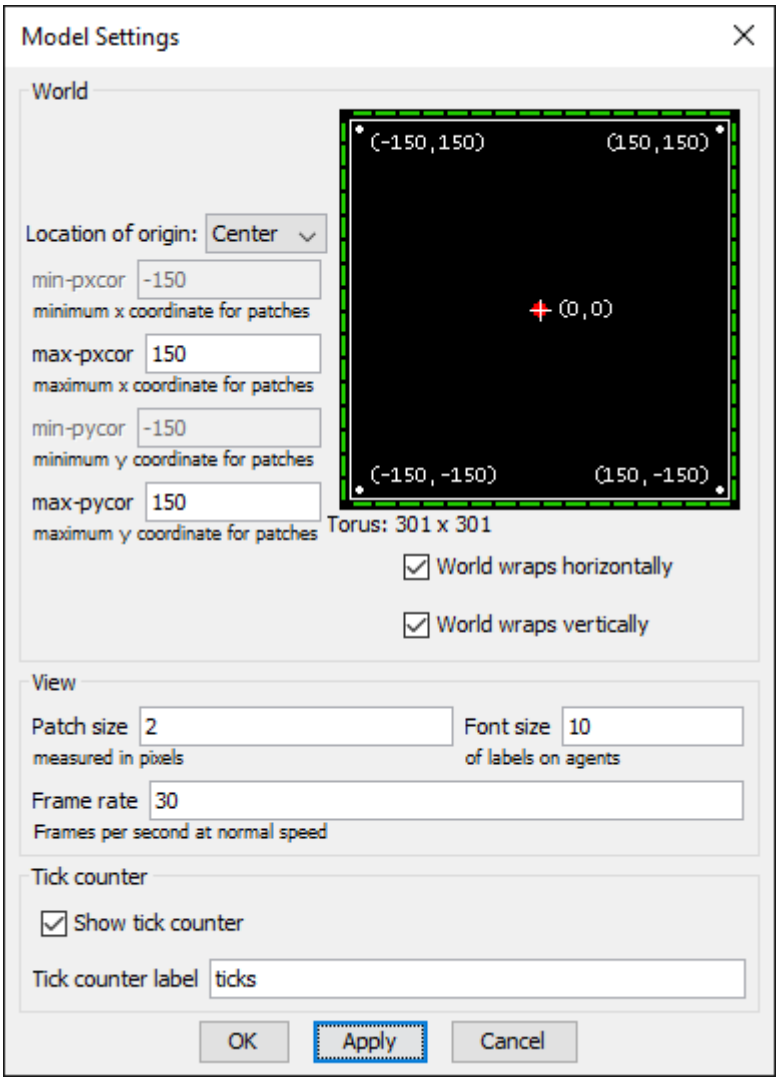
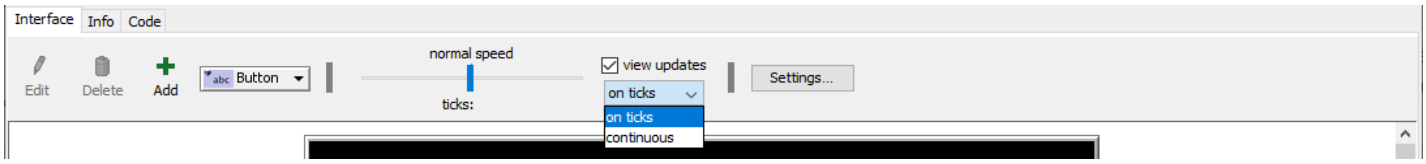
Step 1 - Setting up the environment

- Open NetLogo and save the file in an appropriate location (student drive)
- Click on the setting button and modify the environment to:
    a. extend 150 patches from the centre both horizontally and vertically
    b. wraps both vertically and horizontally by ensuring boxes are ticked (creating infinate world)
    c. set the patch size to 2
    d. set framerate to 30
    e. show the tick counter



Once this is done on the interface set the make sure view updates is ticked and ensure the world updates on ticks rather than continuously as shown below.



Step 2 – Creating the inhabitants

In this world there will be 2 types of inhabitant's people and butterflies. To create these types of populations, enter the following code in the code tab:

```
breed [ peoples people ]
breed [ butterflies butterfly ]
```

Step 3 – creating agent variables and global variables using code

In this model we want the butterflies to keep a log of the number of people they have avoided and the number of people they have hit. In addition we will create a global variable called rad to demonstrate how these work. To create these variables use the following code:

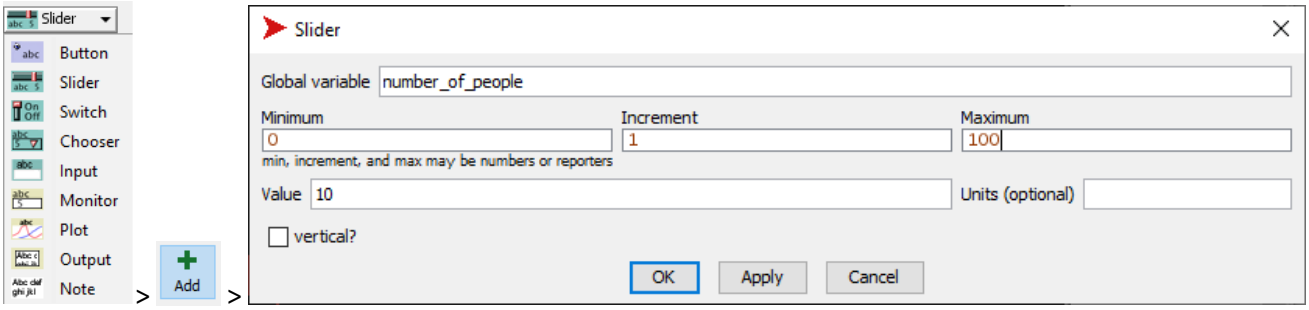
```
butterflies-own [ people_seen people_hit ]

globals [rad]
```

Step 4 – creating an interface and global variables

In this model we want to be able to easily change the parameters of the model, we will therefore create some global variables and buttons to activate commands easily in the interface tab. Create the

Step 4.1 – Creating a global variable with a slider

Select slider from the option list then click the add button and the slider dialogue box will appear where you can name the variable and enter the constraints and default value of the variable.

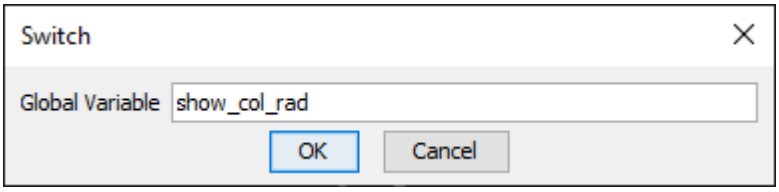


Create the following sliders with the stated parameters:

Name	Min	Incr	Max	Val
number_of_people	1	1	100	40
people_speed	1	1	10	1
pwr	10	1	180	10
number_of_butterflies	1	1	20	5
butterflies_speed	1	1	10	1
bwr	10	1	180	10
vis_rad	0	1	50	25
vis_ang	0	1	180	45

Step 4.2 – Creating a global variable with a switch

Select switch from the option list then click the add button and the switch dialogue box will appear where you can name the variable.

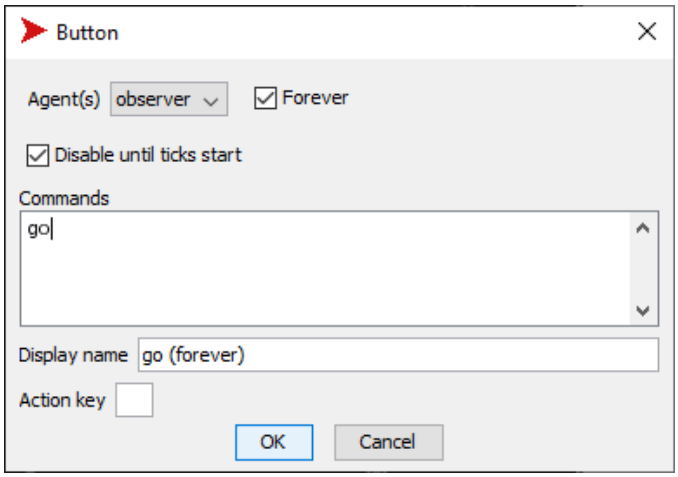


Create the following switches with the stated parameters:

- show_col_rad
- show_vis_cone

Step 4.3 – Creating command buttons

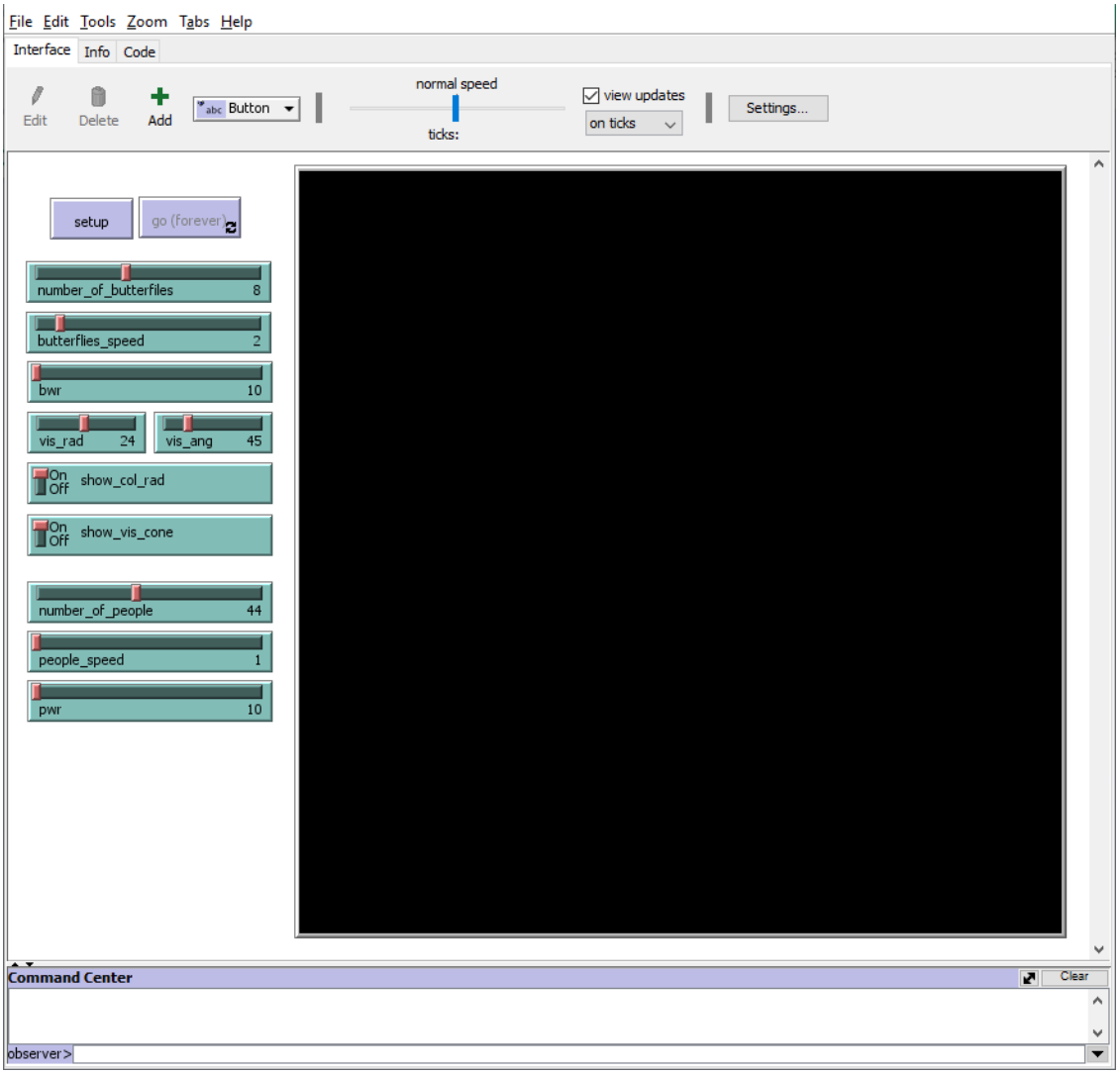
Select button from the option list then click the add button and the button dialogue box will appear where you can name the variable and sent the command for it to activate.



Create the following buttons with the stated parameters:

Commands	Agent(s)	Forever	Disable until ticks start
go	Observer	Ticked	Ticked
setup	Observer	Unticked	

Once these steps have been completed arrange the interface as follows:

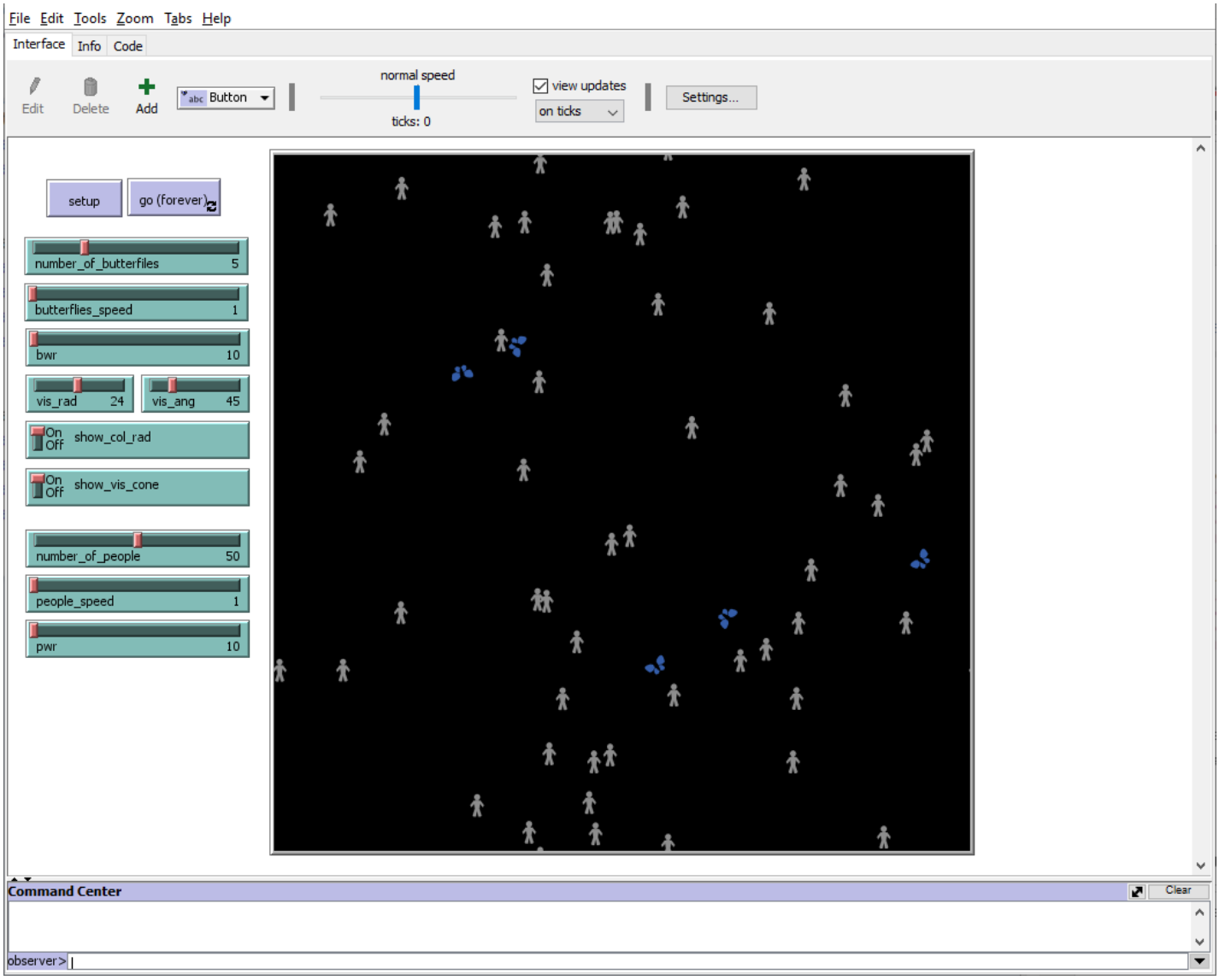


Step 5 – creating the setup function

The following code is used to setup the world which will reset it, place the inhabitants within the world and configure them.

```
to setup                                  ; this creates a function called setup
  clear-all                               ; this clears the world of any previous activities
  reset-ticks                             ; this resets the ticks counter
  set rad 5                               ; this sets the global variable rad to 3

  create-peoples number_of_people [       ; this creates the number of people that your global variable states
    setxy random-xcor random-ycor         ; this sets the starting position of the people to a random location in the world
    set color gray                        ; this sets the color of the people to gray
    set size 10                           ; this sets the size of the people to 10
    set shape "person"                    ; this sets the shape of the people to a person
  ]

  create-butterflies number_of_butterfiles [; this creates the number of butterflies that your global variable states
    setxy random-xcor random-ycor         ; this sets the starting position of the butterflies to a random location in the world
    set color blue                        ; this sets the color of the butterflies to blue
    set size 10                           ; this sets the size of the butterflies to 10
    set shape "butterfly"                 ; this sets the shape of the butterflies to a butterfly
  ]
end
```

If you have entered this correctly when you go back to the interface and click the setup button something similar the this should appear:



Step 6 – creating the go function

In this example we will use several different functions in order to demonstrate the principle. Type the following code but comment out reset_patch_color and make_butterflies_move as we will test the system incrementally.

```
to go                                   ; this creates a function called go
  make_people_move                      ; this calls the make_people_move function
  reset_patch_color                     ; this calls the reset_patch_color function
  make_butterflies_move                 ; this calls the make_butterflies_move function
  tick                                  ; this adds 1 to the tick counter
end
```

Step 7 – making the people move

Creating this function will make the people agents within your environment move. The comments within the code explain exactly what each line does.

```
to make_people_move                     ; this creates a function called make_people_move
  ask peoples [                         ; this asks all of the people in the population to do what is in the brackets
    set color gray                      ; this sets the color of each person to gray
    right ( random pwr - ( pwr / 2))    ; this turns the person right relative to its current heading by a random degree number
    forward people_speed                ; this sets the speed at which the people move
  ]
end
```

Step 8 – test your model and experiment with the parameters

Test your model after entering the code and you will see that the agents will move around using the parameters set within the sliders. To add more agents to the model you will need to press the setup button again after moving the slider.

Step 9 – setting up the butterflies

Step 9.1 – making the butterflies move and setting up local agent variables

The following code will setup some local variables which we will use later and make the butterflies move forward in a single direction. NOTE: uncomment the make_butterflies_move function in the go function.

```
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

    set color blue                      ; this sets the color of each butterfly to blue
    let seen [false]                    ; this creates a local variable called seen
    let hit [false]                     ; this creates a local variable called hit

    forward butterflies_speed           ; moves butterfly forward by the butterflies_speed variable
  ]
end
```

Test the code in the interface and you will see the butterflies just go straight. The next step is to make the butterfly see.

Step 9.2 – setup make the butterfly see by setting up a vision cone

The top part of the code sets up a field of view in front of each butterfly. The lower section visualises the cone by changing the color of the patches. Add this code under the local variables within the make_butterflies_move function

```
    ask peoples in-cone vis_rad vis_ang [   ; this sets up a vison cone with the parameters from vis_rad vis_ang to detects people
      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
    ]

    if show_vis_cone = true [               ; this will switch on the visualisation of the vision cone if the switch is set to true
      ask patches in-cone vis_rad vis_ang [ ; this sets up a vision cone to display the size of the cone by changing the patch color
        set pcolor red                      ; this sets the patch color to red
      ]
    ]
```

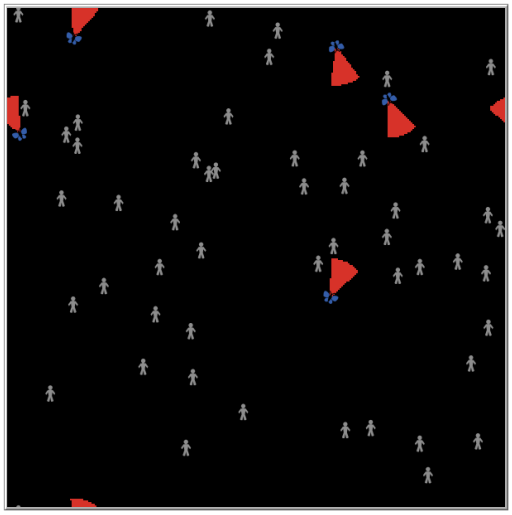
Once you have entered this code see what happens when you run it in the interface. You will notice that the vision cone leaves a trail which is not particularly helpful. To address this issue you will need to add some code to return the patch colors to black.

Step 9.3 – resetting the patch colors to black

In the go function uncomment the reset_patch_color function then add the following function below the go function (ensure it is outside of the go function!)

```
to reset_patch_color
  ask patches [                          ; this asks all of the patches in the population to do what is in the brackets
    set pcolor black                     ; this sets the color of each patch to black
  ]
end
```

Run the model again and see what happens, the vison cone should now work correctly.



Step 9.4 – creating a collision detection radius to detect collisions with people

In order to detect collisions with people we will setup a circle around the butterflies with the following code just below the vison cone in the make_butterflies_move function. The upper part detects collisions with people the lower part creates a visualisation.

```
    ask peoples in-radius rad [          ; this sets up a radius around the butterfly for collision detection with people using rad
      set hit true                       ; this sets the local variable called hit to true indicating that a person has collided with
    ]

    if show_col_rad = true [             ; this will switch on the visualisation of the collision radius if the switch is set to true
      ask patches in-radius rad [        ; this sets up a radius around the butterfly to display the size of the collison radius
        set pcolor orange                ; this sets the patch color to orange
      ]
    ]
```

Once you have entered this code run the model in the interface to see what happens.

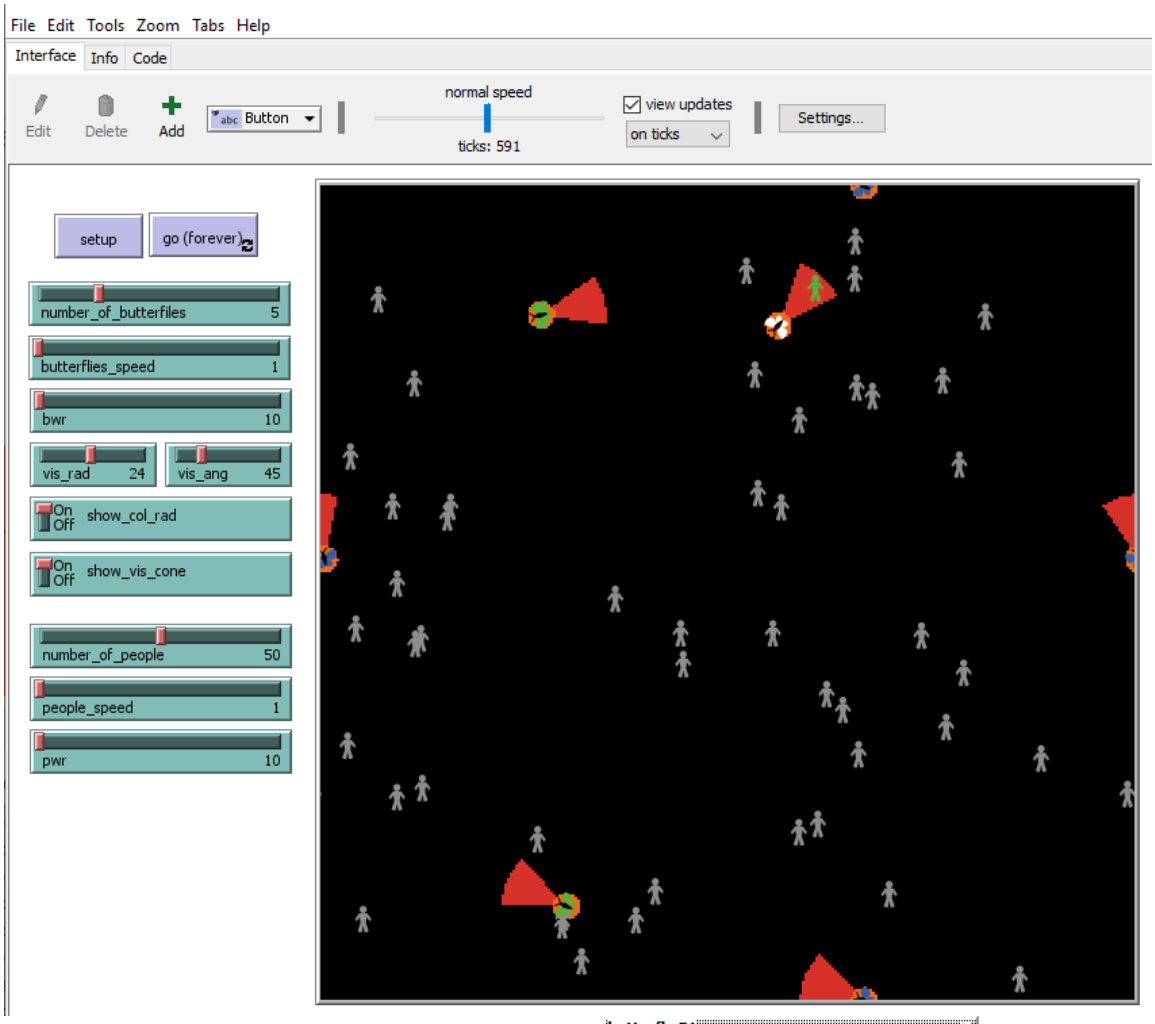
Step 9.5 – updating the hit counters and making the butterflies reactive

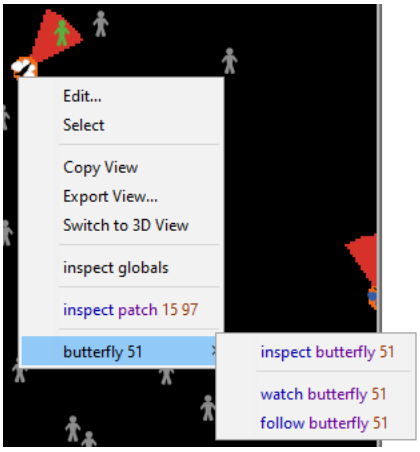
The following code uses if statements based on the local variables to update the behaviour and hit counters of the butterflies. Add this code below the collision detection code.

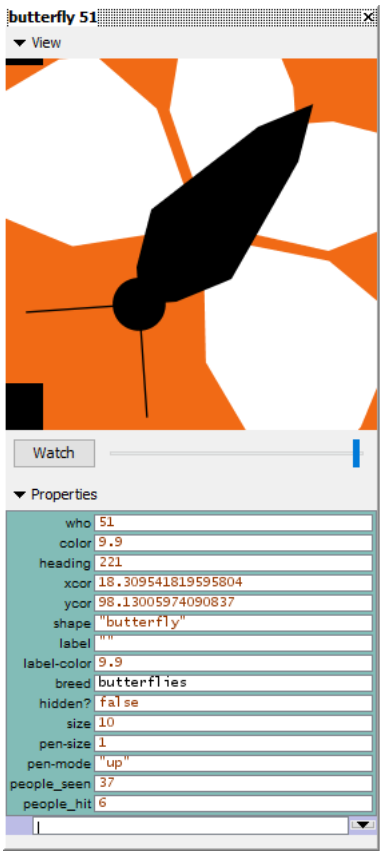
```
    ifelse seen = true [                 ; if then else statement based on the local variable seen, if seen = true then...
      set people_seen people_seen + 1    ; add 1 to the people_seen count
      set color white                    ; set color of butterfly to white
      right 180                          ; 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
    ]

    if hit = true [                      ; if statement based on the local variable hit, if seen = true then...
      set people_hit people_hit + 1      ; add 1 to the people_hit count
      set color green                    ; et color of butterfly to green
    ]
```

Step 10 – Observe and play

Once you have added this code run the model in the interface. The butterflies should try to avoid the people when they see them. By stopping the model and right clicking on a butterfly you can see the variables of that butterfly. You can also click watch to make it easier to track the butterfly







The full code

```
breed [ peoples people ]                    ; creating a population of people who will move around aimlessly
breed [ butterflies butterfly ]             ; creating a population of butterflies who will move around aimlessly but also seen the people

butterflies-own [ people_seen people_hit ]  ; this creates 2 variable which will be used to count the total people seen and total people hit

globals [rad]                               ; this creates a global variable called rad

to setup                                    ; this creates a function called setup
  clear-all                                 ; this clears the world of any previous activities
  reset-ticks                               ; this resets the ticks counter
  set rad 5                                 ; this sets the global variable rad to 3

  create-peoples number_of_people [         ; this creates the number of people that your global variable states
    setxy random-xcor random-ycor           ; this sets the starting position of the people to a random location in the world
    set color gray                          ; this sets the color of the people to gray
    set size 10                             ; this sets the size of the people to 10
    set shape "person"                      ; this sets the shape of the people to a person
  ]

  create-butterflies number_of_butterfiles [; this creates the number of butterflies that your global variable states
    setxy random-xcor random-ycor           ; this sets the starting position of the butterflies to a random location in the world
    set color blue                          ; this sets the color of the butterflies to blue
    set size 10                             ; this sets the size of the butterflies to 10
    set shape "butterfly"                   ; this sets the shape of the butterflies to a butterfly
  ]
end

to go                                       ; this creates a function called go
  make_people_move                          ; this calls the make_people_move function
  reset_patch_color                         ; this calls the reset_patch_color function
  make_butterflies_move                     ; this calls the make_butterflies_move function
  tick                                      ; this adds 1 to the tick counter
end

to make_people_move                         ; this creates a function called make_people_move
  ask peoples [                             ; this asks all of the people in the population to do what is in the brackets
    set color gray                          ; this sets the color of each person to gray
    right ( random pwr - ( pwr / 2))        ; this turns the person right relative to its current heading by a random degree number using
    forward people_speed                    ; this sets the speed at which the people move
  ]
end

to reset_patch_color
  ask patches [                             ; this asks all of the patches in the population to do what is in the brackets
    set pcolor black                        ; this sets the color of each patch to black
  ]
end

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

    set color blue                          ; this sets the color of each butterfly to blue
    let seen [false]                        ; this creates a local variable called seen
    let hit [false]                         ; this creates a local variable called hit

    ask peoples in-cone vis_rad vis_ang [   ; this sets up a vison cone with the parameters from vis_rad vis_ang to detects people
      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
    ]

    if show_vis_cone = true [               ; this will switch on the visualisation of the vision cone if the switch is set to true
      ask patches in-cone vis_rad vis_ang [ ; this sets up a vision cone to display the size of the cone by changing the patch color
        set pcolor red                      ; this sets the patch color to red
      ]
    ]

    ask peoples in-radius rad [             ; this sets up a radius around the butterfly for collision detection with people using rad
      set hit true                          ; this sets the local variable called hit to true indicating that a person has collided with
    ]

    if show_col_rad = true [                ; this will switch on the visualisation of the collision radius if the switch is set to true
      ask patches in-radius rad [           ; this sets up a radius around the butterfly to display the size of the collison radius
        set pcolor orange                   ; this sets the patch color to orange
      ]
    ]

    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
    ]

    if hit = true [                         ; if statement based on the local variable hit, if seen = true then...
      set people_hit people_hit + 1         ; add 1 to the people_hit count
      set color green                       ; et color of butterfly to green
    ]

    forward butterflies_speed               ; moves butterfly forward by the butterflies_speed variable
  ]
end
```