Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

División de Ingeniería Mecánica y Eléctrica

"SIMULACION POR COMPUTADORA DEL SISTEMA VISUOMOTOR EN ANFIBIOS"

TESIS PROFESIONAL

Elaborada para obtener el Titulo de

INGENIERO EN COMPUTACION

P o r

VICTOR ARMANDO CRUZ CEBALLOS
JOSE MANUEL CARMONA MENDIETA
FEDERICO DAZA GONZALEZ

Director de Tesis: Dr. Rolando Lara y Zavala

México, D F. Junio de 1984

CAPITULO 1 INTRODUCCION

1.1 INTRODUCCION 1-1
1.2 CONFIGURACION GENERAL DEL SISTEMA VISUAL EN ANFIBIOS 1-2
1.3 ESTUDIOS CONDUCTUALES SOBRE LA RESPUESTA DE ATAQUE EN ANFIBIOS 1-5
1.4 CONFIGURACION DEL ESTIMULO Y DE LA RESPUESTA DE ORIENTACION 1-5
1.5 ESTUDIOS FISIOLOGICOS 1-7
1.6 ESTUDIOS EXPERIMENTALES 1-9
1.7 BIBLIOGRAFIA 1-13

CAPITULO 2 MODELO GLOBAL DEL SISTEMA VISUOMOTOR DE ANFIBIOS

2.1 MODELO GLOBAL DE COORDINACION VISUOMOTORA 2-1
2.2 TEORIA DE ESQUEMAS 2-3
2.3 ETOGRAMA DE ESPACIO DE ESTADOS DE LOS ANFIBIOS . . 2-4
2.4 BIBLIOGRAFIA 2-14

CAPITULO 3 SIMULACION EN LISP

3.1 INTRODUCCION 3-1
3.2 SIMULACION EN LISP 3-2
3.3 BIBLIOGRAFIA 3-12

CAPITULO 4 SIMULACION EN COMPUTADORA DEL MODELO GENERAL

4.1 INTRODUCCION 4-1
4.2 DESCRIPCION DE SUBMODELOS EMPLEADOS 4-1
4.2.1 MODELO DE PERCEPCION DE OBJETOS FIJOS 4-2
4.2.2 MODELO DETECTOR DE HUECOS 4-5
4.2.3 MODELO DE PERCEPCION DE PROFUNDIDAD DE HUECOS . 4-9
4.3 DIAGRAMA GENERAL 4-11
4.3.1 DESCRIPCION DE MODULOS 4-15
4.4 VARIABLES 4-17
4.5 PROGRAMA EN BURROUGHS 7800 4-19
4.5.1 FUNCIONAMIENTO 4-19
4.5.2 SIMULACION DE ATAQUE, MODELO CON HUECOS . . . 4-20
4.5.3 SIMULACION DE ATAQUE, MODELO SIN HUECOS . . . 4-21
4.5.4 SIMULACION DE HUIDA 4-21
4.5.5 SIMULACION DE RESPUESTA FOTOTAXICA 4-22
4.5.6 LISTADO Y RESULTADOS 4-22
4.5.7 ATAQUE 4-28
4.5.8 HUIDA 4-35
4.5.9 FOTOTAXICA 4-38
4.6 GRAFICACION EN CROMEMCO 4-41
4.6.1 LISTADO Y RESULTADOS CROMEMCO 4-44
4.7 BIBLIOGRAFIA 4-48

APENDICE A

A.1	LENGUAJES DE PROGRAMACION EN LA INTELIGENCIA ARTIFICIAL	A-1
A.2	HISTORIA Y CARACTERISTICAS DE LISP	A-2
A.3	LENGUAJE LISP	A-3
A.4	EXPRESIONES SIMBOLICAS	A-4
A.5	ATOMOS ESPECIALES	A-5
A.6	FUNCIONES ARITMETICAS	A-6
A.7	FUNCIONES ELEMENTALES	A-8
A.8	FUNCIONES PARA MANEJO DE LISTAS	A-11
A.9	ENTRADA Y SALIDA	A-13
A.10	LISTA DE PROPIEDADES	A-14
A.11	META-EXPRESIONES EN LISP	A-16
A.12	PREDICADOS	A-19
A.13	EXPRESIONES CONDICIONALES	A-23
A.14	FUNCIONES CONDICIONALES	A-25
A.15	EXPRESION ITERATIVA DO	A-26
A.16	FUNCIONES QUE MANIPULAN FUNCIONES	A-28
A.17	BIBLIOGRAFIA	A-29

APENDICE B

APENDICE C

APENDICE D

CAPITULO 1

INTRODUCCION

1.1 INTRODUCCION

El cerebro es una estructura de gran complejidad que tiene como función primordial permitir al animal definir cuál es la conducta más adecuada ante una situación especifica del mundo externo. Por esta razón es el centro de interacción sensorimotora, cuya función es relacionar cada estímulo externo con una respuesta motora especifica; tal respuesta puede ser desde un acto reflejo simple hasta la interacción con el mundo externo, dependiendo del grado de complejidad del sistema nervioso del animal.

Se tiene ya un conocimiento bastante amplio sobre los mecanismos por medio de los cuales dos neuronas pueden comunicarse entre sí; sin embargo, es poco el conocimiento que se tiene para saber cómo todo un grupo de neuronas pueden dar origen a una función especifica, o cómo estas pueden ser capaces de regular la conducta del animal.

En este sentido en la actualidad se conocen los mecanismos de respuesta de protección de varios animales, en invertebrados tenemos como ejemplo los estudios en la **Aplysia** o el **Acocil**, en los cuales se ha estudiado detalladamente los mecanismos neuronales de varias conductas y sus modificaciones plásticas como **habituación**, **sensibilización** e incluso **condicionamiento**.

Todos los estudios que se han hecho en invertebrados respecto a la forma cómo su sistema nervioso procesa la información no son fácilmente extrapolables a los sistemas nerviosos de los vertebrados, debido, entre otras cosas, a que la conducta de estos es mucho más compleja. Así pues ha surgido la necesidad de elegir un modelo biológico que tenga las siguientes características:

1) Que no sea tan simple como el de los invertebrados, ni tan complejo como el de los vertebrados superiores.

2) Que pueda actuar como un puente para hacer estudios en ambos casos.

De esta manera se ha elegido al **SISTEMA VISUOMOTOR DE LOS ANFIBIOS** (1) en particular el de **ranas y sapos**, como el modelo que nos servirá para el estudio de la coordinación sensorimotora. Ya que presenta ciertas características entre las que sobresalen:

1) Animales casi estáticos.

2) Conducta controla[illegible] su totalidad por su sistema visual.

3) Tienen una forma [illegible]ponder muy específica.

1.2 CONFIGURACION GENERAL DEL SISTEMA VISUAL EN ANFIBIOS

El ojo de los anfibios es muy similar al de los humanos, con la diferencia de que el cristalino es de mayor tamaño y la distancia del lente hacia la retina es más pequeña. Por ello, su agudeza visual es mucho menor que la de los humanos. La posición periscópica de los ojos de estos animales les permite tener un campo visual de casi 360 grados, con lo cual casi cada punto del espacio externo se proyecta hacia un punto en la retina.

La retina de los anfibios está constituída por diversos grupos de células entre las más importantes se hallan las células bipolares, las horizontales, las amacrinas y las ganglionares que son las células de la retina que envían sus axones a otras zonas del cerebro (Figura 1).

Desde un punto de vista fisiológico, se han descrito varios tipos de células ganglionares:

1) Tipo 1, detectores de bordes fijos;

2) Tipo 2, detectores de bordes convexos;

3) Tipo 3, detectores de contraste;

4) Tipo 4, detectores de oscurecimiento.

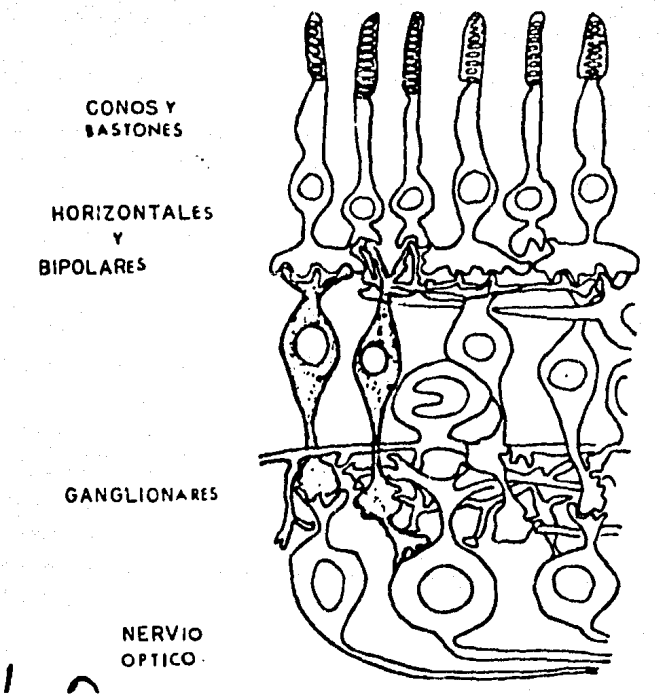


Figura 1.- Circuito de las células de la retina. Los receptores actúan sobre las células bipolares y horizontales; las células bipolares a su vez actúan sobre las células amacrinas y las células ganglionares que son las células eferentes.

Las fibras eferentes de la retina de los anfibios proyectan retinotópicamente es decir, punto a punto y con la misma distribución espacial hacia varias regiones tanto del diencéfalo como del mesencéfalo (Figura 2).

Las regiones diencefálicas que reciben directamente fibras ópticas son el cuerpo geniculado; el núcleo de Bellonci, el pretectum y el cuerpo uncinado, mientras que las zonas mesencefálicas visuales son el tectum óptico y el tegmentum.

El tectum óptico; además de recibir aferentes ópticos, recibe información de otros centros visuales como el cuerpo geniculado lateral, el pretectum y tectum contralateral. El tectum, a su vez, envía fibras hacia regiones mesencefálicas

y diencefálicas. Algunas de las zonas mesencefálicas que reciben información del tectum están relacionadas con la actividad motora ya que sus fibras llegan a la médula espinal, al núcleo oculomotor, al cerebelo, etc. Esto sugiere que el tectum es un centro de coordinación visuomotor puesto que recibe aferentes visuales y proyecta directamente a centros motores. Además, el tectum envía fibras hacia otros centros visuales como el pretectum y el cuerpo geniculado. De esta manera, el tectum forma redes de interacción entre tectum-pretectum y tectum-cuerpo geniculado, lo cual sugiere que el procesamiento visual dentro de estas tres zonas debe ser determinante para regular la actividad visuomotora del animal.

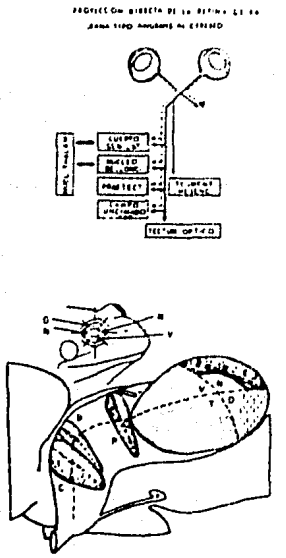


Figura 2.- Proyección retinotópica de la retina hacia el diencéfalo y el mesencéfalo. La retina proyecta hacia el cuerpo geniculado lateral, el núcleo de Bellonci y el tálamo posterior incluyendo al pretectum y al cuerpo uncinado en la región del diencéfalo y hacia el tectum y el tegmentum en la región del mesencéfalo. En la figura sólo se muestra el tectum óptico.

1.3 ESTUDIOS CONDUCTUALES SOBRE LA RESPUESTA DE ATAQUE EN ANFIBIOS

En los sapos, la respuesta de ataque hacia la presa comprende una secuencia fija de actividades motoras:

1) Respuesta de orientación hacia la presa

2) Seguimiento de la presa

3) Fijación y ataque

4) Tragar

5) Limpiar

Esta secuencia conductual es una cadena de estímulo-respuesta donde cada reacción desencadena la siguiente conducta.

Las lesiones del tectum óptico traen como consecuencia que tanto la respuesta de orientación hacia una presa como la respuesta de huida ante un predador desaparezcan. Una estimulación eléctrica puntual en diferentes regiones del tectum -por medio de un tren de pulsos- desencadena la respuesta motora de orientación y ataque hacia la presa. La respuesta de orientación se dirigirá a la misma posición en el espacio codificada por la proyección retino-tectal. Esto sugiere que el tectum es una región donde el espacio visual se codifica punto a punto y donde cada punto representa a su vez la posición hacia la cual se va a orientar el animal cuando la presa se encuentra en una posición específica. Por esto, el tectum es considerado un modelo ideal para estudiar el proceso mediante el cual una información visual se transforma en una conducta motora.

Si existe una lesión unilateral en la región del pretectum, la respuesta de ataque hacia estímulos que se mueven en el campo colateral se ve considerablemente facilitada, aún cuando el estímulo sea de tipo predador. Con una lesión bilateral, este efecto se expande hacia todo el campo visual. Esta evidencia sugiere que el pretectum podría controlar la respuesta de ataque del animal por medio de una acción inhibitoria sobre el tectum.

1.4 CONFIGURACION DEL ESTIMULO Y DE LA RESPUESTA DE ORIENTACION

Se ha observado que la respuesta de orientación en anfibios depende de la forma, velocidad, tamaño,

configuración y contraste respecto del fondo, de la presa.

Ewert (*) demostró que la expansión de un rectángulo de 2 grados en la dirección del movimiento del estímulo facilita la frecuencia de la respuesta de orientación. Si la expansión se realiza perpendicularmente a la dirección del movimiento, la respuesta de orientación se inhibe considerablemente; si la expansión se produce en ambos ejes, se observa una facilitación inicial y una inhibición posterior (Figura 3A). Ewert también ha mostrado que el reconocimiento entre los diferentes configuraciones del estímulo es independiente de la dirección del movimiento. Por otra parte, Ewert descubrió que, cuando se presentan estímulos dobles alineados en la dirección del movimiento, la respuesta de orientación no se desencadena tan eficazmente como en el caso de utilizar un solo objeto. Si los dos estímulos son alineados perpendicularmente a la dirección del movimiento, la respuesta de orientación se inhibe profundamente. Cuando la distancia entre los objetos aumenta, el efecto inhibitorio desaparece. Ingle (**) estudió con más detalle los efectos de estímulos dobles, alineados en la dirección del movimiento y halló que, cuando los dos estímulos se mueven hacia la cabeza del animal, éste tiene la tendencia a atacar al primero de los objetos, mientras que ataca al centro de los estímulos cuando éstos se alejan. Además observó que, si se presentan estímulos de 2 grados, con una distancia entre sí de 4 grados, se produce un efecto de facilitación. Esto sugiere que el animal considera los dos estímulos como un solo objeto. Si la distancia entre estímulos se incrementa de 8 a 16 grados, el animal reacciona ante los dos estímulos en forma independiente.

Ingle ha estudiado también la conducta del animal enfrentado a varios estímulos presentes en su campo visual y ha demostrado que los animales prefieren determinado tipo de estímulo, según su estado motivacional: los animales hambrientos prefieren estímulos de 16 grados a estímulos de 6 grados; esta relación se invierte en animales normales. Asimismo prefiere objetos cercanos y novedosos a objetos alejados y conocidos.

(*) Jorg-Peter Ewert, Profesor e Investigador de la Universidad de Kassel, Alemania del Oeste

(**) David Ingle, Profesor e Investigador de la Universidad de Brandeis: E.U.A.

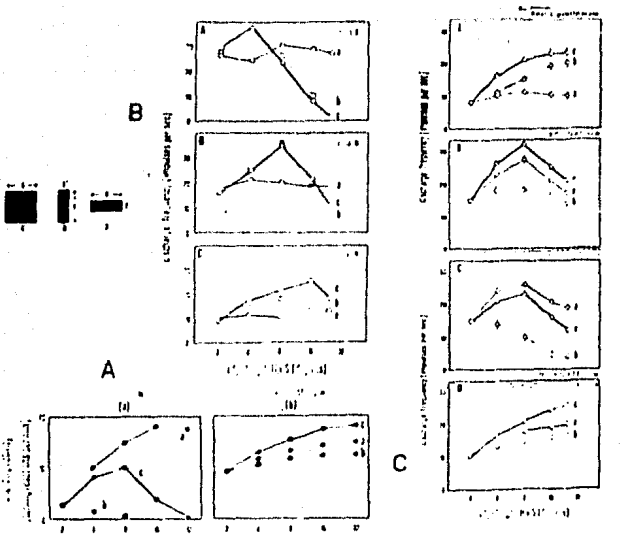


Figura 3.- (A) Número de respuestas de orientación ante diferentes tipos de estímulos. La respuesta de orientación se facilita ante estímulos tipo a) pero se inhibe ante los del tipo b). Una combinación de ambos efectos se observa ante estímulos del tipo c). A la derecha de la figura se observa que esta discriminación desaparece cuando se lesiona el pretectum. (B) Respuesta de las células ganglionares 2, 3 y 4 a las configuraciones a,b y c del estímulo. (C) Respuesta de las células tectales y pretectales ante las diferentes configuraciones del estímulo. Nótese la semejanza entre la respuesta conductual y la respuesta de la célula tectal.

1.5 ESTUDIOS FISIOLOGICOS

En gran cantidad de estudios experimentales se ha intentado analizar los mecanismos neuronales de estas conductas. Ewert ha mostrado que los animales con lesiones en el tectum pierden tanto la capacidad de orientación hacia la presa como la de huida ante el predador. Asimismo, una estimulación puntual en el tectum provoca la respuesta de orientación del animal a la zona respectiva que corresponde a la proyección retino-tectal. Esto sugiere que el tectum

juega un papel preponderante en la respuesta de orientación. Ewert ha demostrado también que, si existen lesiones en el pretectum, el animal será incapaz de discriminar entre las diferentes configuraciones del estímulo. Asimismo, responderá indiscriminadamente a todo tipo de estímulos, incluyendo aquellos que normalmente son considerados como predadores. Esto sugiere que la interacción entre la retina-tectum y pretectum (2) podría ser el factor que permite al animal diferenciar las distintas configuraciones del estímulo y seleccionar un tipo específico de estímulo.

Ewert trató de determinar el papel que juega cada una de estas zonas en el control de estas conductas y, con ese fin, estudió la respuesta neuronal de la retina, tectum y pretectum ante las distintas configuraciones del estímulo. Las células de la retina de los tipos 2, 3 y 4 no modifican considerablemente su respuesta ante estímulos de tipo presa de diferentes dimensiones. En el caso de estímulos que se expanden perpendicularmente a la dirección del movimiento, se facilita la respuesta de las células ganglionares de los tipos 2 y 3 dependiendo de su campo receptivo; cuando el estímulo es mayor que el campo receptivo se produce una inhibición. La inhibición es más fuerte en las células ganglionares del tipo 2 que en las del tipo 3. Las células ganglionares del tipo 4 incrementan su frecuencia de respuesta según el tamaño del objeto. Todas las células ganglionares incrementan su respuesta en función de la velocidad y el contraste del estímulo (Figura 3B).

Gracias a estos estudios, Ewert concluyó que el procesamiento de información en la retina no explicaba de manera satisfactoria la sensibilidad conductual ante las diferentes configuraciones del estímulo. Estudió entonces la respuesta de las células tectales y pretectales ante las diferentes configuraciones de un estímulo y descubrió que algunas células tectales presentan una respuesta de facilitación ante estímulos que se alargan en la dirección del movimiento; una respuesta de inhibición ante estímulos que se alargan perpendicularmente a la dirección del movimiento, y una facilitación inicial seguida de una inhibición ante estímulos que se alargan en ambas direcciones. Cuando se lesiona el pretectum, las células pierden toda la capacidad de discriminación (Figura 3C). Ewert también mostró que algunas de las células tectales reconocen las distintas configuraciones del estímulo independientemente de la dirección, mientras que otras células tectales son sensibles a la dirección y responden más intensamente a estímulos de tipo predador. De acuerdo con esto, postuló que las células tectales capaces de discriminar entre estímulos de tipo presa o predador podrían estar regulando la conducta del animal ante estos estímulos así como la dirección hacia donde el animal debe moverse. Los estudios han mostrado que la mayor parte de las células

pretende ser la óptima en un tiempo mínimo para preservar la vida del animal.

Se ha observado que la respuesta motora en animales con sistema nervioso simple es una acción específica, en cambio, cuando el sistema nervioso del animal es más complejo, la respuesta motora dada por éste es mucho más versátil, y no es tan predecible como en los primeros. Las disciplinas antes mencionadas postulan también que el cerebro contiene estructuras de información, las cuales dado un estímulo sensorial específico producen una respuesta motora determinada. Este tipo de estructuras son llamadas **ESQUEMAS**, y dependiendo de la capacidad del sistema nervioso del animal será posible que éste genere nuevos esquemas o superestructuras. Sin embargo aún no se sabe como se lleva a cabo este complicado proceso de información.

La mayoría de los estudios realizados por estas disciplinas se han llevado a cabo en animales cuyo sistema nervioso es relativamente sencillo, en los cuales es posible hacer estudios de estos procesos a un nivel neuronal y de esta forma tratar de obtener información acerca del mecanismo que usa el sistema nervioso para controlar la respuesta motora del animal. En algunos estudios llevados a cabo en invertebrados, se ha encontrado que aportan muy poco cuando se trata de relacionarlos con sistemas nerviosos más sofisticados, ya que tanto el número de neuronas como la complejidad en las interacciones aumenta considerablemente, lo cual puede dar lugar a una gran cantidad de mecanismos diferentes.

Lo anteriormente expuesto nos sugiere conformar estudios en animales cuyo sistema nervioso sea lo suficientemente simple para realizar un análisis significativo pero que al mismo tiempo sea lo suficientemente complejo para que éste pueda ser aplicado a vertebrados con un sistema nervioso más complejo, y nos de así una idea más completa de cómo el cerebro procesa la información sensorimotora para dar una respuesta motora con un nivel mínimo de complejidad.

Las ranas y sapos poseen un sistema nervioso que cumple con las características antes mencionadas, de los estudios realizados en estos vertebrados se han podido postular algunas respuestas sensorimotoras típicas en estos animales, las cuales se exponen a continuación:

1) Se orientan hacia animales cuya longitud mayor coincide con la dirección de avance, y la respuesta de orientación se inhibe con animales con características contrarias.(3)

pretectales tienen un campo receptivo amplio y que son más sensibles a estímulos tipo predador que a estímulos tipo presa. Sin embargo, una neurona pretectal que posea un campo receptivo relativamente pequeño responde principalmente a estímulos de tipo predador (Figura 3C). Por esta razón, Ewert ha postulado que esta neurona podría inhibir la actividad de las células tectales cuando un estímulo se presente tipo predador. En esta forma, Ewert sugiere que la interacción entre retina-tectum y pretectum podría controlar el reconocimiento predador-presa.

Por su parte, Ingle propone que el pretectum está relacionado con las preferencias de tamaño de la presa y la selección entre varios estímulos presentes simultáneamente en el campo visual del animal. Asimismo, postula que, en condiciones normales, las células tectales son controladas básicamente por aferentes ganglionares del tipo 2, mientras que las células ganglionares de los 3 y 4 son inhibidas por el pretectum. Propone que las células de la retina del tipo 2 pueden contrarrestar la inhibición pretectal por medio de sistemas de facilitación a través de circuitos recurrentes, pero con gran latencia en la respuesta. De esta forma, explica por qué el animal prefiere estímulos pequeños. Cuando la inhibición pretectal se anula -ya sea por un estado motivacional incrementado o por lesiones-, las células tectales son controladas por las células ganglionares de los tipos 3 y 4. Con ello, se modifica el campo receptivo de las células tectales y se reduce la latencia de la respuesta ya que estas fibras llegan a regiones más cercanas al soma de las neuronas.

1.6 ESTUDIOS EXPERIMENTALES

El hecho de que la mayoría de los animales, incluyendo al hombre, sean capaces de enfrentarse inteligente y adaptablemente al medio ambiente, ha sido objeto de estudio tanto de la filosofía como de otras ciencias.

Hasta hace sólo algunos años la filosofía era la única rama del conocimiento que trataba de explicarse, entre otras cosas, cómo se llevaban a cabo los procesos de el entendimiento, la inteligencia, y planeación de los seres vivos, así como su concepción del universo.

En la actualidad existen otras disciplinas, tales como la Etología, los estudios teóricos sobre el cerebro, la Psicología y la Inteligencia Artificial, que tratan de dar una respuesta más científica a las incógnitas del comportamiento animal. Estas disciplinas postulan que el cerebro es básicamente el centro de coordinación sensorimotora, y que la respuesta motora generada por éste

2) Evitan o huyen de animales que cubren un ángulo de visión mayor a 20 grados.(3,4)

3) Muestran constancia dentro de un radio de 30 cm.(5-6)

4) Se habitúan a un estímulo que se presenta frecuentemente.(7-8)

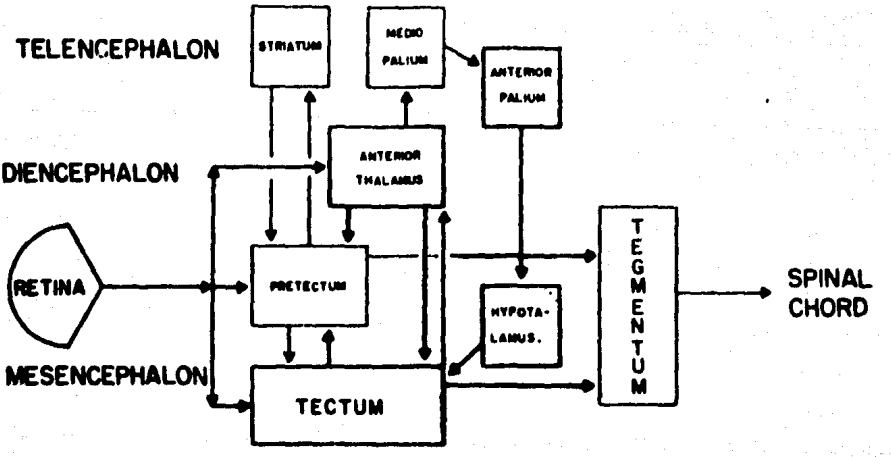
5) Planean su ruta hacia la presa sin retroalimentar la información del medio ambiente. Si el estímulo se mueve fuera del campo visual del anfibio, éste no busca al estímulo, lo que nos indica que no poseen una representación permanente del objetivo.(9-11)

6) En un medio ambiente complejo, compuesto por barreras y zanjas, planean su trayectoria hacia la presa, dependiendo de la situación especica, en términos de relación de la presa y los obstáculos, así como de las dimensiones de los obstáculos. Esto sugiere que la respuesta motora es seleccionada en términos de la representación tridimensional del medio ambiente.(10-13)

Los estudios realizados en este tipo de animales nos dan argumentos para pensar que la respuesta, o secuencia de respuestas motoras dadas corresponden a un estímulo específico que se recibe del medio en términos de objetos fijos y objetos en movimiento.

De experimentos y observaciones llevadas a cabo en anfibios podríamos a grandes rasgos definir cuáles son las principales zonas cerebrales que intervienen en la selección de la respuesta motora, y el papel que juegan en este proceso. El pretectum (Figura 4) aparentemente es sensible a estímulos de tipo predatorio, también en interacción con el telencéfalo tiene células que intervienen en procesos de memoria y aprendizaje; esta zona juega un papel importante en la respuesta de huida y en el control de la respuesta de orientación. El tálamo anterior controla la percepción de objetos fijos. Se ha observado que cuando esta zona es lesionada, el animal es incapaz de evadir objetos inmóviles. El tálamo también juega un papel importante en la respuesta fototáxica, es decir es la preferencia que presentan los animales a buscar lugares con determinado color, el azul en este caso. Por ejemplo si el animal por alguna razón debe esconderse, huir de algún predador, o simplemente buscar algún nuevo lugar para satisfacer alguna de sus necesidades, tenderá a irse a lugares cuyo tono se parezca más al azul.

Figura 4.- Organización general del sistema visuomotor de las ranas. La retina envía fibras al tectum, pretectum y al tálamo anterior, en forma retinotópica. El tectum establece interacciones bidireccionales con el pretectum y el tálamo anterior. El pretectum a su vez mantiene interacciones bidireccionales con la estructura telencefálica striatum. El tectum, mediante el tálamo anterior, el palium anterior y medio, y el hipotálamo, interactúa con el telencefálo. El tectum y el pretectum, posiblemente en combinación con el tegmentum, pueden controlar la respuesta motora de el animal por medio de sus proyecciones a la medula dorsal.



1.7 BIBLIOGRAFIA

1. Lara y Zavala, R. Neural models of the visuomotor system of amphibia. Tesis de Doctorado (Ph.D.) Universidad de Massachusetts en Amherst.

2. Lara, R., Arbib, M.A. A neural model of interaction between pretectum and tectum in prey selection. Cognition and Brain Theory. 5(2) 149-171, 1982.

3. Ewert, J.P., The visual system of the toad: Behavioral and physiological studies on pattern recognition system. K.V. Fite ed. Academic Press, New York, San Francisco, London, 1976

4. Ewert, J.P., Neuroethology. An introduction to the neurophysiological fundaments of behavior. Springer Verlag, Berlin, 1980

5. Ewert, J.P., and Gebauer, L.: Groessenkonstanzphaenomene im Beutef ang verhalten der Erdkroete (Bufo bufo L.). J. Comp. Physiol. 85:303-315, 1973

6. Ingle, D. and Cook, J. The effect of viewing distance upon size preference of frogs for prey. Vision Res. 17:1009-1013, 1977.

7. Ewert, J.P. and Ingle, D.: Excitatory effects following habituation of prey-catching activity in frogs and toads. J. Comp. Physiol. Psych. 77:367-374, 1971.

8. Ewert, J.P. and Kehl, W.: Configurational prey selection by individual experience in the toad Bufo bufo. J. Comp. Physiol. 126,105-114, 1978.

9. Ingle, D. Detection of stationary objects by frogs (Rana pipiens). J. Comp. Physiol. Psych. 91:1359-1364, 1977.

10. Ingle, D. Spatial Vision in Anurans. In: The Amphibian Visual System. K. V. Fite ed. Academic press, New York, San Francisco, London. 1976.

11. Lock, A. and Collet, T.: A toad's devious approach to its prey: a study of some comlex uses of depth vision. J. Comp. Physiol. 131:179-189, 1979.

12. Collet, T. The three dimensional world of the toad. Proc. Roy. Soc. Lond. 206:481-497.

13. Collet, T. Do toads plan routes? A study of the detour behavior of Bufo viridis. J. Comp. Physiol. 146:261-267; 1982.

14. Lara R., Carmona, M., Daza, F. y Cruz, A. A Global Model of Visuomotor Coordination in Toads. Journal of Theoretical Biology. En prensa. 1984

CAPITULO 2

MODELO GLOBAL DEL SISTEMA VISUOMOTOR DE ANFIBIOS

2.1 MODELO GLOBAL DE COORDINACION VISUOMOTORA

Se ha tratado de explicar, mediante modelos teóricos, como puede en sí un anfibio o cualquier otro animal reconocer una presa de un predador, cómo puede tener noción de la profundidad o alejamiento de los objetos que percibe del medio ambiente, cómo se lleva a cabo la respuesta de orientación y otras actitudes, que a pesar de ser sencillas resultan de gran complejidad cuando tratan de explicarse a nivel neuronal.

Trataremos ahora de proponer un modelo global de los posibles mecanismos neuronales responsables de la coordinación visuomotora en ranas y sapos, y los procesos de información necesarios para que se produzca una respuesta motora especifica dependiendo del estímulo sensorial. El modelo se basa principalmente en el reconocimiento de presa y el reconocimiento de predador.

Los postulados del módulo de los mecanismos que controlan la conducta visuomotora del animal se enumeran a continuación:

1) Las células ganglionares de la retina inician el procesamiento visual de la información definiendo la velocidad, el contraste y el tamaño de los objetos fijos así como el de los objetos en movimiento.

2) El conjunto de las células ganglionares da origen a una representación matricial, plana del universo.

3) La escena de la matriz retiniana se envía a varias regiones cerebrales donde se generan nuevas matrices, las cuales procesan diferentes características del estímulo, tanto por procesos internos como por la interacción con otros zonas cerebrales.

4) El tectum está constituido por varias matrices con diferentes funciones y cuyo sustrato anatómico es la columna tectal.

   a) Reconocimiento de la presa.

   b) Reconocimiento del predador.

   c) Percepción de profundidad tanto hacia la presa como hacia el predador.

   d) Selección de la presa.

   e) Posición de la presa.

5) El tálamo-pretectum está constituido por varias matrices con las siguientes funciones:

   a) Reconocimiento del predador.

   b) Reconocimiento de objetos fijos (barreras o zanjas)

   c) Detector de huecos.

   d) Percepción de profundidad tanto hacia el predador como a los objetos fijos.

   e) Constancia del tamaño real de los objetos.

   f) Sensibilidad a los colores del medio ambiente.(respuesta fototáxica).

   g) Reconocimiento y selección de la presa.

6) El telencéfalo realiza las siguientes funciones:

   a) Control del estado motivacional del animal.

   b) Habituación a la respuesta de orientación y huida

   c) Procesos relacionados con memoria y aprendizaje.

7) La información proveniente tanto del tectum como del pretectum puede activar diversas conductas motoras las cuales parece que estan preprogramadas y que han sido denominadas **ESQUEMAS MOTORES.** Los

esquemas motores necesitan de dos fuentes de información para ser activados: La señal de comando, dependiendo del tipo de células activadas y el sistema de información que, una vez definido el comando, envía la información al esquema motor para que éste realice la acción correspondiente. Por ejemplo, la distancia que debe de recorrer el animal antes de reorientarse, la magnitud del salto en caso de que tenga que pasar por encima de algún obstáculo, etc. Para este modelo se han considerado los siguientes esquemas motores:

a) Caminar.

b) Orientar.

c) Atacar.

d) Asechar.

e) Saltar.

f) Bajar.

g) Brincar.

h) Permanecer inmóvil.

2.2 TEORIA DE ESQUEMAS

Un esquema, es pues, una estructura de información que relaciona una circunstancia visual específica con una serie de conductas motoras (1). La teoría de esquemas propuesta en este trabajo nos permite explicar la conducta de diversos animales en función de su representación interna del universo y nos permite postular el tipo de operaciones que tiene que realizar el sistema nervioso de estos animales. El esquema está constituido por los siguientes elementos:

1) Esquemas perceptuales que reconocen objetos significativos en la vida del animal, como son una presa, un predador, una barrera, etc. Asimismo los esquemas perceptuales definen la métrica del espacio externo; es decir, la distancia de la presa, la altura de la barrera, etc.

2) Esquemas motores que representan las pautas motoras fundamentales del animal o del robot, como son caminar, orientar, brincar, etc.

3) Programa de coordinación sensorimotora que define la secuencia de activación de esquemas perceptuales y motores para alcanzar un fin específico.

4) Existe competencia entre diversos esquemas del mismo nivel, es decir una relación heterárquica para controlar la conducta motora.

5) Existe una cooperatividad entre diferentes esquemas para alcanzar el estado meta resultándose una relación jerárquica donde el esquema principal define la meta y llama a un conjunto de esquemas, cada uno con una submeta específica para alcanzar el objetivo.

6) Existe un estado meta al cual tiende la actividad concertada de un grupo de esquemas sensorimotores.

7) Existen estados fallidos que representan que el esquema principal no pudo llegar a su objetivo.

8) El conjunto de esquemas que define la conducta del animal ante diferentes circunstancias se le llama el etograma de espacio de estados o esquema global del animal o del robot.

Con estos elementos procederemos a ilustrar cómo esta teoría nos permite explicar la conducta visuomotora de los anfibios, en particular de las ranas y los sapos.

2.3 ETOGRAMA DE ESPACIO DE ESTADOS DE LOS ANFIBIOS

El modelo se representa por medio de un etograma de espacio de estados (2). El etograma es un diagrama que nos muestra las alternativas o estados que puede alcanzar el animal a partir de un punto inicial. El etograma de espacio de estados de los anfibios lo mostramos en la figura 1. Existen 4 tipos de esquemas principales:

1) En presencia de un predador.

2) En presencia de un presa.

3) En presencia de un consugue.

4) En un ambiente donde no hay estímulos en movimiento.

MODELO GLOBAL DEL SISTEMA VISUOMOTOR DE ANFIBIOS

Los estímulos en movimiento están modulados por un sistema jerárquico que define quien de ellos va a ser activado dependiendo de su experiencia pasada. Estos tres esquemas iniciales están en competencia entre sí para tomar control de la conducta del animal; ésta competencia, como mencionamos antes, está dirigida por el esquema general de estímulos en movimiento. Si este esquema no está activo, entonces el esquema de una circunstancia sin objetos móviles se activa.

Al activarse cualquiera de los esquemas principales, se define inmediatamente el objetivo o meta del mismo y se empiezan a activar diversos esquemas para alcanzarlo.

En el caso de que la rana detecte la presencia de un predador (figura 2), ésta actuará de la manera más conveniente para evadirlo; es decir, si se trata de un predador de tipo aéreo la rana tenderá a agacharse para presentar menos vulnerabilidad ante un intento de ataque. Si el predador es de tipo terrestre y se encuentra a una distancia considerable, el anfibio se dirigirá hacia el hueco más atractivo que se encuentre en la trayectoria menos cercana del predador, si por el contrario el predador se encuentra demasiado cerca, éste representa alto grado de peligrosidad para la supervivencia de la rana, por lo que huye a saltos hacia el hueco que considere más seguro y que se encuentre fuera de la trayectoria de peligro.

Es importante mencionar que el anfibio ante la presencia de una presa y un predador, el esquema de mayor prioridad es el de supervivencia, por lo que la rana hace caso omiso de la presa y procede a seguir las acciones descritas anteriormente, en el que la rana detectaba la presencia de algún predador.

Cuando no existen ni presas ni predadores en el medio ambiente de la rana, ésta permanece inmóvil esperando la llegada de una presa; pero si se presentan estímulos que alteren el estado "emocional" del animal, tales como sed, frío, calor, hambre, temor, etc. el anfibio se dirigirá instintivamente hacia el lugar más adecuado para reestablecer su estado de pasividad; por ejemplo, si lo que motivo al animal a moverse de su lugar fué la falta de humedad éste tenderá a irse a lugares cercanos al agua, si fué el temor tratará de encontrar lugares donde se sienta más seguro.

En la figura 3 se muestra que al activarse el esquema de presencia de una presa, la primer acción motora que llama es orientarse a la presa; en estas condiciones tres posibles esquemas empiezan a competir entre sí para tomar control de la conducta del animal:

MODELO GLOBAL DEL SISTEMA VISUOMOTOR DE ANFIBIOS

1) Presa sin obstáculos.

2) Presa detrás de una barrera.

3) Presa detrás de una zanja.

Una vez que se activa uno de ellos, dependiendo de la condición métrica del espacio, es decir, la distancia de la presa o la distancia de la barrera, etc., se activan diferentes esquemas que a su vez compiten entre sí para tomar control de la conducta animal. En el caso de que se active el esquema de presa sin obstáculos, dependiendo de la distancia de la presa se activan el esquema de aproximación a la presa, en caso de que ésta esté alejada, regresando el control al esquema perceptual para definir si se activa el esquema de atacar y comer, en caso de que la presa esté cerca, lo cual lleva al animal al estado meta. Si el esquema perceptual que se activa es presa en presencia de una barrera, dependiendo de la distancia presa-barrera y de la altura de la barrera se pueden activar tres subesquemas.

En caso de que la presa esté cerca de la barrera el esquema: 'aproximate a la barrera' se activa, el cual, a su vez, envía el control de la conducta al esquema de ataque a la presa; en el caso de que la presa esté alejada y la altura de la barrera sea pequeña, entonces el esquema salta la barrera se activa, retornando el control al esquema perceptual inicial para redefinir la acción presente. En el caso de que la presa esté alejada de una barrera alta, entonces el esquema de rodeo se activa. Este esquema se activa si en el espacio del animal se encuentra un hueco por donde pueda alcanzar la presa. En caso de que exista este hueco, entonces el animal se oriente y camina hacia él, retornando el control al esquema inicial.

En la misma forma se activan los diversos esquemas en presencia de una zanja entre el animal y la presa, pero ahora dependiendo de la profundidad y anchura de la zanja.

El etograma del esquema en presencia de una presa nos muestra cómo el animal utiliza y coordina diversos esquemas sensorimotores para alcanzar un objetivo; los esquemas perceptuales no sólo reconocen objetos sino circunstancias que le permiten definir adecuadamente las conductas necesarias para confrontarlas adecuadamente. En forma similar se han propuesto los esquemas para la huida de un predador ante diferentes circunstancias o para el apareamiento con un conyugue. En el caso de un robot similarmente se definirían sus objetivos y los esquemas perceptuales y motores que regularían su conducta.

MODELO GLOBAL DEL SISTEMA VISUOMOTOR DE ANFIBIOS

El esquema Fototáxica se presenta cuando no hay objetos móviles, y efectúa lo siguiente: evalúa huecos y escoge el más atractivo en ancho, profundidad, posición y color de fondo, camina hacia ese hueco, y permanece en esa posición. Los anfibios tienen preferencia por los huecos de fondo color azul (3).

Es importante tratar de definir como las funciones postuladas podrían llevarse a cabo en el sistema nervioso de estos animales.

Debido a la proyección retinotópica de la retina a varias zonas cerebrales, el modelo está formado por un conjunto de matrices , cada matriz procesa una propiedad especifica del estímulo y varios matrices pueden estar localizadas en una misma región cerebral.

En el tectum se encuentran las matrices que procesan el reconocimiento de la presa o predador, así como su posición y alejamiento. La información que se genera en estas matrices da lugar a los siguientes tipos de comandos:

1) De orientación. Cuando hay una presa en el campo visual pero fuera del campo binocular (b-).

2) De ataque. Cuando la presa se encuentra a una distancia considerablemente cerca (d-) y dentro del campo binocular (b+).

3) De acercamiento. Cuando la presa se encuentra fuera de la zona de ataque (d+) y dentro del campo binicular (b+).

4) Huida. Esta respuesta está controlada por la acción conjunta de células tectales y pretectales.

El tálamo tiene un grupo de matrices que procesan las siguientes características de la imagen visual:

1) Predadores y su posición.

2) Objetos fijos y su posición.

3) Profundidad de objetos fijos y predador.

4) Color del medio ambiente.

La combinación de estas matrices generan el valor de las funciones propuestas anteriormente, cuando se describió los diferentes cursos de acción que podía tomar el anfibio, por ejemplo:

1) De acercamiento a la barrera cuando la distancia entre la presa y el obstáculo esté dentro del rango de ataque del anfibio (d=o-). De saltar la barrera si la distancia entre la presa y el obstáculo esta fuera del alcance de la rana y además la altura de la barrera es salvable (d=o+,h-).

2) De rodear la barrera cuando la presa esté alejada del obstáculo y la rana sea incapaz de saltarla (d=o+,h+).

3) De brincar una zanja cuando el ancho de la misma no sea muy grande (w-).

4) De cruzarla caminando si es ancha y poco profunda (w+,d-).

5) De dar por fructrado el ataque cuando la zanja es ancha y además profunda (w+,d+).

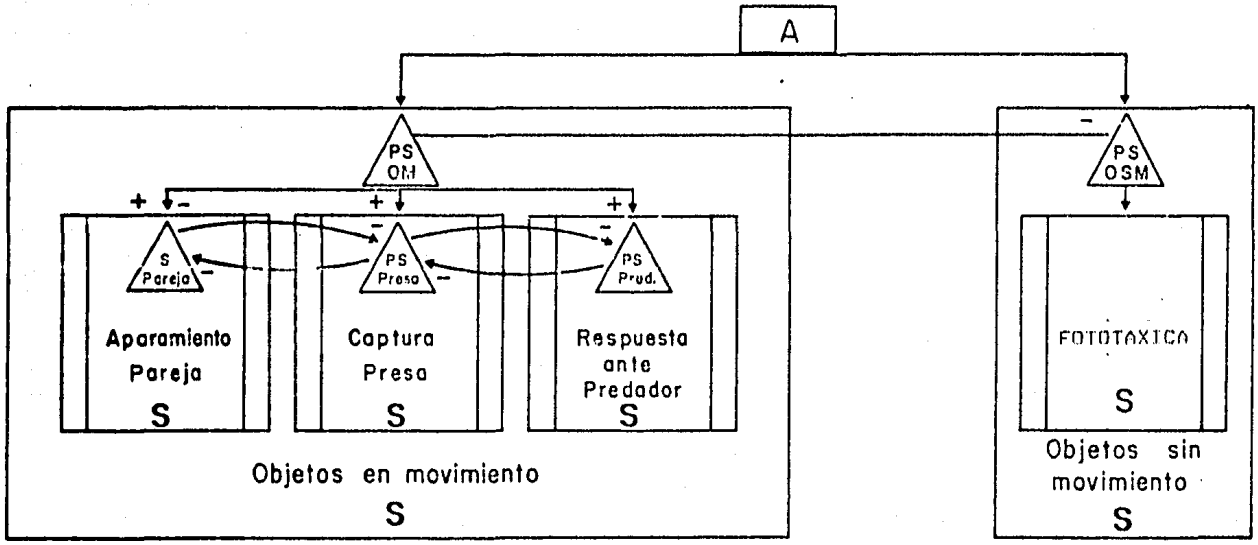
Análogamente a cómo se ha esquematizado en el etograma de estados, el modelo implementado en la computadora asimila la información proporcionada por el usuario; la cual describe las caracteristicas del medio ambiente, y la procesa para generar 'matrices de información' que al mismo tiempo son pasadas como datos a otras zonas cerebrales, las cuales realizan una evaluación de estos y, según el esquema del que se trate, generan una respuesta motora o procesan nuevamente la información para interactuar con otras zonas cerebrales.

Cabe mencionar que una conducta simple puede estar constituida por un esquema motor simple o por la coordinación de varios esquemas motores unidos, por ejemplo, cuando se activa el esquema motor de atravezar una zanja, se activan también los esquemas de acercarse a la zanja, bajar la zanja, atravezarla, y finalmente treparla.

Figura 1.- Esquemas principales que regulan la conducta de las ranas y sapos ante estímulos en movimiento y estímulos fijos. Si existe un estímulo en el campo visual del animal, tres esquemas perceptuales empiezan a competir entre si, dependiendo si el estímulo es una presa, un predador, o un compañero sexual. El esquema perceptual superior modula la competencia dependiendo del estado motivacional del animal y de su experiencia. En caso de que no haya objetos en movimiento, se activa el esquema de búsqueda de un sitio adecuado.

Figura 2.- Esquema de presencia de predador. Al activarse este esquema existen dos posibles esquemas por activar: predador aéreo (aereal-predator) o predador terrestre (terrestrial-predator); para el caso aéreo si la distancia rana-predador es pequeña, entonces se activa el esquema agachar (duck), en caso contrario existen tres posibles esquemas activar: si no existe un lugar en donde esconderse se activa el esquema retire (go-away) de lo contrario analiza si el lugar es un charco entonces se activa sumerge-en-charco (plunge-pond), pero si el lugar es un hueco se activa el esquema esconde-en-hueco (hide-hole). Pero si el esquema que se activo primero es el terrestre existen dos posibles esquemas por activar ya sea que se trate de una culebra o de otro animal terrestre. Si se trata de una vibora que se encuentre cerca de la rana(d-), entonces el animal se infla, toma una postura rigida (SLP)y orienta su cabeza hacia la culebra inclinando su cuerpo (TILT); estas acciones definen la respuesta ante predador. En el caso de que la culebra se encuentre lejos (d+), entonces el el esquema de búsqueda de lugar donde esconderse es activado (hiding place). Si el predador es diferente a una culebra, la única diferencia es que el animal sólo se inclina (TILT) cuando el predador esta cerca, en cambio si el predador se encuentra lejos, entonces el esquema ante predador (hiding place) es activado como con los otros predadores.

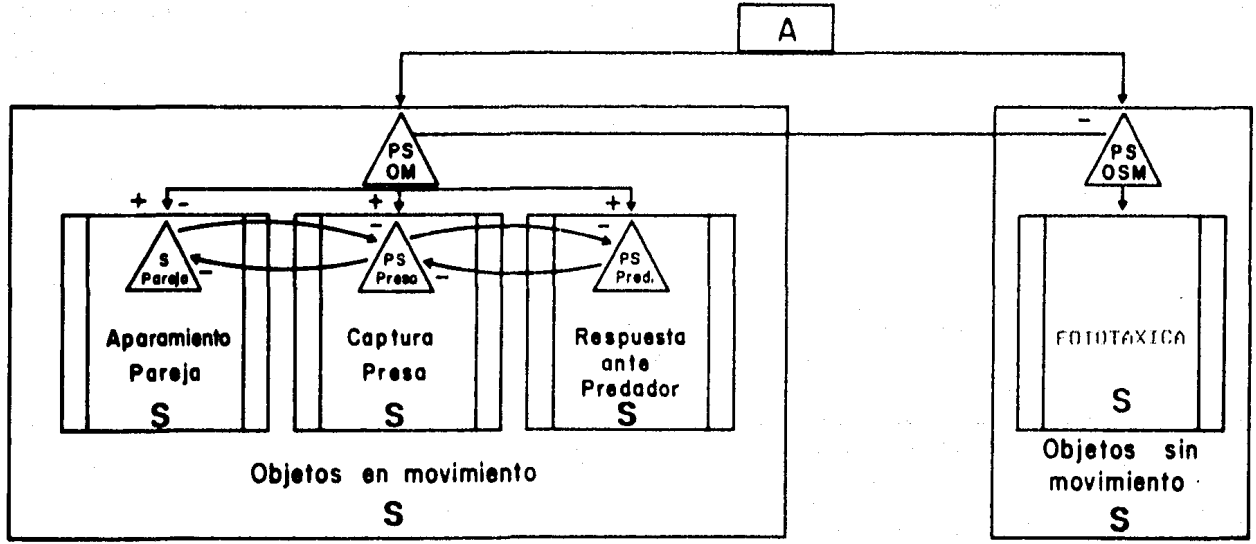
Figura 3.- Esquema de la adquisición de la presa. Al activarse este esquema el animal se orienta hacia la presa y se inicia la competencia entre tres posibles esquemas perceptuales: presa sin obstáculos, presa detrás de una barrera y presa detrás de una zanja. En caso de que se active el esquema de presa sin obstáculos, se pueden activar el esquema de ataque a la presa si la distancia rana y presa es pequeña, o el de caminar hacia hacia la presa, en caso que la distancia sea grande. En el caso de que la presa se encuentre detrás de un obstáculo, dependiendo de las condiciones métricas del mismo, el animal puede activar diferentes esquemas para evadirlo y alcanzar el estado meta.



ESQUEMA PRINCIPAL (S)

ESQUEMA PERCEPTUAL (PS)

FIGURA 1



ESQUEMA PRINCIPAL (S)

ESQUEMA PERCEPTUAL (PS)

FIGURA 1

ESQUEMA MAESTRO

RESPUESTA ANTE PREDADOR

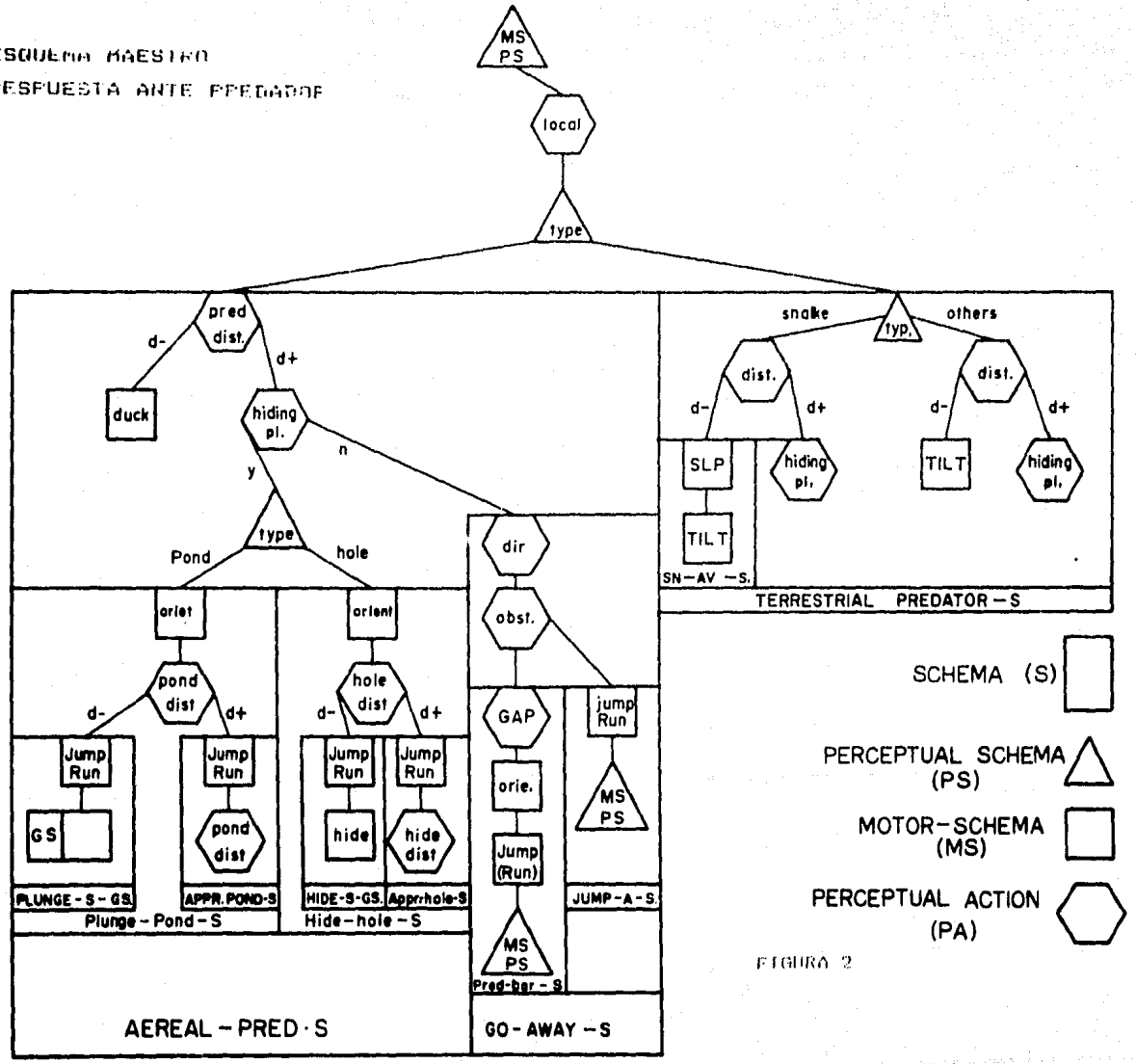


FIGURA 2

ESQUEMA MAESTRO
RESPUESTA ANTE PREDATOR

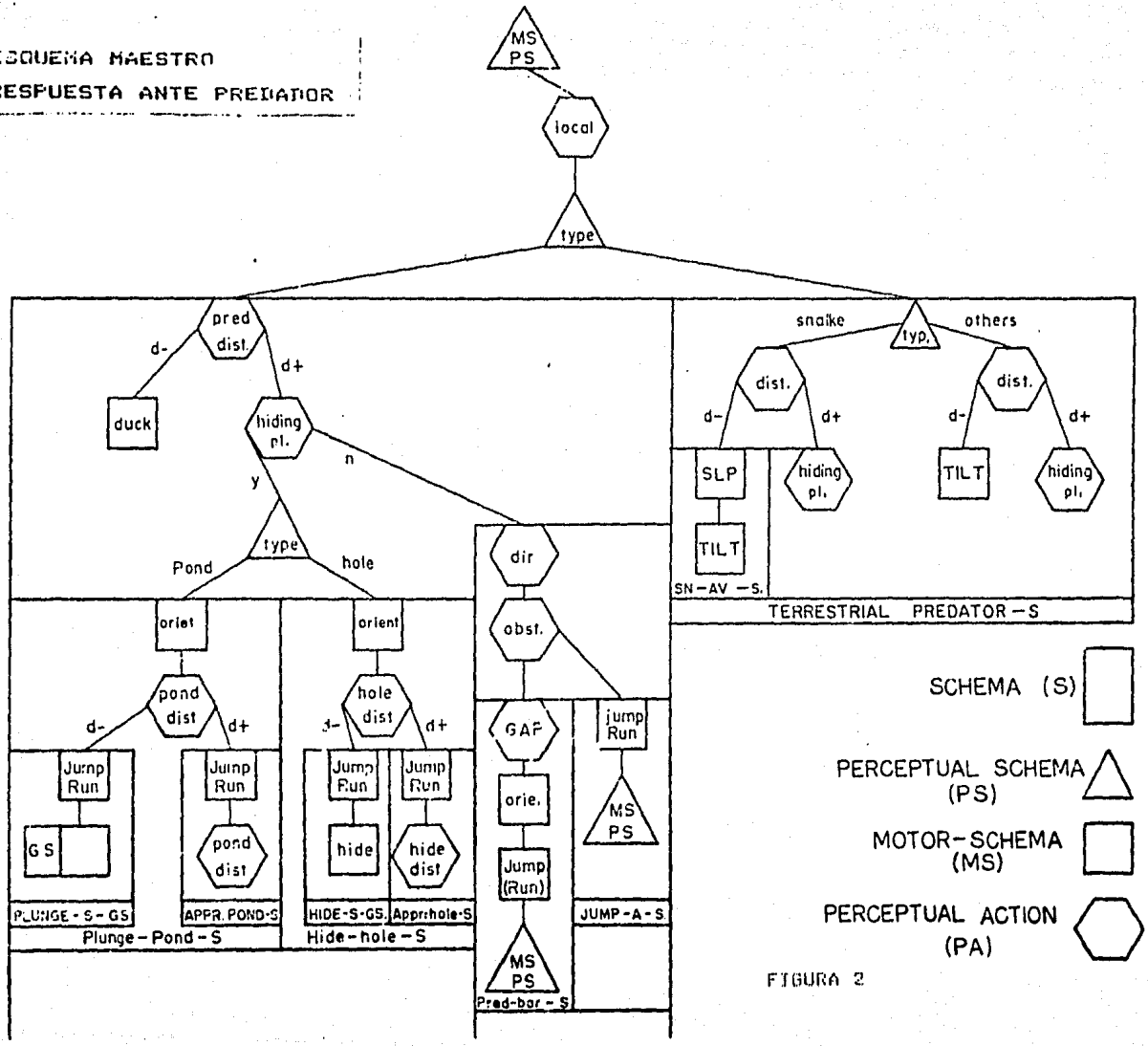


FIGURA 2

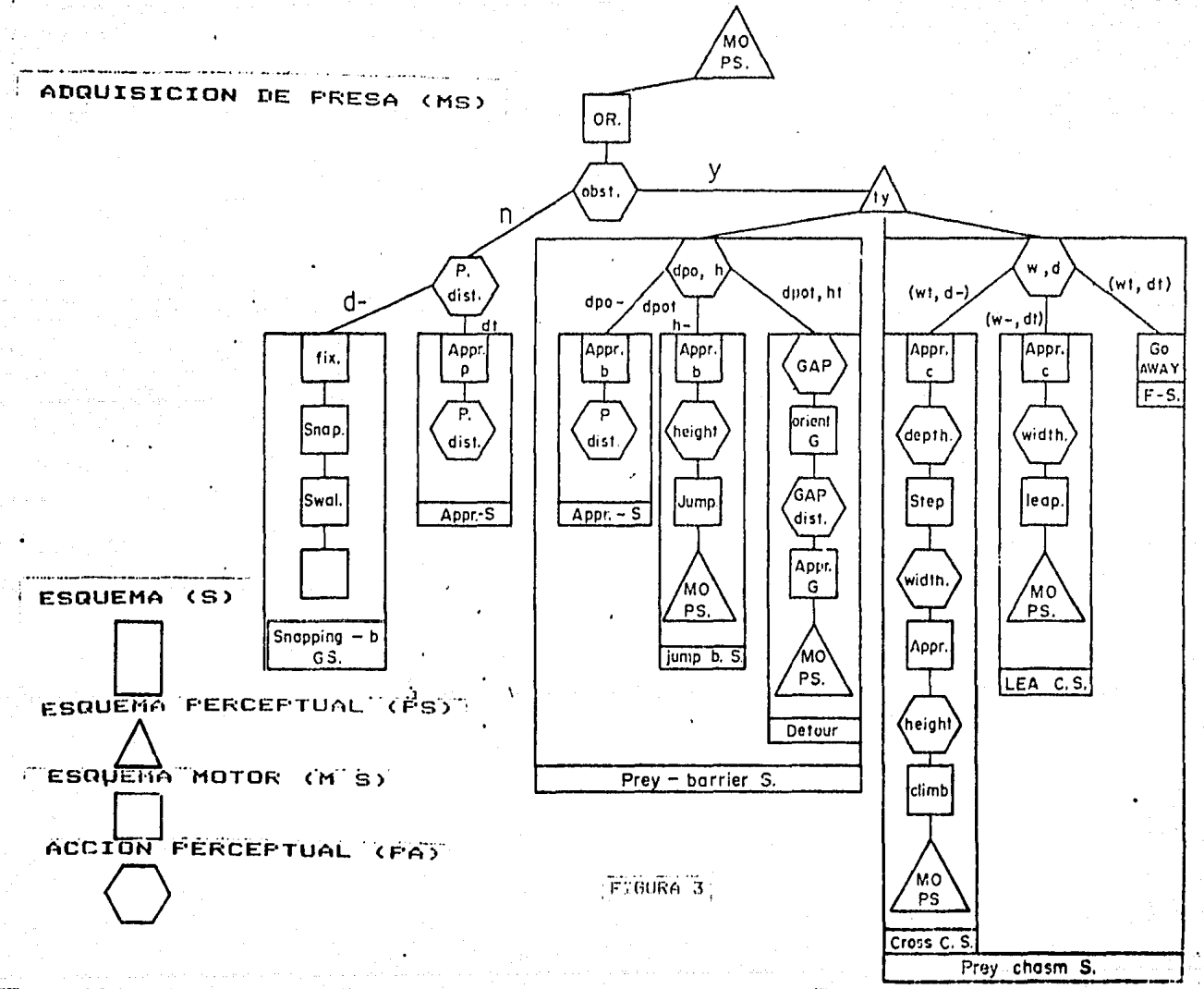

ADQUISICION DE PRESA (MS)
MO PS.
OR.
obst.
y
n
ly
P. dist.
d-
dt
fix.
Snap.
Swal.
Snapping – b GS.
Appr. p
P. dist.
Appr.-S
dpo, h
dpo-
dpot
h-
dpot, ht
Appr. b
P dist.
Appr. – S
Appr. b
height
Jump
MO PS.
jump b. S.
GAP
orient G
GAP dist.
Appr. G
MO PS.
Detour
Prey – barrier S.
w, d
(wt, d-)
(w-, dt)
(wt, dt)
Appr. c
depth.
Step
width.
Appr.
height
climb
MO PS
Cross C. S.
Appr. c
width.
leap.
MO PS.
LEA C. S.
Go AWAY
F-S.
Prey chasm S.
ESQUEMA (S)
ESQUEMA PERCEPTUAL (PS)
ESQUEMA MOTOR (MS)
ACCION PERCEPTUAL (PA)

FIGURA 3

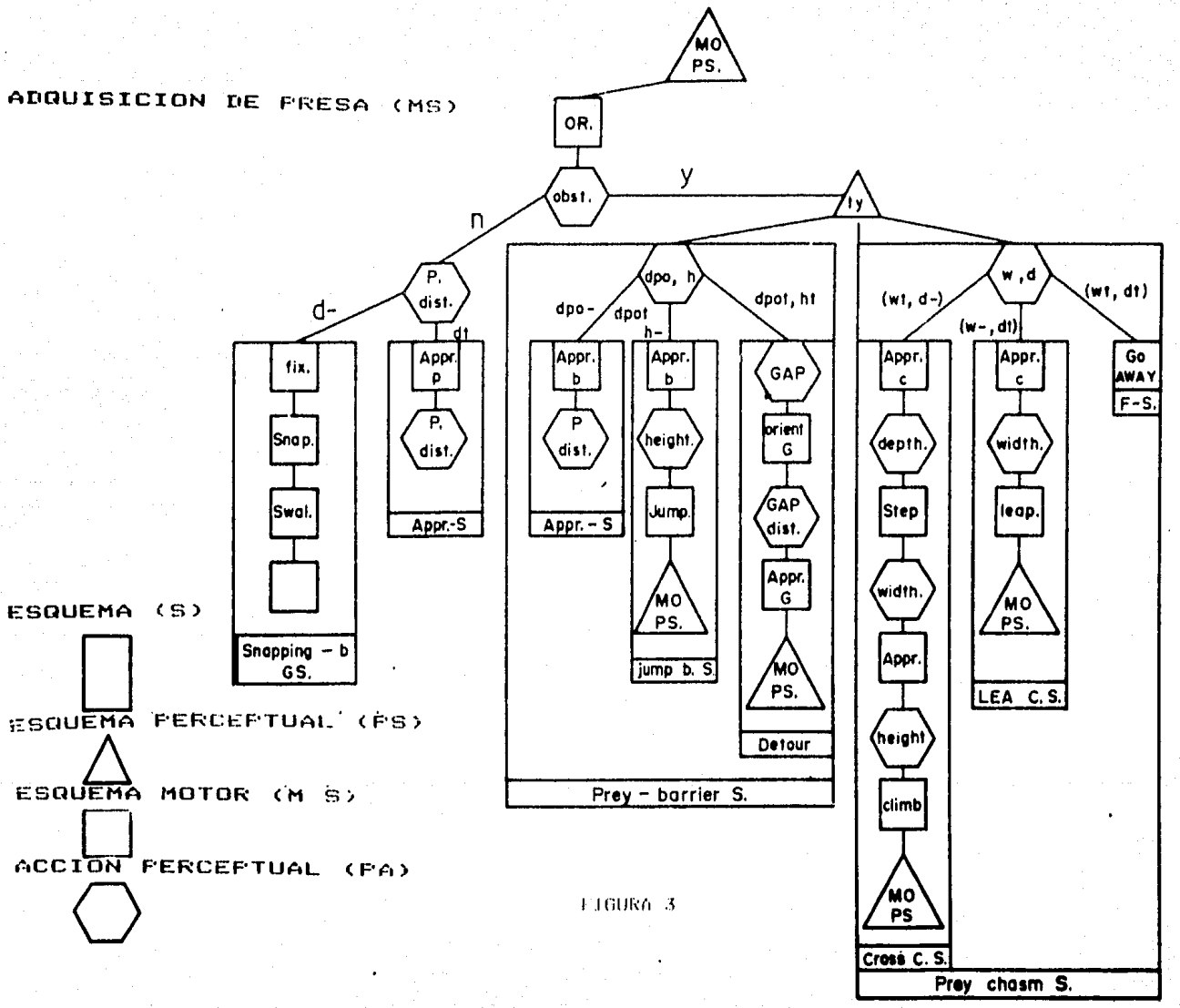

ADQUISICION DE PRESA (MS)
MO PS.
OR.
obst.
y
n
ly
P. dist.
d-
dt
fix.
Snap.
Swal.
Snapping - b GS.
Appr. p
P. dist.
Appr.-S
dpo, h
dpo-
dpot
h-
dpot, ht
Appr. b
P dist.
Appr. - S
Appr. b
height.
Jump.
MO PS.
jump b. S.
GAP
orient G
GAP dist.
Appr. G
MO PS.
Detour
Prey - barrier S.
w, d
(wt, d-)
(w-, dt)
(wt, dt)
Appr. c
depth.
Step
width.
Appr.
height
climb
MO PS
Crosé C. S.
Appr. c
width.
leap.
MO PS.
LEA C. S.
Go AWAY
F-S.
Prey chasm S.
ESQUEMA (S)
ESQUEMA PERCEPTUAL (PS)
ESQUEMA MOTOR (M S)
ACCION PERCEPTUAL (PA)

FIGURA 3

2.4 BIBLIOGRAFIA

1. Lara, R., Carmona, M., Cruz, A, Daza, F. . A Global Model of Visuomotor Coordination in Toads. Journal of Theoretical Biology. En prensa. 1984

2. Lara, R., Cruz. A. .Modelo Global de Coordinación Visuomotora en Animales y Robots. Revista 101 editada por la Fundación Arturo Rosenblueth. Mayo de 1984.

3. Vision in Frogs. W.R.A. Muntz. Perception: Mechanisms and Models. Freeman Co. San Francisco. 1964.

CAPITULO 3:

SIMULACION EN LISP

3.1 INTRODUCCION

La coordinación visuomotora en animales y robots es uno de los tópicos que mayor interes ha adquirido recientemente tanto en disciplinas relacionadas con la conducta del animal, como la Etología o la Psicología del conocimiento; o en estudios de Inteligencia Artificial y Robótica. La interacción entre estas disciplinas apenas se ha iniciado, pero en algunos casos ya se han dado resultados interesantes, como es el caso en el estudio del reconocimiento de formas en máquinas y animales. Esta interacción es fructífera por el hecho de que estudios sobre el procesamiento de información en animales generan ideas para implementarlos en sistemas computacionales; mientras que el desarrollo de algoritmos computacionales para la realización de ciertas funciones inteligentes pueden sugerir nuevas hipótesis para el estudio del procesamiento de la información en los seres vivos.

Diversas teorías han surgido tratando de explicar las estructuras de información que controlan la respuesta motora en animales y su posible aplicación al diseño de computadoras inteligentes. Piaget (1) ha postulado que la inteligencia humana esta constituida por estructuras de información, que el llama esquemas, de tipo sensorimotor o representativos que se van desarrollando en el niño hasta alcanzar un estado de madurez que le permiten confrontar adecuadamente su universo. Arbib (2), por otro lado, ha intentado relacionar los esquemas piagetianos con algunas funciones del cerebro; mientras que Minsky (3 y 4) ha desarrollado una teoría sobre las estructuras de información que deben tener las computadoras con posibilidades de realizar funciones inteligentes. Estas hipótesis, sin embargo no han sido lo suficientemente específicas como para poder explicar adecuadamente el procesamiento de información del sistema nervioso para el control de la conducta animal o para la generación de algoritmos que puedan controlar la conducta de un robot (5).

SIMULACION EN LISP

En el capítulo 2 postulamos una teoría general de las estructuras de información que podrían utilizar animales y robots para poder definir su respuesta motora ante una circunstancia específica. A estas estructuras de información las hemos denominado ESQUEMAS (6). Asimismo, la teoría de esquemas nos permite definir los algoritmos necesarios para controlar la respuesta motora de un robot específico.

3.2 SIMULACION EN LISP

La simulación en computadora del sistema visuomotor en anfibios (*), fué implementada en LISP; debido a que existen las siguientes ventajas:

1) LISP (7) es un lenguaje para procesamiento simbólico de datos. La importancia que tiene este último, es que al programar en LISP no se pierde la visión de la simulación requerida, es decir los datos son procesados como tales, sin sufrir ninguna transformación debida a la implementación en el lenguaje.

2) Las FUNCIONES en LISP se asocian perfectamente al concepto de ESQUEMA. Así como los esquemas pueden tener una estructura jerárquica, para lograr un objetivo, también LISP permite que las funciones tengan una organización jerárquica para lograrlo. Con esto queda incluido también el concepto de cooperatividad de esquemas y de funciones para alcanzar un fin. Otra característica de los esquemas, es la competitividad entre estos, las funciones en LISP pueden tener comportamiento heterárquico.

3) LISP es un lenguaje altamente interactivo, lo que permite verificar y corregir el buen funcionamiento del algoritmo programado.

A continuación se explica el diseño general, y en las figuras 1,2,3 y 4 se muestra el diseño gráfico del sistema, en donde hacemos uso de acopladores de datos (O--➔) y acopladores de control o banderas (●--➔).

SIMULA (Fig. 1) es la función que inicia la simulación, se auxilia directamente de las tres siguientes

*** En el apéndice A se presenta una breve introducción al lenguaje Lisp para la mejor comprensión del sistema implementado, el cual es mostrado en el apéndice B.**

funciones:

1) UNIVERSO, realiza la petición e inspección de los datos necesarios para efectuar la simulación.

2) ORIENTA, es la función central, la cual realiza el analisis conductual de la rana con su medio externo.

3) GRAFICA, realiza la gráfica en el video, de la simulación experimentada.

A su vez UNIVERSO, ORIENTA y GRAFICA hacen uso de funciones subordinadas.

UNIVERSO (Fig. 2) usa las siguientes funciones:

a) EJE-X-Y define el espacio de movimiento de la rana.

b) RANA-DATOS solicita las caracteristicas de la rana: posición, capacidad de salto de longitud (para evadir zanjas), capacidad de salto de altura (para evadir barreras) etc. A su vez esta función usa a la funciones EXAM-DATO y EXAM-COORDXY (nótese que éstas son RECURSIVAS) para examinar que los datos sean congruentes, por ejemplo: que la posición de la rana no esté fuera del espacio de simulación definido por EJE-X-Y. Elabora la lista de propiedades de la rana.

c) PRESA-DATOS solicita la posición de la presa, verifica que ésta sea válida llamando a EXAM-COORDXY también. Elabora la lista de propiedades de la presa.

d) OBSTACULOS-DATOS pregunta si existen obstáculos en caso afirmativo llama a la función DESC-OBST la que define la lista de propiedades de los obstáculos, se auxilia de las funciones recursivas: EXAM-TIPO, POSICION y ALT-PROF. EXAM-TIPO se encarga de obtener el tipo de obstáculo: barrera o zanja. POSICION obtiene las coordenadas de los puntos que indican la posición del obstáculo, si el tipo de obstáculo definido es una zanja entonces se requieren cuatro puntos de coordenadas, y para una barrera sólo dos, verificando a su vez que cada par de coordenadas de cada punto sean congruentes. ALT-PROF deduce la dimensión del obstáculo, dependiendo del tipo de obstáculo que se defina, por ejemplo si se trata de una barrera esté podrá ser: alta o baja, o bien una zanja podrá ser profunda o poco-profunda etc.

ORIENTA (Fig. 3) es la función central, la cual se encarga de procesar la información obtenida por UNIVERSO, de acuerdo a el modelo propuesto. Hace uso directo de las siguientes funciones:

a) ANALIZA-TRAYECTO obtiene la lista de obstáculos y puntos que interfieren (lista-obst-inter) en la trayectoria rana-presa incluyendo el punto y obstáculo de interferencia más cercano a la rana, auxiliandose de las funciones: MODULA-PUNTOS, INTER-LINEAS y OBJ-PTO-CERCA. MODULA-PUNTOS genera una lista de puntos intermedios (lp) entre dos puntos dados (pto-1 y pto-2); INTER-LINEAS determina si hay un punto común a dos listas de puntos (lp1 y lp2), es decir obtiene el punto de intersección (p-i); en caso de que exista por lo menos un p-i se forma una lista de estos (l-pi), los cuales son la entrada a la función OBJ-PTO-CERCA para que ésta encuentre el punto y el obstáculo de interferencia (obj-pto) más cercano a la posición de la rana.

b) ATACA-PRESA, se activa cuando la trayectoria rana-presa está libre de obstáculos, es decir no existe p-i alguno.

c) La función ANALIZA-OBSTACULO se activa cuando existe por lo menos un obstáculo en la trayectoria rana-presa, hace uso a su vez de: ANALIZA-BARRERA y de ANALIZA-ZANJA, dependiendo del tipo de obstáculo del que se trate.

A continuación se muestra, las conductas posibles, cuando el obstáculo que se interpone entre la rana y la presa es una barrera.

a) Si la rana, puede atacar a su presa, introduciendo su lengua por la barrera, entonces la rana se acerca (ACERCAR).

b) Si la barrera es baja, entonces deberá saltarla (SALTAR), de lo contrario se activa la función RODEAR, es decir la rana rodea el obstáculo.

c) Si en la trayectoria rana-presa, existe un arreglo de barreras en forma de cuadrado, pero con una entrada, entonces la rana penetra dentro y olvida su presa.

Si el obstáculo es una zanja, entonces:

d) Si la zanja es poco profunda, entonces la rana bajará y cruzará la zanja (CRUZAR).

e) Si la zanja es profunda, entonces la rana tratará de brincarla (BRINCAR), pero si tampoco la puede brincar, entonces la rana olvidará su presa.

Se observa que las funciones SALTAR-BARRERA, RODEAR, BRINCAR-ZANJA y CRUZAR hacen a su vez uso de la función iniciadora ORIENTA, es decir existe RETROALIMENTACION DE INFORMACION, la salida de este CICLO se puede presentar de dos maneras: que la rana ataque a la presa o bien que se retire.

GRAFICA (Fig. 4) es la función que coordina la graficación del modelo en la pantalla de la computadora. Utiliza a TRAZA directamente para hacer la graficación de las posición inicial de la rana y la de la presa; además utiliza a GRAFICA-OBSTACULOS y a GRAFICA-TRAYECTO las que a su vez se auxilian de MODULA-PUNTOS y de TRAZA para graficar cada uno de los puntos que comforman los obstáculos así como el trayecto que realizó la rana durante la simulación.

A continuación se muestran 4 ejemplos procesados con este programa simulador. Cada uno de ellos plantea una situación diferente para el animal.

```
(simula)
-----------------------------
---DEFINICION-DE-ESPACIO---
-----------------------------

(EL ESPACIO DE MOVIMIENTO EN EJE-X ES DESDE 0 A 30)
(EL ESPACIO DE MOVIMIENTO EN EJE-Y ES DESDE 0 A 30)
(SI DESEAS MODIFICARLO LO PUEDES HACER [S/N]) N

-----------------------------
---DEFINICION-DE-LA-RANA---
-----------------------------

(POSICION DE LA RANA (X Y)) (5 2)

(LONGITUD DE SALTO DE ALTURA) 2

(LONGITUD DE SALTO DE LONGITUD) 2

(LONGITUD DE DESCENSO) 2

(LONGITUD DE LENGUA) 1


-----------------------------
---DEFINICION-DE-LA-PRESA--
-----------------------------

(POSICION DE LA PRESA (X Y)) (5 9)

-----------------------------
--DEFINICION-DE-OBSTACULOS-
-----------------------------

(NUMERO DE OBSTACULOS) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((2 6) (10 6))
(DAME LA ALTURA) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((1 4) (4 4))
(DAME LA ALTURA) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((7 4) (11 4))
(DAME LA ALTURA) 3

((DESCRIPCION (BARRERA ((2 6) (10 6)) ALTA) (BARRERA ((1 4) (4 4)) ALTA) (
BARRERA ((7 4) (11 4)) ALTA)) (NUMERO . 3))



(LA-RANA-RODEA-LA-BARRERA-POR (4 4))

(LA-RANA-SE-ACERCA-A-LA-BARRERA-EN (2 6))

(LA-RANA-ATACA-A-SU-PRESA-EN (5 9))
(FIN-DE-SIMULACION)
NIL
```

```
(SIMULA)

-------------------------------------
---DEFINICION-DE-ESPACIO---
-------------------------------------

(EL ESPACIO DE MOVIMIENTO EN EJE-X ES DESDE [illegible])
(EL ESPACIO DE MOVIMIENTO EN EJE-Y ES DESDE [illegible])
(SI DESEAS MODIFICARLO [illegible] HACER [illegible])

-------------------------------------
---DEFINICION-DE-LA-RANA---
-------------------------------------

(POSICION DE LA RANA [illegible])

(LONGITUD DE SALTO DE ALTURA) 1

(LONGITUD DE SALTO DE LONGITUD) 2

(LONGITUD DE DESCENSO) 7

(LONGITUD DE LENGUA) 1


-------------------------------------
---DEFINICION-DE-LA-PRESA---
-------------------------------------

(POSICION DE LA PRESA (X Y)) (10 12)

-------------------------------------
---DEFINICION-DE-OBSTACULOS---
-------------------------------------

(NUMERO DE OBSTACULOS) 3
(TIPO DE OBSTACULO (BARRERA/ZANJA) O (B/Z)) Z
(COORDENADAS ((X1 Y1) [illegible] (X4 Y4))) ((6 7) (6 10) (14 10) (14 7))
(DAME LA PROFUNDIDAD) [illegible]
(TIPO DE OBSTACULO (BARRERA/ZANJA) O (B/Z)) B
(COORDENADAS ((X1 Y1) (X2 Y2))) ((4 5) (9 5))
(DAME LA ALTURA) 4
(TIPO DE OBSTACULO (BARRERA/ZANJA) O (B/Z)) B
(COORDENADAS ((X1 Y1) (X2 Y2))) ((11 5) (14 5))
(DAME LA ALTURA) 4

((DESCRIPCION (ZANJA ((6 7) (6 10) (14 10) (14 7)) NO-PROFUNDA) (BARRERA ((
4 5) (9 5)) ALTA) (BARRERA ((11 5) (14 5)) ALTA)) (NUMERO . 3))


(LA-RANA-RODEA-LA-BARRERA-POR (4 5))
(LA-RANA-SE-ACERCA-A-LA-ZANJA-EN (6 [illegible]))
(LA-RANA-CRUZA-LA-ZANJA-DE (6 [illegible]) A (7 10))
(LA-RANA-ATACA-A-SU-PRESA-EN (10 12))
(FIN-DE-SIMULACION)
NIL
```

```
P.(simula)
-----------------------------
---DEFINICION-DE-ESPACIO---
-----------------------------

(EL ESPACIO DE MOVIMIENTO EN EJE-X ES DESDE 0 A 30)
(EL ESPACIO DE MOVIMIENTO EN EJE-Y ES DESDE 0 A 30)
(SI DESEAS MODIFICARLO LO PUEDES HACER [S/N]) n

-----------------------------
---DEFINICION-DE-LA-RANA---
-----------------------------

(POSICION DE LA RANA (X Y)) (9 1)

(LONGITUD DE SALTO DE ALTURA) 4

(LONGITUD DE SALTO DE LONGITUD) 3

(LONGITUD DE DESCENSO) 3

(LONGITUD DE LENGUA) 1


-----------------------------
---DEFINICION-DE-LA-PRESA--
-----------------------------

(POSICION DE LA PRESA (X Y)) (12 9)

-----------------------------
--DEFINICION-DE-OBSTACULOS-
-----------------------------

(NUMERO DE OBSTACULOS) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((6 3) (13 3))
(DAME LA ALTURA) 2
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((5 6) (10 6))
(DAME LA ALTURA) 5
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((12 6) (15 16))
(DAME LA ALTURA) 5

((DESCRIPCION (BARRERA ((6 3) (13 3)) BAJA) (BARRERA ((5 6) (10 6)) ALTA) (
BARRERA ((12 6) (15 16)) ALTA)) (NUMERO . 3))



(LA-RANA-SE-APROXIMA-A-LA-BARRERA-EN (9 2))
(LA-RANA-SALTA-LA-BARRERA-HASTA (9 4))
(LA-RANA-RODEA-LA-BARRERA-POR (10 6))
(LA-RANA-SE-ACERCA-A-LA-BARRERA-EN (12 9))
(LA-RANA-ATACA-A-SU-PRESA-EN (12 9))
(FIN-DE-SIMULACION)
NIL
```

```
> (simula)

------------------------------
---DEFINICION-DE-ESPACIO---
------------------------------

(EL ESPACIO DE MOVIMIENTO EN EJE-X ES DESDE 0 A 30)
(EL ESPACIO DE MOVIMIENTO EN EJE-Y ES DESDE 0 A 30)
(SI DESEAS MODIFICARLO LO PUEDES HACER [S/N]) n

------------------------------
---DEFINICION-DE-LA-RANA---
------------------------------

(POSICION DE LA RANA (X Y)) (4 3)

(LONGITUD DE SALTO DE ALTURA) 4

(LONGITUD DE SALTO DE LONGITUD) 1

(LONGITUD DE DESCENSO) 2

(LONGITUD DE LENGUA) 1


------------------------------
---DEFINICION-DE-LA-PRESA--
- ------------------------------

(POSICION DE LA PRESA (X Y)) (10 12)

------------------------------
--DEFINICION-DE-OBSTACULOS-
------------------------------

(NUMERO DE OBSTACULOS) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((4 5) (9 5) )
(DAME LA ALTURA) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) z
(COORDENADAS ((X1 Y1) A (X4 Y4))) ((6 7) (6 10) (14 10) (14 7))
(DAME LA PROFUNDIDAD) 3
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((11 5) (14 5))
(DAME LA ALTURA) 5

((DESCRIPCION (BARRERA ((4 5) (9 5)) BAJA) (ZANJA ((6 7) (6 10) (14 10) (14
7)) PROFUNDA) (BARRERA ((11 5) (14 5)) ALTA)) (NUMERO . 3))


(LA-RANA-SE-APROXIMA-A-LA-BARRERA-EN (6 4))
(LA-RANA-SALTA-LA-BARRERA-HASTA (6 6))
(LA-RANA-SE-ACERCA-A-LA-ZANJA-EN (6 7))
(LA-RANA-OLVIDA-SU-PRESA-EN (6 7))
(FIN-DE-LA-SIMULACION)
NIL
```

```
(simula)
----------------------------
---DEFINICION-DE-ESPACIO---
----------------------------

(EL ESPACIO DE MOVIMIENTO EN EJE-X ES DESDE 0 A 30)
(EL ESPACIO DE MOVIMIENTO EN EJE-Y ES DESDE 0 A 30)
(SI DESEAS MODIFICARLO LO PUEDES HACER [S/N]) n

----------------------------
---DEFINICION-DE-LA-RANA---
----------------------------

(POSICION DE LA RANA (X Y)) (10 0)

(LONGITUD DE SALTO DE ALTURA) 2

(LONGITUD DE SALTO DE LONGITUD) 2

(LONGITUD DE DESCENSO) 2

(LONGITUD DE LENGUA) 1


----------------------------
---DEFINICION-DE-LA-PRESA--
----------------------------

(POSICION DE LA PRESA (X Y)) (10 10)

----------------------------
--DEFINICION-DE-OBSTACULOS-
----------------------------

(NUMERO DE OBSTACULOS) 5
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((7 5) (7 8))
(DAME LA ALTURA) 4
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((7 8) (13 8))
(DAME LA ALTURA) 4
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((13 8) (13 5))
(DAME LA ALTURA) 4
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((7 5) (9 5))
(DAME LA ALTURA) 4
(TIPO DE OBSTACULO [BARRERA/ZANJA] O [B/Z]) b
(COORDENADAS ((X1 Y1) (X2 Y2))) ((13 5) (11 5))
(DAME LA ALTURA) 4

((DESCRIPCION (BARRERA ((7 5) (7 8)) ALTA) (BARRERA ((7 8) (13 8)) ALTA) (
BARRERA ((13 8) (13 5)) ALTA) (BARRERA ((7 5) (9 5)) ALTA) (BARRERA ((13 5)
11 5)) ALTA)) (NUMERO . 5))
```

```
(LA-RANA-SE-ACERCA-A-LA-BARRERA-EN (12 5))
(LA-RANA-SE-ACERCA-A-LA-BARRERA-EN (13 5))
(LA-RANA-SE-ACERCA-A-LA-BARRERA-EN (13 5))
(LA-RANA-PERMANECE-EN (13 5) INDEFINIDAMENTE)
(FIN-DE-SIMULACION)
NIL
```

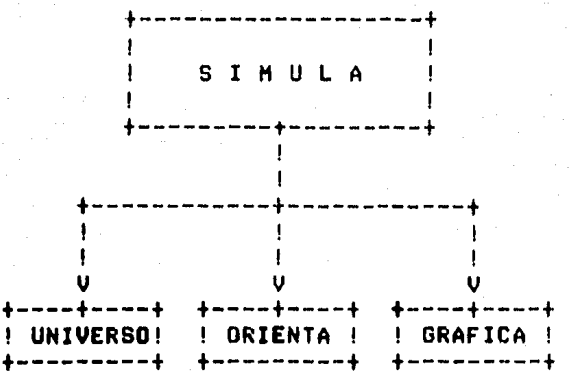


Figura 1

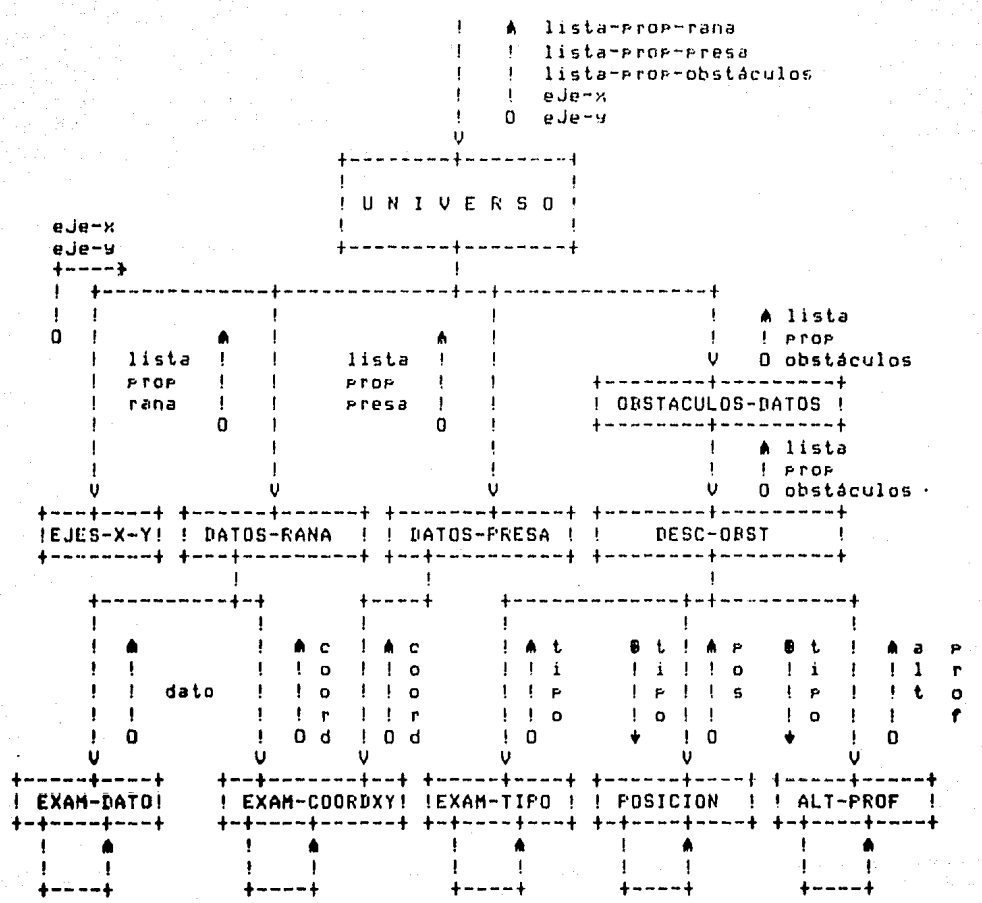

lista-prop-rana
lista-prop-presa
lista-prop-obstáculos
eje-x
eje-y
U N I V E R S O
eje-x
eje-y
lista prop rana
lista prop presa
lista prop obstáculos
OBSTACULOS-DATOS
lista prop obstáculos
EJES-X-Y
DATOS-RANA
DATOS-PRESA
DESC-OBST
dato
EXAM-DATO
EXAM-COORDXY
EXAM-TIPO
POSICION
ALT-PROF

Figura 2

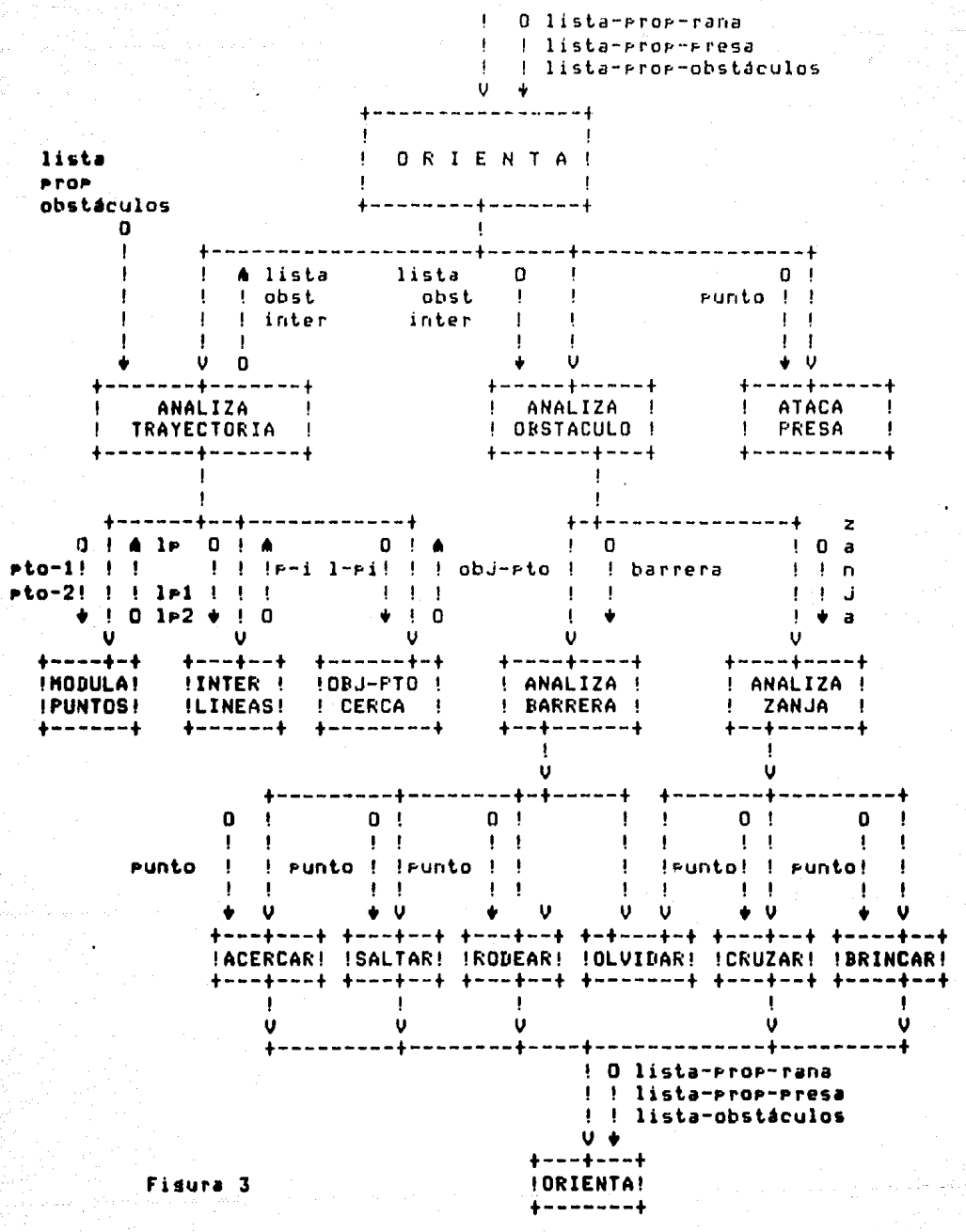

lista-prop-rana
lista-prop-presa
lista-prop-obstáculos
ORIENTA
lista prop obstáculos
lista obst inter
lista obst inter
punto
ANALIZA TRAYECTORIA
ANALIZA OBSTACULO
ATACA PRESA
lp
pto-1
pto-2
lp1
lp2
lp-i
l-pi
obj-pto
barrera
MODULA PUNTOS
INTER LINEAS
OBJ-PTO CERCA
ANALIZA BARRERA
ANALIZA ZANJA
ACERCAR
SALTAR
RODEAR
OLVIDAR
CRUZAR
BRINCAR
lista-prop-rana
lista-prop-presa
lista-obstáculos
ORIENTA

Figura 3

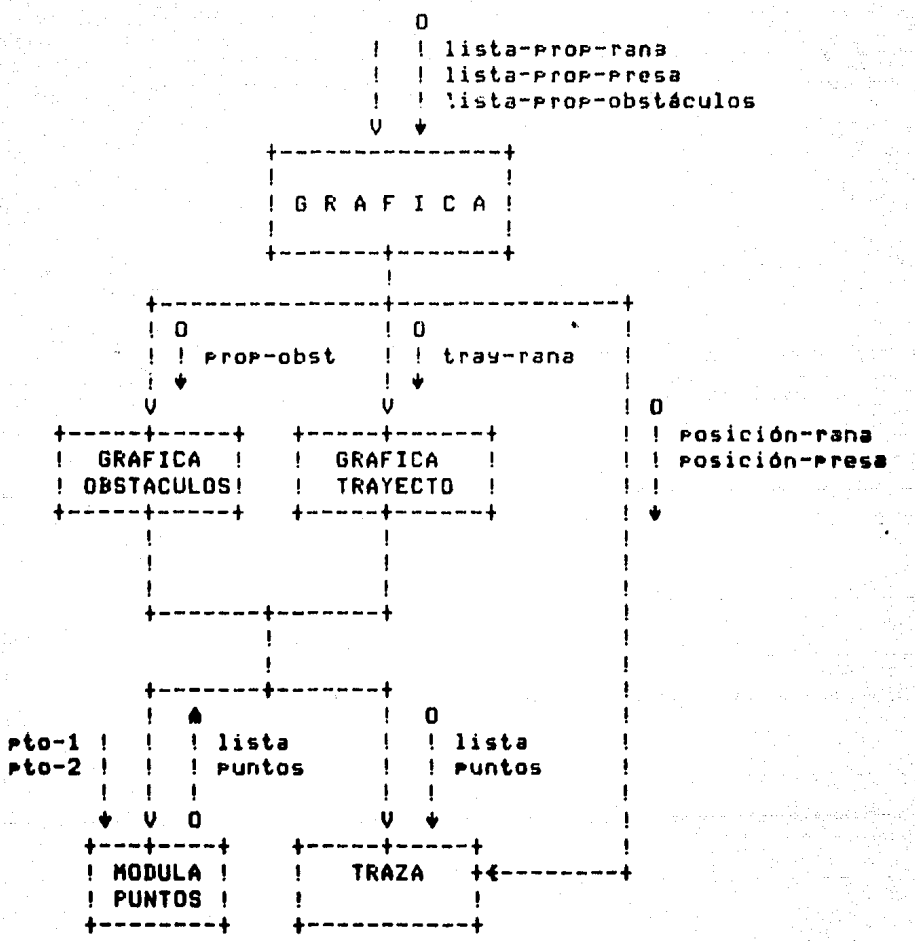

lista-prop-rana
lista-prop-presa
lista-prop-obstáculos
G R A F I C A
prop-obst
tray-rana
posición-rana
posición-presa
GRAFICA OBSTACULOS
GRAFICA TRAYECTO
pto-1
pto-2
lista puntos
lista puntos
MODULA PUNTOS
TRAZA

Figura 4

3.3 BIBLIOGRAFIA

(1) Piaget, J. Introducción a la epistemología genetica. 1975. Editorial Paidos. Buenos Aires. Argentina.

(2) Arbib, M.A. Perceptual structures and distributed motor control. En: Studies in Mathematical Biology (s. Levin, ed.). The Mathematical Association of America. 1978.

(3) Minsky, M. A framework for representing knowledge. En: The Psychology of Computer Vision. Winston, P.H. ed. New York, McGraw Hill, 1975.

(4) Minsky, M. The Society Theory of Thinking. Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, Mass. 1977.

(5) Lara, R. La Cibernética del Cerebro. Compañia Editorial Continental. En prensa.

(6) Lara, R., Carmona, M., Daza, F. y Cruz, A. A Global Model of Visuomotor Coordination in Toads. Journal of Theoretical Biology. En prensa.

(7) Cruz, A., Figueroa, Y. Apuntes de Lenguaje LISP. Facultad de Ingenieria UNAM. En prensa.

CAPITULO 4

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

4.1 INTRODUCCION

Una vez comprendido el modelo global del sistema visuomotor de anfibios, procederemos a dar una explicación de como se realizó la simulación en computadora.

Se eligieron dos computadoras para llevar a cabo la simulación, siendo estas una macrocomputadora (Burroughs 7800) y una microcomputadora (Cromemco System III); ambas computadoras fueron usadas debido a que en un principio el programa se diseñó para procesarlo en la macrocomputadora, pero la emisión de resultados en forma gráfica eran demasiado limitados en resolución, por lo que fue necesario utilizar la microcomputadora Cromemco System Three (la cual posee un sistema de graficación de alta resolución) para efectuar el despliegue gráfico.

Cabe señalar que en esta microcomputadora unicamente se diseñaron programas que fueran capaces de aceptar como entrada los resultados que generaba la macrocomputadora referentes al comportamiento del modelo, y sólo se encargaran de hacer el despliegue gráfico en una pantalla de color con alta resolución.

4.2 DESCRIPCION DE SUBMODELOS EMPLEADOS

El programa hace uso de tres submodelos muy importantes, ya que de ellos depende en gran parte el funcionamiento del mismo, estos son:

1) MODELO DE PERCEPCION DE OBJETOS FIJOS.

2) MODELO DETECTOR DE HUECOS.

3) MODELO DE PERCEPCION DE PROFUNDIDAD DE HUECOS.

los cuales se describen en las siguientes secciones.

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

4.2.1 MODELO DE PERCEPCION DE OBJETOS FIJOS

El algoritmo empleado en este modelo se detalla en los siguientes puntos:

1) Se toma a la trayectoria RANA-PRESA como el eje central del campo visual del animal.

2) Una vez definido el eje central, se recorre el espacio visual desde 85 grados a la izquierda hasta 85 grados a la derecha con respecto a éste (tomando como foco el punto de la posición de la rana).

3) Con incrementos de un grado se trazan rectas que nos definen trayectorias (referirse a las fórmulas de la siguiente página), siendo cada una de estas examinadas para obtener las coordenadas del punto de intersección con algún obstáculo determinado, ya sea que se trate de una barrera, una presa o bien una barrera limitadora del espacio de simulación (encontrando este punto por medio del método de intersección de dos rectas detallado en la página que sigue a las fórmulas anteriores); ya determinado el punto de intersección, se procede a calcular la profundidad de la trayectoria como la distancia que existe entre el punto de intersección y el punto de posición de la rana (mediante la fórmula de la distancia entre dos puntos).

4) Una vez obtenidas las profundidades de las trayectorias, las coordenadas del punto de intersección y el número del obstáculo con el que se intersecta; esta información es almacenada en la matriz de objetos fijos (matriz que representa a las células pretectales), ver figura 1. Esta matriz es fundamental para el modelo detector de huecos.

5) Existe una profundidad máxima dada por la suma de la distancia rana-presa mas 10 unidades. Cualquier profundidad de un objeto que sea mayor a la profundidad máxima, será considerada con profundidad máxima. Esto debido a que el modelo postula que el animal no se interesa por aquellos objetos que estén localizados a profundidades superiores a la de la presa más 10 unidades.

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

PARA ENCONTRAR EL PUNTO OBJETIVO HACIA EL CUAL SE TRAZAN LAS **TRAYECTORIAS** DE INCREMENTOS DE UN GRADO A PARTIR DE LA **POSICION DE LA RANA:**

```
X1 = - B + SQRT (B**2 - 4*A*C)

X2 = - B - SQRT (B**2 - 4*A*C)
```

En donde:

```
A = M**2

B = (2*M*ORD) - (2*RX) - (2*RY*M)

C = RX**2 + RY**2 - DIST**2 - (2*RY*ORD) + ORD**2

ORD = RY - (M*RX)
```

Y además:

M --> Pendiente de la trayectoria a examinar.

DIST --> Distancia a la cual se localiza el punto objetivo.

RX,RY --> Coordenadas de la rana.

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

LAS ECUACIONES QUE NOS PERMITEN ENCONTRAR EL PUNTO DE INTERSECCION ESTAN DADAS POR:

$$INTX = \frac{(B1*C0) - (B0*C1)}{(A0*B1) - (A1*B0)} \qquad INTY = \frac{(A0*C1) - (A1*C0)}{(A0*B1) - (A1*B0)}$$

En donde :

A0 = Y2 -Y1 B0 = X1 - X2 C0 = (X1*Y2) - (X2*Y1)

A1 = Y4 - Y3 B1 = X3 - X4 C1 = (X3*Y4) - (X4*Y3)

Y además:

X1,Y1 --> Coordenadas de la posición de la Rana.

X2,Y2 --> Punto objetivo hacia el cual se esta trazando una linea para el reconocimiento.

X3,Y3 --> Coordenadas de un extremo del obstáculo con el cual se desea obtener el punto de intersección.

X4,Y4 --> Coordenadas del otro extremo del obstáculo con el cual se desea obtener el punto de intersección.

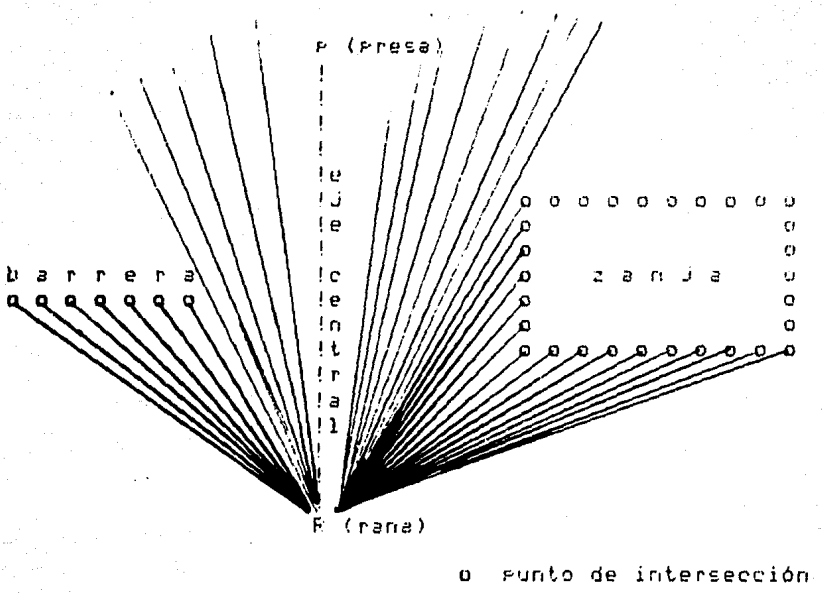


FIGURA 1.- Esta figura muestra como el animal percibe en su campo visual los objetos fijos, sus puntos de intersección, etc.

4.2.2 MODELO DETECTOR DE HUECOS

Como se ha podido apreciar en el modelo global, los huecos juegan un papel muy importante en la selección de la trayectoria a seguir, ya sea cuando se trata de evadir un obstáculo para atrapar a la presa o cuando se trata de huir de algún predador.

La matriz de huecos se genera a partir de la matriz de objetos fijos de la siguiente manera: se hace un recorrido de esta matriz desde la célula -85 hasta la célula 85 (declarada con límites de 0 a 170). Se van comparando los valores de la distancia de profundidad que tiene una célula con su vecina, si estos presentan una disparidad significante y están captando diferentes obstáculos se asume el índice de esta célula como el borde inicial de un hueco, el cual finaliza hasta que al analizar las siguientes células se encuentre una disparidad similar que originará escoger a este índice como el borde final del hueco.

Cada par de bordes, junto con el ancho del hueco (número de grados entre bordes) se guardarán en un renglón

de la matriz de huecos.

En lo que respecta al ancho del hueco, existe un ancho máximo igual a 15 grados, por lo tanto todos aquellos anchos mayores a 15 tomarán éste valor. Para determinar la profundidad del hueco se utilize el modelo de percepción de profundidad, que se describe mas adelante.

Posteriormente se procede a encontrar el índice del hueco óptimo, éste índice óptimo nos indica el número del grado de la trayectoria que es seleccionada a seguir por la rana, y es obtenido en función de la matriz de huecos de acuerdo a la siguiente función:

fr = a*d + b*w + c*f(p) (ver figura 2)

donde:

fr = frecuencia del disparo de la célula.

d = profundidad del hueco

w = anchura del hueco

f(p) = función gaussiana de la forma:

f(p) = EXP - abs (((85-i)**2) / k)

cuyo punto máximo se encuentra en la posición de la presa, considerando que el valor de fr representa la probabilidad de que ese hueco sea elegido.

i = índice de la célula en cuestión, y varia entre 85 y -85.

k = valor que nos representa el grado de la función.

a = 20 %

b = 40 %

c = 40 %

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

En el caso de que se trate de evaluar los huecos ante la presencia de un predador, la frecuencia de disparo esta dada por la función:

fr = a*d + b*w - c*f(p) + g(color)

donde:

f(p) = función cuyo punto máximo se da en la posición o trayectoria hacia el predador de la forma:

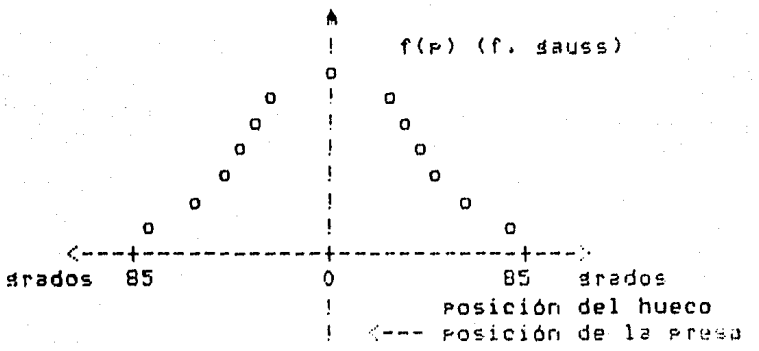
f(p) = EXP - abs (((pp-i)**2) / k)

pp = posición del predador.

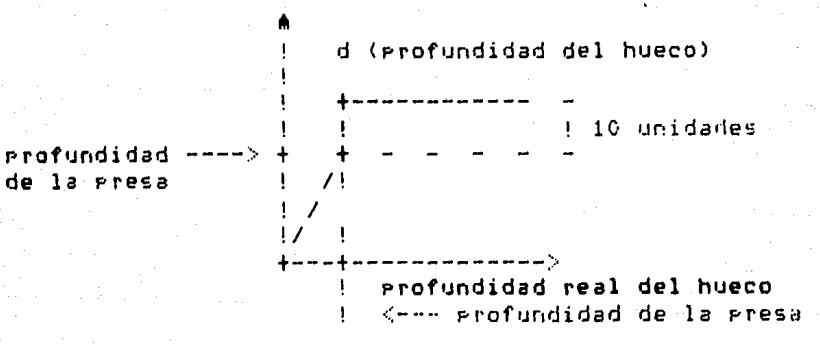
g(color) = función cuyo valor máximo se presenta cuando el color es azul.

En el caso de que se traten de evaluar los huecos en la respuesta fototáxica, la frecuencia de disparo esta dada por la función:

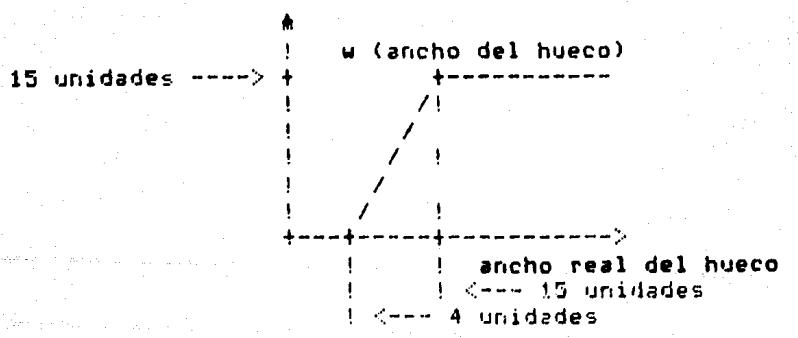
fr = a*d + b*w + g(color)



2a) Función de Gauss



2b) Función de profundidad



2c) Función de anchura.

FIGURA 2.- Se observa que la selección del hueco óptimo esta en función de tres parámetros: la

posición (del hueco respecto la presa), la profundidad y el ancho del hueco. Por lo que respecta a la posición (figura 2a) la función gaussiana tiene su valor máximo en dirección hacia la presa. En cuanto a la profundidad de la presa (figura 2b), la profundidad del hueco es obtenida del promedio de las profundidades del hueco en sus extremos; ahora consideremos una profundidad máxima dada por la distancia rana-presa mas una compensación de 10 unidades, si la profundidad del hueco es mayor a la profundidad máxima, el valor de la profundidad del hueco se hará igual a la profundidad máxima. Por último el ancho del hueco (figura 2c), se consideran con valor aquellos huecos mayores a 4 unidades, si el ancho del hueco rebasa el valor de 15 unidades, tendrá como valor éste último.

4.2.3 MODELO DE PERCEPCION DE PROFUNDIDAD DE HUECOS

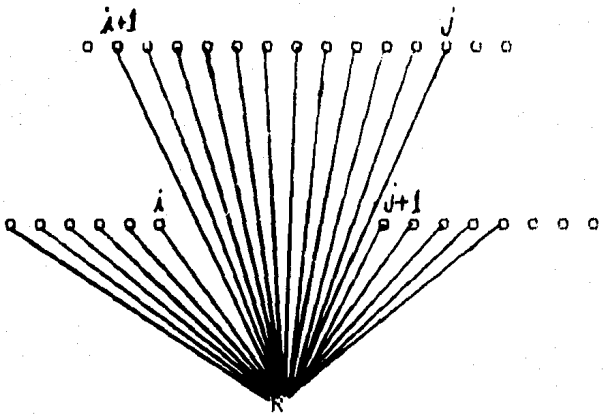
La percepción de profundidad en un hueco es un parámetro útil en la selección del hueco óptimo, la forma en que se detecta es:

1) De la matriz de huecos se obtiene tanto el índice del borde inicial como el índice del borde final del hueco.

2) Con los índices encontrados en el punto anterior se accesa la matriz de objetos fijos para obtener las profundidades de las trayectorias de los bordes (inicial y final), así como también las profundidades de las trayectorias de las células vecinas (que se encuentren dentro del hueco).

3) Se calcula la profundidad izquierda del hueco como la diferencia de profundidades del borde inicial con su célula vecina, el mismo procedimiento se sigue para obtener la profundidad derecha.

4) Por último se suman las profundidades derecha e izquierda y se obtiene su promedio, el cual nos da un valor representativo de la profundidad del hueco.

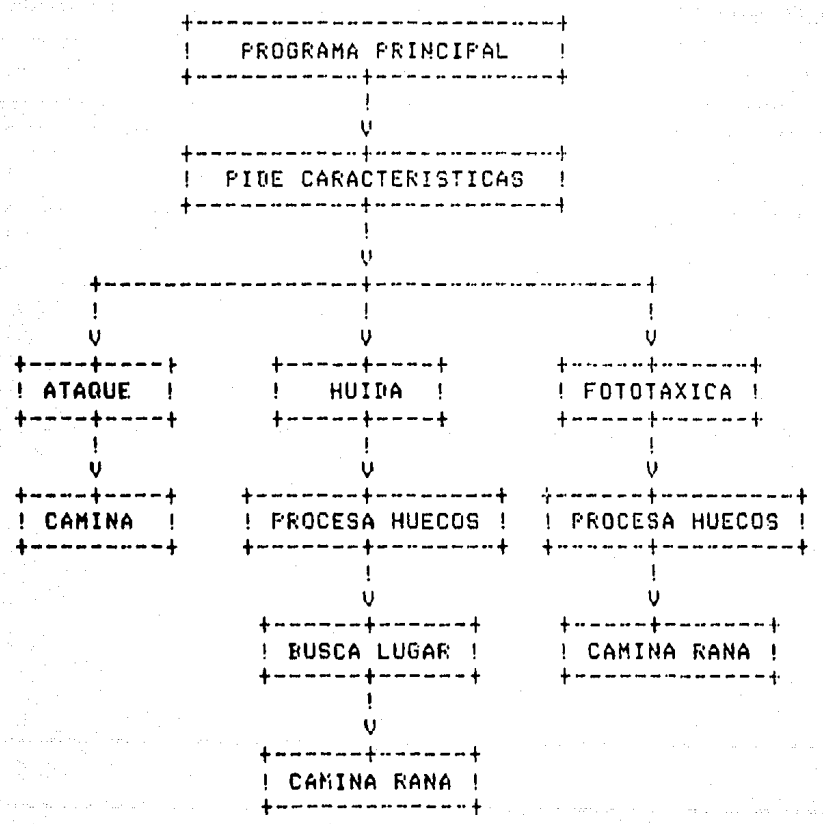


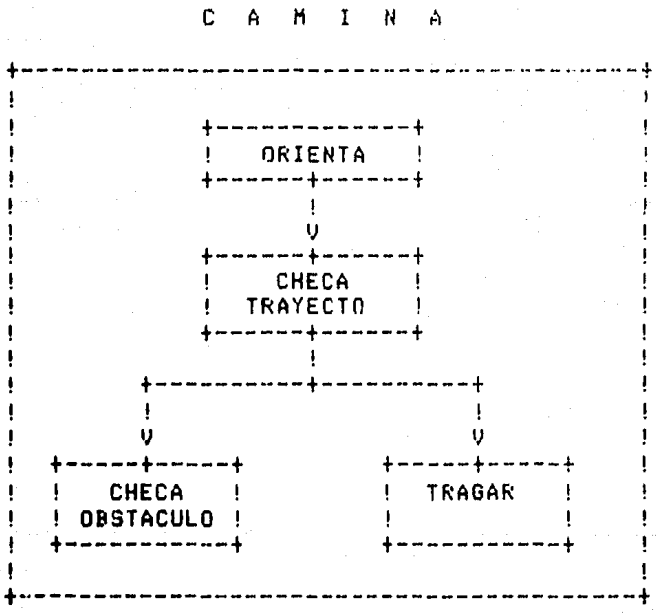
$$\text{prof-hueco} = \frac{(P_i - P_{i+1}) + (P_j - P_{j+1})}{2}$$

FIGURA 3.-En esta figura se puede observar cómo es calculada la profundidad de un hueco, en base a los valores de la distancia de profundidad de las células del borde y sus vecinas.

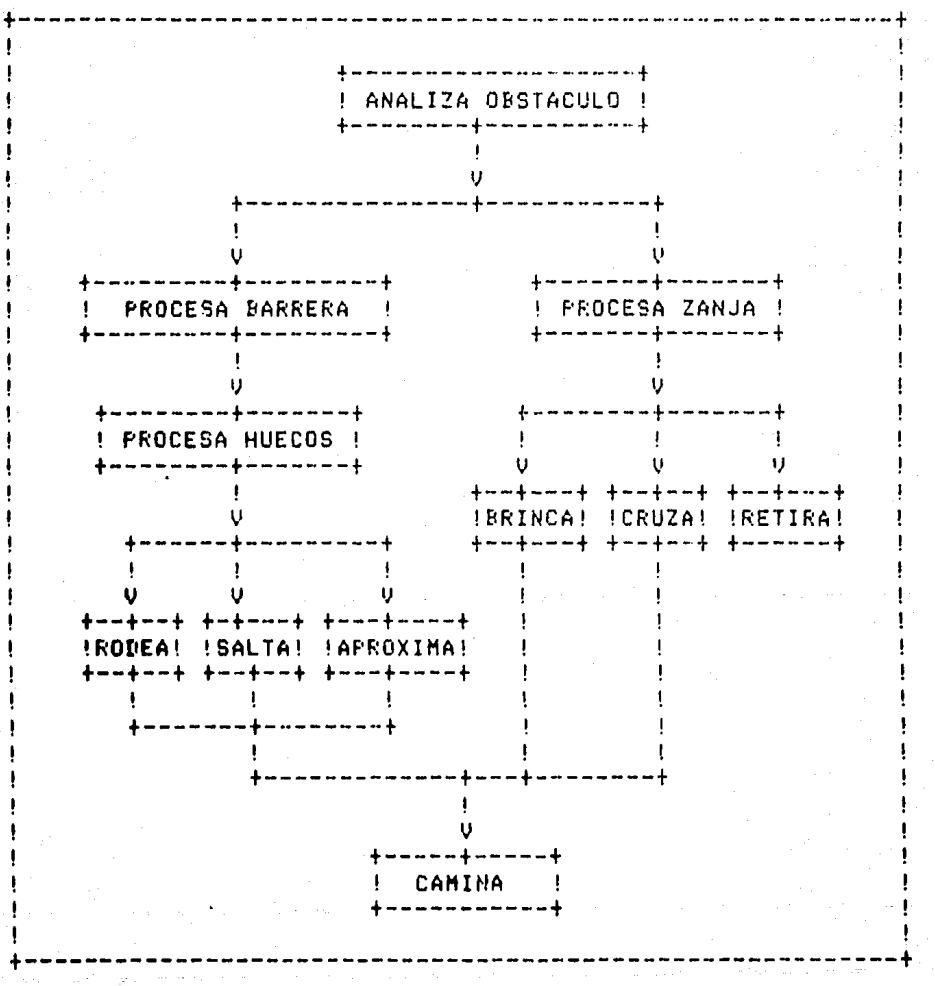
4.3 DIAGRAMA GENERAL

El comportamiento general del programa se muestra en los siguientes diagramas:

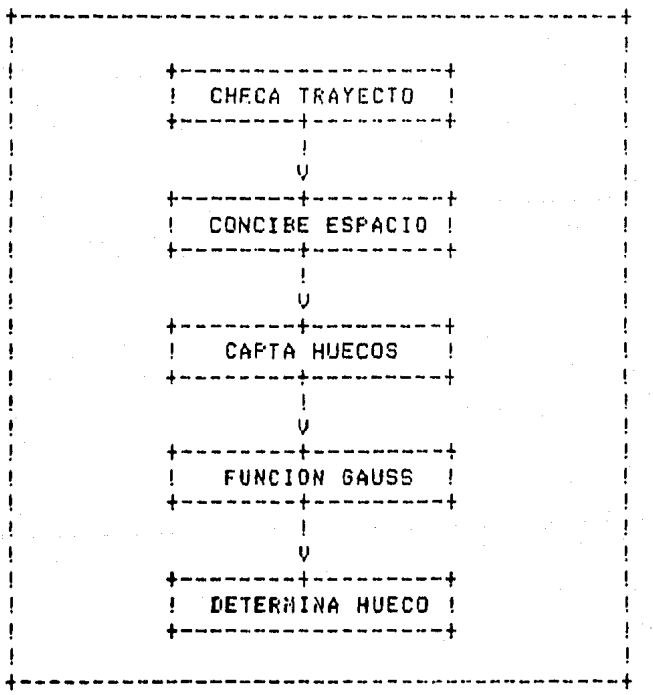



C A M I N A
ORIENTA
CHECA TRAYECTO
CHECA OBSTACULO
TRAGAR

C H E C A O B S T A C U L O



P R O C E S A H U E C O S



SIMULACION EN COMPUTADORA DEL MODELO GENERAL

4.3.1 DESCRIPCION DE MODULOS

En forma general describiremos la función de cada bloque, en orden alfabético; para una mejor comprensión de cada bloque se puede referir al listado del programa, el cual se presenta autodocumentado.

ANALIZA OBSTACULO .- Determina con que tipo de obstáculo se ha cruzado la rana en su trayectoria hacia la presa o punto de orientación.

APROXIMA .- Determina si el alcance de la lengua es tal que se pueda traer a la presa por entre los barrotes de la barrera.

ATAQUE .- Coordina la conducta de ataque del modelo.

BRINCA .- Asigna a la rana nuevas coordenadas, debido a que está brinca al otro lado de la zanja.

BUSCA LUGAR .- Localiza en la matriz de visión el mejor punto de huida.

CAMINA .- Rutina recursiva que simula el caminar de la rana mientras esta no se haya retirado, o tenga obstáculos enfrente, o no pueda comerse a la presa entre los barrotes de la barrera.

CAMINA RANA .- Se encarga de asignar nuevas coordenadas a la posición de la rana, en el espacio de simulación, dentro de un hueco seleccionado.

CAPTA HUECOS .- Capta los huecos que la rana ve de acuerdo a la posición de la presa o el punto a la cual está orientada.

CHECA OBSTACULO .- Coordina las rutinas de reconocimiento del obstáculo y la acción motora realizar en función de las características de dicho obstáculo.

CHECA TRAYECTO .- Examina la trayectoria que debe seguir la rana y detecta el obstáculo más próximo a librar para llegar al objetivo.

CONCIBE ESPACIO .- Hace un reconocimiento del espacio visual de la rana en lo referente a barreras, alturas, zanjas, profundidades, etc. y deposita la información en la matriz de objetos fijos.

CRUZA .- Asigna nuevas coordenadas a la rana de tal forma que haya cruzado la zanja en cuestión.

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

DETERMINA HUECO .- Rutina que escoge el hueco más atractivo, es decir de acuerdo a una competencia entre las células pretectales, escoge aquella que tenga mayor valor.

FOTOTAXICA .- Se encarga de hacer las llamadas a las rutinas necesarias para llevar a cabo la simulación de la respuesta fototáxica.

FUNCION GAUSS .- Genera una función gaussiana de la forma:

$$f = EXP (- abs ((posición) ** 2) / K)$$

donde la constante K es solicitada al usuario en forma interactiva.

HUIDA .- LLama a las rutinas necesarias para simular la respuesta de huida.

ORIENTA .- Rutina que planea la trayectoria rana-presa para su posterior análisis.

PIDE CARACTERISTICAS .- Solicita, ya sea en forma interactiva o por medio de un archivo en disco, los datos referentes a características de los obstáculos, y posiciones de rana, presa o predador.

PROCESA BARRERA .- Determina la conducta a efectuar con respecto a la barrera en turno, si la lengua alcanza a atrapar a la presa entre los barrotes de la barrera, si la salta o si la rodea.

PROCESA HUECOS .- Se encarga de evaluar los huecos en cuanto a ancho, profundidad, color del fondo y posición de la presa o predador.

PROCESA ZANJA .- Determina que es lo que se va a hacer con la zanja en cuestión, si la puede brincar, cruzar hasta el otro extremo o si se retira.

RETIRA .- Finaliza la simulación, ya que no fué posible evadir la zanja que se interpone entre la rana y la presa, debido a las dimensiones de la zanja.

RODEA .- Asignación de nuevas coordenadas en el borde de una barrera con el fin de poderla evadir.

SALTA .- Asignación de nuevas coordenadas al otro lado de la barrera, debido a que ésta es saltada por la rana.

TRAGAR .- Finaliza el proceso de simulación debido a que ahora si fué posible trasarse a la presa.

4.4 VARIABLES

A continuación describiremos las variables utilizadas en el programa simulador, señalando su función y estructura.

RANA: Arreglo de tipo real de una dimensión, compuesto de cinco elementos en cuyos valores se almacena:

RANA[0]: Coordenada X de la rana.

RANA[1]: Coordenada Y de la rana.

RANA[2]: Capacidad de salto de longitud (para evadir zanjas).

RANA[3]: Capacidad de salto de altura (para evadir barreras).

RANA[4]: Longitud de alcance de la lengua.

NOTA: Para las simulaciones de huida y fototáxica solamente se usan RANA[0] y RANA[1].

PRESA: Arreglo de tipo real de una dimensión, que representa a la presa o al predador, compuesto de dos elementos en cuyos valores se almacena:

PRESA[0]: Coordenada X de la presa o predador.

PRESA[1]: Coordenada Y de la presa o predador.

ESPACIO: Arreglo tipo carácter, de dos dimensiones con 61 elementos en una, y 21 elementos en la otra. Representa en una dimensión las coordenadas X (0-60) y en la otra a las coordenada Y (0-20). En cada elemento se colocarán los obstáculos que contendrá la simulación, llenándolo con 'X' si es una barrera, o con una 'Z' si es una zanja; asimismo se colocará en la posición de la rana una 'R', en el de la presa o predador una 'P' y en el punto hacia el cual mira la rana un '.'. También se rellena con '.' aquellas coordenadas por donde la rana realiza su trayectoria.

OBSTACULOS: Arreglo real de dos dimensiones (104 por 10), en el cual se pueden almacenar las características de hasta 104 obstáculos, los primeros 4 obstáculos corresponden a las barreras que limitan el espacio de simulación, a continuación se define el significado de los renglones y columnas de esta matriz:

OBSTACULOS[-4,*]: Coordenadas para la barrera que limitará el norte del espacio de visión.

OBSTACULOS[-3,*]: Coordenadas para la barrera que limitará el este del espacio de visión.

OBSTACULOS[-2,*]: Coordenadas para la barrera que limitará el sur del espacio de visión.

OBSTACULOS[-1,*]: Coordenadas para la barrera que limitará el oeste del espacio de visión.

OBSTACULOS[*,0]: Tipo de obstáculo (0 zanja, 1 barrera).

OBSTACULOS[*,1]: Coordenada X1.

OBSTACULOS[*,2]: Coordenada Y1.

OBSTACULOS[*,3]: Coordenada X2.

OBSTACULOS[*,4]: Coordenada Y2.

OBSTACULOS[*,5]: Coordenada X3 (sólo para zanjas).

OBSTACULOS[*,6]: Coordenada Y3 (sólo para zanjas).

OBSTACULOS[*,7]: Coordenada X4 (sólo para zanjas).

OBSTACULOS[*,8]: Coordenada Y4 (sólo para zanjas).

OBSTACULOS[*,9]: Profundidad de la zanja o altura de la barrera.

OBSTACULOS[*,10]: Color del obstáculo (1 azul, 2 verde, 3 amarillo, 4 rojo, 5 negro).

HUECOS: Arreglo real de 2 dimensiones (20 por 5), en el cual se almacenan las caracteristicas de hasta 20 huecos, las cuales son:

HUECOS [*,1]: Indice del inicio del hueco (o grado inicial del hueco, que va de 1 a 170).

HUECOS [*,2]: Indice del final del hueco (o grado final del hueco, que va de 1 a 170).

HUECOS [*,3]: Ancho en grados del hueco.

HUECOS [*,4]: Valor de la profundidad del hueco en el lado izquierdo.

HUECOS [*,5]: Valor de la profundidad del hueco en el lado derecho.

MATCEL: Arreglo real de 2 dimensiones (170 por 5) en el cual se almacenan las caracteristicas del campo visual que capta la rana, las cuales son:

MATCEL [*,1]: Profundidad.

MATCEL [*,2]: Número del obstáculo que capta para ese ángulo.

MATCEL [*,3]: Coordenada X del punto de intersección del ángulo de visión con el fondo del espacio o algún obstáculo.

MATCEL [*,4]: Coordenada Y del punto de intersección del ángulo de visión con el fondo del espacio o algún obstáculo.

MATCEL [*,5]: Color del fondo.

4.5 PROGRAMA EN BURROUGHS 7800

El programa que se procesa en Burroughs 7800, está implementado en un lenguaje de alto nivel, ALGOL (1,2, 3).

4.5.1 FUNCIONAMIENTO

En forma general el programa principal pide al usuario la opción que se desea simular, siendo estas las siguientes:

1) Simulación de ataque (con evaluación de huecos o sin ella).

2) Simulación de huida.

3) Simulación de respuesta fototáxica.

4) Terminar la ejecución.

Ya sea que se haya escogido cualesquiera de las simulaciones, el programa pide al usuario la forma en que se darán los datos que sirven para alimentar los parametros referentes a número de obstáculos, tipo del mismo, coordenadas, altura o profundidad, posición de la rana, parámetros de la misma (largo de lengua, capacidad de salto de altura y de longitud),y posición de la presa; los datos pueden ser facilitados en forma interactiva por terminal o bien pueden ser tomados de un archivo en disco. Posteriormente se presenta la opción de cómo se desea la emisión de resultados, si se emitirán a la terminal. si se imprimirán a papel, o si se generan en un archivo en disco bajo el nombre RANA/RES. Una vez alimentado el sistema, se procede a iniciar la simulación. A continuación se explicará en forma detallada cómo opera cada una de las opciones de simulación.

4.5.2 SIMULACION DE ATAQUE, MODELO CON HUECOS

La rana se ORIENTA hacia la presa en la rutina CAMINA, analiza el espacio de visión que se le presenta en CHECATRAYECTO para determinar si se retira o sigue adelante. Después llama a la rutina CHECAOBSTACULO para analizar si hay algún obstáculo que se interponga entre su posición y la posición de la presa. En caso de no existir obstáculo, camina hacia la presa y se la traga; pero si hay obstáculo que se interponga, procede a evaluar los huecos de la siguiente manera:

Hace un barrido del espacio de visión de izquierda a derecha con un campo visual de 170 grados, con incrementos de un grado; de esta forma se genera una matriz de visión en la que existen los parámetros de profundidad, coordenadas de cada punto y tipo de obstáculo con el que se intersecta. Al encontrar el borde de un obstáculo empieza a reconocer un hueco, introduciendo las caracteristicas de cada uno de estos en la matriz de huecos, en la que existen los parámetros de ancho del hueco, profundidad del mismo y coordenadas de inicio y fin.

El programa elimina aquellos huecos que son muy pequeños en ancho, debido a que estos son demasiado estrechos para que la rana pueda caminar entre ellos, asimismo se limita a cada hueco a no tener una amplitud visual mayor a 15 grados, y una profundidad máxima de 10 unidades más de la distancia rana-presa.

Evaluados los huecos se genera una matriz de competencia en la cual se calculan los valores de los elementos en función de: el ancho del hueco, profundidad del hueco y posición de la presa; se busca en esta matriz el elemento de mayor valor y se escoge a su índice como el grado hacia el cual se dirigirá la rana, se accesa la matriz de visión y se eligen las coordenadas apropiadas (calculando un tercio de la distancia de la profundidad), y se camina la rana hacia esa nueva posición inicial, para que nuevamente la rutina CAMINA tome el control (debido a la recursividad empleada) y vuelva a simular a partir de la nueva situación de la rana.

4.5.3 SIMULACION DE ATAQUE, MODELO SIN HUECOS

Como en el caso anterior se orienta a la rana hacia la presa. Después llama a la rutina CHECAOBSTACULO para analizar si hay algún obstáculo que se interponga entre su posición y la posición de la presa. En caso de no existir obstáculo, camina hacia la presa y se la traga; pero si hay obstáculo que se interponga entonces se checa el tipo de obstáculo del cual se trata.

En caso de que el obstáculo sea una barrera, y no se pueda ni saltar, ni sea posible tragar a la presa entre los barrotes, se procederá a rodearla por uno de los bordes, escogiendo aquél en el cual la distancia hacia el punto de intersección con la trayectoria sea menor.

Y en el caso en el cual el obstáculo sea una zanja, se determinará si la salta o si la cruza.

En cualquiera de los dos casos, el saltar o rodear la barrera, o cruzar o brincar la zanja, se regresa a la rutina CAMINA para definir la conducta del modelo, a partir de la nueva posición de la rana.

4.5.4 SIMULACION DE HUIDA

Adicionalmente a los datos de la posición y características de la rana y del predador, se pide como dato hacia que punto del espacio (coordenadas x y) está mirando la rana, para tomar esta dirección como orientación base y centro del campo visual.

Asimismo para este caso, y para la simulación de Respuesta Fototáxica, se pide adicionalmente como dato que color tendrán los bordes que delimitan el espacio de acción de la simulación, ya que el color del fondo del hueco

interviene para la evaluación del índice óptimo. Para este caso se tiene la variante de que el peso que ahora tendrá la posición del predador influye en forma negativa en la evaluación del índice óptimo.

En el momento en que la rana detecta en su campo visual al predador, se generan los huecos de la misma manera que en el caso anterior, pero ahora en la matriz de competencia intervienen positivamente el ancho, profundidad y color del fondo del hueco, y negativamente la posición del predador (el cual no esta centrado en el campo visual); se escoge igualmente el índice óptimo (de mayor valor) el cual nos permite obtener de la matriz de visión el punto hacia el cual la rana se dirigirá, caminando solo el 90de de la profundidad del hueco.

4.5.5 SIMULACION DE RESPUESTA FOTOTAXICA

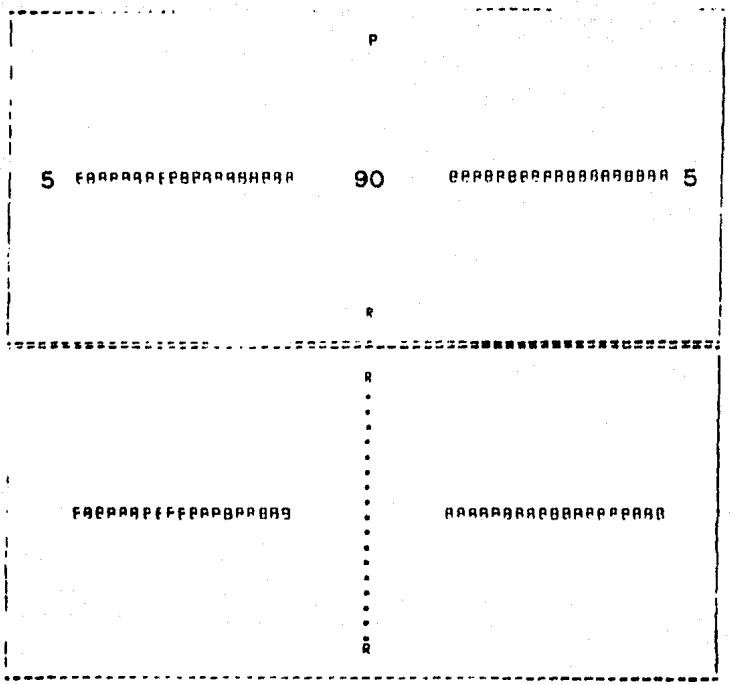
Similarmente al caso anterior, la rana estará orientada hacia cierto punto del espacio y los bordes de éste tendrán cierto código de color que afectará la evaluación del índice óptimo. Así pués, la rana se encuentra en esa posición inicial, barre el espacio de visión, genera los huecos y escoge el índice óptimo, obtiene las coordenadas x,y de la matriz de visión, y camina hacia ellas en un 60 por ciento de la profundidad del hueco.

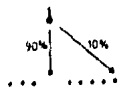
4.5.6 LISTADO Y RESULTADOS

En el apéndice C se muestra el programa en Algol que se encarga de realizar la simulación en BURROUGHS A continuación se muestran algunos resultados de la simulación destacando en la parte derecha los experimentos realizados por Tom Collet (4), con la misma configuración.

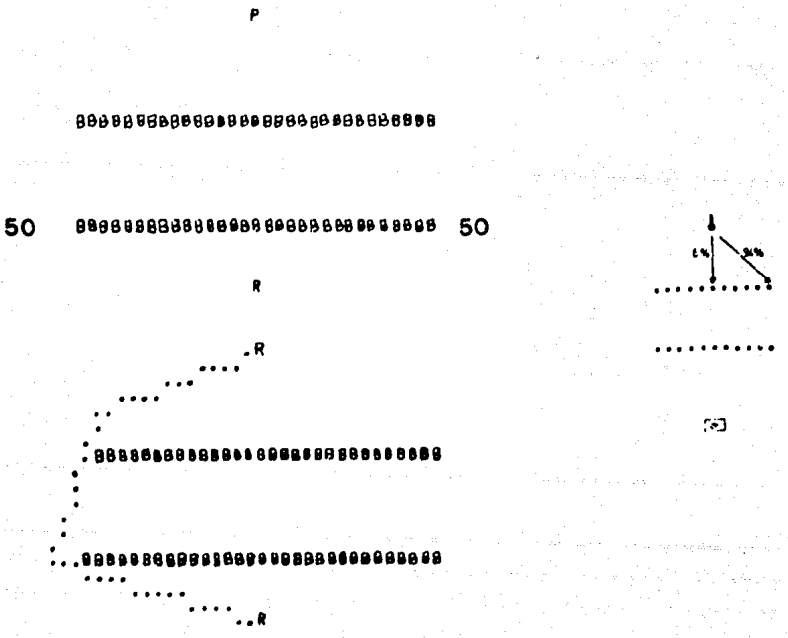
SIMULACION EN COMPUTADORA DEL MODELO GENERAL

A)



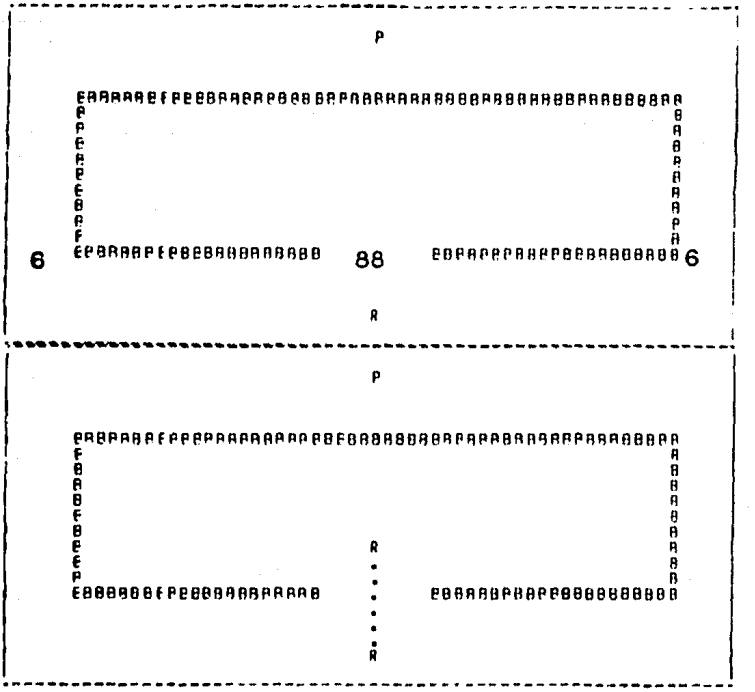


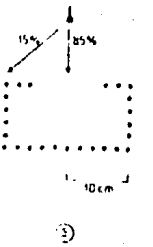
B)



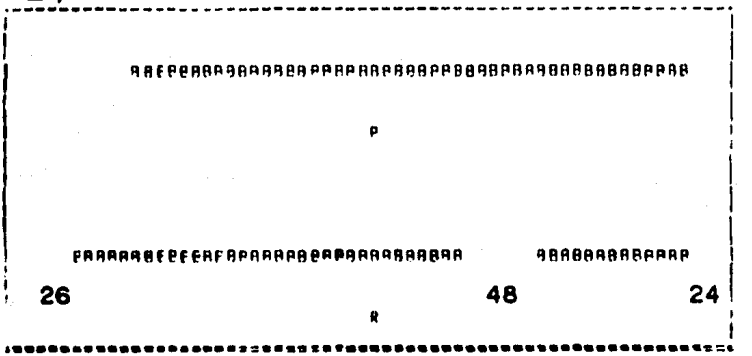
SIMULACION EN COMPUTADORA DEL MODELO GENERAL

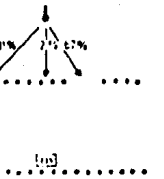
A)

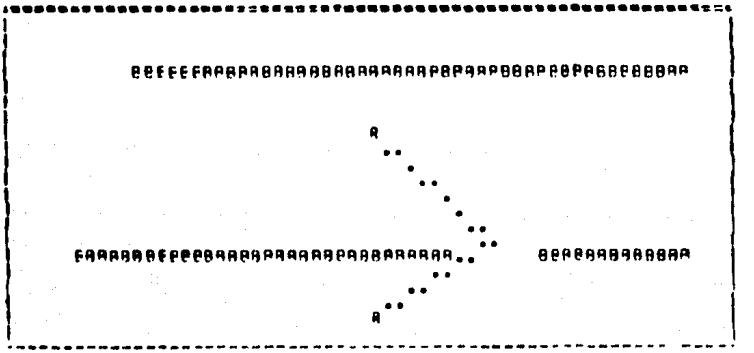




B)

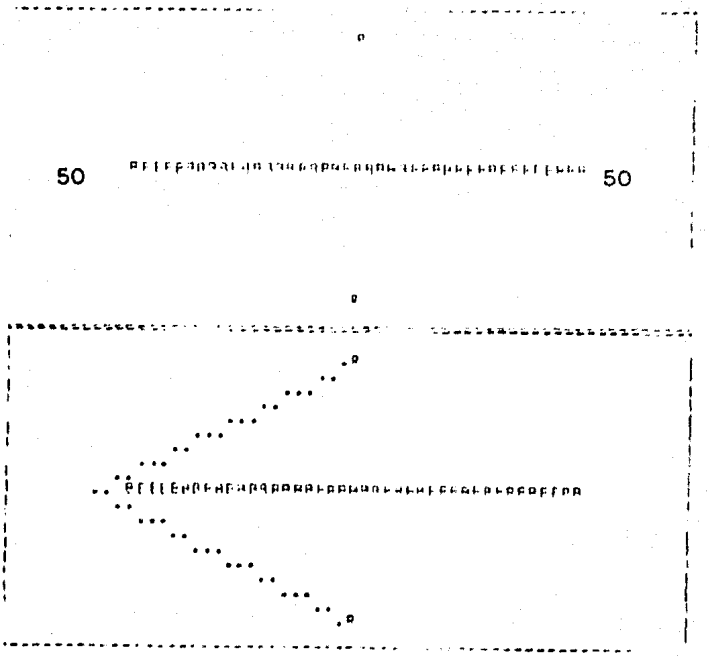




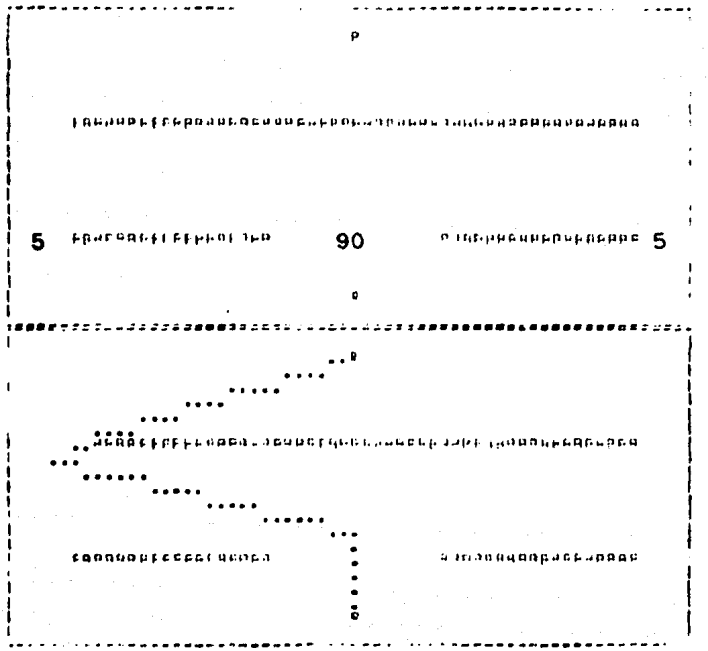


SIMULACION EN COMPUTADORA DEL MODELO GENERAL

A)

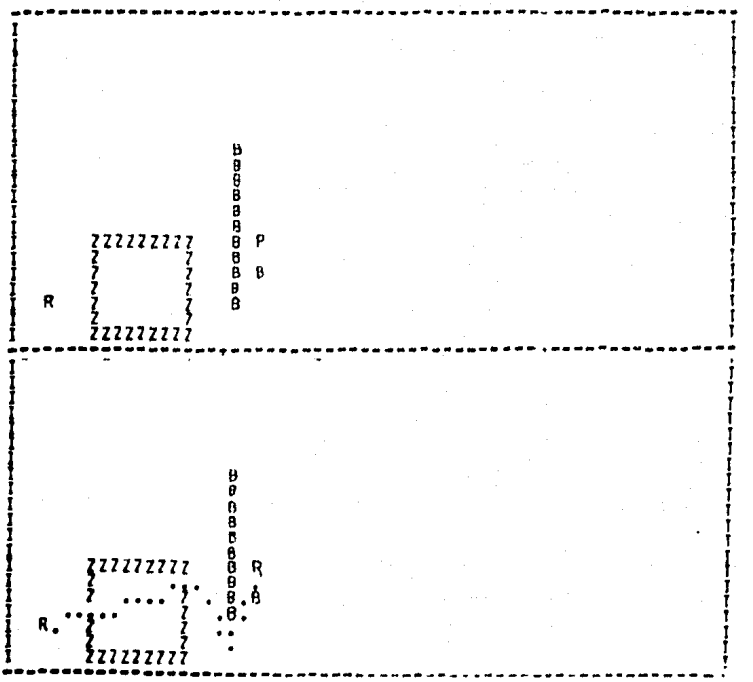


B)

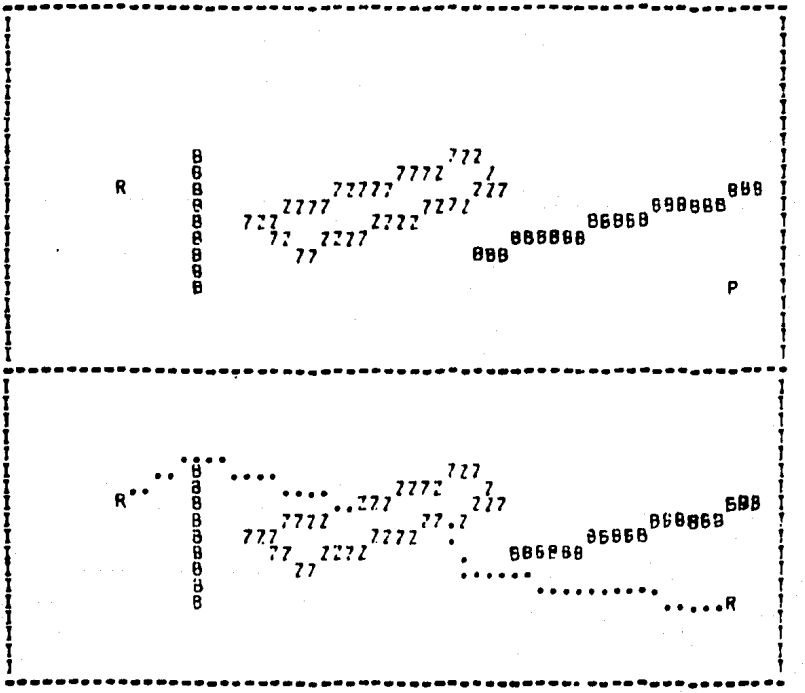


SIMULACION EN COMPUTADORA DEL MODELO GENERAL

A)



B)



SIMULACION EN COMPUTADORA DEL MODELO GENERAL

En los siguientes resultados se muestran la simulación de ataque con algunos de los valores que toman los vectores de anchos, profundidad, posición de la presa o predador (función gausiana), y evaluación de huecos, para tener una mejor visión del comportamiento del modelo; y en las simulaciones de huida y respuesta fototáctica solo se presentan las condiciones para la simulación, el estado inicial y el estado final.

5.7 ATAQUE

CARACTERISTICAS DE LA RANA:

POSICION X Y-->	20.00	5.00
SALTO ALTURA-->	1.00	
SALTO LONGITUD-->	1.00	
ALCANCE LENGUA-->	1.00	

CARACTERISTICAS DE LA PRESA:

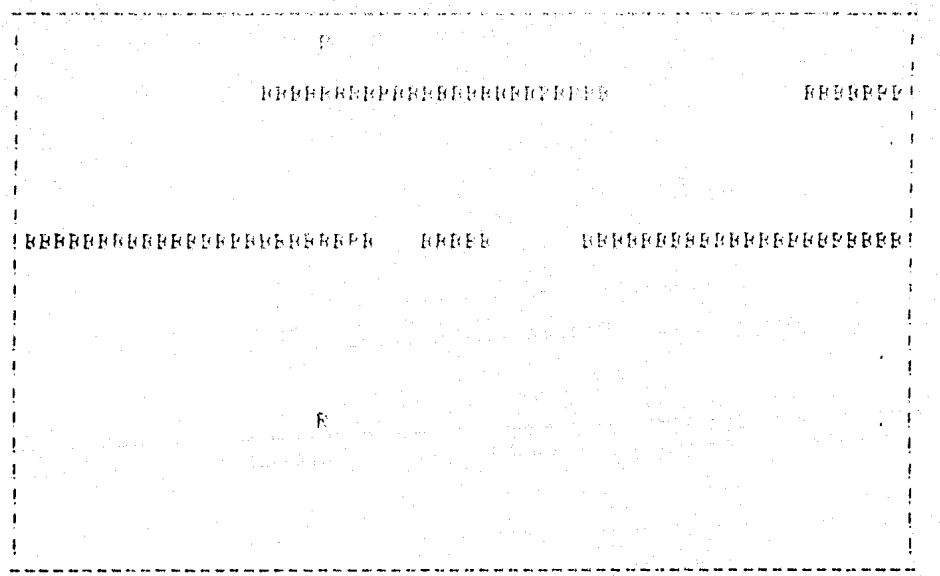
POSICION X Y-->	20.00	20.00

CARACTERISTICAS DE LOS OBSTACULOS:

# OBST	TIPO OBST	X1-Y1	X2-Y2	X3-Y3	X4-Y4	ALTURA O PROF.
0	BARDA	0.00	23.00	0.00	0.00	33.00
		12.00	12.00	0.00	0.00	
1	BARDA	27.00	31.00	0.00	0.00	32.00
		12.00	12.00	0.00	0.00	
2	BARDA	38.00	60.00	0.00	0.00	44.00
		12.00	12.00	0.00	0.00	
3	BARDA	16.00	39.00	0.00	0.00	33.00
		18.00	18.00	0.00	0.00	
4	BARDA	53.00	60.00	0.00	0.00	33.00
		18.00	18.00	0.00	0.00	

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

EL ESTADO INICIAL ES:



LA RANA SE ENCUENTRA EN EL PUNTO
(20.00, 5.00)
Y LA PRESA EN EL PUNTO
(20.00,20.00)

SE TOPA CON LA BARDA 0

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

EL EDO DE LAS CELULAS DEL OBSTACULO:

1 --->	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
11 --->	-1.00	-1.00	-1.00	-1.00	0.00	0.00	0.00	0.00	0.00	0.00
21 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
31 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
41 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
51 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
61 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
71 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
81 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
91 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
101---->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.00	3.00
111---->	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
121---->	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	1.00
131---->	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
141---->	1.00	1.00	-4.00	-4.00	-4.00	-4.00	-4.00	-4.00	-4.00	-4.00
151---->	-4.00	-4.00	-4.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
161---->	2.00	2.00	2.00	2.00	2.00	-3.00	-3.00	-3.00	-3.00	-3.00

EL NUMERO DE HUECOS ENCONTRADOS FUE 2

NUM HUECO	INICIO	FIN	ANCHO	DIFIZQ	DIFDER
1	109	129	15	7	8
2	143	153	11	15	21

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

LA MATRIZ DE EVALUACION DE HUECOS QUEDA:

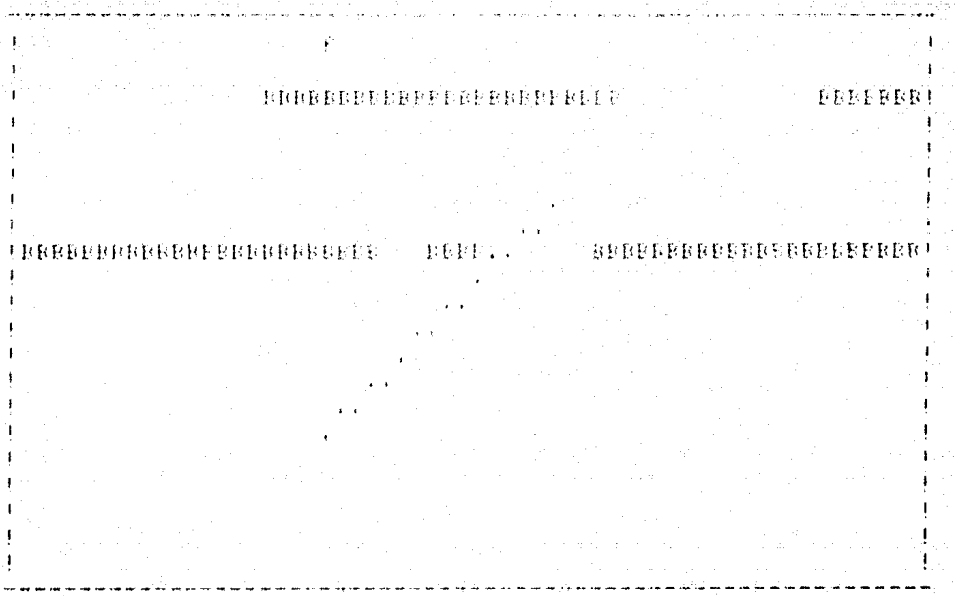
0.00

1 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
21 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
31 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
41 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
51 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
61 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
71 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
81 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
91 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
101 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.04	7.56
111 --->	7.16	6.84	6.59	6.40	6.24	6.13	6.04	5.97	5.92	5.88
121 --->	5.86	5.84	5.83	5.82	5.81	5.81	5.81	5.80	5.80	0.00
131 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
141 --->	0.00	0.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
151 --->	9.00	9.00	9.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
161 --->	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

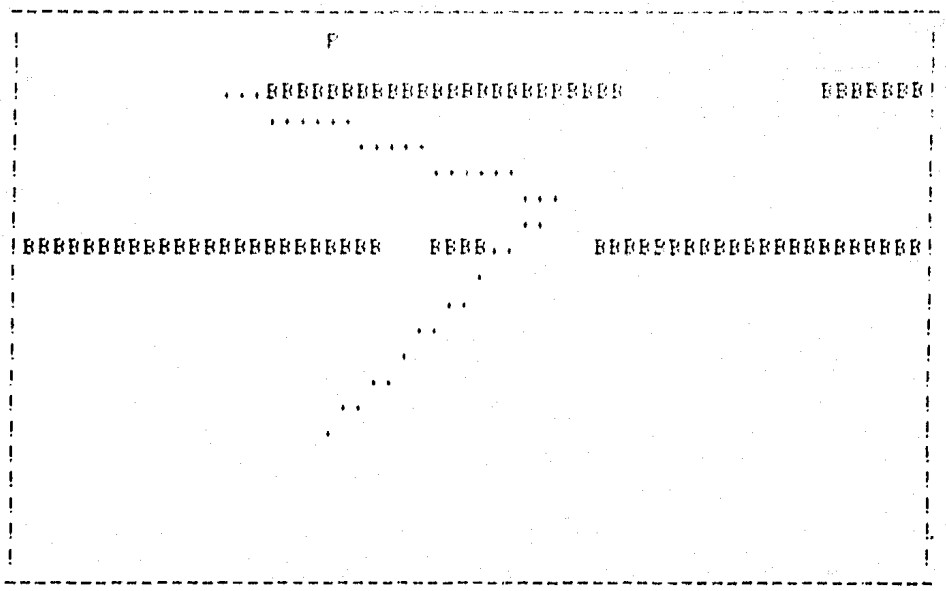
LOS VALORES DE LAS CONSTANTES ANCHO, PROFUNDIDAD Y GAUSS SON:
20 40 40

Y EL GRADO DE LA FUNCION GAUSIANA ES:
200

EL INDICE OPTIMO ES 143

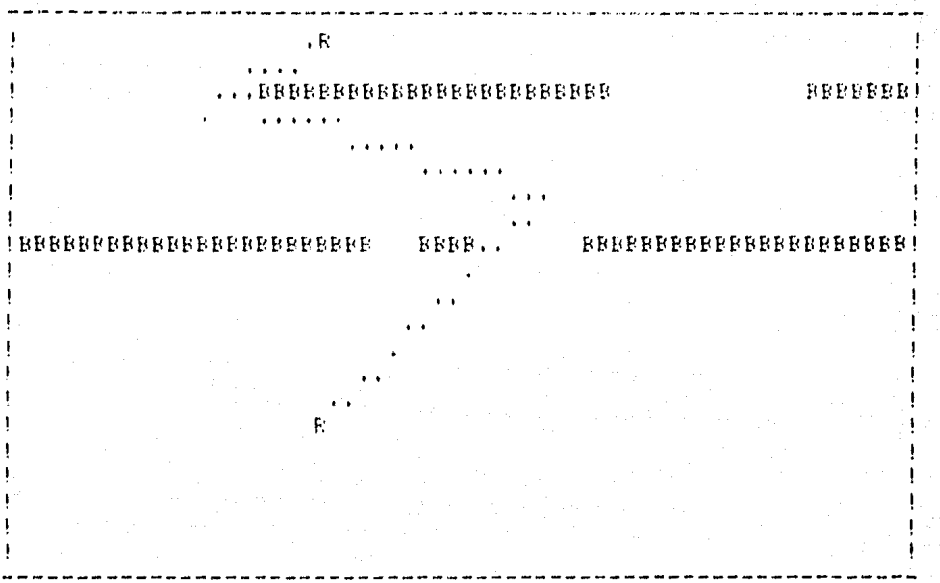


```
==================================
LA RANA SE ENCUENTRA EN EL PUNTO
      (35.35,13.87)
Y LA PRESA EN EL PUNTO
      (20.00,20.00)
==================================

**************************************
SE TOPA CON LA BARDA   3
**************************************
```



```
==================================
LA RANA SE ENCUENTRA EN EL PUNTO
      (12.75,17.77)
Y LA PRESA EN EL PUNTO
      (20.00,20.00)
==================================

LA RANA SE DIRIGE HACIA LA PRESA Y SE LA COME
```

EL ESTADO FINAL ES:



SIMULACION DE ATAQUE TERMINADA

4.5.8 HUIDA

CARACTERISTICAS DE LA RANA:
POSICION X Y--> 22.00 2.00

CARACTERISTICAS DEL PREDADOR:
ANGULO, DIST--> -55 14

POSICION X Y--> 10.03 10.03

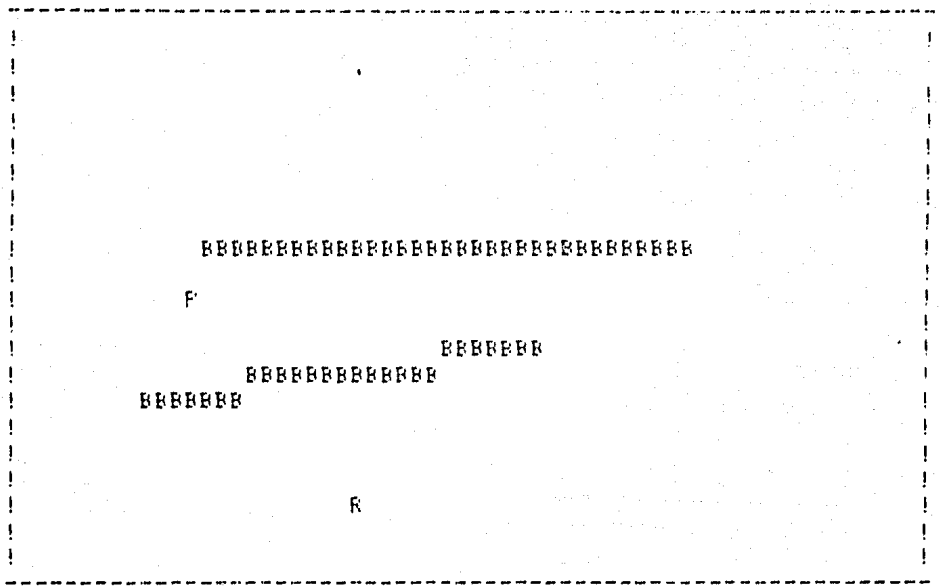
ORIENTADA HACIA EL PUNTO X Y--> 22.00 19.00

CARACTERISTICAS DE LOS OBSTACULOS:

# OBST	TIPO OBST	X1-Y1	X2-Y2	X3-Y3	X4-Y4	ALTURA O PROF.	COLOR
-4	BARDA	0.00 20.00	60.00 20.00	0.00 0.00	0.00 0.00	999.00	AZUL
-3	BARDA	60.00 0.00	60.00 20.00	0.00 0.00	0.00 0.00	999.00	AMARIL
-2	BARDA	0.00 0.00	60.00 0.00	0.00 0.00	0.00 0.00	999.00	VERDE
-1	BARDA	0.00 0.00	0.00 20.00	0.00 0.00	0.00 0.00	999.00	ROJO
0	BARDA	12.00 12.00	44.00 12.00	0.00 0.00	0.00 0.00	6.00	NEGRO
1	BARDA	34.00 8.00	8.00 6.00	0.00 0.00	0.00 0.00	6.00	NEGRO

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

EL ESTADO INICIAL EN EL CUAL
LA RANA SE DA CUENTA DEL PREDADOR ES:



SIMULACION EN COMPUTADORA DEL MODELO GENERAL

EL ESTADO FINAL DE LA RANA AL HUIR ES:

```
----------------------------------------------------------------
!                                                              !
!                       .                                      !
!                                                              !
!                                                              !
!                                                         .R   !
!                                                       ..     !
!                                                    ...       !
!                                                  ..          !
!            BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB ...            !
!                                             ..               !
!           P                              ...                 !
!                                         ..                   !
!                            BBBBBBB ...                       !
!               BBBBBBBBBBBBBB      ..                         !
!        BBBBBBB                 ...                           !
!                              ..                              !
!                           ...                                !
!                         ..                                   !
!                      R.                                      !
!                                                              !
!                                                              !
----------------------------------------------------------------
```

4.5.9 FOTOTAXICA

CARACTERISTICAS DE LA RANA:

POSICION X Y--> 33.00 2.00

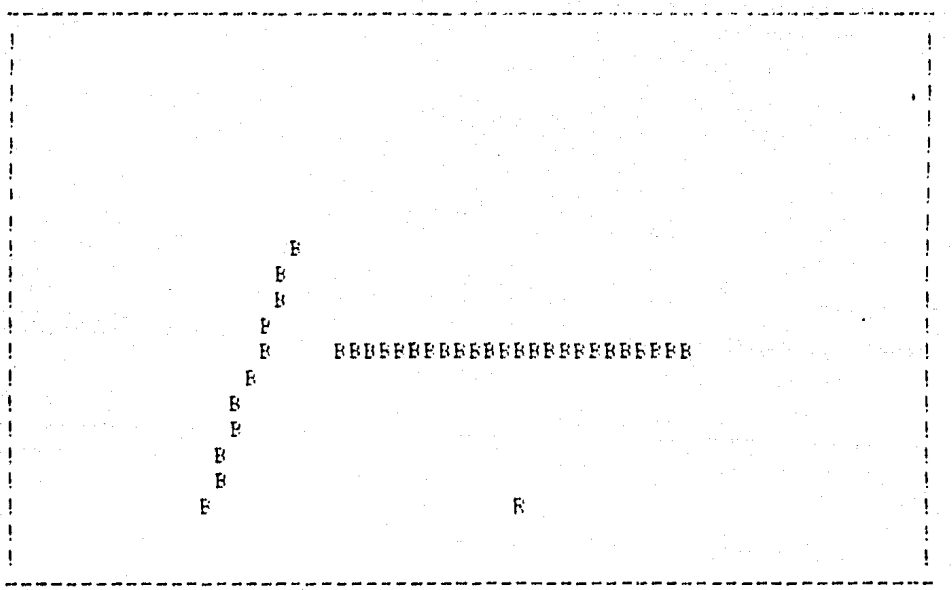
ORIENTADA HACIA EL PUNTO X Y--> 59.00 18.00

CARACTERISTICAS DE LOS OBSTACULOS:

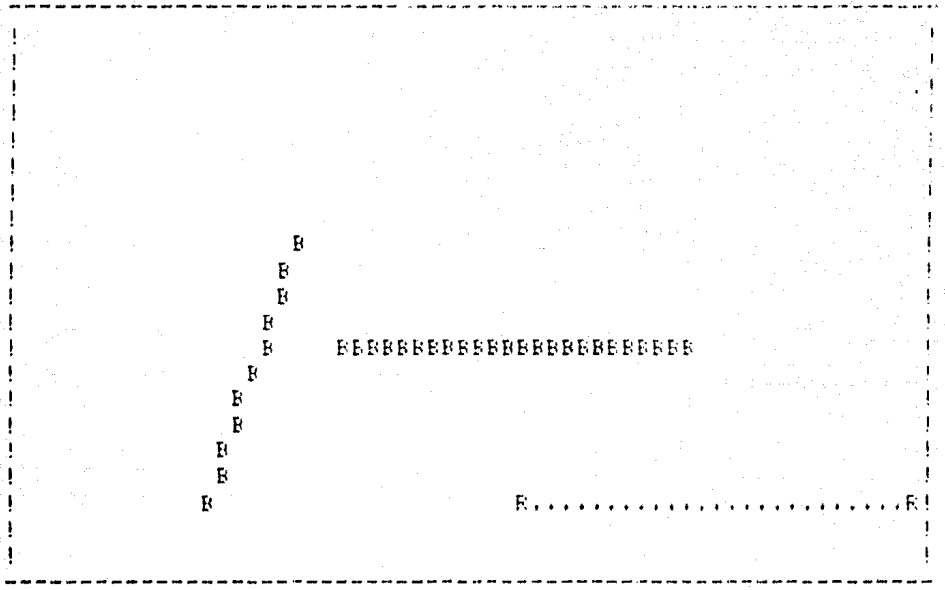
# OBST	TIPO OBST	X1-Y1	X2-Y2	X3-Y3	X4-Y4	ALTURA O PROF.	COLOR
-4	BARDA	0.00 20.00	60.00 20.00	0.00 0.00	0.00 0.00	999.00	AMARIL
-3	BARDA	60.00 0.00	60.00 20.00	0.00 0.00	0.00 0.00	999.00	AMARIL
-2	BARDA	0.00 0.00	60.00 0.00	0.00 0.00	0.00 0.00	999.00	ROJO
-1	BARDA	0.00 0.00	0.00 20.00	0.00 0.00	0.00 0.00	999.00	AZUL
0	BARDA	18.00 12.00	12.00 2.00	0.00 0.00	0.00 0.00	2.00	NEGRO
1	BARDA	44.00 8.00	21.00 8.00	0.00 0.00	0.00 0.00	4.00	NEGRO

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

EL ESTADO INICIAL PARA SIMULAR LA
RESPUESTA FOTOTAXICA ES:



EL ESTADO FINAL DE REPOSO DE
LA RANA ES:



4.6 GRAFICACION EN CROMEMCO

En la parte anterior se ha hecho una descripción del funcionamiento del programa implementado en la computadora BURROUGHS, en lenguaje ALGOL, para simular el modelo del aparato visiomotor de los anfibios. Como se pudo observar en las gráficas de los resultados obtenidos se presentan principalmente tres inconvenientes:

1) Debido las restricciones de las dimensiones, tanto en pantalla como en la impresión a papel, la resolución resulta muy baja (20 x 60).

2) Es difícil poder apreciar la orientación de la rana debido a que la representación de ésta se hace por medio de un sólo carácter.

3) No es posible representar visualmente la respuesta fototáxica del animal debido a la falta de tonos contrastantes.

Por las razones anteriormenter expuestas, y por la necesidad dar una mejor presentación a los resultados obtenidos en el modelo se optó por diseñar un sistema en la CROMEMCO el cual pudiera superar los inconvenientes que presentaba la representación gráfica en BURROUGHS, utilizando el paquete SDI-GRAPHICS (5).

Algunas de las características del sistema de graficación de CROMEMCO son las siguientes:

1) Ofrece dos niveles de resolución: Alto (482 x 756), y medio (241 x 378). El primero sólo cuenta con dos tonos de contraste, el blanco y el negro. En cambio el segundo tiene hasta 15 colores diferentes a escoger para el trazo de líneas y figuras, con la flexibilidad de definir un color en especial utilizando como colores base el rojo, el azul y el verde, según la cantidad de tinte que se use en la mezcla de estos, será el color resultante.

2) Contiene rutinas de propósito especial para trazos específicos: puntos, líneas, círculos, circunferencias, cuadrados, rectángulos, polígonos, etc, etc.

3) Maneja dos páginas de trabajo, lo cual resulta de gran utilidad cuando se trata de simular animación o movimiento de figuras.

4) Permite el uso de 'ventanas', es decir que al mismo tiempo que se observa la página de trabajo, se puede ver parte de la página alterna.

El programa diseñado en la CROMEMCO tiene como objetivo mover cualquier figura sobre cualquier trayectoria, en este caso en especial simularemos el movimiento de la rana a lo largo de las diferentes trayectorias que se obtuvieron en los resultados de la simulación.

El programa utiliza dos archivos como fuente de información. El primero contiene tanto el número de puntos como los puntos que forman la figura a graficar, los puntos deberán estar dados en coordenadas cartesianas tomando el centro de la figura como origen del plano. Para ejemplificar esto consideremos que deseamos graficar el movimiento de un cuadrado cuya longitud de lado es 2: si tomamos el centro de la figura como origen del plano los vértices tendrán las coordenadas (1,1), (1,-1), (-1,-1) y (-1,1) respectivamente en el sentido de las manecillas del reloj, así pues el archivo deberá contener los siguientes datos:

```
 4                    % numero de puntos
 1,1                  % vértice # 1
 1,-1                 % vértice # 2
-1,-1                 % Vértice # 3
-1,1                  % vértice # 4
```

El segundo de los archivos que consulta el programa de graficación es el que contiene tanto los datos del espacio, como los de la trayectoria a seguir. Este archivo debe estar configurado de la siguiente forma:

1) Número de obstáculos que existen en el espacio.

2) Descripción de cada uno de los obstáculos

   a) Tipo y color

   b) Coordenadas de los puntos que componen dicho obstáculo (por pares)

3) Posición de la presa o predador.

4) Número de veces que se re-orienta el anfibio

5) Coordenadas de orientación (por pares)

SIMULACION EN COMPUTADORA DEL MODELO GENERAL

En el punto No. 2, el tipo de obstáculo debe tomar cualquiera de los siguientes valores:

1) 0 si se trata de una zanja.

2) 1 si se trata de una barrera.

3) cualquier otro (se graficará como un circulo).

El color se designará de acuerdo a las siguientes características del obstáculo:

1) 15 (blanco), para las barreras en general.

2) 3 (verde), para las zanjas poco profundas.

3) 10 (rojo), para zanjas profundas.

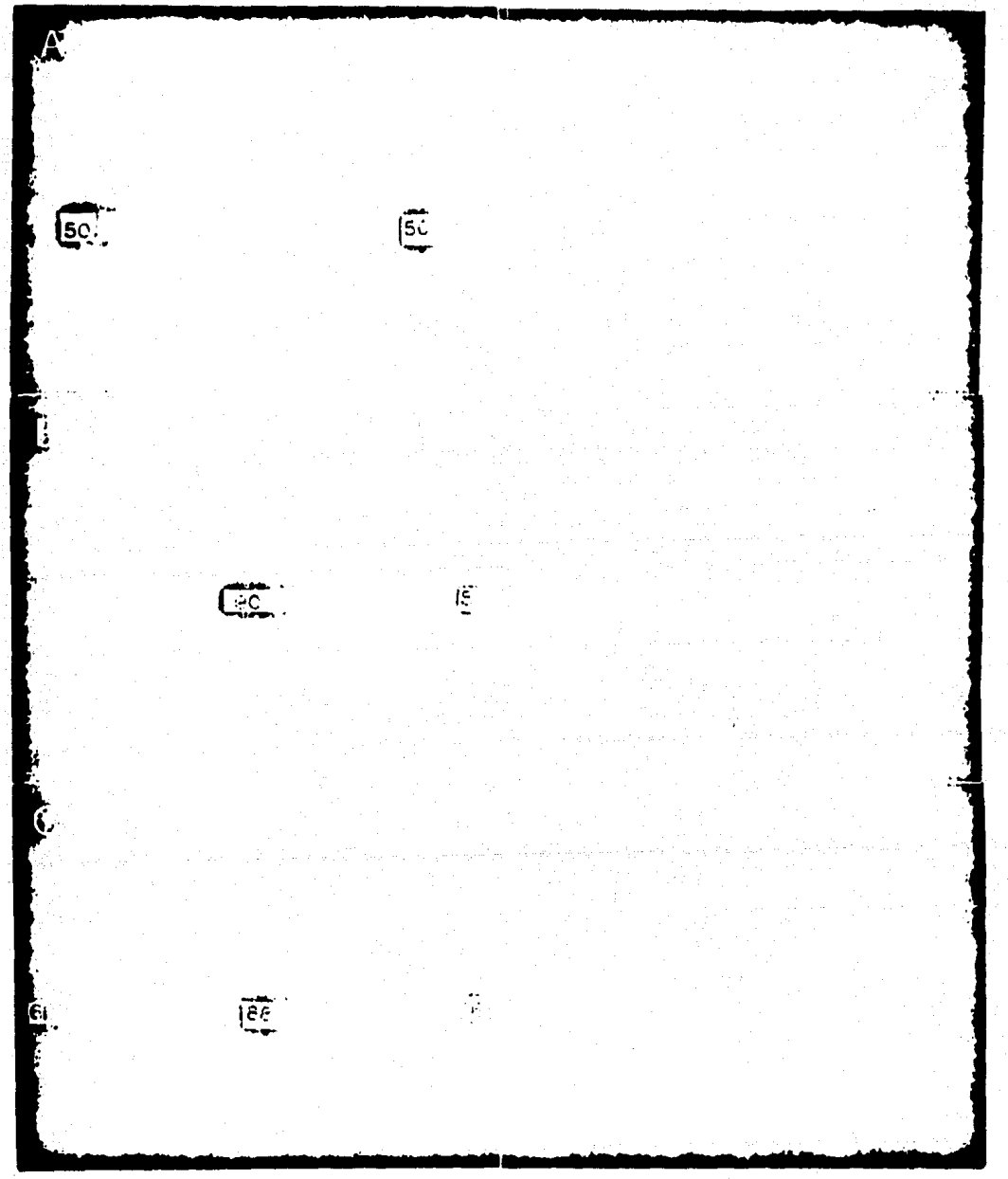
Como ejemplo mostraremos el archivo utilizado para la graficación de un caso con barreras y zanjas.

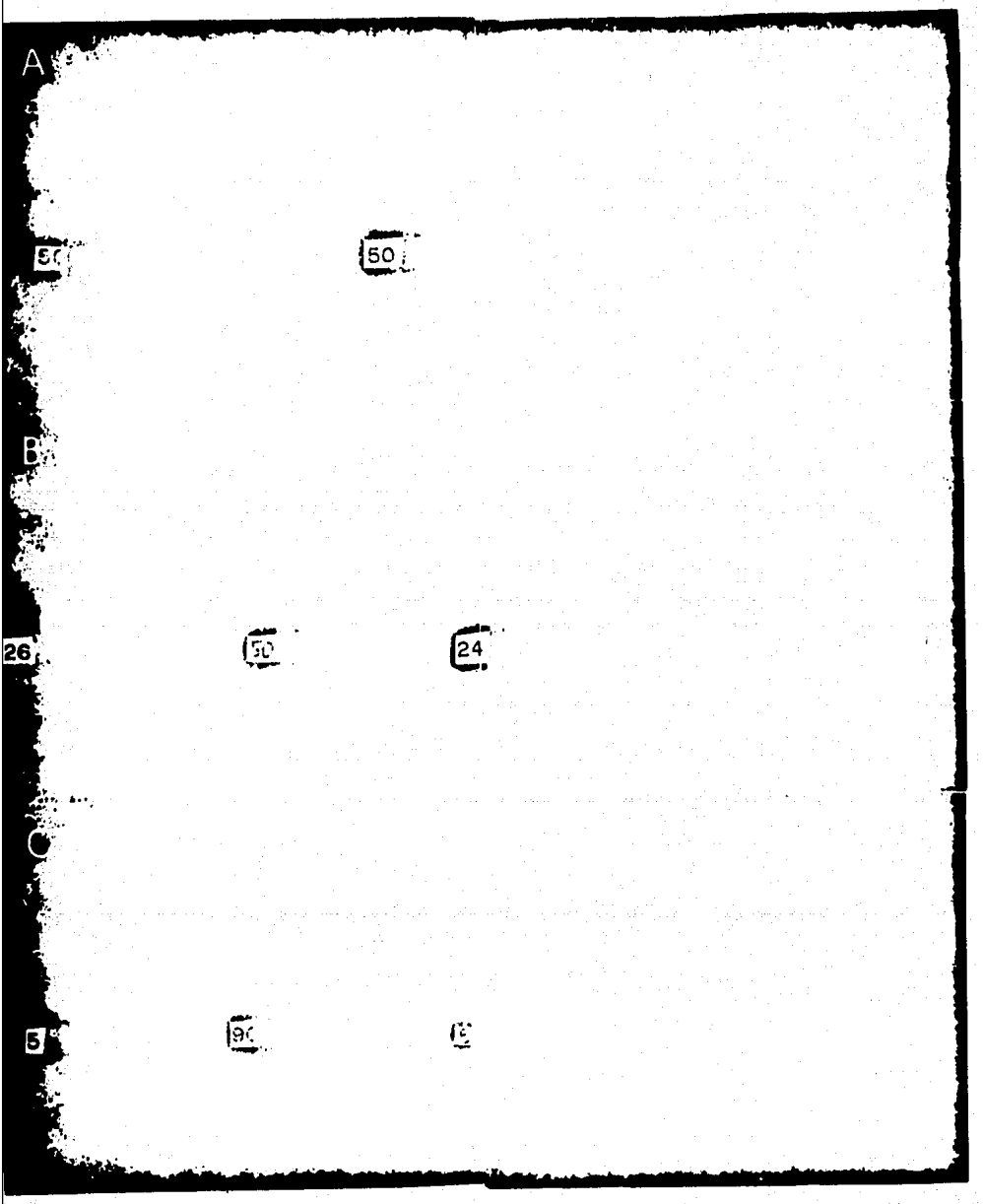
```
2                   % 2 obstáculos.
0,3                 % Primer obstáculo,
                    %   zanja poco profunda.
40,40,80,40         % Coordenadas de los
80,80 40,80         % 4 vértices de la zanja.
1,15                % Segundo obstáculo,
                    %   una barrera.
150,60,150,20       % Coordenadas de los extremos
                    % de la barrera.
180,80              % Posición de la presa.
4                   % La rana se re-orientará 4 veces.
20,50,20,51         % Se orientará y caminará
                    % de (20,50) a (20,51).
20,51,100,75        % Se orientará y caminará
                    % de (20,51) a (100,75).
100,75,150,30       % Se orientará y caminará
                    % de (100,75) a (150,30).
150,30,179,79       % Finalmente caminará
                    % de (150,30) a (179,79).
```

Puede parecer redundante el hecho de que se repita el punto final de una trayectoria de orientación con el punto inicial de la siguiente, pero en el caso en que la rana deba brincar una zanja debe especificarse tanto el punto donde se encuentra colocado el anfibio después del salto, así como el punto hacia donde se dirigirá posteriormente, en tal caso el punto final de la trayectoria de orientación previa es diferente del punto inicial despues del salto, por tal razón es necesario especificar las coordenadas de ambos puntos cada vez que exista un cambio en la ruta a seguir.

4.6.1 LISTADO Y RESULTADOS CROMEMCO

En el apéndice D se muestra el programa en Basic que se encarga de realizar la graficación en CROMEMCO, y se explica cada una de sus partes así como el objetivo de cada una de las rutinas específicas de SDI-GRAPHICS usadas en el programa; y en las páginas siguientes se encuentran fotografías de los resultados obtenidos sobre la pantalla de graficación.




A
50
50
B
26
50
24
C
5
90

A

C

4.7 BIBLIOGRAFIA

1) Algol on the B6700: a complete primer Vol. 1. Donald J. Gregory. Gregory Publishing Company. Sonoma, California. 1979.

2) Algol on the B6700: a complete primer Vol. 2. Donald J. Gregory. Gregory Publishing Company. Sonoma, California. 1979.

3) B7000/B6000 ALGOL Reference Manual. Burroughs Corporation. 1977.

4) Collet, T. Do toads plan routes?. A study of the detour behavior of bufovirids. J. Comp. Physiol. 146:261-267, 1982

5) SDI Graphics Software. Instruction Manual. 1980.

APENDICE A

APENDICE A

A.1 LENGUAJES DE PROGRAMACION EN LA INTELIGENCIA ARTIFICIAL

La investigación en Inteligencia Artificial involucra escribir en un lenguaje simbólico algoritmos que intentan lograr algún tipo de comportamiento inteligente. Esto último nos hace reflexionar, para asegurar que la herramienta más importante en Inteligencia Artificial son los lenguajes de programación, a partir de los cuales son implementados los algoritmos aludidos.

El diseño e implementación de los lenguajes de I.A. es una labor desprendida de la necesidad de utilizar la computadora, de manera que al momento de programarla, **no se pierda la visión del problema a resolver**, al tener que diseñar un programa a partir de una secuencia de pasos cuidadosamente ordenados.

Este tipo de situación se presenta al programar la computadora a través de lenguajes funcionales como son: FORTRAN, COBOL, PL/1, BASIC, etc., los cuales desde temprano se contemplaron insuficientes para los objetivos establecidos por la I.A. Es así que los lenguajes simbólicos de programación han tenido un papel preponderante en la historia de la I.A..

Existen dos grandes aportaciones logradas con estos lenguajes:

1) permiten la conveniente implementación y modificación de programas que demuestran y prueban algoritmos de la I.A.

2) son un vehículo para el pensamiento, permiten al usuario concentrarse sobre conceptos de alto nivel

Con frecuencia, nuevas ideas en I.A. son acompañadas por un nuevo lenguaje que resulta natural para estas ideas. Los lenguajes más renombrados, desarrollados hasta la fecha para aplicación de I.A. son: IPL, LISP, PLANNER, CONNIVER, SLISP, POP-2, SAIL, FUZZY y PROLOG.

APENDICE A

A.2 HISTORIA Y CARACTERISTICAS DE LISP

LISP (LIST PROCESSING) cuyo significado es 'procesamiento de listas' es uno de los lenguajes pioneros en la programación de algoritmos de I.A., surge en el año de 1958; fué creado por **John McCarthy** y su grupo de colaboradores en el **Instituto de Tecnología de Massachusetts (M. I. T.)**. A partir de entonces han surgido cantidad de versiones de este lenguaje, entre las más conocidas están:

UCI-LISP
INTER-LISP
V-LISP
MAC-LISP etc.

Estas versiones hacen variadas modificaciones sobre el LISP original, proporcionando con ello que no exista un lenguaje standard como el propuesto por McCarthy.

Desde su creación , LISP ha sido el principal lenguaje de programación en la I.A., la razón de ello es en parte histórica; LISP ha sido, y es actualmente, el lenguaje más trabajado y experimentado de los existentes en este campo y sigue siendo un vehículo natural en las investigaciones en I.A., debido a que existen características en LISP que son de importancia crítica en este tipo de tareas.

LISP difiere de manera relevante de otros lenguajes de programación en su estilo de describir instrucciones. En lugar de ser descritas como secuencias de pasos, los programas en LISP consisten en **FUNCIONES definidas con formato lógico matemático.** Cada función es representada como una lista, en donde el valor de su primer elemento es el nombre de la función y el valor de sus otros elementos son los argumentos. Una función en LISP puede manipularse a sí mismo o a otra(s) funcion(es) tan sencillamente como si fuese un dato más.

Una característica de LISP, que es única entre los lenguajes de programación de alto nivel y que resulta particularmente importante en trabajos de **I.A.**, es la representación de los programas mismos con la estructura de lista.

Otra característica de LISP es que su ambiente de programación es **altamente interactivo.** Es por ello que la **facilidad** de manipular programas por medio de otros programas en LISP, permite que sean escritos editores y correctores de errores (debuggers, trace, etc), en este **mismo lenguaje.**

LISP posee además el manejo sencillo y elegante de

lista de propiedades (property lists). Los símbolos en LISP son llamados **'átomos'** y cada átomo puede tener varias propiedades asociadas con él; es decir, el programador puede definir propiedades y asociarlas con cada átomo. Por ejemplo, es posible asociar a un átomo que represente un objeto, una propiedad llamada COLOR con los símbolos AZUL, AMARILLO, ROJO, etc., como valores.

A.3 LENGUAJE LISP

El lenguaje LISP fue diseñado principalmente para **procesamiento simbólico de datos**. Ha sido usado para cálculos simbólicos en cálculo diferencial e integral, teoría de circuitos eléctricos, lógica matemática, teoría de juegos, robótica, reconocimiento de formas y otros campos de la I.A. .

LISP difiere de los demás lenguajes de programación en tres características importantes. La primera es la naturaleza de los datos; en el lenguaje LISP, todos los datos están en forma de **expresiones simbólicas**, usualmente llamadas **expresiones-S**, estas son de longitud indefinida y tienen una estructura tipo árbol . En el sistema de programación LISP, la memoria disponible es usada para almacenar expresiones-S en forma de estructura de lista. Este tipo de organización de memoria libera al programador de la necesidad de alojar almacenamiento para los diferentes secciones de su programa.

La segunda diferencia importante es el lenguaje mismo, el cual especifica en que forma las expresiones-S serán procesadas. Es posible tener **FUNCIONES RECURSIVAS** de expresiones-S, debido a que la escritura de funciones recursivas de expresiones-S están a su vez fuera de la notación de expresion-S, a estas funciones se les denomina **expresiones metasimbólicas** o **expresiones-M**.

Por último, la tercera diferencia consiste en que LISP puede interpretar y ejecutar programas escritos en forma de expresiones-S. LISP al igual que un lenguaje de máquina y a diferencia de los lenguajes de alto nivel, genera programas de **rápida ejecución**.

A.4 EXPRESIONES SIMBOLICAS

El tipo más elemental de expresion-S es el átomo. En notación de Backus los siguientes símbolos son usados: "::=", "<", ">", y "!". La siguiente regla: <expresión-S> ::= <átomo> ! (<expresión-S> <expresión-S>),

significa que una expresión-S se define ya sea como un átomo, o un paréntesis izquierdo, expresión-S, expresión-S, paréntesis derecho. El símbolo '|' significa una 'ó', y los dos paréntesis encierran elementos de la sintáxis que esta siendo definida. A continuación se utiliza esta notación para la definición de un átomo:

```
<vacio>                 ::=
<menos>                 ::= -
<letra>                 ::= A|B|C|.....|X|Y|Z
<número>                ::= 0|1|2|.....|7|8|9
<parte-atómica>         ::= <vacio>|<letra><parte-atómica>
                            |<número><parte-atómica>
                            |<menos><parte-atómica>
<átomo>                 ::= <letra><parte-atómica>
```

ejemplos de átomos: a atomo-x ckd-91 color-bloque1

Los átomos son las entidades más pequeñas en LISP, su descomposición en caracteres carece de significado. La longitud máxima standard de un átomo es de 31 caracteres.

Todas las expresiones-S son construidas a base de átomos y los separadores de puntuación '(' y ')'. De acuerdo a la forma de Backus tenemos lo siguiente:

```
<expresión-S> ::=  <átomo>
                |(<expresión-S><expresión-S>)
                |(<expresión-S>...<expresión-S>)
```

Aunque los tres puntos (...) no forman parte de la notación de Backus aquí los utilizamos para evitar el nombramiento innecesario de tipos sintácticos; estos tres puntos significan que cualquier número de tipos simbólicos pueden ocurrir, incluyendo ninguno. De acuerdo con esto se establece que () es una expresión-S válida. Esta definición formal no contempla a constantes numéricas como elementos de una expresión-S, sin embargo frecuentemente en la práctica sucede que las constantes numéricas son elementos importantes en la programación de un algoritmo. Es por ello que proponemos las siguientes reglas para la definición de una expresion-S:

```
<signo>                 ::= <vacio>|+|-
<constante>             ::= <número>|<número><constante>
<constante-real>        ::= <signo><constante>.<constante>
<constante-entera>      ::= <signo><constante>
<constante-numérica>    ::= <constante-real>|<constante-entera>
<expresión-S>           ::= átomo
                            |<constante-numérica>
                            |(<expresión-S><expresión-S>)
                            |(<expresión-S>...<expresión-S>)
```

ejemplos de expresiones-S:

- ATOMO-X

  (A-1 B C 1 2 3)

  (1 (B C D) (A B))

- (LET (1 2 3 (X-1)))

A.5 ATOMOS ESPECIALES

En LISP existen dos átomos con valor definido, independientemente de la máquina o la versión de LISP que se use;se trata de 'T' y 'NIL',a los que les podemos asociar los valores lógicos: TRUE y FALSE que tienen otros lenguajes. El valor asociado a estos átomos es el nombre mismo, es decir T vale T y NIL vale NIL. La inicialización de átomos es de acuerdo a la máquina y el LISP empleado por ésta, por ejemplo LISP de APPLE inicializa todos los átomos a NIL, en cambio LISP de CROMEMCO no lo hace, dando con ésto una ventaja al programador ya que le permitirá hacer un uso totalmente estructurado del lenguaje. El trabajo aquí presentado fué desarrollado en la microcomputadora CROMEMCO. Como mencionamos al inicio de este tema, LISP es un lenguaje interactivo, y funciona de la siguiente forma:

CICLO: LEE EXPRESION-S

EVALUA EXPRESION-S

IMPRIME LA EVALUACION DE LA EXPRESION-S

GO TO CICLO

Por ejemplo si nosotros tecleamos una T, nos devolverá en la pantalla o teletipo una T, para NIL retornará NIL, cuando se teclee un número devolverá el mismo número. En cambio si tecleamos una X nos tratará de devolver el valor que contiene éste átomo y como éste no ha sido inicializado, es decir no tiene valor alguno, el resultado será un error, pero si en lugar de X se introduce 'X LISP responderá con X. Lo que nos sugiere pensar que el apóstrofe tiene la función de distinguir aquello que deberá interpretarse como una constante, definición válida por el momento, ya que más adelante se explicará detenidamente su interpretación.

NOTA:

En lo sucesivo tomaremos la convención de que "=>" significa lo que devuelve LISP.

APENDICE A

Como ya se explicó anteriormente, LISP es un lenguaje interactivo; por lo tanto a cualquier petición del programador LISP contestará siempre; cuando estas peticiones esten mal planteadas, como respuesta tendremos un mensaje de error.

Ejemplos:

T => T

NIL => NIL

X => ERROR: UNBOUND-VARIABLE

'X => X

'LISP => LISP

'CADENA-DE-CARACTERES => CADENA-DE-CARACTERES

A.6 FUNCIONES ARITMETICAS

Enseguida se hace una introducción a las funciones aritméticas, utilizando para ello átomos numéricos. Obsérvese que las funciones son expresiones-S.

El nombre de la función para realizar una suma es: ADD o PLUS o bien basta el símbolo "+"; ejemplo:

(ADD 5 5 5) retornará un valor de 15

siendo la forma general (ADD arg-1 arg-n); donde arg-n indica que pueden existir 'n' sumandos.

A continuación se describirá la forma general de cada una de las operaciones aritméticas; nótese que la única función de 'n' argumentos es la suma, ya que las demás solo tienen uno o dos argumentos; con '*' distinguimos las utilizadas por CROMEMCO-LISP.

APENDICE A

OPERACION	FORMA GENERAL	EJEMPLO
suma	(ADD arg-1 arg-n)* (PLUS arg-1 arg-2) (+ arg-1 arg-2)*	(ADD 5 -6) => -1
incrementa uno	(ADD1 arg-1)*	(ADD1 9) => 10
resta	(SUB arg-1 arg-2)* (DIFFERENCE arg-1 arg-2) (- arg-1 arg-2)*	(SUB 3 1) => 2
decrementa uno	(SUB1 arg-1)*	(SUB1 4) => 3
multiplicación	(MUL arg-1 arg-2)* (TIMES arg-1 arg-2) (* arg-1 arg-2)*	(MUL 5 -4) => -20
división	(DIV arg-1 arg-2)* (QUOTIENT arg-1 arg-2) (/ arg-1 arg-2)*	(DIV 5 2) => 2
residuo	(REM arg-1 arg-2)* (MOD arg-1 arg-2)	(REM 5 2) => 1
valor absoluto	(ABS arg-1)*	(ABS -2) => 2

La manera de efectuar operaciones algebráicas se realiza utilizando notación de CAMBRIDGE: por ejemplo:

ALGEBRA	LISP
$\frac{X + Y + Z}{5}$	(DIV (ADD (ADD X Y) Z) 5)
$\frac{(X*Y) - 3}{(A + B + C/D)}$	(/(-(* X Y) 3)(+ A B (/ C D)))
C*(A + B + K)**2	(* C (*(+ A B K)(+ A B K)))

NOTA: ** indica exponenciación

A.7 FUNCIONES ELEMENTALES

En esta sección se hace una introducción a las funciones elementales de LISP; a continuación se describen: **SETQ, CONS, CAR, CDR, C...R, ATOM Y EQ.**

(SETQ <arg-1> <arg-2>)

SETQ permite asignar valores numéricos o expresiones-S a variables; las variables siguen las mismas reglas gramaticales que un átomo. De esta forma

Ejemplo:

(SETQ A 10) => 10

A => 10

(SETQ LNG '(LISP BASIC C)) => (LISP BASIC C)

LNG => (LISP BASIC C)

'LNG => LNG

(CONS <arg-1> <arg-2>)

CONS se utiliza para construir expresiones-S de otras expresiones-S.

Ejemplos:

(CONS 'A 'B) => (A B)

(CONS '(A B) 'C) => ((A B) C)

(CONS 'APL LNG) => (APL LISP BASIC C)

Nótese que la función CONS no altera el contenido de ninguno de sus argumentos. Es la función más común para generar nuevas expresiones-S a partir de otras expresiones-S más pequeñas. La siguiente expresión-S si alteraría el valor del átomo LNG:

(SETQ LNG (CONS 'APL LENGUAJES)) => (APL LISP BASIC C)

LNG => (APL LISP BASIC C)

(CAR <arg-1>)

CAR obtiene el primer elemento de la expresion-S dada como argumento. El CAR de un simbolo atómico no lista es un

error.

Ejemplos:

(CAR '(1 2 3 4)) => 1

(CAR '((A B) C D)) => (A B)

(CAR LNG) => APL

(CAR 'LNG) => FATAL ERROR

(CAR (CONS 'A 'B)) => A

(CDR <arg-1>)

CDR obtiene el valor del argumento mismo pero sin incluir el primer elemento: se puede decir que ésta función es el complemento de la función CAR. El CDR de un símbolo atómico no lista es un error.

Ejemplos:

(CDR '(A B C D)) => (B C D)

(CDR '((A B C) D)) => (D)

(CDR '(A)) => NIL

(CDR LNG) => (LISP BASIC C)

(CDR 'LNG) => ERROR

Una vez conocidas las funciones CAR y CDR, definiremos el significado del apóstrofe, el que hasta ahora a sido interpretado como la indicación de una expresión-S constante.

Tomemos como ejemplo lo siguiente: tenemos una lista (A B C) y deseamos extraer el segundo elemento de dicha lista, el problema resulta sencillo ya que solo basta aplicar primeramente la función CDR y sobre el resultado aplicar la función CAR, es decir: (CAR (CDR (A B C))), pero existe un grave inconveniente, debido a que LISP trataría de evaluar una función A con argumentos B y C, y como dicha función aún no ha sido definida generaría un error,es por ello que necesitamos una señal que nos indique que (A B C) es una expresión-S que no necesita evaluarse y tal señal es el apóstrofe. La colocación del apóstrofe es también muy importante, ya que el resultado esperado depende de dicha posición.

Ejemplos:

(CAR '(CDR (A B C))) => CDR

(CAR (CDR '(A B C))) => B

(CAR (CDR (A B C))) => ERROR: FUNCION A NO DEFINIDA

(CAR (CDR LNG)) => LISP

LNG => (APL LISP BASIC C)

(C...R <arg-1>)

C...R permite realizar hasta en 3 ocasiones las funciones CAR y CDR en cascada, existiendo 12 diferentes combinaciones.

Ejemplos:

(CADR <arg-1>) equivale a (CAR (CDR <arg-1>))

(CAAAR <arg-1>) equivale a (CAR (CAR (CAR <arg-1>)))

(CDDAR <arg-1>) equivale a (CDR (CDR (CAR <arg-1>)))

Las siguientes dos funciones elementales son conocidas como predicados debido a que devuelven como resultado T o NIL.

(ATOM <arg-1>)

ATOM prueba si su argumento es un elemento átomico, si es este el caso devuelve T, de lo contrario devuelve NIL.

Ejemplos:

(ATOM LNG) => NIL

(ATOM 'LNG) => T

(ATOM '(A B)) => NIL

(EQ <arg-1> <arg-2>)

EQ realiza una prueba de igualdad entre elementos átomicos solamente. En la sección de predicados se describe la función que compara elementos no-átomicos.

Ejemplos:

(EQ 'A 'B) => NIL

(SETQ C 5) => 5

(EQ C (ADD1 4)) => T

(EQ NIL '()) => T

A.8 FUNCIONES PARA MANEJO DE LISTAS

En LISP existen funciones cuyo objetivo es manipular listas de elementos átomicos. Se explicará el funcionamiento de las siguientes funciones: **LIST, APPEND, REVERSE, LENGTH, NTH Y ASSOC.**

(LIST <arg-1> <arg-2> ... <arg-n>)

LIST permite generar una lista, cuyos elementos son las evaluaciones de todos sus argumentos.

Ejemplos:

(SETQ A '(1 2 3 4)) => (1 2 3 4)

(SETQ B '(K L M N)) => (K L M N)

(SETQ C 'A) => A

(LIST A B C) => ((1 2 3 4) (K L M N) A)

(LIST 'A (CAR B)) => (A K)

(LIST (CONS 'A 'B) C) => ((A B) A)

(LIST (LIST B) C) => (((K L M N)) A)

(APPEND <arg-1> <arg-2>)

APPEND permite generar una lista que contiene todos los componentes de ambos argumentos.

Ejemplos:

(APPEND A B) => (1 2 3 4 K L M N)

(APPEND (LIST C) A) => (A 1 2 3 4)

(APPEND (APPEND A B) (LIST C)) => (1 2 3 4 K L M N A)

(APPEND (LIST (CAR A)) (LIST (CAR B))) => (1 K)

(REVERSE <arg-1>)

REVERSE tiene por objeto retornar en forma contraria el contenido de una lista

Ejemplos:

(REVERSE A) => (1 3 2 1)

(REVERSE (LIST A B)) => ((K L M N) (1 2 3 4))

(REVERSE (APPEND (LIST C) A)) = (A 3 2 1 A)

(REVERSE '(S S T)) => (T S R)

(LENGTH <arg-1>)

LENGTH obtiene el número de componentes directos de una lista.

Ejemplos:

(LENGTH A) => 4

(LENGTH (LIST A B)) => 2

(LENGTH (LIST C)) => 1

(LENGTH (APPEND A B)) => 8

(NTH <arg-1> <arg-2>)

NTH obtiene el elemento n de una lista, n esta dado por el segundo argumento y la lista por el primero

Ejemplos:

(NTH 3 7) => L

(NTH 2 (NTH A 2)) => L

(NTH A (ADD (CAR A) (NTH A 2))) => 3

(ASSOC <arg-1> <arg-2>)

ASSOC permite encontrar la sublista del argumento dos cuyo primer elemento es igual al argumento uno. Si esto

condición no se cumple la función retorna NIL. Esta función es útil para el manejo de Bases de Datos.

Ejemplos:

(SETQ L '((A B) (D F) (K L))) => ((A B) (D F) (K L))

(ASSOC 'D L) => (D F)

(ASSOC 'Q L) => NIL

A.9 ENTRADA Y SALIDA

Uno de los mayores problemas en LISP es su entrada y salida de información, debido a que es poco estética para llevarla a cabo se usan cuatro funciones: **PRINT, PRINO, TERPRI Y READ.**

(PRINT <arg-1>)

PRINT permite imprimir un sólo resultado de una expresión-S, si se requiere imprimir varios resultados, se necesita hacer una pequeña implementación con la expresión LIST.

Ejemplos:

(SETQ ALFA '(A B C)) => (A B C)

(SETQ BETA '(1 2 3)) => (1 2 3)

(SETQ GAMA 'MC-22) => MC-22

(PRINT ALFA) => (A B C)

(PRINT (LIST 'VALOR-ALFA ALFA)) => (VALOR-ALFA (A B C))

(PRINT (LIST ALFA GAMA)) => ((A B C) MC-22)

(PRINO <arg-1>)

PRINO funciona al igual que PRINT; solo que además detiene el cursor al final de la impresión; en cambio PRINT coloca el cursor después de imprimir al principio de la siguiente línea.

(TERPRI)

TERPRI es la función que nos permite hacer un salto de

renglón.

(READ)

READ es la indicación para leer datos del usuario por la terminal. Si se introduce un dato tipo lista, bastará con el último paréntesis para que LISP haga la asignación, si el dato a introducir es un átomo entonces necesitará además un RETURN en el teclado.

Ejemplos:

```
(SETQ A (READ)) ; LISP espera el dato a asignar.
(1 2 3)         ; dato introducido.
(1 2 3)         ; contesta con la evaluación de A.
(SETQ X (READ)) ; LISP espera el dato a asignar.
MC-22 <RETURN>  ; dato seguido por un RETURN.
MC-22           ; contesta con la evaluación de X.
```

A.10 LISTA DE PROPIEDADES

Al principio de éste capitulo, se mencionó que una característica importante de LISP es el uso de **listas de propiedades (property list)**.

Las ventajas que se tienen al manejar lista de propiedades, se enumeran a continuación:

1) Permiten asociar una serie de características a un átomo.

2) Tales características o propiedades pueden ser creadas, consultadas, modificadas y destruidas.

3) Permiten implementar pequeñas Bases de Datos, tipo relacional.

Las expresiones que nos permiten usar propiedades son: **PUTPROP, GETPROP, REMPROP** y **PRINTPROP**. En otros lenguajes LISP's a las primeras dos funciones se les conoce como **PUT** y **GET**.

(PUTPROP <arg-1> <arg-2> <arg-3>)

Donde: <arg-1> es el átomo.

<arg-2> es la propiedad.
<arg-3> es el valor de la propiedad.

PUTPROP crea la propiedad de un átomo y le asigna valor, si una vez creada esta propiedad, se quisiera modificar el valor de dicha propiedad, utilizaríamos otro PUTPROP similar, con el nuevo valor de la propiedad; esta función retorna como resultado, el valor de la propiedad.

Ejemplos:

```
(PUTPROP 'ANA 'PROFESION  FISICO) => FISICO

(PUTPROP 'ANA 'IDIOMAS '(ALEMAN FRANCES INGLES))
                        => (ALEMAN FRANCES INGLES)

(PUTPROP 'ANA 'EDAD '24) => 24

(PUTPROP 'ANA 'PROFESION 'FISICO-NUCLEAR)
                          => FISICO-NUCLEAR
```

(GETPROP <arg-1> <arg-2>)

Donde: <arg-1> es el átomo
<arg-2> es la propiedad.

GETPROP nos permite consultar una determinada propiedad de un átomo.

Ejemplos:

```
(GETPROP 'ANA 'IDIOMAS) => (ALEMAN FRANCES INGLES)

(GETPROP 'ANA 'PROFESION) => FISICO-NUCLEAR
```

(REMPROP <arg-1> <arg-2>)

donde: <arg-1> es el átomo
<arg-2> es la propiedad.

REMPROP destruye la propiedad (incluyendo el valor de ésta) de un cierto átomo.

Ejemplos

```
(REMPROP 'ANA 'EDAD) => 24
```

Si en este momento quisiéramos consultar una propiedad destruida, el valor retornado por LISP, sería NIL.

Ejemplo:

(GETPROP 'ANA 'EDAD) => NIL

(PLIST <arg-1>)

donde: <arg-1> es el átomo.

PRINTPROP es llamado en CROMEMCO: **PLIST**, sirve para obtener la descripción de todas las propiedades de un átomo, con su respectivo valor.

Ejemplo:

(PLIST 'ANA) => ((PROFESION FISICO-NUCLEAR)
(IDIOMAS (ALEMAN FRANCES INGLES)))

A.11 META-EXPRESIONES EN LISP

Hasta el momento solo hemos manejado el concepto de expresiones-S u ó funciones elementales. Ahora discutiremos las características de una Meta-expresión, también denominada **FUNCION**. A continuación se muestra la forma general de una función en LISP.

```
(DE <NOMBRE> (<ARGUMENTO>...<ARGUMENTO>)
                <EXPRESION-S>
                <EXPRESION-S>
                      .
                      .
                      .
                <EXPRESION-S>(*)
  );END-<NOMBRE>
```

1) **Nombres de función y nombres de variables (incluyendo las utilizadas como argumentos) son definidos de igual manera que los símbolos atómicos. Para definir una función se utiliza la palabra DE, en el caso de CROMEMCO, en otros LISPs la palabra varía entre las siguientes formas: DEFINE, DEFINEQ, DEFUN, etc.**

2) **Los argumentos de la función son agrupados entre paréntesis y separados entre sí por espacios en blanco. Al definir una función tales paréntesis deberán incluirse aunque la función no tenga argumentos. Al invocar una función el paso de argumentos se realiza por valor.**

3) **La composición de una función es mediante conJuntos anidados de expresiones-S. Toda función devuelve siempre un resultado, por lo general tal resultado es la evaluación de la última expresión-S, indicada por (*) en la forma general.**

NOTA:

Los comentarios en LISP son reconocidos al usar ';', existen dos alternativas al usarlos. La primera es que al encontrarse un punto y coma, el resto de la línea sea interpretado como comentario; la segunda alternativa es que el comentario esté limitado por un punto y coma en cada extremo (esto sucede al usar el IF-THEN-ELSE). En la forma general anterior fué usado un comentario al final de la función.

Tomando en consideración estas reglas deducimos que el siguiente procedimiento representa la definición de una función o bien de una Meta-expresión:

```
(DE CUARTO (X) (CAR(CDR(CDR(CDR X)))))
```

donde el nombre es CUARTO, la variable-dummy es X y cuyo objetivo es obtener el cuarto elemento de una lista. La forma de ejecutarla es invocándola con el nombre y el parámetro, por ejemplo:

```
(CUARTO '(A B C D E F)) => D

(SETQ L '(1 3 4 7 2)) => (1 3 4 7 2)

(CUARTO L) => 7

(ADD (CUARTO L) 5) => 12
```

La colocación de los argumentos es muy importante ya que el resultado esperado depende de la posición correcta de dichos argumentos. Veamos el siguiente ejemplo:

```
(DE F-1 (X Y)
        (SETQ X (ADD1 X))
        (SETQ Y (SUB1 Y))
        (CONS X Y)
);END-F-1

(SETQ A 8) => 8

(SETQ B 5) => 5

(F-1 A B) => (6 7)

A => 8
```

```
B => 5

(F-1 B A) => (5 8)

A => 8

B => 3
```

Como se puede observar el paso de argumentos a la función es realizado por valor, aunque aparentemente en la función F-1 tales argumentos sufren cambios.

Ahora veamos el siguiente ejemplo:

```
(DE F-2 ()
    (SETQ X (ADD1 X))
    (SETQ Y (SUB1 Y))
    (CONS X Y)
);END-F-2

(SETQ X 2) => 2

(SETQ Y 4) => 4

(F-2) => (3 3)

X => 3

Y => 3
```

La función anterior no tiene argumentos y sí en cambio modifica los valores asignados a las variables X y Y, debido a que éstas variables son globales a la función. En CROMEMCO existe una opción que nos permite definir variables locales a una función, esta opción se realiza colocando la palabra %AUX seguida de las variables locales a declarar, en la lista de argumentos. Veamos esta nueva organización de una función, con esta opción:

```
(DE <NOMBRE> (<ARG>...<ARG> [%AUX <VAR>...<VAR>])
             <EXPRESION-S>
             <EXPRESION-S>
                  .
                  .
                  .
             <EXPRESION-S>
);END-<NOMBRE>
```

Las funciones anteriormente descritas son ilustrativas pero a su vez limitadas y poco interesantes, en cambio una gran cantidad de funciones pueden ser definidas utilizando expresiones de condición, las cuales serán definidas en los temas que siguen.

A.12 PREDICADOS

Por predicado se conoce a toda aquella función cuyo valor es falso (NIL) o verdadero (T). Hasta el momento solo conocemos dos predicados: **ATOM** y **EQ**; en esta sección explicaremos los predicados: **EQUAL, NULL, LISTP, NUMBERP, BOUNDP, MINUSP, GE, GT, ZEROP, LE, LT, NOT, AND** y **OR** que nos serán de gran utilidad en las secciones posteriores. Para los predicados: MINUSP, GE, GT, ZEROP, LE y LT el valor T no existe como tal, en cambio el valor NIL si, **CROMEMCO-LISP tiene la característica de que todo aquel valor de un predicado que sea diferente de NIL es considerado como verdadero, es decir con valor T**; es así que un número, un átomo etc. es considerado con valor T.

(EQUAL <arg-1> <arg-2>)

EQUAL es verdadero si el valor de sus argumentos son iguales.

Ejemplos:

```
(SETQ N '(A 1 2 B (D E))) => (A 1 2 B (D E))

(SETQ M 6) => 6

(SETQ L '(A B)) => (A B)

(EQUAL (CADR L) (CAR(CDDDR N))) => T

(EQUAL '(A B) L) => T

(EQUAL 'A (CAR L)) => T

(EQUAL M (CADR L)) => NIL

(EQUAL M (DIV 13 2)) => T

(EQUAL (SUB M 2) (CADR N)) => NIL
```

(NULL <arg-1>)

NULL es verdadero si el valor de su argumento es NIL o lista vacia.

Ejemplos:

```
(NULL L) => NIL

(NULL ()) => T

(NULL (CADR(CDDR N))) => NIL
```

(LISTP <arg-1>)

LISTP es verdadero si el valor de su argumento es una lista.

Ejemplos:

(LISTP L) => T

(LISTP B) => NIL

(LISTP (CAR N)) => NIL

(LISTP (CONS 'A NIL)) => T

(NUMBERP <arg-1>)

NUMBERP es verdadero si el valor de su argumento es numérico.

Ejemplos:

(NUMBERP (CADR N)) => T

(NUMBERP L) => NIL

(NUMBERP (CAR L)) => NIL

(BOUNDP '<arg-1>)

BOUNDP es verdadero si el argumento no evaluado (nótese el apóstrofe antes del argumento), ha sido inicializado, es decir tiene valor.

Ejemplos:

(BOUNDP 'N) => T

(BOUNDP 'Y) => F

(BOUNDP 'M) => T

(MINUSP <arg-1>)

MINUSP es verdadero si el valor de su argumento es negativo.

Ejemplos:

(MINUSP (CADR N)) => NIL

(MINUSP (SUB -4 (CADR N))) => -5

(MINUSP (REM -5 2)) => -1

(GE <ars-1> <ars-2>)

GE es verdadero si el valor de su primer argumento es mayor o igual al valor de su segundo argumento.

Ejemplos:

(GE 5 (CADR N)) => 5

(GE (ADD1 M) 7) => 7

(GE (CADR N) M) => NIL

(GT <ars-1> <ars-2>)

GT es verdadero si el valor de su primer argumento es mayor al valor de su segundo argumento.

Ejemplos:

(GT (ADD1 M) 7) => NIL

(GT 7 (CADR N)) => 7

(GT M (+ 2 (CADR N))) => 6

(ZEROP <ars-1>)

ZEROP es verdadero si el valor de su argumento es cero.

Ejemplos:

(ZEROP M) => NIL

(ZEROP (REM M 2)) => 0

(LE <ars-1> <ars-2>)

LE es verdadero si el valor de su primer argumento es menor o igual al valor de su segundo argumento.

Ejemplos:

(LE M 2) => NIL

(LE (CADR N) M) => T

(LE M 6) => 6

(LT <arg-1> <arg-2>)

LT es verdadero si el valor de su primer argumento es menor al valor de su segundo argumento.

Ejemplos:

(LT M 6) => NIL

(LT 0 (CADR N)) => 0

(LT (ADD1 M) 8) => T

(NOT <arg-1>)

NOT es verdadero si su argumento es falso y falso en caso de que su argumento sea verdadero.

Ejemplos:

(NOT T) => NIL

(NOT NIL) => T

(NOT (ATOM L)) => T

(NOT (ZEROP (REM M 2))) => NIL

(AND <arg-1> <arg-2> ... <arg-n>)

Los argumentos de **AND** son evaluados en secuencia, de izquierda a derecha, existen dos alternativas, que se encuentre el primer argumento cuyo valor sea falso; entonces el valor de AND será falso, pero si se encontró que ninguno de los valores de los argumentos fué falso, entonces el valor de AND es verdadero.

Ejemplos:

(AND (LISTP L) (NUMBERP M)) => T

(AND (NUMBERP (CADR L)) (ZEROP (REM M 2))) => NIL

(AND (GT M 1) (NULL N) (LISTP (CONS M NIL))) => NIL

(AND (LISTP M) (NUMBERP (CADR L))) => NIL

(OR <arg-1> <arg-2> ... <arg-n>)

Los argumentos de **OR** son evaluados en secuencia, de izquierda a derecha, existen dos alternativas: que se encuentre el primer argumento cuyo valor sea verdadero, entonces el valor de OR será verdadero, pero si se encuentra que ninguno de los valores de los argumentos es verdadero, entonces el valor de OR es falso.

Ejemplos:

```
(OR (LISTP L) (NUMBERP M)) => T

(OR (NUMBERP (CADR L)) (ZEROP (REM M 2))) => T

(OR (GT M 1) (NULL N) (LISTP (CONS M NIL))) => T

(OR (LISTP M) (NUMBERP (CADR L))) => NIL
```

A.13 EXPRESIONES CONDICIONALES

Las expresiones condicionales dan a LISP una gran ventaja, sobre los demás lenguajes, en esta sección describiremos la expresión **COND** y la expresión **IF-THEN-ELSE**.

Iniciaremos describiendo la forma general de la expresión condicional **COND**:

```
(COND (<Pred-1> <exp-S> [<exp-S>...<exp-S>])
      (<Pred-2> <exp-S> [<exp-S>...<exp-S>])
      (<Pred-3> <exp-S> [<exp-S>...<exp-S>])
          .         .           .
          .         .           .
          .         .           .
      (<Pred-n> <exp-S> [<exp-S>...<exp-S>])
);END-COND
```

Donde cada expresión <Pred-i> es un predicado. Conviene resaltar que todo aquel predicado cuyo valor sea distinto de **NIL** es interpretado como verdadero al igual que uno T. La expresión condicional funciona de la siguiente forma: analiza predicados, de <Pred-1> a <Pred-n>; y cuando encuentre el primer <Pred-i> que sea verdadero entonces se ejecuta(n) la(s) expresion(es) simbólica(s) <exp-S> que sigue(n) a continuación. Se observa que deberá existir por lo menos una <exp-S>, ya que los paréntesis cuadrados los utilizamos para designar que pueden existir opcionalmente cero o más <exp-S>; retornando finalmente como resultado la evaluación de la última expresión simbólica ejecutada.

En caso de que ninguno de los predicados <Pred-i> sea

verdadero, entonces el valor de la expresión completa es NIL.

Conviene aclarar que cada <pred-i> y cada <exp-S> pueden ser una función, una composición de funciones o a su vez pueden ser otra expresión condicional.

A continuación se sugiere utilizar la siguiente organización para una expresión condicional.

```
(COND (<pred-1> <exp-S> [<exp-S>...<exp-S>])
      (<pred-2> <exp-S> [<exp-S>...<exp-S>])
           .         .          .
           .         .          .
           .         .          .
      (<pred-n> <exp-S> [<exp-S>...<exp-S>])
      (   T     <exp-S> [<exp-S>...<exp-S>])
);END-COND
```

De esta manera prevenimos el caso de que ninguno de los predicados sea verdadero, ya que de esta forma obligamos a tener siempre una alternativa, como consecuencia de que T es verdadero por definición.

A continuación se define la forma general de la expresión condicional **IF-THEN-ELSE**.

```
(IF <predicado>
    ;THEN; <expresión-S>
    ;ELSE; <expresión-S>
           <expresión-S>
                .
                .
                .
           <expresión-S>
);END-IF
```

Esta expresión funciona de la siguiente forma: si el predicado es verdadero sólo puede ejecutar una sola expresión-S, si por lo contrario, resulta falso puede ejecutar una o más expresiones-S.

A continuación se plantea un problema, que sucede si necesitamos hacer n-expresiones-S tanto en el 'then', como en el 'else' ? bien, utilizando la forma general del IF-THEN-ELSE, no será posible, este inconveniente se soluciona si utilizamos la expresión **PROGN** o **PROG1**.

La forma general de la funciones PROGN y PROG1 es la misma, por eso razón solo describiremos una sola:

(PROGN <exp-S> [<exp-S> ... <exp-S>])

Tanto PROGN como PROG1, nos permiten agrupar n-expresiones-S, PROGN devuelve como resultado el valor de la última expresión ejecutada; en cambio PROG1, devuelve el valor de la primera expresión simbólica ejecutada.

Entonces el problema queda solucionado de la siguiente forma:

```
(IF <predicado>
    ;THEN; (PROGN <expresión-S>
                  <expresión-S>
                        .
                        .
                        .
                  <expresión-S>
           );END-PROGN
    ;ELSE; <expresión-S>
           <expresión-S>
                  .
                  .
                  .
           <expresión-S>
);END-IF
```

A.14 FUNCIONES CONDICIONALES

La aplicación principal de las expresiones condicionales está en la definición de funciones recursivas.

Ejemplo No.1:

```
(DE FACT (N)
    (COND ((EQ N 0) 1)
          (T (* N (FAC (SUB1 N))))
    );END-COND
);END-FACT
```

Se puede observar que esta función realiza la operación factorial de un número, utilizando la definición matemática N!=N*(N-1)!.

```
(FACT 5) => 120

(SETQ N 4) => 4

(FACT N) => 24
```

Ejemplo No.2:

```
(DE MEMBER (A L)
    (COND ((NULL L) NIL)
          ((EQ (CAR L) A) T)
          (T   (MEMBER A (CDR L))))
)(END-COND
)(END-MEMBER
```

Este ejemplo muestra a la función MEMBER, la cual devuelve un valor verdadero si el elemento A se encuentra en la lista L; ver ejemplo:

```
(MEMBER 'C '(A B C D E F)) => T

(SETQ LISTA '(A B C)) => (A B C)

(SETQ ELEM 'B) => B

(MEMBER ELEM LISTA) => T

ELEM => B

LISTA => (A B C)
```

A.15 EXPRESION ITERATIVA DO

LISP también ofrece un mecanismo iterativo, que nos permite ejecutar una serie de expresiones-S, un determinado número de veces. Con esto evitamos el uso de instrucciones incondicionales de transferencia (GO TO's) y además podemos aplicar más ampliamente los conceptos de la programación estructurada.

A continuación mostramos la forma general de la expresión iterativa DO:

```
(DO ((<v-1> <va-i> <va-s>) ... (<v-n> <va-i> <va-s>))
    ((<pred-1> <exp-S> ... <exp-S>)
       .        .          .
       .        .          .
       .        .          .
     (<pred-n> <exp-S> ... <exp-S>))
     <exp-S> ; c
     <exp-S> ; u  d
        .    ; e  e  D
        .    ; r  l  O
        .    ; P.
     <exp-S> ; O
)(end-do
```

APENDICE A

Para su fácil comprensión, dividiremos el DO en tres partes.

La primera es la sección de definición de variables locales (v-i), con su respectivo valor inicial (ve-i) y el siguiente valor que adquiere (vs-i) en cada iteración. Como se observa esta primera parte o sección, aun cuando no tuviese variables locales, deberá existir como lista vacía.

La segunda parte prueba los predicados, secuencialmente de izquierda a derecha, de tal manera que cuando encuentre el primero verdadero, ejecutará las expresiones-S que aparecen a continuación, de otra forma seguirá iterando sobre las expresiones-S que forman el cuerpo del DO.

La tercera parte consiste en una serie de expresiones-S, que constituyen el cuerpo del DO; puede existir el caso de que no exista cuerpo en un DO, al no existir expresiones-S. Cabe aclarar que tales expresiones-S pueden ser: expresiones condicionales, funciones o a su vez expresiones iterativas.

Definiremos ahora el funcionamiento de la expresión DO, primero ejecuta la definición o inicialización de variables, posteriormente analiza los predicados que tienen el papel de condiciones de salida, en caso de que ninguno de los predicados sea verdadero, ejecuta uno a uno de las expresiones-S que conforman el cuerpo del DO. Posteriormente ejecuta la asignación de los nuevos valores de las variables, analiza los predicados etc, y así sucesivamente itera hasta que existe una condición de salida verdadera.

Observando detenidamente la forma general, podemos concluir que un **DO-WHILE** puede implementarse así:

```
(DO ()
    (((NOT <Pred>)))
    <exp-S>
    <exp-S>
       .
       .
       .
    <exp-S>
);end-do
```

Ejemplos:

```
(DE FACT (N)
    (DO ((N N (SUB1 N))
         (F 1 (* F N)))
        (((ZEROP N)(PRINT 'EL-FACTORIAL-ES-1))
         ((LT N 0) (PRINT 'ERROR-NUMERO-NEGATIVO))
         ((EQ N 1) (PRINT (LIST 'EL-FACTORIAL-ES F))))
    );END-DO
);END-FACT

(FACT 0) => EL-FACTORIAL-ES-1

(FACT 5) => (EL-FACTORIAL-ES 120)

(FACT -1) => ERROR-NUMERO-NEGATIVO

(DE MEMBER (A L)
    (DO ((L L (CDR L)))
        (((NULL L) NIL)
         ((EQUAL A (CAR L)) T))
    );END-DO
);END-MEMBER

(MEMBER '(A) '(A B C)) => NIL

(MEMBER '(A) '((A) (B) (C))) => T
```

A.16 **FUNCIONES QUE MANIPULAN FUNCIONES**

Las funciones que nos permiten manipular otras funciones son: **APPLY** y **MAP**, su funcionamiento aparenta ser similar, pero tienen diferente objetivo. Ambas pueden tener uno o más parámetros.

(APPLY <func> <lista>)

APPLY aplica la función <func> a la lista de argumentos evaluados, representados por <lista>.

Ejemplo:

(APPLY ADD (LIST (ADD1 5) (MUL 2 3))) => 12

(MAP <func> <lista>)

MAP aplica la función <func> sucesivamente a cada uno de los argumentos evaluados de la lista. El valor que retorna es NIL.

Ejemplo:

```
(MAP PRINT '(A B C)) -> (A B C)
                        (B C)
                        (C)
                        NIL
```

A.17 BIBLIOGRAFIA

1) LISP 1.5 PROGRAMMER'S MANUAL. JOHN McCARTHY
   THE M.I.T PRESS. 1981

2) ARTIFICIAL INTELIGENCE. PATRICK WINSTON
   THE M.I.T PRESS. 1982

3) BYTE MAGAZINE. Agosto de 1979

4) MANUAL DE LISP-CROMEMCO. 1980

APENDICE B

PROGRAMA SIMULADOR EN LISP.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                   ;;
;;;  funcion para elevar a la po-     ;;
;;;  tencia numero 2 (cuadrado)       ;;
;;;                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de ** (x) (* (abs x) (abs x)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                   ;;
;;;   funcion de acercamiento a la    ;;
;;;   barrera.                        ;;
;;;                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de acercar (pto)
    (print (list 'la-rana-se-acerca-a-la-barrera-en pto))
    (actualiza-rana pto)
    (orienta)
);end-acercar


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                   ;;
;;;   funcion que actualiza la po-    ;;
;;;   sicion de la rana, asi como     ;;
;;;   la  lista  de  puntos  de su    ;;
;;;   trayectoria                     ;;
;;;                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de actualiza-rana (pto &aux trayec)
    (putprop 'rana 'posicion pto)
    (setq trayec (cons pto (getprop 'rana 'trayecto)))
    (putprop 'rana 'trayecto trayec)
);end-actualiza-rana


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;   funcion que pide la altura de la   ;
;   barrera o bien la profundidad de   ;
;   la  zanja  dependiendo  esto del   ;
;   objeto del cual se trate.          ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de alt-prof (tipo &aux dato)
    (terpri)
    (cond ((equal tipo 'zanja)
                   (prin0 '(dame la profundidad))
                   (setq dato (exam-dato (read) 20))
                   (cond ((gt dato (getprop 'rana 'botada))
                                                  'profunda)
```

```
                    (t                              'no-profunda)
                )fend-cond:)
          (t    (prin0 '(dame la altura))
                (setq dato (exam-dato (read) 20))
                (cond ((gt dato (getprop 'rana 'salto))
                                                   'alta)
                      (t                           'baja)
                )fend-cond:)
    )fend-cond
)fend-alt-prof


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                  ;;
;;;   analiza-barrera es la funci-   ;;
;;;   cion que  examina las carac-   ;;
;;;   teristicas  de  la  barrera,   ;;
;;;   para asi  poder decidir cual   ;;
;;;   es la funcion a activar:       ;;
;;;           1) acercar             ;;
;;;           2) rodear              ;;
;;;           3) saltar              ;;
;;;                                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de analiza-barrera (l &aux au d obj pto l-1 l-2 travec)
    (setq obj (nth (getprop 'obstaculos 'descripcion) (car l)))
    (setq d (dist-puntos (getprop 'presa 'posicion) (cadr l)))
    (cond ((le d (** (getprop 'rana 'lengua)))

                ; si la distancia entre la presa y
                ; la barrera es pequena entonces la
                ; la rana procede a acercarse para
                ; para atacar a su presa entre los
                ; barrotes de la barrera

                (acercar (cadr l)))

          ((equal (nth obj 3) 'alta)

                ; si la barrera es alta entonces
                ; se procede a realizar un rodeo

                (rodear obj))

          (t    ; deducimos entonces que se trata

                ; de una barera que puede ser
                ; saltada por la rana

                (saltar (cadr l)))

    );end-cond
);end-analiza-barrera
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                           ;
;  funcion que  inspecciona que tipo        ;
;  de obstaculo interfiere en la tra-       ;
;  yectoria  rana-presa  dependiendo        ;
;  del tipo activa analiza-barrera o        ;
;  analiza-zanja                            ;
;                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de analiza-obstaculo (l)
    (putprop 'obstaculos 'actual (caar l))
    (if (and
        (equal (cadar l) (car  (getprop 'rana 'trayecto)))
        (equal (cadar l) (cadr (getprop 'rana 'trayecto))))
       ;then; (permanencia-rana (cadar l))
       ;else; (if (equal (car (nth
                              (getprop 'obstaculos 'descripcion)
                              (caar l))) 'barrera)
                 ;then; (analiza-barrera (car l))
                 ;else; (analiza-zanja l)
             );endif
    );end-if
);end-analiza-obstaculo


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                         ;
;  analiza-trayecto es la  funcion        ;
;  que examina si existen obstacu-        ;
;  los que interfieran  en la ruta        ;
;  rana-presa.  el valor  de  esta        ;
;  funcion puede ser:                     ;
;     * nil si no hay obstaculos          ;
;          que interfieran                ;
;     * una lista que contiene el         ;
;       el  obstaculo y el  punto         ;
;       mas cercano, asi como to-         ;
;       dos  aquellos puntos  que         ;
;       interfieren en su ruta            ;
;                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de analiza-trayecto (l pto &aux tray n p-i l-p l-p-i)
    (setq tray (modula (getprop 'rana 'posicion) pto ))
    (setq l-p-i nil)
    (do ((l l (cdr l)) (n 1 (add1 n)))
        (((not (not(null l)))))
        (if (not (equal n (getprop 'obstaculos 'actual)))
        ;then; (do ((l-p (if (eq (caar l) 'zanja)
                            ;then; (reverse(cons(car (cadar l))
                                              (reverse (cadar l))))
                            ;else; (cadar l)
                            );end-if
                          (cdr l-p)))
                   (((not(not(null(cdr l-p))))))
```

```
                              (seta p-i (inter-lineas tray
                                         (modula (car l-p) (cadr l-p))))

                          (if (not(eq p-i nil))
                          :then:(seta l-p-i
                                      (cons(list n p-i) l-p-i))
                          :else:nil
                          ):end-if
                     ):end-do
          :else: nil
          ):endif
     ):end-do:
     (if (not(null l-p- ))
        :then: (list (pp ato-pto-cerca l-p-i) l-p-i)
        :else: nil
     ):end-if
):end-analiza-trayecto


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                  ;;
;;;   analiza-zanja es la funcion    ;;
;;;   que inspecciona las carac-     ;;
;;;   teristicas de la zanja, pa-    ;;
;;;   ra asi poder decidir cual      ;;
;;;   es la funcion a activar:       ;;
;;;            1) brincar            ;;
;;;            2) cruzar             ;;
;;;            3) retirar            ;;
;;;                                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de analiza-zanja (l &aux p-1 p-2 dist trayec)
     (setq p-1 (cadar l))
     (print (list 'la-rana-se-acerca-a-la-zanja-en p-1))
     (actualiza-rana p-1)
     (setq p-2 (busca-extremo (car l) (cadr l)))
     (setq dist (dist-puntos p-1 p-2))
     (cond ((le dist (** (getprop 'rana 'brinco)))

                          ; la rana brincara la zanja

                          (brincar p-1 p-2))

           ((equal (nth
                    (nth (getprop 'obstaculos 'descripcion) (caar l))
                     3) 'no-profunda)

                          ; si la zanja es no profunda
                          ; la rana procede a descender
                          ; la zanja y cruzarla

                          (cruzar p-1 p-2))

           (t             ; de otra manera la rana
                          ; olvidara su presa y se retirara
```

```
                    (retirar p-1))
      ):end-cond
 ):end-analiza-zanja


 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;                                           ;
 ;    funcion que finaliza la simulac -      ;
 ;    on exitosamente, con el ataque a       ;
 ;    su presa.                              ;
 ;                                           ;
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 (de ataca ()
     (print (list 'la-rana-ataca-a-su-presa en
                  (getprop 'presa 'posicion)))
     (setq traz (cons (getprop 'presa 'posicion)
                      (getprop 'rana 'trayecto)))
     (putprop 'rana 'trayecto traz)
     (print '(fin-de-simulacion))
     nil
 ):end-ataca


 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;                                           ;
 ;  funcion que simula el brinco de la       ;
 ;  rana colocandola en una nueva po-        ;
 ;  sicion.                                  ;
 ;                                           ;
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 (de brincar (p-1 p-2)
     (print (list 'la-rana-brinca-la-zanja-de p-1 'a p-2))
     (actualiza-rana p-2)
     (orienta)
 ):end-brincar


 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;;;                                       ;;
 ;;;  funcion que dado  un punto de       ;;
 ;;;  un zanja obtiene de una lista       ;;
 ;;;  el otro punto  extremo de di-       ;;
 ;;;  cha zanja, por el cual la ra-       ;;
 ;;;  na probablemente pase.              ;;
 ;;;                                       ;;
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 (de busca-extremo (pto l &aux l-ptos)
     (setq l-ptos (assoc (car pto) l))
     (if (equal (cadr pto) (cadar l-ptos))
         ;then; (cadar (assoc (car pto) (cdr l-ptos)))
         ;else; (cadar l-ptos)
     ):end-if
 ):end-busca-extremo
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;    funcion que determina los puntos  ;
;    que existen entre dos coordena-   ;
;    das siempre y cuando:             ;
;       delta-y > 0  y  delta-x > 0    ;
;    la salida de recursividad se      ;
;    presenta cuando :                 ;
;    delta-x = 0     y    delta-y = 1  ;
;             o bien si                ;
;    delta-x = 1     y    delta-y = 0  ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de caso-1 (punto-inf punto-sup &aux delta-x delta-y punto-med x y)
    (setq delta-x (sub (car punto-sup) (car punto-inf)))
    (setq delta-y (sub (cadr punto-sup) (cadr punto-inf)))
    (cond ((not (or (and (zerop delta-x) (equal delta-y 1))
                    (and (equal delta-x 1) (zerop delta-y))))
                (setq x (add (div delta-x 2) (car punto-inf)))
                (setq y (add (div delta-y 2) (cadr punto-inf)))
                (setq x (add (rem delta-x 2) x))
                (setq punto-med (list x y))
                (caso-1 punto-inf punto-med )
                (setq lista-puntos
                      (append lista-puntos (list punto-med)))
                (caso-1 punto-med punto-sup))
    );end-cond
);end-caso-1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;    funcion que determina los puntos  ;
;    que existen  entre dos coordena-  ;
;    das siempre y cuando:             ;
;       delta-y < 0  y  delta-x > 0    ;
;    la salida de recursividad se      ;
;    presenta cuando :                 ;
;    delta-x = 0     y    delta-y = 1  ;
;             o bien si                ;
;    delta-x = -1    y    delta-y = 0  ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de caso-2 (punto-inf punto-sup &aux delta-x delta-y punto-med x y)
    (setq delta-x (sub (car punto-sup) (car punto-inf)))
    (setq delta-y (sub (cadr punto-sup) (cadr punto-inf)))
    (cond ((not (or (and (equal delta-x 1) (zerop delta-y))
                    (and (zerop delta-x) (equal delta-y -1))))
                (setq delta-y (add1 delta-y))
                (setq x (add (div delta-x 2) (rem delta-x 2)))
                (setq x (add x (car punto-inf)))
                (setq y (add (div delta-y 2) (cadr punto-inf)))
                (setq punto-med (list x y))
                (caso-2 punto-inf punto-med)
```

```
                (setq lista-puntos (append lista-puntos
                                            (list punto-med)))
                (caso-2 punto-med punto-sup))
     );end-cond
 );end-caso-2


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                       ;
;   funcion que determina los puntos    ;
;   que existen entre  dos coordena-    ;
;   das siempre y cuando:               ;
;       delta-y < 0  y  delta-x < 0     ;
;   la salida de recursividad se        ;
;   presenta cuando :                   ;
;   delta-y = -1      y     delta-x = 0 ;
;            o bien si                  ;
;   delta-y = 0       y     delta-x = -1 ;
;                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de caso-3 (punto-inf punto-sup &aux delta-x delta-y punto-med x y)
    (setq delta-x (sub (car punto-sup) (car punto-inf)))
    (setq delta-y (sub (cadr punto-sup) (cadr punto-inf)))
    (cond ((not (or (and (zerop delta-x) (equal delta-y -1))
                    (and (equal delta-x -1) (zerop delta-y))))
                (setq x (add (div delta-x 2) (car punto-inf)))
                (setq x (sub x (rem delta-x 2)))
                (setq y (add (div delta-y 2) (cadr punto-inf)))
                (setq punto-med (list x y))
                (caso-3 punto-inf punto-med)
                (setq lista-puntos (append lista-puntos
                                            (list punto-med)))
                (caso-3 punto-med punto-sup))
    );end-cond
);end-caso-3


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                       ;
;   funcion que determina los puntos    ;
;   que existen entre dos  coordena-    ;
;   das siempre y cuando:               ;
;      delta-y > 0  y  delta-x < 0      ;
;   la salida de recursividad se        ;
;   presenta cuando :                   ;
;   delta-y = 0       y     delta-x =-1 ;
;            o bien si                  ;
;   delta-y = 1       y     delta-x = 0 ;
;                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de caso-4 (punto-inf punto-sup &aux delta-x delta-y punto-med x y)
    (setq delta-x (sub (car punto-sup) (car punto-inf)))
    (setq delta-y (sub (cadr punto-sup) (cadr punto-inf)))
    (cond ((not (or (and (zerop delta-x) (equal delta-y 1))
```

```
                        (and (equal delta-x -1) (zerop delta-y))))
                   (setq x (add (div delta-x 2) (car punto-inf)))
                   (setq y (add (div delta-y 2) (cadr punto-inf)))
                   (setq punto-med (list x y))
                   (caso-4 punto-inf punto-med)
                   (setq lista-puntos (append lista-puntos
                                              (list punto-med)))
                   (caso-4 punto-med punto-sup))
     );end-cond
);end-caso-4

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                     ;
;   funcion que determina los puntos  ;
;   que existen entre dos  coordena-  ;
;   das siempre y cuando:             ;
;      delta-y <> 0 y  delta-x = 0    ;
;   la salida de recursividad se      ;
;   presenta cuando :                 ;
;   delta-y = -1     o    delta-y = 1 ;
;                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de caso-5 (punto-inf punto-sup &aux delta-y punto-med y)
    (setq delta-y (sub (cadr punto-sup) (cadr punto-inf)))
    (cond ((not (or (equal delta-y -1) (equal delta-y 1)))
                   (setq y (add (div delta-y 2) (cadr punto-inf)))
                   (setq punto-med (list (car punto-inf) y))
                   (caso-5 punto-inf punto-med)
                   (setq lista-puntos (append lista-puntos
                                              (list punto-med)))
                   (caso-5 punto-med punto-sup))
     );end-cond
);end-caso-5

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                     ;
;   funcion que determina los puntos  ;
;   que existen entre dos  coordena-  ;
;   das siempre y cuando:             ;
;      delta-y = 0  y  delta-x <> 0   ;
;   la salida de recursividad se      ;
;   presenta cuando :                 ;
;   delta-x = -1     y    delta-y = 1 ;
;                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de caso-6 (punto-inf punto-sup &aux delta-x punto-med x)
    (setq delta-x (sub (car punto-sup) (car punto-inf)))
    (cond ((not (or (equal delta-x -1) (equal delta-x 1)))
                   (setq x (add (div delta-x 2) (car punto-inf)))
                   (setq punto-med (list x (cadr punto-inf)))
                   (caso-6 punto-inf punto-med)
                   (setq lista-puntos (append lista-puntos
```

```
                                          (list punto-med)))
                     (caso-6 punto-med punto-sur))
    ');end-cond
);end-caso-6


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;  funcion  que simula el descenso y   ;
;  cruze de la zanja por la rana, la   ;
;  que  permanece ahora en nueva po-   ;
;  sicion.                             ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de cruzar (p-1 p-2)
    (print (list 'la-rana-cruza-la-zanja-de p-1 'a p-2))
    (actualiza-rana p-2)
    (orienta)
);end-cruzar


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;  funcion coordenadas-obstaculos      ;
;  tiene por objetivo pedir las ca-    ;
;  racteristicas de cada obstaculo     ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de desc-obst (n &aux tipo i l-obst l-aux)
    (putprop 'obstaculos 'numero n)
    (setq l-obst nil)
    (do ((i 1 (add1 i)))
        (((not (le i n))))
        (prin0 '(tipo de obstaculo [barrera/zanja] o [b/z]))
        (setq tipo (exam-tipo (read)))
        (setq l-aux (list tipo (posicion-obstaculo tipo)
                                   (alt-prof tipo)))
        (setq l-obst (cons l-aux l-obst))
    );end-do
    (putprop 'obstaculos 'descripcion (reverse l-obst))
    (terpri) (print (plist 'obstaculos)) (terpri)
);end-desc-obst


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                  ;;
;;;  funcion que devuelve la suma    ;;
;;;  de los cuadrados de la dife-    ;;
;;;  rencia de dos puntos            ;;
;;;                                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de dist-puntos (p-1 p-2)
    (+ (** (-  (car p-1)  (car p-2)))
```

```
        (** (- (cadr p-1) (cadr p-2))))
  );end-dist-puntos


  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;                                               ;
  ;    esta funcion tiene por objeto              ;
  ;    definir las dimensiones del pla-           ;
  ;    no de simulacion.                          ;
  ;                                               ;
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (de ejes-x-y ()
      (setq eje-x 30)
      (setq eje-y 30)
      (terpri)
      (print '(el espacio de movimiento en eje-x es desde 0 a 30))
      (print '(el espacio de movimiento en eje-y es desde 0 a 30))
      (prin0 '(si deseas modificarlo lo puedes hacer [s/n]))
      (if (not (eq (read) 's))
         ;then: nil
         ;else: (terpri)
                (prin0 '(longitud maxima de eje-x))
                (setq eje-x (exam-dato (read) 100))
                (terpri)
                (prin0 '(longitud maxima de eje-y))
                (setq eje-y (exam-dato (read) 100))
    ));end-if
  );end-ejes-x-y

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;                                               ;
  ;    funcion recursiva que verifica             ;
  ;    si cada una de las coordenadas             ;
  ;    que definen la posicion de obs-            ;
  ;    taculo  son correctas tanto en             ;
  ;    numero como en componentes. si             ;
  ;    existe  algun  error  entonces             ;
  ;    volvera a pedir la posicion co-            ;
  ;    rrecta,  llamandose  asimisma.             ;
  ;    devuelve como valor la lista de            ;
  ;    coordenadas que definen  la po-            ;
  ;    sicion del obstaculo.                      ;
  ;                                               ;
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (de examina-coords (l tipo n )
      (cond ((and (listp l)
                  (equal n (length l))
                  (valida (nth l 1))
                  (valida (nth l 2))
                  (if (equal n 4)
                     ;then: (and (valida (nth l 3))
                                 (valida (nth l 4)))
                     ;else;t
```

```
                    );end-if;)       ())
               (t    (print '(dato invalido repitelo))
                     (examina-coords (read) tipo n))
          );end-cond
     );end-examina-coords


     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     ;                                           ;
     ;    esta funcion se encarga de veri-       ;
     ;    ficar que las  coordenadas  sean       ;
     ;    una lista  de dos numeros y  que       ;
     ;    estos se  encuentren  dentro del       ;
     ;    espacio definido en eje-x-y            ;
     ;                                           ;
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

     (de exam-coordxy (c)
          (cond ((and (listp c) (eq (length c) 2)
                      (numberp (car c)) (numberp (cadr c)))
                      (cond((and (le (car c) eje-x)
                                 (ge (car c) 0))
                           (cond ((and (le (cadr c) eje-y)
                                       (ge (cadr c) 0)) c)
                                 (t (print '(error coord-y fuera de rango))
                                    (prin0 '(repite coordenadas (x y)))
                                    (exam-coordxy (read)))
                           );end-cond;)
                      (t    (print '(error coord-x fuera de rango))
                            (prin0 '(repite coordenadas (x y)))
                            (exam-coordxy (read)))
                      );end-cond;)
                (t    (print ''error se requiere lista con x y numericos))
                      (prin0 '(repite coordenadas (x y)))
                      (exam-coordxy (read)))
          );end-cond
     );end-exam-coordxy


     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     ;                                           ;
     ;    la  funcion  exam-dato  verifica       ;
     ;    que el dato sea numerico, no ne-       ;
     ;    negativo y  a su vez no mayor al       ;
     ;    limite dado como parametro.            ;
     ;                                           ;
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

     (de exam-dato (dat lim)
          (cond ((numberp dat)
                 (cond ((lt dat 0)
                             (prin0 '(dato menor que cero repitelo))
                             (exam-dato (read) lim))
                       ((gt dat lim)
                             (prin0 (list 'dato 'mayor 'al 'limite
                                          lim 'repitelo))
```

```
                    (exam-dato (read) lim))
                (t    dat)
            );end-cond;)
            (t     (prin0 '(error dato no numerico repitelo))
                   (exam-dato (read) lim))
    );end-cond
);end-exam-dato

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;   funcion  recursiva que examina el  ;
;   tipo de obstaculo a definir es va- ;
;   lido, es decir es barrera o zanja  ;
;   en caso de  definir  un tipo inco- ;
;   rrecto,  la funcion  envia un men- ;
;   saje de error y se llama asimisma  ;
;   devuelve como  resultado el  tipo  ;
;   de obstaculo                       ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de exam-tipo (tipo)
    (cond ((or (equal tipo 'barrera) (equal tipo 'b)) 'barrera)
          ((or (equal tipo 'zanja) (equal tipo 'z)) 'zanja)
          (t    (prin0 '(tipo incorrecto repitelo
                         [barrera/zanja] o [b/z]))
                (exam-tipo (read)))
    );end-cond
);end-exam-tipo

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;   funcion que grafica el modelo de   ;
;   simulacion en el video. grafica:   ;
;     a) posicion inicial de la rana   ;
;     b) posicion de la presa          ;
;     c) trayectoria rana-presa        ;
;     d) obstaculos                    ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de grafica-modelo ()
    (traza (list (car (reverse (getprop 'rana 'trayecto)))) 'r)
    (traza (list (getprop 'presa 'posicion)) 'p)
    (grafica-obstaculos)
    (grafica-trayecto)
);end-grafica-modelo

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;    funcion que coordina la grafica-  ;
;    cion de cada uno de los obstacu-  ;
;    los.                              ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(de grafica-obstaculos ()
    (do ((l (getprop 'obstaculos 'descripcion) (cdr l)))
        (((not (not (null l)))))
        (do ((l-p (if (eq (caar l) 'zanja)
                      ;then: (reverse (cons (car (cadar l))
                                           (reverse (cadar l))))
                      ;else: (cadar l)
                  );end-if
             (cdr l-p)))
            (((not (not (null (cdr l-p))))))
            (traza (mudula (car l-p) (cadr l-p)) 'o)
        );end-do
    );end-do
);end-grafica-obstaculos


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                          ;
;  funcion que coordina la grafica-        ;
;  cion de la trayectoria rana-presa       ;
;                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de grafica-trayecto ()
    (do ((l (getprop 'rana 'trayecto) (cdr l)))
        (((not (not (null (cdr l))))))
        (traza (modula (car l) (cadr l)) '*)
    );end-do
);end-grafica-trayecto


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                            ;
;   funcion de impresion     ;
;                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de imprime (l)
    (terpri)
    (print '----------------------------)
    (print l)
    (print '----------------------------)
);end-imprime


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                            ;
;  funcion que inspecciona si exis-          ;
;  te algun punto comun a dos  lis-          ;
;  tas de puntos, devolviendo como           ;
;  resultado  este  punto, de otra           ;
;  manera devuelve nil.                      ;
;                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de inter-lineas (l1 l2)
```

```
    (cond ((null l1) nil)
          ((member (car l1) l2) (car l1))
          (t (inter-lineas (cdr l1) l2))
    );end-cond
);end int-linea


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                               ;
;   member funciona como un predica-            ;
;   el cual  examina si un  elemento            ;
;   "a" es parte de la lista "l"                ;
;                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de member (a l)
    (cond ((null l) nil)
          ((equal (first l) a) t)
          (t (member a (rest l)))
    );end-cond
);end-member


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                               ;
;   dados dos puntos (x y) la funci-            ;
;   on  modula  genera una  lista de            ;
;   puntos intermedios, cuyos extre-            ;
;   mos son los puntos de  entrada a            ;
;   la funcion, para ello se auxilia            ;
;   de las funciones caso-1 a caso-6            ;
;   las cuales son usadas de acuerdo            ;
;   al tipo de pendiente  que exista            ;
;   entre los dos puntos.                       ;
;                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de modula (pto-inf pto-sup &aux delta-x delta-y)
    (setq delta-x (sub (car pto-sup) (car pto-inf)))
    (setq delta-y (sub (cadr pto-sup) (cadr pto-inf)))
    (setq lista-puntos (list pto-inf))
    (cond ((and (gt delta-y 0) (gt delta-x 0))
                (caso-1 pto-inf pto-sup))
          ((and (lt delta-y 0) (gt delta-x 0))
                (caso-2 pto-inf pto-sup))
          ((and (lt delta-y 0) (lt delta-x 0))
                (caso-3 pto-inf pto-sup))
          ((and (gt delta-y 0) (lt delta-x 0))
                (caso-4 pto-inf pto-sup))
          ((and (zerop delta-x) (not(zerop delta-y)))
                (caso-5 pto-inf pto-sup))
          ((and (not(zerop delta-x)) (zerop delta-y))
                (caso-6 pto-inf pto-sup))
          (t    (setq lista-puntos ())))
    );end-cond
    (append lista-puntos (list pto-sup))
```

```
);end-modula


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                         ;;
;;;  funcion que determina cual es          ;;
;;;  el punto y el  obstaculo  mas          ;;
;;;  cercano a  la posicion  de la          ;;
;;;  rana                                   ;;
;;;                                         ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de objeto-pto-cerca (l &aux d-min rad obj-pto)
    (setq d-min 1000)
    (do ((l l (cdr l)))
        (((not(not(null l)))))
        (setq rad (dist-puntos (getprop 'rana 'posicion) (cadar l)))
        (if (ge rad d-min)
                ;then; nil
                ;else; (setq d-min rad)
                       (setq obj-pto (car l))
        );end-if
    );end-do
    obj-pto
);end-objeto-pto-cerca


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                           ;
;  la funcion obstaculos-datos pide         ;
;  el numero de obstaculos para des-        ;
;  pues solicitar la descripcion de         ;
;  los mismos.                              ;
;  nota:la funcion acepta un maximo         ;
;  de  10  obstaculos (cantidad que         ;
;  puede ser modificada)                    ;
;                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de obstaculos-datos (&aux nume-obst)
    (terpri)
    (prin0 '(numero de obstaculos ))
    (setq nume-obst (exam-dato (read) 10))  *
    (cond  ((gt nume-obst 0)  (desc-obst nume-obst))
           (t                 (putprop 'obstaculos 'numero 0)))
    );end-cond
    (putprop 'obstaculos 'actual 0)
);end-obstaculos-datos
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;  la funcion orienta es la que lle-   ;
;  va  acabo la parte  central de la   ;
;  simulacion: define la accion  in-   ;
;  mediata a  ejecutar  por  la rana   ;
;  en funcion del medio ambiente que   ;
;  se  le presenta.  tales  acciones   ;
;  son:  atacar o  bien  analizar el   ;
;  el obstaculo  inmediato que se le   ;
;  interpone.                          ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de orienta ( &aux accion)
    (setq accion (analiza-trayecto
                            (getprop 'obstaculos 'descripcion)
                            (getprop 'presa 'posicion )))
    (if (equal (getprop 'presa 'inicial)
               (getprop 'presa 'posicion))

        ;then: (if (equal accion nil)
                ;then: (ataca)
                ;else: (analiza-obstaculo accion)
               );end-if
        ;else: (if (equal accion nil)
                ;then: (progn
                       (remprop 'presa 'posicion)
                       (orienta)
                       );end-progn
                ;else: (analiza-obstaculo accion)
               );end-if
    );end-if
);end-orienta


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;  en esta funcion la rana olvida su   ;
;  objetivo y permanece inmovil.       ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de permanencia-rana (pto &aux tray)
    (print (list 'la-rana-permanece-en pto 'indefinidamente))
    (setq tray (cons pto (getprop 'rana 'trayecto)))
    (putprop 'rana 'trayecto tray)
    (print '(fin-de-simulacion))
    nil
);end-permanencia-rana
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                    ;
;   funcion que solicita las coorde- ;
;   das del obstaculo en funcion del ;
;   tipo de obstaculo a definir:     ;
;        2 para definir una barrera  ;
;        4 para definir una zanja    ;
;   hace uso de  examina-coordenadas ;
;   para verificar  esta informacion ;
;   retorna como  resultado la lista ;
;   de coordenadas correctas que li- ;
;   mitan la posicion del obstaculo. ;
;                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de posicion-obstaculo (tipo &aux n-coord)
    (cond ((equal tipo 'barrera)
                  (setq n-coord 2)
                  (prin0 '(coordenadas ( (x1 y1) (x2 y2) ))))
          (t      (setq n-coord 4)
                  (prin0 '(coordenadas ( (x1 y1) a (x4 y4) ))))
    );end-cond
    (examina-coords (read) tipo n-coord)
);end-posicion-obstaculo


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                    ;
;   funcion que permite obtener la   ;
;   posicion de la presa             ;
;                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de presa-datos ()
    (terpri)
    (prin0 '(posicion de la presa (x y)))
    (putprop 'presa 'posicion (exam-coordxy (read)))
    (putprop 'presa 'inicial (getprop 'rana 'posicion))
    (terpri)
);end-presa-datos
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                       ;
;   esta rutina se encarga de pedir     ;
;   datos para la simulacion.           ;
;   los  datos  de  entrada  son la     ;
;   posicion de la rana, tomando en     ;
;   consideracion   que   existe  la    ;
;   siguiente restriccion:              ;
;   el espacio de la rana es de x=0     ;
;   a  x=eje-x  y  y=0 a  y=eje-y       ;
;   la capacidades maximas para         ;
;        salto altura  = 4 unidades     ;
;        salto long.   = 5 unidades     ;
;        descenso      = 4 unidades     ;
;        lengua        = 1 unidad       ;
;                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de rana-datos ()
     (terpri)
     (prin0 '(posicion de la rana  (x y) ))
     (putprop 'rana 'posicion (exam-coordxy (read)))
     (putprop 'rana 'trayecto (cons (getprop 'rana 'posicion) nil))
     (terpri) (terpri)
     (prin0 '(longitud de salto de altura ))
     (putprop 'rana 'salto (exam-dato (read) 4))
     (terpri)
     (prin0 '(longitud de salto de longitud ))
     (putprop 'rana 'brinco (exam-dato (read) 5))
     (terpri)                        (prin0 '(longitud de descenso ))
     (putprop 'rana 'bajada (exam-dato (read) 4))
     (terpri)
     (prin0 '(longitud de lengua ))
     (putprop 'rana 'lengua (exam-dato (read) 1))
     (terpri)
);end-rana-datos


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                       ;
;  funcion que finaliza la simulacion   ;
;  con  la retirada del animal, debi-   ;
;  do a  que  la zanja que interfiere   ;
;  en su trayectoria  hacia  la presa   ;
;  es ancha y profunda.                 ;
;                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de retirar (p-1)
     (print (list 'la-rana-olvida-su-presa-en p-1))
     (print '(fin-de-la-simulacion))
     nil
);end-retirar
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                  ;;
;;;   la funcion rodear tiene por    ;;
;;;   objeto obtener el punto por    ;;
;;;   el cual la rana va a rodear    ;;
;;;   a la barrera, este punto es    ;;
;;;   aquel el cual este mas cerca   ;;
;;;   de la posicion de la rana      ;;
;;;                                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de rodear (obj &aux p-int1 p-int2 l-1 l-2 pto lado d-1 d-2)
    (setq p-int1 (analiza-trayecto
                          (getprop 'obstaculos 'descripcion)
                          (caadr obj)))
    (setq p-int2 (analiza-trayecto
                          (getprop 'obstaculos 'descripcion)
                          (car (cdadr obj))))
    (if (null (append p-int1 p-int2)) ;then; (progn

    ; se calcula la longitud que existe
    ; entre la posicion de la rana y cada
    ; uno de los extremos de la barrera

    (setq l-1 (length (modula (caadr obj))
                             (getprop 'rana 'posicion))))
    (setq l-2 (length (modula (car (cdadr obj))
                             (getprop 'rana 'posicion))))

    ; dichas longitudes son comparadas
    ; para obtener el extremo mas cercano

    (if (gt l-2 l-1)
         ;then; (setq pto (caadr obj))
         ;else; (setq pto (car (cdadr obj)))
    );end-if
    (print (list 'la-rana-rodea-la-barrera-por pto))
    (actualiza-rana pto)
                              );end-progn
    ;else; (setq lado
                  (if (not(null p-int1))
                       (if (not (null p-int2))
                       ;then; (progn
                               (setq d-1 (dist-puntos
                                         (cadar p-int1)
                                         (getprop 'rana 'posicion)))
                               (setq d-2 (dist-puntos
                                         (cadar p-int2)
                                         (getprop 'rana 'posicion)))
                                (if (gt d-1 d-2)
                                ;then; 'p-int2
                                ;else; 'p-int1
                                );end-if
                                );end-progn
                       ;else; 'p-int1
                       );end-if
```

```
                    :else 'fin2
                    )(:end-if))
               (if (equal lado 'f-int1)
                (:then (addfrof 'fresa 'fosicion (cadr obj)))
                (:else (addfrof 'fresa 'fosicion (car (cdadr obj))))
               )(:end-if
               (futfrof 'obstaculos 'actual (car obj))
     )(:end-if
     (orienta)
 )(:end-rodear


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                        ;;
;;;  la funcion salta nos permite          ;;
;;;  determinar cuales son los             ;;
;;;  puntos: antes y despues de            ;;
;;;  saltar la barrera, para que           ;;
;;;  esta accion sea realizada             ;;
;;;                                        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de saltar (pto &aux pto-sig pto-ant)

     ; se obtiene el punto antes de saltar

     (setq pto-ant (cadr (reverse (modula (getfrof 'rana 'fosicion)
                                          pto))))
     (print (list 'la-rana-se-aproxima-a-la-barrera-en pto-ant))
     (actualiza-rana pto-ant)
     ; se obtiene el punto despues de saltar

     (setq pto-sig (cadr (modula pto (getfrof 'fresa 'fosicion))))
     (print (list 'la-rana-salta-la-barrera-hasta pto-sig))
     (actualiza-rana pto-sig)
     (orienta)
)(:end-saltar


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                          ;
;    funcion  maestra  que coordina        ;
;       la simulacion del modelo           ;
;                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de simula ()
     (universo)
     (orienta)
     (grafica-modelo)
)(:end-modelo
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;  funcion que hace la graficacion de  ;
;  listas en la pantalla de la comp.   ;
;                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de traza (l a)
    (if (null l)
        ;then; nil
        ;else; (crt (caar l) (cadar l))
               (print a)
               (traza (cdr l) a)
    );end-if
);end-traza


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                      ;
;   funcion que hace la peticion de    ;
;   los  datos  necesarios  para la    ;
;   simulacion, analizando cada uno    ;
;   de ellos.                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de universo ()
    (imprime '---definicion-de-espacio---)
    (ejes-x-y)
    (imprime '---definicion-de-la-rana---)
    (rana-datos)
    (imprime '---definicion-de-la-presa--)
    (presa-datos)
    (imprime '--definicion-de-obstaculos-)
    (obstaculos-datos)
);end-universo
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                       ;
;   funcion que examina lo siguien-     ;
;   te:                                 ;
;       * la coordenada tenga dos       ;
;         componentes                   ;
;                                       ;
;       * los componentes sean          ;
;         numeros                       ;
;                                       ;
;       * los numeros esten entre       ;
;         los limites definidos         ;
;         en eje-x-y.                   ;
;                                       ;
;   basta con que no se cumpla una      ;
;   de las restricciones anteriores     ;
;   para que la coordenada sea re-      ;
;   chazada y se  envie un mensaje      ;
;   de error, devuelve como resul-      ;
;   tado una bandera que indica si      ;
;   la  coordenada  examinada  es       ;
;   correcta.                           ;
;                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(de valida (l)
    (cond ((and (listp l)
                (equal 2 (length l))
                (numberp (car l))
                (le (car l) eje-x)
                (ge (car l) 0)
                (numberp (cadr l))
                (le (cadr l) eje-y)
                (ge (cadr l) 0))          t)
          (t    (print (list 'la 'coordenada l 'es 'invalida))
                nil)
    );end-cond
);end-valida
```

APENDICE C

```
$SET LIST
$SET FORMAT
BEGIN
DEFINE
       BORRAPANTALLA=<T35, 'SUERTE <FF>  <ESC>E ^G ^G ^G      '>#,
       BORRASCREEN=<T35, ' <FF> <ESC>E ^G ^G ^G      '>#,
       S              =SCAN#,
       R              =REPLACE#,
       P              =POINTER#,
       THENN          =THEN BEGIN#,
       ELSEE          =END ELSE BEGIN#,
       ENDIFF         =END;#,
       DOO            =DO BEGIN#,
       ENDFORR        =END;#,
       ENDWHILEE      =END;#,
       ENDIF          = #,
       NOELSE         = #,
       ENDFOR         = #,
       ENDWHILE       = #,
       TO             =STEP 1 UNTIL#,
       MAX            =MAXRECSIZE#,
       BLOCK          =BLOCKSIZE#;

FILE ENTT (KIND=REMOTE),
     TERM (KIND=REMOTE, MYUSE=IO),
     ENTD (KIND=DISK,   FILETYPE=7),
     SAL  (KIND=REMOTE, MAXRECSIZE=22),
     IMP  (KIND=PRINTER,MAXRECSIZE=22),
     DIS  (KIND=DISK,   MAX=22, TITLE='RANA/RES.',
                        PROTECTION=SAVE),
     REM  (KIND=REMOTE, MAX=22);

SWITCH FILE ENT    := ENTT, ENTD;
SWITCH FILE SALIDA := REM,  IMP, DIS;

DEFINE OUT=SALIDA[SA]#;
```

APENDICE C

```
FORMAT OPCIONES (8(/),T20,' QUE ES LO QUE DESEAS HACER??',///,
                 T20,'SIMULACION DE ATAQUE          [0]',/,
                 T20,'SIMULACION DE HUIDA           [1]',/,
                 T20,'SIMULACION DE RESP. FOTOTAX.  [2]',/,
                 T20,'TERMINAR LA EJECUCION         [3]',/,/,'&?');

ARRAY
      ESPACIO     [0:60,0:120],     % MATRIZ DE ESPACIO DE SIMULACION

      OBSTACULOS [-4:100,0:10],     % GUARDAR LAS CARACTERISTICAS DE LOS
                                    %          OBSTACULOS

      HUECO       [1:20, 1:5],      % CARACTERISTICAS DE LOS HUECOS CAPTADOS

      MATCEL      [-1:171, 1:5],    % MATRIZ DE VISION DEL ESPACIO

      PRESA,                        % CARACTERISTICAS DE LA PRESA
      PREDADOR,   [0:5],            % CARACTERISTICAS DEL PREDADOR

      RANA,                         % CARACTERISTICAS DE LA RANA
      RANANT,                       % POSICION ANTERIOR DE LA RANA
      RANAINIC,                     % CARACTERISTICAS INICIALES DE LA RANA
      RANAP       [0:5],            % AUXILIAR EN ALGUNA POSICION DE LA RANA

      DENTRO,                       % COORDENADAS DENTRO DE LA BARRERA
      FUERA,      [0:1],            % COORDENADAS AL CRUZAR LA ZANJA

      NOMBARCH    [0:9],            % NOMBRE DEL ARCHIVO DE DATOS

      PUNTO,                        % PUNTO AUXILIAR PARA TRAYECTORIAS
      PUNTO_VISION,                 % PUNTO HACIA EL CUAL ESTA ORIENTADA LA RANA
      PUNTOP,                       % PARA CHECAR SI PUEDE SALTAR LA ZANJA
      PUNTOOR     [0:1],            % PUNTO HACIA EL CUAL ESTA ORIENTADA PARA LA
                                    %       RESPUESTA FOTOTAXICA

      VECPROF,                      % VECTOR DE PROFUNDIDADES
      VECANCHOS,                    % VECTOR DE ANCHOS
      VECGAUSS,                     % VECTOR DE LA FUNCION GAUSIANA
      VECCOLOR    [0:170];          % VECTOR DE COLORES
```

```
BOOLEAN
        ATACO,                          % ATACO A LA PRESA?
        CABE,                           % CABE SU LENGUA ENTRE LOS BARROTES PARA
                                        %       TRAGAR LA PRESA?
        FIN,                            % FIN DE LA SIMULACION
        HAYOBSTACULO,                   % PARA VER SI HAY OBSTACULO QUE SE CRUZE
                                        %       EN LA TRAYECTORIA DE LA RANA
        RETIRO,                         % SE RETIRA POR NO PODER TRAGAR A LA PRESA
        SIALCANZA,                      % DETECTAR QUE SI ALCANZA A LA PRESA CON SU
                                        % LENGUA
        SIBAJA,                         % SI ALCANZA A BAJAR LA ZANJA
        SIBRINCA,                       % SI ALCANZA A BRINCAR LA ZANJA
        SISALTA;                        % SI ALCANZA A SALTAR LA BARRERA


INTEGER
        AR,                             % PARA ACCESAR A CIERTO ARCHIVO
        ANG_DIREC,                      % ANGULO DE DIRECCION DEL PREDADOR
        C1, C2, C3,                     % CONSTANTES PARA EL PESO DE LOS HUECOS
        CASO_PRED,                      % PARA CUANDO EL PREDADOR VIENE DE FRENTE
        CUAL,                           % AUXILIAR PARA LA DECISION DE CUAL ARCHIVO
        DIREC,                          % DIRECCION DEL PREDADOR EN GRADOS
        DIST_DIREC,                     % DISTANCIA DEL PREDADOR
        DRP,                            % DISTANCIA RANA PRESA
        GRADO,                          % GRADO SOBRE EL CUAL SE VA A DIVIDIR LA
                                        %       LA FUNCION GAUSIANA
        HUECOS,                         % NUMERO DE HUECOS CAPTADOS
        INDICE,                         % INDICE PARA ACCESAR LA MATRIZ DE VISION
        METODO,                         % METODO DE SIMULACION
        NUMOBST,                        % NUMERO DE OBSTACULOS
        SA,                             % ACCESAR EL ARCHIVO DE SALIDA CORRECTO
        TIPOSIM;                        % TIPO DE SIMULACION QUE SE DESEA.


POINTER P4;                             % AUXILIAR EN EL RECONOCIMIENTO DEL TPO
                                        %           DE OBSTACULO
PROCEDURE ACOMODA_RANA (RANA, PO);
          ARRAY RANA,
                PO [0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                            %%
%%        RUTINA QUE SE ENCARGA DE DARLE UNA NUEVA UBICACION A LA             %%
%%  RANA CUANDO SE ESTA PROCESANDO LA RESPUESTA FOTOTAXICA.                   %%
%%                                                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
DIREC := 85;
PROCESAHLECOS (NUMOBST, OBSTACULOS, RANA, PO, MATCEL, HUECO,
               HUECOS, VECANCHOS, VECPROF, VECGAUSS, INDICE, 3);
IGUALA (RANANT, RANA, 2);
CAMINARANA   (INDICE, MATCEL, RANANT, RANA, 3);
LLENAESPACIO (RANANT[0], RANANT[1], RANA[0], RANA[1], 2);
END
ENDPROCEDURE ACOMODA_RANA;

PROCEDURE ANALIZA (OBST, RANA, OBJETIVO, DIXT, J, K, L, ESTORBA,
                                                  OBSNUM,PCOMUN);
```

```
            ARRAY   OBST [*],
                    RANA,
                    OBJETIVO [*],
                    PCOMUN   [*];

            BOOLEAN ESTORBA;

            INTEGER J,
                    K,
                    L,
                    OBSNUM;

            REAL    DIXT;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%        RUTINA QUE SE ENCARGA DE REALIZAR EL ANALISIS DE SI EXIS-     %%
%%   TE UN OBSTACULO QUE SE CRUZE EN EL TRAYECTO DE LA RANA HACIA LA    %%
%%   PRESA, Y NOS REGRESA TAMBIEN EL NUMERO DE OBTACULO DE CRUZE ASI    %%
%%   COMO SUS COORDENADAS.                                              %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL X,
         Y,
         SEGMOBS,
         LONGOBS,
         SEGMRAPRE,
         DISTRAPRE,
         DISTRAOBS;

INTERSECCION (RANA[0], RANA[1],   OBJETIVO[0], OBJETIVO[1],
              OBST[K], OBST[K+1], OBST[L],     OBST[L+1], X, Y);
IF (ABS(X) LEQ 1000) AND (ABS(Y) LEQ 1000) THEN % POSIBLE OBSTACULO
    BEGIN
    LONGOBS :=   DSTN (OBST[K], OBST[K+1], OBST[L], OBST[L+1]);
    SEGMOBS :=   DSTN (X,       Y,         OBST[K], OBST[K+1]) +
                 DSTN (X,       Y,         OBST[L], OBST[L+1]);
    DISTRAPRE := DSTN (RANA[0], RANA[1],   OBJETIVO[0], OBJETIVO[1]);
    SEGMRAPRE := DSTN (X,       Y,         RANA[0],     RANA[1]) +
                 DSTN (X,       Y,         OBJETIVO[0], OBJETIVO[1]);
    DISTRAOBS := DSTN (RANA[0], RANA[1],   X, Y);
    IF (IGUAL (SEGMOBS,   LONGOBS)   AND
        IGUAL (SEGMRAPRE, DISTRAPRE) AND
           (DISTRAOBS < DIXT))             THEN         % EXISTE OBST.
        BEGIN
        ESTORBA     := TRUE;
        DIXT        := DISTRAOBS;
        OBSNUM      := J;
        PCOMUN [0] := X;
        PCOMUN [1] := Y;
        END
     NOELSE
     ENDIF;
     END
ELSE          % NO EXISTE PROBABILIDAD DE OBSTACULO.
     ESTORBA := FALSE;
ENDIF;
```

```
END
ENDPROCEDURE ANALIZA;

PROCEDURE ATAQUE;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%     RUTINA QUE SE ENCARGA DE COORDINAR LAS LLAMADAS A LAS RUTINAS %%
%%  NECESARIAS PARA QUE LA RANA ATAQUE A LA PRESA.                    %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
INICIALIZA (NUMOBST, RANA, PRESA, OBSTACULOS);
IF FIN THENN
    % FIN DE LA RUTINA
ELSEE
    WRITE (OUT [SKIP 1]);
    WRITE (OUT,<T40,'EL ESTADO INICIAL ES:'>);
    IMPRIME;
    IGUALA (RANAINIC,RANA,2);
    WRITE (TERM,BORRASCREEN);
    WRITE (TERM,<8(/),T20,'QUE METODO SE USARA PARA LA SIMULACION??',//,
                  T10,'SIN HUECOS         [0]',/,
                  T10,'CON HUECOS         [1]',/,'*?'>);
    READ (TERM,/,METODO);
    CHECADATOT (METODO, 1);
    WRITE (SAL,<////,T40,'ESPERA UN MOMENTO....!'>);
    DIREC := 85;
    CAMINA (NUMOBST, RANA, PRESA, OBSTACULOS);
    IF ((NOT RETIRO) AND (NOT CABE)) THEN
        BEGIN
        LLENAESPACIO (RANA[0], RANA[1], PRESA[0], PRESA[1], 2);
        WRITE (OUT,<' LA RANA SE DIRIGE HACIA LA PRESA Y SE LA COME'>);
        IGUALA (RANA, PRESA, 2);
        ESPACIO[INTEGER(RANA[0]),INTEGER(RANA[1])] := '     R';
        END
    NOELSE
    ENDIF;
    ESPACIO[INTEGER(RANAINIC[0]),INTEGER(RANAINIC[1])] := '     R';
    WRITE(OUT [SKIP 1]);
    WRITE(OUT,<T40,'EL ESTADO FINAL ES:'>);
    IMPRIME;
ENDIFF;
WRITE(OUT,<////,T30,'SIMULACION DE ATAQUE TERMINADA'>);
END
ENDPROCEDURE ATAQUE;

PROCEDURE BUSCALUGAR ( PH, IND, MAT);
          ARRAY  PH[0];
          ARRAY MAT [-1,1];

          INTEGER IND;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%     RUTINA QUE ASIGNA LAS COORDENADAS QUE ESTAN DADAS SOBRE LA     %%
%%  MATRIZ DE VISION.                                                 %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
BEGIN
PH[0] := MAT[IND,3];
PH[1] := MAT[IND,4];
END
ENDPROCEDURE BUSCALUGAR;

PROCEDURE BUSCAPRESA( RANA, PRESA);
          ARRAY RANA,
                PRESA[0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                 %%
%%        RUTINA QUE MANDA EL AVISO AL ARCHIVO DE SALIDA DE QUE LA   %%
%%  POSICION DE LA RANA Y PRESA EN CIERTO MOMENTO DE LA SIMULACION   %%
%%                                                                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT FO (' ================================== ',/,
              ' LA RANA SE ENCUENTRA EN EL PUNTO    ',/,
              '       (',F5.2,',',F5.2,')'          ,/,
              ' Y LA PRESA EN EL PUNTO              ',/,
              '       (',F5.2,',',F5.2,')'          ,/,
              ' ================================== ',/);

WRITE(OUT, FO, RANA[0], RANA[1], PRESA[0], PRESA[1]);
ESPACIO[INTEGER(RANA[0]),INTEGER(RANA[1])] := '     R';
END
ENDPROCEDURE BUSCAPRESA;

PROCEDURE BUSCAPUNTO (RX, RY, PX, PY, ALCANCE, APERTURA, X, Y);
          REAL RX, RY,
               PX, PY,
               APERTURA,
               ALCANCE,
               X, Y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                 %%
%%      RUTINA QUE NOS DA EN 'X' 'Y' LAS COORDENADAS DE UN NUEVO PUN- %%
%%  TO, EL CUAL ESTA A LA DISTANCIA 'ALCANCE' Y ANGULO 'APERTURA' EN %%
%%  RADIANES, MAS CERCANO O AL PUNTO DEFINIDO POR 'RX, RY'           %%
%%                                                                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL PI,
         PENDIENTE,
         PENDPCPAL;

PI := 3.141592654;
IF RX NEQ PX THEN
    PENDPCPAL := ARCTAN ((PY-RY)/(PX-RX))
ELSE
    PENDPCPAL := PI/2;
ENDIF;
PENDIENTE := PENDPCPAL - APERTURA*PI/180;
PUNTOOBJ (RX, RY, PX, PY, PENDIENTE, ALCANCE, X, Y);
END
```

```
ENDPROCEDURE BUSCAPUNTO;

PROCEDURE CAMINA(NUMOBST, RANA, PRESA, OBSTACULOS);
          ARRAY  OBSTACULOS[*,*],
                 RANA,
                 PRESA[0];

          INTEGER NUMOBST;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%     RUTINA RECURSIVA QUE SIMULA EL CAMINAR DE LA RANA MIENTRAS         %%
%%  ESTA NO SE HAYA RETIRADO, O TENGA OBSTACULOS ENFRENTE, O NO -         %%
%%  PUEDA COMERSE A LA PRESA ENTRE LAS REJAS DE LA BARDA.                 %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
IF (NOT RETIRO) AND (NOT CABE) THEN
    BEGIN
    BUSCAPRESA( RANA, PRESA);
    CHECATRAYECTO (NUMOBST, OBSTACULOS, RANA, PRESA, HAYOBSTACULO,
                   CUAL, PUNTO);
    END
NOELSE
ENDIF;
IF CUAL GEQ 0 THEN
    BEGIN
    IF (HAYOBSTACULO) AND (NOT RETIRO) AND (NOT CABE) THEN
        BEGIN
        CHECAOBSTACULO (NUMOBST, RANA, PRESA, OBSTACULOS, CUAL,
                                          RETIRO, CABE, PUNTO);
        WRITE (OUT [SKIP 1]);
        IMPRIME;
        CAMINA (NUMOBST, RANA, PRESA, OBSTACULOS);
        END
    NOELSE
    ENDIF;
    END
ENDIF;
END
ENDPROCEDURE CAMINA;

PROCEDURE CAMINARANA(IND, CEL, R, RANA, PP);
          ARRAY R,
                RANA[0],
                CEL[*,*];

          INTEGER IND,
                  PP;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%       RUTINA QUE ASIGNARA NUEVAS COORDENADAS A LA RANA, ESTO ES        %%
%%  LA RANA CAMINARA, DENTRO DE UN HUECO PREVIAMENTE SELECCIONADO.        %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL INC,
```

```
          M,
          FEND,
          PTOX,
          PTOY,
          0,
          INCRDIST,
          SEPAR);

IF (CELCIND+1,2]=CELCIND-1,2]) OR (IND<2) OR (IND>168) THEN
    BEGIN
    INC     := 0;
    PTOX    := CELCIND,3];
    PTOY    := CELCIND,4];
    CASE PP OF
        BEGIN
        1:                          %PRESA
          INCRDIST := -(DSTN(PTOX,PTOY,RANAC0],RANAC1])) / 1.5;
        2:3:4:                      %PREDADOR Y RESPUESTA FOTOTAXICA
          INCRDIST :- -1;
        END
    ENDCASE;
    END
ELSE
    BEGIN
    IF CELCIND+1,1] > CELCIND-1,1] THEN
        BEGIN
        INC  := -0.052;
        PTOX := CELCIND-1,3];
        PTOY := CELCIND-1,4];
        END
    ELSE
        BEGIN
        INC  := 0.052;
        PTOX := CELCIND+1,3];
        PTOY := CELCIND+1,4];
        END
    ENDIF;
    IF PP=3 THENN
        INC  := 0;
        PTOX := CELCIND,3];
        PTOY :- CELCIND,4];
    NOELSE
    ENDIFF;
%
%   PARA DEFINIR EL INCREMENTO DE DISTANCIA A DARLE AL HUECO
%
    SEPAR := ABS(CELCIND+1,1] - CELCIND-1,1]);
    CASE PP OF
       BEGIN
       1:              % PRESA
         INCRDIST := SEPAR * 0.3;
       2:4:            % PREDADOR
         INCRDIST := SEPAR * 0.82;
       3:              % RESPUESTA FOTOTAXICA
         INCRDIST := SEPAR * 0.6;
       END
    ENDCASE;
    END
```

```
ENDIF;
IF IGUAL(R[0],PTOX) THEN
    PEND := (3.1416/2)+INC
ELSE
    BEGIN
    M    := (PTOY-R[1])/(PTOX-R[0]);
    PEND := ARCTAN(M)+INC;
    END
ENDIF;
D := DSTN (R[0], R[1], PTOX, PTOY) + INCRDIST;
PUNTOOBJ(R[0], R[1], PTOX, PTOY, PEND, D, RANA[0], RANA[1]);
END
ENDPROCEDURE CAMINARANA;

PROCEDURE CAPTAHUECOS (CEL, HUECO, HUECOS);
          ARRAY   CEL,
                  HUECO[*,*];

          INTEGER HUECOS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%        RUTINA QUE SE ENCARGA DE CAPTAR LOS HUECOS QUE LA RANA ES-   %%
%%  TA CAPTANDO ENTRE SU POSICION Y LA POSICION QUE TIENE LA PRESA     %%
%%  O EL PUNTO HACIA EL CUAL ESTA VIENDO.                              %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I;
    REAL    DIFIZQ,
            DIFDER;

I      := 0;
HUECOS:= 0;
WHILE I < 171 DOO
    IF CEL[I,2] < 0 THENN
        HUECOS            := *+1;
        HUECO[HUECOS,1] := I;
        WHILE CEL[I,2] < 0 AND I < 170 DOO
            IF I < 171 THEN
               I := *+1;
            ENDIF;
        ENDWHILEE;
        IF I < 170 THENN
           HUECO[HUECOS,2] := I-1;
        ELSEE
           HUECO[HUECOS,2] := I;
        ENDIFF;
        HUECO[HUECOS,3] := HUECO[HUECOS,2]-HUECO[HUECOS,1]+1;
    ENDIFF;
    IF I < 171 THEN
       I := *+1;
    ENDIF;
ENDWHILEE;
%%
%%% PROCESO PARA DEFINIR DIFIZQ Y DIFDER
%%
FOR I:=1 STEP 1 UNTIL HUECOS DOO
```

```
    DIFIZQ      := ABS ( CELCHUECO[I,1]  ,1]-
                         CELCHUECO[I,1]-1,1]);
    DIFDER      := ABS ( CELCHUECO[I,2]  ,1] -
                         CELCHUECO[I,2]+1,1] );
    HUECO[I,4] := DIFIZQ;
    HUECO[I,5] := DIFDER;
ENDFORR;
END
ENDPROCEDURE CARTAHUECOS;

PROCEDURE CARACOBSTACULOS (MATOBST, NUMOBST);
          ARRAY   MATOBST [*,4];
          INTEGER NUMOBST;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                   %%
%%       RUTOINA QUE PIDE EL NUMERO DE OBSTACULOS QUE VAN A COLOCAR  %%
%%                                                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT F0 (8(/),T20,' DAME LAS CARACTERISTICAS DE LOS OBSTACULOS'),
           F1 (/,/,' CUAL ES EL NUMERO DE OBSTACULOS?',/,'$?'),
           F2 (' NO DESEA OBSTACULOS..OK!!!!',/);

WRITE (SAL,BORRASCREEN);
WRITE (SAL,F0);
WRITE (SAL ,F1);
READ  (ENT[ARI],/,NUMOBST);
WRITE (SAL,/,NUMOBST);
WHILE NUMOBST>20 OR NUMOBST<0 DO
    BEGIN
    WRITE(SAL, ' EL NUMERO DE OBSTACULOS NO PUEDE SER MAYOR A',
               ' 20 NI MENOR QUE CERO.',/,
               ' QUIERES DARLO NUEVAMENTE??',/,'$?'>);
    READ(ENT[ARI],/,NUMOBST);
    END
ENDWHILE;
IF NUMOBST > 0 THEN
   DEFINEOBSTACULO (NUMOBST, MATOBST);
ELSE
   WRITE (SAL,F2);
ENDIF;
END
ENDPROCEDURE CARACOBSTACULOS;

PROCEDURE CARACPRESAPRED (PRESA);
          ARRAY PRESA [*];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                   %%
%%       RUTINA QUE PIDE LA POSICION DE LA PRESA.                    %%
%%                                                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT F  (8(/),T20,' DAME LAS CARACTERISTICAS DE LA PRESA ',
                    'O PREDADOR',///),
           F0 (' POSICION DE LA COORDENADA X',/,'$?'),
           F1 (' POSICION DE LA COORDENADA Y',/,'$?');
```

```
     SWITCH FORMAT SF := F0,F1;
     INTEGER I;

 WRITE (SAL, BORRASCREEN);
 WRITE (SAL,F);
 FOR I:=0 STEP 1 UNTIL 1 DO
     BEGIN
     WRITE (SAL,SF[I]);
     READ  (ENT[AR],/,PRESA [I]);
     WRITE (SAL,/,PRESA[I]);
     CHECADATO (PRESA[I],60-I*40);
     END
 ENDFOR;
 ESPACIO[INTEGER(PRESA[0]),INTEGER(PRESA[1])]:='      P';
 END
 ENDPROCEDURE CARACPRESAPRED;

 PROCEDURE CARACRANA (RANA, PP);
          ARRAY    RANA [*];
          INTEGER  PP;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                 %%
%%      RITINA QUE SOLICITA LAS CARACTERISTICAS DE LA RANA.        %%
%%                                                                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 BEGIN
     FORMAT F  (B(/),T20,' DAME LAS CARACTERISTICAS DE LA RANA',////),
            F0 (' POSICION DE LA COORDENADA X',/,'#?'),
            F1 (' POSICION DE LA COORDENADA Y',/,'#?'),
            F2 (' ALTURA DE SALTO DE BARDA',/,'#?'),
            F3 (' LONGITUD DE SALTO DE ZANJA',/,'#?'),
            F4 (' LONGITUD DE LA LENGUA',/,'#?'),
            F5 (' LONGITUD DE SALTO DE HUIDA',/,'#?');
     SWITCH FORMAT SF := F0,F1,F2,F3,F4,F5 ;
     INTEGER I ;

 WRITE (SAL,BORRASCREEN);
 WRITE (SAL,F);
 WRITE (SAL,SF[0]);
 READ  (ENT[AR],/,RANA[0]);
 WRITE (SAL,/,RANA[0]);
 CHECADATO (RANA[0],60);
 WRITE (SAL,SF[1]);
 READ  (ENT[AR],/,RANA[1]);
 WRITE (SAL,/,RANA[1]);
 CHECADATO (RANA[1],20);
 IF PP NEQ 3 THENN
     IF PP=2 THENN
         WRITE (SAL, SF[5]);
         READ  (ENT[AR],/,RANA[5]);
         WRITE (SAL,/,RANA[5]);
         CHECADATO (RANA[5], 20);
     ELSEE
         FOR I:=2 STEP 1 UNTIL 4 DO
             BEGIN
             WRITE (SAL,SF[I]);
             READ  (ENT[AR],/,RANA[I]);
```

```
            WRITE (SAL,/,RANA[I]);
            CHECADATO (RANA[I], 100);
            END
        ENDFOR;
    ENDIFF;
ENDIFF;
ESPACIO[INTEGER(RANA[0]),INTEGER(RANA[1])]:='      R';
END
ENDPROCEDURE CARACRANA;

PROCEDURE CHECADATO (DATO, LIM);
          INTEGER LIM;
          REAL    DATO;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%      RUTINA QUE VERIFICA QUE EL DATO DADO SEA CONGRUENTE.                %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT F ('EL ULTIMO DATO QUE DISTE ES INCORRECTO ',
              /,' EL VALOR TIENE QUE ESTAR ENTRE O Y ',J3,
              /,' QUIERES DARLO NUEVAMENTE..?',/,'#?');

WHILE (DATO>LIM OR DATO<0) DO
    BEGIN
    WRITE (SAL ,F,LIM);
    READ  (ENT[AR],/,DATO);
    END
ENDWHILE;
END
ENDPROCEDURE CHECADATO;

PROCEDURE CHECADATOT (DATO,LIM);
          INTEGER LIM;
          REAL    DATO;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%       RUTINA QUE VERIFICA QUE EL DATO DADO POR TERMINAL SEA CON-         %%
%%  GRUENTE.                                                                %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT F ('EL ULTIMO DATO QUE DISTE ES INCORRECTO ',
              /,' EL VALOR TIENE QUE ESTAR ENTRE O Y ',J3,
              /,' QUIERES DARLO NUEVAMENTE..?',/,'#?');

WHILE (DATO>LIM OR DATO<0) DO
    BEGIN
    WRITE (TERM[0],F,LIM);
    READ  (TERM,/,DATO);
    END
ENDWHILE;
END
ENDPROCEDURE CHECADATOT;

PROCEDURE CHECAOBSTACULO( NUMOBST, RANA, FRESA, OBST, CUAL,
```

```
                        RETIRO, CABE, PUNTO);
         ARRAY OBST[*,*],
               RANA,
               PRESA,
               PUNTO[0];

        BOOLEAN RETIRO,
                CABE;

        INTEGER CUAL,
                NUMOBST;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%     RUTINA QUE SE ENCARAGA DE DETERMINAR CON QUE TIPO DE OBSTA-     %%
%%  CULO SE HA CRUZADO LA RANA, Y LLAMA A LA RUTINA ADECUADA PARA      %%
%%  CONTINUAR CON EL PROCESO DE SIMULACION.                            %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY PUNTO1[0:1];
    REAL DIST;

IF IGUAL (OBST[CUAL,0],1) THEN
    BEGIN
    MENSAJE (OBST, 7);      %SE TOPA CON LA BARDA 'CUAL'
    SIALCANZA := (SQRT(((PUNTO[0]-PRESA[0])**2)+
                    ((PUNTO[1]-PRESA[1])**2))) LEQ (RANA[4]-1);
    SISALTA   :=  OBST[CUAL,9] <= RANA[2];
    PROCESABARDA (RANA, PUNTO, OBST, CUAL, SIALCANZA, SISALTA);
    END
ELSE
    BEGIN
    IGUALA (RANANT, RANA, 2);
    NUEVOPUNTO (RANA[0], RANA[1], PUNTO[0], PUNTO[1],
                RANA[0], RANA[1], PRESA[0], PRESA[1], FALSE);
                                        % ASIGNA NUEVAS COORDENADAS
                                        % A RANA UN POCO ANTES
                                        % DE LA ZANJA.
    MENSAJE (OBST,8);         % SE TOPA CON LA ZANJA 'CUAL'
                              % EN LAS COORD X= PUNTO[0],
                              % Y=PUNTO[1]
    LLENAESPACIO (RANA[0], RANA[1], RANANT[0], RANANT[1], 2);
    SIBAJA  := OBST[CUAL,9] <= RANA[2];
    IGUALA (RANAP,  RANA, 6);
    IGUALA (PUNTOP, PUNTO, 2);
    SIBRINCA := PUEDEBRINCAR (OBST, RANAP, PRESA, PUNTOP, DIST);
    PROCESAZANJA (RANA, PUNTO, SIBAJA, SIBRINCA, DIST);
    END
ENDIF;
END
ENDPROCEDURE CHECAOBSTACULO;

PROCEDURE CHECATIPO (TIPO);
        REAL TIPO;
```

APENDICE C

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                  %%
%%       RUTINA QUE VERIFICA QUE EL TIPO DE OBJETO SEA El CORRECTO,  %%
%%  BARDA O ZANJA.                                                  %%
%%                                                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT F (' EL ULTIMO DATO DEBERA SER : ',
           /,' 0 ---> PARA ZANJA',
           /,' 1 ---> PARA BARDA',
           /,' DA EL TIPO CORRECTO NUEVAMENTE..',/,'$?');

WHILE ((TIPO NEQ 1.) AND (TIPO NEQ 0.)) DO
    BEGIN
    WRITE (SAL,F);
    READ  (ENT[AR],/,TIPO);
    WRITE (SAL,/,TIPO);
    END
ENDWHILE;
END
ENDPROCEDURE CHECATIPO;

PROCEDURE CHECATRAYECTO (NUMOBS, OBS, RANA, OBJETIVO,
                            ESTORBA, OBSNUM, PCOMUN);
         ARRAY   OBS [*,*],
                 RANA,
                 PCOMUN,
                 OBJETIVO [*];

         BOOLEAN ESTORBA;

         INTEGER NUMOBS,
                 OBSNUM;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                 %%
%%      LA SUBRUTINA 'CHECATRAYECTO' SE ENCARGA DE INSPECCIONAR -  %%
%% LA TRAYECTORIA QUE DEBE SEGUIR LA RANA Y DETECTAR EL OBSTACULO %%
%% MAS PROXIMO A LIBRAR PARA LLEGAR AL OBJETIVO.                   %%
%%                                                                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     INTEGER J,
             K,
             L;

     REAL    DISTANCIA;

ESTORBA   := FALSE;
DISTANCIA := 99999;
FOR J := -4 STEP 1 UNTIL NUMOBS DO
    BEGIN
    CASE OBS[J,0] OF              % TIPO DE OBSTACULO
         BEGIN
         0:BEGIN                  % SE TRATA DE UN POZO
           FOR K := 1 STEP 2 UNTIL 5 DO
               BEGIN
```

```
                L:=K+2;
                ANALIZA (OBS[J,*], RANA, OBJETIVO, DISTANCIA, J, K, L,
                                    ESTORBA,  OBSNUM,    PCOMUN );
                END
             ENDFOR;
             K:=7;
             L:=1;
             ANALIZA (OBS[J,*], RANA, OBJETIVO, DISTANCIA, J, K, L,
                                 ESTORBA,  OBSNUM, PCOMUN );
             END;
           1:BEGIN              %SE TRATA DE UNA BARDA
             K:=1;
             L:=3;
             ANALIZA (OBS[J,*], RANA, OBJETIVO, DISTANCIA, J, K, L,
                                 ESTORBA,  OBSNUM, PCOMUN );
             END;
           END
        ENDCASE;
        END
ENDFOR;
END
ENDPROCEDURE CHECATRAYECTO;

PROCEDURE COLOCAANCHO (MATHUECOS, NHUECOS, VECANCHOS);
          REAL ARRAY MATHUECOS [*,*],  % MATRIZ DE HUECOS
                     VECANCHOS [*]  ;  % VECTOR DE ANCHOS DE HUECOS

          INTEGER    NHUECOS        ;  % NUMERO DE HUECOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%       RUTINA QUE EXTRAE LOS VALORES DE LOS ANCHOS DE LOS HUECOS    %%
%%   PARA INTRODUCIRLOS A UN VECTOR.                                  %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I,J;              % VARIABLES PARA FOR'S

FOR I:=1 STEP 1 UNTIL NHUECOS DO
    FOR J:=MATHUECOS[I,1] STEP 1 UNTIL MATHUECOS[I,2] DOO
        VECANCHOS[J]:=MATHUECOS[I,3];
    ENDFORR;
ENDFOR;
END
ENDPROCEDURE COLOCAANCHO;

PROCEDURE CONTROLAESPACIO (I, MATOBST);
          ARRAY   MATOBST [*,*];
          INTEGER I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%       RUTINA QUE CONTROLA LA COLOCACION DE LOS OBJETOS EN LA MA-   %%
%%   TRIZ DE ESPACIO.                                                 %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL TIPO;
```

```
TIPO := MATOBST[I,0];
LLENAESPACIO (MATOBST[I,1], MATOBST[I,2], MATOBST[I,3],
                            MATOBST[I,4], TIPO);
IF TIPO EQL 0.0 THEN
    BEGIN
    LLENAESPACIO (MATOBST[I,1], MATOBST[I,2], MATOBST[I,7],
                                MATOBST[I,8], TIPO);
    LLENAESPACIO (MATOBST[I,5], MATOBST[I,6], MATOBST[I,7],
                                MATOBST[I,8], TIPO);
    LLENAESPACIO (MATOBST[I,5], MATOBST[I,6], MATOBST[I,3],
                                MATOBST[I,4], TIPO);
    END;
NOELSE
ENDIF
END
ENDPROCEDURE CONTROLAESPACIO;

PROCEDURE DEFINEOBSTACULO (NUMOBST, MATOBST);
          ARRAY   MATOBST [*,*];
          INTEGER NUMOBST;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%       RUTINA QUE PIDE LA DESCRIPCION DEL TIPO DE OBTACULO.             %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I;
    FORMAT FX (3(/),T20,' TECLE LOS CUATRO PUNTOS QUE FORMAN',/,
               T20,' LA ZANJA EN SENTIDO DEL RELOJ....OK? '),
           F  (//,T20,' SI EL OBSTACULO ES ZANJA TECLEA [0]',/,
                  T20,' SI ES BARDA TECLEA               [1]'),
           F0 (////,T20,' DAME LAS CARACTERISTICAS DEL OBSTACULO #',J3),
           F1 (//,T20,' QUE TIPO DE OBSTACULO ES?',/,'$?');

WRITE (SAL,BORRASCREEN);
WRITE (SAL,FX);
WRITE (SAL,F);
FOR I := 0 STEP 1 UNTIL NUMOBST-1 DO
    BEGIN
    WRITE (SAL,F0,I+1);
    WRITE (SAL,F1);
    READ  (ENT[AR],/,MATOBST[I,0]);
    WRITE (SAL,/,MATOBST[I,0]);
    CHECATIPO  (MATOBST[I,0]);
    PIDECOORDS (I, MATOBST);
    END
ENDFOR;
END
ENDPROCEDURE DEFINEOBSTACULO;

PROCEDURE DETERMINAHUECO (MATHUECOS, NHUECOS, VECANCHOS, VECPROF,
                          VECGAUSS, INDICE, VECCOLOR, C1, C2, C3, PP);
    REAL ARRAY MATHUECOS [*,*] ,    % MATRIZ DE HUECOS
               VECANCHOS [*]   ,    % VECTOR DE ANCHOS DE HUECOS
               VECPROF   [*]   ,    % VECTOR DE PROFUNDIDAD DE HUECOS
               VECGAUSS  [*]   ,    % VECTOR DE VALORES GAUSSIANOS
```

```
                 VECCOLOR  [*]    ;    % VECTOR DE COLORES DEL FONDO

     INTEGER     NHUECOS          ,    % NUMERO DE HUECOS
                 INDICE           ,    % INDICE DEL HUECO OPTIMO
                 C1               ,    % CTE PARA EL ANCHO
                 C2               ,    % CTE PARA LA PROFUNDIDAD
                 C3               ,    % CTE PARA LA GAUSSIANA
                 PF               ;    % CASO SIMULACION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%       RUTINA QUE DETERMINA EL HUECO OPTIMO A SER UTILIZADO EN LA         %%
%%  RUTINA 'PANA-PRESA'.    DETERMINA EL INDICE DEL HUECO EN LA MA-         %%
%%  TRIZ DE DISTANCIAS.                                                     %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL ARRAY VECEVAL [0:171] ;    % VECTOR DE EVALUACION DE HUECO
    REAL       A,B,C            ,    % VALORES PORCENTUALES DE C1,C2 Y C3
               MAYOR            ;    % VALOR MAXIMO DEL HUECO
    INTEGER    I,J              ;    % VARIABLES PARA LOS FOR'S

A := C1/100;
B := C2/100;
C := C3/100;
FOR I := 1 STEP 1 UNTIL NHUECOS DO
    FOR J := MATHUECOS[I,1] STEP 1 UNTIL MATHUECOS[I,2] DO
        VECEVAL[J] :=  A*VECANCHOS[J]
                      +B*VECPROF  [J]
                      +C*VECGAUSS [J];
    ENDFOR;
ENDFOR;
FOR J := 0 TO 170 DOO
    VECEVAL [J] := *+VECCOLOR [J];
ENDFORR;
IF PF=2 THENN                          %PREDADOR
    FOR J := 0 TO 170 DOO
        VECEVAL [J] := *+(C*VECGAUSS[J]);
    ENDFORR;
ENDIFF;
WRITE(OUT [SKIP 1]);
WRITE(OUT,<10(/),T40    ,'LA MATRIZ DE EVALUACION DE HUECOS QUEDA:'>);
WRITE(OUT,<3(/),T54,F7.2>,VECEVAL[0]);
FOR I := 0 STEP 1 UNTIL 16 DO
    WRITE (OUT,<T10,I3,' ----> ',10F7.2,'  ----> ',I3,/>, (I*10+1),
            FOR J:=1 STEP 1 UNTIL 10 DO VECEVAL[I*10+J], (I+1)*10);
ENDFOR;
WRITE (OUT,<//,T20,'LOS VALORES DE LAS CONSTANTES ANCHO, ',
                   'PROFUNDIDAD Y GAUSS SON:',/,T40,3I6,//,
              T20,'Y EL GRADO DE LA FUNCION GAUSIANA ES:',/,
              T40,I6,////> ,C1, C2, C3, GRADO);
MAYOR  := VECEVAL[85];
INDICE := 85;
FOR I := 84 STEP -1 UNTIL 0 DO
    BEGIN
    IF VECEVAL[I] > MAYOR THEN
       BEGIN
       MAYOR  := VECEVAL[I];
```

```
          INDICE := I;
          END;
     ENDIF
     END;
ENDFOR;
FOR I := 86 STEP 1 UNTIL 170 DO
     BEGIN
     IF VECEVAL[I] > MAYOR THEN
          BEGIN
          MAYOR  := VECEVAL[I];
          INDICE := I;
          END;
     ENDIF
     END;
ENDFOR
END
ENDPROCEDURE DETERMINAHUECO;

PROCEDURE DETERMINAHUECOINT (PP);
          INTEGER PP;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%        RUTINA QUE SE UTILIZO PARA DETERMINAR LAS CONSTANTES DE           %%
%%  ANCHO, PROFUNDIDAD Y FACTOR DE GAUSS, CON EL FIN DE QUE EL MO-          %%
%%  DELO TRABAJARA LO MAS ADECUADAMENTE POSIBLE.  LA DETERMINACION          %%
%%  SE HACIA EN BASE A DAR DE DATO INTERACTIVAMENTE ESTAS CONSTAN-          %%
%%  TES DE PROPORCION Y OBSERVAR LOS RESULTADOS DEL MODELO.                 %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


BEGIN
WRITE (SAL,<//, ' DAME LOS VALORES DE LAS CONSTANTES ',/,
                ' EN EL ORDEN : ANCHO,PROFUNDIDAD Y GAUSS'>,);
READ(TERM,/,C1,C2,C3);
WHILE (C1+C2+C3) > 0 DO
     BEGIN
     IF (C1+C2+C3) EQL 100 THEN
          BEGIN
          DETERMINAHUECO ( HUECO, HUECOS, VECANCHOS, VECPROF,
                           VECGAUSS, INDICE, VECCOLOR, C1, C2, C3, PP);
          END
     ELSE
          BEGIN
          WRITE (SAL,<' LA DATOS ANTERIORES SON INCORRECTOS',/,
                      ' DEBERAN SUMAR 100 '>,);
          WRITE (SAL,<' Y SUMAN -->',F10.1>,C1+C2+C3);
          END
     ENDIF;
     WRITE (SAL,< ' DAME LAS CONSTANTES DE ANCHO,PROF Y GAUSS'>,);
     READ(TERM,/,C1,C2,C3);
     END
ENDWHILE;
END
ENDPROCEDURE DETERMINAHUECOINT;

REAL PROCEDURE DSTN (X1, Y1, X2, Y2);
               REAL X1, Y1,
```

```
                X2, Y2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%      LA SUBRUTINA 'DSTN' CALCULA LA DISTANCIA QUE EXISTE ENTRE      %%
%% LOS PUNTOS (X1,Y1) Y (X2,Y2).                                       %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
DSTN := ((X1-X2)*(X1-X2)+(Y1-Y2)*(Y1-Y2))**0.5;
END
ENDPROCEDURE DSTN;

PROCEDURE ELIMINAPEQUES (HUECO, HUECOS);
          ARRAY   HUECO[*,*];
          INTEGER HUECOS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%       RUTINA QUE SE ENCARGA DE  DESCARTAR DEL VECTOR DE HUECOS,     %%
%%  A AQUELLOS  QUE TIENEN UN ANCHO DEMASIADO PEQUE%O.                 %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I,
            J,
            K;
I := 1;
WHILE I LEQ HUECOS DO
    BEGIN
    IF HUECO[I,3] < 4 THEN
       BEGIN
       FOR J := I STEP 1 UNTIL HUECOS DO
           FOR K := 1 STEP 1 UNTIL 5 DO
               HUECO[J,K] := HUECO[(J+1),K];
           ENDFOR;
       ENDFOR;
       I        := I-1;
       HUECOS := HUECOS-1;
       END
    NOELSE
    ENDIF;
    I := I+1;
    END
ENDWHILE;
END
ENDPROCEDURE ELIMINAPEQUES;

PROCEDURE ERROROBST (NUM,PXY,I);
          ARRAY   PXY [*];

          INTEGER NUM,
                  I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%       RUTINA DE AVISO DE COORDENADAS ERRONEAS PARA LOS OBSTACULOS %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
BEGIN
    FORMAT FO (' EXISTE ERROR EN TUS DATOS!!!!!!',
                ' LAS COORDENADAS DE TU ULTIMO OBSTACULO',
                ' SE INTERSECTAN CON EL OBSTACULO # ',J3,/,
                ' EN EL PUNTO (',F5.2,',',F5.2,')',/,
                ' DA LAS COORDENADAS NUEVAMENTE..OK  ?? ');
I:=*-1;
WRITE (SAL,FO,NUM+1,PXY[0],PXY[1]);
END
ENDPROCEDURE ERROROBST;

PROCEDURE ESCOGE_IND (IND, HUECO, NHUECOS, MAT);
          ARRAY   HUECO,
                  MAT[*,*];

          INTEGER IND,
                  NHUECOS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                %%
%%         RUTINA QUE SE ENCARGA DE SELECCIONAR EL EL HUECO EN QUE %%
%%  SE ENCUENTRA CUAL ES EL LUGAR MAS CERCANO A LA RANA, ES DECIR, %%
%%  COMO EN RESPUESTA FOTOTAXICA LA RANA ESCOGIO  UN HUECO QUE PO- %%
%%  SEIA LAS CARACTERISTICAS MAS IDONEAS,  ESTA  RUTINA  ESCOGE EL %%
%%  LUGAR MAS CERCANO EN DISTANCIA A LA RANA.                      %%
%%                                                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


BEGIN
    INTEGER I,
            CUAL;
    REAL    MENOR;

CUAL  := 0;
FOR I := 1 TO NHUECOS DOO
    IF (IND GEQ HUECO[I,1]) AND (IND LEQ HUECO[I,2]) THENN
       CUAL := I;
    NOELSE
    ENDIFF;
ENDFORR;
IF CUAL=0 THENN                        %IND EN HUECO?
    %NADA
ELSEE
    MENOR := MAT[IND,1];
    FOR I := HUECO[CUAL,1] TO HUECO[CUAL,2] DOO
        IF MAT[I,1] < MENOR  AND MAT[I,5]=MAT[IND,5] THENN
           MENOR := MAT[I,1];
           IND   := I;
        NOELSE
        ENDIFF;
    ENDFORR;
ENDIFF;
END
ENDPROCEDURE ESCOGE_IND;

PROCEDURE ESCRIBECARACTERISTICAS (OBST, NUMOBST, CASO);
          ARRAY   OBST[*,*];
```

```
        INTEGER NUMOBST,
                CASO;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%      RUTINA QUE SE ENCARGA DE IMPRIMIR AL ARCHIVO DE SALIDA LAS        %%
%%  CARACTERISTICAS QUE TENDRA LA RANA, PRESA/PREDADOR Y OBSTACULOS       %%
%%  PARA LA SIMULACION.                                                   %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     ARRAY   TIPO,
             COLOR [0:0];
     INTEGER I;

WRITE (OUT [SKIP 1]);
WRITE (OUT [SKIP 1]);
CASE CASO OF
    BEGIN
    0: WRITE (OUT, <//////,T8,'CARACTERISTICAS DE LA RANA:',/,
                         T23,'POSICION X Y--> ', 2F7.2,/,
                         T22,'SALTO ALTURA--> ', F7.2,/,
                         T20,'SALTO LONGITUD--> ',F7.2,/,
                         T20,'ALCANCE LENGUA--> ',F7.2 >,
                                        RANA[0], RANA[1],
                                RANA[2], RANA[3], RANA[4]);
       WRITE (OUT, <//////,T8,'CARACTERISTICAS DE LA PRESA:',/,
                         T23,'POSICION X Y--> ',2F7.2 >,
                                        PRESA[0], PRESA[1]);
       WRITE (OUT, <//////,T24,'CARACTERISTICAS DE LOS OBSTACULOS:',/,
                           T8, 65('*'),/,
                           T8,'* # OBST ! TIPO OBST ! X1-Y1 ! X2-Y2 ! ',
                              'X3-Y3 ! X4-Y4 ! ALTURA O *',/,
                           T8,'*',T17,'!',T29,'!',T37,'!',T45,'!',
                              T53,'!',T61,'!',T65,'PROF.   *',/,
                           T8,'*',63('='),'*' >);
       FOR I:=0 STEP 1 UNTIL NUMOBST-1 DO
           BEGIN
           TIPO[0]:='BARDA ';
           IF OBST[I,0] = 0 THEN
              TIPO[0] := 'ZANJA ';
           ENDIF;
           WRITE (OUT, <T8,'*',I5,'   !    ',A5,'  !',4(F6.2,' !'),
                        F7.2,'   *',/,
                    T8,'*',T17,'!',T29,'!',4(F6.2,' !'),T72,'*',/,
                    T8,'*',T17,'!',T29,'!',T37,'!',T45,'!',T53,'!',
                       T61,'!',T72,'*'>, I, P(TIPO), OBST[I,1],
                               OBST[I,3], OBST[I,5], OBST[I,7],
                               OBST[I,9], OBST[I,2], OBST[I,4],
                                          OBST[I,6], OBST[I,8]);
           WRITE (OUT,<T8,'*',63('-'),'*'>);
           END
       ENDFOR;
       WRITE (OUT, <T8,65('*')>);
    1: WRITE (OUT, <//////,T8,'CARACTERISTICAS DE LA RANA:',/,
                       T23,'POSICION X Y--> ',2F7.2>,
                                        RANA[0], RANA[1]);
       WRITE (OUT, <//////,T8,'CARACTERISTICAS DEL PREDADOR:',/,
```

```
                        T23,'ANGULO, DIST--> ',2I7,/,
                        T23,'POSICION X Y--> ',2F7.2, ANG_DIREC,
                                    DIST_DIREC, PREDADOR[0],
                                                PREDADOR[1]);
        WRITE (OUT,</////,T6,'ORIENTADA HACIA EL PUNTO X Y--> ',2F7.2),
                                    PUNTO_VISION[0], PUNTO_VISION[1]);
    2: WRITE (OUT,</////,T9,'CARACTERISTICAS DE LA RANA:',/,
                        T22,'POSICION X Y--> ',2F7.2,/,
                        T6,'ORIENTADA HACIA EL PUNTO X Y--> ',2F7.2),
                                    RANA[0],    RANA[1],
                                    PUNTODOR[0], PUNTODOR[1]);
    END
ENDCASE;
IF CASO>0 THENN
    WRITE (OUT, </////,T24,'CARACTERISTICAS DE LOS OBSTACULOS:',/,
                        T5,74('*'),/,
                        T5,'* # OBST ! TIPO OBST ! X1-Y1 ! X2-Y2 ! X3-Y3',
                            ' ! X4-Y4 ! ALTURA O   COLOR  *',/,
                        T5,'*',T14,'!',T26,'!',T34,'!',T42,'!',T50,'!',T5,
                            T62,'PROF.  ',T79,'*',/,
                        T5,'*',72('-'),'*' );
    FOR I := -4 STEP 1 UNTIL NUMOBST-1 DO
        BEGIN
        TIPO[0] := 'BARDA ';
        IF OBST[I,0] = 0 THEN
            TIPO[0] := 'ZANJA ';
        ENDIF;
        CASE OBST[I,1] OF
            BEGIN
            0: COLOR[0] := 'NEGRO ';
            1: COLOR[0] := 'AZUL  ';
            2: COLOR[0] := 'AMARIL';
            3: COLOR[0] := 'VERDE ';
            4: COLOR[0] := 'ROJO  ';
        ELSE: COLOR[0]  := 'NEGRO ';
            END
        ENDCASE;
        WRITE (OUT, <T5,'*',I5,'  !    ',A5,'  !',4(F6.2,' !'),
                        F7.2,'   ! ',A6,' *',/,
                    T5,'*',T14,'!',T26,'!',4(F6.2,' !'),T69,'!',
                        T78,'*',/,
                    T5,'*',T14,'!',T26,'!',T34,'!',T42,'!',T50,'!',
                        T58,'!',T69,'!',T78,'*'>, I, P(TIPO),
                                    OBST[I,1], OBST[I,3], OBST[I,5],
                                    OBST[I,7], OBST[I,9], P(COLOR),
                                    OBST[I,2], OBST[I,4], OBST[I,6],
                                    OBST[I,8]);
        WRITE (OUT,<T5,'*',72('-'),'*'>);
        END
    ENDFOR;
    WRITE (OUT,<T5,74('*')>);
ENDIFF;
END
ENDPROCEDURE ESCRIBECARACTERISTICAS;

PROCEDURE EVALUACOLOR (VEC, MAT);
          ARRAY VEC [0],
                MAT[-1,1];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%        RUTINA QUE DETERMINA EL PESO QUE TENDRA EL COLOR DEL HUECO      %%
%%   PARA LA EVALUACION DEL MISMO.                                        %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY   PESOCOLOR [0:5];
    INTEGER J;
    REAL    VALOR;

FILL PESOCOLOR [*] WITH 0, 7, 5, 4, 2.5, 1;
FOR J := 0 TO 170 DOO
    VALOR  := MAT[J,5];
    VEC[J] := PESOCOLOR[VALOR];
ENDFORR;
END
ENDPROCEDURE EVALUACOLOR;

PROCEDURE EVITA (RANA, PRED, OBST, CASO_PRED);
          ARRAY RANA,
                PRED [0],
                OBST [-4, 0];

          INTEGER CASO_PRED;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                        %%
%%      RUTINA QUE SE ENCARGA DE COORDINAR LAS LLAMDAS A LAS RUTINAS      %%
%%   NECESARIAS PARA QUE LA RANA HUYA DEL PREDADOR QUE LA ACOSA.          %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY PUNTOHUIDA [0:1];

WRITE (OUT [SKIP 1]);
WRITE (OUT,<T38,'EL ESTADO INICIAL EN EL CUAL',/,
            T33,'LA RANA SE DA CUENTA DEL PREDADOR ES:'>);
ESPACIO [INTEGER(RANA[0]),   INTEGER(RANA[1])]    := '     R';
IMPRIME;
PROCESAHUECOS (NUMOBST, OBST, RANA, PUNTO_VISION, MATCEL, HUECO,
               HUECOS, VECANCHOS, VECPROF, VECGAUSS, INDICE, 2);
BUSCALUGAR    ( PUNTOHUIDA, INDICE, MATCEL);
WRITE (OUT,<////,T20,'EL MEJOR LUGAR X-Y PARA HUIR ES:',
         2F10.2>,PUNTOHUIDA[0], PUNTOHUIDA[1]);
IF CASO_PRED=0 THENN
  PROCESAHUECOS (NUMOBST, OBST, RANA, PUNTOHUIDA, MATCEL, HUECO,
                 HUECOS, VECANCHOS, VECPROF, VECGAUSS, INDICE, 2);
  BUSCALUGAR    (PUNTOHUIDA, INDICE, MATCEL);
ENDIFF;
IGUALA      (RANAINIC, RANA, 2);
IGUALA      (RANANT,   RANA, 2);
CAMINARANA (INDICE, MATCEL, RANANT, RANA, 4);
LLENAESPACIO (RANANT[0], RANANT[1], RANA[0], RANA[1], 2);
ESPACIO [INTEGER(RANAINIC[0]), INTEGER(RANAINIC[1])]:='     R';
ESPACIO [INTEGER(RANA[0]),     INTEGER(RANA[1])]    :='     R';
WRITE (OUT [SKIP 1]);
```

```
WRITE (OUT,<T20,'EL ESTADO FINAL DE LA RANA          ES'   >);
IMPRIME;
CLOSE (ENT[AR]);
END
ENDPROCEDURE EVITA;

PROCEDURE FOTOTAXICA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%      RUTINA QUE SE ENCARGA DE LLAMAR A LAS RUTINAS NECESARIAS PA-   %%
%%  SIMULAR LA RESPUESTA FOTOTAXICA.                                    %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I,
            J;

FOR I := 0 TO 39 DO
    FOR J := 0 TO 119 DO
        ESPACIO [I,J] := '       '
    ENDFOR
ENDFOR;
FOR I := -4 STEP 1 UNTIL 20 DO
    FILL OBSTACULOS[I,*] WITH 11(0)
ENDFOR;
PIDEARCHIVO (AR);
IF FIN THENN
    % FIN DE LA RUTINA
ELSEE
    PIDECOLORES     (OBSTACULOS);
    CLOSE (TERM);
    LIMITAESPACIO   (OBSTACULOS);
    CARACOBSTACULOS (OBSTACULOS, NUMOBST);
    CARACRANA       (RANA, 3);
    PIDE_ORIENTACION (PUNTOOR);
    WRITE (SAL,<///,T40,'ESPERA UN MOMENTO... !'>);
    ESCRIBECARACTERISTICAS (OBSTACULOS, NUMOBST, 2);
    WRITE (OUT [SKIP 1]);
    WRITE (OUT,<///,T20,'EL ESTADO INICIAL PARA SIMULAR LA',/,
                    T20,'    RESPUESTA FOTOTAXICA ES:'>);
    IMPRIME;
    IGUALA (RANAINIC, RANA, 2);
    ACOMODA_RANA (RANA, PUNTOOR);
    ESPACIO[INTEGER(RANA[0]),     INTEGER(RANA[1]) ]      :='     R';
    ESPACIO[INTEGER(RANAINIC[0]), INTEGER(RANAINIC[1])] :='     R';
    WRITE (OUT [SKIP 1]);
    WRITE (OUT,<T20,'EL ESTADO FINAL DE REPOSO DE',/,
                    T20,'          LA RANA ES:'>);
    IMPRIME;
    CLOSE (ENT[AR]);
ENDIFF;
END
ENDPROCEDURE FOTOTAXICA;

PROCEDURE FUNCIONGAUSS (POSPRESA, NHUECOS, VECGAUSS);
         REAL ARRAY VECGAUSS [*];    % VECTOR CON VALORES DE GAUSS
```

```
          INTEGER     NHUECOS       ,    % NUMERO DE HUECOS
                      POSPRESA      ;    % POCISION RELATIVA DE LA PRESA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                  %%
%%        RUTINA QUE CALCULA LOS VALORES DE UNA FUNCION GAUSIANA Y  %%
%%  LOS INTRODUCE A UN VECTOR.                                      %%
%%                                                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER
          I            ;    % VARIABLE PARA EL FOR

FOR I := 0 STEP 1 UNTIL 170 DO
    VECGAUSS[I] := 100*EXP(-ABS(((POSPRESA-I)**2)/GRADO));
ENDFOR;
END
ENDPROCEDURE FUNCIONGAUSS;

PROCEDURE GRAFICA_PRED (DIREC, A, D, R, P);
          ARRAY    R,
                   P [0];

          INTEGER DIREC,
                   A,
                   D;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                  %%
%%           RUTINA QUE SE ENCARGA DE IMPRIMIR AL ARCHIVO DE SALIDA %%
%%  LA POSICION DEL PREDADOR EN EL ESPACIO, EL CUAL ESTA  SEPARADO  %%
%%  DE LA RANA UNA DISTANCIA 'D' Y UN ANGULO DADO POR  'DIREC'  EN  %%
%%  GRADOS.                                                         %%
%%                                                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL PX, PY;

DIREC := A+90;
IF DIREC > 170 THEN
    DIREC := 170;
ENDIF;
BUSCAPUNTO (R[0], R[1], P[0], P[1], D, A, PX, PY);
IF PX > 59 THEN PX := 59;
IF PX < 1  THEN PX := 1;
IF PY > 19 THEN PY := 19;
IF PY < 1  THEN PY := 1;
PREDADOR[0] := PX;
PREDADOR[1] := PY;
ESPACIO [INTEGER (PX), INTEGER (PY)] := '     P';
END
ENDPROCEDURE GRAFICA_PRED;

PROCEDURE HUECOINVALIDO (CEL, HUECO, HUECOS);
          ARRAY     CEL[-1,1],
                    HUECO[*,*];

          INTEGER  HUECOS;
```

```
BEGIN
     INTEGER I,J,K;
     INTEGER DIF;
      INTEGER BORDE;

IF HUECOS GTR 0 THENN
     IF (HUECO[1,1] = 0) THENN
          BORDE := HUECO[1,2];
          DIF := CEL[BORDE,1] - CEL [BORDE+1,1];
          IF ABS(DIF) < 4 THENN
               FOR J:= 1 TO HUECOS-1 DOO
                    FOR K:= 1 TO 5 DOO
                         HUECO[J,K]:=HUECO[J+1,K];
                    ENDFORR;
               ENDFORR;
               HUECOS := * - 1;
          ENDIFF;
     ENDIFF;
ENDIFF;
IF HUECOS GTR 0 THENN
     IF (HUECO[HUECOS,2] = 170) THENN
          BORDE := HUECO[HUECOS,1];
          DIF := CEL[BORDE,1] - CEL[BORDE-1,1];
          IF ABS(DIF) < 4 THENN
               HUECOS := * - 1;
          ENDIFF;
     ENDIFF;
ENDIFF;
END
ENDPROCEDURE HUECOINVALIDO;

PROCEDURE HUIDA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%     RUTINA QUE SE ENCARGA DE DETERMINAR COMO ES EL CASO EN QUE       %%
%%  SE VA A SIMULAR LA HUIDA DE LA RANA DEL PREDADOR.                   %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     INTEGER I,J;

FOR I := 0 TO 39 DO
     FOR J := 0 TO 119 DO
          ESPACIO [I,J] := '      '
     ENDFOR
ENDFOR;
FOR I := -4 STEP 1 UNTIL 20 DO
     FILL OBSTACULOS[I,*] WITH 11(0)
ENDFOR;
PIDEARCHIVO (AR);
IF FIN THENN
     % FIN DE LA RUTINA
ELSEE
     PIDECOLORES      (OBSTACULOS);
     WRITE(TERM,<//,T10,'QUE DIRECCION TIENE EL PREDADOR RESPECTO ',
                 /,T10,'  A LA RANA Y EL PUNTO HACIA EL QUE ESTA ',
                 /,T10,'  VIENDO??   [-90 A 90 GRADOS, DE FRENTE',
```

```
                  /,T10,'                    IGUAL A CERO GRADOS]',
                  /,'#?'>);
    READ (TERM,/,ANG_DIREC);
    WHILE (ANG_DIREC < -90) OR (ANG_DIREC > 90) DOO
        WRITE (TERM,<' EL DATO DEBE ESTAR ENTRE 90 Y -90, TECLEALO',
                  /,' NUEVAMENTE POR FAVOR',/,'#?'>);
        READ (TERM,/,ANG_DIREC);
    ENDWHILEE;
    IF ANG_DIREC=0 THENN
        CASO_PRED := 0;
    ELSEE
        CASO_PRED := 1;
    ENDIFF;
    WRITE (TERM, <,T10,'A QUE DISTANCIA SE ENCUENTRA?? [0-60]'>);
    READ (TERM,/,DIST_DIREC);
    CHECADATOT (DIST_DIREC, 60);
    CLOSE (TERM);
    LIMITAESPACIO     (OBSTACULOS);
    CARACOBSTACULOS (OBSTACULOS, NUMOBST);
    CARACRANA         (RANA, 2);
    PIDE_ORIENTACION (PUNTO_VISION);
    WRITE (SAL,<///,T40,'ESPERA UN MOMENTO...!'>);
    GRAFICA_PRED (DIREC, ANG_DIREC, DIST_DIREC, RANA, PUNTO_VISION);
    ESCRIBECARACTERISTICAS (OBSTACULOS, NUMOBST,1);
    EVITA (RANA, PREDADOR, OBSTACULOS, CASO_PRED);
    CLOSE (ENTCAR]);
ENDIFF;
END
ENDPROCEDURE HUIDA;

BOOLEAN PROCEDURE IGUAL (A, B);
        VALUE A, B;
        REAL  A, B;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;                                                                          %
;         RUTINA CUYA FUNCION ES ESTABLECER SI DOS VALORES SON IGUA-       %
%  LES O CASI IGUALES EN TERMINO DE DIEZMILESIMAS.                         %
%                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
IGUAL := (INTEGER (A*10000)=INTEGER (B*10000));
END
ENDPROCEDURE IGUAL;

PROCEDURE IGUALA (A, B, CUANTOS);
          ARRAY     A,
                    B[1];

          INTEGER CUANTOS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                          %
%         RUTINA QUE HACE LA IGUALACION DE DOS VECTORES 'A' Y 'B' DE       %
%  DIMENSION 'CUANTOS'.                                                    %
%                                                                        . %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
```

```
     INTEGER I;

FOR I := 1 STEP 1 UNTIL CUANTOS DO
     A[I] := B[I];
ENDFOR;
END
ENDPROCEDURE IGUALA;

PROCEDURE IGUALA_ANCHOS (HUECO, HUECOS);
          ARRAY HUECO [*,*];
          INTEGER HUECOS;

BEGIN
     INTEGER I;

FOR I:=1 TO HUECOS DOO
     IF HUECO[I,3] > 15 THENN
          HUECO[I,3] := 15;
     ENDIFF;
ENDFORR;
END
ENDPROCEDURE IGUALA_ANCHOS;

PROCEDURE IMPRIME;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%     RUTINA QUE IMPRIME EL ESPACIO DE LA RANA (OBSTACULOS, PRESA    %%
%%  Y PREDADOR)                                                       %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     INTEGER K,L;
     FORMAT F3(X5,62('-'));

WRITE(OUT,</////>);
WRITE (OUT,F3);
FOR K := 20 STEP -1 UNTIL 0 DO
     WRITE(OUT,<X5,'!',60A1,'!'>,FOR L := 0 STEP 1
                         UNTIL 59 DO ESPACIO[L,K]);
ENDFOR;
WRITE(OUT,F3);
END
ENDPROCEDURE IMPRIME;

PROCEDURE INICIALIZA(NUMOBST,RANA,PRESA,OBSTACULOS);
          ARRAY OBSTACULOS[*,*],
                RANA,
                PRESA[0];

          INTEGER NUMOBST;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%     RUTINA QUE PIDE EL MEDIO AMBIENTE DE LA RANA.                  %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
```

```
     INTEGER I,
             J;

RETIRO := FALSE;
CABE   := FALSE;
FOR I := 0 STEP 1 UNTIL 59 DO
    FOR J := 0 STEP 1 UNTIL 119 DO
        ESPACIO[I,J] := '      '
    ENDFOR
ENDFOR;
FOR I := -4 STEP 1 UNTIL 20 DO
    FOR J := 0 STEP 1 UNTIL 9 DO
        OBSTACULOS[I,J] := 0;
    ENDFOR;
ENDFOR;
PIDEARCHIVO (AR);
IF FIN THENN
    % FIN DE LA RUTINA
ELSEE
    CLOSE              (TERM);
    CARACOBSTACULOS (OBSTACULOS, NUMOBST);
    CARACRANA          (RANA, 1);
    CARACPRESAPRED    (PRESA);
    LIMITAESPACIO     (OBSTACULOS);
    ESCRIBECARACTERISTICAS (OBSTACULOS, NUMOBST, 0);
    CLOSE              (ENT[AR]);
ENDIFF;
END
ENDPROCEDURE INICIALIZA;

PROCEDURE INTERSECCION (X1, Y1, X2, Y2, X3, Y3, X4, Y4, INTX, INTY);
           REAL X1, Y1,
                X2, Y2,
                X3, Y3,
                X4, Y4,
                INTX,
                INTY;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%      LA SUBRUTINA 'INTERSECCION' CALCULA EL PUNTO DE INTERRSEC-     %%
%% CION (INTX,INTY) DE DOS RECTAS, DEFINIDAS ESTAS POR DOS PUNTOS      %%
%% CONTENIDAS EN CADA UNA DE ELLAS .                                   %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL A0, A1,
         B0, B1,
         C0, C1,
         DENOMINADOR;

A0 := Y2-Y1;
B0 := X1-X2;
C0 := (X1*Y2)-(X2*Y1);
A1 := Y4-Y3;
B1 := X3-X4;
C1 := (X3*Y4)-(X4*Y3);
DENOMINADOR := (A0*B1)-(A1*B0);
```

```
IF (DENOMINADOR NEQ 0) THEN
     BEGIN
     INTX := ((B1*C0)-(B0*C1))/DENOMINADOR;
     INTY := ((A0*C1)-(A1*C0))/DENOMINADOR;
     END
ELSE
     IF ((DSTN(X1,Y1,X2,Y2)) < (DSTN(X1,Y1,X4,Y4))) THEN
          BEGIN
          INTX := X3;
          INTY := Y3;
          END
     ELSE
          BEGIN
          INTX := X4;
          INTY := Y4;
          END
     ENDIF
ENDIF;
END
ENDPROCEDURE INTERSECCION;

PROCEDURE LIMITAESPACIO (OBST);
          ARRAY OBST[-4,0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%          RUTINA QUE LIMITARA LOS BORDES DEL ESPACIO CON            %%
%% EL FIN DE QUE LA RANA NO SE SALGA DE SU TRAYECTORIA.               %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BEGIN

OBST[-4,0]:=     OBST[-3,0]:=     OBST[-2,0]:=     OBST[-1,0]:= 1; %TIPO
OBST[-4,9]:=     OBST[-3,9]:=     OBST[-2,9]:=     OBST[-1,9]:=999;%ALTO
OBST[-4,1]:= 0; OBST[-4,2]:=20; OBST[-4,3]:=60; OBST[-4,4]:=20; %SUP
OBST[-3,1]:=60; OBST[-3,2]:= 0; OBST[-3,3]:=60; OBST[-3,4]:=20; %DER
OBST[-2,1]:= 0; OBST[-2,2]:= 0; OBST[-2,3]:=60; OBST[-2,4]:= 0; %INF
OBST[-1,1]:= 0; OBST[-1,2]:= 0; OBST[-1,3]:= 0; OBST[-1,4]:=20; %IZQ
END
ENDPROCEDURE LIMITAESPACIO;

PROCEDURE LLENAESPACIO (CX1, CY1, CX2, CY2, TIPO);
          INTEGER TIPO;

          REAL     CX1, CY1,
                   CX2, CY2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%       RUTINA QUE LLENA LA MATRIZ ESPACIO CON LA POSICION DE LOS    %%
%%  OBSTACULOS.                                                       %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     ALPHA   CH;
     INTEGER X, X1, X2,
             Y, Y1, Y2;
     REAL    H;
CASE TIPO OF
```

```
        BEGIN
        0:      CH := '      Z';
        1:      CH := '      B';
        ELSE:   CH := '      .';
        END
    ENDCASE;
    X1 := INTEGER(CX1);
    X2 := INTEGER(CX2);
    Y1 := INTEGER(CY1);
    Y2 := INTEGER(CY2);
    IF (X2 NEQ X1) THEN
        BEGIN
        M := (Y2-Y1)/(X2-X1);
        FOR X:=X1 STEP (SIGN(X2-X1)) UNTIL X2 DO
            BEGIN
            Y := INTEGER(M*(X-X1) + Y1);
            IF (X<0 OR X>60) OR (Y<0 OR Y>20) THEN
                BEGIN
                CX2 := X-(SIGN(X2-X1));
                CY2 := (M*(X-X1-(SIGN(X2-X1))) + Y1);
                X   := X2;
                END
             ELSE
                ESPACIO[X,Y]:=CH;
             ENDIF
             END
         ENDFOR;
         END
    NOELSE
    ENDIF;
    IF (Y2 NEQ Y1) THEN
        BEGIN
        M := (X2-X1)/(Y2-Y1);
        FOR Y := Y1 STEP (SIGN(Y2-Y1)) UNTIL Y2 DO
            BEGIN
            X := INTEGER(M*(Y-Y1) + X1);
            IF (X<0 OR X>60) OR (Y<0 OR Y>20) THEN
                BEGIN
                CY2 := Y-(SIGN(Y2-Y1));
                CX2 := (M*(Y-Y1-(SIGN(Y2-Y1))) + X1);
                Y   := Y2;
                END
            ELSE
               ESPACIO[X,Y] := CH;
            ENDIF;
            END
        ENDFOR;
        END
    NOELSE
    ENDIF;
    END
    ENDPROCEDURE LLENAESPACIO;

    PROCEDURE MASHUECOS(CEL, HUECO, HUECOS);
              ARRAY   CEL[-1,1],
                      HUECO[*,*];

              INTEGER HUECOS;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%          RUTINA QUE HACE EL ANALISIS NUEVAMENTE DE QUE SI HAY       %%
%%   MAS HUECOS DENTRO DE UN HUECO MAYOR.                              %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I, J;

I := 1;
J := 0;
WHILE I LEQ HUECOS DOO
    WHILE J+1 < HUECO[I,1]  DOO
        IF CEL[J,2] = CEL[J+1,2] THENN
            % MISMA BARDA O NO LA HAY
        ELSEE
            RECORRE (HUECO, HUECOS, I);
            IF CEL[J,1] LEQ CEL[J+1,1] THENN
                OBTENTAMANO(CEL, HUECO,I,1,J+1);
            ELSEE
                OBTENTAMANO(CEL, HUECO,I,-1,J);
            ENDIFF;
        ENDIFF;
        J := J+1;
    ENDWHILEE;
    J := HUECO[I,2]+1;
    I := I+1;
ENDWHILEE;
WHILE J+1 <  170 DOO
    IF CEL[J,2] = CEL[J+1,2] THENN
        % MISMA BARDA O NO LA HAY
    ELSEE
        IF CEL[J,1] LEQ CEL[J+1,1] THENN
            OBTENTAMANO(CEL, HUECO,I,1,J+1);
        ELSEE
            OBTENTAMANO(CEL, HUECO,I,-1,J);
        ENDIFF;
        I       := @+1;
        HUECOS := @+1;
    ENDIFF;
    J := J+1;
ENDWHILEE;
HUECOINVALIDO (CEL, HUECO, HUECOS);
END
ENDPROCEDURE MASHUECOS;

PROCEDURE MENSAJE (OBST, I);
          ARRAY   OBST[*,*];
          INTEGER I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%        RUTINA QUE MANDA AL ARCHIVO DE SALIDA LA SITUACION DE LA     %%
%%   RAMA CON RESPECTO A LOS OBSTACULOS, EN CIERTO MOMENTO  DE  LA     %%
%%   SIMULACION.                                                       %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
BEGIN
WRITE (OUT,<///>);
CASE I OF
    BEGIN
    1: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' METIO LA LENGUA ENTRE LA BARDA Y SE  ',/,
                  ' COMIO A LA PRESA..CROAC..CROAC       ',/,
                  ' ************************************ '>);
       END;
    2: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' SALTA LA BARDA',I3                     ,/,
                  ' ************************************ '>,CUAL);
       END;
    3: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' BAJA LA ZANJA Y PROCEDE A CRUZARLA   ',/,
                  ' ************************************ '>);
       END;
    4: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' HA CRUZADO LA ZANJA                  ',/,
                  ' ************************************ '>);
       END;
    5: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' HA BRINCADO LA ZANJA                 ',/,
                  ' ************************************ '>);
       END;
    6: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ******************************************',/,
                  ' NO PUDO NI BAJAR NI SALTAR LA ZANJA ...',/,
                  ' YA QUE LA CAPACIDAD PARA BAJAR DE LA   ',/,
                  ' RANA ES DE: ',F5.2,/,
                  ' Y LA PROFUNDIDAD DE LA ZANJA ES: ',F5.2,/,
                  ' POR LO TANTO SE RETIRA.........',/,
                  ' ******************************************'>,
                   RANA[2],OBST[CUAL,2]);
       END;
    7: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' SE TOPA CON LA BARDA ',I3              ,/,
                  ' ************************************ '>, CUAL);
       END;
    8: BEGIN
       BUSCAPRESA (RANA, PRESA);
       WRITE (OUT,<' ************************************ ',/,
                  ' SE TOPA CON LA ZANJA ',I3              ,/,
                  ' ************************************ '>,
                    CUAL);
```

```
         END;
      9: BEGIN
         BUSCAPRESA (RANA, PRESA);
         WRITE (OUT,<' ********************************* ',/,
                     ' NO ENCONTRO HUECOS NI PUDO SALTAR ',/,
                     ' ********************************* '>);
         END;
      END
   ENDCASE;
   WRITE (OUT,<///>);
   END
   ENDPROCEDURE MENSAJE;

   PROCEDURE NUEVO(NX, NY, RX1, RY1, RX, RY, RX2, RY2, BAND);

             BOOLEAN  BAND;

             REAL     NX, NY,
                      RX1, RY1,
                      RX, RY,
                      RX2, RY2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                    %%
%%      ASIGNA NUEVAS COORDENADAS A NX Y NY SOBRE LA RECTA DADA POR   %%
%%  RX1, RY1, RX2, RY2,  UN POCO MAS ADELANTE DE RX1, RY1.            %%
%%      SI 'BAND'=TRUE ESPECIFICA QUE SE TRATA DE LA RANA Y QUE       %%
%%  ESTA SE ENCUENTRA EN LA MISMA LINEA QUE LA BARDA.                 %%
%%                                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL Y;

Y := (((RY2-RY1)/(RX2-RX1))*(RX-RX1))+RY1;
IF ((IGUAL(Y,RY)) AND (BAND) AND (IGUAL(RY1,RY2))) THEN
                                       %SOBRE LA MISMA LINEA QUE LA BARDA
    BEGIN
    NX := RX1;
    NY := RY1+1;
    END
ELSE
    BEGIN
    IF RX1<RX2 THEN
        NX := RX1-1
    ELSE
        NX := RX1+1
    ENDIF;
    NY := (((RY2-RY1)/(RX2-RX1))*(NX-RX1))+RY1;
    END
ENDIF;
END
ENDPROCEDURE NUEVO;

PROCEDURE NUEVOPUNTO (NX, NY, RX1, RY1, RX, RY, RX2, RY2, BAND);
          BOOLEAN BAND;

          REAL    NX, NY,
```

```
                    RX1, RY1,
                    RX, RY,
                    RX2, RY2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                           %%
%%        RUTINA QUE SE ENCARGA DE HACER LA LLAMADA CORRECTA AL PRO-         %%
%%  DERURE 'NUEVO', DE TAL FORMA QUE SI SE QUIEREN HACER CALCULOS           %%
%%  CON RECTAS A 90 GRADOS, SE HACE UN CAMBIO DE EJES.                       %%
%%                                                                           %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
IF IGUAL(RX1,RX2) THEN
    NUEVO (NY, NX, RY1, RX1, RY, RX, RY2, RX2, BAND) %RECTA VERTICAL
ELSE
    NUEVO (NX, NY, RX1, RY1, RX, RY, RX2, RY2, BAND);
ENDIF;
END
ENDPROCEDURE NUEVOPUNTO;

PROCEDURE OBJFIJOS (NUMDEOBS, MATDEOBS, R, P, MATREFER);
          ARRAY   R[0],
                  P[0],
                  MATDEOBS[*,*],
                  MATREFER[-86,1];

          INTEGER NUMDEOBS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                           %%
%%      LA SUBRUTINA 'OBJFIJOS' (OBJETOS-FIJOS) SE ENCARGA DE DEPO-          %%
%% SITAR EN LA MATRIZ DE REFERENCIA (MATREFER) LOS VALORES  DE  LAS          %%
%% DISTANCIAS A CADA 'PRIMER OBSTACULO' QUE PERCIBE LA RANA EN UN  -         %%
%% RADIO DE 200 UNIDADES DE -86 A +85 GRADOS, TOMANDO COMO REFEREN-          %%
%% LA TRAYECTORIA QUE UNA LA POSICION DE LA RANA Y DE LA PRESA. TAM-         %%
%% BIEN DEPOSITA EN LA MATRIZ EL NUMERO DEL OBSTACULO DEL QUE SE --          %%
%% TRATA, ASI COMO LAS COORDENADAS (X,Y) DEL PUNTO DE CRUCE.                 %%
%%                                                                           %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN

ARRAY   OBJ,
        INT[0:1],

BOOLEAN HAYOBS;

INTEGER COLUMNA,
        OBSTACT,
        DIVPORCUAD,
        ALCANCE,
        GRADRESOL;

REAL    PI,
        PENDIENTE,
        PENDPRIN,
        DISTRANAPRESA,
        DELTA;
```

```
PROCEDURE CONCIBEESPACIO (OBSTACT, PINT, COLUMNA, MATREFER);

          ARRAY   PINT[0],
                  MATREFER[*,*];

          INTEGER OBSTACT,
                  COLUMNA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                            %%
%%      RUTINA LOCAL A LA  RUTINA OBJFIJOS, Y QUE SE ENCARGA DE HA-           %%
%%  CER UN RECONOCIMIENTO DEL ESPACIO, EN CUANTO A BARDAS, PROFUNDI-          %%
%%  DAD ETC., Y DEPOSITA LA INFORMACION EN 'MATREFER'.                        %%
%%                                                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    MATREFER [COLUMNA,1]  := DSTN(R[0],R[1],PINT[0],PINT[1]);
    MATREFER [COLUMNA, 2] := OBSTACT;
    MATREFER [COLUMNA, 3] := PINT[0];
    MATREFER [COLUMNA, 4] := PINT[1];
    MATREFER [COLUMNA, 5] := OBSTACULOS[OBSTACT,10];
END
ENDPROCEDURE CONCIBEESPACIO;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%           R U T I N A            P R I N C I P A L               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ALCANCE   := 200;
GRADRESOL := 1;
PI        := 3.141592654;
DISTRANAPRESA := DSTN(R[0],R[1],P[0],P[1]);
IF R[0] NEQ P[0] THEN
    PENDPRIN := ARCTAN((R[1]-P[1])/(R[0]-P[0]))
ELSE
    PENDPRIN := PI/2;
ENDIF;
DELTA        := PI * GRADRESOL/180;
DIVPORCUAD  := 85/GRADRESOL;
FOR COLUMNA := -DIVPORCUAD STEP 1 UNTIL DIVPORCUAD DO
    BEGIN
    PENDIENTE := PENDPRIN - COLUMNA *DELTA;
    PUNTOOBJ (R[0], R[1], P[0], P[1], PENDIENTE, ALCANCE,
                                        OBJ[0], OBJ[1]);
    CHECATRAYECTO  (NUMDEOBS, MATDEOBS, R, OBJ, HAYOBS, OBSTACT, INT);
    CONCIBEESPACIO (OBSTACT, INT, COLUMNA, MATREFER);
    END
ENDFOR;
END PROCOBJFIJOS;

PROCEDURE OBTENTAMANO (CEL, HUECO, I, INC, K);
         VALUE   K;

         ARRAY   CEL,
                 HUECO[*,*];

         INTEGER INC,
                 I,
```

```
                    K;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%          RUTINA QUE SE ENCARGA DE OBTENER EL TAMA%O DEL HUECO        %%
%%  QUE ESTA SIENDO PERCIBIDO EN ESE MOMENTO POR LA RANA.               %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     BOOLEAN PROCEDURE IGUAL (A, CEL, B);
             VALUE    A, B;
             ARRAY    CEL[-1,1];

             INTEGER B;
             REAL     A;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %%                                                                %%
     %%         RUTINA LOCAL A LA  RUTINA OBTENTAMA%O, EL CUAL SE      %%
     %%  ENCARGA DE DECIRNOS SI EL HUECO QUE ESTA ANALIZANDO TIENE     %%
     %%  EN SUS BORDES TODAVIA AL MISMO OBSTACULO.                     %%
     %%                                                                %%
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     BEGIN
     IF B = 0 THENN
         IGUAL := FALSE;
     ELSEE
         IF (( CEL[B-1,1] < A ) AND ( CEL[B+1,1] > A )) OR
            (( CEL[B-1,1] > A ) AND ( CEL[B+1,1] < A )) THENN
              IGUAL := TRUE;
         ELSEE
              IGUAL := FALSE;
         ENDIFF;
     ENDIFF;
     END
     ENDPROCEDURE BOOLEAN IGUAL;

     INTEGER J,
             K1,
             LARGO;

LARGO := 1;
K1    := K-INC;
WHILE ( NOT IGUAL(CEL[K1,1],CEL,K) )  AND
      ( CEL[K,2] = CEL[K+INC,2] )     DOO
    LARGO := *+1;
    K     := *+INC;
ENDWHILEE;
IF INC > 0 THENN
    HUECO[I,1] := K1+INC;
    HUECO[I,2] := K;
    HUECO[I,3] := LARGO;
    HUECO[I,4] := ABS(CEL[K1,1]-CEL[K1+INC,1]);
    HUECO[I,5] := ABS(CEL[K,1] -CEL[K+INC,1 ]);
ELSEE
    HUECO[I,2] := K1+INC;
    HUECO[I,1] := K;
    HUECO[I,3] := LARGO;
```

```
    HUECO[I,5] := ABS(CEL[K1,1]-CEL[K1+INC,1]);
    HUECO[I,4] := ABS(CEL[K,1] -CEL[K+INC,1 ]);
ENDIFF;
END
ENDPROCEDURE OBTENTAMANO;

PROCEDURE PIDEARCHIVO (CUAL);
          INTEGER CUAL;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%%        RUTINA QUE PIDE DISPOSITIVO EN EL CUAL VIENEN LOS DATOS, Y            %%
%%  EN CASO DE SER DISCO PIDE EL NOMBRE DEL ARCHIVO EN EL CUAL VIE-             %%
%%  NEN LOS DATOS PARA HACER LA SIMULACION.                                     %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
WRITE(TERM, BORRASCREEN);
WRITE(TERM,<8(/),T20,'COMO SE VAN A DAR LOS DATOS:',////,
               T20,'POR TERMINAL      [0]',/,
               T20,'POR DISCO         [1]',/,'#?'>);
READ (TERM,/,CUAL);
CHECADATOT (CUAL, 1);
IF CUAL=1 THENN
    WRITE(TERM, BORRASCREEN);
    WRITE(TERM,<8(/),T20,' NOMBRE ARCHIVO DE DATOS  ?',/,'#?'>);
    READ (TERM,10,NOMBARCH);
    PA := P(NOMBARCH);
    S PA:PA UNTIL EQL ' ';
    R PA BY '.';
    R ENT[CUAL].TITLE BY P(NOMBARCH);
    IF ENT[CUAL].RESIDENT  THENN
       %SIGUE
    ELSEE
        WRITE(TERM, BORRASCREEN);
        WRITE(TERM,<8(/),T20,'REVISA EL DIRECTORIO DE LA CLAVE PUES',/,
                   ' NO EXISTE EL ARCHIVO ',A36>,P(NOMBARCH));
        FIN := TRUE;
    ENDIFF;
ENDIFF;
IF NOT FIN THENN
    WRITE(TERM, BORRASCREEN);
    WRITE(TERM,<8(/),T20,'A DONDE DESEAS LA SALIDA DE RESULTADOS:',
             ////,T20,'A TERMINAL         [0]',/,
                  T20,'A IMPRESORA        [1]',/,
                  T20,'A DISCO            [2]   (CON EL NOMBRE ',
                      'RANA/RES)',/,'#?'>);
    READ (TERM,/,SA);
    CHECADATOT (SA, 2);
    IF SA=2 THENN
        IF SALIDA[SA].RESIDENT THENN
            SALIDA[SA].NEWFILE := TRUE;
        NOELSE
        ENDIFF;
    NOELSE
    ENDIFF;
ENDIFF;
END
```

```
ENDPROCEDURE PIDEARCHIVO;

PROCEDURE PIDECOLORES (OBST);
          ARRAY OBST[*,*];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%          RUTINA QUE LIMITARA LOS BORDES DEL ESPACIO CON             %%
%%  LOS COLORES NECESARIOS PARA QUE LA RANA VEA EL FONDO DE            %%
%%  COLORES, Y ASI EVALUAR EN FORMA CORRECTA LOS HUECOS.               %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY   C[-4:-1];
    INTEGER I;
WRITE(TERM,BORRASCREEN);
WRITE(TERM,<8(/),T20,' EXISTEN LOS SIGUIENTES CODIGOS DE COLORES:',/,
          T10,'AZUL     [1]',/,
          T10,'VERDE    [2]',/,
          T10,'AMARILLO[3]',/,
          T10,'ROJO     [4]',/,
          T10,'NEGRO    [CUALQUIER OTRO NUMERO]'>);
WRITE(TERM,<//,T20,'DAME SEPARADOS POR COMAS LOS COLORES QUE QUIERES ',
        /,' QUE TENGAN LOS BORDES NORTE, ESTE, SUR Y OESTE',/,
          '#?'>);
READ (TERM,/,C[-4], C[-3], C[-2], C[-1]);
FOR I:=-4 STEP 1 UNTIL -1 DOO
    IF C[I]<1 OR C[I]>4 THENN
       OBST[I,10] := 5;
    ELSEE
       OBST[I,10] := C[I];
    ENDIFF;
ENDFORR;
END
ENDPROCEDURE PIDECOLORES;

PROCEDURE PIDECOORDS (I, MATOBST);
          ARRAY   MATOBST [*,*];
          INTEGER I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%       RUTINA QUE PIDE LAS COORDENADAS DEL OBSTACULO A COLOCAR.      %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT FO (/,' DAME LA COORDENADA X',I1,/,,'#?'),
           F1 (' DAME LA COORDENADA Y',I1,/,'#?');
    INTEGER J,K;

K := 1;
FOR J := 1 STEP 2 UNTIL (7-3*MATOBST[I,0]) DO
    BEGIN
    WRITE (SAL,FO,K);
    READ  (ENT[AR],/,MATOBST[I,J]);
    WRITE (SAL,/,MATOBST[I,J]);
    CHECADATO (MATOBST[I,J],60);
    WRITE (SAL,F1,K);
```

```
     READ  (ENT[AR],/,MATOBST[I,J+1]);
     WRITE (SAL,/,MATOBST[I,J+1]);
     CHECADATO (MATOBST[I,J+1],20);
     K := * + 1;
     END
ENDFOR;
K := I;
IF (I NEQ 0) THEN
     VERIFINTER (I, MATOBST);
NOELSE
ENDIF;
IF K EQL I THEN
   PIDEPROFYALT (I, MATOBST);
NOELSE
ENDIF;
CONTROLAESPACIO (I, MATOBST);
END
ENDPROCEDURE PIDECOORDS;

PROCEDURE PIDECOORDS (I, MATOBST);
          ARRAY   MATOBST [*,*];
          INTEGER I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%        RUTINA QUE PIDE LAS COORDENADAS DEL OBSTACULO A COLOCAR.      %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT FO (/,' DAME LA COORDENADA X',I1,/,,'#?'),
           F1 (' DAME LA COORDENADA Y',I1,/,'#?');
    INTEGER J,K;

K := 1;
FOR J := 1 STEP 2 UNTIL (7-3*MATOBST[I,0]) DO
    BEGIN
    WRITE (SAL,FO,K);
    READ  (ENT[AR],/,MATOBST[I,J]);
    WRITE (SAL,/,MATOBST[I,J]);
    CHECADATO (MATOBST[I,J],40);
    WRITE (SAL,F1,K);
    READ  (ENT[AR],/,MATOBST[I,J+1]);
    WRITE (SAL,/,MATOBST[I,J+1]);
    CHECADATO (MATOBST[I,J+1],20);
    K := * + 1;
    END
ENDFOR;
K := I;
IF (I NEQ 0) THEN
    VERIFINTER (I, MATOBST);
NOELSE
ENDIF;
IF K EQL I THEN
   PIDEPROFYALT (I, MATOBST);
NOELSE
ENDIF;
CONTROLAESPACIO (I, MATOBST);
END
```

```
ENDPROCEDURE PIDECOORDS;

PROCEDURE PIDEPROFYALT (I,MATOBST);
          ARRAY   MATOBST [*,*];
          INTEGER I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%     RUTINA QUE PIDE LA PROFUNDIDAD O LA ALTURA DEL OBSTACULO.        %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    FORMAT FO (' PROFUNDIDAD DE LA ZANJA ',/,'#?') ,
           F1 (' ALTURA DE LA BARDA ',/,'#?') ;

IF (MATOBST[I,0] EQL 0.) THEN
    WRITE (SAL,FO)
ELSE
    WRITE (SAL,F1);
ENDIF;
READ  (ENT[AR],/,MATOBST[I,9]);
WRITE (SAL,/,MATOBST[I,9]);
CHECADATO (MATOBST[I,9],100);
END
ENDPROCEDURE PIDEPORFYALT;

PROCEDURE PIDE_ORIENTACION (PO);
          ARRAY PO[0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%     RUTINA QUE PIDE COMO DATO EL PUNTO HACIA EL CUAL LA RANA ES-     %%
%%  TA VIENDO, ESTO CON EL FIN DE SIMULAR LA RESPUESTA FOTOTAXICA.      %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
WRITE (SAL,<//,' HACIA QUE PUNTO ESTA VIENDO LA RANA??',/,
           'DAME LA COORDENADA X',/,'#?'>);
READ  (ENT[AR],/,PO[0]);
WRITE (SAL,/,PO[0]);
CHECADATO (PO[0],60);
WRITE (SAL,<'DAME LA COORDENADA Y',/,'#?'>);
READ  (ENT[AR],/,PO[1]);
WRITE (SAL,/,PO[1]);
CHECADATO (PO[1],20);
ESPACIO[ INTEGER (PO[0]), INTEGER (PO[1]) ]:='      .';
END
ENDPROCEDURE PIDE_ORIENTACION;

PROCEDURE PROCESABARDA(RANA,PUNTO,OBST,CUAL,SIALCANZA,SISALTA);
          ARRAY   RANA,
                  PUNTO[0],
                  OBST[*,*];

          INTEGER CUAL;

          BOOLEAN SIALCANZA,
                  SISALTA;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%      RUTINA QUE DETERMINA QUE ES LO QUE SE VA A HACER CON LA BAR-   %%
%%   DA EN CUESTION; SI LA PUEDE BRINCAR, SI LA LENGUA DE LA RANA      %%
%%   ALCANZA A LA RANA ENTRE LAS REJAS DE LA BARDA, O SI BORDEA A      %%
%%   ESTA.                                                             %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY PUNTO1 [0:1];
    REAL  DIST1,
          DIST2;

IF ( (METODO=1) AND (NOT(SIALCANZA OR SISALTA)) ) THEN
    PROCESAHUECOS (NUMOBST, OBST, RANA, PRESA, MATCEL, HUECO,
                   HUECOS,  VECANCHOS,  VECPROF, VECGAUSS, INDICE, 1);
NOELSE
ENDIF;
IF SIALCANZA THEN
    BEGIN
    CABE := TRUE;
    IGUALA (RANANT, RANA, 2);
    NUEVOPUNTO (RANA[0], RANA[1], PUNTO[0], PUNTO[1],
                RANA[0], RANA[1], PRESA[0], PRESA[1], TRUE);
    MENSAJE(OBST,1);        %METIO LA LENGUA ENTRE LA BARDA Y COMIO
    LLENAESPACIO (RANA[0], RANA[1], RANANT[0], RANANT[1], 2);
    END
ELSE
    BEGIN
    IF SISALTA THEN
        BEGIN
        MENSAJE (OBST,2);      %PUDO SALTAR ESA BARDA
        IGUALA  (PUNTO1, RANA, 2);
        IGUALA  (RANANT, RANA, 2);
        NUEVOPUNTO (RANA[0],RANA[1], PUNTO[0], PUNTO[1],
                    RANA[0],RANA[1], PUNTO1[0],PUNTO1[1],FALSE);
        LLENAESPACIO(RANA[0], RANA[1], RANANT[0], RANANT[1], 2);
        END
    ELSE
        BEGIN
        IGUALA (RANANT, RANA, 2);
        IF METODO=0 THENN
            DIST1 := SQRT(((PUNTO[0]-OBST[CUAL,1])**2)+
                          ((PUNTO[1]-OBST[CUAL,2])**2));
            DIST2 := SQRT(((PUNTO[0]-OBST[CUAL,3])**2)+
                          ((PUNTO[1]-OBST[CUAL,4])**2));
            IGUALA(RANANT, RANA, 2);
            IF DIST1 <= DIST2 THEN
                NUEVOPUNTO (RANA[0], RANA[1], OBST[CUAL,1],
                            OBST[CUAL,2], RANA[0], RANA[1],
                            OBST[CUAL,3], OBST[CUAL,4], TRUE)
                               % ASIGNA NUEVAS COORD A RANA[0], RANA[1]
                               % UN POCO MAS ADELANTE DE LA RECTA  DADA
                               % POR   OBST[CUAL,1] ,    OBST[CUAL,2] Y
                               % OBST[CUAL,3], OBST[CUAL,4] ;     HACIA
                               % OBST[CUAL,1], OBST[CUAL,2]
            ELSE
```

```
                    NUEVOPUNTO (RANA[0], RANA[1], OBST[CUAL,3],
                                OBST[CUAL,4], RANA[0], RANA[1],
                                OBST[CUAL,1], OBST[CUAL,2], TRUE)
                ENDIF;
            ELSEE
                IF HUECOS > 0 THEN
                    CAMINARANA( INDICE, MATCEL, RANANT, RANA, 1)
                ELSE
                    BEGIN
                    MENSAJE (OBST, 9);
                    RETIRO := TRUE;
                    END
                ENDIF;
            ENDIFF;
            LLENAESPACIO(RANANT[0], RANANT[1], RANA[0], RANA[1], 2);
            END
        ENDIF;
        END
ENDIF;
END
ENDPROCEDURE PROCESABARDA;

PROCEDURE PROCESAHUECOS (NUMOBST, OBST, RANA, PRESA, MATCEL,
                         HUECO, HUECOS, VECANCHOS, VECPROF,
                         VECGAUSS, INDICE, PP);
          ARRAY   RANA,
                  PRESA,
                  VECANCHOS,
                  VECPROF,
                  VECGAUSS [0];

          ARRAY   OBST,
                  MATCEL,
                  HUECO [*,*];

          INTEGER NUMOBST,
                  HUECOS,
                  INDICE,
                  PP;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                              %%
%%          LA RUTINA 'PROCESAHUECOS' SE ENCARGA DE COORDINAR   %%
%%    LAS LLAMADAS  A  LAS  RUTINAS NECESARIAS PARA  DETERMI-   %%
%%    NAR LOS HUECOS QUE VISUALIZA LA RANA EN CIERTA POSICION   %%
%%    DADA.                                                     %%
%%       TAMBIEN, A PARTIR DEL VALOR DE LA VARIABLE 'PP'(PRE-   %%
%%    SA-PREDADOR) DETERMINA SI SE VAN A EVALUAR TAMBIEN  LOS   %%
%%    COLORES DEL FONDO; ESTO ES, SOLO SE TOMARAN  EN  CUENTA   %%
%%    LOS COLORES DEL FONDO, PARA CUANDO LA RANA SE ENCUENTRE   %%
%%    ASEDIADA POR EL PREDADOR (PP=2), PARA LA RESPUESTA        %%
%%    FOTOTAXICA (PP=3) O HUIDA DEL PREDADOR (PP=3).            %%
%%                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY   CV[0:0];
    INTEGER I;
```

```
FOR I := -1 TO 171 DO
    FILL MATCEL[I,*] WITH 5(0);
ENDFOR;
OBJFIJOS (NUMOBST, OBST, RANA, PRESA, MATCEL);
WRITE    (OUT [SKIP 1]);
WRITE    (OUT,<10(/),T40,'EL EDO DE LAS CELULAS DEL OBSTACULO:'>);
REPORTA  (MATCEL, 2);
CAPTAHUECOS      (MATCEL, HUECO, HUECOS);
MASHUECOS        (MATCEL, HUECO, HUECOS);
ELIMINAPEQUES    (HUECO, HUECOS);
IGUALA_ANCHOS    (HUECO, HUECOS);
REPORTAHUECOS    (HUECO, HUECOS);
COLOCAANCHO      (HUECO, HUECOS, VECANCHOS);
WRITE        (OUT [SKIP 1]);
WRITE        (OUT,<10(/),T40,'EL VECTOR DE ANCHOS QUEDA:'>);
REPORTA_VEC (VECANCHOS);
PROMPROFUNDIDAD (HUECO, HUECOS, VECPROF);
WRITE        (OUT [SKIP 1]);
WRITE        (OUT,<10(/),T40,'EL VECTOR DE PROFUNDIDAD QUEDA:'>);
REPORTA_VEC (VECPROF);
FUNCIONGAUSS     (DIREC,   HUECOS, VECGAUSS);
WRITE        (OUT [SKIP 1]);
WRITE        (OUT,<10(/),T40,'EL VECTOR DE GAUSS QUEDA:'>);
REPORTA_VEC (VECGAUSS);
CASE PP OF
    BEGIN
    1:                      %PARA PRESA
       FILL VECCOLOR[*] WITH 170(0);
    2:                      %PARA PREDADOR
       EVALUACOLOR (VECCOLOR, MATCEL);
       DRP := 12-DSTN (RANA[0],RANA[1],PREDADOR[0],PREDADOR[1]);
       IF DRP<0 THEN   DRP := 0;
       FOR I := 0 TO 170 DOO
          VECGAUSS[I] := (VECGAUSS[I]*(-1))
       ENDFORR;
    3:                      %PARA RESPUESTA FOTOTAXICA
       FILL VECGAUSS[*] WITH 170(0);
       EVALUACOLOR (VECCOLOR, MATCEL);
    4:                      %HUIDA DEL PREDADOR
       EVALUACOLOR (VECCOLOR, MATCEL);
    END
ENDCASE;
%DETERMINAHUECOINT (PP); %PARA LLAMRSE SI SE QUIEREN ENCONTRAR LAS
                         %CONSTANTES C1, C2, C3 EN FORMA INTERACTIVA
DETERMINAHUECO (HUECO, HUECOS, VECANCHOS,
                VECPROF, VECGAUSS, INDICE, VECCOLOR,
                                    C1, C2, C3, PP);
IF PP=3 THENN                   %EN RESPUESTA FOTOTAXICA
    ESCOGE_IND (INDICE, HUECO, HUECOS, MATCEL);
NOELSE
ENDIFF;
WRITE (OUT,<////,T50,' EL INDICE OPTIMO ES ',I3>,INDICE);
END
ENDPROCEDURE PROCESAHUECOS;

PROCEDURE PROCESAZANJA (RANA, PUNTO, SIBAJA, SIBRINCA, DIST);
          ARRAY RANA,
                PUNTO[0];
```

```
          BOOLEAN SIBAJA,
                  SIBRINCA;

          REAL    DIST;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                       %%
%%      RUTINA QUE DETERMINA QUE ES LO QUE SE VA A HACER CON LA ZANJA %%
%%  EN CUESTION; SI LA PUEDE SALTAR, SI LA BAJA POR AHI PARA LLEGAR   %%
%%  AL OTRO EXTREMO, O SI SE RETIRA.                                  %%
%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
IF SIBAJA THEN
    BEGIN
    LLENAESPACIO (RANA[0], RANA[1], DENTRO[0], DENTRO[1], 2);
    IGUALA       (RANA, DENTRO, 2);
    MENSAJE      (OBSTACULOS, 3);        % BAJA LA ZANJA Y
                                         % PROCEDE A CRUZARLA
    LLENAESPACIO (RANA[0], RANA[1], FUERA[0], FUERA[1], 2);
    IGUALA       (RANA, FUERA, 2);
    MENSAJE      (OBSTACULOS,4);    %LA CRUZO, SUBE Y ESTA ARRIBA,
    END
ELSE
    BEGIN
    IF SIBRINCA THEN
        BEGIN
        IGUALA  (RANA, FUERA, 2);
        MENSAJE (OBSTACULOS,5); % HA BRINCADO LA ZANJA 'CUAL' Y HA
                                % LLEGADO HASTA RANA[0], RANA[1]
        END
    ELSE
        BEGIN
        MENSAJE (OBSTACULOS,6);   % NO PUDO NI BAJAR LA ZANJA PUES
                                  % SU PROFUNDIDAD ERA OBST[CUAL,9]
                                  % NI SALTARLA PUES SU DIST DE CRU-
                                  % CE ERA DIST,Y ELLA TENIA RANA[4]
        RETIRO := TRUE;
        END
    ENDIF;
    END
ENDIF;
END
ENDPROCEDURE PROCESAZANJA;

PROCEDURE PROMPROFUNDIDAD (MATHUECOS, NHUECOS, VECPROF);
          REAL ARRAY MATHUECOS [*,*],  % MATRIZ DE HUECOS
                     VECPROF   [*]  ;  % VECTOR DE PROFUNDIDADES

          INTEGER    NHUECOS        ;  % NUMERO DE HUECOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                       %%
%%       RUTINA QUE EXTRAE EL PROMEDIO DE LAS DIFERENCIAS DE PRO-     %%
%%  FUNDIDAD EN UN HUECO Y LAS DEPOSITA EN UN VECTOR PARALELO.        %%
%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
```

```
    INTEGER I,J             ;  % VARIABLES PARA FOR'S

FOR I := 1 STEP 1 UNTIL NHUECOS DO
    FOR J := MATHUECOS[I,1] STEP 1 UNTIL MATHUECOS[I,2] DO
        VECPROF[J] := (MATHUECOS[I,4] + MATHUECOS[I,5]) DIV 2 ;
    ENDFOR;
ENDFOR;
END
ENDPROCEDURE PROMPROFUNDIDAD;

BOOLEAN PROCEDURE PUEDEBRINCAR(OBST,RANA1,PRESA,PUNTO,DIST);
        ARRAY   OBST[*,*],
                RANA1,
                PRESA,
                PUNTO[0];

        REAL    DIST;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                            %%
%%          RUTINA QUE CHECA SI LA RANA ES CAPAZ DE SALTAR LA ZANJA.          %%
%%  ES DECIR SI SU CAPACIDA DE SALTO ES MAYOR O IGUAL A LA DISTAN-            %%
%%  CIA EXISTENTE ENTRE LOS BORDES DE LA ZANJA QUE SE ENCONTRO.               %%
%%                                                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY RANA [0:6];
    BOOLEAN CRUZO;

IGUALA (RANA, RANA1,6);
NUEVOPUNTO (DENTRO[0], DENTRO[1], PUNTO[0], PUNTO[1],
              RANA[0], RANA[1], RANA[0], RANA[1], FALSE);
                   %ASIGNA NUEVAS COORD A DENTRO UN POCO MAS
                   %ADELANTE DE LA RECTA DADA POR RANA Y PUNTO
                   %HACIA PUNTO
CHECATRAYECTO (NUMOBST,OBST,DENTRO,PRESA,CRUZO,CUAL,PUNTO);
NUEVOPUNTO     (FUERA[0], FUERA[1], PUNTO[0], PUNTO[1],
                DENTRO[0], DENTRO[1], DENTRO[0], DENTRO[1], FALSE);
                   %ASIGNA NUEVAS COORD A FUERA UN POCO MAS
                   %ADELANTE DE LA RECTA DADA POR DENTRO Y PUNTO
                   %HACIA PUNTO
DIST := SQRT(((FUERA[0]-RANA[0])**2)+
             ((FUERA[1]-RANA[1])**2));
IGUALA (RANA1, FUERA, 2);
IF DIST <= RANA[3] THEN
    PUEDEBRINCAR := TRUE
ELSE
    PUEDEBRINCAR := FALSE
ENDIF;
END
ENDPROCEDURE PUEDEBRINCAR;

PROCEDURE PUNTOOBJ (RX, RY, PX, PY, PENDIENTE, ALCANCE, X, Y);
          REAL PX, PY,
               RX, RY,
               ALCANCE,
               PENDIENTE,
```

```
              X, Y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                     %%
%%        LA RUTINA 'PUNTOOBJ' CALCULA EL PUNTO LOCALIZADO A 'ALCANCE' %%
%% UNIDADES DEL PUNTO (RX, RY) TOMANDO COMO DIRECCION LA INCLINACION %%
%% INDICADA POR 'PENDIENTE' (EN RADIANES) ESCOGIENDO EL MAS CERCANO   %%
%% A EL PUNTO (PX, PY).                                                %%
%%                                                                     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    REAL  A, B, C,
          M, ORD,
          MEDIOPI,
          X1, X2, Y1, Y2,
          DIST1, DIST2;

MEDIOPI := 3.141592654/2;
IF NOT IGUAL( ABS(PENDIENTE) ,MEDIOPI) THEN
    BEGIN
    M   := TAN (PENDIENTE);
    ORD := RY - M*RX;
    A   := 1 + M*M;
    B   := 2*M*ORD - 2*RX - 2*RY*M;
    C   := RX*RX + RY*RY - ALCANCE**2 - 2*RY*ORD + ORD*ORD;
    X1  := (-B+(B*B - 4*A*C)**0.5)/(2*A);
    X2  := (-B-(B*B - 4*A*C)**0.5)/(2*A);
    Y1  := X1*M + ORD;
    Y2  := X2*M + ORD;
    END
 ELSE
    BEGIN
    X1 := RX;
    X2 := RX;
    Y1 := RY + ALCANCE;
    Y2 := RY - ALCANCE;
    END
 ENDIF;
 DIST1 := DSTN (X1,Y1,PX,PY);
 DIST2 := DSTN (X2,Y2,PX,PY);
 IF DIST1 < DIST2 THEN
      BEGIN
      X := X1;
      Y := Y1;
      END
 ELSE
      BEGIN
      X := X2;
      Y := Y2;
      END
ENDIF;
END
ENDPROCEDURE PUNTOOBJ;

PROCEDURE RECORRE (A, LIM, REN);
          ARRAY   A[*,*];
          INTEGER LIM,
                  REN;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%       RUTINA QUE COLOCA EN UNA NUEVA POSICION AL NUEVO HUECO CAP-    %%
%%  TADO, Y A LOS DEMAS LOS RECORRE UN LUGAR ABAJO DE ELLOS.            %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     ARRAY   B[1:LIM+1,1:5];
     INTEGER I, J;

FOR I := 1 STEP 1 UNTIL LIM+1 DO
     FOR J := 1 STEP 1 UNTIL 5 DO
          B[I,J] := A[I,J];
     ENDFOR;
ENDFOR;
FOR I := REN STEP 1 UNTIL LIM DO
     FOR J := 1 STEP 1 UNTIL 5 DO
          A[I+1,J] := B[I,J];
     ENDFOR;
ENDFOR;
LIM := *+1;
END
ENDPROCEDURE RECORRE;

PROCEDURE REPORTA (A, K).

          ARRAY    A[*,*];
          INTEGER K;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
%          RUTINA QUE IMPRIME EL VALOR DE LAS NEURONAS DE DISTANCIA,     %
%  COORDENADA X, COORDENADA Y O TIPO DE OBSTACULO.                       %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
     INTEGER I,J;

WRITE (OUT,<///,T54,F7.2>,A[0,K]);
FOR I := 0 STEP 1 UNTIL 16 DO
     WRITE (OUT,<T10,I3,' ----> ',10F7.2,'  ----> ',I3,/>, (I*10+1),
                 FOR J:=1 STEP 1 UNTIL 10 DO A[I*10+J, K], (I+1)*10);
ENDFOR;
END
ENDPROCEDURE REPORTA;

PROCEDURE REPORTAHUECOS (HUECO, HUECOS);

          ARRAY    HUECO [*,*];
          INTEGER HUECOS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
%        RUTINA QUE TIENA COMO FIN EL MANDAR A IMPRIMIR EL NUMERO DE     %
%  HUECOS QUE PERCIBE LA RANA EN CIERTO MOMENTO, ASI COMO EL  TAMA&O     %
%  DE ESTOS EN ANCHO Y PROFUNDIDAD.                                      %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
BEGIN
    INTEGER I;

WRITE (OUT [SKIP 1]);
WRITE (OUT,<5(/),T20,' EL NUMERO DE HUECOS ENCONTRADOS FUE ',I3>,
                                                          HUECOS);
WRITE (OUT,<///,'    NUM HUECO       INICIO          FIN     ANCHO ',
                                        '      DIFIZQ       DIFDER'>);
FOR I := 1 STEP 1 UNTIL HUECOS DO
    WRITE (OUT,<T7,I2,T20,I3,T33,I3,T40,I3,T49,I3,T62,I3>,I,
                               HUECO[I,1],HUECO[I,2],HUECO[I,3],
                                          HUECO[I,4],HUECO[I,5]);
ENDFOR;
END
ENDPROCEDURE REPORTAHUECOS;

PROCEDURE REPORTA_VEC (A);
          ARRAY   A[*];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
%          RUTINA QUE SE ENCARGA DE EMITIR LOS VALORES QUE TIENEN        %
%  LOS VECTORES DE ANCHO, PROFUNDIDAD Y GAUSIANO.                        %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    INTEGER I,J;

WRITE (OUT,<///,T54,F7.2>,A[0]);
FOR I := 0 STEP 1 UNTIL 16 DO
    WRITE (OUT,<T10,I3,' ----> ',10F7.2,'  ----> ',I3,/>, (I*10+1),
               FOR J:=1 STEP 1 UNTIL 10 DO A[I*10+J], (I+1)*10);
ENDFOR;
END
ENDPROCEDURE REPORTA_VEC;

PROCEDURE VERIFINTER (I, MATOBST);
          ARRAY   MATOBST [*,*];
          INTEGER I;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%        RUTINA QUE VERIFICA QUE LOS OBSTACULOS NO SE SOBREPONGAN.     %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BEGIN
    ARRAY   COORD1,
            COORD2,
            PTOCOMUN [0:1];
    BOOLEAN INTERSEC;
    INTEGER OBSTINTER;

INTERSEC  := FALSE;
COORD1[0] := MATOBST[I,1];
COORD1[1] := MATOBST[I,2];
COORD2[0] := MATOBST[I,3];
COORD2[1] := MATOBST[I,4];
```

```
CHECATRAYECTO (I-1, MATOBST, COORD1, COORD2, INTERSEC,
                              OBSTINTER, PTOCOMUN);
IF INTERSEC THEN
    ERROROBST (OBSTINTER, PTOCOMUN, I)
ELSE
    BEGIN
    IF (MATOBST[I,0] EQL 0.0) THEN
        BEGIN
        COORD2[0] := MATOBST[I,7];
        COORD2[1] := MATOBST[I,8];
        CHECATRAYECTO (I-1, MATOBST, COORD1, COORD2,
                      INTERSEC, OBSTINTER, PTOCOMUN);
        IF INTERSEC THEN
            ERROROBST (OBSTINTER,PTOCOMUN,I)
        ELSE
            BEGIN
            COORD1[0] := MATOBST[I,5];
            COORD1[1] := MATOBST[I,6];
            CHECATRAYECTO (I-1, MATOBST, COORD1, COORD2,
                          INTERSEC, OBSTINTER, PTOCOMUN);
            IF INTERSEC THEN
                ERROROBST (OBSTINTER,PTOCOMUN,I)
            ELSE
                BEGIN
                COORD2[0] := MATOBST[I,3];
                COORD2[1] := MATOBST[I,4];
                CHECATRAYECTO (I-1, MATOBST, COORD1, COORD2,
                              INTERSEC, OBSTINTER, PTOCOMUN);
                IF INTERSEC THEN
                    ERROROBST (OBSTINTER,PTOCOMUN,I);
                ENDIF;
                END
            ENDIF;
            END
        ENDIF;
        END
    NOELSE
    ENDIF;
    END
ENDIF;
END
ENDPROCEDURE VERIFINTER;

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%          P R O G R A M A   P R I N C I P A L                         %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

WRITE(TERM,BORRAPANTALLA);
WRITE(TERM,<T30,'E M P E Z A M O S',/////>);
FIN := FALSE;
WHILE NOT FIN DO
    BEGIN
    WRITE (TERM,<///,T62,'<RETURN >  '>);
    READ  (TERM);
    WRITE (TERM,BORRAPANTALLA);
```

```
    WRITE (TERM,OPCIONES);
    READ  (TERM,/,TIPOSIM);
    CHECADATOT (TIPOSIM, 3);
    C1:=C2:=C3:=0;
    WHILE (C1+C2+C3) NEQ 100  AND  TIPOSIM NEQ 3  DOO
        WRITE (TERM,<///, T20,' DAME LOS VALORES DE LAS CONSTANTES ',/,
                    T20,' EN EL ORDEN : ANCHO,PROFUNDIDAD Y GAUSS'>,);
        READ(TERM,/,C1,C2,C3);
        IF (C1+C2+C3) NEQ 100 THENN
            WRITE (SAL,/' LA DATOS ANTERIORES SON INCORRECTOS',/,
                       ' DEBERAN SUMAR 100 '>,);
            WRITE (SAL,' Y SUMAN --',F10.1>,C1+C2+C3);
        ELSEE
            WRITE (TERM,<///,T20,'CUAL SERA LA CONSTANTE PARA LA ',/,
                     T20,'CURVA EXPONENCIAL DE GAUSS??',/,' #?'>);
            READ  (TERM,/,GRADO);
        ENDIFF;
    ENDWHILEE;
    CASE TIPOSIM OF
        BEGIN
        0: ATACO := TRUE;
           ATAQUE;
        1: ATACO := FALSE;
           HUIDA;
        2: FOTOTAXICA;
        3: WRITE (TERM,<////,T10,'FIN NORMAL DEL TRABAJO !!! '>);
           FIN := TRUE;
        END
    ENDCASE;
    END
ENDWHILE;
END.
```

APENDICE D

En este apéndice se muestra el programa en Basic que se encarga de realizar la graficación en CROMEMCO, y se explica cada una de sus partes así como el objetivo de cada una de las rutinas específicas de SDI-GRAPHICS usadas en el programa.

```
      % Habilita graficación.
1090   Q=Usr(316,68)

1095   Dim Radio(80),Pend(80),Trax(25),Tray(25)
1100   Input"Nombre del archivo donde se encuentran los datos? ",Na$
1110   Open\1\Na$

      % A continuación se hace la lectura de
      % los puntos que componen la figura.
1170   Input\1\Nptos
1200     For I=1 To Nptos
1220     Input\1\X,Y
1240     If X=0 Then X=1E-07
1270     Pend(I)=Atn(Y/X)
1280     If X<0 Then Pend(I)=Pend(I)+3.141592654
1300     Radio(I)=Sqr(X*X+Y*Y)
1310     Next I

      % Lectura de las coordenadas y radio de los ojos.
1320   Input\1\H(1),K(1),R(1)
1330   Input\1\H(2),K(2),R(2)
1340   Close\1\

      % Inicializa Ngra (Número de gráficas desplegadas).
1350   Ngra=0

      % Pide nombre del archivo donde se encuentran
      % los datos del espacio.
1351   Input"Cual es el nombre del espacio a seguir?...", Esp$
1360   Open\2\Esp$

      % Lee número de obstáculos y sus características.
1370   Input\2\Nobst
1379   I=0
1380     For Cont=1 To Nobst
1381     I=I+1
1382     Input\2\Tipobs(I),Clr(I)
1383     If Tipobs(I)=0 Then Do

          % Lee coordenadas de los 4 puntos
          % que forman la xanja.
1384       Input\2\Bx1(I),By1(I),Bx2(I),By2(I)
1385       I=I+1
1386       Input\2\Bx1(I),By1(I),Bx2(I),By2(I)
1387       Else
1388       If Tipobs(I)=1 Then Do

            % Lee coordenadas de los extremos
            % de la barrera.
1390          Input\2\Bx1(I),By1(I),Bx2(I),By2(I)
1391          Else
```

```
            % Lee coordenadas del centro y radio
            % de circulo sombreado.
            % (Utilizado para un caso particular
            % de respuesta fototáxica).
1392         Input\2\Xc(I),Yc(I),Rc(I)
1393         Enddo
1394       Enddo
1400     Next Cont

       % Lee coordenadas donde se encuentra
       % presa o predador.
1410    Input\2\Xpr,Ypr

       % Número de veces que se orienta el anfibio
       % antes de lograr su objetivo.
1420    Input\2\Norient

1430     For J=1 To Norient

         % Coordenadas de orientación y de inicio
         % y fin de tramo a caminar.
1440      Input\2\X1,Y1,X2,Y2

         % Para evitar división entre cero al
         % calcular pendiente.
1450      If X1=X2 Then X2=X1+1E-07

         % Cálculo de la pendiente en Radianes
         % (con ajuste).
1460      Pdte=Atn((Y2-Y1)/(X2-X1))-(3.1416/2)
1470      If(X2-X1)<0 Then Pdte=Pdte+3.1416
1480      If((X2-X1)<>0) And (Abs(Y2-Y1)<Abs(X2-X1)) Then Do

           % Selecciona a (Y0) como variable dependiente.
1510        M=(Y2-Y1)/(X2-X1)
1520        Signx=Abs(X2-X1)/(X2-X1)

           % Calcula abscisa final del tramo
           % recto a caminar.
1530        Xf=X2-15*Signx

             % Procede a calcular los puntos por donde
             % pasará el anfibio.
1570          Y0=M*(X0-X1)+Y1

             % Llama a la rutina que grafica la rana
             % en el punto (X0,Y0).
1590          Gosub 2020
1600          Next X0

           % Si al finalizar X0 esta muy alejado de la
           % abscisa final haz (X0,Y0) = (Xf,Yf)
           % y llama a la rutina de graficación.
```

```
1601        If(Xf+20)>X0 Then X0=Xf : Y0=M*(X0-X1)+Y1 : Gosub 2020

1620        Else

           % Selecciona a (X0) como variable dependiente.
1630        M=(X2-X1)/(Y2-Y1)
1650        Signy=Abs(Y2-Y1)/(Y2-Y1)
1660        Yf=Y2-Signy*15

             % Procede a calcular los puntos por donde
             % pasará el anfibio.
1670          For Y0=Y1 To Yf Step Signy*23
1680          X0=M*(Y0-Y1)+X1

             % Llama a la rutina que grafica a la rana
             % en el punto (X0,Y0).
1700          Gosub 2020
1710          Next Y0

           % Si al finalizar Y0 esta muy alejado de la
           % ordenada final haz (X0,Y0) = (Xf,Yf)
           % y llama a la rutina de graficación.
1715        If(Yf+20)>Y0 Then Y0=Yf : X0=M*(Y0-Y1)+X1 : Gosub 2020
1720        Enddo
1740      Next J
1745    Close\2\

       % Calcula la distancia de la rana a la presa.
1750    Dist=Sqr((Xinit-Xpr)**2+(Yinit-Ypr)**

       % Si la distancia rana-presa es pequeña
       % procede a hacer parpadear la pantalla.
1770    If Dist<11 Then Do

         % Dibuja una línea que parta de el hocico
         % de la rana a la presa.
1780      Q=Usr(316,0,Xinit,Yinit,Xpr,Ypr,9)
1790        For T=1 To 10

             % Apaga la pantalla de graficación.
1800          For T1=1 To 100 : Next T1 : Q=Usr(316,64)

             % Prende la pantalla de graficación.
1810          For T2=1 To 100 : Next T2 : Q=Usr(316,63)
1820        Next T
1830      Else
1840      Print : Print 'ATAQUE FRUSTRADO' : Print : Print
1850      Enddo
1860    Input 'Desea hacer otra simulación ??',R$
        If R$ = 'S' Then Goto 1350
1870    End
```

```
      % Rutina para graficar la rana (o cualquier otra
      % figura) tomando como centro las coordenadas
      % (X0,Y0), con una inclinación (Pdte) en radianes.
      % Realiza cambio de página de trabajo para
      % graficación.
2080   If P=0 Then Do
2090     P=1
2100     Else
2110     P=0
2120     Enddo

      % Asigna a (P) como página de trabajo.
2130   Q=Usr(316,69,P) : Q=Usr(316,71)
2131   I=0

        % Grafica obstáculos.
2132     For Cont=1 To Nobs
2134     I=I+1
2136     If Tipobs(I)=0 Then Do

          % Une los puntos que describen la
          % zanja con color (Clr(I)).
2138       Q = Usr(316, 0, Bx1(I), By1(I), Bx2(I), By2(I), Clr(I))
2140       Q = Usr(316, 0, Bx2(I), By2(I), Bx1(I+1), By1(I+1), Clr(I))
2142       Q = Usr(316, 0, Bx1(I+1), By1(I+1), Bx2(I+1), By2(I+1),
                           Clr(I))
2144       Q = Usr(316, 0, Bx2(I+1), By2(I+1), Bx1(I), By1(I), Clr(I))
2146       I=I+1
2148       Else
2149       If Tipobs(I)=1 Then Do

            % Une los puntos que describen
            % la barrera con color (Clr(I)).
2150          Q=Usr(316,0,Bx1(I),By1(I),Bx2(I),By2(I),Clr(I))
2152          Else

            % Grafica un circulo con centro (Xc(I),Yc(I)),
            % Radio (Rc(I)) y color (Clr(i)).
2154          Q=Usr(316,8,Xc(I),Yc(I),Rc(I),Clr(I))
2156          Enddo
2158       Enddo
2159     Next Cont
2160   Xinit=Cos(Pdte+Pend(1))*Radio(1)+X0
2161   Xx=Xinit
2162   Yinit=Sin(Pdte+Pend(1))*Radio(1)+Y0
2163   Yy=Yinit

        % Procede a graficar la rana uniendo
        % los puntos que la forman
2165     For I=2 To Nptos

        % Calcula nuevas coordenadas tomando
```

```
        % como referencia (X0,Y0).
2170     X=Cos(Pdte+Pend(I))*Radio(I)+X0
2180     Y=Sin(Pdte+Pend(I))*Radio(I)+Y0
2190     Q=usr(316,0,Xx,Yy,X,Y,4)
2210     Xx=X
2220     Yy=y
2230     Next I
2240   Q=Usr(316,0,Xx,Yy,Xinit,Yinit,4)

        % Grafica los ojos de la rana.
2250     For I=1 To 2
2270     If H(I)=0 Then H(I)=1E-07
2280     Pend=Atn(K(I)/H(I))
2290     If H(I)<0 Then Pend=Pend+3.1416
2300     Radio=Sqr(H(I)*H(I)+K(I)*K(I))
2310     H=Cos(Pdte+Pend)*Radio + X0
2320     K=Sin(Pdte+Pend)*Radio + Y0
2330     R=R(I)
2350     Q=Usr(316,8,H,K,R,10)
2360     Next I

      % Marca con un punto la posición de la Presa.
2370   Q=Usr(316,8,Xpr,Ypr,1,9)

      % Incrementa en 1 el número de gráficas desplegadas.
2371   Ngra=Ngra+1

      % Guarda en los vectores Trax y Tray la posición donde
      % se ha posicionado la rana para marcar ruta.
2372   Trax(Ngra)=X0
2373   Tray(Ngra)=Y0

        % Marca los puntos por donde ha pasado la rana.
2374     For K=1 To (Ngra-1)
2375     Q=Usr(316,8,Trax(K),Tray(K),1,4)
2376     Next K

      % Despliega en la graficadora la página de trabajo.
2380   Q=Usr(316,70,P)
2390   Return
```