



SWARM ROBOTS FOR MAPPING AND PATH PLANNING

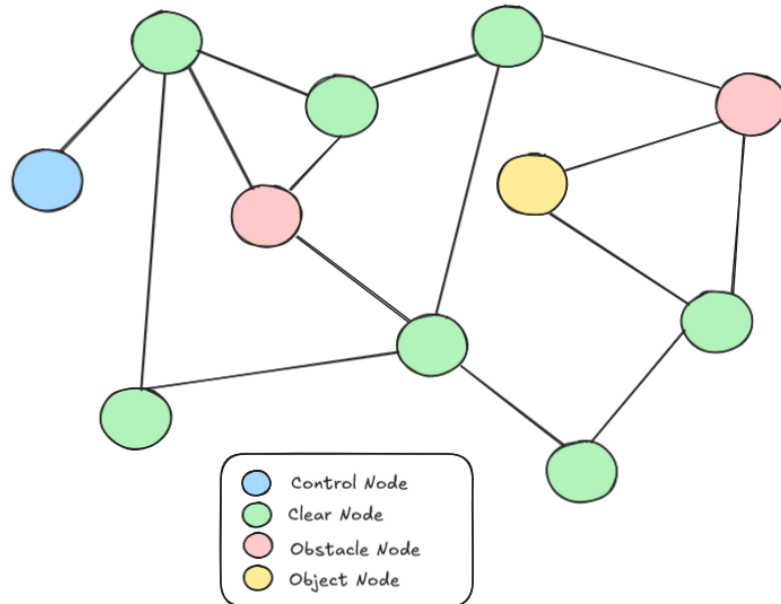
JACK GLADOWSKY, ISHAAN DESAI, WILLIAM FOX

PROJECT SCOPE AND OBJECTIVES

Project Scope

- Develop a system for generating graph environments for robots to traverse and map, so that they can retrieve various objects throughout the environment.

Graph Environment for Swarm Robot Mapping and Planning



OBJECTIVES

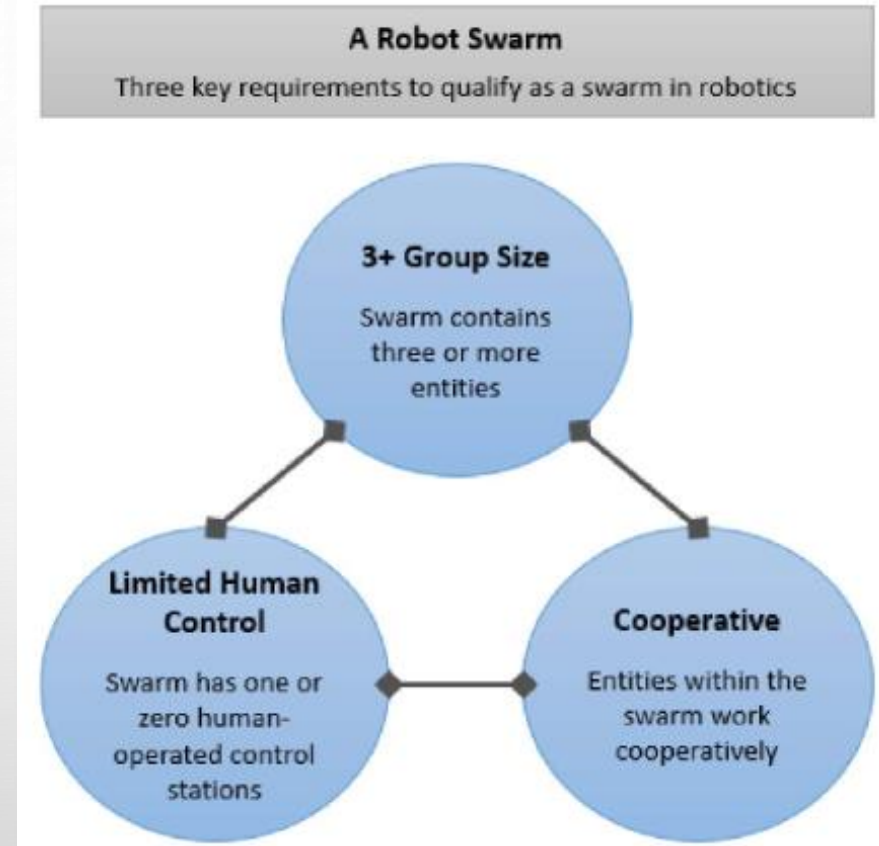
1. Develop efficient mapping algorithm for mapping graphs
2. Implementing optimized path planning strategies
3. Generating graphical representation of an explorable environment

Control Station



LITERATURE REVIEW

- What is a Robot Swarm: A Definition for Swarming Robotics – [LINK](#)
 - Defines a robot swarm as a group of 3+ entities, the entities work together, and the swarm has limited human control
 - 'Many real-world problems can be addressed through use of swarms'
- Real Applications for Swarm Robotics Applications – [LINK](#)
 - Swarm robotics can coordinate multiple robots for search-and-rescue missions, navigating complex environments to locate survivors and deliver assistance efficiently, not path constrained.
 - Groups of robotic units can work together to monitor crop health, plant seeds, and optimize irrigation, enhancing productivity while minimizing resource use.



METHODOLOGY – NODES AND GRAPHS

- A graph is just a list of nodes, where each node keeps track of its neighbors
- Nodes are generated with states assigned randomly (80% chance of Clear, 10% Obstacle, 10% Object) with 1 control point
- Nodes also store their respective neighbors, allows for robot to know where to go next
- Using stacks to hold node neighbor

Graph Generation Pseudocode

1. Loop through the number of nodes and generate randomly based on state distributions
2. Set one nodes state as the control point
3. Loop through all the nodes again
 - A. Choose a random number of neighbors
 - B. Loop through the number of neighbors and choose a random node to add as a neighbor
 - C. Check if the neighbor is already a current neighbor
 - D. Push each node to the neighbor list of each other

Time Complexity: $O(n*m)$

enum NodeState:

- CLEAR = 0
- OBSTACLE = 1
- OBJECT = 2
- CONTROL = 3

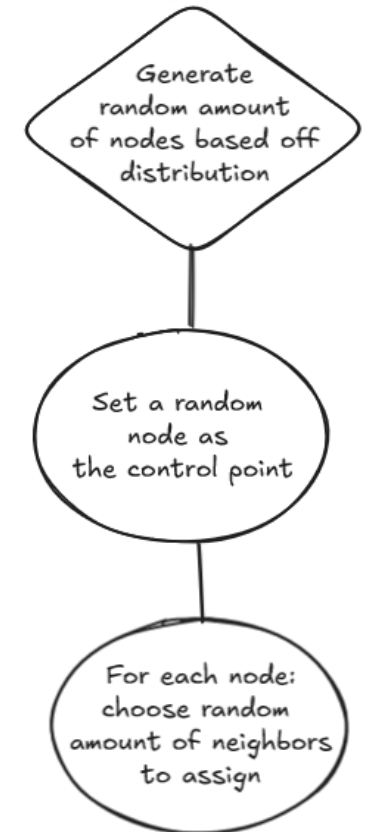
class GraphNode:

- int NodeID
- NodeState state
- Vector<GraphNode*> neighbors

class Graph:

- int numNodes
- Vector<GraphNode*> nodes
- GraphNode* controlPoint

Generating the Graph



METHODOLOGY – ROBOTS AND CONTROLLER

Controller::Map Function

- Create the robots
- Move robots to control point
- Initialize *exploredMap* with control point and its neighbors
- Loop through the robots and Move them one node at a time, updating controller's *exploredMap* with new explored nodes
- Track the number of "seen" nodes vs "visited" nodes
- Backtracking when no neighbor nodes to visit
- Stop mapping when "seen" nodes = "visited" nodes

Mapping Time Complexity

- $O(N + E^2)$ (worst case)
- $O(N + E)$ average case
- N is number of Nodes
- E is number of edges

Data Structures Used

- Graphs
- Vectors
- Stacks

- Robots are objects that traverse the graph node by node, communicating with controller to relay information about surrounding nodes and retrieve objects
- Controller handles the robots, sends/stores data from/to robots while traversing, also plans the paths for the robots to move

class Controller:

- `vector<Robot*> robots`
- `vector<GraphNode*> visitedNodes`
- `Graph* graphMap`
- `Graph* exploredMap`

class Robot:

- `int robotID`
- `GraphNode* currNode`
- `stack<GraphNode*> prevNodeStack`

Robot Move Function



HOURS SPENT PER WEEK

- 5 HOURS SPENT PER WEEK
- 4 WEEKS PER MONTH
- 20 HOURS PER MONTH
- GROUP CODING
- DISCUSSIONS

ANALYSIS AND RESULTS

RESULTS

- Created an algorithm to successfully create random graph environments
- Developed methods for robots to travel through graphs efficiently and ensuring they explore every node
- Created a platform for robots to map graphs completely to perform path planning

Hello Swarm Robots!			
Node ID: 0	ROBOT 0 CREATED.	Size of explored: 10	Size of explored: 10
Node State: OBSTACLE	ROBOT 1 CREATED.	Size of visited: 5	Size of visited: 10
Neighbors: 5 4 7 8 9	ROBOT 2 CREATED.		all nodes have been visited
Node ID: 1	Start Mapping...	Current Robot: 1	Printing Explored Graph...
Node State: CONTROL	On node: 1	value pushed to prevNodeStack: 8	
Neighbors: 4 5 8 9	Control Node neighbors: 4 5 8 9	rand neighbor selected: 3	
Node ID: 2	Pushing node1 to explored map	Node ID: 3	Node ID: 1
Node State: CLEAR	Pushing node 4 to explored map	Node State: OBSTACLE	Node State: CONTROL
Neighbors: 6 3 8 7	Pushing node 5 to explored map	Neighbors: 2 5 6 8 9	Neighbors: 4 5 8 9
Node ID: 3	Pushing node 8 to explored map	pushing 3 to visited	
Node State: OBSTACLE	Pushing node 9 to explored map	added 3 <=> 2	Node ID: 4
Neighbors: 2 5 6 8 9		added 3 <=> 5	Node State: CLEAR
Node ID: 4	Current Robot: 0	added 3 <=> 6	Neighbors: 1 0 7 9 5
Node State: CLEAR	value pushed to prevNodeStack: 1	Size of explored: 10	
Neighbors: 0 1 7 9 5	rand neighbor selected: 4	Size of visited: 6	Node ID: 5
Node ID: 5	Node ID: 4	Current Robot: 2	Node State: CLEAR
Node State: CLEAR	Node State: CLEAR	value pushed to prevNodeStack: 9	Neighbors: 1 4 0 3 7
Neighbors: 0 1 3 4 7	Neighbors: 0 1 7 9 5	rand neighbor selected: 6	
Node ID: 6	pushing 4 to visited	Node ID: 6	Node ID: 8
Node State: OBSTACLE	Pushing node 0 to explored map	Node State: OBSTACLE	Node State: CLEAR
Neighbors: 2 9 3	added 4 <=> 0	Neighbors: 2 9 3	Neighbors: 1 2 3 0
Node ID: 7	Pushing node 7 to explored map	pushing 6 to visited	
Node State: CLEAR	added 4 <=> 7	added 6 <=> 2	Node ID: 9
Neighbors: 4 0 2 5	added 4 <=> 9	Size of explored: 10	Node State: CLEAR
Node ID: 8	added 4 <=> 5	Size of visited: 7	Neighbors: 1 4 6 3 0
Node State: CLEAR	Size of explored: 7		
Neighbors: 2 1 3 0	Size of visited: 2	Current Robot: 0	Node ID: 0
Node ID: 9	Current Robot: 1	value pushed to prevNodeStack: 0	Node State: OBSTACLE
Node State: CLEAR	value pushed to prevNodeStack: 1	rand neighbor selected: 5	Neighbors: 4 8 9 5 7
Neighbors: 4 6 3 1 0	rand neighbor selected: 8	Node ID: 5	
Object Count: 1	Node ID: 8	Node State: CLEAR	Node ID: 7
Obstacle Count: 3	Neighbors: 2 1 3 0	Neighbors: 0 1 3 4 7	Node State: CLEAR
Clear Count: 6	pushing 8 to visited	pushing 5 to visited	Neighbors: 4 0 5 2
	Pushing node 2 to explored map	added 5 <=> 7	
	added 8 <=> 2	Size of explored: 10	Node ID: 2
	Pushing node 3 to explored map	Size of visited: 8	Node State: CLEAR
	added 8 <=> 3		Neighbors: 8 3 6 7
	added 8 <=> 0	Current Robot: 1	
	Size of explored: 9	value pushed to prevNodeStack: 3	Node ID: 3
	Size of visited: 3	rand neighbor selected: 2	Node State: OBSTACLE
	Current Robot: 2	Node ID: 2	Neighbors: 8 9 2 5 6
	value pushed to prevNodeStack: 1	Node State: CLEAR	
	rand neighbor selected: 9	Neighbors: 6 3 8 7	Node ID: 6
	Node ID: 9	pushing 2 to visited	Node State: OBSTACLE
	Node State: CLEAR	added 2 <=> 7	Neighbors: 9 3 2
		Size of explored: 10	
		Size of visited: 9	
		Current Robot: 2	Object Count: 0
		All possible neighbor nodes have been visited.	Obstacle Count: 0
			Clear Count: 0

CONCLUSION

- Overall, gave idea of how professional software development teams operate
 - Improved our understanding of graph theory
- Strengthened our problem solving and algorithm design skills

Projects Limitations/Need to Implement

- Have platform for path finding and object retrieval but still need to implement
 - Going to use a modified Dijkstra's algorithm
 - Implement obstacle node avoidance

Future Work

- Create a better interface for managing and analyzing swarm status
 - Add different node states for more variability