Land Memory System Design

Overview

The Land Memory System is a specialized implementation of the Eliza Memory System designed to handle futuristic real estate data. It combines vector-based semantic search capabilities with structured metadata to enable natural language queries for property information.

Core Components

1. Data Structure

Property Metadata

- **Categorical Data**
  - Plot Sizes (Nano to Giga)
  - Zoning Types (Residential to Legendary)
  - Building Types (Lowrise to Megatall)
  - Distance Categories (Close/Medium/Far)

Memory Structure

- Extends base Memory interface
- Combines natural language description with structured metadata
- Includes spatial and numerical data (coordinates, distances, dimensions)

2. Type System

```
LandPlotMemory extends Memory {
    content: {
        text: string;        // Natural language description
        metadata: LandPlotMetadata;  // Structured data
    }
}
```

3. Key Features

1. **Semantic Search**
   - Vector embeddings for natural language queries
   - OpenAI's text-embedding-3-small model
   - Configurable similarity thresholds
2. **Dual Storage**
   - Text descriptions for semantic search
   - Structured metadata for precise filtering

   - Combined query capabilities

3. **Distance Management**
   - Ocean and bay proximity tracking
   - Automatic distance categorization
   - Spatial coordinate system

4. **Property Classification**
   - Rarity ranking system (1-3000+)
   - Building type classification
   - Plot size categorization

Query System

Query Types

1. **Natural Language Queries**

```
"find me a luxury apartment with ocean views in North Star"
```

2. **Metadata Filters**

```
{
    neighborhood: ["North Star"],
    zoningTypes: ["Residential"],
    maxDistance: { ocean: 300 }
}
```

3. **Combined Queries**
   - Text similarity + metadata filtering
   - Ranking-based sorting
   - Distance-based filtering

Implementation Details

1. Memory Management

- Uses PostgreSQL for persistent storage
- Vector similarity search capabilities
- Caching system for embeddings

2. Helper Functions

```
generateDescription(): string
createLandPlotMemory(): LandPlotMemory
categorizeDistance(): DistanceCategory
categorizeRarity(): string
```

3. Integration Points

- Eliza Memory System
- OpenAI Embeddings API
- PostgreSQL Database
- Agent Runtime System

Usage Patterns

1. Property Creation

```
const plotMemory = createLandPlotMemory(
    id,
    metadata,
    agentId,
    roomId
);
await memoryManager.createMemory(plotMemory);
```

2. Property Search

```
const results = await memoryManager.searchMemoriesByEmbedding(
    queryEmbedding,
    { match_threshold: 0.1 }
);
```

3. Property Updates

- Atomic metadata updates
- Description regeneration
- Embedding recalculation

Performance Considerations

1. Optimization Strategies

- Embedding caching
- Batch processing for bulk operations
- Indexed metadata fields

2. Scaling Considerations

- Horizontal scaling of database
- Embedding model distribution
- Query optimization

Future Enhancements

1. Multi-modal search capabilities
2. Real-time property updates
3. Advanced spatial queries
4. Market dynamics integration
5. Historical data tracking

Security Considerations

1. Access control per property
2. Metadata validation
3. Query rate limiting
4. Secure embedding storage

Dependencies

- @ai16z/eliza
- @ai16z/adapter-postgres
- OpenAI API
- PostgreSQL