Complete Mind Map System - All Concepts and Ideas

Overview

This document captures all the concepts, ideas, and technical details discussed for creating a mind map-based reasoning system for interactive storytelling and AI character thinking.

Core Architecture Concepts

1. Mind Map as Reasoning Engine

Your Idea: Use a mind map structure where thoughts and reasoning processes flow through nodes, with each node representing concepts, situations, or decision points.

Implementation:

- Nodes represent mental concepts (scenes, objects, reasoning states)
- Connections between nodes represent logical relationships
- Character reasoning happens by traversing this network

2. Wrapped Objects as Traveling Thoughts

Your Idea: Create wrapped objects that move through the mind map like packages flowing through a shipping system. These objects carry data, context, and executable functions.

Key Features:

```javascript
WrappedObject {
  data: // The actual thought/question/reasoning
  context: // Closure scope for functions
  path_history: // Nodes visited
  callback: // Function that travels with the object
  plot_context: // Story position and character state
}
```

Your Vision: Objects flow like water through a stream, carrying reasoning from node to node, getting modified and copied as they go.

3. Tree Flow System with Callback Planting

Your Idea: Push wrapped objects at the root of a tree structure. Objects flow down branches based on matching conditions, copying themselves to multiple paths, and planting callbacks at leaf nodes.

The Process:

1. Object starts at root
2. Flows down branches where conditions match
3. Copies itself to multiple matching children
4. Plants callbacks at endpoints
5. Callbacks wait for specific triggers to execute

Your Analogy: Like tracing paper - you can overlay different tree states and compare patterns.

4. Separation of Concerns - Three-Part Architecture

Your Requirement: Separate the system into distinct parts:

1. **Frontend Interface:** Text input, WebSocket client, display output
2. **Node.js Server:** Message routing, WebSocket management, no game logic
3. **Game Engine:** Tree system + mind map + event loop + state management

Communication Flow:

```
Frontend ↔ WebSocket ↔ Server ↔ Game Engine
```

5. Plot Alignment with Mind Map

Your Insight: The mind map structure must align with the story's plot progression.

Implementation:

- Mind map nodes have `plot_position` properties
- Character thoughts constrained by current story position
- Wrapped objects carry plot context
- Callbacks only trigger when narratively appropriate

Advanced Reasoning Concepts

6. Circular Dependency Detection and Question Generation

Your Core Insight: When the system detects circular reasoning (A depends on B depends on A), instead of trying to solve it algorithmically, it should recognize the limitation and generate questions.

The Dishwasher Example:

- Can't unload dishwasher until full
- Can't fill dishwasher until empty
- But someone else controls filling permission
- System detects loop → generates question: "What external factor breaks this cycle?"

Your Philosophy: Turn logical limitations into curiosity. The character becomes genuinely curious because the system discovers the shape of its own ignorance.

7. Back-Propagation with State Callbacks

Your Concept: When blocked, the system should work backward through previous states, checking callback lists at each step to find where dependencies could be broken.

Callback Lists as Comparison Functions:

- Each state property has callbacks that compare previous vs current state
- Callbacks detect patterns, dependencies, deadlocks
- Back-propagation re-analyzes historical states with current knowledge
- Looks for points where external factors could change the situation

8. Context Comparison - "Tracing Paper" Effect

Your Metaphor: Save mind map states as snapshots, then overlay them like tracing paper to find matching patterns.

Implementation:

- Save node states at different story moments
- Compare snapshots to find similar situations
- Apply successful reasoning patterns from one context to another
- Use vector-like comparison for similarity scoring

9. Message Passing Between Components

Your Design: Callback wrappers that send messages between tree nodes and character knowledge system.

Example: Book reading scenario:

- Tree node detects book interaction
- Sends "KNOWLEDGE_GAINED" message
- Character catch callback receives it
- Updates character knowledge state

Technical Implementation Details

10. Node Structure

Your Specification:

javascript

```javascript
Node {
  id: // unique identifier
  concept: // what this represents
  connections: // paths to other nodes
  callbacks: // functions that can be triggered
  state: // current properties/values
  plot_position: // where in story this belongs
  global_state_refs: // references to shared state
  branch_state: // concept application history
}
```

11. Global Data State System

Your Requirement: Centralized asset loading and state management system separate from core algorithms.

Features:

- Load story scenes, tree structures, mind map definitions
- Manage character knowledge and game state
- Decouple story content from reasoning engine

12. Event Loop and Real-Time Processing

Your Vision: Continuous event loop that:

- Processes wrapped object movement
- Executes callbacks when triggered
- Handles message passing
- Updates character state
- Provides diagnostic information

13. Versatile Callback System

Your Requirement: Callbacks should be general-purpose functions that can:

- Modify the tree structure itself
- Copy other callbacks between nodes
- Pass things around the system
- Facilitate any type of action
- Work with closure scopes and message passing

Character and Story Integration

14. Character as Reasoning Entity

Your Vision: The character isn't just responding to input - they have ongoing internal mental processes driven by curiosity and the need to understand.

Implementation:

- Character knowledge grows through message system
- Reasoning happens through wrapped object flow
- Questions generate automatically from logical limitations
- Character thoughts appear as system output

15. Story Actions and World Manipulation

Your Concept: The reasoning system needs to interface with story world manipulation.

Features:

- Move characters between locations
- Add/remove objects from scenes
- Trigger dialogue between characters
- Modify world state based on reasoning outcomes

16. Decision Trees Within Event Loop

Your Question: How to design decision trees inside an event loop for text-based interaction?

Solution:

- Tree structure defines possible decisions
- Event loop processes player commands
- Wrapped objects flow through decision branches
- Callbacks execute appropriate story responses

Advanced System Features

17. Diagnostic and Development Tools

Your Need: Real-time visibility into system operation.

Features:

- Live view of tree state and planted callbacks
- Active wrapped object tracking
- Mind map node status
- Character knowledge and state
- Back-propagation analysis results

18. Recursive Reference Handling

Your Concern: Prevent infinite loops when objects reference themselves.

Solution: Reference counters and loop detection in wrapped object path history.

19. Time-Based Callbacks

Your Challenge: How to handle situations where characters need to wait for future events?

Solution:

- Resolution lists that work backward from goals
- Movement signals in processes
- Callback planting for future conditions
- Back-propagation to identify waiting conditions

20. Ambiguous Text Processing

Your Goal: Handle ambiguity in text input and maintain context consistency.

Approach:

- Save example contexts with scaffolding processes
- Stack comparisons of mind maps
- Substitute variables in templates until contexts are analogous
- Use pattern matching for disambiguation

Core Algorithms Developed

21. Tree Flow Algorithm

- Wrapped object creation and propagation
- Recursive tree traversal with object copying
- Condition matching and callback planting
- Branch selection based on data matching

22. Message Passing Algorithm

- Subscribe/broadcast messaging system
- Target-specific delivery
- Message queue processing
- Component communication backbone

23. Circular Dependency Algorithm

- Path analysis for detecting loops
- Loop classification (simple/complex/deep)
- Question generation from logical limitations
- Loop-breaking strategy suggestions

24. Context Comparison Algorithm

- Mind map state snapshots
- Node similarity calculation with fuzzy matching
- Pattern overlay and matching
- Apply patterns from one context to another

25. Back-Propagation Algorithm

- Move backward through state history
- Check callback lists at each historical state
- Find dependency break points
- Generate resolution strategies

Key Insights and Philosophy

26. Intelligence as Process Discovery

Your Core Insight: True artificial intelligence isn't about having the right answers, but about recognizing when you don't know something and actively seeking to fill that gap.

27. Curiosity Through Limitation Recognition

Your Philosophy: Genuine curiosity emerges when a system discovers the boundaries of its own logic and turns those boundaries into questions.

28. Reasoning as Physical Flow

Your Metaphor: Thoughts should behave like physical objects flowing through a landscape, getting shaped and modified by the terrain they pass through.

29. Story Structure as Mental Architecture

Your Vision: The story's plot structure should literally be the architecture through which the character's thoughts flow, ensuring narrative consistency.

30. Emergent Behavior from Simple Rules

Your Approach: Complex reasoning behavior should emerge from simple propagation rules rather than being explicitly programmed.

Implementation Challenges Addressed

31. State Consistency

- How to maintain coherent character knowledge across reasoning sessions
- Preventing contradictory conclusions from different reasoning paths
- Ensuring story logic remains consistent

32. Performance and Scalability

- Managing large numbers of wrapped objects efficiently
- Preventing callback explosion
- Optimizing tree traversal algorithms

33. Debugging and Development

- Making the reasoning process visible and debuggable
- Providing tools to trace wrapped object paths
- Understanding why certain decisions were made

34. Integration Complexity

- Separating concerns while maintaining system coherence
- Clean interfaces between components
- Modular design that allows independent development

Future Extensions and Possibilities

35. Multi-Character Reasoning

- Each character has their own mind map
- Characters can share knowledge through interaction
- Reasoning processes can influence each other

36. Learning and Adaptation

- Mind map structure evolves based on experience
- Successful reasoning patterns get reinforced
- Failed approaches get modified or abandoned

37. Emotional and Personality Integration

- Character emotions influence reasoning flow
- Personality traits modify callback behavior
- Emotional state affects question generation

38. Complex Story Branching

- Multiple simultaneous plot threads
- Character reasoning affects story direction
- Dynamic story generation based on character thoughts

Technical Specifications

39. File Organization

```
project/
├── frontend/index.html        # Text interface
├── server/server.js           # WebSocket server
├── game/game-engine.js        # Main controller
├── game/tree-system.js        # Tree flow system
├── game/mindmap-system.js     # Mind map processing
├── shared/message-types.js    # Communication protocol
└── shared/global-data.js      # Asset and state management
```

40. Message Protocol

javascript

```javascript
// Frontend → Server → Game
PLAYER_COMMAND: { command: "examine door" }
DIAGNOSTIC_REQUEST: { show: "tree_state" }

// Game → Server → Frontend
STORY_OUTPUT: { text: "...", category: "narrative" }
CHARACTER_THOUGHT: { text: "...", category: "reasoning" }
DIAGNOSTIC_DATA: { tree_state: {...}, active_objects: [...] }
```

Key Quotes and Concepts

Your Vision Statements:

1. **"I wanted the idea of overlaying the maps and looking for similarities like tracing paper"**
2. **"When faced with uncertainty, couldn't the character ask another character?"**
3. **"The wrapped object has to maintain a context for closures, and it has to be able to have a function callback that can be put anywhere and run in that context"**
4. **"I want clearly defined modules that have purposes"**
5. **"The tree structure should be separate from the wrapped object. The wrapped object needs to be a container that moves through the nodes. Like think that its flowing on a stream through the node/branch highway"**
6. **"Time based callbacks... if you have to wait for something, this is a problem i have thought about for a long time"**
7. **"Back-propagation should identify and create a question maybe? My thoughts here aren't solving the problem but having a question prepared for the problem"**
8. **"This is the piece of magic that can happen. When conflicting interests collide I want diagnostic information"**
9. **"The mind map has to align with the plot line. The plot line can be expressed as an outline"**
10. **"I am more interested in how to get the character to think"**

Complete System Summary

This mind map reasoning system represents a novel approach to AI character thinking that:

- **Simulates genuine reasoning** through physical-like object flow
- **Generates authentic curiosity** by recognizing logical limitations
- **Maintains narrative consistency** through plot-aligned architecture
- **Handles complex dependencies** through back-propagation and question generation
- **Provides real-time insight** into character thought processes
- **Scales from simple interactions** to complex reasoning scenarios

The system achieves what traditional AI often misses: **the process of thinking itself**, not just the results of thinking. It creates characters that genuinely wonder, question, and seek understanding because the architecture naturally produces these behaviors from the interaction of simple components.

This is fundamentally a **reasoning engine** that happens to be applied to interactive storytelling, but the core algorithms could be used for any domain requiring dynamic, context-aware decision making with genuine uncertainty handling.