AMONGUS-AI

github link - amongus-ai

Team: 2301AI28, 2301AI50, 2301CS01, 2301CS26, 2301CS27, 2301CS63

Inspiration

This game is inspired from https://blogs.cornell.edu/info2040/2021/09/22/use-game-theory-to-decide-your-next-action-in-werewolf/. We have an uninformed majority and we play as the informed minority.

Game utility

Majority i.e bots have tasks assigned to them around which the move and if all bots complete their task before count drops below 3 then bots win otherwise we win i.e imposter.

Real life-agents:

Bots have been given a sight radius which depends upon a constant we could change and they store the following information:

```
def update(self, dt):
        """ Updates bot movement, pauses randomly, and changes direction at intervals """
        # int cnt=0;

        if(stop==1):
            return

        cnt=0
        for j in killed_players:
           if(j==self.i):
               cnt=cnt+1
        if(cnt!=0):
            return
        num=random.randint(1,20)

        # if((self.rect.center[0]-my_player_x)**2 +(self.rect.center[1]-my_player_y)**2<=min_see_distance**2):
        #     list_player_tasks.append([my_player_x,my_player_y])


        if(num%3==0):
            mini=0
            min_dist=999999999
            for j in range(0,10):
                if(j==self.i):
                    continue
                if((self.rect.center[0]-mpp[j][0])**2 +(self.rect.center[1]-mpp[j][1])**2<=min_dist**2):
```

```
                    mini=j
                    min_dist=(self.rect.center[0]-mpp[j][0])**2 +(self.rect.center[1]-mpp[j][1])**2

            # if((self.rect.center[0]-my_player_x)**2 +(self.rect.center[1]-my_player_y)**2<=min_dist**2):
            #     mini=my_id

            frequency_counter[self.i][mini]+=1


        cur_x=self.rect.center[0]
        cur_y=self.rect.center[1]

        for i,(j,k) in recent_killed:
            if((self.rect.center[0]-j)**2+(self.rect.center[1]-k)**2<=min_body_found**2):
                meeting_called()
```

There is a kill cooldown of 20 sec adjustable according to the user.

We have chosen a probabilistic model to train bots. For individual bot i am checking whether their probability towards me i.e imposter id-12 is maxium or not among other alive players if yes + rewards else -.

Probability calculation:

```
def past_haunts_you(id):
        for i in range(13):
            if i == id:
                continue
            probability_for_each[id][i] *= weights[0]

    def how_many_times_i_saw_you(id):
        for i in range(10):
            if i == id:
                continue
            probability_for_each[id][i] += weights[1] * frequency_counter[id][i]

    def kill_radius_found(id):
        center_x, center_y = recent_killed[-1][1]
        time_elapsed = time.time() - min_kill_time

        for j in range(10):
            probability_for_each[j][12] += weights[2] * time_elapsed

        for i in range(10):
            dist = ((center_x - mpp[i][0]) ** 2 + (center_y - mpp[i][1]) ** 2) ** 0.5
            for j in range(10):
                probability_for_each[j][i] += (weights[3] * time_elapsed - dist)

    def i_saw_you_do_that(id):
        if id in killed_players:
            return
```

```
        bnt = sum(1 for j in list_of_tasks if j not in list_player_tasks)
        probability_for_each[id][my_id] += bnt * weights[4]
        for i in range(10):
            if i == id:
                continue
            probability_for_each[id][i] += weights[5]

    # **State before voting**
    state = []
    for i in range(10):
        if i in killed_players:
            continue
        state.append(probability_for_each[i][12])  # Probability of voting against impostor
```

Functions

Past_haunts_you- Takes previous round data multiplies by a constant and adds to the new cumulative probability and called first among other functions for obvious reasons.

How_many_times_i_saw_you- Kind of inspired by real life game play where if someone follows you for a long time he automatically becomes sus and probability inc towards him/her.

kill_radius_found- This is the probability given by the admin i.e after the latest kill sort of timer will start and people inside a certain radius decided by that time.

I_saw_you_do_that- Every bot moves and observed id=12 in particular, and checks what tasks am i faking and notes them and while answering in terminal if said tasks match with our tasks then probability towards us decreases else increase.

Future Prospects

We could define better movement algorithms for bot movement.

Define better reinforcement learning for bot training.

Do a learning over constants defined in constants.py

Deploy this game over web and make it much more difficult.