CSYS5010

Assignment 4

Fire Evacuation

Vun Chun Edwin Chang

Matthew Laurence William Graham

Huiting Xu

Contents

Introduction

The study of human behaviour is important in order to develop plans for efficient and safe evacuation in the event of a building fire. Models of human movement during a fire can also assist in strengthening the design of buildings such that they facilitate safe evacuation and offer the greatest degree of protection in the event of a disaster.

When a fire starts, the reactions of individual humans will always be shaped by their own personal response to the surrounding people and situations[2]. This could be argued that human evacuation process during fire is not merely dependent on individual human response, the environment is are also an influential factor that influence the efficiency of fire evacuation process[8].

In this report, we discussed our investigations into different exit placements in a stadium for fire evacuation in relation to the human evacuation process.

1. Model Design

Aims

The aims for this investigation include:

- To measure the effects of different exit placements for evacuation scenarios and propose optimal exit locations for a specific building environment in order to identify the best building structural design of the layouts we have used for safe and efficient evacuation.

- To measure the effect that the rate at which fire and smoke spreads and the impact that has on behaviour in an evacuation scenario in order to suggest material flammability requirements or ventilation requirements to reduce the likelihood of death or accident in an evacuation scenario.

- To minimise the number of deaths that occur in an evacuation scenario by imposing capacity limits in density to a specific building environment.

- Predict expected sequence of dangers to optimize evacuation maneuvers.

Scenario

The Building Environment, Fire and Smoke were represented as *Patches* and the Humans were represented as *Turtles*.

The Building environment consisted of:

- Walls, coloured as *Black*, which turtles would not pass through.

- Inner Region which consisted of: 1. Field coloured *Green* and 2. Stands coloured *Light Grey*.
- Outer Region (Lobby/Foyer) coloured *White.*

- Inner Exits, coloured as *Yellow*, which provided the turtles with the means of moving from the innermost region to the outermost region.

- Outer Exits, colour as *Orange*, which provided turtles with being removed from the scenario - these turtles were counted as those that had evacuated safely.
- Three different layouts for the exits:

1. Exits at the intersections of the walls

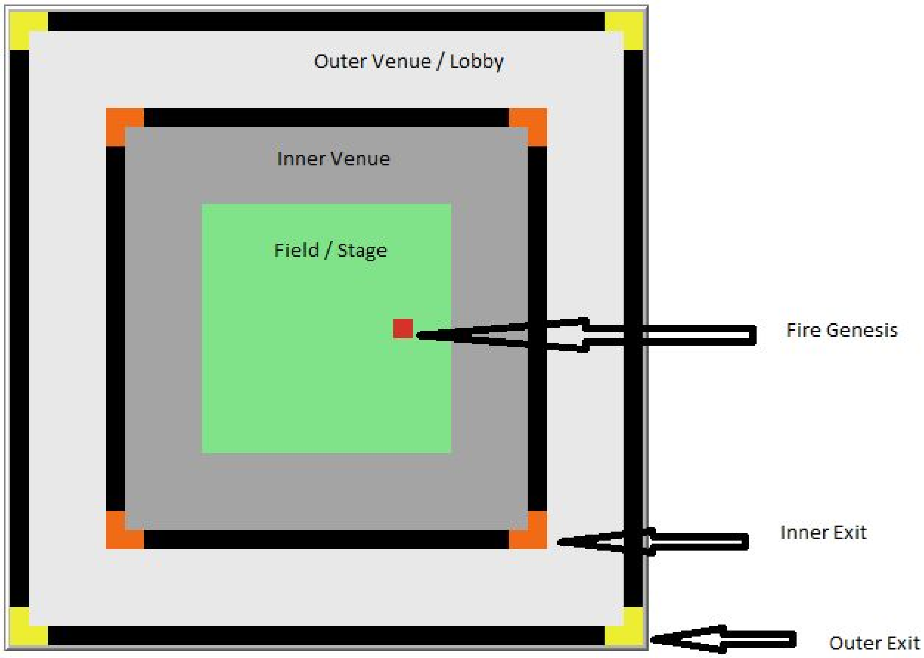


Diagram 1a

2. Exits within the wall segments

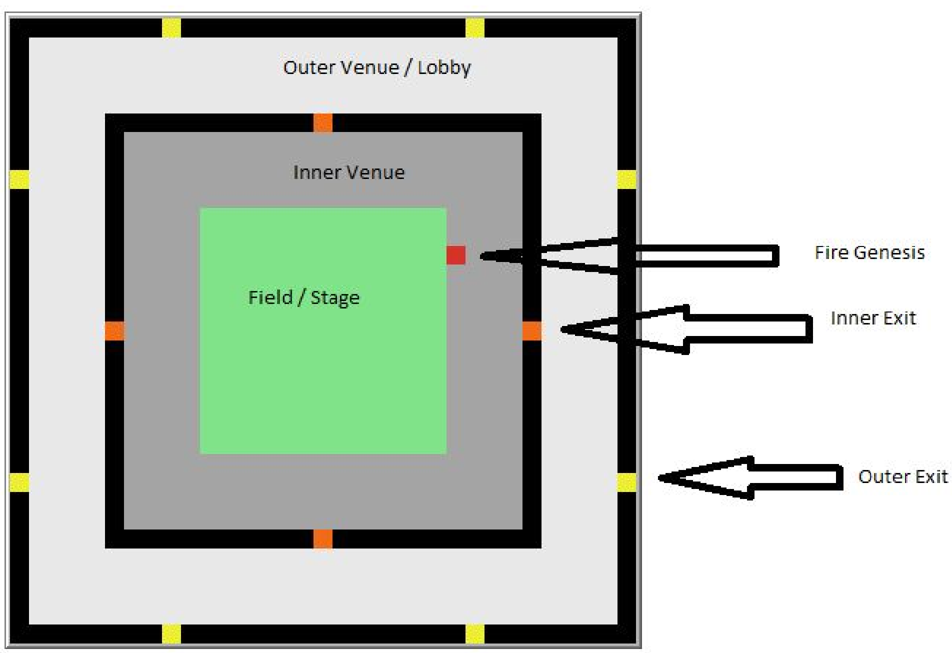


Diagram 1b

3. Randomly placed exit

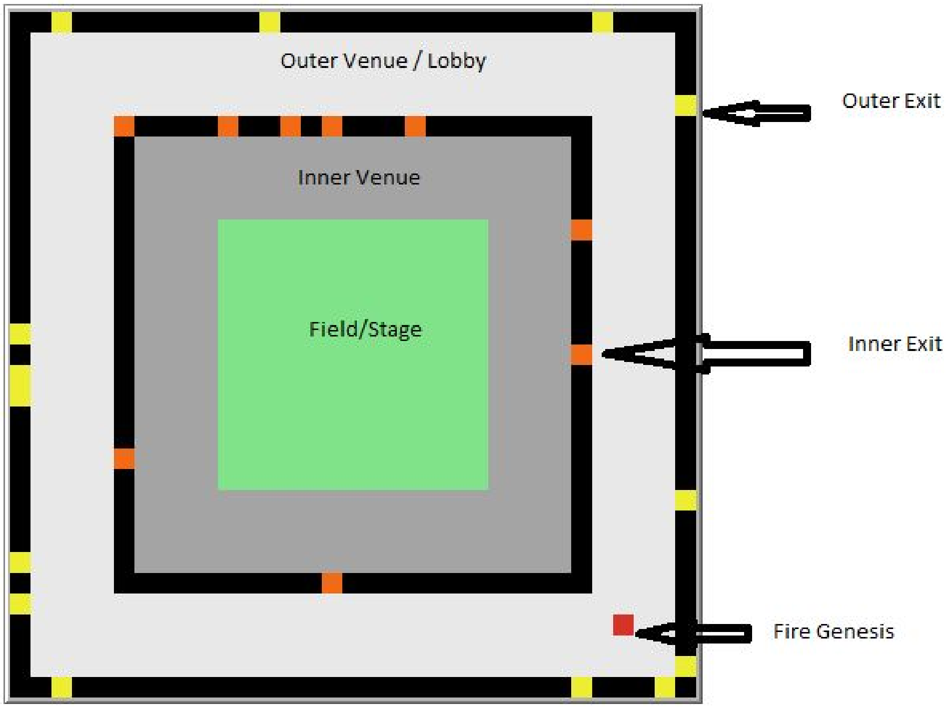


Diagram 1c

Behaviour

The Fire was coloured *Red.* Fire spreads from other fire patches.

The Smoke was coloured *Brown*.

- *Light Brown* => Light Smoke,
- *Medium Brown* => Medium Smoke
- *Dark Brown* => Heavy Smoke

The Turtles vary in colour depending on their behaviour:

- Calm, coloured as *Blue*, where the turtles will move randomly however keep out of a comfort distance of their neighbours.

- Panic with Exits in vision, coloured as *Purple*, where the turtles will move towards a linear combination of: The nearest exit, the nearest neighbour, the reflection point of the nearest patch of fire and; will keep out of a minimum comfort distance.

- Panic with no exits, coloured as *Cyan*, where the turtles will move towards a linear combination of The nearest neighbour, the reflection point of the nearest patch of fire; and will keep out of a minimum comfort distance.

- Panic with no exits and no neighbours coloured as *Green*, where the turtles will move towards a linear combination of: The nearest exit and the reflection point of the nearest patch of fire; and will keep out of a minimum comfort distance.

The turtles possess several internal variables (imposed by the conditions the turtle is exposed to) that will determine the coefficients for the linear combinations of movement behaviours:

- *Sight*; the distance from which turtles will search for nearby turtle and patch neighbours - Reduced by the smoke density in the turtle's current patch (smoke impairs vision)[1][3] and reduced by the number of turtles in the simulation (other turtles blocking vision)[2][4].

- *Comfort Distance*; the distance at which the turtles will keep away from each other - Reduced by the *fear* of the turtle[5][6](scared people more likely to scramble over each other and trample fellow turtles for its own survival).

- *Speed*; the movement distance of the turtle at each tick - Increased by the *fear* of the turtle (scared people more likely to exert more energy for survival) and reduced by *toxicity* (smoke inhalation can reduce aerobic capacity and thus physical performance)[1].

- *Fear*; how scared a turtle is - Increases the *speed* (see *speed*) and decreases the *comfort distance* (see *Comfort Distance*). Increased by the number of deaths a turtle has witnessed[5][6].

- *Toxicity*; how much smoke inhalation has occurred - Decreases the *speed* of the turtle (see *speed*) and increases *fear* (see *fear*). Rise in toxicity was dependant on the density of smoke of the turtle's patch[1][7].

There are several categories for removal of a turtle from the simulation and no possibilities of turtles being added implying the number of turtles is strictly monotonic decreasing over the number of ticks for any given simulation. These conditions are:

- *Evacuation* - Turtles that had reached the *outer* exits

- Death by *Fire* - Turtles that are in a patch of fire

- Death by *Smoke Inhalation* - Turtles that had accumulated excessive toxicity

- Death by *Trampling* - Turtles that have too many neighbours within a fixed distance (independent from *comfort distance*)

The Simulation terminates when the number of turtles reaches 0.

2. Implementation Process

To simulate this behaviour, we prepared a model in the NetLogo agent-based programming language and integrated modelling environment (version 6.0.1).

Observer Interactivity

The observer has access to manipulate the simulation via the NetLogo GUI interface. The ones included in this model include:

<u>Buttons</u>

- *Go* - Proceeds with the procedure loop (default is continuous rather than single tick).
- *SetScenario* - Implements the initial environmental conditions.
- *ResetData* - Removes all data currently stored
- *ExportData* - Writes a csv file containing the stored data

<u>Sliders</u>

- *FireSpreadRate* - Rate at which the fire will spread.
- *SmokeSpreadRate* - Rate at which the smoke will spread.
- *ExitTime* - Time taken for turtles to pass through an exit (represents something like time it takes to open a door).
- *TurtleMultiplier* - Varies number of Turtles

<u>Switches</u>

- *ReRun*? - Model begins a new simulation at termination.
- *RandomFireSpread?* - Randomises *FireSpreadRate* (gamma distribution).

- *RandomSmokeSpread?* - Randomises *SmokeSpreadRate* (gamma distibtuion).
- *RandomTurtles?* - Randomises number of initial turtles (uniform distribution)

Choosers

- *Scenario* - Chooses which exit scenario will be generated (includes a random scenario option that will randomly select one of the other options).
- *FirePosition* - Chooses whether the genesis patch of the fire will be in the innermost region or the outermost region (includes a random option that will randomly select either “Inner” or “Outer”)

Smoke would spreads from fire patches and smoke patches of equal or greater intensity.

Equations

Number of turtles spawned:

$$\begin{bmatrix} Field/Stage \\ Inner\ Arena \\ Outer\ Arena/Lobby \end{bmatrix} = \begin{bmatrix} 10 \\ 80 \\ 20 \end{bmatrix} * TurtleMultiplier$$

Fire Spread:

$$unif\{0, 1000\} < FireSpreadRate * PPFireDensity \Rightarrow FireSpreads$$

Smoke Spread:

$$unif\{0, 1000\} < SmokeSpreadRate * \begin{bmatrix} PPLightSmokeDensity \\ PPMediumSmokeDesnity \\ PPHeavySmokeDesnity \end{bmatrix} \Rightarrow \begin{bmatrix} Light\ Smoke\ Spreads \\ Medium\ Smoke\ Spread \\ Heavy\ Smoke\ Spreads \end{bmatrix}$$

Vision:

$$\begin{bmatrix} \textit{Vision No Smoke} \\ \textit{Vision Light Smoke} \\ \textit{Vision Medium Smoke} \\ \textit{Vision Heavy Smoke} \end{bmatrix} = \begin{bmatrix} 25 \\ 20 \\ 15 \\ 10 \end{bmatrix} * \left(\frac{2}{1+e^{\frac{\textit{Turtle Count}}{220}}} \right)$$

<u>Toxicity:</u>

$$\textit{Toxicity}_{t+1} = \textit{Toxicity}_t + \left\| \begin{bmatrix} \delta_{\textit{Smoke of Turtle's Patch}, \textit{No Smoke}} \\ \delta_{\textit{Smoke of Turtle's Patch}, \textit{Light Smoke}} \\ \delta_{\textit{Smoke of Turtle's Patch}, \textit{Medium Smoke}} \\ \delta_{\textit{Smoke of Turtle's Patch}, \textit{Heavy Smoke}} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0.261497 \\ 0.577215 \\ 1.353077 \end{bmatrix} \right\|$$

<u>Fear:</u>

$$\textit{Fear} = 137 + 2 * \textit{Toxicity} + e^{\pi} * \ln(\textit{DeathsWitnessed} + 1) + 10 * \textit{Number of Turtles} \in 1\,m\,\textit{radius} + \frac{1000}{\textit{Distance of nearest fire} + 0.01}$$

<u>ComfortSpace:</u>

$$\textit{ComfortSpace} = e^{-0.001 * \textit{Fear}}$$

<u>Speed:</u>

$$\textit{Speed} = \frac{e^{-\pi}}{1 + \frac{\pi}{\ln(\textit{Toxicity} + 1)}}$$

<u>FirePatchMultiplier (coefficient for linear combination of fire in behaviour):</u>

$$\textit{FirePatchMultiplier} = \frac{1}{\textit{Distance of nearest fire} + 0.000001} + e^{-0.567 * \textit{Distance of nearest fire} + \pi}$$

<u>ExitPatchMultiplier (coefficient for linear combination of exits in behaviour):</u>

$$ExitPatchMultiplier = \frac{3}{Distance\ of\ nearest\ exit + 0.001} * e^{e^{Distance\ of\ nearest\ exit} - 1}$$

<u>NearestTurtleMultiplier (coefficient for linear combination of nearest neighbour in behaviour)</u>:

$$NearestTurtleMultiplier_{Distance > ComfortSpace} = \frac{1}{2} * NearestTurtleDistance * e^{\frac{1}{Vision} * NearestTurtleDistance}$$

$$NearestTurtleMultiplier_{Distance \leq ComfortSpace} = \frac{1}{NearestTurtleDistance + 0.0001}$$

Difficulties

In the process of implementation, there were a multitude of difficulties in attempting to coerce the agents to behaving in a way that was qualitatively accurately.

One specific behaviour that was difficult to implement was preventing turtles from running into patches of fire if either their neighbours or the exit was on the opposite side of the fire. This was made difficult due to the nature of the direction function used “facexy” which points towards a specific point.

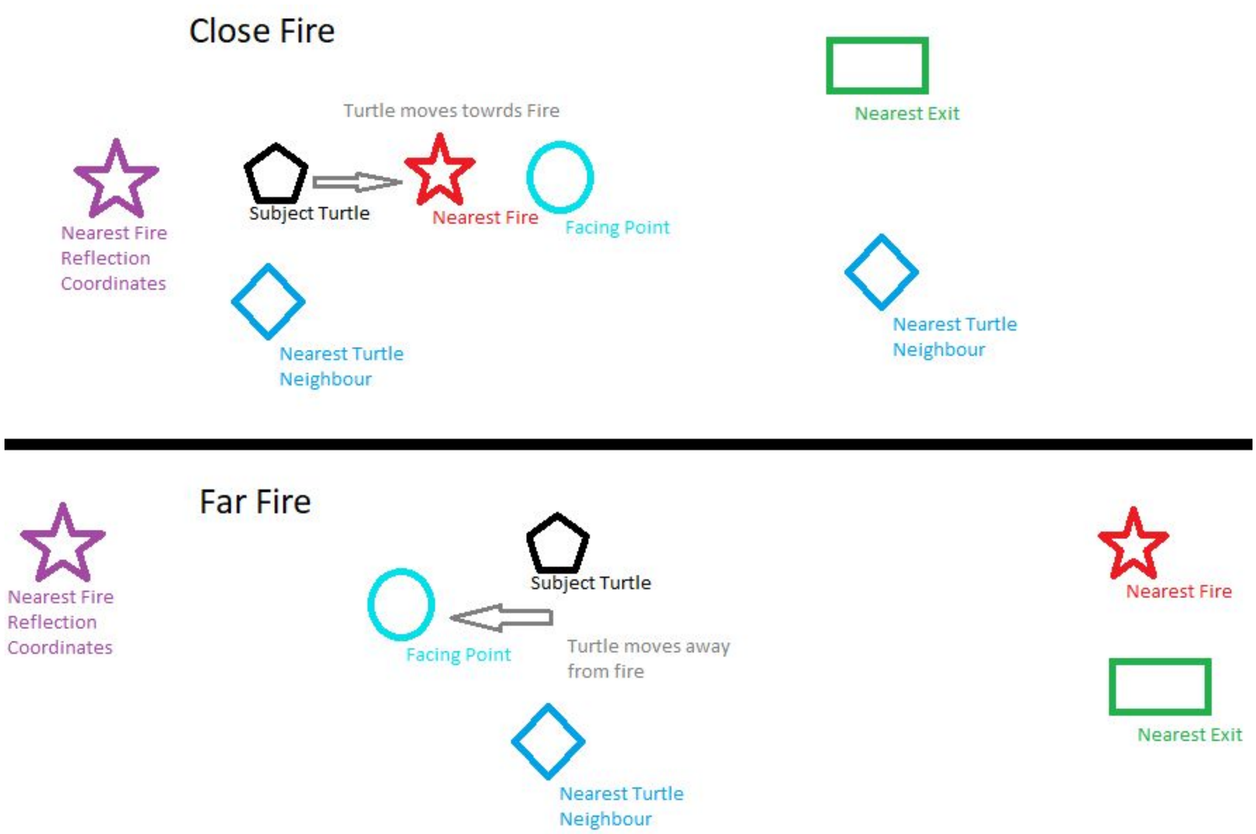


Diagram 2a

In order to combat this behaviour, the linear combination coefficient was large as the distance to the fire approached 0 and small for large distances (see Equations and Diagrams).

3. Results and Analysis

To collect the data and run the analysis, we used the NetLogo R extension that allowed us to export the data we collected from each run. We first ran 1024 (2^{10}) simulations with the variables randomised collecting the data. Every run was initialised on a 32x32 grid with coercing the inner walls into dimensions 22x22.

Variables (each run)

Independent Variables

- Number of Turtles, isomorphic to “Occupational Load” (Number / Area)
- Exit Scenario
- FireSpreadRate
- SmokeSpreadRate
- FirePosition
- Distance of genesis fire from nearest inner exit

Dependant Variables

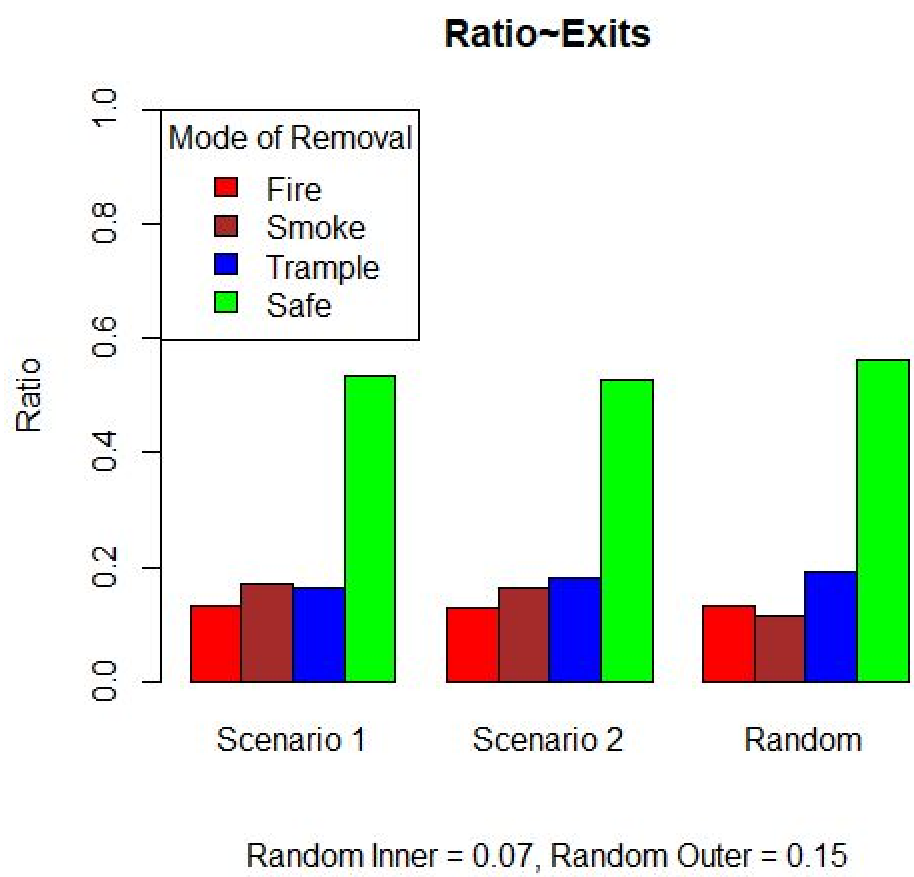
- Duration of Scenario (ticks)
- Number of deaths/evacuees for each category
- Mean and Standard Deviation of time (ticks) of death/evacuation for each category

Data

Upon analysis of the data in regards to the exit position, it appeared as though the "Random Exits" scenario was equally effective at evacuating the patrons with 56% evacuation rate compared to 53% for both Scenario 1 and Scenario 2. (Graph 1a)

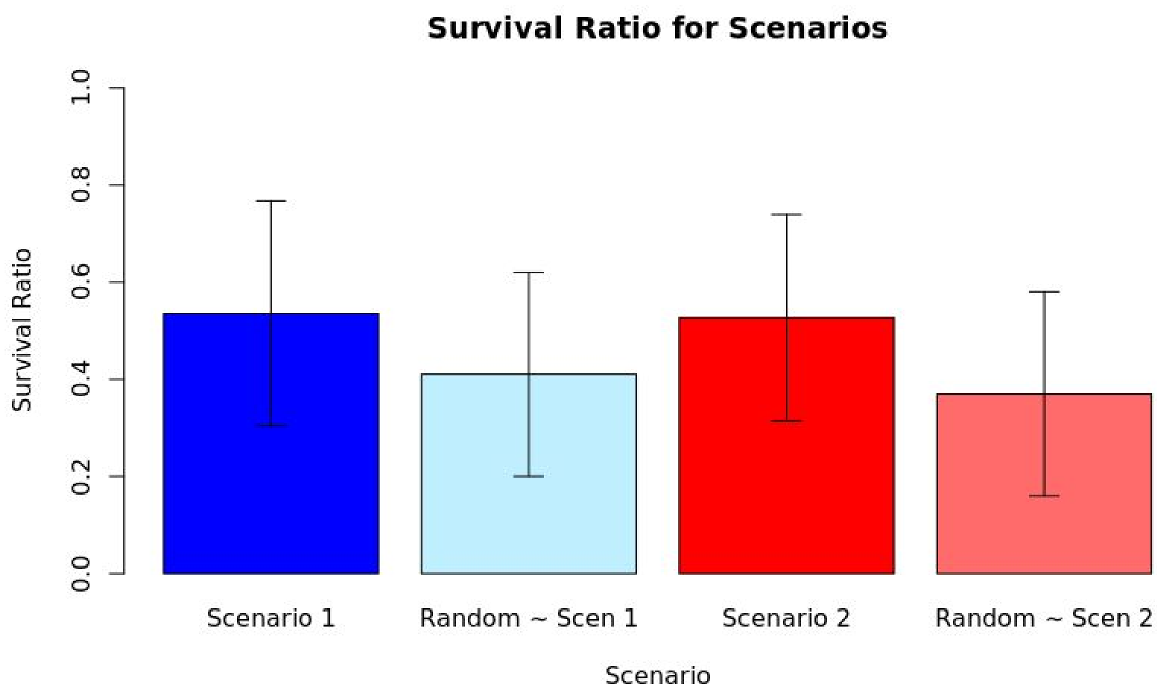
The number of "inner" exits in the "Random Exits" case was on average 5.88 (7% chance over 84 patches) and "outer exits" on average 18.6 (15% chance over 124 patches).

Scenario 1 has a fixed number of 12 "inner" and 12 "outer" exit patches (although only 8 for each inner and outer were accessible for any turtle given the nature of the corners) and scenario 2 has a fixed number of 4 "inner" and 8 "outer" exit patches.



Graph 1a

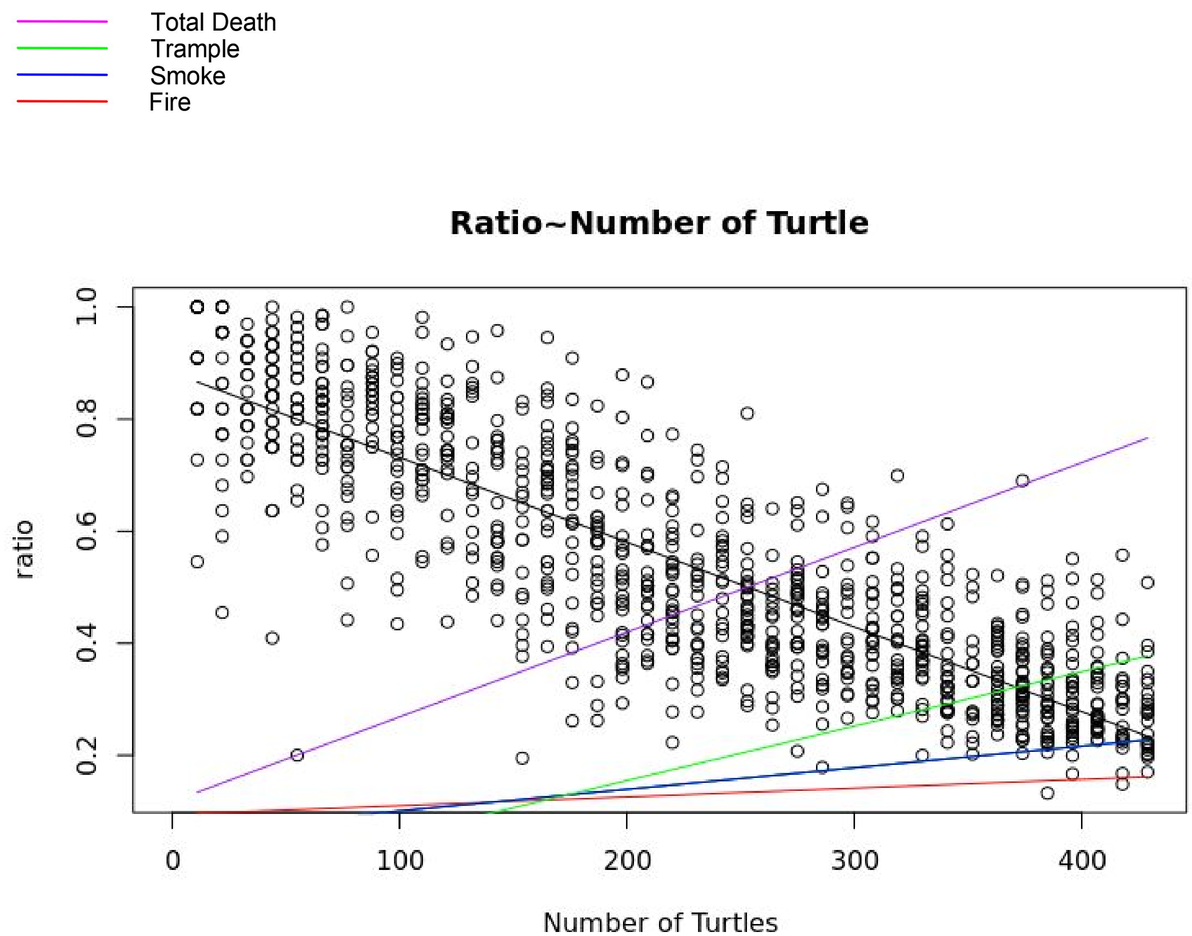
In order to address this, another 256 simulations were run over the “Random Exits” scenario, where the number of exits was randomised to use as a control case.
Upon running the simulations varying the number of “inner” exits and “outer” exits, a multivariate linear regression was used to map the survival rate. When adjusting the “inner” and “outer” exit values to match those of Scenario 1 and Scenario 2. (Graphs 1b)



Graph 1b

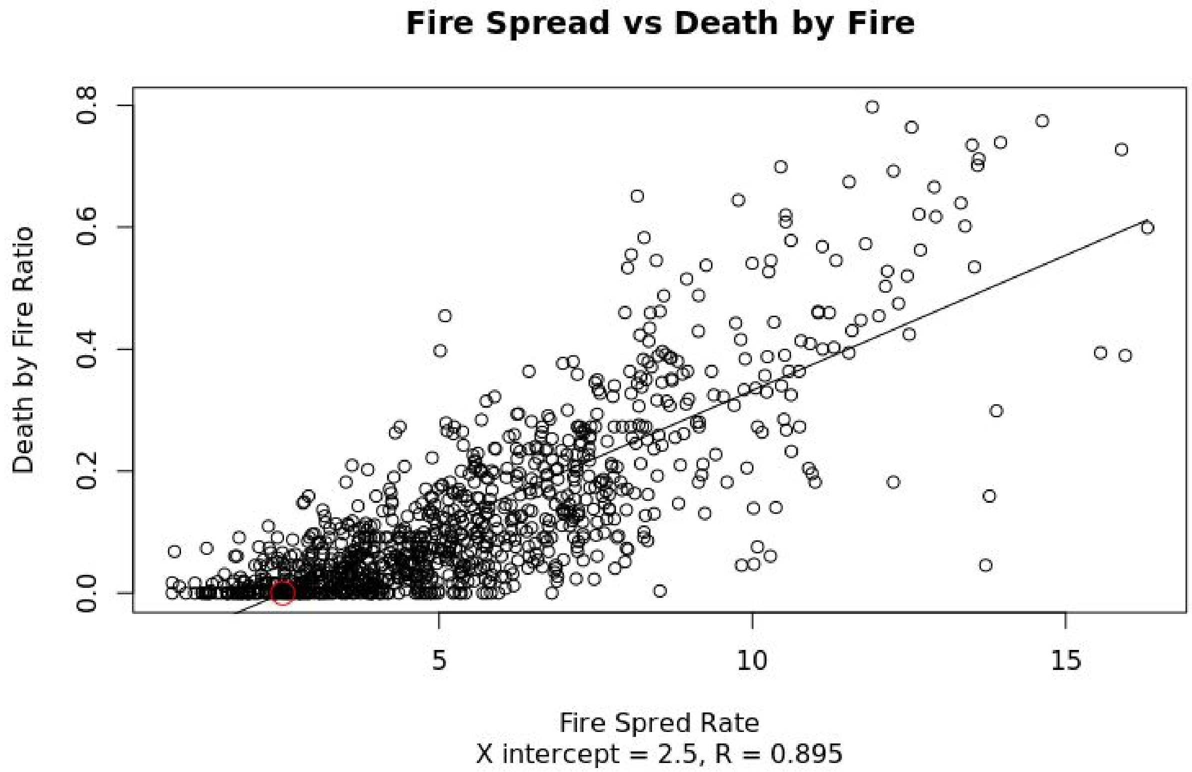
Graph 1b shows that the average survival rate for the randomly placed exits is less than the average for the chosen scenarios, however due to the large spread of results, this result is not statistically significant.

To optimize the capacity density of the stadium, we looked at the ratio of removal modes compared to the number of turtles that the arena originally contains. The points in plot 1a depict the samples of the survival ratio for a given number of turtles and the black line represents the linear regression of the survival ratio. The other coloured lines represent the linear regression of the modes of removal. As the infimum of the survival rate has consistent results below 0.2 occurring from the number of turtles at 143, our recommendation is that the density is kept below 0.18 turtles/m^2. (Plot 1a)



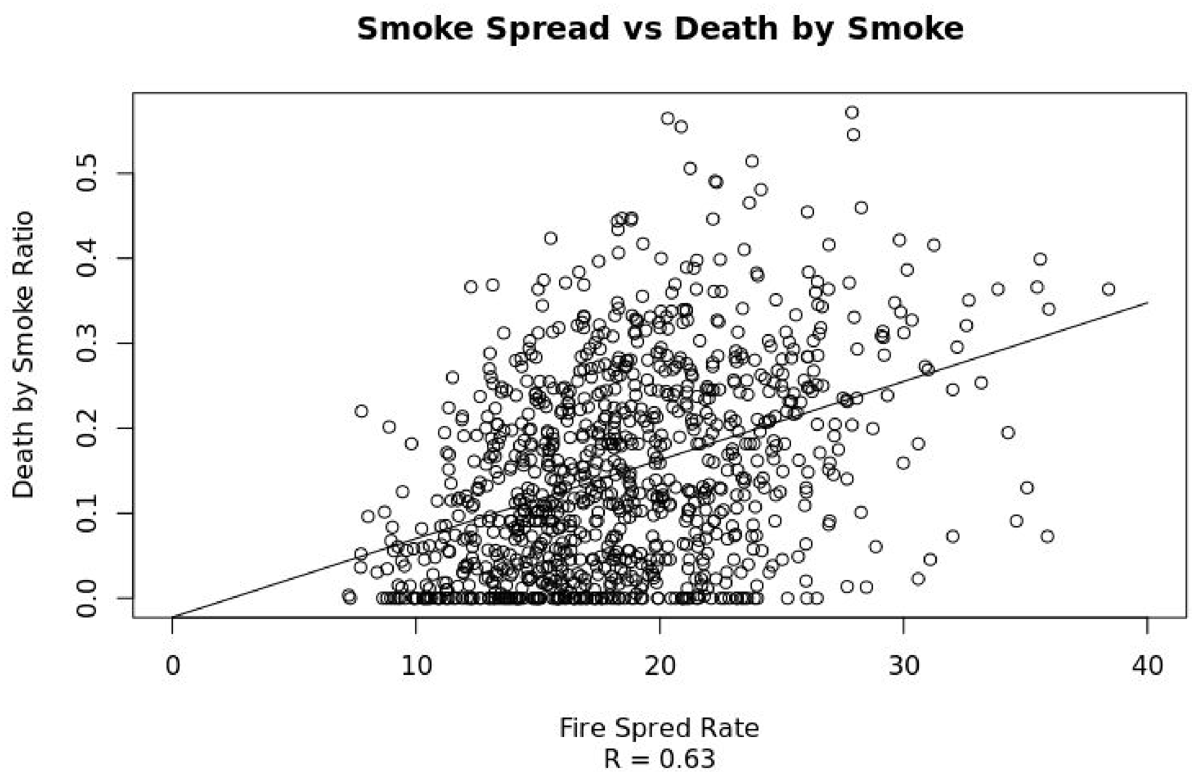
Plot 1

To approximate optimal levels of material flammability or sprinkler abundance, we measured the relationship between the ratio of deaths from fire relative to the total turtles for the scenario (Plot 2a). The intercept occurs when the FireSpreadRate is 2.5 which is a reasonable recommendation.



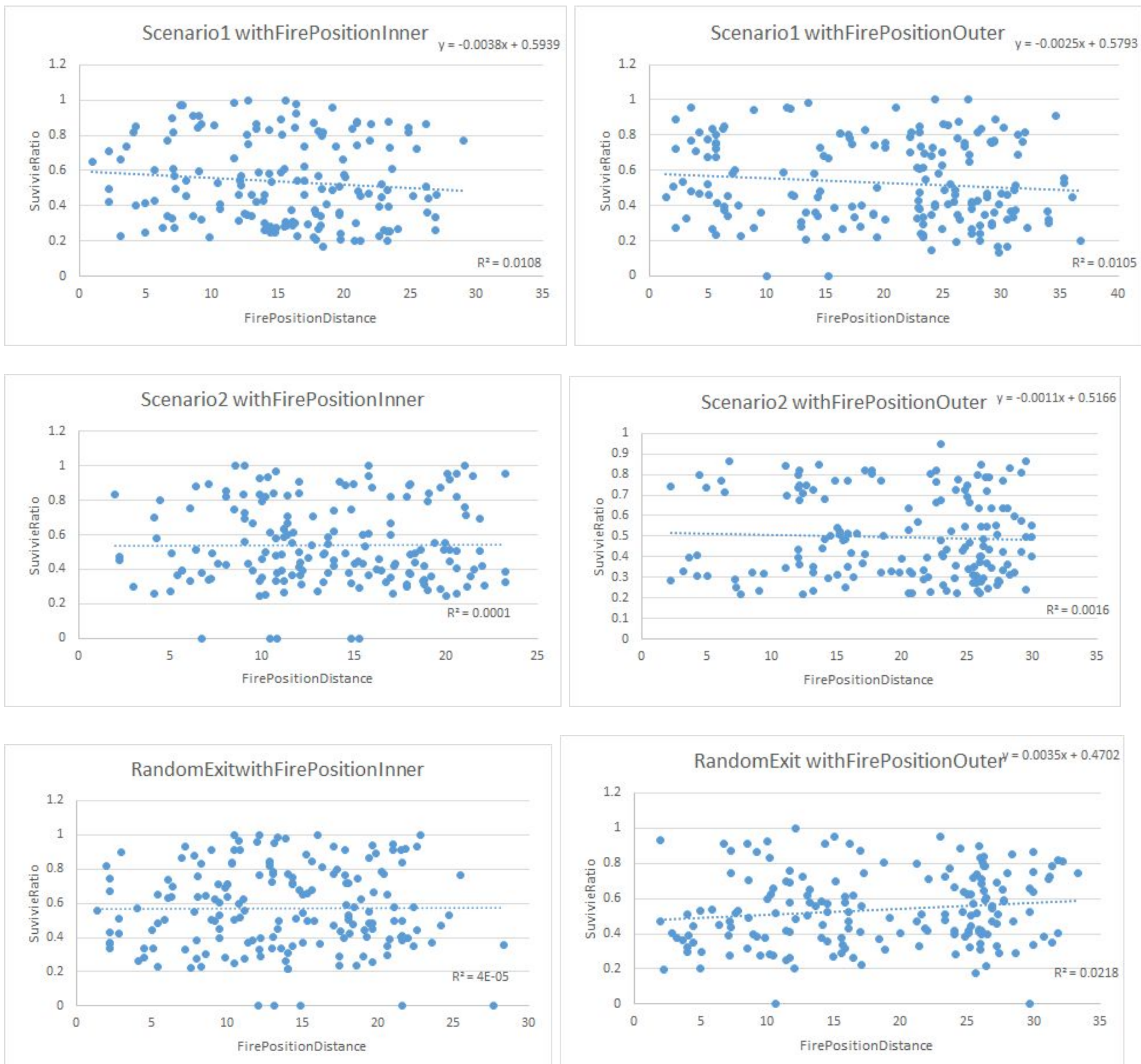
Plot 2a

To estimate the approximate optimal levels of the size of the stadium and the ventilation required on smoke spread during fire, we measured the relationship between the ratio of deaths from smoke relative to the number of turtles. The graph showed that the supremum of the graph spikes when the smoke rate is approximately 13 and thus our recommendation is keeping SmokeSpreadRate below 12 (Plot 2b).



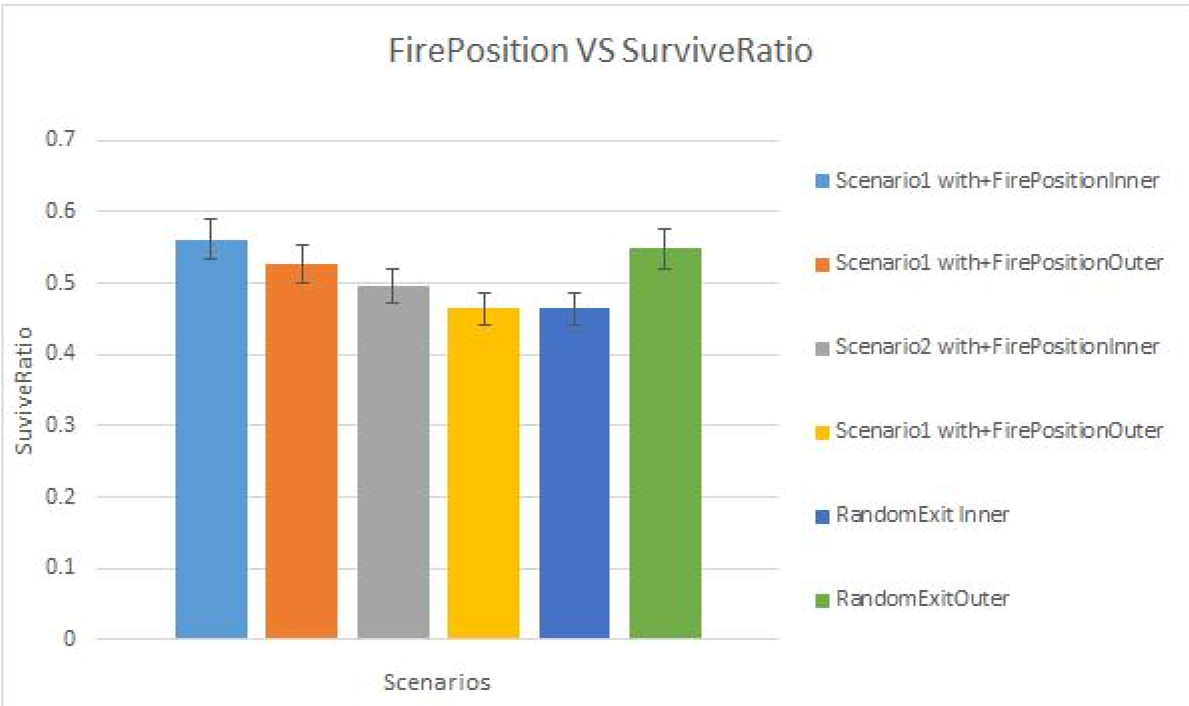
Plot 2b

We looked at the different scenarios and how the distance from the fire genesis patch to its closest inner exit varied. All scenarios showed no relationship. (Plots 3a,b,c,d,e,f)



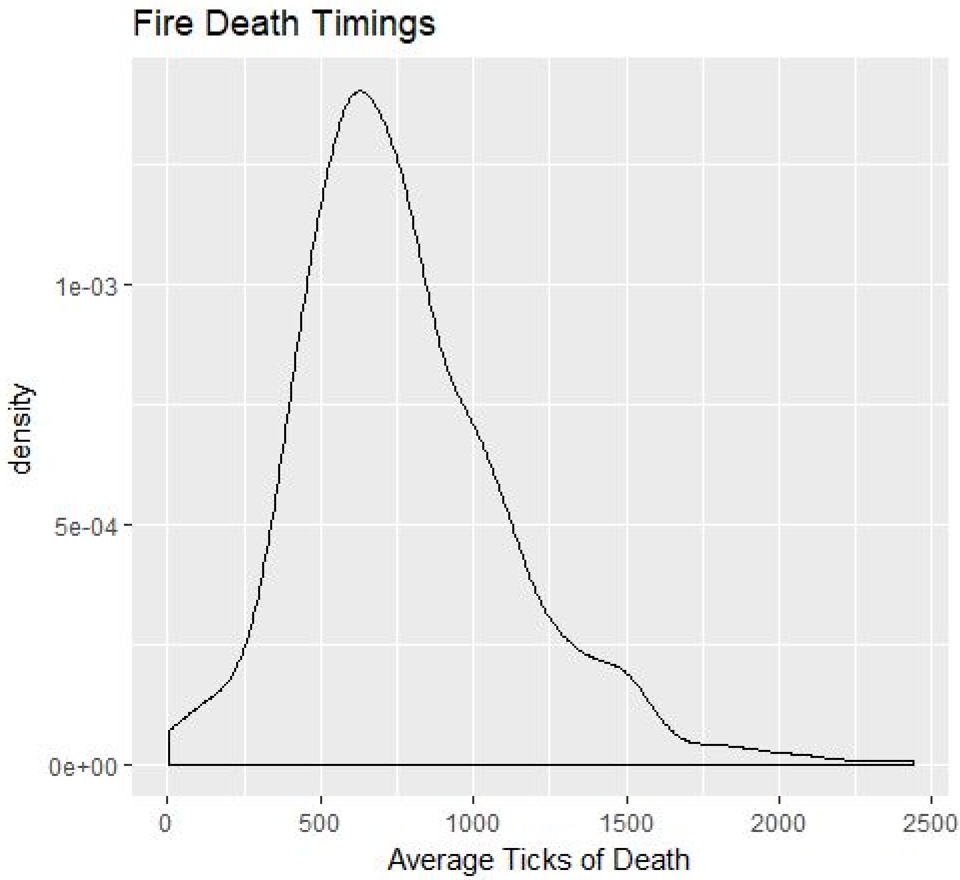
Plots 3a,b,c,d,e,f

The survival ratio was then measured for each scenario and fire starting position. Splitting the data into these respective categories reduces the error bars and gives a finer distribution of the survival rates. A significant difference in the survival ratio for Scenario 1 with an inner fire position and Scenario 2 with an outer fire position indicates that Scenario 1 is a more optimal exit design.

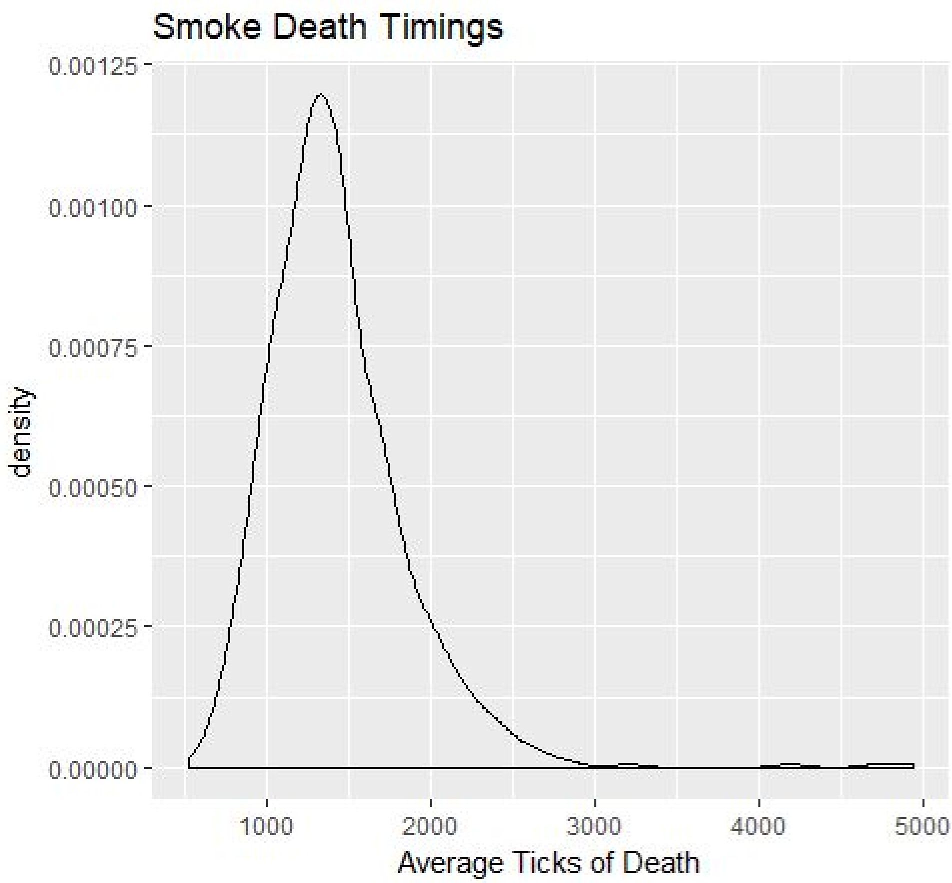


Graph 2

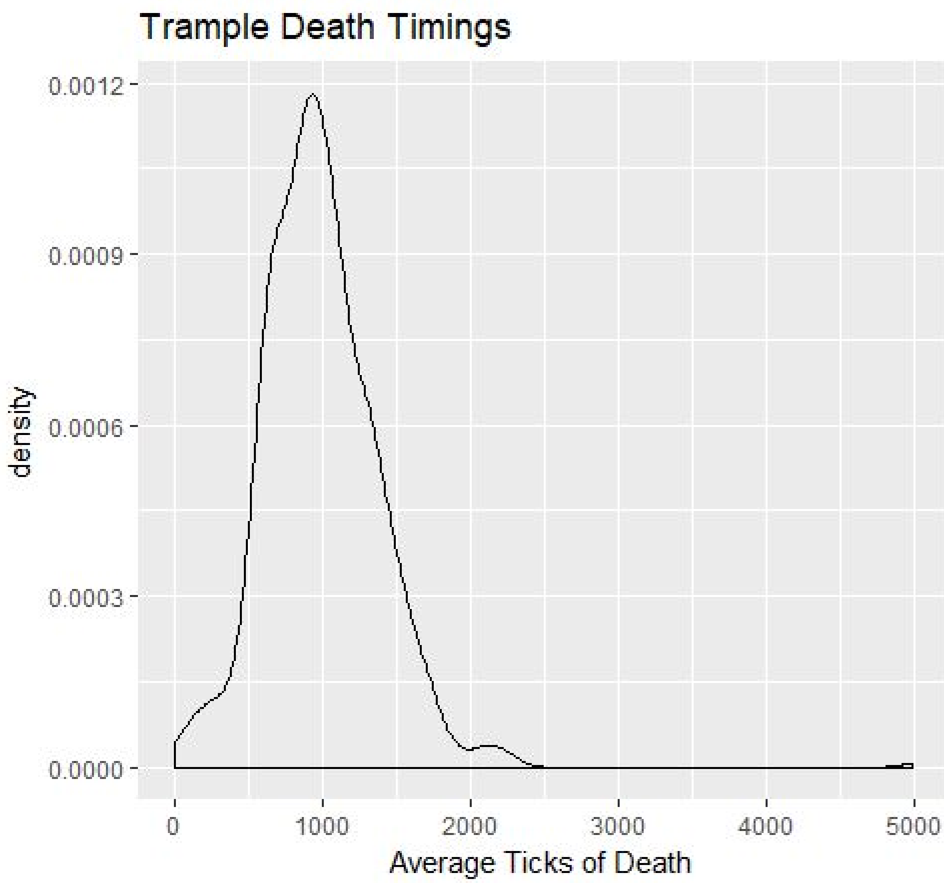
The average timings for each mode of removal for each simulation was plotted by the timing density to measure the relative order in which the modes take place to allow for evacuation drill maneuvers optimised for the expected sequence of dangers.



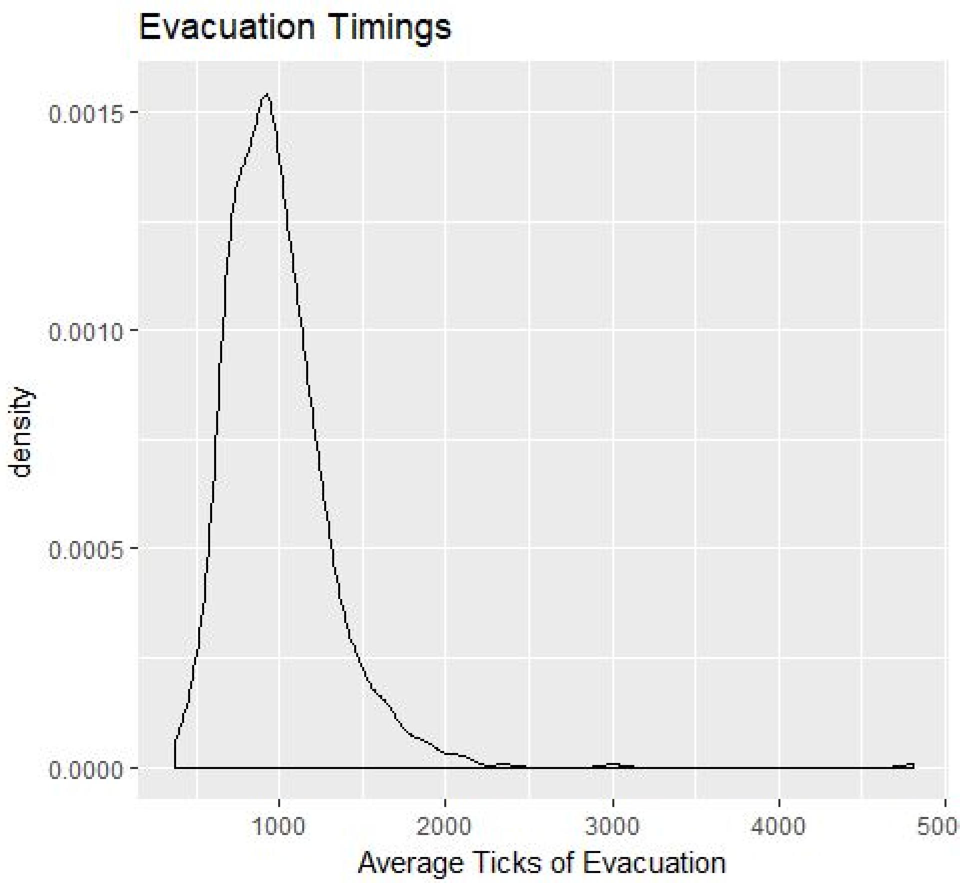
Plot 4a


Smoke Death Timings
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
density
1000
2000
3000
4000
5000
Average Ticks of Death

Plot 4b


Trample Death Timings
0.0012
0.0009
0.0006
0.0003
0.0000
density
0
1000
2000
3000
4000
5000
Average Ticks of Death

Plot 4c



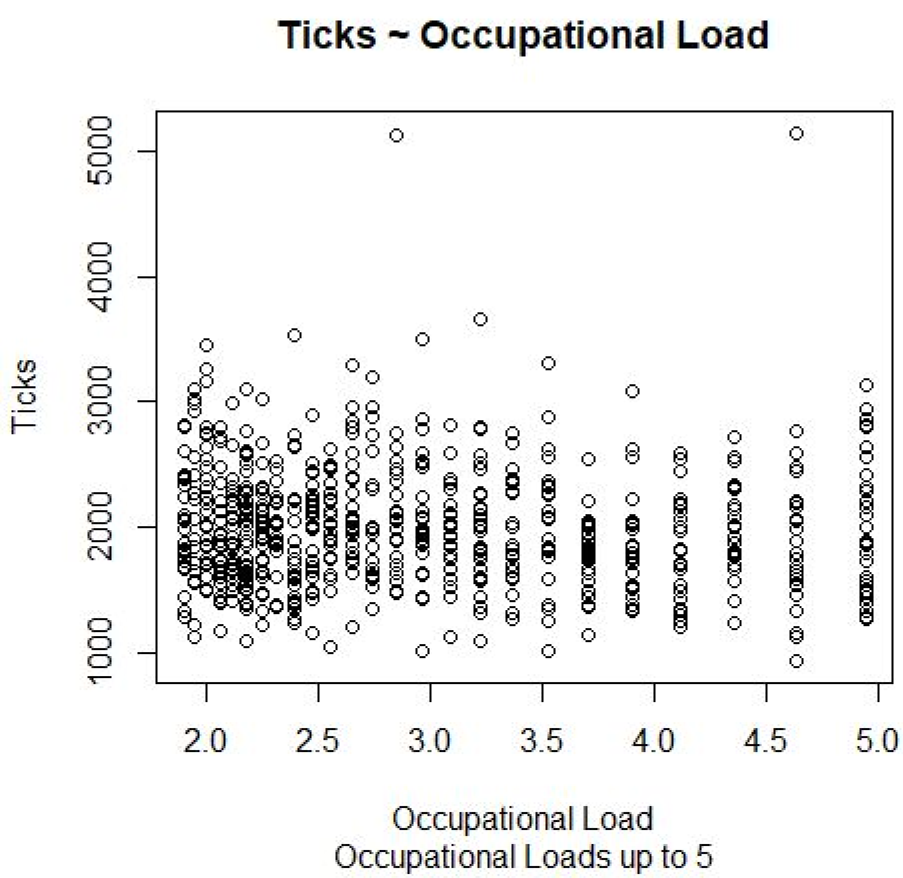
Plot 4d

The plots display that Fire deaths take place first at a mode of approximately 700, Trampling and Exiting occur at roughly the same time with a mode of approximately 1000 and Smoke approximately 1200. This suggests that a large number of turtle are trampling when the turtles are exiting as they will conglomerate trying to get through.

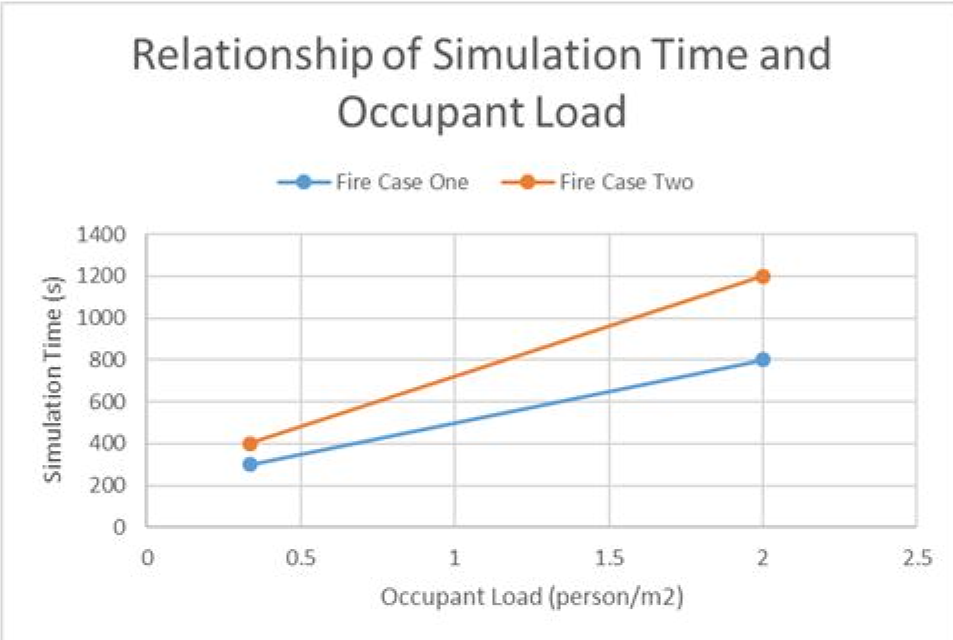
4 .Critical Assessment

i) Conformation with Literature

To verify our data, we compare our results (Plot 5a) of Tick to Occupational Load to the results from previous research by Peizhong Yang, Chao Li and Dehu Chen. (Plot 5b)



Plot 5a



Plot 5b

Source: Peizhong Yang, Chao Li, Dehu Chen[3]

The occupant load has a positive relation in the previous research however this is not reflected in our simulations. This could be due to different building layout adopted for the investigated simulation. The simulation duration was measured in seconds and the simulation duration in our simulation is measured in ticks so without knowing the mapping between them, cannot compare values.

ii) Critique

<u>Strengths</u>

The strengths of the model include somewhat qualitatively accurate behaviour of the turtles.

Weaknesses

Turtles behave uniformly, having no intrinsic variables other than those imposed by the conditions they are exposed to. In reality, people have qualities and features that will dictate nuances in behaviour.

When a set of turtles are within a density of smoke that prevents them from being able to see any nearest exit or any other turtles that can see a nearest exit, they huddle together and end up dying. This may or may not be an accurate representation of reality.

There is no mechanism for which turtles can be injured in the simulation from environmental factors such as furniture.

The mechanism of smoke and fire spread is not modelled on physics which, although seemingly accurate in many situations, has some inconsistent haviour such as smoke spreading faster in a separate region from the fire once an initial patch of smoke has been spawned. The creation of smoke patches is transmitted from both smoke and fire patches, whereas in reality the smoke would only spawn from the fire and any spreading from other smoke would reduce the density of smoke in the transmitting patch.

Ventilation was not specifically imbedded, rather it is convolved into the SmokeSpreadRate variable. This does not allow for analysis of optimal ventilation position.

Turtles do not possess a memory other than how many deaths they have witnessed which prevents turtles from being able to move towards an exit they had previously seen but have since lost (if the smoke density of their residing patch had changed).

Optimally there be a memory with an error that increased the longer the exit had remained out of vision.

If there is fire just outside the inner exit that a turtle is heading towards; they will not see it until they have left that exit resulting in many fire deaths.

Turtles cannot go back through an inner exit; even if no other exits are available the turtle will not go back through the inner exit and attempt a new route.

5. Final Code

```
;; CONTENTS
;;
;; Contents :: (Recursion)
;;
;; Extensions
;;
;; Colours Glossary
;;;;;; Patch Colours
;;;;;; Turtle Colours
;;
;;Variables
;;;;;; Globals
;;;;;; Turtles
;;;;;; Patches
;;
;; Buttons
;;;;;; Set Scenario
;;;;;;;;;; Regions :: (Patch Set) - Outer Section, Inner Section, Field
;;;;;;;;;; Walls :: (Patch Set)
;;;;;;;;;;;;;; Outer Walls - North and South, East and West
;;;;;;;;;;;;;; Inner Walls - North, South, East, West
;;;;;;;;;; Exits :: (Patch Set)
;;;;;;;;;;;;;;Inner Exits - North, South, East, West
;;;;;;;;;;;;;;Outer Exits - North and South, East and West
;;;;;;;;;; Static Patch Variables :: (Patch Variable)
;;;;;;;;;;;;;; OriginalSections
;;;;;;;;;;;;;; PPNeighbours
;;;;;;;;;; People :: (Turtle Set)
;;;;;;;;;;;;;; Field
;;;;;;;;;;;;;; Lobby
;;;;;;;;;;;;;; Field

;; EXTENSIONS
```

```
;;
;; R
extensions [r]

;; COLOURS GLOSSARY
;;;;;; Patch Colours
;;;;;;;;;; 0 = Black = Walls
;;;;;;;;;; 6 = Light Grey = Outer Region
;;;;;;;;;; 9 = White = Inner Seating
;;;;;;;;;; 67 = Light Green = Field
;;;;;;;;;; 15 = Red = Fire
;;;;;;;;;; 37 = Light Brown = Light Smoke
;;;;;;;;;; 35 = Medium Brown = Medium Smoke
;;;;;;;;;; 33 = Dark Brown = Heavy Smoke "Dark Smoke" - lol
;;;;;;Turtle Colours
;;;;;;;;;;

;; VARIABLES
;;
;; Global Variables
;;
globals [
  DeathFireCount
  DeathSmokeCount
  DeathTrampleCount
  SafeCount
  WasRandomScenario?
  WasRandomFire?
  VisionMultiplier
  DeathsThisTick
]
;;
;; Turtle Variables
;;
turtles-own [
 TTneighbours
```

```
 TTneighbourMain
 TPexit
 TPexitsRange
 TPneighbours
 TPfires
 TPfireMain
 InnerExitDuration
 OuterExitDuration
 Walls
 Vision
 WallMain
 WallSub
 Section
 TPexitNew
 TPSmokeNeighbours
 Toxicity
 Fear
 TTneighbourMultiplier
 TPfireMultiplier
 SafestPatchMultiplier
 TPexitMultiplier
 subrandom
 SafestPatch
 SafeXcor
 SafeYcor
 MinDangerLevel
 DeathsWitnessed
 Speed
 ComfortSpace
]
;;
;;Patch Variables
;;
patches-own [
 PPneighbours
 OriginalSection
```

```
 PPFireDensity
 PPLightSmokeDensity
 PPMediumSmokeDensity
 PPHeavySmokeDensity
 DangerLevel
 ClosestInnerExit
]

;; BUTTONS
;;
;; Set Scenario
;;
to SetScenario
;;;;;; Reset
;;;;;;
;;;;;;;;; Reset Patches and Ticks
  clear-all
  reset-ticks
;;;;;;;;; Reset R Temp Data
  r:__evaldirect "rm(Fire, envir = nl.env)"
  r:__evaldirect "rm(Smoke, envir = nl.env)"
  r:__evaldirect "rm(Trample, envir = nl.env)"
  r:__evaldirect "rm(EvacuatedSafely, envir = nl.env)"
  r:__evaldirect "rm(Ticks, envir = nl.env)"
  r:__evaldirect "rm(Exits, envir = nl.env)"
  r:__evaldirect "rm(FirePosition, envir = nl.env)"
  r:__evaldirect "rm(FireDistanceToExit, envir = nl.env)"
  r:__evaldirect "rm(Turtles, envir = nl.env)"
  r:__evaldirect "rm(FireSpread, envir = nl.env)"
  r:__evaldirect "rm(SmokeSpread, envir = nl.env)"
  r:__evaldirect "rm(RandomInnerExitRatio, envir = nl.env)"
  r:__evaldirect "rm(RandomOuterExitRatio, envir = nl.env)"
  r:__evaldirect "rm(TempRow)"
  r:__evaldirect "rm(DeathFrame)"
  r:gc
;;;;;;;;; Global Variable
```

```
  set DeathFireCount 0
  set DeathSmokeCount 0
  set DeathTrampleCount 0
  set SafeCount 0
  if (RandomFireSpread?) [set FireSpreadRate random-gamma 4 0.75]
  if (RandomSmokeSpread?) [set SmokeSpreadRate random-gamma (exp(pi / 1.2)) 0.75]
  if (RandomTurtles?) [set TurtleMultiplier (random 40) / 10]
  if (RandomRandomExitNum?) [set RandomInnerExitNumber (random 10) + 3]
  if (RandomRandomExitNum?) [set RandomOuterExitNumber (random 20) + 1]
;;
  set-current-plot "Death Plot"
  ifelse (Scenario = "Random Scenario") [set WasRandomScenario? TRUE set Scenario one-of ["Random Exits" "Scenario 1" "Scenario 2"]] [set WasRandomScenario? False]
  ifelse (FirePosition = "Random") [set WasRandomFire? TRUE set FirePosition one-of ["Inner" "Outer"]] [set WasRandomFire? False]
;;;;;; Set Regions :: (Patch Set)
;;;;;;;;;; Case 1
;;;;;;;;;;
;;;;;;;;;;;;;; Set Outer Section :: (Square Set)
  ask patches [
    if ((pxcor < max-pxcor) and  (pxcor > min-pxcor) and (pycor < max-pycor) and (pycor > min-pycor))
    [ set pcolor 9 ]
;;;;;;;;;;;;;; Set Inner Section :: (Square Set)
    if (pxcor < ((2 * min-pxcor + 12 * max-pxcor) / 14) and  pxcor > ((12 * min-pxcor + 2 * max-pxcor) / 14) and pycor < ((2 * min-pycor + 12 * max-pycor) / 14) and pycor > ((12 * min-pycor + 2 * max-pycor) / 14))
    [ set pcolor 6 ]
;;;;;;;;;;;;;; Set Field :: (Sqaure Set)
    if (pxcor < ((2 * min-pxcor + 5 * max-pxcor) / 7) and pxcor > ((5 * min-pxcor + 2 * max-pxcor) / 7) and pycor < ((2 * min-pycor + 5 * max-pxcor) / 7) and pycor > ((5 * min-pycor + 2 * max-pycor) / 7))
      [ set pcolor 67 ]
]
;;
;;;;;; Set Walls :: (Patch Set)
;;;;;;;;;; Case 1
;;;;;;;;;;;;;; Outer Walls :: (Line Set)
;;;;;;;;;;;;;;;;;; North and South
```

```
  ask patches with [pxcor = min-pxcor or pxcor = max-pxcor] [set pcolor 0]
;;;;;;;;;;;;;;;; East and West
  ask patches with [ pxcor = max-pxcor or  pycor = max-pycor] [set pcolor 0]
;;;;;;;;;;;; Inner Walls :: (Line Set)
  ask patches [
;;;;;;;;;;;;;;;; North Wall
    if ((pycor = round(((12 * max-pxcor + 2 * min-pxcor) / 14))) and (pxcor < ((2 * min-pycor + 12 *
max-pycor) / 14)) and (pxcor > ((12 * min-pycor + 2 * max-pycor) / 14)))
    [ set pcolor 0]
;;;;;;;;;;;;;;;; South Wall
    if ((pycor = round(((12 * min-pycor + 2 * max-pycor) / 14))) and (pxcor < ((2 * min-pxcor + 12 *
max-pxcor) / 14)) and (pxcor > ((12 * min-pxcor + 2 * max-pxcor) / 14)))
    [ set pcolor 0]
;;;;;;;;;;;;;;;; East Wall
    if ((pxcor = round(((2 * min-pxcor + 12 * max-pxcor) / 14))) and (pycor < ((2 * min-pycor + 12 *
max-pycor) / 14)) and (pycor > ((12 * min-pycor + 2 * max-pycor) / 14)))
    [ set pcolor 0]
;;;;;;;;;;;;;;;; West Wall
    if ((pxcor = round(((12 * min-pxcor + 2 * max-pxcor) / 14))) and (pycor < ((2 * min-pycor + 12 *
max-pycor) / 14)) and (pycor > ((12 * min-pycor + 2 * max-pycor) / 14) ))
    [ set pcolor 0]
  ]
;;
;;;;; Set Exits :: (Patch Set)
;;;;;;;;;;;; Random
  ask patches [
;;;;;;;;;;;;;;;; Inner Exits
    if (Scenario = "Random Exits") [
;;;;;;;;;;;;;;;;;;;; East Exits
      if ((pxcor = round(((12 * max-pxcor + 2 * min-pxcor) / 14))) and (pycor < ((2 * min-pycor + 12 *
max-pycor) / 14)) and (pycor > ((12 * min-pycor + 2 * max-pycor) / 14)) and (random 100) <
RandomInnerExitNumber)
      [ set pcolor 25]
;;;;;;;;;;;;;;;;;;;; West Exits
```

```
      if ((pxcor = round(((12 * min-pxcor + 2 * max-pxcor) / 14))) and (pycor < ((2 * min-pycor + 12 *
max-pycor) / 14)) and (pycor > ((12 * min-pycor + 2 * max-pycor) / 14)) and (random 100) <
RandomInnerExitNumber)
      [ set pcolor 25]
;;;;;;;;;;;;;;;;;;;; North Exits
      if ((pycor = round(((12 * max-pycor + 2 * min-pycor) / 14))) and (pxcor < ((2 * min-pxcor + 12 *
max-pxcor) / 14)) and (pxcor > ((12 * min-pxcor + 2 * max-pxcor) / 14)) and (random 100) <
RandomInnerExitNumber)
      [ set pcolor 25]
;;;;;;;;;;;;;;;;;;;; South Exits
      if ((pycor = round(((12 * min-pycor + 2 * max-pycor) / 14))) and (pxcor < ((2 * min-pxcor + 12 *
max-pxcor) / 14)) and (pxcor > ((12 * min-pxcor + 2 * max-pxcor) / 14)) and (random 100) <
RandomInnerExitNumber)
      [ set pcolor 25]

;;;;;;;;;;;;; Outer Exits
;;;;;;;;;;;;;;;; North and South Exits
      if (pycor = min-pycor or  pycor = max-pycor and (random 100) < RandomOuterExitNumber) [set
pcolor 45]
;;;;;;;;;;;;;;;; East and West Exits
      if (pxcor = min-pxcor or pxcor = max-pxcor and (random 100) < RandomOuterExitNumber) [set pcolor
45]
    ]
;;;;;;;;;;;;; Scenario 1
  if (Scenario = "Scenario 1") [
;;;;;;;;;;;;;;;; Inner Exits
;;;;;;;;;;;;;;;;;;;; North-East Exits
      if ((pxcor = round(((12 * max-pxcor + 2 * min-pxcor) / 14))) and (pycor = round(((12 * max-pycor + 2 *
min-pycor) / 14)))) [set pcolor 25]
      if ((pxcor = round(((12 * max-pxcor + 2 * min-pxcor) / 14))) and (pycor = round(((12 * max-pycor + 2 *
min-pycor) / 14)) - 1)) [set pcolor 25]
      if ((pxcor = round(((12 * max-pxcor + 2 * min-pxcor) / 14)) - 1) and (pycor = round(((12 * max-pycor +
2 * min-pycor) / 14)))) [set pcolor 25]
;;;;;;;;;;;;;;;;;;;; North-West Exits
      if ((pycor = round(((12 * max-pycor + 2 * min-pycor) / 14))) and (pxcor = round(((2 * max-pxcor + 12 *
min-pxcor) / 14)))) [set pcolor 25]
```

```
      if ((pycor = round(((12 * max-pycor + 2 * min-pycor) / 14))) and (pxcor = round(((2 * max-pxcor + 12 * min-pxcor) / 14)) + 1)) [set pcolor 25]
      if ((pycor = round(((12 * max-pycor + 2 * min-pycor) / 14) - 1)) and (pxcor = round(((2 * max-pxcor + 12 * min-pxcor) / 14)))) [set pcolor 25]
;;;;;;;;;;;;;;;;;;;; South-East Exits
      if ((pycor = round(((12 * min-pycor + 2 * max-pycor) / 14))) and (pxcor = round(((2 * min-pxcor + 12 * max-pxcor) / 14)))) [set pcolor 25]
      if ((pycor = round(((12 * min-pycor + 2 * max-pycor) / 14))) and (pxcor = round(((2 * min-pxcor + 12 * max-pxcor) / 14)) - 1)) [set pcolor 25]
      if ((pycor = round(((12 * min-pycor + 2 * max-pycor) / 14) + 1)) and (pxcor = round(((2 * min-pxcor + 12 * max-pxcor) / 14)))) [set pcolor 25]
;;;;;;;;;;;;;;;;;;;; South-West Exits
      if ((pxcor = round(((12 * min-pxcor + 2 * max-pxcor) / 14))) and (pycor = round(((12 * min-pycor + 2 * max-pycor) / 14)))) [set pcolor 25]
      if ((pxcor = round(((12 * min-pxcor + 2 * max-pxcor) / 14))) and (pycor = round(((12 * min-pycor + 2 * max-pycor) / 14)) + 1)) [set pcolor 25]
      if ((pxcor = round(((12 * min-pxcor + 2 * max-pxcor) / 14)) + 1) and (pycor = round(((12 * min-pycor + 2 * max-pycor) / 14)))) [set pcolor 25]
;;;;;;;;;;;;;;;; Outer Exits
;;;;;;;;;;;;;;;;;;;; North-East Exits
      if (pxcor = max-pxcor and pycor = max-pycor) [set pcolor 45]
      if (pxcor = max-pxcor - 1 and pycor = max-pycor) [set pcolor 45]
      if (pxcor = max-pxcor and pycor = max-pycor - 1) [set pcolor 45]
;;;;;;;;;;;;;;;;;;;; North-West Exits
      if (pycor = max-pycor and pxcor = min-pxcor) [set pcolor 45]
      if (pycor = max-pycor and pxcor = min-pxcor + 1) [set pcolor 45]
      if (pycor = max-pycor - 1 and pxcor = min-pxcor) [set pcolor 45]
;;;;;;;;;;;;;;;;;;;; South-East Exits
      if (pycor = min-pycor and pxcor = max-pxcor) [set pcolor 45]
      if (pycor = min-pycor and pxcor = max-pxcor - 1) [set pcolor 45]
      if (pycor = min-pycor + 1 and pxcor = max-pxcor) [set pcolor 45]
;;;;;;;;;;;;;;;;;;;; South-West Exits
      if (pxcor = min-pxcor and pycor = min-pycor) [set pcolor 45]
      if (pxcor = min-pxcor + 1 and pycor = min-pycor) [set pcolor 45]
      if (pxcor = min-pxcor and pycor = min-pycor + 1) [set pcolor 45]
    ]
```

```
;;;;;;;;; Scenario 2
  if (Scenario = "Scenario 2") [
;;;;;;;;;;;;; Inner Exits
;;;;;;;;;;;;;;;;; North Exits
      if ((pycor = round(((12 * max-pycor + 2 * min-pycor) / 14))) and pxcor = 0) [set pcolor 25]
;;;;;;;;;;;;;;;;; South Exits
      if ((pycor = round(((12 * min-pycor + 2 * max-pycor) / 14))) and pxcor = 0) [set pcolor 25]
;;;;;;;;;;;;;;;;; East Exits
      if ((pxcor = round(((12 * max-pxcor + 2 * min-pxcor) / 14))) and pycor = 0) [set pcolor 25]
;;;;;;;;;;;;;;;;; West Exits
      if ((pxcor = round(((12 * min-pxcor + 2 * max-pxcor) / 14))) and pycor = 0) [set pcolor 25]
;;;;;;;;;;;;; Outer Exits
;;;;;;;;;;;;;;;;; North Exits
      if (pycor = max-pycor and pxcor = round (0.5 * min-pxcor)) [set pcolor 45]
      if (pycor = max-pycor and pxcor = round ( 0.5 * max-pxcor)) [set pcolor 45]
;;;;;;;;;;;;;;;;; South Exits
      if (pycor = min-pycor and pxcor = round (0.5 * min-pxcor)) [set pcolor 45]
      if (pycor = min-pycor and pxcor = round ( 0.5 * max-pxcor)) [set pcolor 45]
;;;;;;;;;;;;;;;;; East Exits
      if (pxcor = max-pxcor and pycor = round (0.5 * min-pycor)) [set pcolor 45]
      if (pxcor = max-pxcor and pycor = round ( 0.5 * max-pycor)) [set pcolor 45]
;;;;;;;;;;;;;;;;; West Exits
      if (pxcor = min-pxcor and pycor = round (0.5 * min-pycor)) [set pcolor 45]
      if (pxcor = min-pxcor and pycor = round ( 0.5 * max-pycor)) [set pcolor 45]
    ]
  ]
;;
;;;;; Patch Variables :: (Patch Variable)
;;;;;
;;;;;;;; Section :: (Integer)
  ask patches [
;;;;;;;;;;;;; Field)
    if (pcolor = 67) [set OriginalSection 0]
;;;;;;;;;;;;; Stands
    if (pcolor = 6) [set OriginalSection 0]
;;;;;;;;;;;;; Lobby
```

```
    if (pcolor = 9) [set OriginalSection 1]
;;;;;;;;;;;;; Walls
    if (pcolor = 0) [set OriginalSection 4]
;;;;;;;;;;;;; Inner Exits
    if (pcolor = 25) [set OriginalSection 2]
;;;;;;;;;;;;; Outer Exits
    if (pcolor = 45) [set OriginalSection 3]
  ]
;;;;;;;;; PPneighbours :: (Patch Set)
  ask patches [
    set PPneighbours (other patches) in-radius 1
  ]
;;
;;;;; Set Fire
;;;;;
;;;;;;;;; Inner Fire
  if (FirePosition = "Inner") [
    ask one-of patches with [OriginalSection = 0 or OriginalSection = 2] [
      set ClosestInnerExit min-one-of patches with [OriginalSection = 2] [distance self]
      set pcolor 15
      r:put "FireDistanceToExit" distance ClosestInnerExit
    ]
  ]
;;;;;;;;; Outer Fire
  if (FirePosition = "Outer") [
    ask one-of patches with [OriginalSection = 1 or OriginalSection = 3] [
      set ClosestInnerExit min-one-of patches with [OriginalSection = 2] [distance self]
      set pcolor 15
      r:put "FireDistanceToExit" distance ClosestInnerExit
    ]
  ]
;;
;;;;; Set People :: (Turtle Set)
;;;;;
;;;;;;;;; 10:Field, 80:Stands, 20:Lobby
;;;;;;;;;;;;; Stands
```

```
  create-turtles round(TurtleMultiplier * 80) [set Fear 0 set Toxicity 0 set Vision 32 set shape "person"
move-to one-of (patches with [pcolor = 6])]
;;;;;;;;;;;;; Lobby
  create-turtles round(TurtleMultiplier * 20) [set Fear 0 set Toxicity 0 set Vision 32 set shape "person"
move-to one-of (patches with [pcolor = 9])]
;;;;;;;;;;;;; Field
  create-turtles round(TurtleMultiplier * 10) [set Fear 0 set Toxicity 0 set Vision 32 set shape "person"
move-to one-of (patches with [pcolor = 67])]
;;;;;; Set TTneighbours
  ask turtles [
    if ([Section] of self = 0) [set TTneighbours (other turtles with [Section = 0]) in-radius Vision]
    if ([Section] of self = 1) [set TTneighbours (other turtles with [Section = 1]) in-radius Vision]
    if ([Section] of self = 2) [set TTneighbours (other turtles with [Section = 2]) in-radius Vision]
    if ([Section] of self = 3) [set TTneighbours (other turtles with [Section = 3]) in-radius Vision]
    if ([Section] of self = 4) [set TTneighbours (other turtles with [Section = 4]) in-radius Vision]
  ]
;;;;;; Set TPneighbours
  ask turtles [
    if ([Section] of self = 0) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
0])]
    if ([Section] of self = 1) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
1])]
    if ([Section] of self = 2) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
2])]
    if ([Section] of self = 3) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
3])]
    if ([Section] of self = 4) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
4])]
  ]

  r:put "Exits" Scenario
  r:put "FirePosition" FirePosition
  r:put "Turtles" count Turtles
  r:__evaldirect "DeathFrame <- as.data.frame(matrix(ncol = 4, nrow = nl.env$Turtles))"
  ;;r:__evaldirect "TempRow <- c(nl.env$Exits, nl.env$FirePosition, nl.env$FireDistanceToExit,
nl.env$Turtles)"
```

```
  if (WasRandomScenario?) [set Scenario "Random Scenario"]
  if (WasRandomFire?) [set FirePosition "Random"]
end
;;
;;Go
;;
to go
;;;;;;
  set DeathsThisTick 0
;;;;;; Stop
  if (count turtles = 0) [
   r:put "Fire" DeathFireCount
   r:put "Smoke" DeathSmokeCount
   r:put "Trample" DeathTrampleCount
   r:put "EvacuatedSafely" SafeCount
   r:put "Ticks" ticks
   r:put "FireSpread" FireSpreadRate
   r:put "SmokeSpread" SmokeSpreadRate
   r:put "RandomInnerExitRatio" RandomInnerExitNumber / 100
   r:put "RandomOuterExitRatio" RandomOuterExitNumber / 100
   ;;r:__evaldirect "TempRow <- c(TempRow, mean(DeathFrame[[1]], na.rm = TRUE),
mean(DeathFrame[[2]], na.rm = TRUE), mean(DeathFrame[[3]], na.rm = TRUE), mean(DeathFrame[[4]],
na.rm = TRUE))"
   ;;r:__evaldirect "TempRow <- c(nl.env$Fire, nl.env$Smoke, nl.env$Trample, nl.env$EvacuatedSafely,
nl.env$Ticks, TempRow)"
   ;;r:__evaldirect "TempRow <- c(nl.env$Fire, nl.env$Smoke, nl.env$Trample, nl.env$EvacuatedSafely,
nl.env$Ticks,nl.env$Exits, nl.env$FirePosition, nl.env$FireDistanceToExit, nl.env$Turtles,
mean(DeathFrame[[1]], na.rm = TRUE), mean(DeathFrame[[2]], na.rm = TRUE), mean(DeathFrame[[3]],
na.rm = TRUE), mean(DeathFrame[[4]], na.rm = TRUE))"
   r:__evaldirect "TempRow <- c(as.integer(nl.env$Fire))"
   r:__evaldirect "TempRow <- c(TempRow, as.integer(nl.env$Smoke))"
   r:__evaldirect "TempRow <- c(TempRow, as.integer(nl.env$Trample))"
   r:__evaldirect "TempRow <- c(TempRow, as.integer(nl.env$EvacuatedSafely))"
   r:__evaldirect "TempRow <- c(TempRow, as.integer(nl.env$Ticks))"
   r:__evaldirect "TempRow <- c(TempRow, toString(nl.env$Exits))"
   r:__evaldirect "TempRow <- c(TempRow, toString(nl.env$FirePosition))"
```

```
r:__evaldirect "TempRow <- c(TempRow, as.numeric(nl.env$FireDistanceToExit))"
r:__evaldirect "TempRow <- c(TempRow, as.integer(nl.env$Turtles))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(mean(DeathFrame[[1]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(mean(DeathFrame[[2]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(mean(DeathFrame[[3]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(mean(DeathFrame[[4]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(sd(DeathFrame[[1]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(sd(DeathFrame[[2]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(sd(DeathFrame[[3]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(sd(DeathFrame[[4]], na.rm = TRUE)))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(nl.env$FireSpread))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(nl.env$SmokeSpread))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(nl.env$RandomInnerExitRatio))"
r:__evaldirect "TempRow <- c(TempRow, as.numeric(nl.env$RandomOuterExitRatio))"
r:__evaldirect "DataFrame <- rbind.data.frame(DataFrame, TempRow, stringsAsFactors = FALSE)"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[1] <- 'Fire'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[2] <- 'Smoke'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[3] <- 'Trample'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[4] <- 'Evacuated'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[5] <- 'Ticks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[6] <- 'Exits'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[7] <- 'FirePosition'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[8] <- 'FireDistanceToExit'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[9] <- 'Turtles'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[10] <- 'FireDeathAverageTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[11] <- 'SmokeDeathAverageTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[12] <- 'TrampleDeathAverageTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[13] <- 'EvacuatedAverageTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[14] <- 'FireDeathSDTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[15] <- 'SmokeDeathSDTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[16] <- 'TrampleDeathSDTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[17] <- 'EvacuatedSDTicks'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[18] <- 'FireSpreadRate'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[19] <- 'SmokeSpreadRate'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[20] <- 'RandomInnerExitRatio'}"
r:__evaldirect "if (nrow(DataFrame) == 1){colnames(DataFrame)[21] <- 'RandomOuterExitRatio'}"
```

```
   ifelse ReRun? [SetScenario] [Stop]
  ]
  set VisionMultiplier 2 / (1 + exp(count turtles / 220))
;;;;;; Patches
;;;;;;
  ask patches [
;;;;;;;;;; Patch Variables :: (Patch Variables)
;;;;;;;;;;
;;;;;;;;;;;;;; Counters :: (Integer)
;;;;;;;;;;;;;;;;;; Fire Density Counter
    set PPFireDensity count PPneighbours with [pcolor = 15]
;;;;;;;;;;;;;;;;;; Smoke Density Counters
;;;;;;;;;;;;;;;;;;;;;; Light
    set PPLightSmokeDensity (count PPneighbours with [pcolor = 37 or pcolor = 15])
;;;;;;;;;;;;;;;;;;;;;; Medium
    set PPMediumSmokeDensity (count PPneighbours with [pcolor = 35 or pcolor = 15]) * (count
PPneighbours with [pcolor = 37] + 1) / 4.669201 ;;Feigenbaum Constant
;;;;;;;;;;;;;;;;;;;;;; Heavy
    set PPHeavySmokeDensity (count PPneighbours with [pcolor = 33 or pcolor = 15]) * (count
PPneighbours with [pcolor = 35] + 1) / (pi ^ 2)
;;;;;;;;;;;;;; Danger Level :: (Integer)
;;;;;;;;;;;;;;;;;; Exits
    if (pcolor = 25 or pcolor = 45) [set DangerLevel 0]
;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;; Clear Space
    if (pcolor = 6 or pcolor = 67 or pcolor = 9) [set DangerLevel 1]
;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;; Smoke
;;;;;;;;;;;;;;;;;;;;;; Light
    if (pcolor = 37) [set DangerLevel 2]
;;;;;;;;;;;;;;;;;;;;;; Medium
    if (pcolor = 35) [set DangerLevel 3]
;;;;;;;;;;;;;;;;;;;;;; Heavy
    if (pcolor = 33) [set DangerLevel 4]
;;;;;;;;;;;;;;;;;; Walls and Fire
    if (pcolor = 0 or pcolor = 15) [set DangerLevel 5]
```

```
;;;;;;;;;;;;;
;;;;;;;;;;;;; Patch Changes
;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;; Spread Fire
   if (pcolor != 0 and (random 1000)  < FireSpreadRate * PPFireDensity) [set pcolor 15]
;;;;;;;;;;;;;;;;; Spread Smoke
;;;;;;;;;;;;;;;;;;;; Light Smoke
   if (pcolor = 6 or pcolor = 9 or pcolor = 25 or pcolor = 45 or pcolor = 67 and (random 1000)  <
SmokeSpreadRate * PPLightSmokeDensity) [set pcolor 37]
;;;;;;;;;;;;;;;;;;;; Medium Smoke
   if (pcolor = 6 or pcolor = 9 or pcolor = 25 or pcolor = 45 or pcolor = 67 or pcolor = 37  and (random
1000)  < SmokeSpreadRate * PPMediumSmokeDensity) [set pcolor 35]
;;;;;;;;;;;;;;;;;;;; Heavy Smoke
   if (pcolor = 6 or pcolor = 9 or pcolor = 25 or pcolor = 45 or pcolor = 67 or pcolor = 37 or pcolor = 35 and
(random 1000)  < SmokeSpreadRate * PPHeavySmokeDensity) [set pcolor 33]
  ]
;;
;;;;; Turtles
;;;;;
          ask turtles [
;;;;;;;;; Set Patch-Here Related Variables
;;;;;;;;;;;;; Set Section
   if ([OriginalSection] of patch-here = 0) [set Section 0]
   if ([OriginalSection] of patch-here = 1) [set Section 1]
   if ([OriginalSection] of patch-here = 2) [set Section 2]
   if ([OriginalSection] of patch-here = 3) [set Section 3]
   if ([OriginalSection] of patch-here = 4) [set Section 4]
;;;;;;;;;;;;; No Smoke
   if ([pcolor] of patch-here = 6 or [pcolor] of patch-here = 67 or [pcolor] of patch-here = 9) [set Vision 25 *
VisionMultiplier]
;;;;;;;;;;;;; Light Smoke
   if ([pcolor] of patch-here = 37) [Set Vision 20 * VisionMultiplier set Toxicity Toxicity + 0.261497] ;;
Meissel-Mertenz Constant
;;;;;;;;;;;;; Medium Smoke
   if ([pcolor] of patch-here = 35) [Set Vision 15 * VisionMultiplier set Toxicity Toxicity + 0.577215] ;;
Euler-Mascheroni Constant
```

```
;;;;;;;;;;;;; Heavy Smoke
    if ([pcolor] of patch-here = 33) [Set Vision 10 * VisionMultiplier set Toxicity Toxicity + 1.353077] ;;
Conways Constant
;;;;;;;;; Kill Turtles
;;;;;;;;;;;;; Death By Fire
    if ([pcolor] of patch-here = 15) [set DeathFireCount (DeathFireCount + 1) ask TTneighbours [set
DeathsWitnessed DeathsWitnessed + 1] set DeathsThisTick DeathsThisTick + 1 die]
;;;;;;;;;;;;; Death By Trample
    if (any? TTneighbours and (count TTneighbours in-radius 0.567143 >= 3)) [set DeathTrampleCount
DeathTrampleCount + 1 ask TTneighbours [set DeathsWitnessed DeathsWitnessed + 1] set
DeathsThisTick DeathsThisTick + 1 die] ;; Omega Constant
;;;;;;;;;;;;; Death By Asphyxiation
    if ([Toxicity] of self > pi * 137) [set DeathSmokeCount DeathSmokeCount + 1 ask TTneighbours [set
DeathsWitnessed DeathsWitnessed + 1] set DeathsThisTick DeathsThisTick + 1 die]
;;;;;;;;; Set Nearby Patch Variables :: (Patch Set)
    if ([Section] of self = 0) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
0])]
    if ([Section] of self = 1) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
1])]
    if ([Section] of self = 2) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
2])]
    if ([Section] of self = 3) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
3])]
    if ([Section] of self = 4) [set TPneighbours (patches in-radius ([Vision] of self) with [OriginalSection =
4])]
;;;;;;;;;; Set Nearby Turtle Variables :: (Turtle Set)
    if ([Section] of self = 0) [set TTneighbours (other turtles with [Section = 0]) in-radius Vision]
    if ([Section] of self = 1) [set TTneighbours (other turtles with [Section = 1]) in-radius Vision]
    if ([Section] of self = 2) [set TTneighbours (other turtles with [Section = 2]) in-radius Vision]
    if ([Section] of self = 3) [set TTneighbours (other turtles with [Section = 3]) in-radius Vision]
    if (any? TTneighbours) [set TTneighbourMain min-one-of TTneighbours [distance myself]]
;;;;;;;;; Set Nearby Fires
    if ([Section] of self = 0) [set TPfires TPneighbours with [OriginalSection = 0 and pcolor = 15] in-radius
([Vision] of self)]
    if ([Section] of self = 1) [set TPfires TPneighbours with [OriginalSection = 1 and pcolor = 15] in-radius
([Vision] of self)]
```

```
    if ([Section] of self = 2) [set TPfires TPneighbours with [OriginalSection = 2 and pcolor = 15] in-radius
([Vision] of self)]
    if ([Section] of self = 3) [set TPfires TPneighbours with [OriginalSection = 3 and pcolor = 15] in-radius
([Vision] of self)]
    if ([Section] of self = 4) [set TPfires TPneighbours with [OriginalSection = 4 and pcolor = 15] in-radius
([Vision] of self)]
    if (any? TPfires) [set TPfireMain min-one-of TPfires [distance self]]
;;;;;;;;; Set Nearby Walls
    set Walls (patches) in-radius 0.75 with [pcolor = 0]
    if (any? Walls) [set WallMain min-one-of Walls [distance self]]
;;;;;;;;; Set Nearby Smoke
    set TPSmokeNeighbours TPneighbours with ([pcolor = 33 or pcolor = 35 or pcolor = 37 or pcolor = 15])
;;;;;;;;; Set Nearby Exits
    if ([Section] of self = 0) [set TPexitsRange (patches in-radius Vision) with [OriginalSection = 2 and
pcolor != 15]]
    if ([Section] of self = 1) [set TPexitsRange (patches in-radius Vision) with [OriginalSection = 3 and
pcolor != 15]]
    if (any? TPexitsRange) [set TPexit min-one-of TPexitsRange [distance myself]]
;;;;;;;;; Set Other Variables
;;;;;;;;;;;;; Fear
    if (any? TPneighbours with [pcolor = 33 or pcolor = 35 or pcolor = 37] and any? TPfires) [set Fear (137
+ 2 * Toxicity + ((e ^ pi) * ln (DeathsWitnessed + 1)) + 10 * (count TTneighbours in-radius 1)) + (1000 /
(0.01 + distance [self] of TPfireMain))]
    if (any? TPneighbours with [pcolor = 33 or pcolor = 35 or pcolor = 37] and not any? TPfires) [set Fear
(137 + 2 * Toxicity + ((e ^ pi) * ln (DeathsWitnessed + 1)) + 10 * (count TTneighbours in-radius 1))]
;;;;;;;;;;;;; Speed
    set Speed exp (-1 * pi) * (1 / (1 + (log (Toxicity + 1) (e ^ pi))))
    ;;set speed (30000 - (Toxicity ^ 2)) / 1000000
;;;;;;;;;;;;; ComfortSpace
    set ComfortSpace exp (-0.001 * Fear)

    if ([Section] of self = 2) [set TPexitNew min-one-of (patches with [OriginalSection = 1]) [distance
myself]]
    ;;If turtle is in an exit, sets the patch it will exit towards
```

```
;;CODE IN WORKS
;;set MinDangerLevel min [DangerLevel] of TPneighbours
;;set SafestPatch TPneighbours with [DangerLevel = [MinDangerLevel] of myself]
;;Sets the safest neighbouring patch
set SafestPatchMultiplier 0
;;set SafeXcor sum ([pxcor] of SafestPatch / distance SafestPatch)


if (any? TPfires) [set TPfireMultiplier ((1 / (distance [self] of TPfireMain + 0.000001)) + exp (-0.567 * distance [self] of TPfireMain + pi))]
if (any? TTneighbours and distance [self] of TTneighbourMain > ComfortSpace) [set TTneighbourMultiplier (0.5) * (distance [myself] of TTneighbourMain) * (exp ((1 / Vision) * (distance [myself] of TTneighbourMain)))]
if (any? TTneighbours and distance [self] of TTneighbourMain <= ComfortSpace) [set TTneighbourMultiplier 1 / (distance [myself] of TTneighbourMain + 0.0001)]
;;set TTneighbourMultiplier 1
;;set TTneighbourMultiplier 0
if (any? TPexitsRange) [set TPexitMultiplier (3 / (distance [self] of TPexit + 0.001)) * (exp (exp (-1 * ((distance [self] of TPexit) ^ (1))) + 1))]

      ;;SETTING COLOURS
if (not any? TPneighbours with [pcolor = 33 or pcolor = 35 or pcolor = 37 or pcolor = 15] and not any? Walls) [set color 95 if ([Fear] of self < 100) [set Fear 100]]
      ;;if there is no smoke
if (any? TPSmokeNeighbours and any? TPexitsRange and any? TTneighbours and not any? Walls) [set color 115]
      ;; if there is smoke and exits in range turtles in range
      if (any? TPSmokeNeighbours and any? TPexitsRange and not any? TTneighbours and not any? Walls) [set color 65]
      ;; if there is smoke and exits but no neighbours
      if (any? TPSmokeNeighbours and not any? TPexitsRange and any? TTneighbours and not any? Walls) [set color 85]
      ;;if there is smoke and neighbours but no exits
      if (any? TPSmokeNeighbours and any? TPFires and not any? TPexitsRange and not any? TTneighbours and not any? Walls) [set color 125]
      ;;if there is smoke with no neighbours and no exits
```

```
    ;;ACTIONING
    if ([Section] of self = 2) [act-InnerExit]
    if ([Section] of self = 3) [act-OuterExit]
    ;;If turtle is in an exit
    if any? Walls [set color 12]
    ;;if ( [pcolor] of myself = 45) [act-OuterExit]
    if (color = 95 and [Section] of self != 2 and [Section] of self != 3) [act-Calm]
    if (color = 85 and [Section] of self != 2 and [Section] of self != 3) [act-Flock]
    if (color = 65 and [Section] of self != 2 and [Section] of self != 3) [act-Escape]
    if (color = 115 and [Section] of self != 2 and [Section] of self != 3) [act-Hybrid]
    if (color = 125 and [Section] of self != 2 and [Section] of self != 3) [act-RunFire]
    if (color = 12) [act-Wall]
                    ;; colours 95,85,65,115,125 are blue, teal, green, purple and fuschia respectively
          ]
  if (DeathsThisTick > 0) [
    r:put "DeathTickCount" ticks
    r:put "TempFireCount" DeathFireCount
    r:put "TempSmokeCount" DeathSmokeCount
    r:put "TempTrampleCount" DeathTrampleCount
    r:put "TempSafeCount" SafeCount
    r:__evaldirect "if(match(NA,DeathFrame[[1]]) <=
nl.env$TempFireCount){DeathFrame[[1]][match(NA,DeathFrame[[1]]):nl.env$TempFireCount] <-
nl.env$DeathTickCount}"
    r:__evaldirect "if(match(NA,DeathFrame[[2]]) <=
nl.env$TempSmokeCount){DeathFrame[[2]][match(NA,DeathFrame[[2]]):nl.env$TempSmokeCount] <-
nl.env$DeathTickCount}"
    r:__evaldirect "if(match(NA,DeathFrame[[3]]) <=
nl.env$TempTrampleCount){DeathFrame[[3]][match(NA,DeathFrame[[3]]):nl.env$TempTrampleCount] <-
nl.env$DeathTickCount}"
    r:__evaldirect "if(match(NA,DeathFrame[[4]]) <=
nl.env$TempSafeCount){DeathFrame[[4]][match(NA,DeathFrame[[4]]):nl.env$TempSafeCount] <-
nl.env$DeathTickCount}"
    r:__evaldirect "rm(DeathTickCount, TempFireCount, TempSmokeCount, TempTrampleCount,
TempSafeCount, envir = nl.env)"
    r:gc
```

```
  ]
        tick
end

to act-Calm
        left random 180
        right random 180
  if (any? TTneighbours and distance [self] of TTneighbourMain <= ComfortSpace)[
    facexy (2 * [xcor] of self - [xcor] of TTneighbourMain)
    (2 * [ycor] of self - [ycor] of TTneighbourMain)
  ]
        fd Speed
end

to act-Flock
  if (is-turtle? TTneighbourMain)[
    if (any? TPfires and distance [self] of TTneighbourMain > ComfortSpace)[
        facexy  ((TTneighbourMultiplier) * [xcor] of TTneighbourMain + [heading] of TTneighbourMain +
(TPFireMultiplier) * ([xcor] of self - [pxcor] of TPfireMain) + [xcor] of self ) / (TPFireMultiplier +
TTneighbourMultiplier)
         ((TTneighbourMultiplier) * [ycor] of TTneighbourMain + [heading] of TTneighbourMain +
(TPfireMultiplier) * ([ycor] of self - [pycor] of TPfireMain) + [ycor] of self) / (TPFireMultiplier +
TTneighbourMultiplier)
        fd Speed
  ]
    if (not any? TPfires and distance [self] of TTneighbourMain > ComfortSpace)[
        facexy  (((TTneighbourMultiplier) * [xcor] of TTneighbourMain + 2 * (random max-pxcor) - 16) / (1
+ TTneighbourMultiplier))
         ((TTneighbourMultiplier) * [ycor] of TTneighbourMain + 2 * (random max-pycor) - 16) / (1 +
TTneighbourMultiplier)
        fd Speed
  ]
      if (any? TPfires and distance [self] of TTneighbourMain <= ComfortSpace)[
        facexy  ((TTneighbourMultiplier) * ([xcor] of self - [xcor] of TTneighbourMain) + [xcor] of self + 2 *
(random max-pxcor) - 16 + (TPFireMultiplier) * ([xcor] of self - [pxcor] of TPfireMain) + [xcor] of self) / (3)
```

```
            ((TTneighbourMultiplier) * ([ycor] of self - [ycor] of TTneighbourMain) + [ycor] of self + 2 *
(random max-pxcor) - 16 + (TPfireMultiplier) * ([ycor] of self - [pycor] of TPfireMain) + [xcor] of self) / (3)
            fd Speed
  ]
    if (not any? TPfires and distance [self] of TTneighbourMain <= ComfortSpace)[
            facexy  ((TTneighbourMultiplier) * ([xcor] of self - [xcor] of TTneighbourMain) + [ycor] of self) / (1)
             ((TTneighbourMultiplier) * ([ycor] of self - [ycor] of TTneighbourMain) + [ycor] of self) / (1)
            fd Speed
  ]
  ]
end

to act-Escape
            facexy  ([pxcor] of TPexit)
                        ([pycor] of TPexit)
            fd Speed
end


to act-Hybrid
  if (is-turtle? TTneighbourMain)[
    if (not any? TPfires and distance [self] of TTneighbourMain > ComfortSpace)[
            facexy  (TPexitMultiplier * [pxcor] of TPexit + TTneighbourMultiplier * [xcor] of TTneighbourMain) /
(TTneighbourMultiplier + TPexitMultiplier)
                        ((TPexitMultiplier) * [pycor] of TPexit + TTneighbourMultiplier * [ycor] of
TTneighbourMain) / (TTneighbourMultiplier + TPexitMultiplier)
  ]
    if (any? TPfires and distance [self] of TTneighbourMain > ComfortSpace)[
            facexy  ((TPexitMultiplier) * [pxcor] of TPexit + TTneighbourMultiplier * [xcor] of TTneighbourMain
+ (TPfireMultiplier) * ([xcor] of self - [pxcor] of TPfireMain) + [xcor] of self) / (1 + TTneighbourMultiplier +
TPexitMultiplier + SafestPatchMultiplier)
                        ((TPexitMultiplier) * [pycor] of TPexit + TTneighbourMultiplier * [ycor] of TTneighbourMain
+ (TPfireMultiplier) * ([ycor] of self - [pycor] of TPfireMain) + [ycor] of self) / (1 + TTneighbourMultiplier +
TPexitMultiplier + SafestPatchMultiplier)
  ]
    if (not any? TPfires and distance [self] of TTneighbourMain <= ComfortSpace)[
```

```
        facexy  (TPexitMultiplier * [pxcor] of TPexit + TTneighbourMultiplier * ([xcor] of self - [xcor] of
TTneighbourMain) + [xcor] of self) / (1 + TPexitMultiplier)
                ((TPexitMultiplier) * [pycor] of TPexit + TTneighbourMultiplier * ([ycor] of self - [ycor] of
TTneighbourMain) + [ycor] of self) / (1 + TPexitMultiplier)
  ]
   if (any? TPfires and distance [self] of TTneighbourMain <= ComfortSpace)[
        facexy  ((TPexitMultiplier) * [pxcor] of TPexit + TTneighbourMultiplier * ([xcor] of self - [xcor] of
TTneighbourMain) + [xcor] of self + (TPfireMultiplier) * ([xcor] of self - [pxcor] of TPfireMain) + [xcor] of
self) / (2 + TPexitMultiplier + SafestPatchMultiplier)
                ((TPexitMultiplier) * [pycor] of TPexit + TTneighbourMultiplier * ([ycor] of self - [ycor] of
TTneighbourMain) + [ycor] of self + (TPfireMultiplier) * ([ycor] of self - [pycor] of TPfireMain) + [ycor] of
self) / (2 + TPexitMultiplier + SafestPatchMultiplier)
  ]
        fd Speed
  ]
end

to act-RunFire
  facexy  ((TPfireMultiplier) * ([xcor] of self - [pxcor] of TPfireMain) + [xcor] of self) / (SafestPatchMultiplier
+ 1)
  ((TPfireMultiplier) * ([ycor] of self - [pycor] of TPfireMain) + [ycor] of self) / (SafestPatchMultiplier + 1)

  fd Speed
end



to act-Wall
  ;;bk 0.1
        facexy (2 * [xcor] of self - [pxcor] of WallMain)
                (2 * [ycor] of self - [pycor] of WallMain)
  set subrandom (random 2)
  if (subrandom = 1)[
  rt 30
  ]
  if (subrandom = 2)[
```

```
   lt 30
  ]
            fd Speed
end



to act-InnerExit
            set InnerExitDuration InnerExitDuration + 1
            if (InnerExitDuration >= ExitTime) [
                        facexy  ([pxcor] of TPexitNew)
                                    ([pycor] of TPexitNew)
   move-to TPexitNew
   set Section 1
            ]
end

to act-OuterExit
            set OuterExitDuration OuterExitDuration + 1
  if (OuterExitDuration >= ExitTime) [
    set SafeCount SafeCount + 1
    die
  ]
end

to-report FireDeaths
  report DeathFireCount
end

to-report SmokeDeaths
  report DeathSmokeCount
end

to-report TrampleDeaths
  report DeathTrampleCount
end
```

```
to-report Safe
  report SafeCount
end

to ResetData
  r:__evaldirect "rm()"
  r:gc
  r:__evaldirect "DataFrame <- as.data.frame(matrix(ncol = 21, nrow = 0))"
  ;;r:__evaldirect "colnames(DataFrame)[1] <- 'Fire'"
  ;;r:__evaldirect "colnames(DataFrame)[2] <- 'Smoke'"
  ;;r:__evaldirect "colnames(DataFrame)[3] <- 'Trample'"
  ;;r:__evaldirect "colnames(DataFrame)[4] <- 'Evacuated'"
  ;;r:__evaldirect "colnames(DataFrame)[5] <- 'Ticks'"
end

to ExportData
  r:__evaldirect "write.csv(DataFrame, file = 'C:/Users/Peng/Desktop/Assignment4DATA.csv')"
  ;;r:__evaldirect "write.csv(DataFrame, file = '/home/think/Desktop/DATA.csv')"
end
```

References

1. S. Gwynne, E. R. Galea, P. J. Lawrence, L. Filippidis, *Modelling occupant Interaction with fire conditions using the building EXODUS evacuation model*, Fire Safety Journal 36 (2001) 327–357.

2. M.Kobes, I. Helsloot, B. Varies, J. G. Post, N. Oberije, K. Groenewegena, Way finding during fire evacuation; an analysis of unannounced fire drills in a hotel at night, Built and Environment, Vol 45., (2010) 537-548.

3. Peizhong Yang, Chao Li, Dehu Chen, *Fire emergency evacuation simulation based on integrated fire–evacuation model with discrete design method,* Advances in Engineering Software 65 (2013) 101–111.

4. Manh Hung Nguyen, Tuong Vinh Ho, Jean-Daniel Zucker, *Integration of Smoke Effect and Blind Evacuation Strategy (SEBES) within fire evacuation simulation,* Simulation Modelling Practice and Theory 36 (2013) 44–59.

5. Ren C., Yang C., Jin S., *Agent-Based Modeling and Simulation on Emergency Evacuation, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Vol 5., (2009) 1451-1451.

6. Pan X., Han C. S., Dauber K., *A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations*, AI & Society, Vol 22., (2007) 113-132.

7. D.J. O'Connor, ***Integrating human behaviour factors into design,*** Journal of Fire Protection Engineering, 28 (2005), pp. 8-20

8. J. D. Sime, Crowd psychology and engineering, Safety Science, 21 (1995), 1-14.