

Agent Spatial Embedding in 2D Landscapes (Bonus: Discrete Time)

Nathaniel Osgood

MIT 15.879

March 23, 2012

Lecture Outline

- AnyLogic's Spatial embedding types
 - Overview
 - Reminder of continuous space
 - A glimpse of a discrete space & discrete time model
- Agent Mobility

Agent Spatial Embedding

- Spatial embedding of agents is key to
 - Expressing essential dynamics for problems
Locality of influence/Transmission
 - Insight into certain phenomena (spatial concentration, percolation, spatial reference modes)
- Spatial embedding can permit GIS integration

2D Spatial Embedding: Two Options

- Continuous embedding (e.g. Wandering elephants, our built-up model)
 - No physical exclusion: Agents are assumed to be small compared to landscape scale, and exhibit arbitrary spatial density without interfering
 - We have seen this much with distributing agents initially around the space, adding agents in
- Discrete cells (e.g. The Game of Life, Agent-based predator prey, Schelling Segregation)
 - Divided into “Columns” and “Rows”
 - Physical exclusion: Only one agent in a cell at a time

The Locus of Control: Environment

- The Anylogic Environment sets the parameters for the nature of the 2D landscape
 - Width
 - Breadth
 - Continuous vs. Discrete
 - Character of discrete neighbourhoods (cardinal directions vs. Euclidian { N,NE,E,SE,S,SW,W,NW})

Lecture Outline

- AnyLogic's Spatial embedding types
 - √ Overview
 - Reminder of continuous space
 - A glimpse of a discrete space & discrete time model
- Agent Mobility

Continuous Environment

The screenshot displays the AnyLogic Advanced software interface, titled "AnyLogic Advanced [EDUCATIONAL USE ONLY]". The main workspace shows a simulation environment with a grid and various components. A pink box highlights the "environment" component, which is currently selected. The environment is a continuous space with a width of 500 and a height of 500, using a Moore neighborhood type. The environment is currently empty, with a color palette on the right side showing a gradient from green to brown. The interface includes a menu bar (File, Edit, View, Model, Window, Help), a toolbar, and several panels: Project, Search, Properties, Console, and Palette. The Properties panel is open, showing the "environment - Environment" settings. The Palette panel on the right lists various components such as Parameter, Flow Aux Variable, Stock Variable, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, Entry Point, State, Transition, Initial State Pointer, Branch, History State, Final State, and Environment. The Environment component is highlighted in the Palette.

environment - Environment

General

Space type: Continuous Discrete GIS

Width: 500

Height: 500

Columns: 100

Rows: 100

Neighborhood type: Moore

Layout type: User-defined Apply on startup

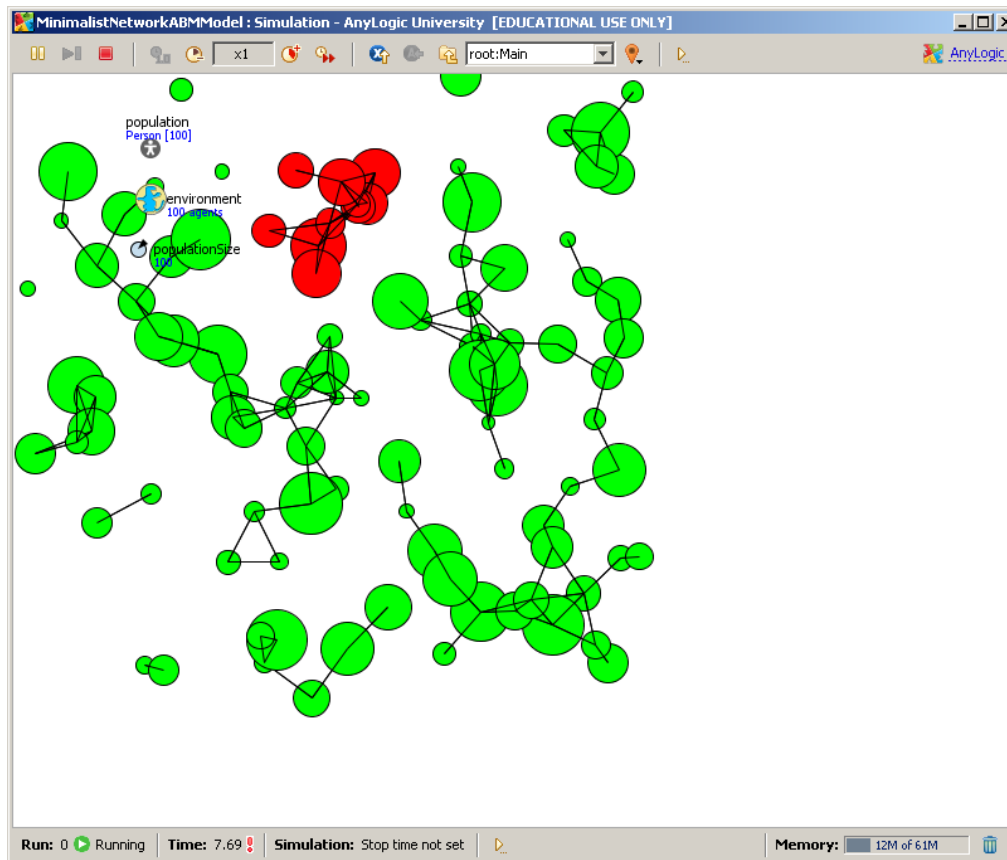
Network type: User-defined Apply on startup

Connections per agent: 2

Capacity per agent: 50

Continuous Environment: Your Model

- We've already seen the continuous embedding in our built up model.



Lecture Outline

- AnyLogic's Spatial embedding types
 - √ Overview
 - √ Reminder of continuous space
 - A glimpse of a discrete space & discrete time model
- Agent Mobility

By Comparison: Discrete Environment

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a grid-based environment with various objects and variables. A color palette is visible on the right side of the workspace. The Properties panel at the bottom is open to the 'environment - Environment' section, where the 'Space type' is set to 'Discrete'. The 'Advanced' section of the Properties panel is highlighted, showing the following settings:

Property	Value
Space type	Discrete
Width	500
Height	500
Columns	100
Rows	100
Neighborhood type	Moore

The Properties panel also shows 'Layout type' and 'Network type' both set to 'User-defined', and 'Connections per agent' set to 2. A blue text overlay on the right side of the Properties panel reads: "Note extra presence of 'Columns' and 'Rows'".

The interface includes a Project browser on the left, a Palette on the right, and a Console at the bottom. The main workspace contains several objects, including 'SmokingInitiationByAgeAndSmokingStatusForSexualActivityGroup1', 'makeUpVegetation', 'environment', 'vegetationToColor', 'placeElephants', 'DisplacementTable', 'altitude', 'AngleTable', 'altcolor', 'viewVegetation', 'vegetation', 'DistrDisplacement', 'mapDrawing', 'DistrAngle', 'altitudesDrawn', and 'updateVegetation'.



Hands on Model Use Ahead



Load AnyLogic Sample Model: The Game
of Life

The “Game” of Life: Background

- Invented in 1970 by Mathematician Conway (modifying ideas from Von Neumann)
- Inspiration: Lifecourse of cells
 - Key dichotomy: A space contains a living element or not
 - Stylized rules for birth, death
- Cellular automaton: Uses Discrete Time (Steps) & Discrete Space (Cells) with evolving cell state
- Deterministic rules
- Illustrates the emergence of tremendous complexity from very simple rules
 - Computationally universal

The Behavioral Rules of the Game of Life

- Cells are viewed as surrounded by 4 neighbors (in cardinal directions)
- Living cells require some neighboring empty space, but also some proximity to nearby living cells
- Birth: An empty cell becomes occupied if it has an “ideal nurturing environment around it (3 surrounding cells)
- An existing cell dies if
 - Too isolated: It has too few neighbors (1 or 0)
 - Too crowded: It is surrounded by other cells (4 surrounding cells)
- No mobility: Cells are born, live and die in same location

Open "Main" Class

Scroll Left to See Population & Environ.

The screenshot displays the AnyLogic University software interface. The main workspace shows a simulation titled "The Game of Life" at "Step: 2". The workspace contains a red rectangular area representing the simulation environment, with a yellow rectangular area inside it. To the left of the workspace, there are two objects: "cells [...]" and "environment".

The left sidebar shows the project hierarchy for "The Game of Life":

- IMainAction
- Simulation: Main
- HierarchicalCityPopulationModelW
- City
- Main
- Person
- Baseline: Main
- RecoveryTime10: Main
- RecoveryTime100: Main
- The Game of Life
 - Cell
 - Variables
 - alive
 - nAliveAround
 - neighbors
 - Functions
 - toggleState
 - Presentation
 - Main
 - Environments
 - Embedded Objects
 - Presentation
 - Simulation: Main

The bottom panel shows the "Main - Active Object Class" properties:

- Name: Main
- Ignore:
- Agent: Agent Generic
- Startup code:
- Destroy code:

The right sidebar shows the "Palette" with various components:

- General
 - Parameter
 - Event
 - Dynamic Event
 - Variable
 - Collection
 - Function
 - Table Function
 - Schedule
 - Port
 - Connector
 - Environment
 - Agent Population
- System Dynamics
- Statechart
- Actionchart
- Analysis
- Presentation
- 3D
- Controls
- Connectivity
- Enterprise Library
- Pedestrian Library
- Rail Library
- Road Traffic Library - Preview
- Pictures
- 3D Objects
- Palettes...

The bottom status bar shows "Selection" and coordinates "X=288, Y=421".

Imposing the Regular 2D Structure

The screenshot displays the AnyLogic University software interface. The main window shows a simulation titled "The Game of Life" at "Step: 2", featuring a yellow rectangular area on a grid. The left sidebar contains a project tree with "The Game of Life" selected, showing sub-elements like "Cell", "Variables", and "Functions". The bottom-left pane shows a "Problems" table with "No problems" listed. The bottom-right pane shows the "environment - Environment" properties, with the "Advanced" tab selected. The "Space type" is set to "Discrete2D", and "Columns" and "Rows" are both set to 100. A red arrow points from the "100x100 grid defined here" text to the "Columns" and "Rows" fields. A blue arrow points from the "Indicated that cells should be laid out in a regular grid in space" text to the "Layout type" dropdown, which is set to "Arranged".

100x100 grid defined here

Indicated that cells should be laid out in a regular grid in space

environment - Environment

Space type: Continuous2D Continuous3D Discrete2D GIS

Width: 500

Height: 500

Z-Height: 0

Columns: 100

Rows: 100

Neighborhood type: Moore

Layout type: Arranged Apply on startup

Network type: User-defined Apply on startup

Connections per agent: 2

Connection range: 50

Neighbor link fraction: 0.95

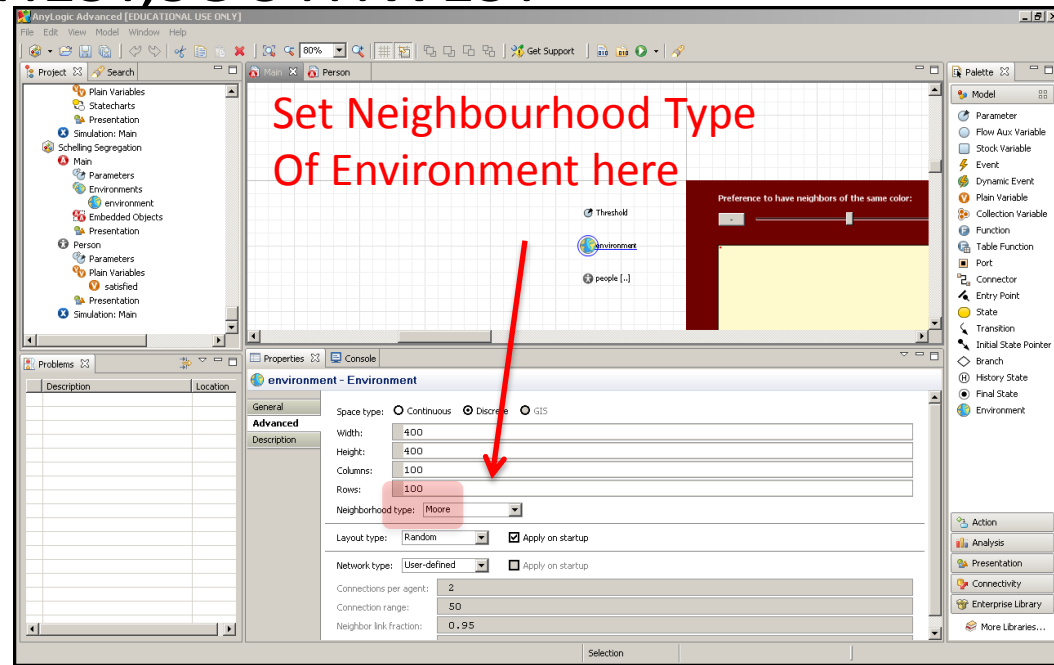
M: 10

Environment: Enabling Discrete Space (Cells)

The screenshot shows the AnyLogic software interface. The main workspace displays a simulation titled "The Game of Life" at "Step: 2", showing a grid of cells. The left sidebar shows a project tree with "The Game of Life" selected, containing a "Cell" object with variables like "alive", "nAliveAround", and "neighbors", and functions like "toggleState". The bottom panel shows the "environment - Environment" properties window. In the "General" tab, the "Space type" is set to "Discrete2D" (highlighted with a red arrow). Other settings include "Width: 500", "Height: 500", "Z-Height: 0", "Columns: 100", "Rows: 100", and "Neighborhood type: Moore" (highlighted with a blue arrow). A red text box on the left says "Discrete2D selected" with an arrow pointing to the "Discrete2D" radio button. A blue text box on the right says "Defines logical neighborhood (here, each cell has 4 neighbors)" with an arrow pointing to the "Moore" dropdown.

Neighbourhood Models

- Moore: Cardinal directions
 - NORTH,SOUTH,EAST, WEST
- Euclidean
 - NORTH, SOUTH, EAST, WEST, NORTHEAST, NORTHWEST, SOUTHEAST,SOUTHWEST



Population: One Cell Agent per Grid Point

The screenshot displays the AnyLogic University interface for configuring a simulation. The central workspace shows a grid titled "The Game of Life" with "Step: 2" indicated in red. A yellow rectangular area represents the grid, and a red border highlights the current step. The left sidebar shows a project tree with "The Game of Life" selected, and "Cell" is the active agent type. The bottom panel shows the "cells - Cell" configuration window. In the "General" tab, the "Initial number of objects" is set to 10000, which is highlighted with a red box and a red arrow pointing to the text "10,000 (= 100*100) agents" written in red below it. The "Replicated" checkbox is checked, and the "Package" is set to "the_game_of_life". The right sidebar shows a palette of simulation components, including "Agent Population".

cells - Cell

General

Name: cells Show name Ignore Public Show at runtime

Type: Cell

Environment: environment

Package: the_game_of_life

Replicated

Initial number of objects: 10000

Optimize: Access by index (ArrayList) Add/remove operations (LinkedHashSet)

10,000 (= 100*100) agents

View the “Cell” Class

This class represents each cell in the entire space – whether it is alive or not

Cell - Active Object Class

General

Advanced

Agent

Preview

Description

Space type: Continuous2D Continuous3D Discrete2D GIS

Environment defines initial location

Initial coordinates:

Column:

Row:

On arrival:

On message received:

Forward message to:

Statecharts

Cell Variables: “alive”

The screenshot displays the AnyLogic University interface. On the left, a project tree shows the hierarchy: Simulation: Main > Cell > Variables > alive. The main workspace shows a grid with several variables: 'alive' (a blue circle with a 'V'), 'neighbors' (a brown circle with a 'V'), 'nAliveAround' (a brown circle with a 'V'), and 'toggleState' (a blue circle with an 'F'). A red arrow points from the text 'Boolean (true/false) variable' to the 'boolean' type selection in the properties panel. A blue arrow points from the text 'Name would be clearer as “isAlive”' to the 'Name: alive' field. A green arrow points from the text '10% initial likelihood of being occupied' to the 'randomTrue(0.1)' initial value field.

Boolean (true/false) variable

Name would be clearer as “isAlive”

10% initial likelihood of being occupied

alive - Variable

General

Name: alive Show name Ignore Show at runtime

Access: public Static Constant Save in snapshot

Type: boolean int double String Other: boolean

Initial value: randomTrue(0.1)

Use Units Unit:

Cell Variables: “neighbors”

The screenshot displays the AnyLogic University interface. On the left, a project tree shows a hierarchy: IMainAction > Simulation: Main > HierarchicalCityPopulationModelW > City > Main > Person > Cell > Variables > neighbors. The main workspace shows a grid with several variables: 'alive' (orange circle), 'neighbors' (orange circle with a red arrow pointing to it), 'nAliveAround' (orange circle), and 'toggleState' (blue circle). A red text box is overlaid on the workspace, stating: "This will reference a Collection (“Array”) that Contains references to each neighbor of the current cell".

At the bottom, the Properties window for the 'neighbors' variable is open. The 'General' tab shows the following settings:

- Name: neighbors
- Show name:
- Ignore:
- Show at runtime:
- Access: public
- Static:
- Constant:
- Save in snapshot:
- Type: Other: Agent[] (highlighted with a green box and a green arrow pointing to it)
- Initial value: [Empty text box]
- Use Units:
- Unit: [Empty text box]

At the bottom of the Properties window, a green text box states: "Reference to the collection has an “Array” type".

Cell Variables: “nAliveAround”

This will count the number Of neighbors around this cell that are alive at the current time (i.e. during the current step)

The “type” of this variable is an “integer”

AnyLogic University [EDUCATIONAL USE ONLY]
File Edit View Draw Model Tools Help
100%
Get Support...

Projects Simulation Main Cell

IMainAction
Simulation: Main
HierarchicalCityPopulationModelW
City
Main
Person
Baseline: Main
RecoveryTime10: Main
RecoveryTime100: Main
The Game of Life
Cell
Variables
alive
nAliveAround
neighbors
Functions
toggleState
Presentation
Main
Environments
Embedded Objects
Simulation: Main

Properties Console
nAliveAround - Variable

General
Name: nAliveAround Show name Ignore Show at runtime
Access: public Static Constant Save in snapshot
Type: boolean int double String Other: int
Initial value:
 Use Units Unit:

Palette
General
Parameter
Event
Dynamic Event
Variable
Collection
Function
Table Function
Schedule
Port
Connector
Environment
Agent Population
System Dynamics
Statechart
Actionchart
Analysis
Presentation
3D
Controls
Connectivity
Enterprise Library
Pedestrian Library
Rail Library
Road Traffic Library - Preview
Pictures
3D Objects
Palettes...

nAliveAround - Variable Selection X=362, Y=173

Visual Representation of Cell (Click on Cell Icon at Origin)

The screenshot displays the AnyLogic software interface. On the left, the 'Projects' tree shows a hierarchy including 'Cell' with sub-items 'Variables' (alive, nAliveAround, neighbors) and 'Functions' (toggleState). The main workspace shows a grid with a red rectangle at the origin (0,0). A red arrow points to this rectangle with the text 'Select this item'. Below the grid, the 'Properties' window is open for a 'rectangle - Rectangle' object. A blue arrow points from the text 'Selects appearance depending on whether alive or not' to the 'Fill color' field, which contains the expression 'alive ? red : lemonChiffon'. The 'Dynamic' section of the properties window is highlighted.

Select this item

Selects appearance depending on whether alive or not

rectangle - Rectangle

General	Replication:
Advanced	Visible:
Dynamic	
Description	
	X: <input type="text"/>
	Y: <input type="text"/>
	Z: <input type="text"/>
	Fill color: <input type="text" value="alive ? red : lemonChiffon"/>
	Width: <input type="text"/>
	Height: <input type="text"/>
	Z-Height: <input type="text"/>

Cell Update Logic ("Agent" Properties of "Cell")

The screenshot displays the AnyLogic University interface. On the left, the Project Explorer shows a tree structure with 'Cell' selected under 'The Game of Life'. The main workspace shows a statechart with variables 'alive', 'neighbors', and 'nAliveAround', and a function 'toggleState'. The Properties window for 'Cell - Active Object Class' is open, showing the logic for 'On message received', 'Forward message to', 'On before step', and 'On step'.

Cell - Active Object Class

General

Advanced

Agent

Preview

Description

On message received:

Forward message to:

Statecharts

On before step:

```
//count the number of alive neighbors  
nAliveAround = 0;  
for( Agent a : neighbors )  
    if( ((Cell)a).alive )  
        nAliveAround++;
```

On step:

```
//evaluate the next state:  
//alive cell stays alive if it has 2 or 3 alive neighbors  
//dead cell becomes alive if there are exactly 3 neighbors  
alive = alive && ( 2 <= nAliveAround && nAliveAround <= 3 ) ||  
        nAliveAround == 3;
```


Two Key Models of Time in Anylogic: Continuous (Asynchronous) Time

- This is what we have dealt with to this point
- Here, every agent is updated at a different time, according to events
- No two agents are typically likely to be updated at exactly the same time during most of model execution, so when considering the state of other agents they “see” the last situation where the other agent has been updated

Two Key Models of Time in Anylogic:

Discrete (Synchronous) Time

- Here, agents all change in lockstep, separated by fixed “time steps”
- When computing agent behavior (to determine agent state in the next timestep), our enquiries about agent state (e.g. using *getAgentAtCell* or *getAgentNextToMe*) need to ask about the situation ***in the current timestep***
 - We gather needed information regarding current state in “On Before Step”, and changes are performed in “On Step”.
- This is similar to what we saw in System Dynamics – the changes over the next small interval of time (Δt) depend on the current value of the stocks
 - These changes are then applied at once, and all stocks are updated

Enabling Discrete (Synchronous) Time

- When enable the steps, the various handlers for synchronized time (e.g. “On before step”, “On step”, “On after step”) etc.) are executed
 - Both environment and agents have “On before step” and “On after step” handlers
 - “On before step” for environments is executed before the corresponding method for agents
 - “On after step” for environments is executed after the corresponding method for agents
- Synchronous time can be enabled via the **environment** “General” page
 - Click checkbox “Enable steps”

Environment: Enabling Discrete Time

The screenshot displays the AnyLogic University software interface for a simulation project titled "The Game of Life". The main workspace shows a grid environment with a red border and a yellow center, labeled "The Game of Life" and "Step: 2". The left sidebar shows a project tree with "The Game of Life" selected, containing elements like "Cell", "Variables", "Functions", and "Presentation". The bottom panel shows the "environment - Environment" properties, with the "Enable steps" checkbox checked. A red arrow points to this checkbox, and a red text box below it says "Notice checkmark to enable discrete time (steps)".

Notice checkmark to enable discrete time (steps)

Cell Update Logic ("Agent" Properties of "Cell")

1) On Before Step
(collects information)

2) On Step (Acts on
Collected Information)

Cell - Active Object Class

On message received:

Forward message to:

Statecharts

On before step:

```
//count the number of alive neighbors
nAliveAround = 0;
for( Agent a : neighbors )
    if( ((Cell)a).alive )
        nAliveAround++;
```

On step:

```
//evaluate the next state:
//alive cell stays alive if it has 2 or 3 alive neighbors
//dead cell becomes alive if there are exactly 3 neighbors
alive = alive && ( 2 <= nAliveAround && nAliveAround <= 3 ) ||
nAliveAround == 3;
```

On Before Step: Collecting the Information

This records a running count of # seen so far (initially 0)

On before step:

```
//count the number of alive nei  
nAliveAround = 0;  
for( Agent a : neighbors )  
    if( ((Cell)a).alive )  
        nAliveAround++;
```

2) Loops through each of the neighbors. Every time we see a live neighbor, increment the count of alive neighbors

On Step: Performing the Update based on Observed Information

Reminder: This is the information collected in “On Before Step”

On step:

```
//evaluate the next state:  
//alive cell stays alive if it has 2 or 3 alive neighbors  
//dead cell becomes alive if there are exactly 3 neighbors  
alive = alive && ( 2 <= nAliveAround && nAliveAround <= 3 ) ||  
nAliveAround == 3;
```

Here, we are updating our aliveness status (represented by the “alive” variable) based on our current status & characteristics of the local environment.

Obtaining the List of Neighboring Cells at Startup

The screenshot displays the AnyLogic University software interface. The main workspace shows a grid with several variables: 'alive' (orange circle), 'neighbors' (orange circle), 'nAliveAround' (orange circle), and 'toggleState' (blue circle). A red text overlay reads: "For performance reasons, this obtains a reference to a set of neighboring cells, and stores it in the variable 'neighbors'". A red arrow points from this text to the 'Startup code' field in the 'Cell - Active Object Class' properties window. The 'Startup code' field contains the following code:

```
//initialize the array of neighbors - it won't change over time  
neighbors = getNeighbors();
```

The 'Cell - Active Object Class' properties window is open, showing the 'General' tab. The 'Name' field is set to 'Cell'. The 'Agent' checkbox is checked, and the 'Generic' checkbox is unchecked. The 'Startup code' field is highlighted with a pink background. The 'Destroy code' field is empty.

The left sidebar shows the project hierarchy, including 'Main', 'Simulation: Main', 'HierarchicalCityPopulationModelW', 'City', 'Main', 'Person', 'Baseline: Main', 'RecoveryTime10: Main', 'RecoveryTime100: Main', 'The Game of Life', 'Cell', 'Variables', 'alive', 'nAliveAround', 'neighbors', 'Functions', 'toggleState', 'Presentation', 'Main', 'Environments', 'Embedded Objects', 'Presentation', 'Simulation: Main'. The right sidebar shows the 'Palette' with various components like 'Parameter', 'Event', 'Dynamic Event', 'Variable', 'Collection', 'Function', 'Table Function', 'Schedule', 'Port', 'Connector', 'Environment', 'Agent Population', 'System Dynamics', 'Statechart', 'Actionchart', 'Analysis', 'Presentation', '3D', 'Controls', 'Connectivity', 'Enterprise Library', 'Pedestrian Library', 'Rail Library', 'Road Traffic Library - Preview', 'Pictures', '3D Objects', and 'Palettes...'. The bottom left shows the 'Problems' window with 'No problems'.

Running the Model

The Game of Life Step: 15

Click on a cell to toggle its status

Run: 0 ▶ Running | Time: 15.80 | ▶ | EPS: 2 | FPS: 12.0 | 8.7 sec