Aqua Nova Mission System Guide

Overview

The mission system is a data-driven engine that creates guided experiences for players. Instead of hardcoding story events, missions are defined in JSON files and processed by the Mission Manager, making it easy to create, modify, and debug content.

Architecture Components

Core Files

- `/game/systems/missionManager.js` - The brain of the system
- `/utils/interactiveElements.js` - Spawns clickable objects in scenes
- `/data/missions/index.json` - Lists all available missions
- `/data/missions/*.json` - Individual mission definitions

Integration Points

- **Game State** - Tracks progress and properties
- **Event System** - Communicates between systems
- **PDA Tasks** - Shows current objectives to player
- **Interactive Elements** - Creates discoverable objects
- **Communicator System** - Handles crew interactions

Data Flow

```
Mission JSON File
    ↓
Mission Manager loads and processes
    ↓
Checks prerequisites and auto-starts
    ↓
Creates objectives in game state
    ↓
Listens for trigger events
    ↓
When triggered: executes actions
    ↓
Updates game state and UI
```

↓
Checks for mission completion
↓
Executes completion actions
↓
Starts follow-up missions

Mission Lifecycle

1. **Loading Phase**

- Mission Manager reads `/data/missions/index.json`
- Loads each mission file listed
- Validates mission structure
- Stores in memory for quick access

2. **Auto-Start Check**

- Checks all missions with `"autoStart": true`
- Evaluates prerequisites for each
- Starts missions that meet all conditions

3. **Active Phase**

- Adds objectives to game state as "active"
- Sets up event listeners for triggers
- Monitors game state changes
- Responds to player actions

4. **Progression Phase**

- Player performs actions (pickup item, enter location, etc.)
- Events are fired with relevant data
- Mission Manager checks if events match objective triggers
- When matched, executes completion actions

5. **Completion Phase**

- All required objectives completed
- Mission marked as complete

- Completion actions executed
- Follow-up missions may start

Event System

The mission system communicates through custom DOM events:

Core Events

```javascript
// Location changes
document.dispatchEvent(new CustomEvent('location-enter', {
    detail: { location: 'quarters' }
}));

// Item interactions
document.dispatchEvent(new CustomEvent('item-pickup', {
    detail: { itemId: 'communicator' }
}));

// Dialogue completion
document.dispatchEvent(new CustomEvent('dialogue-complete', {
    detail: {
        characterId: 'executiveOfficer',
        dialogueId: 'first_contact'
    }
}));

// Interactive element spawning
document.dispatchEvent(new CustomEvent('spawn-interactive', {
    detail: { elementId: 'communicator_pickup', location: 'quarters' }
}));
```

Where Events Are Fired

- **quarters.js** - `location-enter` when page loads
- **communicatorOverlay.js** - `dialogue-complete` after conversations
- **interactiveElements.js** - `item-pickup` when clicking objects
- **missionManager.js** - `spawn-interactive` from actions

Where Events Are Handled

- **missionManager.js** - All mission-related events
- **interactiveElements.js** - `spawn-interactive` events
- **gameState.js** - Property change monitoring

Mission JSON Structure

Required Fields

```json
{
  "id": "unique_mission_identifier",
  "name": "Human-readable mission name",
  "description": "What this mission is about",
  "objectives": [...]
}
```

Full Template

```json
```

```
{
  "id": "mission_template",
  "name": "Template Mission",
  "description": "Description shown to player",
  "category": "tutorial|main|side",
  "priority": "low|medium|high",
  "autoStart": true,

  "prerequisites": [
    {
      "type": "property_equals",
      "property": "path.to.property",
      "value": expectedValue
    }
  ],

  "objectives": [
    {
      "id": "objective_1",
      "description": "Task shown in PDA",
      "required": true,
      "triggers": [
        {
          "type": "location_enter",
          "location": "quarters"
        }
      ],
      "onComplete": [
        {
          "type": "set_property",
          "property": "some.property",
          "value": "new_value"
        }
      ]
    }
  ],

  "onStart": [
    {
      "type": "show_dialogue",
      "characterId": "system",
      "dialogueId": "mission_intro"
    }
```

```
  ],

  "onComplete": [
    {
      "type": "unlock_crew",
      "crewId": "science"
    }
  ],

  "followUpMissions": ["next_mission_id"],

  "rewards": {
    "experience": 100,
    "items": [{"id": "reward_item", "quantity": 1}]
  },

  "hints": [
    {
      "condition": {
        "type": "objective_active_for",
        "objectiveId": "objective_1",
        "seconds": 30
      },
      "message": "Try looking in your quarters."
    }
  ]
}
```

Trigger Types

Location Triggers

```
json

{
  "type": "location_enter",
  "location": "quarters|bridge|logbook"
}
```

Fired when: Player navigates to a page/location

Item Triggers

```
json
```

```
{
  "type": "item_pickup",
  "itemId": "communicator"
}
```

Fired when: Player clicks an interactive element

Property Triggers

```
json

{
  "type": "property_change",
  "property": "tutorial.communicator_opened",
  "value": true
}
```

Fired when: Game state property changes to specific value

Dialogue Triggers

```
json

{
  "type": "dialogue_complete",
  "characterId": "executiveOfficer",
  "dialogueId": "first_contact"
}
```

Fired when: Player completes conversation with character

Action Types

Property Actions

```
json

{
  "type": "set_property",
  "property": "contacts.crew.captain.communicator",
  "value": true
}
```

Effect: Updates game state property

Crew Actions

```json
{
  "type": "unlock_crew",
  "crewId": "executiveOfficer"
}
```

Effect: Makes crew member available in communicator

Item Actions

```json
{
  "type": "add_item",
  "itemId": "communicator",
  "quantity": 1
}
```

Effect: Adds item to player inventory

Interactive Actions

```json
{
  "type": "spawn_interactive",
  "elementId": "communicator_pickup",
  "location": "quarters_desk"
}
```

Effect: Creates clickable object in scene

Dialogue Actions

```json
```

```
{
  "type": "show_dialogue",
  "characterId": "system",
  "dialogueId": "tutorial_message"
}
```

Effect: Shows message/dialogue to player

Contextual Actions

```
json

{
  "type": "add_contextual",
  "characterId": "executiveOfficer",
  "context": "post_mission_chat"
}
```

Effect: Adds conversation options for character

Creating New Missions: Step-by-Step

Step 1: Plan Your Mission

1. **Define the goal** - What should the player accomplish?
2. **Break into objectives** - What steps lead to the goal?
3. **Identify triggers** - How will you detect each step?
4. **Plan rewards** - What happens when complete?

Step 2: Create Mission File

1. Copy the template above
2. Give it a unique ID
3. Write clear objective descriptions
4. Set up appropriate triggers
5. Define completion actions

Step 3: Add to Index

```
json
```

```
{
  "missions": [
    "existing_mission.json",
    "your_new_mission.json"
  ]
}
```

Step 4: Test the Mission

```javascript
// In browser console
missionManager.debugMissionState();
missionManager.startMission('your_mission_id');
missionManager.triggerCustomEvent('item_pickup', {itemId: 'test'});
```

Step 5: Add Required Elements

- Interactive elements (if using `spawn_interactive`)
- Crew members (if unlocking characters)
- Property paths in game state

Example: Simple Exploration Mission

```json
```

```
{
  "id": "explore_bridge",
  "name": "Bridge Inspection",
  "description": "Familiarize yourself with the bridge systems",
  "category": "tutorial",
  "autoStart": false,

  "prerequisites": [
    {
      "type": "property_equals",
      "property": "tutorial.communicator_complete",
      "value": true
    }
  ],

  "objectives": [
    {
      "id": "visit_bridge",
      "description": "Go to the ship's bridge",
      "required": true,
      "triggers": [
        {
          "type": "location_enter",
          "location": "bridge"
        }
      ],
      "onComplete": [
        {
          "type": "set_property",
          "property": "locations.bridge.visited",
          "value": true
        }
      ]
    },
    {
      "id": "check_systems",
      "description": "Examine the helm console",
      "required": true,
      "triggers": [
        {
          "type": "item_examine",
          "itemId": "helm_console"
        }
```

```
        ],
        "onComplete": [
          {
            "type": "unlock_station",
            "station": "Navigation"
          }
        ]
      }
    ],

    "onComplete": [
      {
        "type": "set_property",
        "property": "tutorial.bridge_complete",
        "value": true
      }
    ],

    "rewards": {
      "experience": 50
    }
}
```

Interactive Elements

Interactive elements are objects that players can click in scenes. They're defined in `interactiveElements.js`:

javascript

```
this.elementDefinitions.set('helm_console', {
    type: 'examine',
    cssClass: 'interactive-examine helm-examine',
    position: { left: '45%', top: '30%', width: '15%', height: '20%' },
    cursor: 'pointer',
    tooltip: 'Helm Console - Click to examine',
    onClick: () => {
        this.examineItem('helm_console');
    }
});
```

Element Properties

- **type** - `'pickup'` or `'examine'`
- **cssClass** - CSS styling for element
- **position** - Where to place in scene (percentages)
- **cursor** - Mouse cursor style
- **tooltip** - Help text on hover
- **onClick** - Function to call when clicked

Debugging Missions

Common Console Commands

```javascript
// Check mission status
missionManager.debugMissionState();

// Check current objectives
console.log(missionManager.getCurrentObjectives());

// Check game state
console.log(gameStateInstance.getState());

// Force complete objective
missionManager.completeObjective('missionId', 'objectiveId');

// Trigger events manually
missionManager.triggerCustomEvent('location_enter', {location: 'bridge'});

// Spawn interactive element
interactiveElementManager.forceSpawn('element_id', 'location');
```

Common Issues

Mission Won't Start

- Check prerequisites match game state
- Verify `autoStart: true`
- Check console for loading errors

Objectives Won't Complete

- Verify trigger events are firing
- Check trigger data matches exactly
- Ensure event listeners are set up

Interactive Elements Won't Spawn

- Check container selectors exist
- Verify element definitions
- Check CSS positioning

Actions Don't Execute

- Check property paths exist in game state
- Verify crew IDs match contacts.json
- Check console for execution errors

Best Practices

Mission Design

1. **One clear goal per mission** - Keep focus simple
2. **Logical progression** - Each objective builds on previous
3. **Clear descriptions** - Player should understand what to do
4. **Meaningful rewards** - Unlock something useful
5. **Appropriate difficulty** - Match player's current abilities

Technical Implementation

1. **Unique IDs everywhere** - No duplicate mission/objective/element IDs
2. **Consistent naming** - Use camelCase for IDs, clear names
3. **Test prerequisites** - Ensure missions start when expected
4. **Validate triggers** - Test all trigger conditions work
5. **Handle edge cases** - What if player does things out of order?

File Organization

```
/data/missions/
  ├── index.json
  ├── tutorial/
```

```
│   ├── 01_find_communicator.json
│   ├── 02_meet_crew.json
│   └── 03_ship_systems.json
├── main_story/
│   ├── chapter_1_discovery.json
│   └── chapter_2_contact.json
└── side_missions/
    ├── explore_ocean_floor.json
    └── crew_personal_stories.json
```

This system gives you powerful tools to create rich, interactive storytelling experiences while keeping the technical complexity manageable. Each mission is self-contained but can connect to others, creating a flowing narrative experience for your players.