Path Planning Using RRT Algorithm

Based on WEBOTS Simulation Environment

Xiong Junlin
Advisor:Zoltán Gyenes

Research Background

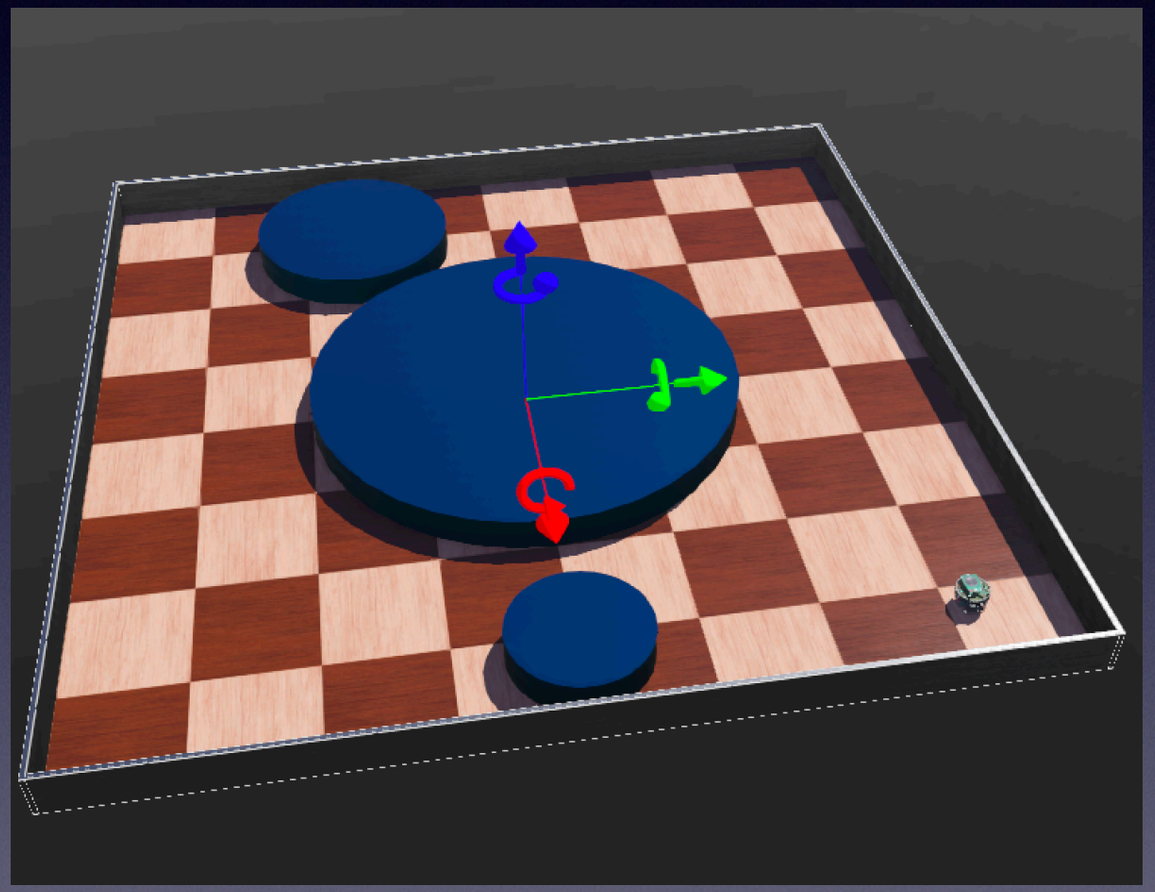
- Path planning is a crucial aspect of robotics, involving the **determination of an optimal path** from a start point to an endpoint while avoiding obstacles.

- The Rapidly-exploring Random Tree (RRT) algorithm is a popular method for solving path planning problems due to **its efficiency in high-dimensional spaces** and it is **easy to implement and scalable**.

Webots Simulation Environment

- WEBOTS is a professional mobile robot simulation software that allows the **modeling, programming, and simulation** of mobile robots.

- It provides a comprehensive development environment for robots, making it an ideal platform for testing and validating path planning algorithms like RRT.



Webots World Setup

- Map Setup

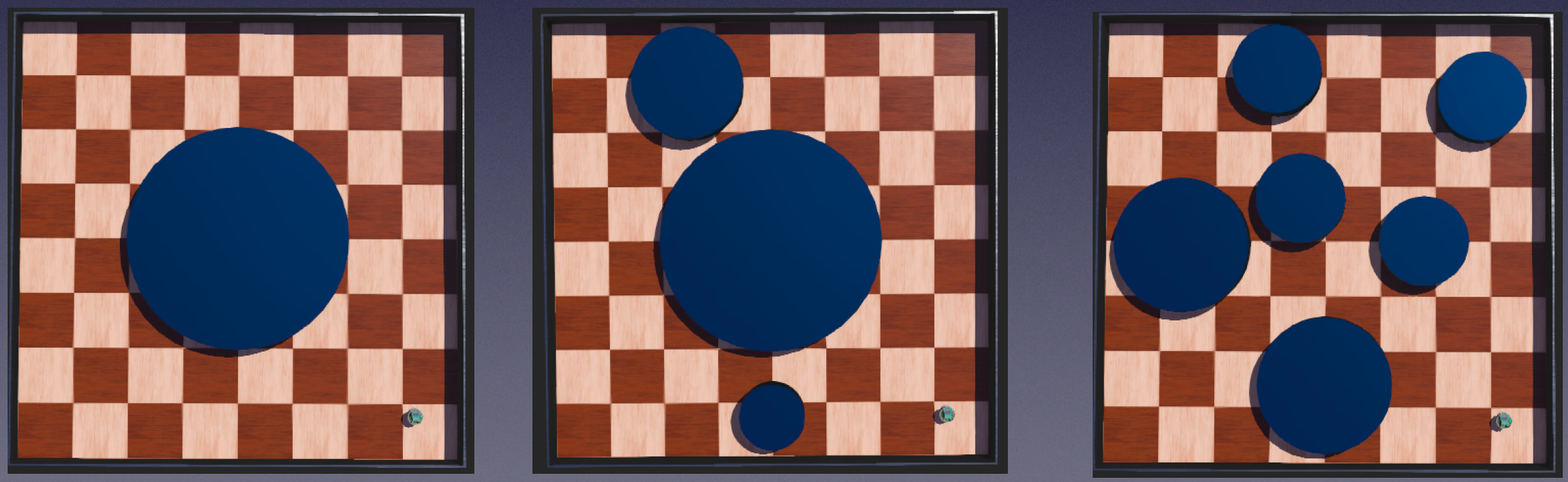
Description of three 2*2 worlds

Number of cylindrical obstacles in each map (1, 3, 6)

- Path Planning Start and Goal Points

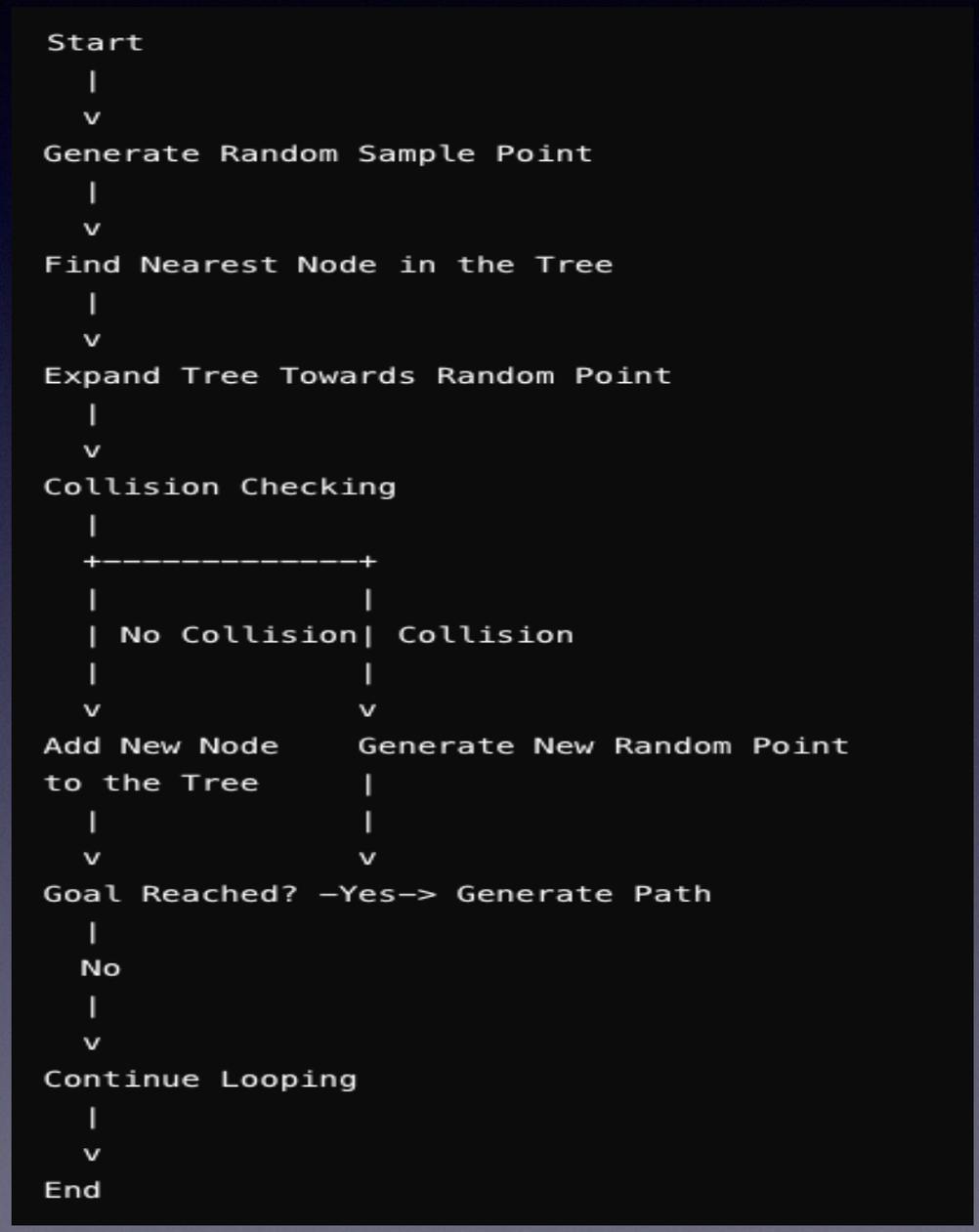
Start Point: (0.8, 0.8)

Goal Point: (-0.8, -0.8)



RRT Algorithm

1. **Start**
2. **Generate Random Sample Point**
3. **Find Nearest Node in the Tree**
4. **Expand Tree Towards Random Point**
5. **Collision Checking**
   - No Collision -> **Add New Node to the Tree**
   - Collision -> **Generate New Random Point**
6. **Goal Reached?**
   - Yes -> **Generate Path**
   - No -> **Continue Looping**
7. **End**



Problem and solution

1. Obstacle and Safety Distance Handling

Solution:

Implement collision detection with a safety margin using the is_collision function.

```python
def is_collision(self, q_new):
    """Check if the new vertex is within any of the obstacles"""
    for center, radius in self.obstacles:
        distance_to_obstacle = math.dist(q_new, center)
        if distance_to_obstacle <= radius + 0.04:  # Safety distance of 0.04
            return True
    return False
```

Problem and solution

2. Goal Reaching and Infinite Loop Prevention

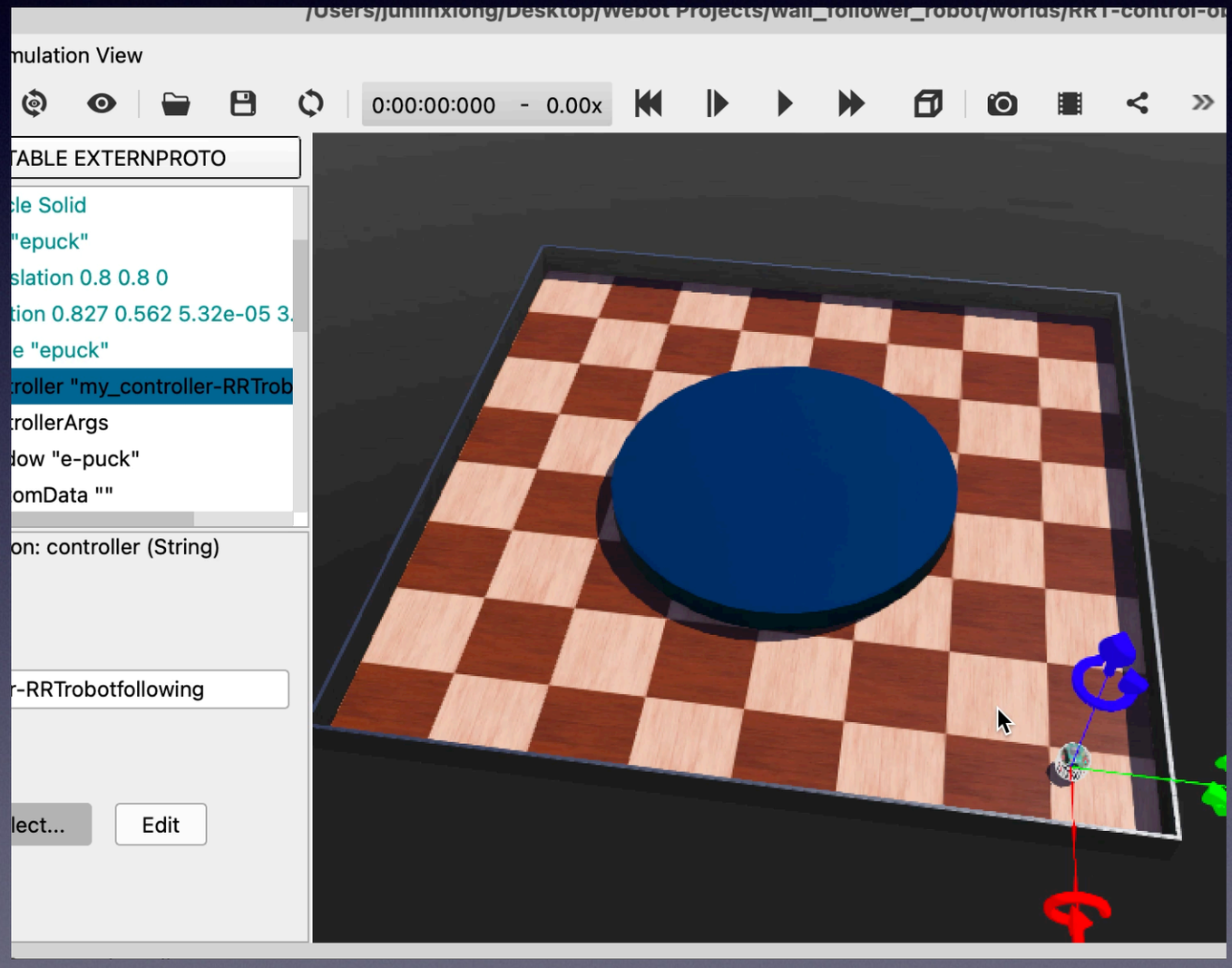
Solution:

Limit iterations and check the distance to the goal in each step.

```
if math.dist(q_new, self.q_goal) <= self.delta:
    self.q_list.append(self.q_goal)
    self.parent[tuple(self.q_goal)] = tuple(q_new)
    break
```

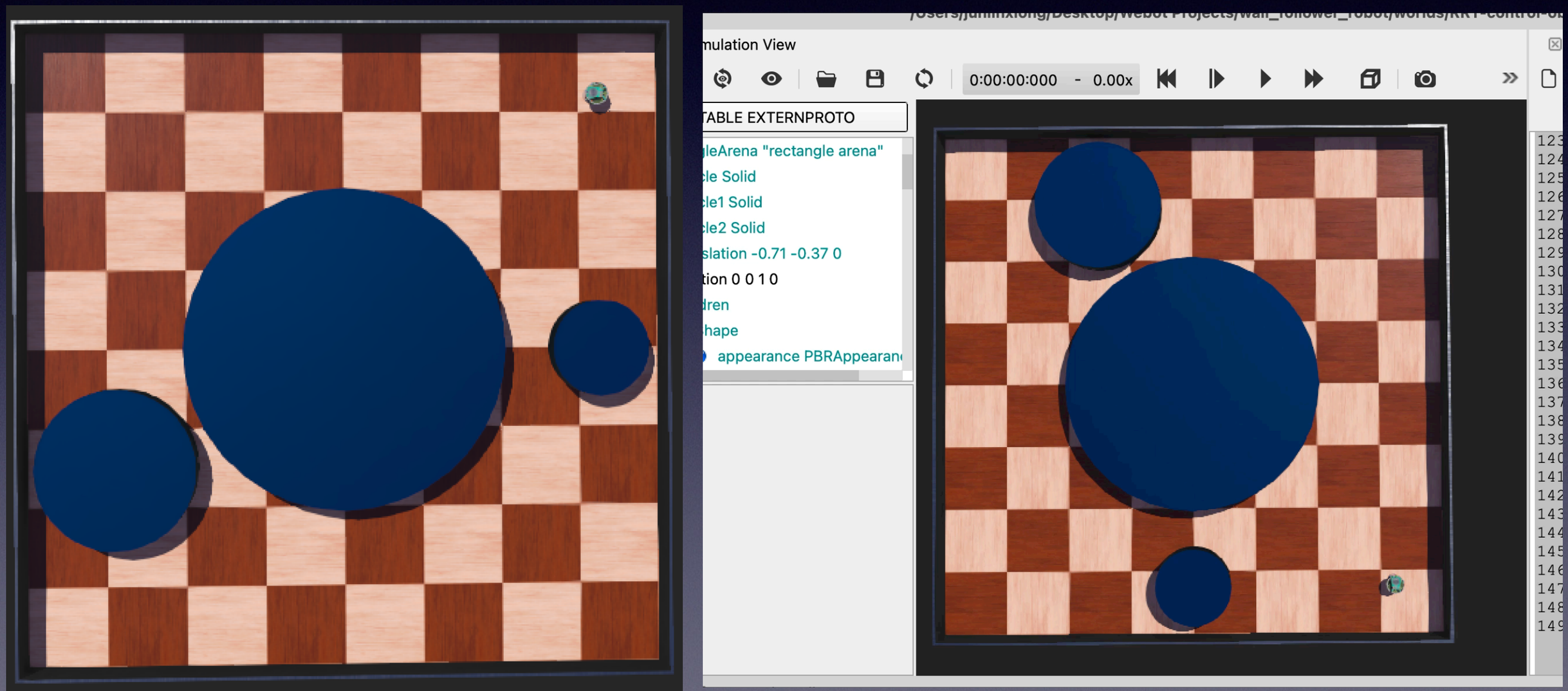
Example

1. Obstacal = 1 Without Safety Distance Handling



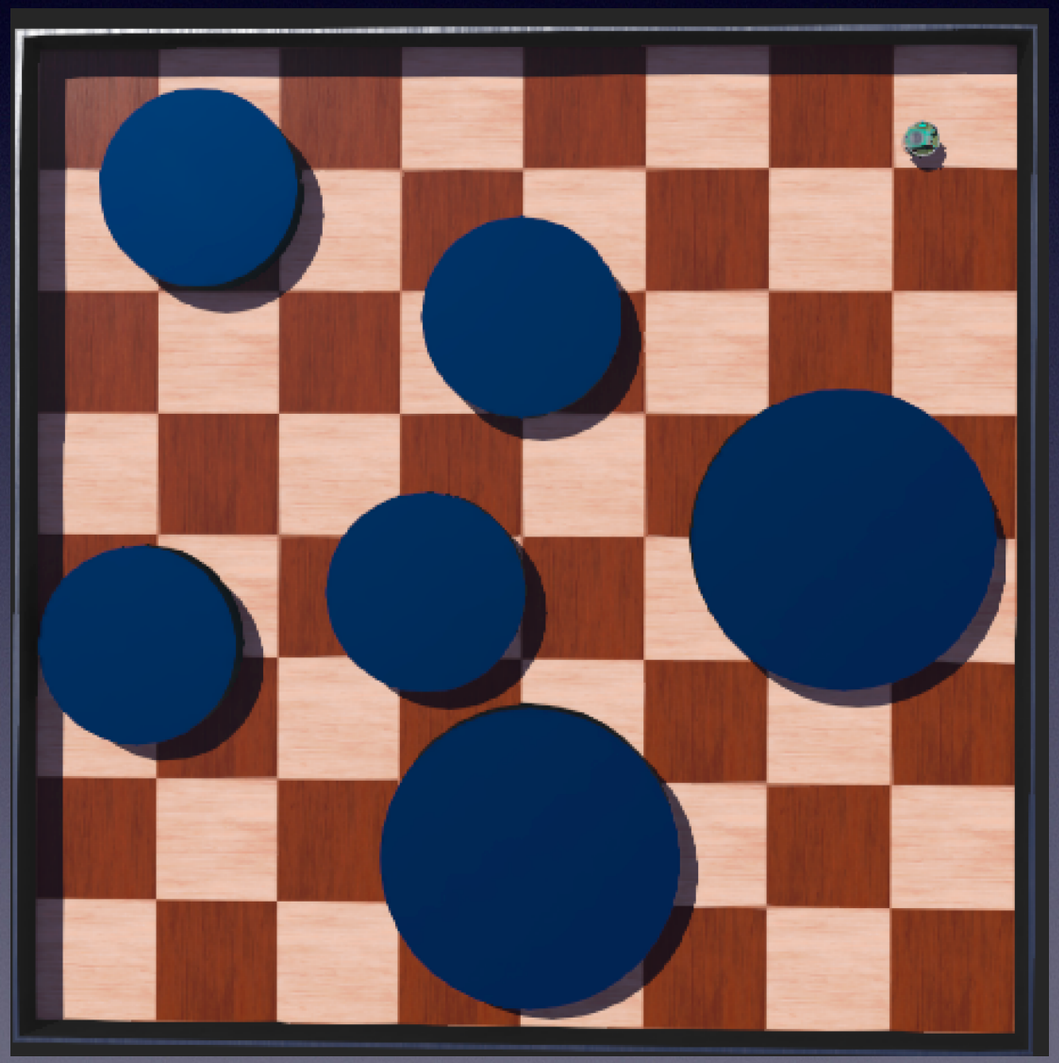
Example

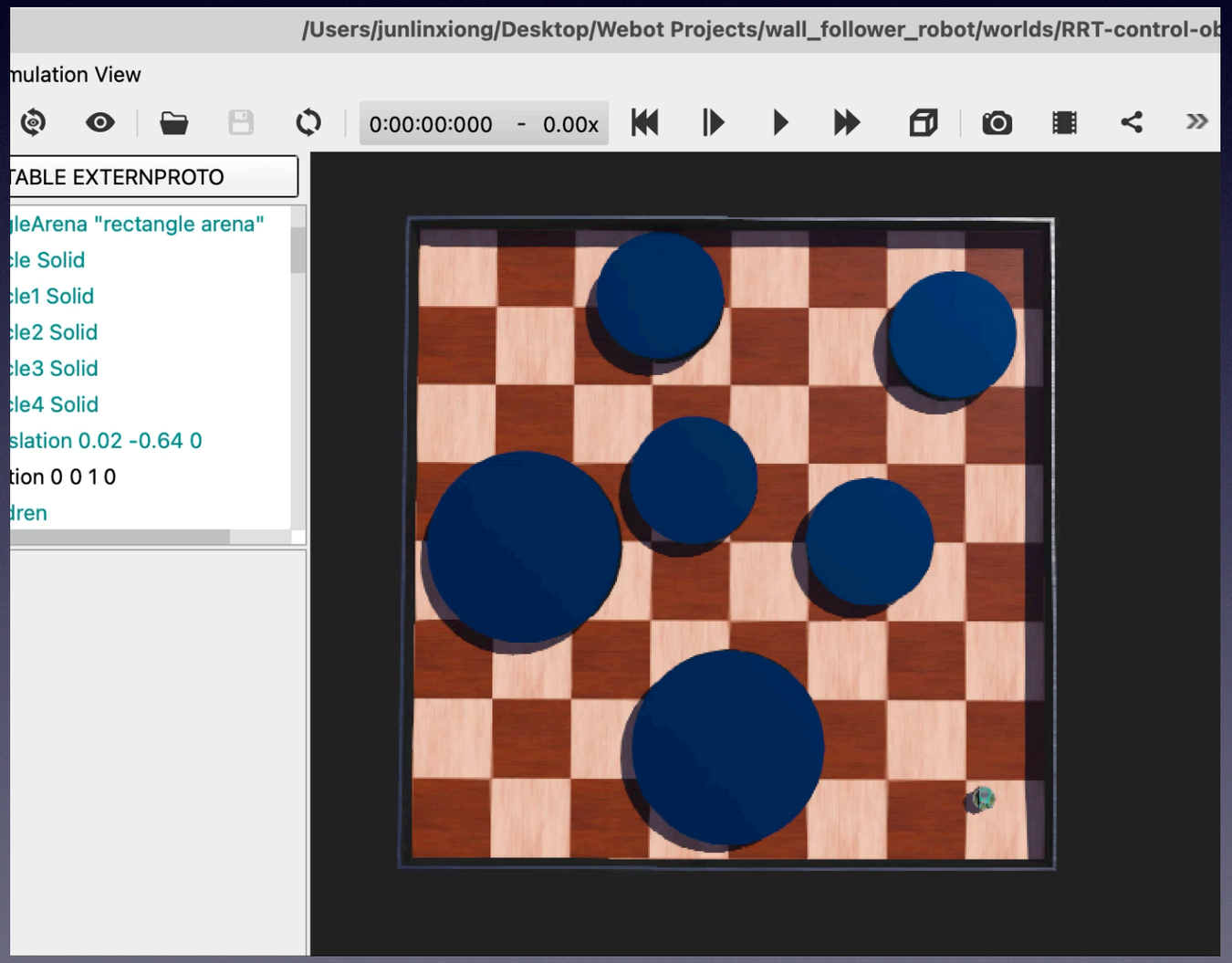
2. Obstacals = 3 With Safety Distance Handling



Example

3. Obstacals = 6 With Safety Distance Handling





Future Plan

- Integrate RRT with dynamic path planning algorithms to achieve more complex environment navigation.

- Implement robot path tracking in Webots to follow the planned route.

Thank You