ASSIGNMENT V (continued)

THE TRAVELING WORM

You will be writing a number of *MATHEMATICA* procedures whose end product will be a movie of a “WORM” traveling in a user chosen area (*house*) avoiding a number of user chosen obstacles (*obs*) (see the attached sample picture).

For efficient production of the successive frames of the movie, it is suggested that the construction of the frames be divided into three steps.

1. In the first step the successive worm positions are constructed simply as numerical vectors giving the coordinates of the “centers” of the sections of the worm (see worm picture below).
2. In the second step, the numerical data is converted in successive worm pictures each preceded by a copy of the background.
3. In the third step the successive frames are made into a movie by, means of the *MATHEMATICA* “ListAnimate” command.

These three steps may be implemented within a procedure, (which calls all the procedures described below), with heading

\[ \text{mkshow}[N,\ n,\ \text{house},\ \text{obs},\ \alpha,\ \beta,\ \gamma,\ \delta] \]

where \( N \) gives the desired number of frames, \( n \) gives the length of the worm (the number of its sections), *house* and *obs* encode the “background” which consists of the region of the plane where the worm is constrained to travel, and the “obstacles” it is to avoid (see figures above to get an idea of what the background can be), and finally \( \alpha,\ \beta,\ \gamma,\ \delta \) give “probabilities” governing the motion of the worm as described below.

To speed up computation a worm will consist of \( n \) sections each of which is represented by a pair of integers \((a, b)\) giving the “center” of the section. In *MATHEMATICA*, this is simply a sequence

\[ \text{worm} = \{ \{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_n, b_n\} \} \]

where \( \{a_n, b_n\} \) gives the position of the “head” of the worm \( \{a_{n-1}, b_{n-1}\} \) the “neckbone” and \( \{a_1, b_1\} \) the “tail”.

A step in the motion consists of three parts:

1. Computing the vector \( \mathbf{v} \) that gives the present “direction” of the worm. That is defined as

\[ \mathbf{v} = \{a_n - a_{n-1}, b_n - b_{n-1}\} \]

which may be computed by the command \( \mathbf{v} = \text{worm}[n] - \text{worm}[n-1] \)

2. Determining the point \( \text{newhead} = \{a, b\} \) giving the new position of the head by successively calling the procedure with heading “\( \text{turn} [\mathbf{v},\ \alpha,\ \beta,\ \gamma,\ \delta] \)”, we shall describe below, until the resulting point \( \{a, b\} \) is “inside” *house* and “outside” *obs*. 
(3) Removing the “tail” and appending the new “head” to construct the **newworm**. This can be achieved by the _Mathematica_ command:

```
newworm = Append[Delete[worm, 1], newhead]
```

These three steps can be carried out by a call of a procedure with heading

```
mkhead[n_, house_, obs_, α_, β_, γ_, δ_]
```

where `n` gives the number of sections of the worm (i.e. its “length”), `house` should give the region within which the worm is restricted to travel, `obs` should be the sequence of obstacles the worm should avoid, and finally `α, β, γ, δ` determine the probabilities which the procedure `turn` should be using in the construction of the new head.

Before we can describe the procedure `turn` we need to make some observations. First of all, since each section of the worm, including the “head” is represented by a pair of integers. The new “head” `{a,b}` must be obtained by adding to `{an,bn}` one of the vectors on the left side of the following display:

Except that, since we do not want the new head `{a,b}` to fall on top of the worm’s neckbone (that is `{an−1,bn−1}`), we need to exclude that one of these vectors that is equal to −v = {an−1 − an, bn−1 − bn).

For instance the figure on the right in the display above depicts the vectors we are allowed to add to `{an,bn}` in the case that v turns out to be `{1,0}`. We have also indicated how the probabilities α, β, γ, δ are to be distributed among these vectors. Since the programming of this selection is the most difficult part of this project, to help you out, I have included a collection of commands and procedures that you may use for this purpose. Study them and understand what they can do for you.

```
Rin[n_] := Random[Integer, {0, n}]

Ra[tot_] := Random[Real, 0, tot]

dins = {{0, 1}, {−1, 1}, {−1, 0}, {−1, −1}, {0, −1}, {1, −1}, {1, 0}, {1, 1}}

Do[where[dins[[i]]] = i, {i, 1, 8}]

modmod[k_, n_] := If[Mod[k, n] == 0, n, Mod[k, n]]
```
mksel[a_, b_, c_, d_] := (w = Ra[a + b + c + d];
  out = Which[w <= a, 0,
    (a < w) && (w <= a + b),
    2*Rin[1] - 1, 
    (a + b < w) && (w <= a + b + c),
    4*Rin[1] - 2, 
    a + b + c < w, 
    6*Rin[1] - 3];
  Return[out])

shify[v_, x_] := (pl = where[v];
  npl = modmod[pl + x, 8];
  Return[dins[[npl]]])

turn[v_, a_, b_, c_, d_] := (dis = mksel[a, b, c, d];
  Return[shify[v, dis]])

A word of caution must be added here, to make sure that no portion of the worm enters the obstacles the worm is supposed to avoid, the actual data using to check the “goodness” of the “head” should represent a slightly larger obstacle than the one you will end up depicting in the background of your frames. The same suggestion applies to the “house”. That is in the depicted house should be slightly larger than the one used to determine “inside-ness” of the new head.

When the worm is dangerously close to an obstacle, the program we have described will require a number of calls of the procedure turn before a good new head is finally obtained. An interesting variation of the display we had in class is to join as many frames of the “stand still” worm as the number of trials required to obtain the new head. This will indicate “hesitation” on the part of the worm in the presence of obstacles.

We should also mention that experimentation with our program revealed that a most realistic behaviour of the worm is obtained by choosing $\alpha = 20$, $\beta = 3$, $\gamma = 2$ and $\delta = 1$. This will assure that the worm “prefers” to go straight and turn only when forced to.

In the figure above we have depicted an 8-section “worm” constructed from a sequence of 8 adjacent grid points \{a1, b1\}, \ldots, \{a8, b8\}. Of course, you may dress up your sequence \{a1, b1\}, \ldots, \{a8, b8\} with a worm “dress” of your choice. What is important is that you do this only after all the successive worm positions have been precalculated. To do this you may use three global variable worm, newworm and worms. Then at the start worm is initialized to lie in a region, within the house, where you know there are no obstacles. This done worms is initialized to \{worm\} and everytime newworm constructed you simply append it to worms by a call of worms = Join[worms, newworm] followed by worm = newworm.

At the end of this process, determined by the number of frames you want in your movie you will have your sequence of worm positions and all you have to is dress them up for the show! In the two figures above we show our worm avoiding three obstacle yet remaining within the “house”.