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

Luka Obuljen

ALGORITMI UMJETNE INTELIGENCIJE U RAČUNALNIM IGRAMA

ZAVRŠNI RAD

Varaždin, 2011.

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

Luka Obuljen
Redoviti student
Broj indeksa: 35986/07-R
Smjer: Informacijski sustavi
Preddiplomski studij

ALGORITMI UMJETNE INTELIGENCIJE U RAČUNALNIM IGRAMA

ZAVRŠNI RAD

Mentor:
Tihomir Orehovački, mag. inf.

Varaždin, rujan 2011

Sadržaj

Popis slika

1. Uvod

Računalne igre danas su jedan od najvećih izvora zabave dostupne svima. Igrači žele da svijet i protivnici u igri budu što realniji, a samim time teži i zabavniji za igranje. U igračkoj se industriji ulažu značajna sredstva za razvoj umjetne inteligencije kako bi se zadovoljile potrebe igrača i postigli stvarniji protivnici. Cilj ovog rada jest pojasniti način na koji funkcioniraju likovi kojima upravlja računalo. Oni u igri mogu biti naši saveznici ili protivnici, ali u oba slučaja o njima ovisi koliko će igra biti uzbudljiva, nepredvidljiva i teška za igranje.

Mnogo je definicija umjetne inteligencije. Jedna od njih kaže kako umjetna inteligencija omogućava računalu da izvršava zadatke koji zahtijevaju misaoni proces, kao kod ljudi ili životinja [I. Millington, J. Funge, 2009, str. 29.]. Druga definicija opisuje umjetnu inteligenciju kao omogućavanje računalima da misle kao čovjek ili da donose racionalne odluke kao on [S. Russel, P. Norvig, 2009, str. 20].

Pojavom računalnih igara, započeo je i razvoj umjetne inteligencije jer su igrači htjeli da im računalo kao protivnik bude što sličniji stvarnom igraču. Od primitivnih početaka u igrama u kojima su se likovi mogli samo kretati pa sve do danas, kada postoje igre u kojima se izvršava nekoliko naprednih algoritama.

Svrha rada jest prikazati osnovne algoritme umjetne inteligencije koji se koriste u računalnim igrama te njihovo korištenje i primjenu u različitim tipovima igara. Uz opis algoritma prikazan je njegov dijagram toka te je naveden primjer njegovog korištenja u današnjim igrama.

Umjetna inteligencija u igrama može se podijeliti na nekoliko dijelova: kretanje, traženje puta, donošenje odluka te strateško razmišljanje. Svi navedeni dijelovi rade zajedno i omogućavaju računalu da upravlja likovima kao što bi to čovjek činio. Kretanje i traženje puta omogućava likovima da dođu od jedne točke na razini do druge. Donošenjem odluka određuje se koje će radnje likovi izvršavati. Kod strategije i taktike likovi iskorištavaju dodatne informacije o okolišu, međusobno usklađuju svoje odluke te zajednički djeluju. Rad opisuje svaki od ovih dijelova te prikazuje algoritme vezane uz njega.

Danas umjetna inteligencija postoji u svim tipovima igara. Najviše je razvijana u pucačinama (eng. shooters), strategijama u realnom vremenu, strategijama na potez te sportskim igrama. U posljednjem je poglavlju prikazano kako su algoritmi umjetne inteligencije implementirani u pojedinim žanrovima računalnih igara.

2. Umjetna inteligencija u igrama

2.1 Povijest umjetne inteligencije u igrama

Jedna od prvih igara koja je bila javno objavljena 1972. godine jest **Pong** [I. Millington, J. Funge, 2009, str. 32.]. U njoj računalo i igrač kontroliraju reket sprječavajući da im loptica dođe do ruba ekrana. To jednostavno upravljanje reketom gore-dolje na temelju pozicije loptice značilo je početak razvoja umjetne inteligencije u igrama.

Prvi koraci na razvojnom putu javljaju se su se pojavili u stolnoj igri **Backgammon** u kojoj je računalo radilo poteze na temelju trenutnog stanja na ploči [Mott, 2009, str. 1.]. Zatim slijedi igra **Space Invaders** u kojoj se likovi kreću po određenim uzorcima.

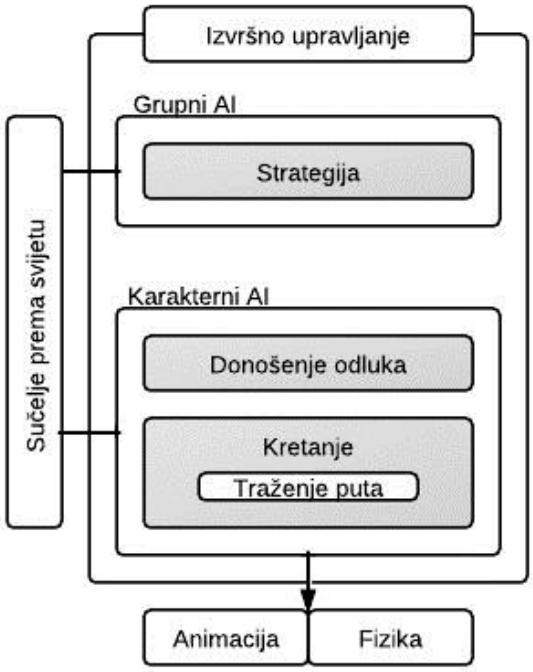
U igri **Pac-man** 1979. već nalazimo složeniju umjetnu inteligencija [I. Millington, J. Funge, 2009, str. 32.]. U njoj je igrač morao izbjegavati neprijatelje. Umjetna inteligencija u toj igri izgrađena je na temelju automata stanja.

U idućih petnaest godina razvoj umjetne inteligencije nije previše napredovao sve do pojave „sneak“ igara poput **Metal Gear Solid** ili **Thief** u kojima se razvio sustav simulacije osjeta. Na primjer, neprijatelj bi znao prepoznati je li njegov suborac živ ili mrtav i po potrebi uključiti alarm. Tada je započelo i intenzivno oblikovanje strategija u realnom vremenu poput **Warcrafta** i serijala **Command & Conquer**. U njima su poboljšani algoritmi traženja puta te strateški i taktički aspekti umjetne inteligencije.

Jedna od najvažnijih igri za razvoj umjetne inteligencije jest **Black and White** iz 2001. godine [Wexler, 2002, str. 17.], koja i danas ima jedan od najboljih i najsloženijih sustava umjetne inteligencije. U njoj se koriste mnogi algoritmi, tako na primjer za interakciju likova sa okolinom koriste se sustavi temeljeni na pravilima dok se odluke donose uz pomoć dinamičkih stabla odlučivanja. Većina igara danas radi na temeljnim načelima u upotrebi prije 30 godina zato što nema potrebe za naprednijima.

2.2 Model umjetne inteligencije u igrama

U igrama postoje mnogi modeli umjetne inteligencije. Jedan od njih je prikazan na slici 2.1. On se sastoji od kretanja, donošenja odluka i strategije. Kretanje i donošenje odluka odnose se na pojedini lik u igri dok se strategija odnosi na skupinu likova. Traženje puta zapravo je planiranje kretanja likova.



Slika 2.1 – Model umjetne inteligencije u igrama
[I. Millington, J. Funge, 2009, str. 34.]

Središte za vođenje likova neprestano dobiva podatke o okolini, to jest svijetu koji ih okružuje i većina algoritama umjetne inteligencije te podatke koristi za djelovanje. Takvim djelovanjem lik može utjecati ili mijenjati svijet oko sebe. Djelovanje u igrama prikazuje se pomoću animacija i fizike koja se koristi u igri.

Ne trebaju sve igre koristiti cijeli model umjetne inteligencije. Na primjer igra **Super Mario** ne koriste strategiju nego se temelje na algoritmima pokreta i donošenja odluka.

3. Kretanje

Kretanje (eng. movement) je prisutno u svim igrama i ono je jedno od najvažnijih dijelova umjetne inteligencije u njima. Već su prve igre poput **Pac Mana** imale razvijen algoritam kretanja. Postoje mnogi algoritmi kretanja, od onih jednostavnih kada se lik kreće od jedne točke do druge pa sve do onih složenih koji uključuju upravljanje ili skakanje.

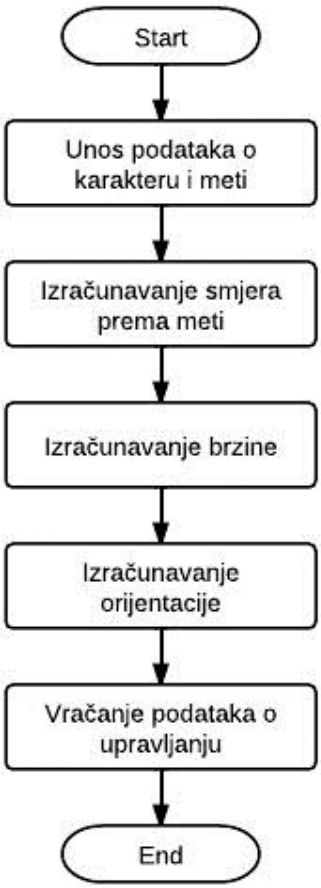
Algoritmi za kretanje rade na načelu uzimanja geometrijskih podatka lika i njegove okoline te izvode geometrijske podatke o kretanju koje lik mora napraviti. Danas se sve više stvaraju igre s dinamičkim upravljanjem koje se odnosi na trenutno kretanje likova.

3.1 Kinematski algoritam kretanja

Kinematski algoritam kretanja (eng. kinematic movement algorithm) koristi poziciju i orijentaciju te računa željenu brzinu. Kod ovog algoritma nema ubrzanja iako se tako ne čini. Algoritam pokreće lik punom brzinom ili ga zaustavlja. Postoje dva osnovna kinematska algoritma kretanja. Algoritam traženja i algoritam lutanja.

Algoritam traženja uzima poziciju od lika i podatke o meti koju traži. Zatim se računa smjer i dobiva se linija između njega i mete. Nakon toga se izračunava brzina kojom će se kretati po toj liniji. Dijagram toka algoritma za traženje je prikazan na slici 3.1.

Kod lutanja (eng. wandering) lik se uvijek kreće naprijed u smjeru prema kojem je trenutno usmjeren. Brzina kojom se kreće je maksimalna. Upravljanje mijenja usmjerenje lika i tako mu omogućava da ne luta samo u jednom smjeru.



Slika 3.1 – Dijagram toka algoritma traženja

3.2 Upravljanje

Upravljanje (eng. steering) daje kretanju brzinu i rotaciju. Mnogo je različitih načina upravljanja poput traženja, bježanja ili zaobilaženja prepreka.

Podudaranje varijabli (eng. variable matching) najčešći je način upravljanja u kojem se podaci o kretanju lika pokušavaju podudarati s podacima kretanja mete. Lik će se kretati sve dok se varijable ne budu podudarale. Najčešće se gledaju podaci o položaju ili smjeru. Ako se gleda smjer, lik će rotirati sve dok se smjer i meta ne podudaraju.

Kod hvatanja mete koja se kreće, to jest bježi, nije dovoljno znati samo njezin trenutni položaj jer se on cijelo vrijeme mijenja. Umjesto ciljanja njezina trenutnog položaja, potrebno je predvidjeti joj i putanju. Tome mogu poslužiti razni dodatni algoritmi predviđanja. Osim

hvatanja tu je i traženje puta koje je slično traženju mete samo se umjesto mete gleda cijeli ciljani put.

3.3 Skakanje

Skakanje je jedan od temeljnih dijelova igri, pogotovo pucačina. Uobičajeni algoritmi kretanja ne sadrže mogućnost skakanja.

Problem sa skakanjem jest u tome da kontrola za upravljanje mora točno odrediti brzinu, točan smjer i trenutak za skok. Na razinama se mogu označiti točke skakanja te brzina kojom se lik treba kretati da bi skočio. Problem s točkama je u tome da nema informacije potrebne za sve mogućnosti skakanja. Tako se može dogoditi da lik dođe iz krivog kuta te skoči na pogrešno mjesto i slično. Točke skakanja i informacije koje one pružaju određuje dizajner razine ili igre.

4. Traženje puta

Danas gotovo sve igre nude velika područja po kojima se likovi mogu kretati. Da bi igrač imao osjećaj da se likovi kojima upravlja umjetna inteligencija slobodno i nepredvidljivo kreću, potrebni su algoritmi koji će im omogućiti kretanje po nekoj karti ili razini.

Kod kretanja postoji početna i završna točka, to jest točka do koje lik mora doći. Karte ili razine u igrama mogu se prikazati pomoću grafa na kojem su prikazani čvorovi i njihove međusobne veze. Lik se može kretati preko međusobno povezanih čvorova. Cilj algoritama za traženje puta (eng. pathfinding) jest da nađu put, to jest čvorove uz pomoć kojih će lik što brže doći od početnog do završnog čvora.

Traženje puta najčešće se odvija prije samog početka kretanja i njime se planira put kretanja likova u igri.

4.1 Dijkstra algoritam

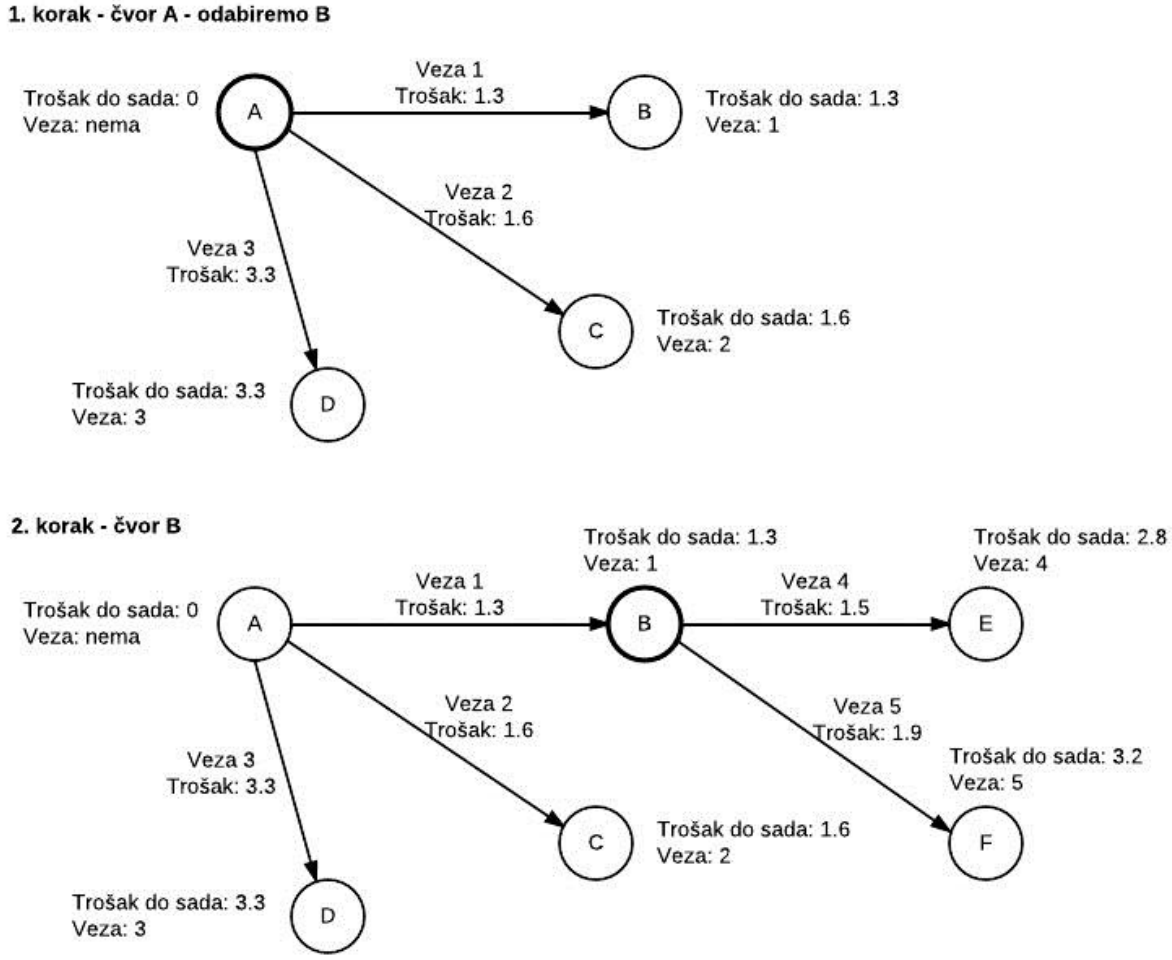
Dijkstra algoritam dobio je naziv po svom autoru Edsgeru Dijkstri [I. Millington, J. Funge, 2009, str. 204.] i on služi za traženje najkraćeg puta. Traženje puta podrazumijeva početnu i završnu točku, a u igrama je najčešće potrebno pronaći najkraći put između te dvije točke. Problem s ovim algoritmom jest da je prilično neučinkovit te zanemaruje sve putove rute osim najkraće između početne i završne točke.

Zbog toga Dijkstra algoritam najčešće služi za traženje puta na samom početku da se, na primjer, dobije pregled i analiza karte i tako omogući donošenje nekih taktičkih odluka. Za traženje puta tijekom igre najčešće se koriste neke naprednije verzije ovog algoritma.

Kod traženja puta imamo grafikon na kojemu su prikazani čvorovi te međusobne veze između čvorova koje također prikazuju koliko je vremena potrebno da prođemo između dva čvora.

Dijkstra algoritam radi na načelu petlje koja se događa u svakom pojedinom čvoru. Gledaju se veze kojima je taj čvor povezan s drugim čvorovima, trošak tih veza te ukupan trošak do pojedinog čvora. Trošak do pojedinog čvora računa se kao trošak veze do tog čvora i trošak koji je potreban do trenutnog čvora koji se promatra. Odabire se čvor koji ima najmanji ukupan trošak do njega te se prelazi na njega. Kao primjer imamo sliku 4.1 koja prikazuje početni čvor A

i nekoliko drugih međusobno povezanih čvorova, te njihov ukupan trenutni trošak i trošak veza između njih. U prvom koraku odabire se čvor B koji ima ukupni trošak 1.3. Zatim slijedi korak do čvora B i njegove veze F i E.



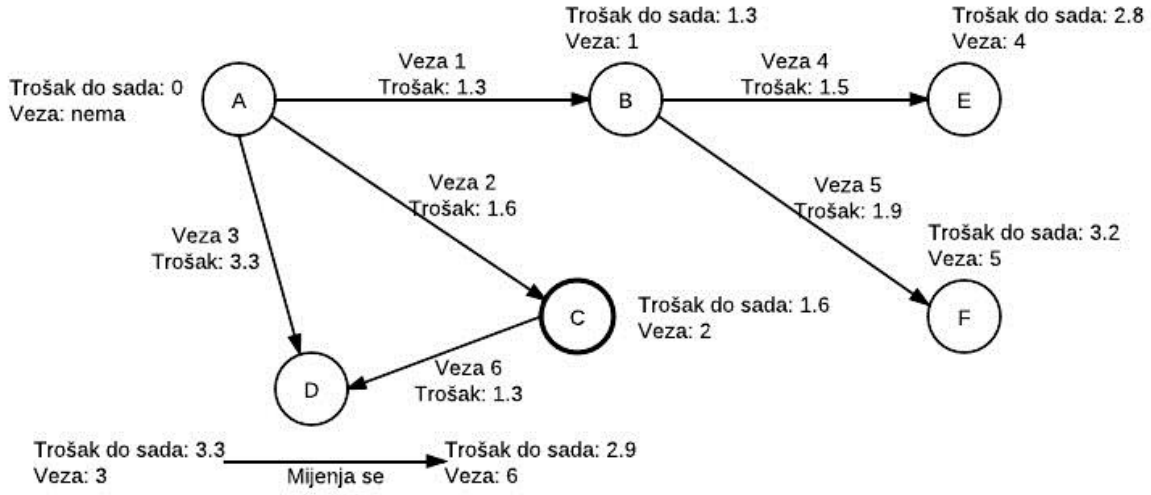
Slika 4.1 - Koraci u Dijkstra algoritmu

[I. Millington, J. Funge, 2009, str. 206.]

Da bi algoritam radio moraju postojati dvije liste čvorova. Prva je otvorena lista na kojoj su zapisani svi čvorovi koje algoritam poznaje, ali još na njima nije radio iteracije. Druga je zatvorena lista na kojoj se nalaze svi dotad obrađeni čvorovi. U svakoj iteraciji algoritam uzima čvor iz otvorene liste čija je vrijednost ukupnog troška najmanja. Nakon obrade, taj čvor prelazi iz otvorene liste u zatvorenu.

Tijekom iteracija algoritam može doći do čvorova koji su već bili otkriveni, to jest, ima pregled veza tog čvora i njegov ukupni trošak, ali taj čvor nije ni na listi otvorenih ni zatvorenih čvorova. U tom slučaju razmotrimo je li novi put kojim smo došli do njega bolji, to jest ima li

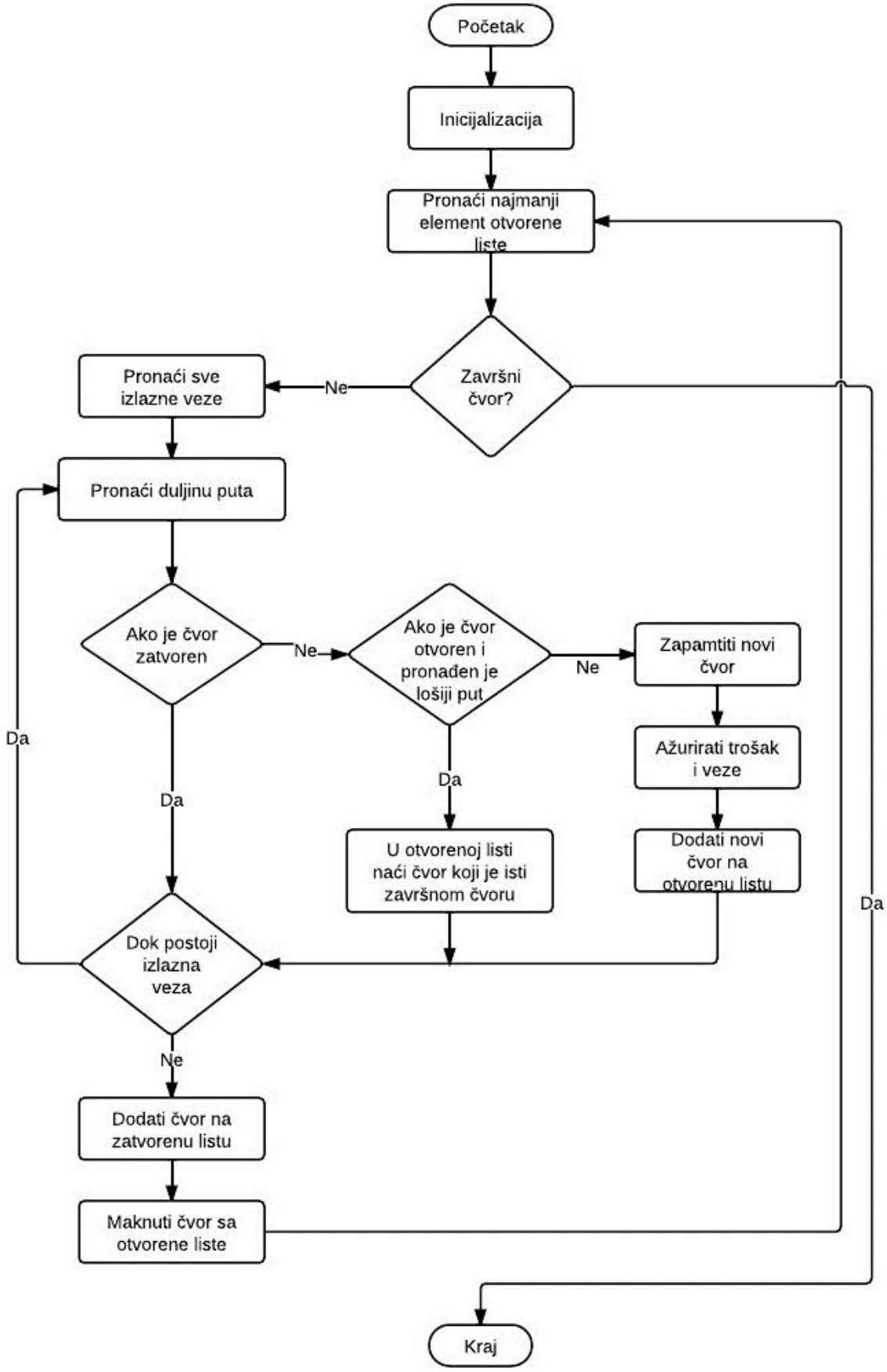
manji ukupan trošak nego je ukupan trošak na njemu zapisan. Ako je novi ukupni trošak bolji, njega zapisujemo i čvor stavimo na otvorenu listu. Takav je slučaj prikazan na slici 4.2.



Slika 4.2 - Prikaz odabira puta sa manjim ukupnim troškom

[I. Millington, J. Funge, 2009, str. 207.]

Algoritam završava kada je otvorena lista čvorova prazna, to jest kada ciljani čvor bude najmanji čvor na otvorenoj listi. Problem kod Dijkstra algoritma izaziva činjenica da se može dogoditi da neki čvorovi ne budu ni otkriveni. Prikaz dijagrama toka Dijkstra algoritma je na slici 4.3.



Slika 4.3 – Dijagram toka Dijkstra algoritma

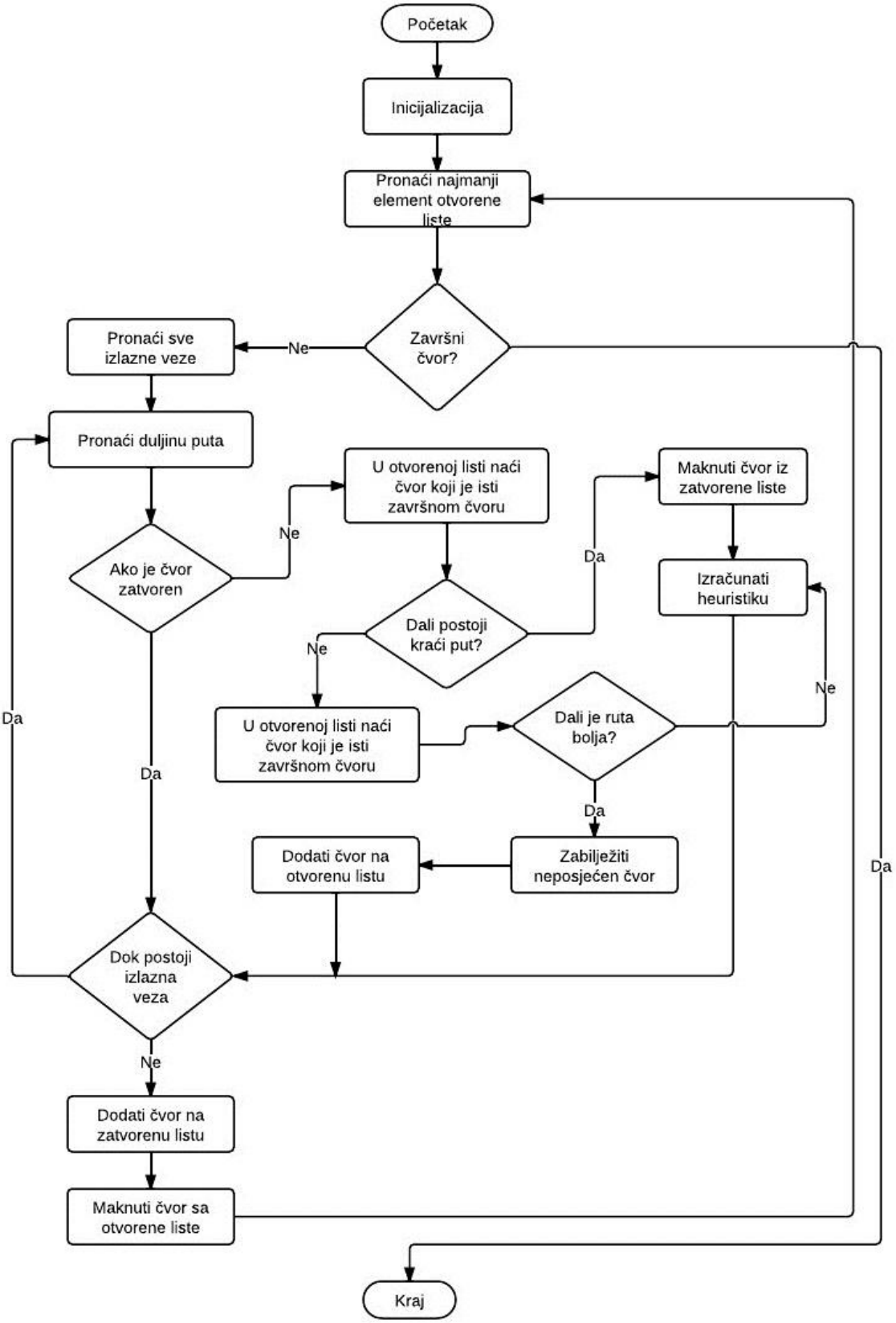
4.2 A*

Za traženje puta najčešće se upotrebljava A* algoritam. On je učinkovit, jednostavno se provodi i ima mnogo varijanti. A* algoritam sličan je Dijkstra algoritmu, a glavna je razlika u tome da Dijkstra algoritam uvijek bira čvor iz otvorene liste s najmanjim troškom, dok heuristika A* algoritma bira čvor koji će najvjerojatnije biti dio najkraćeg puta do cilja. O heuristici ovisi učinkovitost algoritma. Ako A* dobije pogrešnu procjenu traženje puta može trajati duže. Ona je najčešće implementirana izvan A* algoritma.

A* djeluje u iteracijama u kojima se obrađuje određeni čvor. Promatraju se veze, to jest čvorovi s kojima je trenutni čvor povezan, računa se dotadanji trošak, te se on i broj veze zapisuju u te čvorove. Uz trošak se izračunava još jedna vrijednost, procijenjena ukupna vrijednost puta od početka do trenutnog čvora, te do krajnjega, ciljanog čvora. Procijenjenu ukupnu vrijednost izračunava drugi kod, koji nije dio A* algoritma.

Kao i kod Dijkstra algoritma, postoji otvorena i zatvorena lista čvorova. Na otvorenoj se listi nalaze neobrađeni čvorovi za koje algoritam zna da postoje, a na zatvorenoj čvorovi koji su obrađeni svojom iteracijom. Iz liste se biraju za obradu oni čvorovi čiji je procijenjeni trošak najmanji. Na taj način algoritam odabire čvor koji je najvjerojatniji dio najkraćeg puta i takvim se pretraživanjem usmjerava na dio koji najviše obećava da će biti dio najkraćeg puta.

Algoritam se u većini implementacija prekida kad je ciljani čvor najmanji na otvorenoj listi, no to nije nužno i najkraći put između početnoga i završnog čvora. Kako bi se pretražili i drugi, još možda neotkriveni čvorovi, često se algoritmu daje više vremena za otkrivanje. Dijagram toka A* algoritma je prikazan na slici 4.4.



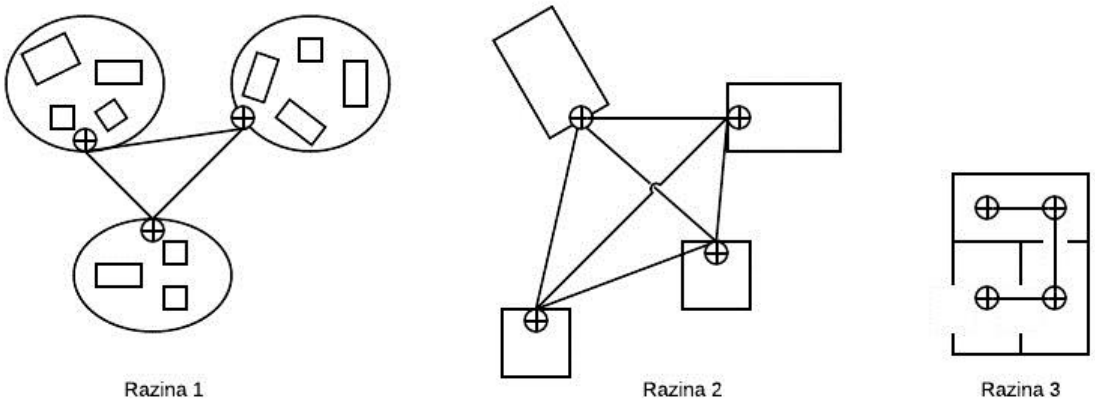
Slika 4.4– Dijagram toka A* algoritma

4.3 Hijerarhijsko traženje puta

Problemi pri traženju puta nastaju kada treba isplanirati put na velikim područjima gdje su algoritmi poput Dijkstre ili A* prespori i zahtijevaju previše računalnih resursa. Primjer takvih planiranja su **MMRPG** igre (eng. Massively multiplayer online role-playing game) u kojima lik treba prijeći veliku udaljenost između dva grada ili kod strateških igara u kojima cijele vojske moraju imati isplanirani put kroz bojišnicu.

Da bi se riješio taj problem učinkovitosti postoji dinamički način traženja puta, to jest, hijerarhijsko traženje puta (eng. hierarchical pathfinding). Kod hijerarhijskog traženja puta prvo se planira apstraktna ruta, zatim se poduzima prvi korak plana, ponovno pronađe ruta do kraja i tako sve dok se lik ne počne kretati. Nakon početnog planiranja, kod svakog koraka detaljno planiramo samo sljedeći korak na putu. Planiranje jednog djela puta je jednostavno i zato cijelu rutu dijelimo na manje dijelove i planiramo dio po dio. Kod pojedinačnih dijelova puta i početnog pretraživanja puta može se koristiti bilo koji algoritam za traženje puta, poput A*.

Primjer grafa traženja puta je na slici 4.5 na kojoj su prikazane grupe čvorova. Najviša razina predstavlja naselja, niža razina predstavlja građevine u tim naseljima, dok u najnižoj razini imamo prikazane čvorove unutar pojedinih zgrada.

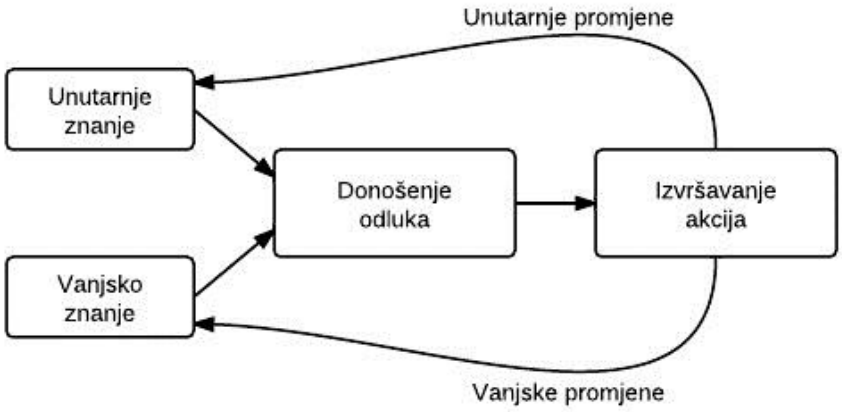


Slika 4.5 – Razine i čvorovi kod hijerarhijskog traženja puta

5. Donošenje odluka

Donošenje odluka (eng. decision making) omogućava likovima kojima računalo upravlja da sami određuju svoje djelovanje. Jednako tako omogućava rad i drugih dijelova umjetne inteligencije kao što je kretanje ili strateško razmišljanje. Danas se većinom koriste jednostavna odlučivanja uz pomoć stabla odlučivanja, automata stanja ili sustava temeljnog na pravilima.

Donesene se odluke većinom temelje na nekim vanjskim ili unutarnjim podacima, a nakon donošenja odluke obavljaju se akcije koje te podatke mogu mijenjati, kao što je prikazano na slici 5.1.



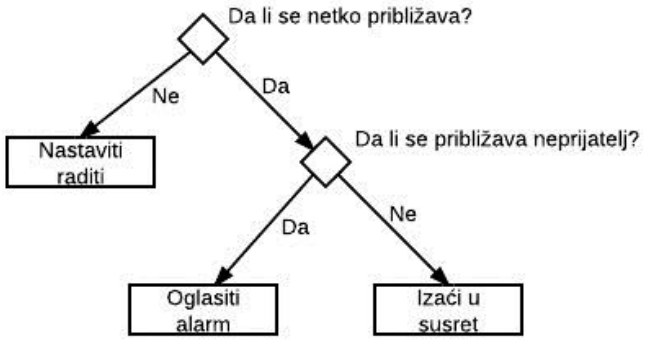
Slika 5.1 – Donošenje odluka

[I. Millington, J. Funge, 2009, str. 294.]

5.1 Stabla odlučivanja

Postoji nekoliko algoritama koji omogućavaju donošenje odluka. Najlakši i najbrži algoritam je algoritam stabla odlučivanja (eng. decision tree). U današnjim računalnim igrama stabla odlučivanja rijetko se koriste, ali ona su veoma jednostavna i laka za razumijevanje. Međutim, uz dodatne modifikacije algoritam može postati dosta složen i učinkovit.

Stabla odlučivanja posjeduju određena znanja te moguće akcije od kojih se odabiru potrebne. Sastoje se od povezanih točaka odluka. Svaka točka nudi odabir od nekoliko mogućih odluka. Početna točka stabla se naziva korijen. Primjer stabla odlučivanja prikazan je na slici 5.2.

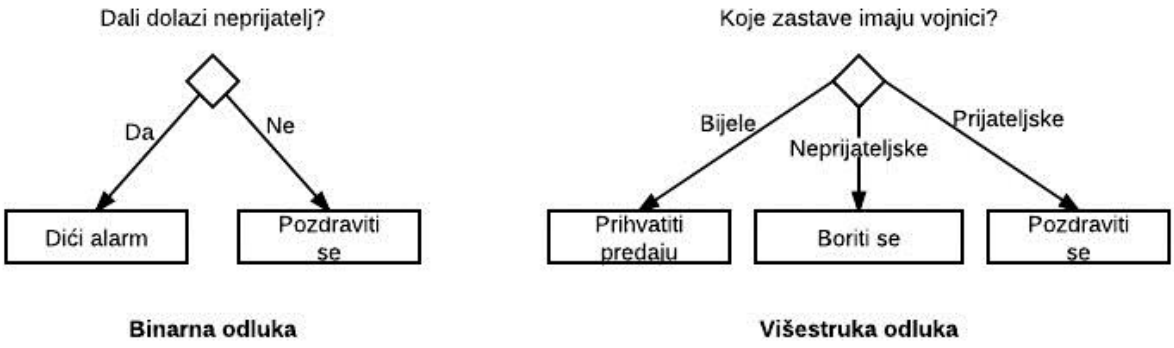


Slika 5.2 - Primjer stabla odlučivanja

Svaka se donijeta odluka temelji na znanju dostupnom liku koji donosi odluke. Algoritam prolazi kroz stablo sve dok ima odluka na raspolaganju. Na kraju algoritma obavljamo određenu akciju koja se nalazi na vrhu grane stabla do kojeg smo došli odlukama. Podebljane crte na slici predstavljaju odluke koje su donesene.

Stabla odlučivanja veoma su učinkovita jer su odluke koje se donose najčešće jako jednostavne. Isto tako, do istih je odluka moguće doći na više različitih načina.

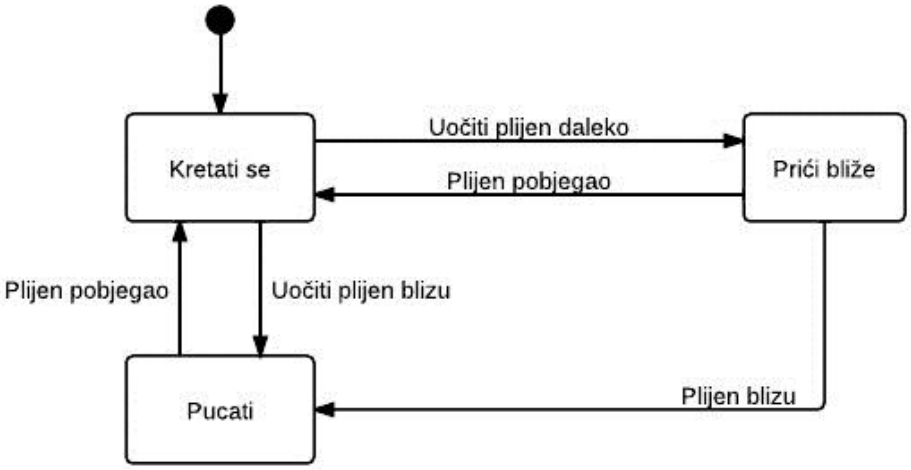
Moguće odluke mogu biti binarne, to jest imati dvije mogućnosti ili mogu biti višestruke i imati više mogućnosti. Primjer binarnih i višestrukih odluka je prikazan na slici 5.3.



Slika 5.3 – Primjer binarnih i višestrukih odluka

5.2 Automati stanja

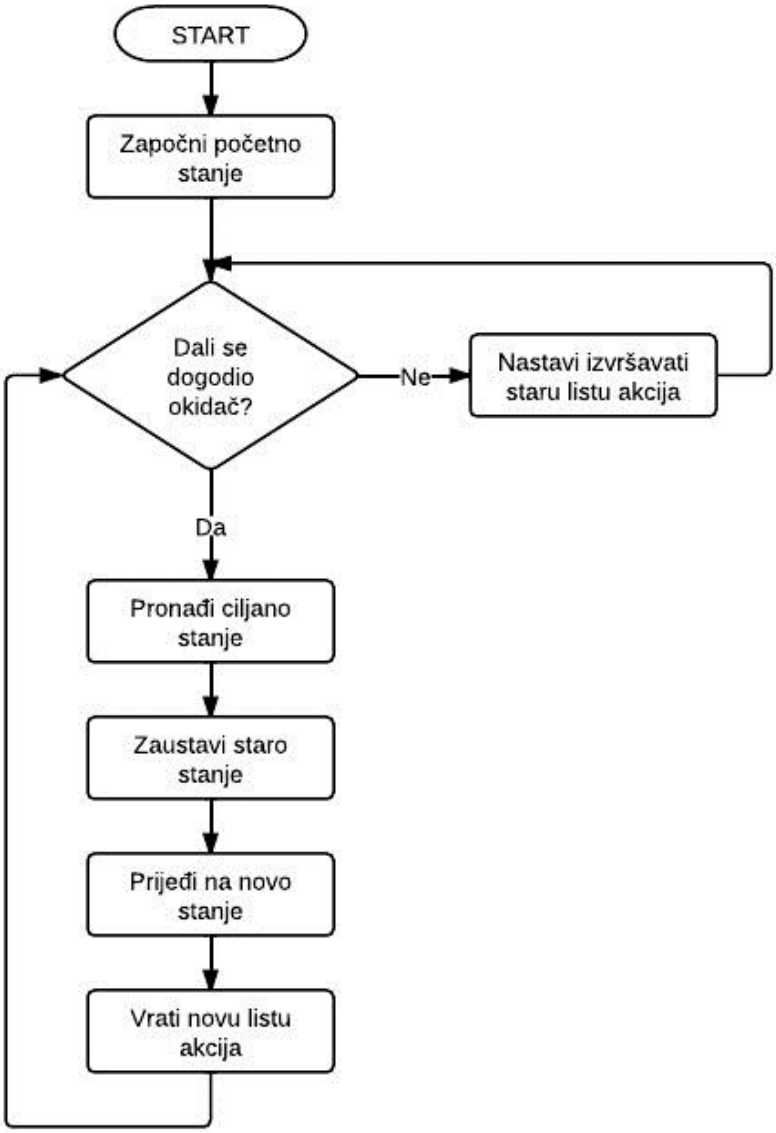
Uz stablo odlučivanja, automati stanja (eng. state machine) najčešće su korištene tehnike odlučivanja u računalnim igrama. Svaki lik u igrama ima neko stanje i neko djelovanje koje u tom stanju izvodi. Također se lik može nalaziti u nekoliko mogućih stanja i ta su stanja međusobno povezana prijelazima koji imaju svoje uvjete. Primjer prikaza stanja i njihovih veza je na slici 5.4. Lovac tijekom lova ima tri stanja u kojima se može biti i tri akcije koje može obavljati: kretati se, pucati i prići bliže. Ako se tijekom igre ostvare uvjeti prijelaza tada lik prelazi iz jednog stanja u drugo, te započinje s djelovanjem u novom stanju.



Slika 5.4 - Primjer automata stanja za lovca u lovu

U igrama postoji struktura stanja, to jest povezanost stanja i uvjeta koji se moraju ispuniti da bi se ostvario neki prijelaz iz jednog stanja u drugo. Algoritam može zauzeti samo jedno stanje i izvršavati ga. U svakoj iteraciji algoritam provjerava je li se neki prijelaz stanja dogodio.

Sam algoritam se sastoji od popisa stanja na kojemu je označeno aktivno stanje, te funkcija izvršavanja prijelaza stanja i funkcije dohvaćanja akcija za određeno stanje. Dijagram toka automata stanja je prikazan na slici 5.5.



Slika 5.5 – Dijagram toka automati stanja

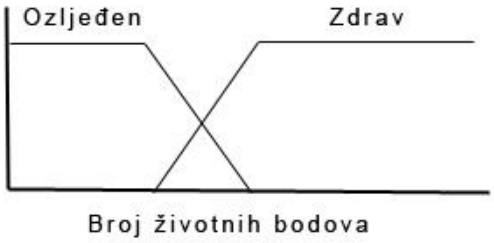
5.3 Neizrazita logika

Algoritmi odlučivanja kao što su stabla odlučivanja ili automati stanja imali su jasne moguće ishode i odluke koje se donose na načelu istine ili laži. Kod donošenja odluka postoje i one odluke koje su negdje između dvije krajnosti, u takozvanoj sivoj zoni. Neizrazita logika (eng. fuzzy logic) je napravljena tako da pokrije odluke koje se donose u njoj.

Neizrazita logika radi na načelu da osim predikata koje dajemo likovima u igri, dodjeljujemo im i određenu vrijednost. Na primjer, broj životnih bodova u pucačinama. Raspon vrijednosti koje se mogu dodati predikatima zove se neizraziti set. Vrijednost koja se dodjeljuje predikatima može predstavljati i stupanj članstva u pojedinom setu.

Pojedini objekti mogu biti unutar više setova, primjerice lik može biti ozlijeđen u igri i u isto vrijeme imati jak štit. U tradicionalnoj logici se dva ili više predikata mogu isključivati, kao na primjer lik ne može u isto vrijeme biti ozlijeđen i zdrav, dok je u neizrazitoj logici to moguće, to jest može biti ozlijeđen ili zdrav.

Kod neizrazite logike ulazne vrijednosti u algoritam najčešće se pretvaraju u neke setove i odrediti stupanj članstva u pojedinim setovima. Izlaz iz algoritma iz oblika koji se nalazi u setovima mora se pretvoriti u neku jednostavnu, najčešće brojčanu vrijednost kako bi se mogla kasnije koristiti. Primjer mapiranog seta se nalazi na slici 5.6.



Slika 5.6 – Mapirani set u neizrazitoj logici

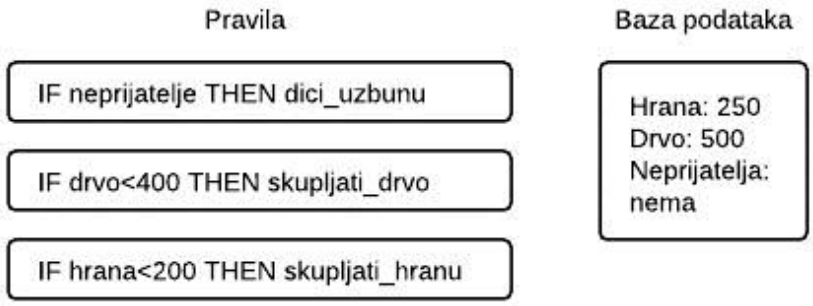
Uz set i članove pojedinog seta imamo i neizrazita pravila. Ona sadrže ulaze međusobno povezane logičkim operatorom AND i izlaze koji određuju koja će se akcija izvoditi.

5.4 Sustavi temeljeni na pravilima

Sustavi temeljeni na pravilima (eng. ruled-based systems) napravljeni su i korišteni na samom početku razvoja računalnih igara i umjetne inteligencije u njima. Koriste se već preko 15 godina iako su dosta neučinkoviti te ih je teško implementirati. U današnje vrijeme zamjenjuje ih jednostavniji algoritmi poput stabla odlučivanja i automati stanja.

Struktura sustava temeljenih na pravilima ima dva dijela: bazu podataka koja sadrži znanje koje lik može primijeniti te skup „if-then“ pravila, to jest pravila koja se primjenjuju ako se dogodi neka promjena [I. Millington, J. Funge, 2009, str. 427].

Pravila provjeravaju bazu znanja da utvrde je li se *if* uvjet ispunio, a ako jest, onda se pokreće okidač akcije koja se događa unutar uvjeta. Sustavi često sadrže i suca koji odlučuje koji će se okidač pokrenuti. Često se u sustavima nalazi i sudac koji odlučuje koji će se okidač izvršiti. Prikaz primjera sustava temeljenog na pravilima nalazi se na slici 5.7.

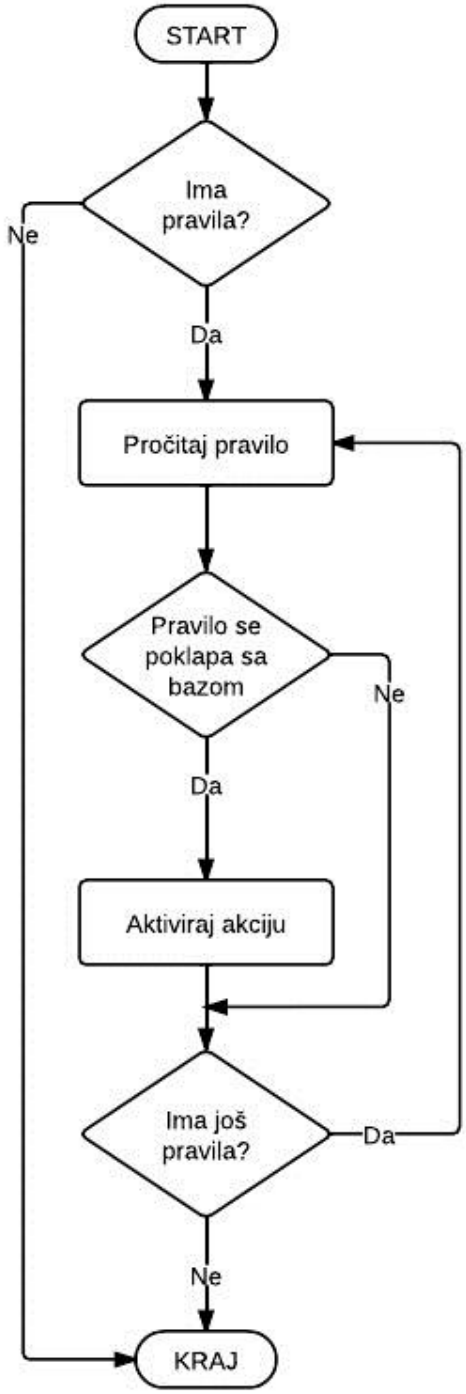


Slika 5.7 - Primjer sustava temeljenog na pravilima

Za uspoređivanje uvjeta najčešće se koriste logički operatori: *and, or, not*. U složenijim uvjetima služimo se i drugim operatorima uspoređivanja poput: veće od *(>)* ili manje od *(<)*.

Algoritam radi na načelu iteracija. Pravila se primjenjuju unutar iteracije te se uzastopno može primijeniti neodređeni broj pravila. Baza podataka u kojoj se nalazi znanje neprestano se može mijenjati ili djelovanjem ili nekim drugim vanjskim kodom.

Sustav neprestano provjerava svako pravilo za bazu podataka i ako se pravilo aktivira, započinje akcija tog pravila. Dijagram toka sustava temeljenog na pravilima prikazan je na slici 5.8.



Slika 5.8 – Dijagram toka sustava temeljenog na pravilima

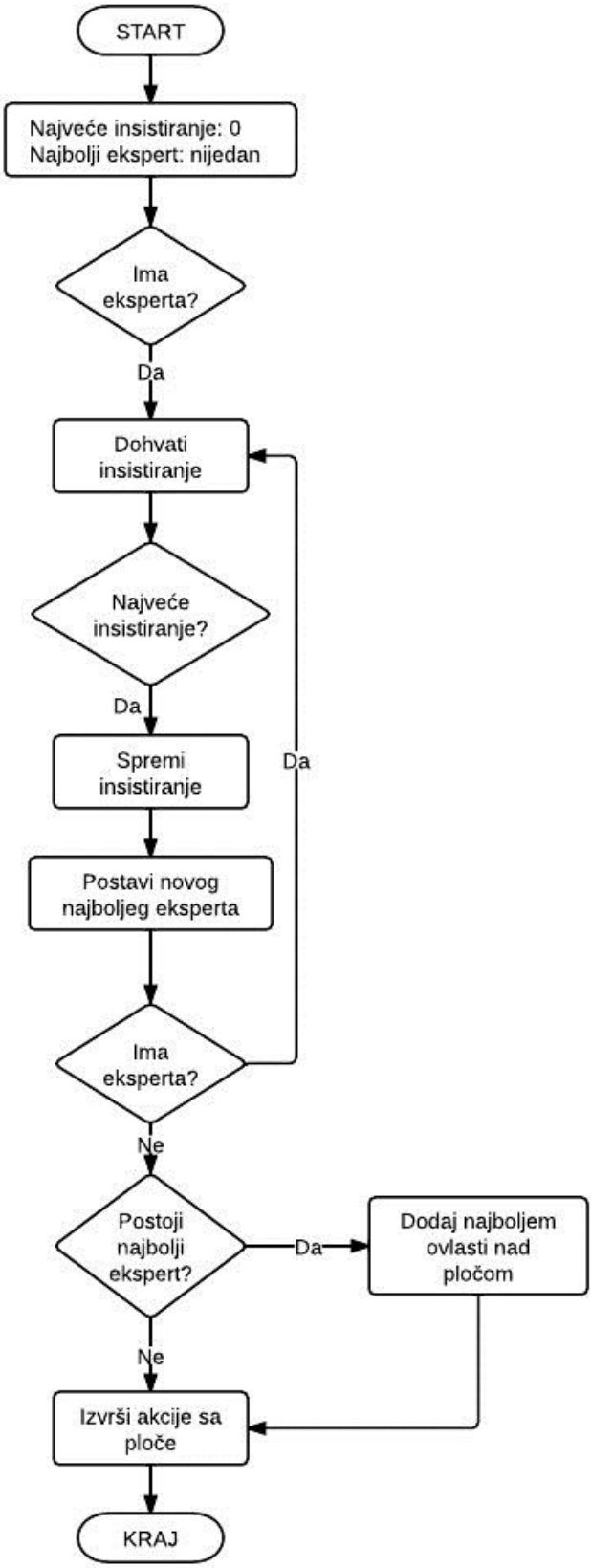
5.5 Arhitektura crne ploče

Arhitektura crne ploče (eng. blackboard architecture) nije algoritam odlučivanja sam po sebi nego služi za koordiniranje djelovanja više sudionika u odlučivanju [I. Millington, J. Funge, 2009, str. 459]. Ona se sastoji od nekoliko dijelova. Prvi dio je skup različitih alata za donošenje odluke koji se nazivaju ekspertima, drugi je dio ploča, a treći sudac.

Crna ploča je mjesto, memorija u koju eksperti zapisuju i čitaju informacije. Svaki ekspert čita informacije s ploče i ako odluči da mu je neka informacija potrebna, traži mogućnost upravljanja pločom. Kada pojedini ekspert dobije tu mogućnost, može obrisati informaciju s ploče i dodati novu. Nakon nekog vremena ekspert mora predati svoje ovlasti i omogućiti upravljanje pločom drugom ekspertu. Na slici 5.9 prikazan je primjer crne ploče i eksperata.

Slika 5.9 - Primjer arhitekture crne ploče

Sudac određuje hoće li neki ekspert dobiti ovlasti ili ne. Najčešće samo jedan ekspert želi dobiti ovlasti tako da sudac nije ni potreban. Djelovanje i informacije mogu se zapisivati na ploču. Nakon što sve iteracije završe, algoritam započinje izvoditi akcije zapisane na ploči. Sudac najčešće daje ovlasti ekspertu koji je najuporniji. Dijagram toka arhitekture crne ploče prikazan je na slici 5.10.



Slika 5.10 – Dijagram toka arhitekture crne ploče

5.6 Ponašanje određeno ciljem

Ponašanje određeno ciljem (eng. goal-oriented behavior) je tehnika određivanja ponašanja lika preko ciljeva ili želja koje želi postići. Svaki lik u igri može imati jedan ili više ciljeva kojima stremi. Cilj može biti: ubiti neprijatelja ili skupiti što više resursa. Svaki je cilj obilježen numeričkom vrijednošću koja se naziva insistiranjem (eng. insistence) ili navaljivanjem. Cilj koji ima najveću vrijednost insistiranja najviše će utjecati na ponašanje lika i na njegove odluke.

U nekim je igrama dopušteno da se ciljevi u potpunosti ispune dok se u nekima oni ispunjavaju samo djelomično, to jest potrebno ih ispunjavati dok im se vrijednost insistiranja ne smanji. Najpoznatiji primjer ponašanja određenog ciljem je igra **The Sims** u kojoj su likovi određeni stupnjem gladi, higijene ili socijalnim druženjem. Igrač će rješavati cilj koji ima manju vrijednost.

Uz ciljeve mora postojati i djelovanje koje možemo obavljati. Djelovanje je najčešće uvjetovano okolinom. Na primjer, ako lik želi jesti u okolini mora biti hrana ili načina da se do nje dođe.

Slika 5.11 – Primjer ciljeva u igri The Sims

[http://img.gamespot.com/gamespot/images/2005/055/reviews/925292_20050225_screen004.jpg]

Odabir između više ciljeva može se riješiti na više načina. Najjednostavniji je odabiranje cilja čija je potreba ili insistiranje najveće. Na slici 5.11 vidi se primjer ponašanja određenog ciljem i vrijednosti pojedinih ciljeva koji se moraju postići. Na slici 5.12 je prikazan dijagram toka ponašanja određenog ciljem.

Slika 5.12 – Dijagram toka ponašanja određenog ciljem

6. Taktička i strateška umjetna inteligencija

6.1 Taktičke točke

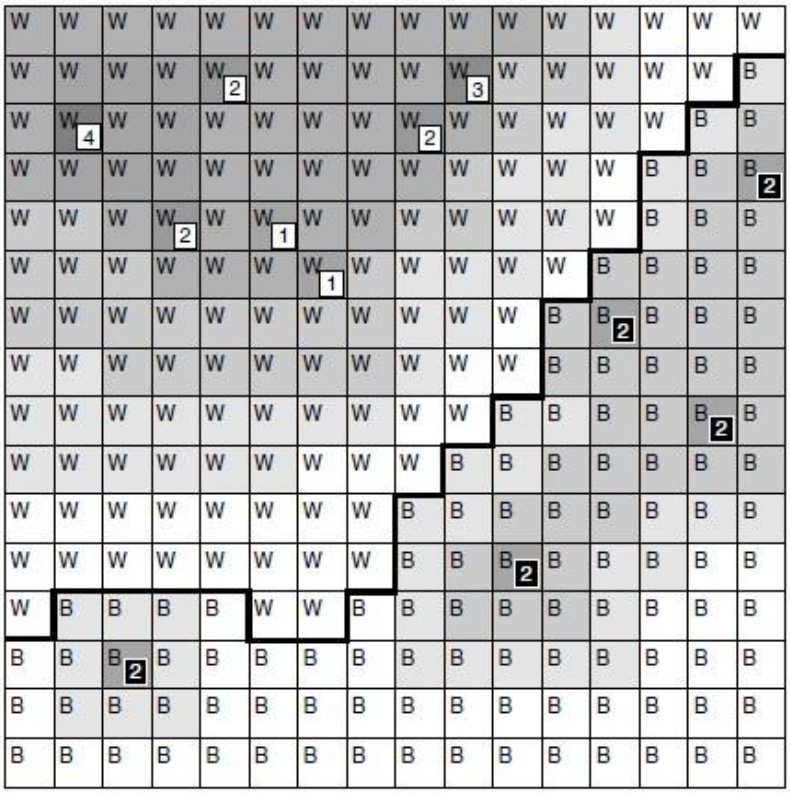
Taktičke točke u igrama slične su čvorovima u grafu traženja puta. Različite su po tome što uz lokaciju, susjedne točke i veze s njima, sadrže i neke taktičke podatke, kao na primjer postoji li zaklon, dobra snajperska pozicija, je li točka u sjeni i slično, što ovisi o vrsti igre. Pojedina taktička točka može imati više taktičkih obilježja.

Taktičke točke se koriste u drugim algoritmima, kao na primjer kod stabla odlučivanja ili neizrazite logike. Algoritam odlučivanja odlučuje koju vrstu taktičke točke treba, zatim se lik, ako je to moguće, kreće do najbliže zadane točke.

6.2 Taktička analiza

Taktička analiza je zapravo analiza razine ili karte, najčešće terena na kojem se razina nalazi. Katkad se za taktičku analizu radi mapa utjecaja koja prikazuje koliko su neki strateški čimbenici prisutni na pojedinim dijelovima mape. Na primjer, tamo gdje protivnika očekuje zid, na karti utjecaja označena je jaka obrana. Mnogi čimbenici mogu utjecati na izgled i sadržaj mapa utjecaja. Primjer mape utjecaja je na slici 6.1.

Igrač kojim upravlja računalo najčešće nema pristup cijeloj karti utjecaja nego samo onim dijelovima koje je istražio, to jest sve u vidokrugu njegovih jedinica.



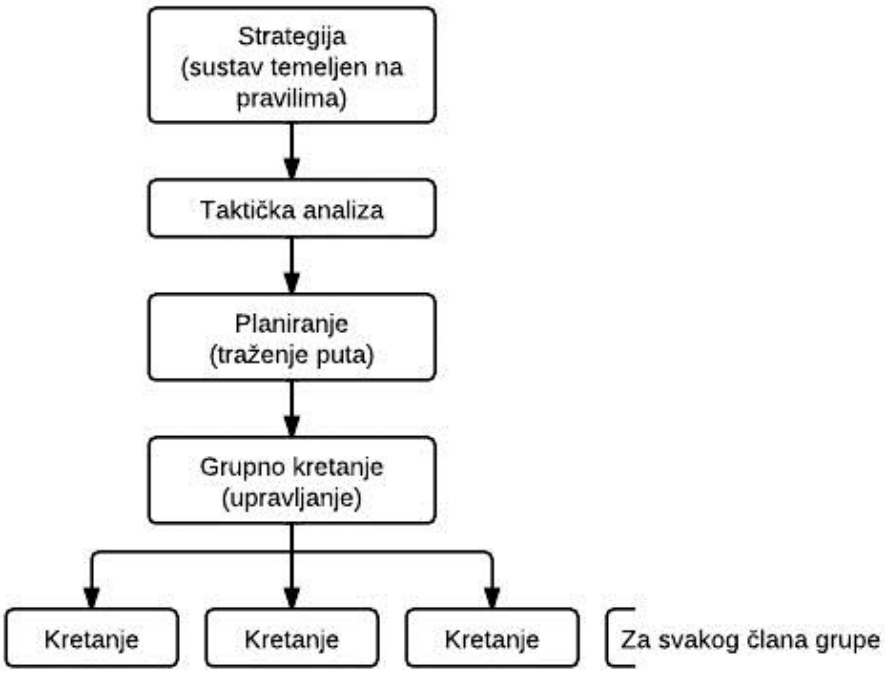
Slika 6.1 – Primjer mape utjecaja

[I. Millington, J. Funge, 2009, str. 546.]

6.3 Koordinirane akcije

Koordinirane akcije služe za usklađivanje radnje koju izvodi više likova. Primjer takvih radnji imamo u strateškim igrama sa skupinama ratnika ili timovima specijalaca u igri **SWAT.** Također, u igri **SWAT** tim s kojim upravlja računalo mora djelovati zajedno s igračem koji igra igru. Tim mora predvidjeti igračeve namjere i uskladiti s njime svoje djelovanje.

Da bi se omogućila koordinacija više likova, koristi se višerazinska umjetna inteligencija. Na najnižoj razini će postojati algoritmi za pojedine likove, a što idemo prema višoj razini to će se algoritmi primjenjivati na više likova. Primjer višerazinske umjetne inteligencije prikazan je na slici 6.2.



Slika 6.2 – Višerazinska umjetna inteligencija

[I. Millington, J. Funge, 2009, str. 585.]

Postoje mnogi pristupi izvršavanja algoritama, ali se najčešće koristi odozgo prema dolje (eng. top–down approach). Najviša razina donosi odluke i šalje upute nižoj razini. Zatim niža razina donosi odluke prema uputama dobivenim odozgo i nove odluke šalje idućoj nižoj razini.

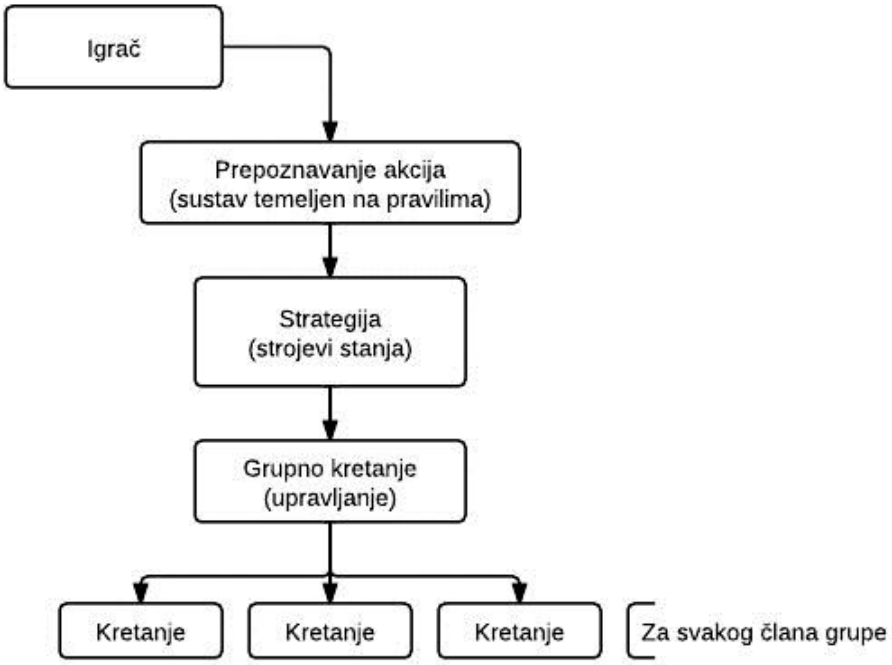
Odluke koje se donose na višim razinama najčešće su strateškoga ili taktičkog tipa. Kako se ide prema nižim razinama to se odluke odnose na manji broj likova. Algoritmi za donošenje taktičkih i strateških odluka isti su kao i algoritmi za donošenje običnih odluka.

Algoritmi kretanja ne izvršavaju se na višim razinama nego samo na najnižoj razini. Kretanje se izvodi za svaki lik pojedinačno, ali u skladu s kretanjem cijelog tima ili grupe.

Traženje puta za skupinu teže je nego za pojedini lik. Više likova koji se kreću zajedno u skupini trebaju veći prostor za kretanje ili se može dogoditi da različiti tipovi likova trebaju različite strateške ili taktičke prednosti. Tako, na primjer, u skupini može biti vojnik s teškim naoružanjem i izvidnik. Izvidnik se brže i lakše kreće po terenu od teško naoružanog vojnika. Postoje različiti načini rješavanja takvih problema. Jedno od njih jest da se cijela skupina prilagodi najslabijem članu.

Koordinirano djelovanje koje uključuje i stvarnog igrača teže je ostvariti. Stvarni igrač je nepredvidljiv i ima veću slobodu od likova kojima upravlja računalo. Najčešće igrač preuzima

donošenje odluke za cijelu skupinu. Da bi omogućili neke zabrane i ograničenja dizajneri igri najčešće oblikuju okolinu koja usmjerava igrača.



Slika 6.3 – Višerazinska umjetna inteligencija sa uključenim stvarnim igračem
[I. Millington, J. Funge, 2009, str. 589.]

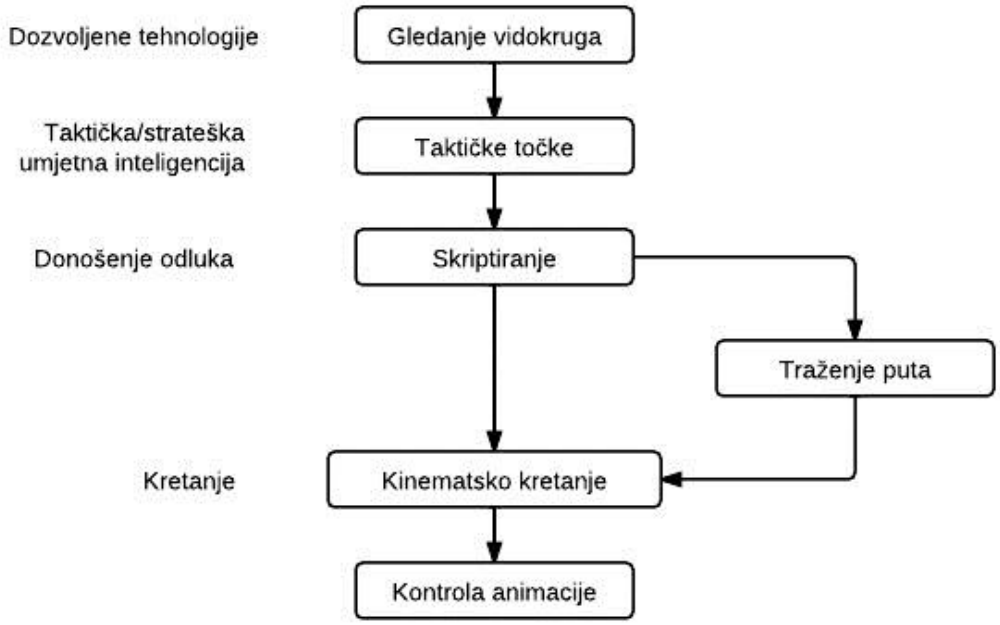
Na slici 6.3 nalazi se višerazinska umjetna inteligencija u usklađenom djelovanju sa stvarnim igračem. Igračeve su odluke na najvišoj razini, a druge se odluke i likovi njima prilagođavaju.

7. Umjetna inteligencija u pojedinim žanrovima igara

7.1 Pucačine

Pucačine (eng. shooters) su jedna od najvažnijih žanrova igri. Postoje pucačine u prvom licu (eng. FPS – first-person shooter) i pucačine u trećem licu (eng.TPS – third person shooter). U njima igrač preuzima ulogu lika u igri i može se kretati, pucati, te obavljati određeno djelovanje. Cilj u pucačinama je poraziti neprijatelja ili ostvariti određene ciljeve. Neprijatelji su najčešće likovi sličnih osobina kao i igrač, samo njima upravlja umjetna inteligencija.

Tipična umjetna inteligencija koja se koristi za upravljanje neprijateljima sadrži: kretanje, pucanje, donošenje odluka, opažanje, traženje puta i taktičku umjetnu inteligenciju [I. Millington, J. Funge, 2009, str. 814]. Arhitektura umjetne inteligencije u pucačinama prikazana je na slici 7.1.



Slika 7.1 – Arhitektura umjetne inteligencije u pucačinama
[I. Millington, J. Funge, 2009, str. 840.]

Kretanje je najvažniji dio ponašanja likova u igri. U svim modernim pucačinama poput **Medal of Honor, Call of Duty, Unreal Tournamentm, Crysis** i slično, istodobno je dopušteno vodoravno kretanje, skakanje i rotiranje (okretanje) kamere u svim smjerovima. Pucanje se ostvaruje u smjeru viđenja kamere. Sve to djelovanje treba predočiti kao animaciju. U igrama

poseban dio koda planira akcije, to jest, pokrete koje lik obavlja, dok drugi dio koda to predočuje u nizu animacija.

Najvažniji dio u pucačinama je, naravno, pucanje koje je u početku razvoja pucačina bilo previše točno i igrač se nije mogao izmaknuti mecima. Na kasnijem stupnju razvoja likovima je omogućeno da griješe.

Odlučivanje se najčešće ostvaruje pomoću automata stanja i stabla ponašanja. Na primjer, ako lik uoči neprijatelj, izvrši akciju „pucaj“, ako ne, izvrši akciju „ne pucaj“. Mnoga su ponašanja određena i okruženjem u kojem se lik nalazi ili drugim likovima. Da bi se dobio privid različitog ponašanja likova, na raspolaganju su rane slučajnosti

Percepcija označava kako lik reagira na svijet oko sebe. U starijim igrama svaki je lik okružen područjem koje reagira kada igrač uđe u njega. Kod je novih igri percipiranje je unaprijeđeno i likovi dobivaju poruke s obavijestima o okolini.

Pucačine imaju graf traženja puta na kojemu su veze koje predstavljaju razno djelovanje ili ponašanje potrebno liku da dođe do određenog mjesta pa tako igrač ima osjećaj da se lik prilagođuje okruženju u kojemu se nalazi. Velike su razine najčešće podijeljene na manje dijelove koji se zasebno učitavaju, a među njima se nalazi portal.

Slika 7.2 – Primjer pucačine Crysis 2

7.2 Simulacije vožnje

U igrama u kojima se najviše vozi najpotrebnija je umjetna inteligencija kretanja. Mogu postojati i drugi vidovi umjetne inteligencije, ali kretanje je primarno i najvažnije. Na slici 7.3 prikazana je arhitektura umjetne inteligencije u igrama utrka autima na stazi, kao na primjer **Toca Race Driver.**

Slika 7.3 – Arhitektura umjetne inteligencije u simulacijama vožnje
[I. Millington, J. Funge, 2009, str. 845.]

Kretanje u igrama vožnje može se ostvariti na dva načina. Nekad su se takve igre radile tako da je dizajner odredio najbolju putanju na stazi koju prate auti kojima upravlja umjetna inteligencija. Staza se određuje uz pomoć matematičkih krivulja, a također se određenim dijelovima staze mogu odrediti dodate osobine. Umjetna inteligencija neprestano provjerava položaj i brzinu auta i prema potrebi je podešava. U ranijim je igrama problem bio da protivnici nisu mogli prolaziti jedan pokraj drugoga ili, ako se dogodila nesreća, auti je nisu pokušali zaobići nego su se jednostavno zaletjeti u druge aute i slično. Da bi se takve situacije izbjegle i protivnici bili što stvarniji, dodaju se dijelovi koda koji omogućuju otkrivanje sudarenih auta i njihovo zaobilaženje. Na slici 7.4 je prikazana igra **Toca Race Driver**.

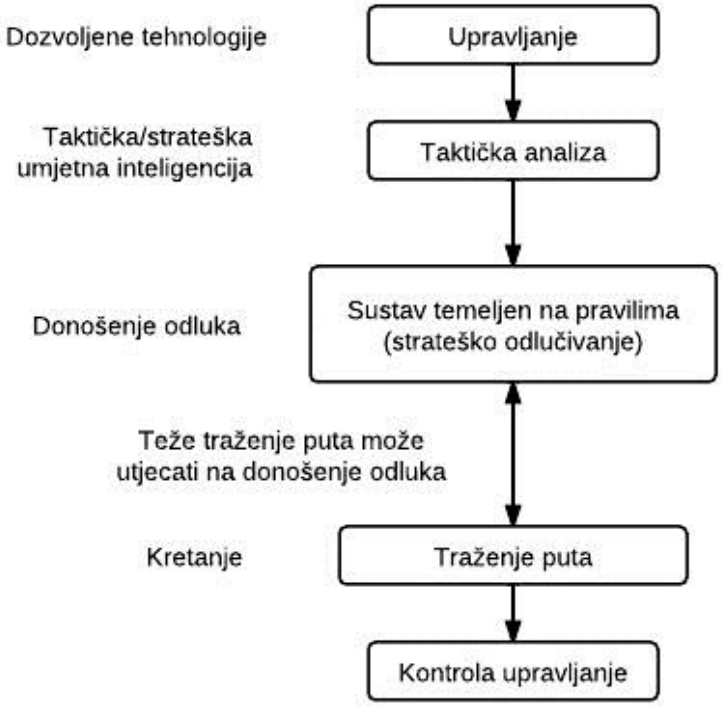
Postoji i drugi pristup izradi umjetne inteligencije kod vozačkih igri koji se pretežito koristi u današnjim igrama. U tom pristupu umjetna se inteligencija ponaša kao vozač i upravlja autom uz pomoć simulacije fizike kako bi se auto ponašao što stvarnije.

Slika 7.4 – Prikaz igre Toca Race Driver [http://ps2.ign.com/dor/objects/690291/toca-race-driver-2/images/toca-race-driver-2-20040701041433876.html]

Kod ostalih simulacija poput simulacija vožnje zrakoplova ili podmornice koriste se algoritmi slično kao i u drugim tipovima igra. Upravljanje zrakoplovima je realizirano kao upravljanje likom u pucačini uz mogućnost kretanja u svim smjerovima. Kod simulacija podmornice važni su strateški i taktički algoritmi. Simulacije pokušavaju što bolje predočiti stvarni osijecaj upravljanja nekim vozilom stoga moraju uključivati simulaciju fizike i prikaz što realnije okoline.

7.3 Strategije u realnom vremenu

Strategije su jedna od dominantnih žanrova igri u kojima igrač gradi bazu ili grad, skuplja resurse, stvara i vodi svoju vojsku kako bi porazio protivnika. Igrači kojima upravlja računalo, to jest umjetna inteligencija, moraju se ponašati kao igrači, to jest, moraju poraziti protivnika skupljanjem resursa, izgradnjom, proizvodnjom i taktičkim kretanjem jedinica. Primjer strategija u realnom vremenu (eng. RTS – real-time strategy) su **Age of Empires, Command & Conquer** te **Warcraft.** Umjetna inteligencija u njima koristi traženje puta, skupna kretanja, taktičku i stratešku umjetnu inteligenciju i odlučivanje [I. Millington, J. Funge, 2009, str. 824]. Na slici 7.4 prikazana je najčešće korištena arhitektura umjetne inteligencije.



Slika 7.5 – Arhitektura umjetne inteligencije u strategijama u realnom vremenu
[I. Millington, J. Funge, 2009, str. 849.]

Traženje puta jedno je od ključnih dijelova umjetne inteligencije u strategijama. Karte na kojima se odvija igra jako su velike kao i količina jedinica koje se mogu kretati. Kretanje po karti je načinjeno tako da je karta podijeljena na jako malo dijelove, kocke, koje su u suvremenim igrama nevidljive igraču. Formacija kojom se više jedina kreće u skupini najčešće je predodređena nekim uzorkom, ali danas može biti i dinamički načinjena.

Strategija i taktika među najvažnijim su dijelovima strateških igri. Primjer strategije je odabir lokacija za gradnju zgrada. Zidovi bi trebali biti izgrađeni tako da štite druge najvažnije građevine i slično. U današnjim strateškim igrama poput **Empire Eartha** i **Age of Empires** 2, strateška gradnja omogućena je uz pomoć označavanja važnosti na karti. Važna područja zabilježena su na drugačiji način i umjetna inteligencija zna da oko tog područja treba postaviti zid.

Taktički vid možemo naći kod sukoba velikih vojski u igrama poput serijala **Total War.** Manje pokretne jedinice drže se izvan dosega topništva, konjica pokušava napasti bokove, napadi se usredotočuju na jedinice sa slabijim moralom i slično.

Odlučivanje u strategijama uvijek se odvija na više razina i uvijek je zastupljeno više načina odlučivanja [I. Millington, J. Funge, 2009, str. 826]. U igrama se događa mnogo stvari,

zastupljeno je mnogo informacija, uvjeti i stanja neprestano se mijenjaju, odluke koje treba donijeti, za pojedince ili skupine, stalno se izmjenjuju. Zato je odlučivanje u strateškim igrama veoma važno jer se tako stvara privid ozbiljnijeg i stvarnijeg protivnika.

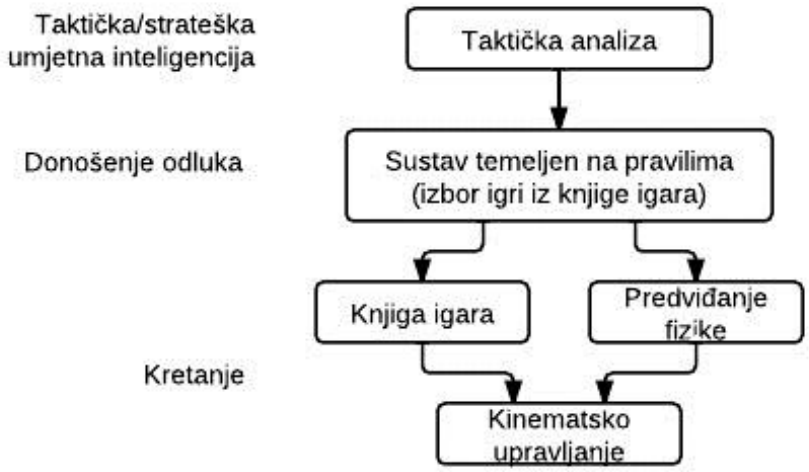
U starijim strateškim igrama odlučivanje je podijeljeno na dijelove. Na primjer, određeni dio donosi odluke o resursima neovisno o dijelu koji odlučuje što treba graditi. U novijim strategijama postoje središnjice koji mogu donositi odluke za sve dijelove odlučivanja. Kod nekih strateških igri postoji i hijerarhijsko odlučivanje s činovima kao u vojsci. Na primjer, ako je prisutan, general ima najveće ovlasti.

Najčešće upotrebljavani algoritmi su automati stanja, stabla odlučivanja ili neki sustavi temeljeni na pravilima. Na slici 7.6 je prikazana igra **Age of Empires 2.**

Slika 7.6 – Prikaz igre Age Of Empires 2
[http://gameland.jutarnji.hr/wp-content/gallery/age-of-empires-2/age-of-empires-2_2.jpg]

7.4 Sportske igre

Danas ima mnogo sportskih igara poput **Fife** ili **NHL**. Problem kod sportskih igri je taj što igrač očekuje da protivnik, kojim upravlja računalo, bude što sličniji stvarnom protivniku. Prednost je što već postoje gotove strategije, taktike i velika baza znanja sa svim podacima potrebnim za igru.



Slika 7.7 – Arhitektura umjetne inteligencije kod sportskih igri
[I. Millington, J. Funge, 2009, str. 852.]

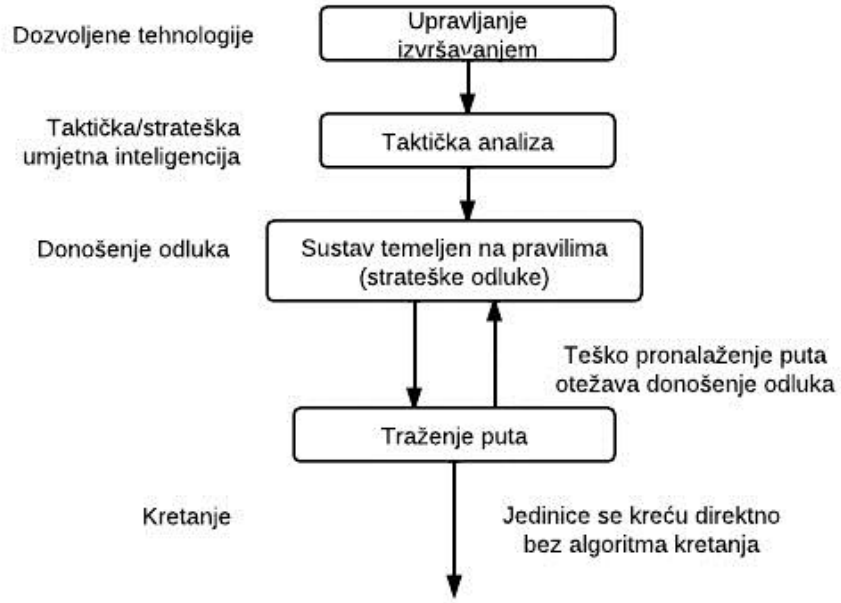
Arhitektura sportskih igara prikazana je na slici 7.5. Kako je većina sportova timska, najveći je problem postići da pojedini likovi reagiraju na stanje tako da uključe i ostatak tima. Umjetna inteligencija u sportskim igrama radi na više razina. Najviša razina bavi se strateškim odlukama. Na srednjoj razini odvijaju se koordinirani pokreti po uzorku koji je odgovor na neku situaciju i na zadnjoj razini svaki pojedini lik odlučuje kako će se ponašati s obzirom na globalnu strategiju [I. Millington, J. Funge, 2009, str. 827]. Kod sportova koji nisu timski nema srednje razine.

Slika 7.8 – Prikaz igre Fifa 2009
[http://rifaiy.blogspot.com/2011/03/download-fifa-2009.html]

Odlučivanje u sportskim igrama nije složeno. Podaci iz knjige igara upućuju na to kako treba reagirati na određenu situaciju. Ta knjiga sadrži odluke o tome kako likovi moraju djelovati. Važan dio kod odlučivanja je i predviđanje fizike. Ono je jako važno za predviđanje ponašanja likova da bi što bolje igrali.

7.5 Strategije na potez

Strategije na potez (eng. TBS – turn-based strategy) slične su strategijama u realnom vremenu. Danas ih ima jako mnogo, a najpoznatiji i najpopularniji primjeri su serijal **Civilization, Heroes of Might and Magic** i **Worms.** Kod ovog tipa strategija algoritmi kretanja nisu potrebni. Lik se najčešće samo prebaci s jednog položaja na drugi, ali uključeno je traženje puta, odlučivanje i strateško planiranje. Sve se to izvodi kao i kod strategija u realnom vremenu.



Slika 7.9 – Arhitektura umjetne inteligencije u strategijama na potez
[I. Millington, J. Funge, 2009, str. 854.]

Glavna razlika između strategija u realnom vremenu i strategija na potez je u vremenu. Kod strategija na potez vrijeme nije važno. Igrač može uzeti neograničeno vremena i razmišljati o svom potezu. Također, računalo može duže i podrobnije raditi taktičke analize i donositi odluke. Odluke se najčešće donose uz pomoć sustava temeljenog na pravilima. Iako nema kretanja, može sadržavati algoritme za traženje puta i omogućilo strateško prebacivanje jedinica na karti.

Zanimljiv primjer je serijal igara **Total War** u kojima imamo dio koji je strategija na potez i drugi dio igre koji je strategija u realnom vremenu. U prvom dijelu imamo stratešku kartu određenog dijela svijeta i upravljamo jedinicama naizmjence sa protivničkim igračima sve dok se ne dogodi neka bitka u kojoj preuzimamo upravljanje jedinicama na bojom polju u realnom vremenu. Primjer jednog i drugog dijela imamo na slikama 7.7 i 7.8. Na prvoj slici je prikazan dio koji se odvija na potez u igri Napoleon Total War, a na drugoj slici dio koji se odvija u realnom vremenu.

Slika 7.10 – Dio igre koji se odvija na potez u igri Napoleon Total War

Slika 7.11 – Dio igre koji se odvija u realnom vremenu u igri Napoleon Total War

8. Zaključak

Umjetna inteligencija važan je dio računalnih igri. S razvojem igara razvija se i ona. U igrama ona omogućuje da se likovi kojima upravlja računalo ponašaju onako kako bi se ponašali da njima upravlja čovjek

U počecima razvoja igre su imale samo jednostavne algoritme kretanja koje omogućuje liku da se kreće po razini ili mapi. Današnji algoritmi kretanja omogućavaju napredno upravljanje, izbjegavanje prepreka, skakanje, bježanje ili traženje mete. Najbolji primjer za kretanje može se vidjeti u pucačinama gdje su najčešće ostvareni svi algoritmi kretanja.

Uz kretanje važni su i algoritmi za traženje puta. Najvažnija dva su Dijkstra algoritam i A* algoritma koji se i najviše koriste. Svaka razina u igrama može se predstaviti kao graf čvorova međusobno povezanih vezama i troškovima prolaska tih veza. Algoritmi traženja rade tako da gledaju preko kojih čvorova mogu doći od početnog do završnog čvora uz minimalan trošak, najčešće vremena. Kod strateških igara imamo grupe jedinica koje, kada im damo naredbu da dođu na neko mjesto na karti, sami traže najbržu rutu i zaobilaze prepreke kako bi došli do zadanog cilja.

Odlučivanje je jedno od glavnih dijelova umjetne inteligencije. Ono omogućuje računalu da sam odabire potrebno djelovanje. Može se ostvariti pomoću stabla odlučivanja, automata stanja, neizrazite logike i drugih algoritama i sustava. Danas se najčešće koristi više algoritama za odlučivanje istovremeno. Na primjer može postojati odlučivanje uz pomoć automata stanja i stabla odluka. Jednostavan primjer odlučivanja je kada se lik nalazi u stanju stražarenja i u slučaju nailaska neprijatelja on ga napada.

Taktičko i strateško razmišljanje ključno je u današnjim igrama. Ono se najčešće ostvaruje spojem različitih jednostavnijih algoritama za odlučivanje, kao na primjer sustava temeljenog na pravilima. Strategija i taktika omogućavaju računalu da razmišlja na višoj razini od pojedinca. Da se skupina likova zajednički i usklađeno organiziraju u donošenju odluka, kretanju i djelovanju.

Postoje mnogi žanrovi računalnih igara. U nekima se koriste napredni algoritmi kretanja kao na primjer u simulacijama vožnje dok u nekim imamo zastupljene sve dijelove modela umjetne inteligencije kao što su to strategije ili pucačine.

Umjetna inteligencija u igrama se stalno razvija i pokušavaju se naći novi algoritmi ili poboljšati stari. Danas igre imaju jako naprednu inteligenciju koja je napravljena uz pomoć algoritama prikazanim u ovom radu ili nekih njihovih varijanata. Osim napredne grafike, velikih prostora za kretanje, postoje i napredniji protivnici koji čine igru težom i realnijom za igrati.

Razvoj tehnologije i povećanje računalnih performansi omogućit će igrama korištenje još kompleksnijih algoritama i većeg broja podataka. Inteligencija u igrama još uvijek nije dosegla svoj maksimum i ne može zamijeniti stvarnog protivnika.

Literatura

1. Aiolli F. and Palazzi C. E., *Enhancing Artificial Intelligence on a Real Mobile Game*, International Journal of Computer Games Technology, vol. 2009, http://www.hindawi.com/journals/ijcgt/2009/456169/ (učitano 17.7.2011.)
2. Allis L. V., *Searching for Solutions in Games and Artificial Intelligence*
3. Anderson E. F., *Playing smart - artificial intelligence in computer games,* CiteSeerx, 2003.
4. Baumgarten R., Colton S., Morris M., *Combining AI Methods for Learning Bots in a Real-Time Strategy Game*, International Journal of Computer Games Technology, vol. 2009, http://www.hindawi.com/journals/ijcgt/2009/129075/ (učitano 17.7.2011.)
5. Cormen T. H., Leiserson C. E., Rivest Clifford Stein R. L., *Introduction to Algorithm, Third Edition s*, MIT, Cambridge, 2009.
6. Doss P., *Artificial Intelligence in Computer Games* , Bowie State University, 2003. http://ac-support.europe.umuc.edu/~sdean/ProfPaps/Bowie/T1-0607/Doss.pdf (učitano 10.7.2011.)
7. El Rhalibi A., Kok Wai Wong, Price M., *Artificial Intelligence for Computer Games*, International Journal of Computer Games Technology, vol. 2009, http://www.hindawi.com/journals/ijcgt/2009/251652/ (učitano 17.7.2011.)
8. González-Calero P. A., Gómez-Martín M. A., *Artificial Intelligence for Computer Games*, Springer, 2010.
9. Logan S., *The Future of Artificial Intelligence in Games*, 2010. http://api.ning.com/files/V68k0DlvXmH-Y02kYEmOexKanB*ScgumEHXr0CbQTBlZinQLR8gcyVZxqb0Df1H0mFtJApq-9vvBTIG-43OesSZR5uPR-h*w/SLogF.pdf (učitano 13.9.2011.)
10. Millington I., Funge J., *Artificial Intelligence for Games, Second Edition*, Morgan Kaufmann Publisher, 2009.

11. Mott K., *Evolution of Artificial Intelligence In Video Games: A Survey*, University Of Northern Iowa, 2009. http://www.cs.uni.edu/~schafer/courses/previous/161/spring2009/proceedings/papers/paperG.pdf (učitano 10.7.2011.)

12. Russel S., Norvig P., *Artificial Intelligence A Modern Approach, Third Edition*, Prentice Hall, 2009.

13. Wexler J., *Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White" and where it can and will go in the future*, 2002. http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf (učitano 10.7.2011.)

14. Schaeffer J, *One jump ahead: computer perfection at checkers*, Springer, 2009.

15. Schaeffer J., J. van den Herik H., *Games, computers, and artificial intelligence*, Elsevier http://www.acso.uneb.br/marcosimoes/Arquivos/IA/sdarticle_games.pdf (učitano 10.7.2011.)