```
__includes ["communication.nls" "bdi.nls"]

breed [ civilians civilian]
breed [ obstacles obstacle]
breed [ rescue-units rescue-unit]
breed [ rescued rescue-1]
breed [ bases base]
breed [ ambulances ambulance]

globals [rescued-cvls picked-up distance-traveled]
ambulances-own [beliefs intentions incoming-queue load]
bases-own [incoming-queue]

;;; Setting up the environment
to setup
   ;; (for this model to work with NetLogo's new plotting features,
   ;; __clear-all-and-reset-ticks should be replaced with clear-all at
   ;; the beginning of your setup procedure and reset-ticks at the end
   ;; of the procedure.)
   ;;__clear-all-and-reset-ticks
   clear-all
   reset-ticks
   random-seed seed
   setup-civilians
   setup-obstacles
   setup-ambulances
   setup-rescue-units
   setup-base
   set rescued-cvls 0
   set distance-traveled 0
   set picked-up 0
end

;;; creating disaster victims (civilians)
to setup-civilians
   create-civilians num-victims [
     rand-xy-co
     set shape "person"
     set color red
   ]
end

;;; creating obstacles
to setup-obstacles
   create-obstacles num-obstacles [
     rand-xy-co
     set shape "box"
     set color yellow
   ]
end

;;; creating ambulances
to setup-ambulances
```

```
   create-ambulances num-ambulances [
     rand-xy-co
     set shape "abulance"
     set color red
     set beliefs []
     set intentions []
     set incoming-queue []
     set load 0
     add-intention "do-nothing" timeout_expired 30
   ]
end
;;; creating Rescue units
to setup-rescue-units
   create-rescue-units num-rescue-units [
     rand-xy-co
     set shape "rescue-unit"
     set color blue
   ]
end

to setup-base
   create-bases 1 [
     set shape "triangle 2"
     set color red
     setxy 0 0
     set incoming-queue []
   ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;; END of SETUP

;;;; Experiment
to run-rescue
   if count civilians = 0 and count rescued = 0 [stop]
   ask bases [base-behaviour]
   ask ambulances [ambulance-behaviour]
   ask rescue-units [rescue-unit-behaviour]
   tick
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to base-behaviour
   let msg 0
   let performative 0

   while [not empty? incoming-queue]
   [
      set msg get-message
      set performative get-performative msg
      if performative = "inform" [allocate-the-rescue msg]
   ]
end
```

```
to allocate-the-rescue [msg]
   let coords (item 1 get-content msg)
   broadcast-to ambulances add-content (list "collect" coords) create-message
"request"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;
;;; Ambulance Agent
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Hybrid Layer.
to ambulance-behaviour
   if reactive-ambulance-unit [stop]
   collect-msg-update-intentions-unit
   execute-intentions
end

;;; Reactive layer of ambulance Agent.
to-report reactive-ambulance-unit
   if detect-ambulance [avoid-obstacle report true]
   if load >= maximum_load and at-dest base-id [set load 0] ;; unload
patients
   if load >= maximum_load [move-towards-dest base-id report true]
   if detect-civilian [rescue-civilian pick-up-victim report true]
   report false
end

;;; Ambulance unit proactive behaviour
to collect-msg-update-intentions-unit
   let msg 0
   let performative 0

   while [not empty? incoming-queue]
   [
     set msg get-message
     set performative get-performative msg
     if performative = "request" [
       add-belief get-content msg
       set intentions []
     ]
     if performative = "saved" [
       remove-belief get-content msg
       set intentions []
     ]
   ]

   if exist-beliefs-of-type "collect" and empty? intentions [
       let bel closer beliefs-of-type "collect"
       let coords item 1 bel
       add-intention "pick-up-victim" "true"
       if not at-dest coords [
```

```
         add-intention (word "move-towards-dest " coords) (word "at-dest "
coords)
       ]
    ]
end

;;; Reports the closest item in list.
;;; This reports the closer to the agent item from a list of items. The
coordinates of the
;;; different members in the list of items must be in a list as well. For
example
;;; the list must be of the form [ ["collect" [12 3] ["collect" [14 7]]]
to-report closer [itemlist]
   let closest first itemlist
   foreach itemlist
   [
      if distance-coords (item 1 ?) < distance-coords (item 1 closest)
        [set closest ?]
   ]
   report closest
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Rescue Units
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to rescue-unit-behaviour
   if detect-civilian [rescue-civilian inform-base stop]
   if detect-obstacle [avoid-obstacle stop]
   if true [move-randomly]
end

;;; Informing base for victim
;;; creates a message for the location of the victim, where the content is
;;; "victim-at" [xcor ycor]
to inform-base
   send add-receiver base-id add-content (list "victim-at" (list (round xcor)
(round ycor))) create-message "inform"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;; Sensors
;; Detecting obstacles
;; Obstacles are obstacles and other rescue agents.
to-report detect-obstacle
   foreach (list patches in-cone 2 30)
   [
     if any? obstacles-on ? [show "obstacle-on" report true]
     if any? other rescue-units-on ? [show "rescue-on" report true]
   ]
   report false
end

to-report detect-ambulance
```

```
   foreach (list patches in-cone 2 30)
   [
     if any? other ambulances-on ? [report true]
   ]
   report false
end

;;; detecting a civilian
to-report detect-civilian
   ifelse any? civilians-here
   [report true]
   [report false]
end

;;;; Returns true if an agent is at the specific destination.
to-report at-dest [dest]
   if is-number? dest [
      ifelse ([who] of one-of turtles-here = dest)
      [report true]
      [report false]
    ]

    if is-list? dest [
      ifelse (abs (xcor - first dest) < 0.4 ) and (abs (ycor - item 1 dest) <
0.4)
      [report true]
      [report false]
    ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;; Actions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; rescueing a civilian
to rescue-civilian
   set rescued-cvls rescued-cvls + 1
   ask one-of civilians-here [
      set breed rescued
      set shape "person"
      set color green
   ]
end

;;; Actions that move the agent around.
;;; Turning randomly to avod an obstacle
to avoid-obstacle
   set heading heading + random 360
end

;; moving randomly. First move then turn
to move-randomly
   fd 1
   set heading heading + random 30 - random 30
```

```
end

;;;;;;;;;;;;;;;;
to pick-up-victim
   ask rescued-here [die]
   set picked-up picked-up + 1
   set load load + 1
   broadcast-to ambulances add-content (list "collect" (list (round xcor)
(round ycor))) create-message "saved"
end

;;; Top level Reactive-traveling.
to move-towards-dest [dest]
   if true [travel-towards dest stop]
end

;;; Traveling towars a destination.
to travel-towards [dest]
    fd 0.2
    set distance-traveled distance-traveled + 0.2
    if is-number? dest
    [
      if not ((xcor = [xcor] of turtle dest) and (ycor = [ycor] of turtle
dest))
      [
        set heading towards-nowrap turtle dest
      ]
    ];; safe towards

    if is-list? dest
    [
      if not ((xcor = first dest) and (ycor = item 1 dest))
      [
        set heading towardsxy-nowrap (first dest) (item 1 dest)
      ]
    ];; safe towards
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Utilities
to rand-xy-co
   let x 0
   let y 0

   loop [
      set x random-pxcor
      set y random-pycor
      if not any? turtles-on patch x y and not (abs x < 4 and abs y < 4)
[setxy x y stop]
   ]
end
```

```
;;; Reports the distance from a set of coordinates [x y] that are given in  a
list eg [3 4]
to-report distance-coords [crds]
   report distancexy-nowrap (first crds) (item 1 crds)
end

;;; base ID is required to broadcasy a message to the base.
;;; This is intended for use with the add-receiver reporter.
to-report base-id
   report first [who] of bases
end
```