**Overview**

1. **Purpose:**

This model aims to simulate a procession that travels from the Serapeum to the seafront or river harbour over a spatial urban plan of Ostia. It allows the user to explore how different forms of static and moving spectatorship affect the processional leader’s route selection. Within the model, the processional leader does not have global knowledge of their environment, instead, they determine their route based on travel past spaces and groups of people that would give the procession the greatest visibility at each timestep.

1. **Entitles, State Variables, and Scales:**

Each run includes two sets of agents: one processional leader and processional observers.

The model requires several different GIS shapefiles to create the landscape, which is shown in fig. 1. A DEM is not included since there is very little elevation variation across the site of Ostia. For the present model, commercial and production building classifications are given the same values since there are very few production buildings within the areas surrounding the temple of Serapis.



Fig. 1. Simulation environment. Green = commercial structures, orange = production structures, yellow = residential structures, blue = public structures, pink = religious structures, grey = unexcavated cityscape, black = streets, brown = seafront.

The global variables are shown in Table 1, the patch variables in Table 2, the processional leader variables in Table 3, and the observer variables in Table 4.

*Table 1: Global Variables*

**Variable Description**

cityscape Use with GIS extension, imports the extended cityscape

streets Use with GIS extension, imports the city’s streets

street-network Use with GIS extension, imports a center-line street network

width Use with GIS extension, imports streets over 7m in width

portico Use with GIS extension, imports street porticoes

commercial Use with GIS extension, used to denote buildings defined as having a commercial function

production Use with GIS extension, used to denote buildings defined as having a production function.

domestic Use with GIS extension, used to denote buildings defined as have a residential function

religious Use with GIS extension, used to denote buildings defined as having a religious function

public Use with GIS extension, used to denote buildings defined as having a public function

coastline Use with GIS extension, used to define the seafront coastline

destination Parameter used as the final endpoint of the procession

arrived-goal Parameter used to stop the simulation if the procession has reached the designated area at the seafront or river harbour

watch-location Parameter used to help identify where observers can watch the procession

temp-tar Parameter that sets a temporary target for the leader, but its visibility has to first be checked

target Parameter that determines which patch the leader is moving towards

viewshed? Parameter used to check the viewshed of a target patch

vis? Parameter used to determine if surrounding patches with a viewshed are visible

the-leader Parameter that designates one turtle as the leader.

bldg-designation Parameter that denotes if a patch is defined as any type of building

bldg-list Parameter used for determining which patches belong to a list of buildings

*Table 2: Patch Variables*

**Variable Description**

street? True if patch is part of a street

comm Patches defined as commercial

prod Patches defined as production

dom Patches defined as domestic

rel Patches defined as religious

pub Patches defined as public

width\_ Patches of wide streets

portico\_ Patches defined as porticoes

scape Patches defined as belonging to the extended cityscape

coast Patches defined as belonging to the coastline

elevation Parameter used to track that a building exists which affects visibility

obstructed? Parameter that determines if a patch obstructs the leader's view

street-nw Patches defined as belonging to the center-line street network

influence Influence value of each patch

distance-goal Parameter that tracks distance of each patch from a specified goal. Used for processions going to the seafront and for returning to a temple

n-distance-goal Parameter that normalizes all distance values to designated goal

r-distance-goal Parameter that reverses the normalized values so the highest values are now closest the designated goal

obs-inf Parameter that attracts observers to a location

radius-range Parameter used for determining where Observers begin

export-results Parameter used to record patches that have been traversed

visited? Leaders, designate if a patch has been previously visited

ptype Parameter that differentiates between building and street patches associated with different buildings

procession-route Parameter that tracks every patch passed by the processional leader.

*Table 3: Processional Leader Variables*

**Variable Description**

traveled? Parameter used to keep track of if a leader has traveled to a target yet or not

visited-list Parameter that tracks which patches a leader has already travelled across so that they do not return to patches previously crossed

viewshed-list Parameter that discounts patches that have already been used to test for viewshed

*\*\* The following variables have been kept in the model but are not used in the present simulation \*\**

home-xy Parameter that can be used to set the leader's start patch as the return destination

path Pathfinding parameter, used for determining best route back to the temple using A\* algorithm

*Table 4: Observer Variables*

**Variable Description**

watching Parameter that determines if an observer is actively watching a procession

nearest-group Parameter used to indicate a group of observers in a single location

count-down Parameter used to determine how long an observer has been watching a procession

1. **Process Overview:**

The simulation begins at setup, once the environment has been fully loaded, by having observers populate the environment and then determine a location to watch the procession. This observer procedure occurs separately rather than simultaneously with the processional leader’s movement patterns as a way to ease the length each simulation takes to run. Each observer first determines if it is within a specified radius of the processional leader. If they are, then they have to find a patch to “watch” the procession. Each observer first determines if a nearby group of observers exists, and if there is a group then the observer will move to join that group. If no close group exists, then they find their own “watch” location which is computed as a nearby area that either has a portico or is part of a wide-street. The “watch” location selected by each observer will be the location that the agent remains at for the duration of the simulation. Their presence along the street increases the influence value of the directly surrounding street area.

At each tick, the processional leader evaluates its immediate surroundings and determines where to move on its way to the seafront or river harbour. The leader calculates each step as moving towards patches with the highest influence value, which represent areas corresponding to the highest influence value buildings and with the greatest number of watching spectators. At each step, a temporary target is first selected which represents a patch with the greatest influence accumulation relative to the leaders current position. Then, the viewshed of the temporary target is calculated. If the temporary target is not located behind a building, then the temporary target is set as the primary target and the leader then moves towards it. If the temporary target is not visible, then a viewshed algorithm is run until a target is identified that has both the next highest influence value and that is not blocked by surrounding buildings. This new patch is then set as the next target and the processional leader moves towards it.

Once a patch is walked on by the processional leader, then it cannot be considered as a potential target for the next 500 ticks, called by the procedure (check-leader-list). After 500 ticks the patch may be considered as a potential target. This ensures that the leader does not travel in circles, but it also allows for some backtracking to avoid not being able to find any route to the seafront or river harbour without traversing a portion of previously travelled patches. The patch chosen by the processional leader is recorded as a red line and can be exported at the end of each run.

1. **Design Concepts**

**Basic Principles**

This model tries to mimic processional route selection based on the importance of passing buildings and areas of the cityscape hosting spectators. It goes beyond previous least-cost path or A\* algorithms in which the agent has knowledge of the entire environment for determining a route. Instead at each timestep, the processional leader has to select a movement direction that is influenced by several variables.

**Emergence**

Different routes occur as the position of observers change, forcing the processional leader to select routes not only determined by the influence values of individual buildings. Since observers are randomly placed, they serve to produce route emergence that will differ during each simulation run. If just building influence values are used and no observers are used to populate the simulation, a single route is produced with no aspect of emergence.

**Adaptation**

The processional leader can choose an alternative next-target movement location if the current one is not visible, meaning it’s obscured by surrounding buildings and not directly accessible from their current position. The processional leader and observers can also adapt to obstacles, ensuring that they stay on street patches and do not run into buildings. During each step, the processional leader is trying to move towards a street patch that reflects buildings with the highest influence value and the greatest number of nearby observers.

**Objectives**

The processional leader’s direction is influenced by the weighted values of surrounding buildings and groups of spectators on the side of streets. These two values determine the choice in route through the landscape which travels from the temple to the seafront or river harbour. During each step, this is represented as a target, which the processional leader will move towards until it is reached, at which point a new target will be calculated. The objective of the model is to see how routes change as the building influence values are adjusted and as observers choose different watch locations within the environment.

**Learning**

Patches that are considered by the processional leader as potential movement targets change their influence value if they are walked on so they are not considered as potential targets in future steps. The same occurs during the viewshed calculations, where previous patches are discounted. Agents do not learn in this version of the model.

**Prediction**

The processional leader considers patches within a 200 patch radius when determining their next target. This assumes they have at least some knowledge of the surrounding topography.

**Sensing**

Observers, in their algorithm to determine their watch location draw on distance knowledge of either existing groups of other observers or the definition of streets as wide or having porticoes.

The processional leader uses an algorithm to determine which street patches have the highest influence value relative to their present position while discounting patches previously passed.

**Interaction**

The observers and processional leader interact directly with the street patches as they evaluate their distance and determine their movement target.

**Stochasticity**

Stochasticity exists when observers are included within the model. Where observers are located at the model setup are randomized and change as the number of observers are altered and as their radius distance from the processional leader is increased or decreased. The location where observers choose to ‘watch’ a procession and their added influence value all create stochasticity in the model output results in where the processional leader will choose to go.

**Collectives**

No collectives

**Observation**

The interface outputs the route results as an image export-image file. The results of each weighted simulation run were subsequently aggregated within GIS. Future work would benefit from creating future optimization for route aggregation.

1. **Initialization**

The world is created as a gridded matrix, and does not wrap around. It provides a full view of the extent of Ostia’s excavated and un-excavated cityscape. Patch size and resolution can be changed by the user.

The following parameters need to be set by the user before initializing the model with the button *‘setup’*:

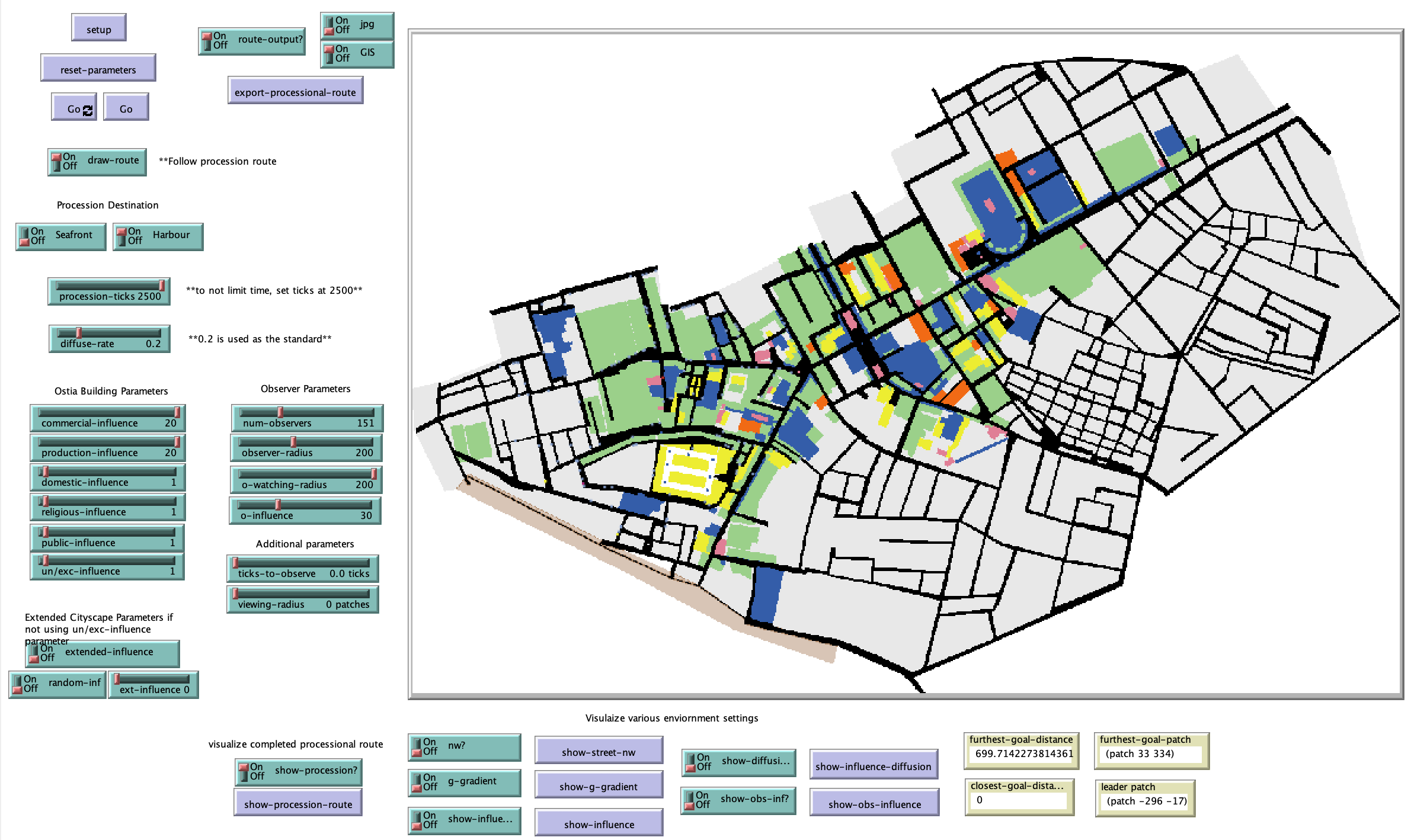


Fig. 2. View of the simulation interface.

*Procession-ticks –* the number of ticks that procession is allowed to run.

*Diffuse-rate* – the rate at which the street influence values are diffused together to average out the building values from either side of the street.

*Draw-route –* a pen-down function that traces the route traversed by the processional leader throughout the simulation.

Procession Destination Selections:

*Seafront* – sets the seafront as the final destination

*Harbour* – sets the river harbour as the final destination

Ostia Building Parameters *–* weights of each building type need to be chosen on a scale of 1–20.

*commercial-influence*

*production-influence*

*domestic-influence*

*religious-influence*

*public-influence*

*un/exc-influence*

*num-observers –* the total number of observers that will populate the model.

*observer-radius* – the radius surrounding the Serapeum that observers will be created.

*o-watching-radius –* the radius from the Serapeum at which observers are classified as “watching” a procession and must then find a location to watch it.

*o-influence* – the influence that each observer exerts on the street network once the find a “watch” location.

Parameters if observers continue to move during simulation, these parameters have been left in but are not used in the present simulation runs, to replicate current results they should be left at 0:

*Ticks-to-observe –* number of ticks that an observer will stay in one position watching the procession before moving

*Viewing-radius –* the distance from the processional leader that will trigger an observer to stop and watch the procession.

1. **Input data**

The environment is based on GIS shapefiles, all of which are located in the folder called *‘data’.* These consist of the shapefiles associated with the various building classifications, the extended cityscape, the street network, the center-line street network, the seafront, street width, and street porticoes. All of these need to be imported into the model using the GIS extension, they are added during the initialization procedure. These files form the environment of the model.

1. **Submodels**

Setting the background environment and loading observers before the model run starts:

* *To create-environment:* 
  + All related GIS files are loaded using the GIS extension. Patches that interest with each GIS file type are displayed using a different color representing different building classification categories. Building patches are given a *ptype* of ‘building’ and an *elevation* of 50. Streets are given a *ptype* ‘street’ and an elevation of 3. Patches consisting of the center-line *street-nw,* wide street patches of *width,* and *porticoes* are all set as ‘true’.
* *To setup-edges*
  + patches associated with individual pcolors and that border a street patch are set with the influence values determined within the interface.
  + These influence values are then expanded with the procedure *expand-influence* which is run 3 times. This serves to transfer the building influence values to the streets located directly in front of the associated buildings.
    - *To expand-influence –* Any patches that are street patches and that are direct neighbors with a single building patch gain the influence value of that building
* *To diffuse-influence*
  + Patches with influence values are diffused using the *diffuse-rate* specified in the interface. This serves to even out the influence values so that the center-line street network values reflect the combined influence value of the buildings on either side of the street. Any patches that are not part of the street-network are set with a new influence value of 0.
* *To create-goal-gradient* 
  + This procedure is used to have the seafront or river harbour as the overall movement goal. All patches that comprise the *street-nw* calculate their distance from either a predefined seafront patch or river harbour patch. These values are then normalized and reversed using the *reverse-gradient* procedure so that the seafront or river harbour patch has the highest value.
  + *To update-influence-with-goal* – the normalized *reverse-gradient* street network values are then added to the influence value of each *street-nw* patch.
* *To setup-observer-influence*
  + Patches associated with wide streets *width\_* or porticoes *portico\_* which border each building classification are given an observer influence value, *obs-inf,* which is determined within the interface as *o-influence.*
* *To display-agents* 
  + The processional leader (see below) and observers are created within the simulation environment
  + *To observers-select-viewing-location*
    - This procedure runs before the processional leader starts moving, and is run 40 times to enable each observer to find and travel to a ‘watch’ location.
    - Observers first *find-watch-location.* This consists of determining if there are any observers within the *o-watching-radius* that have *watching* set as ‘true’. If there are any, then the observer will *find-nearest-group*
      * *To find-nearest-group* – possible groups are determined by the reporter *to-report possible-observer-groups* which consist of groups within a 10 patch radius. The group is then set as the nearest group of observers with *watching* set as ‘true’ relative to a particular observer’s position. The observer will face and then move to join the group using the procedure *join-group,* moving forward one step at a time.
    - If there are no nearby groups of observers, then an observer will determine the best location to ‘watch’ the procession using the procedure *define-watching-location*
      * *To define-watching-location* – possible watching locations are set as any patches with an observer influence value greater than 1, reflecting areas of the street that are wide or have a portico. This is determined using a reporter function of *to-report possible-watching-location*
    - Once all observers find a location, then they add an observer influence value to the centerline street network
      * *To add-to-influence* – patches that are in radius 6 of each observer that are part of the *street-nw* gain the *o-influence* value which is added to their existing influence and reverse gradient values

*To setup-processionals:*

Creates a single processional leader associated with the Serapeum. The coordinates of the temple have been manually entered, but can be changed if another temple starting location is added. When created, the variables of *visited-list* and *viewshed-list* are set to 0. Any patches that are within the *observer-radius* of the leader are defined as in-range (1) or out of range (0).

*To setup-observers:*

Creates the number of observers specified within the interface. When created they are not watching any procession. They also move to a street patch with a *radius-range* of 1, meaning that is within the defined *observer-radius*, determined within the procedure *setup-processionals.*

*To Go:*

If the leader has reached the end of the procession the model stops. If not, the *process* procedure is called.

*To process:*

Calls the sub-procedures of *move-leaders* and *check-leader-list.*

*To move-leaders:*

The leader:

* Determines their movement target
  + *To define-target:*
    - Sets a temporary target, *temp-tar,* as the reporter value *highest-influence.* 
      * *To-report highest-influence –* calculated from street patches within a 200 patch radius of the leader, discounting the leaders’ current patch, and includes patches that have an influence value greater than 0, that are not within the *visited-list* or *viewshed-list.* The reported patch is one that has the highest influence value relative to the leader’s current standing position.
    - The leader sets this temporary patch target *vis?* as false. The leader then faces *temp-tar.* At this point the viewshed or visibility of the target is checked to ensure that the patch is one that can be reached from the current street position and is not behind buildings.
    - *To check-viewshed* is called. If the *temp-tar* is less than one patch away, then *vis?* is set as true and the patch is set as *target.* If the *temp-tar* is more than one patch away, then *mark-viewshed* and *determine-target* are called. These procedures go in a loop until a *temp-target* is determined to be visible.
      * *To mark-viewshed –* all patches between the leader and the *temp-tar* are considered. Visibility is determined based on elevation. Buildings were given an elevation value of 50 during setup and streets an elevation value of 2. For each patch moving away from the leader towards the *temp-tar,* patch calculations occur to determine if the next patch is at a higher elevation or not until the *temp-tar* is reached. If there is no higher elevation, then the *temp-tar* is considered visible and *viewshed?* is set as ‘true’. If a higher elevation patch is encountered, then the *temp-tar* is not visible and *viewshed?* is set as ‘false’.
      * *To determine-target* – if *viewshed?* was true, then *temp-tar* is set as *target* and the *viewshed-list* is cleared. If *viewshed?* is false, then that patch is placed within the *viewshed-list* as a patch that will be discounted. A new *temp-tar* is then calculated discounting the previous one using the reporter *highest-influence* calculation. A new *temp-tar* is created and its viewshed checked. This occurs in a loop until the *viewshed?* comes back as ‘true’.
    - If the model’s total ticks equal the value set for *procession-ticks,* the simulation will stop.
* *To move-leaders* 
  + If *target* is the seafront or river harbour patch, then the simulation stops
  + If the distance to the *target* is <= 0, then a new target is selected using the *define-target* procedure
  + If the distance to the *target* is more than one patch away, then the leader moves towards the target
    - *To travel-leaders* – calls a *move-towards-target* function. If the next movement patch is black, indicating a street, then a *move-function* is called. If the next movement patch is not black, indicating a non-street patch, then an *avoid-function* is called.
      * *To move-function – all-possible-moves* is calculated as neighbors4 in radius of the leader within the street at their current location from a reporter. Leader faces one of the *all-possible-moves* patches relative to their distance and move forward one patch. This patch is then set with the parameters of *visited?* with a value of 1, and *procession-route* with a value of 1. These two values are used to report the full route at the end of the simulation.
      * *To avoid-function* – *all-possible-moves* is calculated and the next patched faced but not moved towards.

*To check-leader-list:*

If more than 500 ticks have occurred, then the first patch is removed from the *visited-list* to increase the speed of the target calculations.