

Agent Spatial Embedding in 2D Landscapes

Nathaniel Osgood

Using Modeling to Prepare for Changing
Healthcare Needs

Duke-NUS

April 16, 2014

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

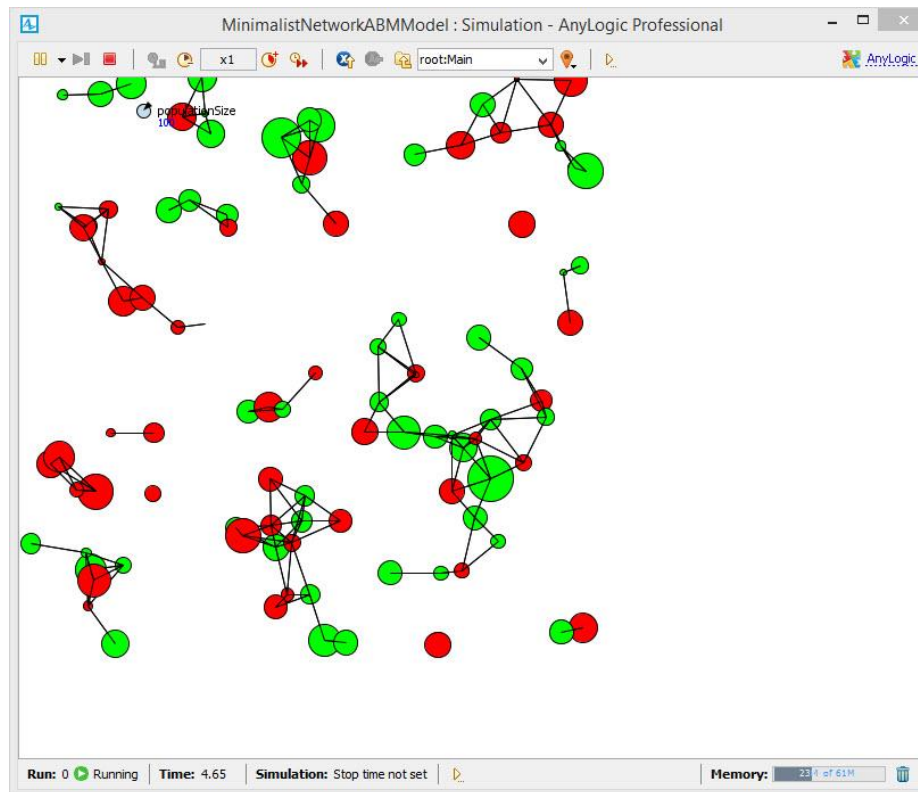
- AnyLogic's Spatial embedding types
 - √ Overview
 - Reminder of continuous space
 - A glimpse of a discrete space & discrete time model
- Agent Mobility

Continuous Environment

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a simulation model for "Wandering Elephants" on a grid. The model includes several components: "NumberOfElephants", "makeUpAltitudes", "elephants [...]", "altitudeToColor", "makeUpVegetation", "vegetationToColor", "placeElephants", "DisplacementTable", "altcolor", "altitude", "AngleTable", and "viewVegetation". A "View" window is open, showing a 2D plot of the environment with a color gradient representing altitude, ranging from 0 to 80. The "Properties" panel is open, showing the "Main - Agent Type" configuration. Under "Environment for other agents", there is a section for "Select the agents you want to place in the environment:" with a blue selection area and a "Z-Height" input field set to 0. The "Palette" panel on the right shows the "Statechart" palette with various elements like "Statechart Entry Point", "State", "Transition", "Initial State Pointer", "Branch", "History State", and "Final State". The bottom status bar indicates "Time units: days".

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 Professional software interface. The main workspace shows a grid-based environment with several objects and processes:

- NumberOfElephants**: A process icon.
- makeUpAltitudes**: A process icon.
- elephants [.]**: A state icon.
- altitudeToColor**: A process icon.
- makeUpVegetation**: A process icon.
- vegetationToColor**: A process icon.
- View**: A rectangular area representing the environment view, containing a small landscape with a brown area and a yellow area, both labeled with the number 80.

The left sidebar shows the project structure:

- Wandering Elephants*
- Elephant
- Main
- Simulation: Main

The bottom panel shows the **Main - Agent Type** configuration:

- Movement parameters**
- Environment for other agents**
 - Select the agents you want to place in the environment:
 - elephants
- Neighborhood type: Moore
- Layout type: User-defined Apply on startup

Time units: days

Note extra presence of "Columns" and "Rows"



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

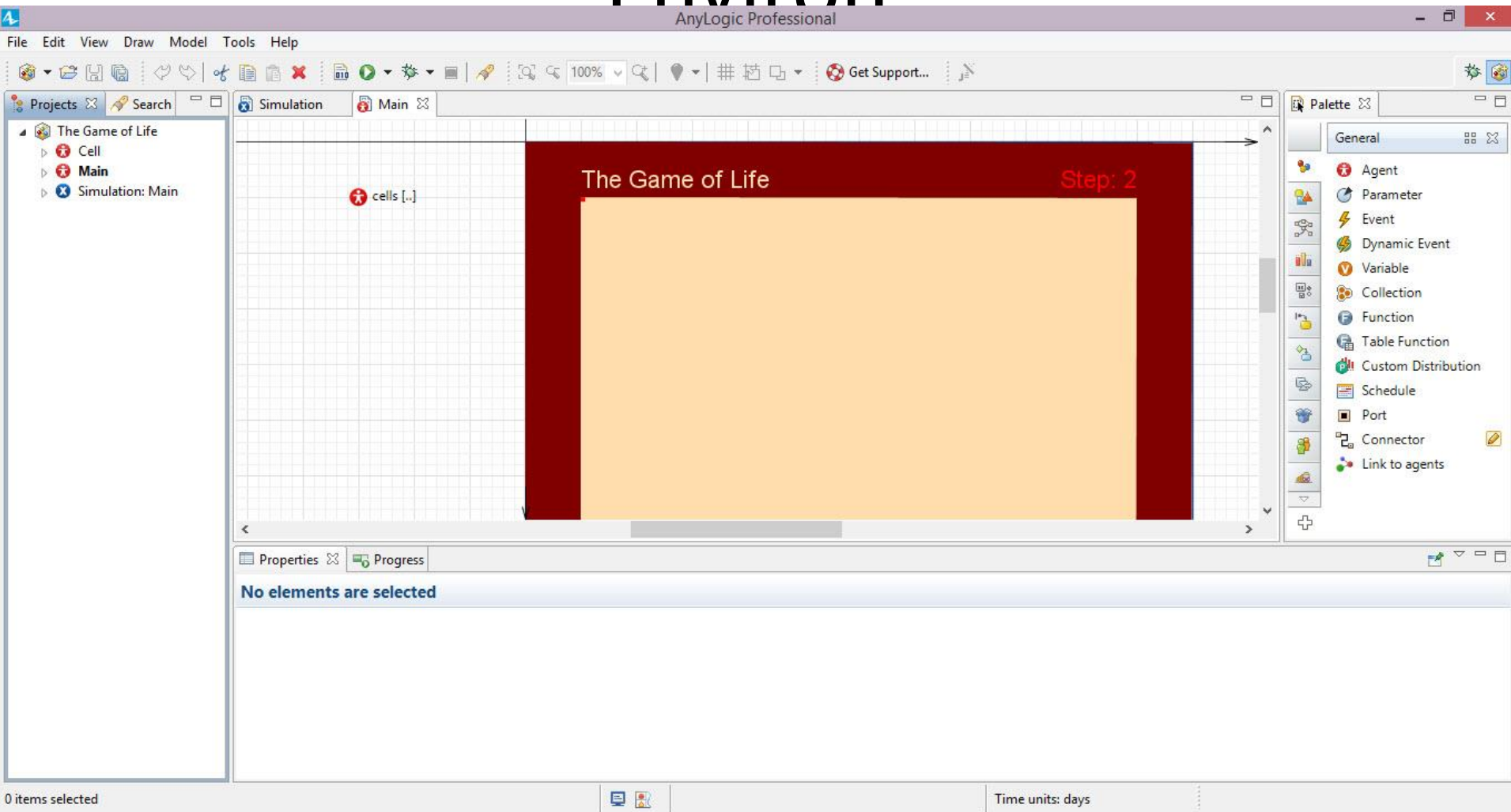
The Behavioral Rules of the Game of

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

Open Main Class

Scroll Left to See Population & Environ



Imposing the Regular 2D Structure

The screenshot displays the AnyLogic Professional interface. The main workspace shows a simulation titled "The Game of Life" at "Step: 2", with a grid of cells. The "Properties" panel is open to "Main - Agent Type", showing the "Environment for other agents" section. The "cells" agent is selected, and the "Space type" is set to "Discrete". The "Space dimensions" are defined as 500 columns and 500 rows. The "Neighborhood type" is set to "Moore".

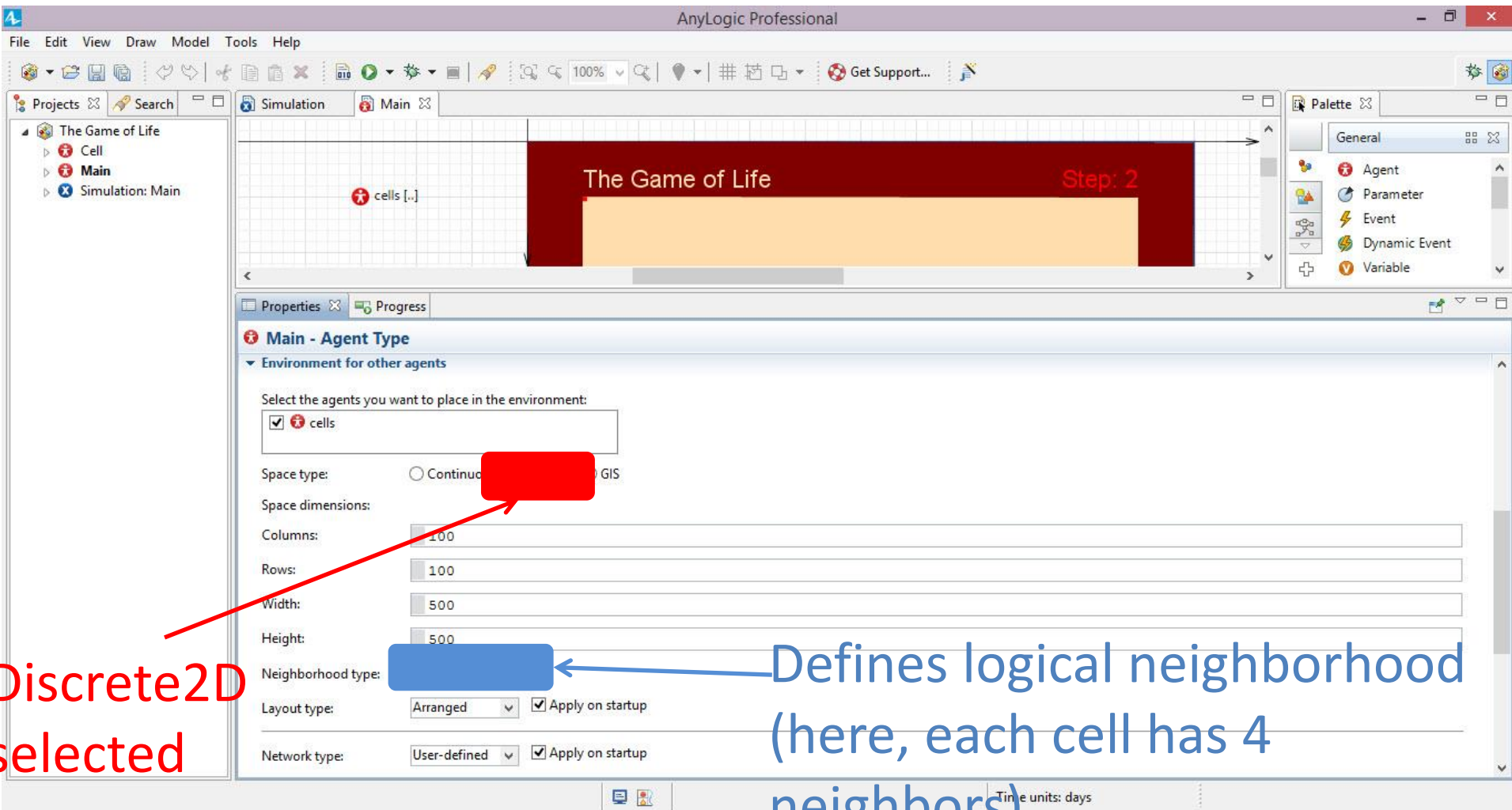
Annotations on the image include:

- A red box highlights the "Rows" field, with a red arrow pointing to it from the text "100x100 grid defined here".
- A blue box highlights the "Layout type" field, with a blue arrow pointing to it from the text "Indicated that cells should be laid out in a regular grid in space".

100x100
grid defined
here

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

Environment: Enabling Discrete Space (Cells)



Discrete2D selected

Defines logical neighborhood (here, each cell has 4 neighbors)

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 Professional interface for a simulation titled "The Game of Life". The main workspace shows a grid with a central orange square labeled "The Game of Life" and "Step: 2". A red box highlights the "Initial number of agents" field in the "cells - Cell" properties panel, which is set to 10,000. A red arrow points from the text "10,000 (= 100*100) agents" at the bottom to this field. The "Initial location" section is also visible, with a note that settings are applied only if the "User-defined" layout type is set.

10,000 (= 100*100) agents

View the “Cell” Class

The screenshot displays the AnyLogic Professional interface. The main workspace shows a class diagram for the 'Cell' class, which includes the following attributes and methods:

- Attributes: `alive`, `neighbors`, `nAliveAround`
- Method: `toggleState`

Overlaid on the diagram is the text: "This class represents each cell in the entire space – whether it is alive or not".

The Properties panel at the bottom shows the configuration for the 'Cell - Agent Type':

- Name: Cell Ignore
- Agent actions: (empty)
- Entity actions:
 - Use in flowcharts as: Entity
 - On enter flowchart block: (empty text box)
 - On exit flowchart block: (empty text box)
 - On seize resource: (empty text box)

The status bar at the bottom indicates 'Time units: days'.

Cell Variables: "alive"

The screenshot shows the AnyLogic Professional interface. The main workspace displays a grid with several variables: 'alive', 'neighbors', 'aliveAround', and 'toogleStat'. The 'alive' variable is highlighted with a red circle. A red arrow points from the text 'Boolean (true/false) variable' to the 'alive' variable in the workspace. Another red arrow points from the same text to the 'Type' field in the Properties panel, which is highlighted in red. A blue arrow points from the text 'Name would be clearer as "isAlive"' to the 'Name' field in the Properties panel. A green arrow points from the text '10% initial likelihood of being occupied' to the 'Initial value' field in the Properties panel, which is highlighted in green. The Properties panel shows the following details for the 'alive' variable:

- Name: [Redacted]
- Visible: yes
- Type: [Redacted]
- Initial value: [Redacted]

Time units: days

Cell Variables: “neighbors”

This will reference a Collection (“Array”) that Contains references to each neighbor of the current cell

neighbors - Variable

Name: neighbors Show name Ignore

Visible: yes

Type: [Redacted]

Initial value: [Redacted]

Advanced

Description

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”

The Game of Life

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects Search

Simulation Main Cell

The Game of Life*

- Cell
- Main
- Simulation: Main

alive

neighbors

toggleState

Cell Variables: "alive"

Properties Progress

nAliveAround - Variable

Name: nAliveAround Show name Ignore

Visible: yes

Type: integer

Initial value:

Advanced

Description

Time units: days X=...-6

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

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a grid with a red square icon at the origin (0,0). A red arrow points to this icon with the text "Select this item". Below the grid, a list of variables is visible: "alive", "neighbors", "nAlive_neighbors", and "toggleState". A blue arrow points from the text "Selects appearance depending on whether alive or not" to the "alive" variable. The Properties panel at the bottom shows the selected "rectangle - Rectangle" with various settings. The "Appearance" section is expanded, showing a blue fill color swatch, "Line color: No color", and "Line width: 1 pt". The Palette on the right lists various modeling elements like Agent, Parameter, Event, etc. The status bar at the bottom indicates "Time units: days" and "X=...16".

Select this item

Selects appearance depending on whether alive or not

rectangle - Rectangle

Name: rectangle Ignore Visible on upper level Icon Lock

Visible: no yes

Appearance

Fill color:

Line color: No color

Line width: 1 pt

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

The screenshot displays the AnyLogic Professional interface for configuring a 'Cell' agent type in a simulation titled 'The Game of Life'. The main workspace shows a grid with two variables, 'alive' and 'neighbors', represented by orange circles. The 'Properties' pane on the right is set to 'Cell - Agent Type' and contains the following logic:

On arrival to target location:

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;
```

The bottom status bar shows 'Time units: days' and 'X=...42'.

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)

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 class's “General” page
 - Click checkbox “Enable steps”

Environment: Enabling Discrete Time

The screenshot displays the AnyLogic Professional interface for a simulation titled "The Game of Life". The main workspace shows a grid with two variables, "alive" and "neighbors", plotted over time. The Properties panel is open to the "Main - Agent Type" section, where various simulation parameters are configured. A red box highlights the "Step duration (in model time units)" field, which is set to 1.0. A red arrow points from the text below to this field.

Properties Progress

Main - Agent Type

Columns: 100

Rows: 100

Width: 500

Height: 500

Neighborhood type: Moore

Layout type: Arranged Apply on startup

Network type: User-defined Apply on startup

Step duration (in model time units): 1.0

On before step:

Notice checkmark to enable
discrete time (steps)

Time units: days

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

1) On Before Step
(collects information)

2) On Step (Acts on
Collected Information)

Cell - Agent Type

On arrival to target location:

On before step:

On step:

Entity actions

Time units: days X=...42

On Before Step: Collecting the Information

On before step:

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

```
//count the number of alive nei
```

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 & neighbors == 2 || neighbors == 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

For performance reasons, this obtains a reference to a set of neighboring cells, and stores it in the variable "neighbors"

Cell - Agent Type

Name: Cell Ignore

Agent actions

On startup:  neighbors - it won't change over time

On destroy:

On arrival to target location:

On before step:

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

Time units: days

Running the Model

The screenshot shows a simulation window titled "The Game of Life : Simulation - AnyLogic Professional". The window includes a toolbar with icons for simulation control (stop, play, step forward, step back, search, zoom) and a zoom level of "x2". The main area displays a 2D grid with red squares and circles representing the state of the simulation at "Step: 7". The text "The Game of Life" is on the left and "Step: 7" is on the right. A red instruction at the bottom left says "Click on a cell to toggle its status". The bottom status bar shows "Run: 0" with a green running icon, "Time: 7.50", "EPS: 2", "FPS: 40.0", and "3.1 sec".