Wumpusov svijet uz pomoć Mindstorm robota

Dmitrović, Matej

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

Sveučilišni diplomski studij računarstva – informacijske i podatkovne znanosti

WUMPUSOV SVIJET UZ POMOĆ MINDSTORM ROBOTA

Diplomski rad

Matej Dmitrović

Osijek, 2022.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 08.09.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Matej Dmitrović
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1045R, 06.10.2019.
OIB studenta:	67929836880
Mentor:	Izv. prof. dr. sc. Mirko Köhler
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 1:	Izv. prof. dr. sc. Mirko Köhler
Član Povjerenstva 2:	Prof. dr. sc. Krešimir Nenadić
Naslov diplomskog rada:	Wumpusov svijet uz pomoć Mindstorm robota
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Objasniti i riješiti problem Wumpusovog svijeta pomoću LEGO Mindstorm robota.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/bodaPostignuti rezultati u odnosu na složenost zadatka: 2 bod/bodaJasnoća pismenog izražavanja: 2 bod/bodaRazina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	08.09.2022.

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

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 28.09.2022.

Ime i prezime studenta:	Matej Dmitrović
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1045R, 06.10.2019.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Wumpusov svijet uz pomoć Mindstorm robota**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Mirko Köhler

i sumentora ,

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

Potpis studenta:

SADRŽAJ

1. UVOD

Još u davnim vremenima, čovjek je težio za automatiziranjem poslova kako bi sebi olakšao život. Automatizacija je postala izraženija industrijskom revolucijom kada su se počele graditi tvornice koje su svojim automatizmom imale veliki utjecaj na cjelokupni tijek čovječanstva. Pojavom modernih računala, želja za automatizacijom poslova se pretvorila u želju za izgradnju autonomnih jedinica – robota. Umjetna inteligencija je ključ njihove izrade – ali što više želimo da roboti sliče ljudima, taj ključ je sve teže iskovati. Težiti velikim stvarima nije pogrešno, no potrebno je ponekad vratiti se unatrag radi utvrđivanja osnova. U ovom radu ne će se istraživati roboti koji se ne mogu razlikovati od čovjeka, nego roboti koji su specijalizirani za odrađivanje jednog određenog scenarija. Teorija o umjetnoj inteligenciji će biti praktički prikazana prilikom rješavanja *Wumpusovog svijeta* - problema umjetne inteligencije u kojem agent mora pronaći siguran put do zlata izbjegavajući opasnosti tako što će opažati svoju okolinu. Sam agent je napravljen pomoću Lego Mindstorms robot-igračke s mogućnosti kretanja i razlikovanja boja.

Rad je podijeljen u pet poglavlja. U drugom poglavlju daju se osnovne definicije potrebne za razmatranje umjetne inteligencije. U trećem poglavlju opisuju se karakteristike problema *Wumpusovog svijeta* i daje se primjer rješavanja jedne od mogućih konfiguracija. Četvrto poglavlje opisuje Lego Mindstorms, njegove značajke i tehnička ograničenja. U petom poglavlju daje se rješenje *Wumpusovog svijeta* pomoću Lego Mindstormsa na način da se opisuje konstrukcija agenta, i programski kod korišten za agentovo razmišljanje. Šesto poglavlje obuhvaća rad i daje viđenja o budućnosti umjetne inteligencije.

1.1. Zadatak rada

Objasniti i riješiti problem Wumpusovog svijeta pomoću Lego Mindstorm robota.

2. INTELIGENCIJA

Kao temelj ovog rada, potrebno je definirati što je to točno inteligencija, ili barem pokušati. Problem je što se inteligencija ne može jednoznačno definirati, u čemu se slažu [1] i [2]. Earl B. Hunt [1] pokušava definirati inteligenciju pragmatičnim pristupom: “Inteligencija je ono što testovi za inteligenciju mjere.” Dakako, definiranje pojmova kružnom logikom nije znanstveni način tako da [3] definira inteligenciju kao sposobnost nositi se s umnom složenošću. Prema [4] poistovjećuju se inteligencija i dobro prosudba, s time da dan naglasak na prilagodbu ovisnu o situaciji, te samokritičnost. Par istraživača u [5] inteligenciju definiraju da je prilagodljivo ponašanje usmjereno cilju, što nalaže da je potreban cilj kako bi se ponašanje moglo smatrati inteligentnim. Inteligencija u svakom kontekstu proizlazi iz sposobnosti da se inteligentno biće može prilagoditi na trenutnu situaciju kako bi izvršilo određeni cilj, tako što će razmisliti s čime raspolaže, te o iskustvu koje je već steklo..

2.1. Prirodna inteligencija

Inteligenciju živih bića nazivamo prirodnom inteligencijom. To je biološki određena stavka ovisna o vrsti živog bića. Prema [6] posebno je definira prirodna inteligencija da je sposobnost da se djeluje onako kako treba s obzirom na situaciju bez pravljenja grešaka. Iako se inteligencija najviše povezuje s ljudskom vrstom, ona se može naći i u drugim, čak beskralježnjačkim vrstama: u članku [7] provedeno je istraživanje u kojem se pokazalo da zemljani crvi imaju sposobnost praćenja jednostavnih puteva i izbjegavanja štetnih kemikalija. Veća razina inteligencije je primijećena kod čovjekolikih majmuna u [8] gdje su čovjekoliki majmuni koristi i čuvali alate za buduću upotrebu. Ipak, ljudska vrsta (latin. *homo sapiens*) smatra se najinteligentnijim bićem za kojeg trenutno znamo. Philip C. Jackson [6] kao dokaz ljudske inteligencije navodi četiri opsežna polja:

- povijest - evidentirana povijest je sama po sebi dokaz o ljudskoj inteligenciji
- samoispitivanje - ovaj dokaz svodi se na Descartesov poznati citat: “mislim, dakle postojim”. Kao glavnu razliku između ljudske i životinjske inteligencije, Descartes napominje ljudsku sposobnost komunikacije - pisanje, rečenice, simbolika i ostale stvari koje nisu rezultat instinktivnog ponašanja. Ljudska komunikacija je rezultat učenja i opažanje okoline u kojoj čovjek odrasta, te je sama sposobnost učenja te komunikacije rezultat ljudske inteligencije.
- društvene znanosti - sociološki gledano, ljudska inteligencija je osobina koju posjeduju svi ljudi. Razvoj ljudske inteligencije se odvija u fazama: u svakoj fazi potrebno je skupiti

dovoljno iskustva kako bi se mogla započeti sljedeća faza. Faze razvoja inteligencije se najviše razvijaju od rođenja do 11 godine života. Svaka faza objedinjuje neku od cjelina poput vladanja senzo-motoričkim osobinama, razlikovanje simbola, razumijevanje formalnog jezika i sl. Uz vrijeme, za razvoj inteligencije nadovezuje se i okolina. Okolina u kojoj se čovjek razvija može podosta utjecati na cjelokupan razvoj. Primijećeno je da identični blizanci koji se razvijaju u istoj okolini imaju slične kvocijent inteligencije, dok identični blizanci koji su se razvijali u potpuno različitim okolinama mogu imati kvocijent inteligencije različit 20 bodova.

- biologija - mozak se smatra najkompleksnijim organom u čovjeka, stoga još nemamo cjelokupno znanje o njemu. Do sada je zaključeno da pojedini dijelovi mozga kontroliraju određeni dio funkcije tijela poput upravljanje hormonima, motorikom; obrada slike, njuha i ostalih osjetila. Ono što je također poznato jest način na koji mozak komunicira s tijelom: pomoću neurona. Neuroni su živčane stanice koje prenose informacije od ili do mozga. Koriste električne impulse kako bi se informacija prenijela s jednog neurona na drugi. Električni impuls nastaje kada dođe do razlike potencijala na granama neurona zbog razlike u koncentraciji K^+ iona. Sam prijenos se razlikuje od prijenosa informacije žicom unutar računala u tome što električni impulsi nisu diskretne vrijednosti 0 i 1, nego su kontinuirane.

Zajednička stvar svih inteligentnih bića je primanje vanjskih podražaja (vid, miris, sluh, dodir, razina svjetlosti, itd.) te reagiranje na te podražaje. Kako bi bića mogla primiti vanjske podražaje, koriste se osjetilima. Osjetila su organi koji služe za opažanje okoline i kreiranje informacije o njoj. Način prijenosa informacije ovisi o biološkom carstvu u kojem se biće nalazi. Tako, na primjer, listovi biljaka su organi koji reagiraju na sunčevu svjetlost tako što će slati informacije za povečanje duljine svojih stanica prema svjetlosti koristeći talk vode [9], dok će čovjekovo oko slati električne signale prema mozgu [6]. Najbolji primjer ovog toka informacije jest čovjekovo oko. Oko je organ kojim ljudi primaju i osjete Sunčevu svjetlost. Padanje zraka svjetlosti na čunjiće i štapiće unutar oka uzrokuje stvaranje električnih signala. Jakost tih električnih signala ovisi o raznim karakteristikama svjetla što je upalo u oko. Ti signali potom putuju prema mozgu preko živčanih stanica - neurona. Mozak zna kako pretvoriti te električne signale u smislenu informaciju. Tako se električni signali koji dođu od oka pretvaraju u sliku što ljudi vide. Ta slika je informacija trenutnog stanja. Na temelju te slike odlučuje se što će sada dogoditi, tj. kako će čovjek reagirati. Vidi li čovjek primamljivu kantu sladoleda? Ako vidi samo tu kantu sladoleda i samo o njoj razmišlja, tada razmišlja samo o trenutnom stanju i o tome kako bi je htio uzeti. Prisjeća li se

Commented [s1]: Povečavanje duljine

Commented [s2]: Redoslijed riječi

čovjek da ne može jesti sladoled jer ne podnosi laktozu? Tada čovjek razmišlja o trenutnom, ali i prijašnjim stanjima te na temelju prijašnjih iskustava odlučuje da ipak neće uzeti kantu sladoleda.

2.2. Umjetna inteligencija

Umjetna inteligencija pojavljuje se već u prvoj polovici 20. stoljeća, ali u obliku znanstvene fantastike. Pojam umjetne inteligencije nastaje tek 1956. godine kada je počeo značajniji razvoj računala i računalnih tehnologija. Do danas su se razvila brojna znanstvena polja vezana uz umjetnu inteligenciju poput strojnog učenja, računalnog vida i robotike.

Kako je samu inteligenciju teško jednoznačno definirati, umjetnu inteligenciju je još teže. Već u 2. poglavlju je navedena pragmatična definicija iz [1], no bolju definiciju možda nudi [10] koja glasi: "umjetna inteligencija je sposobnost računala da obavljaju zadatke za koje bi ljudi rekli da je potrebna inteligencija". [11] nudi sličnu definiciju u kojoj se umjetna inteligencija ne gleda kao pojam nego kao nauka o stvaranju računala koja mogu obnašati zadatke za koje je potreban um. Za umjetnu inteligenciju tako možemo reći da je to sposobnost računala da rješava probleme na sličan način kako bi ih čovjek riješio - tako što će računalo analizirati okolinu u kojoj se nalazi, pokušati postići cilj na temelju stečenog znanja, te prilagoditi na novonastale uvjete i stanja.

2.2.1. Matematička i logička pozadina

Primarna svrha umjetne inteligencije je rješavanje problema. Problem se može definirati u obliku trenutnog stanja i željenog stanja, s operacijama kojim se mijenjaju stanja. [1] definira stanje kao uređenu n-torku vrijednosti koji opisuju problem u određenom trenutku. Ako imamo neko početno stanje (a_1, a_2, a_3) i željeno stanje (z_1, z_2, z_3), tada operacije mijenjaju trenutno stanje (t_1, t_2, t_3) u novo stanje (n_1, n_2, n_3). Ista operacija možda davati drugačiji rezultat ovisno o trenutnom stanju u kojem se nalazimo, tako da operaciju možemo smatrati funkcijom Θ koja iz domene stanja preslikava u kodomenu stanja. Problem umjetne inteligencije se bazira na pronalaženju kombinacije operacija koje će dovesti do traženog stanja $\Theta_1(\Theta_2(...\Theta_n((a_1, a_2, a_3))...)) = (z_1, z_2, z_3)$, odnosno pronalaženju kompozicija operacija za koje vrijedi $(\Theta_1 \circ \Theta_2 \circ \ldots \circ \Theta_n)((a_1, a_2, a_3)) = (z_1, z_2, z_3)$. U kompleksnim, realnim problemima, nisu poznata sva moguća stanja unaprijed već se trebaju iskustvom saznati, što znači da se domena i kodomena operacija mogu proširivati tijekom samog izvršavanja problema.

Inteligentno biće koje je svjesno trenutnog stanja i operacijama što može iskoristiti, ne može sa sigurnošću znati koju operaciju treba napraviti. Kriva operacija u određenom stanju može dovesti do završnog stanja - stanja s kojeg se ne može prijeći u ostala stanja. Kao pomoć pri rješavanju

problema umjetne inteligencije, koriste se logički izrazi pomoću kojih se odabire sljedeća operacija. Logika se temelji na istinitim vrijednostima, tako da odabir operacije se vrši ukoliko se stanje smatra da zadovoljava uvjete za izvršenje operacija. Razina logike koju inteligentno biće koristiti i koliko prošlih (ili budućih) stanja uzima u obzir prilikom definiranja uvjeta ovisi o razini inteligencije koje ono posjeduje. Najjednostavnija inteligentna bića provjeravaju samo trenutno stanje pri odabiru operacije, dok složenija bića zaključke donose i na temelju stanja koje je već svjedočio.

2.2.2. Tehnologija inteligencije

U prijašnjim poglavljima pokazano je kako su osjetila ključna za bilo kakvu vrstu prirodne inteligencije, budući da nam osjetila omogućuju promatranje okoline što opisuje trenutno stanje. Kada govorimo umjetnoj inteligenciji, ulogu osjetila preuzimaju senzori. Senzori, poput osjetila, hvataju vanjske podražaje te ih pretvaraju u električne signale. Ovi signali mogu biti analogni ili digitalni ovisno o vrsti senzora i što se tim senzorom želi mjeriti. Senzori najčešće očitavaju tzv. čiste podatke koje odmah nakon očitavanja pretvore u neku od razumljivih vrijednosti. Čisti podaci imaju najčešće mogu imati vrijednosti od 0 do 2 na broj bitova koji senzor koristi prilikom rada. Dobivena vrijednost se potom prenose do drugih dijelova putem žica i električne struje.

Signali koji dođu iz osjetila i senzora moraju se obraditi kako bi inteligentno biće moglo odrediti kako reagirati na podražaj. Kod bioloških bića, ovu ulogu igra mozak. Kod tehnoloških kreacija nije moguća jednoznačna usporedba s mozgom, ali se najčešće uzima da je to središnja procesorska jedinica (eng. *Central Processing Unit - CPU*) u Neumanovoj arhitekturi. Dolaskom signala iz senzora, u središnjoj procesorskoj jedinici odlučuje se koja će biti reakcija. Ovisno o tipu samog inteligentnog agenta, ova reakcija može biti reakcija na podražaj koja će se uvijek ponavljati neovisno o situaciji, ili pomno razmišljanje i zaključivanje isplati li se odraditi jednu reakciju ili pak drugu. Od velike važnosti je i programski kod prilikom samog razmišljanja. Pomoću skupa naredbi, kod može reprogramirati način na koji agent razmišlja i reagira - od jednostavnog agenta moguće je napraviti složenije i obrnuto.

Nakon obrade podražaja i odluke kako reagirati, potrebna je sama reakcija. Reakcija se najčešće svodi na pomicanje dijelova tijela. U umjetnoj inteligenciji svaka komponenta koja služi za pomicanje ili kontroliranje nekog mehanizma se naziva aktuatorom. Najčešći aktuatori su elektromotori koji koriste električnu energiju kako bi proizveli rotaciju. Mogući su i oni koji će raditi translaciju, ali, za razliku od rotacije koja se može nastaviti nakon 360°, takvi aktuatori imaju ograničenu sposobnost kretanja. Osim podjele po načinu kretanja, postoji podjela po izvoru

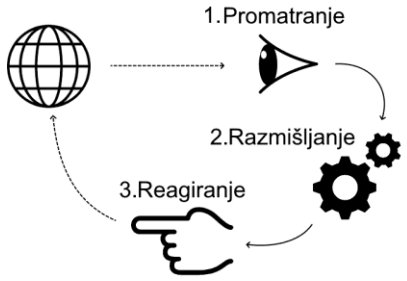
Commented [MK3]: Previše se u jednom poglavlju spominje riječ aktuatori. Izbaciti gdje nije nužno.

energije: električni, hidraulički, pneumatski, termalni, magnetski i zavojnice polimera. U nastavku ovoga rada, razmatraju se samo električni rotacijski aktuatori.

2.2.3. Inteligentni agenti

Prema [12] inteligentni agent je bilo što što može osjetiti okolinu i samostalno reagirati kako bi postigao željeni cilj, uz mogućnost učenja na stečenim iskustvima. Agenti za tu svrhu posjeduju umjetnu inteligenciju koja im omogućava to.

Već je par puta naglašeno da je karakteristika inteligentnog bića da reagira na okolinu. Ta činjenica se odnosi i na inteligentne agente te je dio same definicije inteligentnih agenata. Općeniti princip po kojem radi svaki inteligentni agent jest da agent osjeti okolinu pomoću senzora, na temelju očitanih vrijednosti odluči kako će reagirati, te sama reakcija.

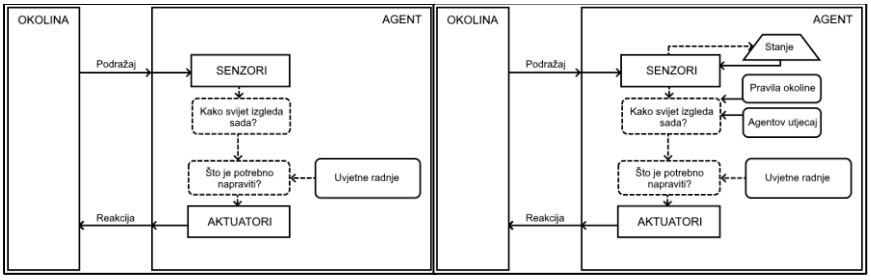


Sl.2.1. Opći dijagram rada inteligentnog agenta.

S. J. Russell and P. Norvig [12] inteligentne agente klasificiraju u ovisnosti o stupnju percipirane inteligencije. Jednostavni refleksivni agenti (Slika 2.2) primaju podražaje iz okoline i samo reagiraju na njih na temelju danih uvjeta (ako vidim ovo, činim ono). Ne postoji čuvanje prošlih osjeta i stanja već se reagira isključivo na ono što agent vidi. Zbog tog svojstva, agenti ove vrste uspijevaju u samo potpuno vidljivim okolinama jer u djelomično vidljivim okolinama, ovi agenti često naiđu na beskonačnu petlju. Upravo zbog svoje jednostavnosti i predvidljivosti, jednostavni refleksni agenti imaju najveću industrijsku upotrebu u automatizaciji.

Refleksivni agenti temeljeni na modelu (Slika 2.2) u sebi imaju znanje kako okolina oko njih funkcionira i u kojem stanju se agent trenutno nalazi. Ovi agenti imaju mogućnost da koriste

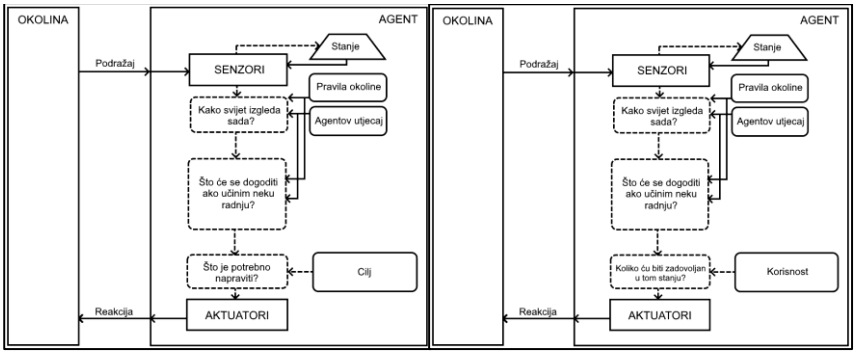
saznanja stečena o prošlim stanjima kako bi stvorili predodžbu okoline koju još nisu otkrili. Zbog navedenih karakteristika, ovi agenti mogu rješavati probleme u djelomično vidljivoj okolini.



Sl.2.2. Model refleksivnog agenta (lijevo) i model refleksivnog agenta temeljenog na modelu (desno) napravljen prema uzoru na [12].

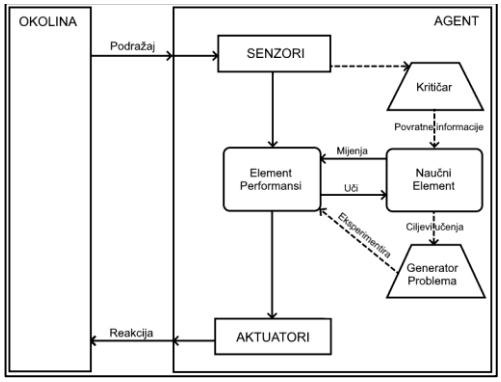
Agenti bazirani prema cilju (Slika 2.3) su nadogradnja agenata temeljenih na modelu. Ovaj tip agenata koristi informacije o krajnjem cilju kako bi odabrao način na koji će reagirati. To znači da ako je agent u situaciji da mora birati, agent će odabrati onu akciju koja će ga dovesti bliže zadanom cilju.

Agenti temeljeni na korisnosti (Slika 2.3) razlikuju samo ciljna i ne-ciljna stanja. Ovi agenti koriste funkciju za preslikavanje okolnih podražaja u vrijednost koja iskazuje korisnost stanja. Cilj te funkcije jest odabiranje sljedećeg stanja koji najviše odgovara ciljnom stanju. Tako agent može odlučiti kojem stanjem teži ako na raspolaganju ima više sljedećih stanja. Zbog načina na koji ovi agenti funkcioniraju, agenti ovoga tipa moraju imati mogućnost detaljnog opažanja okoline kako bi mogli što točnije odrediti sljedeće stanje. Razlika između agenata temeljenih na korisnosti i agenata baziranih prema cilju jest to što agenti bazirani prema cilju mjere samo to jesu li uspjeli doći do cilja ili ne (diskretne vrijednosti), dok agenti temeljeni prema korisnosti osim uspjeha mjere i performanse koje su ih dovele do tog cilja (kontinuirane vrijednosti). Njihova sličnost se očituje u strukturi modela prikazanih na slici 2.3.



Sl.2.3. Model inteligentnog agenta baziranog prema cilju (lijevo) i model inteligentnog agenta temeljen na korisnosti (desno) napravljen prema uzoru na [12].

Zadnji tip inteligentnih agenata prema [12] su učeći agenti (Slika 2.4). Ovi agenti učenjem mogu se snalaziti u nepoznatim okolinama i stjecati sposobnosti koje nisu imali pri početku rada. Sposobnost učenja imaju zbog načina na koji su modelirani - dobivanjem podražaja, ovi agenti osim što odlučuju što treba napraviti sljedeće, također su samokritični o podražaju koje prima te o podražajima koji su bili primljeni i akcijama koji su poduzeti. Samokritičnost daje način na koji se ovi agenti poboljšavaju tijekom obavljanja samog problema. Kako bi nastavili učiti, ovi agenti su modelirani sa generatorom problema koji predlaže pothvate koji će dovesti do novih iskustava i saznanja.



Sl.2.4. Model učećih agenata

2.2.4. Okolina

Okolina je područje u kojem se agent(i) nalazi i u kojoj je potrebno rješavati problem. Ona ima ključnu ulogu pri definiraju inteligencije budući da je jedna od temeljnih karakteristika inteligentnih bića reagiranje na vanjske (okolišne) podražaje. Zato su u [12] dane dimenzije okolina na temelju kojih je moguće odrediti njihovu kvalitetu, te lakše odlučiti kakvog inteligentnog agenta treba odabrati.

- **Potpuna/djelomična vidljivost** - okolina u kojoj je moguće potpuno odrediti njeno stanje naziva se potpuno vidljivom okolinom, suprotnom naziva se djelomičnom vidljivom okolinom. U djelomično vidljivoj okolini najčešće se koristi vrsta memorije kako bi se sačuvale informacije o prošlim stanjima te na temelju njih stvorila potpuna slika okoline
- **Determinizam/stohastičnost** - ako je slijedeći stanje u potpunosti moguće odrediti na temelju trenutno stanja, tada govorimo o deterministčnoj okolini; inače o stohastičkoj
- **Epizodičnost/sekvencijalnost** - u epizodičnim okolinama, ne postoji ovisnost između prošle (i buduće) i trenutne radnje. Kod sekvencijalnih okolina, ta ovisnost postoji i ona može znatno utjecati na ponašanje agenta
- **Diskretnost/kontinuiranost** - okolina u kojoj je moguće postići konačan broj akcija smatra se diskretnom okolinom, dok okoline u kojoj akcije ne mogu biti numerirane smatraju se kontinuiranim okolinama
- **Statičnost/dinamičnost** - okolina koja ima sposobnost mijenjanja tijekom rada agenta je dinamična okolina; u suprotnom statična okolina
- **Jedan ili više agenata** - u okolini se može nalazi samo jedan ili više od jednog agenata. U okolinama s više agenata još postoji podjela na **natjecateljsku** ili **suparničku** okolinu. Agenti u natjecateljskoj okolini pokušavaju maksimizirati svoje performanse dok minimiziraju performanse ostalih agenata. U suparničkoj okolini agenti međusobno nastoje maksimizirati svoje performanse.

Commented [s4]: stohastičnost

Commented [s5]: slijedeći

3. *WUMPUSOV SVIJET*

Zasnovan na igri iz 1973. "Hunt the Wumpus", *Wumpusov svijet* je problem u polju umjetne inteligencije. Problem se sastoji od jednog agenta koji se kreće labirintom određenim pravilima kako bi pokupio zlato i vratio se na početak, bez da ne upadne u brazdu ili naiđe na živo čudovište - *Wumpusa*. Labirint je kvadratnog oblika raspodijeljeno na 4 retka i 4 stupca, dok u samom labirintu nalazi se 1 *Wumpus* i nasumičan broj brazdi. Mjesto na kojem se sijeku stupci i redci labirinta nazivaju se poljima (tako da postoji sveukupno 16 polja). Agent zna samo za polje na kojem se nalazi, polja kroz koje je već prošao i polja koja je logički zaključio. Kako agent ne bi slijepo padao u brazde i zabijao se u čudovišta, svaka od prepreka ima naznaku na susjednim poljima: na poljima koja su pored braze osjeti se propuh, dok na poljima koja su pored *Wumpusa* osjeti se smrad. Agent može pokušati ubiti *Wumpusa*, ali na raspolaganju ima samo jednu strijelu. Ako strijela pogodi *Wumpusa*, njegov vrisak se može čuti na bilo kojem polju.

Prema [12], [13], ovaj problem je moguće opisati u tri dijela: radnje agenta, vanjski podražaji i svojstva problema.

Kako bi agent uspješno obavio svoj zadatak, na raspolaganju ima određene akcije:

- **Rotacija u lijevo** - rotacija agenta u lijevo kako bi promijenio stranu u koju gleda
- **Rotacija u desno** - rotacija agenta u lijevo kako bi promijenio stranu u koju gleda
- **Pomicanje naprijed** - pomicanje agenta u smjeru kojem gleda
- **Gađanje strijelom** - ispaljivanje strijele u smjeru kojem agent gleda
- **Hvatanje** - uzimanje zlata s poda
- **Puštanje** - puštanje zlata na pod

Agent također na svakom polju može osjetiti navedene stvari:

- **Propuh** ako se u barem jednom od susjednih polja nalazi brazda
- **Smrad** ako se u jednom od susjednih polja nalazi *Wumpus*
- **Sjaj** ako se na polju nalazi zlato
- **Udarac** ako agent udari u zid
- **Vrisak** ako agent pogodi *Wumpusa* strijelom

Svojstva okoline *Wumpusovog svijeta* su:

- **Djelomična vidljivost** - agent od početka ne zna stanje cijelog labirinta

- **Determinizam** - labirint je konstruiran tako da se može predvidjeti koje akcije će agent poduzeti, tj. krajnji rezultat je već unaprijed poznat
- **Sekvencijalnost** - redoslijed obavljanja akcija je bitan
- **Statičnost** - lokacija *Wumpusa* i brazda se ne mijenja
- **Diskretnost** - prostor je podijeljen na diskretne cjeline po kojima se agent kreće - polja
- **Jedan agent** - sastoji je od samo jednog agenta koji se kreće i razmišlja. *Wumpus* se ne smatra agentom zbog statičnosti

Zašto se *Wumpusov svijet* smatra problemom umjetne inteligencije? Zato što agent na temelju podražaja iz polja mora odlučiti kako će se kretati ovisno o kombinaciji novih i starih podražaja. Agent u ovom problemu se smatra refleksivnim agentom temeljenom na modelu zbog potrebe pamćenja prošlih podražaja i zaključivanja o ostalim, neistraženim poljima. Iako ovaj agent ima cilj pronaći blago, ne može se smatrati agentom temeljenim na cilju budući da informacije o cilju postoje tek kad agent dođe na sam cilj, tj. agent se ne može orijentirati prema cilju i tako prilagoditi svoje ponašanje jer ne zna gdje je cilj sve dok ne dođe na njega.

Kako bi se ustanovila kvaliteta agentovog rješenja, određeni su bodovi koje je moguće sakupiti ili izgubiti tijekom rješavanja:

- +1000 bodova ako agent uzme zlato te izađe iz spilje
- -1000 bodova ako agenta pojede *Wumpus* ili agent upadne u brazdu (u oba slučaja umire)
- -10 bodova za korištenje strijele
- -1 bod za svaku ostalu radnju

Kraj je kada agent umre ili izađe iz spilje.

3.1. Logika *Wumpusovog svijeta*

U ovom radu koristit će se iskazna logika za rješavanja problema *Wumpusovog svijeta*. Postavljeni su iskazi $S_{i,j}$, $P_{i,}$, $Ru_{i,j}$, $Wu_{i,}$. Indeks i,j označava polje labirinta koji se nalazi u i-tom stupcu i j-tom retku. Iskazi su definirani:

- $S_{i,j}$ iskazuje da se na polju osjeti smrad;
- $P_{i,j}$ iskazuje da se na polju osjeti propuh;
- $Ru_{i,j}$ iskazuje da se u barem jednom od susjednih polja nalazi brazda;
- $Wu_{i,j}$ iskazuje da se u točno jednom od susjednih polja nalazi *Wumpus*;

Susjedna polja polja $Po_{i,j}$ su polja koja imaju zajedničku stranicu s poljem $Po_{i,j}$; odnosno to su polja $Po_{i+1,j}$, $Po_{i-1,j}$, $Po_{i,j+1}$ i $Po_{i,j-1}$. Na temelju zadanih iskaza i pravila *Wumpusovog svijeta*, moguće je definirati dva suda ključnih za zaključivanje na kojim poljima se nalaze opasnosti:

$$S_{i,j} \leftrightarrow Wu_{i,j} \tag{3-1}$$

$$P_{i,j} \leftrightarrow Ru_{i,j} \tag{3-2}$$

, koji iskazuju ako i samo ako se na polju osjeti smrad, tada se u točno jednom od susjednih polja nalazi *Wumpus* (3-1), te ako i samo ako se na polju osjeti propuh, tada se u barem jednom od susjednih polja nalazi brazda (3-2).

Ako na polju vrijedi da se osjete i smrad i propuh ($S_{i,j} \wedge P_{i,j}$), tada pomoću modus ponensa iz (3-1) i (3-2) dobije se sud:

$$(S_{i,j} \wedge P_{i,j}) \rightarrow (Wu_{i,j} \wedge Ru_{i,j}) \tag{3-3}$$

Također, ako na polju vrijedi da se u susjednom polju nalazi točno jedan *Wumpus* i barem jedna brazda $(Wu_{i,j} \wedge Ru_{i,j})$, tada pomoću modus ponensa iz (3-1) i (3-2) dobije se sud:

$$(Wu_{i,j} \wedge Ru_{i,j}) \rightarrow (S_{i,j} \wedge P_{i,j}) \tag{3-4}$$

, koji kada se konjugira sa (3-3), dobije se konačni sud:

$$(S_{i,j} \wedge P_{i,j}) \rightarrow (Wu_{i,j} \wedge Ru_{i,j}) \wedge (Wu_{i,j} \wedge Ru_{i,j}) \rightarrow (S_{i,j} \wedge P_{i,j}) \tag{3-5}$$

$$(S_{i,j} \wedge P_{i,j}) \leftrightarrow (Wu_{i,j} \wedge Ru_{i,j}) \tag{3-6}$$

, koji iskazuje ako i samo ako se na polju osjeti smrad i propuh, tada se u susjednom polju nalazi točno jedan *Wumpus* i u barem jednom od susjednih polja se nalazi brazda (3-6).

Vrijede i obratni slučajevi budući da su sudovi (3-1) i (3-2) ekvivalencije. Ako pretpostavimo da vrijedi $(\neg S_{i,j} \wedge \neg P_{i,j})$, tada iz (3-1) i (3-2) možemo zaključiti:

$$\neg S_{i,j} \leftrightarrow \neg Wu_{i,j} \tag{3-7}$$

$$\neg P_{i,j} \leftrightarrow \neg Ru_{i,j} \tag{3-8}$$

, odnosno ako i samo ako se na polju ne osjeti smrad, tada *Wumpus* nije niti na jednom od susjednih polja (3-7), te ako i samo ako se na polju ne osjeti propuh, tada nema brazda u niti jednom od susjednih polja.

Ako se na polju ne osjeti smrad niti propuh $(\neg S_{i,j} \wedge \neg P_{i,j})$, tada se pomoću modus ponensa i sudova (3-7) i (3-8) dobije sud:

$$(\neg S_{i,j} \wedge \neg P_{i,j}) \rightarrow (\neg Wu_{i,j} \wedge \neg Ru_{i,j}) \tag{3-9}$$

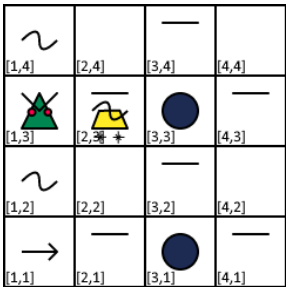
, koji nalaže da ako se na polju ne osjeti ni smrad ni propuh, tada u susjednim poljima nema ni *Wumpusa* ni brazdi.

Navedene sudove moguće je prikazati pomoću predikatne logike; ali zbog jednostavnosti i manjeg broja iskaza, korištena je iskazna logika.

3.2. Primjer rješavanja *Wumpsovog svijeta*

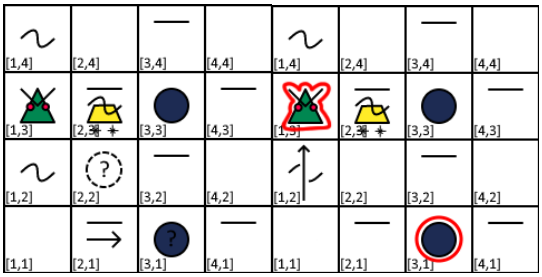
Na slici 3.1. dan je prikaz jedne konfiguracije *Wumpusovog svijeta*. Na slici agent je prikazan strelicom, *Wumpus* je prikazan zelenim trokutom, brazde su prikazane tamno plavim krugovima, a zlato je prikazano žutim trapezom. Podražaji što agent može osjetiti su smrad koji je prikazan valovitom crtom, propuh koji je prikazan ravnom crtom, i sjaj koji je prikazam dvjema zvjezdicama.

Agent uvijek započinje u donjem lijevom kutu (polje [1,1]) gledajući u desnu stranu (prema polju [2,1]).



Sl.3.1. Primjer jednog od mogućih rasporeda labirinta. Agent i smjer u kojem gleda je prikazan strelicom.

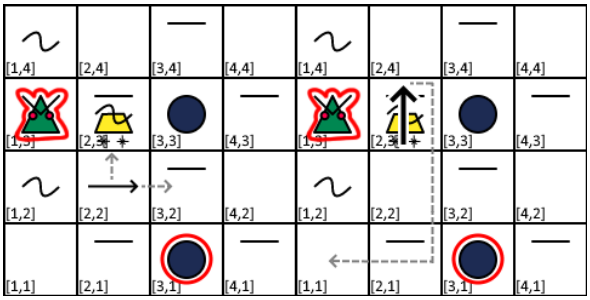
U ovom primjeru, agent na polju [1,1] ne osjeti nikakve podražaje, što znači da ne osjeti ni smrad ni propuh. Korištenjem suda (3-9) zaključuje da u susjednim poljima ([1,2] i [2,1]) nema nikakve opasnosti te da su sigurna za kretanje. Pomicanjem na polje [2,1], agent osjeća propuh te pomoću suda (3-2) zaključuje da se u barem jednom od susjednih polja nalazi propuh. Budući da nije siguran, agent se vraća na polje [1,1] potom nastavlja na jedino polje [1,2] za koje je siguran da nema opasnosti.



Sl.3.2. Prikaz agenta na polju [2,1] (lijevo). Agent zna da postoji barem jedna brazda, ali ne i gdje, stoga se vraća na jedino preostalo polje za koje zna da nema opasnosti (desno). Zbog prethodnih stanja, agent zna na kojem polju se nalaze *Wumpus* i brazda.

Na polju [1,2] agent osjeća samo smrad - zbog (3-1) to znači da se *Wumpus* nalazi u jednom od susjednih polja: [1,3] ili [2,2]. Agent zna da se *Wumpus* ne može nalaziti na polju [2,2] budući da na polju [2,1] nije osjetio smrad, te koristeći (3-7) zaključuje da se *Wumpus* ne može nalaziti na polju [1,3], stoga polje [2,2] nema opasnosti. Zbog toga što agent ima još polja za koje zna da se može kretati, agent odlučuje nastaviti na polje [2,2]. Na tom polju nema nikakvih podražaja te zbog (3-9) zaključuje da polja [2,3] i [3,2] su sigurna za kretanje. Ako agent nastavi na polje [3,2], osjetit će propuh. Budući da vrijedi (3-2) i može biti više od jedne brazde u labirintu, agent ne može zaključiti postoji li brazda na polju [3,3] ili [4,2] te odlučuje se vratiti na jedino sigurno polje [2,3]. Dolaskom na to polje, agent primjećuje sjaj, ne obazire se na ostale podražaje i uzima zlato. Agent nakon toga se vraća na polje [1,1]. Poznajući gdje se može kretati, te zbog statičnosti opasnosti, agent pronalazi najkraći put do polja [1,1] i izlazi iz labirinta sa zlatom.

Commented [s6]: Poznajući



Sl.3.3. Dolaskom na polje [2,2] agent se može kretati u jedno polje desno ili gore. Ako se pomakne desno, naići će na propuh, ali ne će znati koliko brazda ima u susjednim poljima (lijevo). Preostaje samo polje [2,3] bez opasnosti. Agent na polju pronalazi zlato te počinje povratak na početno polje [1,1] (desno).

4. LEGO MINDSTORMS

Prema [14] robotika je više-disciplinsko polje koja obuhvaća znanja iz mnoštvo drugih polja. Zbog toga, robotika može služiti kao poseban alat za sve razine školovanja i nauke. Jedan od pristupa robotici je pomoću Lego Mindstorms tehničke igračke koja je na tržište izašla 1. rujna 1998. godine, izrađena od strane danske tvrtke LEGO.

4.1. Svojstva Lego Mindstorms-a

Lego Mindstorms dolazi u par verzija. Ovaj rad se odnosi na EV3 31313 verziju Lego Mindstorma. Ova verzija sadrži senzore za: dodir, boju i infracrveno svjetlo. Senzor za dodir reagira na kontakt senzora s nekim predmetom. Senzor za boju omogućuje razlikovanje do sedam različitih boja na bliskoj udaljenosti. Svaka boja ima svoju enkodiranu vrijednost koje je moguće vidjeti u tablici 4.1.

Tab.4.1. Tablica preslikavanja boje u vrijednost.

Boja	Izlazna vrijednost
Bez boje[1]	0
Crna	1
Plava	2
Zelena	3
Žuta	4
Crvena	5
Bijela	6
Smeđa	7

Infracrveni senzor ima tri načina rada. Prvi način rada je mjerenje udaljenosti u kojem senzor pomoću ispaljenih i reflektirajućih infracrvenih valova može odrediti udaljenost između senzora i objekta. Udaljenost se iščita u ovisnosti o jačini reflektirajuće zrake. Najveća udaljenost koju ovaj senzor može iščitati je 100 centimetara. Drugi način rada je odašiljački način rada u kojem senzor

[1] Označava da senzor prilikom preslikavanja elektromagnetnih valova u naziv/vrijednost boje nije mogao pronaći odgovarajuću boju

može detektirati infracrvene odašiljače. Senzor može odrediti koliko je udaljen odašiljač, te je li odašiljač ispred senzora ili je pak s lijeve ili desne strane. Treći način rada je korištenje infracrvenog odašiljača kao daljinski upravljač. Senzor može na daljinu odrediti koja tipka ili kombinacija tipki su pritisnute na odašiljaču.

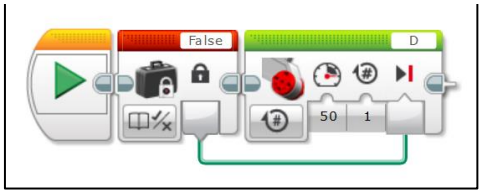
Komplet također sadrži aktuatore kako bi se robot mogao djelovati na okolinu: dva velika motora i jedan manji motor. Motori pretvaraju električnu energiju u rotaciju. Moguće je podesiti jačinu rotacije i broj stupnjeva što će se motori rotirati. Također je moguće podesiti da se motori vrte određeno vrijeme, no to nije poželjno ako se motori koriste za kretanje na specificirane daljine. Korištenjem oba velika motora daje robotu potrebnu mobilnost da se kreće okolišem pomoću četiri kotača ili dva lančanika.

4.2. Razvojno okruženje

Za programski dio Lego Mindstorms-a koristi se LEGO MINDSTORMS Education EV3 razvojno okruženje. Razvojno okruženje dolazi s uputama za izradu četiri tipa robota i desecima edukacijskih video isječaka.

Programira se pomoću vizualnog programiranja. Svaka naredba ili skup naredbi naziva se blok. Međusobno spajanje blokova radi na principu spajanja izlaza i ulaza putem žica koje prenose podatke od početka do odredišta. Sve varijable su globalne varijable, a njihovo čitanje i pisanje moguće je pomoću bloka varijabla. Moguće je skupiti više blokova u jednu cjelinu koja se naziva moj blok. Moj blok ima istu upotrebu kao funkcija kod ostalih programskih jezika, uključujući definiranje ulaza i izlaza, stoga će se u nastavku ovog rada za njih koristiti izraz funkcija ili funkcijski blok. Blokovi su podijeljeni u par kategorija: akcije, tok programa, senzori, obrada podataka, napredno i moj blokovi.

Commented [MK7]: razmak



Sl.4.1. Primjer EV3 programskih blokova. Podaci s jednog bloka se mogu slati pomoću žica (zelena crta) na drugi blok - od izlaza do ulaza. Blokovi se izvršavaju slijedno od lijeve strane prema desnoj, ali postoji mogućnost paralelizma.

Blokovi za akcije koriste se kako bi se direktno manipuliralo aktuatorima. Moguće je podesiti rotaciju jednog od motora, a kako su neke konfiguracije motora često korištene za kretanja poput vozila, dana su dva dodatna bloka koja upravljaju velikim motorima: blok za upravljanje poput auta i blok za upravljanje poput tenka. Oba bloka automatski koriste oba priključena motora kako bi se dobio željeni način kretanja robota. Osim korištenje malog i velikih motora, ovim blokovima može se ispisivati tekst ili slika na ekran, reproducirati zvučni zapis, i uključiti svjetlo na samom robotu.

Tok programa može se podešavati pomoću bloka za petlju, bloka za uvjet i bloka za čekanje. Blok za čekanje ne utječe na sam logički slijed programa već zaustavlja njegovo izvršenje na određeni broj sekundi. Blok za uvjet će izvršiti jedan dio koda ako je uvjet zadovoljen, inače će se izvršiti drugi. Kod koji se izvršava nalazi se u samom bloku te je moguće staviti bilo koju kombinaciju blokova unutra - uključujući novi blok za uvjet. Blok za petlju može raditi kao klasična petlja i izvršiti određeni broj ponavljanja ili dok je neki logički uvjet istinit. U samoj petlji je ugrađeno direktno čitanje senzora, tako da je moguće vriti petlju sve dok, na primjer, senzor svjetla očitava više od 100. Također može raditi beskonačno bez prestanka. Svaka programska petlja ima svoj naziv koji se može koristiti za njeno zaustavljanje. To je poželjno ako se naiđe na neku neočekivanu ili traženu vrijednost prije izvršenja svih iteracije petlje. Ali, zbog korištenja naziva petlje prilikom njenog prekidanja, moguće je prekinuti vanjsku petlju ako se petlja nalazi u petlji, ili prekinuti petlju u drugom dijelu programa. Prekidanje petlje u drugom dijelu programa je moguće zbog toga što je podržan paralelan način rada. Paralelizam se postiže tako što se koristi više od jednog start bloka na početku programa.

Senzorski blokovi služe za čitanje podataka sa senzora. Svaki senzora ima svoj posvećeni blok koji signale senzora može iščitati u valjanu vrijednost. Ako su potrebni čisti podaci, tada je moguće koristiti „čisti podaci" (engl. *raw data*) blok u naprednoj sekciji. Kako bi se smanjio broj korištenja blokova za obradu podataka, svaki od senzorskih blokova osim mogućnosti čitanja vrijednosti može odmah uspoređivati sa željenom vrijednošću.

Commented [MK8]: Ovo provjeriti

Blokovi za obradu podataka su standardni blokovi koji omogućuju čitanje podataka, uspoređivanje brojeva, validacija logičkih izraza, obradom nizova, dodavanjem teksta i odabiranjem nasumičnih brojeva.

Napredni blokovi su dodatni blokovi najmanje razine apstrakcije. Pomoću ovih blokova moguća je komunikacija među više EV3 robota, korištenje bluetooth mogućnosti, spremanja i čitanja iz tekstualnih datoteka, čitanja čistih podataka sa senzora i zaustavljanje cijelog programa. Ovi

blokovi su rjeđe korišteni budući da blokovi ostalih tipova rade ono što je u većini slučajeva potrebno.

Podržano je pet tipova podataka: broj, niz brojeva, logička vrijednost, niz logičkih vrijednosti i tekst. Brojevi su definirani u skupu realnih brojeva. Nizovi se ponašaju poput vektora u jeziku C: uvijek je moguće dodati element na kraj niza, tj. ne postoji programska granica koliko veliki niz može biti. Nizovi brojeva sadrže samo elemente brojčanog tipa. Logička vrijednost se može usporediti s boolean tipom podataka iz općih programskih jezika gdje vrijednost logičke varijable može biti samo ili *true* ili *false*. Logički nizovi sadrže samo elemente logičkog tipa. Tekst je tip podatka koji sadrži niz simbola. Tekst se može poistovjetiti sa *stringovima* iz općih programskih jezika s time da je rad nad tekstom dosta ograničenije. Moguć je samo pristup određenom simbolu teksta i spajanje više tekstova u jedan (engl. *concatenation*).

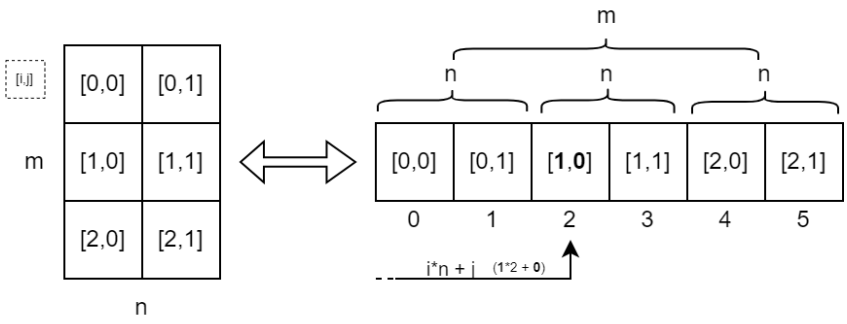
Commented [s9]: ?

Ne znam postoji li ova riječ
Možda koristiti neki sinonim

4.3. Tehnička ograničenja

Jedno od tehničkih ograničenja LEGO MINDSTORMS Education EV3 razvojnog okruženja je nedostatak direktne potpore za višedimenzionalne nizove (eng. *multi-dimensional arrays*). Budući da u razvojnom okruženju postoje nizovi kao vrsta podataka, moguće je izraditi i koristiti višedimenzionalne nizove. Stvaranje višedimenzionalnih nizova se temelji na tome da su ti nizovi reprezentirani kao jednodimenzionalni nizovi u računalnoj memoriji. Višedimenzionalni nizovi su nizovi koji kao svoje elemente sadrže nizove niže dimenzije: niz dimenzije N sadrži nizove dimenzije N-1. Nizovi dimenzije N-1 sadrže nizove dimenzije N-2 i ta rekurzija se nastavlja sve dok se ne dođe do niza dimenzije 1, tj. jednodimenzionalnih nizova. Niz dimenzije N tada opisuje N razina nizova.

Kod standardnih, jednodimenzionalnih nizova, pokazivač trenutne pozicije niza indeks i se povećava ili smanjuje za 1 kada je potrebno pristupiti susjednom elementu niza. Kod dvodimenzionalnih nizova (matrica), indeks i se mijenja za n, gdje je n veličina unutarnjeg jednodimenzionalnog niza, te mu se dodaje indeks j koji označava j-ti element unutarnjeg niza. Tako da, element koji se nalazi na polju i,j dvodimenzionalnog niza, nalazit će se na $i*n + j$ polju jednodimenzionalnog niza.



Sl.4.2. Prikaz ekvivalencije između matrice/dvodimenzionalnog niza i jednodimenzionalnog niza. Indeks *i* označava redak dok indeks *j* označava stupac matrice.

Još jedno ograničenje navedenog razvojnog okruženja je nedostatak rekurzija. Postoji mogućnost pravljenja vlastitih blokova/funkcija, ali ti blokovi/funkcije ne mogu pozivati sami sebe. Rješenja koja koriste rekurzije moguće je napraviti pomoću stogova, ali razvojno okruženje nema direktnu podržanost stogova. Stog je moguće napraviti, ali ne postoji jednostavan način budući da brisanje elemenata polja nije moguće.

5. IZRADA RJEŠENJA ZA RJEŠAVANJE *WUMPUSOVOG SVIJETA*

5.1. Tehničko rješenje

Commented [s10]: Kratki opis između naslova poglavlja i potpoglavlja

U ovom poglavlju daju se programska i tehnička rješenja inteligentnog agenta koji služi za rješavanje problema *Wumpusovog svijeta*. Također se opisuje konstrukcija okoline samog problema te način na koji su prikazani podražaji što agent može osjetiti.

5.1.1. Ploča

Pločice po kojima se robot kreće kodirane su po bojama. Svaka boja opisuje jednu moguću kombinaciju opasnosti ili stvari koje se nalaze na polju. Kodiranje po boji je opisano u tablici 4.1. Budući da senzor svjetla može razlikovati samo sedam boja, nije bilo moguće kodirati po boji sve kombinacije. Zbog toga je definirano da kada robot dođeW na polje sa zlatom, moguće je saznati da se samo zlato nalazi na njemu (iako to nije nužno slučaj). Ovaj pristup je uzet jer kada robot dođe na polje sa zlatom, ono je ispunilo svoj cilj, stoga ga ne zanimaju nalazi li se smrad ili propuh na tom polju. Nakon toga je potrebno vratiti se na početak. Povratak na početak zbog nedostatka osjeta na polju sa zlatom ne predstavlja problem zbog toga što je okolina statična.

Baza ploče, okvir i šesnaest polja su napravljena od dvije MDF (engl. *Medium Density Fiberboard*) ploče. Polja su izrezana u identične kvadrate kako bi bilo koja konfiguracija polja stala unutar okvira. Umetanjem svih polja osigurava stabilnost polja dok se agent kreće na njima. Svako polje sadrži jednu bijelu stranicu te jednu obojenu stranicu. Sveukupno ima 1 žuto polje, 1 smeđe polje, 3 crna polja, 3 crvena polja, 4 zelena polja i 4 plavih polja.

Sl.5.1. Slika ploče i polja po kojima će se agent kretati.

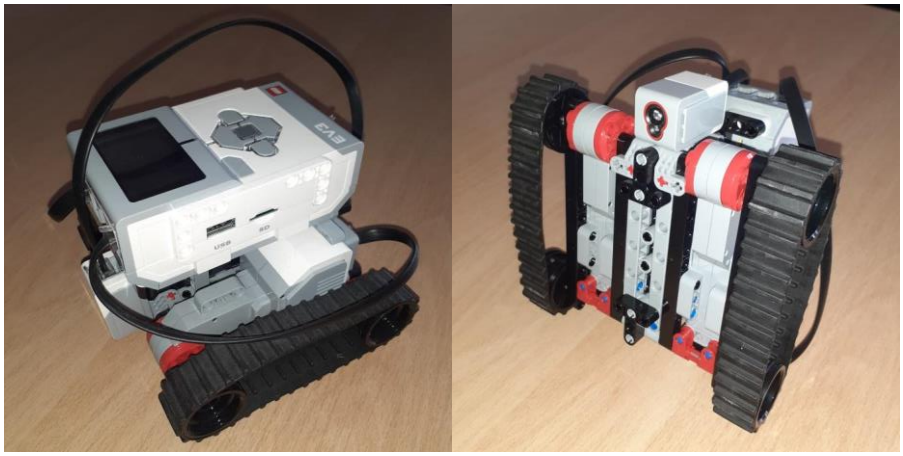
5.1.2. Agent

Inteligenti agent za rješavanje *Wumpusovog svijeta* napravljen je pomoću *Lego Mindstorms EV3* paketa. Kako bi agent mogao početi rješavati problem *Wumpusovog svijeta*, ponajprije mora imati sposobnost kretanja. U centru same konstrukcije se nalazi naredbeni blok koji služio kao mozak robota. Zbog svoje veličine i uloge, naredbeni blok se uzima kao mjerodavna veličina - konstrukcija ne može biti manja od naredbenog bloka.

Kako je hod na dvije ili četiri noge znatan zadatak, robot je konstruiran na izgled vojnog tenka. Za svrhu kretanja namijenjena su dva gumena lančanika - jedan s desne strane i jedan s lijeve strane naredbenog bloka. Osim jednostavne implementacije, gumeni lančanici pružaju stabilnost prilikom kretanja i rotaciju konstrukcije. Pokreti naprijed/nazad se dobiju kada se oba gumena lančanika kreću u istom smjeru, dok se rotacija oko težišta dobiva kada se gumeni lančanici kreću u suprotnim smjerovima.

Commented [s11]: ?

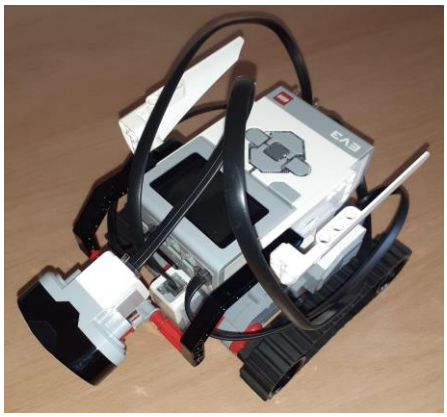
Prva inačica agenta, prikazana na slici 5.2., je imala mogućnost kretanja i prepoznavanja boja, ali nije podržavala slušanje vriskova. Slušanje u ovom kontekstu ne znači slušanje zvučnih valova pomoću senzora zvuka, nego očitavanje infracrvenih zraka koje odskaču od *Wumpusa*. Detaljnije o slušanju *Wumpusa* je opisano u poglavlju 5.1.3.



Sl.5.2. Prikaz prve inačice agenta. Konstrukcija je u obliku vojnog tenka radi jednostavnijeg načina kretanja. Detektor svjetla gleda dolje kako bi se mogla detektirati boja na ploči.

Zato druga inačica agenta, koja je nadogradnja na prvu inačicu, je imala tu mogućnost tako što je infracrveni senzor na prednju stranu agenta. Iako lagan, senzor je dovoljno težak da promjeni

centar mase agenta što je predstavljalo problem prilikom rotacije u stranu. Prva i druga inačica su imali problem slabe konstrukcije. Motori i lančanici nisu bili dovoljno čvrsto konstruirani da podnose sveukupnu težinu agenta. Događalo se da tijekom kretanja, ili čak stajanja, agent lagano propadne.



Sl.5.3. Prikaz druge inačice agenta. Ova inačica ima mogućnost slušanja *Wumpusovog* vriska, ali je zato centar mase znatno pomaknut.

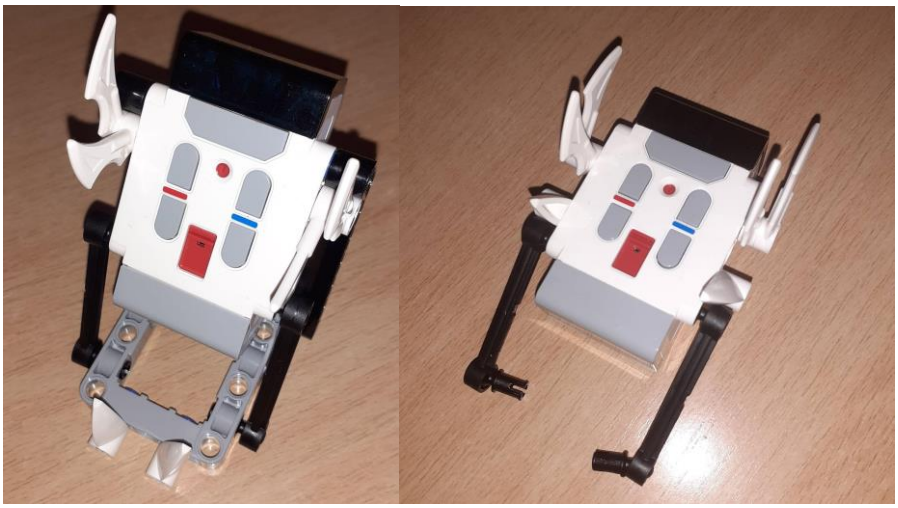
Zbog navedenih razloga, treća inačica agenta ima potpuno novu konstrukciju. Konstrukcija je još uvijek u stilu vojnog tenka, ali je napravljena dovoljno čvrsto da motori ne propadaju. Također, senzor za boju je premješten na stražnju stranu agenta, dok je infracrveni senzor ostao na prednjoj strani, ali bliže centru mase.

Sl.5.4. Prikaz treće inačice agenta. U usporedbi s drugom inačica, infracrveni senzor je još uvijek na prednjoj strani (lijevo), ali više ne utječe na centar mase. Senzor boja je premješten na stražnju stranu (sredina). Cijela konstrukcija je viša od prošlih inačica. Konstrukcija dna je jednostavnija (desno), ali puno čvršća u usporedbi s prošlim inačicama.

5.1.3. Wumpus

Iako *Wumpus* nije agent, dobio je vlastitu konstrukciju kako bi bilo moguće simulirati ispaljivanje strijele i vrisak. Prva inačica je infracrveni odašiljač s čudovišnim detaljima. Ovisno koji način rada koristi infracrveni senzor, infracrveni odašiljač može služiti kao daljinski upravljač ili kao vodilja. U ovom slučaju, odašiljač se koristi kao vodilja. Kada agent pokuša pogoditi *Wumpusa* strijelom, on zapravo s infracrvenog senzora odašilje infracrvene zrake. Odašiljać tada može uhvatiti te zrake te ih ponovno poslati prema senzoru. Ako se zrake vrate u senzor s odgovarajućim vrijednostima, tada agent može zaključiti da je pogodio *Wumpusa*; u suprotnom agent zaključuje da je promašio.

Druga inačica *Wumpusa* je proizašla zbog treće inačice agenta. Kako se infracrveni senzor na agentu nalazi bliže tlu, agent nije bio u mogućnosti precizno (ili uopće) registrirati signale prve inačice *Wumpusa*. Stoga, druga inačica je reducirana na ležanje na tlu kako bi senzor i odašiljač bili u istoj ravnini. Ova inačica funkcijski se ne razlikuje od prve.



Sl.5.5. Prikaz prve inačice *Wumpusa* (lijevo). Konstrukcija se sastoji od infracrvenog odašiljača i par detalja koji daju čudovišni izgled. Prikaz druge inačice *Wumpusa* (desno). Ova inačica je reducirana verzija prve inačice zbog toga što sam odašiljač mora biti bliže tlu.

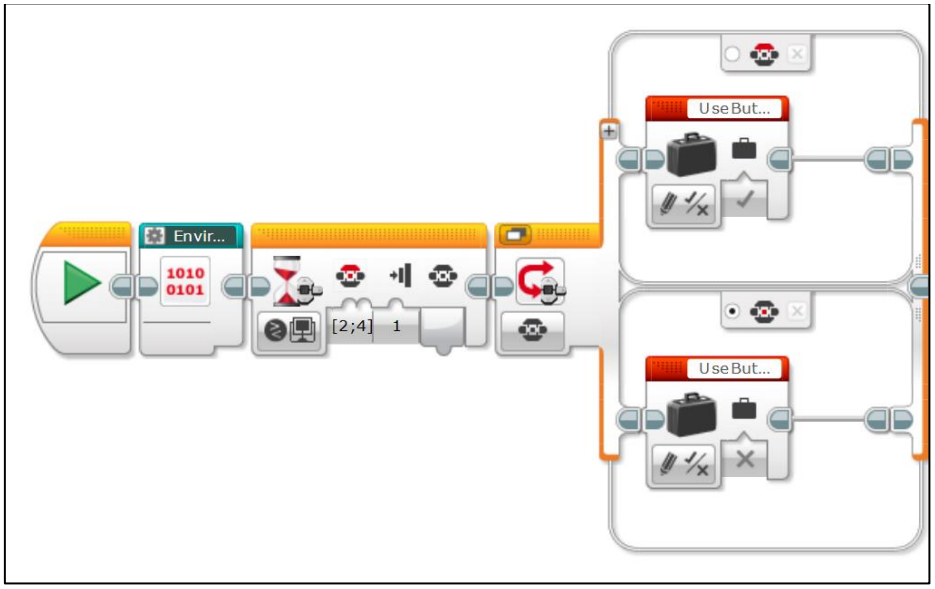
Commented [MK12]: Opet previše puta riječ Wumpus.

5.2. Programsko rješenje

Programsko rješenje se sastoji od vizualnog programiranja pomoću blokova u "LEGO MINDSTORMS Education EV3" razvojnom sučelju. Glavni dio programa se može podijeliti u tri dijela: postavljanje varijabli, glavna petlja i završne akcije. Sva napravljene funkcije korištene u ovom radu opisane su u prilogu P.5.2.

5.2.1. Postavljanje varijabli

Iako najmanji, ovaj dio je ključan za pravilno kretanje agenta i pomoć pri pisanju ostatka koda. Sastoji se od jednog bloka *EnvironmentSetup* koji u sebi sadrži naredbe za postavljanje varijabli na početne vrijednosti i konstanti, te naredbe za odabiranje načina rada agenta..



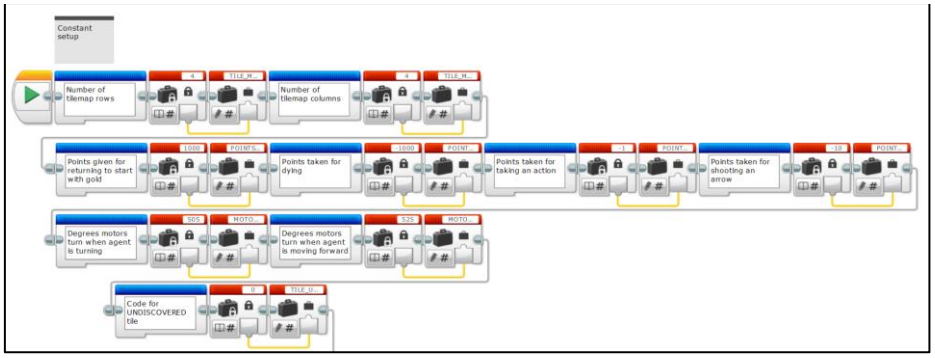
Sl.5.6. Prikaz blokova za postavljanje varijabli.

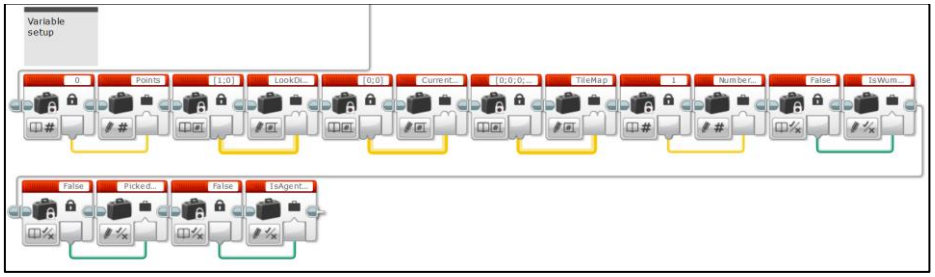
Konstante u ovom kontekstu su obične varijable koje je moguće naknadno mijenjati budući da u razvojnom okruženju ne postoje imenovane konstante. Postoji blok za kreiranje konstanti, ali njihovo korištenje je za definiranje zajedničkih vrijednosti u samo jednom dijelu koda - a ne u cijelom programu.

Commented [s13]: ostatka

Commented [MK14]: Ukositi bez navodnika,

Tako u čitavom radu za tekst između navodnika





Sl.5.7. i **5.8.** Djelomični prikazi koda bloka *EnvironmentSetup*. Kod je podijeljen u dva dijela: dio za postavljanje konstanti i dio za postavljanje varijabli na početne vrijednosti.

Konstante su definirane kako bi se pojedini podaci mogli jedinstveno koristiti te ako ih je potrebno promijeniti, moguće ih je promijeniti na samo jednom mjestu. U ovom dijelu se također definira način na koje se boje očitane senzoru preslikavaju u brojeve i što ti brojevi označavaju. Svaki broj označava stanje polja koje je agent mapirao. Detaljan opis preslikavanja je dan u tablici 3.1. U ovom dijelu se također mijenjaju vrijednosti okretaja motora, odnosno daleko će agent ići naprijed te koliko okretaja motora je potrebno da se agent okrene za 90°. Vrijednosti rotacije aktuatora potrebne da se dobije rotacija agenta za 90° i pomicanje jedno polje unaprijed dobivene su empirijski te iznose 505° za rotaciju i 832° za pomak unaprijed. navedene vrijednosti prilagođene su da valjano rade na već spomenutim pločicama. Površine koje nisu pločice mogu rezultirati pogrešnom rotacijom agenta zbog drugačijeg trenja između površine i agentovih lančanika.

Tab.5.1. Tablica preslikavanja vrijednosti u njihovo značenje i boje. Redci bez napisane boje ne mogu se direktno iščitati putem senzora, ali su ključna tijekom agentovog razmišljanja.

Vrijednost	Boja	Značenje
0	-	Neistraženo polje
1	Crna	Polje a brazdom
2	Plava	Polje s propuhom
3	Zelena	Polje sa smradom
4	Žuta	Polje sa zlatom
5	Crvena	Polje s propuhom i smradom
6	Bijela	Polje bez podražaja
7	Smeđa	Polje sa *Wumpusom*
8	-	Polje s brazdom ili *Wumpusom*
9	-	Neistraženo polje koje sigurno ne sadrži ni *Wumpusa* ni brazdu

U drugom dijelu *EnvironmentSetup* bloka nalazi se postavljanje varijabli na njihove početne vrijednosti. Tu se postavljaju početi uvjeti i stanja poput sveukupnog broja bodova, prvobitnu stranu na koju agent gleda i prikaz labirinta koju agent vidi/zaključuje, te zastavice za provjeru je li agent umro, je li *Wumpus* umro i je li zlato uzeto.

Prikaz labirinta je jednodimenzionalno polje od 16 elemenata. Koristeći tehnike opisane u poglavlju 4.3., to je zapravo 4x4 matrica. Ona opisuje svako realno polje što je agent očitao senzorima i logički zaključio. Svaki element ove matrice prvo se postavlja na 0, a njihova vrijednost može biti jedna od vrijednosti opisanih u tablici 5.1.

Nakon bloka za postavljanja varijabli, dolazi blok za postavljanje načina rada agenta. Agent može raditi u dva načina rada: kontinuirano ili po koraku. Kontinuirano se odabire pritiskom srednje tipke na agentu. U ovom načinu rada agent se kreće kroz labirint bez prestanka sve dok ne dođe do jednog od mogućih krajeva. Način rada po koraku se odabire pritiskom na gornju tipku agenta. U ovom načinu rada agent stoji nakon što učini jedan korak. Napominje se da nijedan korak nije nužno pomicanje iz jednog susjednog polja na drugo. Pod jedan korak se podrazumijeva i pomicanje iz jednog poznatog polja u nepoznato polje - koliko god da su udaljeni. Moguće je vidjeti kako agent percipira svijet nakon svakog koraka.

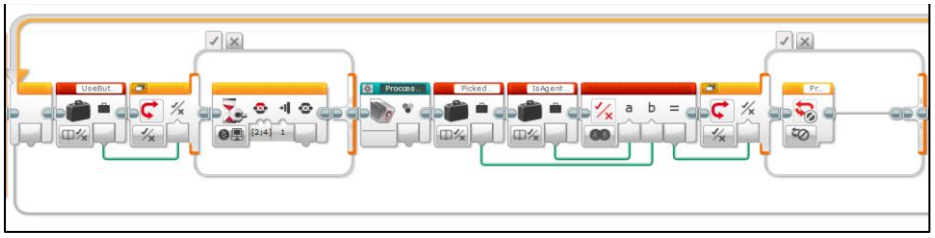
5.2.2. Glavna petlja

Glavna petlja programa može se svesti na sljedeći algoritam:

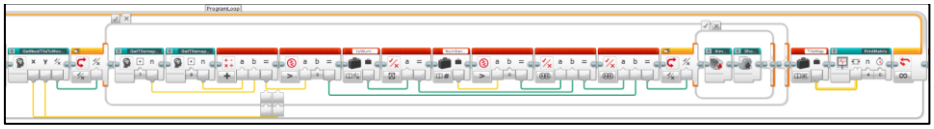
```
Sve dok nije KRAJ nakon svakog koraka radi:
      Iščitaj polje
      Ako se na polju nalazi Wumpus i Wumpus nije mrtav:
            KRAJ
      Ako se na polju nalazi brazda:
            KRAJ
      Inače ako se na polju osjeti sjaj:
             uzmi zlato i vrati se na početak i KRAJ
      Inače:
            Odredi koja polja su bez opasnosti
            Odredi koja polja su moguće opasna
            Ako postoji polje bez opasnosti:
                  Otiđi na to polje
            Inače ako imaš strijelu i Wumpus nije mrtav i poznato je
      gdje bi se Wumpus mogao nalaziti:
                  Orijentiraj se prema Wumpusu i ispucaj strijelu
                  Ako je Wumpus pogođen:
                        Označi sva potencijalna polja s Wumpusom kao
                  polja bez Wumpusa
                  Inače:
                        Označi sva potencijalna polja s Wumpusom u
                  smjeru ispaljivanja kao polja bez Wumpusa
            Inače:
                  Vrati se na početak i KRAJ
```

Program koristi razne funkcije kako bi izvršenje ovog algoritma mogli biti moguće.

Glavna petlje se može prekinuti u tri slučaja: smrt agenta (agent je ušao na polje s brazdom ili živim *Wumpusom*), uzimanje zlata, i nemogućnost da se pronađe put bez opasnosti. Posljednji slučaj je moguće jasnije definirati da agent ne vidi put bez moguće opasnosti te ako je *Wumpus* živ, agent nije pronašao potencijalno polje sa *Wumpusom* ili nema više strijela.



Sl.5.9. Prvi dio glavne petlje koji provjerava treba li se prekinuti. Neki od uvjeta prekida su smrt agenta ili pronalazak zlata.



Sl.5.10. Drugi dio glavne petlje koji traži sljedeće polje do kojeg se agent mora kretati. Ako ne postoji takvo polje, tada se gleda je li moguće ubiti *Wumpusa*.

Kako bi agent našao najkraći put po kojem se može kretati, implementiran je Dijkstrin algoritam za pronalaženje najkraćeg puta. Za samo kretanje, koristi sve dvije funkcijama *TurnAgent* *MoveAgent* koje rotiraju agenta za 90° u lijevu ili desnu strani i pomiču agenta za jedno polje u smjeru gledanja. Rotacija i translacija agenta vrši se rotiranjem elektromotornih aktuatora. Glavna razlika između navedene dvije akcije jest to što u slučaju rotacije agenta, dva elektromotora će se rotirati u suprotnim smjerovima, dok se rotiraju u istim smjerovima u slučaju translacije agenta naprijed.

U svakoj iteraciji petlje (i nakon što agent napravi korak naprijed), prvo se gleda na kojem polju se agent nalazi, tj. koje osjete može primijetiti. Osjeti su kodirani pomoću boje i mogu se iščitati iz tablice 5.1. Agent će, osim čitanja i pamćenja trenutnog polja, modificirati svoju umnu mapu labirinta ovisno o boji polja na koje je naišao:

- **bijelo** - neistražena susjedna polja (0) označava da su neistražena polja bez opasnosti (9)
- **plavo** - neistražena susjedna polja (0) označava da se na njima nalazi brazda (1). Ovdje se koristi pristup: “kriv dok se ne dokaže suprotno”; tj. smatra se da je opasnost u svakom polju dok se logikom ne dokaže da nije
- **zeleno** - ago agent već nije naišao na zeleno ili crveno polje, neistražena susjedna polja (0) označava da se na njima nalazi *Wumpus* (7); inače ih označava da su neistražena bez opasnosti (9), i sva ne susjedna polja sa *Wumpusom* (7) označava da su neistražena bez opasnosti (9), i sva ne susjedna polja sa *Wumpusom* ili brazdom (8) označava da se na njima nalazi brazda (1)
- **crveno** - ako agent već nije naišao na crveno ili zeleno polje, neistražena susjedna polja (0) označava da se na njima nalazi *Wumpus* ili brazda (8), inače ih označava da se na njima nalazi brazda (1), i sva ne susjedna polja sa *Wumpusom* (7) označava da su neistražena bez

Commented [MK15]: Ne susjedna

Commented [MK16]: isto

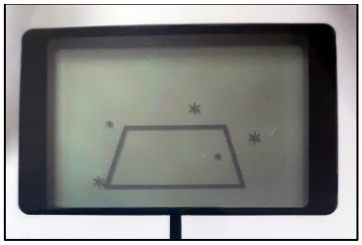
Commented [MK17]: isto

opasnosti (9), i sva ne susjedna polja sa *Wumpusom* ili brazdom (8) označava da se na njima nalazi brazda (1)

- **crno** - agent umire i prekida se glavna petlja
- **smeđe** - ako je *Wumpus* živ, agent umire i prekida se glavna petlja; inače susjedna polja označava da su neistražena polja bez opasnosti (9)
- **žuto** - agent uzima zlato i prekida se glavna petlja

Commented [MK18]: isto provjeriti sami dalje

Slučajevi za zeleno i crveno polje su kompliciraniji zbog toga što se u labirintu nalazi točno jedan *Wumpus*. Oba slučaja mogu se svesti na to da se *Wumpus* nalazi na presjeku susjednih polja svih polja sa smradom, te ako to vrijedi, nije moguće da *Wumpus* nije susjed barem jednom polju sa smradom. Ovaj sud implementiran je tako da se sva ne susjedna polja sa smradom označavaju bez *Wumpusa*.



Sl.5.11. Prikaz agentovog zaslona nakon što pokupi zlato. Prikazani simbol označava da je agent pokupio zlato i da ga drži na sebi.

Potencijalna greška može nastati ako agent dođe na polje s mrtvim *Wumpsom*, a u jednom od susjednih polja nalazi se brazda. Agent na tom polju ne će osjetiti propuh te zbog toga može upasti u brazdu. Greška nastaje zbog ograničenog broja vrijednosti što Lego Mindstorms senzor može razlikovati, stoga ne postoji dovoljno vrijednosti senzora da se preslikaju u sva moguća stanja jednog polja. U ovom slučaju bi bila potrebna boja koja označava da se na polju nalazi *Wumpus* i propuh.

Agent će tražiti *Wumpusa* ako ne može više naći put bez opasnosti, a poznato je barem jedno potencijalno polje s živim *Wumpusom* te agent ima barem jednu strijelu. Agent se pozicionira prema *Wumpusu* tako što će prvo naći jedno polje koje je u istom retku ili stupcu kao što je polje sa *Wumpusom*. Nakon što je polje pronađeno, agent se kreće prema tom polju pomoću funkcija *MoveAgent* i *TurnAgent* i Dijkstrinog algoritma za pronalaženje najkraćeg puta. Dolaskom na to polje, agent se okreće prema ciljanom polju te ispaljuje strijelu. Ako se čuje vrisak, agent postavlja

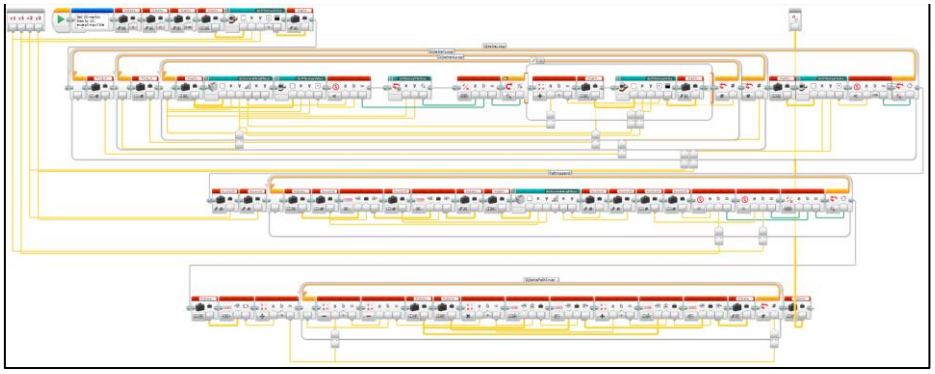
označava sva polja s *Wumpusom* (7) na neistražena polja bez opasnosti (9), inače označava samo polja s *Wumpusom* (7) u smjeru u kojem agent gleda.

Glavna petlja zaustavlja se ako agent ne može pronaći put bez opasnosti čak i kada je *Wumpus* mrtav.

5.2.3. Dijkstrin algoritam za pronalaženje najkraćeg puta

Kako bi se agent mogao lako kretati s jednog polja na drugo (koliko god da su udaljeno) bilo je potrebno definirati funkciju za pronalaženje najkraćeg puta. Za tu svrhu korišten je Dijkstrin algoritam za pronalaženje najkraćeg puta. On radi na principu da svake polje ima jednu vrijednost od 0 do ∞ koja označava udaljenost od početnog polja. U ovom kontekstu koristi se Manhattan udaljenost, a ne Euklidska. Manhattan udaljenost je udaljenost između dvije točke čija je vrijednost zbroj komponenti vektora udaljenosti između te dvije točke. Za početno polje se stavlja vrijednost 0. Za svako susjedno polje kroz koje je moguće kretanje stavlja se vrijednost trenutnog polja plus 1 ako je dobivena vrijednost manja od vrijednosti susjednog polja. Ovo se ponavlja za svako polje čija je vrijednost promijenjena, počevši od početnog polja. Izvršavanje traje sve dok se ne dođe do traženog polja - tada se putanja sprema od traženog do početnog polja tako da pravi put uzimajući uvijek susjedno polje s najmanjom vrijednosti. Dobiven put sadrži sekvencu od krajnjeg polja do početnog tako da je potrebno obrnuti redoslijed.

Dijkstrin algoritam je u ovom radu implementiran pomoću tri bloka petlje. Prva petlja označava broj iteracija - maksimalno 16 budući da je to maksimum što se agent može kretati. Druga i treća petlja se koriste za dohvaćanje pojedinačnog polja labirinta. Nad svakim poljem koristi se funkcija za dohvaćanje susjeda tog polja. Susjedi polja $W_{i,j}$ mogu se jednostavno dohvatiti jer su to polja $W_{i+1,j}$, $W_{i-1,j}$, $W_{i,j+1}$ i $W_{i,j-1}$, s time da $i \pm 1$ i $j \pm 1$ moraju biti unutar intervala [0,3]. Svakom susjedu stavlja se vrijednost polja $W_{i,j}+1$ ako je dobivena vrijednost manja od one koja je već na tom polju. Glavna petlja staje ako dođe do maksimalnog broja iteracija (u tom slučaju to označava da nije moguće doći od zadanog do traženog polja) ili traženo polje ima vrijednost različitu od ∞.



Sl.5.12. Umanjen prikaz implementacije Dijkstrinog algoritma. Implementiranje složenijih stvari u vizualnom programiranju može uzrokovati vizualni šum zbog većeg broja isprepletenih žica i blokova.

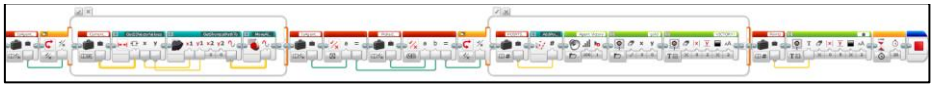
Ako je moguće kretanje do traženog polja, polja po kojima se agent mora kretati spremaju se u niz od traženog do početnog polja. Niz se popunjava tako da se nad trenutačnim poljem traže susjed s najmanjom vrijednošću. Koordinate tog susjeda se spremaju u niz te tada on postaje trenutno polje. Početno trenutno polje je odredišno polje. Iteracija od početnog do odredišnog polja nije moguća jer već u prvom koraku može biti više susjeda s istom najmanjom vrijednošću - dok to kod iteracije od odredišta to nije moguće. Kada petlja dođe do početnog polja, pokreće se nova petlja koja će niz koordinata samo obrnuti tako da prvi element bude početno polje, a zadnji element odredišno polje.

5.2.4. Završne akcije

Nakon što se prekine glavna petlja, na redu se izvršavaju završne akcije. Ovisno zbog čega je glavna petlja prekinuta, izvršit će se određena akcija:

- Agent je pao u rupu ili je agenta pojeo *Wumpus* - oglasit će se gubitnički zvuk i ispisati “DEFEAT” na ekran agenta
- Agent nije uspio naći polje bez opasnosti, a nema više strijela ili nije pronašao *Wumpusa* - agent se vraća na početničko polje, te ako ne upadne u rupu ili ga ne pojede *Wumpus*, oglasit će se gubitnički zvuk i ispisati “DEFEAT” na ekran agenta. Agent pronalazi put pomoću već opisanog Dijkstrinog algoritma za pronalazak najkraćeg puta.
- Agent je pokupio zlato - agent se vraća na početno polje, te ako ne upadne u rupu ili ga ne pojede *Wumpus*, oglasit će se pobjednički zvuk i ispisati “VICTORY” na ekran agenta

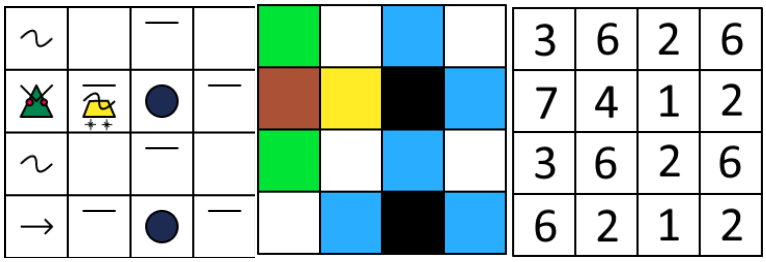
Nakon bilo kojeg slučaja is ekran se dodatno ispisuje broj bodova koji je agent sakupio. Ispis ostaje prikazan 20 sekundi nakon kojih agent potpuno izlazi iz programa.



Sl.5.13. Prikaz blokova u završnom dijelu programa.

Sl.5.14. Prikaz agentovog zaslona nakon što agent umre (lijevo) ili nakon što agent izađe sa zlatom (desno). U oba slučaja, ispisuje se broj postignutih bodova.

5.3. Prikaz rješenja



Sl.5.15. Prikaz problema koji se rješava (lijevo), dijagram realne ploče (sredina) i enkodirana matrica koju agent vidi ako je istraženo svako polje (desno).

Ovo je realni prikaz primjera rješavanja problema opisanog u poglavlju 3.2. Agent započinje u donje-lijevom kutu koji predstavlja polje [1,1].

0	0	0	0
0	0	0	0
9	0	0	0
6	9	0	0

Sl.5.16. Slika realnog agenta u prvom koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno). Iscrtani krug predstavlja trenutnu poziciju agenta.

Agent dolazi na polje [2,1] gdje osjeti propuh.

0	0	0	0
0	0	0	0
9	1	0	0
6	2	1	0

Sl.5.17. Slika realnog agenta u drugom koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno).

Agent se vraća na jedino polje bez opasnosti [1,2] gdje osjeti smrad.

0	0	0	0
7	0	0	0
3	9	0	0
6	2	1	0

Sl.5.18. Slika realnog agenta u trećem koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno).

Agent zaključuje da ni Wumpus ni brazda ne mogu biti na polju [2,2] stoga se tamo pomiče.

0	0	0	0
7	9	0	0
3	6	9	0
6	2	1	0

Sl.5.19. Slika realnog agenta u četvrtom koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno).

Od dva moguća puta, agent odabire put koji se prvi nalazi u nizu što opisuje labirint [3,2].

0	0	0	0
7	9	1	0
3	6	2	1
6	2	1	0

Sl.5.20. Slika realnog agenta u petom koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno). Iako se na polju [4,2] ne nalazi brazda, agent nema dovoljno dokaza da zna suprotno.

Agent na tom polju osjeća propuh, stoga se vraća na preostalo polje bez opasnosti [2,3] na kojem pronalazi zlato.

0	0	0	0
7	4	1	0
3	6	2	1
6	2	1	0

Sl.5.21. Slika realnog agenta u šestom koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno).

Agent uzima zlato te vraća se na početno polje [1,1]. Povratkom reproducira se pobjednički zvuk i ispisuje se ukupan broj bodova na agentovom ekranu.

0	0	0	0
7	4	1	0
3	6	2	1
6	2	1	0

Sl.5.22. Slika realnog agenta u sedmom i zadnjem koraku (lijevo) i prikaz labirinta kako ga agent u tom koraku percipira (desno). Prikaz labirinta je isti kao i u šestom koraku budući da agent nije pristupio novim poljima.

Agent ne radi na najučinkovitiji način zbog tehničkih ograničenja razvojnog okruženja. Ovaj manjak učinkovitosti se ponajviše vidi kada agent pronalazi sljedeće polje ili traži mjesto s kojeg može naciljati *Wumpusa*. Trenutna implementacija, budući da u suštini koristi niz za prikaz labirinta, ide slijedno od polja do polja kako su pozicionirani u nizu neovisno koliko je agent udaljen od njih. Korištenjem rekurzije moguće je eliminirati ovaj problem ako se rekurzivno gledaju samo susjedna polja.

6. ZAKLJUČAK

Inteligencija je pojam koji za sada nije moguće točno definirati niti u potpunosti opisati. Znamo da bića imaju različite razine inteligencije koje koriste kako bi osigurali svoj opstanak. Kao i u živoj prirodi, neživa inteligentna bića zvani inteligentni agenti, posjeduju različite razine umjetne inteligencije u ovisnosti po potrebi i svrsi. Neki agenti samo prate slijed instrukcija koje je potrebno izvršiti u ovisnosti o dobivenom podražaju, dok neki agenti razmišljaju koja akcija je bolja u određenim situaciji te kako će ta akcija utjecati na svijet oko agenta. Ali, zajednička stvar svih agenata je reagiranje na vanjske podražaje - reagiranje na svijet oko agenta.

Wumpusov svijet je igra namijenjena za inteligentne agente koji moraju zaključiti na temelju ograničenih informacija gdje se nalazi siguran put do zlata, tj. cilja. Jedna od mogućih realizacija agenata za Wumpusov svijet je pomoću Lego Mindstorms EV3 robota-igračke koji sadrži potrebne senzore i aktuatore kako bi agent mogao primati podražaje te reagirati na iste. Agent koristi pravila *Wumpusovog svijeta* kako bi pomoću logike zaključio gdje se nalaze moguće opasnosti na temelju podražaja što osjeti na trenutnom polju, ali i podražaje koje je osjetio već prijeđenim poljima.

Agent *Wumpusovog svijeta* je pokazatelj kako neživo biće može imitirati inteligenciju živih bića tako što je svjestan okoline u kojom se nalazi i pravila po kojoj se okolina ponaša. No, zbog svoje nemogućnosti da agent sam razluči pravila okoline, ne može se reći da je njegova umjetna inteligencija na razini prirodne inteligencije. Razvoj umjetne inteligencije se dalje temelji na sposobnosti da agent uči - da agent ulazi u okolinu nepoznatih pravila, ali da iskustvom nauči i razumije kako se ona ponaša.

LITERATURA

[1] Earl B. Hunt, *Artificial Intelligence*. New York, New York: Academic Press, 1975.
[2] S. Legg and M. Hutter, "A Collection of Definitions of Intelligence," in *Proceedings of the 2007 conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006*, Jun. 2007, vol. 157, pp. 17–24.
[3] L. Gottfredson, "The General Intelligence Factor," *Scientific American, Inc.*, vol. 9, pp. 24–29, 1998.
[4] E. S. kite (Trans.) and Baltimore: Williams & Wilkins, "New methods for the diagnosis of the intellectual level of subnormals," *The development of intelligence in children: The Binet-Simon Scale*, pp. 37–90, 1905.
[5] RJ Sternberg and W Salter, *Handbook of human intelligence*. Cambridge, UK: Cambridge University Press.
[6] Phillip C. Jackson, Jr., "Natural intelligence," in *Introduction to Artificial Intelligence*, Third edition., Mineola, New York: Dover Publications, Inc., 2019, pp. 5–28.
[7] Robert M. Yerkes, "The intelligence of earthworms.," *Journal of Animal Behavior*, vol. 2, no. 5, pp. 332–352, doi: 10.1037/h0072456.
[8] Nicholas J. Mulcahy and Josep Call, "Apes Save Tools for Future Use," *Science*, vol. 312, no. 5576, pp. 1038–1040, May 2006, doi: 10.1126/science.1125456.
[9] Björn C. Willige *et al.*, "D6PK AGCVIII Kinases Are Required for Auxin Transport and Phototropic Hypocotyl Bending in Arabidopsis," *The Plant Cell*, vol. 25, no. 5, pp. 1674–1688, May 2013, doi: https://doi.org/10.1105/tpc.113.111484.
[10] Mariusz Flasiński, *Introduction to Artificial Intelligence*. Jagiellonian University, Kraków, Poland: Springer, 2019.
[11] Margaret A. Boden, *Artificial Intelligence*, Second Edition. Academic Press, 1996.
[12] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall.
[13] JavaTpoint, "The Wumpus World in Artificial intelligence," *javaTpoint*. https://www.javatpoint.com/the-wumpus-world-in-artificial-intelligence (accessed Oct. 08, 2022).
[14] E. Afari and M. S. Khine, "Robotics as an Educational Tool: Impact of Lego Mindstorms," *International Journal of Information and Education Technology*, vol. 7, no. 6, pp. 437–442, Jun. 2017, doi: 10.18178/ijiet.2017.7.6.908.

SAŽETAK

U ovom radu daju se razne definicije inteligencije po kojima se temelji razvoj umjetne inteligencije. Definiraju se načela po kojoj inteligenti agenti rade, koje su glavne komponente koje omogućuju inteligentno ponašanje i okolina u kojoj agent mora djelovati. Kao primjer jedne okoline, istražuje se *Wumpusov svijet* i njegov agent. Opisuju se pravila *Wumpusovog svijeta* i logički sudovi koji proizlaze iz njih. Za stvaranje stvarnog agenta *Wumpusovog svijeta*, korištena je LEGO Mindstorms EV3 robot-igračka s pripadajućim senzorima i aktuatorima kako bi bila moguća interakcija s okolinom. Programski dio napravljen je pomoću razvojnog okruženja LEGO MINDSTORMS Education EV3 koje je namijenjeno za programiranje EV3 robota. Koristeći izvedene logičke sudove, agent se kreće kroz labirint izbjegavajući opasnosti i dolazi do zlata. Agent ne radi po najvećoj učinkovitosti zbog tehničkih ograničenja razvojnog okruženja, ali pokazuje kako umjetna inteligencija može oponašati prirodnu inteligenciju u načinu razmišljanja.

Ključne riječi: inteligentni agent, LEGO Mindstorms, logika, umjetna inteligencija, Wumpus

ABSTRACT

Wumpus World Using Mindstorm Robot

This graduate paper gives various definitions of intelligence by which artificial intelligence is being developed. Definitions are given for principles by which intelligent agents abide, the main components which enable intelligent behavior, and the environment in which the agent needs to act. *Wumpus' world* is taken as an example of one such environment. Rules are described for *Wumpus' world* which derive logical statements from them. For creation of *Wumpus' world* agent, LEGO Mindstorms EV3 robot-toy is used with appropriate sensors and actuators to make interaction with the environment possible. The programming is done using the LEGO MINDSTORMS Education EV3 programming environment intended for EV3 robots. By using the derived logical statements, the agent moves through the labyrinth avoiding danger and finds the gold. The agent does not work at maximum efficiency due to technical limitations of the programming environment, but it shows how artificial intelligence can imitate natural intelligence in the way of reason.

Key words: artificial intelligence, intelligent agent, LEGO Mindstorms, logic, Wumpus

Commented [s19]: Na novu stranicu prebacite anstract i oddamh ispod ubaciti
Wumpus World Using Mindstorm Robot
To običnim fontom samo bold

Commented [s20]: Graduate paper

Commented [s21]: By using - možda

ŽIVOTOPIS

Matej Dmitrović rođen je 18. prosinca 1997. godine u Osijeku. Pohađao je osnovnu školu August Šenoa. Nakon osnovnoškolskog obrazovanja, uspije se u Prirodoslovnu-matematičku gimnaziju, Osijek. 2014. osvaja 23. mjesto na državnom natjecanju iz logike. 2016. godine upisuje se na Fakultet elektrotehnike, računarstva i informacijskih tehnologija, Osijek smjer računarstvo koji završava 2019. godine te dobiva titulu sveučilišnog prvostupnika računalnog inženjerstva.

Matej Dmitrović

PRILOG 1

Video datoteke s rješenjima nalaze se na optičkom disku.

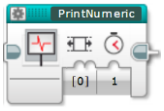
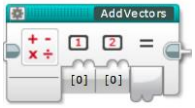
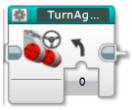
Video *wumpus_normal* prikazuje agenta kako uspješno skuplja zlato i vraća se na početno polje.

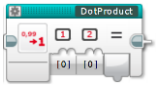
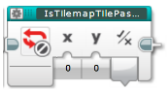
Video *wumpus_clash* prikazuje agenta kako koristi strijelu da bi saznao lokaciju Wumpusa, te uspješno skuplja zlato i vraća se na početno polje.

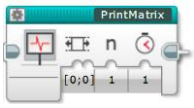
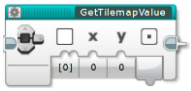
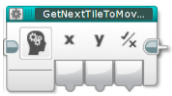
Video *wumpus_impossible* prikazuje agenta kako ne može pronaći siguran put te se vraća na početno polje bez zlata.

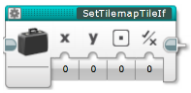
PRILOG 2

P.5.2. Tablica moj blok funkcijskih blokova korištenih u izradi ovog rada

Slika	Naziv	Opis
	*AddPoints*	Dodaje ulaznu vrijednost u ukupan zbroj bodova što je agent sakupio tijekom prolaska kroz labirint.
	*EnvironmentSetup*	Postavlja konstante i varijable na početne vrijednosti. Detaljnije opisan u poglavlju 3.2.1.
	*PrintNumeric*	Ispisuje broj ili niz brojeva na agentov ekran. Drugi parametar opisuje koliko dugo će ispis ostati na ekranu (u sekundama).
	*AddVectors*	Zbraja dva vektora s 2 elementa i rezultat daje na izlaz.
	*TurnAgent*	Agenta rotira u lijevo za 90° ako je na ulazu 1, ili u desno za 90° ako je na ulazu -1. Svako pozivanje agentu oduzima jedan bod.
	*MoveAgent*	Pomiče agenta za jedno polje u smjeru u kojem gleda. Svako pozivanje agentu oduzima jedan bod.
	*ProcessCurrentTile*	Agent očitava vrijednost sa senzora i ovisno o boji koju očitava radi određene akcije. Boju daje kao izlaznu vrijednost ukoliko se mora dalje koristiti.
	*ShootArrow*	Agent ispaljuje strijelu ako ima više od 0. Ako se *Wumpus* nalazi u smjeru pucanja, čuje se zvuk. Svako pozivanje s brojem strijela većeg od 0 oduzima agentu deset bodova.

	*GetLowestNeighbour*	Pomoća funkcija korištena prilikom Dijkstrinog algoritma za pronalaženje najkraćega puta. Prvi ulaz specificira matricu koja se koristi. Drugi i treći ulaz specificiraju koordinate polje oko kojeg se gledaju susjedna polja. Prvi izlaz daje vrijednost najmanjeg susjeda, dok drugi i treći izlaz daju koordinate tog susjeda.
	*GetShortestPathTo*	Funkcija koja implementira Dijkstrin algoritam za pronalaženje najkraćeg puta. Prva dva ulaza su koordinate od kojeg se počinje tražiti put, dok druga dva ulaza su koordinate odredišnog polja. Izlaz je niz uređenih parova koordinata kojim se mora kretati da se dođe do odredišnog polja.
	*InverseVector*	Svaki element vektora pomnoži sa -1.
	*DotProduct*	Računa skalarni umnožak dva vektora s dva elementa.
	*Get2DVectorValues*	Pomoćna funkcija za pristupanje elementima vektora s dva elementa.
	*MoveAlongPath*	Ulaz je niz uređenih parova koordinata polja po kojim se agent mora kretati. Agent se sekvencijalno kreće po svakom polju u nizu. Koristi funkcije “MoveAgent” i “TurnAgent” kako bi omogućio kretanje. Nakon svakog pokreta, poziva se funkcija “ProcessCurrentTile”.
	*IsTilemapTilePassable*	Ulaz su koordinate polja za koje se provjerava je li moguće kretanje po njemu bez opasnosti. Polje je sigurno za kretanje ako zadovoljava sljedeće uvjete: ● na polju ne postoji mogućnost da se nalazi brazda ● ako *Wumpus* nije mrtav, na polju ne postoji mogućnost da se nalazi *Wumpus*
	*GetNeighbouringTiles*	Kao ulaz prihvaća koordinate polja kojem se traže susjedna polja. Izlaz je niz uređenih parova koordinata polja koji su susjedi ulaznog polja. Niz može sadržavati maksimalno četiri uređena para, a

		minimalno dva.
	*PrintMatrix*	Ispisuje matricu danu prvim ulazom na ekran. Drugi ulaz označava broj stupaca matrice, dok treći ulaz označava koliko dugo će ispis biti prikazan na ekranu (u sekundama).
	*SetTilemapValu-e*	Upisuje vrijednost na polje specificirane matrice. Prvi ulaz je matrica u kojem se polje nalazi. Drugi i treći ulaz su koordinate polja. Četvrti ulaz je nova vrijednost tog polja. Izlaz je modificirana matrica.
	*GetTilemapVal-ue*	Dohvaća vrijednost polja specificirane matrice. Prvi ulaz je matrice u kojem se polje nalazi. Drugi i treći ulaz u koordinate polja. Izlaz je vrijednost tog polja.
	*GetNextTileTo-MoveTo*	Vraća koordinate sljedećeg polja do kojeg se agent mora kretati. Ako polje nije moguće pronaći, treći izlaz daje “false”, inače “true”. Polje nije moguće pronaći ako ne postoji neistraženo polje koje nema barem jedno susjedno polje bez opasnosti.
	*GetTilemapCo-unt*	Vraća broj polja labirinta koji imaju ulaznu vrijednost.
	*SetAllTilemapT-iles*	Postavlja sva polja labirinta na novu vrijednost ako imaju određenu vrijednost. Prvi ulaz je nova vrijednost polja. Drugi ulaz je vrijednost koje polje mora imati da bi se postavila nova vrijednost.
	*SetNeighbouri-ngTiles*	Postavlja sva susjedna polja trenutnog polja labirinta na novu vrijednost ako imaju određenu vrijednost. Prvi ulaz je nova vrijednost polja. Drugi ulaz je vrijednost koje polje mora imati da bi se postavila nova vrijednost.
	*GetFirstWump-usTile*	Vraća koordinate prvog polja labirinta na kojem se *Wumpus* moguće nalazi. Polja se gledaju u smjeru lijevo-desno gore-dolje.
	*RotateAgentTo-LookAtDir*	Rotira agenta tako da gleda u specificiranom smjeru. Smjer je vektor s dva elementa. Elementi smjera moraju biti cijeli brojevi.

	*AimAtWumpus*	Agent, ako može, pozicionira se na polje labirinta tako da gleda u smjeru polja s mogućim *Wumpusom*.
	*VectorsEqual*	Uspoređuje dva vektora. Vraća "true" ako su elementi prvog vektora jednaki elementima drugog vektora, inače vraća "false".
	*SetTileMapIfTil -e*	Postavlja vrijednost polja labirinta na novu vrijednost ukoliko to polje ima specificiranu vrijednost. Prva dva ulaza su koordinate polja. Treći ulaz je nova vrijednost polja. Četvrti ulaz je vrijednost polja koju polje mora imati da bi se postavila nova vrijednost.