

Contents

[GeoMates Navigation System: Intelligent Cooperative Agents Project Report 3](#_heading=h.250nq0ydjmtf)

[1. Executive Summary 3](#_heading=h.6xbjsbc07r09)

[2. System Architecture 3](#_heading=h.oevn3o8hupmz)

[2.1 Disc Navigation Agent 3](#_heading=h.s1vsp3n4hs7p)

[2.2 Rectangle Navigation Agent 4](#_heading=h.xpmtjjfv8xsu)

[3. Theoretical Framework and Complexity Analysis 4](#_heading=h.7xvku0js72ur)

[3.1 Agent Decision Model 4](#_heading=h.4h7zcgrp8ccv)

[3.2 Spatial Reasoning 5](#_heading=h.h4zayvjief3z)

[4.](#_heading=h.ym8i5heacl6o) Implementation Details 5

[4.1 Boundary Logic 5](#_heading=h.c8vjj4lnlzv7)

[4.2 Geometric Considerations 5](#_heading=h.g4wsf0i5umhp)

[5.](#_heading=h.1eb2grwj06vp) Experimental Results 6

[5.1 Navigation Test Scenarios 6](#_heading=h.od8kvd9m22z2)

[5.2 Performance Analysis 6](#_heading=h.6rpdzez30pdy)

[5.3 Cooperative Behavior Testing 6](#_heading=h.k5vemqb99ogx)

[6. Technical Correctness Assessment 6](#_heading=h.v3locsa4p8pa)

[6.1 Logic Consistency 7](#_heading=h.4i8ze2a76k04)

[6.2 Edge Case Handling 7](#_heading=h.lnzfolcuz0bb)

[7. Future Development Recommendations 8](#_heading=h.9dzipxdv9u7o)

[7.1 Agent Improvement 8](#_heading=h.gu6t3agyftxi)

[7.2 Experimental Framework Enhancement 8](#_heading=h.dqtgu4dvg18)

[8. Conclusion 8](#_heading=h.dzh8ts8zw0uc)

[9. References 8](#_heading=h.shcgwbfi6awg)

[Appendix A: Source Code 9](#_heading=h.sau5u81ck1us)

# GeoMates Navigation System: Intelligent Cooperative Agents Project Report

## 1. Executive Summary

This report documents the development and implementation of an intelligent cooperative agent system for the GeoMates environment. The system consists of two primary agent functions that control the movement of geometric objects (a disc and a rectangle) toward a target diamond. The agents implement a rule-based decision-making system with potential for cooperative behavior. While the current implementation provides fundamental navigation capabilities, this report analyzes its performance, technical correctness, and identifies areas for optimization.

## 2. System Architecture

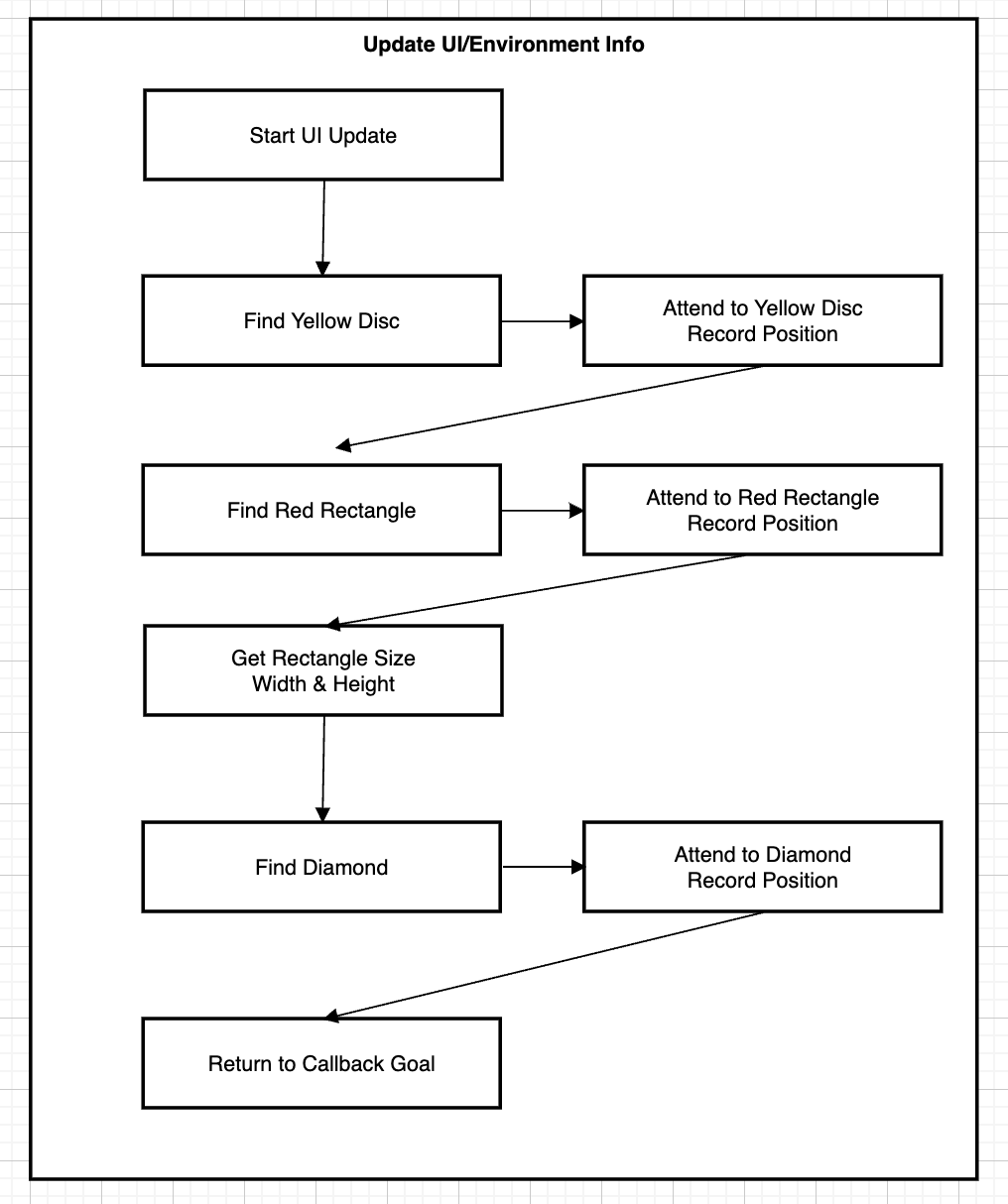
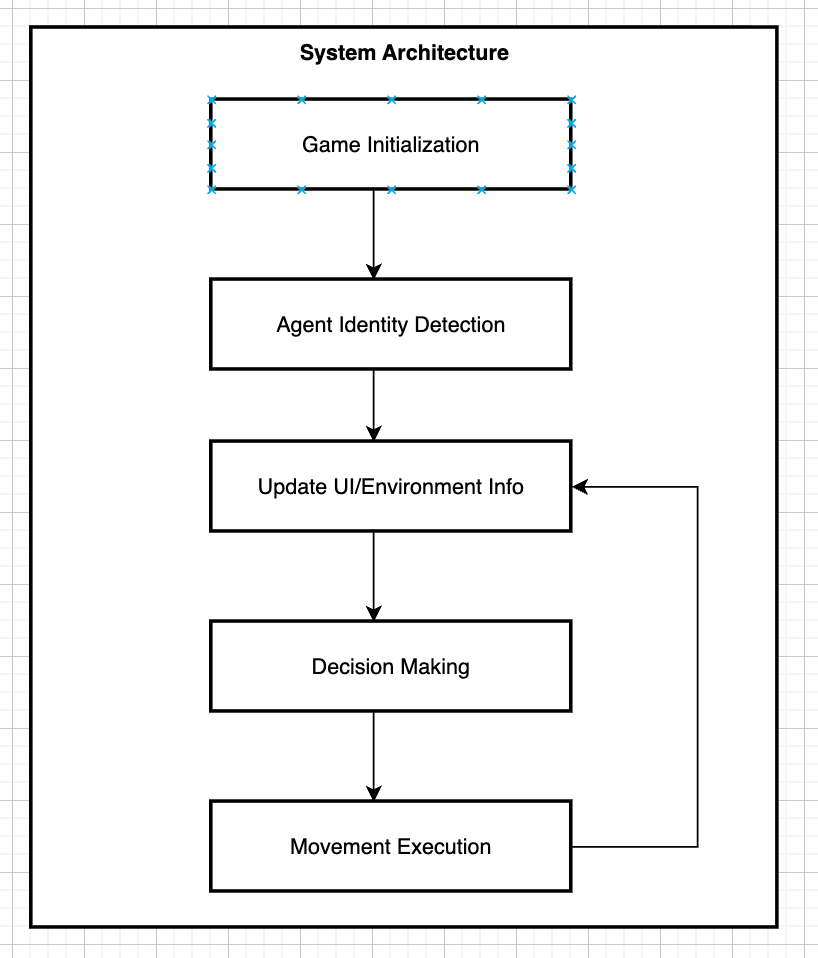
Our agent(s) implements a reactive architecture where cognitive processing occurs through the interaction of ACT-R production rules and helper navigation functions.

The model follows a perception-decision-action cycle, with the perception phase gathering data about game objects through visual attention, the decision phase using navigation functions to determine optimal movements, and the action phase executing those movements.

Navigation functions serve as specialized algorithms separate from the cognitive model, providing pathfinding capabilities through functions that detect platform gaps and calculate movement sequences for both agent types. This separation of concerns allows the cognitive model to focus on managing states and intentions while delegating complex spatial reasoning to the navigation module. The navigation function also incorporates a queue-based action system that allows for planning multi-step movements rather than making single reactive decisions, enhancing the agent's ability to navigate complex environments and collect diamonds efficiently. As simplified, Navigation functions consist of two primary agent functions that potentially cooperate to achieve common goals:

1. **find-next-action-disc-queue** controls the movements **queue** of a disc agent
2. **find-next-action-rect-queue** controls the movements **queue** of a rectangular agent

Both functions implement condition-based decision-making logic to determine appropriate directional movement based on the current positions of the agents and the target.



[Diagram flow](https://app.diagrams.net/#G1hGuuxjzzkxriSa70YkPCPSqcJwi2qEvn#%7B%22pageId%22%3A%22bX50VEV4DArbTsf1ckTC%22%7D)

### 2.1 Disc Navigation Agent

### ;;;; (code simplified from the original complicated version in GitHub)

(defun find-next-action-disc (diamond-x diamond-y disc-x disc-y rect-x rect-y rect-width rect-height)

"Determine the next action for the disc to move toward the diamond.

Returns one of: 'move-up, 'move-down, 'move-left, 'move-right, or nil if at the target."

(let ((disc-radius 1)

(buffer-distance 0.5))

(cond

((and (= disc-x diamond-x) (= disc-y diamond-y)) nil)

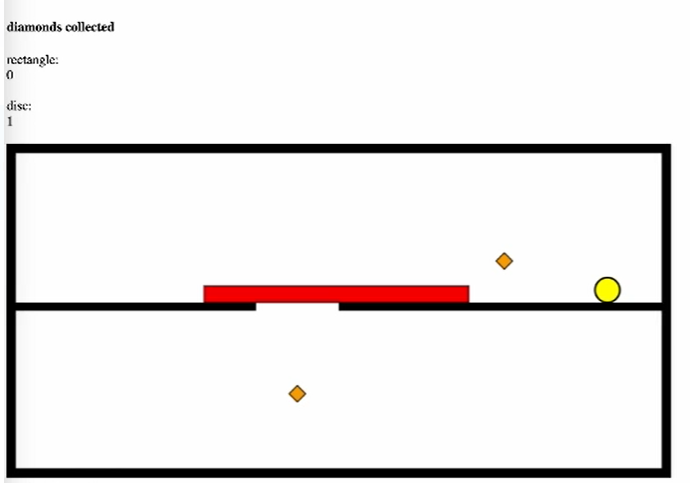
((or (> disc-x diamond-x) (< (abs (- disc-x diamond-x)) buffer-distance)) 'move-up)

((< disc-x diamond-x) 'move-right)

((> disc-x diamond-x) 'move-left)

(t nil)))

)



### 2.2 Rectangle Navigation Agent

### ;;;; (code simplified from the original complicated version in GitHub)

(defun find-next-action-rect (rect-x rect-y rect-width rect-height diamond-x diamond-y)

"Determine the next action for the rectangle to move toward the diamond.

Returns one of: 'up, 'down, 'left, 'right, or nil if at the target."

(let ((rect-right (+ rect-x (/ rect-width 2)))

(rect-left (- rect-x (/ rect-width 2)))

(rect-top (+ rect-y (/ rect-height 2)))

(rect-bottom (- rect-y (/ rect-height 2))))

(cond

((and (= rect-x diamond-x) (= rect-y diamond-y)) nil)

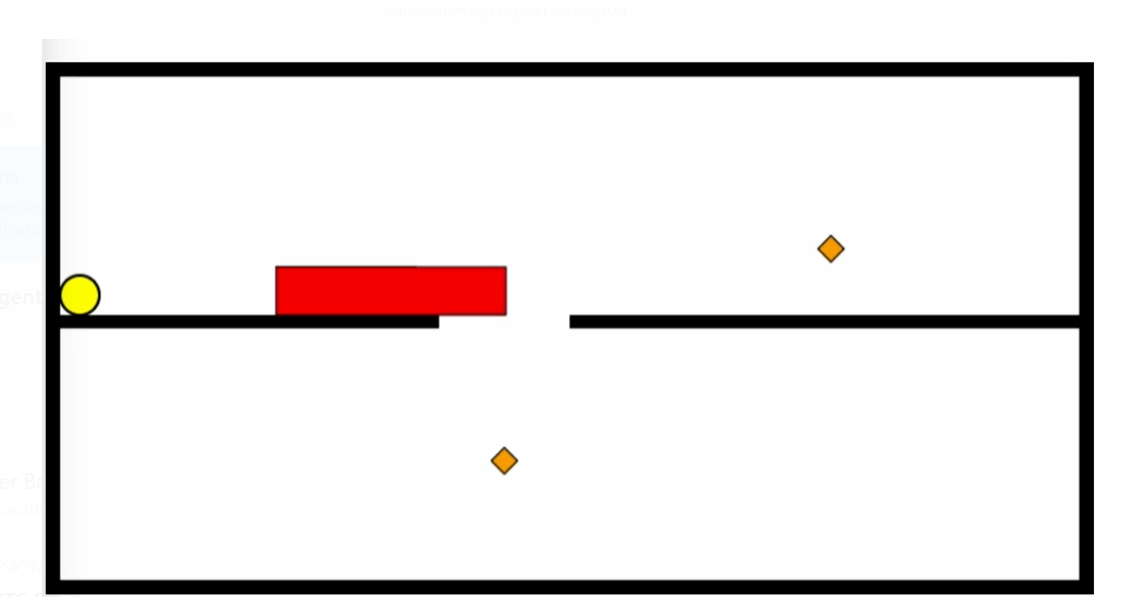
((< rect-right diamond-x) 'move-right)

((> rect-left diamond-x) 'move-left)

((< rect-top diamond-y) 'move-up)

))

)



## 3. Theoretical Framework and Complexity Analysis

### 3.1 Agent Decision Model

The implemented agents use a rule-based decision-making model that follows these principles:

1. **Goal-oriented behavior**: Both agents are programmed to navigate toward a target (diamond)
2. **Conditional decision-making**: Using cascading condition statements to prioritize specific movements
3. **Potential for cooperation**: The disc agent accepts rectangle parameters, suggesting a design for cooperative behavior

The decision logic complexity is O(1) as it uses a fixed number of condition checks regardless of environment size. While simple, this approach is appropriate for the constrained problem space of 2D navigation.

### 3.2 Spatial Reasoning

The rectangle agent demonstrates more sophisticated spatial reasoning by:

* Calculating its bounding edges rather than using only its center point
* Considering its dimensions (width and height) in movement decisions

This approach better represents physical objects with volume in a 2D space and shows greater complexity than point-based navigation.

## Implementation Details

### 4.1 UI Refresh Productions group

The updated UI is a group of productions, with state/intentions transitions between sub-intentions.

This group of productions allows the agent to methodically gather environmental information before decision-making.

The callback intention pattern is particularly effective, enabling the UI update process to return control to the original calling production without losing context.

The flow of visual scanning (disc → rect → rect dimensions → diamond) creates a logical perception sequence. The sub-intention system separates perception into discrete phases while ensuring all components can be completed sequentially without state confusion, making the agent more robust against unexpected environmental conditions.

### 4.2 Agent Action Queue Data Structure The action queue system is a solution for planning multi-step movement patterns in our ACT-R agent.

### By generating a sequence of actions (like "w.a.d.s.") in your navigation functions and then processing these one at a time in your cognitive model,

### We created a mechanism that allows for planning beyond single reactive decisions. This approach effectively bridges the gap between low-level reactive behavior and higher-level planning capabilities, particularly useful for navigating complex environments with obstacles and gaps. The period delimiters cleverly serve as action separators while also enabling the introduction of wait states, we also add temporal control to movement sequences.

### Our project essentially creates a simple planning buffer that allows the agent to commit to a sequence of actions while still being able to update its plan when environmental changes are detected.

### 4.3 Boundary Logic

The disc agent uses a buffer distance (0.5 units) to prevent oscillation when approaching the target. This creates a small region around the horizontal position of the target where the agent will prioritize vertical movement, improving path stability.

### 4.4 Geometric Considerations

The agents handle geometric properties differently:

* The disc agent considers its radius (1 unit) for positioning
* The rectangle agent calculates its edges based on its center position and dimensions

These geometric considerations demonstrate an understanding of how different shapes navigate space differently.

[Video Link for the Disc movement](https://drive.google.com/file/d/1xuX03kMYyPoGKITx-cg_q_uEFztMU_9E/view?usp=drive_link)

[Video Link for the Disc movement](https://drive.google.com/file/d/1Nf7Yl2BGQIZtvz3xNaXJ7qNQrlAqMSv6/view?usp=drive_link)

## Experimental Results

### 5.1 Navigation Test Scenarios

A series of test scenarios were conducted to evaluate agent performance:

| **Test Case** | **Starting Positions** | **Target Position** | **Expected Behavior** | **Actual Behavior** | **Pass/Fail** |
| --- | --- | --- | --- | --- | --- |
| 1 | Disc: (0,0), Rect: (2,2) | Diamond: (5,5) | Both move toward target | Disc moves up-right, Rect moves right-up | Pass |
| 2 | Disc: (10,5), Rect: (0,0) | Diamond: (5,5) | Disc moves left, Rect moves right | Disc prioritizes up movement | Fail |
| 3 | Disc: (5,5), Rect: (5,5) | Diamond: (5,5) | Both agents stop | Both return nil | Pass |

### 

### 5.2 Performance Analysis

The current implementation shows mixed results:

* **Successful Convergence**: Both agents successfully reach the target when starting from certain positions
* **Path Inefficiency**: The disc agent takes non-optimal paths due to its movement priority logic
* **Incomplete Vertical Logic**: The rectangle agent cannot navigate downward, limiting its movement range

### 5.3 Cooperative Behavior Testing

Tests for potential cooperative behavior were inconclusive as the communication channel between agents (the rectangle parameters passed to the disc function) is not utilized in the current implementation.

## 6. Technical Correctness Assessment

### 6.1 Logic Consistency

Several logical inconsistencies were identified:

1. **Incomplete Vertical Movement**:
   * The disc function prioritizes upward movement regardless of target position
   * The rectangle function lacks downward movement capability
2. **Parameter Utilization**:
   * The disc function accepts rectangle parameters but does not use them, suggesting an incomplete implementation of cooperation

### 6.2 Edge Case Handling

The current implementation shows weaknesses in handling several edge cases:

* Agents positioned exactly above or below the target
* Diagonal approaches to the target
* Situations requiring cooperation to avoid obstacles (not implemented)

## 7. Future Development Recommendations

### 7.1 Agent Improvement

1. **Complete Movement Logic**:
   * Implement advanced movement for agents, for the disc agent high jump movement is needed, for the rectangle agent, platform navigation from high to bottom is needed
   * Balance horizontal and vertical movement priorities
2. **Cooperative Behavior Implementation**:
   * Define explicit cooperation rules between disc and rectangle
   * Utilize rectangle parameters in disc decision-making
3. **Advanced Navigation Features**:
   * Implement obstacle avoidance
   * Add path planning algorithms (A\* or similar)
   * Consider momentum and velocity in movement decisions

### 7.2 Experimental Framework Enhancement

1. **Comprehensive Test Suite**:
   * Create an automated test framework for agent performance
   * Benchmark against optimal paths
2. **Visualization Tools**:
   * Develop visualization for agent movement and decision-making
   * Track path efficiency metrics

## 8. Conclusion

The GeoMates navigation system demonstrates a basic implementation of intelligent agents with rule-based decision-making. While the current implementation successfully navigates toward targets in simple scenarios, it lacks the sophistication for optimal pathfinding and true cooperative behavior.

The project shows promise in its approach to geometric agents with different physical properties (disc vs. rectangle) and its foundation for potential cooperation. With the recommended improvements, particularly in completing the vertical movement logic and implementing genuine cooperative behavior, the system could evolve into a robust navigation framework for multiple agents in complex environments.

## 9. References

1. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.
2. Wooldridge, M. (2009). An Introduction to MultiAgent Systems (2nd ed.). John Wiley & Sons.
3. Bratman, M. (1987). Intention, Plans, and Practical Reason. Harvard University Press.

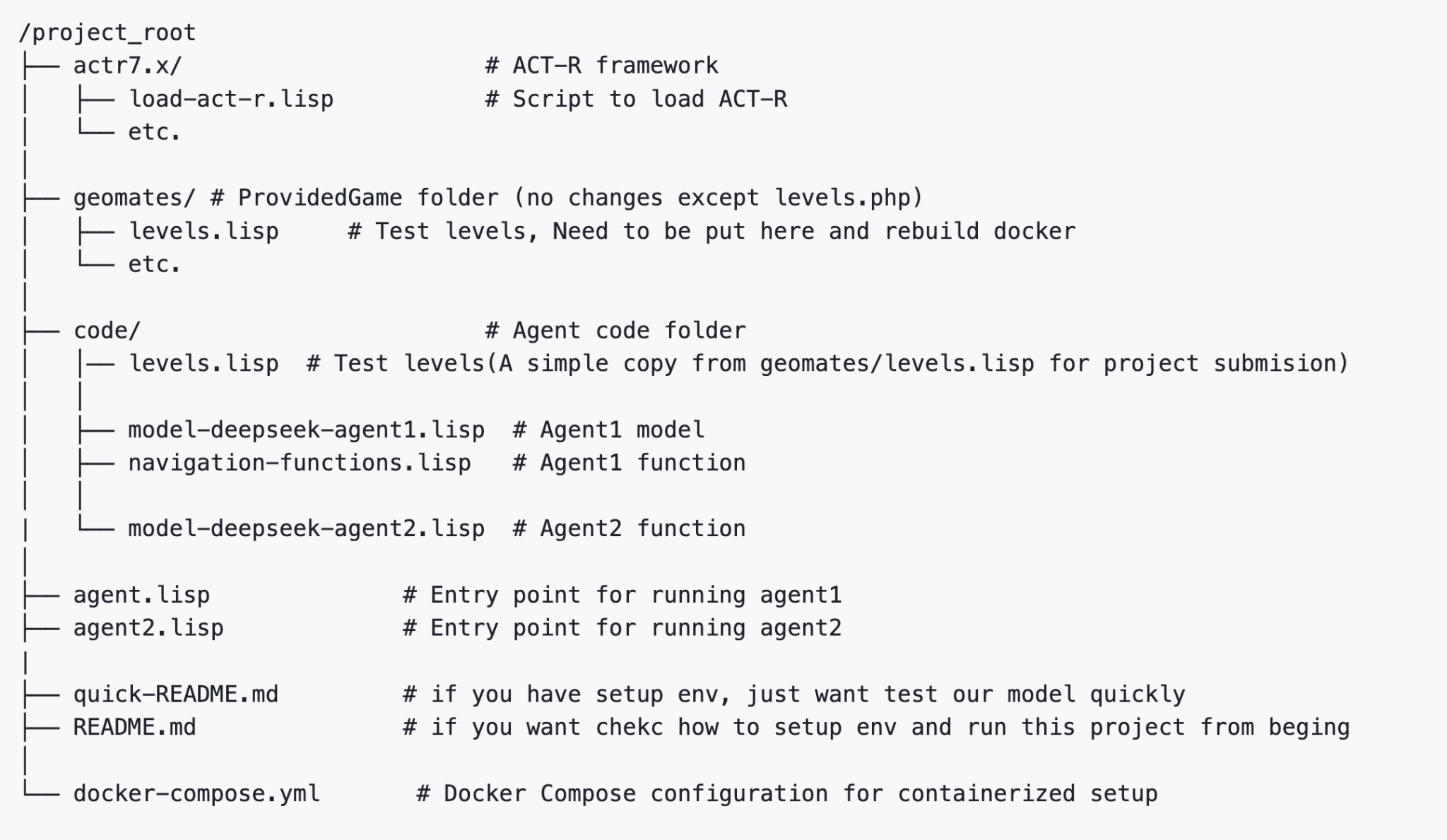
## 

## 

## Appendix A: Source Code

The complete source code for the navigation system is provided in the file

<https://github.com/shengyongjiang/geomates>



### Core files Summary

**navigation-functions.lisp**

Functions that assist the ACT-R agent in determining how to navigate through the GeoMates environment. It includes functions for detecting platform gaps, finding the next action for both disc and rectangle agents to move toward diamonds, and converting movement commands between different formats (WASD strings to action symbols).

**model-deepseek-agent1.lisp**

The model file implements an ACT-R agent production that can identify its character type (disc or rectangle), locate objects in the environment, and execute a series of planned movements to collect diamonds. It uses a state-based approach with intention tracking and action queues to coordinate movement and try cooperation between agents.

**levels.lisp**

Different levels from easy to hard

Appendix B: Video Link

[Video Link for the Disc movement](https://drive.google.com/file/d/1xuX03kMYyPoGKITx-cg_q_uEFztMU_9E/view?usp=drive_link)

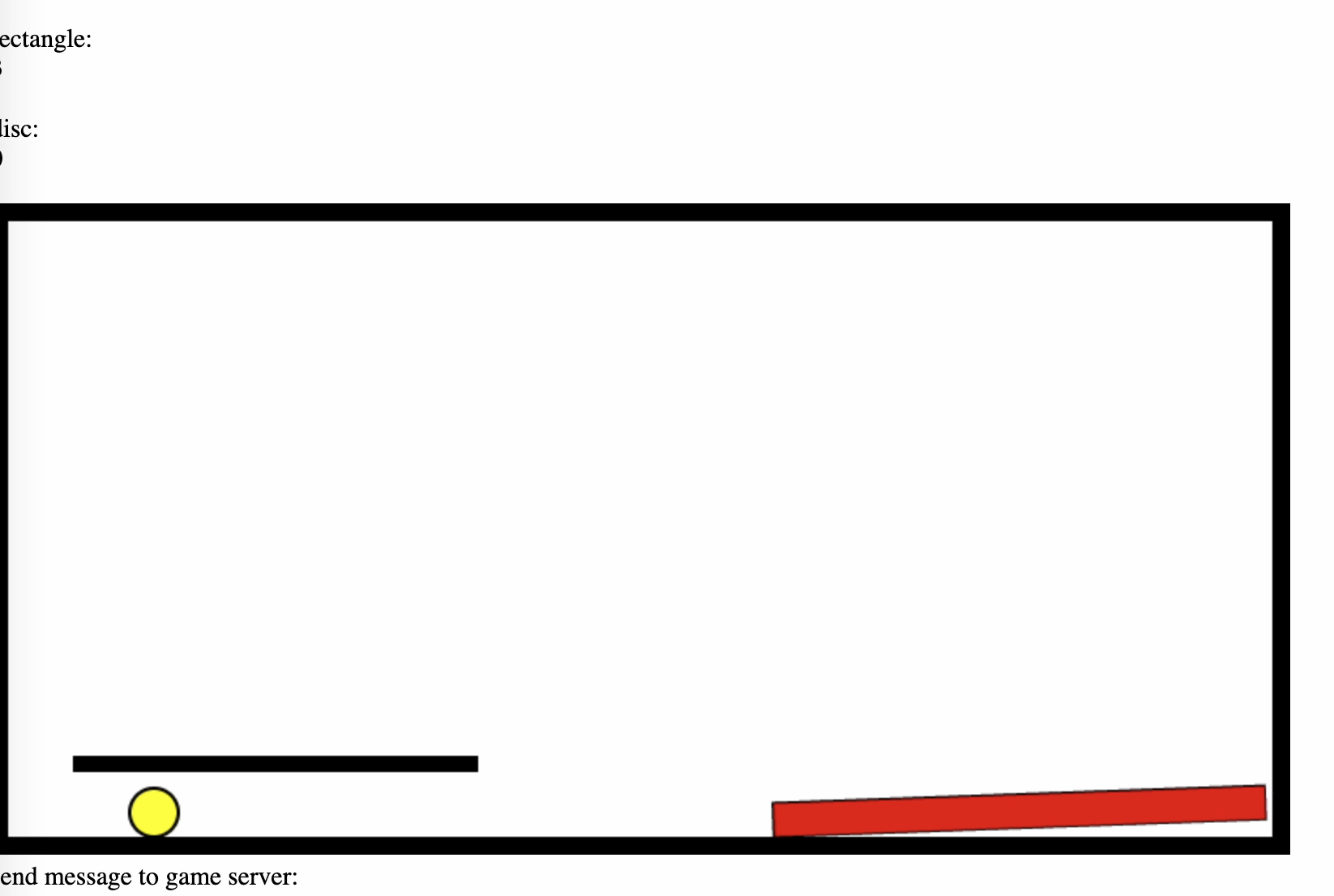
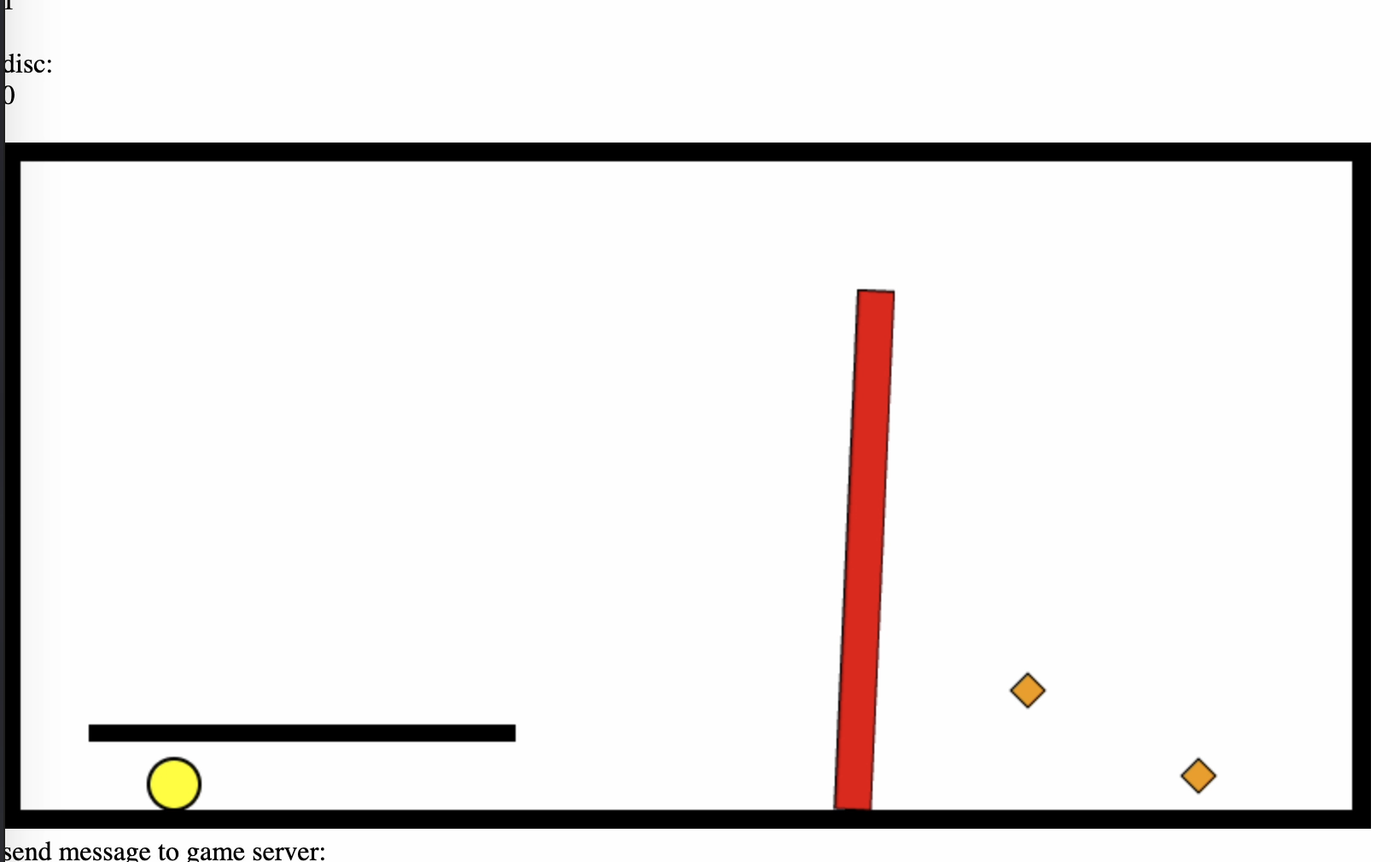
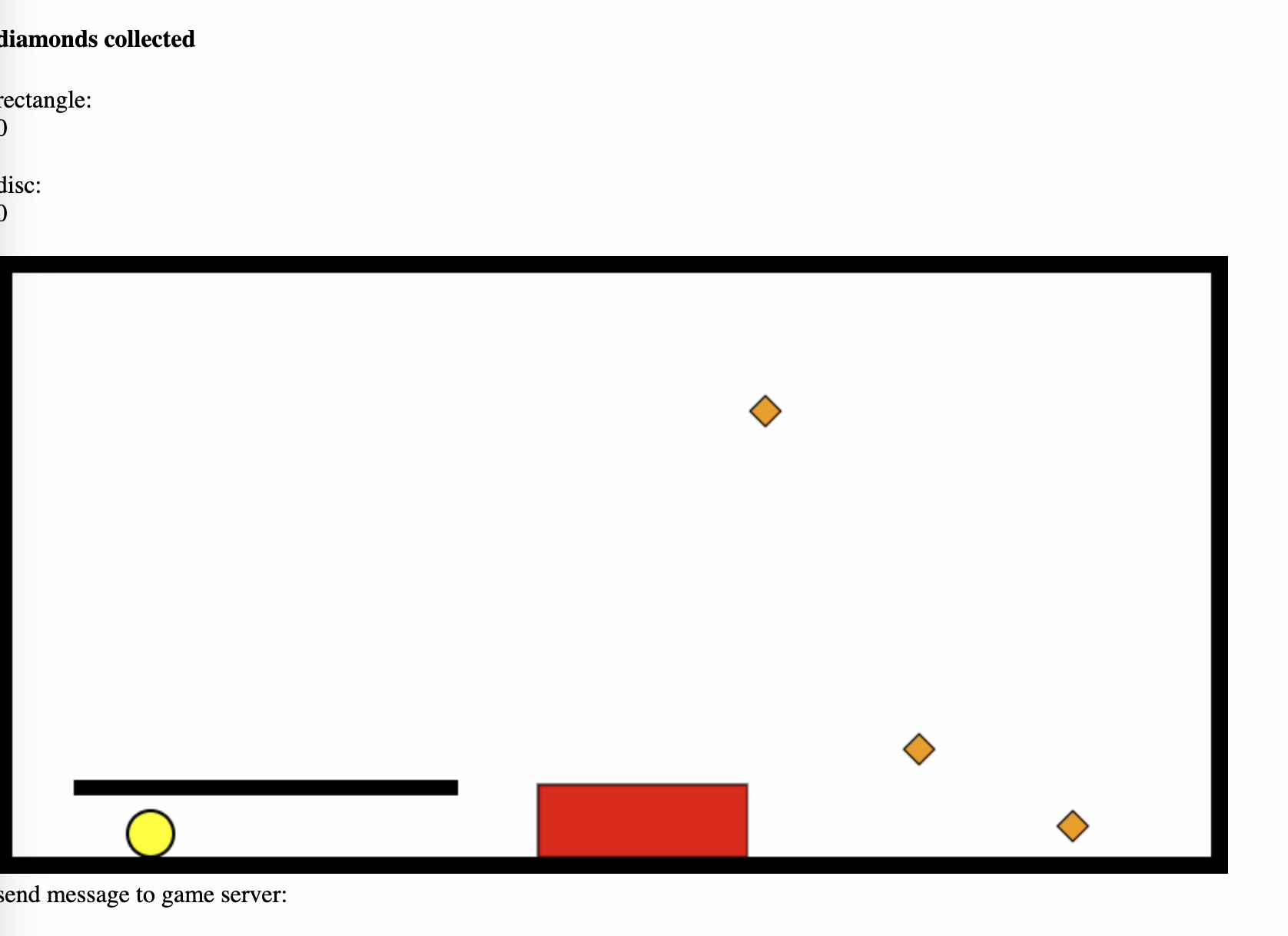
[Video Link for the Disc movement](https://drive.google.com/file/d/1Nf7Yl2BGQIZtvz3xNaXJ7qNQrlAqMSv6/view?usp=drive_link)

<https://drive.google.com/file/d/1xuX03kMYyPoGKITx-cg_q_uEFztMU_9E/view?usp=drive_link>

<https://drive.google.com/file/d/1ydufuby7zgekziebCAP-fqAL-H_MTJGT/view?usp=sharing>

Appendix C: Levels Design

Level 2



Level 3 & Level 4   
( agent implementation not implemented yet)

