EXPANDED LEARNING BY CITIZEN SCHOOLS

# Amazing Mazes
## Materials and Teaching Checklist – lesson 4

Lesson Name: Introduction to Maze Walker Programming
Date to be taught: 3/26/2013

**"I Can" Skills:**

| Last Time | This Time | Next time |
|---|---|---|
| **Build mazes with different complexities** | **Create and manually control maze walkers** | **Create a basic maze walker program to control the walkers** |

**Before the Lesson:**

| Copies to Make | Materials to Bring | Visuals to Make |
|---|---|---|
| | - Projector<br>- Computers with Internet connectivity<br>- | |

**During the Lesson:**

| Section | Time (min) | I Say / Do | They Say / Do |
|---|---|---|---|
| **Hook** | 10 | **Review**<br>- Quick review and **teach-back** of last lesson:<br>○ How to draw a line using (x,y) coordinates?<br>○ What's the meaning of the "history box"?<br>○ What is a "draw-path" line called?<br>○ What is a collection of "draw-path" lines?<br>○ Is the order of lines in a program important? | - **Students review key points of last lesson**<br>○ **Use (x1,y1)(x2,y2) for the start and end points**<br>○ **History of actions taken**<br>○ **Draw-line command**<br>○ **Maze-building program**<br>○ **Not in the case of maze building, but may be important in other cases (e.g. in directions/commands given to a maze walker)** |
| **Mini Lesson** | 10 | **Manually controlling a maze walker**<br>**Teacher Demo:**<br>- Use the 2-player NetLogo program http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-2-players-2.html , create a simple maze (by clicking the "Draw simple maze" button to draw an "H" shape similar to last lesson's).<br>- Create 1 maze walker using the UI drop-down and button, and position it at (3,5) | - **students discuss and answer questions**<br>- **including the meaning and purpose of the program in the "history window" at the top right of the User Interface (UI)** |

- show the students how to control 1 walker, by moving it from the top left (3,5), to the bottom right (6,1)

**- Questions/discussion**: In this lesson, the “history box” (upper right corner of the UI) shows something different from maze drawing commands. It shows:

    - move-down1
    - move-down1
    - move-right1
    - move-right1
    - move-right1
    - move-down1
    - move-down1

  - Q: What is the meaning of these commands?
    - A: it's another **program**. This time to control the walker, not to build a maze (which we covered in lesson 3)!
  - Q: is the order of the commands in this program important? (A: yes)

- **Teacher demo**: In order to show that the walker program works: delete the walker and create another one, then copy and paste the walker program from the history box into the command-line box, and run the walker program.

- **Question/discussion**: in how many ways can we make the maze walker program “fail” (to reach the target)?
  - A: we can change the program by swapping lines/commands, removing lines/commands, delete lines in the maze itself.
  - The teacher will point out that this program is “brittle”: small changes can “break” the program. Later in the apprenticeship we'll learn to create “stronger” programs, which have less chance to fail.

| | | | |
|---|---|---|---|
| **Activity 1** | 15 | **2 player competition/game:**<br><br>- The object of the game/competition is for each student to walk their walker as quickly as possible from one corner of the maze (their starting point) to the other opposite/kitty-corner<br><br>- Teacher explains how to play the game: students create the maze and **2 walkers at opposite corners of the maze** – one for each student in the pair.<br><br>- **One walker will start at coordinate (1,1), the other at (17,11)**<br><br>- Each student controls their walker using the 5 navigation buttons (up, down, left, right, back), using the keyboard shortcuts (the letters W, Z, A, D, S respectively for walker 1, and I, M, J, L, K respectively for walker 2). Students need to use the keyboard not the mouse, since there are 2 players competing! | - **Students use the 2-player NetLogo program [http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-2-players-2.html](http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-2-players-2.html) to play the game, controlling their walker, and trying to get to the opposite corner first.** |
| **Activity 2** | 15 | **Walker program efficiency and effectiveness**<br><br>- Since after the game/competition, the history box will contain a mix of the programs of BOTH walkers (intertwined), the teacher will ask the students to "figure out" which program was "better" (regardless of which student won!)<br><br>- The idea is that a student can be very fast in making their walker move (and therefore they will win), but they could do more "dead-end moves" in the maze.<br><br>- The "better" program (as opposed to "better/faster execution or control speed) is the program that takes less commands or moves to execute. It is more efficient.<br><br>- The teacher will tabulate the length of programs from each pair of students (after they had agreed which program amongst themselves is the best/shortest)<br><br>- Teacher will ask the students to take their "best" program and try to make it fail to walk the walker from one corner of the maze to the other<br><br>    - Students can do this by:<br><br>        - mixing the order of the lines/commands in the program<br><br>        - deleting one or more commands<br><br>        - adding another command | - **Students need to count and compare the number of commands (walker moves) of each program, and determine if it's the "best program possible" (is it executing the shortest path from the origin to the destination in the maze, and is therefore more efficient?)**<br><br>- **Students call out the length of their best walker program, and teacher tabulates the results.**<br><br>- **Students will run their "best program" in the maze and then try to make it fail in various ways** |

| | | | |
|---|---|---|---|
| | | - changing the maze itself (using the “delete-path” button or command) | |
| **Activity 3** | 15 | **Bird's-eye view vs. inside maze view programming**<br><br>**First Bird's-eye view programming**<br><br>○ One very important way in which the walker program is brittle is that it depends on the orientation of the maze: if you just turn the maze 90 (or any other angle!) degrees, the walker program will fail to reach the destination!<br><br>○ Teacher demo:<br><br>○ Create very simple “H” shaped maze (even simpler that the one from lesson 3), by pushing the “Create simple maze” button on the UI:<br>○ draw-path (3,5)(3,1)<br>○ draw-path (3,3)(6,3)<br>○ draw-path (6,5)(6,1)<br><br>○ Create 1 maze walker at the top left of the maze: (3,5). Then turn the walker back (using the turn back button) to face in the direction of the path<br><br>○ With students' help: create a walker program to walk from the top left corner of the “H”<br>○ move-down1<br>○ move-down1<br>○ move-right1<br>○ move-right1<br>○ move-right1<br>○ move-down1<br>○ move-down1<br><br>○ Run the program to make sure it works in the maze | - **Students help build the simple “H” maze**<br><br>- Students help program the maze walker from a bird's eye view/perspective |

| | | | |
|---|---|---|---|
| | | ○ Then, create a different maze by turning the "H" shape on its side (turned 90 degrees), by swapping the x and y values of each draw-path command:<br>○ draw-path (5,3)(1,3)<br>○ draw-path (3,3)(3,6)<br>○ draw-path (5,6)(1,6)<br>○ Create and position a new walker at the same relative position: now at (5,6)<br>○ Turn the walker around so it faces the maze path:<br>○ move-left1<br>○ move-right1<br>○ back1<br>○ And run the same walker program, to see that now it failed, even though the maze itself did not change at all! | - Students help create a turned H shaped maze |
| **Activity 4** | 15 | **Now to inside maze view programming**<br>○ Make sure the walker faces the maze path again (it would have turned as a result of the failed program above)<br>○ With the help of the students create a new and different program – using an inside maze view/perspective (think like the walker itself, where forward, left, right are relative to itself, NOT to someone from above), here in sort format notation:<br>○ fd<br>○ fd<br>○ tl<br>○ fd<br>○ fd<br>○ fd<br>○ tr<br>○ fd | - Students help create an inside maze view program for the walker<br>- The commands can be entered in either be long format or short format:<br>- forward<br>- forward<br>- turn-left<br>- forward<br>- forward<br>- forward<br>- turn-right<br>- forward<br>- forward |

|  |  |  |  |
|---|---|---|---|
|  |  | ○ fd<br><br>○ Ask the kids to articulate what this demonstrates. Namely, that an inside view program is more "robust" and less prone to errors. But not by much!<br><br>○ We will learn later in the apprenticeship how to program "fool proof" programs, but we'll do it in stages, refining and strengthening our programs every time. | - Students discuss if and why the inside view program is better (more "robust", less "brittle") compared to the bird's eye view program. |
| **Exit Tix** | 10 | If we have time:<br><br>- Hands-on activity:<br><br>- the students can try to program an inside view program for the complex maze<br><br>- | **Students in pairs, program an inside view program for the complex maze** |
| **Dismiss** |  | Remind students of our goal regarding programming:<br><br>○ We will learn in the next lessons how to program "fool proof" programs (programs that don't fail or break), but we'll do it in stages, refining and strengthening our programs every time. |  |

## Thumbnails lesson outline:

- **Review and teach-back** of lesson 3 – (x,y) coordinates, "history box", command/line, program/group of commands

- **Teacher - NetLogo 2-Players, creates simple "H" shape** – 1 walker at:
  (3,5) going with the arrow buttons to (6,1)

- Teacher - NetLogo – **2-Players** – history window
  - **History window** – a collection of instructions/commands is a program (move-down1, etc.)
  - Delete walker, recreate one, and copy and paste the program to run the walker through the maze

- Brainstorm with Students – what are ways to make the program fail?
  - Swap program lines, delete lines, change the maze

- **Student Competition - NetLogo 2-Players, create complex maze** – 2 walkers at: (1,1) and (17,11)

- **Program effectiveness and efficiency -**
  - Students count length of best program
  - Teacher tabulates on board
- **Teacher - Bird's-eye view vs. inside maze view programming**
  - Create the H maze and run the program

- Then with students – flip the H maze on its side – run same program – it'll fail
    - This is Bird's eye and it's brittle
    - Then write program from Inside Maze view: forward, turn-left, etc.
    - show that it runs ok with both H maze and flipped H maze!

- **If we have time:**
    - **students program Inside Maze view program for the Complex Maze**