Tactibots Project Proposal #4

1. Data Features Visualization Plan

1.1 Table Presentation Details

Feature for Table Display (A)	Statistical Values to Present (B)
Robot Position Tracking	Mean distance from optimal path, Maximum deviation from path, Standard deviation of position
Recovery Attempts	Total count, Average per stage, Maximum in single stage
Time to Completion	Minimum (best time), Average completion time, Standard deviation
Obstacle Collision Count	Total collision count, Average collisions per stage, Collision rate (collisions/minute)
Algorithm Card Combinations	Success rate percentage, Average recovery count per combination

1.2 Graph Visualization Plan

Graph Type	Features Used	Purpose of Visualization
Heatmap	Robot Position Tracking	Visualize movement patterns across the game map, highlighting areas where robots frequently travel or get stuck
Line Graph	Time to Completion (y-axis) vs. Attempt Number (x-axis)	Track player improvement over time with different algorithm combinations, showing learning curve and effectiveness of different strategies
Bar Chart	Recovery Attempts (grouped by algorithm combinations)	Compare recovery attempts across different algorithm combinations to identify the most effective approaches

2. Project Implementation Plan

2.1 Weekly Planning

26 March - 2 April

- Finalize UML class diagram and system architecture
- Implement Robot class with basic movement mechanics
- Implement Card class with parameter modification functionality
- Begin implementation of Algorithm base class

3 April - 9 April

- Complete Algorithm class and implement algorithm subclasses:
    - PathfindingAlgorithm with A* implementation
    - CollisionAvoidanceAlgorithm with Dynamic Window Approach
    - RecoveryAlgorithm with rule-based system
- Begin GameController implementation with game state management
- Implement basic user interface for card selection
- Start DataRecorder implementation for tracking basic metrics

10 April - 16 April

- Complete DataRecorder with CSV export functionality
- Implement collision detection system in Robot class
- Create first playable stage with static obstacles
- Implement event-driven mechanics for real-time responses
- Design basic visualization of collected data using Tkinter
- Begin implementation of dynamic obstacles

17 April - 23 April

- Complete all planned data visualizations:
    - Position heatmap visualization
    - Time-to-completion line graph
    - Recovery attempts bar chart
- Implement statistical analysis functions in DataRecorder
- Add multiple stages with varying difficulties
- Implement leaderboard functionality
- Complete movement efficiency calculations
- Conduct initial testing and gather sample data

24 April - 11 May

- Refine algorithm behaviors based on testing results

- Complete dynamic obstacle implementations
- Add UI polish and improve user experience
- Finalize all documentation and code comments
- Create comprehensive testing suite
- Prepare final presentation and demo
- Package project for submission

2.2 50% Completion Milestone (by 16 April)

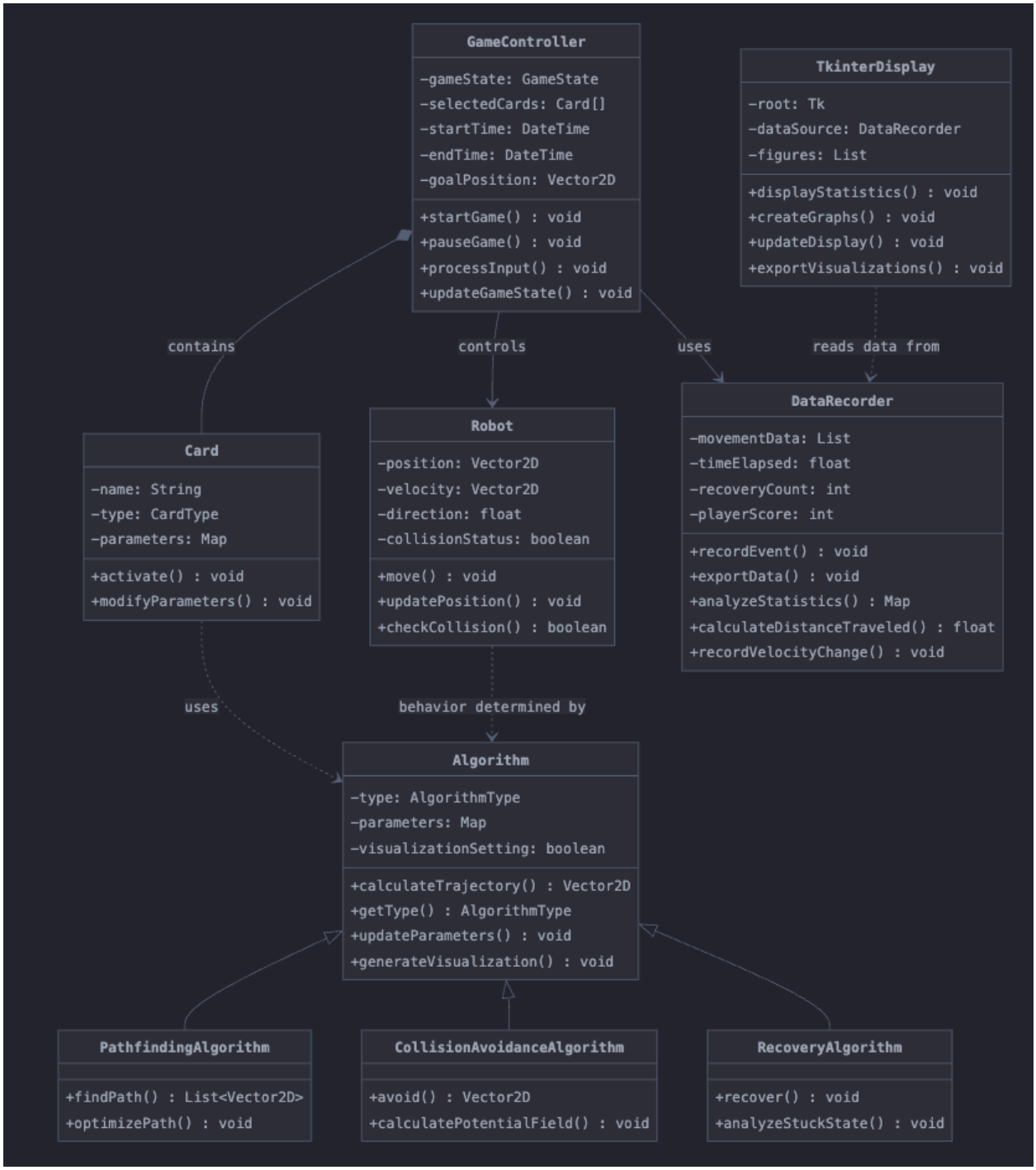
- Complete implementation of all core classes:
    - Robot class with movement and collision detection
    - Card class with parameters and modification capabilities
    - Algorithm base class and all three algorithm subclasses
    - GameController with basic game loop functionality
    - DataRecorder with data collection and CSV export
- Implement first playable stage with static obstacles
- Create basic user interface for card selection and game control
- Implement collision detection and recovery mechanics
- Begin implementation of data visualization components

2.3 75% Completion Milestone (by 23 April)

- Complete all three data visualizations (heatmap, line graph, bar chart)
- Implement statistical analysis functions for all tracked metrics
- Add multiple game stages with varying difficulty levels
- Implement dynamic obstacles in game environments
- Complete leaderboard functionality
- Implement movement efficiency calculations
- Conduct initial testing with sample data collection
- Begin comprehensive documentation

2.4 Final 25% Completion (by 11 May)

- Refine algorithm behaviors for improved gameplay experience
- Complete dynamic obstacle implementations with varied behaviors
- Polish user interface and improve overall user experience
- Finalize all documentation including code comments and user guide
- Create comprehensive testing suite to validate all features
- Prepare final presentation and demonstration
- Package project for final submission


GameController
-gameState: GameState
-selectedCards: Card[]
-startTime: DateTime
-endTime: DateTime
-goalPosition: Vector2D
+startGame() : void
+pauseGame() : void
+processInput() : void
+updateGameState() : void
TkinterDisplay
-root: Tk
-dataSource: DataRecorder
-figures: List
+displayStatistics() : void
+createGraphs() : void
+updateDisplay() : void
+exportVisualizations() : void
contains
controls
uses
reads data from
Card
-name: String
-type: CardType
-parameters: Map
+activate() : void
+modifyParameters() : void
Robot
-position: Vector2D
-velocity: Vector2D
-direction: float
-collisionStatus: boolean
+move() : void
+updatePosition() : void
+checkCollision() : boolean
DataRecorder
-movementData: List
-timeElapsed: float
-recoveryCount: int
-playerScore: int
+recordEvent() : void
+exportData() : void
+analyzeStatistics() : Map
+calculateDistanceTraveled() : float
+recordVelocityChange() : void
uses
behavior determined by
Algorithm
-type: AlgorithmType
-parameters: Map
-visualizationSetting: boolean
+calculateTrajectory() : Vector2D
+getType() : AlgorithmType
+updateParameters() : void
+generateVisualization() : void
PathfindingAlgorithm
+findPath() : List<Vector2D>
+optimizePath() : void
CollisionAvoidanceAlgorithm
+avoid() : Vector2D
+calculatePotentialField() : void
RecoveryAlgorithm
+recover() : void
+analyzeStuckState() : void