Tehničko veleučilište u Zagrebu

Preddiplomski stručni studij računarstva

**UMJETNA INTELIGENCIJA U IGRAMA**

Seminarski rad

Kolegij: Metodologija stručnog i istraživačkog rada

Ime i prezime nastavnika: Sara Slamić Tarade, univ.mag.rel.publ., pred.

Ime i prezime studenta: Josip Štanfelj

Zagreb, siječanj 2025.

Sažetak

U ovom radu objasnit ću pojam umjetne inteligencije i na koji način se ona implementira u videoigrama. Istaknut ću videoigre koje su svojim rješenjima postavile nove standarde i na taj način prikazati napredak ove tehnologije kroz povijest. Navest ću jedne od najpoznatijih metoda njene primjene u videoigrama te ih objasniti uz grafičke i programske primjere. Objasnit ću kako svaka pojedina metoda utječe na ponašanje i odlučivanje automatskih likova, kao i na korisničko iskustvo. Ukratko, u radu ću prikazati na koji način umjetna inteligencija omogućuje stvaranje realistične virtualne stvarnosti.

Ključne riječi: umjetna inteligencija, videoigre, automatski likovi, korisničko iskustvo

Sadržaj

Sažetak……………………………………………………………………………………………2

Sadržaj…………………………………………………………………………………………….3

Popis kratica i oznaka…………………………………………………………………………..…4

Popis tablica…………………………………………………………………………………….....5

Popis slika…………………………………………………………………………………………5

[1. UVOD 1](#_Toc189475918)

[2. Ukratko o umjetnoj inteligenciji u igrama 1](#_Toc189475919)

[3. Povijest primjene umjetne inteligencije u video igrama 2](#_Toc189475920)

[3.1. Rani razvoj 2](#_Toc189475921)

[3.2. Suvremena primjeri primjene u 3D igrama 3](#_Toc189475922)

[4. Tehnike umjetne inteligencije u videoigrama 3](#_Toc189475923)

[4.1. Osnovni oblik odlučivanja 3](#_Toc189475924)

[4.2. Stabla odlučivanja 4](#_Toc189475925)

[4.3. Konačni automati 6](#_Toc189475926)

[4.4. Stabla ponašanja 9](#_Toc189475927)

[4.5. Neuronska mreža 11](#_Toc189475928)

[4.6. Neizrazita logika 12](#_Toc189475929)

[4.7. Traženje puta 13](#_Toc189475930)

[4.8. Genetski algoritmi 14](#_Toc189475931)

[5. Zaključak 15](#_Toc189475932)

[6. Literatura 16](#_Toc189475933)

Popis kratica i oznaka

NPC – Non-Player Character

JPS algoritam – Jump point search algoritam

Popis tablica

[Table 1: Primjer tablice stanja 7](#_Toc189432111)

Popis slika

[Slika 1: Osnovni oblik odlučivanja 4](#_Toc189338924)

[Slika 2: Primjer stabla odlučivanja 5](#_Toc189338925)

[Slika 3: Stablo odlučivanja AND operatora 5](#_Toc189338926)

[Slika 4: Stablo odlučivanja OR operatora 5](#_Toc189338927)

[Slika 5: Primjer funkcije u stablu odlučivanja 6](#_Toc189338928)

[Slika 6: Primjer konačnog automata na automatskom igraču 7](#_Toc189338929)

[Slika 7: Implementacija stroja konačnih stanja 8](#_Toc189338930)

[Slika 8: Grafički prikaz selektora 9](#_Toc189338931)

[Slika 9: Grafički prikaz sekvence 9](#_Toc189338932)

[Slika 10: Primjer stabla ponašanja - Ulazak NPC-a u prostoriju 10](#_Toc189338933)

[Slika 11: Implementacija selektora sa slučajnim odabirom 11](#_Toc189338934)

[Slika 12: Grafički prikaz umjetne neuronske mreže 12](#_Toc189338935)

[Slika 13: Grafički prikaz stroja za zaključivanje 13](#_Toc189338936)

[Slika 14: Prikaz mape za navođenje automatskog lika sa svojim čvorovima i vezama 14](#_Toc189338937)

# UVOD

Kao današnja karakteristika mnogih naprednih sustava, umjetna inteligencija predstavlja rješenje za zahtjevne logičke i programske probleme. Iako umjetna inteligencija još uvijek nije dosegla razinu i mogućnosti čovjekova razuma, osigurava osnovne pokazatelje inteligencije za razvoj ciljanih sustava. Omogućuje analizu velike količine podataka u vrlo kratkom vremenu, a na temelju njih nudi učinkovita rješenja. Zbog svoje efikasnosti ima vrlo široku primjenu, a svojim mogućnostima značajno olakšava život ljudi. Bazirajući se na neuronskoj mreži koja omogućuje sustavnu obradu podataka, automatski poboljšava svoje performanse na temelju stečenog iskustva. Zbog svojeg širokog spektra mogućnosti primjenjuje se i u razvoju različitih videoigra. Programeri neprestano nastoje pronaći nove metode kojima će stvoriti inteligentna ponašanja u igricama. Neke od metoda se pokažu neučinkovitima pa se brzo prestaju koristiti, dok se druge nastavljaju unaprjeđivati. Na ovaj način igrice svake godine postaju sve zanimljivije , a njihova radnja sve dinamičnija.

U ovom radu najprije se objašnjava na koji način se umjetna inteligencija primjenjuje u igrama i koje vrste tehnika postoje. Nakon toga, slijedi njezina povijest kroz koju ćemo pratiti njen razvoj i način implementacije u pojedinim videoigrama. Zatim ću navesti najpopularnije metode zajedno s njihovim pozitivnim i negativnim stranama. Svaka od njih bit će detaljno objašnjena uz grafički, a ponekad i programski prikaz.

# Ukratko o umjetnoj inteligenciji u igrama

Umjetna inteligencija u igrama odnosi se na skup algoritama kojom se automatskim virtualnim likovima (eng. NPC – Non-Player Character) pridodaju inteligentne karakteristike. Uz pomoć nje automatski igrači mogu aktivno promatrati svoju okolinu i na temelju toga donositi odluke. Takve algoritme opisuje ciklus osjeti-misli-djeluj. NPC najprije uočava promjene u svojoj okolini , a zatim donosi odluku kako će se ponašati. Na temelju donesene odluke NPC primjenjuje odgovarajuće ponašanje. Takvo se ponašanje omogućuje različitim tehnikama kao što su stabla odlučivanja, strojevi konačnih stanja, stabla ponašanja, neuronske mreže , neizrazita logika i ostale. Tehnike umjetne inteligencije možemo podijeliti na determinističke i nedeterminističke [1]. Kod determinističkih tehnika ponašanja su unaprijed određena , dok kod nedeterminističkih nisu. Zbog toga su nedeterminističke videoigre puno zanimljivije jer ponašanja automatskih igrača postaju nepredvidljiva.

# Povijest primjene umjetne inteligencije u video igrama

## Rani razvoj

Jednu od prvih primjera arkadnih igra s programiranim kretnjama protivnika predstavlja Space Invaders (1978.). U toj videoigri protivnicima je bilo određeno kretanje te postavljeni uvjeti koji će u određenom trenutku zakomplicirati ishod igre. Tada je ponašanje neprijatelja u igrama bilo osnovno te se igra uvijek izvodila na isti način. Razvoj umjetne inteligencije nastavlja se 1979. godine kada izlazi videoigra Pacman. Ona je bila igra s naprednijim oblikom umjetne inteligencije u kojoj su neprijatelji pokazivali značajniji napredak u ponašanju. Neprijatelji bi se kretali u smjeru koji bi igraču otežavao igranje, a time i igricu činio izazovnijom. Oslanjala se na tehniku strojeva konačnih stanja kod koje se stanje mijenja na temelju redoslijeda događaja unutar igre. Upravo su na temelju te tehnike likovi neprijatelja na svakom raskrižju birali put koji bi potencijalno igraču mogao stvoriti probleme. U trenutku kada je igrač mogao uhvatiti i onesposobiti protivnika , svaki protivnik nasumično počinje bježali po istoj navedenoj logici. Narednih godina se u nadolazećim videoigrama ne vidi značajan napredak na području umjetne inteligencije. To se mijenja 1997. godine kada izlazi videoigra Goldeneye 007 koja je likovima dodala simulaciju osjetila [2]. Likovi su bili sposobni pratiti stanja svojih kopija te se na taj način uvjetovalo njihovo ponašanje. Važno je istaknuti i videoigru Creatures koja izlazi iste godine, a danas i dalje predstavlja jedan od složenijih primjera upotrebe umjetne inteligencije u igrama. U njoj svaki lik (Norn) ima sposobnost vlastitog razmišljanja koji se temelji na neuronskoj mreži. Ta je neuronska mreža njima omogućavala razmišljanje i učenje na temelju vaših postupka i brige o njima [3]. Kao igrač morali ste se brinuti o bićima Nornama sve dok ne postanu dovoljno zreli. Ukratko, igrica je predstavljala simulator stvarnog života u kojoj morate uspostaviti stabilnu populaciju bića Norna.

## Suvremena primjeri primjene u 3D igrama

Suvremena primjena umjetne inteligencije u kompleksnijim igricama vrlo je bliska onoj iz 1990-ih. Napredniji oblici njezine uporabe vide se u mnoštvu igara kao što su Counter Strike: Global Offensive , GTA 5, The Last of Us i ostale. U igrici Counter Strike protivnici posjeduju stupanj inteligencije koji im omogućuje planiranje napada i stvaranje taktike kojom nastoje poraziti igrača [4]. Za razliku od nje, likovi u videoigrici “The Last of Us” koriste umjetnu inteligenciju koja im određuje ponašanje na temelju događaja iz okoline. Lovci, posjedujući vidna i audio osjetila, svojom inteligencijom mogu odrediti što predstavlja prijetnju, a što ne. Pronalaskom tijela svojih saveznika se automatski agresivnije ponašaju. S druge strane, Zaraženi lutaju otvorenim svijetom sa smanjenim vidnim i audio osjetilima [5]. Međutim, mogu spoznati kada su u blizini glavnog lika te se na temelju toga kreću prema njemu. Igrica koja je pomaknula granice umjetne inteligencije na još višu razinu je GTA 5 [6]. U njoj su smješteni jedni od automatskih igrača s najprirodnijim ponašanjem i reakcijama. Svaki automatski igrač unutar nje obavlja svoj posao i ponaša se u skladu s tim. Tako je u njoj moguće pronaći likove koji se druže i vode smislene razgovore, vozače koje prate prometne znakove, ali i taksiste koji vas mogu odvesti do željene točke u gradu. Automatski igrači reagiraju na vaše ponašanje, mogu odrediti prijetnju, ali i detektirati i zaobići prepreku kao što su pješaci. Osim toga, policija reagira na vaše prekršaje te smišlja načine kako vas najlakše zaustaviti.

# Tehnike umjetne inteligencije u videoigrama

## Osnovni oblik odlučivanja

Osnovni oblik odlučivanja predstavljaju najjednostavnije if-else naredbe. Sljedeći primjer prikazuje osnovnu logiku prema kojoj će program moći detektirati promjene i donositi odluke (Slika 1).

U ovom slučaju reket će se kretati u stranu prema kojoj se kreće loptica. U samom trenutku kada loptica udari u igračev reket, program će već znati u kojem smjeru se protivnički reket treba kretati [7]. Slika predstavlja programski kod igrice Pong, gdje se protivnički reket ne smije kretati brže od loptice kako bi težina igrice bila adekvatna.

A screen shot of a computer code

AI-generated content may be incorrect.

Slika 1: Osnovni oblik odlučivanja

Izvor: Autor

## Stabla odlučivanja

Stabla odlučivanja predstavljaju prikaze u obliku stabla sastavljenog od čvorova odluke i završnih čvorova. Svaki čvor odluke sadrži uvjet koji ima 2 moguća ishoda, a kroz njih prolazimo kako bismo došli do konačnog stanja. Završni čvorovi nazivaju se i listovima , a predstavljaju konačnu odluku neke situacije. Stablo odlučivanja smatra se jednim od jednostavnijih determinističkih tehnika u donošenju odluka. Karakterizira ih brzina, jednostavna implementacija i jasnoća, a koriste se u stvaranju složenih i taktičkih sustava umjetne inteligencije. Pomoću njih se na jednostavan način opisuju opcije i ishodi te na taj način predviđaju akcije. Pružaju mogućnost nadogradnje tako što se na stablo u bilo kojem trenutku mogu dodati novi slučajevi i ponašanja.

Algoritam kojim prolazi kroz stablo na svakom čvoru donosi odluku i određuje ishod tog čvora. Taj se postupak nastavlja sve dok ne dođe do jednog od listova gdje se izvršava određena naredba. Kada bi čvorovi imali više od dvije grane, struktura bi bila jednostavnija i zahtijevala bi puno manje odluka. Međutim, bilo bi ih puno teže optimizirati. Zbog toga se za ovakvo odlučivanje isključivo koriste stabla s dvije grane. Stablo bi trebalo biti balansirano kako bismo ostvarili najkraće moguće vrijeme pretraživanja unutar stabla. Slika broj 2 prikazuje stablo odlučivanja za pokretanje radnji nad protivnikom. List s istom akcijom može se ponavljati više puta unutar stabla. U našem primjeru to je list s akcijom “Napadni”. Akcija “Napadni” će biti odabrana uvijek osim kada neprijatelj nije vidljiv ili u slučaju kada se nalazi na boku.

A diagram of a diagram

AI-generated content may be incorrect.

Slika 2: Primjer stabla odlučivanja

Izvor : Autor

Stabla odlučivanja primjenjuju se i u određivanju rezultata u izrazima Booleove algebre, ali za njih koristimo serijski oblik testova. Kod izraza s operatorom AND, i lijeva i desna strana izraza moraju biti istinite kako bi se izvršila akcija 1. U suprotnom slučaju izvršava se akcija broj 2. U izrazima s operatorom OR akcija 2 se izvršava samo ako nijedan test nije istinit. Ako je bilo koji od dva izraza istinit, izvršit će se akcija broj 1.

|  |  |
| --- | --- |
| A diagram of a flowchart  Description automatically generated  Slika 3: Stablo odlučivanja AND operatora | A diagram of a decision tree  Description automatically generated  Slika 4: Stablo odlučivanja OR operatora |

Izvor : [2] Izvor : [2]

Kod pisanja programskog koda stabla odlučivanja , odluke i akcije se programiraju kao klase koje nasljeđuju apstraktnu klasu. Apstraktna klasa definira metodu koja se naziva čistom virtualnom funkcijom, a ona će se implementirati u klasama koje ju nasljeđuju. Klasa akcije sadrži pojedinosti o radnji, dok klasa odluke sadrži pokazivače na čvorove koji su povezani s istinitom i lažnom granom. Sljedeća slika prikazuje završnu klasu „DonesiOdluku” koja svojom funkcijom „donesiGranu“ vraća odgovarajuću granu na temelju provedenih provjera.

A computer screen shot of a program code

AI-generated content may be incorrect.

Slika 5: Primjer funkcije u stablu odlučivanja

Izvor : Autor

## Konačni automati

U današnjim igricama automatske igrače opisuje to što provode jedno ponašanje sve do opažanja promjena u okolini. Na primjer, prilikom napada u igrici kao što je FIFA, protivnički će se likovi kretati na svojim uobičajenim pozicijama sve dok im se igrač dovoljno ne približi. Tada oni prelaze u agresivniji način, gdje igraču intenzivno nastoje oduzeti loptu. Iako se za te slučajeve mogu koristiti i stabla odlučivanja, tehnika konačnih automata je jednostavnija pa i zato češće korištena. Dakle, konačni automati, isto kao i stabla odlučivanja, omogućuju automatskim igračima da prelaze iz jednog stanja u drugo. Za razliku od njih, oni se obično prikazuju usmjerenim grafovima. Svaki čvor grafa predstavlja stanje, a svaka usmjerena dužina događaj koji izaziva promjenu stanja. Automatski lik prvo zauzima početno stanje u kojem nastavlja izvoditi pripadajuću radnju. Iz tog stanja može prijeći samo u stanja koja su s njim povezana, a preostala u tom trenutku nisu dostupna.

A diagram of a diagram

AI-generated content may be incorrect.

Slika 6: Primjer konačnog automata na automatskom igraču

Izvor : Autor

U našem slučaju automatski igrač ima početno stanje “Stražari“. Iz tog će stanja, u slučaju kada je igrač detektiran, moći prijeći samo u stanje “Lovi igrača”. U svako iduće stanje prijelazi samo kada je uvjet na prijelazu zadovoljen. Nakon što zauzme stanje “Povuci se”, tek se onda može vratiti u početno stanje “Stražari”. Na svakom je grafu moguće dodati nove uvjete i prijelaze tijekom stvaranja video igrice. Tako je na grafu na slici 6 moguće dodati novu poveznicu koja je usmjerena od stanja “Lovi igrača” do stanja “Stražari”. U ovom slučaju, taj bi se prijelaz aktivirao kada bi igrač otišao predaleko pa ga protivnik više ne bi mogao uloviti.

Osim usmjerenih grafova, konačne automate moguće je prikazati i tablicama stanja.

|  |  |  |
| --- | --- | --- |
| **Trenutno stanje** | **Uvjet** | **Sljedeće stanje** |
| Stražari | Igrač detektiran | Lovi igrača |
| Lovi igrača | Igrač u dometu | Napadni |
| Napadni | Niska razina zdravlja | Povuci se |
| Povuci se | Zdravlje obnovljeno | Stražari |

Table 1: Primjer tablice stanja

Izvor : Autor

Na slici 7 prikazali smo opću implementaciju koda za stvaranje konačnog automata. Konačni automat je obično implementiran kao klasa koja definira objekte mogućih stanja, objekt početnog stanja i objekt trenutnog stanja. Prilikom inicijalizacije, trenutno stanje ima vrijednost početnog stanja koje je dano u konstruktoru objekta. U svakoj iteraciji okvira igrice bit će pokrenuta metoda “ažuriranje”. Ona provjerava sve moguće prijelaze iz trenutnog stanja te, ukoliko je prijelaz pokrenut, pokreće njegovo izvršavanje. Prilikom pokretanja prijelaza dohvaća se stanje u koje lik mora prijeći, a zatim izlazna akcija koju lik mora izvršiti prije izvršenja prijelaza. Ukoliko prijelaz zahtjeva izvršenje određene akcije ona će se isto dodati u akcije koje je potrebno izvršiti. Nakon svih akcija koje se izvršavaju tijekom prijelaza, izvršava se akcija ciljnog stanja, a trenutnom stanju dodana je vrijednost novog stanja.

A screenshot of a computer program

AI-generated content may be incorrect.

Slika 7: Implementacija stroja konačnih stanja

Izvor : Autor

Iako je konačne automate vrlo jednostavno dizajnirati i primijetiti stanja koja nedostaju, kod složenijih implementacija nerijetko se događaju problemi kod dodavanja novih stanja. Veliki broj stanja u prikazu može dovesti do zbunjenosti i otežati implementaciju. Također, kod složenijih izračuna prijelaza kao što je provjera vidljivosti protivnika, izračuni mogu preopteretiti procesor računala što dovodi do dužeg vremena izvršavanja. Stoga, takve se provjere izvode rjeđe kako bi se poboljšale performanse. Tehnika konačnih automata spada pod determinističke jer su ovom tehnikom reakcije NPC-a unaprijed zadane. Ovo predstavlja još jedan nedostatak , a to je predvidljivost. Tako igrač s lakoćom može odrediti buduće ponašanje automatskih likova pa igrica može postati dosadna. Međutim, taj se problem rješava definiranjem nasumičnih prijelaza i različitih varijanti stanja.

## Stabla ponašanja

Stabla ponašanja potkategorija su konačnih automata i funkcioniraju na vrlo sličan način. Za razliku od konačnih automata njima su čvorovi hijerarhijski raspoređeni. Sadrže akcije umjesto stanja, stoga imaju naglasak na aktivnostima koje automatski igrač izvršava u određenom trenutku. Njima je vrlo jednostavno dodati nove čvorove, a obično se koriste u kombinaciji s konačnim automatima. Štoviše, njihovim prikazima omogućujemo izgradnju složenog skupa ponašanja na vrlo jednostavan način. Stabla ponašanja sadrže dvije vrste kompozitnih zadataka sa specifičnim ponašanjem, a to su sekvence i selektori. Obje vrste kompozitnih zadataka sastavljene su od više podzadataka. Funkcioniraju tako da prolaze redoslijedom kroz podzadatke, a kao povratnu vrijednost vračaju uspjeh ili neuspjeh. Sekvenca se izvodi sve dok ne dođe do podzadatka koji završava neuspjehom. Ako sekvenca naiđe na neuspjeli podzadatak, automatski će vratiti neuspjeh. Ukoliko su svi podzadatci uspjeli, sekvenca završava uspjehom. Za razliku od nje, selektor prolazi kroz podzadatke sve dok ne dođe do podzadatka koji završava uspjehom. Ako su svi podzadatci završili neuspjehom, selektor vraća neuspjeh. U suprotnom, ako je bilo koji od podzadataka završio uspjehom , selektor će vratiti uspjeh.

Selektor možemo koristiti u primjeru kada automatski igrač želi ući u neku građevinu. Najprije će pokušati ući kroz vrata. Ukoliko to ne uspije, automatski igrač pokušava ući kroz bočna vrata. U slučaju ponovnog neuspjeha, automatski igrač pokušava ući kroz prozor. Kada bi posljednji ulaz kroz prozor uspio, selektor bi stao s izvođenjem zadataka i vratio uspjeh.

|  |  |
| --- | --- |
| A diagram of a question mark  AI-generated content may be incorrect.  Slika 8: Grafički prikaz selektora. | A diagram of a diagram  AI-generated content may be incorrect.  Slika 9: Grafički prikaz sekvence. |

Izvor : Autor Izvor : Autor

Za sekvencu uzimamo primjer napada na neprijatelja u kojem igrač mora ispuniti više uvjeta kako bi realizirao napad. Ukoliko je neprijatelj u blizini, igrač ima oružje i mjesto za sklonište, onda će se napad realizirati. Ako jedan od slučajeva nije moguće odrediti, tada će se napad na neprijatelja otkazati [8].

A diagram of a building

AI-generated content may be incorrect.

Slika 10: Primjer stabla ponašanja - Ulazak NPC-a u prostoriju

Izvor: [8]

Uporabu sekvenci i selektora možemo vidjeti na stablu na slici 10 gdje izvođenje počinje sa selektorom. Na lijevoj strani stabla imamo sekvencu koja ispituje jesu li vrata širom otvorena. U tom slučaju, NPC će odmah ući u prostoriju. Ako vrata nisu otvorena, lijeva sekvenca vraća neuspjeh i kretanje u prostoriju izostaje. Tada selektor prelazi na sekvencu na desnoj strani stabla gdje se NPC najprije približava vratima. Nakon toga se izvršava središnji selektor treće razine. On pokreće sekvencu na četvrtoj razini koja će otvoriti vrata ukoliko su ona otključana. Ako to nije slučaj, vraćamo se na središnji selektor koji pokušava razbiti vrata. Ako su vrata uspješno razbijena, vraćamo se na sekvencu druge razine koja provodi ulazak NPC-a u prostoriju.

U slučaju kada vrata neće biti širom otvorena, moći ćemo ući u prostoriju pod uvjetima:

1. A) Pomaknuli smo se do vrata , B) Vrata su otključana, C) Otvorili smo vrata
2. A) Pomaknuli smo se do vrata , B) Razvalili smo vrata

Razlikujemo potpuno determinističke, djelomično determinističke i potpuno nedeterminističke tipove stabla ponašanja. Prethodni tip stabla ponašanja je deterministički jer se zadatci odvijaju u strogo definiranom redoslijedu. Iako nisu uvijek poželjni, vrlo često se koriste. Prije ulaska u prostoriju morali bismo otvoriti vrata, a ne obrnuto. Slučaj u kojem prvo ulazimo u prostoriju, a tek onda otvaramo vrata, nema smisla. Determinističko ponašanje unutar sekvenci i selektora nije uvijek poželjno jer se stvara fiksno virtualno okruženje. U tom slučaju moramo koristiti drugačije tipove selektora i sekvenci kako bi igrici dali stupanj nepredvidljivosti. Takvi selektori i sekvence svoje podzadatke izvršavaju slučajnim redoslijedom, a taj se princip koristi u primjerima gdje redoslijed izvođenja nije bitan.

A computer screen shot of a program code

AI-generated content may be incorrect.

Slika 11: Implementacija selektora sa slučajnim odabirom

Izvor : Autor

## Neuronska mreža

Jedan od bitnijih koncepta umjetne inteligencije u igrama jest neuronska mreža. Neuronska mreža sastavljena je od mnoštva umjetnih neurona koji rade po istom principu kao i ljudski mozak. Ona automatskim igračima omogućuje rješavanje problema i učenje na temelju niza primjera. Računalno implementirane neurone nazivamo perceptronima, a svaki od njih ima mnoštvo ulaznih podataka. Svaka ulazna vrijednost množi se težinama koje određuju utjecaj tog ulaza na aktivaciju neurona. Aktivacijska funkcija tada svaku vrijednost zbraja u sumu koja se uspoređuje s pragom koju određuje faktor korelacije. Kada će suma aktivacijske funkcije biti veća od prethodno određenog praga, tada će se neuron vratiti 1. U suprotnom, ako će suma biti manja od praga, neuron će vratiti 0.

Povezivanjem umjetnih neurona u cjelinu dobivamo umjetnu neuronsku mrežu. U takvoj strukturi razlikujemo tri vrste slojeva, a to su ulazni , skriveni i izlazni slojevi [9]. Ulazni sloj neurona prima ulazne signale, skriveni sloj ih prenosi, a izlazni daje zaključnu vrijednost. Kao primjer, željeli bismo naučiti neuronsku mrežu da prepoznaje znak 2. U našem slučaju imamo panel 8x8 pa će neuronska mreža sveukupno imati 64 ulaza. Najprije ćemo nasumično odrediti težine umjetnih neurona, a nakon toga na ulaze unositi različite konfiguracije panela. Za svaki niz ulaza koji ne predstavljaju broj 2 , težine će se prilagođavati da izlaz bude bliži 0. U suprotnom slučaju , težine će se prilagođavati da izlaz bude bliži 1. Na taj se način neuronska mreža trenira kako bi rezultati bili čim bliži očekivanim ishodima.

A diagram of a network

AI-generated content may be incorrect.

Slika 12: Grafički prikaz umjetne neuronske mreže

Izvor : Autor

## Neizrazita logika

Neizrazita logika je tehnika koja dopušta obradu vrijednosti koje mogu biti između istine i laži. Za razliku od tradicionalne logike, omogućuje djelomičnu pripadnost nekom skupu. Tako automatski igrač može biti umoran s vrijednošću 0.4 i gladan s vrijednošću 0.7. Pravilo neizrazite logike govori da zbroj stupnjeva pripadnosti međusobno isključivih skupova mora biti jednak 1. Time je igrač sa stupnjem povrijeđenosti 0.4 istovremeno zdrav s vrijednošću 0.6. Te se vrijednosti nazivaju stupnjevima pripadnosti i određuju u kojoj mjeri neki objekt pripada određenom skupu. Ukoliko je stupanj pripadnosti jednak jedan , tada ćemo reći da određena stvar u potpunosti pripada određenom skupu. Sustav temeljen na neizrazitoj logici ima tri glavna dijela , a to su varijable, pravila i stroj za zaključivanje. Proces kojim se pomoću numeričkih vrijednosti dobivaju stupnjevi pripadnosti naziva se fuzzifikacija [10]. Vrijednosti koje ulaze u stroj najprije se fuzzificiraju funkcijama pripadnosti, a onda prosljeđuju stroju za zaključivanje. Nad tim vrijednostima stroj primjenjuje različita pravila i postepeno oblikuje izlaznu vrijednost. Ta pravila su sastavljena od if naredbi i osnovnih operatora booleove algebre. Svaka se tako dobivena vrijednost može vratiti na prethodnu jasniju vrijednost procesom de-fuzzifikacije.

A diagram of a rule

AI-generated content may be incorrect.

Izvor : [10]

Slika 13: Grafički prikaz stroja za zaključivanje

Neizrazita logika u igrama omogućuje modeliranje automatskih igrača s različitim ponašanjem. Zbog toga su igre koje koriste ovu logiku nedeterminističke, odnosno nepredvidljive. Simulacija ljudskog ponašanja zahtjeva realistično i raznoliko ponašanje, a ova joj metoda baš to omogućuje nasumičnim odabirom stanja.

## Traženje puta

Traženje puta je tehnika kojom se osim pronalaska optimalnog puta određuju i načini kako stići do određenog mjesta u igrici. Razlikujemo slijepa, optimalno usmjerena i heuristička pretraživanja. Poznati algoritam pretraživanje u širinu spada pod slijepa pretraživanja , a radi na principu pretraživanja svojih susjednih čvorova. Pretraga ovim algoritmom može se zamisliti kao valna fronta koja se širi u svim smjerovima sve dok ne nađe traženo mjesto. Većina algoritma koji se koriste za pronalaženje optimalnog puta temeljeni su na A\* algoritmu pretraživanja. Optimizacijom A\* algoritma pretraživanja nastaje JPS algoritam [11]. On radi s dvodimenzionalnom ravninom koja se predstavlja u obliku grafa kojeg čine čvorovi i veze. Svaki čvor predstavlja lokaciju ili određeni objekt na mapi, a svaki pojedini čvor može imati najviše osam susjeda. Čvorovi se međusobno povezuju vezama u slučaju da postoji put kojim igrač do njih može doći. Za razliku od klasičnog A\* algoritma, JPS algoritam uklanja sve nepotrebna čvorove na temelju smjera kretanja. Time se smanjuje broj čvorova koje algoritam mora provjeriti, a zajedno s tim i vrijeme izvršavanja. Svaka veza nosi određeni trošak koji predstavlja vrijeme ili udaljenost od željene lokacije. Ovaj način pretraživanja traži put do željenog mjesta na način da najprije provjeri troškove svih susjednih čvorova. Da bi došao do cilja, algoritam će izabrati put prema onom čvoru koji u tom trenutku ima najmanji trošak. Trošak određenog puta izračunava se po formuli f(x) = g(x) + h(x). Nepoznanica g predstavlja cijenu za dolazak do susjednog čvora, a nepoznanica h heurističku procjenu puta. Kod A\* algoritma pretraživanja heuristička procjena puta predstavlja broj čvorova koje je potrebno prijeći kako bismo došli do ciljane lokacije. Za razliku od njega, JPS algoritam uzima u obzir i uvjete na koje će naići odabirom određenog puta. Zbog toga će formula za heurističku procjenu puta glasiti : .

D(n) – udaljenost od početnog čvora do završnog (n) , u – faktor prohodnosti, Wd - težinski faktor udaljenosti , Wu – težinski faktor prohodnosti

A diagram of a network

AI-generated content may be incorrect.

Slika 14: Prikaz mape za navođenje automatskog lika sa svojim čvorovima i vezama

Izvor : [2]

## Genetski algoritmi

Genetski algoritmi temelje se na Darwinovoj teoriji evolucije, odnosno na ideji prirodne selekcije i genetike. Ovom se metodom pokušava oponašati prirodna selekcija u kojoj se odabiru najbolji primjerci u populaciji za stvaranje potomstva. Svaka jedinka posjeduje jedinstveni fenotip koji predstavlja njene karakteristike. Fenotip će se mijenjati tijekom reprodukcije u populaciji, a prevladat će onaj koji se najbolje pokazao u interakciji s okolinom. Vrste koje su se uspjele prilagoditi određenim promjenama biti će temelj za nastanak budućih generacija. Tako se stvara generacija s najboljim genima koja ujedno i predstavlja najbolje rješenje. Proces počinje odabirom populacije, odnosno odabirom niza primjeraka koje će se uspoređivati. Jedinke koje se ističu u populaciji bit će izabrane na prvom koraku kako bi se smanjilo vrijeme pronalaska optimalnog rješenja. Za svaku drugu jedinku izračunava se vrijednost podobnosti koja predstavlja stupanj sposobnosti te jedinke. Na temelju dobivenih vrijednosti odabiru se jedinke sa najboljim rezultatima koje će križanjem stvoriti nove potomke. Prilikom križanja primjenjuje se binarni operator varijacije koji spaja genetske informacije dviju jedinki [12]. Taj se proces ponavlja sve dok se ne dobije populacija koja nema značajnih razlika od prethodne, a iz nje se tada odabire najbolji kandidat. Tijekom procesa se obično izvršavaju mutacije na genima kojom se dodaju nasumični geni u populaciju. To omogućuje heterogenu strukturu genoma koja će algoritmu omogućiti ispitivanje većeg broja mogućnosti i pronalazak najoptimalnijeg rješenja. Genetski algoritmi idealni su za različite vrste optimizacija i pretraživanja, ali i za kompleksno pronalaženje minimuma i maksimuma.

# Zaključak

Umjetna inteligencija otvorila je vrata novim rješenjima i omogućila pristup problemima na učinkovitiji način. Svaka od njezinih tehnika pruža specifičnu logiku kojom se omogućuju inteligentna ponašanja kod automatskih igrača. Iako neke od njih predstavljaju malo lošija rješenja od ostalih, omogućuju stvaranje vrlo učinkovitih algoritma. Također, jedna tehnika drugoj može biti idealna nadopuna pa se njihovom kombinacijom gube njihovi nedostatci. Prve igrice nisu implementirale ove metode zbog čega su se smatrale statičnim. Pružajući vrlo mali spektar mogućnosti, igrači su s lakoćom mogli predvidjeti ishod svake situacije u igri. Zbog navedenog problema javlja se potreba za podizanjem virtualnog iskustva na više i kompleksnije razine. Videoigra stari čim ona igračima postaje neizazovna i dosadna, a rješenje tome pronađeno je upravo u umjetnoj inteligenciji. Implementiranjem umjetne inteligencije protivnici će prilagođavati svoje ponašanje na temelju igračevih poteza. Automatski igrač više nije pasivan, već donosi različite odluke u različitim situacijama. Time se stvara promjenjivo virtualno okruženje koje će igraču pružiti nepredvidljive izazove. Uz pomoć ove tehnologije, industrija računalnih igara nastoji stvoriti čim bolje i izazovnije korisničko iskustvo te virtualnu stvarnost na jedno potpuno novoj razini.

# Literatura

|  |  |
| --- | --- |
| [1] | D. Stipaničev, L. Šerić i M. Braović, (2021). Uvod u umjetnu inteligenciju, Split: Fakultet elektrotehnike, strojarstva i brodogradnje u Splitu, Dostupno: https://ai.fesb.hr/knjiga/AI-knjiga-FINAL.pdf. |
| [2] | I. Millington i J. Funge,(2006). Artificial Intelligence for Games, Boca Raton, Elsevier, Dostupno: https://theswissbay.ch/pdf/Gentoomen%20Library/Game% 20Development/Programming/Artificial%20Intelligence%20for%20Games.pdf |
| [3] | A. Zucconi, "The AI of Creatures", 27.7.2020. alanzucconi.com. Preuzeto s: https://www.alanzucconi.com/2020/07/27/the-ai-of-creatures/. |
| [4] | J. Balen, "Umjetna inteligencija u igrama", 2.7.2023. cgda.hr. Preuzeto s: https://www.cgda.hr/umjetna-inteligencija-u-igrama-kako-implementirati-pametne-protivnike/. |
| [5] | T. Thompson, "Endure and Survive: the AI of The Last of Us" , 17.6.2020. gamedeveloper.com. Preuzeto s: https://www.gamedeveloper.com/design/ endure-and-survive-the-ai-of-the-last-of-us. |
| [6] | The Evolution of AI in Gaming, bigcloud.global. Preuzeto s: https://bigcloud.global/ the-evolution-of-ai-in-gaming/. |
| [7] | B. Sizer, "The Total Beginner's Guide to Game AI" , 13.8.2018. gamedev.net. Available: https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/. |
| [8] | D. Mount i R. Eastman, (2018). "Lecture 21- Artificial Intelligence for Games: Decision Making", aigamedev.com, Dostupno: https://www.cs.umd.edu/class/spring2018/cmsc425/Lects/lect21-ai-dec-making.pdf. |
| [9] | M. Buckland i A. LaMothe, (2002). AI techniques for game programming, Cincinnati, Ohio: Premier Press, Dostupno: https://archive.org/details/ aitechniquesforg0000buck/page/n3/mode/2up |
| [10] | R. Kovačević, I. Cesar i D. Cafuta, (2019). "UMJETNA INTELIGENCIJA U RAČUNALNIM IGRAMA", *Polytechnic and design,* sve. 7, br. 2, str. 117-124, Dostupno: Hrčak, http://hrcak.srce.hr/. |
| [11] | D. Zhang, Y. Chen, C. Huang i M. Gao, (2016). "Improved Path Planning Algorithm on the Rugged Road", *Automatika,* sve. 57, br. 2, str. 0-0, Dostupno: Hrčak, http://hrcak.srce.hr/. |
| [12] | N. Bosner i T. Droždjek, (2015). "Genetski algoritmi i biomorfi", M*ath.e,* sve. 27, br. 1, str. 1-11, Dostupno: Hrčak, http://hrcak.srce.hr/ |