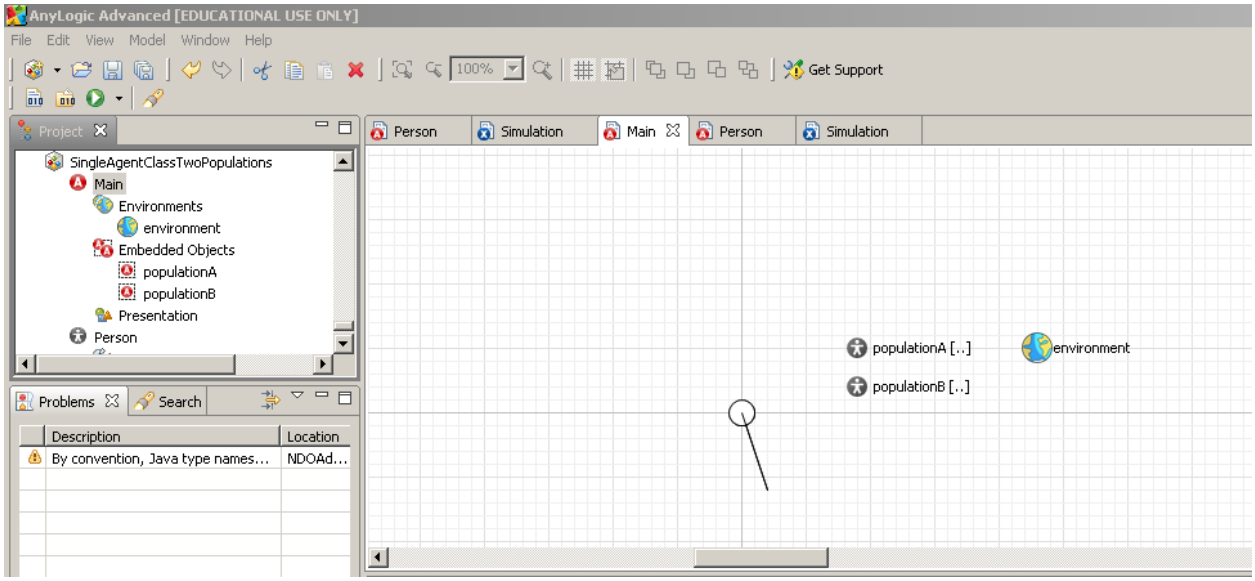
Rating: Basic
Prerequisites: "Building A Minimalist Network-Based Model Framework"
Estimated Time Required: 1 hour

This exercise shows one way of create a very simple meta-population model that includes multiple populations, each of which is well-connected inside, and where there are limited connections existing between populations.

Please note that as the number of such populations rise, and particularly if the count of populations varies over time or in response to interventions, the essentials of the approach below are likely to carry over well, but the details of implementing this approach will need to be modified. Specifically, we suggest considering a single "population" in AnyLogic, with more sophisticated logic to determine location of different members. The learning from this exercise should aid the student in addressing such greater challenges.

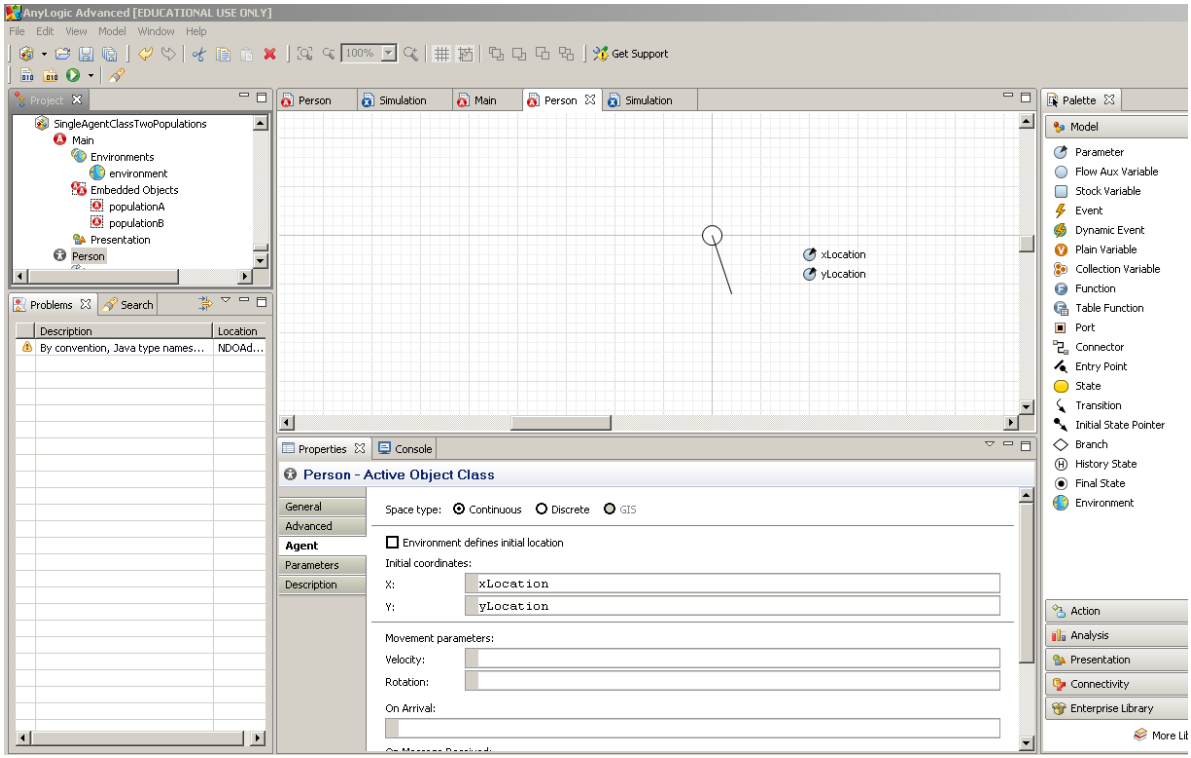
1. Follow the directions in the exercise "Building a Minimalist Network-Based Model Framework", or copy a model previously built using this exercise, and open it up for extension and modification.
2. Save the model to a new model, entitled "SingleAgentClassTwoPopulations"
    a. In anticipation of creating a second population, rename "population" as "populationA" in the canvas for "Main"
    b. In the same manner that you used "drag and drop" to create the original population in Step 0, use the same procedure to create a second population. Click on this population to view its properties. Set these properties as follows
        i. In the name Field, call this population "populationB"
        ii. For the "Environment, enter "Environment"
        iii. In the "Replication" field, enter "50". This means that population B includes 50 people. Recall that "populationA" was set to be larger (100 people).

The canvas for "Main" should now look as follows:



3. The model now has two working populations. Because these are associated with the same environment, these can be connected with one another. However, right now both populations are both displayed in the same area. Suppose we would like them to appear in different spatial locations, such that the populations are more tightly connected within their populations, and less so to the other population. Accomplishing this is a straightforward but multi-step process.
    a. Firstly, we must realize that because the environment doesn't distinguish between these different populations, we cannot depend on it to set the positions of the people. The next few steps will allow us to take more explicit control of the location process.
        i. Double click on "Person" to open up the canvas for the "Person" class.
        ii. Because we will no longer depend on the environment to set our location, we will need to use another mechanism for setting our location. Specifically, we will add two parameters to this class – one for the x coordinate, and one for the y coordinate.
            1. Open the "Model" tab of the "Palette" window. Add a parameter to the canvas of Person. Using the properties window, name this parameter "xLocation. For the default value, enter "0". All other settings can be left as specified.
            2. In a similar way, create another parameter entitled "yLocation", with a default value of 0.
        iii. Click on "Person" to view the canvas for the "Person" class. Click on the "Agent" tab under the "Properties" window. We will set several items here to reflect the fact that we are taking control of agent locations
            1. Currently, the checkbox entitled "Environment defines initial location" should be checked. Uncheck that checkbox.
            2. We will now indicate that the X and Y locations for this person will depend on the two parameters that we created
                a. Under the field for "X", enter "xLocation"
                b. Under the field for "Y", enter "yLocation"

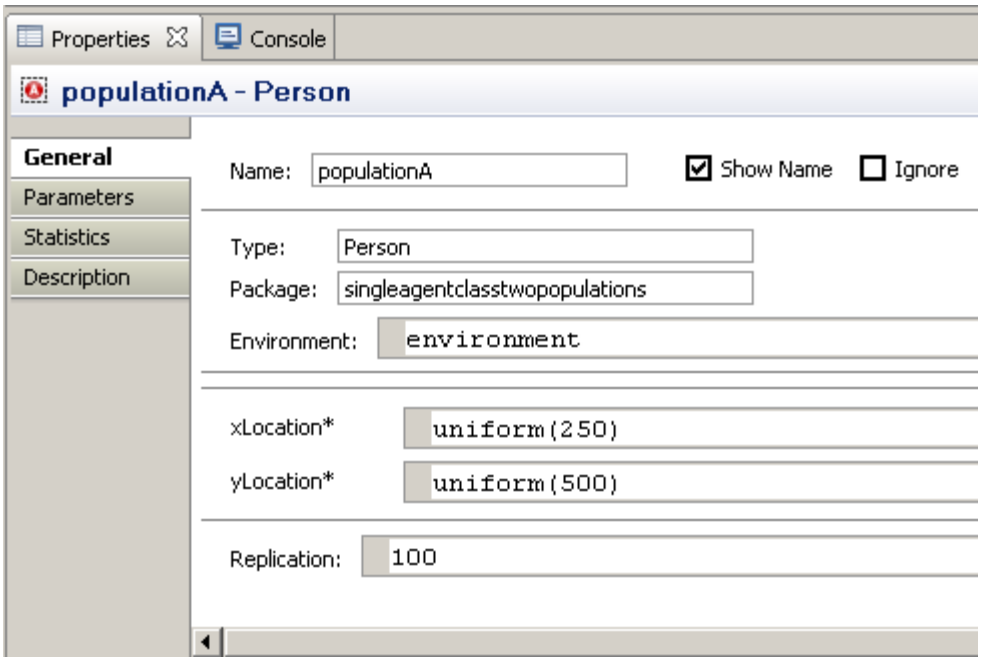
    The "Agent" tab for the properties for "Person" should now look as follows:



b. Based on the above, our agents (Persons) can now have their location explicitly specified. The only thing remaining is to actually specify that location. To illustrate this process, we will set the locations separately for each of populationA and populationB
    i. Double click on "Main" to open up the canvas for the "Main" class.
    ii. Click on "populationA" to display the properties for populationA. Suppose that we wish to have persons associated with populationA displayed in the left side of the running visual area, but both reaching down to the bottom of that area. Recall that both the width and height of the space – as given by the environment – is 500 units.
        1. For the X location for persons in population A, we will therefore choose a value uniformly distributed between 0 and 250. (Note that "250" was chosen because it is half of 500). To specify the value of the "xLocation" field for a given agent, we therefore use the expression "uniform(250)", which draws a value (specific to this person) between 0 and 250.
        2. In this case, we will seek to have the For the Y location for persons in population B, we will choose a value uniformly distributed between 0 and 500 – the entire height of the area. To specify the value of the "yLocation" field for a given agent, we therefore use the expression "uniform(500)".

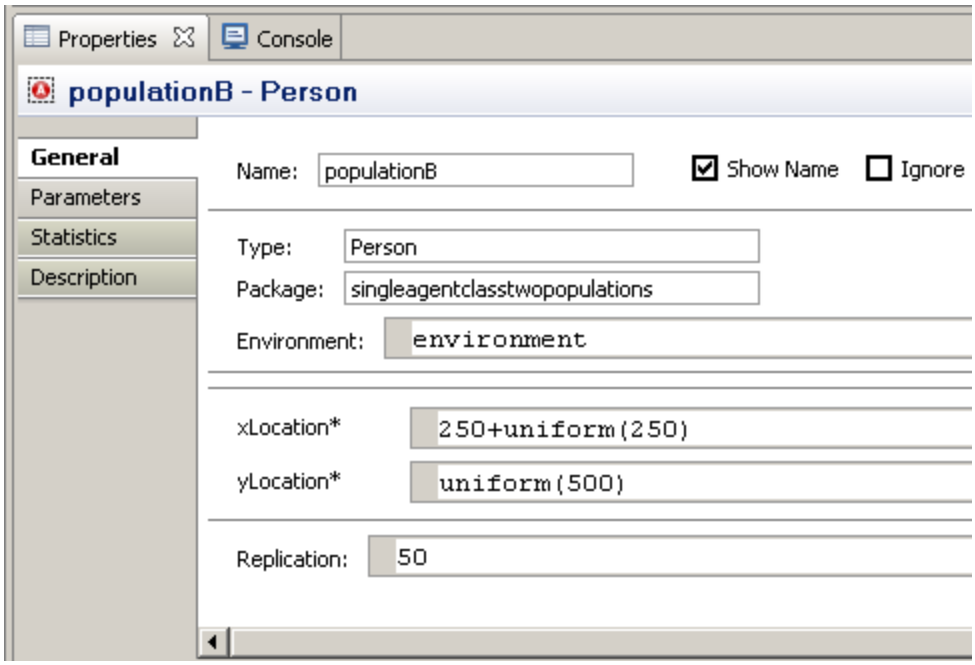
    The properties for "populationA" are as follows:

Exercise Rating: Basic



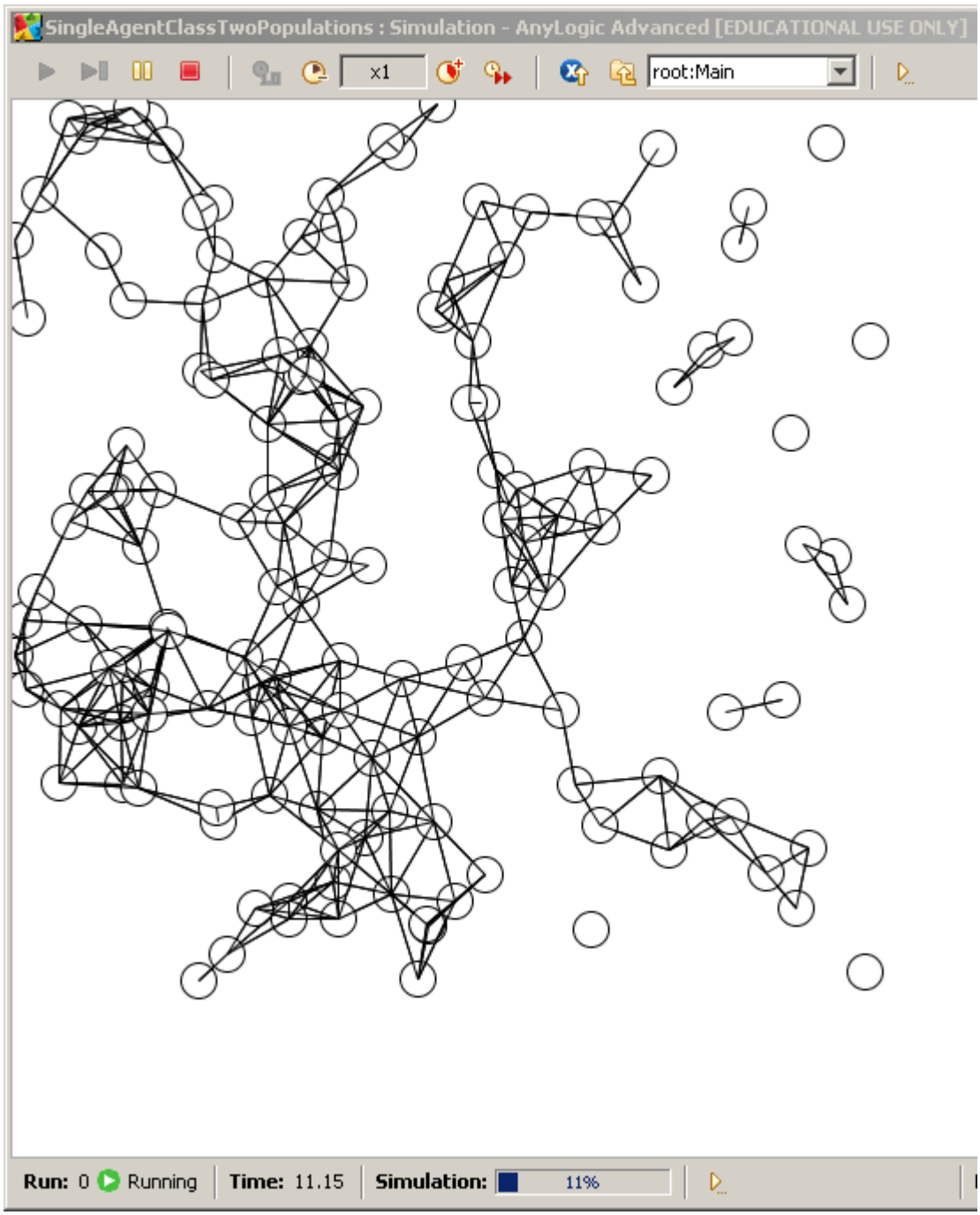
iii. Now click on "populationB" to display the properties for populationB. The logic is similar to the above, except that populationB is to be displayed to the right side of the visual area – that is, in the locations with X coordinates 250 and above.
    1. As was the case for population A, the X location associated with a given person in populationB is to be chosen randomly. However, the random variable associated with such coordinates is different than for populationA. We assume that individuals in population B are uniformly distributed between 250 and 500. We can treat this as the sum of 250 and a value uniformly distributed between 0 and 250. To do so, we enter the value "250+uniform(250)" as the expression specifying the value of "xLocation" for population.
    2. For the value of "yLocation", we enter an identical expression to that used in for populationA – that is, "uniform(500)"

The properties for "populationB" are thus as follows:



You may now run the model. You should see two poulations side-by-side, with limited connections between them.

Exercise Rating: Basic



Exercise Rating: Basic