turtles-own [ home-pos new-heading ]

enemys-own [ flag? move-patch]

patches-own[father Cost-path visited? active?]

breed [ players player ]

breed [ monsters monster ]

breed [ enemys enemy ]

breed [ flags flag]

globals [

p-valids ;; holds valid patches for search algorithim

Start ;; holds start point for algorithim

Goal ;;holds goal point for algorithim

Final-Cost ;; holds the cost of the path for the algorithim

level ;; current level

level-over? ;; true when a level is complete

player-lives ;; remaining lives

red-lives ;; enemy 1 lives

yellow-lives ;; enemy 2 lives

blue-lives ;; enemy 3 lives

player-score ;; player score

red-score ;; enemy 1 score

yellow-score ;; enemy 2 score

blue-score ;; enemy 3 score

dead? ;; true when player loses a life

red-dead? ;; enemy death

yellow-dead? ;; enemy death

blue-dead? ;; enemy death

speed ;; player speed

red-speed ;; enemy 1 speed

yellow-speed ;; enemy 2 speed

blue-speed ;; enemy 3 speed

monster-speed ;; monster speed

count-down ;; Timer

visible? ;; Controls flag-turtle interaction

all-dead ;; if all enemies are dead

gamer-over? ;; Terminates

tool which-enemy ;; variable needed to properly load level.

]

;; Player, color green, turtle 1, index 0

;; Monster, color brown, turtle 2, index 1

;; Flag, color purple, turtle 3, index 2

;; Enemy 1, color red (15), turtle 4, index 3

;; Enemy 2, color yellow (45), turtle 5, index 4

;; Enemy 3, color blue (95), turtle 6, index 5

;;;;;;;;;;;;;;;;;;;;;;;;

;;; Setup Procedures ;;;

;;;;;;;;;;;;;;;;;;;;;;;;

to new ;; Observer Button

clear-all

set level 1

load-map

set level-over? false

reset-ticks

end

to load-map ;; Observer Procedure

;; Filename of Level Files

let maps ["playermap1.csv" "playermap2.csv" "playermap3.csv"]

ifelse ((level - 1) < length maps)

[ import-world item (level - 1) maps

set player-lives 3

set red-lives 3

set yellow-lives 3

set blue-lives 3

set player-score 0

set red-score 0

set yellow-score 0

set blue-score 0

set dead? false

set red-dead? false

set yellow-dead? false

set blue-dead? false

set speed 0.6

set red-speed 0.6

set yellow-speed 0.6

set blue-speed 0.6

set monster-speed 1

set all-dead 0

set visible? true

ask players

[

set home-pos list xcor ycor

set count-down 10

]

ask monsters

[ set home-pos list xcor ycor ]

ask patches

[

set father nobody

set Cost-path 0

set visited? false

set active? false

]

set p-valids patches with [pcolor != grey and pcolor != orange ]

ask enemys

[

set move-patch nobody

set home-pos list xcor ycor

set count-down 10

set flag? false

]

ask flags

[ set home-pos list xcor ycor ]

]

[ set level 1

load-map ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Runtime Procedures ;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;

; Patch report to estimate the total expected cost of the path starting from

; in Start, passing through it, and reaching the #Goal

to-report Total-expected-cost [#Goal]

report Cost-path + Heuristic #Goal

end

; Patch report to reurtn the heuristic (expected length) from the current patch

; to the #Goal

to-report Heuristic [#Goal]

report distance #Goal

end

to map-enemy

;;Set the relative starting point and goal point for the enemy turtle

ask enemys [

set Start patch-here

]

;;Find the goal aka the flag

(ifelse

flag visible? = true

[

ask flags [set Goal patch-here]

]

flag visible? = false

[

ask players [set goal patch-here]

])

end

; A\* algorithm. Inputs:

; - #Start : starting point of the search.

; - #Goal : the goal to reach.

; - #valid-map : set of agents (patches) valid to visit.

; Returns:

; - If there is a path : list of the agents of the path.

; - Otherwise : false

to-report A\* [#Start #Goal #valid-map]

; clear all the information in the patches

ask #valid-map with [visited?]

[

set father nobody

set Cost-path 0

set visited? false

set active? false

]

; Active the staring point to begin the searching loop

ask #Start

[

set father self

set visited? true

set active? true

]

; exists? indicates if in some instant of the search there are no options to

; continue. In this case, there is no path connecting #Start and #Goal

let exists? true

; The searching loop is executed while we don't reach the #Goal and we think

; a path exists

while [not [visited?] of #Goal and exists?]

[

; We only work on the valid pacthes that are active

let options #valid-map with [active?]

; If any

ifelse any? options

[

; Take one of the active patches with minimal expected cost

ask min-one-of options [Total-expected-cost #Goal]

[

; Store its real cost (to reach it) to compute the real cost

; of its children

let Cost-path-father Cost-path

; and deactivate it, because its children will be computed right now

set active? false

; Compute its valid neighbors

let valid-neighbors neighbors4 with [member? self #valid-map]

ask valid-neighbors

[

let t ifelse-value visited? [ Total-expected-cost #Goal] [2 ^ 20]

if t > (Cost-path-father + distance myself + Heuristic #Goal)

[

; The current patch becomes the father of its neighbor in the new path

set father myself

set visited? true

set active? true

; and store the real cost in the neighbor from the real cost of its father

set Cost-path Cost-path-father + distance father

set Final-Cost precision Cost-path 3

]

]

]

]

; If there are no more options, there is no path between #Start and #Goal

[

set exists? false

]

]

; After the searching loop, if there exists a path

ifelse exists?

[

; We extract the list of patches in the path, form #Start to #Goal

; by jumping back from #Goal to #Start by using the fathers of every patch

let current #Goal

set Final-Cost (precision [Cost-path] of #Goal 3)

let rep (list current)

While [current != #Start]

[

set current [father] of current

set rep fput current rep

]

report rep

]

[

; Otherwise, there is no path, and we return False

report false

]

end

; Axiliary procedure to lunch the A\* algorithm between patches

to look-for-goal

set Start patch-here

; Compute the path between Start and Goal

let path A\* Start Goal p-valids

; If any...

if path != false [if color = 15

[ ask enemys with [color = 15][

move-to one-of neighbors4 with [member? self path]

move-enemys

;;move-to one-of patches with[ neighbors4 = true and visited? = true]

]

]

if color = 45

[ ask enemys with [color = 45][

move-to one-of neighbors4 with [member? self path]

move-enemys

;; move-to one-of patches with[ neighbors4 = true and path = true]

]

]

if color = 95

[ ask enemys with [color = 95][

move-to one-of neighbors4 with [member? self path]

move-enemys

;;move-to one-of patches with[ neighbors4 = true and path = true]

]

]]

; Set the Goal and the new Start point

set Start Goal

end

to play ;; Observer Forever Button

player-lives-control

player-score-control

enemy-lives-control

enemy-score-control

;; special win condition

if all-dead = 3

[ set level-over? true

user-message "You Win - Last Man Standing" ]

(ifelse

level-over? = true and level != 3

[ (ifelse

player-score = 3 or all-dead = 3

[ user-message "Prepare - Next Level!"

set level level + 1

load-map

set level-over? false

stop ]

player-score != 3 or all-dead != 3

[ user-message "Unlucky - Try Again!"

set level level

load-map

set level-over? false

stop ])

]

level-over? = true and level = 3

[ user-message "Congratulations - Game Completed!"

stop ])

every speed ;; player speed

[ move-player ]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

every red-speed ;; red speed

[

ask enemys with [ color = 15 ]

[ ;;Find the goal aka the flag

ask flags [if visible? = true

[

ask flags[set Goal patch-here]

]]

ask flags [if visible? = false

[ask players[

if shape = "playerflag"[

ask players [set goal patch-here]]]]

ask enemys[if color = 15 and shape = "enemyflag"

[ask enemys with [color = 15][set goal patch -9 9]]

]

ask enemys[if color = 45 and shape = "enemyflag"

[ask enemys with [color = 45][set goal patch-here]]

]

ask enemys[if color = 95 and shape = "enemyflag"

[ask enemys with [color = 95][set goal patch-here]]

]]

look-for-goal]

;;[ move-enemys ]

]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

every yellow-speed ;; yellow speed

[

ask enemys with [ color = 45 ]

[

;;Find the goal aka the flag

ask flags [if visible? = true

[

ask flags[set Goal patch-here]

]]

ask flags [if visible? = false

[ask players[

if shape = "playerflag"[

ask players [set goal patch-here]]]]

ask enemys[if color = 15 and shape = "enemyflag"

[ask enemys with [color = 15][set goal patch-here]]

]

ask enemys[if color = 45 and shape = "enemyflag"

[ask enemys with [color = 45][set goal patch 9 9]]

]

ask enemys[if color = 95 and shape = "enemyflag"

[ask enemys with [color = 95][set goal patch-here]]

]]

look-for-goal]

;; [ move-enemys ]

]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

every blue-speed ;; blue speed

[

ask enemys with [ color = 95 ]

;;[ move-enemys ]

[

;;Find the goal aka the flag

ask flags [if visible? = true

[

ask flags[set Goal patch-here]

]]

ask flags [if visible? = false

[ask players[

if shape = "playerflag"[

ask players [set goal patch-here]]]]

ask enemys[if color = 15 and shape = "enemyflag"

[ask enemys with [color = 15][set goal patch-here]]

]

ask enemys[if color = 45 and shape = "enemyflag"

[ask enemys with [color = 45][set goal patch-here]]

]

ask enemys[if color = 95 and shape = "enemyflag"

[ask enemys with [color = 95][set goal patch 9 -9]]

]]

look-for-goal]

]

every monster-speed ;; monster speed

[

ask monsters [ move-monsters ]

]

display

end

to player-speed-control

(ifelse

shape = "player"

[ set speed 0.6 ]

shape = "playerflag"

[ set speed 0.8 ])

end

to player-lives-control

(ifelse

dead? = true

[

if player-lives != 0

[ set player-lives player-lives - 1

if player-lives = 0

[ set level-over? true

user-message "Game Over - No Lives Left!" ]

set dead? false

]

]

player-lives = 0

[ user-message "Game Over - No Lives Left!" ])

end

to player-score-control

if player-score = 3

[ set level-over? true

user-message "You Win - Mission Complete!" ]

end

to enemy-speed-control

if color = 15

[ (ifelse

shape = "enemy"

[ set red-speed 0.6 ]

shape = "enemyflag"

[ set red-speed 0.8 ])

ask patch-here

[ if pcolor = 53

[ ask enemys-here

[ if shape = "enemy-flag"

[ set red-speed 7 ]

]

set red-speed 5

]

]

]

if color = 45

[ (ifelse

shape = "enemy"

[ set yellow-speed 0.6 ]

shape = "enemyflag"

[ set yellow-speed 0.8 ])

ask patch-here

[ if pcolor = 53

[ ask enemys-here

[ if shape = "enemy-flag"

[ set yellow-speed 7 ]

]

set yellow-speed 5

]

]

]

if color = 95

[ (ifelse

shape = "enemy"

[ set blue-speed 0.6 ]

shape = "enemyflag"

[ set blue-speed 0.8 ])

ask patch-here

[ if pcolor = 53

[ ask enemys-here

[ if shape = "enemy-flag"

[ set blue-speed 7 ]

]

set blue-speed 5

]

]

]

end

to enemy-lives-control

if red-dead? = true

[

if red-lives != 0

[ set red-lives red-lives - 1

if red-lives = 0

[ set all-dead all-dead + 1

let red-dead (enemys) with [ color = 15 ]

ask red-dead

[ die ]

]

set red-dead? false

]

]

if yellow-dead? = true

[

if yellow-lives != 0

[ set yellow-lives yellow-lives - 1

if yellow-lives = 0

[ set all-dead all-dead + 1

let yellow-dead (enemys) with [ color = 45 ]

ask yellow-dead

[ die ]

]

set yellow-dead? false

]

]

if blue-dead? = true

[

if blue-lives != 0

[ set blue-lives blue-lives - 1

if blue-lives = 0

[ set all-dead all-dead + 1

let blue-dead (enemys) with [ color = 95 ]

ask blue-dead

[ die ]

]

set blue-dead? false

]

]

end

to enemy-score-control

(ifelse

red-score = 3

[ set level-over? true

user-message "Game Over - Red Wins!" ]

yellow-score = 3

[ set level-over? true

user-message "Game Over - Yellow Wins!" ]

blue-score = 3

[ set level-over? true

user-message "Game Over - Blue Wins!" ])

end

;; Move player

to move-player ;; Observer Procedure

ask players

[ player-speed-control

set heading new-heading

;; Player moves forward unless blocked by wall

if [pcolor] of patch-ahead 1 != gray

[

if [pcolor] of patch-here = black or [pcolor] of patch-here = brown

or [pcolor] of patch-here = cyan ;; Safe Tiles

[ fd 1 ]

if [pcolor] of patch-here = orange ;; Lava/Dead

[ set dead? true

setxy (item 0 home-pos) (item 1 home-pos)

if shape = "playerflag"

[ set shape "player"

ask flags [ setxy (item 0 home-pos) (item 1 home-pos)

set visible? true

show-turtle ]

]

]

if [pcolor] of patch-here = 53 ;; Marsh

[ fd 0

set count-down count-down - 1

if count-down = 0

[ fd 1

set count-down count-down + 10 ;; Reset timer

]

]

]

;; If player (flag) walks into enemy

if any? enemys-here with [ shape = "enemy" ] and shape = "playerflag" ;; WORKS

[

ask enemys-here

[ set shape "enemyflag"

set flag? true

]

set shape "player"

set dead? true

setxy (item 0 home-pos) (item 1 home-pos)

]

;; Copy from enemy

;; If player walks into enemy (flag)

if any? enemys-here with [ shape = "enemyflag" ] and shape = "player" ;; WORKS

[

ask enemys-here

[

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true])

set shape "enemy"

set flag? false

setxy (item 0 home-pos) (item 1 home-pos)

]

set shape "playerflag"

]

;; If player walks into monster

if any? monsters-here ;; If player has flag and dies to monster

[ set dead? true

if shape = "playerflag"

[ let flag-drop-pos list xcor ycor

ask flags [ setxy (item 0 flag-drop-pos) (item 1 flag-drop-pos)

set visible? true

show-turtle ]

set shape "player"

]

setxy (item 0 home-pos) (item 1 home-pos)

]

;; If player walks into flag

if any? flags-here and visible? = true ;; Pick up flag

[

set shape "playerflag"

ask flags [ hide-turtle

set visible? false ]

]

;; If player walks into home

if [pxcor] of patch-here = item 0 home-pos and [pycor] of patch-here = item 1 home-pos and shape = "playerflag" ;; Reset if flag is returned to spawn point

[ set shape "player"

set player-score player-score + 1

ask flags [ setxy (item 0 home-pos) (item 1 home-pos)

set visible? true

show-turtle ]

]

]

end

;; Move enemy

to move-enemys ;; Observer Procedure

;; Makes swapping enemy to enemy flag easier and more effecient - controls interaction of enemys on same patch

let nearby-enemy (enemys-on patch-here) with [ flag? = false ]

let flag-enemy (enemys-on patch-here) with [ flag? = true ]

ask enemys

[ enemy-heading

enemy-speed-control ]

;; If enemy walks into lava

if [pcolor] of patch-here = orange ;; Lava/Dead

[ ask enemys-here

[

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true ])

setxy (item 0 home-pos) (item 1 home-pos)

if shape = "enemyflag"

[ set shape "enemy"

set flag? false

ask flags [ setxy (item 0 home-pos) (item 1 home-pos)

set visible? true

show-turtle ]

]

]

]

;; If enemy (flag) walks into player

if any? players-here with [ shape = "player" ] and shape = "enemyflag" ;; WORKS

[

ask players-here

[ set shape "playerflag" ]

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true ])

set shape "enemy"

set flag? false

setxy (item 0 home-pos) (item 1 home-pos)

]

;; Copy from player

;; If enemy walks into player (flag)

if any? players-here with [ shape = "playerflag" ] and shape = "enemy" ;; WORKS

[

ask players-here

[ set shape "player"

set dead? true

setxy (item 0 home-pos) (item 1 home-pos)

]

set shape "enemyflag"

set flag? true

]

;; If enemy (flag) walks into enemy

if flag? = true

[

ask nearby-enemy

[

if shape = "enemy"

[ set shape "enemyflag"

set flag? true

]

ask flag-enemy

[

if shape = "enemyflag"

[

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true ])

set shape "enemy"

set flag? false

setxy (item 0 home-pos) (item 1 home-pos)

]

]

]

]

;; COPY ABOVE CODE

;; If enemy walks into enemy (flag)

if flag? = false

[

ask flag-enemy

[

if shape = "enemyflag"

[

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true ])

set shape "enemy"

set flag? false

setxy (item 0 home-pos) (item 1 home-pos)

]

ask nearby-enemy

[

if shape = "enemy"

[ set shape "enemyflag"

set flag? true

]

]

]

]

;; If enemy walks into monster

if any? monsters-here ;; If enemy has flag and dies to monster

[ ask enemys-here

[ if shape = "enemyflag"

[ let flag-drop-pos list xcor ycor

ask flags [ setxy (item 0 flag-drop-pos) (item 1 flag-drop-pos)

set visible? true

show-turtle ]

set shape "enemy"

set flag? false

]

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true ])

setxy (item 0 home-pos) (item 1 home-pos)

]

]

;; If enemy walks into flag

if any? flags-here and visible? = true ;; Pick up flag

[

ask enemys-here

[ set shape "enemyflag"

set flag? true ]

ask flags-here [ hide-turtle

set visible? false ]

]

;; If enemy walks into home

if [pxcor] of patch-here = item 0 home-pos and [pycor] of patch-here = item 1 home-pos and shape = "enemyflag" ;; Reset if flag is returned to spawn point

[ ask enemys-here

[

(ifelse

color = 15

[ set red-score red-score + 1 ]

color = 45

[ set yellow-score yellow-score + 1 ]

color = 95

[ set blue-score blue-score + 1 ])

set shape "enemy"

set flag? false ]

ask flags [ setxy (item 0 home-pos) (item 1 home-pos)

set visible? true

show-turtle ]

]

end

;; Enemy direction

to enemy-heading ;; Monster Procedure

let dirs enemy-clear-headings

let new-dirs remove enemy-opposite heading dirs

let monster-dir false

if length dirs = 1

[ set heading item 0 dirs ]

if length dirs = 2

[ ifelse see-monster item 0 dirs

[ set monster-dir item 0 dirs ]

[ ifelse see-monster item 1 dirs

[ set monster-dir item 1 dirs ]

[ set heading one-of new-dirs ]

]

]

if length dirs = 3

[ ifelse see-monster item 0 dirs

[ set monster-dir item 0 dirs ]

[ ifelse see-monster item 1 dirs

[ set monster-dir item 1 dirs ]

[ ifelse see-monster item 2 dirs

[ set monster-dir item 2 dirs ]

[ set heading one-of new-dirs ]

]

]

]

if length dirs = 4

[ ifelse see-monster item 0 dirs

[ set monster-dir item 0 dirs ]

[ ifelse see-monster item 1 dirs

[ set monster-dir item 1 dirs ]

[ ifelse see-monster item 2 dirs

[ set monster-dir item 2 dirs ]

[ ifelse see-monster item 3 dirs

[ set monster-dir item 3 dirs ]

[ set heading one-of new-dirs ]

]

]

]

]

if monster-dir != false and [pcolor] of patch-ahead 1 != black

[ set heading enemy-opposite heading ]

end

;; Enemy moving

to-report enemy-clear-headings ;; Enemy procedure

let dirs []

if [pcolor] of patch-at 0 1 != gray

[ set dirs lput 0 dirs ]

if [pcolor] of patch-at 1 0 != gray

[ set dirs lput 90 dirs ]

if [pcolor] of patch-at 0 -1 != gray

[ set dirs lput 180 dirs ]

if [pcolor] of patch-at -1 0 != gray

[ set dirs lput 270 dirs ]

report dirs

end

to-report enemy-opposite [dir]

ifelse dir < 180

[ report dir + 180 ]

[ report dir - 180 ]

end

;; If nps sees monster

to-report see-monster [dir] ;; Monster procedure

let saw-monster? false

let p patch-here

while [[pcolor] of p = black]

[ ask p

[ if any? monsters-here

[ set saw-monster? true ]

set p patch-at sin dir cos dir ;; next patch in direction dir

]

;; stop looking if you loop around the whole world

if p = patch-here [ report saw-monster? ]

]

report saw-monster?

end

;; Move monster

to move-monsters ;; Observer Procedure

ask monsters

[monsters-heading]

;; Monster moves forward on these tiles

if [pcolor] of patch-ahead 1 = black or [pcolor] of patch-ahead 1 = orange or [pcolor] of patch-ahead 1 = 53

[ fd 1 ]

;; Monster stops before these tiles

if [pcolor] of patch-ahead 1 = cyan or [pcolor] of patch-ahead 1 = brown ;; Bug fixed - caused contradicting directions

[

if [pcolor] of patch-at 0 1 = black

[ set heading 0 ]

if [pcolor] of patch-at 1 0 = black

[ set heading 90 ]

if [pcolor] of patch-at 0 -1 = black

[ set heading 180 ]

if [pcolor] of patch-at -1 0 = black

[ set heading 270 ]

]

;; Copy from player

;; If monster walks into player

if any? players-here ;; If player has flag and dies to monster

[ ask players-here

[ set dead? true

if shape = "playerflag"

[ let flag-drop-pos list xcor ycor

ask flags [ setxy (item 0 flag-drop-pos) (item 1 flag-drop-pos)

set visible? true

show-turtle ]

set shape "player"

]

setxy (item 0 home-pos) (item 1 home-pos)

]

]

;; Copy from enemy

;; If monster walks into enemy

if any? enemys-here ;; If enemy has flag and dies to monster

[ ask enemys-here

[ if shape = "enemyflag"

[ let flag-drop-pos list xcor ycor

ask flags [ setxy (item 0 flag-drop-pos) (item 1 flag-drop-pos)

set visible? true

show-turtle ]

set shape "enemy"

set flag? false

]

(ifelse

color = 15

[ set red-dead? true ]

color = 45

[ set yellow-dead? true ]

color = 95

[ set blue-dead? true ])

setxy (item 0 home-pos) (item 1 home-pos)

]

]

end

;; Monster direction based on player

to monsters-heading ;; Monster Procedure

let dirs monsters-clear-headings

let new-dirs remove monster-opposite heading dirs

let player-dir false

let enemy-dir false

if length dirs = 1

[ set heading item 0 dirs ]

if length dirs = 2

[ ifelse see-player item 0 dirs or see-enemy item 0 dirs

[ set player-dir item 0 dirs set enemy-dir item 0 dirs ]

[ ifelse see-player item 1 dirs or see-enemy item 1 dirs

[ set player-dir item 1 dirs set enemy-dir item 1 dirs ]

[ set heading one-of new-dirs ]

]

]

if length dirs = 3

[ ifelse see-player item 0 dirs or see-enemy item 0 dirs

[ set player-dir item 0 dirs set enemy-dir item 0 dirs ]

[ ifelse see-player item 1 dirs or see-enemy item 1 dirs

[ set player-dir item 1 dirs set enemy-dir item 1 dirs ]

[ ifelse see-player item 2 dirs or see-enemy item 2 dirs

[ set player-dir item 2 dirs set enemy-dir item 2 dirs ]

[ set heading one-of new-dirs ]

]

]

]

if length dirs = 4

[ ifelse see-player item 0 dirs or see-enemy item 0 dirs

[ set player-dir item 0 dirs set enemy-dir item 0 dirs ]

[ ifelse see-player item 1 dirs or see-enemy item 1 dirs

[ set player-dir item 1 dirs set enemy-dir item 1 dirs ]

[ ifelse see-player item 2 dirs or see-enemy item 2 dirs

[ set player-dir item 2 dirs set enemy-dir item 2 dirs ]

[ ifelse see-player item 3 dirs or see-enemy item 3 dirs

[ set player-dir item 3 dirs set enemy-dir item 3 dirs ]

[ set heading one-of new-dirs ]

]

]

]

]

(ifelse

player-dir != false

[ set heading player-dir

set monster-speed 0.6 ]

player-dir = false

[ set monster-speed 1 ])

(ifelse

enemy-dir != false

[ set heading enemy-dir

set monster-speed 0.6 ]

enemy-dir = false

[ set monster-speed 1 ])

;;if player-dir != false

;;[ set heading player-dir ]

;;if enemy-dir != false

;;[ set heading enemy-dir ]

end

;; Monster moving

to-report monsters-clear-headings;; Monster procedure

let dirs []

if [pcolor] of patch-at 0 1 != gray

[ set dirs lput 0 dirs ]

if [pcolor] of patch-at 1 0 != gray

[ set dirs lput 90 dirs ]

if [pcolor] of patch-at 0 -1 != gray

[ set dirs lput 180 dirs ]

if [pcolor] of patch-at -1 0 != gray

[ set dirs lput 270 dirs ]

report dirs

end

to-report monster-opposite [dir]

ifelse dir < 180

[ report dir + 180 ]

[ report dir - 180 ]

end

;; If monster sees player

to-report see-player [dir] ;; Monster procedure

let saw-player? false

let p patch-here

while [[pcolor] of p = black]

[ ask p

[ if any? players-here

[ set saw-player? true ]

set p patch-at sin dir cos dir ;; next patch in direction dir

]

;; stop looking if you loop around the whole world

if p = patch-here [ report saw-player? ]

]

report saw-player?

end

;; If monster sees npc

to-report see-enemy [dir] ;; Monster procedure

let saw-enemy? false

let p patch-here

while [[pcolor] of p = black]

[ ask p

[ if any? enemys-here

[ set saw-enemy? true ]

set p patch-at sin dir cos dir ;; next patch in direction dir

]

;; stop looking if you loop around the whole world

if p = patch-here [ report saw-enemy? ]

]

report saw-enemy?

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Interface Procedures ;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to move-up

ask players [ set new-heading 0 ]

end

to move-right

ask players [ set new-heading 90 ]

end

to move-down

ask players [ set new-heading 180 ]

end

to move-left

ask players [ set new-heading 270 ]

end