Robot Motion Planning
Info Sheet

***Materials Needed**: Printed copy and pencil, or digital copy with drawing capabilities.*

Activity Overview

1. Icebreaker!
2. Ensure that students have all the necessary materials
3. Go through Introduction questions
    a. Encourage them to think about what robots could be used for. They should be thinking outside the box, wild ideas are good
    b. Transition to thinking more technically about how robots would need to move to be able to accomplish these tasks, how they would need to plan, make decisions, etc
4. For each experiment
    a. Go over the directions **first**
    b. Let them try things out
    c. Go over the discussion questions (this can be brief)
5. Go through conclusion questions
    a. The first question should be a "gimme". Motion planning makes it easier to move by giving an easy to follow path (as demonstrated in experiment 4)
    b. Second question is tricky and intentionally open ended. Looking for answers that get at the idea of changing some of the assumptions we made in these experiments. For example:
        i. Use something besides a 2D grid to represent the state
        ii. Have more complicated actions than up/down/left/right
        iii. Have different costs associated with different movements

Robot Motion Planning Background (*Bolded terms are in the PPT*)

- Motion planning is part of a 3-step process that many robots follow (and some consider a definition of what is a robot):
    - **Sense** – gather information about the task with sensors. For motion planning this is typically some notion of your environment (e.g. a map)
    - **Plan** – Determine how you want the robot to achieve the task. In this case determine the path that you will follow to reach the goal
    - **Act** – Execute the plan by determining how to control your actuators.
- Key planning concepts

- **Map** (also terrain, environment, etc) – Data of some form that contains information about what is around the robot. Varieties include:
  - Occupancy grid (what we are using, a grid where each element is either 0 or 1 whether it is impassible or passible)
  - Cost map (grid or other data structure where each element contains the cost induced by moving to that location)
  - Height map (2D grid that contains the height of the terrain at each point)
  - Graph (just a series of vertices and edges that connect them, this is closer to what Google Maps uses). Technically the other options all correspond to equivalent graphs

- **States** – Some numbers that represent where something is. Think the 2D or 3D Cartesian position of the robot, the desired robot position, etc. Can also include orientation, velocity, and any number of things. We will just be using position (specifically as indices in an occupancy grid).
  - **Robot State** – where the robot is
  - **Goal State** – where you want to go

- **Actions** – A description of how the robot can move (i.e. "transition between states"). We will be using up/down/left/right (known as a 4-connected grid) but you could also move diagonally (an 8-connected grid) or any number of ways. Defining these to match your system but also work numerically is where the research is.

- **Algorithms** – The procedures you use to plan the sequence of states and actions that lead from the robot state to the goal state. Examples include:
  - **Dijkstra's Algorithm** – a very common one which they will use in Experiments 2 & 3. Guarantees the shortest path to the goal. Works by recursively computing the cost to travel to certain states.
  - A* - A variation of Dijkstra that includes a guess (a "heuristic") of how far a given state is to the goal. Guarantees the shortest path to the goal while also checking the fewest number of states (more computationally efficient). Probably the most common algorithm in most motion planning applications.
  - Breadth-First Search – Search all the successors of the start state. Then search all the successors of those successors. Continue until you find the goal. Not very efficient.

- Depth-First Search – Search through all the successors of one state until you reach the end. Then backtrack and search all the successors of the next state. Even less efficient than Breadth-First Search.

Robot Motion Planning Experiment 1 - Naïve Approach

- In this experiment they will try their hand at motion planning without following any algorithms.
- The purpose of this is to have a baseline to compare path length against Dijkstra (which should be optimal)
- Go through the Experiment 1 instructions **first** all the way through, then have them start planning paths in each of the scenes
- **Important:** make sure they know to add one to their Total Move Cost every time they move around the maze, even if they backtrack. This isn't a test of how smart they are or how good they are at solving mazes, it is just getting a baseline
- Discussion questions: Just try to get them thinking analytically about how they attempted to plan the path. Did they explicitly or implicitly create any rules for themselves? Anything better than "I dunno, I just kinda moved randomly" is great!

Experiment 1 & 2 "Solutions":

	Scene 1 Total Move Cost	**Scene 2 Total Move Cost**	**Scene 3 Total Move Cost**
Naïve Approach	>=12	>=30	>=28
Dijkstra's Algorithm	12	30	28

Robot Motion Planning Experiment 2– Dijkstra's Algorithm (*with precomputed costs*)

- During Experiments 2 & 3 the students will execute Dijkstra's Algorithm. The steps are provided in the activity sheet, so familiarize yourself with those.
- In this experiment they are given pre-computed costs so all they should have to do is move to the cell with the lowest cost until they have a complete path
- Discussion questions: Make sure they are clear that Dijkstra is better than other approaches, and that the purpose of using an algorithm is that you don't really have

to make decisions (although bonus points if they recognize that if they have multiple options for moving to the lowest cost it doesn't matter which they choose)

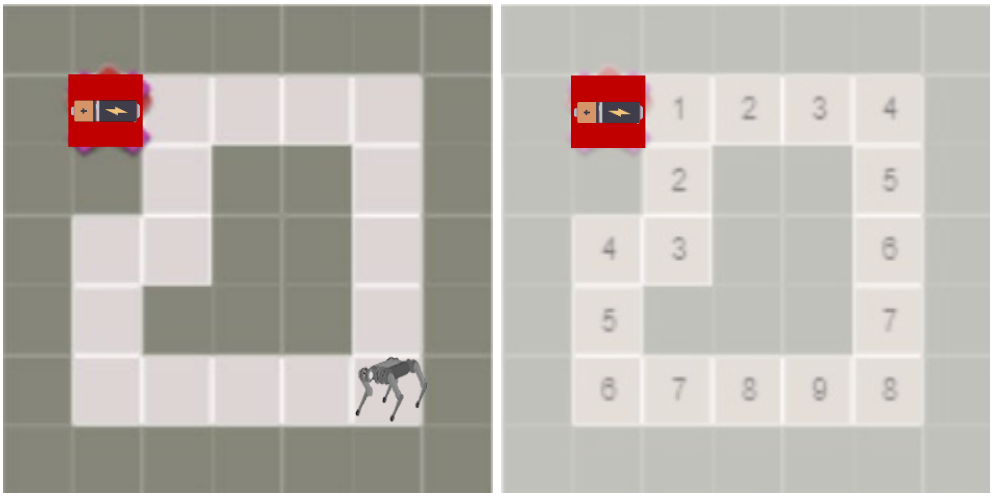
Robot Motion Planning Experiment 3 – Dijkstra's Algorithm (*from scratch*)

- In this experiment they will actually implement the full Dijkstra's Algorithm, including computing the costs themselves. This may be tricky on some of the harder scenes, so don't worry if they don't finish. Make sure there is time left for Experiment 4
- Again refer to the activity sheet for detailed instructions
- If done incorrectly Dijkstra will still yield an answer, it just might not be optimal. If they get sub-optimal solutions (which is okay), there are two likely reasons:
    - You need to explore states in a certain order – always choosing the one with the lowest cost (ties don't matter). Choosing the wrong order, exploring states multiple times, or exiting before the goal is the lowest-cost unexplored state could give sub-optimal paths. Keep track of which states you've already explored (this is why the instructions say to put a dot in the corner to indicate this).
    - Some states might be the successor of multiple states – you need to make sure you update the value of an already-updated state **if and only if** the new value would be lower. Note that already explored states (ones with dots) should never need to be updated (if they are, something went wrong).

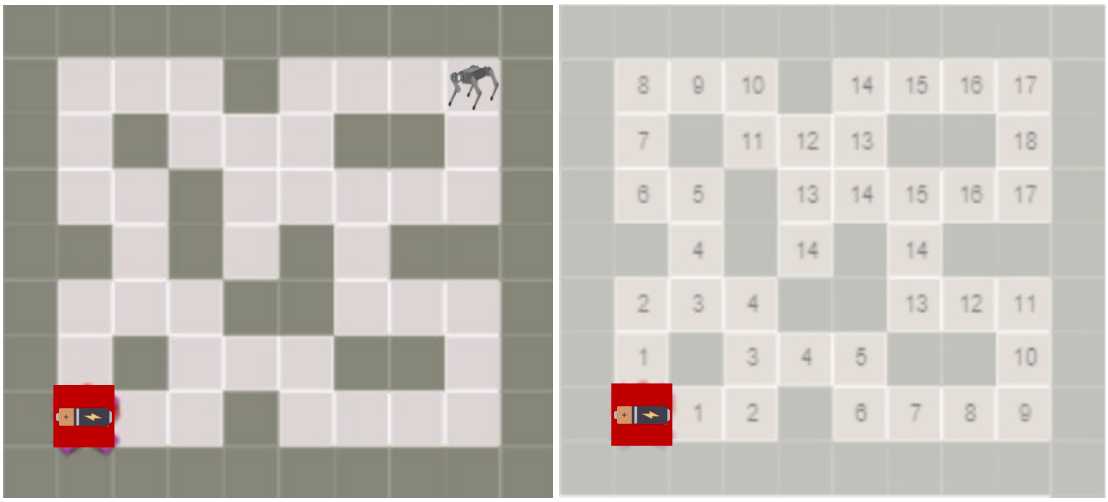
Experiment 3 Solutions:

Scene 1 (Total Move Cost = 5):

Scene 2 (Total Move Cost = 8):



Scene 3 (Total Move Cost = 17):



Robot Motion Planning Experiment 4 – Path Following

- This experience involves a webpage that creates obstacles and generates paths, along with the ability to explore them in first person. It's a fun little applet that they could explore for a while or could just play with briefly, it is meant to be flexible for timing
- Play around with the webpage beforehand so you are comfortable adding/removing obstacles and solved paths, and transitioning between first-person and overhead views
- The purpose of the experiment is for them to see how useful path planning is when you cannot see the whole map. They should time themselves solving the maze in

the first person view, both with and without paths to guide them. It should be **far** faster with a path.

- Note: the Recursive Division environment without a map is extremely easy to get lost in. Make sure they know that they should stop the experiment if they exceed two minutes.
- If they finish with time to spare, they can just play with the webpage (adding their own obstacles, looking at the other path planning algorithms, etc.)

Sources

Webpage for Experiment 4: https://dhruvmisra.github.io/Pathfinding-Visualizer-ThreeJS/
Webpage for interactive walkthrough of all this content and more:
https://www.redblobgames.com/pathfinding/a-star/introduction.html