EXPANDED LEARNING BY CITIZEN SCHOOLS

Amazing Mazes

Materials and Teaching Checklist – lesson 5

Lesson Name: Maze Walker Programming
Date to be taught: 4/2/2013

"I Can" Skills:

Last Time	This Time	Next time
Create and manually control maze walkers	**Create a basic maze walker program to control the walkers**	**Create and test different maze walker programs and find their limitations**

Before the Lesson:

Copies to Make	Materials to Bring	Visuals to Make
	- Projector - Computers with Internet connectivity -	

During the Lesson:

Section	Time (min)	I Say / Do	They Say / Do
Hook	10	**Review and Teach-back** - Quick review and **teach-back** of last lesson: - differences between maze **building program** and maze **walking program**? - Bird's-eye view walker program to run through a simple maze - Use the 2-player NetLogo program http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-2-players-2.html , draw the simple H maze, ask students to call out the maze walking commands ○ Ask students to list a couple of ways to "mess up the walker" and demonstrate it	- **Students review key points of last lesson** - **Maze building program instructs the computer how to create the maze. Maze walking program instructs the computer how to control the maze walker to go through the maze.** - **move-down1, move-right1, etc.** ○ **One way is to change or delete any line/command in the walker program.** ○ **Another way is to change the maze itself (using delete-path)**
Mini Lesson	10	**Now to inside maze view programming** ○ In the H shaped maze, create a walker at (3,5) and make sure the walker faces the maze path (facing down) ○ With the help of the students enter a new and different program into the command-line area in the UI – using an inside maze view/perspective (think like	- Students help create an inside maze view program for the walker - The commands can be entered in either be long format or short format: - forward - forward

		the walker itself, where forward, left, right are relative to itself, NOT to someone from above), here in short format notation: ○ fd ○ fd ○ tl ○ fd ○ fd ○ fd ○ tr ○ fd ○ fd ○ Copy and paste the program from the command-line area (to save it in the computer clipboard), and then delete and recreate the walker at (3,5) and rerun the inside view program again, just to make sure students see that the program works every time. ○ 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: ○ draw-path (5,3)(1,3) ○ draw-path (3,3)(3,6) ○ draw-path (5,6)(1,6) ○ Create and position a new walker at the same relative position: now at (5,6), and turn it so it faces the maze path. ○ Run the same inside view walker program which you copied and saved in your clipboard (fd, fd, tl, fd, fd, fd, tr, fd, fd), to show that this time the walker successfully gets to the destination ○ 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! ○ 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.	- turn-left - forward - forward - forward - turn-right - forward - forward - Students discuss if and why the inside view program is better (more "robust", less "brittle") compared to the bird's eye view program.

Activity 1	15	**Hands-on: creating a maze and a matching maze walking program:** **- Ask the students, using the 2-player NetLogo program** http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-2-players-2.html **to create a maze, then create 1 maze walker, and 1 target anywhere in their maze, and write an inside view program to walk their maze to the target.** **Note: in order for the walker to be able to walk the maze, all the maze paths/lines need to be either horizontal or vertical, but cannot be diagonal!** - Teacher needs to make sure that the program works, by asking the students to delete the maze walker, recreate and reposition it in the same initial position and rerun their walker program, to show that it repeatedly/reliably works.	- **Students use the 2-player NetLogo program** http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-2-players-2.html **to write a maze walker program.** - **Students rerun the walker program a couple of times to make sure it works.**
Activity 2	15	**Maze Walker algorithms** * Left-hand walk inside a maze – student participation: – example 1: * arrange 3 classroom desks in a U shape, and declare a maze target at the end of one arm of the U shaped "maze" (it can be on the outside of the arm) * ask for a student volunteer and place them at the end of the other arm of the U shaped maze. * ask the volunteer to put their left hand on the edge of the table, close their eyes and just follow the edge, always staying in touch with the tables ("the maze wall") until they reach the target – example 2: * draw a simple H shaped maze on the board, where each part of the shape is wide like a path or "street" * put a target at the end of one arm of the H, and with a marker placed as the walker at the end of a different arm, trace a left-hand walk all the way to the target – Example 3: * ask a student to draw another, more complex maze on the board, with wide paths (like "streets"), and have another volunteer trace a left-hand walk from a starting point to a target.	- **Students volunteer and participate in the activities**

		– Teacher explains that an algorithm is a certain way and a sequence of steps to solve a problem in a way that the computer can execute. An algorithm is like a recipe for a dish that can be executed by a cook/chef.	
Activity 3	15	**Inside maze view program effectiveness – playing a game: "hide that cheese" (target)** * Teacher shows the new Java applet at http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-12-programming-algorithms-1.html and explains the new elements on the User Interface: ○ "Draw with mouse", "Left-hand walk", "Right-hand walk" * Then the teacher will explain the game "Hide that cheese" (target): • Students will build a maze of their own design, either using the "draw-path" button of the User Interface, or the "Draw with mouse" button. • One student in the pair will then place 1 walker anywhere in the maze, and the other student will place the target in the maze, in a way that they think will take the walker the most time/steps to reach. • Then they will start the left-hand walk algorithm and count the steps the walker takes to get to the target. • The students will switch roles, and see if they can find a longer path (basically trying to "hide" the target from the walker).	- **Students will play the game "Hide that cheese" (target)**
Activity 4	15	**Summary of left/right-hand algorithms** * The teacher copies and pastes the maze below the program at http://employees.org/~hmark/courses/amazingmazes/amazing-mazes-12-programming-algorithms-1.html (this is a "replicated H shape" from previous lessons). * The teacher creates a maze walker at (1,5), and asks the students to call out the location of the target which would result in the longest search (path, number of steps) by the walker. ○ This is a trick question, since the target location that's "hidden the best" depends on the	- Students call out the "best location" for the target. But this actually depends on the walker algorithm/walk that we select to execute: - for left-hand walk (3,4) - for right-hand walk (3,6)

		walker algorithm. ○ For a left-hand walk the best target location is at (3,4) ○ For a right-hand walk the best target location is at (3,6) * The teacher places the target in both places and shows how both algorithms/walks find the target	
Exit Tix	10	If we have time: - Hands-on activity: - the students can try to create a maze and hide the target in a way that **neither a left-hand walker nor a right-hand walker will find the target** (it is possible!). -	**Students in pairs, build a maze and try to hide the target so that neither walking algorithm hits/finds it.**
Dismiss		Remind students of our goal regarding programming: ○ 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 – what's the difference between a maze building program and a maze walking program?

- **Teacher - NetLogo 2-Players, creates simple "H" shape** – 1 walker at: (3,5)
  - **teach students** inside maze view commands: forward, turn-left, turn-right

- **Teacher - NetLogo 2-Players, creates turned "H" shape** – 1 walker at: (5,6)
  - **show that the same** inside maze view program works here too!
  -
- **Students - NetLogo 2-Players** – create **their own maze**, a walker, a target
  - Program the walker using inside maze view commands to run through their maze

- **Walking algorithms** – demo with Students
  - rearrange tables, Teacher draws "street maze" on board, Students draw "street maze" on board
  - Definition of algorithm

- **Student Game "Hide the cheese" - NetLogo -Programming Algorithms**
  - **Students create a complex maze** – 1 student places walker, 1 places target
  - see if left-hand or right-hand algorithms can find the target in the largest number of steps

- **Teacher demos/summarizes "Hide the cheese" - NetLogo -Programming Algorithms**
    - using the "replicated H shaped maze", place walker at (1,5), ask students where the best location for hiding the target is? (depends on which type of walk/algorithm)

- **If we have time:**
    - students program Inside Maze view program for the Complex Maze, trying to create a maze and target location which both left and right hand walks will not find