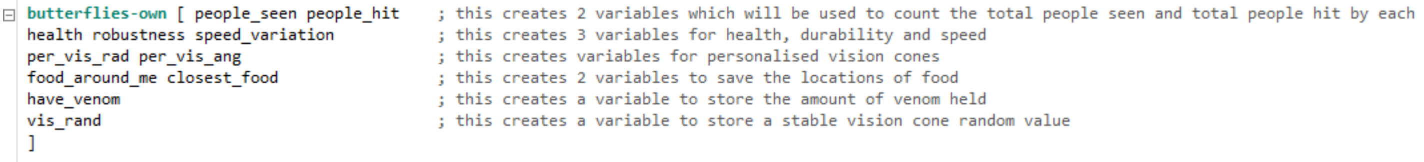
NetLogo – building on #4, making our world more complex

Step 1 – download and open the netlogo #4 file from canvas

Please download model file from Practical Resources 4 on canvas as this will be our starting point.

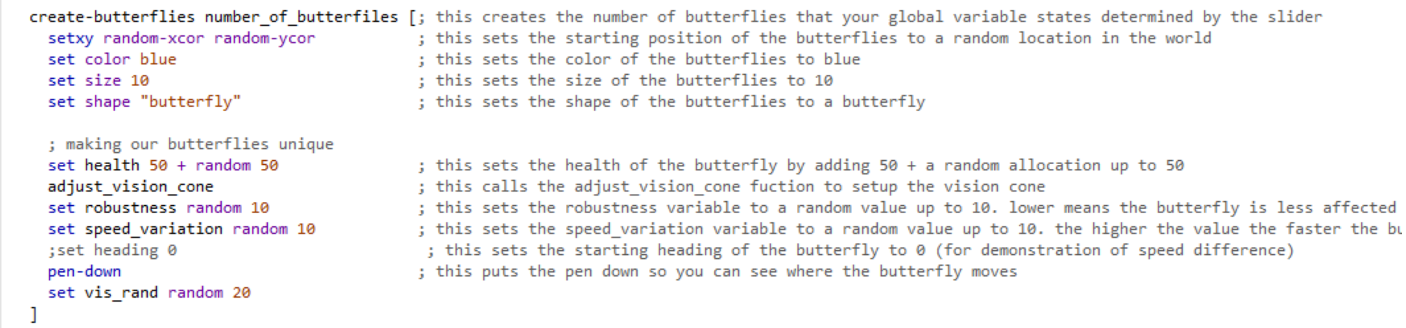
Step 2 – fixing an error in our existing code

The adjust_vision_cone code has a small error can you spot it? To fix this error we will create a butterfly variable called vis_rand and sent this to random 20 in the setup function. The code should look as follows.



```
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 each
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
vis_rand                                    ; this creates a variable to store a stable vision cone random value
]
```

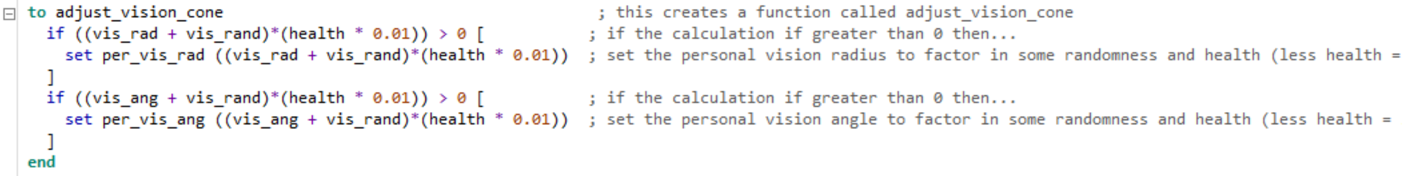
In the setup function add 1 line to the bottom of the create-butterflies code.



```
  create-butterflies number_of_butterfiles [; this creates the number of butterflies that your global variable states determined by the slider
    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

    ; making our butterflies unique
    set health 50 + random 50              ; this sets the health of the butterfly by adding 50 + a random allocation up to 50
    adjust_vision_cone                     ; this calls the adjust_vision_cone fuction to setup the vision cone
    set robustness random 10               ; this sets the robustness variable to a random value up to 10. lower means the butterfly is less affected
    set speed_variation random 10          ; this sets the speed_variation variable to a random value up to 10. the higher the value the faster the bu
    ;set heading 0                          ; this sets the starting heading of the butterfly to 0 (for demonstration of speed difference)
    pen-down                               ; this puts the pen down so you can see where the butterfly moves
    set vis_rand random 20
  ]
```

Now in the adjust_vison_cone function edit the code by replacing random 20 with vis_rand as seen below.

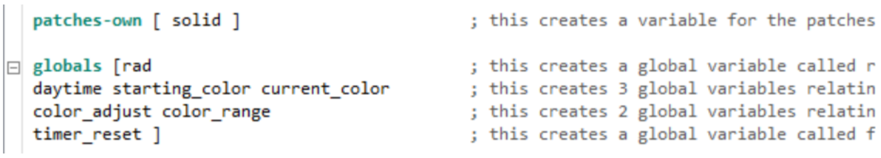


```
to adjust_vision_cone                                 ; this creates a function called adjust_vision_cone
  if ((vis_rad + vis_rand)*(health * 0.01)) > 0 [    ; if the calculation if greater than 0 then...
    set per_vis_rad ((vis_rad + vis_rand)*(health * 0.01))  ; set the personal vision radius to factor in some randomness and health (less health =
  ]
  if ((vis_ang + vis_rand)*(health * 0.01)) > 0 [    ; if the calculation if greater than 0 then...
    set per_vis_ang ((vis_ang + vis_rand)*(health * 0.01))  ; set the personal vision angle to factor in some randomness and health (less health =
  ]
end
```

Changing this code means that the initialised random value stays the same rather than being adjusted every time the function is called.

Step 3 – creating more global variable to make our model more flexible

In order make our model easy to modify we will create some additional global variables to regulate day and night along with the timer reset. Add the following global variables.

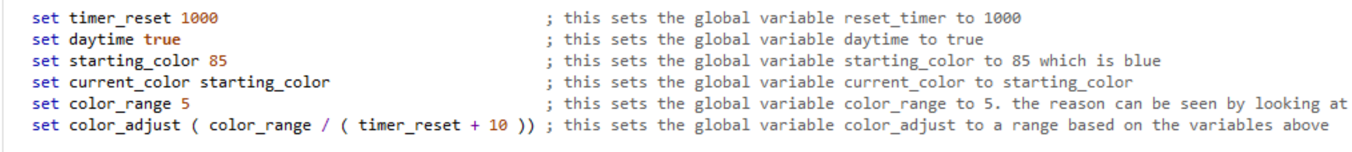


```
patches-own [ solid ]                   ; this creates a variable for the patches

globals [rad                            ; this creates a global variable called r
daytime starting_color current_color    ; this creates 3 global variables relatin
color_adjust color_range                ; this creates 2 global variables relatin
timer_reset ]                           ; this creates a global variable called f
```

Step 3.1 – setting our variables up

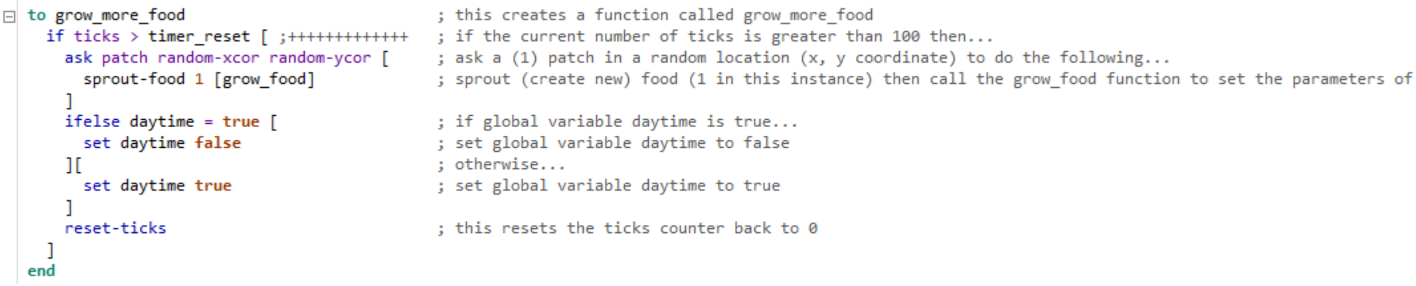
Setup the variable in the setup function to the following parameters.



```
  set timer_reset 1000                              ; this sets the global variable reset_timer to 1000
  set daytime true                                  ; this sets the global variable daytime to true
  set starting_color 85                             ; this sets the global variable starting_color to 85 which is blue
  set current_color starting_color                  ; this sets the global variable current_color to starting_color
  set color_range 5                                 ; this sets the global variable color_range to 5. the reason can be seen by looking at
  set color_adjust ( color_range / ( timer_reset + 10 )) ; this sets the global variable color_adjust to a range based on the variables above
```

Step 4 – adjusting the timer and adding functionality

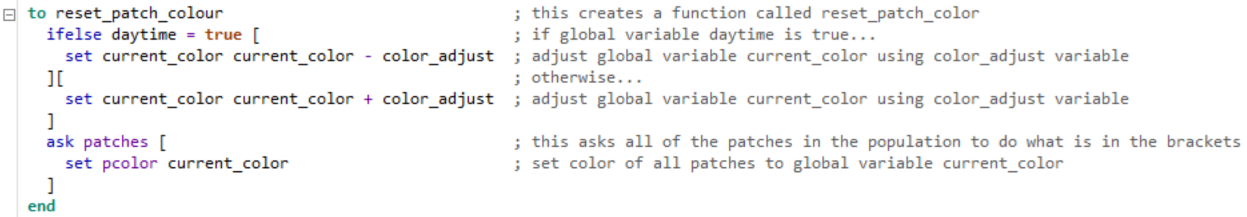
The grow_more_food function currently resets the ticks in our model every 1000 ticks. Go to this function and adjust the code as follows. This will create the beginning of a system we can use to create day and night within our model.



```
to grow_more_food                        ; this creates a function called grow_more_food
  if ticks > timer_reset [ ;+++++++++++++  ; if the current number of ticks is greater than 100 then...
    ask patch random-xcor random-ycor [   ; ask a (1) patch in a random location (x, y coordinate) to do the following...
      sprout-food 1 [grow_food]           ; sprout (create new) food (1 in this instance) then call the grow_food function to set the parameters of
    ]
    ifelse daytime = true [               ; if global variable daytime is true...
      set daytime false                   ; set global variable daytime to false
    ][                                    ; otherwise...
      set daytime true                    ; set global variable daytime to true
    ]
    reset-ticks                           ; this resets the ticks counter back to 0
  ]
end
```

Step 5 – creating day and night in our world

To make our world more interesting we will revisit the reset_patch_color code. Edit the code as seen below to create night and day in the model.

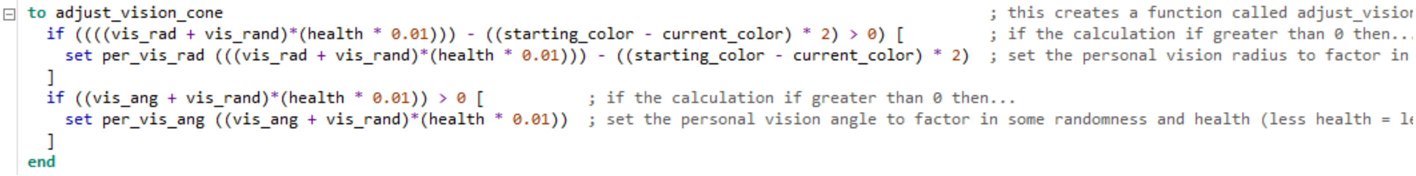


```
to reset_patch_colour                              ; this creates a function called reset_patch_color
  ifelse daytime = true [                          ; if global variable daytime is true...
    set current_color current_color - color_adjust  ; adjust global variable current_color using color_adjust variable
  ][                                               ; otherwise...
    set current_color current_color + color_adjust  ; adjust global variable current_color using color_adjust variable
  ]
  ask patches [                                    ; this asks all of the patches in the population to do what is in the brackets
    set pcolor current_color                       ; set color of all patches to global variable current_color
  ]
end
```

* Now test your model to see what happens. If all is working correctly you will have a background that changes color.

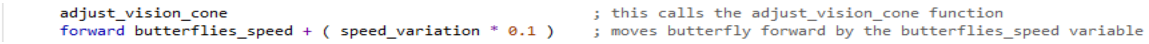
Step 6 – making day and night have an impact on the butterflies vision

In order for the day and night to have an impact on our butterflies range of vison we need to go back to the adjust_vison_cone code and modify this as follows.



```
to adjust_vision_cone                                                                  ; this creates a function called adjust_visior
  if ((((vis_rad + vis_rand)*(health * 0.01))) - ((starting_color - current_color) * 2) > 0) [    ; if the calculation if greater than 0 then...
    set per_vis_rad (((vis_rad + vis_rand)*(health * 0.01))) - ((starting_color - current_color) * 2)  ; set the personal vision radius to factor in
  ]
  if ((vis_ang + vis_rand)*(health * 0.01)) > 0 [          ; if the calculation if greater than 0 then...
    set per_vis_ang ((vis_ang + vis_rand)*(health * 0.01))  ; set the personal vision angle to factor in some randomness and health (less health = le
  ]
end
```

After you have done this you need to call the adjust_vison_cone function in the make_butterflies_move function as this will continually change now.

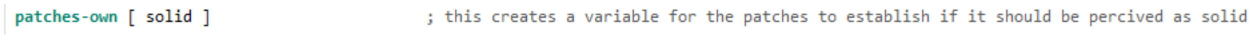


```
    adjust_vision_cone                               ; this calls the adjust_vision_cone function
    forward butterflies_speed + ( speed_variation * 0.1 )   ; moves butterfly forward by the butterflies_speed variable
```

It would also be good to remove the calls to the adjust_vison_cone function else where in the code as it will no longer be needed in the other locations. This can be done by searching using CTRL F.

Step 7 – creating buildings in the environment

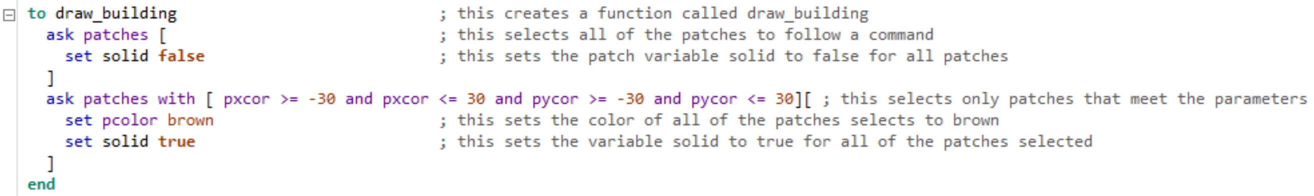
In order to create solid structures in the environment we need to add a patch variable at the beginning of the code as follows.



```
patches-own [ solid ]                     ; this creates a variable for the patches to establish if it should be percived as solid
```

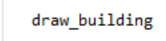
Step 7.1 – creating a function to setup our buildings

To create our building(s) we will create a function that we will then call in the setup. In this function we will change the color of the patches and also set the variable for each patch.



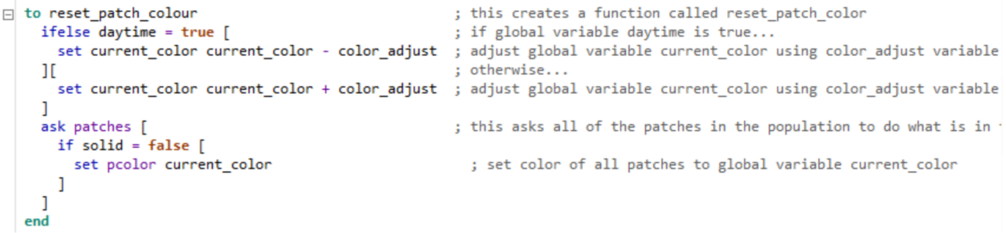
```
to draw_building                            ; this creates a function called draw_building
  ask patches [                             ; this selects all of the patches to follow a command
    set solid false                         ; this sets the patch variable solid to false for all patches
  ]
  ask patches with [ pxcor >= -30 and pxcor <= 30 and pycor >= -30 and pycor <= 30][ ; this selects only patches that meet the parameters
    set pcolor brown                        ; this sets the color of all of the patches selects to brown
    set solid true                          ; this sets the variable solid to true for all of the patches selected
  ]
end
```

Then call the function in setup.



```
  draw_building
```

Finally edit the reset_patch_colour function to only change the color of patches that are not solid.

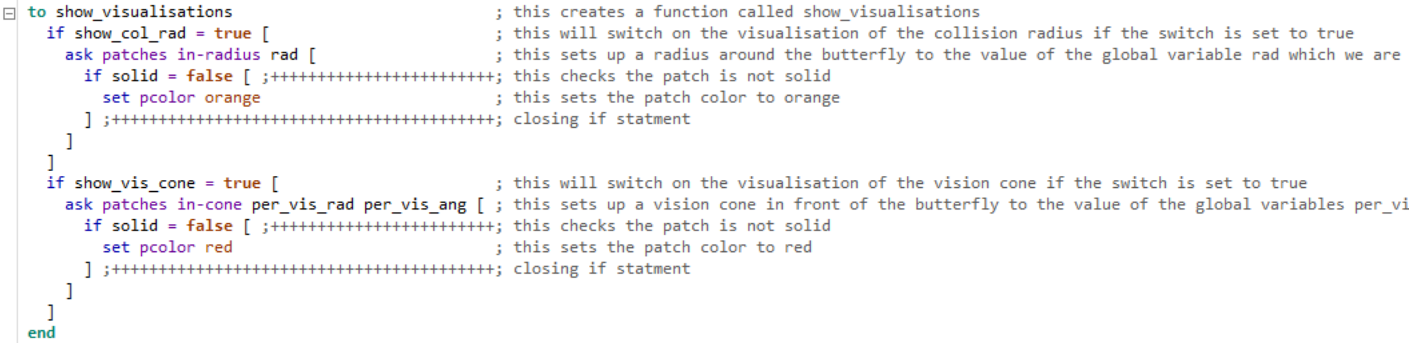


```
to reset_patch_colour                               ; this creates a function called reset_patch_color
  ifelse daytime = true [                           ; if global variable daytime is true...
    set current_color current_color - color_adjust  ; adjust global variable current_color using color_adjust variable
  ][                                                ; otherwise...
    set current_color current_color + color_adjust  ; adjust global variable current_color using color_adjust variable
  ]
  ask patches [                                     ; this asks all of the patches in the population to do what is in
    if solid = false [
      set pcolor current_color                        ; set color of all patches to global variable current_color
    ]
  ]
end
```

* Now test your model to see what happens. If all is working correctly you will have a brown square in the middle of your model, but see what happened when you run the model. Your agents will change the color of the brown square and go through it.

Step 7.2 – fixing the color changing issue

To fix this issue you need to revisit the show_visualisations function and add 2 if statements to check if the patch is solid or not as shown below.

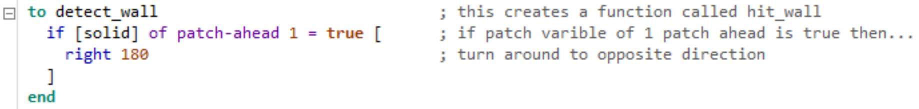


```
to show_visualisations                            ; this creates a function called show_visualisations
  if show_col_rad = true [                        ; this will switch on the visualisation of the collision radius if the switch is set to true
    ask patches in-radius rad [                   ; this sets up a radius around the butterfly to the value of the global variable rad which we are
      if solid = false [ ;++++++++++++++++++++++++; this checks the patch is not solid
        set pcolor orange                         ; this sets the patch color to orange
      ] ;+++++++++++++++++++++++++++++++++++++++++; closing if statment
    ]
  ]
  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 per_vis_rad per_vis_ang [ ; this sets up a vision cone in front of the butterfly to the value of the global variables per_vi
      if solid = false [ ;++++++++++++++++++++++++; this checks the patch is not solid
        set pcolor red                            ; this sets the patch color to red
      ] ;+++++++++++++++++++++++++++++++++++++++++; closing if statment
    ]
  ]
end
```

* Now test your model to see what happens. If all is working correctly the color changing issue is resolved but the agents in the model still pass through the solid object so this needs to be fixed.

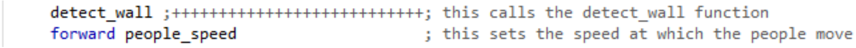
Step 7.3 – making the object solid to our agents

We next need to create a function for our agents to see/hit the wall, we will call this detected_wall, the code is as follows.



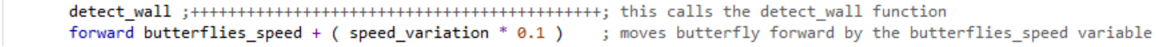
```
to detect_wall                              ; this creates a function called hit_wall
  if [solid] of patch-ahead 1 = true [      ; if patch varible of 1 patch ahead is true then...
    right 180                               ; turn around to opposite direction
  ]
end
```

Once you have created this function call the function in both the make_people_move code and the make_butterflies_move code just before the piece of code that moves the agent forward.



```
    detect_wall ;+++++++++++++++++++++++++++; this calls the detect_wall function
    forward people_speed                    ; this sets the speed at which the people move
```

And…



```
      detect_wall ;+++++++++++++++++++++++++++++++++++++++++++++; this calls the detect_wall function
      forward butterflies_speed + ( speed_variation * 0.1 )     ; moves butterfly forward by the butterflies_speed variable
```

* Now test your model to see what happens. If all is working correctly the agents will no longer pass through the building.