|  |
| --- |
| **Notes** |
| **ZOMBIE APOCALYPSE SIMULATION V12**  **WHAT IS IT?**  The agents in this model are HUMANS and ZOMBIES however, humans turn into zombies if they encounter a zombie and get bitten.  Zombies are reflex agents that always attack when confronted by a human. Humans are rational agents that make decisions based on their immediate enviroment and the actions they can take at that moment.  **HOW IT WORKS**  There are two main agents in this model, Zombies and Humans. The enviroment spawns two objects randomly anywhere on the map to help assist the humans to defeat the zombies. Food to regain their health when neccesary and weapons to have a fighting chance to defeat and kill the zombies. Food are denoted as a yellow fish icon and weapons are denoted as yellow 'x's.  In addition, the humans own the following abilities: Robustness to become strong, Speed variation to increase speed and Vision cone to be able to see the zombies and flee.  Furthermore, there are building/s that humans can pass through however, the zombies CANNOT pass through the buildings which makes the building a sort of SAFE ZONE for the humans.  If a zombie wins a fight and reduces the humans health to 0 then it bites the human, and the human turns into a zombie. Otherwise, the human may flee from the zombie or fight and kill the zombie using their weapon to avoid becoming infected.  **HOW TO USE IT**  Buttons   * SET-UP resets the simulation and to get ready * GO (FOREVER) to run the simulation and watch in action   Sliders   * number\_of\_humans to increase the amount of humans that are placed * humans\_speed to increase the speed variation of the humans * bwr to increase the range of the degree humans turn away from zombies * vis\_rad to increase the radius of the vision cone * vis\_ang to increase the angle of the vision cone * number\_of\_zombie to increase the amount of zombies placed * zombie\_speed to increase the speed variation of the zombies * pwr to increase the range of the degree zombies turn to humans   Switch   * show\_col\_rad to enable the radius around the human * show\_vis\_cone to enable the vision cone function   Monitors   * Zombies monitors the amount zombies are left * Humans monitors the amount of humans left * Food monitors the amount of food left * Weapon monitors the amount of weapons left     **Credits**  This script was created by Anthony Constant (AC). If you have any questions or suggestions, you can contact him at [anthonyconstant.co.uk/](https://anthonyconstant.co.uk/)  **License**  This script is released under the MIT License. See the LICENSE file for more details.  **GitHub**  Share Link: |

**NETLOGO COPY & PASTED LOCAL SOURCE CODE**

breed [ zombies zombie ] ; creating a population of zombie who will move around aimlessly

breed [ humans human ] ; creating a population of humans who will move around aimlessly but also seen the zombie

breed [ food fish ] ;creating a population of fish for food for the humans to eat to regain health!

breed [ weapon ammo ] ; creating a population of weapons/ammo as weapons to defend the humans!

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

humans-own [ zombie\_seen zombie\_encounter ;this creates 2 variables which will be used to count the total zombies seen and zombies encountered

health robustness speed\_variation ;this creates 3 variables for health, durability and speed

per\_vis\_rad per\_vis\_ang ;this creates variables for personlised vision cones

food\_around\_me closest\_food ;this creates 2 variables to save the location of food

have\_weapon ;this creates a variable to store the amount of weapon held

vis\_rand ;this creates a variable to store a stable vision cone random value

]

zombies-own [

human\_around\_zombie ;this creates a variable for the zombie to detect a human in it's radius

closest\_human ;this creates a variable to detect the closest human

]

food-own [ amount ] ;this creates a variable for the food to establish amount of the resource

globals [rad ;this creates a global variable called rad

daytime starting\_color current\_color ;this creates 3 global variables relating to creating day and night within our model

color\_adjust color\_range

timer\_reset ] ;this creates a global variable called for resetting the timer

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

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 95 ;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.

set color\_adjust ( color\_range / ( timer\_reset + 10 )) ;this sets the global variable color\_adjust to a range based on the variable above

create-zombies number\_of\_zombie [ ; this creates the number of zombie that your global variable states

setxy random-xcor random-ycor ; this sets the starting position of the zombie to a random location in the world

set color gray ; this sets the color of the zombie to gray

set size 10 ; this sets the size of the zombie to 10

set shape "person" ; this sets the shape of the zombie to a person

]

create-humans number\_of\_humans [; this creates the number of humans that your global variable states

setxy random-xcor random-ycor ; this sets the starting position of the humans to a random location in the world

set color red ; this sets the color of the humans to blue

set size 10 ; this sets the size of the humans to 10

set shape "person" ; this sets the shape of the humans to a human

set health 30 + random 10 ;sets the health of the human by adding 50 + a random allocation up to 50

adjust\_vision\_cone ;set up the vision cone

set robustness random 10 ;sets the robustness variable to a random value up to 10

set speed\_variation random 10 ;higher the number the faster they will go

;set heading 0 ; demonstrate it has impact

;pen-down ; this puts the pen down to see where the human moves (history of the human)

set vis\_rand random 5

ifelse show-health? ;show-health? switch

[ set label health ] ;show the health stat for humans

[ set label "" ] ;set string label

]

create-weapon 10 [ ;this creates X number of new weapons for the humans to store and use against the zombies

make\_weapon ;this calls the make\_weapon function

]

create-food 2 [ ;this creates X number of new food plants for the humans to feed from to regain health

grow\_food ;this calls the grow\_food function

]

draw\_building ;this calls the draw\_building function

end

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 the patches selects to brown

set solid true

]

end

to detect\_wall ;this creates a function called detect\_wall

if [solid] of patch-ahead 1 = true [ ;if patch variable of 1 patch ahead is true then...

right 180 ;turn around to opposite direction

]

end

to convert

ask turtles-on patch-here [ set breed zombies

set color gray

set size 10

set shape "person"]

end

to make\_weapon ;this creates a function called make\_weapon

setxy random-xcor random-ycor ;this sets the position of the weapon to a random location in the world

set color green ;this sets the color of the weapon to green

set size 5 ;this sets the size of the weapon to 5

set shape "x" ;this sets the weapon shape to an x

end

to grow\_food ;this creates a function called grow\_food

setxy random-xcor random-ycor ;this sets the position of the food to a random location in the world

set color yellow ;this sets the color of the food to yellow

set size 10 ;this sets the size of the food to 10

set shape "fish" ;this sets the shape of the food to a fish

set amount random 10 ;this sets the amount of food per plant to a random value up to ''

end

to go ; this creates a function called go

make\_zombie\_move ; this calls the make\_zombie\_move function

reset\_patch\_color ; this calls the reset\_patch\_color function

make\_humans\_move ; this calls the make\_humans\_move function

draw\_building ; calls the draw building function

tick ; this adds 1 to the tick counter

grow\_more\_food ; this calls the grow\_more\_food function

if not any? humans [ stop ] ; exits if there are no more humans

if not any? zombies [ stop ] ;exits if there are no more zombies

end

to make\_zombie\_move ; this creates a function called make\_zombie\_move

ask zombies[ ; this asks all of the zombie in the population to do what is in the brackets

set color gray ; this sets the color of each person to gray

let can\_see\_human human\_functions 30 ;set can\_see\_human radius to 30

ifelse ( can\_see\_human = true ) [ ;if can\_see\_human is true then...

set heading (towards closest\_human ) ;set zombie heading towards the closest human

]

[right ( random pwr - ( pwr / 2))] ; this turns the person right relative to its current heading by a random degree number using the range set within pwr NOTE: if negative it will turn left

detect\_wall ;this calls the detect\_wall function

forward zombie\_speed ; this sets the speed at which the zombie move

]

end

to-report human\_functions [sensitivity] ;this creates a reporting function called human\_functions and expects a value for sensitivity

set human\_around\_zombie other ( humans in-radius sensitivity ) ;this sets the human\_around\_zombie variable to the ID's of the human within the sensitivity radius

set closest\_human min-one-of human\_around\_zombie [ distance myself ] ;this sets the closest\_human variable to the ID of the closest human source

let can\_see\_human [false] ;set can\_see\_human to false

if (closest\_human != nobody ) [ ;if closest\_human is equal to nobody then...

set can\_see\_human true ;set can\_see\_human to true

]

report can\_see\_human ;return value of can\_see\_human to location where function was called

end

to reset\_patch\_color ;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

if solid = false [

set pcolor current\_color ; this sets the color of each patch to current\_color

]

]

end

to make\_humans\_move ;this is defining a function called make\_humans\_move

ask humans [ ;this asks all of the humans in the population to do what is in the brackets

ifelse health > 0 [ ;if health is greater than 0 ( still alive)...

show\_visualisations ;call the show\_visualtions function

set color red ;this sets the color of each human to red

let have\_seen\_zombie human\_function ;this creates a local variable called have\_seen\_zombie 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 value

pickup\_weapon ;this calls the pickup\_weapon function

ifelse ( have\_seen\_zombie = true ) [ ;if local variable have\_seen\_zombie is true...

right 180 ;set the heading of human to 180 (turn around to avoid zombie!)

][

ifelse ( can\_smell\_food = true ) [ ;if local variable can\_smell\_food is true...

set heading ( towards closest\_food ) ;set heading towards closest food source

][

right (random bwr - (bwr / 2)) ;this turns the human right relative to its current heading by a random degree number

]]

forward humans\_speed + ( speed\_variation \* 0.01 ) ;moves human forward by the humans\_speed variable

][

set color gray ;set color to gray to indicate dead human

convert ;this kills the human off

]

]

end

to show\_visualisations ; this creates a new function called show\_visualisation

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 zombie to display the size of the collison radius

if solid = false [

set pcolor orange ; this sets the patch color to orange

]

]

]

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 to display the size of the cone by changing the patch color

if solid = false [

set pcolor red ; this sets the patch color to red

] ;++++++++++++++++++++++++++++++++++ closing if statement

]

]

end

to-report food\_function [sensitivity] ;this creates a reporting function called food\_function abd expects a value

set food\_around\_me other ( food in-radius sensitivity )

set closest\_food min-one-of food\_around\_me [distance myself]

let can\_smell\_food [false] ;this creates a local variable valled can\_smell\_food and sets it to false

let eating\_food [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

ifelse amount > 0 [ ;if amount is greater than 0

set eating\_food true ;set the local variable called eating\_food to true indicating the human is eating

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

][

die ;there is no food left so kill the agent

]

]

]

if eating\_food = true [ ;if eating\_food is true then...

set health health + 5 ;add 5 health to the human

]

if (closest\_food != nobody) [ ;if closest\_food is not empty (the human 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 is being called

end

to-report human\_function

let seen [false] ; this creates a local variable called seen

let hit [false] ; this creates a local variable called hit

let zombie\_hit 0 ;this creates a local variable calls upon zombie\_hit and sets it to 0

ask zombies in-cone per\_vis\_rad per\_vis\_ang [ ; this sets up a vison cone with the parameters from vis\_rad vis\_ang to detects zombie

set color green ; this sets the color of the person detected within the vision code of the human to green

set seen true ; this sets the local variable called seen to true indicating that a person has been seen

]

ask zombies in-radius rad [ ; this sets up a radius around the human for collision detection with zombie using rad

set hit true ; this sets the local variable called hit to true indicating that a person has collided with

set zombie\_hit who ;this sets the local variable called person\_hit to the individual who

]

ifelse seen = true [ ; if then else statement based on the local variable seen, if seen = true then...

set zombie\_seen zombie\_seen + 1 ; add 1 to the zombie\_seen count

set color green ; set color of human to white

;right 180 ; set heading of the human to 180 (turn around to avoid!)

][ ; if seen = false...

;right (random bwr - (bwr / 2)) ; this turns the human right relative to its current heading by a random degree number using the range set within bwr NOTE: if negative it will turn left

]

if hit = true [ ; if statement based on the local variable hit, if seen = true then...

ifelse have\_weapon > 0 [ ;if have\_weapon is greater than 0 then...

ask zombie zombie\_hit [die] ;kills of the zombie hit

set have\_weapon have\_weapon - 1 ; remove 1 from the have\_weapon of the human

][

set zombie\_hit zombie\_hit + 1 ; add 1 to the zombie\_encounter count

set color green ;if hit by a zombie set human colour to green

set health health - robustness ;+++++++++++++

adjust\_vision\_cone ;+++++++++++++++++++++++++++++

]

]

report seen ;+++++++++++++++++++++++++++++++++++++

end

to pickup\_weapon ; this creates a function called pickup\_weapon

let pickup [false] ;this creates a loval variable called pickup and sets it to false

ask weapon in-radius rad [ ;this sets up a radius around the human to the value of the global variable rad which we are using for collision detection with the weapons to pick it up

set pickup true ;this sets thew local variable pickup to true

die

]

if pickup = true [ ;if pick is true then...

set have\_weapon have\_weapon + 1 ;add 1 to the have\_weapon count on the human

]

end

to adjust\_vision\_cone ;if statement is to check if health drop below 0 ( error checking )

if ((((vis\_rad + vis\_rand)\*(health \* 0.01))) - ((starting\_color - current\_color) \* 2) > 0) [ ;if not as healthy what they can see is being reduced

set per\_vis\_rad (((vis\_rad + vis\_rand)\*(health \* 0.01)) - ((starting\_color - current\_color) \* 2)) ;set the pesonal vision radius to factor in some randomness and health ( less health = less vision )

]

if ((vis\_ang + vis\_rand)\*(health \* 0.01)) > 0 [ ;if the calculation is 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 = less vision)

]

end

to grow\_more\_food ;this creates a new function called grow\_more\_food

if ticks > timer\_reset [ ;+++++++++++++++++++++ ;if the current numer of ticks is greater than 100 then...

ask patch random-xcor random-ycor [ ;ask the patch to do the following...

sprout-food 1 [grow\_food] ;sprout (create new) food (1 in this instance) then call grow\_food function to set the parametres

]

ifelse daytime = true [ ;if global variable daytime is true then...

set daytime false ;set global variable daytime to false

][ ;otherwise...

set daytime true ;set global variable daytime to true

]

reset-ticks ; this resets the tick counter back to default

]

end