Emergency Egress

New Mexico

Supercomputing Challenge

Final Report

March 29, 2007

Team 15

Artesia High School

Team Members

Jeff Mayberry

Casey Haldeman

Destry Kinnibrugh

Shane Wilson

Teachers

Mr. Gaylor

Unofficial Mentor

Nick Bennett

Executive Summary

Any environment is suitable for only a limited amount of occupants due to the limits of particular resources and shelter. In a building environment such as the AHS Auditorium resources and shelter are limited. Thus when buildings are viewed as an environment, these same principles can apply. A building can only suit a certain capacity without endangering the entire population. When buildings are overcrowded, the evacuation of the people in the building is hindered because of the congestion of doorway. People adopt a "survival of the fittest" attitude to save their own skins. When large masses of people rush an exit, the disorder that ensues causes an extreme slowing of the exit rate which is dangerous for those inside.

Building capacities have been set in place for this very reason. A building is given a maximum capacity based on its architectural design, number of exits, and other similar factors.

The auditorium at Artesia High School is a very old building. We were curious if the maximum capacity was still in accordance with current standards. To test this, we built an agent-based model of peoples' behavior in a panic situation inside our high school's auditorium.

Problem Description

We choose to model the evacuation of people in an emergency event. We meandered from setting off explosives inside tightly packed football stadiums (a wild idea from back before we understood the technical problems with coding such an event), to the more practical approach of simulating a fire evacuation inside our high school's auditorium.

Until recently, the team has been undecided, and for the most part unconcerned, about the actual problem our model was based off of. When it became evident that a solvable problem was crucial, we realized that our model was very suited to testing the maximum capacity of the auditorium.

To do so, our model has focused primarily on the behavioral aspects of human instincts during a crisis situation. This includes the basic instinct to stay with a crowd for safety, the occasional occurrence of bad decisions due to panic, the tendency of people to trample others in an attempt to escape, also, we have a source of danger for them to avoid and escape from. Due to the pyrotechnic outlook we have retained throughout the challenge, this is labeled as fire. Although it is referred to as a fire, it is not limited to being a source of flame. It represents any danger that would force people to evacuate the building.

In any iteration of our simulation, it is assumed to be a panic situation whether a threat to their health is actually present or imagined. This is not a model that simulates normal exiting procedures under normal circumstances. In any run of this model, the agents are subjected to a crisis situation with which they must react.

Methods

I: Intro

We would like to specify that in the model the "fire" is not necessarily fire. Do not be alarmed when I don't explain the movement of the fire. It is the basic diffusion of heat. The "fire" could be anything that would cause panic in a crowd i.e. gas, smoke etc. We will however use this section to explain the behavior of the people trying to escape the building. We used Netlogo to model our evacuation. In this section I will cover the code and explain how it models each particular behavior.

II: Basic Movement Through the Model's World

We wrote a "step" command (line 328 in the code appendix) to allow the turtles to walk through the world while evading fire and walls.

"if (any? ((patches in-cone 3 60) with [on-fire?]))"

It first asks the agent to check its cone of sight for burning patches.

"let danger-patches ((patches in-radius 5) with [on-fire?]) face max-one-of neighbors [min values-from danger-patches [distance myself]] forward 1"

If it finds itself less than three patches away from fire, it checks the surrounding five patches in any direction for other burning patches and chooses the safest near patch to step onto. It then walks onto that patch and repeats this process.

"if is-wall?-of patch-ahead 1 [let x dx + xcor let y dy + ycor face min-one-of neighbors with [not is-wall?] [distancexy x y]] fd 1"

If it is one patch away from running into a wall it looks around to the nearest patch that reports false to the "is-wall?" variable and steps onto it. Writing this particular piece of code was one of our most important breakthroughs. Before we had it we made the turtles do a 180 and jump the other way. It often sent them straight through a nearby wall if they were in a hallway. This new piece of code allowed for much more organized, orderly movement through the building.

Movement through a crowded auditorium filled with panicked people would have to be chaotic. We included this code to induce some confusion (code appendix line 342).

```
 let min-scent min values-from neighbors with [not is-wall?] [scent] - 4
 let selected-neighbor max-one-of neighbors with [not is-wall?] [(scent - min-scent) * random-float 1]
 face selected-neighbor (step)
```

The agents decide their path by following the scent coming from the doors. Here we multiply the scent by a random floating point number that is less than one. It could easily cause the turtle to head directly toward the door but could also send it off in a totally random direction, but the more common case involves the turtles heading in the correct direction. We will go more into detail on the nest scent in **Section IV: Turtle Behavior.**

III: The Dangers of the World

Obviously people can (and often are) killed in emergency evacuations of burning buildings. In order to model this reality the turtles have a variable called "energy". This variable allows us to modify the health of the turtles as they move through the building. There are various dangers in the virtual world of the model where turtles can be hurt or killed, be it by means of trampling, burning or otherwise.

“if not nest? [let victim one-of (other-turtles-here with [energy <= energy-of myself]) if victim != nobody [ask victim [get-trampled]]] to get-trampled set trampled? true set energy energy - 2
if energy <= 0 [die] end” (line 362 in appendix)

In this piece of code the turtle asks the patch it is standing on if there are other agents on the same patch so that it can trample them. It first declares any other turtles on the patch as a “victim”. It then checks to see if the energy of that turtles is not greater than its own, and if it is not, it asks the turtle to set its “trampled?” variable to true. When a turtles sets its “trampled?” variable to true it simply subtracts two energy points and reports its remaining energy. If its energy is reported to be less than or equal to zero, it tells the agent to die. The code reports the value of the “nest?” variable before it executes anything else. This is in place to ensure that turtles don't trample each other in the safe zone.

IV: Turtle Behavior

Turtles follow a flocking behavior in the model which causes them to run in groups. This can have catastrophic effects because if one turtles gets confused and runs of towards the fire the rest of the flock tends to follow suite and take off after him. Of course their path can easily be redirected by a turtle that turns the right way as the flock will follow him just as easily as they were steered off course by the rouge.

We inserted Uri Wilensky’s flocking code into our model and adapted it to work for our purposes.

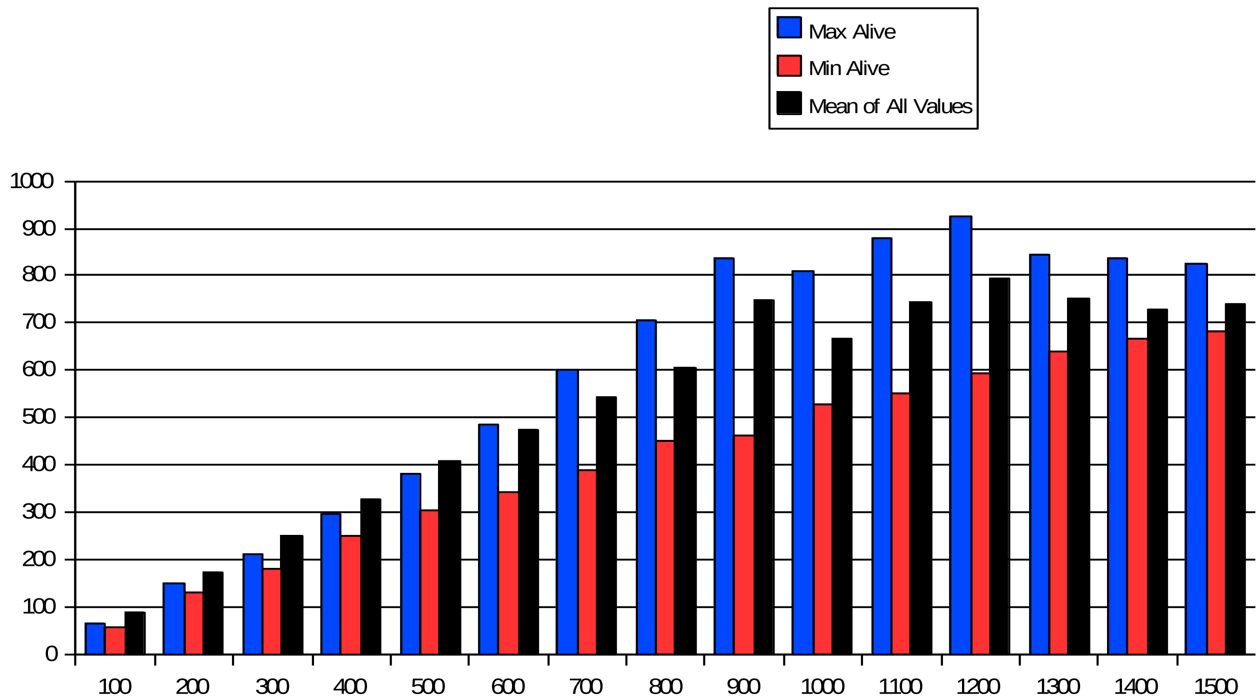
```
set flockmates (turtles in-radius vision) with [self != myself] end to find-nearest-neighbor
 set nearest-neighbor min-one-of flockmates [distance myself] end to separate
  turn-away (heading-of nearest-neighbor) max-separate-turn end to align
  turn-towards average-flock?mate-heading max-align-turn
end to-report average-flock?mate-heading  report atan sum values-from flockmates [sin heading]
     sum values-from flockmates [cos heading] end to cohere
  turn-towards average-heading-towards-flockmates max-cohere-turn
end to-report average-heading-towards-flockmates
  report atan mean values-from flockmates [sin (towards myself + 180)]
```

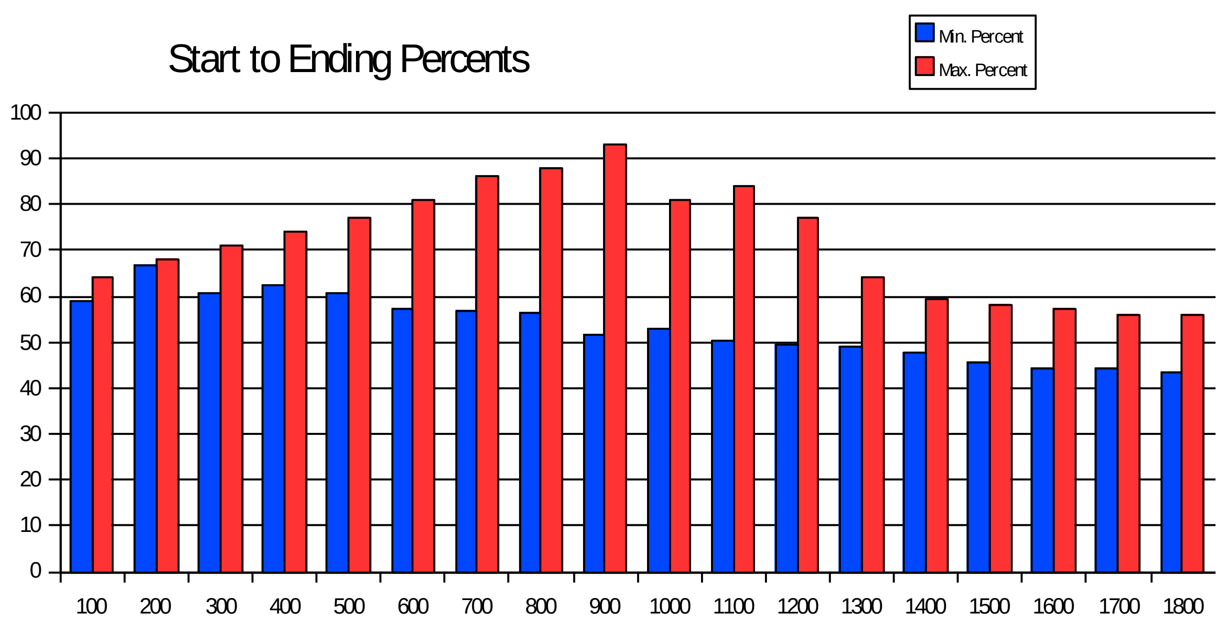
```
        mean values-from flockmates [cos (towards myself + 180)]
end to turn-towards [new-heading max-turn]
  turn-at-most (subtract-headings new-heading heading) max-turn
end to turn-away [new-heading max-turn]
  turn-at-most (subtract-headings heading new-heading) max-turn
end to turn-at-most [turn max-turn]
  ifelse abs turn > max-turn [ ifelse turn > 0 [ rt max-turn ] [ lt max-turn ] ] [ rt turn ] end
```

"*The birds follow three rules: 'alignment', 'separation', and 'cohesion'. 'Alignment' means that a bird tends to turn so that it is moving in the same direction that nearby birds are moving. 'Separation' means that a bird will turn to avoid another bird which gets too close. 'Cohesion' means that a bird will move towards other nearby birds (unless another bird is too close). When two birds are too close, the 'separation' rule overrides the other two, which are deactivated until the minimum separation is achieved.*"—**Uri Wilensky**

The agents get their sense of direction from the diffusion of scent across the model. The "chemical" cannot pass through walls and it is stopped dead in its tracks by fire. Path-finding was one of our original problems. It was a common problem with such models. People came up with various ideas for a solution such as using elevation and the turtles always moving downhill or turtles following a color gradient to find their way around. We were looking at an ant model that used pheromones to help the ants get back to the nest. We took that code and spent several weeks modifying it and adjusting the variable values to suite our needs. We decided to use that particular code because we already had a color gradient illustrating the spread of the "fire". (see section II or check the code appendix at line 342)

Results


Max Alive
Min Alive
Mean of All Values
1000
900
800
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500


Start to Ending Percents
Min. Percent
Max. Percent
100
90
80
70
60
50
40
30
20
10
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800

By using Netlogo's Behavior Space program, we were able to run our agent based model hundreds of times and collect results from the data which was collected. Since trying to calculate the building's ideal capacity we were able to set fire expansion rate, and fire start positions at fixed rates while varying amounts of people. The results extracted from the hundreds of runs can be summed up with one phrase- The Law Of Diminishing Returns.

The Law of Diminishing Returns states that in a system with fixed and variable inputs , beyond some point, each additional unit of variable input yields less and less additional output. While mainly used in referring to economics, this law can encompass everything from studying to car buying. In terms of our project this means that with building size and fire/smoke diffusion as fixed inputs and number of people in the building as a variable input, as more and more people are put in the building output levels (People remaining alive) will increase to maximum amount and then will start to decline. The charts above help explain this. With a fire set at a set spot we see that over 90% of people survived in the nine-hundred person run of the simulation. The sudden decrease at nine-hundred is caused by variables such as- trampling, over crowding, etc.

While not completely realistic, our project can still be inferred in real life. In February of 2003, a fire consumed a concert in Rhode Island, leaving 97 dead and 184 wounded. The results of our project reflect those in that fire. With a proper building limit in place and state of the arc fire safety equipment, we hope to prevent such occurrences from happening again.

Conclusions

According to the data we have compiled, our High school auditorium isn't exactly the safest building in Artesia. Using our Netlogo model we have run our program, which has gone through many debugging stages, many times over and come to the conclusion that the safest capacity of people the auditorium can hold is approximately nine hundred as opposed to the two thousand thirtyseven recommended by current standards. Our research has shown a bell curve type of statistics. Nine hundred people tops the curve as the safest while lesser or greater amounts endangers the audience. I would like to believe that our most significant achievement regarding the project was the fact that our agent based model became the most advanced that has come to the challenge this year when it comes to modeling Emergency Egress.

Panic can kill as easily as the fire we have emulated for the people to flee from, this has relevance with our project due to the fact that a lot of the children are killed by being trampled rather than the fire. Because people panic, more lives are lost, panic can kill you even if nothing else wants to, as shown by our model when it is run without the fire.

Works Cited

Helbing, Dirk, Tamas Vicsek, Illes J. Farkas, and Peter Molnar. Simulation of Pedestrian Crowds inNormal and Evacuation Situations. Eotvos U.,Clark Atlanta U. Oo. 27 Oct. 2006.

Pan, Xiaoshan, Charles S. Han, and Kincho H. Law. A MULTI-AGENT BASED SIMULATIONFRAMEWORK FOR THE STUDY OF HUMAN AND SOCIAL BEHAVIOR IN EGRESS ANALYSIS. Stanford U. 1-12. 3 Jan. 2007.

Wilensky, Uri. Flocking Model. 1998

Acknowledgments

Our thanks to:

Mr. Gaylor,

Mr. Orth,

Mr. Caton,

Mr. Phipps,

Mr. Bennett,

Mr. and Mrs. Bob Mayberry,

Mr. and Mrs. Charles Haldeman,

Mr. and Mrs. Tracy Kinnibrugh,

Mr. and Mrs. ____ Wilson,

Mrs. Mathis,

Mr. Conner,

Mr. Roberts,

and all others who made the Challenge possible.

Appendix: Code

```
breed [averagers averager]
breed [elderly senior]
breed [children child]
globals [
  fire-temperature
  spontaneous-combustion-threshold
  normal-combustion-threshold
  seat-color
  max-scent
  wall-color
  safe-color
]

patches-own [
  on-fire?
  temperature
  fuel
  is-wall?
  chemical
  food
  nest?
  scent
  is-safe?
]

turtles-own [
  carrying-food?
  flockmates
  nearest-neighbor
  energy
  trampled?
]

; Following are setup procedures:

to setup
    ca
    setup-globals
    setup-patches-from-import
```

```
    diffuse-scent
    setup-turtles
    setup-fire
end

to setup-globals
    set max-scent 1000
    set wall-color blue
    set safe-color cyan
    set seat-color brown
    set fire-temperature 2
    set spontaneous-combustion-threshold 1
    set normal-combustion-threshold 0.5
end

to setup-patches-from-import
    import-pcolors "Aud_5426rsd2.png"
       ask patches
          [set nest?
            (((abs pxcor) = max-pxcor) or ((abs pycor) = max-pycor))
            if (nest?)[set pcolor violet ]]
     ask patches [
        set is-safe? false
        set is-wall? false
        set scent 0
        ifelse (shade-of? pcolor 115) [
            set nest? true
            set scent max-scent
        ] [
            if (shade-of? pcolor wall-color) [
                set is-wall? true
            ]
        ]]
end

to diffuse-scent
    loop [
        let propagation-set (
            patches with [
                not nest?
                and not is-wall?
                and any? neighbors with [
                    not is-wall?
                    and ((scent - scent-of myself) > (0.001 + sqrt 2))
                ]
            ]
```

```
    )
    ifelse (any? propagation-set) [
      ask propagation-set [
        set scent (
          max list
            max values-from neighbors4 [scent - 1]
            max values-from neighbors [scent - sqrt 2]
        )
      ]
    ] [
      stop
    ]
  ]
end

to setup-turtles
 ask (n-of number-of-averagers
      (patches with [pcolor = seat-color])) [
    sprout-averagers 1[
  setxy (xcor - 0.5 + random-float 1) (ycor - 0.5 + random-float 1)
  set size 2
  set color red
  set carrying-food? true
  set energy 100
  set trampled? false
  set breed (averagers)]
]
 ask (n-of number-of-children
      (patches with [pcolor = seat-color])) [
    sprout-children 1[
  setxy (xcor - 0.5 + random-float 1) (ycor - 0.5 + random-float 1)
  set size 2
  set color green
  set carrying-food? true
  set energy 60
  set trampled? false
  set breed (children)]
]
 ask (n-of number-of-elderly
      (patches with [pcolor = seat-color])) [
    sprout-elderly 1[
  setxy (xcor - 0.5 + random-float 1) (ycor - 0.5 + random-float 1)
  set size 2
  set color yellow
  set carrying-food? true
  set energy 40
```

```
    set trampled? false
    set breed (elderly)]
]
end

to setup-fire
   ask patches [
  set on-fire? false
  set temperature 0
  ifelse (shade-of? pcolor cyan) [
    set fuel 0
    set is-safe? true
    ][
    set fuel 50
    set is-safe? false ]
  ifelse (shade-of? pcolor blue) [
    set fuel 1
    set is-wall? true
  ] [
    set fuel 50
    set is-wall? false
  ]
  ifelse (shade-of? pcolor violet) [
  set fuel 0
  set nest? true
  ][
  set fuel 50
  set nest? false
  ]
    draw-color-gradient
  ]
end


; Run Procedures:

to do-fire
  if (mouse-down?) [
      ask patch-at mouse-xcor mouse-ycor [
          catch-fire
          draw-color-gradient
        ]
    ]
end

to iterate
```

```
  dissipate-heat
  ask patches with [on-fire?] [
    burn
    set scent 0
  ]
  ask patches [
    spread-fire
    draw-color-gradient
    set temperature (temperature * 0.99)
  ]
  ask turtles [ flock? ]
  ask turtles [ if carrying-food? [ return-to-nest ]]
    diffuse chemical (diffusion-rate / 100)
  ask turtles [ trample ]
  ask turtles [if pcolor = yellow [die]]

end


to dissipate-heat
  diffuse temperature 0.75
  ask patches with [is-wall?] [
    set temperature (temperature * 0.45)
  ]
  ask patches with [is-safe?] [
    set temperature 0
    ]
   ask patches[
      set chemical (chemical * (100 - evaporation-rate) / 100)]
end

to draw-color-gradient
  if (not is-wall?) [
    ifelse (on-fire?) [
      set pcolor yellow
    ] [
      set pcolor scale-color red temperature 0 2
    ]
  ]
end

to burn
  set temperature 2
  set fuel (fuel - 1)
  if (fuel <= 0) [
    set on-fire? false
```

```
  ]
end

to catch-fire
  set on-fire? true
  set temperature 2
end

to spread-fire
  if ((not on-fire?) and (fuel > 0)) [
    ifelse (temperature >= spontaneous-combustion-threshold) [
      catch-fire
    ] [
      if ((temperature >= normal-combustion-threshold) and (any? neighbors4 with [on-fire?]))
[
        catch-fire
      ]
    ]
  ]
end

to flock?
   if nest?
  [set carrying-food? false
    stop
  ]
  if (not nest? and not is-wall?)
    [set carrying-food? true
    if (is-wall?)
     [ set carrying-food? true
       step
       find-flockmates
       ]
    find-flockmates
      if any? flockmates
        [ find-nearest-neighbor
          ifelse distance nearest-neighbor < minimum-separation
            [ separate ]
            [ align
              cohere ]
              ]]
end

to find-flockmates
  set flockmates (turtles in-radius vision) with [self != myself]
end
```

```
to find-nearest-neighbor
  set nearest-neighbor min-one-of flockmates [distance myself]
end

to separate
  turn-away (heading-of nearest-neighbor) max-separate-turn
end

to align
  turn-towards average-flock?mate-heading max-align-turn
end

to-report average-flock?mate-heading
  report atan sum values-from flockmates [sin heading]
              sum values-from flockmates [cos heading]
end

to cohere
  turn-towards average-heading-towards-flockmates max-cohere-turn
end

to-report average-heading-towards-flockmates
  report atan mean values-from flockmates [sin (towards myself + 180)]
              mean values-from flockmates [cos (towards myself + 180)]
end

to turn-towards [new-heading max-turn]
  turn-at-most (subtract-headings new-heading heading) max-turn
end

to turn-away [new-heading max-turn]
  turn-at-most (subtract-headings heading new-heading) max-turn
end


to turn-at-most [turn max-turn]
  ifelse abs turn > max-turn
    [ ifelse turn > 0
        [ rt max-turn ]
        [ lt max-turn ] ]
    [ rt turn ]
end

to return-to-nest
  ifelse nest?
```

```
  [set carrying-food? false
    stop
  ][
  step
  if (not is-wall? and not nest?)[
    uphill-nest-scent
    wiggle
    ]]
end

to step
    if (any? ((patches in-cone 3 60) with [on-fire?])) [
      let danger-patches ((patches in-radius 5) with [on-fire?])
      face max-one-of neighbors [min values-from danger-patches [distance myself]]
      forward 1
]
    if is-wall?-of patch-ahead 1 [
     let x dx + xcor
     let y dy + ycor
     face min-one-of neighbors with [not is-wall?] [distancexy x y]
 ]
fd 1
end

to uphill-nest-scent
  wiggle
  let min-scent min values-from neighbors with [not is-wall?] [scent] - 4
  let selected-neighbor max-one-of neighbors with [not is-wall?] [(scent - min-scent) *
random-float 1]
  face selected-neighbor (step)
end

to wiggle
  if (not is-wall?) [
    rt random 40 - random 40
    if not can-move? 1
      [ rt 180 ]]
end

to-report get-nest-scent [ angle ]
  let p patch-right-and-ahead angle 1
  if p != nobody
  [ report scent-of p ]
  report 0
end
```

```
to trample
if not nest? [
let victim one-of (other-turtles-here
with [energy <= energy-of myself])
if victim != nobody [
ask victim [
get-trampled
]
]
]
end

to get-trampled
set trampled? true
set energy energy - 2
if energy <= 0 [
die]
end
```