SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Mihael Kožul

PRIMJENA POTICANOG UČENJA NA PRIMJERU JEDNOSTAVNE VIDEOIGRE

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mihael Kožul

Matični broj: 0016141878

Studij: Informacijski sustavi

PRIMJENA POTICANOG UČENJA NA PRIMJERU JEDNOSTAVNE VIDEOIGRE

ZAVRŠNI RAD

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2022.

Zahvala

Veliku zahvalu, u prvom redu, dugujem svom mentoru dr.sc. Bogdanu Okreši Đurić bez kojega ovaj završni rad ne bi bio moguć. Hvala na pomoći pri smišljanju teme rada, hvala na savjetima, ukazanoj potpori i strpljenu i najviše od svega brzom ispravku radnih verzija rada.

Također, zahvaljujem se i svim profesorima, asistentima te FOI-ju općenito na sveukupnom prenesenom znanju, a ponajviše na, s jedne strane vrlo ugodnom i prijateljskom, a s druge strane na vrlo profesionalnom i korektnom međusobnom odnosu.

Prvenstveno hvala mojoj djevojci i svim mojim prijateljima, a onda hvala i kolegama, poznanicima i svima onima koji su bili dio mojih studentskih dana u Varaždinu i Zagrebu (hvala SARS-CoV-2) i zbog kojih su ove tri godine prošle toliko brzo i s toliko zabave da ću ih pamtiti cijeli svoj život.

Kao šećer na kraju, hvala svim članovima moje obitelji. Najveću zaslugu za ono što sam postigao i što ću tek postići pripisujem njima. Hvala im na neizmjernoj i nesebičnoj potpori u apsolutno svakom trenutku mojega života. Posebno hvala mojim roditeljima što su mi dali sve, a moje je bilo samo da učim. Hvala i na uzrečicama "Ne moram ja ništa, moram samo umrijeti" i "Ako nisu plan A i plan B, bit će plan C i D" kojima sam se vodio, kojima se vodim i bez kojih bi moje školovanje i život bili znatno kompliciraniji.

Velika hvala svima, Mihael

Mihael Kožul

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Završni rad bavi se širokim pojmom umjetne inteligencije te kao takav obuhvaća strojno učenje, neuronske mreže, znanost o podacima i sl. No, primarni je fokus rada na poticanom učenju (engl. *reinforcement learning*) i njezinom prikazu na primjeru jednostavne, samostalno napravljene, 2D videoigre. Igra je napravljena po uzoru na poznatu igru "Geometry Dash" u Unityju te je kao takva pisana jezikom C#. Za razvoj inteligentnog agenta jednom od metoda poticanog učenja, korišten je besplatni softver otvorenog koda Unity Machine Learning Agents Toolkit. Kako bi se otežalo treniranje agenta i prikazala realnost razvoja umjetne inteligencije, igra se sastoji od jedne razina koja, koristeći proceduralno generiranje, stvara novi, drukčiji sadržaj mape. Generalni je cilj rada obraditi jednu vrlu zanimljivu temu informatike, koja iz dana u dan svoju primjenu pronalazi u novim aspektima svakodnevnog života.

Ključne riječi: umjetna inteligencija; strojno učenje; poticano učenje; agent; 2D videoigra; proceduralno generiranje; algoritmi

Sadržaj

1. Uvod

Područje informacijskih tehnologija raste iz dana u dan. U svakodnevnom životu sve više se mogu čuti pojmovi poput umjetne inteligencije (engl. *artificial intelligence*), strojnog učenja (engl. *machine learning*), dubokog učenja (engl. *deep learning*) i sl. Od znanosti i obrazovanja, preko sporta i trgovine pa sve do zdravlja, svakodnevnog poslovanja i industrije igara, umjetna inteligencija se predstavlja kao rješenje za sve probleme. Iz tog se razloga može reći kako doživljava novo zlatno doba. Vrijeme, novac i znanje kao nikad do sad ulažu se kako bi se provela njezina implementacija i tako poboljšali i unaprijedili razni procesi svakodnevnog života. Danas, zahvaljujući velikom razvoju, umjetna inteligencija ima mogućnost upravljanja vozilom, otkrivanja bolesti, obrađivanja velike količine podataka i sl., no zbog svoje kompleksnosti često se preuveličavaju njezine trenutačne mogućnosti. Kako stvari stoje, bit će potrebno još vremena prije nego umjetna inteligencija dostigne svoj puni potencijal.

Strojno učenje, kao poddomena umjetne inteligencije, još je jedan izraz koji je vrlo popularan u današnje doba. Koristeći velike količine podataka kao ulaz (engl. *input*), ono pokušava predvidjeti vrijednosti izlaza (engl. *output*) na način da softverska aplikacija nije eksplicitno programirana da to napravi. Kao što će biti u nastavku rada objašnjeno, tri su grane strojnog učenja - nadzirano (engl. *supervised*) učenje, nenadzirano (engl. *unsupervised*) učenje i poticano (engl. *reinforcement*) učenje. Razlika između nadziranog i nenadziranog učenja je u skupu podataka. S jedne strane je skup podataka označen, pri čemu se softver uči precizno definirati svaki podatak, dok je s druge strane skup podataka neoznačen te softver pokušava otkriti različite uzorke bez potrebe za ljudskom intervencijom. Kao posljednja grana, poticano se učenje bavi kreiranjem inteligentnih agenata koji svojim postupcima pokušavaju maksimizirati kumulativ svih dobivenih nagrada. Detaljniji opis uz primjere i dodatna objašnjenja dio je poglavlja "2. Pojam umjetne inteligencije".

Završni rad pokušava objasniti bitne segmente cjelokupne domene umjetne inteligencije - njezinu povijest, primjene i algoritme, pri čemu će se veliki naglasak stavljati na poticano učenje (engl. *reinforcement learning*). Ideja je rada obraditi temu kroz tri poglavlja. Prvo poglavlje funkcionira kao relativno kratak uvod u temu, kako bi se stvorila šira slika umjetne inteligencije i kako bi se lakše razumjeli koncepti strojnog učenja, odnosno poticanog učenja koje je u centru pažnje u naredna dva poglavlja. Drugo poglavlje vrlo detaljno na teorijski način obrađuje poticano učenje kako bi se u trećem poglavlju, s punim shvaćanjem, prezentirao praktični dio rada - primjena poticanog učenja na primjeru agenta unutar jednostavne, samostalno kreirane videoigre. Iako nije primarna tema rada, na vrlo kratak način obrađuje se i proces kreiranja videoigre. Osim toga, objašnjeno je i proceduralno generiranje koje je korišteno kao jedna funkcionalnost kreirane igre. Unutar igre kreira se beskonačno duga mapa koju igrač, u ovom slučaju agent, prolazi do gubitka.

2. Pojam umjetne inteligencije

Kroz povijest, ljudi su pokušali opisati, odnosno shvatiti što je to inteligencija. Danas postoji i relativno dobar način njezinog kvantificiranja, no razvojem tehnologije javlja se novi oblik inteligencije - inteligencija strojeva, odnosno neživih objekata, i s time potpuno novo shvaćanje inteligencije.

Može se reći kako je ljude oduvijek zanimala ideja inteligencije koju posjeduje neživa stvar. To je kroz povijest vidljivo razvojem automata, uređaja koji imaju sposobnost samostalnog obavljanja rada. Među prvim automatima smatra se drveni golub na parni pogon iz 4. stoljeća prije Krista kojeg je izradio Arhita iz Tarenta [1]. O grčkoj fasciniranošću automatima govore brojni mitovi i legende. Najzanimljiviji primjeri su naravno vezani uz grčkog boga kovača, Hefesta. U Homerovoj Ilijadi i Odiseji spominju se različite vrste automata, neki od njih u 18. pjevanju Ilijade su mu služili kao ispomoć što u svakodnevnom životu, što u procesu proizvodnje, u ovom slučaju pri kovanju Ahilejevog novog oružja [2, 18. pjevanje: 372-377, 468–473]. Osim toga, postoji i mit kako je Hefest kreirao veliki brončani automat po nazivu Talos kojemu je zadaća bila štiti otok Kretu od neželjenih osvajača. No, fasciniranost automatima nije bila prisutna samo u staroj Grčkoj već u gotovo svim civilizacijama, kulturama i njihovim razdobljima, od Kine preko Afrike pa sve do renesansne Europe [1]. Zahvaljujući velikim i brojnim tehnološkim dostignućima danas se može reći kako je inteligencija strojeva zapravo i vjerojatnija nego ikad do sad.

Razvoj umjetne inteligencije započeo je u 20. stoljeću te je od tada njezin razvoj prolazio kroz mnoge uspone i padove. Njezinim začetakom smatra se 1943. godina, kada su Warren McCulloch i Walter Pitts predložili model umjetnih neurona kojima pripadaju dva stanja - uključen (engl. *on*) i isključen (engl. *off*) [3, str. 35]. Time su pokazali da se mrežom povezanih neurona može izračunati bilo kakva izračunljiva funkcija. Pokazali su i da se sve logičke veze mogu implementirati koristeći jednostavne strukture umreženih neurona, a na temelju toga sugerirali da neuronske mreže mogu i učiti. Sedam godina kasnije, Harvardski studenti Marvin Minsky i Dean Edmonds kreirali su prvu neuronsku mrežu koja je simulirala 40 neurona [3, str. 35]. Iste godine, Alan Turing, poznati engleski matematičar, informatičar i otac moderne računarske znanosti, u svome članku "Računalni strojevi i inteligencija" (engl. *Computing Machinery and Intelligence*) [4] predstavio je Turingov test, strojno učenje, genetske algoritme i poticano učenje. Turingov test danas je najpoznatiji test provjere inteligencije računala. Test je zamišljen kao igra koja funkcionira na način da jedan izolirani ispitivač vodi razgovor s dva (ili više) igrača pri čemu je jedan (ili više) od njih računalo. Posao je ispitivača otkriti koji je igrač računalo, odnosno koji je ljudsko biće. Nedostatak je testa taj što ne uzima u obzir druge oblike inteligencije. Neki od najinteligentnijih računala danas nemaju nikakve šanse proći Turingov test, no to ne znači da je njihov razvoj manje impresivan. U svome članku je, također, sugerirao i kako je jednostavnije koristeći algoritme učenja stvoriti umjetnu inteligenciju na razini ljudske inteligencije nego ju pokušati ručno isprogramirati [4].

Izraz "umjetna inteligencija", a time i područje unutar računarske znanosti, nastao je kao rezultat organiziranog skupa desetorice znanstvenika u Hanoveru, New Hampshire u sklopu

Dartmouth Collegea [5]. Skup je trajao 2 mjeseca, a njegov cilj bio je ostvariti veliki napredak unutar novokreiranog polja znanosti na način da se istražuju mogućnosti strojeva da koriste jezik i rješavaju razne probleme koji su do tad bili rezervirani isključivo za čovjeka. Sudionici skupa bili su organizatori John McCarthy (Dartmouth College), Marvin Minsky (Harvard), Claude Shannon (Bell Labs) i Nathaniel Rochester (IBM) te uz njih Allen Newell (Carnegie Tech), Herbert Simon (Carnegie Tech), Trenchard More (Princeton), Arthur Samuel (IBM), Ray Solomonoff (MIT) i Oliver Selfridge (MIT). Sam izraz kreirao je John McCarthy zbog čega se često naziva i ocem umjetne inteligencije. Nakon skupa započinje prvo zlatno doba njezinog razvoja koje traje do 1974. godine [6]. Značajan napredak u razvoju računalne moći i interneta pozitivno su djelovali na razvoj umjetne inteligencije upravo zbog omogućavanja stvaranja velike količine podataka. Taj fenomen danas se često naziva "veliki podaci" (engl. *big data*).

Videoigre i umjetna inteligencija, kao dvije teme koje ovaj završni rad obrađuje, dijele dugu i bogatu povijest. Veliki dio razvoja unutar ovog polja zapravo je nastao kreirajući agente koji igraju igre. Time se htjelo vidjeti mogu li računala riješiti zadatke za koje se smatralo da je potrebna inteligencija. U tome im je naravno i uspjelo. Prvog takvog agenta isprogramirao je A.S.Douglas 1952. godine na videoigri križić-kružić (engl. *Tic-Tac-Toe*) [7, str. 8]. Samo godinu dana kasnije Alan Turing je koristeći Minimax algoritam kreirao agenta koji igra šah [7, str. 8]. Poticano učenje, kao grana strojnog učenja, nastalo je 1959. godine te se njezinim ocem smatra Arthur Samuel. On je u svome radu kreirao softver koji je igrajući sam protiv sebe naučio igrati dame. Uspjeh umjetne inteligencije u tradicionalnim igrama vidljiv je i danas. Google je od 2014. godine razvijao AlphaGo softver [8] koji je 2017. godine uspio pobijediti broj 1 Go igrača u svijetu [9]. Usporedbe radi, složenost igre s obzirom na broj legalnih pozicija koje se mogu ostvariti s početne pozicije u igri u šahu je 10^{50}, dok je u igri Go taj broj 10^{172} [10].

2.1. Definiranje pojmova

Često se što zbog jednostavnosti, što zbog neznanja različiti pojmovi unutar polja umjetne inteligencije smatraju sinonimima, odnosno istoznačnicama, iako one to nisu. Tako se na primjer umjetna inteligencija, strojno učenje, duboko učenje i neuronske mreže smatraju istim konceptima, a veliki podaci i znanost o podacima disciplinom unutar umjetne inteligencije i sl. U nastavku su prikazane definicije pojmova.

Umjetna inteligencija: „Sposobnost digitalnog računala ili računalno kontroliranog robota da obavlja zadatke koji se obično povezuju s inteligentnim bićima.” [11]

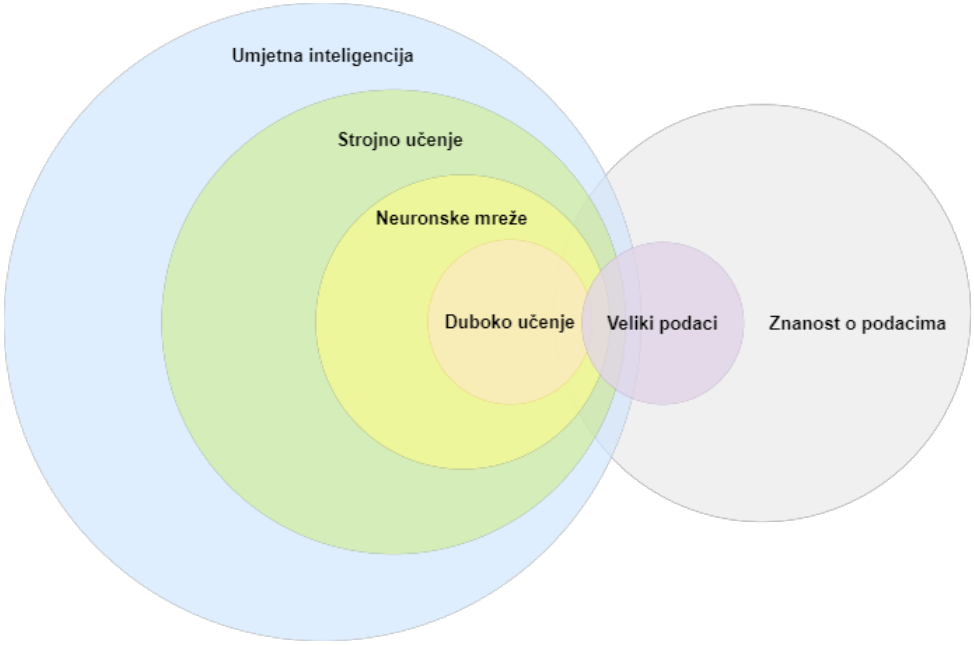
Strojno učenje: „Disciplina unutar umjetne inteligencije koja se bavi implementacijom računalnog softvera koji može učiti autonomno.” [12]

Duboko učenje: „Vrsta umjetne inteligencije koja koristi algoritme (= skupove matematičkih uputa ili pravila) na temelju načina na koji funkcionira ljudski mozak. ” [13]

Neuronske mreže: „Računalni program koji radi na način inspiriran prirodnom neuronskom mrežom u mozgu.” [14]

Znanost o podacima: „Multidisciplinarno područje koje korištenjem znanstvenih metoda izvlači korisne informacije iz računalnih podataka, osobito velikih količina podataka.” [15]

Veliki podaci: „Skupovi informacija koji su preveliki ili presloženi za rukovanje, analizu ili korištenje standardnim metodama.” [16]



Slika 1: Odnos pojmova vezanih za polje umjetne inteligencije

Kao što je vidljivo na slici 1, umjetna inteligencija krovni je pojam za strojno učenje, neuronske mreže i duboko učenje. Znanost o podacima znanstvena je disciplina koja integrira sve izraze unutar umjetne inteligencije kako bi se mogle izvući informacije iz velikih količina podataka i stvoriti predviđanja temeljena na njima. Veliki podaci sami po sebi nemaju šireg značenja osim označavanja velikog skupa podataka.

Duboko učenje i neuronske mreže gotovo su iste prirode, zbog čega je razlika samo u njihovoj kompleksnosti. Duboko je učenje u stvarnosti samo po sebi neuronska mreža. Drugi naziv za duboko učenje je duboka neuronska mreža. Ona je vrlo kompleksna, a njezina kompleksnost ovisi o broju slojeva između početnog (ulaznog) sloja koji predstavlja podatke i završnog (izlaznog) sloja koji predstavlja rezultat. Trenutno ne postoji eksplicitno prihvaćeni kriterij određivanja koja neuronska mreža je duboka. Neuronska mreža od jednog ili nekoliko međuslojeva (skrivenih slojeva) je i dalje neuronska mreža, no ne može se nazivati dubokom neuronskom mrežom, odnosno dubokim učenjem. Iako službeno ne postoji prihvaćeni kriterij, model neuronske mreže mora se sastojati od minimalno dva skrivena sloja kako bi se smatrao dubokim učenjem [17, str. 660].

Neuronske mreže, a shodno tome i duboko učenje, podskup su strojnog učenja te se temelje na umjetno kreiranim neuronima koji kreiraju mrežu inspiriranu ljudskim, organskim neuronskim mrežama. Tako stvoreni sustav uči i prilagođava se na temelju velike količine podataka te prepoznaje skrivene uzorke, kombinira ih i stvara izlazni rezultat.

Budući da je strojno učenje nešto mlađi pojam od umjetne inteligencije, početni načini kreiranja te inteligencije sastojali su se od jasno isprogramiranog slijeda koraka koje računalo prati. To u biti znači da računalo nije samo naučilo kako riješiti zadatak već da mu je unaprijed definirano koje podatke koristiti te kako ih analizirati. Jasno je kako je to vrlo nizak oblik inteligencije. Kako bi se razvili složeniji oblici inteligencija razvijeno je strojno učenje. Strojno učenje podskup je umjetne inteligencije koji se bavi izvlačenjem znanja iz podataka [18]. Njegov je posao izvesti određeni zadatak bez potrebe za eksplicitno isprogramiranim instrukcijama kako taj zadatak riješiti. Poanta je da softver sam shvati kako pristupiti i ispuniti željene ciljeve zadatka. Umjetna inteligencija, kao najširi koncept, jednostavno rečeno vidljivi je prikaz inteligencije strojeva [3, str. 19]. Dakle, to je praktični primjer računala koji rješava zadatke za koje se smatra kako je potrebna inteligencija poput pričanja i razumijevanja jezika (Siri, Bixby, Alexa), vožnje auta (Tesla), hodanja (Boston Dynamics), jednostavnije rečeno sposobnost vida, percepcije, govora i dr.

2.2. Strojno učenje

Strojno učenje poddomena je umjetne inteligencije koja se bavi implemetacijom softvera koji ima sposobnost samostalnog učenja [18]. To je vrlo bitno u razvoju složenijih programa iz razloga što je za njihov razvoj potrebno skriveno znanje. Skriveno, prešutno ili tacitno znanje je znanje koje se teško prenosi na drugu osobu, u ovom slučaju računalo [19]. Definirati pravila jednostavne igre, povezati glavne gradove s državom nije teško prenijeti, no kako voziti, komunicirati i izraziti emocije je gotovo nemoguće. Kako bi se bolje objasnilo, može se uzeti jezik kao primjer važnosti samostalnog učenja softvera, točnije razumijevanje jezika u svrhu, npr. prijevoda s hrvatskog na engleski. Prva ideja koja pada na pamet za razvoj softvera sigurno obuhvaća softver baziran na pravilima jezika. To nažalost ne funkcionira, tj. prirodni jezik nije tako jednostavan. On se sastoji od mnogo iznimaka i mnogo njegovog značenja proizlazi iz konteksta. Kako bi program funkcionirao, potrebno bi bilo svaku tu iznimku fizički zapisati unutar programa, što je moguće jedino u teoriji. Takav program jednostavno je prekompleksan. Kako bi se pronašlo rješenje za taj problem, nastalo je strojno učenje kao disciplina unutar umjetne inteligencije. Sada, uz sposobnost računala da uči na samostalan način, bez potrebe za izravnim kodiranjem "pravila", računalo će uz velike količine podataka samostalno zaključiti koja su "pravila". Zbog toga se može reći kako je strojno učenje vrlo slično učenju čovjeka u njegovom razvoju. Tako kroz mnogobrojne faze pokušaja i pogrešaka čovjek i/ili računalo uči kako hodati, sporazumijevati se, rješavati određene zadatke, ali i učiti.

Tri su glavna pristupa strojnom učenju [3, str. 671]: nadzirano (engl. *supervised*), nenadzirano (engl. *unsupervised*) i poticano (engl. *reinforcement*) učenje. Podjela je nastala s obzirom na tip podataka i prema načinu rada s podacima. Svaki tip objašnjen je u nastavku rada u sekcijama "2.2.1, Nadzirano učenje" i "2.2.2. Nenadzirano učenje" te poglavlju "3. Poticano učenje".

Općenito govoreći, svrha strojnog učenja je pronaći nepoznatu funkciju f na temelju analize skupa podataka D [20]. U nadziranom učenju skup podataka sadrži, osim ulaznih x, i

izlazne vrijednosti y. Takav bi se odnos mogao zapisati u obliku:

$$D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N).$$

Budući da su izlazne vrijednosti poznate, zadatak algoritma je pronaći najbolju približnu funkciju g na način da minimizira odstupanja između poznatih $g(x_N)$ i y_N te da daje dobru približnu vrijednost y za podatkovne točke x koje nisu dio skupa za testiranje [20].

U nenadziranom učenju poznate su samo ulazne vrijednosti x, stoga se takav odnos zapisuje u obliku:

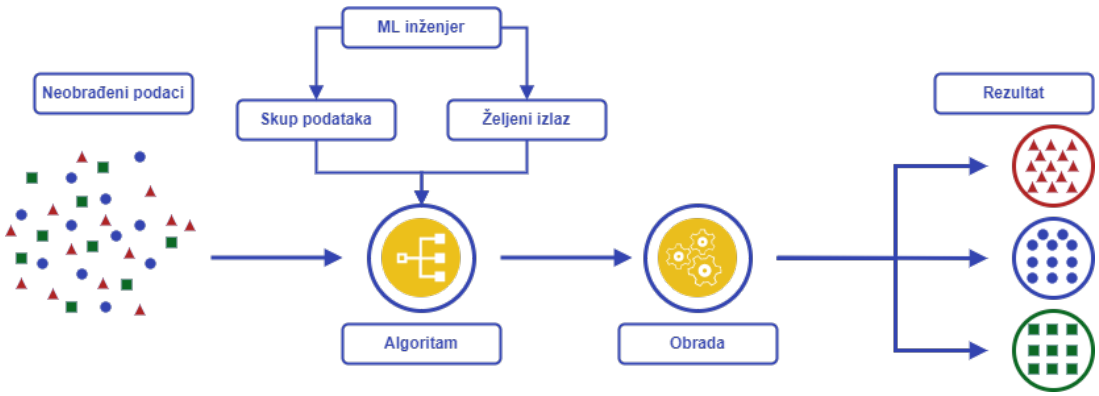
$$D = x_1, x_2, ..., x_N.$$

U ovom slučaju teže je naći približnu funkciju za f, jer nisu dostupni izlazni podaci, zbog čega algoritam ne može znati kako se funkcija ponaša. Zato se ovakav tip učenja koristi za određivanje takvog g koji pronalazi uzorke ili trendove u podacima. [20]

Jedan od većih problema strojnog učenja nastaje kod učenja modela te se zove prenaučenost (engl. *overfitting*). Problem prenaučenosti je problem kod kojeg model ne odgovara generalnim uzorcima već samo danim primjerima prema kojima se vršilo učenje. Zbog toga se model neće moći prilagoditi novom skupu podataka te njegova zapažanja i predviđanja neće biti pouzdana. Jednostavna i efikasna tehnika sprječavanja prenaučenosti je metoda isključivanja (engl. *dropout*). Koristeći tehniku isključivanja nasumično se izlazne vrijednosti neurona postavljaju na nulu te tako zanemaruju. Time se postiže nova, drukčija konfiguracija neuronskih mreža koja za posljedicu ima robusnije znanje.

2.2.1. Nadzirano učenje

Nadzirano učenje vrsta je strojnog učenja koje se temelji na označenom skupu podataka (engl. *labeled data*) i skupu za testiranje (engl. *testing set*) [21]. Uspoređujući predikcije nadziranog modela s ispravnim rezultatima, model stvara veze između svojstava ulaznih podataka i kategorija kojima pripadaju te na taj način uči. U današnje vrijeme modeli nastali nadziranim učenjem koriste se u sklopu bioinformatike, u svrhu predviđanja cijena dionica i/ili (kripto)valuta, prepoznavanja lica i prepoznavanja glasa osobe, u medicini, infektologiji i dr, što je vidljivo i u tablicama 1 i 2.



Slika 2: Nadzirano učenje (engl. *supervised learning*); prema [21]

2.2.1.1. Klasifikacija

Klasifikacija (engl. *classification*) je tip modela nadziranog učenja koji iz skupa podataka kreira zaključke o tome koji podatak pripada kojoj kategoriji [7, str. 58]. Jednostavno rečeno, klasifikacija se bavi grupiranjem podataka. Primjer takve grupacije može biti skup fotografija različitih životinjskih vrsta koje će predstavljati ulazne vrijednosti. Uzimajući takav skup podataka, svaka je fotografija označena kao pas, mačka, kornjača i sl. Posao je algoritma klasificirati nove slike u bilo koju od ovih označenih kategorija.

U nastavku u tabličnom obliku prikazane su četiri vrlo popularne metode korištene u izradi modela klasifikacije

Tablica 1: Odabrane metode klasifikacije; prema [22] [23] [24]

Metoda	Definicija	Prednost	Mane	Primjena
Naivni Bayes	Algoritam koji koristi Bayesov teorem vjerojatnosti za klasifikaciju objekata.	Bazira se na jednostavnim načelima vjerojatnosti.	Ponavljanje proračuna za svaki unos novog zapisa.	Uglavnom se koristi u klasifikaciji teksta.
K-najbližih susjeda	Koristi blizinu za izradu klasifikacija ili predviđanja o grupiranju pojedinačnog podatka.	Lakoća implementacija (nije potrebna obuka algoritma). Novi podaci mogu biti dodani bez većih poteškoća.	Osjetljiv na vrijednosti koje nedostaju i ekstremne vrijednosti.	Sustavi preporuke, prepoznavanje lica, medicina...
Stabla odluke	Stablasto strukturirani klasifikator, unutarnji čvorovi predstavljaju značajke skupa podataka, grane predstavljaju pravila odlučivanja, svaki list predstavlja ishod.	Lakoća implementacije. Jednostavno korištenje kod klasificiranja novih zapisa.	Kompleksnost. Problemi prenaučenosti.	Koristi se za rad s nelinearnim skupovima podataka.
Slučajne šume	Pouzdan skup višestrukih stabala odlučivanja.	Poboljšava točnost stabala odluke (smanjuje prenaučenost). Dobro funkcionira s kategoričkim i kontinuiranim vrijednostima.	Računalno zahtjevno. Dugotrajna obuka.	Financije, maloprodaja, aeronautika...

Definitivno jedna od najpoznatijih metoda klasifikacije je metoda naivni Bayes. Ova vrlo jednostavna metoda najčešće se koristi za filtriranje neželjene e-pošte (engl. *spam email*). Kao što piše u tablici 1, ova metoda koristi Bayesov teorem koji se temelji na uvjetnoj vjerojatnosti te izgleda ovako [17, str. 605]:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)},$$

pri čemu:

- A i B predstavljaju događaje,
- $P(A|B)$ predstavlja vjerojatnost za A ako je B istinit
- $P(B|A)$ predstavlja vjerojatnost za B ako je A istinit
- $P(A)$ i $P(B)$ vjerojatnosti za događaj A i događaj B.

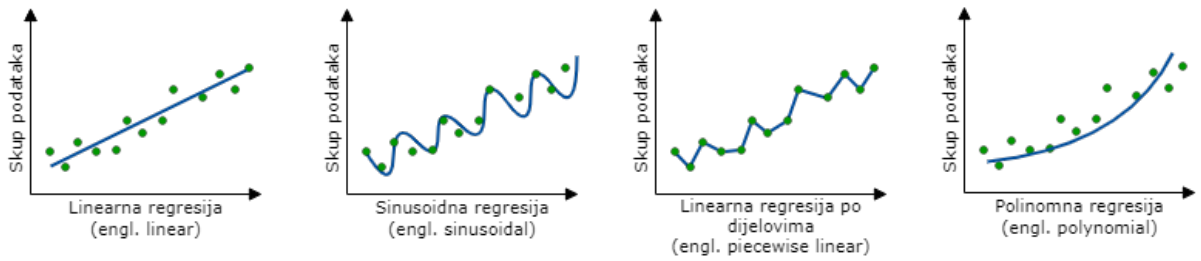
Algoritam klasificira poruke na željenu i neželjenu poštu na temelju pojavnosti određenih ključnih riječi prema kojima se zatim određuje koji je tip poruke. Naivni Bayes uključuje pridjev "naivni" iz razloga što metoda smatra svaku ulaznu varijablu neovisnom o drugima što je rijetko istina.

2.2.1.2. Regresija

Regresija (engl. *regression*) je tip modela nadziranog učenja koji se svodi na kreiranje aproksimacije funkcije [25]. Od trenutka formiranja modela regresije otvara se mogućnost predviđanja rezultata na temelju novog ulaznog podatka. Na taj način se predviđaju kontinuirane varijable, kao što su promjene temperature, fluktuacija električne energije i dr.

Razlike u dobivenim rezultatima ovisno o metodi vidljive su na slici 3 u grafičkom obliku. Prikazane su:

- linearna regresija (engl. *linear regression*),
- sinusoidna regresija (engl. *sinusoidal regression*),
- linearna regresija po djelovima (engl. *piecewise regression*) i
- polinomijalna regresija (engl. *polynomial regression*).



Slika 3: Tipovi regresije (engl. *regression*); prema [3]

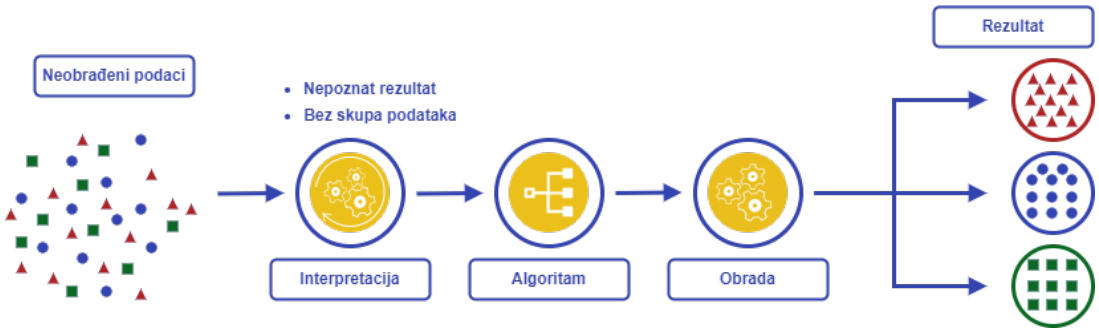
Treba naglasiti kako se velika većina metoda, među kojima i metode prikazane u tablici 1 i tablici 2, koristi u problemima klasifikacije i regresije uz manje ili veće promjene.

Tablica 2: Odabrane metode regresije; prema [26] [27] [28]

Metoda	Definicija	Prednost	Mana	Primjena
Linearna	Pronalazak linearne funkcije kojom se opisuje odnos dvaju svojstava.	Lakoća implementacije. Dobra interpretacija.	Osjetljiv na ekstremne vrijednosti. Kod manje količine podataka dovodi do prenaučenosti.	Predviđanje prodaje, analiza anketnih podataka, predviđanje cijene dionica...
Polinomna	Pronalazak polinomne funkcije n-tog reda kojom se opisuje odnos dvaju svojstava.	Funkcionira na svakom skupu podataka. Funkcionira odlično za nelinearne probleme.	Potrebno izabrati pravi stupanj polinoma za točne rezultate.	Predviđanje stope širenja COVID-19 i drugih zaraznih bolesti...
LASSO	Skuplja vrijednosti podataka prema središnjoj točki kao srednjoj vrijednosti.	Izbjegava prenaučenost.	Pristrani koeficijenti. Nestabilne procjene. Teško procjenjuje standardne pogreške.	Ekonomija, financije...
SVR	Predviđanje diskretnih vrijednosti na linearnim i nelinearnim problemima.	Lako prilagodljiv. Nije pristran prema ekstremnim vrijednostima.	Teško razumljiv. Obvezna primjena skaliranja.	Bioinformatika, predviđanje cijene dionica, obrada slike...

2.2.2. Nenadzirano učenje

Nenadzirano učenje vrsta je strojnog učenja kod kojeg, za razliku od nadziranog učenja, ne postoji skup označenih podataka i testni podaci [21]. Algoritam mora na temelju neoznačenih podataka kao ulaznih varijabli doći do njihovog razumijevanja. Jedna manja prednost nenadziranog učenja je ta što su neoznačeni podaci često dostupniji u izobilju, u odnosu na označene. Takvi algoritmi proizvode generativne modele koji zatim mogu proizvesti realističan tekst, slike, audio i video i dr.



Slika 4: Nenadzirano učenje (engl. *unsupervised learning*); prema [21]

Primjene modela nastalih nenadziranim učenjem su brojne. Zbog mogućnosti grupira-

nja podataka po parametru, modeli se često koriste kako bi se grupirali kupci prema uzorcima kupovine. Osim toga, koristi se i u znanosti, npr. u svrhu analize evolucijske biologije. Zanimljiv primjer korištenja istreniranog modela je i u svrhe detektiranja anomalija unutar podataka, što je vrlo korisno kod otkrivanja prijevara.

Nenadzirano učenje uglavnom obavlja jedan od naredna tri zadatka [29]:

- smanjenje dimenzija (engl. *dimensionality reduction*) - transformacija podataka u nižedimenzionalni prostor uz zadržavanje značajnih svojstava izvornih podataka
- otkrivanje anomalija (engl. *anomaly detection*) - identificiranje ekstremiteta i anomalija unutar skupa podataka
- grupiranje (engl. *clustering*) - identificiranje grupe sličnih podataka.

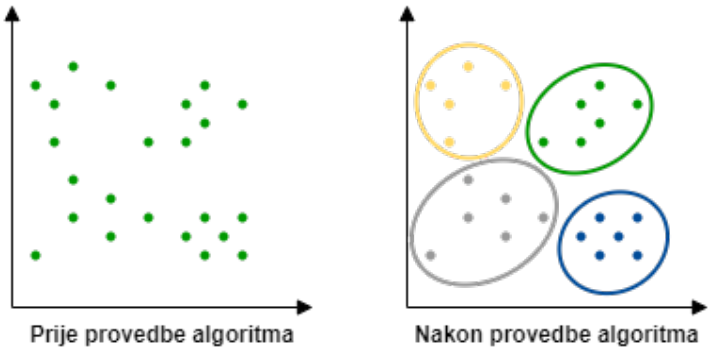
2.2.2.1. Algoritam K-srednjih vrijednosti

Algoritam k-srednjih vrijednosti (engl. *k-means clustering*) smatra se najpopularnijim algoritmom grupiranja. Između ostaloga, razlog popularnosti ovoga algoritma je brzina treninga te vrlo dobra ravnoteža između jednostavnosti i učinkovitosti. Funkcionira na način da algoritam dijeli podatke u k grupa podataka na temelju sličnosti između uzoraka podataka. Jednostavni pseudokod algoritma glasi [7, str. 78]:

Za dani k

1. Podijeliti podatke nasumično u k nepraznih grupa (engl. *cluster*)
2. Izračunati položaj težišta (engl. *centroid*) grupa trenutne particije. Težište je središte, odnosno središnja točka grupe
3. Dodijeliti svaki podatak grupi s najbližim težištem
4. Stati kada se dodjela ne promijeni; inače vratiti se na korak 2.

Jedan od glavnih problema, može se reći izazova, ovog algoritma je upravo to što algoritam provodi samo grupiranje podataka. Potrebno je naknadno provesti analizu značenja grupa pronađenih algoritmom. Pojednostavljeni primjer algoritma za $k = 3$ prikazan je na slici 5.

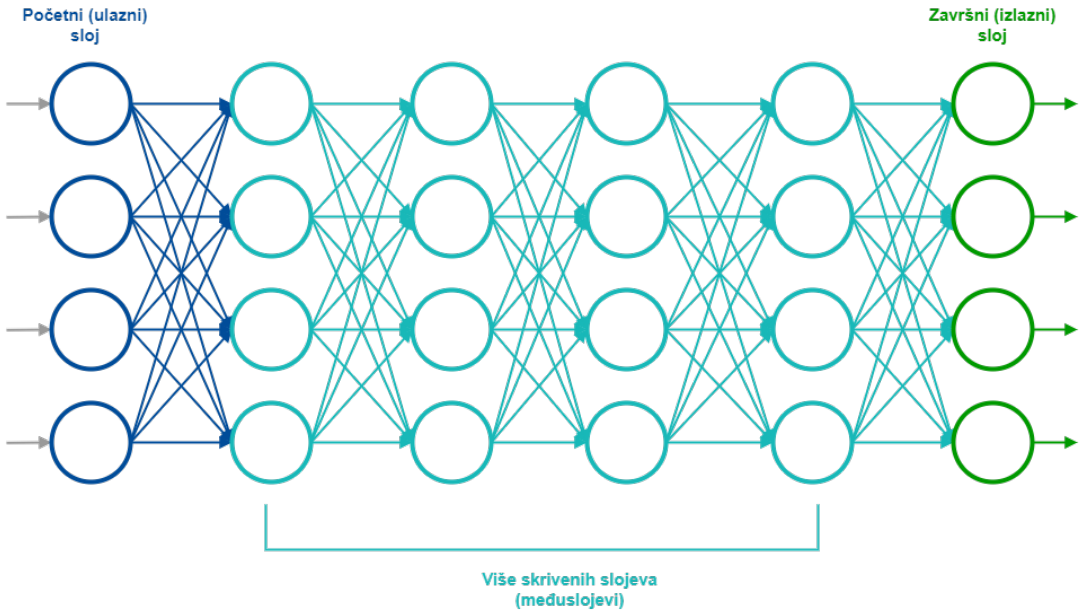


Slika 5: Algoritam K-srednjih vrijednosti (engl. *k-means algorithm*)

2.3. Duboko učenje

Duboko učenje, duboka neuronska mreža ili duboko strukturirano učenje vrsta je strojnog učenja temeljena na umjetnim neuronskim mrežama. Unutar neuronskih mreža višestruki slojevi obrade reprezentacijskim učenjem izvlače značajke sve više razine iz podataka [30]. Reprezentacijsko učenje jedno je od tehnika pristupa strojnom učenju koje sustavu iz neobrađenog seta podataka daje mogućnost otkrivanja željenih značajki ili klasifikacije [31].

Titulu oca dubokog učenja dijeli više znanstvenika koji se bavili njezinim razvojem. Jedan od njih bio je Frank Rosenblatt [32]. On se među prvima bavio područjem dubokog učenja te ga je proučavao sve do svoje tragične smrti 1971. godine. Samo tri godine nakon njegove smrti započinje prva "zima umjetne inteligencije" koja traje sve do 1982. godine i njihovog ponovnog populariziranja i financiranja [6]. No, i dalje je uspio razviti i istražiti sve osnovne dijelove današnjih sustava dubokog učenja poput perceptrona - umjetnog neurona. Time je postavio temelje za daljnji razvoj neuronskih mreža. Osim njega, titulu zaslužuju i Geoffrey Hinton, Yann LeCun i Yoshua Bengio. Radeći neovisno, ali i zajedno, Hinton, LeCun i Bengio su uspjeli razviti konceptualne temelje područja neuronskih mreža i dubokog učenja [33]. Kroz teorijski i eksperimentalni rad pridonijeli su napretku koji je pokazao praktične prednosti dubokih neuronskih mreža za što su i nagrađeni Turingovom nagradom 2019. godine.



Slika 6: Duboka neuronska mreža (engl. *deep neural network*)

Slika 6 pokazuje pojednostavljeni prikaz duboke neuronske mreže. Sastoji se od početnog (ulaznog) sloja i završnog (izlaznog) sloja te više međuslojeva, odnosno skrivenih slojeva (engl. *hidden layers*) [30]. Prvi sloj, tzv. ulazni sloj (engl. *input layer*) je sloj unutar kojeg su informacije poznate, gdje svaki neuron predstavlja pojedinačnu značajku iz danog uzorka našeg skupa podataka. Naredna, u ovom slučaju, 4 sloja su tzv. skriveni slojevi. Nazivaju se skriveni jer su prave vrijednosti unutar svakog neurona nepoznate unutar skupa podataka za treniranje. Izlazni sloj (engl. *output layer*) posljednji je sloj neuronske mreže te je sloj unutar ko-

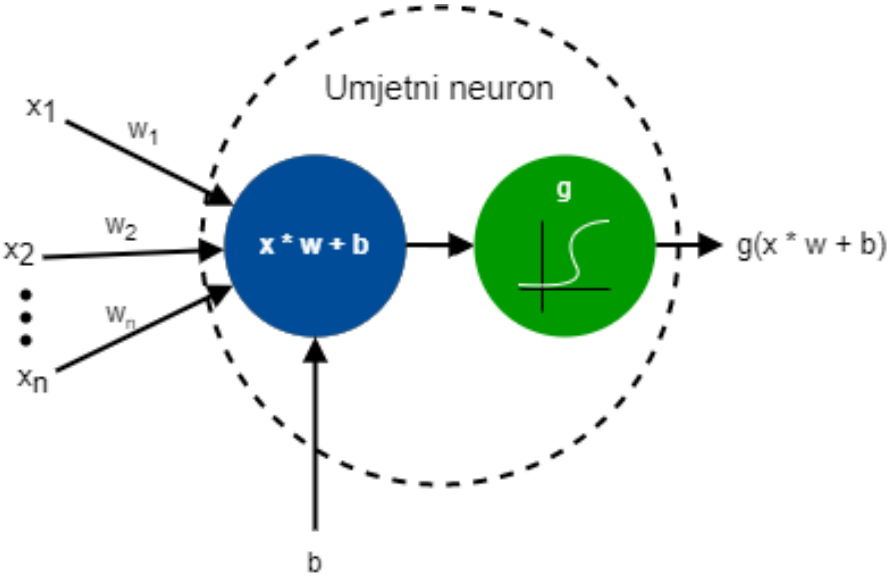
jeg se dobivaju željena predviđanja. Broj neurona svakog sloja ovisi o kompleksnosti zadanog problema.

U nastavku, na slici 7, prikazana je ilustracija umjetnog neurona. Svaki neuron prima n ulaznih vektora x s odgovarajućim vrijednostima težine w. Obrađujući ulazne vektore, odnosno izračunavajući njihov ponderirani zbroj, pri čemu se dodaje i težina pristranosti, dobiva se formula:

$$x \times w + b.$$

Rezultirajuća formula djeluje kao ulazna varijabla aktivacijske funkcije g, pri čemu njezin rezultat u konačnici definira izlaz neurona. Umjetne neuronske mreže [7, str. 59] su, jednostavno rečeno, matematički modeli koji definiraju funkciju:

$$f : x \to y.$$



Slika 7: Struktura umjetnog neurona; prema [7, str. 59]

Zbog velikog potencijala razvoja u gotovo svim smjerovima, duboko se učenje danas primjenjuje u raznim područjima ljudskih života. Samo neke od primjena su u razvoju robota (za vojsku, u medicinske svrhe ili svakodnevni život), koristi se u prepoznavanju i restauraciji slika, u marketingu, otkrivanju prijevara, obradi prirodnog jezika i dr. [30]

2.3.1. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (engl. *convolutional neural network*) najpoznatija je vrsta dubokih neuronskih mreža s posebnom primjenom u analizi vizualnih slika i otkrivanju objekata, tj. s primjenom u procesiranju jedno i višedimenzionalnih podataka [34]. Inspirirane biološkim procesima, konvolucijske neuronske mreže (KNM) predstavljaju veliku prednost u odnosu na druge algoritme za klasifikaciju slika. Razlog tomu je taj što KNM automatski otkriva važne značajke podataka bez ljudskog nadzora. Na primjer, uzmu li se u obzir slike jabuka i

banana, ono samostalno uči važne značajke za svaku od te dvije klase.

Kao i kod tradicionalne neuronske mreže i konvolucijska neuronska mreža sastoji se od jednog ulaznog, jednog izlaznog i jednog ili više skrivenih slojeva. Ono što je specifično za njih su poseban tip skrivenih slojeva među kojima je i jedan po kojemu je mreža dobila i ime. Skriveni slojevi sastoje se od jednog ili više konvolucijskih slojeva (engl. *convolutional layer*) te jednog ili više sloja sažimanja (engl. *pooling layer*) koji se izmjenjuju nekoliko puta nakon čega dolazi na red jedan ili više potpuno povezani sloj (engl. *fully connected layer*) [34].

3. Poticano učenje

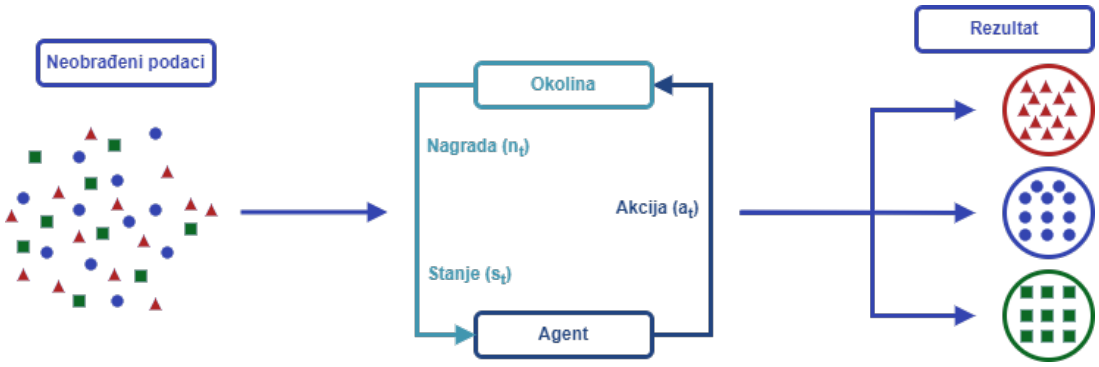
Poticano učenje vrsta je strojnog učenja nastalo po uzoru na biheviorističku psihologiju [7, str. 71], odnosno po uzoru na način na koji uče ljudi i životinje.

Za razliku od nadziranog učenja, poticano učenje nije nadgledano. Algoritam u tom slučaju ne donosi odluke isključivo na temelju podataka danih na početku, već jedna odluka utječe na drugu te se one donose sekvencijalno. Jedina povratna informacija, na temelju koje algoritam uči, prikazana je u obliku pozitivne ili negativne nagrade [7, str. 71]. Povratne informacije mogu biti i odgođene, odnosno dane s određenim vremenskim zakašnjenjem. Primjer može biti šah ili neka druga društvena igra gdje je moguće odigrati puno različitih poteza. Dobri i loši potezi su poznati tek na kraju igre kada se sazna rezultat.

Razlika između poticanog učenja i nenadziranog učenja je u tome što nenadzirano učenje pokušava pronaći nekakvu strukturu u podacima te ih grupirati. U poticanom učenju, algoritam se trudi kroz proces pokušaja i pogrešaka naučiti što napraviti u kojoj situaciji. Za primjer se može uzeti sustav preporuke videa korisnicima platforme YouTube. Algoritam kreiran nenadziranim učenjem sugerirat će videe na temelju korisnikove povijesti gledanja videa, tj. predložit će one videe koji su slični prethodno pogledanima. Algoritam poticanog učenja predlagat će razne videe, dobivati povratne informacije te na temelju toga učiti koji tipovi videa bi se svidjeli osobi.

S obzirom na usporedbe tipova strojnog učenja, prikazan je kratki sažetak karakteristika poticanog učenja [35]:

- vrijeme igra ključnu ulogu u treniranju agenta
- ne postoji osoba koja nadgleda i daje instrukcije agentu
- akcije agenta utječu na podatke koje agent prima kroz sustav nagrada i kazni
- najbolje rješenje problema rezultat je najvećih ostvarenih nagrada tijekom treninga
- odlučivanje agenta je sekvencijalno
- povratna informacija nije trenutna.



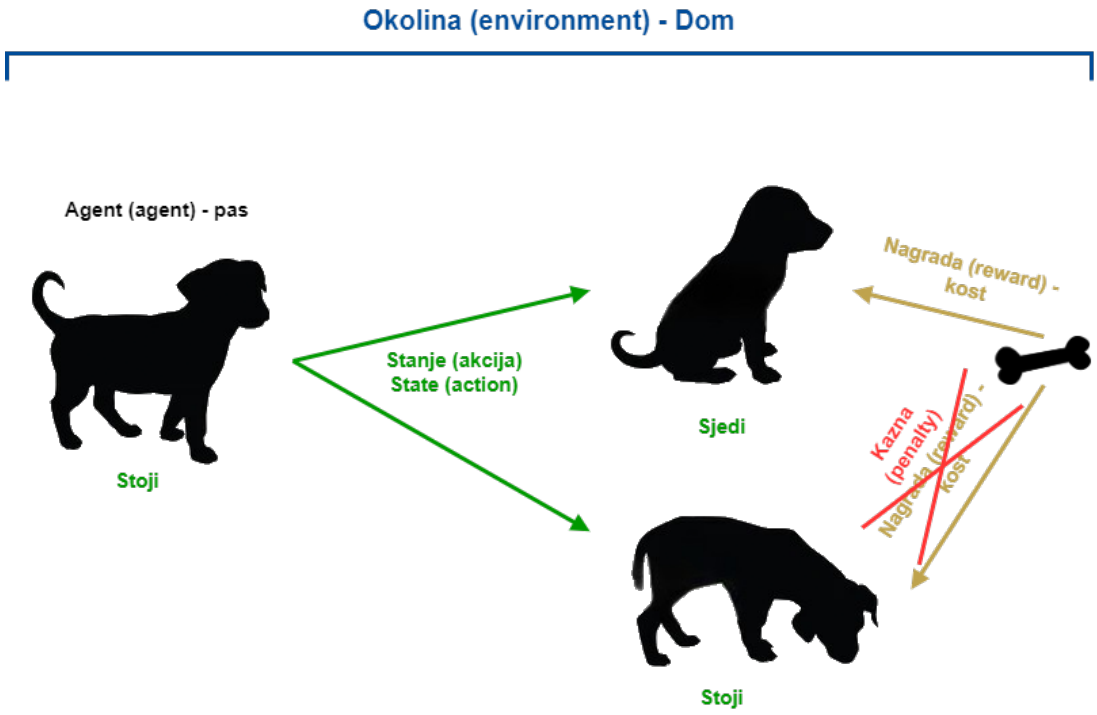
Slika 8: Poticano učenje (engl. *reinforcement learning*); prema [21]

Neki od najvažnijih pojmova unutar poticanog učenja su [35] [7, str. 15, 18, 73]:

- agent (engl. *agent*) - subjekt koji izvodi radnju unutar danog okruženju u svrhu maksimiziranja nagrade
- okolina (engl. *environment*) (e) - scenarij koji okružuje agenta i na koji agent utječe akcijama
- akcija (engl. *action*) (A) - radnja koju agent obavlja radi prijelaza u naredno stanje
- skup akcija (engl. *action space*) - skup svih radnji koje agent može izvesti
- nagrada (engl. *reward*) (R) - mehanizam poticaja i pokude agentu nakon obavljanja određene akcije
- stanje (engl. *state*) (s) - zapažanje koje agent zaprima iz okoline (trenutni položaj)
- skup stanja (engl. *state space*) - skup svih stanja koje agent može poprimiti
- strategija (engl. *policy*) (π) - agentova strategija na temelju koje donosi odluke o sljedećoj akciji ovisno o trenutnom stanju
- vrijednost (engl. *value*) (V) - dugoročni povrat, u usporedbi s nagradom koja je kratkoročni povrat
- epizoda (engl. *episode*) - konačan niz stanja, akcija i nagrada koji završava završnim stanje (e.g. smrt agenta).

Navedeni pojmovi bitni su u treniranju algoritma te se pojavljuju kroz cijelo poglavlje "4. Primjena poticanog učenja na primjeru jednostavne videoigre".

Osnovni pojmovi vidljivi su u nastavku na slici 9 koja pokazuje pojednostavljeni primjer generalnog načina funkcioniranja poticanog učenja. U ovom primjeru potrebno je psa naučiti kako sjesti na danu zapovijed. Iz razloga što pas ne razumije ljudski jezik pa mu nije moguće na taj način objasniti što napraviti, potrebno ga je naučiti poslušati na drugi način. To je moguće ostvariti koristeći nagrade. U trenutku kada pas na danu zapovijed sjedne, dobije kost, a ukoliko ne sjedne nagrada mu neće biti dana. Očekujući nagradu, pas sve više pozitivno reagira na zapovijed te na taj način s vremenom uči što mora napraviti. Ako se primjer translatira na osnovne pojmove, okolina unutar koje se odvija cijela radnja može biti dom, agent bi u tom slučaju bio pas, stanja bi bila položaji u kojima se pas nalazi (sjedi, stoji, trči), a akcija je prijelaz iz stanja stajanja u stanje sjedenja. Nagrada za dobro donesenu odluku je kost, dok je kazna samo nedostatak nagrade. U stvarnosti je, kao što će biti u narednom poglavlju demonstrirano, treniranje virtualnog agenta mnogo zahtjevniji zadatak. Teško je definirati sustav nagradi i kazni koji će na kvalitetan i brz način naučiti agenta što mora činiti.

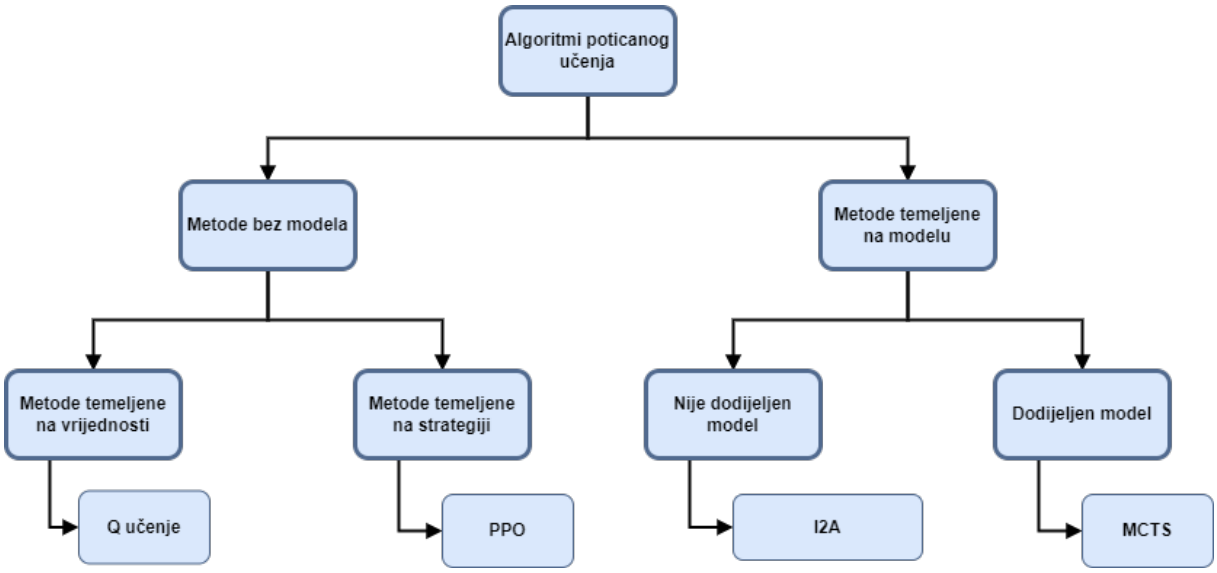


Slika 9: Pojednostavljeni primjer načina funkcioniranja poticanog učenja

Poticano učenje dobro djeluje na tipovima problema gdje postoji više različitih komponenata u interakciji te je najveća prednost ovog tipa učenja lakoća prilagodbe na neizvjesnost. Zbog toga se danas vrlo često koristi u igrama, robotici, računalnom vidu, financijskom sektoru, zdravstvenom sektoru i generalno u raznim strukturama unutar poslovanja. S druge strane, ono nije prikladno za tipove problema kod kojih je jednostavno vidjeti rješenje, kada postoji previše stanja ili kod problema koji nisu stabilni, odnosno ujednačeni. U tom slučaju nema smisla razvijati strategiju ako se ona kroz vrijeme mijenja. Naravno u realnom svijetu postoje i ograničenja resursa, stoga se poticano učenje ne koristi kod problema koji imaju ozbiljne posljedice ili kojima je potrebno previše vremena i/ili novca. Jedan od takvih problema je izgradnja autonomnog vozila gdje jedna greška (koje su temelj poticanog učenja) može nanijeti mnogo štete. Usprkos tome, i dalje se koristi u dijelu njegovog razvoja, ali u simuliranim uvjetima.

3.1. Algoritmi poticanog učenja

S obzirom na prethodno poglavlje koje se bavi definiranjem pojma poticanog učenja, potrebno je objasniti i taksonomiju algoritama korištenih u ovoj vrsti strojnog učenja. Detaljna kategorizacija algoritama prikazana [36, poglavlje 3] je u nastavku, pri čemu je za svaku kategoriju dan jedan primjer.



Slika 10: Taksonomija algoritama poticanog učenja; prema [36, poglavlje 3]

3.1.1. Metode temeljene na modelu

Za metode temeljene na modelu potrebno je napraviti razliku između toga je li algoritmu poznat model okoline, odnosno je li model učinjen dostupnim ili je model nepoznat te ga algoritam mora samostalno naučiti. Zbog toga nastaje razlika unutar ove metode na dani model, gdje je model poznat, i na naučeni model, gdje je model nepoznat.

3.1.1.1. Dodijeljen model

Ukoliko je model poznat, može mu se izravno pristupiti u trenutku učenja optimalne strategije ili funkcije vrijednosti. Simulirajući akcije, odnosno moguće putanje i izračunavajući nagrade provedenih radnji, odabire se ona akcija koja vraća najveću nagradu. Simulacije se obično provode uz pomoć algoritama pretraživanja koji zatim stvaraju stabla odluke. U takvoj konfiguraciji trenutno stanje bi bio korijenski čvor, a svaki sljedeći čvor ili list bi bilo stanje koje se može dohvatiti provodeći određeni niz akcija [37]. Optimalnu strategiju je na ovaj način vrlo lako pronaći.

Problem s klasičnim stablima odluke, odnosno algoritmom pretraživanja, proizlazi kod kompleksnijih problema, odnosno problema s velikim skupom akcija. Razlog tome je taj što broj mogućih akcija raste eksponencijalno sa složenijim problemima. Primjer algoritma koji se primjenjuje na takvim tipovima problema je upravo Monte Carlo pretraživanje stabala (engl. *Monte Carlo Tree Search - MCTS*).

Pretraga stabla Monte Carlo iterativan je algoritam koji pokušava pronaći najbolji potez ponavljajući korake [38]:

- izbora (engl. *selection*)
- proširenja (engl. *expansion*)

- simulacije (engl. *simulation*)
- propagacije unatrag (engl. *backpropagation*).

Prije nego što se do kraja objasni algoritam, potrebno je napomenuti kako svaki čvor pohranjuje dva broja - jedan određuje uspješnost, dakle broj dobivenih partija, a drugi sveukupni broj partija [39, str. 28]. Početna je konfiguracija algoritma korijen stabla, odnosno algoritam se kreće od djelomično izgrađenog stabla te se spušta prema listovima na temelju izabrane mjere poželjnosti poteza. Mjera zapravo određuje koliko će se algoritam posvetiti istraživanju, a koliko iskorištavanju. U trenutku kada algoritam dođe do lista provjerava je li on konačan ili nije. Ako nije, stvara se novi list (ili više njih ovisno o danom modelu) te se odabire jedan list. Zatim se provodi simulacija od izabranog djeteta, dok se ne postigne završna konfiguracija. Na kraju, kada se dođe do lista koji čuva terminalno stanje (kraj igre), ažurira se statistika svakog roditelja i naravno čvora iz kojeg je simulacija krenula. Ovaj algoritam korišten je i u AlphaGo softveru spomenutom na početku završnog rada [37]. Softver je razvila tvrtka DeepMind Technologies, podružnica tvrtke Google, odnosno Alphabet Inc. Nakon pobjede protiv najboljeg Go igrača svijeta, AlphaGo softver je umirovljen te je razvijena još snažnija verzija poznata pod nazivom AlphaZero koji je naučio i dodatne igre poput šaha [40].

3.1.1.2. Nije dodijeljen model

Ukoliko model nije poznat, potrebno je prvo naučiti model okoline prije nego što se implementira strategija ili funkcija vrijednosti. Jedini način na koji je moguće učiti je da algoritam dođe u interakciju s modelom okoline. Jedan od algoritama koji uči na taj način je algoritam agenata s proširenom maštom (engl. *Imagination-Augmented Agents - I2A*).

I2A nova je arhitektura za duboko poticano učenje koja kombinira dijelove metoda temeljenih na modelu i bez modela. Jedna od prednosti ovog algoritma je taj što je sposoban rukovati s naučenim modelima što znači da je sposoban rukovati s potencijalno nesavršenim modelima okoline. Oslanjajući se na modele okoline, koji s obzirom na trenutne informacije mogu pokušati predvidjeti budućnost, agente nastale metodom bez modela i algoritmom I2A moguće je ispuniti maštom. Neuronska mreža koristi modele okoline kako bi simulirala zamišljene putanje i time pružila dodatni kontekst razvoju strategije agenta. Potrebno je naglasiti kako je I2A algoritam nadmašio algoritme nastale isključivo bez modela u proceduralno generiranoj igri Pac-Man i Sokoban zagonetkama. [41]

3.1.2. Metode bez modela

Kao što sam naziv kaže, metode bez modela ni ne pokušavaju izgraditi model okoline. U ovom slučaju, agent komunicira s okolinom i na temelju provedene komunikacije pokušava poboljšati svoj učinak [36, poglavlje 3]. Uspoređujući metode temeljne na modelu s metodama bez modela, metode temeljene na modelu su znatno kompliciranije za implementirati upravo zbog modela okoline koji može biti teško naučiti. No, metode bez modela, upravo jer ne mare za model, kreiraju novi set problema. Zbog nužnog istraživanja okoline, kako bi se naučila

optimalna strategija, cijena razvoja je često neisplativa u smislu vremena, novca, opreme i/ili sigurnosti [36, poglavlje 3]. U stvarnom svijetu razvoj modela za npr. autonomno vozilo si ne može priuštiti cijenu istraživanja okoline. To je prihvatljivo isključivo u izrazito simuliranim uvjetima.

3.1.2.1. Metode temeljene na strategiji

Metode temeljene na strategiji pokušavaju kreirati takvu strategiju koja će za poduzetu akciju u danom stanju donijeti najveću, odnosno maksimalnu nagradu [36, poglavlje 3]. Dodatna podjela može se napraviti s obzirom na to temelji li se algoritam na strategiji ili bez strategije [7, str. 73]. Algoritmi koji se temelje na strategiji (engl. *on-policy*) tijekom učenja koriste tu strategiju kako bi odlučili koju akciju poduzeti ovisno o stanju. S druge strane, kod algoritama koji se temelje na metodi bez strategije (engl. *off-policy*), iako strategija postoji, ona se ne uzima u obzir te svoje odluke donosi pohlepno.

Primjer algoritma koji pripada metodama temeljenima na strategiji može biti algoritam optimizacije proksimalne politike (engl. *Proximal Policy Optimization - PPO*). PPO koristi neuronske mreže za izračun idealne funkcije koja će preslikati agentova opažanja i na temelju njih izabrati najbolju akciju u određenom stanju [42]. Algoritam je izabran iz razloga što je korišten u treniranju agenta unutar praktičnog dijela završnog rada, vidljivo u 4. poglavlju.

3.1.2.2. Metode temeljene na vrijednosti

U metodama temeljnim na vrijednostima, cilj je maksimizirati funkciju vrijednosti $V(s)$ [43]:

$$V^{\pi}(s) = \mathbb{E}\sum_{k=0}^{\infty}[\gamma^{k} r_{t+k} | s_t = s, \pi],$$

pri čemu je $\mathbb{E}$ operator očekivane vrijednosti, π odabrana strategija, r nagrada, s stanje, a γ faktor umanjenja. Optimalni očekivani povrat prethodne funkcije maksimizira očekivanu vrijednost nagrade prema danoj strategiji π [43]:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s).$$

U ovoj metodi smatra se kako je strategija već dana. Ukoliko nije eksplicitno zadana, koristi se tzv. pohlepna strategija (engl. *greedy policy*) koja odabire najveću vrijednost.

3.2. Q-učenje

Q-učenje algoritam je poticanog učenja bez modela, temeljenog na vrijednosti i bez strategije [7, str. 74-75]. To znači da agent uči bez modela okoline u kojoj se nalazi i da ne slijedi danu strategiju, već pokušava maksimizirati konačnu nagradu pohlepnom strategijom. Kao i

kod ostalih algoritama poticanog učenja, potrebno je pronaći adekvatan kompromis između istraživanja i iskorištavanja (engl. *exploration-exploitation trade-off*).

Istraživanje i iskorištavanje dva su načina ponašanja [44] koje algoritam razmatra u trenutku donošenja odluka. S jedne strane, iskorištavanje je siguran pristup u kojem algoritam pokušava donijeti što bolju odluku na temelju skupljenih podataka do trenutka odluke. S druge strane, istraživanje je riskantniji pristup unutar kojeg algoritam zanemaruje trenutne podatke o najboljoj odluci i istražuje nove odluke s ciljem otkrivanja najbolje, ako ih ima.

Korake koje prati algoritam prikazani su na slici 11.

Prvi je korak kreirati Q-tablicu koja se sastoji od n redaka i m stupaca. Broj redaka određen je skupom mogućih stanja, dok je broj stupaca određen skupom mogućih akcija. Tablica se inicijalizira na način da se sve vrijednosti postave proizvoljno. Često se vrijednosti postavljaju na 0.
U drugom koraku algoritam odabire narednu akciju. Kao što je prethodno rečeno, Q-učenje koristi tzv. pohlepnu strategiju, pazeći na kompromis istraživanja i iskorištavanja.
Treći i četvrti korak sastoje se od izvođenja akcije a, odnosno promjene stanja s_t u stanje s_{t+1} te dobivanja povratne informacije u obliku nagrade.
Posljednji je korak ažurirati Q-tablicu te se vratiti na drugi korak. Kako bi nastala kvalitetna Q-tablica, potrebno je odraditi veliki broj iteracija.

Slika 11: Algoritam Q-učenja; prema [45]

Bellmanova jednadžba, vidljiva u nastavku, pokazuje kako je najveća buduća nagrada jednaka nagradi koju je agent ostvario prelaskom u trenutno stanje zbrojeno s maksimalnom budućom nagradom za prelazak u naredno stanje s' [46]:

$$Q(s,a) = r + \gamma \max_{\alpha'} Q(s',a').$$

Koristeći Bellmanovu jednadžbu kao bazu, nastala je formula Q-učenja pomoću koje se provodi ažuriranje Q(s, a) [7, str. 75]:

$$Q^n(s,a) \leftarrow Q(s,a) + \alpha\{r_{t+1} + \gamma \max_{\alpha'} Q(s',a') - Q(s,a)\}.$$

Alternativno, formula se može zapisati u obliku:

$$Q^n(s,a) = (1-\alpha)Q(s,a) + \alpha\{r_{t+1} + \gamma \max_{\alpha'} Q(s',a')\},$$

pri čemu je:

- $Q^n(s, a)$ - nova Q-vrijednost za polje s i akciju a
- α - stopa učenja (engl. *learning rate*) $\alpha \in [0, 1]$
- $Q(s, a)$ - trenutna Q-vrijednost za polje s i akciju a
- r - neposredna nagrada za poduzimanje akcije a u stanju s
- γ - faktor umanjenja (engl. *discount factor*) $\gamma \in [0, 1]$
- $\max_{\alpha'} Q(s', a')$ - najveća buduća nagrada za novo stanje s' i sve dostupne akcije a'.

Faktor umanjenja ponderira važnost nagrada. Što je γ bliža 0, veća se težina pridodaje trenutnoj nagradi, a što je bliža 1, veća se težina dodaje budućoj nagradi. Stopa učenja određuje u kojoj će mjeri nova procjena za Q biti nadjačana u odnosu na staru procjenu. [7, str. 75]

Danas, Q-učenje koristi se u problemima poticanog učenja gdje postoji konačan broj stanja i akcija, što je jasno samo po sebi s obzirom na način na koji se algoritam provodi.

3.3. Markovljev proces odlučivanja

Markovljev proces odlučivanja matematički je okvir nastao 1960. godina [47, str. 1]. Naziv potječe od imena ruskog matematičara 19. i 20. stoljeća Andreja Andrejeviča Markova. Njegova istraživanja teorije stohastičkih procesa, kasnije nazvana Markovljevi lanci, preteča su Markovljeva procesa odlučivanja [48].

Markovljevi lanci ili Markovljevi procesi stohastički je model koji prelazi iz stanja u stanje unutar ograničenog broja mogućih stanja. Model sadrži skup stanja i vjerojatnosti, pri čemu sljedeće stanje znatno ovisi o trenutnom. Razlika između Markovljevih lanaca i Markovljeva procesa odlučivanja (MPO) je u dodatku nagrada i akcija u cijeli proces [49, str. 482]. Kada bi se MPO sastojao od jedne akcije i jednakih nagrada nakon prijelaza u novo stanje, cijeli proces bi se mogao svesti na Markovljeve lance. Akcije, i povratna informacija o nagradi nakon promjene stanja, daju mogućnost izbora o daljnjem koraku.

Parametri Markovljeva procesa odlučivanja dio je petorke (S, A, P, R, γ) [49, str. 482-483]:

- S - skup stanja sustava $s_t \in S$
- A - skup stanja akcija $a_{s_t} \in A$
- P - matrica prijelaznih vrijednosti, $P_{a_t}(s_t|s) = Pr(s(t+1) = s'|s_t, a_t)$ - vjerojatnost da će akcija a_t u stanju s_t u vremenskom intervalu t dovesti do stanja s' u vremenu $t + 1$
- R - neposredna nagrada nakon prijelaz u novo stanje $R_{a_t}(s, s')$

- γ - faktor umanjenja $\gamma \in [0, 1]$, predstavlja razliku važnosti sadašnjih i budućih vrijednosti nagrada.

Metode rješavanja problema pomoću Markovljeva procesa odlučivanja najčešće uključuju dinamičko programiranje. Kako bi se pojednostavio, problem se rekurzivnim dinamičkim programiranjem rastavlja na dijelove, istovremeno pamteći optimalna rješenja svakog rastavljenog dijela [3, str. 553]. Tri su osnovne grane Markovljeva proces odlučivanja: MPO s kontinuiranim vremenom, MPO s diskretnim vremenom i polu-Markovljevi procesi odlučivanja [47, str. 4].

Primjena je moguća u mnogim područjima poput obrade signala, ekonomije i komunikacija. Generalno govoreći Markovljevi procesi odlučivanja koriste se za probleme optimizacije gdje je ishod djelomično pod kontrolom i djelomično slučajan.

4. Primjena poticanog učenja na primjeru jednostavne videoigre

Praktični dio završnog rada prikazan je u ovom poglavlju. Sastoji se od samostalno izrađene jednostavne 2D videoigre te kreiranja agenta koji samostalno pokušava naučiti igrati videoigru. Videoigra "Deometry Gash" nastala je po uzoru na popularnu igru "Geometry Dash", no s određenim promjenama. Osim promjena u vizualu - što promjena generalnog dojma, što promjena objekata koji ubijaju igrača, najveća promjena je u razinama. Geometry Dash sastoji se od više razina različite inkrementalne težine, dok se Deometry Gash sastoji od jedne proceduralno generirane razine.

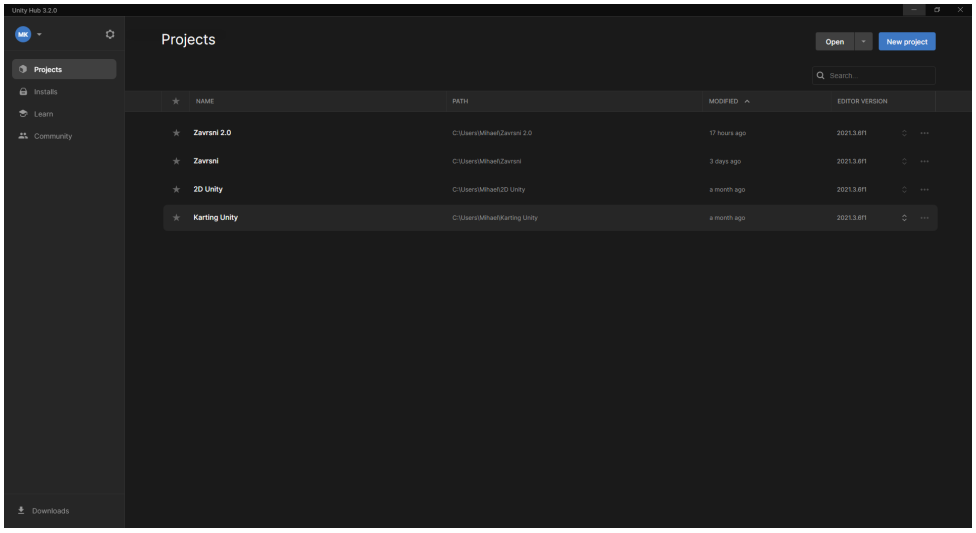
Alati korišteni u izradi praktičnog rada su:

- Unity
  - platforma za razvoj igara, interaktivnih simulacija i drugo
  - korišten za izradu 2D videoigre
- Unity ML-Agents Toolkit
  - softver otvorenog koda za kreiranje inteligentnih virtualnih igrača
  - korštен za izradu, kreiranje i treniranje agenta
- Visual Studio
  - integrirano razvojno okruženje za razvoj softvera
  - korišten za programski dio razvoja videoigre unutar platforme Unity
- Draw.io
  - softver otvorenog koda za crtanje grafikona
  - korišten za izradu slika i dijagrama
- Pixilart
  - internetska stranica posvećena stvaranju i dijeljenju pikseliziranih slika
  - korišten za izradu komponenata videoigre
- Paint
  - uređivač rasterske grafike
  - korišten za obradu slika.

Naravno, središnji alat rada bio je Unity zbog čega je dodatno opisan u nastavku.

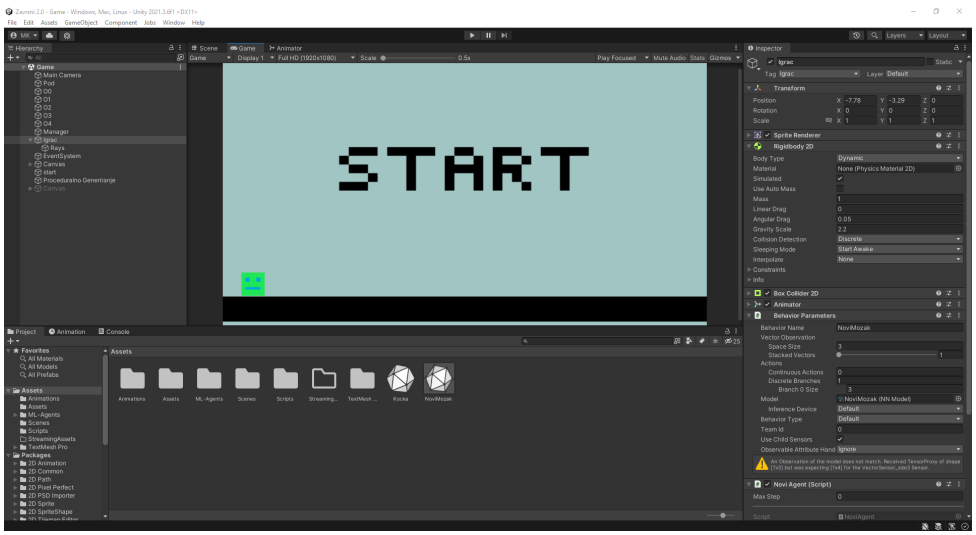
4.1. Unity kao game engine

Unity Hub aplikacija je unutar Unity sustava koja pojednostavljuje način na koji se pronalaze, preuzimaju i upravljaju Unity projekti i instalacije. Unity Hub sustavu pridodaje mogućnost upravljanja računom i licencama, pokretanja različitih verzija Unityja, instalacije drugih komponenata, korištenja predložaka projekata kako bi se ubrzao njihov razvoj i još mnogo toga. Izgled Unity Huba vidljiv je u nastavku na slici 12. [50]



Slika 12: Korisničko sučelje alata Unity Hub

Unity je višeplatformski (engl. *cross-platform*) softver za izradu kako 3D i 2D igara tako i interaktivnih simulacija. Razvijen je od strane tehnološke tvrtke Unity Technologies te podržava razne platforme za računala, konzole, mobilne uređaje i virtualnu stvarnost. Iako je originalno namijenjen za izradu igara, Unity se danas koristi u filmskoj industriji, u arhitekturi, građevinarstvu, automobilskoj industriji, a koriste ga i Oružane snage Sjedinjenih Američkih Država. [51]



Slika 13: Korisničko sučelje alata Unity

4.2. Proceduralno generiranje

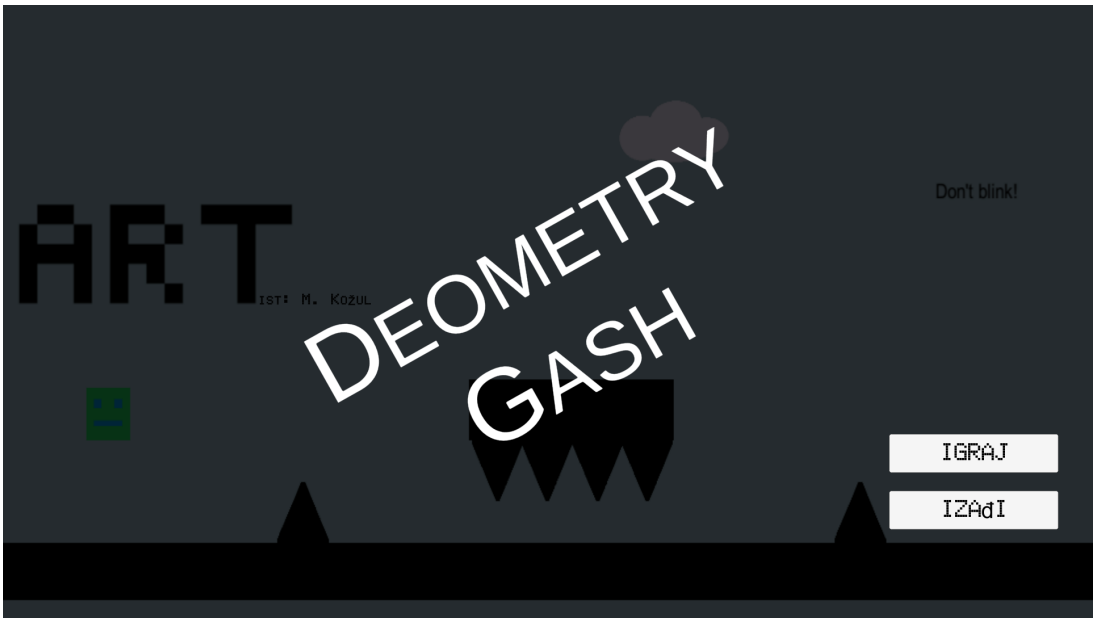
Proceduralno generiranje pojam je kojim se opisuje postupak generiranja sadržaja iz različitih matematičkih funkcija s nasumičnim unosom [52, str. 8]. Suprotnost je tome ručno programiranje i izrada sadržaja. Algoritmi primaju ulazne varijable poput nasumično generiranog broja ili skupa parametara koje koriste u procesu stvaranja objekata. Generiranje samo po sebi uz ulazne vrijednosti koristi i skup različitih funkcija i pravila na temelju kojih kreira sadržaj [52, str. 1].

Prednosti proceduralnog generiranja su mnoge. Jedna od glavnih prednosti je jednostavnost generiranja sadržaja, pri čemu se stvara mogućnost kreiranja razine ili svijeta naizgled beskonačne duljine. Osim što se na taj način štedi na vremenu pri generiranju sadržaja, velika je prednost i znatna ušteda u prostoru. U igri Elite iz 1984. godine, moguće je istraživati preko 2000 zvijezda zauzimajući malo iznad 20 kb [52, str. 4], a u igri No Man's Sky iz 2016. godine, 18 trilijuna planeta zauzimajući 6-11 gb prostora na disku [53]. S druge strane, teško je, no ne i nemoguće, generirati unikatan i zanimljiv sadržaj. Računala, a i algoritmi koji se koriste u proceduralnom generiranju nisu kreativni ili originalni, stoga ni kvaliteta sadržaja nije zadovoljavajuća. Zbog tog problema, veliki je broj kritika bio upućen na račun igre No Man's Sky. Planeta je bilo mnogo, no razlika između njih bila je minimalna. Rješenje se danas često pronalazi u hibridinom pristupu gdje se proceduralno generiranje koristi kao početna točka nakon koje se ručno poprave i/ili nadograde željeni dijelovi [54].

Proceduralno generiranje sadržaja koristi se u industriji igara već 40 godina, što je vidljivo na primjeru prethodno spomenute igre Elite, te njegova popularnost i dalje raste. Samo neke od popularnih igara današnjice koje koriste proceduralno generiranje u nekom obliku su No Man's Sky, Minecraft, Terraria, Left 4 Dead 2, Valheim, Civilization, Subway Surfers i Microsoft Minesweeper [17, str. 670]. Uglavnom se koristi za generiranje 3D, odnosno 2D svjetova, no često se proceduralno generiranje koristi i za dizajn likova, dijalog i animacije.

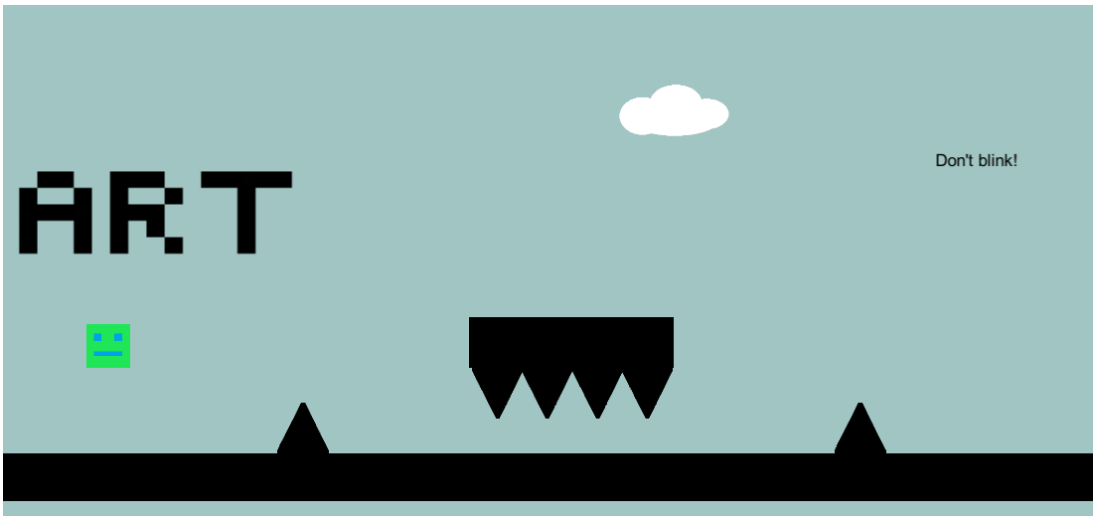
4.3. Razvoj videoigre

Kao što je i ranije rečeno videoigra kretivnog naziva *Deometry Gash* nastala je po uzoru na popularnu, originalno isključivo mobilnu igru *Geometry Dash* razvijenu 2013. godine. Obje igre funkcioniraju po jednostavnom principu kontroliranja pomičnog igrača koji ima mogućnost skakanja. Samostalno kreirana igra, osim toga ima i mogućnost obavljanja akcije čučnja. Ne može se kontrolirati brzina kojom se kreće lik, odnosno okolina, zbog čega su vrlo bitni ritam i preciznost tijekom cijele partije. Za razliku od Geometry Dasha, Deometry Gash sastoji se od jedne proceduralno generirane razine, stoga je jedini cilj igrača preživjeti što je dulje moguće. Također, razlika između igara je i u načinu na koji se igrač kreće. Unutar igrice Geometry Dash lik se kreće prema objektima, dok to nije slučaj unutar Deometry Gasha. U ovom slučaju, lik stoji na mjestu, dok se generirani objekti kreću prema njemu. Treba napomenuti i kako se sami objekti ne pojavljuju u jednakim vremenskim intervalima kako bi se dodatno otežala igra, a kasnije i treniranje agenta.



Slika 14: Glavni izbornik

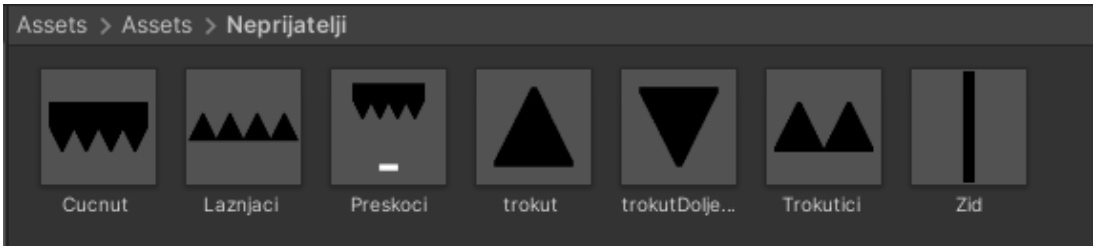
Igra je razvijena unutar Unity platforme za izradu igara, no za izradu korišteni su još i programi Visual Studio, Pixilart i Paint. Budući da je jezik alata Unity C#, u njemu je napisana cijela igra. Visual Studio korišten je za pisanje i uređivanje koda s obzirom na to da Unity sam po sebi nema implementiranu tu mogućnost. Internetski alat Pixilart korišten je za izradu komponenata igre kao što su igrač i objekti izbjegavanja, dok je Microsoft Paint korišten isključivo u svrhe jednostavnog uređivanja slika, poput slike pozadine glavnog izbornika. Za shvaćanje procesa stvaranja videoigre unutar alata Unity korišten je izvor [55], dok je za proceduralno generiranje korišten izvor [56].



Slika 15: Prikaz središnjeg dijela igre

Unutar samog projekta, proceduralno je generiranje korišteno u dvije različite instance. Jedna se implementacija bavi nasumičnom generacijom okoline za što su joj dostupna dva objekta - oblak i ptica. Druga implementacija bavi se bitnijom stavkom i to nasumičnim generi-

ranjem objekata koje igrač mora izbjeći. Oba načina primjenjivanja proceduralnog generiranja ne zahtijevaju veću razinu predznanja o proceduralnom generiranju ili programiranju unutar alata Unity. U prilogu 2 priložena je klasa *ProceduralnoGeneriranje.cs* koja je zadužena za nastanak objekata izbjegavanja. Proceduralna generacija okolnih objekata odvija se pomoću druge skripte.



Slika 16: Objekti izbjegavanja

Sedam je kreiranih objekata koje igrač mora izbjegavati, pri čemu ih tri mora preskočiti, jedan ignorirati, za dva čučnuti kako bi prošao ispod njih, a preko jednog ima mogućnost prolaska skokom ili čučnjem. Gubitkom, odnosno u trenutku doticaja s jednim od objekata, igrač gubi život te kreće ispočetka. Jedini objekt od kojeg igrač gubi život i mora krenuti ispočetka je trokut, uz njegove varijacije. Osim toga, moguće je izgubiti život i tako da se izađe iz vidljivog dijela ekrana. Tako nešto moguće je postići zabijanjem u zid ili određene dijelove drugih objekata.

Kako bi se skočilo unutar videoigre, potrebno je kliknuti gornju strelicu (↑), a kako bi se čučnulo potrebno je kliknuti donju strelicu (↓). Sam čučanj realiziran je koristeći animacije unutar alata. Klikom na gumb *Esc*, igra se zaustavlja te se otvara prozor pauze sa zaustavljenom igrom kao pozadinom. Iz tog prozora može se nastaviti s igrom ili odustati.

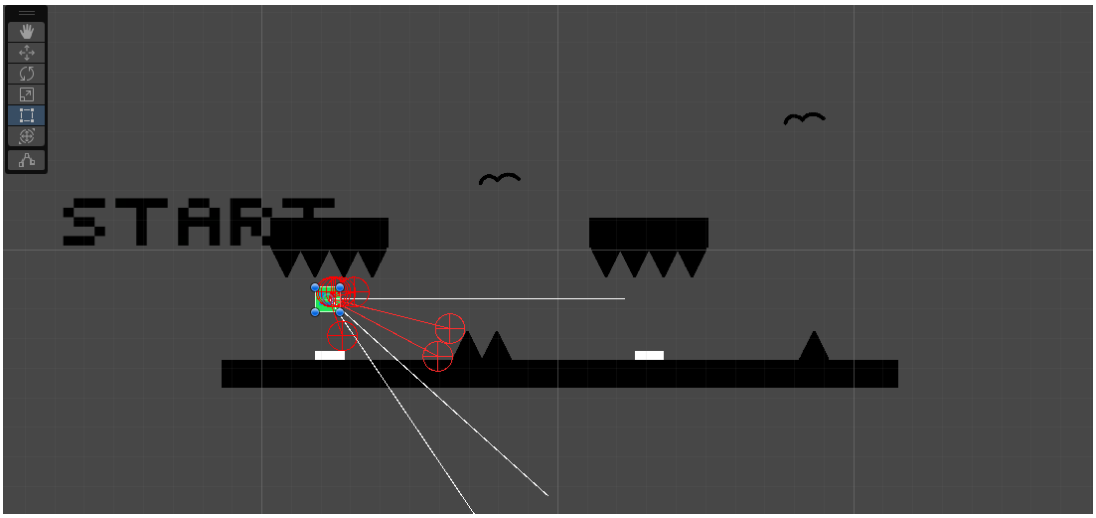
Slika 17: Kraj igre uz srušeni rekord

4.4. Razvoj agenta

Razvoj agenta ostvaren je uz pomoć Unity Machine Learning Agents Toolkita. ML-Agents Toolkit omogućio je ranije kreiranoj videoigri da služi kao okolina unutar koje se provodi obuka inteligentnog agenta. Sam alat baziran je na na okviru (engl. *framework*) strojnog učenja otvorenog koda - PyTorch. Svi dodatni alati potrebni za normalno fukcioniranje ML-Agents alata, vidljivi su unutar GitHub repozitorija [57] uz način instalacije. Kao pomoć u treniranju agenta korišteni su izvori [58] i [59].

Ranije spomenuti PPO algoritam jedan je od korištenih algoritama poticanog učenja unutar ML-Agents alata. Implementiran je unutar besplatne softverske biblioteke otvorenog koda (engl. *open-source*) TensorFlow te se izvodi u zasebnom Python procesu. S Unity aplikacijom, unutar koje je implementiran agent, komunicira preko spojne točke (engl. *socket*) [42].

Za treniranje agenta korištena je skripta vidljiva u prilogu 3. Sustav nagrađivanja funkcionira po principu; što je dulje agent preživio, to je konačna suma nagrada bila veća.



Slika 18: Agentov dodir s okolinom uz komponentu *Ray Perception Sensor 2D*

Kako bi se pokrenulo treniranje agenta potrebno je unijeti sljedeće naredbe unutar naredbenog retka. Prvom naredbom mijenja se trenutni radni direktorij i ulazi u direktorij projekta.

Isječak kôda 1: Promjena direktorija

```
C:\Users\Mihael>cd Zavrsni 2.0
```

Drugom naredbom stvara se lokalna virtualna okolina specifična za projekt kako bi svi projekti mogli biti izolirani jedni od drugih.

Isječak kôda 2: Kreiranje virtualne okoline

```
C:\Users\Mihael\Zavrsni 2.0>venv\Scripts\activate
```

Uz naredbu *mlagent-learn* pokreće se trening agenta. Osim toga, potrebno je definirati i identifikacijsku oznaku treninga, što se ostvaruje s *–run-id=*. Dodavanje prilagođene konfiguracije nije potrebno. U slučaju da se ne definira konfiguracija, koristit će se originalno zadana

(engl. *default*) konfiguracija. U treniranju agenta korištena je prilagođena konfiguracija prikazana u prilogu 1.

Isječak kôda 3: Pokretanje treninga agenta uz prilagođenu konfiguraciju

```
(venv) C:\Users\Mihael\Zavrsni 2.0>mlagents-learn config/NazivKonfiguracije.yaml --run-id=
 TreningID
```

Jedna od mogućnosti za brže treniranje sigurno je kreiranje više uzastopnih okolina i agenata koji se uče u njima. Njihova se zapažanja zatim spremaju u zajednički "mozak". Ono što je prvo potrebno napraviti je izgraditi igru (engl. *build*) i unutar klasične naredbe za trening dodati lokaciju *.exe* datoteke (*–env=*). Za kraj, potrebno je proslijediti željeni broj okolina za stvaranje (*–num-envs=*).

Isječak kôda 4: Poseban trening uz više okolina

```
(venv) C:\Users\Mihael\Zavrsni 2.0>mlagents-learn config/NazivKonfiguracije.yaml --env=
 Lokacija/Gotove/Igre --num-envs=10 --run-id=TreningID
```

Nakon završetka posljednjeg treninga, unutar naredbenog retka upisuje se sljedeća naredba.

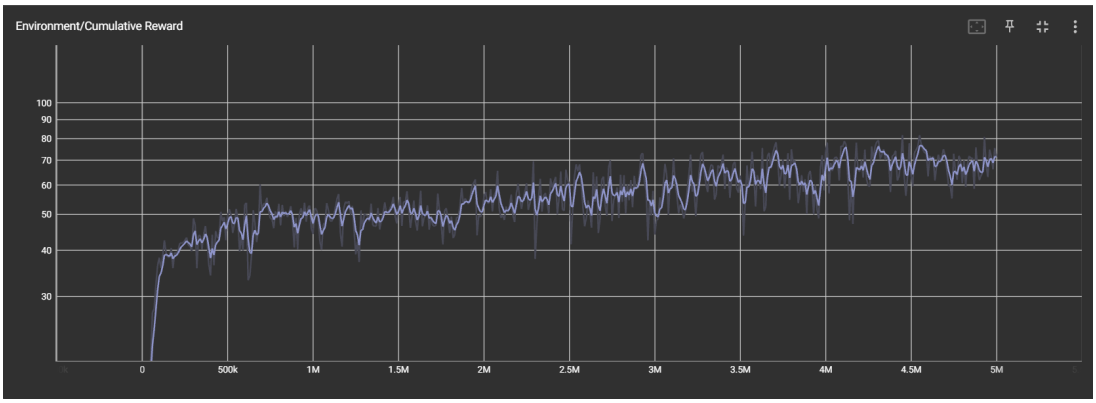
Isječak kôda 5: Vizualizacija rezultata treninga

```
(venv) C:\Users\Mihael\Zavrsni 2.0>tensorboard --logdir results
TensorFlow installation not found - running with reduced feature set.
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.10.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

Pomoću nje otvara se Tensorflowov alat za vizualizaciju podataka koji se otvara na pristupu (engl. *port*) 6006 gdje su prikazani rezultati svih provedenih treninga.

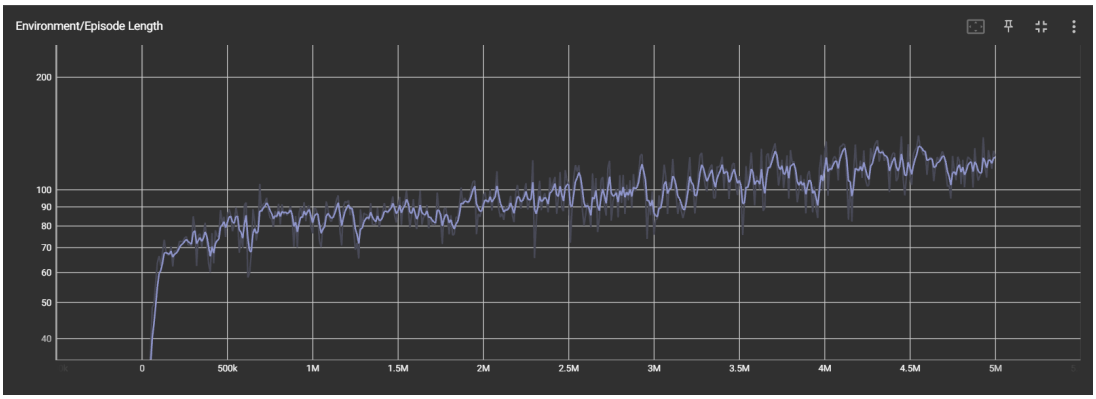
4.4.1. Rezultati

Konačni rezultat treniranja pokazao je koliko je zapravo vremenski i računalno zahtjevno istrenirati inteligentnog agenta. Primjera radi, posljednji trening koji je proveden unutar Unityja i kojemu je maksimalni broj koraka bio postavljen na 5 milijuna, trajao je 15 sati. Analizirajući rezultate treninga, vidljivo je kako je trening bio uspješan. Nagrade koje je dobivao agent kroz trening prikazane su na slici 19, a trajanje epizode na slici 20.



Slika 19: Prikaz dobivenih nagrada kroz trening

Promatrajući trening unutar alata, agent je često ostvarivao rezultate veće od 500, no nakon treninga, iako pokazuje znakove učenja, isti rezultat nije uspio replicirati. Rekord koji je postavio tijekom treninga bio je 1453, a nakon njega 431. Za rezultate koji bi se mogli usporediti s ljudskima potrebno je nastaviti s treningom i agentovim učenjem metodom sirove sile (engl. *brute-force*). Kao što je bilo i očekivano, proceduralno generirana razina te objekti izbjegavanja, koje je potrebno izbjeći različitim pristupima, pokazali su se kao znatno otežanje agentovom učenju.



Slika 20: Progresivno trajanje epizode za vrijeme treninga

Česti padovi Unityja tijekom treniranja agenta na većem broju koraka spriječili su dulje i zahtjevnije oblike treninga. Po uzoru na [60], procijenjeni broj maksimalnog broja koraka nakon kojega bi agent igrao na ljudskoj razini bio bi 25 milijuna.

5. Zaključak

Umjetna je inteligencija pojam koji je vrlo popularan u današnje doba. Jedan od razloga je zasigurno i ljudska fasciniranost inteligencijom i razvojem inteligencije strojeva. Danas, zahvaljujući strojnom učenju, to je i ostvarivo. Razvoj inteligencije neživih stvari započeo je sredinom 20. stoljeća i od tada doživio mnogobrojne uspone i padove. Zahvaljujući velikom tehnološkom napretku, u svakom dijelu svakodnevnog života moguće je pronaći neki oblik umjetne inteligencije. Prisutna je u videoigrama, automobilima, avionima, tvornicama i mnogim drugim tehnološkim aspektima. Zbog toga je potrebno razlikovati pojmove unutar polja umjetne inteligencije. Najkraće rečeno, duboko učenje je složeniji oblik neuronskih mreža, neuronske mreže su podskup strojnog učenja, a strojno učenje je disciplina unutar umjetne inteligencije. Često se inženjeri strojnog učenja znaju našaliti o razlici između strojnog učenja i umjetne inteligencije; ako je napisano u Pythonu onda je vrlo vjerojatno strojno učenje, a ako je napisano u PowerPointu vrlo je vjerojatno umjetna inteligencija.

Poticano učenje jedno je od tri grane strojnog učenja uz nadzirano i nenadzirano učenje. Nastalo po uzoru na način na koji uče ljudi i životinje, algoritam poticanog učenja kroz proces pokušaja i pogrešaka uči što napraviti u kojem trenutku. Osnovne karakteristike strojnog učenja su da ne postoji osoba koja nadgleda i daje instrukcije agentu, agent uči kroz sustav nagradi i kazni, vrijeme igra ključnu ulogu u učenju te je odlučivanje agenta sekvencijalno. Glavna podjela algoritama poticanog učenja je na metode bez modela i metode temeljene na modelu. Iz naziva je jasno kako je jednoj metodi dostupan model okoline iz kojega algoritam uči, dok drugoj nije, zbog čega model uči komunicirajući s okolinom. Najpoznatija metoda temeljena na modelu sigurno je pretraživanje stabala Monte Carlo. Jedan od razloga je i zato što je korišten u izradu poznatog Googleovog softvera AlphaGo. Q-učenje najpoznatiji je algoritam metoda bez modela. Razlozi tomu su jednostavnost i za shvaćanje i za implementiranje, ali i efikasnost.

Platforma Unity znatno olakšava izradu ne samo videoigara, nego i modela i simulacija. Iz tog je razloga izabrana kao glavni alat za izradu praktičnog dijela završnog rada. Kreirana videoigra nastala je po uzoru na poznatu igru Geometry Dash, no samo s jednom proceduralno generiranom razinom. Iako je nastala za treniranje agenta, igru je moguće igrati i samostalno. Razvoj inteligentnog agenta pokazao se vrlo zahtjevnim poslom. Jednostavnosti radi, za njegovu izradu i treniranje korišten je također Unity, točnije Unity Machine Learning Agents Toolkit. Treninzi su bili mnogobrojni i vrlo iscrpni, kako vremenski, tako i računalno. Agent je u konačnici uspješno istreniran, odnosno pokazuje zdravu putanju učenja, no da bi došao na ljudsku razinu potrebno je znatno više vremena.

Popis literature

[1] The Editors of Encyclopaedia Britannica. „Automaton," Encyclopedia Britannica. (19. 11. 2021.), adresa: https://www.britannica.com/technology/automaton (pogledano 1. 8. 2022.).

[2] Homer. „Ilijada." Preveo i protumačio: Tomo Maretić, CARNET. (), adresa: https://lektire.skole.hr/wp-content/uploads/2020/01/homer_ilijada.pdf (pogledano 1. 8. 2022.).

[3] S. Russell i P. Norvig, *Artificial Intelligence: A Modern Approach, Global Edition*, 4. izdanje, S. Russell i P. Norvig, ur. Pearson Education, 4. 2021., sv. 1167, ISBN: 978-1-292-40117-1.

[4] A. M. TURING, „I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, sv. LIX, br. 236, str. 433–460, 10. 1950., ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf.

[5] J. McCarthy, M. Minsky, N. Rochester i C. Shannon, „A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE," sv. 13, 8. 1955.

[6] O. Kaynak, „The golden age of Artificial Intelligence," *Discover Artificial Intelligence*, sv. 1, br. 1, str. 1, 9. 2021., ISSN: 2731-0809. DOI: 10.1007/s44163-021-00009-x.

[7] G. N. Yannakakis i J. Togelius, *Artificial Intelligence and Games*, G. N. Yannakakis i J. Togelius, ur. Springer International Publishing, 2. 2018., sv. 337, ISBN: 978-3-319-63519-4.

[8] PCWorld. „AlphaGo's unusual moves prove its AI prowess, experts say," PCWorld. (3. 2016.), adresa: https://www.pcworld.com/article/420054/alphagos-unusual-moves-prove-its-ai-prowess-experts-say.html.

[9] BBC News. „Google AI defeats human Go champion," BBC. (5. 2021.), adresa: https://www.bbc.com/news/technology-40042581.

[10] A. Rendón i M. Alvarado, „Pattern recognition and monte-carlotree search for go gaming better automation," sv. 7637, str. 11–20, 11. 2012. DOI: 10.1007/978-3-642-34654-5_2.

[11] The Editors of Encyclopaedia Britannica. „Artificial intelligence," Encyclopedia Britannica. (24. 8. 2022.), adresa: https://www.britannica.com/technology/artificial-intelligence (pogledano 24. 8. 2022.).

[12] ——, „Machine learning," Encyclopedia Britannica. (25. 8. 2022.), adresa: https://www.britannica.com/technology/machine-learning (pogledano 25. 8. 2022.).

[13] Cambridge University Press. „Deep learning," Cambridge University. (), adresa: https://dictionary.cambridge.org/dictionary/english/deep-learning (pogledano 4. 8. 2022.).

[14] The Editors of Encyclopaedia Britannica. „Neural network," Encyclopedia Britannica. (9. 1. 2020.), adresa: https://www.britannica.com/technology/neural-network (pogledano 5. 8. 2022.).

[15] Cambridge University Press. „Data science," Cambridge University. (), adresa: https://dictionary.cambridge.org/dictionary/english/data-science (pogledano 4. 8. 2022.).

[16] Oxford University Press. „Big data," University of Oxford. (), adresa: https://www.oxfordlearnersdictionaries.com/definition/english/big-data (pogledano 4. 8. 2022.).

[17] I. Millington, *AI for games, third edition*, 3. izdanje, I. Millington, ur. CRC Press, 3. 2019., sv. 1150, ISBN: 978-1-138-48397-2.

[18] K. Casey. „How to explain machine learning in plain english," The Enterprisers Project. (11. 2020.), adresa: https://enterprisersproject.com/article/2019/7/machine-learning-explained-plain-english (pogledano 6. 9. 2022.).

[19] Wikipedija. „Skriveno znanje," Wikipedia. (2021.), adresa: https://hr.wikipedia.org/wiki/Skriveno_znanje (pogledano 5. 9. 2022.).

[20] stratus inovations group. „Machine learning paradigms: Supervised, unsupervised, and reinforcement learning," stratus inovations group. (), adresa: https://stratusinnovations.com/blog/machine-learning-paradigms-supervised-unsupervised/ (pogledano 30. 8. 2022.).

[21] K. Goel. „Supervised learning vs unsupervised learning vs reinforcement learning," IntelliPaat. (24. 7. 2022.), adresa: https://intellipaat.com/blog/supervised-learning-vs-unsupervised-learning-vs-reinforcement-learning/ (pogledano 5. 8. 2022.).

[22] A. Alashqur, „Representation schemes used by various classification techniques – a comparative assessment," *International Journal of Computer Science Issues*, sv. 12, 11. 2015.

[23] Javatpoint. „Decision tree classification algorithm," Javatpoint. (), adresa: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm (pogledano 15. 8. 2022.).

[24] S. Gupta. „Popular classification models for machine learning," Analytics Vidhya. (30. 11. 2020.), adresa: https://www.analyticsvidhya.com/blog/2020/11/popular-classification-models-for-machine-learning/ (pogledano 4. 9. 2022.).

[25] M. Saeed. „A gentle introduction to approximation," Machine Learning Mastery. (18. 8. 2021.), adresa: https://machinelearningmastery.com/a-gentle-introduction-to-approximation/ (pogledano 4. 9. 2022.).

[26] eculidean. „Pros and cons of common machine learning algorithms," Medium. (), adresa: https://medium.com/@eculidean/pros-and-cons-of-common-machine-learning-algorithms-45e05423264f (pogledano 15. 8. 2022.).

[27] A. Ohri. „10 popular regression algorithms in machine learning of 2022," UNext Jigsaw. (16. 6. 2022.), adresa: https://www.jigsawacademy.com/popular-regression-algorithms-ml/ (pogledano 15. 8. 2022.).

[28] R. Bhatia. „Top 6 regression algorithms used in data mining and their applications in industry," Analytics India Magazine. (19. 9. 2017.), adresa: https://analyticsindiamag.com/top-6-regression-algorithms-used-data-mining-applications-industry/ (pogledano 15. 8. 2022.).

[29] J. Rocca. „The exploration-exploitation trade-off: Intuitions and strategies," Medium. (17. 2. 2022.), adresa: https://towardsdatascience.com/unsupervised-learning-algorithms-cheat-sheet-d391a39de44a (pogledano 20. 8. 2022.).

[30] W. contributors. „Deep learning," Wikipedia. (2022.), adresa: https://en.wikipedia.org/wiki/Deep_learning#cite_note-NatureBengio-2 (pogledano 26. 8. 2022.).

[31] ——, „Feature learning," Wikipedia. (2022.), adresa: https://en.wikipedia.org/wiki/Feature_learning (pogledano 25. 8. 2022.).

[32] Association for Computing Machinery. „Fathers of the deep learning revolution receive acm a.m. turing award," Association for Computing Machinery. (3. 2019.), adresa: https://www.acm.org/media-center/2019/march/turing-award-2018 (pogledano 30. 8. 2022.).

[33] Y. LeCun, Y. Bengio i G. Hinton, „Deep learning," *Nature*, sv. 521, br. 7553, str. 436–444, 5. 2015., ISSN: 1476-4687. DOI: 10.1038/nature14539.

[34] M. Mishra. „Convolutional neural networks, explained," Medium. (8. 2020.), adresa: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 (pogledano 31. 8. 2022.).

[35] D. Johnson. „Reinforcement learning: What is, algorithms, types & examples," Guru99. (16. 7. 2022.), adresa: https://www.guru99.com/reinforcement-learning-tutorial.html (pogledano 1. 9. 2022.).

[36] H. Dong, Z. Ding i S. Zhang, *Deep Reinforcement Learning Fundamentals, Research and Applications: Fundamentals, Research and Applications*, 1. izdanje, H. Dong, Z. Ding i S. Zhang, ur. Springer International Publishing, 1. 2020., ISBN: 978-981-15-4094-3.

[37] S. Dittert. „Introduction to model-based reinforcement learning," Medium. (17. 8. 2020.), adresa: https://medium.com/analytics-vidhya/introduction-to-model-based-reinforcement-learning-6db0573160da (pogledano 28. 8. 2022.).

[38] Wikipedia contributors. „Monte carlo tree search," Wikipedia. (2022.), adresa: https://en.wikipedia.org/wiki/Monte_Carlo_tree_search (pogledano 1. 9. 2022.).

[39] M. Čupić. „Umjetna inteligencija," Sveučičište u Zagrebu FER. (3. 2020.), adresa: http://java.zemris.fer.hr/nastava/ui/games/games-20200316.pdf (pogledano 1. 9. 2022.).

[40] W. contributors. „Alphazero," Wikipedia. (2022.), adresa: https://en.wikipedia.org/wiki/AlphaZero (pogledano 2. 9. 2022.).

[41] S. Racanière, T. Weber, D. P. Reichert i dr., „Imagination-Augmented Agents for Deep Reinforcement Learning," *Advances in Neural Information Processing Systems*, str. 5691–5702, 7. 2017., ISSN: 10495258. DOI: 10.48550/arxiv.1707.06203.

[42] Unity-Technologies. „Training with proximal policy optimization," Unity-Technologies. (2018.), adresa: https://github.com/miyamotok0105/unity-ml-agents/blob/master/docs/Training-PPO.md (pogledano 1. 9. 2022.).

[43] B. Or. „Value-based methods in deep reinforcement learning," Medium. (1. 2021.), adresa: https://towardsdatascience.com/value-based-methods-in-deep-reinforcement-learning-d40ca1086e1 (pogledano 31. 8. 2022.).

[44] D. Nikolaiev. „Unsupervised learning algorithms cheat sheet," Medium. (18. 4. 2021.), adresa: https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82 (pogledano 29. 8. 2022.).

[45] C. Shyalika. „A beginners guide to q-learning," Medium. (15. 11. 2019.), adresa: https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c (pogledano 29. 8. 2022.).

[46] V. Valkov. „Solving an mdp with q-learning from scratch — deep reinforcement learning for hackers (part 1)," Medium. (10. 12. 2017.), adresa: https://medium.com/@curiousily/solving-an-mdp-with-q-learning-from-scratch-deep-reinforcement-learning-for-hackers-part-1-45d1d360c120 (pogledano 29. 8. 2022.).

[47] Q. Hu i W. Yue, *Markov decision processes with their applications*. New York, NY: Springer International Publishing, 2008., ISBN: 978-0-387-36950-1.

[48] Wikipedia contributors. „Andrej markov," Wikipedia. (2019.), adresa: https://bs.wikipedia.org/wiki/Andrej_Markov (pogledano 31. 8. 2022.).

[49] D. Marinescu, *Cloud Computing: Theory and Practice*, 2. izdanje. Morgan Kaufmann, 12. 2017., sv. 588, ISBN: 9780128128114.

[50] Unity-Technologies. „Unity hub," Unity-Technologies. (2022.), adresa: https://docs.unity3d.com/2020.1/Documentation/Manual/GettingStartedUnityHub.html (pogledano 27. 8. 2022.).

[51] Wikipedia contributors. „Unity (game engine),” Wikipedia. (2022.), adresa: https://en.wikipedia.org/w/index.php?title=Unity_(game_engine)&oldid=1105914892 (pogledano 27. 8. 2022.).

[52] J. Blomberg, R. Jemth, A. Lennar, R. Lilius-Lundmark, M. P. Johnsson i T. Svensson, „An Exploration of Procedural Content Generation for Top-Down Level Design,” 2018.

[53] T. Hussain. „No man’s sky’s 18 quintillion planets take up just 6 gb on disc,” GameSpot. (7. 2016.), adresa: https://www.gamespot.com/articles/no-mans-skys-18-quintillion-planets-take-up-just-6/1100-6441663/ (pogledano 2. 9. 2022.).

[54] G. Schier. „Pros and cons of procedural level generation.” (8. 2015.), adresa: https://schier.co/blog/pros-and-cons-of-procedural-level-generation (pogledano 2. 9. 2022.).

[55] bblakeyyy. „How to make a 2d platformer - unity tutorial crash course,” YouTube. (8. 2021.), adresa: https://www.youtube.com/watch?v=nPigL-dIqgE (pogledano 20. 8. 2022.).

[56] ChronoABI. „Procedural generation in unity 2d for beginner,” YouTube. (5. 2020.), adresa: https://www.youtube.com/watch?v=-5OS1s-NWRw (pogledano 21. 8. 2022.).

[57] Unity-Technologies. „Unity ml-agents toolkit,” Unity-Technologies. (2022.), adresa: https://github.com/Unity-Technologies/ml-agents (pogledano 25. 8. 2022.).

[58] Code Monkey. „How to use machine learning ai in unity! (ml-agents),” YouTube. (11. 2020.), adresa: https://www.youtube.com/watch?v=zPFU30tbyKs (pogledano 28. 8. 2022.).

[59] ——, „Ai learns to play flappy bird!” YouTube. (12. 2020.), adresa: https://www.youtube.com/watch?v=fz8D0OZkQGQ (pogledano 28. 8. 2022.).

[60] O. Lieber. „Reinforcement learning for mobile games,” Medium. (12. 2019.), adresa: https://towardsdatascience.com/reinforcement-learning-for-mobile-games-161a62926f7e (pogledano 30. 8. 2022.).

Popis slika

Popis tablica

Popis isječaka koda

Prilozi

1. Prilog 1

Isječak kôda 6: Prikaz hiperparametara korištenih u treniranju agenta

```
behaviors:
  NoviMozak:
    trainer_type: ppo
    hyperparameters:
      batch_size: 10
      buffer_size: 100
      learning_rate: 3.0e-4
      beta: 5.0e-4
      epsilon: 0.2
      lambd: 0.99
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 128
      num_layers: 2
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    max_steps: 5000000
    time_horizon: 64
    summary_freq: 10000
```

2. Prilog 2

Isječak kôda 7: Prikaz klase za proceduralno generiranje neprijatelja

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

public class ProceduralnoGeneriranje : MonoBehaviour
{
    [SerializeField] private Transform[] trokut;
    public float vrijeme;
    private Manager m;
    int randBr;
    public float randVrijemeNastanak;

    // Start is called before the first frame update
    void Start()
    {
        m = GameObject.FindGameObjectWithTag("Manager").GetComponent<Manager>();
    }

    // Update is called once per frame
    void Update()
    {
        vrijeme += Time.deltaTime;
        if(m.ubrzanjeVremena > 10)
        {
            randVrijemeNastanak = 1;
        }
        else
        {
            randVrijemeNastanak = Random.Range(1, 4);
        }

        if (vrijeme > randVrijemeNastanak)
        {
            vrijeme = 0;
            randBr = Random.Range(0, 7);
            spawnObjekt(trokut[randBr]);
        }
    }

    private void spawnObjekt(Transform obj)
    {
        switch (randBr)
        {
            case 3:
                obj = Instantiate(trokut[randBr], new Vector3(11, (float)-1.7),
                 Quaternion.identity);
                break;
            case 4:
                obj = Instantiate(trokut[randBr], new Vector3(11, (float)-2.32),
```

```
                    Quaternion.identity);
49                  break;
50              case 6:
51                  obj = Instantiate(trokut[randBr], new Vector3(11, (float)-2.75),
                    Quaternion.identity);
52                  break;
53              default:
54                  obj = Instantiate(trokut[randBr], new Vector3(11, (float)-3.24),
                    Quaternion.identity);
55                  break;
56          }
57          obj.transform.parent = this.transform;
58      }
59  }
```

3. Prilog 3

Isječak kôda 8: Prikaz klase korištene za treniranje agenta

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Actuators;
using Unity.MLAgents.Sensors;
using UnityEngine.SceneManagement;

public class NoviAgent : Agent
{
    public float nagrada;
    public float ukupnaNagrada;
    public Animator animacija;
    private Rigidbody2D igrac;
    private bool nepotrebanKorak;
    private bool naPodu;
    public override void OnActionReceived(ActionBuffers actions)
    {
        Debug.Log(actions.DiscreteActions[0]);

        nagrada = Time.deltaTime * 3;
        ukupnaNagrada += nagrada;
        AddReward(nagrada);

        if (actions.DiscreteActions[0] == 0)
        {
            nepotrebanKorak = false;
        }
        if (actions.DiscreteActions[0] == 1 && !nepotrebanKorak && naPodu)
        {
            igrac.AddForce(Vector2.up * 450);
            nepotrebanKorak = true;
            //AddReward(-1f);
        }
        if (actions.DiscreteActions[0] == 2 && !nepotrebanKorak)
        {
            animacija.SetTrigger("Cucanj");
            nepotrebanKorak = true;
            //AddReward(-1f);
        }
        //if(igrac.transform.rotation.z == 0)
        //{
        //    AddReward(0.1f);
        //}
    }

    public override void CollectObservations(VectorSensor sensor)
    {
        sensor.AddObservation(transform.position);
```

```
50          sensor.AddObservation(nepotrebanKorak ? -0.7f : 0.7f);
51      }
52
53      public override void Initialize()
54      {
55          igrac = GetComponent<Rigidbody2D>();
56      }
57
58      private void OnTriggerEnter2D(Collider2D collision)
59      {
60          if (collision.gameObject.CompareTag("Neprijatelj"))
61          {
62              SetReward(-1f);
63              //EndEpisode();
64              SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
65              //SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
66          }
67      }
68
69      private void OnBecameInvisible()
70      {
71          SetReward(-1f);
72          //EndEpisode();
73          SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
74          //SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
75      }
76
77      private void OnCollisionEnter2D(Collision2D collision)
78      {
79          if (collision.gameObject.CompareTag("Pod"))
80          {
81              naPodu = true;
82          }
83          if (collision.gameObject.CompareTag("Skok"))
84          {
85              igrac.AddForce(new Vector2(120, 500));
86          }
87      }
88
89      private void OnCollisionExit2D(Collision2D collision)
90      {
91          if (collision.gameObject.CompareTag("Pod"))
92          {
93              naPodu = false;
94          }
95      }
96  }
```