiple" if you have a sprite sheet, or select individual sprites if you have them separated.

5. Configure the "Sprite Mode" settings accordingly, such as setting the sprite sheet's "Sprite Editor" settings or defining individual sprites' pivot and border values.

6. Create an empty GameObject in your scene to serve as a container for your running animation.

7. Attach the Sprite Renderer component to the empty GameObject by dragging and dropping it onto the GameObject or using the "Add Component" button in the Inspector window.

8. Create a new C# script (e.g., "RunningAnimation") and attach it to the GameObject containing the Sprite Renderer component.

9. In the script, write code to animate the running animation by swapping sprites over time. Here's a basic example:

```
```csharp
using UnityEngine;

public class RunningAnimation : MonoBehaviour
{
 public Sprite[] runningFrames; // Array of running animation frames
 public float frameRate = 0.1f; // Delay between frame changes

 private SpriteRenderer spriteRenderer;
 private int currentFrameIndex = 0;
 private float timer = 0f;

 private void Start()
 {
 spriteRenderer = GetComponent<SpriteRenderer>();
 }
}
```

```

 }

 private void Update()
 {
 timer += Time.deltaTime;

 if (timer >= frameRate)
 {
 timer -= frameRate;
 currentFrameIndex = (currentFrameIndex + 1) %
runningFrames.Length;
 spriteRenderer.sprite = runningFrames[currentFrameIndex];
 }
 }
}
...

```

10. Assign the running animation frames to the "runningFrames" array in the Inspector window of the GameObject with the RunningAnimation script. Make sure the frames are in the correct order.

11. Test the running animation by running the scene. The script will cycle through the running frames at the specified frame rate, creating the animation effect.

This approach manually changes the sprite rendered by the Sprite Renderer component over time to create the running animation. Adjust the frame rate, sprite array, and any additional animation logic in the script to suit your specific animation needs.

Again, I apologize for the previous incorrect information, and I hope this updated guide helps you create the running animation using sprites in Unity.

[3]

Kad smo pratili ove upute, sve je radilo iz prvog pokušaja. nije trebalo ni modificirati imena varijabla koja je morao mijenjati u svim ostalim skriptama.

Sljedeća skripta koju smo izradili bila je za nasumično generiranje prepreka koje igrač mora preskakati. Ovo je najkompliciranija skripta u igri i s njom je bilo podosta problema. Autor je puno puta pokušao potaknuti Aida generira ovu skriptu, ali na kraju je za ovaj projekt iskorištena skripta iz tuđeg projekta sa GitLab-a, koju je autor podosta modificirao sam. Do te točke autor je generirao skriptu ili ju sam napisao i ponovno izbrisao i krenuo ispočetka više puta. Jedna od tih verzija koda koja je zapravo funkcionirala, a bila je generirana od strane Alja, pa naknadno modificirana je ova:

```

using UnityEngine;

public class Spawner2 : MonoBehaviour
{
 public GameObject[] obstaclePrefabs; // Array of obstacle prefabs
 private Vector2 spawnPoint; // Point where obstacles will
be spawned
 public float minSpawnDelay = 1f; // Minimum delay between spawns
 public float maxSpawnDelay = 2f; // Maximum delay between spawns

 private float nextSpawnTime; // Time of the next spawn

```

```

private void Start()
{
 // Set the next spawn time initially
 nextSpawnTime = Time.time + 2;
}

private void OnEnable() {
 Invoke(nameof(Spawn), Random.Range(minSpawnDelay,
maxSpawnDelay));
}

private void OnDisable() {
 CancelInvoke();
}

private void Spawn() {
 // Spawn a random obstacle prefab
 int randomIndex = Random.Range(0, obstaclePrefabs.Length);
 GameObject obstacle =
Instantiate(obstaclePrefabs[randomIndex]);
 obstacle.transform.position += transform.position;

 Invoke(nameof(Spawn), Random.Range(minSpawnDelay,
maxSpawnDelay));
}
}

```

[3]

Ova skripta funkcionira i sadržavala je sve glavne komponente i funkcije koje su nam trebale. Na mjestu objekta unutar igre u vremenskom intervalu 1-2 sekunde, treba se stvoriti 1 od 14 različitih *prefab* prepreka koje se kreću sa desna na lijevo. Problem sa skriptom, i zašto je nismo mogli koristiti, je taj što se prepreke odabiru nasumično. To nije dobro za igru jer se brzina igre povećava kako vrijeme prolazi, a sa time što se igrač „brže kreće“ kako vrijeme prolazi, njegov luk skoka postaje veći pa mu je zato lakše preskočiti veće prepreke. Zato je u početku igre, dok je brzina još mala, doslovno nemoguće preskočiti neke prepreke. Za taj problem imali smo 2 moguća rješenja. Prvo je da postavimo da igra u početku kreće od veće brzine, a drugo je da implementiramo sistem stvaranja prepreka tako da različite prepreke imaju različite vjerojatnosti da će se pojaviti u igri. Ovo drugo rješenje je puno elegantnije tako da smo odabrali njega. Nažalost, ono je i znatno teže za implementirati i programirati, i to nismo mogli napraviti uz pomoć AI-ja.

Na kraju je autor na GitLab-u pronašao projekt izrade *infinite runner* igre koja kopira igru dinosaura u Googleovoj tražilici Chrome koja se pojavljuje kada nema internetske veze. Uzeli smo njegov sistem stvaranja prepreka i modificirali kod za korištenje u svojoj igri. Slijedi ta skripta:

```

using UnityEngine;

public class Spawner : MonoBehaviour
{
 [System.Serializable]
 public struct SpawnableObject

```

```

 {
 public GameObject prefab;

 public float spawnChance;
 }

 public SpawnableObject[] objects;

 public float minSpawnRate = 1f;
 public float maxSpawnRate = 2f;

 private void OnEnable() {
 Invoke(nameof(Spawn), Random.Range(minSpawnRate, maxSpawnRate));
 }

 private void OnDisable() {
 CancelInvoke();
 }

 private void Spawn() {
 float spawnChance = Random.value;

 foreach (var obj in objects) {
 if (spawnChance < obj.spawnChance) {

 GameObject obstacle = Instantiate(obj.prefab);
 obstacle.transform.position += transform.position;
 break;
 }

 spawnChance -= obj.spawnChance;
 }

 Invoke(nameof(Spawn), Random.Range(minSpawnRate, maxSpawnRate));
 }
}

```

[3]

U ovom GitHub projektu autor je pronašao još jednu skriptu koju smo koristili u svojoj igri, nakon modifikacija. Ona je GameManager, i služi za upravljanje nekih stanja igri, a to što su početak i kraj, brojanje bodova i UI. U njoj se određuje i brzina igre. Slijedi ta skripta u obliku u kojem je korištena u igri:

```

using TMPro;
using UnityEngine;
using UnityEngine.UI;

[DefaultExecutionOrder(-1)]
public class GameManager : MonoBehaviour
{
 public static GameManager Instance { get; private set; }

 public float initialGameSpeed = 5f;
 public float gameSpeedIncrease = 0.1f;
}

```

```

public float gameSpeed { get; private set; }

public TextMeshProUGUI scoreText;
public TextMeshProUGUI hiscoreText;
public TextMeshProUGUI gameOverText;
public Button retryButton;

private PlayerMovement player;
private Spawner spawner;

private float score;

private void Awake()
{
 if (Instance != null) {
 DestroyImmediate(gameObject);
 } else {
 Instance = this;
 }
}

private void OnDestroy()
{
 if (Instance == this) {
 Instance = null;
 }
}

private void Start()
{
 player = FindObjectOfType<PlayerMovement>();
 spawner = FindObjectOfType<Spawner>();

 NewGame();
}

public void NewGame()
{
 Obstacle[] obstacles = FindObjectsOfType<Obstacle>();

 foreach (var obstacle in obstacles) {
 Destroy(obstacle.gameObject);
 }

 score = 0f;
 gameSpeed = initialGameSpeed;
 enabled = true;

 player.gameObject.SetActive(true);
 spawner.gameObject.SetActive(true);
 gameOverText.gameObject.SetActive(false);
 retryButton.gameObject.SetActive(false);

 UpdateHiscore();
}

public void GameOver()
{
 gameSpeed = 0f;

```

```

 enabled = false;

 player.gameObject.SetActive(false);
 spawner.gameObject.SetActive(false);
 gameOverText.gameObject.SetActive(true);
 retryButton.gameObject.SetActive(true);

 UpdateHiscore();
 }

 private void Update()
 {
 gameSpeed += gameSpeedIncrease * Time.deltaTime;
 score += gameSpeed * Time.deltaTime;
 scoreText.text = Mathf.FloorToInt(score).ToString("D5");
 }

 private void UpdateHiscore()
 {
 float hiscore = PlayerPrefs.GetFloat("hiscore", 0);

 if (score > hiscore)
 {
 hiscore = score;
 PlayerPrefs.SetFloat("hiscore", hiscore);
 }

 hiscoreText.text = Mathf.FloorToInt(hiscore).ToString("D5");
 }
}

```

[3]

Trebalo je razmisliti o tome da li da se ovaj centralni dio funkcionalnosti odradi na ovaj način, u jednoj skripti, ili ne. Tehnički govoreno, ovakva izvedba nije bila potrebna. Igra bi bila funkcionalna i bez toga, ali bila bi puno rudimentarnija i manje zabavna, a na kraju i manje igriva. Autor je već ranije u projektu donosio slične odluke, hoće li ostaviti odluke i kod koji mu je generirao AI, iako zna da to nije dobro rješenje i da će kasnije stvoriti probleme ili da će učiniti igru lošijom, i odlučio se se protiv toga. Odlučio je da ako se na jednostavan način može poboljšati igru, to će se i učiniti.

Sa ovom skriptom igra je bila većinski funkcionalno gotova. Preostalo je samo bilo dodati pozadinsku glazbu.



## 4. Pozadinska glazba

Chat GPT-u je postavljeno pitanje zna li neke AI alate koje bi se moglo koristiti za izradu pozadinske glazbe. Ovo je odgovor koji smo dobili:

1. OpenAI's MuseNet: MuseNet je AI model razvijen od strane OpenAI koji može generirati originalnu glazbu u različitim stilovima i žanrovima. Može se koristiti za stvaranje melodija, harmonija i aranžmana.

2. Jukedeck: Jukedeck je platforma koja koristi AI za generiranje prilagođene glazbe za videozapise, reklame i druge medijske projekte. Omogućuje korisnicima da odaberu žanr, tempo i raspoloženje, a zatim generira odgovarajuću glazbu.

3. Amper Music: Amper Music je alat koji koristi AI za generiranje prilagođene glazbe za različite namjene, uključujući videoigre, filmove, reklame i druge medijske projekte. Omogućuje korisnicima da odaberu stil, tempo i instrumente kako bi generirali željeni zvuk.

4. Google's NSynth: NSynth je eksperimentalni projekt koji koristi neuronske mreže kako bi sintetizirao nove zvukove spajanjem različitih instrumenata. Može generirati zvukove koji kombiniraju karakteristike različitih instrumenata i stvara jedinstvene zvučne teksture.

5. Magenta: Magenta je istraživački projekt Googlea koji istražuje primjenu umjetne inteligencije u glazbi i umjetnosti. Nudi niz alata i modela za generiranje glazbe, uključujući alate za stvaranje melodija, ritmova i harmonija.

Ovi alati koriste napredne tehnike strojnog učenja kako bi generirali glazbu i zvukove, ali imajte na umu da konačni rezultati ovise o postavkama i smjernicama koje im pružite.

[3]

Nakon internetske pretrage o tim alatima, bilo je očito da neki od njih više ni ne postoje, a ostali ili više nisu dostupni za koristiti, ili nisu besplatni. Razlog ovome je taj što je Chat GPT 3.5 koji je korišten, odsječen od izvora informacija 2021., tako da su mu informacije zastarjele. Autor je ipak morao sam potražiti AI alat koji će koristiti.

Nakon što što je isprobao par alata, pronašao je onaj koji se koristio u projektu. Zove se Soundraw. Radi na način da odaberete svrhu za koju vam treba glazbeni zapis, pa odaberete „osjećaj“ koji glazba treba imati, pa odaberete žanrove u koje treba spadati. Može se navesti i raspon BPM (eng. *beats per minute*), i specificirati glazbala koja će se koristiti, ali mi u konkretnom slučaju nismo, željeli smo da nam alat ponudi veći izbor različite glazbe.

Alat vam ponudi veći broj glazbenih zapisa koji svaki ima neku kombinaciju karakteristika koju ste mu zadali. Svaki zapis možete preslušati, a oni su podijeljeni na odsječke ili sektore. Sektori su označeni po „razini energije“ (niska, srednja, visoka, vrlo visoka), koju možete mijenjati u pravom vremenu kako bi dobili glazbeni zapis koji vam više odgovara.

Na ovaj način smo napravili glazbeni klip, svrha: pozadinska glazba, osjećaj: euforično, žanr: funk. Implementirao sam pozadinsku glazbu u Unity. Time je u projekt dodana zadnja funkcionalnost.

## 5. Završne radnje

Zadnje što je preostalo za napraviti je testiranje igre igranjem (eng. *Playtesting*), traženje i ispravljanje računalnih grešaka (eng. *Bugs*). Za ovaj korak nisu korišteni AI alati. Igru su igrale i testirale dvije osobe, i naišle su na nekoliko stvari koje je trebalo ispraviti.

Glavni problemi koji je trebao riješiti su bili podesiti omotavanje (eng. *wrapping*) platformi poda, tako da se ne vidi, podešavanje brzine igre i kretanja pozadine kako bi osigurao da težina igre nije pre teška ni pre lagana, a i da bude vizualno lakše gledati igru. Prije je imala nekakav vizualni hipnotički efekt. Zadnja stvar koju je trebalo mijenjati su *hitbox-ovi* prepreka. Kvadratni *hitboxovi* su promijenjeni u poligonske, koje je trebalo ručno namještati za svaki sprite.

Kada su ti problemi bili riješeni projekt je bio završen. Bilo je vrijeme za spremati zadnju verziju projekta i kompajlirati izvršne datoteke. S time je izrada igre bila gotova.

## 6. Zaključak

Premisa ovog rada bila je provjeriti tezu da je moguće uz generiranje sadržaja AI alatima napraviti računalnu video igru.

Za izradu igre trebalo je generirati grafičke sadržaje za koje smo koristili DALL-E platformu od tvrtke OpenAI, i Bing image create, Microsoftovu tražilicu pogonjenu malo naprednijim DALL-E modelom od onoga koji se nudi korisnicima na njihovoj vlastitoj stranici. Za tehnička pitanja, savjete vezane za oblikovanje projekta i generiranje C# skripti za igru smo koristili Chat GPT 3.5 također od OpenAI-ja. On je trenutno najpoznatiji i korišteniji jezični AI model, iako u zadnje vrijeme sve više konkurenata dolazi na tržište. Za izradu pozadinske glazbe smo koristili Soundraw, AI alat tvrtke Zendesk koji služi za generiranje zvučnih klipova za korištenje u izradi videa, *podcastova*, video igara i slično.

Na kraju jesmo izradili potpuno funkcionalnu igru, iako ne bi bilo potpuno precizno reći da ju je izradio AI. Definitivno se može reći da je autor u ulozi korisnika odradio bar 85% ukupnog posla. Najveći doprinos AI alata izradi igre je bilo generiranje grafičkih elemenata i pozadinske glazbe. Ipak, treba naglasiti da ta sredstva koje smo dobili generiranjem od strane AI-ja, zajedno i sa programskim kodom u obliku C# skripti, bili daleko od toga da se direktno mogu koristiti u izradi igre. Sva sredstva je trebalo naknadno obraditi, ispravljati i preoblikovati u oblik u kojem ih se može koristiti u programskom alatu za izradu igara Unity.

Također treba naglasiti da je taj dio posla iziskivao poznavanje drugih programskih alata koji nisu direktno vezani za izradu računalnih igara. Za obradu slike koristio se Gimp, a za obradu zvučnih zapisa Audacity. Ta oba alata su besplatna ali trebalo ih je učiti koristiti u svrhu obrade izlaznih rezultata AI alata. Također valja napomenuti da je sva sredstva, a pogotovo C# skripte koje je Chat GPT generirao u svrhu korištenja u izradi igre, trebalo generirati mnogo puta kako bi se dobila verzija koja se kasnije koristila. U slučaju skripti vrlo često verzije koje su se koristile su bile spoj mnogo generiranih verzija skripti, plus autorov dodatak koji je ručno programirao.

Sve ovo zajedno znači da korištenje AI alata nikako ne znači da bilo tko može napraviti ovakav projekt. Pogotovo to vrijedi u kontekstu programiranja, jer je u tom slučaju najviše bila potrebna autorova intervencija kao korisnika. Autor je mišljenja da taj dio projekta ne bi bilo moguće napraviti bez pomoći korisnika, a ako i bi, to bi bilo slučajno i to bi bilo jako teško reproducirati drugi put. Razlog ovome je taj što je Chat GPT jezični model i nije dizajniran sa fokusom na programski kod. To također znači da će alat generirati različit kod ako mu postavljamo iste upite, zato je rečeno da je moguće dobiti koristan kod, ali slučajno i to je teško reproducirati.

Kao završnu misao o cijelom projektu možemo reći da je moguće izraditi video igru korištenjem dijelova generiranim od strane AI alata, ali nije baš elegantno. Autor bi definitivno igru napravio brže da je radio sam i koristio besplatne resurse koji se mogu naći preko interneta, ili u Unity-jevoj ugrađenoj trgovini.

Autor je također mišljenja da AI alati imaju svoju svrhu u izradi video igara, ali je ona sekundarna. Ta svrha su vjerojatnije: traženje informacija, savjeta, i inspiracije. Chat GPT je odličan za to. Alati koje smo koristili za generaciju grafike nisu baš elegantni za tu svrhu za koju su korišteni, ali su odlični za nekakvu konceptualnu umjetnost, gdje mi kao korisnik ili neki grafički umjetnik možemo doći do odličnih vizualnih inspiracija.

## Popis literature

- [1] Unity Technologies, »Our Company,« [Mrežno]. Available: <https://unity.com/our-company>. [Pokušaj pristupa 30 june 2023].
- [2] S. Lock, »What is AI chatbot phenomenon ChatGPT and could it replace humans?,« The Guardian, 5 December 2022. [Mrežno]. Available: <https://www.theguardian.com/technology/2022/dec/05/what-is-ai-chatbot-phenomenon-chatgpt-and-could-it-replace-humans>. [Pokušaj pristupa 30 June 2023].
- [3] C. GPT, »Odgovor na upit autora,« OpenAI, [Mrežno]. Available: <https://chat.openai.com/>. [Pokušaj pristupa 30 June 2023].
- [4] G. Marcus, E. Davis i S. Aaronson, »A very preliminary analysis of DALL-E 2,« 2022.
- [5] M. Diaz, »How to use Bing Image Creator (And why It's better than DALL-E 2),« 2023. [Mrežno]. Available: <https://www.zdnet.com/article/how-to-use-bing-image-creator/>. [Pokušaj pristupa 30 June 2023].
- [6] ThereIsAnAIForThat, »Soundraw,« [Mrežno]. Available: <https://theresanaiforthat.com/ai/soundraw/>. [Pokušaj pristupa 30 June 2023].
- [7] Wikipedia the Free Encyclopedia, »Sprite (computer graphics),« [Mrežno]. Available: [https://en.wikipedia.org/wiki/Sprite\\_\(computer\\_graphics\)](https://en.wikipedia.org/wiki/Sprite_(computer_graphics)). [Pokušaj pristupa 30 june 2023].
- [8] DALL-E, »Odgovor na upit autora,« OpenAI, [Mrežno]. Available: <https://labs.openai.com/>. [Pokušaj pristupa 30 june 2023].
- [9] Bing Image Create, »Odgovor na upit autora,« Microsoft, [Mrežno]. Available: <https://www.bing.com/create>. [Pokušaj pristupa 30 june 2023].

# Popis slika

Popis slika treba biti izrađen po uzoru na indeksirani sadržaj, te upućivati na broj stranice na kojoj se slika može pronaći.

Slika 1. „Spriteovi lika igrača“ [8] .....	11
Slika 2. „Pozadinske slike“ [9].....	12
Slika 3. „Platforme za pod“ [9] .....	13
Slika 4. „Tekstura poda“ [9] .....	13
Slika 5. „Automobil“ [9] .....	14
Slika 6. „kanta za smeće“ [9] .....	14
Slika 7. „Kutije“ [9] .....	15
Slika 8. „Hrpa smeća“ [9] .....	15
Slika 9. „Ograda“ [9] .....	16

## 7. Prilog: Izvorni kod

### 7.1. Skripta: GameManager

```
using TMPro;
using UnityEngine;
using UnityEngine.UI;

[DefaultExecutionOrder(-1)]
public class GameManager : MonoBehaviour
{
 public static GameManager Instance { get; private set; }

 public float initialGameSpeed = 5f;
 public float gameSpeedIncrease = 0.1f;
 public float gameSpeed { get; private set; }

 public TextMeshProUGUI scoreText;
 public TextMeshProUGUI hiscoreText;
 public TextMeshProUGUI gameOverText;
 public Button retryButton;

 private PlayerMovement player;
 private Spawner spawner;

 private float score;

 private void Awake()
 {
 if (Instance != null) {
 DestroyImmediate(gameObject);
 } else {
 Instance = this;
 }
 }

 private void OnDestroy()
 {
 if (Instance == this) {
 Instance = null;
 }
 }

 private void Start()
 {
 player = FindObjectOfType<PlayerMovement>();
 spawner = FindObjectOfType<Spawner>();

 NewGame();
 }

 public void NewGame()
 {
 Obstacle[] obstacles = FindObjectsOfType<Obstacle>();
 }
}
```

```

 foreach (var obstacle in obstacles) {
 Destroy(obstacle.gameObject);
 }

 score = 0f;
 gameSpeed = initialGameSpeed;
 enabled = true;

 player.gameObject.SetActive(true);
 spawner.gameObject.SetActive(true);
 gameOverText.gameObject.SetActive(false);
 retryButton.gameObject.SetActive(false);

 UpdateHiscore();
 }

 public void GameOver()
 {
 gameSpeed = 0f;
 enabled = false;

 player.gameObject.SetActive(false);
 spawner.gameObject.SetActive(false);
 gameOverText.gameObject.SetActive(true);
 retryButton.gameObject.SetActive(true);

 UpdateHiscore();
 }

 private void Update()
 {
 gameSpeed += gameSpeedIncrease * Time.deltaTime;
 score += gameSpeed * Time.deltaTime;
 scoreText.text = Mathf.FloorToInt(score).ToString("D5");
 }

 private void UpdateHiscore()
 {
 float hiscore = PlayerPrefs.GetFloat("hiscore", 0);

 if (score > hiscore)
 {
 hiscore = score;
 PlayerPrefs.SetFloat("hiscore", hiscore);
 }

 hiscoreText.text = Mathf.FloorToInt(hiscore).ToString("D5");
 }
}

```

## 7.2. Skripta: BackgroundMovement

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BackgroundMovement : MonoBehaviour
{
 private float currentSpeed;
 public float speedModifier = 1f;

 private void Update()
 {
 // Accelerate the background and floor movement
 currentSpeed = (GameManager.Instance.gameSpeed) *
speedModifier;
 // Move the background and floor based on the current speed
 transform.Translate(Vector3.left * currentSpeed *
Time.deltaTime);
 }
}
```



## 7.3. Skripta: PlayerMovement

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
 public float jumpForce = 5f;
 public float forwardSpeed = 5f;
 private Rigidbody2D rb;
 private bool isJumping = false;

 private void Start()
 {
 rb = GetComponent<Rigidbody2D>();
 }

 private void Update()
 {
 // Check for jump input
 if (Input.GetKeyDown(KeyCode.Space) && !isJumping)
 {
 isJumping = true;
 Jump();
 }
 }

 private void FixedUpdate()
 {
 // Move the player forward based on the background movement
 float movement = forwardSpeed * Time.fixedDeltaTime;
 rb.velocity = new Vector2(movement, rb.velocity.y);
 }

 private void Jump()
 {
 // Apply upward force to make the player jump
 rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
 }

 private void OnCollisionEnter2D(Collision2D collision)
 {
 // Check for collision with the ground or platforms to allow
 the player to jump again
 if (collision.gameObject.CompareTag("Floor"))
 {
 isJumping = false;
 }
 }

 private void OnTriggerEnter2D(Collider2D other) {
 if (other.CompareTag("Obstacle")) {
 GameManager.Instance.GameOver();
 }
 }
}
```

## 7.4. Skripta: RunningAnimation

```
using UnityEngine;

public class RunningAnimation : MonoBehaviour
{
 public Sprite[] runningFrames; // Array of running animation frames
 public float frameRate = 0.1f; // Delay between frame changes

 private SpriteRenderer spriteRenderer;
 private int currentFrameIndex = 0;
 private float timer = 0f;

 private void Start()
 {
 spriteRenderer = GetComponent<SpriteRenderer>();
 }

 private void Update()
 {
 timer += Time.deltaTime;

 if (timer >= frameRate)
 {
 timer -= frameRate;
 currentFrameIndex = (currentFrameIndex + 1) %
runningFrames.Length;
 spriteRenderer.sprite = runningFrames[currentFrameIndex];
 }
 }
}
```

## 7.5. Skripta: Obstacle

```
using UnityEngine;

public class Obstacle : MonoBehaviour
{
 float speed; // Speed at which the obstacle moves
 float despawnXPosition = -20f; // X position at which the obstacle
despawns

 private void Update()
 {
 speed = GameManager.Instance.gameSpeed;
 // Move the obstacle to the left
 transform.position += Vector3.left * speed * Time.deltaTime;

 // Check if the obstacle has reached the despawn X position
 if (transform.position.x <= despawnXPosition)
 {
 // Despawn the obstacle
 Destroy(gameObject);
 }
 }
}
```

## 7.6. Skripta: Spawner

```
using UnityEngine;

public class Spawner3 : MonoBehaviour
{
 [System.Serializable]
 public struct SpawnableObject
 {
 public GameObject prefab;
 public float spawnChance;
 }

 public SpawnableObject[] objects;

 public float numPrefabs;
 public float minSpawnRate = 1f;
 public float maxSpawnRate = 2f;

 private void OnEnable() {
 Invoke(nameof(Spawn), Random.Range(minSpawnRate,
maxSpawnRate));
 }

 private void OnDisable() {
 CancelInvoke();
 }

 private void Start() {
 numPrefabs = objects.Length;
 }

 private void Spawn() {
 float spawnChance = Random.Range(1, numPrefabs);

 foreach (var obj in objects) {
 if (spawnChance >= obj.spawnChance) {
 GameObject obstacle = Instantiate(obj.prefab);
 obstacle.transform.position += transform.position;
 break;
 }
 spawnChance = (objects.Length - spawnChance);
 }
 Invoke(nameof(Spawn), Random.Range(minSpawnRate,
maxSpawnRate));
 }

 /* private void Spawn() {
 float spawnChance = Random.value;

 foreach (var obj in objects) {
 if (spawnChance < obj.spawnChance) {

 GameObject obstacle = Instantiate(obj.prefab);
 }
 }
 }
}
```

```

 obstacle.transform.position += transform.position;
 break;
 }

 spawnChance -= obj.spawnChance;
}

Invoke (nameof (Spawn) , Random.Range (minSpawnRate,
maxSpawnRate));

} */
}

```

## 7.7. Skripta: Wrapper

```

using UnityEngine;

public class Wrapper : MonoBehaviour
{
 public float checkX = -26f; // The target x value to check against
 public float newX = 17f; // The new x value to move the prefab to

 private void Update()
 {
 // Check if the prefab has reached the target x value
 if (transform.position.x <= checkX)
 {
 // Move the prefab to the new x value
 Vector2 newPosition = new Vector2 (newX,
transform.position.y);
 transform.position = newPosition;
 }
 }
}

```