PSEUDOCODE FOR “SHIP\_SET\_SAIL” and associated subroutines

Overview

The module simulates the passage of ships. Each ship has been insured in a previous module by one of the 3 insurers.

The first part of the programme displays data on the ships (typically 8) to the player. The data includes the shortest route calculated allowing for ocean drift and prevailing wind.

The player then chooses to “Set Sail”. Realtime clock simulation starts at this time.

The ships progress is displayed along with weather hazards and static hazards. During the voyage each ship can incur damage eg from being blown against rocks. Some damage simply results in delays. Other more serious damage can slow the progress of the ship, and result in damage which needs to be repaired at the next suitable port. The cost of this damage can be reclaimed from the insurer of that particular ship.

CLASSES USED

Classes used inside and outside of “Ships\_set\_sail’ are in subroutines, otherwise they are at the top of the main ships\_set\_sail\_sub.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** |  |  |  | **Location** |
|  | **Attributes** | **Source** | **Notes** | **subroutines** |
| **Ship** | | | | |
|  | ***Attributes directly loaded from data file*** | | | |
|  | ship\_name | local\_data.ship\_data | ship\_data is indexed according to listing |  |
|  | port , destination | “ | “ | “ |
|  | tons, age, place of build |  |  |  |
|  | hull condition, rig condition, haul |  |  |  |
|  | haul | “ | 1 for long haul, 2 for short haul |  |
|  | ***Calculated attributes*** | | | |
|  | place\_of\_build\_preference | other attributes | all places of build preferred over ‘plantation’ |  |
|  | rig\_speed\_factor | calculated from rig\_condition | faster for better rig condition |  |
|  | hull\_speed\_factor | calculated from hull\_condition | faster for better hull condition |  |
|  | ship\_speed\_pix | calculated from rig\_condition and hul\_condition | a ship with good rig and hull condition crosses one grid square of 16 pixels every hour of simulated time |  |
|  | ship\_speed\_cond | ship\_speed\_pix/8 | one grid square every 2 nautical miles |  |
|  | ***Attributes initialised to Zero*** | | | |
|  | port\_x,port\_y,  destination\_x,destination\_y | for port and destination co-ordinates |  |  |
|  | ship\_x\_last,ship\_y\_last  ship\_x,ship\_y | for ship co-ordinates |  |  |
|  | ports\_tuple | stores port and destination as position tuple |  |  |
|  | ship\_k | step on route as 16 pixel grid |  |  |
|  | ship\_depart\_time, arrive\_time | simulated clock time |  |  |
|  | outbound\_time, inbound\_time | duration |  |  |
|  | port\_delay | days held in port to stagger departures – assigned at run time |  |  |
|  | move\_x, move\_y | amount to move shio on each calculation as pixels |  |  |
|  | weather\_disp\_x,  weather\_disp\_y | ship movement due to weather |  |  |
|  | ship\_hazard\_counter | number of times a ship encounters a particular hazard | if it accumulates to hazard\_counter\_max(5) ship damage results |  |
|  | ship\_premium | stores the amount of premium paid for insuring this ship | | |
|  | ship\_premium\_accum | stores the accumulated premiums paid each year | |  |
|  | ship\_premium\_counter | pay premium just one per year |  |  |
|  | ship\_damage\_accum | accumulates the value of ship damage |  |  |
|  | ***Attributes initialised as Boolean*** | | | |
|  | ship\_go | True is going, False if returning |  |  |
|  | ship\_infoge, infogw | in eastern or wester fog | fog affected |  |
|  | same for storms, hurricanes, pirates |  | affected by storms, hurricanes, pirates |  |
|  | **Attributes initialised as Lists** | | | |
|  | path\_go, path\_back | as list of points as a tuple with x,y values |  |  |
|  | ship\_event\_x\_list, ship\_event\_y\_list | x and y where event occurs |  |  |
|  | ship\_log | list of events |  |  |
|  | ***Revenue and Cost Related Attributes*** | | | |
|  | exports | 0.7 £/ton | move this constant |  |
|  | imports | 1,75 £/ton | “ |  |
|  | revenue out | tons\*exports/haul | longer haul (haul=1) gets higher revenue than haul=2 |  |
|  | revenue in | tons\* exports/haul |  |  |
|  | value | 3000 for a 200 ton ship |  |  |
|  | ship\_value | ratio’s from value | using law of 6 tenths |  |
|  | ship\_repair | 0.33\*ship\_value |  |  |
|  | ship\_balance\_ins | balance of income and expenses as iinsured |  |  |
|  | ship\_balance\_unins | theoretical balance if damage comes from shipowners pocket not insurers | | |
|  | ***Other attributes*** | | | |
|  | marker\_radius | 5 | can be varied e.g. larger when in hazard |  |
|  | ship\_insurer | placeholder for name |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **Ship Class Methods** | | | | |
| **ship\_speed\_reset** | | | | |
|  | resets hull and rig condition speed factors following an event |  |  |  |
| **ship\_repair\_sub** | | | | |
|  | resets hull and rig condition speed factors following a repair |  |  |  |
| **get\_port** | | | | |
|  | finds port and destination and finds shortest routed | uses weighted astar which takes account of currents |  |  |
| **ship\_log\_update** | | | | |
|  | inserts first four lines of fixed log text including premium paid | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** |  |  |  | **Location** |
|  | **Attributes** | **Source** | **Notes** | **subroutines** |
| **Weather Event** | | | | |
|  | ***Attributes directly loaded from data file*** | | | |
|  | event\_type | from local\_data.weather\_events\_list |  |  |
|  | month\_start, month\_end | seasonal cycle |  |  |
|  | duration | 30 \* local\_data to convert to days |  |  |
|  | trajectory | initial trajectorys as an angle |  |  |
|  | speed | speed of movement |  |  |
|  | traj\_boundary\_plus, traj\_boundary\_minus | y values limiting geographical area of weather event |  |  |
|  | wind\_speed\_mi, wind\_speed\_max |  | check duplication of wind\_speed minimum programme |  |
|  | starting\_event\_radius | some events start bigger than others |  |  |
|  |  |  |  |  |
|  | ***Calculated attributes*** | | | |
|  | x\_limit,y\_limit | 16 pixels for one tile |  |  |
|  | event\_x,event\_y | randomised within x\_limit and y\_limit from origin specified in data\_file |  |  |
|  | ***Attributes initialised to Zero*** | | | |
|  | started\_days | day event started |  |  |
|  | age | age of event |  |  |
|  | wind\_speed |  |  |  |
|  | event \_radius |  |  |  |
|  | ***Attributes initialised as Boolean*** | | | |
|  | in\_season |  |  |  |
|  | started |  |  |  |
|  | exists |  |  |  |
|  | ended |  | may be some reduncancy here |  |
|  | **Attributes initialised as Lists** | | | |
|  | event\_x\_list, event\_y\_list | keeps a track of the event |  |  |
|  | month\_end\_reset | -1: prevents recurrence of event within the same month |  |  |
| **Weather Event Class Methods** | | | | |
| **reset (I for weather event, mytotal\_time\_months)** | | | | |
|  | to reset weather event attributes once age is expired | | | |
| **drift\_event( myinterval\_days,event as j)** | | | | |
| 323 | returns a tuple of event position and wind speed | | | |
|  | retrieves the trajectory boundaries as plus and minus. Note that these boundaries are in degrees | | | |
|  | a degree of randomness is applied to trajectory within +- traj\_limit (hardcode : default 5) | | | |
|  | calculates an increment of travel for the weather event based on speed, trajectory | | | |
|  | increments event\_x and event\_y as pixels | | | |
|  | sets periphery\_speed (hardcode,default 24 knots) | | | |
|  | calculates an event fraction as age/duration. | | | |
|  | uses wind\_speed\_max to apply wind\_speed calculation to hurricanes and storms only | | | |
|  | calculates new wind speed based on periphery\_speed, and wind\_speed\_max \* event\_fraction, in order to give max wind speeds half way into the life of the event | | | |
|  | returns event\_x,event\_y, wind\_speed | | | |
|  |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** |  |  |  | **Location** |
|  | **Attributes** | **Source** | **Notes** | **subroutines** |
| **Insurer** | | | | |
|  | ***Attributes directly loaded from data file*** | | | |
|  | insurer\_name | from local\_data.insurer\_data |  |  |
|  | initial\_book\_value | “ |  |  |
|  | percent\_premium | “ | % value charged as annual premium |  |
|  | ***Calculated attributes*** | | | |
|  | premiums\_income | initial\_book\_value-remaining\_book\_value |  |  |
|  | premiums\_income\_accum | accumulated premiums received |  |  |
|  | balance | premums\_income-claims |  |  |
|  | ***Attributes initialised to Zero*** | | | |
|  | claims |  |  |  |
|  | ***Attributes initialised as Boolean*** | | | |
|  | None |  |  |  |
|  | **Attributes initialised as Lists** | | | |
|  | preference\_list | list of ships in preference order before selection |  |  |
|  | ships\_insured\_list | list of ships after selection |  |  |
|  |  |  |  |  |
| **Insurer Class Methods** | | | | |
| **insurer\_update** | | | | |
|  | to recalculates premiums\_income and balance | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** |  |  |  | **Location** |
|  | **Attributes** | **Source** | **Notes** | **ship\_set\_sail** |
| **Shiplog\_Button** | | | | |
|  | ***Attributes directly loaded when Shiplog\_button Class is instantiated*** | | | |
|  | x,y,w,h |  |  |  |
|  | text |  |  |  |
|  | rect\_color |  |  |  |
|  | alt | to distinguish between two lines |  |  |
|  | clicked |  |  |  |
| **Shiplog\_Button Class Methods** | | | | |
| **shiplog\_button\_blit** | | | | |
|  | creates a rect | | | |
|  | blits the rect with text and ship\_clor circle | | | |
| **shiplog\_button\_display** | | | | |
|  | creates buttons as list ( not sure if this should be a Class method) | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** |  |  |  | **Location** |
|  | **Attributes** | **Source** | **Notes** | **ship\_set\_sail** |
| **Ship\_Revenue** | | | | |
|  | ***Attributes directly loaded when Ship Revenue Class is instantiated*** | | | |
|  | x,y,w,h |  |  |  |
|  | text |  |  |  |
| **Ship\_Revenue Class Methods** | | | | |
| **shiprevenue\_blit( requires rect\_color, ship\_color** | | | | |
|  | creates a rect | | | |
|  | blits the rect with tex | | | |
| **blit\_text ( requires text\_lines, rect, color)** | | | | |
|  | displays text within a rect | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **subroutine** | | | | | | |
| **ship\_detail (window,canvas,selected\_ship\_number)** | | | | | | |
| prepares text to be displayed in Part 1 on the ships and routes | | | | | | |
| This enable the player to enquire on any particular ship before  Set Sail time. | | | | | | |
| The player selects the ship on which information is wanted from button\_click | | | | | | |
| tab 0 | tab 1 | tab 2 |  |  |  |  |
|  | prepares ship\_detail\_rect for a block in the bottom right hand corner of the screen | | | | | |
|  | the list ship\_detail\_list is created with each list item corresponding to a line of text | | | | | |
|  | The length of the journey is calulcated from the length of path\_go (needs some science here) | | | | | |
|  | the rect is blanked, then blit\_text is called to display ship\_detail\_list | | | | | |

|  |
| --- |
| **subroutine** |
| **blit\_text (canvas,ship\_detail\_list,ship\_detail\_rect,'blue')** |
| Each line in ship\_detail\_list is blitted for display |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **subroutine** | | | | | | |
| **ship\_log\_display (window,canvas,selected\_ship\_number,ship\_log)** | | | | | | |
| displayed in Part 2 on the ship’s progress, damage etc. | | | | | | |
| This enable the player to enquire on any particular ship before  Set Sail time. | | | | | | |
| The player selects the ship on which information is wanted from button\_click | | | | | | |
| tab 0 | tab 1 | tab 2 |  |  |  |  |
|  | prepares ship\_detail\_rect for a block in the bottom right hand corner of the screen | | | | | |
|  | the list ship\_detail\_list is created with each list item corresponding to a line of text | | | | | |
|  | each log is specific to a particular ship as selected by MOUSEBUTTONDOWN – the value is held in selected\_ship\_number | | | | | |
|  | The rect used for display is deliberately the same as for ship\_display | | | | | |
|  | the rect is blanked, then blit\_text is called to display ship\_log | | | | | |
|  | the log is prepared using subroutine append\_if (below) to avoid duplicate entires | | | | | |
|  |  | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **subroutine** | | | | | | |
| **append\_if (j, append\_text, mytotal\_time\_months, mytotal\_time\_days\_res, time\_stamp)** | | | | | | |
| j is the ship number. Data to be added to the ship\_log for ship j is passed to append\_if. | | | | | | |
| The purpose of this subroutine is to avoid duplicate entries, since the source programme is looping. For example ‘encounter storms’ needs to be displayed only one, whereas is append\_if will be called every loop. | | | | | | |
| There are two types of statement passed to append\_if. One requires a time stamp to be added, one not. | | | | | | |
| tab 0 | tab 1 | tab 2 |  |  |  |  |
|  | load ship\_list\_selected from local data mirror | | | | | |
|  | determine current length of ship\_log for ship j | | | | | |
|  | set append to True | | | | | |
|  | note that log entries created in ship\_set\_sail\_sub have a ‘,’ after the time entry, if there is one. It is therefore important that no other use of ‘,’ is used. | | | | | |
|  | set numb\_test as 5, the default number of log entires which will be tested to avoid duplication | | | | | |
|  | loop through the last numb\_test log entries | | | | | |
|  |  | extract the log entry and create a list, split at ‘,’ | | | | |
|  |  | if the resulting list has two parts, use only the second. | This method detaches the time from the preceding log entry | | | |
|  |  | compare the resulting part (append\_text\_test) with the new part (append\_text) | | | | |
|  |  | if they are identical set append to False | | | | |
|  | carry out the append only if append is True, adding the time stamp or not according to time\_stamp | | | | | |
|  | if the resulting ship\_log exceeds log\_max\_len (default 20) pop the first entry after the fourth - maintains the first 4 lines of the log as headers. Any change here needs also to be coordinated with ship\_log\_update | | | | | |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **subroutine** | | | | | | |
| **damage\_random\_sub (i,iw,damage\_text,mytotal\_time\_months, mytotal\_time\_days\_res,weather\_events\_list,ship\_list\_selected,mmax,insurers\_list)** | | | | | | |
| saves duplication of code | | | | | | |
| purpose is to randomly cause damage to a ship encountering a weather event | | | | | | |
| the degree of randomness depends on the event (iw) and on weather it is rigging, hull or total shipwreck which result. Randomness variable are loaded into Weather\_event from local\_data | | | | | | |
|  | damage\_text hold the string representing the weather event potentially causing damage | | | | | |
|  | damage random is set at 10 , it could be any integer | | | | | |
|  | damage increment is set to £100 for all events | this needs to be reviewed | | | | |
|  | if damage\_random is same as random number between 0 and rig\_damage\_risk | | | | | |
|  |  | append ship\_log with rig damaged by damage\_text using append\_if | | | | |
|  |  | increment ship\_damage\_accum | | | | |
|  |  | run degrade condition subroutine | | | | |
|  |  | loop through insurers |  |  |  |  |
|  |  |  | when insurer for ship I is found |  |  |  |
|  |  |  |  | increment claims | | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  | REPEAT FOR HULL DAMAGE | | | | | |
|  | if SHIPWRECK | | | | | |
|  |  | | | | | |
|  |  | append ship\_log with rig damaged by damage\_text using append\_if | | | | |
|  |  | set ship\_shipwreck==True, which causes any further code processing for this ship to be stopped | | | | |
|  |  | increment ship\_damage\_accum with full value of ship | | | | |
|  |  | loop through insurers |  |  |  |  |
|  |  |  | when insurer is found |  |  |  |
|  |  |  |  | increment claims with value of ship | | |
|  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **subroutine** | |
| **degrade\_condition(I,ship\_list\_selectedship\_condition** | |
| saves duplication of code | |
| moves either hull or rigging condition to lower level after damage | |
| uses a list of hull and rigging coniditions (hardcode) | |
|  | ship\_condition holds either Hull or Rigging |
|  | current condition found from list, index incremented by 1, new condition fund |
|  | unless currnt condition is last in list. Possible upgrade is to shipwreck beyond lowest condition |
|  |  |

**subroutine:**

**ships\_set\_sail\_sub (requires window, and canvas)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **subroutine** | | | | | | |
| **ship\_set\_sail\_sub (requires window and canvas)** | | | | | | |
| ***prerequisites*** | | | | | | |
| a random sample of ships (def 8) has been instantiated in ‘goinside’. This is held in ‘ship\_list\_selected’ which is in turn stored in ‘local\_data. | | | | | | |
| insurers ( def 3) has been instantiated in ‘goinside). This is held in ‘insurers list’ which is in turn help in ‘local\_data’. | | | | | | |
| the ships and insurers have agreed premiums, in premiums\_alt. So each ship has an insurer and a premium sotred as attributes of Ship, and each Insurer has a list of ships and premiums stored as attributes | | | | | | |
| Part 1 displays the map, routes, ships and insurers | | | | | | |
| The player is then asked to click ‘Set Sail’ which initiates the clock. | | | | | | |
| Part 2 shows the progress of the ships | | | | | | |
| tab 0/ line number | tab 1 | tab 2 |  |  |  |  |
| 138 | smax, the number of ships, (def 8) retrieved from local data | | | | | |
|  | ship\_list\_selected, a list of instantiated Ships with their attributes retrieved | | | | | |
|  | mmax, the number of insurers (def 3) retrieved from local\_data | | | | | |
|  | insurers\_list, a list of instantiated Insurers with their attributes retrieved | | | | | |
|  | runs insurer\_update to update balance | | | | | |
|  | sets parameters on mapwidth and height, margins, padding, cell\_width and height | | | | | |
|  | sets empty list and display parameters for insurer\_finances\_nested\_list | | | | | |
|  | sets colors and fonts | | | | | |
| 168 | loads grid as map\_map using TileMap. This shows drift and fixed hazards.  It is not displayed but used in calculating ship drift and collisions with fixed hazards.  The associated pseudocode is in Routes | | | | | |
|  | load the background map of the North Atlantic | | | | | |
|  | set constants relating to ship travel including angle\_haz, angle\_avoid described later when they are used | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | distortion\_factor | initialised to 1 , altered later to reflect compression to bottom of N Atlantic map | | | | |
|  | drift\_speed | this is a master value to tune the movement of the ship due to drift (ospeed) | set without science to 0.001 | | | |
|  | weather\_disp\_fract | this is a master value to due the vovement of the ship due to weather( weather\_disp\_x,y | set without science to 0.01 | | | |
|  | wind\_speed\_min | minimum wind speed for windy events | set to 24 knots , possible duplicate with that set in weather\_event | | | |
|  | hazard\_counter\_max | number of times a ship encounters a hazard before it is listed as damaged | default set to 5 , needs tuned | | | |
|  | convert\_pixel | 16 to convert grid to pixels |  |  |  |  |
| 186 | weather\_sep | separates weather events | default 0 |  |  |  |
|  | game\_speed\_conv | defaut 5000 | 25714 milliseconds game time equals one day of ship travel | | | |
|  | alimit | default 1, if ship is within alimit of destination it is considered to have arrived | | | | |
|  | pirate\_random | default 10, a random integer which might result in pirate damage | needs to be tuned | | | |
|  | damage\_increment | default £100 for each damage event | needs to be tuned, and vary for differet damage events | | | |
|  | display slist | to be toggle to display or not | review | | | |
| 192 | toggle\_ship\_  insurer\_button\_clicked | to be toggled to display of nor | review | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 195 | displays “Displaying Shortest Routes – please wait” | | | | | |
|  | the next section creates a list of ships under each insurer. Each is a button ( as a list ‘button names’, which can be used to display more detail about the ship in the panel at the bottom left. | | | | | |
| 204 | SHIP LOG SHOW BUTTONS | | | | | |
|  | set button sizes and create empty list of button names | | | | |  |
|  | insert two lines of headers | | | | | |
|  | for each insurer m | | | | | |
|  |  | retrieve insurer name |  |  |  |  |
|  |  | prepare insurer heading in button\_names\_list |  |  |  |  |
|  |  | for each ship |  |  |  |  |
|  |  |  | identify if it is insured by this insurer m | | | |
|  |  |  | prepare ship name text in button\_names\_list |  |  |  |
|  |  |  | using the ship name retrieve the ship port and destination and prepare text in button\_names\_list | | | |
|  | create empy lists button and button\_rect | | | | | |
|  | retrieve length of list button\_names | | | | | |
|  | instantiate each Button and blit Rect | | | | | |
|  | for each button | | | | | |
|  |  | if ship\_name is ‘in’ button name | | | | |
|  |  |  | retrieve ship color |  |  |  |
|  |  |  | draw circle in that colour adjacent to text |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 250 | INSTANTIATE WEATHER EVENTS | | | | | |
|  | create empty weather events list | | | | | |
|  | instantiate weather events in weather\_events\_list | | | | | |
| 257 | DISPLAY DOTS FOR SHIP PATHS | | | | | |
|  | for each ship in smax | | | | | |
|  |  | retrieve ship color | | | | |
|  |  | use get\_port method to retrieve tuple of co-ordinates of the ship’s port and destination | | | | |
|  |  | assign a port\_delay to each ship , used after ships sail | | | | |
|  |  | draw circles for ports and destination | | | | |
|  |  | for each point in the ship’s path\_go | | | | |
|  |  |  | draw circle at the point |  |  |  |
|  |  |  | and line to next point |  |  |  |
|  |  | same for path\_back |  |  |  |  |
|  |  | blit |  |  |  |  |
| 295 | WAIT FOR SET SAIL BUTTON | | | | | |
|  | display “Click to Set Sail” | | | | | |
|  | **in pygame MOUSEBUTTONDOWN** | | | | | |
|  |  | **select button to display alternative ship data whilst wating for set sail to be clicked** | **call ship\_detail according to button[i] clicked. See ship\_detail subroutine** |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Part 2 | **Set sail as been clicked** | | | | | |
|  | tab 1 | tab 2 | tab 3 |  |  |  |
| 406 | **while running** | | | | | |
|  |  | blit image of north atlantic | | | | |
|  |  | mytime= pytime.time.get\_ticks() | | | | |
|  |  | derive my time dependancies | | | | |
|  |  | myinterval (milliseconds) | | | | |
|  |  | myinterval\_days (game time)  from myinterval/game\_speed\_conv | | | | |
|  |  | times\_res are game times resolved into years, months, days | | | | |
|  |  | draw and blit journey time rect with time | | | | |
|  |  | blit coffee menu button – returns to coffee shop menu | | | | |
|  |  | blit toggle ship insurer list button – this swiches display of ship financials | | | | |
|  |  | blit toggle dift map button – this switches on or off coloured drift map | | | | |
| 460 |  | blit toggle ship routes button – this either displays all routes for all ships, or for the one selected for ship log display | | | | |
| 469 |  | white out area for ship\_log | | | | |
| 471 |  | **create buttons for each ship listed under each insurer** by creating a list button\_names. Each entry in button\_names is a two part tuple with the first part the name, and the second part 0 or 1 or 2 depending on the color border required. Only button\_names,1 will become true buttons.  Button names, 0 is text, button\_names 2 is for port and destination | | | | |
|  |  |  |  |  |  |  |
|  |  | loop through each insurer |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  | append insurer name to button\_names list | | | |
|  |  |  | loop through each ship | | | |
|  |  |  |  | identify if ship is in insurer list and if so add to button\_names | | |
|  |  | create blank lists for button and buttontext\_rect |  |  |  |  |
|  |  | loop through button\_names | | | |  |
|  |  |  | add to list of buttons and instantiate the button[i] | | | |
|  |  | loop through button\_names |  |  |  |  |
|  |  |  | add to list of buttontext\_rects and blit rect and text | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | (running) | **develop weather events** | | | | |
| tab 0 | tab 1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
|  |  |  | | | | |
| 434 |  | loop through each entry in weather\_events\_list (index iw) | | | | |
|  |  |  | determine if that weather event is in season ( between month\_start and month\_end) | | | |
|  |  |  |  | if time in months is not equal to month\_end\_reset allow new weather event ( not sure how this works) | | |
|  |  |  |  |  | starts weather event if time in days is within a random window (0,30) and if weather event does not currently exist | |
|  |  |  |  |  |  | sets started\_days, started, and exists |
| 451 |  |  | checks that event exists | | | |
|  |  |  |  | updates event age | | |
|  |  |  |  | if age exceeds duration | runs method weather\_events\_list.reset | note duration is a Weather event property calculated in days |
| 460 |  |  | checks that event exists | | | |
|  |  |  |  | runs method **drift\_event** to retrieve position and wind speed from a tuple | | |
| 470 |  |  |  | appends weather\_x and y events list and event radius | | |
|  |  |  |  | DETERMINE DISTANCE FROM NEIGHBOURING WEATHER EVENTS | | |
| 472 |  |  |  | loop through each entry of weather\_events\_list (index ik) | | |
|  |  |  |  | calculate distance from event iw from event ik | | |
|  |  |  |  |  | determines if distance < the sum of the two weather event radii +weather\_sep (default 0) | |
|  |  |  |  |  |  |  |
|  |  |  |  | tab 6 | tab 7 | tab 8 |
|  |  |  |  | exclude Pirate events | | |
|  |  |  |  |  | exclude events of same event\_type | |
|  |  |  |  |  |  | if event radius iw is greather than ik end event ik and reset event ik. |
|  |  |  |  |  |  |  |
| tab 0 | tab 1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
| 490 |  |  |  | if event is a storm or hurricane | | |
|  |  |  |  |  | for various wind speeds | hardcoded 34,64,83,96,>96 |
|  |  |  |  |  | assign color\_ring and proportionately change event radius | proportions hardcoded |
| tab 0 | tab 1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
| 514 |  |  | (for an event iw) | (if windy event) | draw event ring |  |
|  |  |  |  | else(not windy event) draw blue ring of fixed event radius | icebergs,fog,pirates |  |
| 522 |  |  |  | create rect based on event position | | |
|  |  |  |  | slices back two characters (\_E, \_W)from event\_type to create even name to be displayed | | |
|  |  |  |  | blit name |  |  |
|  |  |  |  |  |  |  |
|  |  | ### SHIP ROUTING ### | | | | |
| 532 |  | update ship\_log for selected\_ship\_number using ship)log\_update method | | | | |
|  |  | Note: selected\_ship\_number id default 0 (line 332 apprx) | | | | |
|  |  | selected\_ship\_number is selected by MOUSEBUTTONDOWN (around line 1019) | | | | |
|  |  | run method ship\_log\_display to show log of selected\_ship\_number | | | | |
| 612 |  | loop through all ships | | | | |
|  |  |  | select ship color for display of path, circles etc | | | |
|  |  |  | if ship is shipwrecked execute no more code | | | |
|  |  |  | PAY ANNUAL PREMIUMS | | | |
|  |  |  | compare ship\_premum\_counter with years ( offset by 1 since year 0 is first year of premium payment)  if premium payment required | | | |
|  |  |  |  | add fixed premium to ship\_premium\_accum | | |
|  |  |  |  | find insurer for the ship | | |
|  |  |  |  |  | add premium to premiums\_income\_accum | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 627 |  |  | LOCATE LAST AND NEXT WAYPOINTS ON SHIP PATH | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |
|  |  |  | if ship\_go=True ( default set in Ship Class) | | | |
|  |  |  |  | determine last and next waypoints on path go | | |
|  |  |  |  | since these points are grid boxes convert to pixels and add 8 to pin point centre of box | | |
|  |  |  | if ship\_go=False | | | |
|  |  |  |  | determine last and next waypoints on path go | | |
|  |  |  |  | since these points are grid boxes convert to pixels and add 8 to pin point centre of box | | |
|  |  |  | set ship\_weather\_affected=False | | | |
|  |  |  | loop through weather events list (index iw) | | | |
|  |  |  |  | find if weather\_event exists | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | |
| tab 4 | tab 5 | tab 6 | tab 7 | tab 8 | tab 9 | |
| (weather event exists\_ | calculate distance of ship I from weather event iw | | | | | |
|  | calculate angle by which ship is approaching the event | | | | | |
|  | calculate a fract\_event\_radius as distance/event\_redius | | | | | |
|  | calculate a wind speed fraction as 1- (dist\_x/(eventraidus\* cost angle\_to\_event | | | | | |
|  | calculate a fract\_event\_radius as distance/event\_redius | | | | | |
|  | calculate a wind speed fraction as 1- (dist\_x/(eventraidus\* cost angle\_to\_event | | | | | |
|  | if fract\_event\_radius<1 | ie ship is within the event | | | | |
|  |  | wind speed proportional to difference of wind-spped\_max and wind\_spped\_min time fract\_event\_radius | | | | |
| 600 |  | cacluate weather\_disp\_x,y from weather\_disp\_fact \* days \* wind\_speed\*sin/cos angle\_to\_even \*(1-fract\_event\_radius  \_ | | | | |
| 655 | CONSEQUENCES OF DISTANCE TO WEATHER EVENT | | | | | |
|  | if weather event is Fog | | | | | |
|  |  | if within weather event radius | | | | |
|  |  |  | slice event type and assign infoge or infogew to be True | | | |
|  |  |  | append ‘encounters for’ to ship log list using append\_if | | | |
|  |  |  | set ship\_speed\_cond=0, weather\_disp x,y, to 0, and marker radius to 10 | | | |
|  |  |  | ship is fog bound at zero speed until fog has cleared | | | |
|  |  |  | run damage\_random\_sub subroutine | | | |
|  |  | else set infoge/infog\_w to False | | | | |
|  |  |  | | | | |
| 612 | if weather event is Storms | | | | | |
|  |  | if within weather event radius | | | | |
|  |  |  | slice event tupe and assign instorme or instormw to be True | | | |
|  |  |  | append ‘encounters Storms to ship\_log using append\_if | | | |
|  |  |  | set ship\_speed\_cond=0.5\*ship\_speed\_pix/8, and marker radius to 10 | | | |
|  |  |  | weather disp x,y clculated from weather\_disp\_fract \* myinterval\_days\*wind\_speed\*sin/cos of angle to event | | | |
|  |  |  | note weather\_disp\_fract is set around line 182 as 0.01 and needs some science and tuning | | | |
|  |  |  | run damage\_random\_sub to randomly find damage to rigging, hull, or whipwreck | | | |
|  |  |  | | | | |
|  |  | else set instormw,e as False | | | | |
| 638 | if weather event is Hurricane | | | | | |
|  |  | if within weather event radius | | | | |
|  |  |  | slice event type and assign inhurricanee or inhurricanew as True | | | |
|  |  |  | append ‘encounters hurricanes’ to ship\_log using append\_if | | | |
|  |  |  | set ship\_speed\_cond=0.4\*ship\_spped\_pix/8, and marker radius to 10 | | | |
|  |  |  | weather disp x,y clculated from weather\_disp\_fract \* myinterval\_days\*wind\_speed\*sin/cos of angle to event | | | |
|  |  |  | run damage\_random\_sub to randomly find damage to rigging, hull, or shipwreck | | | |
|  |  | else inhurricaneee/w False | | | | |
| tab 4 | tab 5 | | | | | |
| 666 | if weather event is Icebergs | | | | | |
|  |  | if within weather event\_radius | | | | |
|  |  |  | slice event type and assign inicebergse/w as True | | | |
|  |  |  | append ‘encounters Icebergs’ to ship\_log uusing append\_if | | | |
|  |  |  | set ship\_speed\_cond=0.3\*ship\_spped\_pix/8, and marker radius to 10 | | | |
|  |  |  | weather\_disp\_x and weather\_dispy set to zero (this needs reviewed) | | | |
|  |  |  | run damage\_random\_sub to randomly find damage to rigging, hull, or whipwreck | | | |
|  |  | else set inicebergs to False | | | | |
| 721 | if weather event is Pirates ( although not a weather event, treat the risk a similar way) | | | | | |
|  |  | if within event radius | | | | |
|  |  |  | slice event type and set inpiratee/w accordingly | | | |
|  |  |  | append ‘ encounters Pirates’ to ship\_log using append\_if | | | |
|  |  |  | pirate\_random was set to 10 in 189. If this number is equil to a randomly generated number between 1 and 2000 there is cargo damage. Needs tuned | | | |
|  |  |  | run damage\_random\_sub to randomly find damage to rigging, hull, or whipwreck | | | |
|  |  | else set in pirates to False | | | | |
| 746 | if not in any weather event set weather\_disp\_x,y to zerod, run method ship\_speed\_reset and marker\_radius to 5 | | | | | |
|  |  | | | | | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | |
| tab 0 | tab 1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
|  | DISPLAY A SHIP THORETICAL PATH | | | | | |
|  |  | (cont loop through all ships) | only if display\_all\_routes== False | ie display only route selected for ship log (sj) | | |
|  |  |  |  | for k in pathr\_go or path back draw circles | | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| tab 0 | tab 1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
|  |  | | | | | |
| 829 | DISPLACEMENT DUE TO SHIP SPEED | | | | | |
|  | (running) | (in loop of ships) | fraction x,y ship x,y-wp\_next\_xy – distance to next waypont.  Note w-\_next\_x,y was established around line 555 | | | |
|  |  |  | calculate angle\_r of ship movement | | | |
|  |  |  | adds angle\_haz ( not yet used and set to 0, intended for to model evasive action) | | | |
|  |  |  | move\_x, move y caclulated first as nautical miles and then as pixels using myinterval\_days, ship\_speed\_cond, angle\_m sin/cos and distortion\_factor | | | |
|  |  |  | Note distortion factor set to 1, possible use later for map distortion effects | | | |
| 847 | DISPLACEMENT DUE TO OCEAN DRIFT | | | | | |
|  |  |  | if ship\_go is True | | | |
|  |  |  |  | retrieve grid points from path \_go current and next | | |
|  |  |  | else | retrieve grid points from path \_back current and next | | |
|  |  |  |  | assign grid number as o\_drift and assign an ospeed\_x and ospeed\_y according to drift retrieved | | |
|  |  |  |  | Note: there is no science to the drift speeds inserted | | |
|  |  |  |  | set distortion factor based on latitude 1,6 south of 700 | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | STAGGER DEPARTURE AND CALCULATE NEW POSTION | | | | | |
| tab0 | tab1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
| 899 |  | (in loop of ships) | if length of ship\_log==0 append “in port” plus ship condition to log | | | |
|  |  |  | if ship\_k==0 and port\_delay-0 | ship set sail added to ship log | | |
|  |  |  |  | calculates new ship x,y based on move, ospeed(drift) and weather \_disp | | |
|  |  |  | or if port delay> my total time days | waiting in port added to ship log | | |
|  |  |  | else |  |  |  |
|  |  |  |  | calculates new ship x,y based on move, ospeed(drift) and weather \_disp | | |
|  |  |  | draw circle for ship with marker radius | | | |
| 943 | EVALUATE FIXED HAZARDS AT NEW POSITION | | | | | |
|  |  |  | retrieve grid x,y as grid squares (roundes) plus a residue | this is not exacet | | |
|  |  |  |  | 1=Beach, 2=Rocks,4=Land | | |
|  |  |  | find the hazard index for the square occupied | | | |
|  |  |  | if not in port (k !=0 | | | |
|  |  |  |  | if hazard=1 (beach) | | |
|  |  |  |  |  | append encoutered beach | |
|  |  |  |  | if hazard=2 | | |
|  |  |  |  |  | append encountered rocks | |
|  |  |  |  | if hazard=4 | | |
|  |  |  |  |  | append encountered land (should never get here | |
|  |  |  | hazard\_k\_last-hazard\_k | this is a marker to endure that ship is not damaged twice from the ame hazariod within the same grid square | | |
|  |  |  | if hazard is1,2 or 4 | | | |
|  |  |  |  | if hazard is 1 degrade condition | | |
|  |  |  |  | if hazard is 2 degrade conditon | | |
|  |  |  |  | if hazard is 4 degrade conditon | | |
|  |  |  |  | add damage increment | | |
|  |  |  |  | append text concerning ship damage | | |
|  |  |  |  | add damage to insurers claims | | |
|  |  |  |  | run ship\_speed\_reset | | |
|  |  |  |  | append ship condition | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| tab0 | tab1 | tab 2 | tab 3 | | tab 4 | tab 5 | tab 6 |
|  | UPDATE SHIP PROGRESS | | | | | | |
|  |  | (in loop of ships | update ship\_x,y\_last | | | | |
|  |  |  | append event\_x\_list | | to hold acctual tract of ship | | |
| 1066 | CHECK IF AT DESTINATION OR PORT | | | | | | |
|  |  |  |  |  | |  | |
| tab 2 | tab 3 | tab 4 | tab 5 | | tab 6 | tab 7 |  |
| (in loop of ships) | if ship is go forward from port to destination (ship\_go=True) | | | | | | |
|  |  | if the length of the route has been equalled or exceeded (path\_go-1) | | | | | |
|  |  |  | draw large red circle | | | | |
|  |  |  | append “ reached destination – set sail for return” to ship log | | | | |
|  |  |  | set ship\_go to False to use path back to port | | | | |
|  |  |  | record arrive time and calculate outbound time( elapsed time) | | | | |
|  |  |  | check hardcoded conversion (elapsed time \*0.2/1000) | | | | |
|  |  |  | set depart time | | | | |
|  |  |  | accumulate revenue for trip | | | | |
|  |  |  | run repair sub and append any repairs completed to ship log | | | | |
| 1090 | else ship is returning and is now at port | | | | | | |
|  |  |  | draw large red circle (shoud this not be blue for port?) | | | | |
|  |  |  | run repair sub and any reparis completed to ship log | | | | |
|  |  |  | clear event list to clear tracking | | | | |
|  |  |  | clear damage\_event\_list to clear damage displays otherwise displays overlay and become cluttered | | | | |
|  |  |  | set ship\_go to True to set sail back to destination | | | | |
|  |  |  | record arrive time and elapsed time | | | | |
|  |  |  | accumulate revenue for tri[ | | | | |
| 1111 | DRAW PROGRESS OF SHIP | | | | | | |
|  | if display\_all\_routes==True loop through event list and draw circle in correct ship colour along route for ever y ship | | | | | | |
|  | if False draw event list only for selected\_ship\_number | | | | | | |
|  | if False loop through damage event list of selected ship and print damage events on map (don’t do this for True – map becomes cluttered) | | | | | | |
|  |  |  |  | |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| tab0 | tab1 | tab 2 | tab 3 | tab 4 | tab 5 | tab 6 |
| 1166 | SHOW TABLES | | | | | |
|  |  | loop of ships has ended | | | | |
|  |  | run ship\_log\_display subroutine | | | | |
|  |  | run insurer finances display | | | | |
|  |  | if display s\_list is true ( use toggle menu button) | | | | |
|  |  | display s list – a list of ship finances | | | | |
|  |  | run ship\_list\_by\_insurer sub | | | | |
|  | DISPLAY MANAGEMENT AND MOUSEBUTTONDOWN RESPONSES | | | | | |
| 1079 |  | blit canvas and update display | | | | |
|  |  | set mytime\_last=mytime (milliseconds) for time management | | | | |
| 1082 | EVENT HANDLING | | | | | |
|  | running loop |  |  |  |  |  |
|  |  | retrieves any events using pygame.event.get() | | | | |
|  |  |  | quit if event type = quite and set running to False | | | |
|  |  |  | if mouse button has been clicked |  |  |  |
|  |  |  |  | for button name (in local\_data\_ | | |
|  |  |  |  |  | select ship number sj for use in ship log display | |
|  |  | blit canvas and update | | | | |
|  |  | if coffee menu button clicked return to goinside | | | | |
| 1157 |  | if toggle\_ship\_insurer clicked display\_slist | | | | |
|  |  | if toggle\_drift\_map display coloured drift map | | | | |
|  |  | if toggle\_rutes\_display show all routes else just on in ship\_log | | | | |
|  |  |  |  |  |  |  |