**CSCI 8480 Project Report**

**Multi-agent Rendezvous problem solution using stochastic game algorithm**

**by**

**Mohan Ambati & Venkat Garlapati**

Table of Contents

[1. Description 1](#_Toc481611187)

[2. Algorithms 2](#_Toc481611188)

[2.1 Markov Decision Process: 2](#_Toc481611189)

[2.2 Stochastic Game: 3](#_Toc481611190)

[3. Implementation 4](#_Toc481611191)

[3.1 Programming Environment: 4](#_Toc481611192)

[3.2 Setup 5](#_Toc481611193)

[4. Evaluation Criteria 5](#_Toc481611194)

[Milestones: 6](#_Toc481611195)

[5. Results 7](#_Toc481611196)

[6. Conclusion 10](#_Toc481611197)

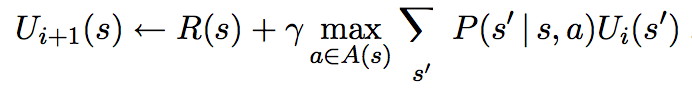
1. **Description**  
    In this project, we are concerned with the collective behavior of a group of n >1 mobile agents, which can all move in a plane. The action set of each agent is {N, W, S, E, Stay}. The multi-agent rendezvous problem is to devise strategies for each agent to cause all the agents to eventually rendezvous at a single specified location. The approach is to use stochastic game where the agents repeatedly play games from the collection of normal form games, and the particular game played at any given iteration depends probabilistically on the previous game played and on the actions taken by all agents in that game. The game is played in sequence of stages. At the beginning of each stage the game is in some state. The players select actions and each player receives a payoff that depends on the current state and the chosen actions. The game then moves to a new random state whose distribution depends on the previous state and the actions chosen by the players. The procedure is repeated at the new state and play continues for a finite number of stages until all players reach a goal location. Here, each state is a normal form game played by ‘n’ agents. The transition probability is the probability of transitioning from one state to other state after joint action. Payoff matrix is generated for each agent after every game using value iteration algorithm of Markov Decision Process. The objective of the project is to plan a collision free path for each player so that all the players in the game reach common goal location by minimizing the length of the path.
2. **Algorithms**

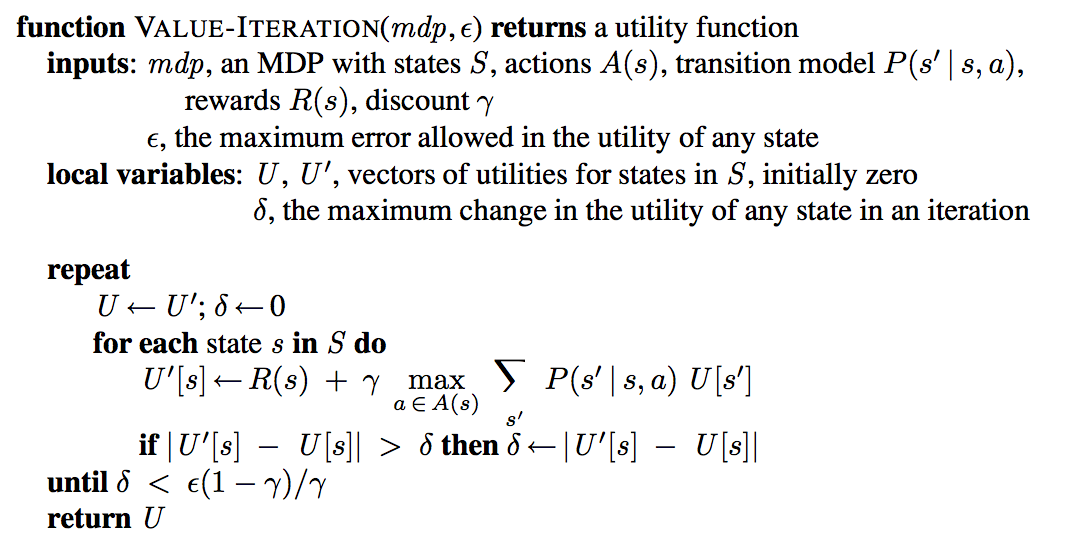
A Multi-Agent Rendezvous Problem (MARP) algorithm essentially consists of the following steps:

* Utility generation: In this step payoff for each robot is generated for every game for their respective actions using value iteration algorithm of Markov Decision Process.
* Stochastic game: At each state, a new game is being played by the agents and their actions leads to new state with different game and this process is continued until they reach the end point.

## **Markov Decision Process:**

The utility for each agent is generated based on the Markov Decision Process value iteration algorithm. The Bellman equation is the basis of the value iteration algorithm for solving MDPs. If there are n possible states, then there are n Bellman equations, one for each state. The n equations contain n unknowns—the utilities of the states. So we would like to solve these simultaneous equations to find the utilities. There is one problem: the equations are nonlinear, because the “max” operator is not a linear operator. Whereas systems of linear equations can be solved quickly using linear algebra techniques, systems of nonlinear equations are more problematic. One thing to try is an iterative approach. We start with arbitrary initial values for the utilities, calculate the right-hand side of the equation, and plug it into the left-hand side—thereby updating the utility of each state from the utilities of its neighbors. We repeat this until we reach an equilibrium. Let U*i*(s) be the utility value for states at the *i* th iteration. The iteration step, called a Bellman update, looks like this





## **Stochastic Game:**

A stochastic game is a collection of normal-form games that the robots play repeatedly. The particular game played at any time depends probabilistically on previous game played and the actions of the robots in that game.

A stochastic (or Markov) game includes the following:

* a finite set Q of states (games),
* a set N = {1, …, n} of robots,
* For each robot i, a finite set Ai of possible actions
* A transition probability function

P : Q × A1 ×· · ·× An × Q → [0, 1] P(q, a1, …, an , q’) = probability of transitioning to state q’ if the action profile (a1, …, an) is used in state q, which is considered as fully deterministic.

* For each robot i, a real-valued payoff function ri : Q × A1 ×· · ·× An → ℜ

The algorithm is as follows:

1. Get the initial locations of the robots.
2. Perform MDP value iteration function and generate payoffs.
3. The generated payoffs are used to define a new game after every stage.
4. Find the Nash Equilibrium to devise the optimal strategy for each robot to move towards the end position by avoiding the collisions with other agents.

# **Implementation**

## **Programming Environment:**

The Value Iteration and Stochastic Game algorithms are implemented using python 2.7 programming language. A desktop GUI application has been developed to simulate the agent’s movement to using Tkinter which is a python standard GUI package. The below figure explains the flow of control between the essential modules of the application. After each game, the agent’s current position and the travelled path are updated on the GUI application.

Stochastic Game

If Goal reached

**Start No**

Value Iteration

Draw GUI

**Yes**

Stop

## **Setup**

The following are the requirements to startup the Multi-Agent Rendezvous application:

* + Python 2.7
  + Tkinter
  + Ubuntu/Windows/MacOS

Steps to run and test the project:

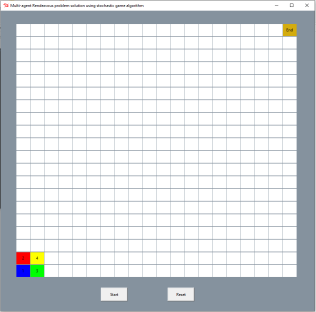
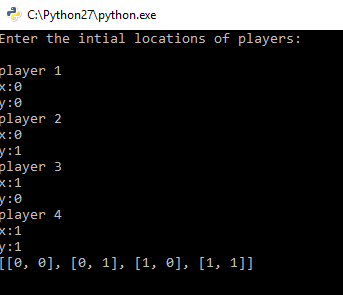
1. Extract “.py” files to same folder eg: - /Users/JohnDoe/Marp
2. Open terminal and navigate to above mentioned folder
3. Execute the below command

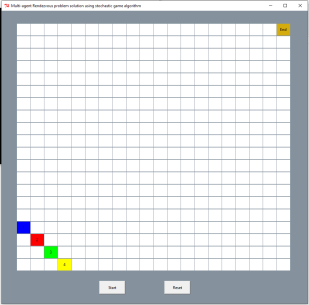
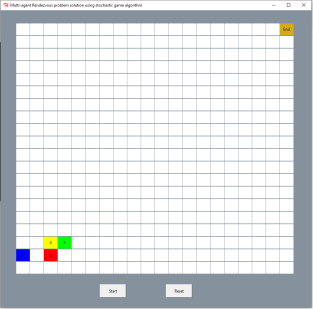
$ python marp.py

1. Initial positions of the agents are given as input on the terminal
2. Push the start button on the GUI

# **Evaluation Criteria**

The implementation of the project is evaluated with various starting locations of the agents on 20x20 grid environment. Below are some of the starting configurations of the multiple agents shown on the GUI desktop application. The agents path is marked with the color of the agent itself. The algorithm is rigorously tested and validated that the agents are not colliding and following the shortest path possible to reach goal location.



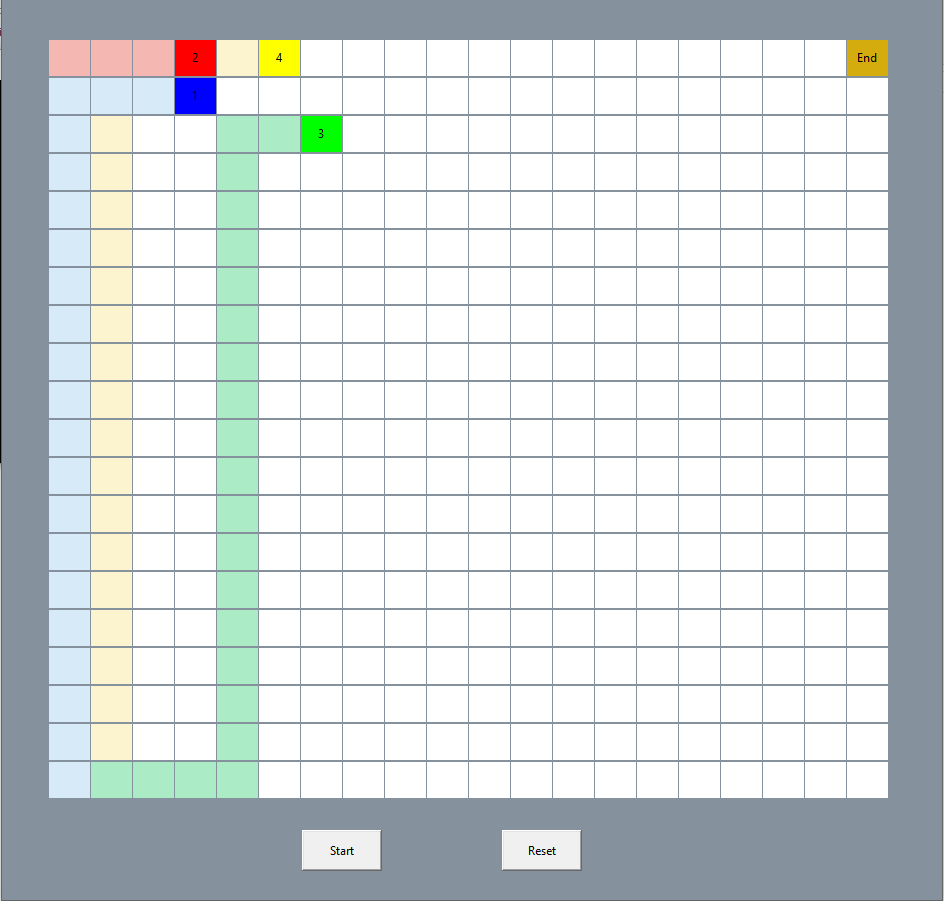
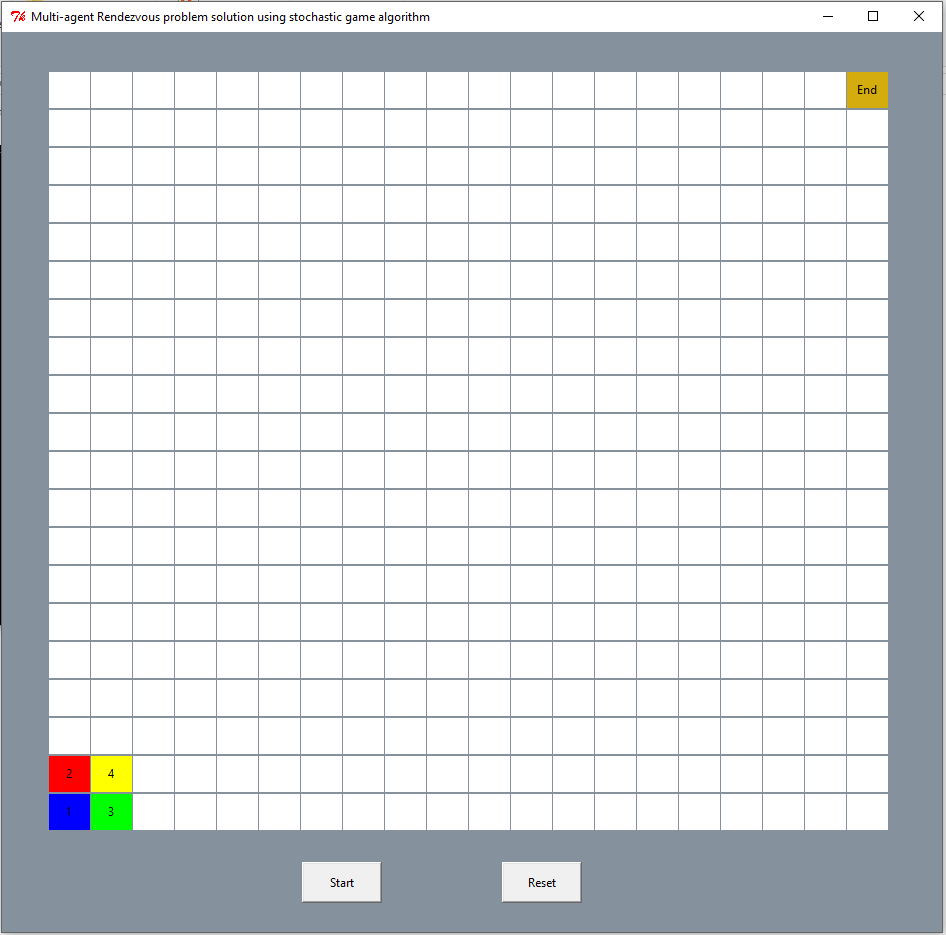
## **Milestones:**

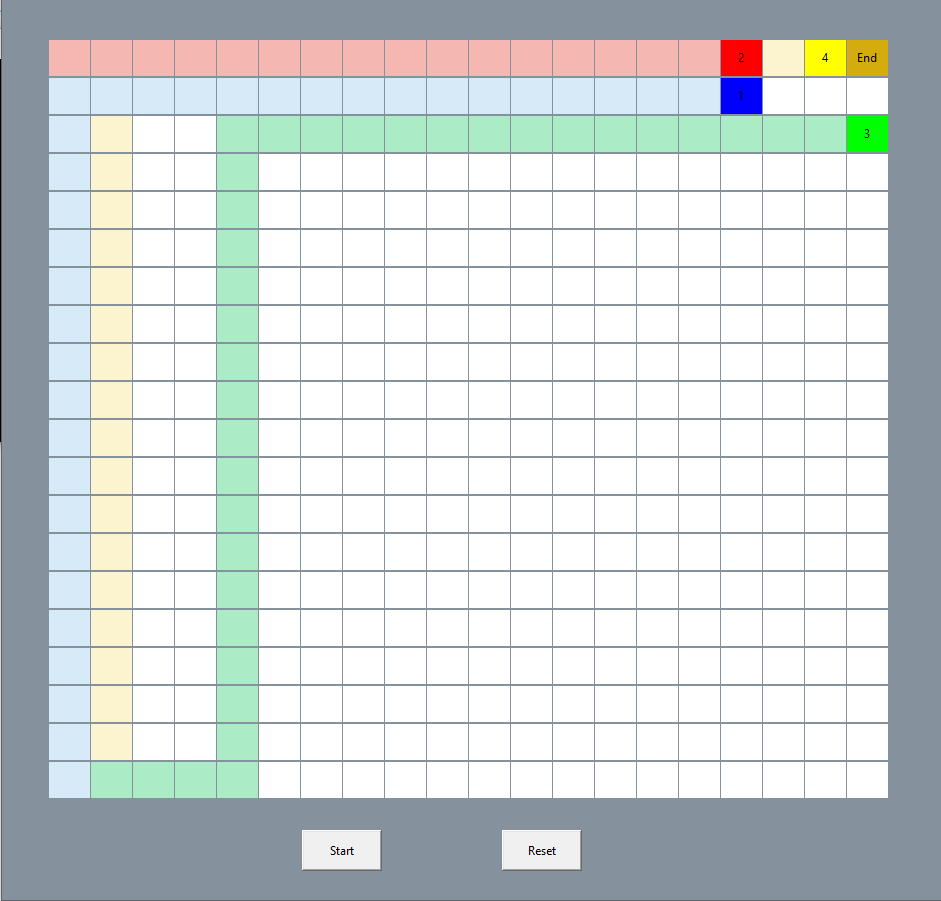
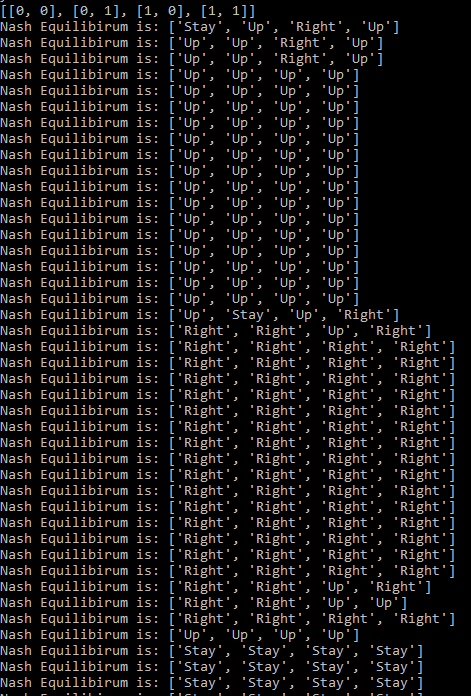
|  |  |  |
| --- | --- | --- |
| Phases | Description | Start Date – End Date |
| Phase 1 | 1. Analyze & Create Model for the problem  * UI Design and Development. * Model Design and Development. | March 6th – March 20th |
| Phase 2 | 1. Define transition probability function.  * We are considering the probability of transition from one game to other is deterministic.  1. Define payoff function(rewards)  * We are considering value iteration algorithm in MDP’s to get the payoffs for robots actions. | March 21st – April 3rd |
| Phase 3 | 1. Implementation | April 4th – April 17th |
| Phase 4 | 1. Test Cases 2. Results 3. Final Report | April 18th – April 27th |

# **Results**

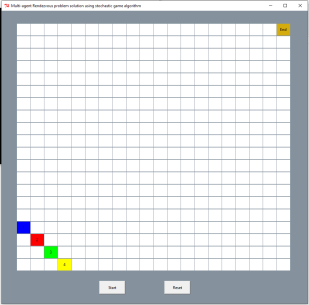
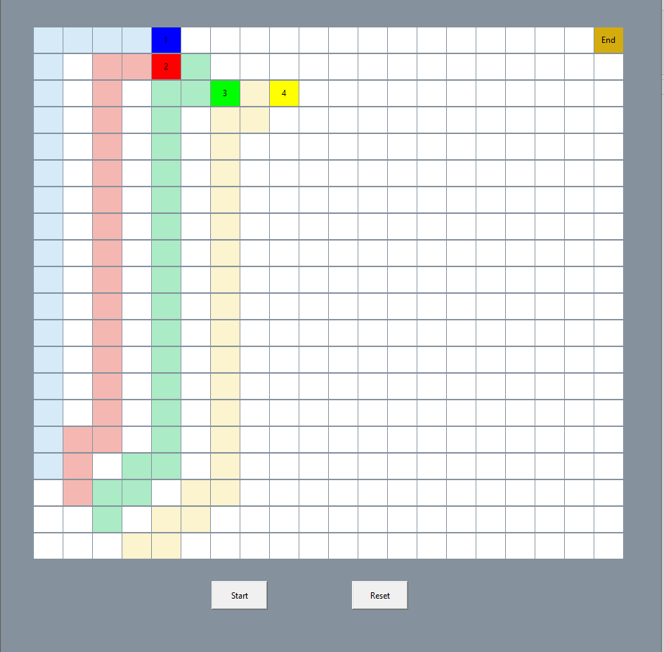
The multi-agent rendezvous algorithm is tested using many test scenarios. A new simulation GUI application is created to view how the agents move from initial location to goal location by avoiding collision. For all the test cases the results clearly show that the agents are choosing their best strategies based on the other agent’s strategies. At every state, a new game is formed and a nash equilibrium is calculated by each agent to move to the next state. Below figures show some of the test scenarios with agents at initial location, agents moving towards goal location, and agents reaching goal location.

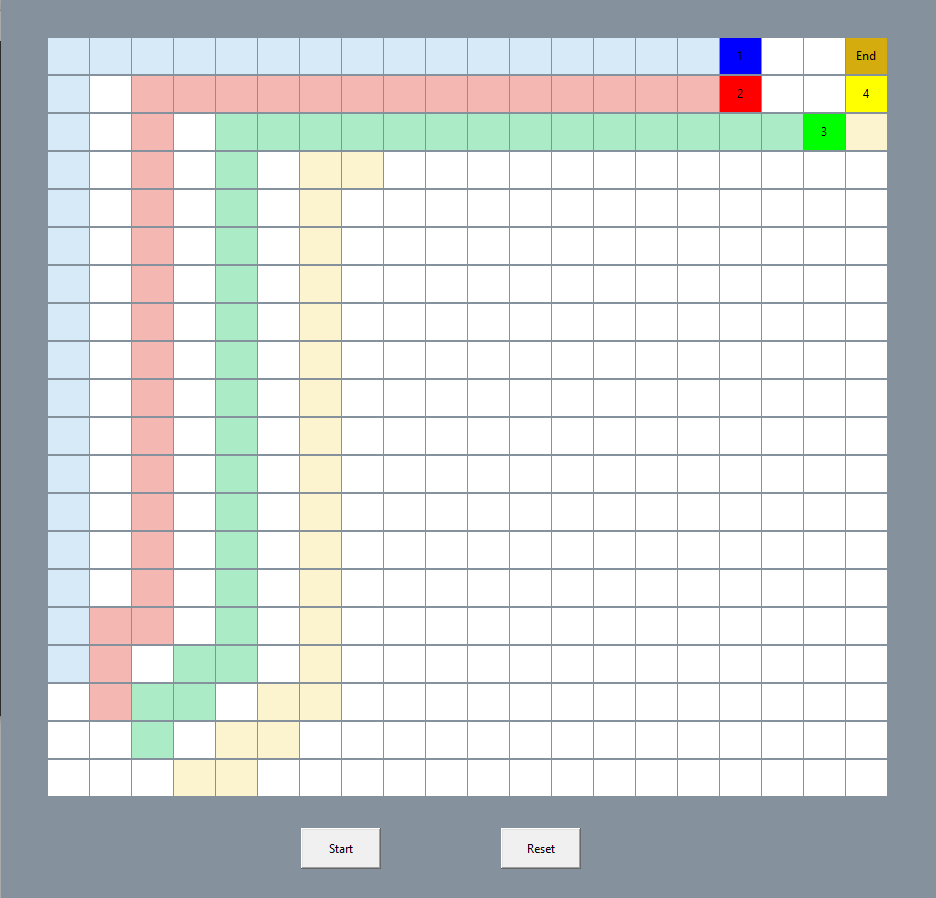
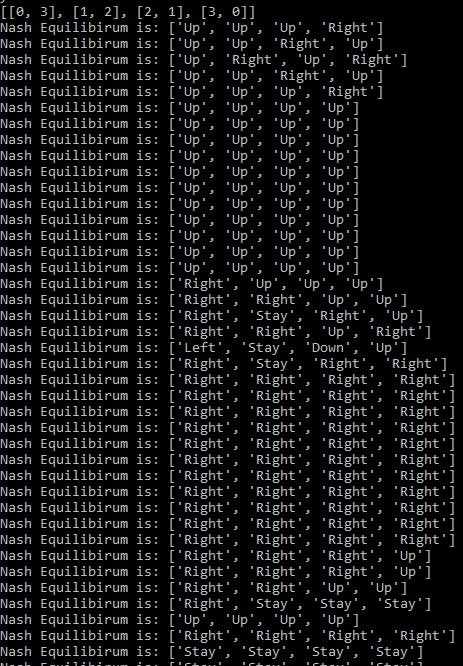
**Test Case 1:**



**Test Case 2:**

# **Conclusion**

The intention of the project was to investigate how to solve multi-agent rendezvous problem using stochastic games. As a summary of the results in this project, it can be said that the state of the art algorithm for Multi-Agent Rendezvous problem was developed and implemented using stochastic games. The algorithm is able to devise strategies for each agent to cause all the agents to eventually rendezvous at a single specified location while avoiding collisions.