GridWorld Case Study

Don Allen
Tracy Ishman
Ann Shen

Background

- New for 2008 Exam
- Developers:
  - Cay Horstmann – framework design and implementation
  - Chris Nevison and Barbara Cloud Wells – narrative
  - Judy Hromcik – solutions manual

MBS Comfort Zone

- Actors (bugs and critters) exist in a grid environment
- `Location` class to represent (row, column) pairings
- All `Actor` objects have an `act` method
- GUI has similar look and feel

Improvements to MBS

- Fewer interacting classes
- More flexible grid environment
  - Use to implement games (tic-tac-toe, Connect Four, races, etc.) without having to create an illogical extension
  - Easy to add a new actor by creating a subclass and GIF image with same name
  - Interactive object instantiation and message sending (a la BlueJ)

Setting Up GridWorld

- Download from AP Central (code, student manual and solutions manual)
- Includes directions for various IDEs
- GridWorld framework stored in packages
- API provided
- Projects for `Bug`, `BoxBug`, and 3 critters

Part 1: Observing & Experimenting w/ GridWorld

- Great for introducing objects and methods without having to write any code (no Java knowledge needed)
- Spend time exploring "what if" scenarios to learn how to interact with the `Grid` and how a `Bug` acts…

Part 2: Bug Variations

- Simple inheritance
- Teach students to create simple subclasses (no loops or arrays)
- `Bug` – acts using the following methods
  - `canMove`
  - `move`
  - `turn`
  - Override any of these four methods
- `BoxBug` overrides `act` & adds instance variables moving in a box pattern

Part 3: GridWorld Classes & Interfaces

- Critical for understanding the `Location` class, `Actor` classes, and `Grid` interface – necessary for `Critter` class in Part 4
- Work through the "Do You Know" sets
- Create worksheets for students as needed to cement understanding

Part 3: Location class

- Similar to MBS
- No `Direction` class
  - Direction represented as integer
  - Static constants for NORTH (0), EAST (90), SOUTH, WEST, NORTHEAST, SOUTHEAST, SOUTHWEST, NORTHWEST
  - Static constants for rotating: LEFT, RIGHT, HALF_LEFT, HALF_RIGHT, FULL_CIRCLE, HALF_CIRCLE, AHEAD

Part 3: `Grid` Interface

- Implemented by `BoundedGrid` and `UnboundedGrid`
- Methods with `Location` parameters assume argument is NOT `null`
- Methods for obtaining `ArrayList` of valid/empty/occupied adjacent locations…

Part 3: Actor Class

- Superclass for `Rock`, `Flower`, `Bug`, `Critter`
- Use `putSelfInGrid` and `removeSelfFromGrid` methods for adding and removing `Grid` objects – avoid `Grid put and remove`
- `act` method reverses its direction

Part 3: Actor Extensions

- `Rock`
  - Acts by doing nothing
  - Overrides `act` with empty body
- `Flowers`
  - Acts by darkening its color
  - Overrides `act` to reduce amount of red, green, and blue in its color

Part 3: Actor Extensions

- Bug
  - Acts by moving forward and leaving behind a flower when possible
  - When unable to move forward, makes 45 degree turn to right
- `instanceof` operator and `Color` class are used by `Bug` and `Critter`
  - Not in AP subset
  - Students expected to understand usage within context of Case Study only

Part 3: Group Activity

- Good design activity to be worked in small groups
- Illustrates need to ask good questions
- Demonstrates the existence of multiple correct solutions

Part 4: Interacting Objects

- `Critter` Class
- Critical to maintain pre- and post-conditions
- Five methods control behavior – override to create new critters instead of `act` method
  - getActors
  - processActors
  - getMoveLocations
  - selectMoveLocation
  - makeMove

Part 4: `Critter` Extensions

- `ChameleonCritter`
  - Instead of "eating" its neighboring actors, adopts color of randomly selected neighbor
  - Overrides `processActors` and `makeMove`
- `CrabCritter` (not tested)
  - Eats what is found in front, right-front, or left-front
  - Overrides `getActors`, `getMoveLocations`, `makeMove`

Part 4: Group Activity

- Demonstrates software development process.
- Groups develop specification for a new Critter
- Groups trade specifications and design and implement Critters

Part 5: Grid Data Structures (AB Only)

- `AbstractGrid<E>` Class – implements 5 methods common to both implementations
  - `getNeighbors`
  - `getValidAdjacentLocations`
  - `getEmptyAdjacentLocations`
  - `getOccupiedAdjacentLocations`
  - `toString`

Part 5: BoundedGrid<E>

- Two-dimensional array:
  `Object[][] occupantArray`
- Important to know time complexities of various methods

Part 5: UnboundedGrid<E>

- Maps Locations to grid occupant
  `Map<Location, E> occupantMap`
- `isValid` always returns `true`
- `numRows` and `numCols` is -1
- Important to know time complexities of various methods

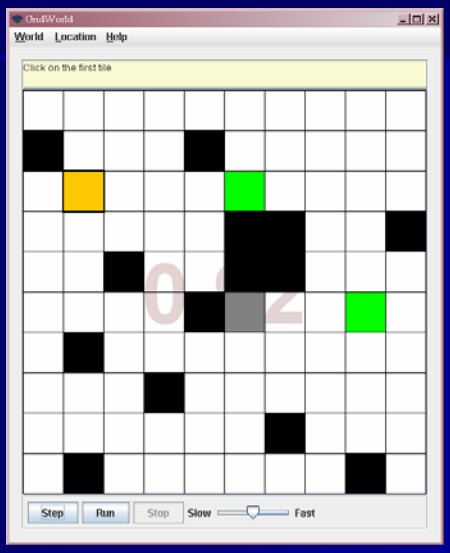
Part 5: Exercises

- Important to explore other implementations of `Grid`
- Be able to compare time complexities of the various implementations

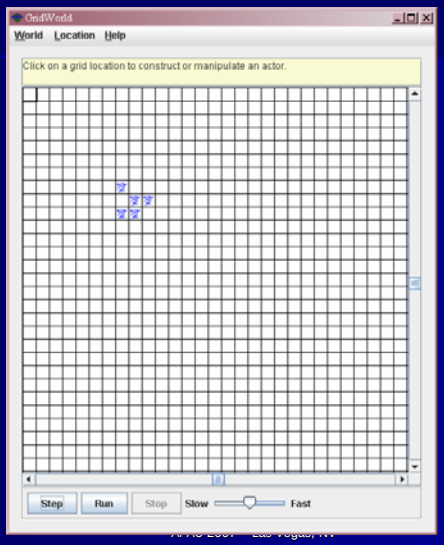
Appendixes

- A – Class Summary and Index
- B – Testable API
- C – Testable Code for A/AB
- D – Testable Code for AB Only
- E – Quick Reference A/AB
- F – Quick Reference AB Only
- G – Index for Source Code

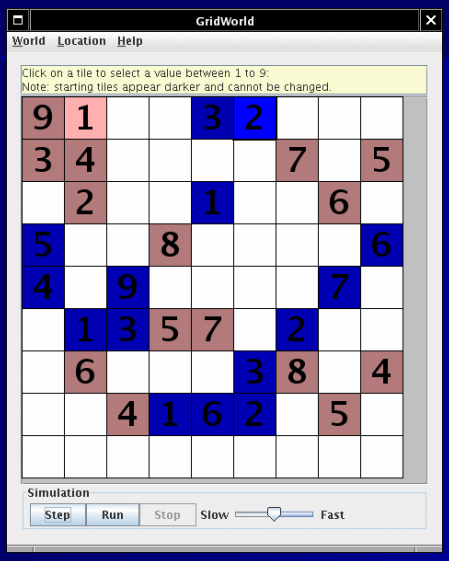
An Extension - TileGame



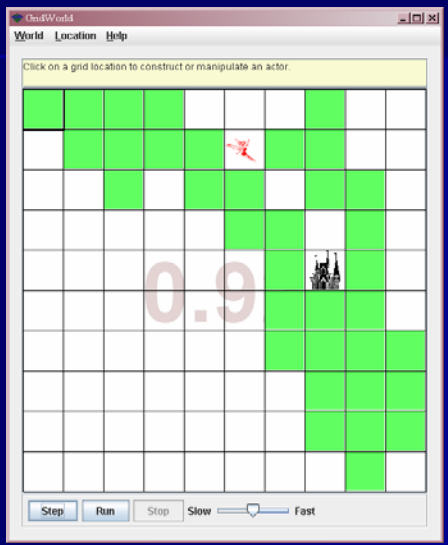
An Extension - GameOfLife



An Extension - Sudoku



An Extension - APACWorld



Finally.....

- CSTA K-12 Computer Science Web Repository: http://csta.villanova.edu/
- Joe Coglianese's GridWorld worksheets: http://coglianese.troyhigh.com/gridworld.html
- Slides to be posted next week: www.dave-reed.com