CS123

Programming Your Personal Robot

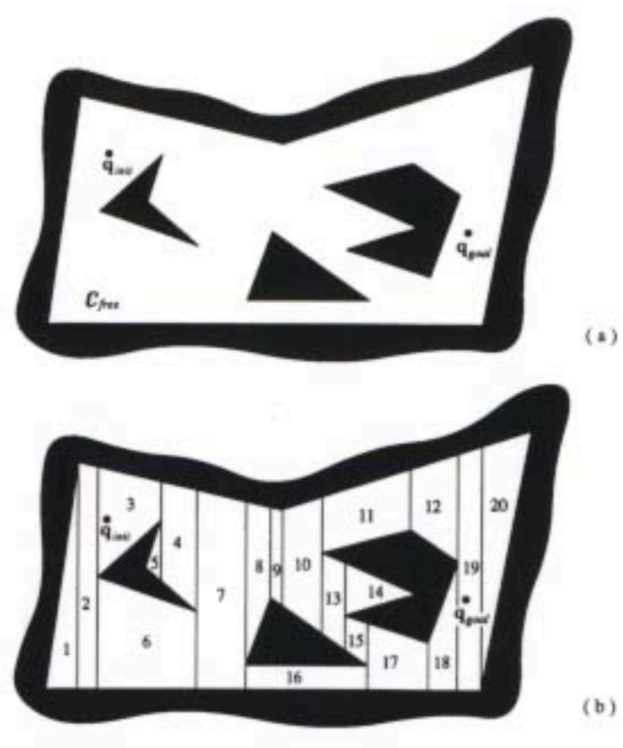
Part 3: Reasoning Under Uncertainty

Topics

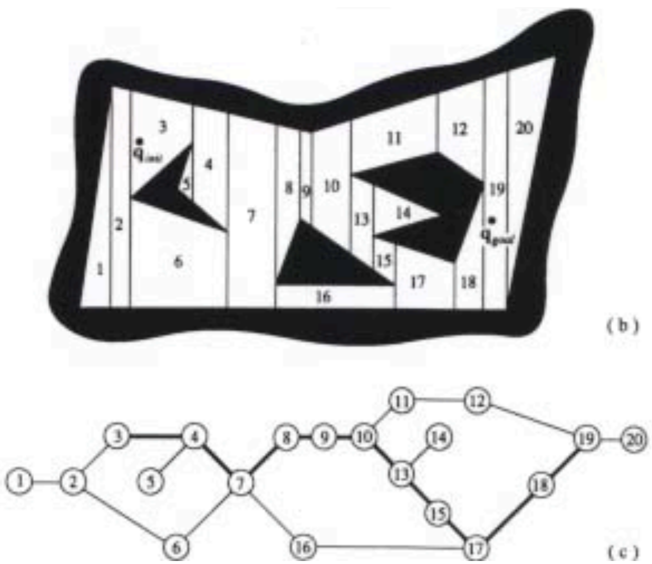
- More On Motion Planning
  - Search
  - Potential Field Method
- More On Control Under Uncertainty
  - Motion "Primitives"
  - Avoiding "Unexpected" Obstacles
- More On Homework Assignment Part # 3-2
  - student demo (Starbuck reward still good)
- Final Project
  - Schedule
  - Project Ideas (and guideline)

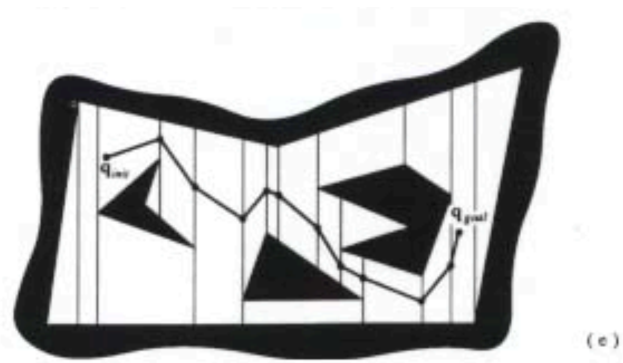
Motion Planning: Discretization of Space

- Different methods for “discretizing” space:
  - Visibility Graph
  - Voronoi Diagram
  - Cell Decomposition



Motion Planning: The Search Problem

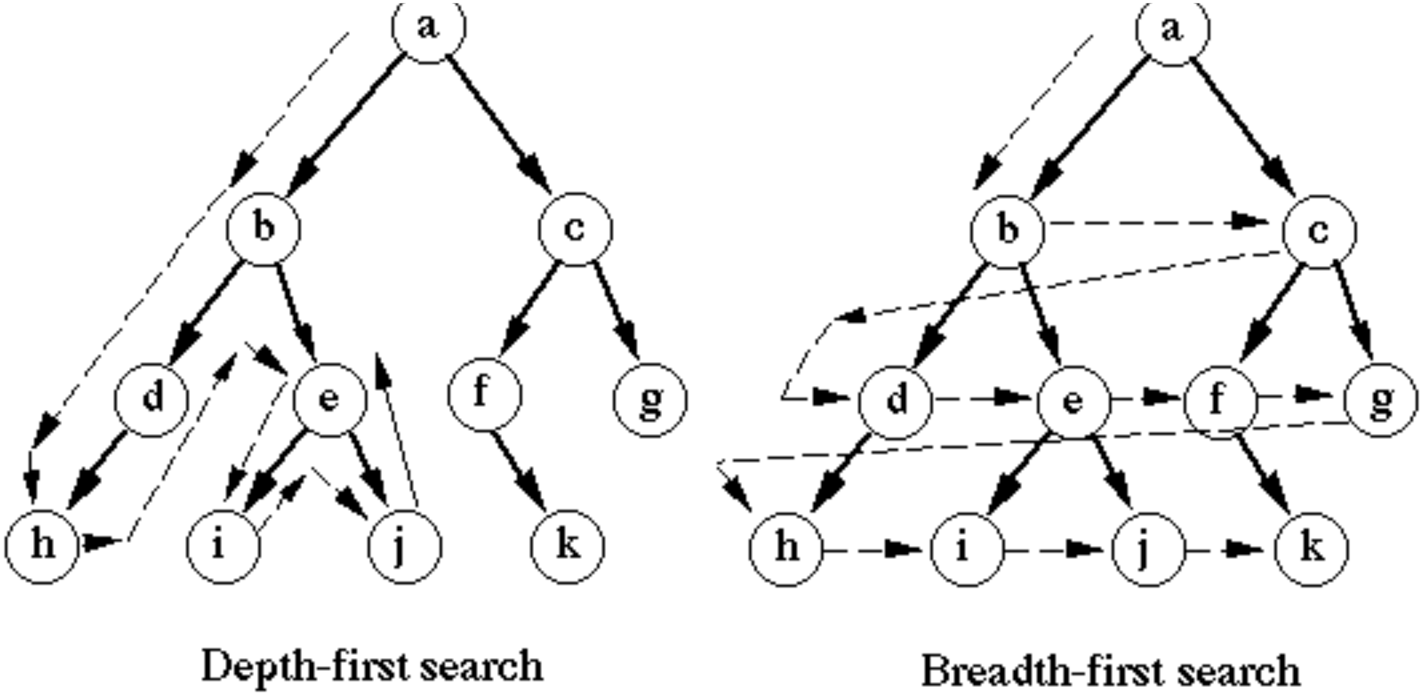




Search

- Uninformed Search
  - Use no information obtained from the environment
  - Blind Search
    - BFS (Breath First)
    - DFS (Depth First)
- Informed Search
  - Use evaluation function
  - Use “Heuristic” to guide the search:
    - Dijkstra’s Algorithm
    - A*

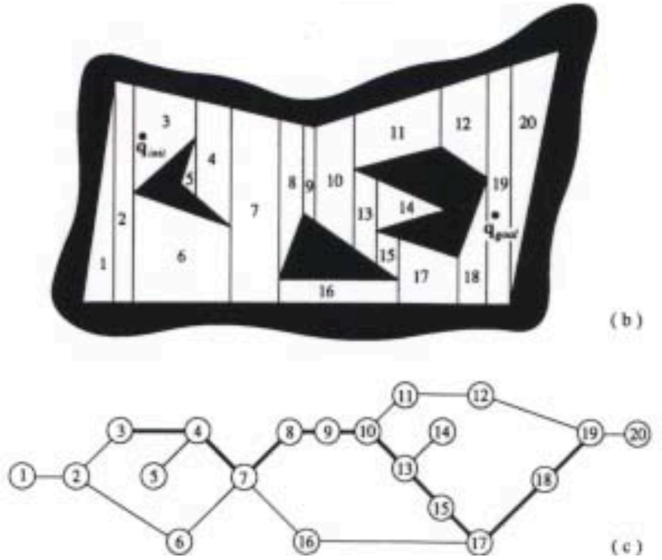
Search: DFS and BFS



No sense of “cost” of the path (how good is the path)

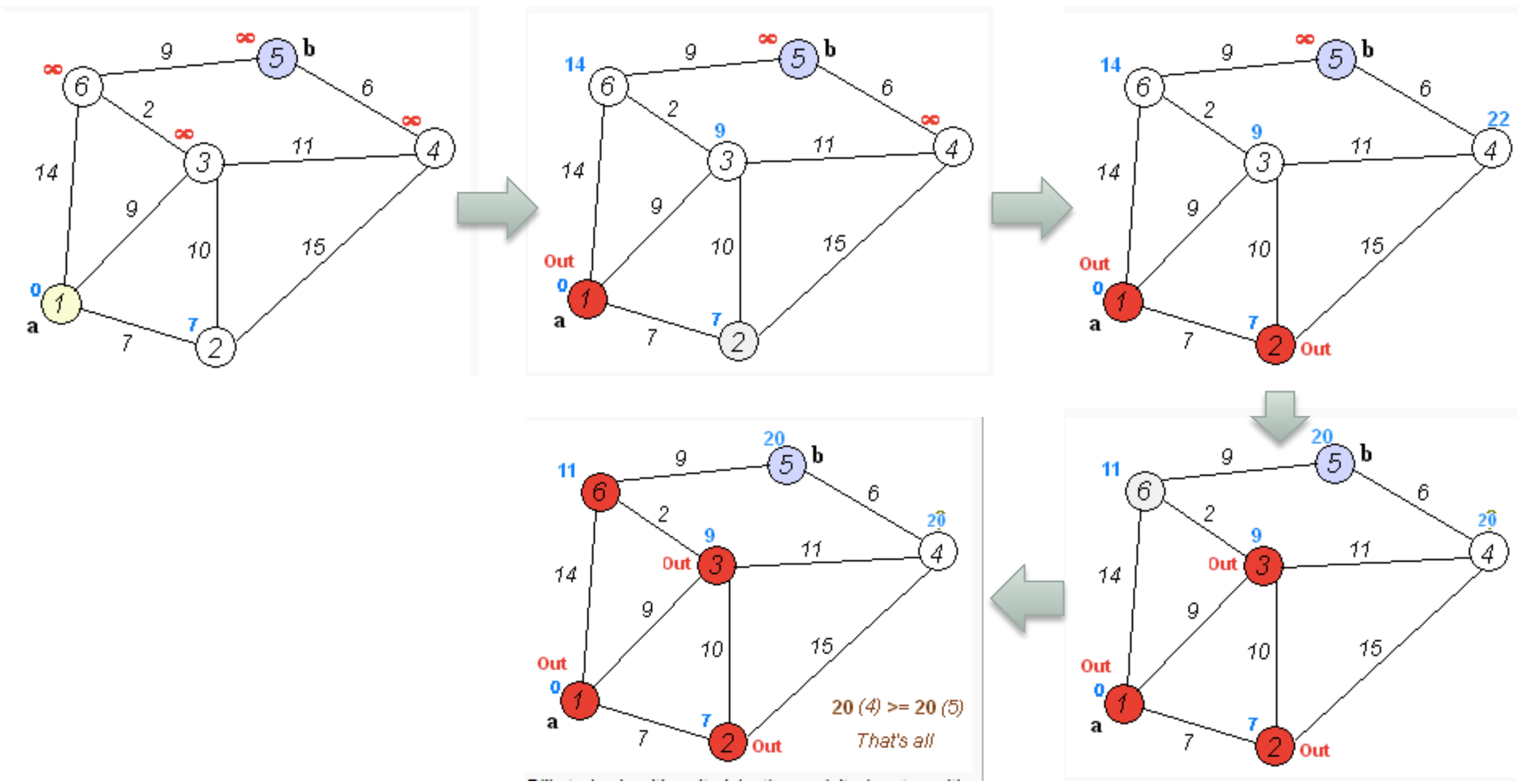
Informed Search:

- Use "domain" information to guide the search
  - Cost of going from one node to the next : distance of travel
  - Edge has a "cost value"
- Nodes are selected for expansion based on an **evaluation function** *f(n)* from the set of generated but not yet explored nodes
- Then select node first with lowest *f(n)* value



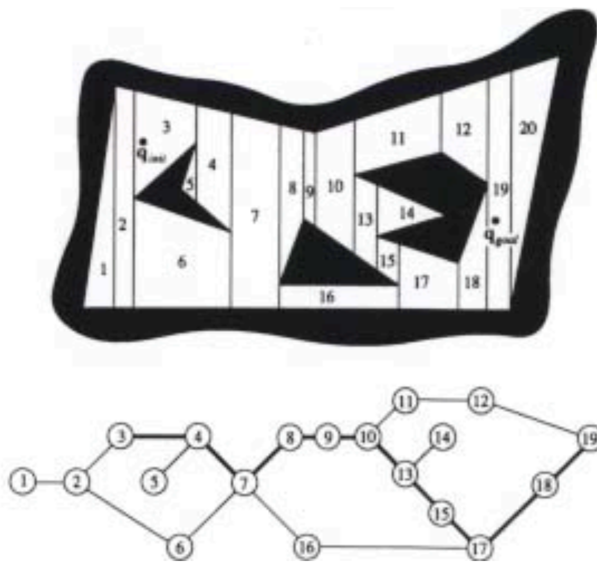
Informed Search: Dijkstra’s Algorithm

- Shortest Path
  - a.k.a “Greedy Method”



Use of Heuristics

- Estimate “Distance to Goal” at each node



(b)

(c)

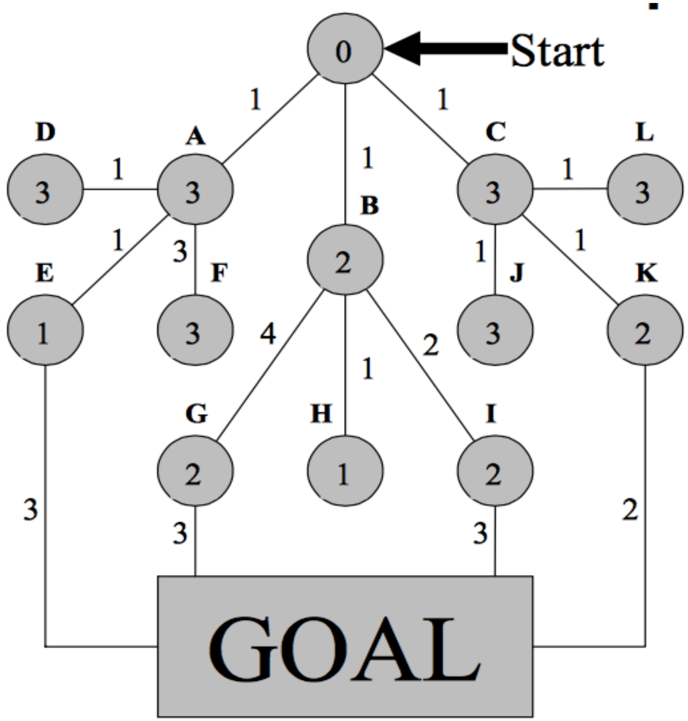
Informed Search: A*

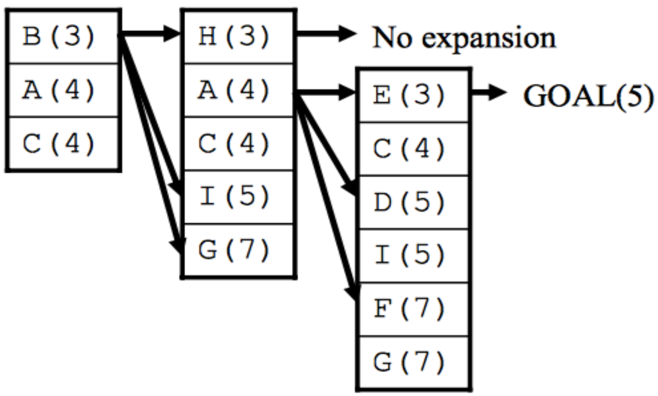
- First suggested by Nils Nilsson (at SRI) as an heuristic approach for **"Shakey" the Robot** to navigate through a room with obstacles
- Later improved by Raphael and Hart

Shakey – The "Grand Daddy" of Modern Robots

Informed Search: A*

- Cost at any expanded note : f = g + h, where
  - g = sum of “edge” costs of best path leading to node
  - h = estimate of cost from node to goal





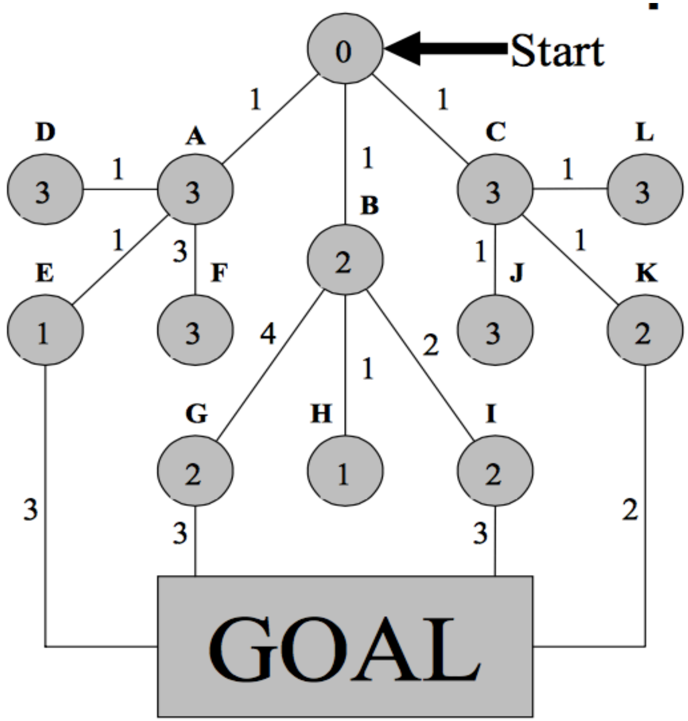
We’ve found a path to the goal:
Start => A => E => Goal
(*from the pointers*)

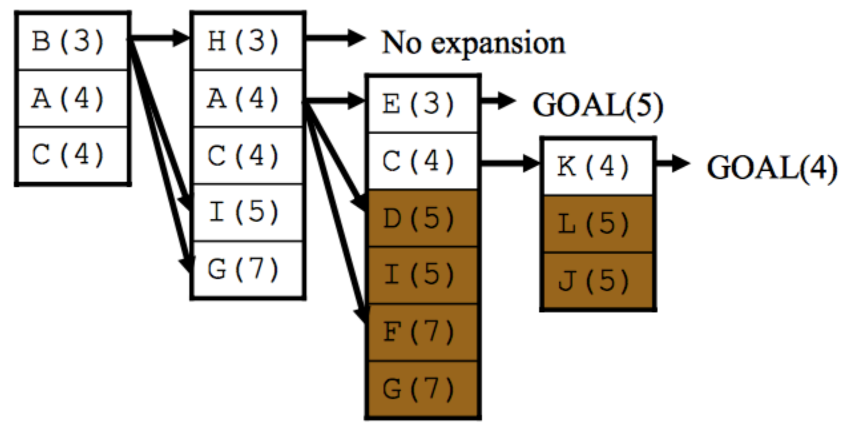
Are we done?

Note: Example taken from Howie Choset CMU Lecture

Informed Search: A*

- Terminate when Priority Queue is empty
  - throw away nodes in the queue has cost more than goal cost found so far





We can continue to throw away nodes with priority levels lower than the lowest goal found.

As we can see from this example, there was a shorter path through node K. To find the path, simply follow the back pointers.

Therefore the path would be:
Start => C => K => Goal

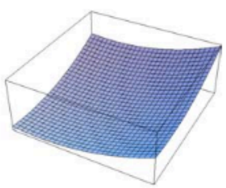
Potential Field Method

- All techniques discussed so far aim at capturing the connectivity of *C_free* into a graph
- **Potential Field Methods** follow a different idea:
  - The robot, represented as a point in *C*, is modeled as a **particle** under the influence of a **artificial potential field U** which superimposes
    - **Repulsive forces** from obstacles
    - **Attractive force** from goal

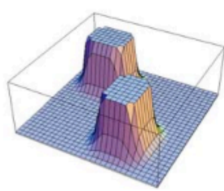
Potential Field Method

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

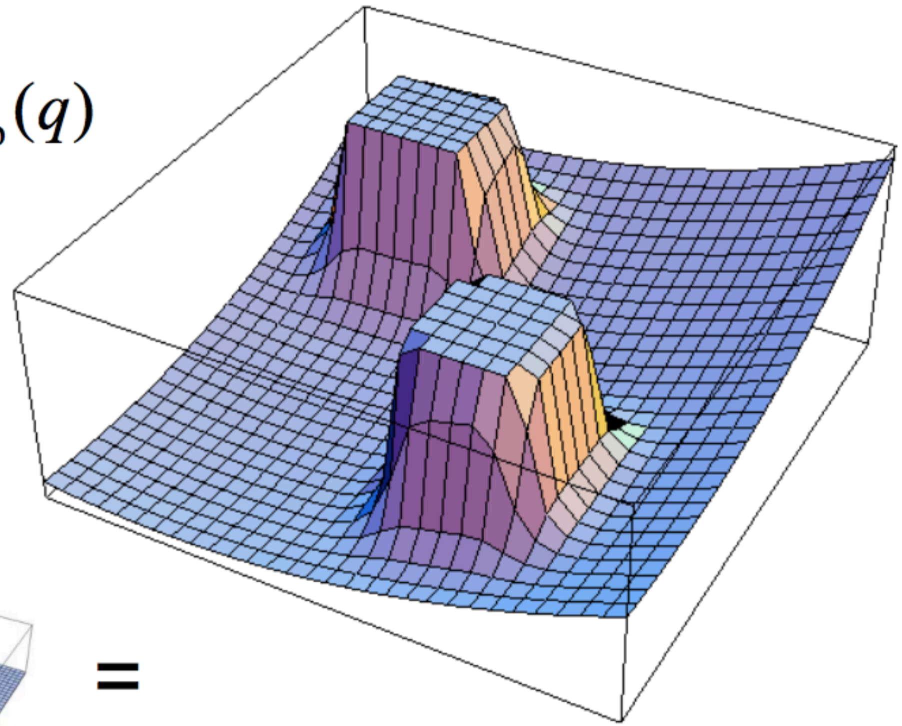
$$F(q) = -\nabla U(q)$$



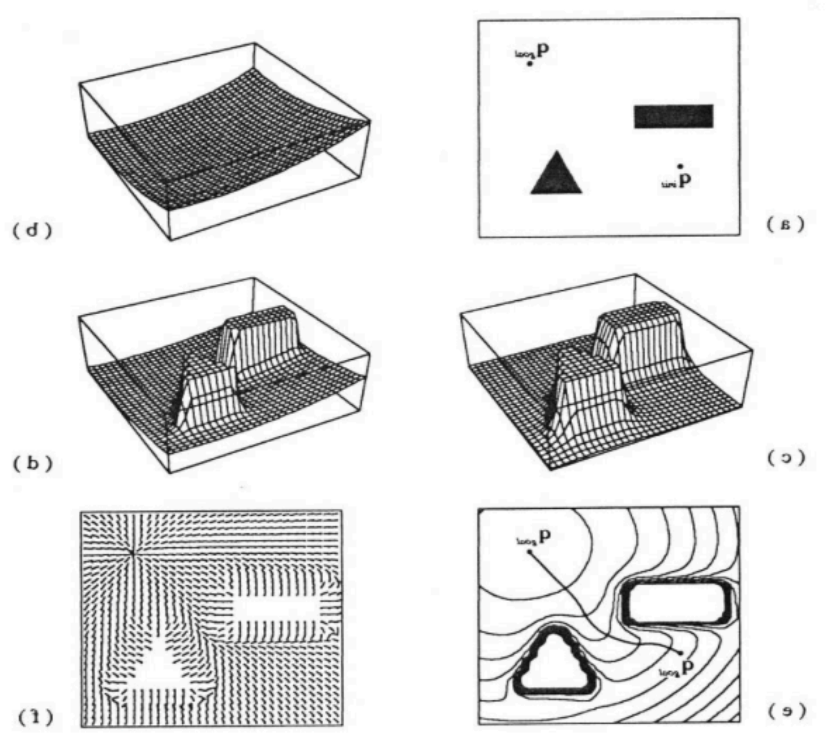
+



=



Potential Field Method: Gradient Descent

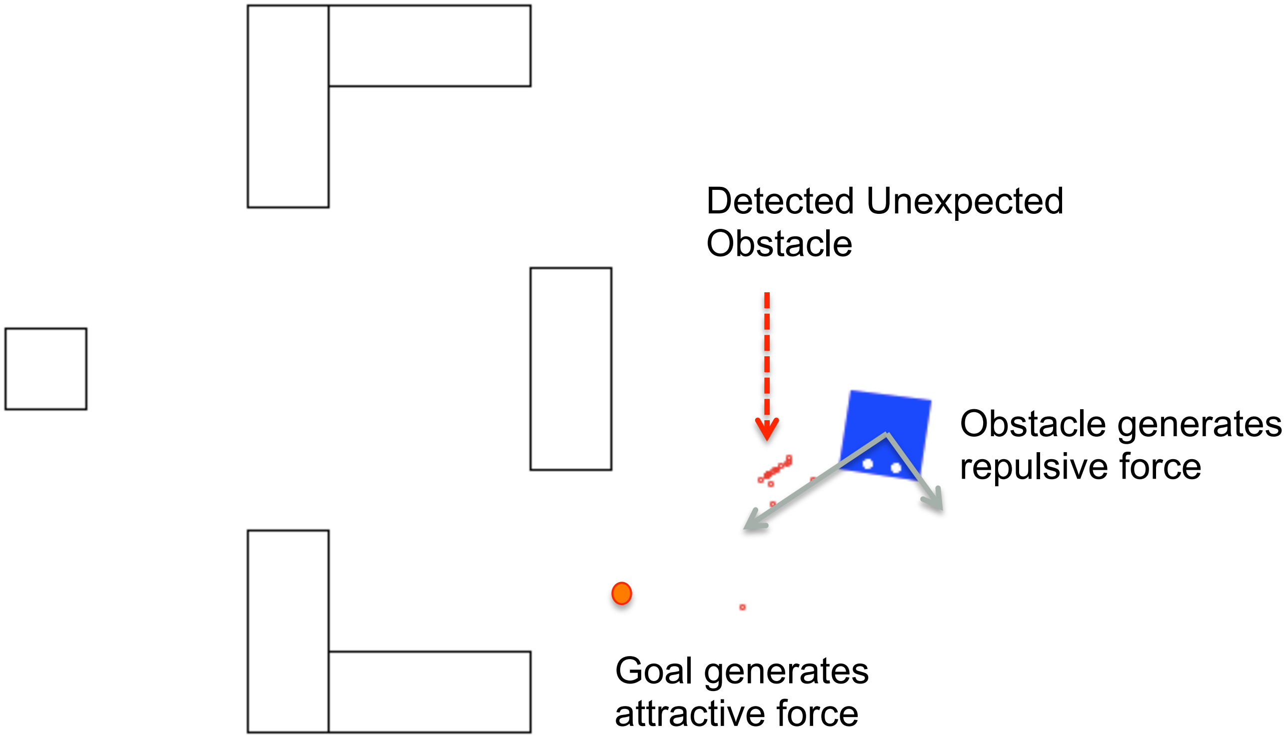


16-735, Howie Choset, with slides from Ji Yeong Lee, G.D. Hager and Z. Dodds

“Unexpected” Obstacle Avoidance

- Simple Potential Field Method has the drawback of getting stuck at “local minimum”
- But is good for “local obstacle” avoidance, such as
  - unexpected obstacles in environment (like moving people)
  - or known obstacle become “unexpected” due to control uncertain

Local Obstacle Avoidance

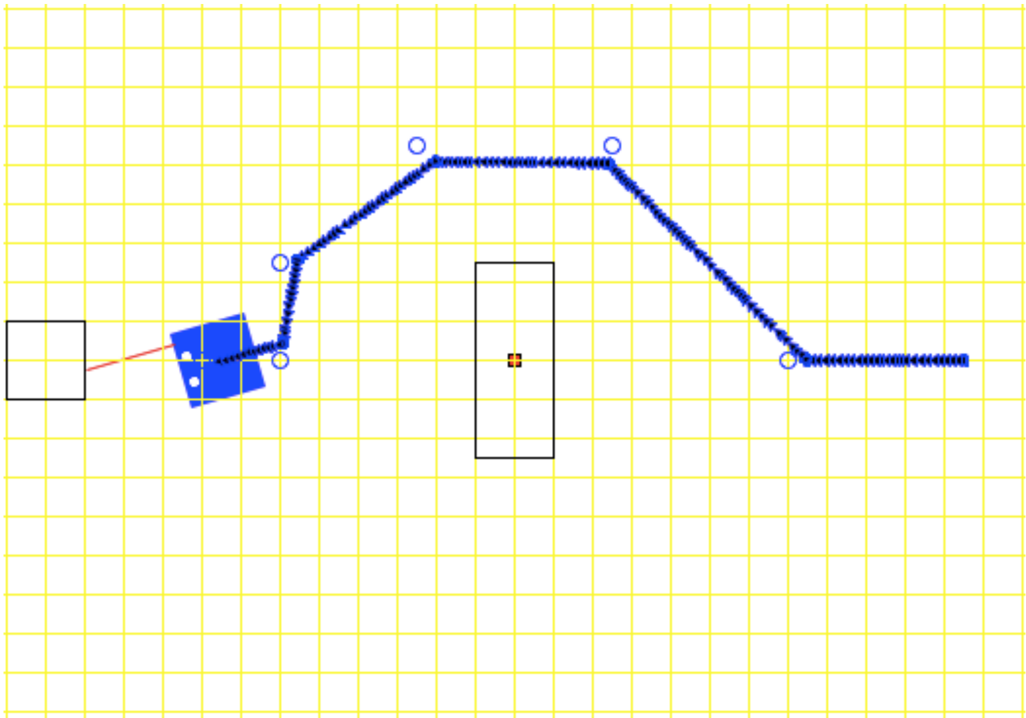


“General” Controller for Hamster

- Separating Planning and Control
  - Should not hard-code the controller together with the planner
  - The planner outputs a list of “sub-goals”
  - The controller translates the sub-goal list into a sequence of executable “motion primitives”

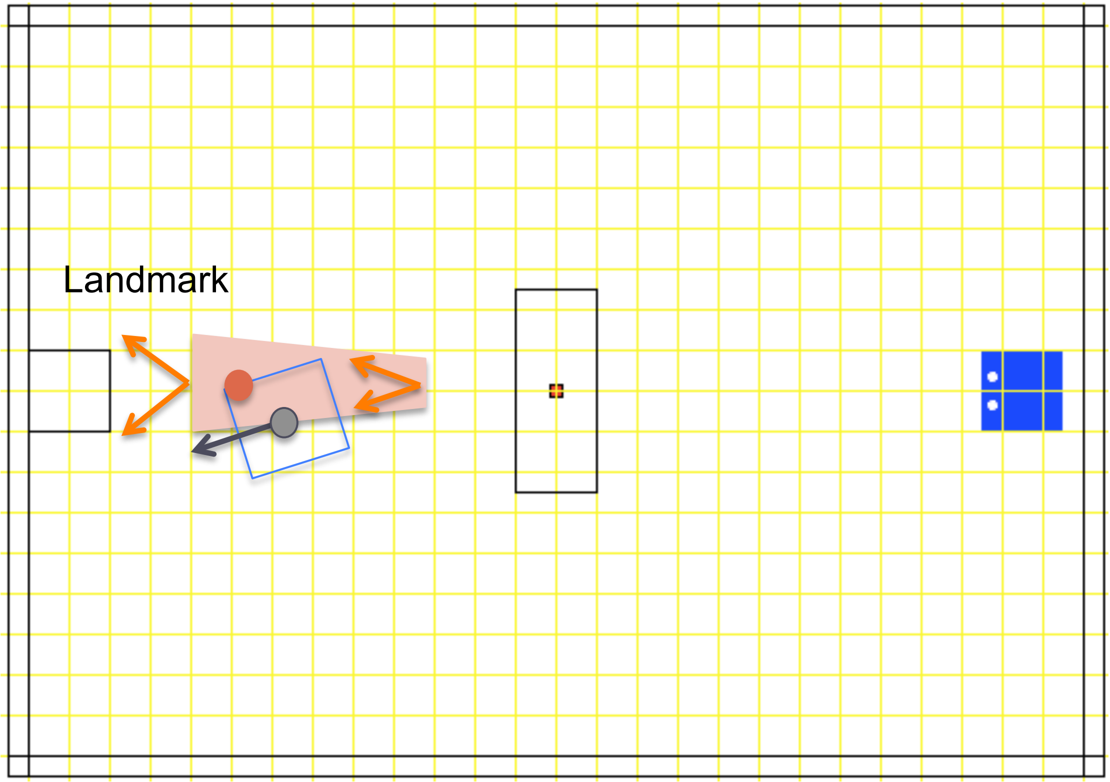
Motion Control: Motion Primitives

- Perfect World:
  - Move to (x, y, a)
  - Terminate when getting close enough to (x, y, a)

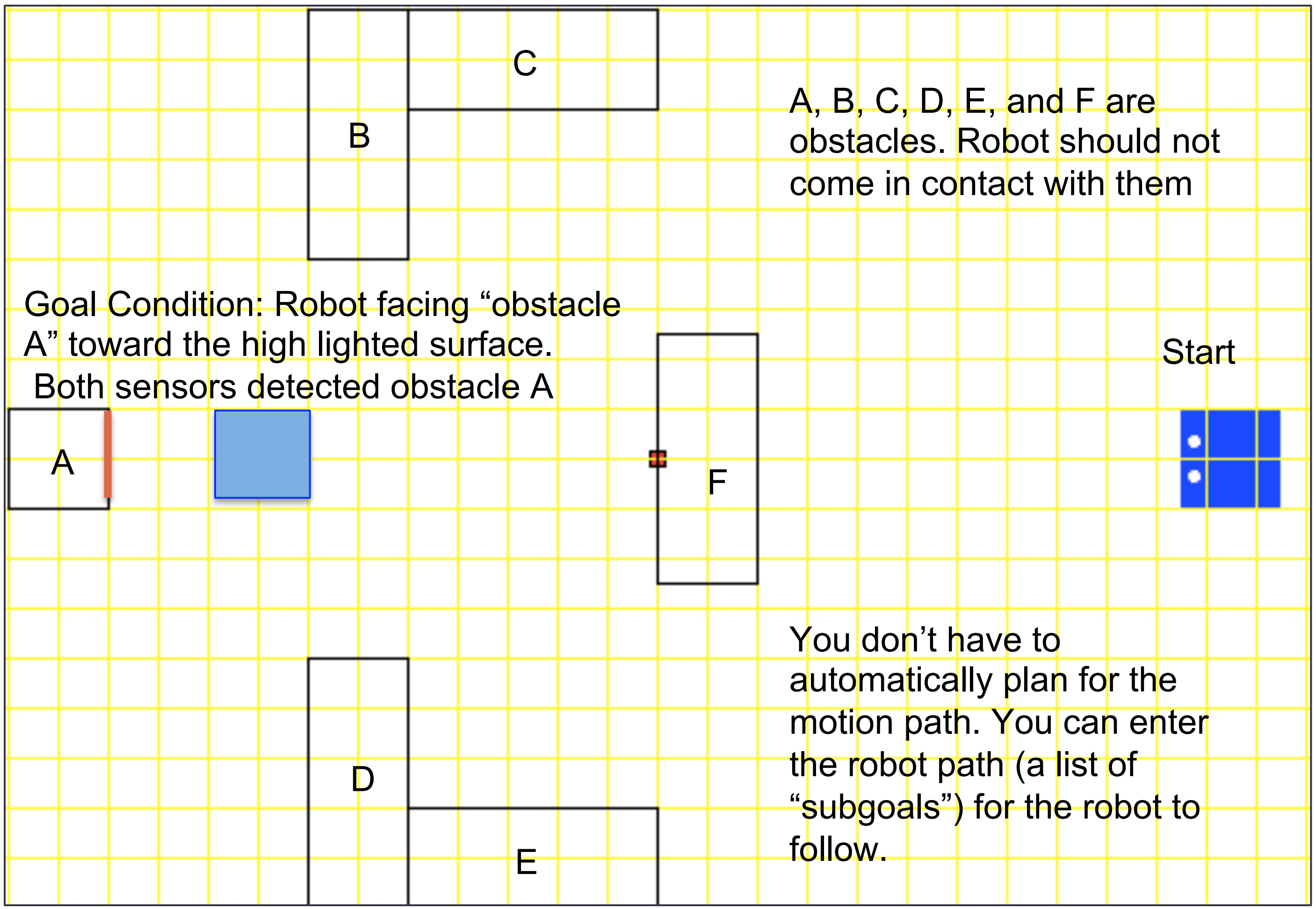


Motion Primitive: Control Uncertainty

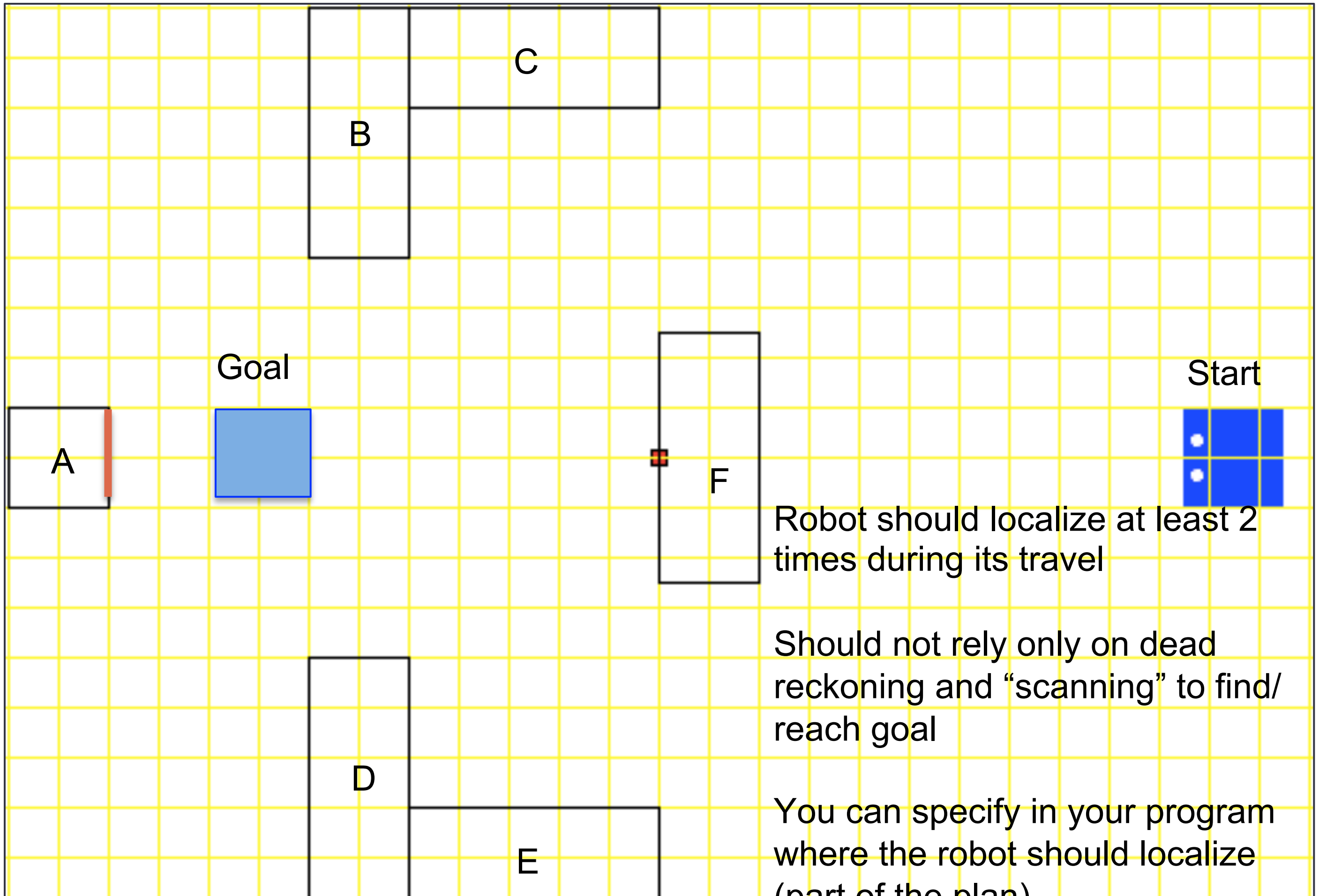
- Real World – Control Uncertainty
  - Move along d (direction)
  - Terminate with some sensor



Home Work Part #3-2



Home Work Part #3-2



Robot should localize at least 2 times during its travel

Should not rely only on dead reckoning and “scanning” to find/reach goal

You can specify in your program where the robot should localize (part of the plan)

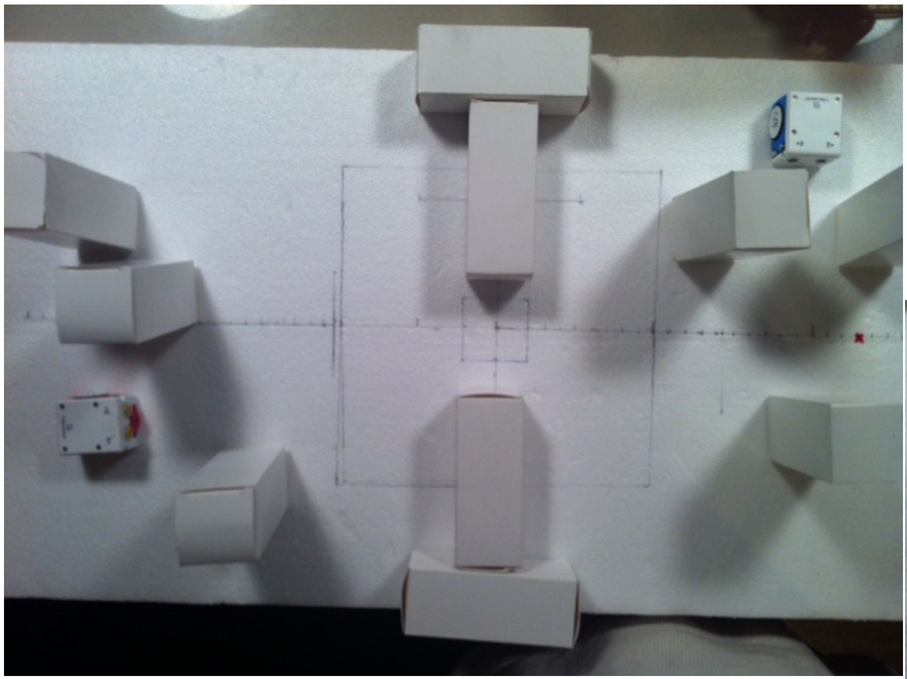
Final Project

- Team of 2 – 4 persons
  - Number of persons should correspond to the number of robots used in the project – and each robot should perform "useful" tasks (no spectator robot ☺)
- Schedule
  - Submit Proposal by Nov 6th
  - Proposal Approval by Nov 13th
  - Schedule for "Demo Event" during Final week (Dec 7th – 11th)
    - would like to have every one attend so can other teams' project

Two Project Ideas

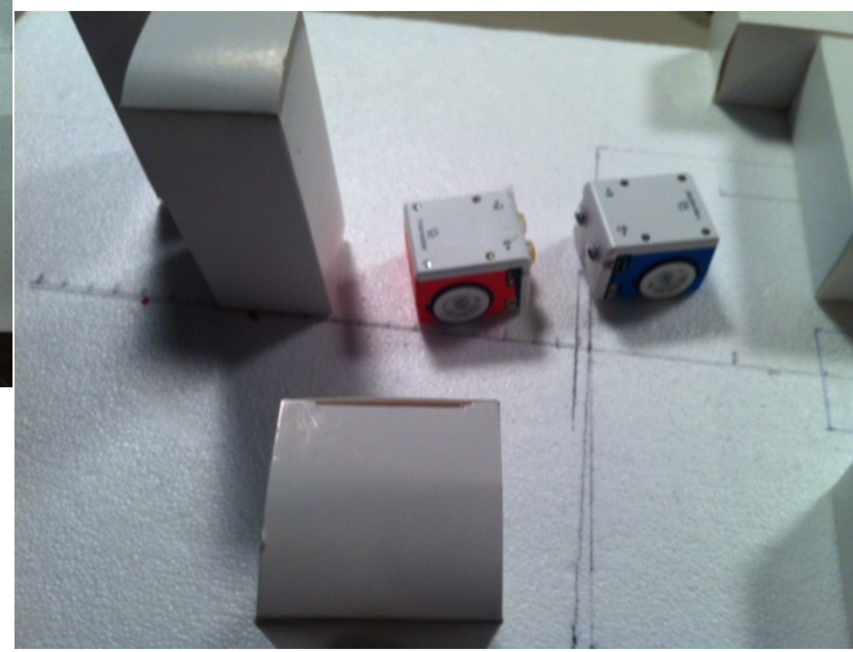
- Purpose
  - Some ideas to get your thinking going
  - You can certainly make various variance of one of these projects (or come up with entirely new ones)
  - To provide a “difficulty” reference
- Hamster “Date”
  - Two Hamsters are put into a know environment (you are given the map), but they don’t know where they are
  - They need to find out where they are, and meet up at a place
- The Couch Mover Problem

Hamster Date

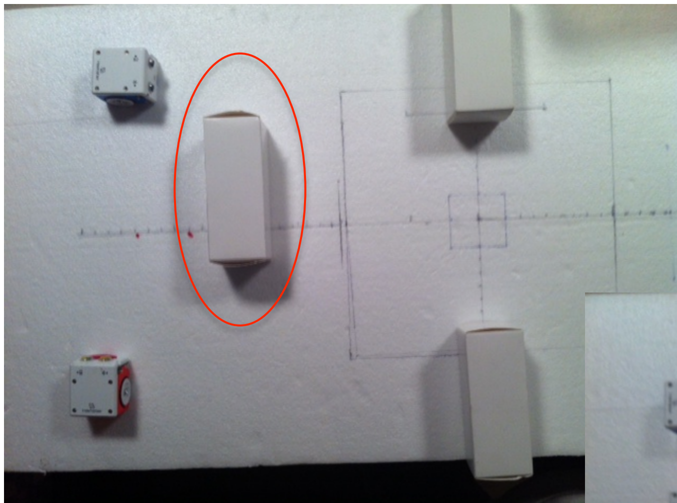


Worlds apart, separating by obstacles

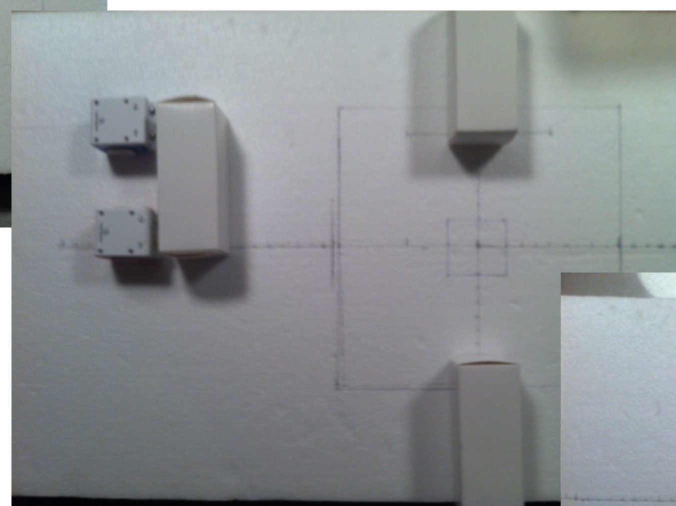
Found True Love



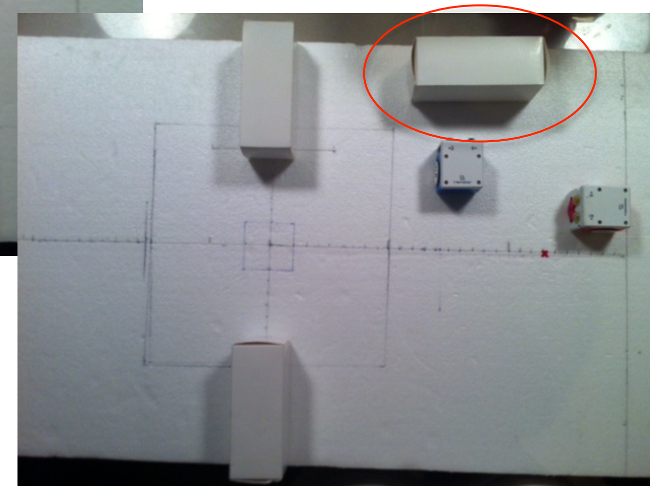
Couch Mover Problem



From Here



To Here



Some Notes on Project Proposal

- Project should be well defined
  - Precise definition of “initial state” and “final state”
  - Clearly definite assumptions

Note: the two sample projects given are not well defined enough – if you want to use one of them for your project, you need to make it more precise