PROJECTE FINAL DE CARRERA

MEMÒRIA

SIMULABER: SOFTWARE PER A LABERINTS

ESPECIALITAT ELECTRÒNICA INDUSTRIAL

Autor: Eva Calderón Agudo
Any: Juny 2003
Pla: 95 Reforma 2002
Ponent: Joan Domingo

ESCOLA UNIVERSITÀRIA D'ENGINYERIA TÈCNICA INDUSTRIAL DE BARCELONA

ÍNDEX

1.- OBJECTE DEL PROJECTE

L'objecte del projecte és realitzar un treball per tal de superar amb bona nota el Projecte Final de Carrera compost per les assignatures PFC1 i PFC2 i obtenir així el títol d'Enginyer Tècnic Industrial especialitat en Electrònica Industrial.

Per tal d'assolir aquest objectiu, es desenvoluparà un projecte la finalitat del qual serà crear un programa informàtic que simuli els moviments d'un robot. En concret, el que es pretén és simular els moviments que faria un robot que es desplacés i resolgués un laberint.

Aquests moviments seran especialment interessants en les interseccions que formen els camins del laberint, on el robot haurà de prendre les decisions adequades que el duguin a la sortida del laberint amb el menor temps possible.

S'ha de tenir en compte que el robot desconeixerà el laberint per tant, el programa ha de poder simular la resolució de diferents laberints. Aquests laberints hauran de complir una sèrie de característiques que vindran determinades pel disseny del robot, ja que el robot estarà pensat per resoldre un tipus en concret de laberint dels molts existents que n'hi ha.

Per fer la simulació de la resolució, caldrà basar-se en algoritmes de control que indicaran quins moviments són els més adequats en cada una de les diferents situacions amb les que pot trobar-se el robot dins del laberint.

Amb el programa simulador, podrà comprovar-se l'eficiència dels diversos algoritmes aplicats a un laberint, la qual cosa ajudarà a escollir el millor algoritme pel software d'un robot en cas que es volgués construir aquest robot.

Cal remarcar que aquest projecte està emmarcat dins de la intensificació d'Automatització i Control de Processos.

2.- MOTIVACIÓ I JUSTIFICACIÓ

La motivació que ha dut a la realització d'aquest projecte ha sigut la de poder aplicar en un exemple pràctic els coneixements adquirits durant la formació com a enginyer.

En un principi, la intenció era dissenyar i construir un robot mòbil que fos capaç de desplaçar-se per un laberint i trobar-ne la sortida, però per motius de temps, es va desestimar aquesta proposta i va redirigir-se el projecte cap a la creació d'un simulador de resolució de laberints. Es va substituir un projecte que tractava principalment la part de hardware d'un robot, per un altre que aprofunditza en la part de software.

En la realització del programa simulador de laberints serà necessari sintetitzar coneixements de diferents matèries com són la informàtica, l'electrònica i, en menor mesura, la mecànica. Aquest projecte és molt interessant i educatiu si es té en compte que l'autor parteix d'uns coneixements molt mínims de la programació en Visual Basic, que serà el programa amb el que es crearà el simulador. A més a més, hi ha el repte afegit de crear un programa executable per poder veure els moviments que faria un robot si el laberint fos real.

Els laberints poden ser interessants des de molts punts de vista. Potser la visió més extensa que es té dels laberints, és que són un mètode d'entreteniment degut a la gran quantitat de jocs relacionats amb laberints que hi ha. Però aquesta no és la única utilitat d'un laberint. Existeixen molts tipus de laberints entre els quals es poden trobar per exemple jardins en forma de laberint. En aquest cas, la utilitat del laberint pot ser com a element de decoració, un lloc per amagar-se i jugar, un lloc on anar a passejar i fins i tot un lloc per relaxar-se. Un exemple real d'aquest laberints és el que mostra la figura 2.1. Per altra banda, cal tenir en compte que els laberints permeten exercitar la intel·ligència d'una persona ja que cal pensar quina és la millor manera de trobar el final. Hi ha diversos col·lectius que utilitzen els laberints com a element per exercitar la ment. Exemples d'aquest col·lectius poden ser els discapacitats psíquics, els nens o fins i tot alguns animals.

Avui en dia existeixen infinitat de programes simuladors orientats a moltes àrees diferents i amb aplicacions molt diverses. Aquests programes simuladors faciliten en gran mesura la realització de determinades tasques i permeten estalviar temps a aquells qui els utilitzen. Generalment els programes simuladors s'utilitzen per verificar el disseny d'algun article, aparell o producte nou per tal de detectar els possibles errors abans d'arribar a construir l'element en qüestió. Això permet a moltes empreses reduir costos abans de treure un producte al mercat.

En el cas del programa simulador de resolució de laberints, servirà per determinar quin algoritme de control hauria de dur un robot segons el tipus de laberint que intentés resoldre. D'aquesta manera es podrà conèixer com actuaria el robot i en el moment de dissenyar-lo es tindrà en compte les característiques que hauria de tenir per poder funcionar amb l'algoritme escollit. Seria el cas, per exemple, dels sensors que hauria de dur el robot, ja que no tots els algoritme adquiriran la informació del medi exterior de la mateix manera.

Per altra banda, pot qüestionar-se la utilitat d'un robot que resolgui laberints. A la vida real, no servirà de molt un robot d'aquestes característiques ja que no es requeriran robots que resolguin laberints. Vist d'aquesta manera, podria pensar-se que no té cap interès el disseny d'un robot que realitzi aquestes funcions. Però, si s'analitzen més detingudament els objectius del robot, s'observa que el que en realitat pretén, és desplaçar-se per unes zones totalment desconegudes pel propi robot, de manera que es van recorrent espais i camins aparentment sense que condueixin a cap lloc en concret però, a mesura que el robot es desplaça, va explorant zones anteriorment desconegudes.

Es podria associar la necessitat de trobar la sortida a un laberint, amb la necessitat de descobrir o investigar llocs, de tal manera que un robot amb característiques similars al que resoldrà el laberint, podria utilitzar-se en l'exploració de llocs on les persones no poden accedir-hi o tenen moltes dificultats per motius com poden ser les condicions ambientals, l'accessibilitat, les dimensions, etc.

Alguns exemples de llocs on serà útil la utilització de robots de reduïdes dimensions poden ser: passos subterranis desconeguts, l'interior de piràmides, el clavegueram de les ciutats, coves, fonts marins, conductes d'aires condicionats, i altres.

Tot i així, en cada aplicació concreta, caldria fer una adaptació del robot de manera que si el robot estigués destinat a explorar fonts marins, hauria de ser resistent a l'aigua; en cas de que explorés coves, l'element de contacte amb el terra hauria de poder superar els obstacles del terra; si explorés l'interior de piràmides, podria anar acompanyat d'una petita camera i una llum per tal de gravar o transmetre les imatges dels llocs per on passa, i així successivament.

Així doncs, s'observa que el que en un principi seria únicament un robot que es desplaça i busca la sortida en un laberint, a la vida real, es podria aplicar aquest mateix robot amb les corresponents modificacions, per explorar llocs inaccessibles per les persones.

Vistes les aplicacions que podria tenir el robot, el primer pas, serà simular el comportament d'aquest robot abans de dissenyar-lo i construir-lo.

Fig. 2.1. Foto del laberint del Parc del Laberint de Barcelona

3.- EXPOSICIÓ PRELIMINAR

Com ja s'ha comentat, el que es pretén en aquest projecte és crear un programa que serveixi per simular els moviments que seguirà un robot mòbil capaç de trobar la sortida a un laberint desconegut pel propi robot. Es farà també una proposta de disseny d'aquest robot i s'orientarà en com seria una possible solució per construir-lo.

Per tal de poder dur a terme tant la creació del simulador com el disseny del robot, en primer lloc caldrà conèixer una mica quins tipus de laberints hi ha i quines característiques tenen.

D'entre tots els tipus de laberints s'escollirà aquell que faciliti més la construcció d'un robot que el resolgui, i serà aquest tipus de laberint el que portarà a la creació d'un programa simulador en el qual es puguin dibuixar laberints i veure com un robot sap trobar-hi la sortida. Així doncs, el primer pas serà conèixer els laberints.

3.1. DESCRIPCIÓ

Tal i com s'acaba d'indicar, es vol arribar a dissenyar un robot que resolgui laberints. Per poder dissenyar aquest robot, caldrà primer conèixer les característiques que haurà de tenir i les funcions exactes que haurà de realitzar.

La millor manera per orientar el disseny del robot, és fer prèviament una simulació del moviments que haurà de realitzar. Si es pretén que el robot resolgui laberints, es començarà construint un programa que permeti fer aquestes simulacions, és a dir, s'haurà de crear un programa en el qual s'hi puguin dibuixar laberints i veure com els resoldria el robot.

Crear un programa d'aquestes característiques requereix uns coneixements sobre laberints, ja que es poden trobar diferents tipus de laberints i per cada un d'ells s'haurà de disposar d'un sistema de resolució adequat a les característiques d'aquest laberint. Seria gairebé impossible crear un robot que pogués desenvolupar la seva funció correctament en tots i cadascun dels diferents tipus de laberints que es poden trobar.

Per veure'n algun exemple, es poden trobar laberints formats per línies pintades al terra; laberints dirigits on cada intersecció només permet girar cap a un determinat sentit; laberints formats per passadissos delimitats per parets; laberints de miralls on els passadissos estan delimitats per miralls; laberints de colors i/o símbols on per anar de l'inici al final només es pot fer passant de color en color o de símbol en símbol; laberints silenciosos on l'objectiu és anar de l'entrada a la sortida sense tocar cap dels objectes sorollosos que hi ha al mig del camí; laberints amb diversos objectes perduts que cal trobar; laberints amb moltes sortides de les quals només una és la vàlida; i altres molts altres tipus de laberints amb formes molt diverses i objectius diferents.

Tot i la gran varietat de laberints diferents que hi ha, es centralitzarà el problema en dos tipus de laberints pels quals serà factible la construcció d'un robot que pugui resoldre'ls i desplaçar-se per ells. Aquests dos laberints seran aquell format per línies fosques pintades sobre un fons clar i aquell format per passadissos delimitats per parets. A continuació s'exposen més detalladament com serien aquests dos laberints i les característiques necessàries que hauria de tenir el robot perquè pogués desenvolupar la seva tasca en un laberint determinat i amb un mètode concret.

3.2.- POSSIBLES SOLUCIONS

3.2.1- Laberint format per passadissos

Quan es parla d'un laberint format per passadissos, es pot pensar en moltes formes de laberints diferents. Efectivament, seran laberints d'aquest tipus tan aquells formats per passadissos rectes amb interseccions de dos, tres o quatre camins i amb girs d'angle recte, com aquells formats per passadissos que poden tenir un cert grau de curvatura, és a dir, aquells que segueixen línies corbes tant en els passadissos com en les interseccions de diferents camins.

La figura següent mostra un exemple de laberint format per passadissos rectes i interseccions que formen també angles rectes:

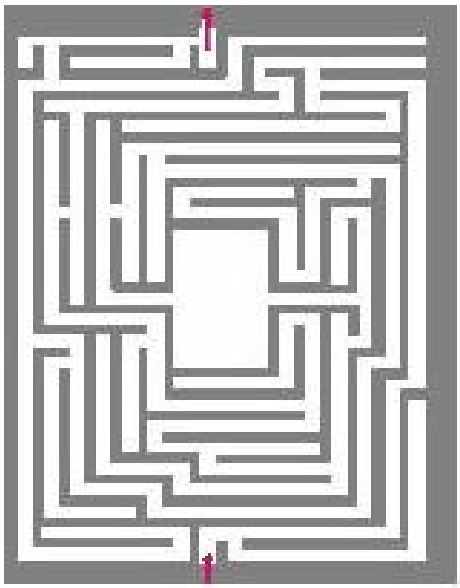


Fig. 3.1. Laberint de passadissos rectes

Si s'observa la figura, es veu que en aquest cas, el laberint té una entrada i una sortida i que per arribar des d'una a l'altra, caldrà desplaçar-se per diferents camins delimitats per parets. Es veu també que tots els girs i interseccions de camins formen angles rectes.

La figura que es mostra a continuació és la d'un laberint format per passadissos que tenen línies corbes en els passadissos:

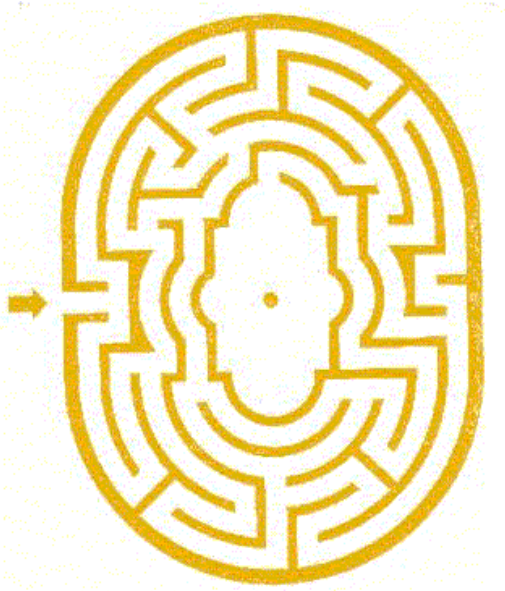


Fig 3.2. Laberint de passadissos amb línies corbes

Seria possible també, trobar algun laberint de passadissos en el que es combinessin els trams rectes amb els trams amb corbes.

Tot i així, dels laberints formats per parets que s'acaben de comentar, el format per trams rectes serà el més interessant alhora de crear un robot que pugui desplaçar-se per ell, ja que és el laberint més fàcil de construir i el que donarà menys problemes.

Possible solució amb robot:

Per poder resoldre un laberint d'aquestes característiques, seria necessari un robot de dimensions reduïdes per tal de poder moure's sense dificultats pel laberint ja que la distància entre les parets del laberint haurà de ser superior a les mides del robot. Per altra banda, caldria que el robot anés equipat amb sensors que li permetessin detectar la presència de parets. Aquests sensors serien detectors de xoc o bumpers, i podrien anar situats un a cada lateral del robot i un tercer sensor a la part davantera. El robot també podria incorporar sensors de so que permetessin detectar la distància a la que es troben les parets. Això serviria per determinar quin és el camí més llarg davant d'una intersecció. Per altra banda, les característiques mecàniques del robot haurien de ser tals que permetessin al robot realitzar girs d'angles de 90º o superiors per facilitar els desplaçaments en les interseccions.

Pel que fa al software del robot, es poden trobar moltes maneres diferents de resoldre un laberint i trobar algoritmes de resolució que prenguin determinades decisions davant les interseccions. El que sí tindran en comú tots ells, serà una lectura contínua de la informació procedent dels sensors del robot i en funció d'aquesta informació l'algoritme de control del robot prendrà unes decisions i donarà les ordres corresponents als mecanismes adequats del robot.

En l'apartat 4 es descriuen alguns exemples d'algoritmes de resolució de laberints.

3.2.2- Laberint format per línies pintades a terra

Aquest laberint és molt similar al laberint format per passadissos però es diferencien en que en un es segueixen línies pintades al terra i en l'altre es segueixen camins delimitats per parets.

Un laberint de línies pintades a terra seria aquell format per un fons clar, normalment blanc i sobre aquest fons, hi haurien unes línies pintades de color negre. Tan el color blanc com el negre haurien de ser mates per evitar reflexes. Les línies negres han de tenir un determinat gruix perquè es puguin diferenciar fàcilment del terra blanc i han d'estar separades una determinada distància les unes de les altres per tal de que no es confonguin. De la mateixa manera, hi haurà una distància mínima entre dues interseccions consecutives.

Les línies poden fromar trams rectes, girs i interseccions, les quals poden ser de dos, tres o quatre camins. En el cas dels girs i les interseccions, estaran formats per angles rectes.

A continuació es mostra un possible laberint d'aquestes característiques:

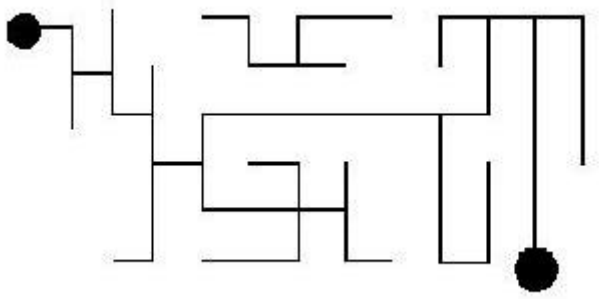


Fig 3.3. Laberint format per línies

Possible solució amb robot:

Un robot capaç de resoldre aquest laberint, seria molt similar al que es necessitaria per resoldre el laberint de les parets, però en el cas de les línies, les dimensions del robot no serien tan estrictes ja que no hi haurà problemes d'espai i, per altra banda, els sensors també serien diferents.

Al haver-se de detectar línies pintades al terra, els sensors que s'utilitzarien en aquest cas, serien sensors òptics que detectarien el contrast entre el color blanc i el color negre i que estarien situats a la part inferior del robot, encarats cap al terra. Serà important doncs, mantenir la superfície del terra neta per evitar problemes de detecció en els sensors.

Les característiques mecàniques del robot, li haurien de permetre també, fer girs de 90º o superior i el mecanisme de contacte del robot amb el terra, que podrien ser rodes, hauria de ser d'un material que permetés una correcte adhesió per evitat lliscaments de les rodes.

Pel que fa al software d'aquest robot, no hi hauria cap diferència amb el del cas de les parets, ja que els criteris de decisió presos pels algoritmes de resolució del laberint, no es veurien afectats pel fet de seguir línies o parets. L'única diferencia seria la informació d'entrada que rebria el microcontrolador del robot, ja que en un cas l'estat de les entrades les indicarien els sensors bumpers i en l'altre cas, els sensors infrarojos.

3.2.3.- Possibles softwares per resoldre el laberint

Per poder resoldre un laberint, cal tenir molt clar els criteris de decisió que permetin escollir quin camí és el més adequat per seguir. Evidentment, el camí més adequat serà sempre aquell que condueixi a la sortida amb el menor temps possible. D'aquesta manera, davant una intersecció de camins, el robot haurà d'escollir per quin d'ells continuarà la seva trajectòria. Per poder fer això, el robot anirà equipat amb un microcontrolador el qual durà un algoritme que permetrà controlar els moviments del robot en tot moment.

Aquest algoritme de control no es veurà influenciat per l'estructura del robot, és a dir, no tindrà importància el tipus de sensors utilitzats ja que tan si la informació procedent de l'exterior es transmet al microcontrolador a partir de sensors bumpers com si és a través de sensors d'infrarojos, l'algoritme actuarà de la mateixa manera. Així doncs alhora de determinar quin algoritme controlarà el robot, no importarà si el robot es desplaça per un laberint format per parets o un format per línies pintades a terra.

Pel que fa al tipus d'algoritme a utilitzar, hi ha moltes possibilitats diferents, ja que davant una intersecció, el robot podria actuar de moltes maneres: girant a l'esquerra, girant a la dreta, seguint recte, seguint recte i si no pot girar a l'esquerra, girant a la dreta i si no pot girar a l'esquerra, ... Es veu doncs, que poden trobar-se molts algoritmes de resolució diferents. Tots ells tenen en comú que s'haurà de fer una comprovació periòdica de l'estat dels sensors ja que a través d'ells es rebrà informació de l'estat del robot dins el laberint.

A continuació s'indiquen alguns exemples d'algoritmes d'aquest tipus.

?? El primer algoritme seria aquell que, davant qualsevol intersecció, faria girar el robot sempre cap a un mateix cantó, dreta o esquerra, és a dir, seria com si el

robot sempre seguís una paret. En cas que no pogués girar cap al sentit determinat, seguiria recte i si tampoc pogués seguir recte, llavors giraria cap a l'altre costat. Aquest procediment, té l'avantatge de que gairebé sempre es troba la sortida i que és molt senzill, però té l'inconvenient de que és un procés molt lent ja que el robot recorrerà un llarg camí abans no trobi la sortida i en alguns casos, passarà dues o més vegades pel mateix lloc. També pot donar-se el cas de que el robot quedi donant tombs en uns determinats camins i no arribi a trobar la sortida. En cas d'estar en un camí tancat, caldria poder fer girar el robot 180º.

?? Una altra opció semblant a l'anterior, seria davant una intersecció, seguir recte, i en cas q no es pugui, girar a un costat. Si tampoc pot girar-se cap a aquest costat (dreta o esquerra), llavors es giraria cap al cantó contrari. Aquest algoritme segueix sent senzill, però tindrà els mateixos problemes que l'anterior.

?? Es podria tenir també un algoritme que davant d'una intersecció de tres camins[1], fes girar el robot cap a un determinat costat, per exemple a la dreta, tan si la intersecció donés la possibilitat de seguir recte o girar, en aquest cas a la dreta, com si es tingués la possibilitat de girar cap a la dreta o l'esquerra. Si aquesta intersecció només permetés girar a l'esquerra o seguir recte, llavors es faria girar el robot a l'esquerra, però en cap cas, davant una intersecció de tres camins, el robot seguiria recte. En canvi, en el cas de tenir una intersecció de quatre camins, llavors, el robot seguiria recte sense fer cap gir. El mateix algoritme es podria aplicar en el cas de que s'escollís girar a l'esquerra davant una intersecció de tres camins. Aquest algoritme, a diferència dels anteriors, ja no es tan senzill, però poden donar-se dues situacions: una es que trobi la sortida amb facilitat i rapidesa; l'altre es que es quedi estancat en un sector del laberint i no en pugui sortir.

?? Un altre tipus d'algoritme, aquest més complex, davant una intersecció faria girar el robot a la dreta. En cas de no poder, es seguiria recte i si tampoc pogués, llavors giraria a l'esquerra. Això es repetiria successivament en cada intersecció a no ser que es detectés que tampoc es pogués girar a l'esquerra la qual cosa significaria que el robot es troba en un camí sense sortida. Si es donés el cas, el robot hauria de fer un gir de 180º i en la pròxima intersecció, en comptes de girar a la dreta, giraria en primer lloc cap a l'esquerra, i si no pogués provaria de seguir recte i per últim a la dreta. Aquest algoritme, quan detectes que el robot està en un camí sense sortida, al tornar enrera, canviaria el sentit de gir per tal que el robot no tornés a passar pels mateixos llocs i no acabés donant tombs en una zona sense poder sortir-ne.

[1] S'entén per intersecció de tres camins, aquella intersecció en la que es creuen dos camins més el camí pel qual ja s'està circulant.

3.3.- SOLUCIÓ ESCOLLIDA

Fins ara s'han comentat les característiques de dos tipus de laberints i els requisits bàsics que hauria de tenir un robot per poder desplaçar-se i trobar la sortida als laberints, tan a nivell d'estructura com a nivell dels algoritmes de resolució.

S'ha pogut comprovar que aquests dos laberints tenen moltes similituds i que el principal problema a l'hora de resoldre els laberints no serà el tipus de laberint en concret sinó l'algoritme de resolució que hauria de dur el software del robot. Així doncs, la importància que tindrà el laberint escollit serà mínima en comparació amb l'elecció de l'algoritme. De totes maneres, s'escollirà aquell laberint que requereixi una estructura del robot més simple i senzilla per tal de facilitat la construcció del robot en cas de que es volgués construir.

Atenent a aquests criteris de simplicitat, s'escollirà el laberint format per parets. El motiu d'aquesta elecció recau en els sensors amb els que hauria d'anar equipat el robot.

Quan es parlava dels possibles laberints, ja es comentava una mica els sensors que necessitaria un robot per resoldre cada un dels laberints. Es va indicar que en el cas d'un laberint format per línies, el robot hauria de dur uns sensors òptics d'infrarojos que li permetessin detectar les línies del terra. Caldrien com a mínim tres sensors d'aquest tipus, un a la part davantera del robot i un a cada lateral. Evidentment, aquests sensors haurien d'anar encarats al terra que és on hi hauria les línies pintades. Per un correcte funcionament d'aquests sensors, s'hauria de tenir en compte que la superfície del terra estigués neta i que el material no fos reflectant ni brillant ja que això podria desviar excessivament els raigs emesos per l'emissor dels sensors. De la mateixa manera, tindria gran importància el correcte posicionament de l'emissor i del receptor ja que en cas d'una mala col·locació, no es capturaria prou bé el raig emès per l'emissor, la qual cosa provocaria una pèrdua d'informació.

Pel que fa als laberint formats per parets, els sensors que hauria de dur el robot serien detectors de xoc o bumpers. Aquests sensors no són més que uns sensors finals de carrera que al topar amb un element, s'activen. En aquest cas, no hi hauria cap problema amb l'estat de neteja del terra ni de les parets del laberint, ja que aquests sensors l'únic que detecten és la presència o no de determinats elements, que pel cas del laberint seran parets. A l'igual que en el cas del laberint de línies, es necessitaria un mínim de tres sensors, els quals anirien col·locats de la mateixa manera però sobresortint de la superfície del robot. El que també s'hauria de tenir en compte són les dimensions del robot. Aquestes, es veurien delimitades per la distància entre paret i paret del laberint, ja que el robot hauria de ser de mida suficientment reduïda per tal de poder-se moure i girar per l'interior del laberint.

Comparant els problemes que durien implícits els sensors utilitzats en els dos casos de laberint, s'escull definitivament el laberint format per parets, ja que el robot que caldria per resoldre'l seria de més fàcil construcció i econòmicament seria més recomanable utilitzar sensors bumpers que sensors infrarojos.

Una vegada feta l'elecció del laberint, el pas següent serà determinar les característiques que hauria de tenir un robot per poder-se desplaçar pel laberint amb l'objectiu de trobar-hi la sortida.

A l'hora de dissenyar el robot, es diferenciaran clarament dues parts: la part de hardware, que comprendrà tota l'estructura i mecanismes del robot, i la part de software, que serà la part encarregada de controlar tots els moviments del robot mitjançant uns criteris que vindran determinats per l'algoritme de control del robot.

Pel que fa a la part de hardware, el robot podria estar construït per una plataforma base, uns motors que accionin les rodes (o qualsevol altre mecanisme), les quals seran el mecanisme de contacte del robot amb el terra, i d'una etapa de potència que adeqüés la senyal procedent del microcontrolador per poder ser aplicada als motors. A més a més, el robot hauria d'anar equipat amb una etapa de control que contingués els sensors i el microcontrolador. Per últim, no s'ha d'oblidar la bateria per alimentar els circuits que ho requereixin.

El microcontrolador serà l'element bàsic per fer funcionar el robot ja que serà l'encarregat d'analitzar la informació procedent dels sensors i de guiar tots els moviments del robot en funció d'aquesta informació. Aquest microcontrolador portarà gravat en memòria el programa corresponent a l'algoritme de resolució que s'utilitzarà per intentar trobar la sortida.

Quan es parlava de les possibles solucions, a l'apartat 3.2.3 es comentaven diferents softwares que podrien aplicar-se per resoldre el laberint. Analitzant detingudament cada un d'ells, s'arriba a la conclusió que aquell que permetrà trobar la sortida d'una forma més intel·ligent, entenent intel·ligent com amb el menor temps possible i sense caure en l'error de quedar atrapat en un sector del laberint sense poder sortir-ne, serà un algoritme que permeti detectar quan el robot es troba en un camí sense sortida i en aquest cas, fer canviar el sentit de gir en la pròxima intersecció o prendre alguna altra decisió adequada. De totes formes, cal tenir en compte que no tots els laberints són iguals i que un únic algoritme no és capaç de resoldre tots els laberints, per tant, caldrien varis algoritmes diferents per poder afrontar diversos tipus de laberints.

Tant la part de hardware com la de software s'explicaran més detingudament en capítols posteriors.

Per concloure aquest capítol, s'indica que el que es pretendrà fer a partir d'aquest moment, és trobar diversos algoritmes que puguin resoldre laberints formats per passadissos delimitats per parets, i un cop trobats aquests algoritmes, es crearà un programa simulador que permeti veure al detall cada un dels moviments que efectuaria un robot per trobar la sortida a un laberint amb un determinat algoritme de resolució.

Per últim, es proposarà un disseny del robot que resolgui laberints.

4.- ALGORITMES DE CONTROL

Per poder trobar la sortida a un laberint és important seleccionar el camí adequat davant una intersecció. En el cas que el laberint sigui desconegut no és possible saber quin és el camí correcte ja que es desconeix cap on conduirà cada camí. Donat aquest problema, l'única solució és seguir uns determinats criteris a l'hora de fer l'elecció del camí.

Com ja s'ha comentat, un sol algoritme de resolució, no és suficient per resoldre tots els laberints de parets que hi poden haver. Degut a això, en aquest capítol s'expliquen quatre tipus d'algoritmes diferents que permeten resoldre laberints molt variats.

4.1.- PARTS D'UN LABERINT

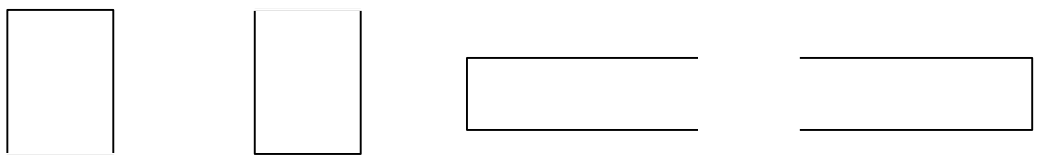
Abans de començar a explicar els algoritmes de resolució, és convenient conèixer de quines parts està compost el tipus de laberint pel qual es proposen els algoritmes.

El laberint escollit és un laberint format per passadissos delimitats per parets. En el laberint hi haurà un punt considerat l'inici i un punt considerat el final. Tant l'inici com el final podran estar situats en qualsevol punt del laberint i l'objectiu del robot serà anar de d'inici al final en el menor temps possible.

Es considera que la distància entre les parets que delimiten un camí, és constant en tot el laberint.

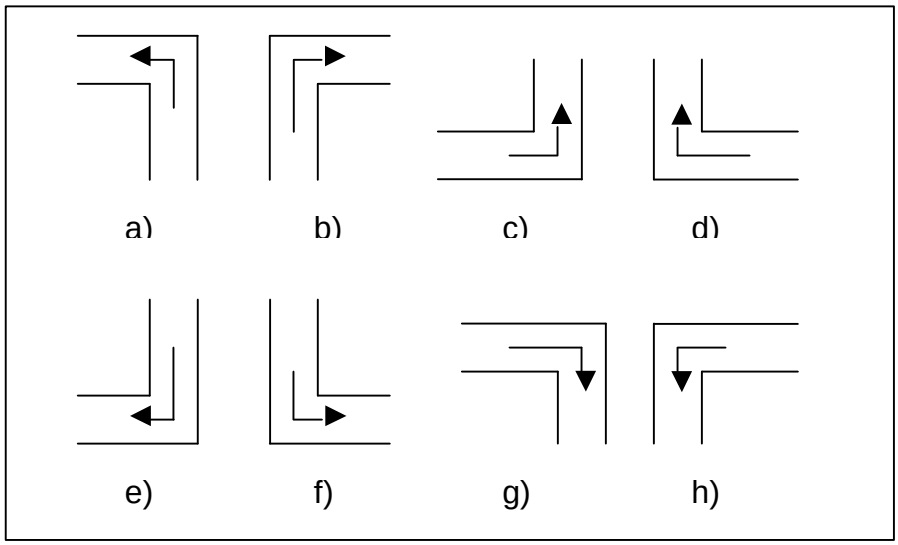
Dins el laberint poden trobar-se trams rectes, interseccions o camins sense sortida. Les interseccions poden ser de dos, tres o quatre camins. A la figura següent es mostren les possibles interseccions que es poden trobar dins el laberint i els camins sense sortida:

?? Camins sense sortida:

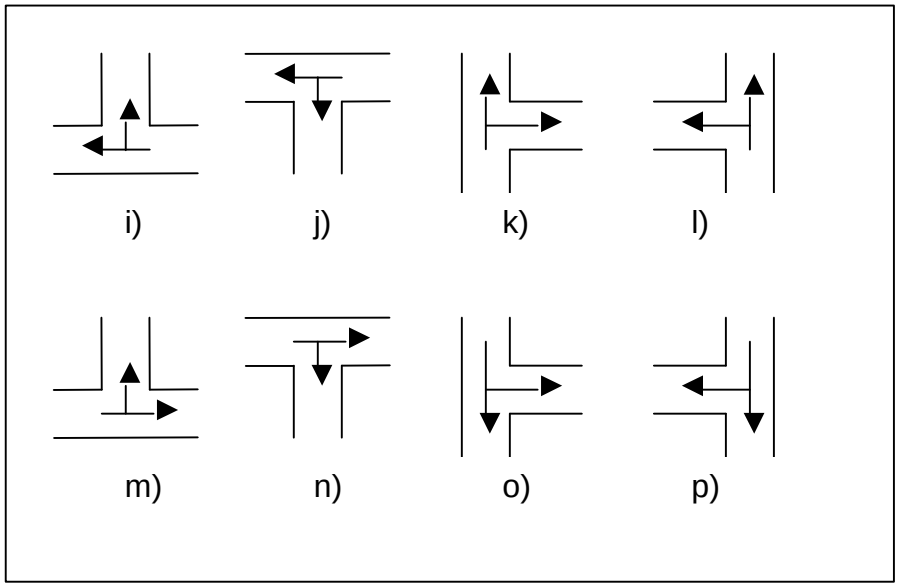


a) b) c) d)

?? Interseccions de dos camins:



?? Interseccions de tres camins:



?? Intersecció de quatre camins

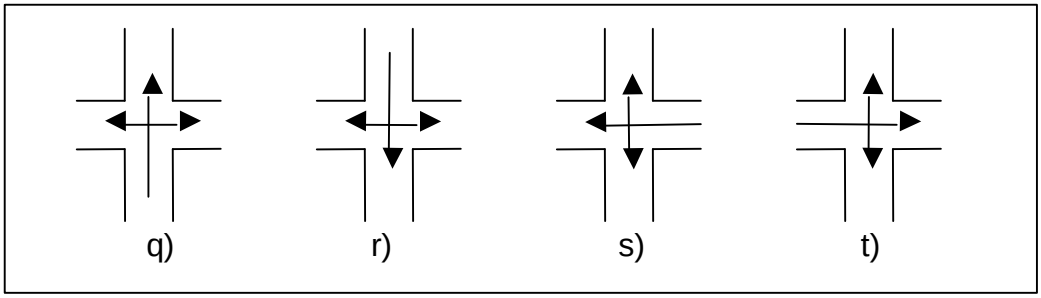


Fig. 4.1. Interseccions i camins sense sortida

4.2.- ALGORITME 1

L'algoritme 1 és un algoritme molt senzill pensat per resoldre laberints petits i amb pocs camins sense sortida. Aquest algoritme, davant una intersecció de tres o quatre camins, sempre farà girar el robot cap a la dreta, i en cas que es trobés en un camí sense sortida, donaria un tomb de 180º i seguiria recte fins a la pròxima intersecció de més de dos camins, on també giraria a la dreta. A continuació s'explica més detingudament com actuaria aquest algoritme.

4.2.1.- Ordinograma de l'algoritme 1

El següent ordinograma mostra la seqüència de decisions i conseqüentment de moviments, que seguiria un robot dins un laberint si el seu algoritme de control fos l'algoritme 1.

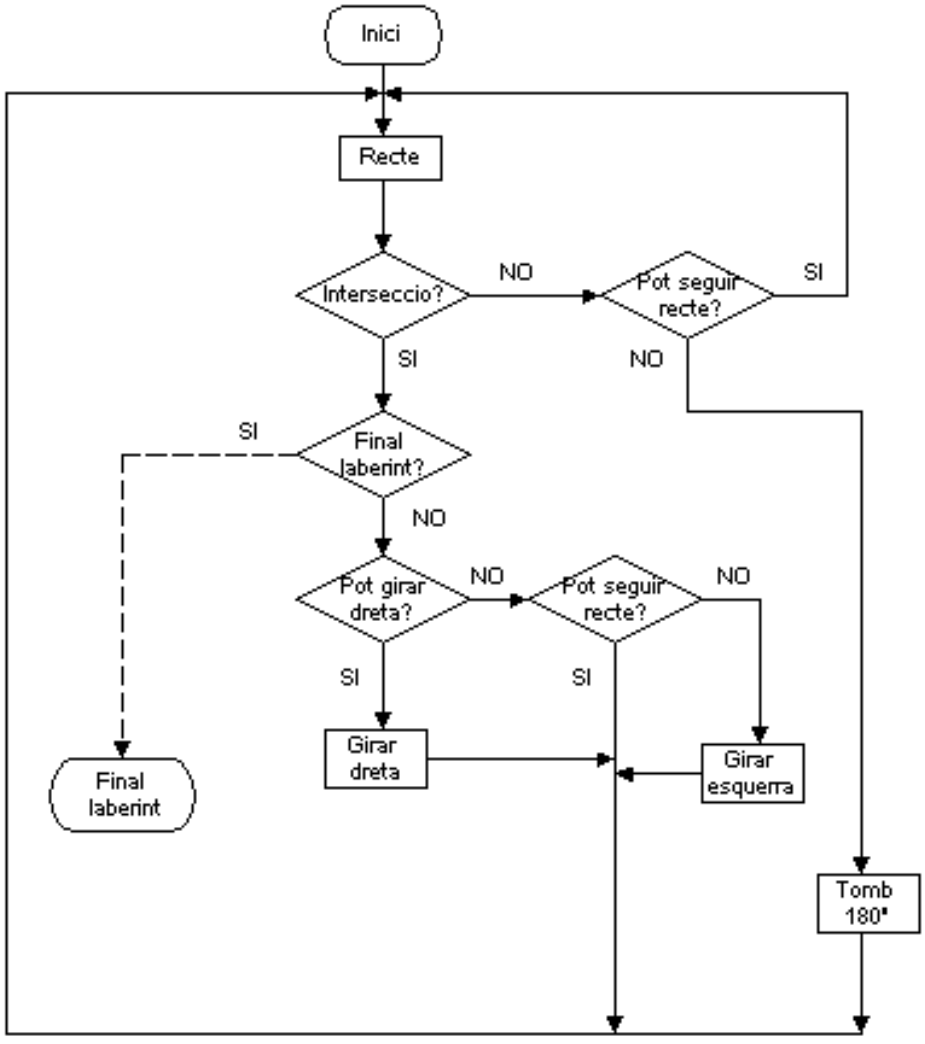


Fig. 4.2. Ordinograma de l'algoritme 1

4.2.2.- Descripció de l'ordinograma de l'algoritme 1

La seqüència de moviments a seguir que ve determinada per l'algoritme 1, parteix del punt marcat com a inici del laberint. El robot que estarà situat en aquest punt inicial, farà un moviment endavant i seguirà recte.

Contínuament s'anirà testant en quina posició es troba el robot dins el laberint. D'aquesta manera es comprovarà si el robot es troba situat en una intersecció o no. Si no es troba en una intersecció i pot seguir recte, el robot seguirà recte. Si pel contrari, el robot detecta que no es troba en una intersecció però tampoc pot seguir recte, llavors entendrà que es troba en un camí sense sortida i donarà un tomb de 180º. Un cop el robot ha girat, tornarà a seguir recte i testant per saber en tot moment en quina posició està.

Si al fer el test el robot detecta que es troba situat en una intersecció, ja sigui de dos, tres o quatre camins, llavors, comprova si en aquesta intersecció es troba el final del laberint. En cas que hi hagi el final del laberint, el robot es desplaça fins al punt final i es considera el laberint resolt.

Una altra situació és que estigui situat en una intersecció però que no hi hagi el final del laberint en ella. Si es dona aquest cas, el robot intentarà fer un gir cap a la dreta (casos b, c, e i f de la figura 4.3.). Si no pot girar a la dreta perquè hi ha una paret, intentarà seguir recte (cas d de la figura 4.3.), però si tampoc fos possible seguir recte, llavors giraria cap a l'esquerra (cas a de la figura 4.3.). Després de fer qualsevol dels girs, el robot segueix recte i continua testant en quina situació es troba dins el laberint.

Aquest procés s'anirà repetint fins que el robot detecti el final del laberint.

A la figura següent es poden veure els moviment que seguiria el robot:

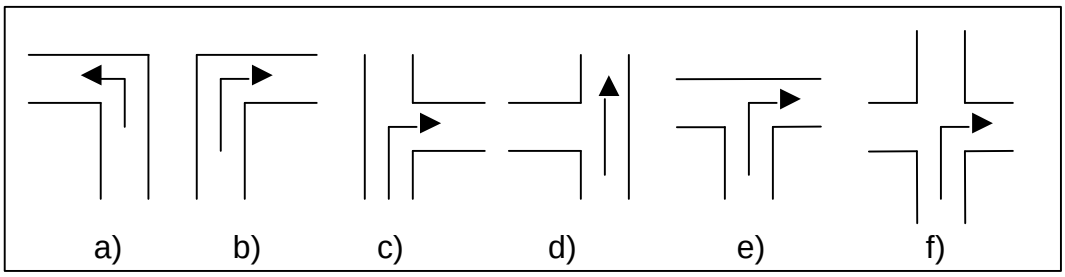


Fig. 4.3. Moviments de l'algoritme 1

4.2.3.- Pseudocodi de l'algoritme 1

El pseudocodi d'aquest algoritme seria el que es mostra a continuació:

Algoritme 1

```
INICI_SEQÜÈNCIA
   REPETIR_FINS_QUE {final de laberint}

        FER {anar recte}
        FINS_QUE {intersecció}o{camí sense sortida}

        SI {camí sense sortida}
           LLAVORS {donar tomb de 180º}
        FI_SI

        SI {intersecció}
           SI {final del laberint}
              LLAVORS{parar robot}
              SI_NO
                 SI {pot girar a la dreta}
                    LLAVORS {girar a la dreta}
                    SI_NO
                       SI {pot seguir recte}
                          LLAVORS {seguir recte}
                          SI_NO {girar a l'esquerra}
                       FI_SI
                 FI_SI
           FI_SI
        FI_SI

   FI_REPETIR
FI_SEQÜENCIA
```

Fig. 4.4. Pseudocodi de l'algoritme 1

4.2.4.- Exemple de laberint resolt amb l'algoritme 1

La millor manera d'entendre com actuaria aquest algoritme, és veure els moviments que seguiria un robot per resoldre un laberint.

El laberint escollit és el que mostra la figura 4.5.

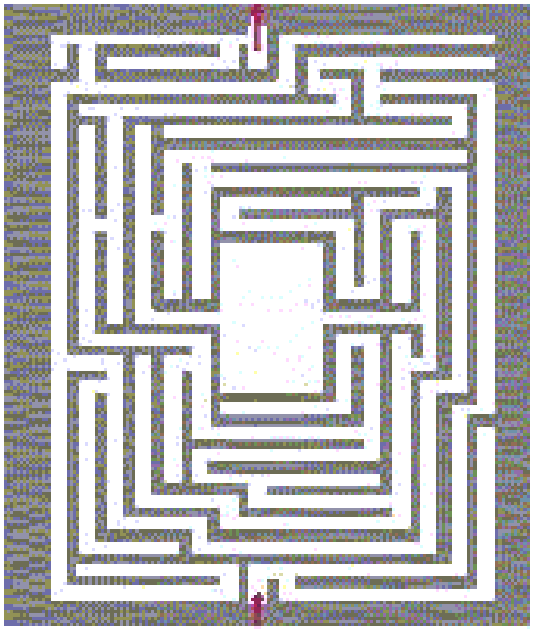


Fig 4.5. Laberint d'exemple per l'algoritme 1

Aquest laberint té l'inici situat a la part inferior i el final a la part superior del laberint, per tant, per resoldre aquest laberint caldrà travessar-lo tot. Aquest laberint no té gaires interseccions però sí té alguns camins sense sortida.

Si s'observa la figura 4.6. es veu com el robot gira a la dreta al trobar la primera intersecció de tres camins. Al girar a la dreta, s'introdueix en un camí sense sortida on segueix recte fins que topa amb la paret, moment en el qual fa un gir de 180º i torna enrere per on ha vingut. Quan es torna a trobar a la primera intersecció, gira cap a la dreta una altra vegada i segueix recte.

Les pròximes tres interseccions amb les que es troba, són interseccions de dos camins on només té la opció de girar cap al sentin que gira el camí. Més endavant, es troba en una intersecció on pot girar a la dreta o a l'esquerra. El robot girarà a la dreta tal i com li indica l'algoritme. Després d'haver girat, torna a trobar-se amb una intersecció de tres camins on pot girar a la dreta i a l'esquerra. El robot tornarà a girar a la dreta i es ficarà en un camí sense sortida. Una vegada detecti que el camí no té sortida, el robot donarà un tomb de 180º i tornarà enrere fins trobar-se de nou amb la última intersecció de tres camins. En aquest cas però, el robot només podrà seguir recte o girar a l'esquerra que seria per on havia vingut. Al

no poder girar a la dreta, agafarà el camí recte, ja que segons l'algoritme té prioritat anar recte abans que seguir recte.

En les següents interseccions i camins sense sortida, el robot continuarà actuant de la mateixa manera: girant sempre a la dreta, si no pot seguirà recte i si tampoc pogués, girarà a l'esquerra.

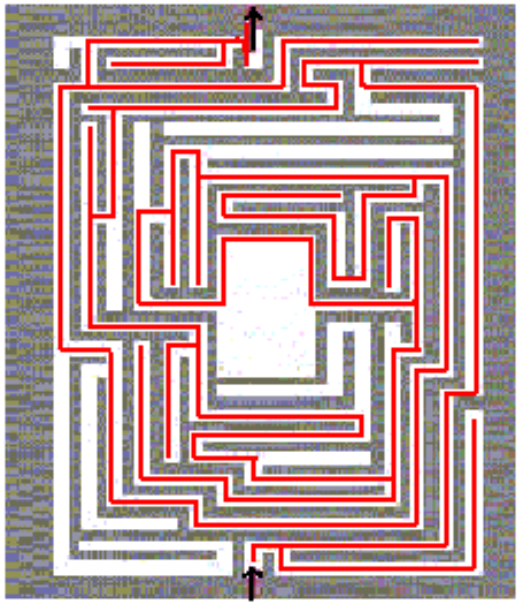


Fig. 4.6. Laberint d'exemple resolt amb l'algoritme 1

4.2.5.- Avantatges i inconvenients de l'algoritme 1

Observant l'exemple anterior es veu la simplicitat que caracteritza aquest algoritme. Això té l'avantatge de que permet que els moviments del robot siguin ràpids ja que no hi ha estructures complexes dins l'algoritme, però té l'inconvenient de que el robot passa per molts camins innecessaris que es podria estalviar amb un altre algoritme que actués de forma més intel·ligent, la qual cosa que fa augmentar el temps de resolució del laberint.

4.3.- ALGORITME 2

L'algoritme 2 és similar a l'algoritme 1 ja que davant una intersecció, sempre farà girar el robot cap a la dreta, si no pogués, seguiria recte i si tampoc fos possible, el faria girar cap a l'esquerra. Es diferencia de l'algoritme 1 en que, al trobar un camí sense sortida, en la primera intersecció que trobi de tres o quatre camins després d'haver girat 180º, canviarà el sentit de gir, és a dir, donarà prioritat a girar el robot cap a l'esquerra i si no fos possible seguiria recte. L'algoritme 1 en canvi, després d'haver trobat un camí sense sortida, seguia donant prioritat de gir cap a la dreta.

L'algoritme 2 també és un algoritme força senzill i està pensat per aquells laberints on calgui, després d'haver entrat en un camí sense sortida, tornar enrere pel camí per on s'havia vingut.

4.3.1.- Ordinograma de l'algoritme 2

El següent ordinograma mostra la seqüència de decisions i conseqüentment de moviments, que seguiria un robot dins un laberint si el seu algoritme de control fos l'algoritme 2

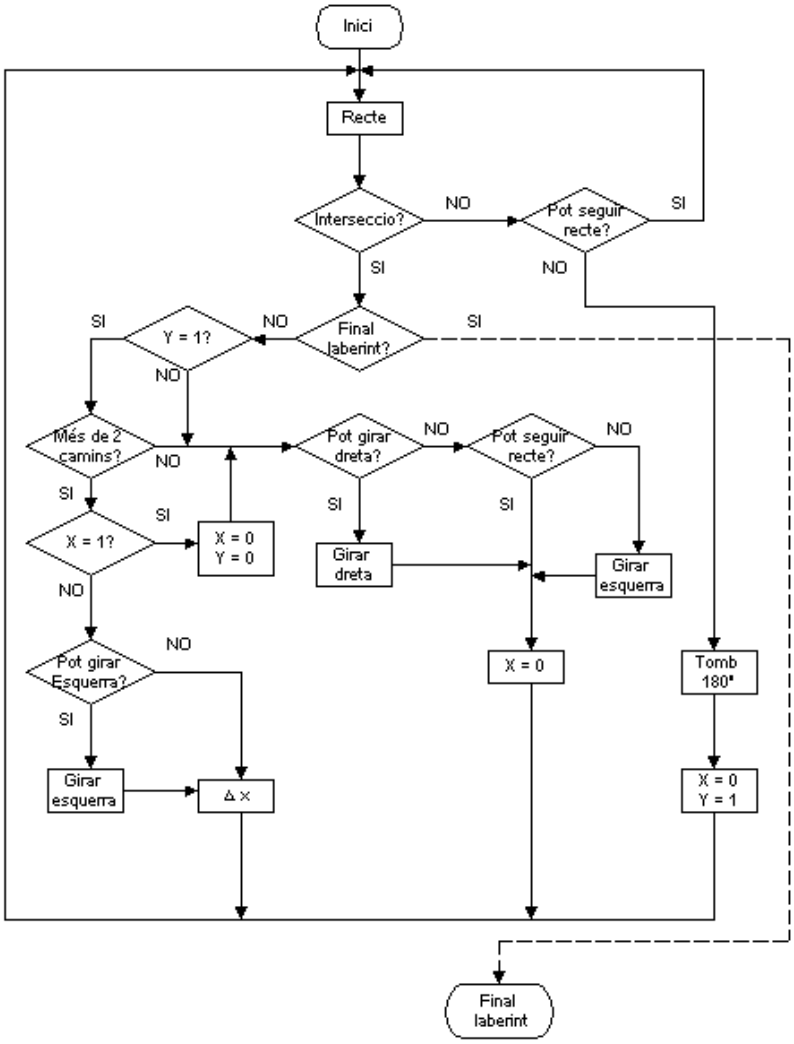


Fig. 4.7. Ordinograma de l'algoritme 2

4.3.2.- Descripció de l'ordinograma de l'algoritme 2

En l'ordinograma que s'acaba de veure, es poden observar els moviments que faria el robot en cada situació.

Es suposa que el robot es troba situat en un punt qualsevol del laberint considerat l'inici. El robot començarà movent-se en línia recte. S'anirà comprovant contínuament la informació procedent dels sensors que durà el robot, la qual cosa permetrà saber si hi ha una intersecció o no. El robot seguirà anant recte fins que trobi una intersecció o bé es trobi en un camí sense sortida.

Si es detecta que no hi ha cap intersecció però el robot tampoc pot seguir recte, llavors significa que hi ha un camí sense sortida. Davant d'aquesta situació, el robot farà un gir de 180º i s'inicialitzaran dues variables: x=0 i y=1. La variable *x* actuarà com un comptador, i la variable *y* indicarà que el robot s'ha trobat amb un camí sense sortida. Els valors de les variables *x* i *y* seran 0 o 1. Després de donar valor a aquestes dues variables, el robot seguirà anant recte fins que trobi la pròxima intersecció de camins.

En el cas de que hi hagi una intersecció ja sigui de dos, tres o quatre camins, es comprova si realment es tracta d'una intersecció o si s'ha trobat la sortida del laberint. Si s'ha trobat la sortida, es considerarà el laberint resolt i el robot s'aturarà. Si es tracta d'una intersecció, es comprova l'estat de la variable *y* i si val 0, significarà que el robot no s'ha trobat cap camí sense sortida en els últims moviments. Si és així, en la intersecció es donarà prioritat a girar cap a la dreta. En cas que no pugui girar a la dreta, el robot seguiria recte (casos a, b, c, d, e i f de la fig. 4.8).

Quan la intersecció és de només dos camins, llavors l'únic que pot fer el robot és girar cap a l'únic sentit possible ja que es tractaria simplement de seguir el camí. Si davant una intersecció de dos camins i amb la variable y=0 el robot no pot girar a la dreta ni seguir recte, farà un gir cap a l'esquerra, la qual cosa equivaldria a seguir el recorregut del camí ja que no es té cap altra opció.

Després d'haver fet qualsevol gir, el robot segueix recte fins tornar a trobar una intersecció.

En el cas que s'acaba de comentar, quan el robot es troba davant una intersecció i la variable *y* val 0, després d'haver girat a la dreta, haver seguit recte o haver girat a l'esquerra, es torna a inicialitzar la variable *x* a 0 encara que el seu valor ja sigui aquest.

Quan el robot es troba davant una intersecció que no és la sortida del laberint i la variable *y* val 1, significarà que en els últims moviments, el robot s'ha trobat en un camí sense sortida. Ara el robot actuarà de maneres diferents segons el tipus d'intersecció en la que es trobi. Si la intersecció és de dos camins, el robot seguirà donant prioritat als girs cap a la dreta, i en cas de no poder seguiria recte o giraria a l'esquerra. De totes maneres, quan hi ha una intersecció de dos camins, la única

opció es seguir el recorregut marcat per aquests dos camins. Aquest seria el cas dels dibuixos g) i h) de la figura 4.8.

Si el robot es troba davant una intersecció de tres o quatre camins i la y = 1, llavors la prioritat dels sentits de gir canviarà.

Abans s'ha vist que quan el robot es troba davant un camí sense sortida, dóna els valor 0 i 1 a les variables *x* i *y* respectivament. Tal i com s'observa en l'ordinograma, quan es troba una intersecció de més de dos camins i y=0, si la *x* és diferent de 1, el robot girarà a l'esquerra i en cas de no poder, seguirà recte. No es tindrà la opció de girar a la dreta perquè tal i com s'observa en i), j), k) i l) de la figura 4.8 , en totes les interseccions de tres o quatre camins, es tindrà la opció de girar a l'esquerra o bé seguir recte. Si la x = 1, es donarà el valor 0 a les dos variables *x* i *y*, i la prioritat dels sentits de gir tornarà a ser primer a la dreta, després recte i per últim a l'esquerra.

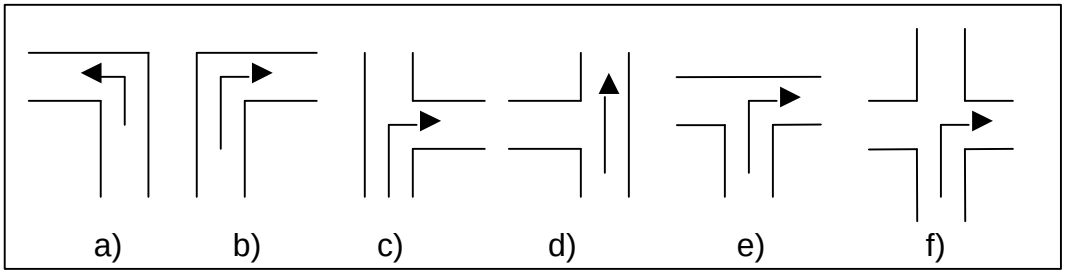
Quan es detecta que la *x* és diferent de 1 i després d'haver fet un gir cap a l'esquerra o haver seguit recte, s'incrementa en una unitat el valor de la variable *x*, que com ja s'ha comentat, servirà de comptador d'interseccions, en concret, de comptador d'interseccions de més de dos camins després d'haver trobat un camí sense sortida.

En el cas de trobar-se en un laberint amb molts camins sense sortida i moltes interseccions, si es fa canviar el sentit de gir després d'un camí sense sortida fins que la variable *x* valgui 1, es possible que el robot no arribi a trobar mai la sortida o tardi molt en trobar-la. Per evitar això, podria modificar-se l'algoritme i en comptes de canviar el sentit de gir en la intersecció posterior a un camí sense sortida, es faria en les tres interseccions posteriors. Es a dir, després d'un camí sense sortida, en la propera intersecció de tres o quatre camins, la prioritat de gir del robot no seria cap a la dreta sinó que seria cap a l'esquerra, llavors s'incrementaria en una unitat la variable x i es repetiria aquesta prioritat de sentit de gir en les pròximes interseccions fins que la x valgués tres.

En el cas de que es canviés una única vegada la prioritat del sentit de gir en una intersecció després d'haver trobat un camí sense sortida, es podria prescindir de la variable *x* ja que no seria necessari tenir un comptador d'interseccions. En l'ordinograma s'ha inclòs aquesta variable per entendre millor com actuaria el robot si es volgués canviar el sentit de gir després d'un camí sense sortida en més d'una intersecció. L'algoritme que durà el programa simulador, només canviarà la prioritat de gir després d'un camí sense sortida, una vegada.

La figura següent mostra els moviments que faria un robot depenent del tipus d'intersecció en la que es trobés i si anteriorment hi ha hagut un camí sense sortida o no.

?? Si anteriorment el robot no s'ha trobat amb un camí sense sortida:



?? Si anteriorment el robot s'ha trobat amb un camí sense sortida:

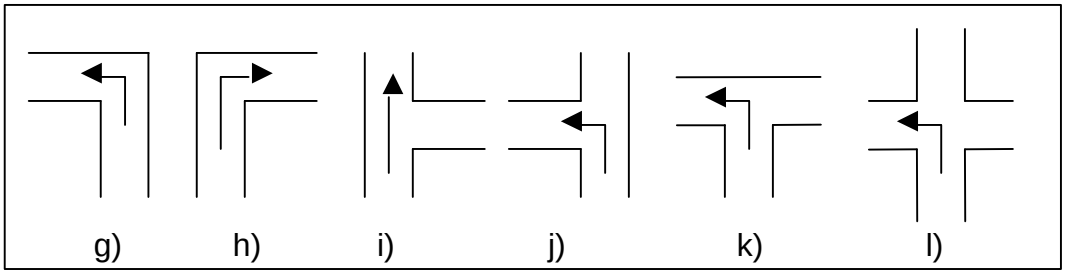


Fig. 4.8. Moviments de l'algoritme 2

Es veu doncs que l'algoritme representat en el diagrama de flux té en compte les diverses situacions en les que es pot trobar el robot dins d'un laberint, i que segons siguin aquestes situacions, el robot faran uns moviments o uns altres.

4.3.3.- Pseudocodi de l'algoritme 2

El pseudocodi de l'algoritme 2 és el següent:

Algoritme 2

```
INICI_SEQÜÈNCIA
   REPETIR_FINS_QUE {final de laberint}

      FER {anar recte}
      FINS_QUE {intersecció}o{camí sense sortida}

      SI {camí sense sortida}
          LLAVORS {tomb de 180º},{X=0},{Y=1}
      FI_SI

      SI {intersecció}
         LLAVORS
            SI {final de laberint}
               LLAVORS {laberint resolt; parar robot}
               SI_NO
                  SI {Y no val 1}
                     LLAVORS {saltar a "pot girar dreta"}
                     SI_NO
                        SI {més de dos camins}
                           LLAVORS
                              SI {X val 1}
                                 LLAVORS {saltar a "pot girar dreta"}, {X=0}, {Y=0}
                                 SI_NO
                                    SI {pot girar esquerra}
                                       LLAVORS {girar esquerra},{incrementar X}
                                       SI_NO {seguir recte},{incrementar X}
                                    FI_SI
                              FI_SI
                           SI_NO
                              SI {pot girar dreta}
                                 LLAVORS {girar dreta},{X=0}
                                 SI_NO
                                    SI {pot seguir recte}
                                       LLAVORS {seguir recte},{X=0}
                                       SI_NO {girar esquerra},{X=0}
                                    FI_SI
                              FI_SI
                        FI_SI
                  FI_SI
            FI_SI
      FI_SI

   FI_REPETIR
FI_SEQÜÈNCIA
```

Fig. 4.8. Pseudocodi de l'algoritme 2

4.3.4.- Exemple de laberint resolt amb l'algoritme 2

Per poder comprendre millor el funcionament d'aquest algoritme, s'expliquen tot seguit els moviments pas a pas que realitzaria un robot dins un laberint. Per entendre l'exemple, s'utilitzarà un laberint senzill en el qual es numeraran les interseccions. Al tractar-se d'un laberint senzill i no gaire gran, després d'un camí sense sortida, en la primera intersecció de més de dos camins, es donarà prioritat de gir cap a l'esquerra però en les properes interseccions ja no.

El laberint que es pren d'exemple és el següent:

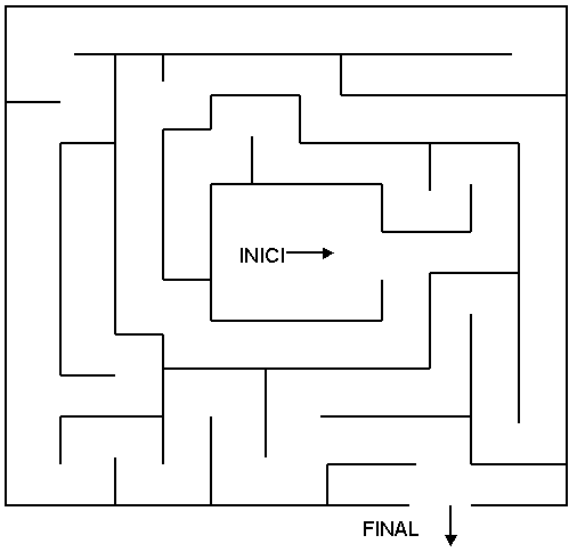


Fig. 4.9. Laberint d'exemple per l'algoritme 2

En aquest laberint el robot inicia el seu recorregut des del centre del laberint. La primera intersecció que es troba (número 1 a la figura 4.10), és una intersecció de tres camins on es té l'opció de seguir recte o girar a la dreta. El robot girarà a la dreta i seguirà recte. Immediatament després, es troba amb una intersecció de dos camins, on la única opció és fer un gir cap a la dreta. Tot seguit, hi ha dues interseccions més cada una de dos camins, on el robot obligadament girarà primer a la dreta i després a l'esquerra. La següent intersecció és de tres camins (número 2 a la figura 4.10) on també es pot seguir recte o girar a la dreta. El robot tornarà a girar a la dreta i seguirà recte.

Fins aquest punt, el robot s'ha trobat amb dues interseccions formades per tres camins on en els dos casos ha fet un gir cap a la dreta. S'ha trobat també amb tres interseccions de dos camins on no ha tingut cap altre opció que girar cap a l'únic sentit possible. Tampoc hi ha hagut cap camí sense sortida, per tant, no s'ha hagut de modificar la prioritat dels sentits de gir.

A partir d'aquí, el robot es trobarà amb nou interseccions més de dos camins fins arribar a una intersecció de tres camins (número 3 a la figura 4.10). En aquest punt el robot pot girar a l'esquerra o seguir recte. Seguirà recte fins trobar-se varies

interseccions de dos camins, fins que arriba de nou a una altra intersecció de tres camins (número 4 a la figura 9.10).En aquesta intersecció, fa un gir a la dreta i continua recte fins que es troba amb un camí sense sortida. Llavors el robot farà un gir de 180º i tornarà enrere fins a la intersecció 4, on aquesta vegada, no girarà a la dreta, sinó a l'esquerra. En aquest punt és on el robot ha fet un canvi en el sentit de gir.

Si continua el seu recorregut, es troba les mateixes interseccions de dos camins que s'havia trobat abans fins que arriba de nou a la intersecció número 4. En aquesta intersecció el robot pot girar a la dreta o a l'esquerra. Com que la prioritat dels sentits de gir torna a ser cap a la dreta, el robot farà un gir cap a aquest sentit i seguirà recte fins trobar-se la intersecció 5. En la intersecció 5, el robot pot girar a la dreta, a l'esquerra o seguir recte. Com que detecta que si segueix recte troba el final del laberint, no gira cap a la dreta sinó que segueix recte i finalitza la resolució del laberint.

En aquest exemple, el robot no s'ha trobat amb cap intersecció de quatre camins, però si se n'hagués trobat alguna, hagués fet un gir cap a la dreta. S'ha vist també que ha sigut suficient canviant una única vegada la prioritat dels sentits de gir després del camí sense sortida, però això no passarà en tots els laberints i en molts casos serà necessari canviar aquesta prioritat dos o tres vegades seguides.

A la figura següent es pot veure el recorregut complert que farà el robot fins trobar la sortida al laberint:

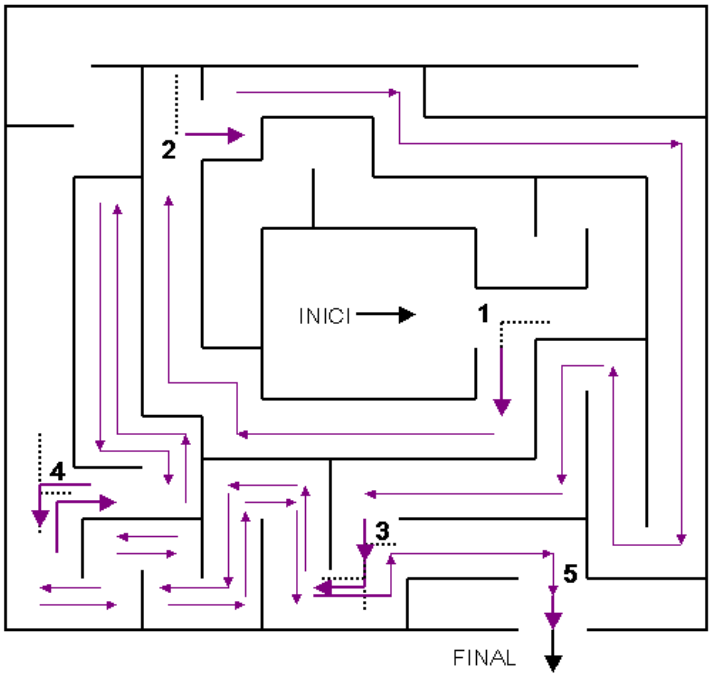


Fig. 4.10. Laberint d'exemple resolt amb l'algoritme 2

4.3.5.- Avantatges i inconvenients de l'algoritme 2

Aquest algoritme, és també un algoritme força simple amb unes estructures no gaire complexes que proporcionen rapidesa al robot. L'inconvenient que té aquest algoritme, és que en determinats laberints, el robot pot quedar-se estancat en uns camins sense poder sortir-ne. Si en el laberint de l'exemple s'intercanviés el punt d'inici i el punt final el robot sortiria del laberint tornant a passar pel punt d'inici, i si es col·loqués per exemple el punt d'inici a la intersecció número 3 i el punt final al lloc de l'inici es veuria com el robot no arribaria a trobar mai el final.

Es veu doncs que tot i ser un algoritme que proporciona una resolució ràpida en determinats laberints, pot donar-se el cas que en algunes situacions, no aconsegueixi trobar mai la sortida.

4.4.- ALGORITME 3

L'algoritme 3 està orientat a la resolució d'un tipus de laberint molt diferent als que poden resoldre els algoritmes 1 i 2. L'algoritme 3 permet resoldre laberints els quals tinguin nombroses sortides. El punt inici i el punt final, no tenen perquè coincidir amb aquestes sortides del laberint, i per tant, la dificultat de la resolució d'aquest laberint recaurà en el fet d'evitar que el robot surti del laberint per error sense haver trobat la sortida. Una altra característica dels laberints que permet resoldre l'algoritme 3, és que no tenen cap camí tancat.

Davant una intersecció de camins, aquest algoritme farà girar el robot cada vegada cap un cantó diferent. Dit d'una altra manera, si en la primera intersecció de tres o quatre camins el robot gira cap a la dreta, en la següent intersecció girarà si pot cap a l'esquerra i si no pot tornarà a girar a la dreta. En tots els casos, evitarà seguir recte.

4.4.1.- Ordinograma de l'algoritme 3

En l'ordinograma de la figura 4.11 es poden observar la seqüència de moviments que seguiria el robot actuant amb aquest algoritme. Es pot comprovar que els moviments seran diferents segons si l'últim gir ha sigut cap a la dreta o cap a l'esquerra.

4.4.2.- Descripció de l'ordinograma de l'algoritme 3

Tal i com indica l'ordinograma, el robot que actuï segons l'algoritme 3, partirà d'un punt inicial i començarà anant recte fins que els sensors que porti li indiquin que es troba en una intersecció.

Al detectar-se una intersecció és comprova si en aquesta intersecció s'hi troba el final del laberint. En cas que es trobi el final del laberint, el robot es desplaçarà fins al punt final i es considerarà el laberint resolt.

Quan es detecta una intersecció on no hi ha el final del laberint, si aquesta intersecció és de dos camins, el que equivaldria a un canvi de direcció dins el mateix camí, el robot girarà a la dreta o a l'esquerra segons el cantó cap el que giri el camí. Si la intersecció és de tres o quatre camins, es comprova quin ha sigut l'últim gir que ha fet el robot. Per fer aquesta comprovació s'utilitza la variable D, el valor de la qual pot ser 0 o 1. Si D=1 significa que l'últim gir ha sigut cap a la dreta. Si D=0 significa que l'últim gir ha sigut a l'esquerra o bé que anteriorment el robot no ha fet cap gir en una intersecció de més de dos camins.

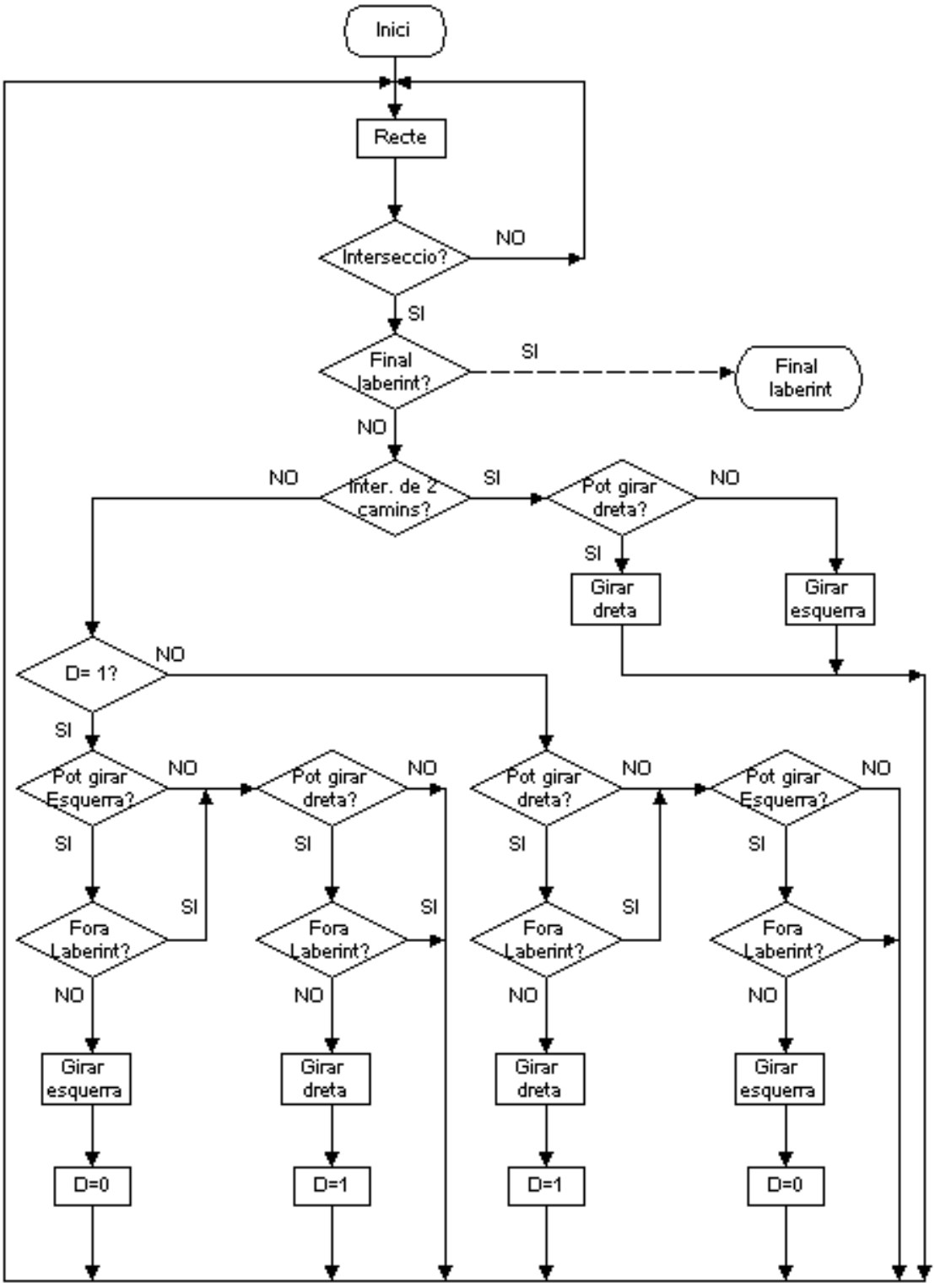


Fig. 4.11. Ordinograma de l'algoritme 3

En cas que l'últim gir l'hagi fet a la dreta, D=1, el robot comprovarà si pot girar a l'esquerra. Si pot girar a l'esquerra i no surt del laberint, llavors girarà a l'esquerra i assignarà el valor 0 a la variable D (D=0) per indicar que l'últim gir no l'ha fet a la dreta. Si girant a l'esquerra surt del laberint, llavors es comprova si pot girar a la dreta. Si pot girar a la dreta i no surt del laberint, el robot gira a la dreta, però si també sortís del laberint, llavors seguiria recte.

Es considera que no pot donar-se el cas d'una intersecció de tres camins on hi hagués l'opció de girar a la dreta o a l'esquerra però que en tots dos casos es sortís del laberint.

Si D=1 i el robot no pot girar a l'esquerra, es comprova si pot girar a la dreta una altra vegada. Si es pot girar a la dreta sense sortir del laberint llavors es gira a la dreta, però si surt del laberint, es segueix recte. Tant si torna a girar a la dreta com si segueix recte, el valor de la variable D segueix sent 1

En cas que D no valgui 1, és a dir D=0, significa que l'últim gir l'ha fet a l'esquerra o el robot es troba en la primera intersecció de més de dos camins després d'haver iniciat la resolució del laberint. En aquest cas, es comprova si el robot pot girar a la dreta. Si pot girar a la dreta sense sortir-se del laberint, llavors gira a la dreta i assigna el valor 1 a la variable D (D=1). Si es detecta que al girar a la dreta el robot pot sortir del laberint, llavors es comprova si pot girar a l'esquerra.

Si el robot no pot girar a la dreta perquè no hi ha camí o bé perquè es sortiria del laberint, es comprova si es pot girar a l'esquerra. Si es pot girar a l'esquerra sense sortir del laberint, llavors es gira a l'esquerra i es torna a assignar el valor 0 a la variable D (D=0), però si al girar a l'esquerra es sortís del laberint, llavors es segueix recte.

Després d'haver girat a la dreta o a l'esquerra i haver assignat el valor corresponent a la variable D, es torna a avançar una posició i a comprovar de nou la posició en la que es troba el robot dins el laberint.

Aquest algoritme es dona per finalitzat quan es detecta que el robot està situat en una intersecció de camins, un dels quals conté el punt considerat el final del laberint.

4.4.3.- Pseudocodi de l'algoritme 3

El pseudocodi de l'algoritme 3 és el següent:

Algoritme 3

```
INICI_SEQÜÈNCIA
   REPETIR_FINS_QUE {final de laberint}

      FER{anar recte}
      FINS_QUE {intersecció}

      SI {intersecció}
         LLAVORS

            SI {final del laberint}
               LLAVORS {parar robot}
               SI_NO

                  SELECCIONAR CAS

                  EN CAS {últim gir, a la dreta}

                     SI {pot girar esquerra}
                        LLAVORS
                           SI {no és fora del laberint}
                              LLAVORS {girar esquerra} i {últim gir, a l'esquerra}
                              SI_NO {saltar a SI pot girar dreta}
                           FI_SI {no és fora de laberint}
                        SI_NO
                           SI {pot girar dreta}
                              LLAVORS
                                 SI {no és fora de laberint}
                                    LLAVORS {girar dreta} i {últim gir, a la dreta}
                                    SI_NO {seguir recte}
                                 FI_SI {no és fora de laberint}
                              SI_NO {seguir recte}
                           FI_SI {pot girar dreta}
                     FI_SI {pot girar esquerra}

                  EN CAS {últim gir, a l'esquerra}

                     SI {pot girar dreta}
                        LLAVORS
                           SI {no és fora de laberint}
                              LLAVORS {girar dreta} i {últim gir, a la dreta}
                              SI_NO {saltar a SI pot girar esquerra}
                           FI_SI {no és fora de laberint}
                        SI_NO
```

```
                    SI {pot girar esquerra}
                       LLAVORS
                          SI {no és fora de laberint}
                             LLAVORS {girar esquerra} i {últim gir, a l'esquerra}
                             SI_NO {seguir recte}
                          FI_SI

                       SI_NO {seguir recte}
                    FI_SI {pot girar esquerra}
                 FI_SI {pot girar dreta}

              FI_SELECCIONAR_CAS

           FI_SI {final laberint}

        FI_SI {intersecció}

     FI_REPETIR
FI_SEQÜÈNCIA
```

Fig. 4.12. Pseudocodi de l'algoritme 3

4.4.4.- Exemple de laberint resolt amb l'algoritme 3

El laberint que mostra la figura 4.13. servirà d'exemple per veure el funcionament d'aquest algoritme.

S'observa que aquest laberint té un punt considerat l'inici i un punt considerat el final. Ambdós coincideixen amb sortides del laberint. Es pot veure també, que hi ha nombroses sortides a l'exterior del laberint i que no hi ha cap camí tancat.

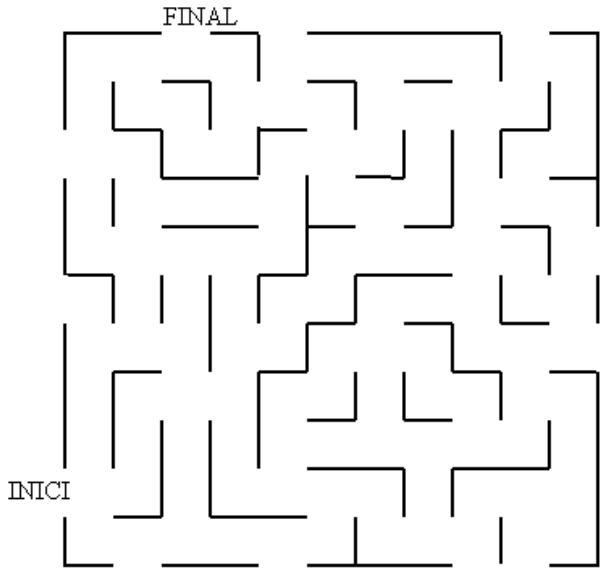


Fig. 4.13. Laberint d'exemple per l'algoritme 3

En aquest laberint, tan aviat el robot comenci la resolució, es trobarà una intersecció de quatre camins en la qual podrà girar a la dreta, a l'esquerra o seguir recte. Com que anteriorment el robot no ha fet cap gir, aquesta vegada girarà a la dreta i guardarà en memòria el valor de la variable D (D=1). Aquesta intersecció és la numero 1 de la figura 4.14.

Tot seguit, el robot es troba amb una alta intersecció on pot girar a la dreta o seguir recte. Si gira a la dreta, surt del laberint sense haver trobat el final, per tant, segueix recte fins arribar a la intersecció 2 (fig. 4.14). En aquesta intersecció, el robot pot girar a l'esquerra o seguir recte. Com que D=1 i pot girar a l'esquerra, llavors gira a l'esquerra i assigna el valor 0 a la variable D (D=0).

Al arribar a la intersecció 3, el robot pot girar a la dreta, a l'esquerra o seguir recte. Com que l'últim gir l'ha fet a l'esquerra (D=0), aquesta vegada gira a la dreta (D=1) i es troba a la intersecció 4. En aquesta intersecció, pot girar a la dreta o a l'esquerra, però com D=1 llavors gira a l'esquerra (D=0).

En la intersecció 5, el robot pot girar a la dreta o seguir recte. La decisió que pren el robot és la de girar a la dreta per dues raons: perquè sempre es dona prioritat als girs abans de seguir recte, i perquè en l'última intersecció el robot ha girat a l'esquerra. Després de girar cap a la dreta en la intersecció 5, la variable D valdrà 1.

Si s'observa la figura 4.14 es veu com en la intersecció 6 el robot gira a l'esquerra (D=0), en la 7 gira a la dreta (D=1), en la 8 gira a l'esquerra (D=0), en la 9 a la dreta (D=1), i en la 10 a l'esquerra (D=0). Al arribar a la intersecció 11, al robot li tocaria girar a la dreta, però com que detecta que si gira a la dreta surt del laberint, gira cap a l'esquerra (D=0). A la intersecció 12 podrà girar a la dreta o a l'esquerra, però com que en la intersecció 11 ha girat cap a l'esquerra, aquesta vegada gira a la dreta (D=1). En la intersecció 13 girarà a l'esquerra (D=0) i en la 14 a la dreta (D=1).

Per últim, quan el robot arriba a la intersecció 15, li toca girar a l'esquerra ja que D=1, i a més a més detecta que si gira a l'esquerra arriba al final del laberint. Per aquestes dues raons, l'últim moviment del robot serà fer un gir cap a l'esquerra (D=0).

A continuació s'observen tots els moviments del robot:

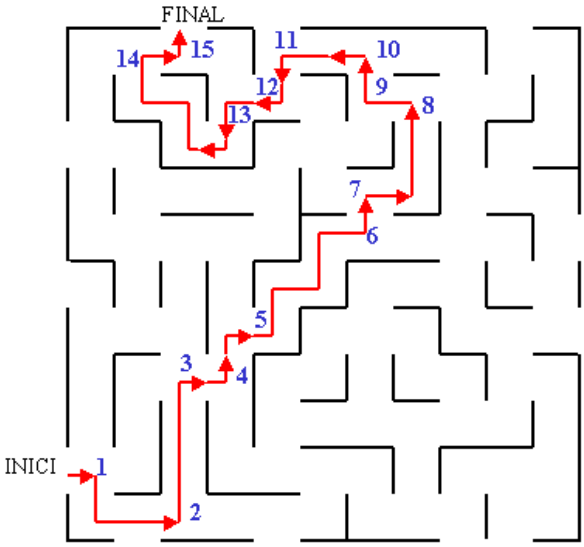


Fig. 4.14. Laberint d'exemple resolt amb l'algoritme 3

4.4.5.- Avantatges i inconvenients de l'algoritme 3

L'algoritme 3 és un algoritme força senzill que permet resoldre els laberints generalment amb rapidesa. Els canvis continus de prioritat en els sentits de gir, fan que el robot difícilment quedi atrapat en una zona sense poder sortir-ne.

Tot i ser un algoritme que proporciona una resolució ràpida, té l'inconvenient de que només pot aplicar-se a un tipus de laberint molt concret. No importarà si els laberint tenen nombroses sortides a l'exterior o no en tenen cap, però el que si que s'ha de tenir en compte és que aquest algoritme no resoldrà laberints que continguin camins sense sortida.

L'única dificultat que presenta aquest algoritme, és la de determinar quins són els límits del laberint, per tal d'evitar que en una intersecció el robot faci un gir erroni i surti a l'exterior del laberint.

4.5.- ALGORITME 4

L'algoritme 4 és un algoritme força complex que es pot aplicar a qualsevol laberint excepte aquells que tinguin nombroses sortides a l'exterior com és el cas del laberint de la figura 4.13. No és recomanable utilitzar-lo en laberints la resolució dels quals requereixi passar més d'una vegada pel mateix camí ja que aquest algoritme evita passar per camins ja fets.

Davant una intersecció de tres o quatre camins, l'algoritme farà que el robot segueixi la resolució del laberint escollint el camí més llarg, sempre i quan no s'hagi passat prèviament per aquell camí.

Si davant una intersecció es troba amb més d'un camí amb la mateixa llargada i per on no hi hagi passat abans, es donarà prioritat a girar, en aquest ordre, cap a la dreta, cap a l'esquerra o recte.

Si es dóna el cas de que el robot s'introdueixi en un camí sense sortida, després de donar un tomb de 180 º , al trobar-se de nou amb la última intersecció de més de dos camins, deixa indicat amb una senyal que es tracta d'un camí tancat.

Quan es detecta que en una intersecció un dels camins condueix al final del laberint, s'escull aquell camí independentment de la llargada dels camins que formin la intersecció.

4.5.1.- Ordinograma de l'algoritme 4

L'ordinograma de la figura 4.15. mostra la seqüència de moviments que seguiria un robot que dugués aquest algoritme de resolució.

Es poden diferenciar tres parts principals en aquest ordinograma. Per una banda, hi ha la seqüència de moviments que seguiria el robot si es trobés en una intersecció de dos camins i per una altra banda, les accions i els moviments del robot davant una intersecció de tres o quatre camins.

La part que tracta les interseccions de més de dos camins, es pot dividir alhora en dues parts: una que comprova cap a quins cantons pot girar el robot (dreta, esquerra i/o recte) i la llargada de cada un d'aquests camins, i l'altra part la funció de la qual és comparar la llargada dels diferents camins i decidir quins moviments farà el robot en relació a aquestes distàncies.

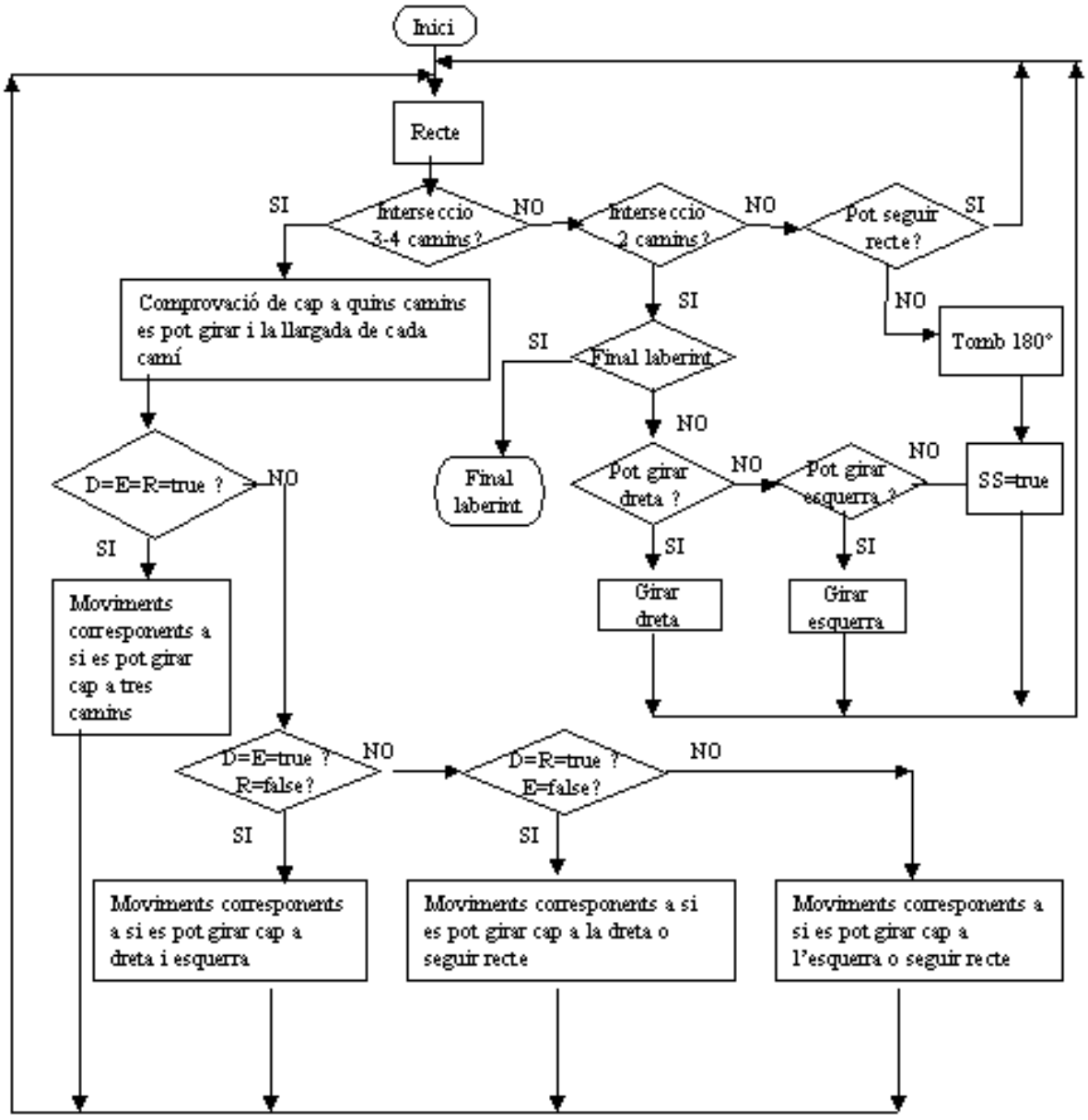


Fig. 4.15. Ordinograma simplificat de l'algoritme 4

Aquest ordinograma, és en realitat un ordinograma simplificat de l'algoritme 4. Les figures 4.16, 4.17, 418, 4.19 i 4.20 amplien l'ordinograma de la figura 4.15. La figura 4.16 mostra l'ordinograma de la part corresponent a la comprovació de cap a quins camins es pot girar i la llargada de cada camí; la figura 4.17 mostra l'ordinograma de la part dels moviments corresponents a si es pot girar cap a tres camins; la figura 4.18 mostra l'ordinograma dels moviments corresponents a si es pot girar cap a la dreta i cap a l'esquerra; la 4.19 mostra els moviments corresponents a si es pot girar cap a la dreta o seguir recte; la figura 4.20 fa referència als moviments corresponents a si es pot girar a l'esquerra o seguir recte.

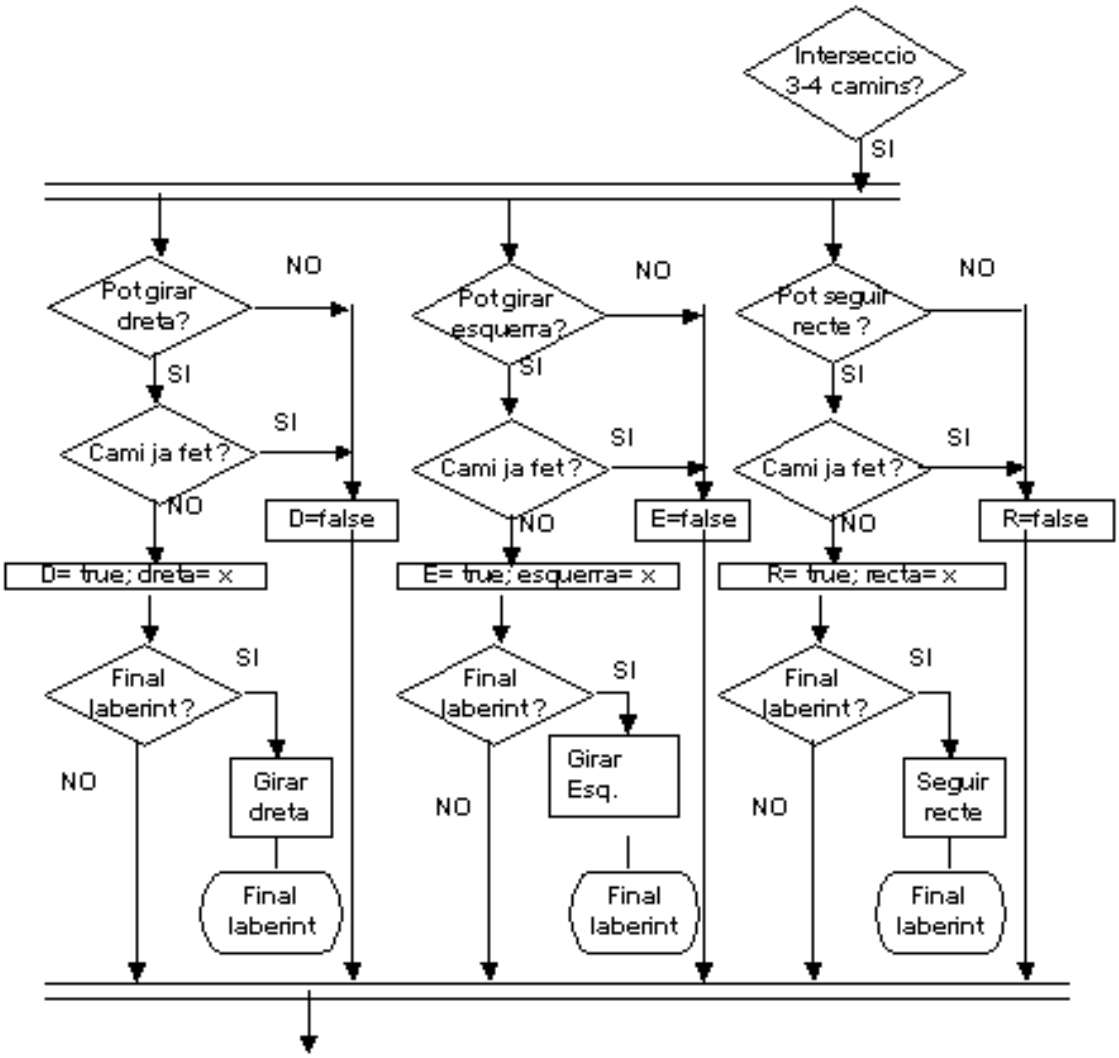


Fig. 4.16. Comprovació de cap a quins camins es pot girar i la llargada de cada camí

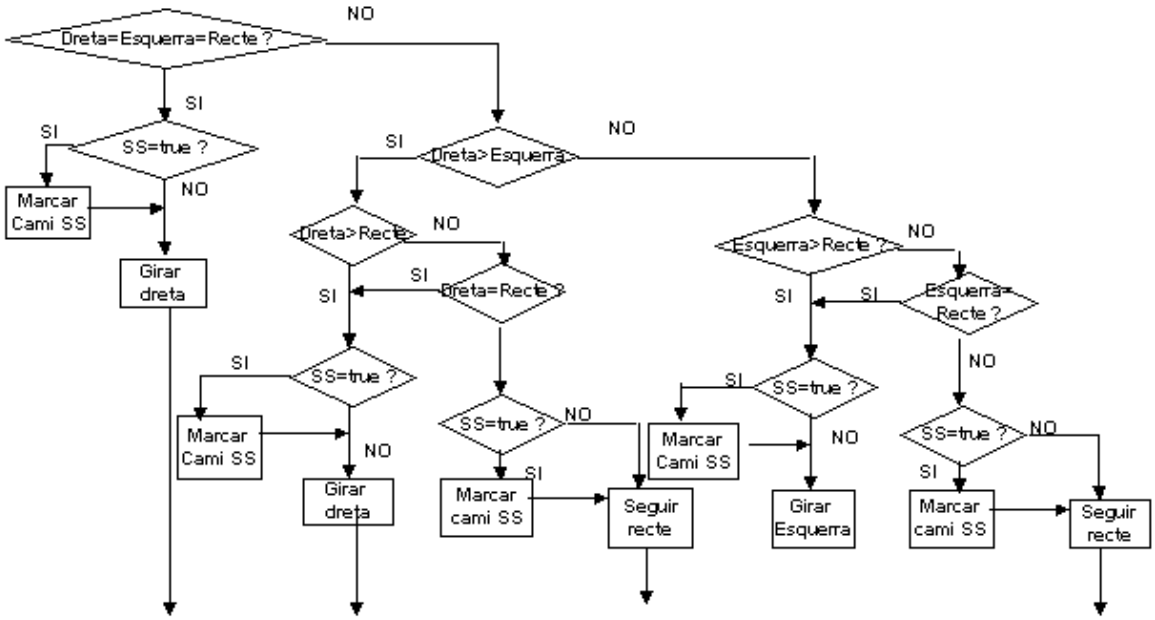


Fig. 4.17. Moviments corresponents a si es pot girar cap a tres camins

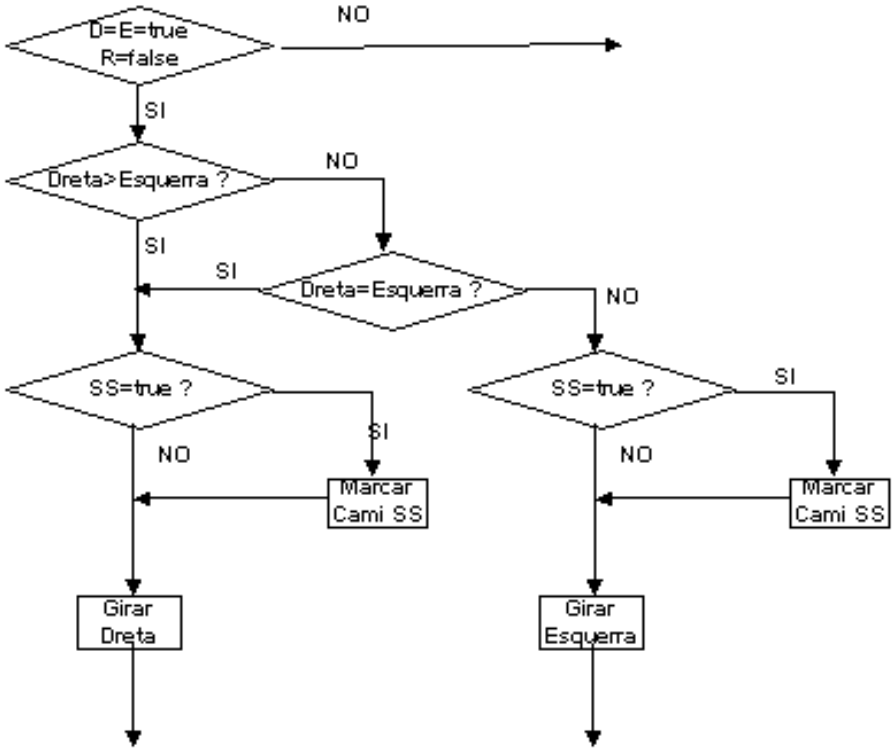


Fig. 4.18. Moviments corresponents a si es pot girar cap a la dreta i l'esquerra

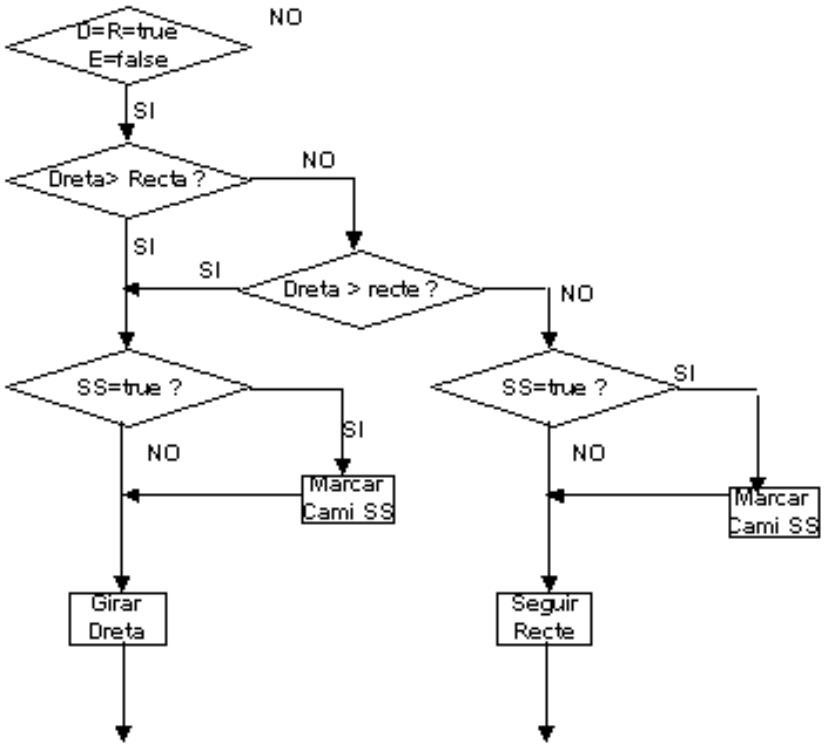


Fig. 4.19. Moviments corresponents a si es pot girar cap a la dreta o seguir recte

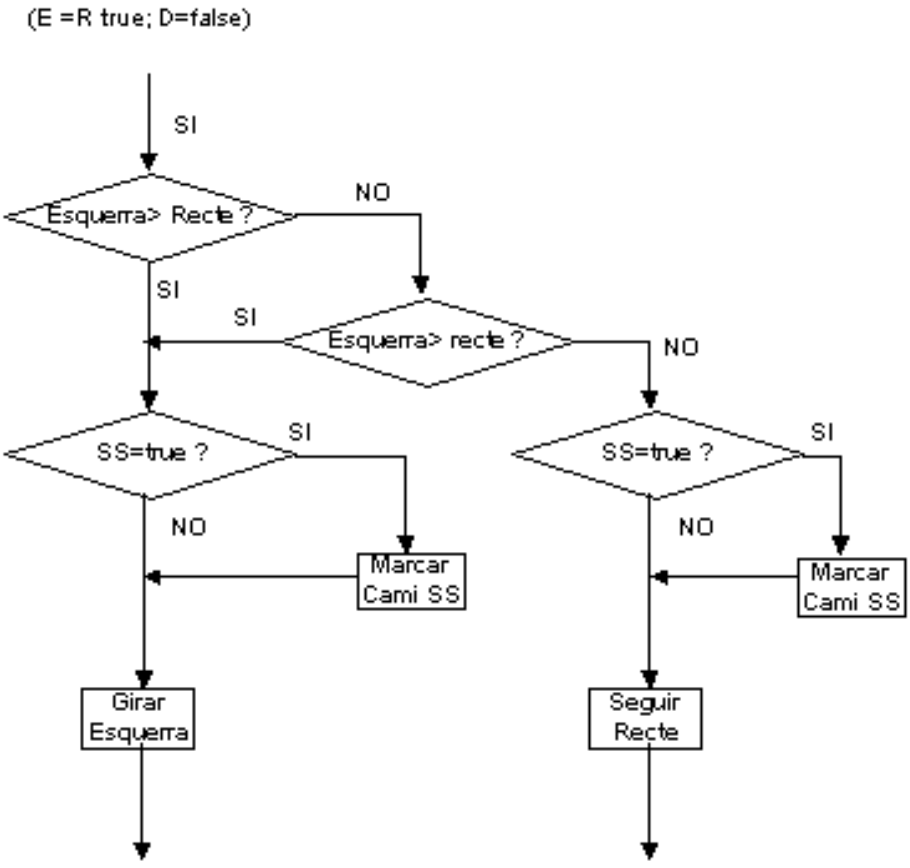


Fig. 4.20 Moviments corresponents a si es pot girar cap a l'esquerra o seguir recte

4.5.2.- Descripció de l'ordinograma de l'algoritme 4

La seqüència de moviments a seguir que ve determinada per l'algoritme 4, parteix d'un punt marcat com a inici del laberint. En aquest punt, s'hi situarà el robot i el primer moviment que farà serà anar recte. El robot contínuament anirà comprovant en quina posició del laberint es troba, per tal de determinar si es troba en un camí o en una intersecció. Mentre el robot no detecti que està situat en una intersecció o un camí tancat, seguirà anant recte.

Si el robot detecta que no es troba en cap intersecció i tampoc pot continuar anant recte, significarà que està en un camí sense sortida. En aquest cas donarà un tomb de 180º, i indicarà amb una variable, SS, que s'ha trobat en un camí sense sortida (SS=true).

Quan el robot detecta que es troba en una intersecció, comprova si es tracta d'una intersecció de dos o de més camins. Si la intersecció és de dos camins, comprova si hi ha el final del laberint i en cas de que sí hi sigui, es considera el laberint resolt. Si en la intersecció de dos camins no hi ha situat el punt final, llavors es comprova si es pot girar a la dreta i en cas de poder, el robot gira a la dreta. Si no es pot girar a la dreta, es comprova si es pot girar a l'esquerra, i si tampoc es pot, es segueix recte.

Si el robot està situat en una intersecció de tres o quatre camins es comprova cap a quins cantons pot girar. Es possible que el robot pugui girar a la dreta, a l'esquerra i/o seguir recte, per tant, es comprovaran cada una d'aquestes possibilitats per separat.

Es farà la comprovació de si el robot pot girar a la dreta. Si no pot girar, s'assigna el valor false a una variable anomenada D que significa dreta (D=false). En cas que sí es pugui girar a la dreta, es comprova si pel camí de la dreta ja s'hi ha passat amb anterioritat. Si es així, s'assigna de nou el valor false a la variable D (D=false) la qual cosa significarà que no es pot girar a la dreta. Si no s'ha passat abans pel camí de la dreta, s'assigna el valor true a la variable D (D=true) que servirà per indicar que es pot girar a la dreta. En aquest últim cas es comprovarà si girant a la dreta s'arriba al final del laberint. Si s'hi arriba, es girarà a la dreta fins trobar el final i es considerarà el laberint resolt.

Després d'haver fet la comprovació de si es pot girar a la dreta o no, s'obté un valor de la variable *D*. Si D=true, significa que sí es pot girar a la dreta. Si D=false, indica que no es pot girar a la dreta. A més a més, també s'obté el valor d'una variable anomenada *dreta*, que indica un número que fa referència a la llargada del camí que hi ha a la dreta.

Aquesta comprovació que s'acaba d'explicar referent a la possibilitat o no de girar cap a la dreta, també es fa per saber si es pot girar a l'esquerra i per si es pot seguir recte. En cas de la comprovació de l'esquerra, s'obtindrà el valor de la variable *E* (E=false o E=true), i de la variable *esquerra* que indicarà la llargada del camí. Per la comprovació de seguir recte, s'obté la variable *R* (R=true o R=false) i la variable *recte* que indica la llargada del camí en cas que n'hi hagi.

Després d'haver fet aquestes tres comprovacions, es coneixeran les variables *D, E* i *R* que indicaran si es pot girar cap a la dreta, l'esquerra o recte, respectivament. També s'obtindran les variables *dreta, esquerra* i *recte,* que indicaran la llargada dels camins en cas que hi siguin.

Un cop es coneix els camins que formen una intersecció (D, E, R), es poden donar els següents casos:

a) Que es pugui girar a la dreta, a l'esquerra o seguir recte: D=E=R=true
b) Que es pugui girar a la dreta o a l'esquerra: D=E=true, R=false
c) Que es pugui girar a la dreta o seguir recte: D=R= true, E=false
d) Que es pugui girar a l'esquerra o seguir recte: E=R=true, D=false

Si D=E=R=true, llavors es comprova quins camins són els més llargs. Si els tres camins són igual de llargs, dreta = esquerra = recte, llavors es girarà a la dreta. Si anteriorment s'ha trobat un camí sense sortida (SS=true) abans de girar a la dreta es senyalitza el camí sense sortida.

Si D=E=R=true però el camí de la dreta és més llarg que el de l'esquerra, (dreta > esquerra) llavors, si el camí de la dreta és igual o més llarg que el camí recte (dreta > = recte), es girarà a la dreta, però si el camí recte és més llarg que el de la dreta (dreta < recte), es seguirà la resolució del laberint pel camí recte.

Si D=E=R=true però el camí de l'esquerra és més llarg que el de la dreta, (dreta < esquerra) llavors, si el camí de l'esquerra també és més llarg que el recte o és igual que el recte (esquerra > = recte), el robot girarà cap a l'esquerra. Si en canvi el camí recte és més gran que el de l'esquerra (esquerra < recte) llavors el robot seguirà pel camí recte.

Amb aquestes comparacions de camins, s'aconsegueix que el robot segueixi el seu recorregut escollint el camí més llarg. En tots els casos, si el robot detecta que anteriorment s'ha trobat amb un camí sense sortida, abans de fer qualsevol gir o seguir recte, senyalitza que el del qual ve, és un camí sense sortida.

En cas que D=E= true i R=false, llavors si el camí de la dreta és més llarg o igual que el de l'esquerra (dreta > = esquerra), el robot girarà cap a la dreta. En canvi, si el camí de la dreta és més curt (dreta < esquerra) el robot girarà cap a l'esquerra.

Si D=R= true i E= false, llavors, si el camí de la dreta és més llarg o igual que el camí recte (dreta >= recte), el robot girarà a la dreta, però si el camí de la dreta és més curt (dreta < recte), el robot seguirà recte.

Per últim, si E=R=true i D=false, llavors si el camí de l'esquerra és més llarg o igual que el recte (esquerra > = recte), el robot girarà a l'esquerra, però si el camí de l'esquerra és més curt (esquerra < recte), el robot seguirà recte.

En tots els casos es comprova si el robot ve d'un camí sense sortida i en cas que sigui així, abans de girar a la dreta, a l'esquerra o seguir recte, es senyalitza el camí com camí tancat.

Una vegada fets aquests moviments, el robot seguirà recte pel camí escollit fins que es torni a trobar una altra intersecció.

El laberint es considera resolt quan en una de les interseccions, ja sigui de dos, tres o quatre camins, el robot detecta que hi ha el final del laberint. Quan el robot detecta el punt considerat el final del laberint, es desplaça fins a aquest punt sigui quina sigui la llargada del camí, i es considera el laberint resolt.

4.5.3.- Pseudocodi de l'algoritme 4

El pseudocodi de l'algoritme 4 és el següent:

Algoritme 4

```
INICI_SEQÜÈNCIA
  REPETIR_FINS_QUE {final de laberint}

    FER {anar recte}
    FINS_QUE {intersecció} o {camí sense sortida}

    SI {camí sense sortida}
      LLAVORS {donar tomb de 180º} i {SS=true}
    FI_SI

    SI {intersecció}
      LLAVORS

        SI {intersecció de dos camins}
          LLAVORS
            SI {pot girar dreta}
              LLAVORS {girar dreta}
              SI_NO
                SI {pot girar esquerra}
                  LLAVORS {girar esquerra}
                  SI_NO {seguir recte}
                FI_SI
            FI_SI
        FI_SI

        SI {intersecció de tres o quatre camins}
          LLAVORS
```

```
    SI {pot girar dreta}
      LLAVORS
        SI {camí no està fet}
          LLAVORS {es pot girar dreta: D=true}
            SI {final de laberint}
              LLAVORS {girar dreta},{laberint resolt}
              SI_NO {saltar a comparació}
            FI_SI
          SI_NO {no es pot girar dreta: D=false},{saltar a comparació}
        FI_SI
      SI_NO {no es pot girar dreta: D=false},{saltar a comparació}
    FI_SI

    SI {pot girar esquerra}
      LLAVORS
        SI {camí no està fet}
          LLAVORS {es pot girar esquerra: E=true}
            SI {final de laberint}
              LLAVORS {girar esquerra},{laberint resolt}
              SI_NO {saltar a comparació}
            FI_SI
          SI_NO {no es pot girar esquerra: E=false},{saltar a comparació}
        FI_SI
      SI_NO {no es pot girar esquerra: E=false},{saltar a comparació}
    FI_SI

    SI {pot seguir recte}
      LLAVORS
        SI {camí no està fet}
          LLAVORS {es pot seguir recte: R=true}
            SI {final de laberint}
              LLAVORS {seguir recte},{laberint resolt}
              SI_NO {saltar a comparació}
            FI_SI
          SI_NO {no es pot seguir recte: R=false},{saltar a comparació}
        FI_SI
      SI_NO {no es pot seguir recte: R=false},{saltar a comparació}
    FI_SI

FI_SI


SELECCIONAR_CAS {comparació}

EN_CAS {D=E=R=false}
 SI {Dreta>Esquerra}
    LLAVORS
      SI {D>R}
        LLAVORS
          SI {SenseSortida=true}
            LLAVORS {marcar camí com SS},{girar dreta}
            SI_NO {girar dreta}
          FI_SI
        SI_NO
          SI {D=R}
```

```
          LLAVORS {saltar a: D ÉS >E}
          SI_NO
            SI {SS=true}
              LLAVORS {marcar camí com SS},{seguir recte}
              SI_NO {seguir recte}
            FI_SI
        FI_SI
      FI_SI
      SI_NO
        SI {Esquerra>Recte}
          LLAVORS
            SI {SS=true}
              LLAVORS {marcar camí com SS},{girar dreta}
              SI_NO {girar esquerra}
            FI_SI
          SI_NO

          SI {E=R}
            LLAVORS {saltar a: E ÉS > R}
            SI_NO
              SI {SS=true}
                LLAVORS {marcar camí com SS},{seguir recte}
                SI_NO {seguir recte}
              FI_SI
          FI_SI
        FI_SI
    FI_SI

EN CAS {D=E=true,R=false}
  SI {Dreta>Esquerra}
    LLAVORS
      SI {SS=true}
        LLAVORS {marcar camí com SS},{girar dreta}
        SI_NO {girar dreta}
      FI_SI
    SI_NO
      SI {Dreta=Esquerra}
        LLAVORS {saltar a: D ÉS > E}
        SI_NO
          SI {SS=true}
            LLAVORS {marcar camí com SS},{girar esquerra}
            SI_NO {girar esquerra}
          FI_SI
      FI_SI
  FI_SI

EN CAS {D=R=true,E=false}
  SI {Dreta>Recte}
    LLAVORS
      SI {SS=true}
        LLAVORS {marcar camí SS},{girar dreta}
```

```
            SI_NO
          FI_SI
        SI_NO
          SI {Dreta=Recte}
            LLAVORS {saltar a: D ÉS > R}
            SI_NO
              SI {SS=true}
                LLAVORS{marcar camí com SS},{seguir recte}
                SI_NO {seguir recte}
              FI_SI
          FI_SI
      FI_SI

    EN CAS {E=R=true,D=false}
      SI {Esquerra>Recte}
        LLAVORS
          SI {SS=true}
            LLAVORS {marcar com camí SS},{girar esquerra}
            SI_NO {girar esquerra}
          FI_SI
        SI_NO
          SI {Esquerra=Recte}
            LLAVORS {saltar a: E ÉS > R}
            SI_NO
              SI {SS=true}
                LLAVORS {marcar com camí SS},{seguir recte}
                SI_NO {seguir recte}
              FI_SI
          FI_SI
      FI_SI

    FI_SELECCIÓ

  FI_SI

 FI_REPETIR
FI_SEQÜÈNCIA
```

Fig. 4.21. Pseudocodi de l'algoritme 4

4.5.4.- Exemple de laberint resolt amb l'algoritme 4

Per veure com actua aquest algoritme, es pren d'exemple el laberint de la figura 4.22. Aquest laberint té una entrada que correspon a l'inici i una sortida que correspon al final. En ell s'hi troben varis camins sense sortida i interseccions de dos, tres i quatre camins. Per poder trobar la sortida, el robot haurà de creuar el laberint.

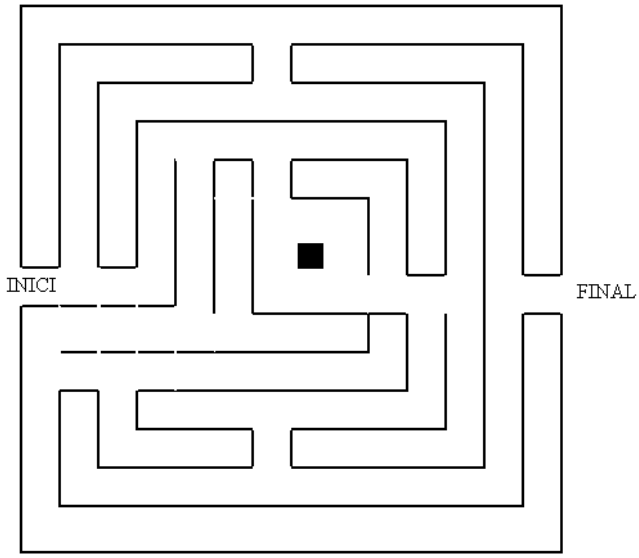


Fig. 4.22. Laberint d'exemple per l'algoritme 4

El robot que actuï segons l'algoritme 4, començarà la resolució del laberint partint del punt indicat com l'inici. La primera intersecció que es troba (número 1 en la figura 4.23) li permet fer un gir a l'esquerra o bé seguir recte. El robot comprovarà quin dels dos camins és el més llarg i finalment farà un gir cap a l'esquerra. El camí de l'esquerra, però, és un camí tancat i quan el robot detecti que no pot continuar, farà un gir de 180º i tornarà enrera pel camí on ha vingut fins arribar de nou a la intersecció 1. Quan es troba a la intersecció 1 per segona vegada, té la possibilitat de girar a la dreta o a l'esquerra, però com que pel camí de la dreta ja hi ha passat, farà un gir a l'esquerra.

La pròxima intersecció de més de dos camins amb la que es troba el robot, és la intersecció 2 on pot seguir recte o girar a la dreta. Com que el camí que hi ha al recte és més llarg que el de la dreta, el robot seguirà recte. Seguidament es troba amb la intersecció 3, on en aquest cas, el camí de la dreta és més gran que el recte i per tant, el robot fa un gir cap a la dreta.

La propera intersecció, és la 4. En aquesta intersecció, el robot pot girar a l'esquerra o seguir recte. Al fer la comprovació de quin camí és el més llarg, detecta que els dos camins són iguals, però com que l'algoritme 4 davant casos on els camins són igual de llargs dona més prioritat a girar cap a l'esquerra que seguir recte, el robot girarà a l'esquerra.

En la intersecció 5 el robot girarà cap a l'esquerra perquè el camí és més llarg que el de la dreta, i en la intersecció 6 torna a girar a l'esquerra pel mateix motiu. El girar a l'esquerra a la intersecció 6, el robot es fica en un camí sense sortida, i haurà de tornar enrere fins tornar-se a trobar a la intersecció 6, on aquesta vegada girarà cap a l'esquerra i es dirigirà a la intersecció 7. En la intersecció 7 el robot gira a la dreta ja que és el camí més llarg.

Després de passar per les interseccions 8 i 9 i fer els corresponents girs segons quin és el camí més llarg i segons si per un dels camins s'hi ha passat abans, el robot es troba finalment a la intersecció 10. En aquesta intersecció el robot pot girar a la dreta o seguir recte. El camí del recte és més gran que el camí de la dreta, però el robot detecta que en el camí de la dreta s'hi troba el final del laberint, per tant, farà un gir cap a la dreta i haurà finalitzat la resolució del laberint.

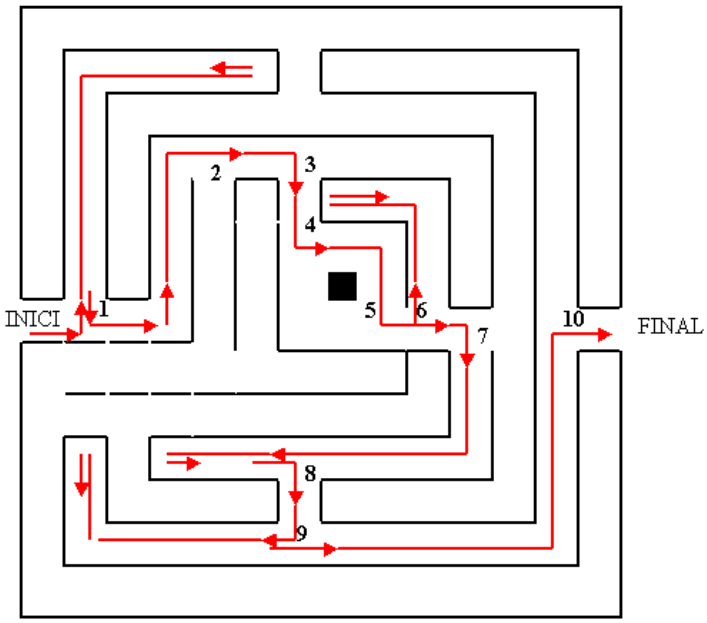


Fig. 4.23. Laberint d'exemple resolt amb l'algoritme 4

4.5.5.- Avantatges i inconvenients de l'algoritme 4

L'algoritme 4 és un algoritme que es pot aplicar a un gran nombre de laberints. El fet d'evitar passar per camins ja fets, fa que el robot no es pugui quedar atrapat en una determinada zona del laberint sense poder sortir-ne, i tampoc es perd temps passant per camins ja fets. Per altra banda, permet conèixer quins camins no tenen sortida. Una altra avantatge d'aquest algoritme, és que quan detecta el final del laberint en una intersecció, dirigeix el robot directament cap al final, sense tenir en compte quin camí és el més llarg.

Aquest algoritme té l'inconvenient de que és força complex i això fa que el robot tardi en decidir cap a quin cantó a de girar davant una intersecció. Aquest fet, fa que el robot es mogui a una velocitat lenta. Un altre inconvenient és que no es pot aplicar a laberints amb nombroses sortides ja que no detecta quan un camí condueix a la sortida del laberint.

5.- PROGRAMA SIMULABER

El programa simulador de resolució de laberints, SIMULABER, és una eina que permet visualitzar els moviments que realitzaria un robot per tal de resoldre un laberint. Es podrà observar com actua aquest robot en cada una de les diferents situacions en les que pot trobar-se dins d'un laberint així com els moviments que farà per anar des de l'inici al final del laberint.

La resolució dels laberints es fa segons uns determinats algoritmes de control. Aquests algoritmes determinen quin criteri de decisió seguirà el robot davant d'una intersecció per tal d'escollir quin camí és el més adequat. El programa disposa de quatre algoritmes diferents, cada un d'ells pensat per resoldre un tipus de laberint.

Cada un dels laberints representen una situació diferent i desconeguda pel propi robot. Davant cada una d'aquestes situacions, es podrà provar la reacció del robot segons l'algoritme escollit, fet que permetrà a l'usuari decidir quins algoritmes són adequats i quins no ho són per cada laberint en concret.

El simulador, permet provar els algoritmes en laberints ja fets, o bé crear laberints nous al gust de l'usuari, la qual cosa proporciona una amplia flexibilitat al programa.

La utilització del simulador és de gran utilitat si es vol dissenyar un robot ja que permet detectar les possibles deficiències del software abans d'arribar a construir-lo. Es veu doncs que aquest programa és una eina útil a l'hora de dissenyar i construir robots, la funció dels quals sigui desplaçar-se per zones desconegudes amb objectiu d'explorar la zona per determinar-ne les característiques, o bé per trobar-hi una sortida.

5.1.- PARTS DEL PROGRAMA I ELEMENTS QUE L'INTEGREN

Per poder entendre i utilitzar correctament el programa caldrà en primer lloc, conèixer com està creat el programa simulador i de quins elements està compost.

5.1.1.- Per què Visual Basic?

El programa simulador de laberints està creat amb el programa Microsoft Visual Basic 6.0. Aquest programa està orientat a la realització de programes per Windows, i pot incorporar tots els elements d'aquest entorn informàtic: finestres, botons, caixes de diàleg i de text, botons d'opció i selecció, barres de desplaçament, gràfics, menús, etc.

Pràcticament tots els elements d'interacció amb l'usuari dels que disposa Windows, poden ser programats en Visual Basic 6.0 de forma senzilla. A més a més, permet desenvolupar aplicacions complexes en poc temps, en comparació amb altres llenguatges de programació com per exemple Visual C++. El preu que s'ha de

pagar per utilitzar Visual Basic 6.0 és una menor velocitat o eficiència en les aplicacions.

Visual Basic 6.0 és un programa basat en objectes, on cada objecte té assignades unes propietats, i sobre els quals hi actuen uns determinats esdeveniments.

Visual Basic 6.0 es a més un llenguatge de programació visual anomenat també llenguatge de 4ª generació. Això significa que un gran número de funcions es realitzen sense escriure codi, simplement amb operacions gràfiques realitzades amb el ratolí sobre la pantalla.

L'elecció de Visual Basic 6.0 com a eina per la creació del programa Simulaber, ve justificada per la similitud amb l'entorn Windows i pel fàcil aprenentatge d'aquest llenguatge.

5.1.2.- Parts del programa Simulaber

El programa Simulaber està format per quatre formularis: Menú, ExempleLab, CrearLab i frmAbout.

Formulari **Menú**:

Fig. 5.1. Menú del formulari Menú

El formulari principal és el formulari Menú. Aquest formulari és el formulari que apareix al executar el programa i és des d'on s'accedeix als altres dos formularis. La funció del formulari Menú, no és cap altre que la de presentar el programa i permetre l'accés als formularis ExempleLab i CrearLab.

Aquest formulari mostra una llista amb els menús disponibles:

- Arxiu
- Exemples
- Crear Laberint
- Ajuda

Menú Arxiu

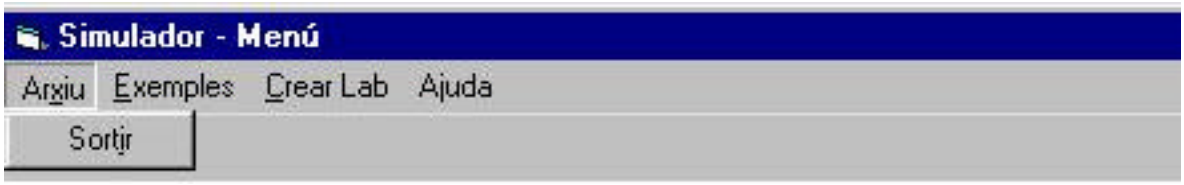


Fig.5.2. Menú Arxiu del formulari Menú

Opcions:

?? Arxiu **Sortir**
Tanca el programa.

Menú Exemples

Fig.5.3. Menú Exemples del formulari Menú

Aquest menú permet accedir directament a un dels quatre laberints d'exemples que hi ha al formulari ExempleLab.

Opcions:
?? Exemple **Laberint 1**
?? Exemple **Laberint 2**
?? Exemple **Laberint 3**
?? Exemple **Laberint 4**

Menú Crear Laberint

Fig.5.4. Menú CrearLab del formulari Menú

Aquest menú permet accedir al formulari CrearLab on es podrà crear un laberint nou o modificar algun de ja existent.

Opcions:
?? Crear Lab **Nou**
Accedeix al formulari crear laberint i surt un missatge informatiu indicant que es seleccioni Nou en el menú Arxiu del formulari CrearLab.

Fig.5.5. Caixa de missatge per crear un arxiu Nou

?? Crear Lab **Modificar**

Accedeix al formulari crear laberint i surt un missatge informatiu indicant que es seleccioni Obrir en el menú Arxiu del formulari CrearLab.

Fig.5.6. Caixa de missatge per obrir un arxiu existent

Menú Ajuda

Fig.5.7. Menú Ajuda del formulari Menú

El menú ajuda permet accedir a l'ajuda del programa on s'explica el funcionament del programa, les parts que el composen, com utilitzar el programa i les regles d'utilització, informació sobre els algoritmes de control, sobre com veure la resolució d'un laberint d'exemple o com crear un laberint nou, i informació sobre el programa.

Aquest menú d'ajuda, és comú als tres formularis i s'hi pot accedir des de qualsevol d'ells.

Opcions:

?? Ajuda **Contingut**

Obre la carpeta Contingut de l'ajuda del programa, des d'on es pot accedir a tots els temes que inclou l'ajuda

?? Ajuda **Índex**
Permet accedir a l'ajuda d'un determinat tema introduint el nom del tema.
?? Ajuda **Ajuda**
Obre el quadre de diàleg d'ajuda de Windows
?? Ajuda **Sobre el SIMULALBER**
Es mostra un formulari amb les característiques del programa.

Formulari **ExempleLab**

Fig.5.8. Menú del formulari Exemple laberint

Al formulari ExempleLab s'hi accedeix des del menú Exemples del formulari Menú. La funció del formulari ExempleLab, és la de veure la resolució de laberints dels quals disposa el programa. Aquest formulari permet escollir un dels quatre laberints disponibles i veure la resolució que efectuaria un robot segons l'algoritme de control escollit. Hi ha quatre algoritmes diferents i es pot provar amb quin d'ells es podria resoldre el laberint de millor forma.

Aquest formulari mostra una llista amb els menús disponibles:

?? Arxiu
?? Edició
?? Ajuda

Menú Arxiu

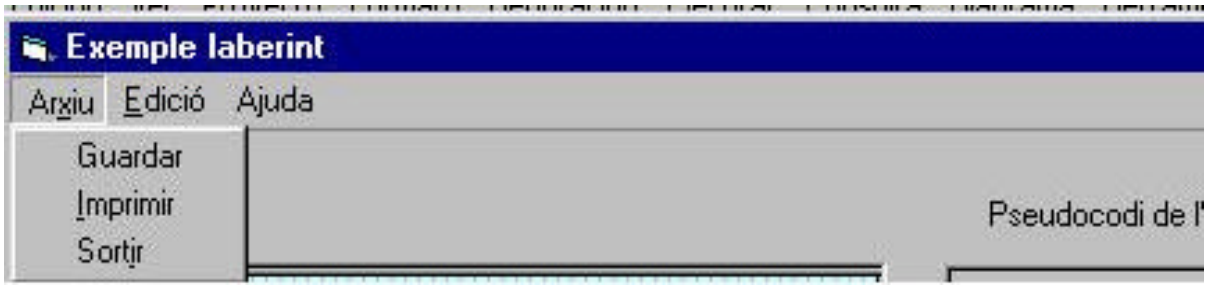


Fig.5.9. Menú Arxiu del formulari ExempleLab

Aquest menú té dues opcions, guardar i sortir.

Opcions:

?? Arxiu **Guardar**
Guarda el dibuix del laberint amb el nom i la ubicació indicats.
?? Arxiu Imprimir
Imprimeix la graella amb el laberint dibuixat.

?? Arxiu **Sortir**

Tanca el formulari de Exemple laberint, després de preguntar si realment es desitja sortir.

Fig.5.10. Caixa de missatge per sortir del formulari CrearLab

Menú Edició

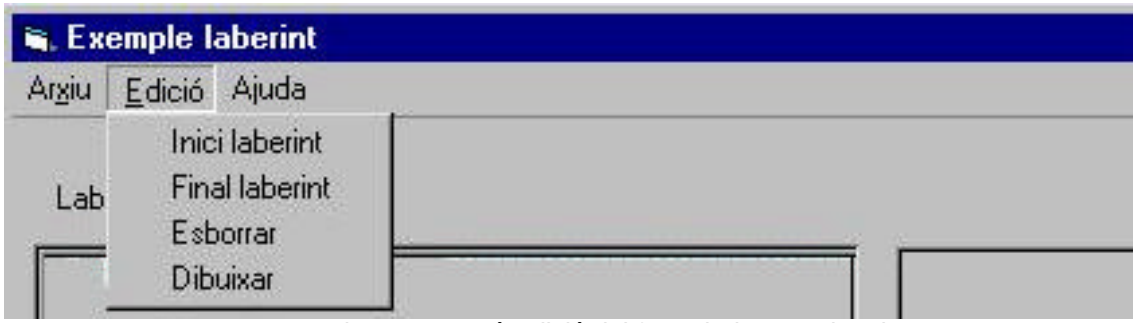


Fig.5.11. Menú Edició del formulari ExempleLab

Aquest menú permet fer modificacions en els laberints que hi ha d'exemple.

Opcions:

?? Edició **Inici Laberint**

Permet establir un nou punt d'inici del laberint. Per situar el punt, cal clicar amb el ratolí sobre el punt desitjat del laberint.

?? Edició **Final Laberint**

Permet establir un nou punt de final del laberint. Per situar el punt, cal clicar amb el ratolí sobre el punt desitjat del laberint.

?? Edició **Esborrar**

Permet esborrar qualsevol part del laberint. Per esborrar un punt, cal clicar amb el ratolí sobre el punt desitjat del laberint.

?? Edició **Dibuixar**

Permet dibuixar parets del laberint. Per dibuixar un punt, cal clicar amb el ratolí sobre el punt desitjat del laberint

Menú Ajuda

S'accedeix a la ajuda del programa.

Formulari **CrearLab**

Fig.5.12. Menú del formulari CrearLab

Al formulari CrearLab s'hi accedeix des del menú CrearLab del formulari Menú. La funció del formulari CrearLab, és la de poder crear laberints i provar de resoldre'ls amb algun dels quatre algoritmes disponibles.

Aquest formulari mostra una llista amb els menús disponibles:

- Arxiu
- Algoritme
- Ajuda

Menú Arxiu

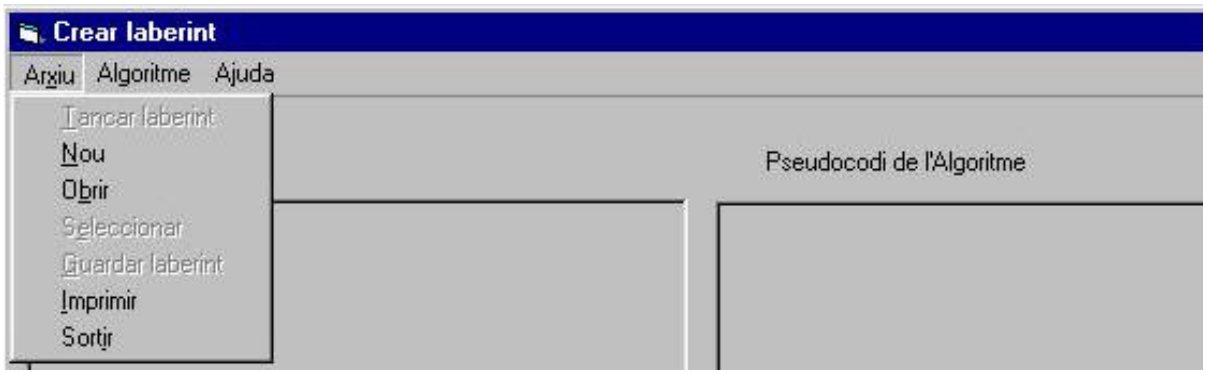


Fig.5.13. Menú Arxiu del formulari CrearLab

El menú arxiu del formulari CrearLab, té les opcions de tancar un laberint obert, crear un laberint nou, obrir un laberint existent, seleccionar el laberint i guardar-lo, i sortir del formulari.

Opcions:

- Arxiu **Tancar laberint**

  Tanca el laberint existent després de preguntar si realment es vol tancar. Aquesta opció s'habilita quan hi ha algun laberint dibuixat.

Fig.5.14. Caixa de diàleg per tancar el laberint

?? Arxiu **Nou**

Crea una graella nova. Si ja existeix una graella amb algun dibuix fet, abans de crear una altre graella nova, pregunta si es vol guardar la graella existent:

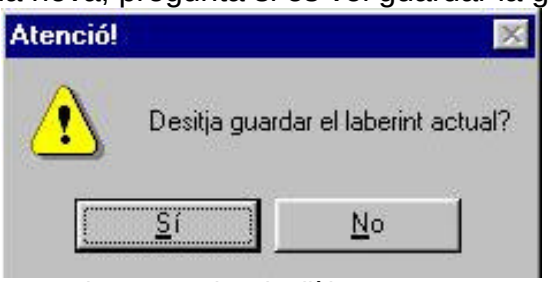


Fig.5.15. Caixa de diàleg per crear un laberint Nou

?? Arxiu **Obrir**

Obra el quadre de diàleg obrir per obrir un arxiu ja existent.

?? Arxiu **Seleccionar**

Selecciona tota la graella.

?? Arxiu **Guardar**

Guarda la graella amb el dibuix del laberint amb el nom i la ubicació indicats. Aquesta opció s'habilita després d'haver seleccionat la graella amb el laberint.

?? Arxiu **Imprimir**

Imprimeix la graella amb el dibuix del laberint.

?? Arxiu **Sortir**

Tanca el formulari i torna al formulari principal després de preguntar si realment es vol sortir.

Menú Algoritme

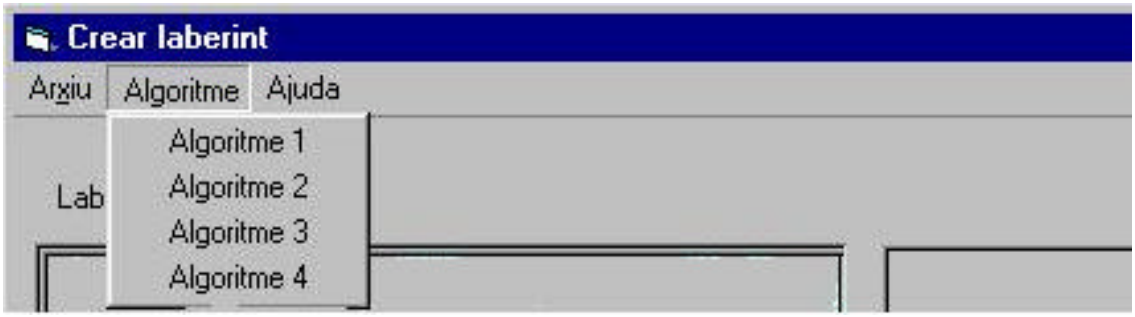


*Fig.5.16. Menú A*lgoritme *del formulari CrearLab*

Aquest menú permet escollir un dels quatre algoritmes que estan disponibles en el programa. Aquest menú està inhabilitat fins que no es guarda el laberint creat.

Opcions:

?? Algoritme 1
?? Algoritme 2
?? Algoritme 3
?? Algoritme 4

Menú Ajuda

S'accedeix a la ajuda del programa.

Formulari frmAbout

Aquest formulari mostra informació sobre el programa Simulab. En ell s'hi indica la versió del programa, la utilitat i una nota sobre les lleis que el protegeixen.

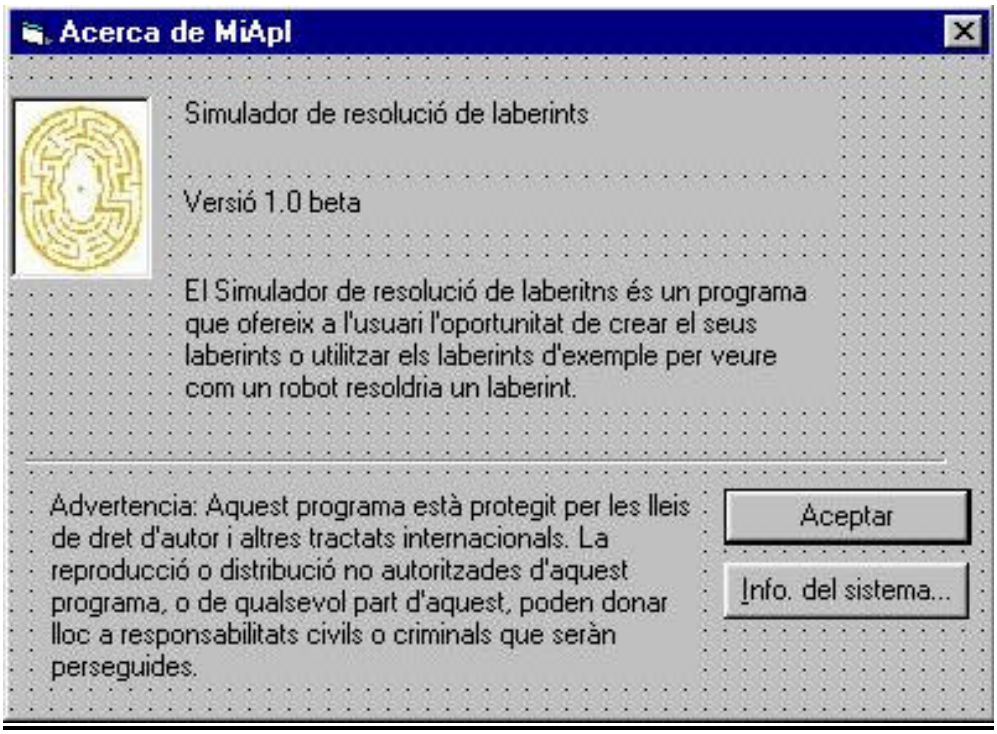


Fig.5.17. Formulari Sobre el SIMULABER

5.1.3.- Elements que integren el programa

Per conèixer bé els elements que formen el programa en primer lloc s'explicaran els controls utilitzats i seguidament s'analitzarà cada formulari per separat per poder explicar detingudament cada un dels elements dels que està compost el formulari.

Controls utilitzats

Els controls que s'expliquen a continuació són els controls utilitzats per crear els formularis. Aquest controls els proporciona el programa Visual Basic 6.0.

Botó de comandament (Command Button)

Permeten iniciar, interrompre o acabar un procés. Quan es fa clic en ell, invoca la comanda escrita en el seu procediment.

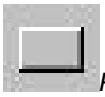


Fig. 5.18. Command Button

Botons d'opció (Option Button)

S'utilitzen per presentar opcions, normalment en grups de botons d'opció, entre els quals l'usuari pot escollir-ne un. Quan un usuari selecciona un botó d'opció, els altres controls Option Button del mateix grup deixen d'estar disponibles automàticament.


Fig. 5.19. Option Button

Barres de desplaçament (Scroll Bars)

Proporcionen un medi senzill de recorre llargues llistes d'elements o grans quantitats d'informació al desplaçar-se horitzontal o verticalment dins d'una aplicació o d'un control.

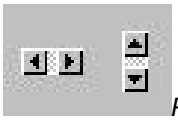


Fig. 5.20. Scroll Bars

Caixes combinades (Combo Box)

Els quadres combinats presenten una llista d'opcions a l'usuari. Permet que l'usuari seleccioni un element; per fer-ho, s'ha d'escriure text en el quadre de text o seleccionar-lo en la llista.


Fig. 5.21. Combo Box

Caixes de diàleg estàndard (Common Dialog)

El control Common Dialog proporciona un conjunt de quadres de diàleg estàndard per operacions com obrir i guardar arxius, establir opcions d'impressió i seleccionar colors o fonts. També permet presentar ajuda mitjançant el sistema d'ajuda de Windows.


Fig. 5.22. Common Dialog

Caixes de missatge (MsgBox)

La caixa de missatge obre una finestra a través de la qual s'envia un missatge a l'usuari i se li demana una resposta.

Caixa de text (Text Box)

S'utilitza per presentar informació escrita per l'usuari en temps d'execució o assignada a la propietat Text del control en temps de disseny o d'execució.


Fig. 5.23. Text Box

Control Timer

Els controls Timer responen al pas del temps. Són independents de l'usuari i poden programar-se perquè executin accions a intervals periòdics de temps.


Fig. 5.24. Timer

Etiquetes (Labels)

S'utilitzen per presentar text i l'usuari no pot modificar-les. Es poden utilitzar per identificar els objectes d'un formulari.


Fig. 5.25. Label

Graella (MSFlexGrid)

És un control de tipus fulla de càlcul que mostra una sèrie de files i columnes que representen registres i camps d'un objecte. Proporciona formes avançades per presentar dades en una quadrícula.


Fig. 5.26. MSFlexGrid

Marc (Frame)

Els controls Frame s'utilitzen per proporcionar una agrupació identificable per altres controls. Separa grups d'altres botons.

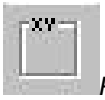


Fig. 5.27.Frame

Menús

Visual Basic permet crear barres de menús amb l'editor de menús. S'obre una finestra on es poden anar introduint les opcions que formaran el menú.


Fig. 5.28. Editor de menús

Quadre de dibuix (Picture Box)

S'utilitza per presentar gràfics, per actuar com contenidor d'altres controls i per presentar el resultat dels mètodes gràfics o text amb el mètode Print.


Fig. 5.29. Picture Box

Controls del formulari Menú

El formulari Menú, no té cap control. En aquest formulari només apareix una barra de menús que ha estat creada amb l'editor de menús.

Controls del formulari ExempleLab

La majoria dels controls que hi ha al formulari ExempleLab, es mostren a la figura 5.31. Es poden observar cinc etiquetes, dos caixes combinades, una barra de desplaçament horitzontal, una caixa de text, un marc, dos botons d'opció, tres botons de comandament i dos graelles. S'explicarà el contingut de cada control, la seva utilitat i el nom amb el que el programa els reconeix.

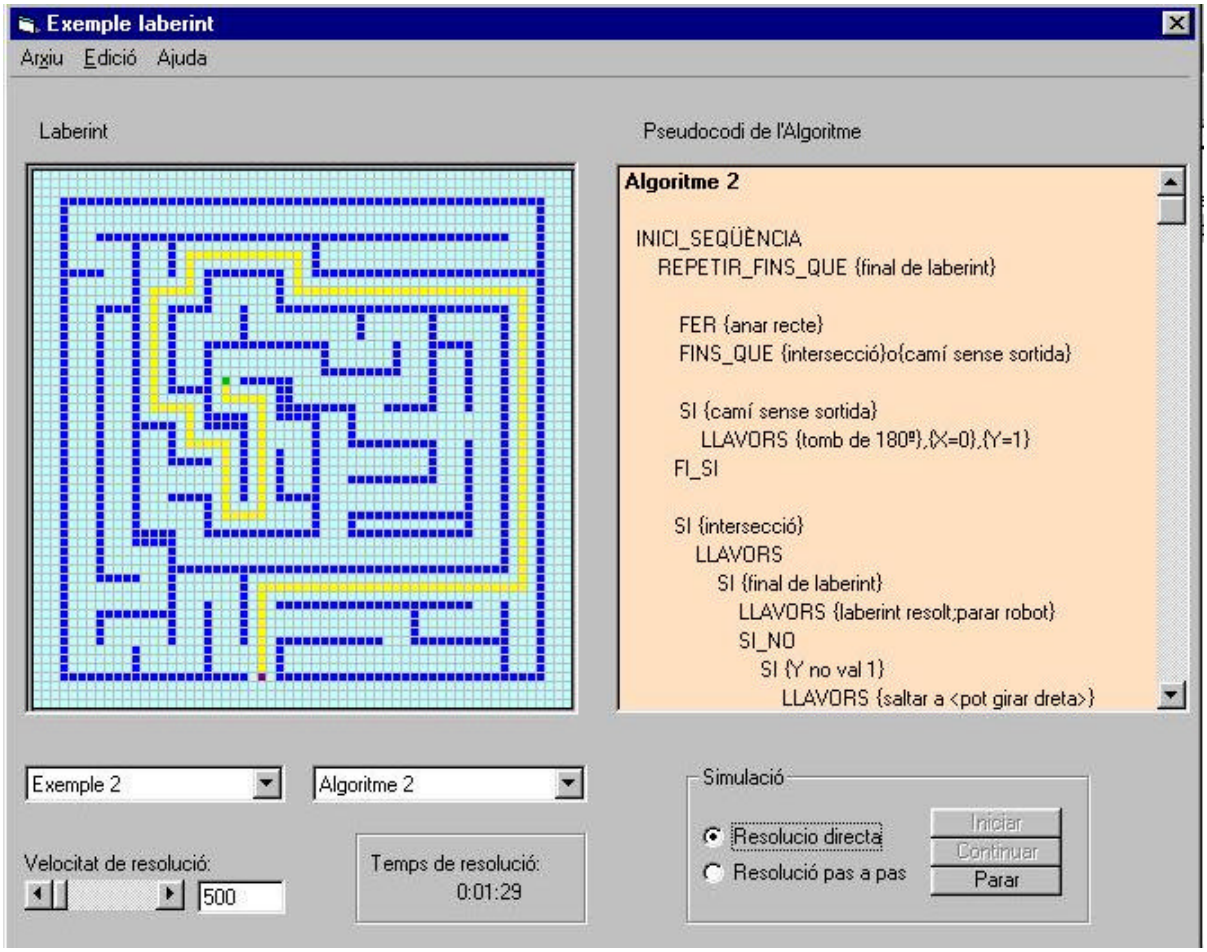


Fig. 5.31. Controls del formulari ExempleLab

lblLab2

És el nom amb el que el programa reconeix l'etiqueta on es llegeix "Laberint". S'utilitza per indicar què és el que es mostra a sota seu.

lblPscodi2

Nom amb el que el programa reconeix l'etiqueta on es llegeix "Pseudocodi de l'algoritme". Indica que el que hi ha a sota seu és el pseudocodi de l'algoritme utilitzat per resoldre el laberint.

cboExe

És el nom amb el que el programa reconeix la caixa combinada on es permet a l'usuari escollir algun dels quatre exemples que hi ha. Els exemples fan referència a laberints els quals es mostren en una graella.

cboAlg

Nom de la caixa combinada on es permet a l'usuari escollir algun dels quatre algoritmes que hi ha. Al escollir un algoritme, apareix el pseudocodi corresponent en

la graella GridAlg. També apareixen visibles els controls fraSim2, optResD2, optResP2, cmdIni2, cmdCon2 i cmdPar2, que inicialment no hi eren.

PicLab2

És el nom que rep el quadre de dibuix que està situat sota l'etiqueta lblLab2. Aquest quadre actua com a contenidor de quatre graelles les quals mostren el dibuix dels laberints d'exemple.

GridExeX

El formulari conté quatre graelles amb el dibuix de quatre laberints d'exemple. Aquestes graelles estan totes contingudes en el quadre de dibuix PicLab2 i segons l'exemple seleccionat en la caixa combinada cboExe, apareix visible una d'elles.

Aquestes graelles estan formades per 60 files i 60 columnes que formen quadres de mida bastant reduïda.

Inicialment es pretenia que aquestes graelles tinguessin 100 files x 100 columnes sense variar les dimensions de la graella, però llavors la mida de les caselles resultava ser molt petita. Això tenia una avantatge i era que permetia dibuixar laberints amb formes corbes ja que hi havia més precisió a l'hora de dibuixar. Si es va decidir augmentar les mides dels quadres i disminuir la quantitat, va ser degut als problemes que presentava una graella amb tantes caselles i tan petites. En primer lloc, es feia difícil dibuixar les parets del laberint degut a la dificultat de distingir els límits d'uns quadrets tant petits la qual cosa requeria forçar molt la vista per poder seleccionar el quadret desitjat. Un altre problema important apareixia quan s'havia de carregar el dibuix. Si es dibuixava un laberint i es guardava, si en algun moment volia recuperar-se aquest laberint guardat, el programa carregava la graella i després revisava quadret a quadret per pintar-lo del color corresponent segons si era paret, camí, inici o final. En una graella de 10 000 quadrets (100 x 100) aquest procés era extremadament lent i no interessava. Per aquestes dues raons, es va decidir que les graelles estiguessin formades per 3600 quadrets (60x60) i la mida dels quadrets fos més gran.

Amb aquests canvis es va solucionar el problema de seleccionar els quadrets, però tot i que es va aconseguir augmentar molt la velocitat a l'hora d'obrir una graella amb un laberint, no es va poder eliminar aquest problema ja que el programa continuava sent lent obrint un arxiu. A conseqüència d'això, es va optar per no obrir les graelles des d'un arxiu guardat cada vegada que es seleccionés un laberint, sinó que els quatre laberints d'exemple estan escrits per codi en el formulari ExempleLab, i en aquest formulari no apareix la possibilitat d'obrir arxius guardats. En el formulari CrearLab, en canvi, sí es podran obrir arxius.

Un altre aspecte a tenir en compte pel que fa a les graelles, és la forma de presentar els laberints. Els laberints d'exemple que hi ha dibuixats, estan formats per un **fons blau cel**, amb les **parets** d'un **blau** més **fosc**, el punt d**'inici de color verd**, i el punt **final de color lila**. Quan es resol el laberint, el recorregut que fa el robot es pinta de color groc. Aquests colors amb els quals es presenta el laberint, en realitat només serveixen perquè l'usuari tingui una visió clara del laberint, ja que el

programa no distingeix els colors de les caselles alhora de diferenciar entre parets, camins, inici i final. El programa només detecta el contingut de les caselles, és a dir, el text que conté cada casella. Així doncs, tot i que l'usuari no ho vegi, cada color té associat un caràcter diferent que està escrit, en mides molt reduïdes, a cada un dels quadrets segons el color que tinguin. Quan el programa resolt un laberint, el que està fent, és comparar el caràcter de cada quadret per comprovar si es tracta de paret, camí, inici o final.

De la mateixa manera, quan es selecciona el laberint i es guarda, només es guarden els caràcters de cada casella i no els colors. Això explica la lentitud del programa quan obre un arxiu, ja que, al obrir un arxiu, apareix directament la graella amb els caràcters corresponents a cada casella, tot i que a la vista de l'usuari, no es veuen. Per poder mostrar bé el laberint, es recorre casella a casella i es canvia el color de cada una segons el caràcter que contingui.

El caràcter associat a: una paret és "X"; l'inici és "I"; al final és "F"; un camí és " "; un camí ja fet és "O".

Cal indicar, que s'han utilitzat graelles (MSFlexGrid) per dibuixar els laberints perquè permeten identificar amb format de fila i columna cada posició de la graella, perquè permeten escriure i canviar les propietats com per exemple el color de cada casella individualment, perquè permeten fer operacions clicant amb el ratolí, perquè permeten escollir el número de files, columnes i l'amplada i altura d'aquestes, perquè permeten guardar molta informació en un sol control i perquè ofereixen un ampli ventall de possibilitats útils per l'aplicació que se'ls hi dona en el programa Simulaber que altres controls de Visual Basic no ofereixen.

La graella que correspon al laberint d'exemple 1 s'anomena GridExe1, la que correspon a l'exemple 2 s'anomena GridExe2, la corresponent al laberint 3 es diu GridExe3 i el nom de la graella de l'exemple quatre és GridExe4. Amb aquests noms és com el programa identifica els controls MSFlexGrid.

?? GridExe1

El laberint que conté aquesta graella és el que mostra la figura següent:

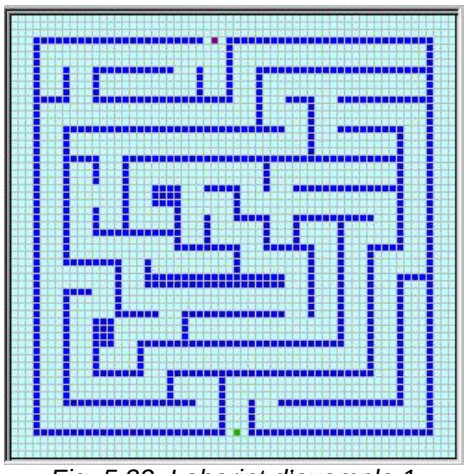


Fig. 5.32. Laberint d'exemple 1

En aquest exemple el robot parteix d'un punt inicial i ha de travessar tot el laberint fins trobar el final que es troba just al costat contrari.

És un laberint on s'hi poden trobar trams rectes, girs, interseccions de tres i quatre camins, i camins sense sortida. Aquest laberint no presenta gaires dificultats i pot resoldre's amb algoritmes senzills com és el cas de l'algoritme 1 i l'algoritme 2.

?? GridExe2

El laberint que conté aquesta graella és el que mostra la figura 5.33.

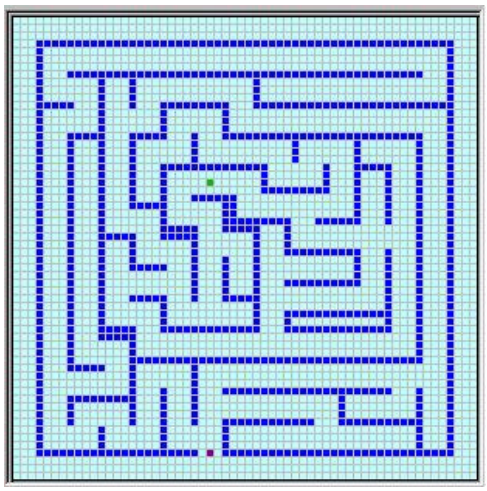


Fig. 5.33. Laberint d'exemple 2

El laberint de l'exemple 2 és molt similar al de l'exemple 1, però es diferencien bàsicament en que el laberint 2 està format per molts camins sense sortida. En aquest cas el robot partiria del centre del laberint.

Aquest tampoc és un laberint amb moltes dificultats, només caldrà trobar un algoritme de resolució que procuri evitar els camins sense sortida. Un dels algoritmes que pot anar bé per resoldre'l, és l'algoritme 2.

?? GridExe3

Aquesta graella conté el laberint de la figura 5.34.

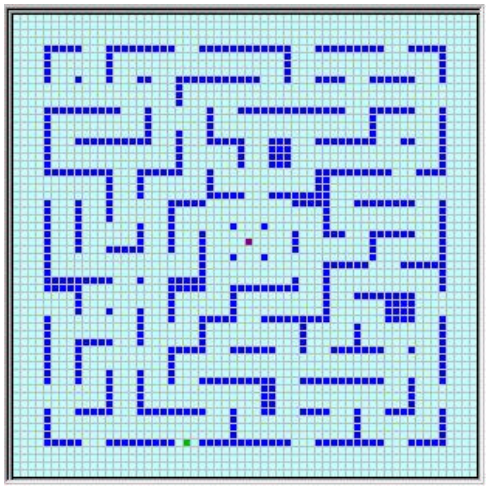


Fig. 5.34. Laberint d'exemple 3

El laberint de l'exemple 3 és un laberint molt diferent als de l'exemple 1 i 2. En aquest cas el laberint no té cap camí sense sortida, però té la característica de que hi ha nombroses sortides a l'exterior. Aquestes sortides, representaran una dificultat pel robot ja que el robot no haurà de sortir per cap d'elles sinó que haurà de dirigir-se cap al punt que indiqui el final, que en el cas de l'exemple, es troba en el centre del laberint.

Aquest és un laberint complicat des del punt de vista d'un robot, ja que caldrà trobar un algoritme de resolució que detecti els límits del laberint per impedir que el robot en surti. L'algoritme adequat per resoldre aquest laberint, és l'algoritme 3.

?? GridExe4

La figura 5.35 mostra el laberint que conté aquesta graella:

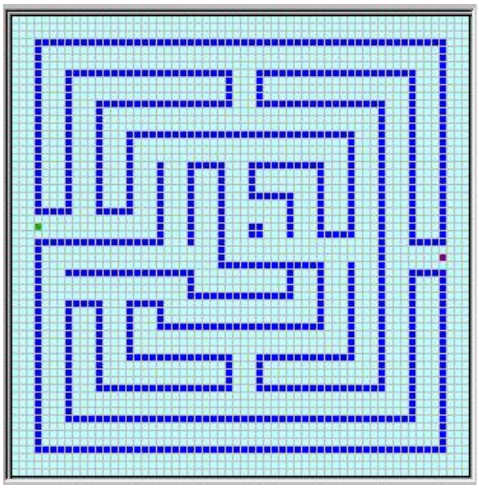


Fig. 5.35 Laberint d'exemple 4

Aquest laberint està format per un seguit de camins continus que es van entrelligant entre ells de manera que formen diversos camins sense sortida dels quals és difícil sortir-ne.

Aquest laberint té una única entrada i una única sortida i si no s'escull l'algoritme adequat per a la seva resolució, es possible tardar molt, o bé no arribar mai a trobar la sortida.

L'algoritme que es necessitaria per resoldre un laberint d'aquestes característiques no seria un algoritme senzill. L'algoritme 4, per exemple, permet resoldre aquest laberint.

lblVel2

Etiqueta que indica que l'objecte que hi ha a sota permet regular la velocitat de resolució.

hsb2

Nom de l'objecte ScrollBar que permet regular la velocitat de resolució del laberint. Apareix quan es selecciona un dels dos botons d'opció per escollir el tipus de resolució. Per defecte, està a la posició màxima.

txtms2

Nom de la caixa de text que mostra la velocitat de regulació escollida amb el hsb2. Apareix en el mateix moment que el hsb2.

GridAlg

Aquest nom fa referència a la graella que mostra els Pseudocodis dels algoritmes. El contingut de les caselles de la graella, canvia en funció de l'algoritme de resolució escollit. Si no hi ha cap algoritme seleccionat, la graella està buida.

La graella GridAlg, té una única columna i un número de files que també estarà en funció de l'algoritme escollit, ja que no tots els pseudocodis són igual de llargs.

Per desplaçar-se per la graella, apareix automàticament una barra de desplaçament vertical.

En aquesta graella, no es mostren les línies de separació entre files per motius estètics ja que queda més uniforme sense separacions.

S'ha escollit un control MSFlexGrid per presentar els pseudocodis perquè permet mostrar molta informació en un sol control i perquè es pot modificar el contingut de les caselles fàcilment.

fraSim2

És el nom que rep el marc que conté les opcions de resolució. Al carregar el formulari, aquest marc, juntament amb els objectes que conté, no està visible. Apareix al seleccionar un algoritme de resolució en el cboAlg.

optResD2

Nom del botó d'opció que permet seleccionar una resolució directa. Aquest botó no està visible fins que no s'ha seleccionat un algoritme de resolució. Al polsar el optResD2, s'habilita el botó de comanda Iniciar.

optResP2

Nom del botó d'opció que permet seleccionar una resolució pas a pas. Aquest botó no està visible fins que no s'ha seleccionat un algoritme de resolució. Al polsar el optResD2, s'habilita el botó de comanda Iniciar.

cmdIni2

Amb aquest nom s'identifica al botó de comandament Iniciar. Aquest botó inicialment no apareix i es fa visible quan es selecciona un algoritme de resolució, però només s'habilita quan s'escull un dels dos botons d'opció que indiquen el tipus de resolució.

El cmdIni2 serveix per iniciar la seqüència de moviments que ha de fer el robot per resoldre el laberint. Una vegada es polsa, queda inhabilitat fins que no es polsa el botó Parar.

En el cas d'escollir una resolució pas a pas, al polsar el botó cmdIni2, aquest queda inhabilitat, però s'habiliten els botons cmdCon2 i cmdPar2. Si el tipus de resolució és la directa, només s'habilita el botó cmdPar2.

cmdCon2

Nom del botó Continuar el qual només s'habilita al polsar el botó cmdIni2 si està escollida la resolució pas a pas.

Serveix per continuar els moviments que ha iniciat el robot per resoldre el laberint.

cmdPar2

Aquest botó serveix per Parar la resolució del laberint un cop iniciada aquesta, però només podrà polsar-se quan el robot estigui aturat perquè es tracta d'una resolució pas a pas o quan el programa dona l'opció de parar la resolució en cas d'haver escollit una resolució directa.

Una vegada trobat el final del laberint, també és convenient polsar el botó Parar.

fratemps2

Aquest marc apareix una vegada finalitzada la resolució del laberint i en el seu interior conté dos etiquetes que indiquen el temps de resolució del laberint.

lblTem2

Fa referència a l'etiqueta que indica "Temps de resolució". Està situada dins el marc fratemps2 i apareix un cop finalitzada la resolució del laberint.

lbltemps2

Nom de l'etiqueta que indica el temps transcorregut fins que s'ha iniciat la resolució del laberint, fins que ha finalitzat. El temps es presenta en hores:minuts:segons.

Apareix un cop finalitzada la resolució.

Menú

El formulari ExempleLab, conté un menú amb una sèrie d'opcions. Aquest menú s'explica a l'apartat 5.1.2 d'aquest mateix capítol.

Controls del formulari CrearLab

Els controls del formulari CrearLab es mostren a la figura 5.36. S'observen dues etiquetes, dos marcs, sis botons d'opció, cinc botons de comanda i dos graelles. S'explicarà el contingut de cada control, la seva utilitat i el nom amb el que el programa els reconeix.

lblLab

És el nom amb el que el programa reconeix l'etiqueta on es llegeix "Laberint". S'utilitza per indicar què és el que es mostra a sota seu.

lblPscodi

Nom amb el que el programa reconeix l'etiqueta on es llegeix "Pseudocodi de l'algoritme". Indica que el que hi ha a sota seu és el pseudocodi de l'algoritme utilitzat per resoldre el laberint.

PicLab

És el nom que rep el quadre de dibuix que està situat sota l'etiqueta lblLab2. Aquest quadre actua com a contenidor de la graella Grid1, la qual permetrà crear laberints o obrir-ne d'existents

Grid1

Aquesta graella és igual que les graelles GridExeX que conté el formulari ExempleLab, però es diferència d'elles en que en el cas de la Grid1, és una graella buida on no hi ha, inicialment, cap laberint dibuixat.

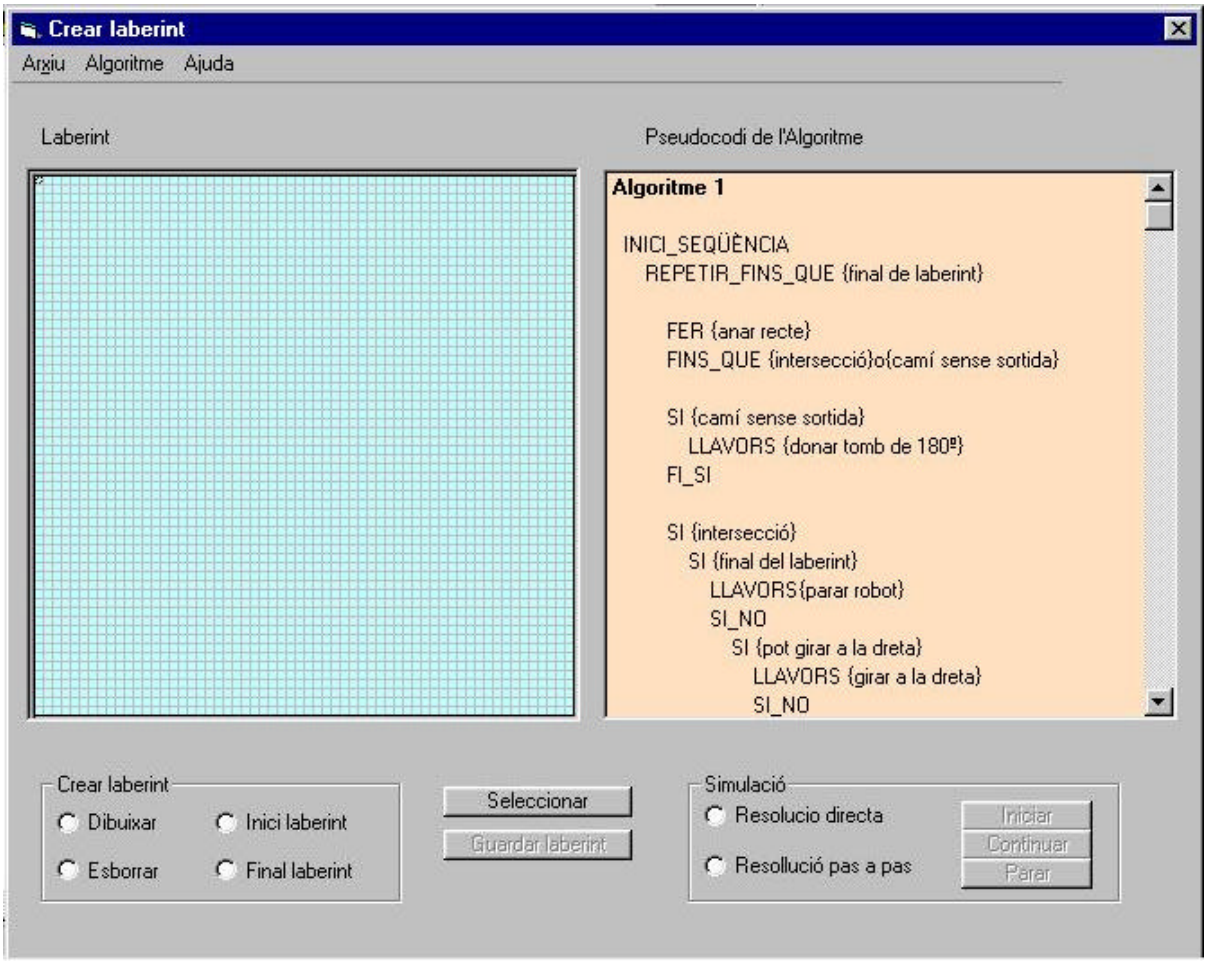


Fig 5.36. Controls del formulari CrearLab

fraCre

Nom del marc que conté els botons d'opció que permeten dibuixar el laberint. Desapareix, juntament amb els controls que conté, quan es selecciona un tipus de resolució.

optDib

Nom del botó d'opció que permet dibuixar les parets del laberint en la graella Grid1. Està dins el fraCre.

optEsb

Nom del botó d'opció que permet esborrar qualsevol dibuix fet en el laberint de la graella Grid1. Està dins el fraCre.

optIni

Nom del botó d'opció que permet situar el punt d'inici en el laberint de la graella Grid1. Està dins el fraCre.

optFin

Nom del botó d'opció que permet situar el punt final en el laberint de la graella Grid1. Està dins el fraCre.

cmdSel

Amb aquest nom es fa referència al botó de comandament "Seleccionar". Aquest botó permet seleccionar la graella per després guardar el seu contingut.

cmdGua

Aquest botó de comandament permet guardar el laberint prèviament seleccionat amb el cmdSel. Al polsar el cmdGua, s'obre un quadre de diàleg estàndard que permetrà assignar un nom al laberint i una ubicació.

lblVel

Etiqueta que indica que l'objecte que hi ha a sota permet regular la velocitat de resolució. Apareix al seleccionar un tipus de resolució

hsb

Nom de l'objecte ScrollBar que permet regular la velocitat de resolució del laberint. Apareix quan es selecciona un dels dos botons d'opció per escollir el tipus de resolució. Per defecte, està a la posició màxima.

txtms2

Nom de la caixa de text que mostra la velocitat de regulació escollida amb el hsb. Apareix en el mateix moment que el hsb.

GridAlg

Aquesta graella és idèntica a la del formulari ExempleLab.

fraSim

És el nom que rep el marc que conté les opcions de resolució. Al carregar el formulari, aquest marc, juntament amb els objectes que conté, no està visible. Apareix al seleccionar un algoritme de resolució en el menú.

optResD

Nom del botó d'opció que permet seleccionar una resolució directa. Aquest botó no està visible fins que no s'ha seleccionat un algoritme de resolució. Al polsar el optResD, s'habilita el botó de comanda Iniciar.

optResP

Nom del botó d'opció que permet seleccionar una resolució pas a pas. Aquest botó no està visible fins que no s'ha seleccionat un algoritme de resolució. Al polsar el optResD, s'habilita el botó de comanda Iniciar.

cmdIni

Amb aquest nom s'identifica al botó de comandament Iniciar. Aquest botó inicialment no apareix i es fa visible quan es selecciona un algoritme de resolució, però només s'habilita quan s'escull un dels dos botons d'opció que indiquen el tipus de resolució.

El cmdIni serveix per iniciar la seqüència de moviments que ha de fer el robot per resoldre el laberint. Una vegada es polsa, queda inhabilitat fins que no es polsa el botó Parar.

En el cas d'escollir una resolució pas a pas, al polsar el botó cmdIni, aquest queda inhabilitat, però s'habiliten els botons cmdCon i cmdPar. Si el tipus de resolució és la directa, només s'habilita el botó cmdPar.

cmdCon

Nom del botó Continuar el qual només s'habilita al polsar el botó cmdIni si està escollida la resolució pas a pas.

Serveix per continuar els moviments que ha iniciat el robot per resoldre el laberint.

cmdPar

Aquest botó serveix per Parar la resolució del laberint un cop iniciada aquesta, però només podrà polsar-se quan el robot estigui aturat perquè es tracta d'una resolució pas a pas o quan el programa dona l'opció de parar la resolució en cas d'haver escollit una resolució directa.

Una vegada trobat el final del laberint, també és convenient polsar el botó Parar.

fratemps

Aquest marc apareix una vegada finalitzada la resolució del laberint i en el seu interior conté dos etiquetes que indiquen el temps de resolució del laberint.

lblTem

Fa referència a l'etiqueta que indica "Temps de resolució". Està situada dins el marc fratemps i apareix un cop finalitzada la resolució del laberint.

lbltemps

Nom de l'etiqueta que indica el temps transcorregut fins que s'ha iniciat la resolució del laberint, fins que ha finalitzat. El temps es presenta en hores:minuts:segons.

Apareix un cop finalitzada la resolució.

Menú

El formulari CrearLab, conté un menú amb una sèrie d'opcions. Aquest menú s'explica a l'apartat 5.1.2 d'aquest mateix capítol.

5.2.- PSEUDOCODI DE LES RUTINES DEL PROGRAMA

El codi del programa Simulaber, està constituït per nombrosos procediments i funcions. La divisió del codi en aquests procediments i funcions, permet simplificar i reduir el codi del programa ja que algunes d'aquestes rutines s'utilitzen varies vegades en diferents parts del programa.

En el codi, es poden trobar procediments que serveixen per habilitar o inhabilitar els objectes dels formularis o canviar-ne les propietats i altres procediments o funcions dedicats a la resolució dels laberints. La major part de codi l'ocupen aquest procediments i funcions dedicats a la resolució de laberints.

En aquest apartat, es veuran alguns pseudocodis corresponents als procediments i rutines que formen el programa. S'han escollit vuit procediments d'entre els molts que hi ha de tal forma que s'intenta mostrar la varietat d'aquestes rutines.

Algunes de les rutines que s'indiquen a continuació, són comunes a més d'un formulari.

Pseudocodi del procediment corresponent al formulari Menú, per seleccionar un exemple:

```
INICI_SEQÜÈNCIA

    {Mostrar el formulari ExempleLab}
    {Seleccionar en la caixa combinada el text corresponent a l'exemple
    seleccionat en el menú}
    {Mostrar l'exemple seleccionat en la graella del formulari ExempleLab}

FI_SEQÜÈNCIA
```

Aquest procediment, és dels procediments que canvien les propietats dels objectes i els habiliten o inhabiliten segons correspongui. En el codi del programa hi ha varies rutines d'aquest tipus, però són una minoria en comparació amb les rutines per resoldre els laberints i a més, són procediments senzills i sense gaires complicacions.

Els pseudocodis que es mostren a continuació, pertanyen tots a rutines més directament relacionades amb els laberints.

Pseudocodi del procediment corresponent a crear una graella nova per dibuixar el laberint, en el formulari CrearLab:

```
INICI_SEQÜÈNCIA
        SI {la graella ja està visible}
                LLAVORS {preguntar si es vol guardar la graella existent}
                        SI {es vol guardar la graella}
                                LLAVORS {obrir quadre estàndard de guardar}
                                SI_NO {esborrar la graella i obrir-ne una de nova}
                        FI_SI
                SI_NO {obrir la graella}
        FI_SI
FI_SEQÜÈNCIA
```

Pseudocodi del procediment corresponent a escriure i canviar el color de les caselles de la graella seleccionades amb el ratolí:

```
INICI_SEQÜÈNCIA
        SI {es selecciona dibuixar}
           LLAVORS {pintar la casella de blau fosc i escriure: "X"}
            SI_NO
                SI {es selecciona esborrar}
                   LLAVORS {pintar la casella de blau cel i escriure: " "}
                   SI_NO
                        SI {es selecciona inici}
                                LLAVORS {pintar la casella de verd i escriure: "I"}
                                SI_NO {pintar la casella de lila i escriure: "F"}
                        FI_SI
                FI_SI
        FI_SI
FI_SEQÜÈNCIA
```

Pseudocodi del procediment corresponent a escollir un dels exemples:

```
INICI_SEQÜÈNCIA

        SELECCIONAR CAS
        EN CAS QUE {s'escull l'exemple 1} FER
                {exemple 1 visible}
                {exemple 2, 3 i 4, no visibles}
                {habilitar / inhabilitar determinats objectes (segons el formulari)}

        EN CAS QUE {s'escull l'exemple 2} FER
                {exemple 2 visible}
                {exemple 1, 3 i 4, no visibles}
                {habilitar / inhabilitar determinats objectes (segons el formulari)}

        EN CAS QUE {s'escull l'exemple 3} FER
```

{exemple 3 visible}
{exemple 1, 2 i 4, no visibles}
{habilitar / inhabilitar determinats objectes (segons el formulari)}

EN CAS QUE {s'escull l'exemple 4} FER
{exemple 4 visible}
{exemple 1, 2 i 3, no visibles}
{habilitar / inhabilitar determinats objectes (segons el formulari)}

FI_SELECCIÓ

FI_SEQÜÈNCIA

Pseudocodi del procediment corresponent a escollir un dels algoritmes:

INICI_SEQÜÈNCIA
SELECCIONAR CAS
EN CAS QUE {s'escull l'algoritme 1} FER
{algoritme 1 visible}
{algoritme 2, 3 i 4, no visibles}
{habilitar botons d'opció per la resolució}

EN CAS QUE {s'escull l'algoritme 2} FER
{algoritme 2 visible}
{algoritme 1, 3 i 4, no visibles}
{habilitar botons d'opció per la resolució}

EN CAS QUE {s'escull l'algoritme 3} FER
{algoritme 3 visible}
{algoritme 1, 2 i 4, no visibles}
{habilitar botons d'opció per la resolució}

EN CAS QUE {s'escull l'algoritme 4} FER
{algoritme 4 visible}
{algoritme 1, 2 i 3, no visibles}
{habilitar botons d'opció per la resolució}

FI_SELECCIÓ
FI_SEQÜÈNCIA

Pseudocodi del procediment corresponent a escollir un tipus de resolució:

INICI_SEQÜÈNCIA

SI {s'escull la resolució directa}
LLAVORS {inhabilitar els botons de comandament Iniciar i Continuar}; {habilitar el botó Parar}
SI_NO

```
                    SI {s'escull la resolució pas a pas} LLAVORS
                        {inhabilitar el botÓs de comandament Iniciar}; {habilitar el
                        botó Parar i Continuar}
                    FI_SI
        FI_SI
FI_SEQÜÈNCIA
```

Pseudocodi del procediment corresponent a iniciar la resolució d'un laberint:

```
INICI_SEQÜÈNCIA
        {iniciar comptador temps}
        {seleccionar la casella de la graella que correspon al punt d'inici}
        {inicialització de variables}

        SI {l'inici està ben situat}
           LLAVORS
                {comprovar què hi ha a les caselles pròximes a la casella
                seleccionada}
                {assignar un valor a una variable en funció del contingut de les caselles
                pròximes a la casella seleccionada}
                {moviments corresponents a la posició de l'inici i en funció de
                l'algoritme escollit}
                {indicar amb una variable que s'ha iniciat el moviment}
           SI_NO
                {missatge informatiu indicant que es posicioni correctament l'inici}
        FI_SI

        SI {s'ha iniciat el moviment}
                SI {resolució pas a pas}
                   LLAVORS {final rutina}
                   SI_NO
                        MENTRE {no trobi final ni es pari la resolució} FER
                                {comprovar què hi ha a les caselles pròximes a la casella
                                seleccionada}
                                {assignar un valor a una variable en funció del contingut
                                de les caselles pròximes a la casella seleccionada}
                                {moviments corresponents a la posició actual i en funció
                                de l'algoritme escollit}
                                {comprovar si és final de laberint}
                        FI_MENTRE
                FI_SI
        FI_SI

        SI {es troba el final del laberint}
                LLAVORS {missatge informatiu}; {indicar temps de resolució}
        FI_SI

FI_SEQÜÈNCIA
```

Pseudocodi del procediment corresponent a continuar la resolució d'un laberint:

INICI_SEQÜÈNCIA

MENTRE {no sigui final de laberint ni hi hagi intersecció de més de 2 camins} FER

{comprovar què hi ha a les caselles pròximes a la casella seleccionada}
{assignar un valor a una variable en funció del contingut de les caselles pròximes a la casella seleccionada}
{moviments corresponents a la posició actual i en funció de l'algoritme escollit}
{comprovar si hi ha final del laberint}

FI_MENTRE

Aquests són alguns dels exemples de rutines que formen el codi. Com ja s'ha comentat, la majoria de rutines que hi ha serveixen per resoldre els laberints i estan relacionades amb els algoritmes explicats en l'apartat 4.

5.3.- INSTAL·LACIÓ DEL PROGRAMA

Abans d'instal·lar el programa, cal comprovar que l'equip compleixi els requisits mínims.

5.3.1.- Requisits del sistema

Per instal·lar el Simulador es requereix el següent hardware i software:

- Microprocessador Pentium 90 MHz o superior.
- Pantalla VGA de 640x480 o de resolució superior compatible amb Microsoft Windows.
- 24 MB de RAM per Windows 95/98/2000/XP, 32 MB per Windows NT.
- Microsoft Windows NT 3.51 o posterior, o Microsoft Windows 95 o posterior.
- Una unitat CD-ROM.
- Un *mouse* (ratolí) o un altre dispositiu punter.

5.3.2.- Instal·lació del programa

Per poder instal·lar el programa Simulador, cal seguir els passos següents:

- Executar l'arxiu Setup.exe de la carpeta VisualBasic/Simulador/Executable
- Apretar el botó acceptar que apareix al quadre benvinguda al programa d'instal·lació.
- Escollir el directori on es vol guardar el programa i polsar el botó que inicia la instal·lació.
- Polsar el botó continuar que apareix en el següent quadre de missatge.
- Polsar finalitzar.

Una vegada realitzats aquests passos, el programa ja estarà disponible per ser utilitzat

5.4.- REGLES DEL PROGRAMA

Hi ha un seguit de regles i consells a tenir en compte per tal d'utilitzar el programa correctament. Són les següents:

Regles referents a la creació o modificació d'un laberint:

- La distància entre dues parets que formen un camí, ha de ser exactament de tres quadrets en tots els camins del laberint.
- L'amplada de les parets serà com a mínim d'un quadret.
- Cal deixar com a mínim tres quadrets entre els límits de la quadrícula i els límits del laberint.
- L'inici i el final del laberint no poden estar fora del laberint ni en mig d'interseccions. S'han de situar sempre en el quadret del mig entre dues parets verticals o horitzontals.
- Si es fa alguna modificació del punt d'inici, cal esborrar el punt original. Només hi pot haver un únic punt d'inici al laberint, però hi poden haver varis punts finals. Si hi hagués més d'un punt d'inici en el laberint, es consideraria com a vàlid l'últim punt d'inici dibuixat.
- Al dibuixar girs o interseccions amb les parets del laberint, cal que la llargada de les parets sigui d'un mínim de tres quadrets abans de tornar a fer un altre gir o trobar-se una altra intersecció.
- Els girs han de formar sempre angles rectes. La graella permet dibuixar altres tipus d'angles o inclòs simular corbes, però els algoritmes del programa no permeten la resolució.

Regles i consells referents a la resolució del laberint:

- Al provar de resoldre un laberint creat per primera vegada, és aconsellable fer-ho amb la resolució pas a pas per tal de localitzar els possibles errors alhora de fer el dibuix del laberint.
- Mentre s'està executant la resolució directa, no polsar el botó parar fins que el programa no doni la possibilitat d'escollir si es vol parar la resolució o no.

Altres regles i consells:

- Per tancar el programa, cal fer-ho des de l'opció sortir del menú arxiu del formulari actual. Cal evitar tancar el programa de qualsevol altre forma.
- És aconsellable guardar els laberint creats o les modificacions fetes en un laberint ja existent abans de començar la resolució.
- Al guardar un laberint, es guarda amb format .GRD que prové de *Grid* (graella). L'arxiu guardat únicament podrà obrir-se per ser aplicat a un control MSFlexGrid de Visual Basic 6.0. Si es vol guardar el dibuix del laberint en un altre format, pot fer-se polsar la tecla Imp. Pant del teclat i copiar la imatge en el programa Paint o similar.
- El programa només permet obrir arxius amb extensió *.GRD. És aconsellable evitar obrir arxius ja existents degut a la lentitud amb la que el programa realitza aquesta acció.

5.5.- FUNCIONAMENT DEL SIMULABER

El programa Simulador de resolució de laberints està pensat perquè qualsevol persona pugui utilitzar-lo i no requereix cap coneixement específic de cap matèria. El fet d'habilitar i inhabilitar les funcions facilita la utilització del programa ja que marca el passos a seguir per fer una correcte utilització del programa. Es doncs, un programa fàcil d'utilitzar.

Tot i l'escassa dificultat d'utilització del programa, és aconsellable que en primer lloc es vegi la resolució d'algun exemple ja fet abans de començar a dibuixar cap laberint.

És molt important tenir present en tot moment les normes i regles d'utilització del programa.

5.5.1.- Com veure la resolució d'un exemple

Per poder veure la resolució d'un exemple cal accedir al formulari ExempleLab. A aquest formulari s'hi accedeix des del menú Exemples del formulari principal. El menú Exemples del formulari Menú conté quatre opcions, cada una de les quals fa referència a un laberint d'exemple diferent. Al seleccionar qualsevol d'aquests quatre exemples, es visualitza per pantalla el formulari ExempleLab, el qual conté el laberint seleccionat al menú Exemples del formulari Menú. La figura 5.37 mostra l'aspecte del formulari ExempleLab amb el laberint d'exemple 1.

Per **seleccionar un dels exemples** des del menú del formulari Menú, pot fer-se clicant al damunt amb el ratolí, o bé per teclat, polsant la tecla Alt més la lletra subratllada del menú, que en el cas del menú dels Exemples és la lletra e (Alt + e).

Tal com es pot observar a la figura 5.37, el formulari ExempleLab conté un menú, el Picture Box (PicLab2) que conté un MSFlexGrid (GridExe1) amb el dibuix del laberint, un altre MSFlexGrid (GridAlg) on més tard es mostrarà el pseudocodi de l'algoritme escollit, i dos Combo Box (caixa combinada), un que permetrà escollir un dels quatre laberints d'exemple que hi ha (cboExe) i l'altre per escollir un dels també quatre algoritmes disponibles (cboAlg).

La caixa combinada cboExe permet canviar el laberint d'exemple i escollir qualsevol dels quatre que hi ha disponibles. Cada exemple seleccionat amb el cboExe està associat a una graella, la qual es mostra sobre el PicLab2.

Es veu doncs que per seleccionar un dels laberints d'exemple, es pot fer a partir del formulari principal o bé un cop dins el formulari ExempleLab, seleccionant l'exemple en el cboExe.

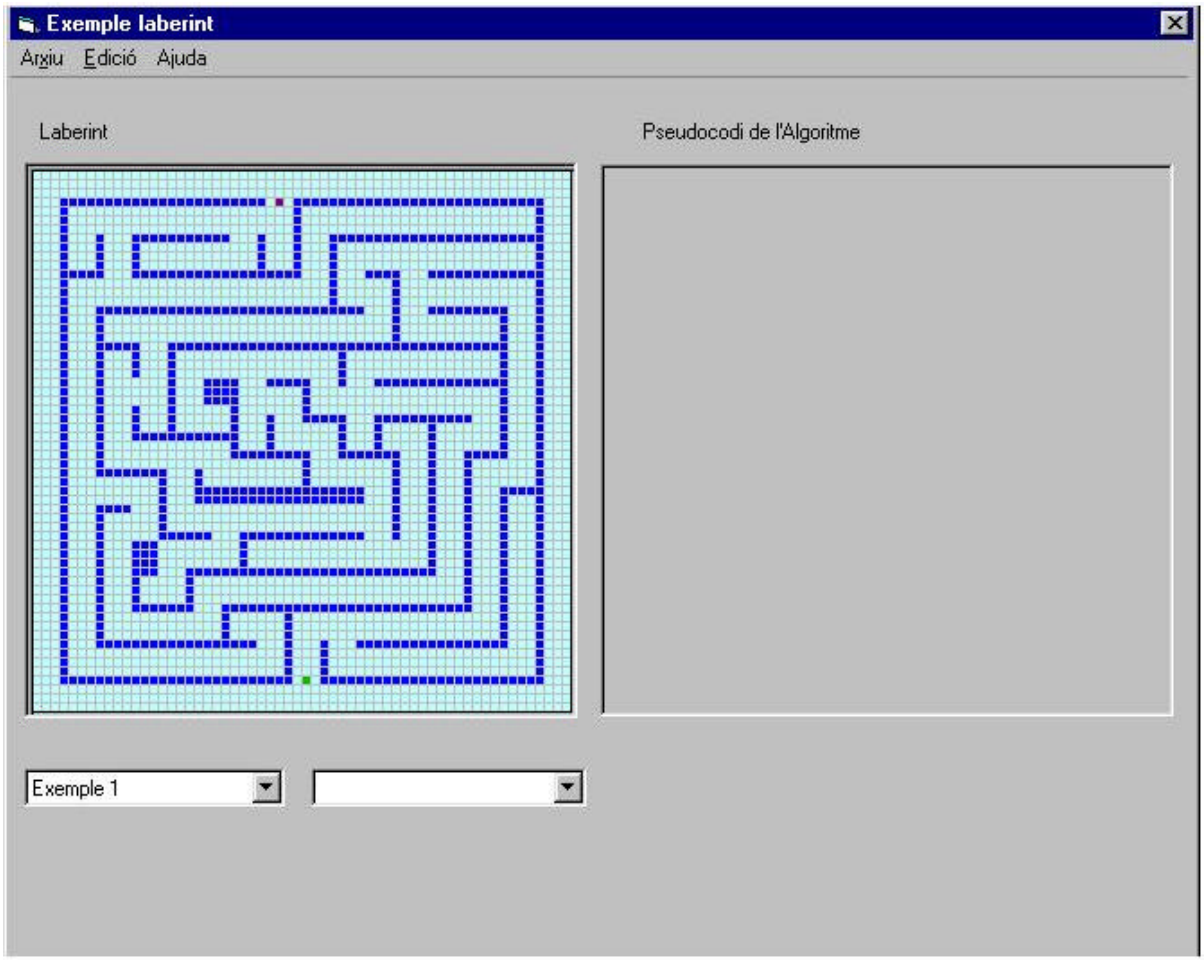


Fig.5.37. Exemple 1 del formulari ExempleLab

Els laberints d'exemple dels que disposa el programa Simulador tenen assignats un punt d'inici i un punt final. Aquests dos punts es poden canviar de posició utilitzant les opcions del menú Edició. També és possible dibuixar parets i esborrar qualsevol dels punts del laberint, ja siguin el d'inici, el de final o una paret.

Una vegada es té el laberint d'exemple triat i amb les **modificacions** que s'hagin volgut fer, el pas següent abans de fer la resolució del laberint, és **escollir l'algoritme de resolució**. Per escollir aquest algoritme cal seleccionar-lo des del cboAlg. Quan es selecciona algun algoritme, apareix el seu pseudocodi a la graella GridAlg. Amb les barres de desplaçament que apareixen automàticament, es pot recorre tot el pseudocodi. A la figura 5.36 es mostra com quedarà el formulari una vegada seleccionat el laberint i l'algoritme.

Es pot comprovar també que al seleccionar un algoritme de resolució, apareix un quadre amb dos botons d'opció, el optResD2 i el optResP2, juntament amb tres command buttons inhabilitats, cmdIni2, cmdCon2 i cmdPar2.

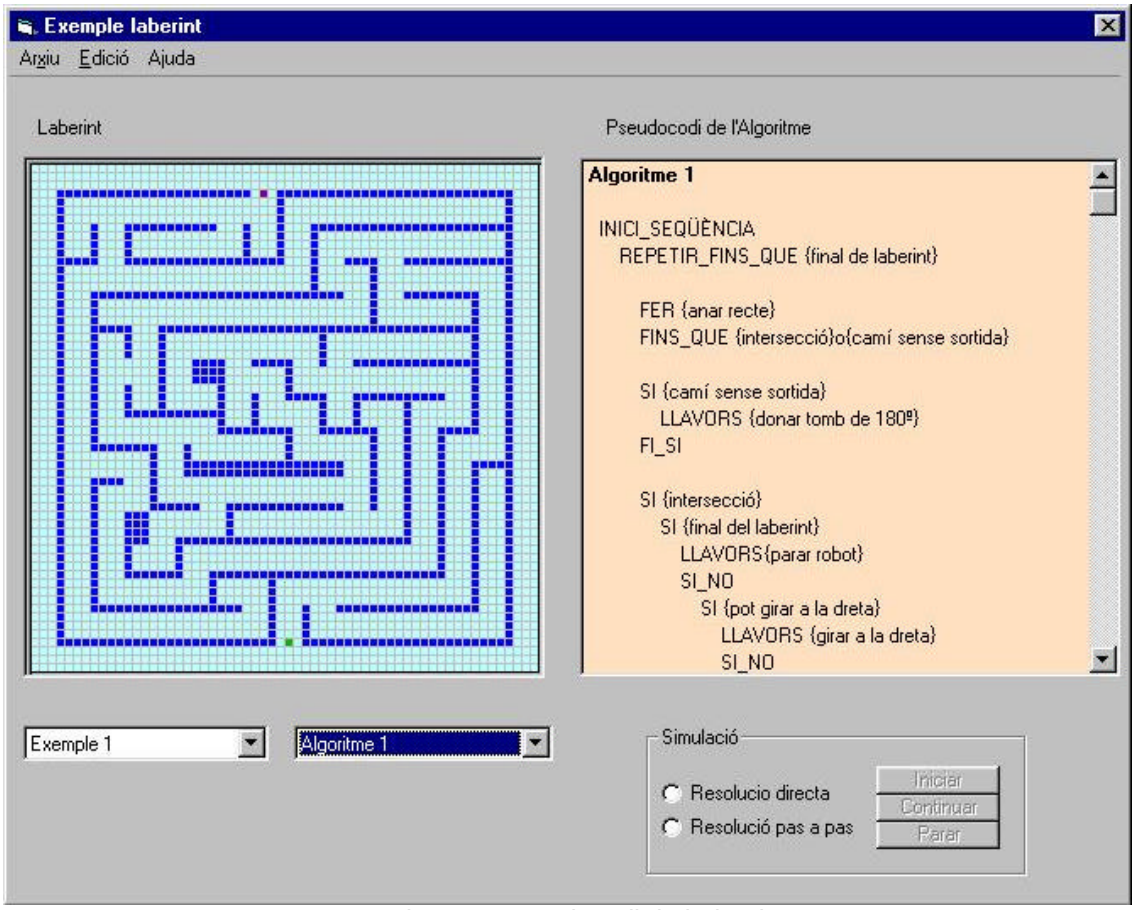


Fig.5.38. Pseudocodi de l'Algoritme 1

El pas següent és **seleccionar un tipus de resolució**. Es pot fer una resolució directa o una resolució pas a pas. En el moment de seleccionar un tipus de resolució, s'habilita el botó Iniciar (cmdIni2) que permetrà iniciar els moviments del robot.

Les opcions del menú Edició, queden inhabilitades mentre es resolt el laberint i es tornen a habilitar quan es polsa el botó parar un cop finalitzada la resolució.

Resolució directa:

Si es selecciona una resolució directa, en el moment de **polsar el botó iniciar**, el robot comença una seqüència de moviments que el conduiran, en cas que l'algoritme sigui l'adequat, al final del laberint.

Abans de començar els moviment però, el programa comprova que el punt d'inici estigui degudament situat, i en cas que no ho estigui apareix un missatge d'error com el de la figura 5.39.

Fig. 5.39. Missatge d'error causat per tenir el punt d'inici mal situat

Una vegada comprovat que el punt d'inici està en un lloc adequat, comença la seqüència de moviments. Si s'ha escollit la resolució directa, l'usuari no pot aturar el robot fins que el programa no li permeti encara que polsi el botó Parar (cmdPar2).

Els algoritmes de resolució estan programats perquè quan el robot es trobi davant una determinada intersecció, s'aturi i es mostri una caixa de diàleg preguntant a l'usuari si desitja continuar amb la resolució o no. Això permet parar la resolució directa ja que sinó no podria parar-se fins que el robot arribés al final del laberint.

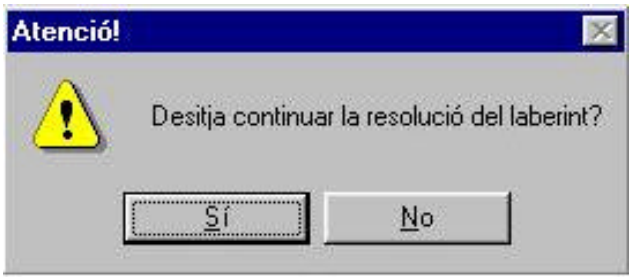


Fig. 5.40. Missatge per continuar la resolució directa o parar-la

El botó Parar, no actua mentre el robot està movent-se, per tant, no servirà per parar la resolució directa una vegada iniciada la seqüència de moviments.

Si quan apareix el missatge que mostra la figura 5.40, s'escull que no es vol seguir amb la resolució, llavors desapareix el missatge i n'apareix un altre que informa de que ha finalitzat la resolució del laberint. Al mateix temps, apareix en el formulari, el temps que ha transcorregut des del moment en que s'ha iniciat la resolució del laberint fins que ha finalitzat.

Fig. 5.41. Missatge que indica que ha finalitzat la resolució del laberint

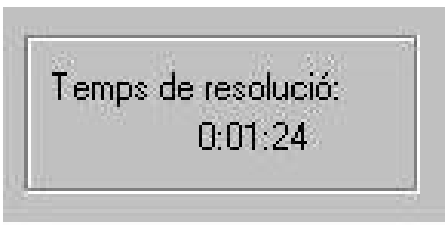


Fig. 5.42. Indicador del temps de resolució

Per tancar la caixa de missatge de la figura 5.41, cal clicar amb el ratolí sobre el botó acceptar o bé fer un enter amb el teclat. Per parar definitivament la resolució, cal polsar el botó **Parar** (cmdPar2).

Si quan apareix la caixa de missatge de la figura 5.40 es decideix continuar la resolució, el robot seguirà desplaçant-se pel laberint fins que torni a aparèixer una altra caixa de missatge com la de la figura 5.38 o bé el robot hagi trobat el final del laberint. En aquest últim cas, apareixerà el missatge de la figura 5.41 i en el formulari s'indicarà el temps de resolució tal i com mostra la figura 5.42.

Resolució pas a pas:

En el cas d'escollir una resolució pas a pas, també caldrà polsar el botó **Iniciar** (cmdIni2) per iniciar la resolució del laberint. Abans d'iniciar la resolució, es comprova també que el punt d'inici estigui situat correctament, igual que es feia en la resolució directa. Si l'inici està ben situat comença la resolució, però en aquest cas, el robot farà un moviment i s'aturarà.

Al tractar-se d'una resolució pas a pas el robot s'aturarà una vegada iniciada la resolució i més tard, davant cada intersecció de tres o quatre camins.

Al contrari del que passava en la resolució directa, ara, al polsar el botó Iniciar, s'habiliten dos botons més: el de Continuar (cmdCon2) i el de Parar (cmdPar2).

El botó **Parar** actuarà igual que en la resolució directa, i només podrà ser polsat quan el robot no estigui executant cap seqüència de moviments.

El botó **Continuar** serà necessari polsar-lo cada vegada que el robot s'aturi en una intersecció si es vol continuar la resolució.

Una característica d'aquesta resolució, és que cada vegada que el robot fa un moviment, queda senyalat en el pseudocodi de l'algoritme l'estructura de l'algoritme corresponent a la situació concreta en la que es troba el robot. Per entendre-ho millor, es suposa que el robot s'ha introduït en un camí sense sortida. En aquest cas, i si el robot està actuant segons l'algoritme de resolució 1, el pseudocodi quedaria de la següent forma:

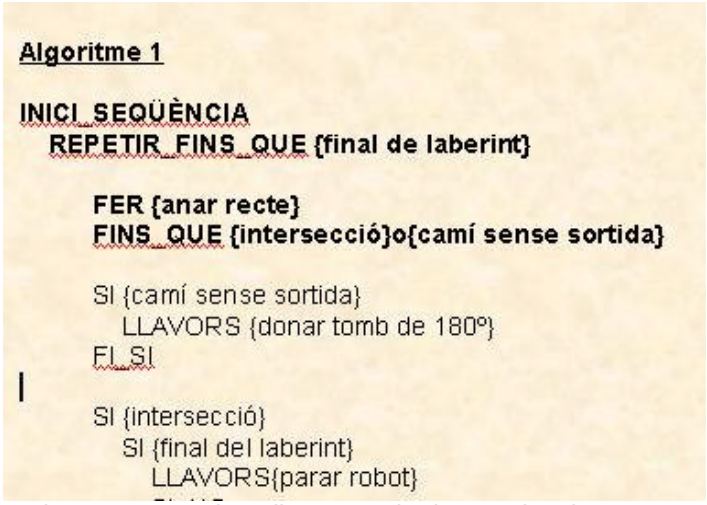


Fig. 5.43. Pseudocodi 1 quan el robot es desplaça recte

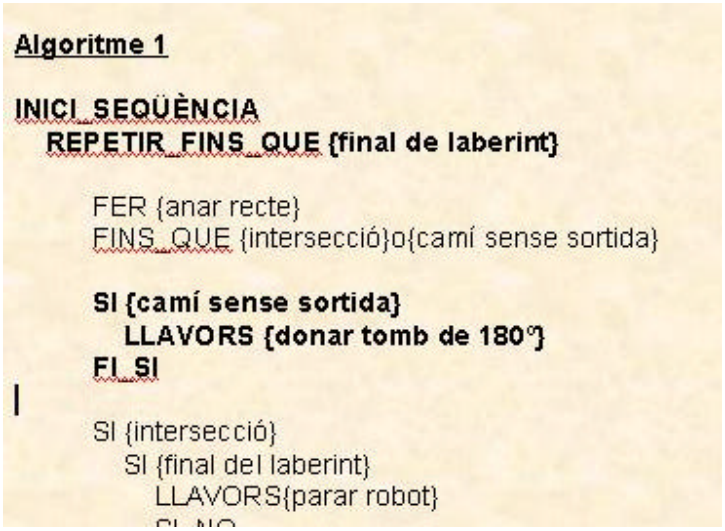


Fig. 5.44. Pseudocodi 1 quan el robot es troba en un camí sense sortida

En aquest mode de resolució, no apareixeran caixes de missatge preguntant si es vol continuar amb la resolució o parar-la, ja que aquesta resolució ja es para en cada intersecció per veure com està actuant el robot.

La resolució pas a pas, també finalitza quan el robot troba el final del laberint, o bé quan l'usuari polsa el botó Parar. En aquest cas també surt el missatge de la figura 5.41 anunciant que ha finalitzat la resolució, però no s'indica el temps tardat ja que dependrà del temps que tarda l'usuari en polsar el botó continuar cada cop que el robot es pari en una intersecció.

Quina resolució utilitzar?

La resolució directa, permet resoldre els laberints més ràpidament ja que l'usuari no ha d'estar constantment polsant el botó Continuar i un cop finalitzada la resolució del laberint, s'indica el temps tardat.

Per altra banda, la resolució pas a pas permet conèixer com està actuant el robot en cada moment i permet veure el pseudocodi de l'algoritme mentre s'està realitzant la resolució. Una altra característica de la resolució pas a pas, és que en cas que l'algoritme utilitzat no sigui l'adequat per resoldre el laberint o bé el dibuix del laberint no compleixi les normes del programa, és possible detectar-ho abans de que es produeixi un error en el programa.

En funció de les necessitats de l'usuari, serà més convenient utilitzar una resolució o una altra. Així, s'aconsella utilitzar la resolució pas a pas en aquells casos en els que es vulgui anar comprovant el pseudocodi de l'algoritme utilitzat i en aquells en els que es faci la resolució d'un laberint per primera vegada i es desconegui si l'algoritme utilitzat permet resoldre el laberint o no.

La resolució directa serà més útil quan es vulgui comparar la resolució que proporcionen diferents algoritmes per un mateix laberint coneixent prèviament que els dos algoritmes permeten la resolució.

5.5.2.- Com crear laberints i resoldre'ls

Per poder crear un laberint i resoldre'l, és necessari accedir al formulari CrearLab. A aquest formulari s'hi accedeix des del menú Crear Lab del formulari principal (Menú). El menú Crear Lab del formulari Menú, conté dues opcions, una anomenada Nou i una altra anomenada Modificar.

Al seleccionar una de dels dues opcions del menú Crear Lab, s'accedeix al formulari CrearLab. La diferència entre l'opció Nou i la Modificar, està en que, quan es selecciona Nou, al mostrar-se el formulari CrearLab, apareix una caixa de missatge informant a l'usuari de què ha de fer per crear un laberint nou. Si l'opció seleccionada és la de Modificar, el missatge informatiu que apareix al mostrar-se el formulari CrearLab, indica els passos a seguir per tal d'obrir un laberint creat i guardat amb anterioritat.

Aquests missatges informatius, són els que mostren les figures 5.5 i 5.6 d'aquest mateix capítol.

L'aparença del formulari CrearLab quan es selecciona l'opció Nou en el menú del formulari principal, és la següent:

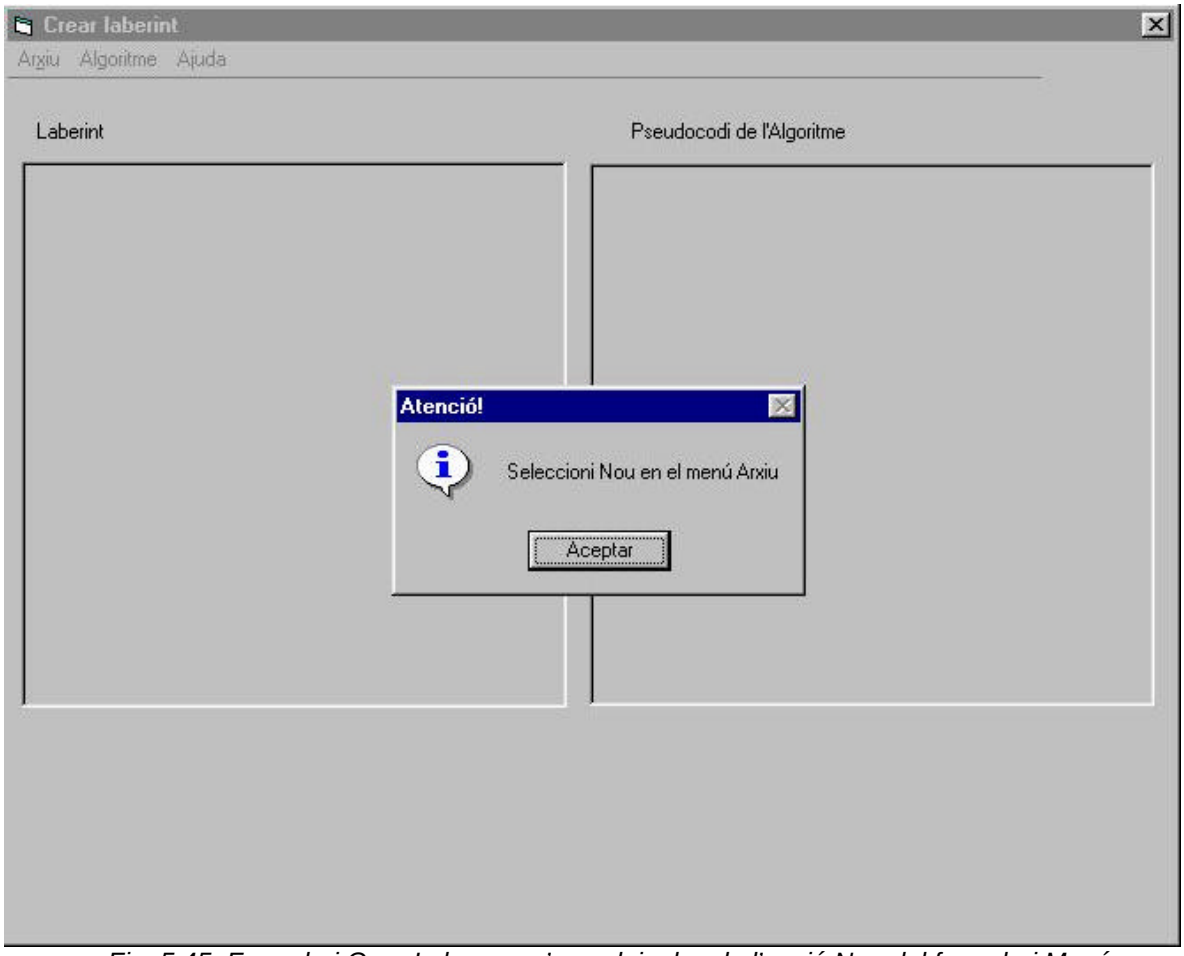


Fig. 5.45. Formulari CrearLab quan s'accedeix des de l'opció Nou del formulari Menú

Per **seleccionar Nou en el menú del formulari Menú**, pot fer-se clicant al damunt amb el ratolí o bé per teclat, polsant la tecla Alt més la lletra subratllada del menú, que en el cas del menú Crear Lab, és la C (Alt + c).

Una vegada seleccionada l'opció Nou o bé l'opció Modificar en el formulari Menú, s'ha de polsar acceptar en el formulari CrearLab per tancar la caixa de missatge. A continuació, s'ha d'accedir al menú Arxiu (amb el ratolí o polsant Alt+x) i seleccionar per exemple Nou.

Inicialment, el menú Arxiu només té habilitades les opcions de Nou, Obrir i Sortir. Les opcions de Tancar, Seleccionar i Guardar, estan inhabilitades ja que no es poden dur a terme si no hi ha algun laberint existent (Fig. 5.13) o la graella per dibuixar-lo.

Al **seleccionar l'opció Nou**, es crea una graella (Grid1) on es dibuixarà el laberint. Al mateix temps, apareixen situats sota la graella quatre botons d'opció que permetran **dibuixar parets, esborrar, posicionar l'inici i el final del laberint**.

També apareixen dos Command Button (cmdSel i cmdGua) que permetran seleccionar el laberint, per després guardar-lo. No es possible guardar el laberint si prèviament no s'ha seleccionat.

El formulari quedarà de la següent forma:

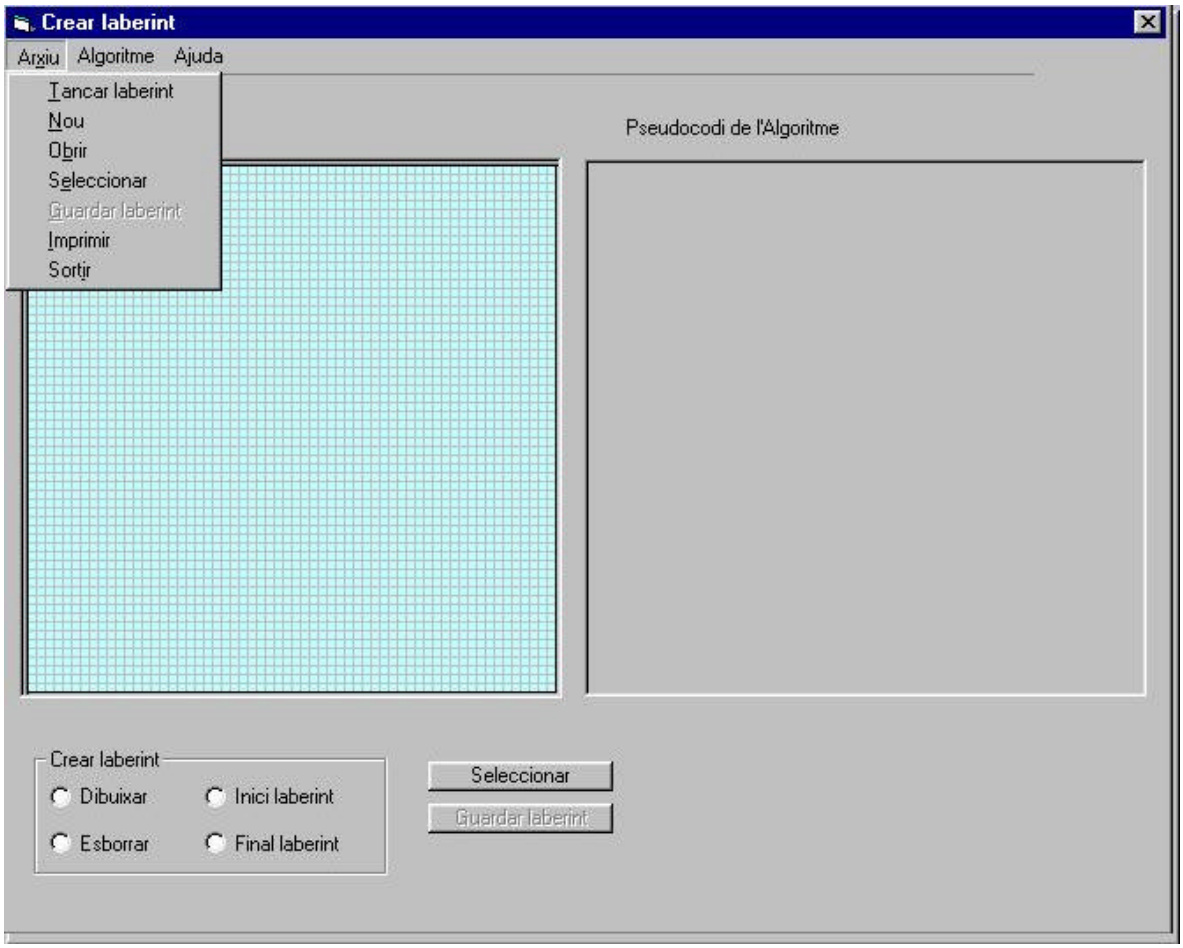


Fig. 5.46. Formulari CrearLab quan es crea un laberint Nou

Si es selecciona l'opció Dibuixar, al clicar amb el ratolí sobre la graella, es marca un punt de color blau que representa una paret. Per no haver d'anar clicant punt per punt, si es manté el botó esquerra del ratolí polsat mentre es mou el punter del ratolí sobre la graella, es van pintant de blau tots els quadrets seleccionats. Això permet dibuixar més ràpidament les parets dels laberints, ja que seria molt laboriós fer el dibuix punt per punt. Amb l'opció Esborrar, passa el mateix. Les opcions Inici laberint i Final laberint, en canvi, només queda marcat sobre la graella el quadret senyalat pel punter del ratolí en el moment de clicar ja que s'entén que només interessarà marcar un únic punt com a inici i un altre com a final.

Al contrari del que passa al formulari CrearLab, en el formulari ExempleLab, les opcions de Dibuixar i Esborrar no permeten seleccionar varis quadrets a la vegada i únicament queda dibuixat el quadret senyalat pel punter del ratolí en el moment de clicar. El programa està configurat d'aquesta manera perquè es suposa que en el cas dels laberints d'exemple, únicament es faran petites modificacions i per tant, no caldrà fer grans seleccions de quadrets.

Si es té un laberint dibuixat i per algun motiu es desitja obrir un altre arxiu Nou, al tornar a seleccionar l'opció Nou del menú Arxiu, el programa pregunta si es

vol guardar el laberint existent. Si es vol guardar, s'obre la caixa de diàleg estàndard de guardar, i si no es vol guardar, es crea una graella nova i es perd el dibuix que hi havia a la graella anterior. També es pot tancar el laberint inicial amb l'opció tancar del menú arxiu i després si es vol, crear una graella nova o obrir un arxiu ja existent.

Un cop finalitzat el dibuix del laberint, s'ha de **seleccionar el laberint** o bé amb el botó cmdSel o bé des del menú Arxiu. Un cop seleccionat, ja es pot guardar.

Al **guardar** el laberint, apareix el quadre de diàleg estàndard de guardar, comú a la majoria de programes de Windows. El laberint podrà guardar-se a qualsevol lloc, però sempre **en format *.GRD** per tal de poder recuperar-lo en un altre moment.

No és possible començar la resolució del laberint si no està guardat.

Una vegada guardat el laberint, s'habiliten les opcions del menú Algoritme que fins ara havien estat inhabilitades. El pas següent és **escollir un dels quatre algoritmes** per poder resoldre el laberint.

Quan s'escull un algoritme, apareix el pseudocodi de l'algoritme escollit en la graella GridAlg i també apareixen els botons d'opció per seleccionar el tipus de resolució i els Command Buttons per iniciar, continuar o parar la resolució.

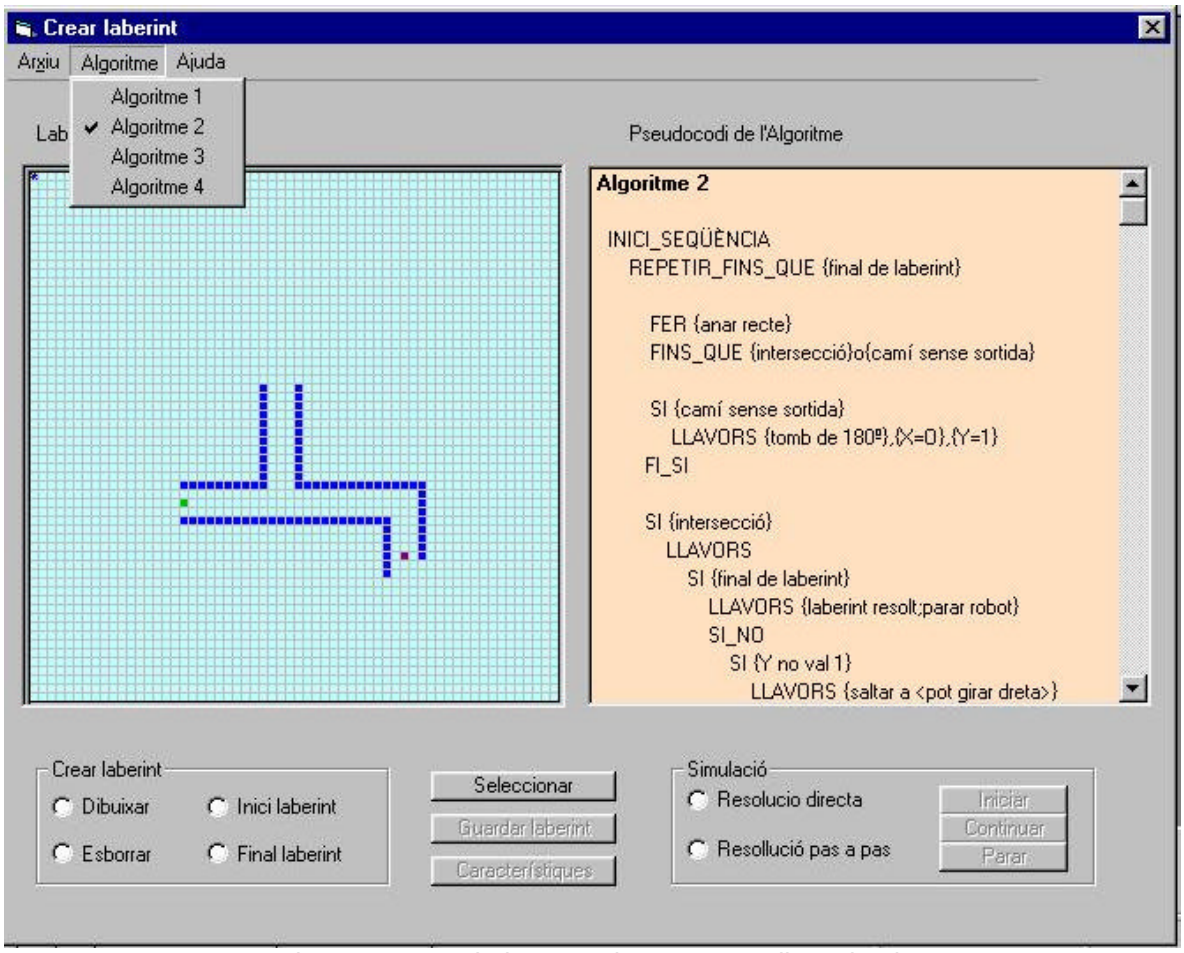


Fig. 5.47. Formulari CrearLab quan s'escull un algoritme

A continuació, caldrà **seleccionar un tipus de resolució** i **polsar el botó Iniciar** per començar la resolució. S'aconsella escollir la resolució pas a pas si és la primera vegada que s'intenta resoldre el laberint creat.

El formulari quedarà modificat de la següent manera:

Fig. 5.48. Formulari CrearLab quan s'inicia la resolució pas a pas.

Es comprova que una vegada iniciada la resolució, no és possible fer cap modificació del laberint. El que si és possible, abans de polsar el botó Iniciar, és regular la velocitat amb la que es vol resoldre el laberint. Per defecte, està seleccionada la màxima velocitat de resolució.

La resolució del laberint, funciona igual que en el cas dels laberints d'exemple, tant si es fa una resolució directa com una resolució pas a pas.

Si al entrar al formulari CrearLab, no es vol dibuixar un laberint sinó recuperar-ne un de ja creat, només cal seleccionar l'opció Obrir en el menú Arxiu. S'obre un quadre de diàleg estàndard que permetrà seleccionar l'arxiu que es vulgui. Només es poden obrir arxius amb extensió *.GRD.

Els passos següents per arribar a resoldre el laberint, seran els mateixos que si es crea el laberint nou.

5.5.3.- Utilització de l'ajuda

L'ajuda de la qual disposa el programa és similar tan a nivell d'estructura com de funcionament, a les ajudes que tenen la majoria dels programes de Windows.

El menú ajuda del programa Simulaber està constituïda per les opcions: Contingut, Índex, Ajuda i Sobre el Simulaber. Cada una d'aquestes opcions accedeix a una carpeta de l'arxiu d'ajuda, excepte la opció Sobre el Simulador la qual obre un formulari amb informació general sobre el programa.

Si es vol veure les opcions de les quals disposa l'arxiu ajuda, cal escollir l'opció Contingut. Per fer una cerca indexada a partir d'algun nom, es fa des de l'opció Índex. Per últim, si s'escull l'opció Ajuda s'obre el quadre de diàleg comú d'ajuda de Windows on també pot fer-se una cerca indexada o bé per contingut.

6.- PRESTACIONS DEL PROGRAMA

El programa Simulaber ofereix a l'usuari que l'utilitza, la possibilitat de veure com actuaria un robot per resoldre un laberint.

El programa disposa de quatre laberints que serveixen d'exemples per veure diferents tipus de resolució. Hi ha quatres algoritmes diferents per resoldre els laberints, i cada un està pensat per ser aplicat a un dels quatre laberints d'exemples que hi ha. Tot i així, es pot provar de resoldre cada laberint amb els quatre algoritmes. En alguns casos, l'algoritme escollit podrà resoldre el laberint i en altres casos, no. Això permetrà a l'usuari conèixer quin algoritme és més adequat per resoldre un tipus de laberint.

Una altre característica del programa és que permet que l'usuari es creï els seus laberints i els pugui resoldre amb un dels quatre algoritmes. L'usuari també pot modificar els laberints d'exemple.

Tots els laberints, inclòs quan estan resolts, poden guardar-se i recuperar-se en qualsevol moment.

En el moment de resoldre els laberints, pot fer-se de dues maneres: de forma directa, veient com el programa fa tota la resolució des de que el robot inicia els moviments fins que troba el final; i una resolució pas a pas, la qual permet anar llegint el pseudocodi de l'algoritme utilitzat i veure quines decisions està prenent el robot en cada moment.

Aquest programa serà útil en aquells casos en els que es vulgui construir un robot per resoldre laberints o que explori zones desconegudes, ja que permetrà escollir l'algoritme de control més adequat que hauria de dur el robot per desenvolupar les seves funcions correctament.

7.- HARDWARE QUE CALDRIA PER IMPLEMENTAR EL ROBOT

En el capítol 3.2 quan es parlava de les possibles solucions, ja es comentava breument com hauria de ser un robot per poder desplaçar-se i resoldre un laberint. En aquest capítol s'ampliaran una mica més aquests conceptes i es farà una proposta de disseny d'un robot adequat pel tipus de laberint que s'ha estat tractant durant tot el projecte: un laberint format per camins delimitats per parets.

Per dissenyar un robot, en primer lloc cal diferenciar dues parts principals: la part de hardware i la part de software.

La part de hardware fa referència a l'estructura del robot, els mecanismes i els elements que l'integren, i la part de software inclou tot el que implica control del robot, és a dir, els algoritmes que durà el programa que farà funcionar el robot. A continuació s'expliquen per separat aquestes dues parts.

7.1.- HARDWARE

En el moment de dissenyar el hardware s'haurà de tenir present les característiques que tindrà el laberint ja que delimitaran certs aspectes en el disseny del robot. Per exemple, s'haurà de tenir en compte la distància entre paret i paret del laberint, ja que el robot no haurà de superar aquestes dimensions. De no ser així, el robot tindria dificultats per desplaçar-se per l'interior del laberint. També caldrà tenir present el material amb el que està fet el terra del laberint ja que segons si aquest és rugós o lliscant, el mecanisme de contacte del robot amb el terra, que pot ser mitjançant rodes, haurà d'estar fet amb un material adequat per assegurar una correcta adhesió.

Un altre aspecte important, és que el robot ha de ser completament autònom, per tant, no durà cap mena de control remot, ni cable que el connecti enlloc. Això significa que el robot haurà de subministrar-se la seva pròpia energia mitjançant bateries. El problema que tenen les bateries, és que acostumen a ser bastant pesades i a més a més, la seva duració és limitada, per tant s'hauria d'evitar que el robot es quedi parat al mig d'un laberint per falta d'energia.

Fins ara, ja sabem que el robot haurà d'anar equipat amb bateries, que el material de les rodes haurà d'anar en funció al material del terra i que les dimensions del robot hauran de ser suficientment reduïdes perquè aquest pugui moure's per l'interior del laberint. Aquests aspectes ja donen una visió general de com hauria de ser el robot.

A continuació s'analitzen més detingudament cada una de les parts fonamentals que constituiran el robot.

7.1.1.- Diagrama de blocs del disseny escollit

Es pot dividir l'estructura del robot en quatre blocs bàsics diferents. A la figura següent s'observen de forma esquemàtica quins serien aquests blocs:

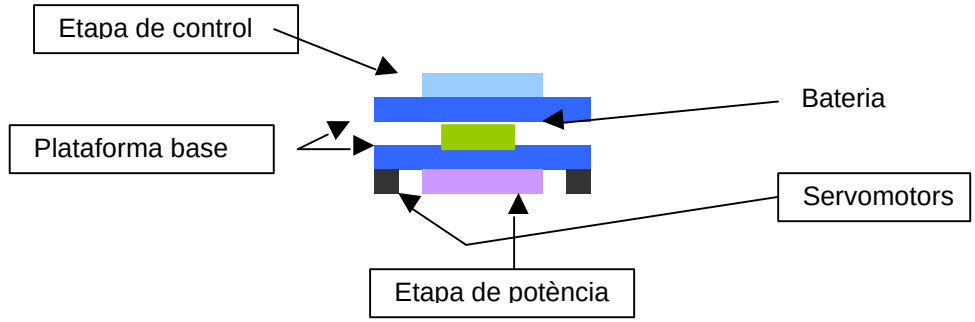


Fig. 7.1. Diagrama de blocs del robot

Es pot veure que el robot està compost per quatre blocs principals: plataforma base, etapa de control, etapa de potència i servomotors. En primer lloc hi hauria la plataforma base que en el cas de la figura, està dividida en dues parts. La part inferior aguantaria els servomotors, les rodes i el circuit de potencia, i la part superior duria els sensors i el circuit de control. Entre les dues parts de la plataforma, podria posar-s'hi la bateria. En l'etapa de control, s'analitzarien les senyals rebudes pels sensors i un microcontrolador donaria les ordres oportunes que controlarien els servomotors, passant primer per l'etapa de potència. En aquesta etapa, s'adequaria la senyal procedent del microcontrolador per tal de poder aplicar-la sobre els servomotors. Per últim, hi ha els servomotors que seran els encarregats de moure el robot de la manera desitjada i segons els hi indiqui el microcontrolador.

Aquest però, és un exemple de com podria ser l'estructura del robot ja que, mentre el robot estigués constituït per aquests blocs principals, podria estructurar-se de moltes maneres diferents. Així doncs, podria construir-se un robot totalment diferent al proposat en el diagrama de blocs i que realitzés les mateixes funcions. Podrien haver-hi canvis pel que fa al tipus de microcontrolador utilitzat, a la quantitat de sensors, el tipus de motors utilitzats (servomotors, motors pas a pas), la plataforma base podria ser d'una sola peça en comptes d'estar dividida en dues, la bateria podria estar situada en un altre lloc,...

En el següent apartat, s'analitzaran amb més detall cada un dels components que formarien el robot

7.1.2.- Components a utilitzar

El disseny d'un robot no és únic i es podrien trobar moltes solucions diferents per una mateixa aplicació, cada una d'elles amb els seus avantatges i els seus inconvenients.

El disseny que aquí es proposa és un disseny bàsic i senzill d'un robot que estaria constituït per tres sensors detectors de xoc, tres sensors d'ultrason, un microcontrolador que comprovaria periòdicament l'estat d'aquests sensors, una bateria que alimentaria tots els components que ho requerissin, dos servomotors que s'activarien en funció de les ordres que els hi enviés el microcontrolador, i tres rodes, dues d'elles accionades cada una amb un servomotor i la tercera, una rodeta lliure o "boja". També hi hauria una placa base que suportaria tots els elements i els circuits de potencia per regular la senyal procedent del microcontrolador abans de ser aplicada als servos. A continuació es descriuen els principals components d'aquest muntatge:

?? Plataforma base

La plataforma serà l'element que subjectarà tots els components que constitueixen l'estructura del robot. Aquesta plataforma estarà dividida en dues parts. A la part inferior s'hi situaran els dos motors que controlen el moviment de les rodes. S'hi enganxaran també l'eix de les dues rodes motrius i una tercera rodeta que es mourà lliurement. Entre la plataforma inferior i la superior s'hi situarà la bateria. A la part superior de la plataforma hi haurà els sensors, el pic, i els circuits de control i de potència. El fet de que la plataforma estigui dividida en dues parts, una al damunt de l'altre, permet reduir l'amplada i llargada del robot a canvi d'augmentar la seva altura. Això és interessant ja que les dimensions del robot es veuen delimitades per les dimensions del laberint i quan més petit sigui el robot, menys dificultats tindrà per avançar i girar per l'interior del laberint.

?? Dos servomotors.

Cada servomotor servirà per controlar els moviments d'una roda motriu. Els servomotors, hauran de poder girar en les dues direccions i uns angles d'almenys 180 º. Aquests servomotors, hauran de poder-se controlar individualment, ja que en el cas de fer girar el robot, els dos servomotors s'accionaran en sentit contrari per tal de fer girar una roda motriu endavant i una altra endarrera. Això permetrà disminuir l'àrea de gir necessària pel robot. Quan els dos motors girin en la mateixa direcció amb la intenció de desplaçar el robot endavant o endarrera, hauran d'anar ben sincronitzats perquè si no hi van, el robot es descentralitzà. Caldrà també poder regular la velocitat d'aquests motors. En el cas de que el robot es trobi descentrat en un passadís s'aconseguirà centrar la seva trajectòria fent moure els dos servomotors a velocitats diferents però en el mateix sentit. Aquests dos servomotors, han de poder ser alimentats per la bateria que dugui el robot i han de respondre a les ordres del microcontrolador.

??Dues rodes de tracció i una roda "boja".

Les dues rodes que aniran connectades als motors, hauran de poder-se adherir bé a la superfície per la qual es desplaci el robot perquè no rellisqui, però tenint en compte que no hi hagi excessiu fregament ja que en aquest cas s'augmentaria el consum del robot. La roda boja, donarà estabilitat al robot i es mourà lliurement en funció al moviment que dugui el robot en cada moment.

?? Circuit de control pels motors. Etapa de potència.

Aquest circuit inclourà components com poden ser: transistors, resistències, diodes, etc. Adaptarà les senyals procedents del pic per poder-les adequar als motors.

?? Bateria

La bateria ha de poder ser suficient per alimentar tots els components, però s'ha de tenir en compte que no interessa que sigui gaire pesant, ja que el robot l'ha de transportar al seu damunt. Una bateria molt pesada faria augmentar el consum del robot i això no interessa. Cal preveure també, la duració de la bateria i procurar evitar que el robot es quedi sense energia al mig d'un laberint.

??Sensors de xoc

Els sensors detectors de xoc o bumpers seran els sensors que el robot utilitzarà per detectar les parets i donar les senyals corresponents al PIC. Se'n col·locaran tres: un a cada lateral del robot i un a la part davantera. D'aquesta manera es controlarà la presència de parets des de qualsevol punt del robot. Aquests sensors aniran col·locats a la plataforma base superior del robot i una mica més sortits que la resta d'elements per tal de poder detectar la presència de parets abans que la resta del robot hi col·lisio-n'hi. Els bumpers són uns sensors finals de carrera que donen una sortida binària que en aquest cas indicarà la presència o no de paret.

??Sensors d'ultrasons

Els sensors d'ultrasons s'utilitzaran per detectar la presència de parets i la distancia que hi ha des del robot fins a la paret. Aquests sensors seran útils quan es vulgui determinar la llargada dels camins que formen una intersecció. Es necessitaran tres sensors d'ultrasons, un per situar-lo a cada lateral del robot i un altre per la part davantera. Aquest sensors, aniran situats de tal manera que si el robot col·lisiona amb una paret no es doni cap cop als sensors, per tant, no podran sobresortir de la superfície del robot.

?? Microcontrolador

El microcontrolador serà l'element més important del robot. Serà l'encarregat de llegir la informació procedent dels sensors i de controlar el moviment dels motors en funció d'aquesta informació. El microcontrolador durà gravat en memòria un programa de l'algoritme o algoritmes que es podran utilitzar per resoldre el laberint.

El microcontrolador que es proposa per la construcció del robot, és el PIC 16F876. Algunes de les característiques d'aquest PIC són: 8 kbytes de memòria RAM, 3 timers, 2 mòduls de captura i comparació, 1 convertidor A/D de 10 bits i 5 entrades. Aquest Pic treballa amb només 35 instruccions i les possibilitats que ofereix són suficients per poder-lo utilitzar en l'aplicació del laberint.

?? Altres

A més a més dels components que s'acaben de descriure, també se'n necessitarien d'altres pel circuit de potència (resistències, diodes, transistors, etc.) i elements per subjectar les diferents parts del robot (claus, rosques, brides, etc).

A continuació es mostra un esquema de com estarien posicionades les rodes del robot i els sentits de gir de cada una per poder desplaçar el robot d'un lloc a un altre.

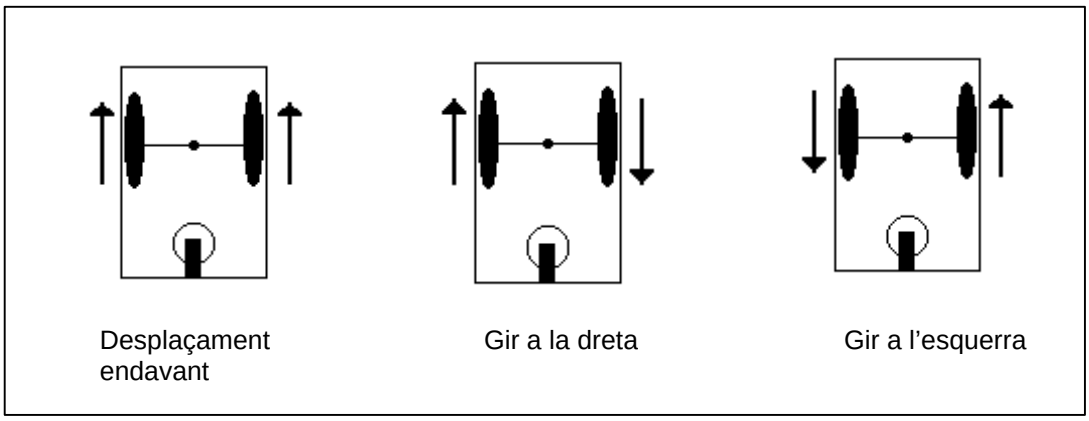


Fig. 7.2. Sentit de gir de les rodes segons el desplaçament del robot

Tal i com s'observa, el moviment de les rodes motrius, serà el que provocarà els girs a la dreta o a l'esquerra, o bé el desplaçament recte. D'igual manera, es podria fer girar les dues rodes cap enrera per tal de desplaçar el robot cap enrera, però en principi, aquesta opció no serà necessària. En comptes de fer moure el robot endarrere, serà més interessant aconseguir que el robot realitzi girs de 180º per poder fer canvis de sentit quan es trobi en un camí sense sortida. Per altra banda, en el moment d'efectuar girs, una roda motriu girarà en un sentit i l'altre roda girarà en sentit contrari. Fent els girs d'aquesta manera s'aconsegueix disminuir l'àrea de gir necessària pel robot.

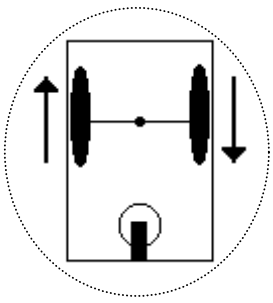


Fig. 7.3. Àrea de gir del robot

7.2.- SOFTWARE

El software del robot estarà constituït principalment per un microcontrolador que durà gravat en memòria un programa mitjançant el qual, i a partir d'uns algoritmes de control, s'analitzarà l'estat dels sensors i en funció de la informació que aquests proporcionin, es transmetran unes ordres determinades cap als motors que faran moure el robot.

Els sensors bumpers que durà el robot donaran una sortida binària (0 o 1) que s'introduirà directament al microcontrolador sense necessitat de ser adaptada. Els sensors d'infrarojos, en canvi, proporcionaran una sortida analògica que anirà en funció de la llargada dels camins. En aquest cas, sí caldrà adaptar la senyal abans de ser introduïda al microcontrolador i això es farà mitjançant un convertidor analògic-digital. La informació procedent dels sensors, una vegada adaptada, pot ser introduïda al microcontrolador a partir d'un port d'entrada que es vagi testejant contínuament a intervals curts de temps. Si només utilitzéssim sensors bumper, també podríem introduir la informació dels sensors mitjançant interrupcions externes que s'activessin quan el robot col·lisionés amb alguna paret, moment en el qual s'activaria algun dels tres bumpers, per tant, caldria disposar de tres entrades d'interrupció externes. Tot i que treballar amb interrupcions suposaria un estalvi de temps degut a que només es coneixeria l'estat dels sensors en els moments necessaris, el programa del microcontrolador no treballarà amb interrupcions sinó que anirà testejant el port d'entrada, ja que d'aquesta manera serà més fàcil portar un control dels últims moviments del robot i es podrà conèixer la llargada dels camins.

El programa del PIC podrà dur un o més algoritmes de resolució, els quals s'han comentat en el capítol quatre. Segons l'algoritme que s'utilitzi per resoldre el laberint, el robot, en arribar a una intersecció, farà uns moviments o uns altres. En funció de l'algoritme també caldrà que el programa disposi d'un sistema per guardar en memòria els últims moviments del robot per poder detectar quan hi ha un camí sense sortida i prendre les decisions adequades.

Una vegada analitzat l'estat dels sensors i els últims moviments del robot, el microcontrolador donarà les ordres oportunes als motors per tal de fer girar el robot cap un sentit o cap a un altre.

Perquè les ordres procedents del microcontrolador es duguin a terme, es necessitarà una etapa de potència (hardware) amb un convertidor digital-analògic (CDA) que adapti les senyals digitals procedents del PIC en senyals analògiques per poder-les aplicar als motors. El microcontrolador que es proposa per la construcció del robot, el 16F876, ja porta incorporat un CDA.

8.- VALORACIÓ ECONÒMICA

En aquesta part del projecte s'estudiarà la viabilitat econòmica del programa Simulador. Caldrà analitzar les despeses de disseny i de programació. A més a més, s'haurà de tenir en compte quins són els possibles destinataris i la quantitat d'ells.

El progrma Simulaber, està destinat principalment a estudiants d'escoles de formació professional, escoles d'enginyeria industrial, d'electrònica, d'informàtica, de telecomunicacions, d'automatització, i a totes aquelles persones, no necessàriament estudiants, interessades en aquestes matèries.

Aquest pot ser un ampli grup de destinataris, però cal tenir en compte que no tots ells necessitaran utilitzar un programa com el Simulaber, la qual cosa fa reduir bastant el nombre de vendes estimat.

Tenint en compte que la majoria de possibles destinataris seran estudiants, caldrà pensar en un preu de venda reduït ja que aquest grup no voldrà comprar un programa que valgui gaires diners.

Els costos del projecte es divideixen en costos d'enginyeria, que inclouran les hores de disseny i les hores de creació de software, i els costos administratius, on s'inclourà els costos referents a les hores d'oficina i els costos de petit material utilitzat. A més a més, caldrà tenir en compte els beneficis i els impostos.

	PREU / HORA	**HORES**	**PREU**
ENGINYERIA			
Disseny	30 €	25 h	750 €
Software	30 €	60 h	1800 €
ADMINISTRATIUS			
Oficina	12 €	40 h	480 €
Petit material	10 €		10 €
SUBTOTAL(1)			3040 €
BENEFICIS	10%		304 €
SUBTOTAL (2)			3344 €
IMPOSTOS (IVA)	16%		535,04 €
TOTAL			**3879,04 €**

Per arrodonir preus i fer més fàcils els càlculs, es suposarà que el cost total del producte serà de 3900 €.

S'estima que el número total d'unitats que es podrien vendre és de 200.

Si es comercialitza el producte a un preu de 30 € la unitat, els beneficis totals obtinguts serien els següents:

	Preu unitat (€)	unitats	Preu
Preu total del producte			3900 €
Despeses fixes per unitat	1	200	200 €
Preu de venta per unitat			30 €

Beneficis totals: (200 unitats * 30 €/unitat) – (200 €+ 3900 €)= 1900 €

9.- PROPOSTES DE MILLORA I TREBALL FUTUR

Per poder augmentar les prestacions del programa i la seva flexibilitat, podrien afegir-se algunes millores en el programa i fer alguns canvis.

Una de les limitacions del programa recau en les dimensions dels laberints. Les graelles on es dibuixen els laberints, tenen unes dimensions determinades i una quantitat de files i columnes invariable. És possible que en algun cas a l'usuari l'interessi crear laberints més grans, amb més caselles, o amb les mides de les caselles diferents. Amb el programa Simulaber no es possible fer aquests canvis i podria ser interessant solucionar-ho.

Relacionat amb les mides de les caselles i la quantitat d'aquestes, apareix indirectament el fet de poder dibuixar laberints amb línies corbes. Quan més petites siguin les caselles que formen la graella, es podran dibuixar corbes amb més precisió i si s'aconsegueix dibuixar laberints amb parets corbes, caldrà també un algoritme de resolució per poder resoldre aquest nou tipus de laberint, ja que cap dels quatre laberints dels que disposa el programa, permet resoldre laberints amb parets corbes.

Una altra millora que es podria afegir al programa, seria la possibilitat de poder crear nous algoritmes de resolució o modificar els existents. Això seria un procés molt complicat ja que cada un dels algoritmes de resolució requereix molt de codi per ser programat i es necessitaria una explicació exhaustivament detallada de com funciona cada algoritme i com es programa perquè funcioni. Tot i així, seria interessant que el propi usuari pogués augmentar les possibilitats del programa.

Un dels principals problemes que té el Simulaber, és la rapidesa. Com s'ha vist en apartats anteriors, el programa tarda bastant en poder carregar un laberint creat i guardat amb anterioritat. Aquest problema s'ha intentar resoldre canviant les estructures del codi per algunes que fossin més ràpides, com per exemple, substituint estructures del tipus If... then... else... end if, per estructures del tipus Select Case.. Case... End Select. Tot i així, no s'ha pogut millorar més aquest aspecte i és un tema important que caldria resoldre.

Per últim, una proposta més de millora referent a la impressió de laberints. El programa Simulab permet imprimir el dibuix dels laberints però no permet canviar les propietats de impressió. No s'obre el quadre de diàleg estàndard de Windows en el qual es poden seleccionar la quantitat de copies o altres característiques d'impressió. El Simulab tampoc permet determinar la posició de la graella en el full on s'imprimeix. Aquesta doncs, podria ser una altra futura millora a fer en el programa.

10.- BIBLIOGRAFIA I ENLLAÇOS

Bibliografia:

Microsoft Visual Basic 6.0. Manual del programador.- Mc Graw Hill.- Madrid, 1998.

PETROUTSOS, EVANGELOS: Visual Basic 6.- ANAYA Multimedia.- Madrid, 1999.

GARCÍA DE JALÓN, JAVIER; RODRÍGUEZ, JOSÉ IGNACIO; BRAZÁLEZ, ALFONSO; FUNES, PATXI; CARRASCO, EDUARDO; CALLEJA, JESÚS: *Aprenda Visual Basic 6.0 como si estuviera en primero.*- Escuela Superior de Ingenieros Industriales, Universidad de Navarra.- San Sebastián, 1998.

CHARTE OJEADA, FRANCISCO: *Visual Basic 6. Guía práctica para usuarios*.- ANAYA Multimedia.- Madrid, 1999.

SMITH, ERIC A.; WHISLER, VALOR; MARQUIS, HANK: *El libro de Visual Basic 6*.- ANAYA Multimedia.- Madrid, 1999.

KERN, HERMAN: T*hrough the Labyrinth: Designs and Meanings over 5,000 years.*- Prestel.- USA, 2000.

CONTY, PATRICK: *The Genesis and Geometry of the Labyrinth: Architecture, Hidden Language, Myths and Rituals*.- Inner Traditions Intl Ltd; 2002.

Enllaços:

Webs relacionades amb Visual Basic:

http://msdn.microsoft.com/vstudio/
http://www.mvps.org/vbnet/
http://www.vbwm.com/
http://www.aprendevisualbasic.com/
http://www.vbsoftware.cl/

Webs de laberints:

http://www.jugarjuegos.com/juegos/java/laberinto/
http://www.juegosliterarios.netfirms.com/laberintos.htm
http://mat.uab.es/aguade/bside.htm
http://www.maromax.net/Juegos/DowloadsJuegos/Estrategia/MazeBall.htm

Webs de robots:

http://www.autric.com/Microbotica%20y%20Mecatronica/granja_de_robot.htm
http://www.microbotica.es/mastall.htm
http://www.rec.ri.cmu.edu/education/webpage/rotorient.htm
http://www.depeca.uah.es/alcabot/alcabot2001/laberinto.htm