18-642 Recitation #6

February 23, 2018
Updates

• Homework:
  – Homeworks #13/15 graded on canvas
  – Homeworks #18a/18b/20 due Sunday night @ 11pm

• Project:
  – Project 5 graded on Canvas
  – Project 6 checkpoint due tonight @ 11 PM
  – Project 6 due 3/2 @ 11 PM
  – Project 7 will be due 3/23 (Spans exam + spring break)

• Exam coming up on 3/7
  – HW due on 3/4 is an exam review HW

• Be sure to fill out question sheet to receive attendance points today!
Today

• Some tips for the projects
• State charts
• Homework discussion
Project #6

• Checkpoint due today is:
  – Brief description of your new algorithm
  – Requirements for new algorithm
  – Screenshots showing you ran the turtle on m1, m2, m3

• Tips:
  – Keep your algorithm simple
  – Leverage visit count array from Project 5
Requirements

• Requirements for Project 6 are **product** requirements
  – Like in HW 18a
  – Do not reference internal state variables (like an enum value)
  – Can reference internal memory (like visit count array)

• Example requirements for left-hand rule:
  – R-1: If the turtle is obstructed by a wall, the turtle shall turn right.
  – R-2: If the turtle is not obstructed by a wall, the turtle shall move one square in the direction it is facing.
  – R-3: If the turtle has just completed a move forward, it shall turn left regardless of obstruction.
  – R-4: If the turtle has reached the end square of the maze, it shall not move.
Bumped() function

- `bumped(x1,y1,x2,y2)` checks to see if there is a wall segment with **endpoints** (x1,y1) and (x2,y2):

```
bumped(2,1,2,2) = true
```

```
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
```

(2,1) and (2,2) indicate the endpoints of the wall segment.
Bumped() Function

• bumped(x1,y1,x2,y2) checks to see if there is a wall segment with endpoints (x1,y1) and (x2,y2):

bumped(1,1,1,2) = false

![Diagram showing a grid with a wall segment and points (1,1) and (1,2)]
More notes on scope

- Separation of student_turtle and student_maze is like robot and its simulation environment

- Make **one** decision per call to moveTurtle:
  - At most **one** call to bumped()
  - **One** call to displayTurtle() (one turn)
  - Move forward at most **one** step

- This is like a real robot system and control loop
  - In real life, the number of tasks in a control loop is restricted because of timing and system consistency
  - Corresponds to time-triggered statechart design
Project #6 Questions?
State Chart Reminders

• Indicate your start state
• No side effects on transitions
• Transition on input and internal variables
• Set output and internal variables in state
• Next week we’ll review how to implement state chart in code
Homework Discussion

• Consider an electronic dimming light switch that has four buttons: ON, OFF, UP, DOWN, and MEM. The output of the system is LIGHT with an intensity of 0-100. Generally speaking, ON and OFF turn the light all the way on and off. UP adds 10% illumination. DOWN subtracts 10% illumination. Pressing MEM sets lights to a predefined setting. Pressing and holding MEM remembers the current light setting as the predefined setting.

• Exercise #1: Create a statechart for this system. Each student either adds ONE state to the diagram OR adds an arc between two states until the statechart is fully defined.

• Exercise #2: Create sequence diagrams for this system according to the following roles: