Vladimir Makarić

Istraživanje nepoznatog terena

U radu se razmatra problem istraživanja nepoznatog terena (INT) od strane entiteta (istraživača) sa suženim vidnim poljem. Cilj rada je bio napraviti inteligentan entitet koji pronalazi putanju do zadatog cilja na terenu sa preprekama, te koji se potom vraća do početne tačke optimalnom (najkaćom) putanjom. Kao preduslov simulacije INT-a realizovani su sistemi za generisanje terena sa preprekama i detekciju sudara entiteta sa preprekom. Kretanje entiteta se zasniva na interiranju Dajkstrinog algoritma, koji se izvršava na grafu koji reprezentuje istraženi teren i trenutno vidno polje. Pri tome se u svakoj iteraciji u graf dodaju dva privremena čvora: jedan koji predstavlja trenutnu poziciju entiteta i drugi koji predstavlja odredište, a koji se povezuje sa čvorovima do kojih postoji pravolinijska putanja na kojoj nema poznatih prepreka (ili delova prepreka). Rezultati simulacije INT-a pokazuju da tako dizajnirano kretanje entiteta zadovoljava postavljene ciljeve projekta.

Uvod

Istraživanje nepoznatog terena (INT) je odlika ljudi i životinja koja je potrebna za pronalazak hrane, vode, skloništa.. Itd. INT predstavlja složen proces u kome entitet koji vrši istraživanje mora imati sposobnost vida, pamćenja i inteligencije. Vid omogućava formiranje slike o okolnom terenu koji se nalazi u vidnom polju entiteta. Ta slika se skladišti u pamćenje. Inteligencija služi da bi na osnovu poznatih karakteristika okolnog terena (pamćenja), entitet mogao donositi odluke o daljem istraživanju. Pored mogućnosti istraživanja nepoznatog terena entitet mora posedovati mogućnost efikasnog kretanja kroz poznat (istražen) teren, između ostalog da bi se nakon istraživanja mogao vratiti kući. Proces INT-a se završava kada je dati teren u celosti istražen to jest kada je moguće za bilo koje dve tačke, koje se nalaze u datom terenu, odrediti najefikasniju putanju od jedne do druge.

Cilj ovog rada je što približnija/realnija implementacija gore navedenih sposobnosti entiteta koje zajedno čine INT.

INT sistemi imaju primenu u sferi autonomnih robota gde se koristi SLAM (Simultaneous localization and mapping) tehnika za mapiranje i kretanje po nepoznatom terenu (web 1). U slične svrhe se koristi i D algoritam (dinamički) koji je implementiran na nasinim mars roverima (web 2; web 3). Svi ovi navedeni INT sistemi, uključujući i implementaciju INT sistema koja je tema ovog rada, kretanje ka cilju vrše sa pretpostavkom da se u nepoznatom terenu ne nalaze prepreke. Najveći problem INT-a je konstantno menjanje poznatog terena koje se dešava prilikom istraživanja, to jest problem je računati putanje do cilja u tom dinamičnom terenu a takođe je problem na efikasan način ga predstaviti u memoriji zbog njegove dinamične prirode.

Ova implementacija je više okrenuta ka virtulenom istraživanju, gde je teren parametar koji je u celosti poznat. Zadatak algoritama za istraživanje je da entitetu prosledjuju informacije samo o onom delu terena u kome se entitet nalazi, to jest samo o onom delu terena koji je istražio. Ova implementacija akcenat stavlja na što vernije predstavljanje istraženog terena i što prirodnije kretanje entiteta koje više

Vladimir Makarić (1993), Novi Sad, Neimarova 15, učenik 3. razreda Gimnazije „J. J. Zmaj" u Novom Sadu

MENTORI:
Dragan Toroman, ISP
Miloš Savić, MATF Novi Sad

podseće na kretanje živih bića nego na robotsko kretanje. Primena ovakve implementacije može biti u naučne shvre kao elemenat simulacije primitivne ljudske zajednice koja istraživanjem nepoznatog terena pronalazi resurse potrebne za opstanak ili kao elemenat game engine-a strategijske igrice.

Problem istraživanja nepoznatog terena je u ovom radu u određenoj meri apstrahovan. Teren u kojem se odvija „istraživanje" predstavljen je dvodimenzionalno. Entitet koji predstavlja čoveka koji vrši istraživanje apstrahovan je kao materijalna tačka. Vidno polje entiteta predstavlja krug opisan oko njegove pozicije. Kretanje je ograničeno na x i y osu. U konfiguraciju terena su uvedene prepreke u vidu složenih poligona koje imaju ulogu virtuelnih planina, reka i sličnih prirodnih tvorevina koje ljudima i životinjama otežavaju kretanje kroz prostor. Prepreke su predstavljene vektorski.

Metode

Grafički prikaz. Program je pisan u programskom jeziku C++, a za prikaz je odabran grafički interfejs, jer je preglednije posmatrati kretanje entiteta i istrazivanje terena vizuelno nego numerički. Za iscrtavanje grafičkih oblika koji predstavljaju elemente terena i entiteta koriste se OpenGL primitive.

Kao preduslov za INT razvijena su 2 sistema

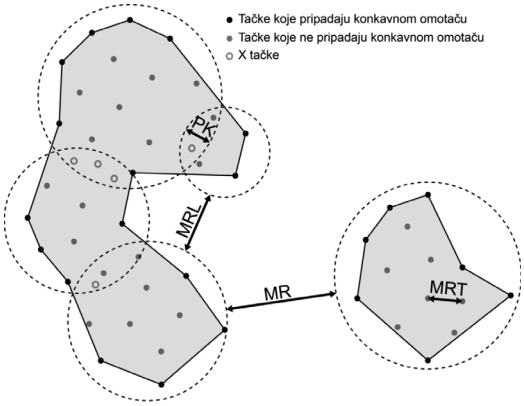
1. Proceduralni generator terena
2. Provera sudara unutar terena

Ovi sistemi nisu direktno vezani za INT i mogu biti primenjeni kao elementi raznovrsnih simulacija ili igrica.

Proceduralni generator terena

Da bi istraživanje bilo što realnije potrebno je da prepreke koje čine konfiguraciju terena imaju osobinu nasumične složenosti koja je karakteristična za prirodne tvorevine koje predstavljaju. Stoga je ručno unošenje prepreka nepraktično. Iz tog razloga je implementiran proceduralni generator terena. Proces PGT-a se sastoji iz tri faze. Prva faza je generisanje krugova na datoj površini koja predstavlja teren, krugovi mogu biti samostalni ili vezani u lance. Druga faza je nasumično popunjavanje krugova sa tačkama. Treća faza je povezivanje tačaka unutar krugova i lanaca krugova u poligone koji predstavljaju prepreke na terenu. Proces je ilustrovan na slici 1.

1. **Generisanje krugova**. Proces dodavanja novog kruga na teren počinje sa nasumicnim određivanjem njegovog prečnika između određene minimalne i maksimalne vrednosti. Svaki krug se ili veže za prošli dodati krug, ili se stvara samostalno u zavisnosti od evaluacije verovatnoće povezivanja. Ukoliko je verovatnoća velika, krugovi će biti pretežno povezani u lance (vezani jedni za druge), a ukoliko nije krugovi će se pretežno stvarati samostalno čineći mala „ostrvca" raštrkana po terenu. Ukoliko se krug stvara samostalno njegova pozicija se nasumično određuje u granicama zadatog terena, tako definisan krug sa određenim prečnikom i pozicijom mora biti na razdaljini većoj ili jednakoj određenoj minimalnoj razdaljini (MINR) u odnosu na ostale krugove. Ukoliko nije, generiše se nova pozicija i tako sve dok se ne nađe odgovarajući krug ili ukoliko se ovaj proces ponovi određeni broj puta, posle čega proces dodavanja novog kruga kreće ispočetka. Ukoliko se krug vezuje u lanac pozicija se generiše nasumično tako da budući krug preseca (ulazi u) krug za koji se veže za određenu dužinu (PK). Budući krug mora biti na razdaljini većoj ili jednakoj MINR samo u odnosu na krugove koji ne pripadaju tom lancu, a postoji i minimalna međusobna razdaljina (MINRL), koja se odnosi samo na članove lanca osim onog člana na kojeg se krug direktno veze. Ta razdaljina (MINRL) omogućava kontrolisanje savijanja lanca. Ukoliko je ta razdaljina mala, lanac ima slobodu da se savija, a ukoliko je velika, lanac će biti pretežno prav. Ukoliko bilo koji od dva testa razdaljine (MINR, MINRL) ne prođe, važi isto kao i za samostalno generisane krugove, jedina razlika je što se pozicija drugačije određuje. Ukoliko se verovatnoća povezivanja podesi na 100%, na terenu će se generisati jedan jedini lanac koji će da bude isavijan tako da popuni prostor na terenu. Postoji i parametar maksimalne veličine lanca, to jest maksimalan broj krugova. Ukoliko lanac dostigne maksimalnu veličinu a sledeći krug prođe verovatnoću vezivanja onda se nađe sledeći „slobodan" krug koji nije vezan u lanac pa se na njega poveže, ukoliko takvog nema nađe se slobodan lanac čija je veličina manja od maksimalne, a ukoliko ni takvog nema, krug se dodaje kao slobodan. Dodavanje krugova se završava ukoliko je određeni broj krugova dodat ili ukoliko je algoritam prešao određeni maksimalan broj neuspelih procesa dodavanja novih krugova, posle čega se smatra da na terenu više nema mesta za nove krugove.



Slika 1.

Figure 1.

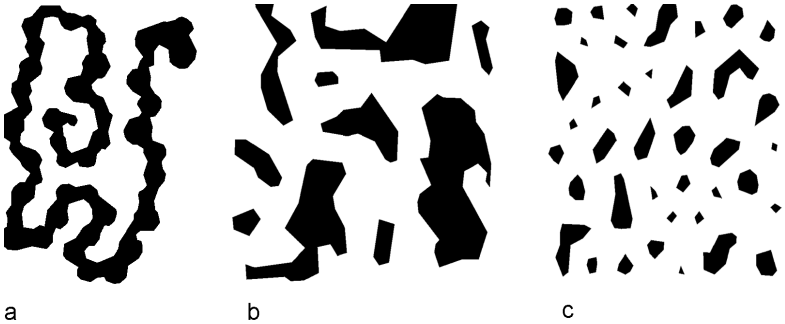
2. **Generisanje tačaka**. U svakom krugu se nasumično generišu tačke. Za svaku tačku kada se generiše proverava se njena razdaljina u odnosu na sve ostale tačke koje su dodate istom krugu; ta razdaljina mora biti veća ili jednaka miminalnoj razdaljini tačke (MRT). Ukoliko nije tačka se odbacuje. Taj proces se ponavlja određeni broj puta. Taj broj je jednak proizvodu određenog odnosa broja tačaka (OBT) i prečnika kruga. Na primer recimo da je OBT 1/20, tada bi se u krugu prečnika 100 generisalo 5 nasumičnih tacaka, od kojih neke mogu biti odbačene. Kada se svi krugovi „popune" tačkama, proverava se razdaljina između tacaka u svakom pojedinačnom lancu i izbacuju se one tacke koje su na manjoj razdaljini od MRT, do čega dolazi zbog međusobnog ulaženja „presecanja" krugova u istom lancu (nazvaćemo ih X tačke).

3. **Generisanje kompleksnih (konkavnih) poligona**. Algoritam koji je implementiran za nalaženje konkavnog omotača (web 1) ima parametar koji određuje složenost poligona, to jest broj najbližih tačaka koje uzima u obzir kada računa ivice poligona, ukoliko je parametar veći, poligon će biti manje slozen, ivice će biti veće, a ukoliko je parametar maksimalan, poligon će biti konveksan. Rezultat ovog algoritma je niz tačaka poređan tako da svaka tačka sa sledećom u nizu gradi stranicu, ove stranice su povezane jedna sa drugom i čine poligon, to jest prepreku.

Na slikama 2 (a, b i c) su primeri terena koji su generisani pomoću PGT-a. Na slici 2a je verovatnoća povezivanja 100% tako da se na tabli nalazi jedan jedini lanac, OBT je prilicno velik a MRT mali, stoga imamo veliki broj tačaka i lanac je detaljan, cak se vide i krugovi iz kojih je lanac generisan. Na slici 2b je verovatnoća povezivanja 80%, OBT je prilicno nizak a MRT veci, tako da imamo manji broj tacaka i oštrije prepreke sa dužim ivicama. Na slici 2c je verovatnoća povezivanja 20% stoga je teren sastavljen pretežno od većeg broja manjih preperka koje su generisane od tačaka koje su se nalazile pretežno u samostalnim krugovima.

Sistem za efikasnu proveru sudara

Ukoliko bi smo hteli da proverimo sudar nekog objekta X sa preprekama na terenu, morali bismo za svaku prepreku da proverimo sudar sa X, a to znaci da za svaku stranicu svake prepreke moramo da proverimo sudar, što je veoma sporo i zavisno od veličine terena, to jest broja prepreka. Da bi provera sudara bila efikasnija možemo da odredimo najmanji opisani krug oko svake prepreke. Onda bi smo prvo proveravali sudar sa opisanim krugovima, ukoliko nema sudara sa opisanim krugovima, onda možemo



Slika 2

Figure 2.

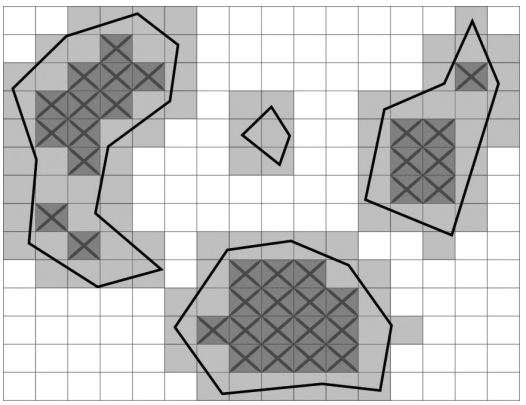
biti sigurni da ga nema ni sa preprekama. Ukoliko ima sudara sa opisanim krugovima, to ne mora da znači da ga ima i sa preprekama, pa moramo da proverimo sudar sa onim preprekama sa čijim opisanim krugovima je došlo do sudara; tako smo značajno ubrzali proces provere sudara zahvaljujuci pojednostavljivanju objekata u prostoru provere sudara. Iako smo ubrzali proveru sudara ona je i dalje zavisna od broja prepreka na terenu.

Pored pojednostavljivanja objekata, možemo da pojednostavimo i sam prostor. U ovoj implementaciji prostor je pojednostavljen tako što je implementirana matrica za brz pristup (MBP). MBP ima određeni broj redova i kolona, time je određena i visina i širina pojedinačnog polja u odnosu na dimenzije prostora u kojem se odvija provera sudara, u ovom slučaju teren u kome se odvija istraživanje. Svako polje MBP-a ima niz sadržanih ivica koje se nalaze u terenu kao delovi poligona, koje ga (polje) presecaju ili se u njemu nalaze. Pošto svaka ivica ima podatak o poligonu kojem pripada, svako polje takođe ima podatke o poligonima koji ga presecaju. Brz pristup znaci da za bilo koju poziciju x, y možemo da odredimo polje MBP-a koje toj poziciji odgovara. Polje je određeno kolonom i redom; kolonu dobijamo tako što x celobrojno podelimo sa širinom polja, a red tako što y celobrojno podelimo sa visinom. Na taj način iz bilo koje pozicije imamo brz pristup okolnim ivicama prepreka.

Da bismo odredili sudare sa X moramo odrediti sva polja MBP-a koja sadrže X, to znači sva polja koja X preseca kao i sva polja koja se nalaze unutar X. Kada smo odredili sva polja, tada iz njih izdvajamo sve ivice poligona, bez duplikata, u niz, jer vise polja moze da sadrzi u svom nizu ivica jednu istu ivicu. Na kraju ovog procesa imamo niz ivica koje proveravamo za sudar sa X, što je daleko efikasnije od proveravanja svih mogućih ivica i od metode sa opisanim krugovima oko prepreka. Najbolja strana MBP je što provera sudara ne zavisi od veličine terena i broja prepreka.

U ovoj implementaciji jedini objekti za koje je potrebno određivati sudar su duž i krug, što znači da su nam potrebni algoritmi za određivanje polja MBP koja sadrži krug i duž. Entitetov vidokrug je predstavljen krugom a svako geometrijsko telo koje nije oblo sastavljeno je od većeg broja duži, poput složenih poligona koji čine prepreke. Tako da je ovaj pristup dovoljan za veliki broj objekata.

Da bi provera sudara pomoću MBP-a bila moguća, za svako polje MBP-a moramo odrediti sadržane ivice. To možemo gledati i sa druge strane to jest možemo iskoristiti algoritam za određivanje polja koja sadrži duž, tako što ćemo za svaku ivicu svih prepreka odrediti sva polja koja ta ivica sadrzi, onda svim tim poljima dodati tu ivicu u niz sadržanih ivica. Kada se proces popunjavanja MBP-a zavrsi, tada se određuje tip svakog pojedinačnog polja (slika 3). Polje može biti jedno od tri tipa, presečeno (svetlo sivo polje, slika 3), što znači da niz sadržanih ivica nije prazan, a ukoliko je prazan tip može biti ili popunjen (polje sa iksom, slika 3), što znači da se polje u potpunosti nalazi u prepreci, u suprotnom polje je oznaceno kao slobodno (prazno, belo polje, slika 5). Kada određuemo tip polja, ukoliko nije presečeno nijednom ivicom, tada proveravamo dali se polje u potpunosti nalazi u nekoj prepreci, dovoljno je da proverimo centar polja za sadržanost u preprekama. Efikasnosti radi prvo odredimo u kojim se opisanim krugovima, koji odgovaraju preprekama, nalazi tačka sredine polja pošto više opisanih krugova mogu da se preklapaju. Posle proveravamo sa preprekama koje odgovaraju tim opisanim krugovima. Za proveru pripadnost sredine polja unutrašnjosti prepreke je implementiran ray casting algoritam za proveru pripadnosti tačke unutrašnjosti složenog poligona (web 2). Ukoliko provera prođe, polje označavamo kao popunjeno, u suprotnom kao slobodno.



Slika 3

Figure 3.

Recimo da hoćemo da proverimo da li postoji prav put od tačke A do tačke B, to jest da li duž od A do B preseca neke prepreke. Da bismo mogli da proverimo sudar sa preprekama, to jest njihovim ivicama, moramo prvo odrediti sva polja koja sadrži duž a onda proveriti sudar sa svim sadržanim ivicama tih polja, ali ukoliko je bilo koje od sadrzanih polja oznaceno kao popunjeno nemoramo da proveravamo sudar sa ivicama jer na osnovu tipa polja znamo da duz prolazi kroz prepreku i da je samim tim doslo do sudara. Time smo znatno ubrzali proveru sudara.

Određivanje polja koja sadrži duž. Ovaj algoritam ima određene sličnosti sa bresenhamovim algoritmom za crtanje duži na ekranu. Oba algoritma vektorski pojam duži rasteriziju u matrični prostor, bresenhamov u matricu piksela koja čini ekran a ovaj algoritam u MBP. Razlika je u tome što bresenhamov algoritam podrazumeva da se početak i kraj duži nalaze u centru piksela i podrazumeva da je piksel kockast. Razlika je takođe u tome što bresenhamov algoritam nema za cilj da odredi svaki piksel kojeg preseca duž, zbog toga što mu cilj nije provera sudara nego što verniji prikaz duži, dok ovaj algoritam zbog svoje svrhe mora odrediti svako presečeno polje. Algoritam je ilustrovan na slici 4.

Algoritmu se prosleđuje duž, dve pozicije. Za te dve pozicije se određuju dva polja MBP-a i određuju se leva i desna kolona i gornji i donji red na osnovu ta dva polja. Ukoliko su ta dva polja na istoj koloni ili u istom redu, algoritam vraća ta dva polja zajedno sa nizom polja koji povezuje ta dva polja bilo po koloni ili po redu. U suprotnom se određuje k parametar prave $y = kx$ čiji isečak predstavlja duž. Objašnjen algoritma koje sledi se ograničava na slučaj kada je k rastuće. Algoritam se „penje“ po duži od donje-leve ivice pa do gornje desne tačke duži, kada dođe do gornje desne tačke, tada algoritam vraća sva posećena polja.

Algoritam ima promenjive: trenutna kolona, trenutni red, prošla kolona, prošli red, uspon, pozicija penjanja, xPomeraj i yPomeraj. Vrednosti ovih promenljivih su na početku jednake: trenutna kolona i prošla kolona levoj koloni, trenutni red i prošli red donjem redu, uspon razdaljini po Y osi od donje tačke duži do donje ivice donjeg reda (PYP), pozicija penjanja donjoj levoj tački duži.

Kod svake iteracije se:

1. Trenutna kolona uvećava za 1

2. Pozicija penjanja se po X osi uvećava za xPomeraj, razdaljina do ivice sledece kolone, koja je jednaka širini polja (šP), osim u prvoj i poslednjoj iteraciji kada je jednaka PXP-u i KXP-u, odnosno.

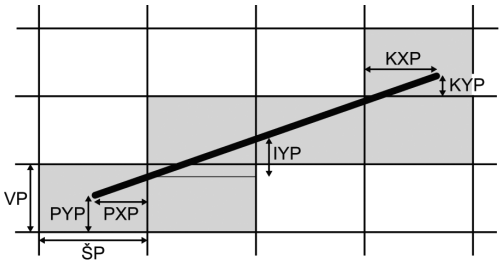
3. Pozicija penjanja se po Y osi uvećava za yPomeraj koji je jednak proizvodu xPomeraja i k parametra, vrednost yPomeraja je unapred izračunata (IYP) za sve iteracije kada je xPomeraj jednak širini polja, to jest za sve iteracije osim prve i poslednje kada je yPomeraj jednak PYP-u i KYP-u, odnosno.

4. Uspon se uvećava za yPomeraj, ukoliko vrednost uspona postane veća od visine polja (VP), to znači da pozicija penjanja nije više u trenutnom redu i tada se odvija proces označavanja posećenih polja:

- sva polja u trenutnom redu od poslednje kolone do trenutne se označavaju kao posećena.
- trenutni red postaje prošli red a novi trenutni red se izračunava tako što se celobrojnim deljenjem uspona sa visinom polja dobija broj pređenih redova, taj broj se dodaje na vrednost prošlog reda da bi se dobio trenutni red.
- sva polja u trenutnoj koloni od trenutnog reda pa do prošlog se označavaju kao posećena.

Tokom procesa označavanja posećenih polja, samo jedan broj, ili broj posećenih polja u trenutnoj koloni ili broj posećenih polja u trenutnom redu može biti veći od 1. Na primer (slika 4) kada duž ima nizak parametar k, proći će više iteracija dok uspon ne pređe visinu polja (usled malog yPomeraja), a kada pređe preći ce u sledeći red a dotle će algoritam proći veći broj kolona. Usled toga će broj posećenih polja (od prošlog reda do trenutnog) u trenutnoj koloni biti 1, a broj posećenih polja (od prošle kolone do trenutne) u trenutnom redu biti veći od 1.

Određivanje polja koja sadrži krug. Da bismo odredili polja koja sadrži krug moramo odrediti polja



Slika 4

Figure 4.

MBP-a koje preseca kružna linija kao i polja unutar kruga. Ukoliko je krug mali ili ukoliko su polja velika nema potrebe precizno određivati presek sa krugom. U tom slučaju možemo krug predstaviti kao kvadrat, na osnovu gornje leve i donje desne tačke kvadrata odrediti gornje levo i donje desno polje, i kao rezultat algoritma vratiti sva polja koja čine kvadrat definisam sa ova dva polja. U ovoj implementaciji drugaciji algoritam nije ni korišćen, iz razloga što bi precizan algoritam bio neefikasan/spor. Pošto postoji mogućnost greške, kada su krugovi veći, neka polja mogu biti van kruga, i stoga njihove sadržane linije ne bi presecale krug. Iz tog razloga pri određivanju jedinstvenih ivica koje sadrži krug, ivice bi se testirale za pripadnost krugu pomoću provere sudara opisanog kruga oko ivice i kruga, ni ovaj test nije definitivan ali odbacuje određeni broj slučajeva, definitivan test je test preseka duži i kruga koji se radi samo ako je to neophodno pošto je spor.

Inteligencija Entiteta (IE). IE kao parametar prima cilj, poziciju koja se nalazi na terenu do koje treba doći. IE se izvršava svake iteracije i vraća vektor brzine konstantnog intenziteta koji se dodaje na entitetovu poziciju, tako se postiže kretanje. Entitet svoje kretanje pamti kao uređeni niz tačaka (putanja entiteta) nova tačka se dodaje svaki put kada se promeni vektor brzine.

Prepreke u terenu su predstavljene vektorski kao niz tačaka. Ovakva predstava nije idealna za algoritme IE, tako da svaka prepreka ima duplo linkovanu listu nodova koji predstavljaju tačke poligona. Svaka prepreka takođe ima linkovanu listu ivica. Svaka ivica sadrži 2 noda iz liste nodova, koji je određuju. MBP u svojim poljima sadrži ove ivice. Svi nodovi i ivice svih prepreka su smešteni u niz nodova i niz ivica, odnosno. Svaki nod i svaka ivica ima redni broj koji odgovara njegovoj/njenoj poziciji u nizu. Oba niza su veličine broja nodova koji je isti kao i broj ivica.

Inteligencija entiteta se sastoji iz dve celine. Prva je struktura podataka koju ćemo nazvati POT (podaci o terenu) koja sadrzi sve informacije o istraženom terenu. Ova struktura se ažurira svake iteracije, analizirajući okolni neistražen teren u vidokrugu entiteta. Drugu celinu predstavlja logička struktura koju ćemo nazvati Menadžer Putanje (MP), MP na osnovu informacija iz POT, generiše putanju do cilja i svake iteracije je ažurira u skladu sa novim informacijama iz POT ukoliko postoje.

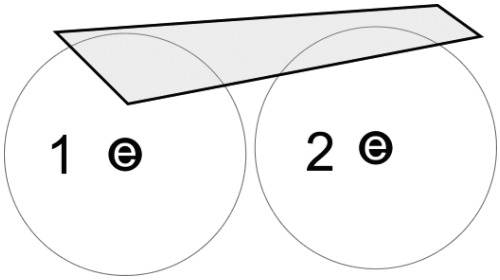
POT (podaci o terenu). Kada entitet istražuje on prolazom kroz teren vidi prepreke, ali ne vidi uvek cele prepreke, nego u većini slučajeva vidi samo jedan njihov deo. Delovi prepreka su opisani lancima (linkovanim listama) POT nodova (PNOD). PNOD-ovi su specijalni nodovi kojima POT opisuje vidljiv teren. Postoji dva tipa PNOD-a, ovde će biti reč o OPNOD-u, nodu koji odgovara „originalnom“ nodu sa prepreka. Kada entitet vidi neki nod na terenu prvi put, POT u niz OPNOD-ova doda novi OPNOD koji odgovara viđenom nodu, a viđeni nod se označava kao originalni nod novog OPNOD-a. Novi OPNOD se povezuje za okolne ranije dodate OPNOD-ove i tako se kreiraju lanci koji predstavljaju viđene delove prepreka. Svaki OPNOD ima dve veze sa OPNOD-ovima (prošli i sledeći). Obe ove veze ne moraju biti popunjene, što omogućava vezivanje u otvorene lance gde je poslednji i prvi član lanca vezan samo za jedan OPNOD, za razliku od običnih nodova koji su povezani u zatvoren lanac koji predstavlja prepreku.

POT sadrži OPNOD tabelu koja za svaki nod vraća odgovarajuci OPNOD, ukoliko postoji, to jest ukoliko je dodat. Ova tabela nam omogućava da znamo koji nod smo ranije videli a koji nismo, jer kada vidimo neki nod prvi put, dodajemo novi OPNOD koji njemu odgovara u niz OPNOD-ova. Taj novi OPNOD dodajemo i u tabelu da ga ne bismo dodali u niz OPNOD-ova ponovo. Ova tabela je običan niz OPNOD-ova, koji odgovara nizu svih nodova, elemenat ovog niza može biti prazan što označava da taj nod nije viđen ili može sadrzati OPNOD što znači da je taj nod već viđen. Pojedinačnom elementu ove tabele se pristupa pomoću rednog broja noda koji hoćemo da proverimo, to jest da proverimo dali smo ga ranije videli.

Problem sa ovakvim pristupom pamćenja istraženog terena su situacije 1 u 2 na slici 5, gde entitet ili vidi samo jedan nod ili ne vidi ni jedan i nema načina da sačuva informacije o ivicama koje mu se nalaze u vidokrugu. Ovaj problem je rešen tako što je implementirana tabela za ivice takođe, slična kao i za nodove, stim što svako polje ove tabele koje predstavlja jednu ivici na terenu može imati tri vrednosti:

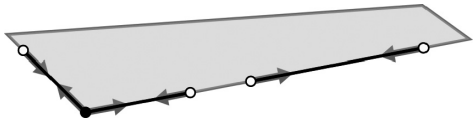
- neistražena (entitet nikada nije video tu ivicu)
- potpuno istražena (entitet je video tu ivicu u celosti)
- ne potpuno istražena, ova vrednost nosi sa sobom niz druge vrste PNOD-ova, presečnih PNOD-ova (PPNOD-ova), koji predstavljaju delove ivice koje je entitet video. Ovi nodovi

ne odgovaraju nodovima na terenu nego su dodati na presecima entitetovog vidokruga. Ovi nodovi imaju samo jednu vezu sa drugim PNOD-ovima (mogu se vezivati za oba tipa PNOD-a), zato što opisuju delove ivice koji su istraženi. Pored toga na jednoj ivici nema ćoškova pa nema potrebe za 2 veze. PPNOD--ove na slici 9 predstavljaju prazni krugovi, dok puni krugovi predstavljaju OPNOD-ove. Sive strelice predstavljaju veze između PNOD--ova. Kao što se vidi sa slike svi PNOD-ovi su povezani tako da predstavljaju istražene delove prepreke (tamno crna linija), ovi istraženi delovi odgovaraju slučajevima 1 i 2 na slici 8 i predstavljaju rešenje problema.



Slika 5.

Figure.



Slika 6.

Figure 6.

Ažuriranje PNOD-ova

1. Pomoću MBP-a se odrede sve ivice i svi nodovi koji se nalaze u vidokrugu entiteta.

2. Svi nodovi čiji odgovarajući OPNOD-ovi (koje određujemo pomoću tabele) imaju samo jednu vezu (krajevi lanca) i nodovi koji nisu dodati, oni prvi put viđeni, se proveravaju za vidljivost. Pošto ako pripadaju vidokrugu to ne znači da su i vidljivi. Mora se proveriti sudar terena sa duzi od entiteta do nodova.

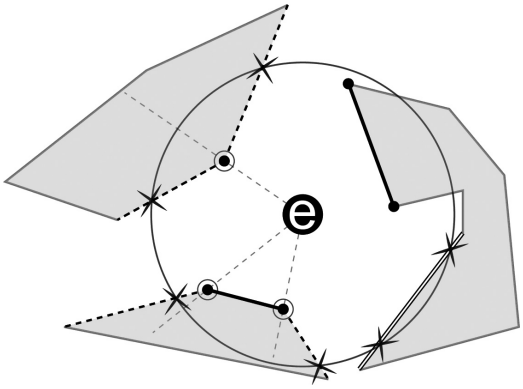
4. Na osnovu svakog noda koji prođe test vidljivosti a nije dodat (prvi put viđen), se kreira OPNOD koji se dodaje u tabelu i u niz svih OPNOD-ova. Takođe se dodaje u niz novih OPNOD-ova.

5. Svaki novi OPNOD se povezuje tako što se proveravaju susedni nodovi njegovog originalnog noda. Ukoliko OPNOD-ovi koji odgovaraju susednim nodovima postoje i ukoliko su prošli test vidljivosti oni se povezuju sa novim OPNOD-om. Krajevi lanaca su proveravani za vidljivost (korak 2) da bi novi OPNOD-ovi mogli da se povežu sa starima.

6. Nađu se vidljivi krajevi lanaca koji su OPNOD--ovi (pošto kraj lanca može biti i PPNOD), (moraju se ponovo tražiti jer je moguće da su dodati novi OPNOD-ovi) čiji originalni nodovi nisu ivični nodovi u odnosu na pravu od pozicije entiteta (sive isprekidane duzi, slika 7). Ovi OPNOD-ovi su na slici 10 predstavljeni crnim punim krugovima sa praznim krugom oko sebe.

7. Svaka ivica čija dva OPNOD-a (koje pomoću tabele dobijamo od njena dva noda) su povezana se označava kao potpuno istražena u tabeli ivica (debele crne duži, slika 7).

8. Od ivica koje se nalaze u vidokrugu a nisu potpuno istražene izdvoje se one čiji jedan OPNOD (koji odgovara jednom nodu od date ivice) pripada



Slika 7.

Figure 7.

OPNOD-ovima opisanim u koraku 6 (crne isprekidane duži, slika 10). Te ivice se presecaju sa vidokrugom, i taj presek je vidljiv entitetu (krstevi na crnim isprekidanim dužima, slika 10). Tada se na mesto preseka kruga i te ivice dodaje novi PPNOD koji se vezuje za vidljivi OPNOD ivice. Ukoliko je ivica neistražena (ukoliko na njoj nema ni jedan PPNOD).

9. Ostale ivice u vidokrugu čiji ni jedan nod nije vidljiv i koje nisu potpuno istražene (duž sa duplom crnom linijom, slika 7) se testiraju za presek sa vidokrugom (pošto ne moraju biti u vidokrugu, zbog neprecizности MBP-a). Ukoliko se nađu dve tačke preseka sa vidokrugom (dva krsta na duploj duži, slika 10) tada se u tim tačkama dodaju dva PPNOD-a međusobno povezana, ukoliko je ivica neistražena.

Dodavanje PPNOD-ova je opisano samo u slučaju kada su presečene ivice ne istražene u koraku 8 i 9 ažuriranja POT-a, u suprotnom kada ivice nisu potpuno istražene moraju se uzeti u obzir PPNOD-ovi koji na toj ivici već postoje. Ovaj problem je ilustrovan primerima na slici 11, crne tačke predstavljaju OPNOD-ove, bele tacke PPNOD-ove, sive strelice veze između njih, crna linija istraženi deo ivice, siva ne istraženi, krstovi presečne tačke.

Kao i sa neistraženim ivicama, postoje dva slučaja:

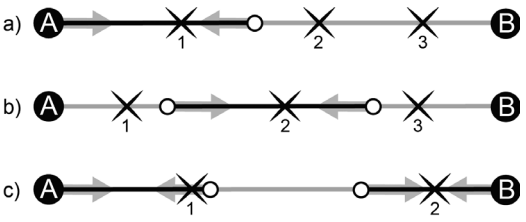
1. Kada postoji samo jedna presečna tačka i jedan vidljiv nod ivice. U ovom slučaju entitet vidi sve od presečne tačke pa do vidljivog OPNOD-a ivice, to mu se nalazi u vidokrugu. Prvo se označe svi PPNOD--ovi na ivici koji se nalaze u vidokrugu (od presecne tačke do vidljvog OPNOD-a), oni se na kraju brišu. Onda se proveravaju veze tih PPNOD-ova. Svaki ima jednu vezu kojom se povezuje za drugi PNOD nod (PPNOD ili OPNOD). Ukoliko bilo koja ta veza, drugi PNOD, nije u vidokrugu to znaci da neće biti dodat novi PPNOD, nego će taj drugi PNOD biti vezan za vidljivi OPNOD ivice. Primer pod b, vidljivi OPNOD ivice je A, presečna tačka je 2. Levi PPNOD je u vidokrugu, ali njegova veza (desni PPNOD) nije, tako da se njegova veza povezuje sa A OPNOD-om, a levi PPNOD nod se briše.

Ukoliko je veza nekog vidljivog PPNOD-a OPNOD ivice koji nije vidljiv, onda se ivica označava kao istrazena i svi PPNOD-ovi se brišu. Primer pod a, vidljivi OPNOD ivice je B, presečna tačka je 1. Jedan PPNOD je u vidokrugu a njegova veza je drugi OPNOD ivice.

Ukoliko su sve veze od vidljivih PPNOD-ova u vidokrugu tada se dodaje novi PPNOD povezan za vidljivi OPNOD ivice. Primer pod b, OPNOD ivice je B, presečna tačka je 1.

Ukoliko ni jedan PPNOD nije u vidokrugu onda se jednostavno dodaje novi PPNOD na mestu preseka i povezuje sa vidljivim OPNOD-om ivice. Ovo ne važi u slučaju da je neki PPNOD na ivici (koji nije vidljiv) povezan za vidljivi OPNOD ivice. Tada ne dolazi ni do kakve promene pošto entitetov vidokrug preseca već viđeni, istraženi deo ivice. Primer pod a, vidljivi OPNOD ivice je A, presečna tačka je 1.

2. Kada postoje dve presečne tacke a ni jedan nod ivice nije vidljiv. Entitet vidi sve između dve



Slika 8.

Figure 8.

presečne tačke, to mu se nalazi u vidokrugu. Prvo se označe svi vidljivi PPNOD-ovi (oni između dve presečne tačke), oni se na kraju brisu. Proveravaju se veze tih PPNOD-ova. Ukoliko bilo koja veza (PNOD) nije vidljiva, određuje se presečna tačka koja je toj vezi, tom PNOD-u, najbliža i ona se odbacuje, to jest na njenoj pozicije se neće dodavati novi PPNOD. Tada se taj PNOD označava kao zamena za presečnu tacku. Ukoliko se se obe presečne tačke odbace, onda se zamene povezu. Ukoliko su zamene dva različita OPNOD-a tada se cela ivica označava kao istražena i svi PPNOD-ovi se brišu. Primer c, presečne tačke 1 i 2, vidljiva su dva PPNODa, obe njihove veze nisu vidljive i predstavljaju OPNOD-ove.

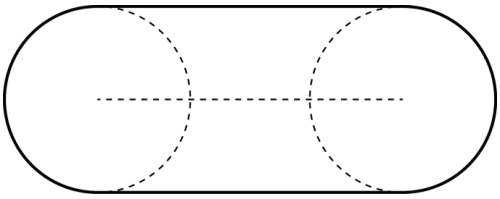
Ukoliko je samo jedna presečna tačka odbačena, tada se PNOD koji predstavlja zamenu vezuje sa novim PPNOD-om koji se dodaje na mesto druge presečne tacke koja nije odbačena. Primer b, presečne tacke 1 i 2, vidljiv je levi PPNOD, njegova

veza desni PPNOD nije vidljiv, on postaje zamena, odbacuje se presečna tačka 2 i povezuje se zamena sa novim PPNOD-om u presečnoj tački 1.

Ukoliko ni jedan PPNOD nije vidljiv mora se proveriti dali se presečne tačke nalaze na istraženom delu ivice, pošto ni jedan PPNOD nije vidljiv dovoljno je proveriti sredinu te dve presečne tačke. Ukoliko se ta sredina ne nalazi na istraženom delu ivice (između dva povezana PNOD-a) tada se dodaju 2 PPNODA na pozicijama presečnih tačaka i međusobno se povezuju. Primer a, presečne tačke 2 i 3.

Svake iteracije na početku ažuriranja PNOD-ova se POT označava kao ne promenjen. Svaki put kada se u POT-u neka ivica označi kao istražena ili se doda novi PPNOD ili OPNOD, tada se POT označava kao promenjen.

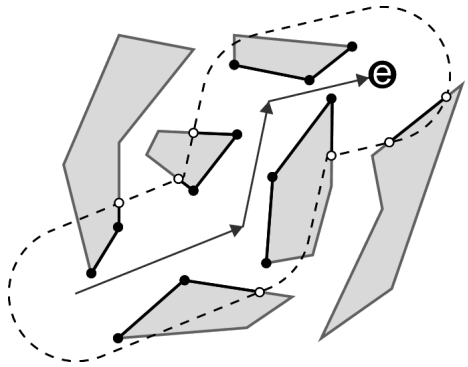
Predstavljanje istražene površine. Za predstavljanje istraženog terena nije dovoljno samo čuvati informacije o viđenim preprekama. Mora se na neki način predstaviti sva istražena površina. Istražena površina je predstavljena pomoću putanje entiteta. Putanja entiteta predstavlja niz tačaka u kojima je on promenio smer kretanja. Kada se ove tačke redom spoje dobija se putanja kojom se igrač kretao. Površina koju je igrač prešao je površina predstavljena kapsulama koje su redom definisane tačkama u putanji i poluprečnikom vidokruga. Kapsula je geometrijsko telo koje se sastoji od centralne duži na čijim krajevima se nalaze dva kruga istog poluprečnika. Ovi krugovi su spojeni dvema zajedničkim tangentama koje su paralelne jedna drugoj i centralnoj duzi (slika 9).



Slika 9.

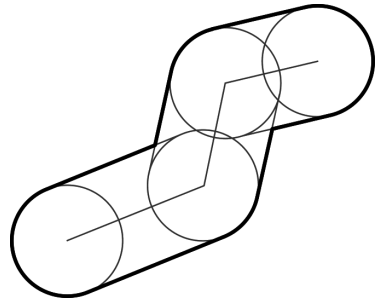
Figure 9.

Primer putanje entiteta vidimo na slici 10. Putanju predstavljaju spojene sive strelice. Na mestu glave svake strelice je entitet promenio smer kretanja i stoga je ta tačka zapamćena u putanji entiteta. Redom spojene ove tačke predstavljaju putanju. Na slici 11 se vidi kako svake dve redom spojene tačke čine jednu kapsulu. Na slici 10 se vidi kako te kapsule čine svu istraženu površinu (isprekidana linija). Na slici 10 vidimo da površina istraživanja koja je definisana kapsulama i istraženi delovi prepreka predstavljeni lancima PNOD-ova koji se nalaze u toj površini, predstavljaju sve informacije o istraženom terenu.



Slika 10.

Figure 10.



Slika 11.

Figure 11.

Povezanost PNOD-ova unutar POT-a. Da bi entitet mogao efikasno da se kreće ka cilju, zaobilazeći poznate prepreke, mora postajati način računanja putanje do cilja. Za bilo koji algoritam koji računa najkraću putanju do cilja potrebno je prvo prostor u kome se računa putanja pojednostaviti. U ovoj implementaciji prostor je pojednostavljen, to jest pred-

stavljen kao graf povezanosti poznatih nodova, PNOD-ova. A za određivanje najkraće putanje do cilja je korišćen Đikstrin (web 3) algoritam. Ovaj algoritam računa putanju od početne (pozicija entiteta) do krajnje (cilj) i krajnja i pocetna tačka moraju pripadati grafu (moraju biti nodovi). Za đisktrin algoritam potrebno je odrediti dužinu svake veze između dva noda. Ta dužina označava dužinu pravolinijskog puta koji treba preći da bi se došlo od jednog do drugog noda koji čine tu vezu. Pomoću ove dužine algoritam određuje putanju do cilja sa najmanjim zbirom ovih dužina.

Svaki OPNOD ima dva niza povezanih PNOD-ova, jedan je niz OPNOD-ova a drugi PPNOD-ova. Svaki PPNOD ima niz povezanih OPNOD-ova, nema smisla određivati povezanost PPNOD-ova međusobno jer nikad se entitet neće kretati od jednog ka drugom PPNOD-u, zato što cim entitet dođe do jednog PPNOD-a on će već biti obrisan. U ove nizove svakog PNOD-a se vremenom kako entitet istražuje teren dodaju novi PNOD-ovi i tako se stvara graf pomoću koga entitet računa putanju do cilja. Kada je jedan PNOD povezan sa drugim PNOD-om (kada oba imaju jedan drugog u nizovima povezanih PNOD-ova) to znači da postoji pravolinijska putanja od jednog do drugog koja je potpuno istražena i koja se ne nalazi ni u jednoj prepreci niti preseca bilo koju. Drugim rečima entitet može nesmetano da se kreće od jednog do drugog noda.

Povezanost PNOD-ova se ažurira samo kada je to neophodno pošto je to zahtevan proces, uglavnom se ažuriranje vrši pre računanja putanje do cilja. Pre svakog procesa ažuriranja svaki niz povezanih PPNOD-ova svakog OPNOD-a se briše, jer PPNOD-ovi se stalno brišu pa ostavljaju rupe u nizovima koje u zavisnosti od implementacije mogu biti opasne.

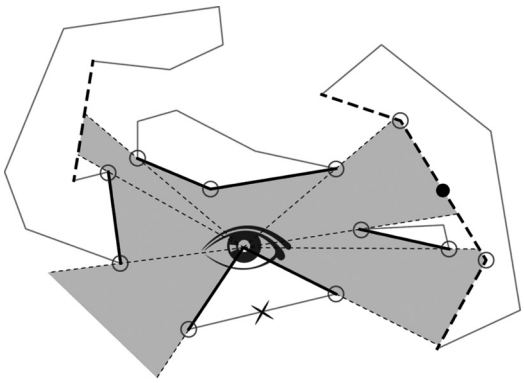
Proces ažuriranja povezanosti PNOD-ova možemo optimizovati tako što ćemo imati informacije o mogućnosti povezivanja nodova i ivica u terenu. Informacije o mogućnosti povezivanja nam pomažu pri procesu povezivanja PNOD-ova. Pošto moramo proveriti dali je putanja između dva PNOD-a istražena i dali se preseca sa poznatim preprekama (lancima PNOD-ova). Ovo proveravanje je zahtevno i ukoliko možemo svesti proveru samo na one nodove koji mogu biti povezani i ivice koje mogu biti povezane sa nodovima onda smo znatno ubrzali proces.

Informacije o mogućnosti povezivanja se čuvaju u dve tabele sa istim brojem redova i kolona koji je jednak broju nodova. Tabela međusobne povezanosti nodova i tabela povezanosti nodova i ivica. Svako polje ovih tabela je promenjiva koja označava povezanost. Tabele povezanosti se određuju na početku simulacije posle generisanja terena.

Tabela međusobne povezanosti nodova (TPN). Dva noda mogu biti ili povezana ili ne povezana, stoga svako polje ove tabele može imati samo te dve vrednosti. Povezanost se određuje proverom sudara terena sa duži koja spaja dva različita noda. Ukoliko se duž ne sudara sa preprekama na terenu ta dva noda se mogu povezati a u suprotnom ne mogu. Ta vrednost se unosi u dva polja tabele. Jednom polju je kolona redni broj jednog noda a red je redni broj drugog noda, a drugom polju obrnuto, tako da je tabela simetrična po dijagonali (polje [5][2] je isto što i [2][5]). Povezanost proveravamo za svaka dva različita noda, da bismo popunili celu tabelu. Ukoliko dva različita noda koja proveravamo pripadaju istoj prepreci, istom poligonu, moramo proveriti pripadnost srednje tačke duži koja ih povezuje unutrašnjosti njihovog zajedničkog poligona. Ukoliko test sudara i test pripadnosti budu negativni tek onda označavamo ova dva noda kao povezana.

Tabela povezanosti nodova i ivica (TPI). Jedan nod je povezan sa svim ivicama koje bi bile vidljive ako bi smo gledali iz pozicije noda sa ne ograničenim vidokrugom, slika 12, nod iz čije pozicije gledamo je označen sa okom a sve vidljive ivice su podebljane. Svaki red tabele predstavlja jedan nod (red odgovara rednom broju noda). Svaka kolona predstavlja ivicu (kolona odgovara rednom broju ivice). U ovoj tabeli svako polje predstavlja povezanost odgovarajućeg noda sa odgovarajućom ivicom. Polje može imati 3 vrednosti: ne povezano, sasvim povezano (kada je cela ivica vidljiva iz pozicije noda) i parcijalno povezano (kada je vidljiv samo deo ivice). Da bismo odredili povezanost nodova sa ivicama moramo testirati povezanost svakog noda sa svakom ivicom. Da bi smo mogli da testiramo povezanost pojedinačnog noda (nod sa okom, slika 12) sa svim ivicama, moramo prvo znati sa kojim sve nodovima je povezan, za to će nam koristiti TPN, ovi nodovi su na slici 12 označeni praznim krugom. Za svaki nod koji testiramo, u ovom slučaju nod sa okom, proveravamo svaku ivicu u nizu svih ivica. Ukoliko su oba noda koja čine ivicu povezana sa nodom koji testiramo, tada testiramo sudar terena sa duži od sredine ivice do noda koji testiramo. Ovaj test je neophodan zbog slučajeva kao na slici 12, ivica koja je prekrštena X-om, njena oba noda su povezana sa nodom sa

okom ali ona nije vidljiva. Ukoliko nema sudara, tada su nod koji testiramo i ivica sasvim povezani. Da bi odredili parcijalno povezane ivice za dati nod, moramo od svih nodova koji su povezani za dati nod odrediti ivične nodove. Ivični nod je onaj nod čija dva susedna noda (prošli i sledeći član u linkovanoj listi) se nalaze na istoj strani prave kojoj pripada duž od datog noda do njega. Datim nodom i ivičnim nodovima su definisane poluprave koje se prostiru u pravcu ivicnih nodova (isprekidane linije, slika 12). Za svaku ovu polupravu se proverava sudar sa terenom. Prva ivica sa kojom je detektovan sudar se označava kao parcijalno povezana i ta vrednost se unosi u tabelu. Test parcijalne povezanosti se radi posle testa totalne povezanosti, jer se može desiti da ivica prođe test srednje duži i da bude označena kao sasvim povezana, kao na slici 12 ivica sa crnom punom tačkom u sredini.



Slika 12

Figure 12

Ažuriranje povezanosti PNOD-ova. Proveravanje povezanosti između dva PNOD-a se sastoji iz dve provere:

1. Provera sudara sa preprekama. Ova provera se obavlja prva, jer je mnogo brža od druge i ukoliko ovde odredimo da veza nije moguća, do druge provere neće ni doći. Sudar duži od jednog do drugog PNOD-a sa terenom proveravamo pomocu MBP-a. To je provera sudara sa svim preprekama u terenu, ne samo sa vidljivim. To nije problem jer to ne daje entitetu informacije o ne istraženom terenu, jer da bi se označila kao povezana, veza između dva PNOD-a mora biti u celosti u istraženom terenu. Tada će prepreke već biti vidljive.

2. Provera istraženosti putanje. Proverava se da li je entitetu poznata pravolinijska putanja između dva PNOD-a čiju povezanost testiramo. Drugim rečima proverava se pripadnosti duži od jednog do drugog PNOD-a istraženoj površini. Istražena površina je predstavljena nizom kapsula redom povezanih, da bi duž u celosti pripadala istraženoj povrsini ona mora da je sadržana u tim kapsulama. Test sadržanosti duži u grupi kapsula se odvija tako što se za svaku kapsulu proverava presek sa duži. Presek duži sa kapsulom moze biti u jednoj ili u dve tačke. Ukoliko postoji jedna tačka preseka to znači da se jedna tačka duži nalazi u kapsuli (zanemaruje se slučaj kada je duž tangenta na kapsulu, tada preseka nema). Kada se odrede preseci, odgovarajući deo duži se označava kao vidljiv, ukoliko postoji jedna tačka preseka, onda se deo od tačke duži koja se nalazi u kapsuli do tačke preseka označava kao vidljiv. Ukoliko postoje dve tačke preseka onda se deo duži od jedne do druge označava kao vidljiv. Označavanje vidljivih delova duži se postiže nodovima sa jednom vezom kao i kod ivica prepreka, jedina razlika je što ivice prepreka preseca vidokrug entiteta dok duži povezanosti PNOD-ova presecaju kapsule. Kako se proveravaju preseci kapsula presečni nodovi se povezuju kao i kod ivica prepreka. Na kraju kada se proveri presek sa svim kapsulama ukoliko je ostalo još presečnih nodova na duzi, to znači da još ima ne istraženih delova i da ta dva PNOD-a još nisu povezana.

Proces ažuriranja povezanosti svih PNOD-ova se odvija u 2 koraka:

1. Ažuriranje međusobne povezanosti OPNOD-ova. Svaki sa svakim OPNOD-ovi se testiraju za povezanost. Pošto OPNOD-ovi odgovaraju nodovima u terenu, rezultati azuriranja, to jest međusobna povezanost OPNOD-ova se čuvaju u istoj tabeli svih nodova kao TPN, na pocetku se vrednosti iz TPN-a kopiraju u ovu tabelu. Pošto vrednosti iz TPN-a predstavljaju vrednosti dobijene prvom proverom povezanosti (provera sudara sa preprekama) ova provera je ovde ne potrebna. Jedina razlika između ove tabele i TPN-a je što polje ove tabele može imati 4 vrednosti koje opisuju vezu 2 OPNOD-a čiji redni brojevi njihovih originalnih nodova ga definisu (svaka dva OPNOD-a definišu dva polja koja imaju istu vrednost jer je tabela simetrična po dijagonali kao TPN):

1. Moguće 1, ovo je kopirana vrednosti iz TPN, to znači da je ova veza moguca ali jedan OPNOD od 2 OPNOD-a koja definišu to polje ne postoji, to jest nije viđen, istražen.
2. Moguće 2, ovo je kopirana vrednost iz TPN, i znači da je ova veza moguća i da dva OPNOD-a koja definišu to polje postoje ali duž koja predstavlja vezu se ne nalazi u celosti u istraženom terenu. Ovu vrednost polje poprima kada se prvi put proveri povezanost (druga provera) i ta provera ne prođe. Ovo polje čuva sve vezano za drugu proveru povezanosti, sve nodove koji definišu vidljivi deo i sve kapsule već testirane. Svaka kapsula koja se jednom testira, bilo da se nađe ili ne nađe presek, se označava kao testirana i više se ne testira. Tako da se testiraju samo nove kapsule, to jest jedino ima smisla ponovo testirati za pripadnost istraženoj površini ukoliko je ta površina promenjena, to jest ukoliko je povećana.
3. Nemoguće, ovo je kopirana vrednost iz TPN, to znači da ova veza nije moguća i neće se ni proveravati. Ovo entitetu ne daje znanje o ne istraženom terenu, isti slučaj kao i u prvoj proveri između povezanosti dva PNOD-a (provera sudara sa preprekama).
4. Postoji, dva OPNOD-a koja ovo polje određuju su povezana (postoji veza).

Svake iteracije se proverava povezanost ona dva OPNOD-a iz niza OPNOD-ova u POT-u čija vrednost u tabeli je Moguće 1 ili 2, ukoliko je 1 to znači da se prvi put provera povezanost ova dva OPNOD--a. Provera povezanosti dva OPNOD-a se sastoji samo iz drugog koraka. Ukoliko provera prođe uspešno tada se veza između ova dva OPNOD-a u tabeli označava kao postojeća i oni se međusobno dodaju jedan drugom u niz povezanih OPNOD-ova. Ukoliko ne prođe tada se u dva polja koja određuje veza ova dva OPNOD-a unosi vrednost moguće 2 i sve informacije vezane za drugu proveru (nodovi i proverene kapsule).

2. Ažuriranje povezanosti PPNOD-ova sa OPNOD-ovima. Pošto PPNOD-ova može biti beskonačno mnogo i zato što njihov broj stalno varira, čas opada čas raste čas se jedni brišu a drugi dodaju, ne moguće je smestiti ih u tabelu i čuvati vrednosti povezanosti sa ostalim OPNOD-ovima. Tako da se u svakom PPNOD-u za vreme njegovog životnog veka čuva niz povezanosti sa OPNOD-ovima veličine broja nodova na tabli, ovaj niz ima vrednosti kao i tabela povezanosti OPNOD-ova. Stim što se početne vrednosti ovog niza određuju u odnosu na vrednosti iz jedne kolone TPI, ta kolona odgovara rednom broju ivice datog PPNOD-a. U toj koloni se nalaze informacije o povezanosti te ivice (samim tim i svakog PPNOD-a na toj ivici) sa svim OPNOD-ovima, to jest njihovim odgovarajucim originalnim nodovima. Ukoliko je vrednost povezanosti određenog noda u toj koloni sasvim povezana, tada se u nizu povezanosti OPNOD-ova element koji odgovara tom OPNOD-u označava kao Moguće 1. Ukoliko nije sasvim povezana (vrednost noda u koloni TPI), tada se proverava povezanost PNOD-ova br. 1, između datog PPNOD-a i OPNOD-a. Ukoliko provera prođe (nema sudara) tada se elemenat takođe označava sa Moguće 1. U suprotnom se označava kao ne moguće. Ukoliko je ne povezana (vrednost noda u koloni TPI) tada se takođe označava kao nemoguće.

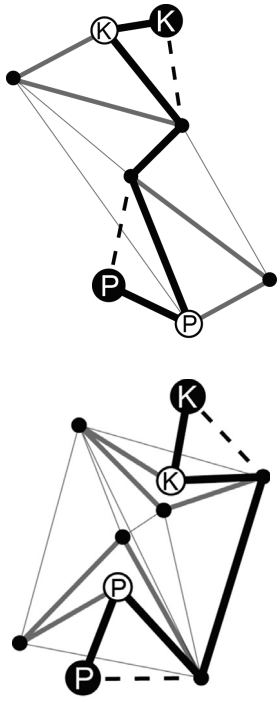
Ažuriranje vrednosti se vrši isto kao i sa vrednostima u tabeli povezanosti OPNOD-ova. Ovakav pristup gde svaki PPNOD ima tabelu svih nodova je memorijski zahtevan, ali u suprotnom kada bi svakog azuriranja povezanosti PPNOD-ova proveravali povezanost svakog PPNOD-a sa svakim OPNOD-om to bi bilo veoma sporo i ne upotrebljivo, pogotovo za velik teren gde je veliki deo istražen. U ovoj implementaciji je brzina veći prioritet nego veličina programa u memoriji.

MP (menadžer putanje)

MP može biti aktivan ili ne aktivan, ukoliko je aktivan on određuje kretanje entiteta,to jest kreirana je putanja i entitet se po njoj kreće. Putanja se kreira kada se MP aktivira (kada više ne postoji prav put do cilja), tada se računa najefikasnija putanja do cilja na osnovu istraženog terena. Nova putanja se takođe kreira kada je MP već aktiviran ali trenutna putanja više nije upotrebljiva.

Problem računanja putanje pomoću đikstrinog algoritma je to što početak i kraj putanje moraju biti članovi grafa, nodovi, u ovom slučaju PNOD-ovi. Naivno rešenje bi bilo odrediti odgovarajući nod u odnosu na vektorsku poziciju početka ili kraja i odrediti putanju sa tim nodovima, a posle dodati vektorski početak i kraj na odgovarajuće krajeve te putanje. Na primer možemo za početak i kraj putanje (pozicija entiteta i cilja) naći najbliže nodove (slika 13a) ili da se za početni nod nađe nod čiji zbir rastojanja od početka do njega i od njega do kraja je najmanji, isto

tako i za kraj (slika 13b). Na slici 13 prepreke su predstavljene debelim sivim linijama, nodovi u grafu su predstavljeni crnim tačkama, veze u grafu (povezanost nodova) tankim sivim linijama, pozicije vektorskog kraja i početka velikim crnim tačkam sa odgovarajućim slovom, nodovi koji ovim vektorski pozicijama odgovaraju su bele tačke na grafu sa odgovarajućim slovom, dobijena putanja je debela crna linija. Na ovim slikama vidimo da putanje dobijene ovim metodama nisu zadovoljavajuće, to jest postoje bolje (isprekidana linija).



Slika 13. a) b)

Figure 13.

Da bismo dobili najbolje putanje ne možemo prevoditi vektorske pozicije početka i kraja u postojeće grafovske, moramo grafu dodati nove privremene nodove u njihovim pozicijama. Prvi nod, nod od kojeg počinje proces računanja putanje ne moramo ni na koji nacin vezati za ostale nodove, što je dobra stvar jer ne moramo menjati graf. Dovoljno je samo da taj privremeni nod početka ima niz svih grafovskih nodova sa kojima može da se veže. Ti grafovski nodovi njega ne moraju da imaju u svojim nizovima jer od njega počinje proces generisanja putanje. Privremeni nod kraja mora promeniti graf jer se proces traženja putanje završava kada se dođe do noda koji može da se veže za krajnji nod, tako da svi nodovi koji mogu da se vežu za njega ga moraju imati u svojim nizovima povezanih nodova. Mogu ga staviti na vrh niza da bi posle pronalaska putanja mogao lako biti obrisan.

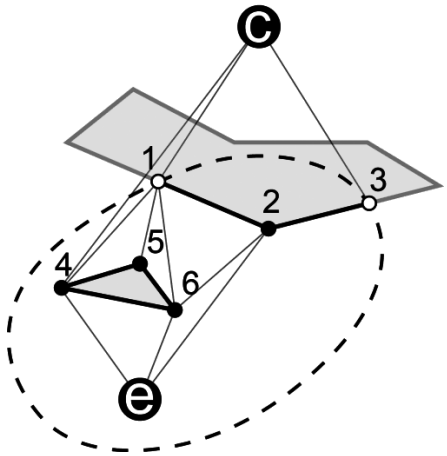
Pre generisanja nove putanje prvo se obavlja ažuriranje povezanosti PNOD-ova. Posle toga se određuju privremeni nod početka (ENOD) i privremeni nod kraja (KNOD). ENOD se kreira u poziciji entiteta i u njegov niz povezanih nodova se dodaju svi PNOD-ovi koji zadovoljavaju obe provere povezanosti dva PNOD-a.

Kreiranje privremenog noda kraja (KNOD).

KNOD određuje sve za njega povezane PNOD--ove i onda sebe dodaje na vrh njihovih nizova povezanih nodova, da bi posle generisanja putanje mogao lako da se obriše. Da bi PNOD bio povezan sa KNOD-om PNOD mora biti ivičan u odnosu na KNOD, duž koja ih spaja ne sme da preseca ni jednu poznatu prepreku (ili deo poznate prepreke). Ukoliko je PNOD kraj svog lanca PNOD-ova onda ne mora biti ivičan u odnosu na KNOD, pošto je u „glavi" entiteta kraj lanca uvek ivičan u odnosu na bilo šta (bilo koju pravu), pošto je entitet optimista, on se nada da iza kraja lanca postoji prav put do cilja, inače ne bi bio dobar istraživač.

Oba noda su privremena u smislu grafa a ENOD je u svakom smislu privremen jer se svaki put kada se generiše putanja kreira novi zato što on predstavlja poziciju entiteta a ona se usled kretanja menja. KNOD se kreira iznova samo kada se promeni cilj koji on predstavlja. Tako da se KNOD svakog generisanja putanje azurira dok se ENOD kreira iznova, podaci o povezanosti nodova sa krajnjim nodom se čuvaju u tabeli sličnoj tabeli povezanosti PPNOD-a.

Na slici 14 vidimo primer privremenih nodova i celokupnog grafa povezanosti. KNOD je označen sa C u crnom krugu, za njega su povezana 3 PNOD-a, PPNOD br. 1, duž do njega ne preseca ni jednu poznatu prepreku, isto važi i za PPNOD br. 3. OPNOD br. 4 je za njega vezan jer je ivičan i duž od njega do KNOD-a ne preseca ni jednu poznatu prepreku. OPNOD br. 2 nije vezan za KNOD jer nije ivičan u odnosu na njega. ENOD je vezan za 3 noda, do kojih putanje moraju u potpunosti biti u istraženoj površini za razliku od nodova povezanih za KNOD gde entitet

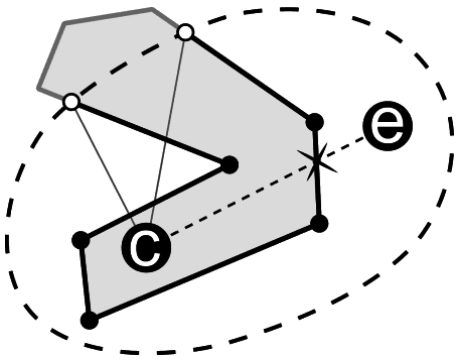


Slika 14.

Figure 14.

samo procenjuje koji nodovi ga mogu odvesti ka cilju jer njihove putanje do cilja nisu istražene.

Na slici 15 vidi se poseban slučaj gde se cilj nalazi u prepreci. Entitet to nezna sve dok ne istraži celu prepreku. Prilikom istraživanja može se desiti situacija kao na slici 15 gde KNOD ne može da se veže ni za jedan PNOD po dosadašnjim pravilima, što bi značilo da entitet ne može da nastavi sa kretanjem pošto ne postoji put do cilja. Ovaj problem je rešen tako što je određena najbliža ivica KNOD-u koju preseca duž od KNOD-a do pozicije entiteta



Slika 15.

Slika 15.

(isprekidana linija, presek u krstu). Određuje se PNOD lanac kome pripada ta ivica i oba kraja tog lanca se povezuju sa KNOD-om. Tada se entitet kreće ka jednom od tih PNOD-ova „gurajući ga" (oktrkivati nove PNOD-ove) sve dok se ne zatvori lanac i ne otkrije poligon, tada se cilj označava kao nedostižan.

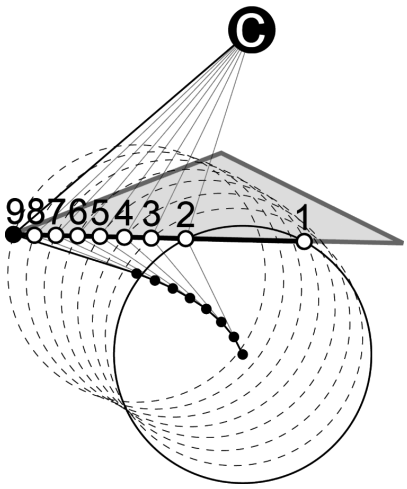
Generisanje nove putanje. Pomoću Đikstrinog algoritma se generiše niz nodova koji imaju najveće šanse da entitet dovedu do cilja. Na grafu sa slike 14 niz nodova putanje bi redom bio ENOD, 6, 1, KNOD. Kada se odredi niz nodova putanje, pretposlednji nod se čuva pošto je putanja od njega do cilja samo pretpostavljena na osnovu ograničenog znanja o terenu. Kao što vidimo na slici 14, od PPNOD-a 1 do KNOD-a ne postoji putanja ali entitet to nezna jer taj deo terena nije istražio pa može jedino da pretpostavi da tu postoji put. Posle svakog ažuriranja PNOD-ova se proverava validnost pretposlednjeg noda. Ukoliko je to OPNOD sa jednom vezon (kraj lanca) proverava se dali i dalje ima jednu vezu, ukoliko ima 2 veze (ukoliko je prostor oko njega otkriven usled kretanja entiteta) tada se proverava dali je ivičan u odnosu na segment putanje koji ide od njega do cilja i ukoliko nije to znači da putanja preseca prepreku i mora se generisati nova putanja. Ukoliko je PPNOD, proverava se dali taj PPNOD još postoji, ukoliko je obrisan to znači da mora da se generiše nova putanja. Na slici 14 na primer entitet će se kretati do PPNOD-a br. 1 sve dok ne dođe do njega i dok se taj PPNOD ne obriše da bi se proširio deo ivice koji je istražen. Kada se obriše tada putanja više neće važiti i generisaće se nova.

Da se ne bi stalno generisala nova putanja u slučaju da je pretposlednji nod PPNOD, MP zabeležava smer u kome će se širiti ovaj PPNOD na svojoj ivici kada entitet dođe do njega. Ovaj smer je suprotan od smera veze ovog PPNOD-a. Takođe se čuva pozicija ovog PPNOD-a i njegova ivica pošto će sam PPNOD biti obrisan kada entitet dođe do njega. Na slici 14 pretposlednji nod u putanji je nod 1 i on je PPNOD, stoga se čuva smer koji je u ovom slučaju suprotan od smera ka nodu 2, a čuva se i njegova pozicija. Kada entitet dođe do ovog PPNOD-a on se briše, i tada putanja više nije validna. Tada se traži odgovarajući PPNOD koji se nalazi na sačuvanoj ivici u sačuvanom smeru u odnosu na sačuvanu poziciju obrisanog PPNOD-a. Ukoliko se takav nađe, putanja se ažurira tako što taj PPNOD postaje pretposlednji nod putanje i njegova pozicija se čuva, brzina

entiteta se ažurira u odnosu na novu promenjenu putanju. Tako se postiže kretanje entiteta uz ivicu u toku kojeg se on ustvari nada da svaki novi PPNOD u smeru širenja na ivici predstavlja njen kraj.

Proces kretanja uz ivicu je ilustrovan na slici 16, prva pozicija entiteta je prvi mali crni krug u putanji koju predstavlja tanka crna linija. Vidokrug koji odgovara prvoj poziciji entiteta je krug crne pune linije, ostali vidokruzi koji odgovaraju ostalim pozicijama entiteta (malim crnim krugovima) su krugovi isprekidane linije. Kada entitet iz prvobitne pozicije računa putanju do cilja, graf povezanosti se sastoji od PPNOD-ova 1 i 2 (preseci njegovog vidokruga) i dva privremena noda (ENOD i KNOD). Entitet generiše putanju kroz nod br. 2 i taj nod se čuva kao pretposlednji zajedno sa potrebnim gore navedenim informacijama. Entitet ažurira brzinu u odnosu na njega. Entitet se pomera za tu brzinu i tada se nalazi u drugoj maloj crnoj tački na putanji (crne tačke predstavjaju pozicije entiteta kroz iteracije). U toj poziciji se ažuriraju PNOD-ovi i nalazi se presek vidokruga sa ivicom i kreira se PPNOD br. 3, br. 2 se briše. Pošto je br. 2 bio pretposlednji nod tada se traži nod koji je u istom smeru na njegovoj ivici i nalazi se br. 3, putanja se ažurira, ažurira se brzina, br. 3 se čuva kao preptoslednji nod itd.. Tako sve do noda br. 9 koji predstavlja kraj ivice, u ovom slučaju putanja kroz br. 9 predstavlja pravu putanju do cilja, iako to entitet nezna. Sa slike vidimo zakrivljenost putanje entiteta dok se kreće niz ivicu, što je posledica toga što entitet pretpostavlja da svaki PPNOD kroz koji u datom trenutku prolazi putanja (2, 3, 4, 5, 6, 7, 8) predstavlja kraj ivice, stoga se kreće direktno ka njemu, sve dok se odmah u sledećoj iteraciji ne nađe novi presek ivice i ispostavi se da to nije kraj i kreira se novi PPNOD, onda se kreće ka njemu i tako redom do kraja ivice, nastaje zakrivljena putanja koja podseća na prirodno kretanje.

Pamćenje smera širenja PPNOD-a na ivici je mnogo efikasnije nego svaki put generisati novu putanju kada pretposlednji nod bude obrisan. Problem sa generisanjem nove putanje je što je to veoma zahtevan proces pošto je potrebno ažurirati povezanost svih PNOD-ova. Još jedna dobra strana pamćenja smera širenja je slučaj sa slike 17, kada entitet odredi putanju kroz PPNOD 2 i krene da se kreće ka njemu, on se obriše i kreira se PPNOD 3, ukoliko bi tada generisali novu putanju tada bi kraća putanja bila kroz PPNOD 1, tada bi se u sledećoj iteraciji u smeru noda 1 kreirao novi PPNOD, dalji od entiteta



Slika 16.

Figure 16.

pa bi sledeća putanja bila kroz nod 3, tako bi entitet išao čas na jednu čas na drugu stranu, tako da bi stajao u mestu. Mora se odrediti smer kretanja duž ivice da bi se izbegla ova situacija.

Pored invalidnosti pretposlednjeg noda u putanji, razlog za generisanje nove putanje može biti dodavanje novih PNOD-ova u toku kretanja kroz već postojeću putanju. Drugi razlog može biti predugo kretanje niz jednu ivicu. Da li će entitet reevaluirati svoj put istraživanja do cilja tako što će što češće generisati novu putanju zavisi od aplikacije u kojoj se koristi, ukoliko je u pitanju igra akcenat će biti na brzini tako da će se izbegavati česta veoma zahtevna ažuriranja povezanosti PNOD-ova kao preduslov za generisanje nove putanje. Ukolo je sa druge strane u pitanju neka simulacija gde izvršavanje algoritama u realnom vremenu nije od presudnog značaja, će se generisanje nove putanje izvršavati što češće moguće, jer je time entitet u boljoj interakciji sa terenom koji istražuje.

Sada kada smo definisali sve strukture koje ga čine možemo opisati interativni algoritam koji čini osnovu IE. Svake iteracije se redom izvršavaju:

1. Proverava se razdaljina do cilja, ukoliko je manja od intenziteta brzine znači da je entitet stigao do cilja. Tada možemo odrediti novi cilj u ne istraženom delu terena tako što ćemo

Slika 17

Figure 17.

naći tačku koja ne pripada istraženoj površini (nizu kapsula) i tako sve dok se ne istraži ceo teren.

2. Ažuriranje POT-a
3. Proverava se dali postoji prav put do cilja, to jest da li duž koja spaja cilj i trenutnu poziciju entiteta preseca bilo koju poznatu ivicu, informacije o poznatim, to jest istraženim ivicama se nalaze u POT-u. Ukoliko postoji, vraća se vektor brzine usmeren ka cilju i prekida se izvršavanje IE.
4. Ukoliko prav put ne postoji, tada se ažurira MP. Ukoliko je MP već aktivan onda se nastavlja kretanje duž već postojeće putanje ukoliko nisu ispunjeni kriterijumi za generisanje nove putanje. Ukoliko je MP ne aktivan, tada se aktivira i kreira se putanja ka cilju.Vraća se vektor brzine ka trenutnoj tački putanje ka kojoj se entitet kreće i prekida se izvršavanje IE.

Posle izvršavanja IE, se dobijeni proizvod, vektor brzine, dodaje na poziciju entiteta.

Rezultati

Svi ovi elementi zajedno čine INT, ova implementacija INT-a podseća na način koji svesni entiteti istražuju teren zahvaljujući svojom složenom predstavom istraženog terena i algoritmima koji pomoću tih informacija stvaraju iluziju inteligentnog kretanja i istraživanja. Rezultate ove implementacije možemo videti na slikama 17 (a, b i c) koje predstavljaju grafički prikaz implementacije gde je entitet prešao put od početne pozicije do cilja istražujući nepoznat teren.

Literatura

web 1.
http://www.wipo.int/patentscope/search/en/detail.jsf?docId=WO2008107859&recNum=1&maxRec=&office=&prevFilter=&sortOption=&queryString=&tab=PCT+Biblio

web 2. http://alienryderflex.com/polygon/

web 3.
http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf

Vladimir Makarić

Exploring of Unknown Terrain

This paper is about unknown terrain exploring (UTE) by an entity (explorer) having narrow view field. The goal was to make an „intelligent“ enitity able to find a path from start to goal points posi-

tioned on an unknown terrain with obstacles, and be able to go back to the starting point using optimized (shortest) path. As a prerequisite for this simulation (UTE), it was needed to first make the systems for obstacled terrain generation, and collision detection for entity-obstacle collisions. Entity movements are based on Dijkstra algorithm, performing on a graph representing the explored terrain and the current field of view. Additionaly, two more nodes are added to the graph on every iterration: one representing the current position of the entity and the other representing the destination point, being connected to all nodes to which there is a straight line path with no known obstacles (or obstacle sections). Results of UTE simulation shows that the movement of the entity designed in such a manner satisfies the predicted goals of the project.