Game AI System Design Document

Casper Fahlman

January 15, 2026

Contents

1 Overview 2

2 High-Level Architecture 2

3 Game Loop 2

3.1 GameLoop Class 2

3.2 Key Responsibilities 2

3.3 Important Methods 3

4 AI Agent System 3

4.1 GameAI 3

4.2 AI States 3

4.3 Movement and Navigation 3

4.4 AI-Brain Connection 3

5 AI Brain 3

5.1 AIBrain Overview 3

5.2 Key Responsibilities 4

5.3 Finite State Machine 4

5.4 World Knowledge 4

6 Task System 4

6.1 Task Representation 4

6.2 Task Allocation 4

6.3 Task Types 5

7 AI Managers 5

7.1 ResourceManager 5

7.2 BuildManager 5

7.3 ManufacturingManager 5

7.4 MilitaryManager 5

8 World Representation 5

8.1 Grid 5

8.2 PathNode 5

8.3 Pathfinding Support 6

9 Design Patterns Used 6

10 Conclusion 6

1 Overview

Assignment from W0045E Game AI Programming.
This document describes the design and architecture of a grid-based game simulation featuring autonomous AI agents, resource management, construction, and military systems. The system is implemented in C++ and uses SDL for rendering and input handling.

The core components include:

- A centralized game loop
- AI-controlled agents with finite state machines
- A cognitive AI brain with task planning
- A grid-based world representation
- Modular AI managers for resources, building, manufacturing, and military logic

2 High-Level Architecture

The system follows a modular, object-oriented design with clear responsibility separation.

- **GameLoop** orchestrates updates, rendering, and input.
- **GameAI** represents individual autonomous agents.
- **AIBrain** handles reasoning, decision-making, and task allocation.
- **Grid** and **PathNode** model the world and navigation space.
- **Managers** encapsulate domain-specific logic (resources, building, military).

3 Game Loop

3.1 GameLoop Class

The `GameLoop` class is implemented as a singleton and is responsible for:

- Initializing the game world
- Running the fixed-timestep update loop
- Managing input, rendering, and debug tools
- Owning global systems such as the grid and renderer

3.2 Key Responsibilities

- Frame timing and FPS regulation
- Player and AI lifecycle management
- Debug visualization
- Scheduling AI deletion

3.3 Important Methods

- `RunGameLoop()`
- `UpdateGameLoop()`
- `InitializeGame()`

4 AI Agent System

4.1 GameAI

`GameAI` represents an autonomous agent in the world and extends the `Movable` base class.

4.2 AI States

Agents use a finite state machine with the following states:

- Idle
- Seek
- Flee
- Arrive
- Wander
- Evade
- Pursue
- Follow Path

4.3 Movement and Navigation

- Agents navigate using grid-based pathfinding
- Targets may be static nodes or moving entities
- Path validation respects fog-of-war constraints

4.4 AI-Brain Connection

Each `GameAI` owns an `AIBrain` instance responsible for high-level decisions.

5 AI Brain

5.1 AIBrain Overview

The `AIBrain` class acts as a cognitive layer for each AI agent. It performs reasoning, planning, and task execution using a modular manager system.

5.2 Key Responsibilities

- Maintaining beliefs about the world (known nodes)
- Managing desires and priorities
- Allocating and executing tasks
- Running a task-based finite state machine

5.3 Finite State Machine

The AI brain FSM:

1. Retrieves the highest-priority task
2. Executes behavior based on task type
3. Generates subtasks if prerequisites are missing
4. Returns the agent to idle when no tasks remain

5.4 World Knowledge

`KnownNode` structures store partial knowledge about the grid:

- Discovery state
- Walkability belief
- Last seen time
- Associated resource

6 Task System

6.1 Task Representation

Tasks are defined using the `Task` struct and include:

- Task type
- Required resources
- Priority
- Time or quantity constraints

6.2 Task Allocation

The `TaskAllocator`:

- Assigns unique task IDs
- Orders tasks by priority
- Ensures only one active task per agent

6.3 Task Types

- Discover
- Gather
- Build
- Manufacture
- Train Soldiers

7 AI Managers

7.1 ResourceManager

Handles inventory tracking, resource consumption, and validation.

7.2 BuildManager

- Manages building templates
- Queues and places buildings
- Tracks constructed structures

7.3 ManufacturingManager

Converts raw resources into processed goods such as steel and swords.

7.4 MilitaryManager

- Manages soldier templates
- Handles training queues
- Tracks active military units

8 World Representation

8.1 Grid

The `Grid` class represents the navigable game world.

- 2D grid of `PathNodes`
- Spatial queries for entities and nodes
- Line-of-sight and clearance calculations

8.2 PathNode

Each `PathNode` represents a cell in the grid.

- Position and neighbors
- Node type (wall, resource, empty)
- Clearance and obstacle detection

8.3 Pathfinding Support

The `NodeRecord` structure supports A* pathfinding with cost tracking.

9 Design Patterns Used

- Singleton (`GameLoop`)
- Strategy (Behaviours)
- Finite State Machine (AI and Brain)
- Component-Based AI Managers
- Data-Oriented Task System

10 Conclusion

This architecture provides a scalable and extensible framework for AI-driven gameplay. The modular manager-based AI brain allows for complex emergent behavior while maintaining clarity and separation of concerns.

Future improvements could include:

- Multi-agent coordination
- Advanced planning
- Persistent world memory