Step 1 – download and open the netlogo #3 file from canvas

Please download model file from Practical Resources 3 on canvas as this will be our starting point.

Step 2 – creating a food function for the butterflies

We will now create a function to enable our butterflies to smell food within a particular range and return a value to a local variable within the make_butterfly_move function in order to decide what the butterfly should do. Add the following code to your program.

```
to-report food_function [sensitivity]                          ; this creates a reporting function called food_function and expects a value for ser
  set food_around_me other ( food in-radius sensitivity )      ; this sets the food_around_me variable to the ID's of the food within the sensitivi
  set closest_food min-one-of food_around_me [distance myself] ; this sets the closest_food variable to the ID of the closest food source
  let can_smell_food [false]                                   ; this creates a local variable called can_smell_food and sets it to false
  let eating_food [false]                                      ; this creates a local variable called eating_food and sets it to false

  if health < 100 [                                            ; if health is less than 100 then...
    ask food in-radius rad [                                   ; this sets up a radius around the food to the value of the global variable rad whic
      ifelse amount > 0 [                                      ; if amount (a food variable) is greater than 0...
        set eating_food true                                   ; set the local variable called eating_food to true indicating the butterfly is eati
        set amount amount - 5                                  ; reduces 5 from the amount variable in the food
        set color color - .25                                  ; reduce the color intensity of the food by .25
      ][                                                       ; otherwise...
        die                                                    ; there is no food left so kill the good agent
      ]
    ]
  ]

  if eating_food = true [                                      ; if eating_food is true then...
    set health health + 5                                      ; add 5 to health of butterfly
    adjust_vision_cone                                         ; call adjust_vision_cone function as health impact on the vison of the butterfly
  ]

  if (closest_food != nobody) [                                ; if closest_food is not empty (the butterfly can smell food in range) then...
    set can_smell_food true                                    ; set can_smell_food to true
  ]
  report can_smell_food                                        ; return value of can_smell_food to location where function was called
end
```

Add the following line to the code in the make_butterfly_move code. The 30 signifies the sense of smell.

```
    let can_smell_food food_function 30                ; this creates a local variable called can_smell_food then fills it with the return c
```

Step 3 – switching your butterflies behaviours

To switch the behaviour of your butterflies between avoiding people, looking for food and random wandering the code should look as follows:

```
to make_butterflies_move                                       ; this is defining a function called make_butterflies_move
  ask butterflies [                                            ; this asks all of the butterflies in the population to do what is in the brackets
    ifelse health > 0 [                                        ; if health is greater than 0 then (still alive)...
      show_visualisations                                      ; call the show_visualisations function
      set color blue                                           ; this sets the color of each butterfly to blue
      let have_seen_person people_function                     ; this creates a local variable called have_seen_person the fills it with the return
      let can_smell_food food_function 30                      ; this creates a local variable called can_smell_food then fills it with the return o
      ifelse ( have_seen_person = true ) [                     ; if local variable have_seen_person is true...
        right 180                                              ; set heading of the butterfly to 180 (turn around to avoid!)
        ][                                                     ; otherwise...
        ifelse ( can_smell_food = true ) and ( health < 100) [ ; if local variable can_smell_food is true...
          set heading ( towards closest_food )                 ; set heading towards closest food source
          ][                                                   ; otherwise...
          right (random bwr - (bwr / 2))                       ; this turns the butterfly right relative to its current heading by a random degree n
      ]]
      forward butterflies_speed + ( speed_variation * 0.1 )    ; moves butterfly forward by the butterflies_speed variable
    ][                                                         ; otherwise...
      set color gray                                           ; set color to gray to indicate dead butterfly
      die                                                      ; this kills off the butterfly
    ]
  ]
end
```

In addition to this you will need to comment out 2 lines in the people_function as outlined below.

```
    ;right 180     ;------------------------------; set heading of the butterfly to 180 (turn around to avoid!)
  ][                                               ; if seen = false...
    ;right (random bwr - (bwr / 2)) ;--------------; this turns the butterfly right relative to its current heading by a random degree number using
  ]
```

* Now test your model and experiment with the various parameters

Step 4 – creating venom pods

Similar to previous steps to create weapons we will breed a new population of agents called vpods as follows.

```
breed [ venom vpod ]                    ; creating a population of venom pods as weapons to defend the butterflies
```

The we will create a function to setup the vpods using the following code.

```
to make_venom                           ; this creates a function called make_venom
  setxy random-xcor random-ycor         ; this sets the position of the venom to a random location in the world
  set color green                       ; this sets the color of the venom to green
  set size 5                            ; this sets the size of the venom to 5
  set shape "x"                         ; this sets the shape of the venom to an x
end
```

We then add some lines to the setup function to add the vpods to the world and set them up by calling the make_venom function.

```
  create-venom 20 [                     ; this creates x number of new venom pods for the butterflies to store and use
    make_venom                          ; this calls the make_venom function
  ]
```

* Now test your model and see if the vpods appear

Step 5 – picking up venom pods

Step 5.1 – adding another variable to the butterflies

In order for the butterflies to keep a log on the vpods they are carrying you need to create an additional variable on the butterfly at the beginning of the code. The butterflies-own variables should now look like this.

```
butterflies-own [ people_seen people_hit    ; this creates 2 variables which will be used to count the total people seen and total people hit by
health robustness speed_variation           ; this creates 3 variables for health, durability and speed
per_vis_rad per_vis_ang                     ; this creates variables for personalised vision cones
food_around_me closest_food                 ; this creates 2 variables to save the locations of food
have_venom                                  ; this creates a variable to store the amount of venom held
]
```

Step 5.2 – creating the pickup function

To enable the butterflies to pickup the vpods we will write a new function similar to what we created for the food function but simpler as we are not actively looking. The code is as follows.

```
to pickup_venom                             ; this creates a function called pickup_venom
  let pickup [false]                        ; this creates a local variable called pickup and sets it to false
  ask venom in-radius rad [                 ; this sets up a radius around the butterfly to the value of the global variable rad which we are
    set pickup true                         ; this sets the local variable pickup to true
    die                                     ; this removes the vpod from the map
  ]
  if pickup = true [                        ; if pickup is true then...
    set have_venom have_venom + 1           ; add 1 to the have_venom count on the butterfly
  ]
end
```

Once you have done this you need to call the function in the make_butterflies_move function as follows. Place this in a logical location within the function.

```
    pickup_venom ;++++++++++++++++++++++++++++++++++++++++; this calls the pickup_venom function
```

Step 5.3 – making the vpods have impact

To make the vpods have impact you need to add some lines of code to the people_function. Essentially you need to add a local variable to store the ID of the person hit, log the ID within the in-radius part of the code and add an if statement to use the vpod, kill the person and update the have_venom counter. The people function should now look as follows:

```
to-report people_function                       ; this creates a reporting function called people_function
  let seen [false]                              ; this creates a local variable called seen
  let hit [false]                               ; this creates a local variable called hit
  let person_hit 0    ;++++++++++++++++++++++++; this creates a local variable calles person_hit and sets it to 0

  ask peoples in-cone per_vis_rad per_vis_ang [   ; this sets up a vison cone on the butterfly with the parameters from per_vis_rad per_vis_ang and
    set color green                             ; this sets the color of the person detected within the vision code of the butterfly to green
    set seen true                               ; this sets the local variable called seen to true indicating that a person has been seen
  ]

  ask peoples in-radius rad [                   ; this sets up a radius around the butterfly to the value of the global variable rad which we are
    set hit true                                ; this sets the local variable called hit to true indicating that a person has collided with the b
    set person_hit who ;++++++++++++++++++++++++; this sets the local variable called person_hit to the individual who
    show person_hit
  ]

  ifelse seen = true [                          ; if then else statement based on the local variable seen, if seen = true then...
    set people_seen people_seen + 1             ; add 1 to the people_seen count
    set color white                             ; set color of butterfly to white
    ;right 180     ;-----------------------------; set heading of the butterfly to 180 (turn around to avoid!)
  ][                                            ; if seen = false...
    ;right (random bwr - (bwr / 2)) ;-----------; this turns the butterfly right relative to its current heading by a random degree number using
  ]

  if hit = true [                               ; if statement based on the local variable hit, if seen = true then...
    ifelse have_venom > 0 [ ;++++++++++++++++++++; if have_venom is greater that 0 then...
      ask people person_hit [die] ;+++++++++++++; kill off the person hit
      set have_venom have_venom - 1 ;++++++++++++; remove 1 from the have_venom of the butterfly
    ][
      set people_hit people_hit + 1             ; add 1 to the people_hit count
      set color green                           ; set color of butterfly to green
      set health health - robustness ;++++++++++; adjust health of butterfly to health - collision penalty (robustness)
      adjust_vision_cone ;++++++++++++++++++++++; calls adjust_vision_cone to update the vision parameters based on heath changes
    ]
  ]
  report seen ;++++++++++++++++++++++++++++++++; return true or false based in local variable seen
end
```

Step 6 – using graphs to keep tabs on our model

As an easy visual indicator you can add a plot to your interface, this can be done by entering the details as per the screenshot below.