Novellium Game Format Documentation

Technical specification for Novellium visual novel game files

Overview

Novellium games are distributed as ZIP files containing JSON configuration and asset files. This document defines the complete format specification for compatibility and development purposes.

File Structure

Required Structure

```
game-name.zip
├── config.json              # Game metadata and settings
├── characters.json          # Character definitions
├── story.json              # Events and narrative flow
└── assets/                 # Game assets (optional)
    ├── backgrounds/        # Background images
    ├── sprites/           # Character sprites
    └── audio/            # Audio files (future)
```

File Requirements

- **ZIP Format**: Standard ZIP compression
- **UTF-8 Encoding**: All JSON files must use UTF-8 encoding
- **Case Sensitivity**: File and folder names are case-sensitive
- **No Subdirectories**: Assets should be in flat folders (backgrounds/, sprites/, audio/)

JSON Schema Definitions

config.json

Purpose: Contains game metadata and engine configuration

Required Fields:

```
{
  "title": "string",           # Game title (1-100 characters)
  "author": "string",          # Author name (1-50 characters)
  "startEvent": "string"       # ID of the first event
}
```

Optional Fields:

```
{
  "description": "string",     # Game description (0-500 characters)
  "version": "string",         # Version identifier
  "tags": ["string"],          # Genre/content tags
  "language": "string",        # Language code (en, es, fr, etc.)
  "rating": "string",          # Content rating (G, PG, PG-13, R)
  "created": "ISO-8601",       # Creation timestamp
  "modified": "ISO-8601"       # Last modified timestamp
}
```

Complete Example:

```
{
  "title": "Love Quest",
  "author": "Jane Developer",
  "description": "A romantic visual novel about finding love in college",
  "startEvent": "intro",
  "version": "1.0.0",
  "tags": ["romance", "school", "choices"],
  "language": "en",
  "rating": "PG",
  "created": "2025-10-26T12:00:00Z",
  "modified": "2025-10-26T15:30:00Z"
}
```

characters.json

Purpose: Defines all characters in the game

Structure:

```
{
  "characters": {
    "character_id": {
      "name": "string",         # Display name
      "color": "string",        # Hex color for dialogue (#RRGGBB)
      "sprites": {              # Character sprites (optional)
        "emotion": "path"       # Sprite paths relative to zip root
      }
    }
  }
}
```

Character ID Rules:

- Must be unique within the game

- Use lowercase letters, numbers, underscores only
- No spaces or special characters
- Examples: `alice`, `main_character`, `villain1`

Sprite Emotion Names:

- Common: `neutral`, `happy`, `sad`, `angry`, `surprised`, `love`
- Custom emotions allowed
- File paths relative to ZIP root: `sprites/alice_happy.png`

Complete Example:

```
{
  "characters": {
    "alice": {
      "name": "Alice Johnson",
      "color": "#ff6b9d",
      "sprites": {
        "neutral": "sprites/alice_neutral.png",
        "happy": "sprites/alice_happy.png",
        "sad": "sprites/alice_sad.png",
        "angry": "sprites/alice_angry.png"
      }
    },
    "bob": {
      "name": "Bob Smith",
      "color": "#4a90e2",
      "sprites": {
        "neutral": "sprites/bob_neutral.png",
        "smiling": "sprites/bob_smile.png"
      }
    },
    "narrator": {
      "name": "Narrator",
      "color": "#666666"
    }
  }
}
```

story.json

Purpose: Defines all events and story flow

Structure:

```
{
  "events": {
    "event_id": {
      "type": "event_type",     # Event type (required)
```

```
      "text": "string",          # Display text (required for most types)
      "next": "event_id",        # Next event ID (optional)
      ...                        # Type-specific fields
    }
  }
}
```

Event ID Rules:

- Must be unique within the game
- Use lowercase letters, numbers, underscores only
- Descriptive names recommended: `intro`, `meet_alice`, `choice_help_friend`

Event Types

1. Dialogue Events

Purpose: Character speaking to player or other characters

```
{
  "type": "dialogue",
  "character": "character_id",  # Must exist in characters.json
  "text": "Hello there!",       # What the character says
  "background": "backgrounds/school.jpg",  # Optional
  "sprite": "happy",            # Optional sprite emotion
  "next": "next_event_id"       # Optional
}
```

2. Narration Events

Purpose: Story text without a specific character

```
{
  "type": "narration",
  "text": "You walk through the empty hallway...",
  "background": "backgrounds/hallway.jpg",  # Optional
  "next": "next_event_id"                   # Optional
}
```

3. Choice Events

Purpose: Present options to the player

```
{
  "type": "choice",
  "text": "What do you do?",      # Question or context
```

```
  "background": "backgrounds/crossroads.jpg",  # Optional
  "choices": [
    {
      "text": "Go left",         # Choice text
      "next": "left_path",       # Where this choice leads
      "condition": "has_key"     # Optional condition
    },
    {
      "text": "Go right",
      "next": "right_path"
    }
  ]
}
```

4. Scene Events

Purpose: Change scene settings without dialogue

```
{
  "type": "scene",
  "text": "The next morning...",   # Optional transition text
  "background": "backgrounds/morning.jpg",
  "audio": "audio/morning_birds.mp3",  # Optional (future)
  "next": "next_event_id"
}
```

Advanced Features

Conditional Logic

Events and choices can include conditions:

```
{
  "type": "dialogue",
  "character": "alice",
  "text": "Thanks for helping me!",
  "condition": "helped_alice == true",  # Only show if condition met
  "next": "grateful_response"
}
```

Condition Syntax:

- Variables: `variable_name`
- Comparisons: `==`, `!=`, `>`, `<`, `>=`, `<=`
- Logical: `&&` (and), `||` (or), `!` (not)
- Values: `true`, `false`, numbers, strings in quotes

Variable Creation:

- Automatically created from choice selections
- Choice with `next: "helped_alice"` creates `helped_alice = true`
- Manual variables possible in future versions

Multiple Endings

Games can have multiple ending events:

```
{
  "good_ending": {
    "type": "narration",
    "text": "You lived happily ever after!",
    "background": "backgrounds/sunset.jpg"
    // No "next" field = game ends
  },
  "bad_ending": {
    "type": "narration",
    "text": "Maybe next time...",
    "background": "backgrounds/rain.jpg"
    // No "next" field = game ends
  }
}
```

Asset Specifications

Images

Background Images:

- **Recommended Size**: 1920x1080 (16:9 ratio) or 1280x720
- **Minimum Size**: 800x600
- **Formats**: PNG, JPG, GIF
- **File Size**: Under 2MB per image recommended
- **Location**: `backgrounds/` folder

Character Sprites:

- **Recommended Size**: 300-500px wide, maintain aspect ratio
- **Format**: PNG recommended (supports transparency)
- **File Size**: Under 1MB per sprite recommended
- **Location**: `sprites/` folder
- **Naming**: `character_emotion.png` (e.g., `alice_happy.png`)

General Image Guidelines:

- Use web-safe formats (PNG, JPG, GIF)
- Optimize file sizes for web delivery

- Consistent art style across all images
- Appropriate resolution for target devices

Audio (Future Feature)

Background Music:

- **Formats**: MP3, OGG, WAV
- **Location**: `audio/` folder
- **File Size**: Under 10MB per track
- **Looping**: Should loop seamlessly

Sound Effects:

- **Formats**: MP3, OGG, WAV
- **Location**: `audio/` folder
- **File Size**: Under 1MB per effect
- **Duration**: Usually under 5 seconds

Validation Rules

Required Validations

1. **File Structure**: All required JSON files present
2. **JSON Syntax**: Valid JSON in all configuration files
3. **Character References**: All character IDs referenced in events exist
4. **Event Flow**: Start event exists, no broken event chains
5. **Asset Paths**: All referenced assets exist in ZIP

Recommended Validations

1. **Event Connectivity**: All events reachable from start event
2. **Image Optimization**: Images under recommended file sizes
3. **Text Quality**: No obvious typos or formatting issues
4. **Choice Balance**: Multiple choice paths available

Error Handling

Common Errors and Solutions

Invalid JSON Syntax:

```
Error: Unexpected token in JSON
Solution: Check for missing commas, quotes, or brackets
```

Missing Character:

```
Error: Character 'alice' not found in characters.json
Solution: Add character definition or fix character ID
```

Broken Event Chain:

```
Error: Event 'missing_event' referenced but not defined
Solution: Create missing event or fix event ID reference
```

Missing Asset:

```
Error: Image 'backgrounds/missing.jpg' not found
Solution: Add image file or fix path in event
```

Version Compatibility

Current Version: 2.0

- Supports all features documented above
- Compatible with cloud deployment
- Backward compatible with 1.x games

Legacy Support (1.x)

- Basic dialogue, narration, choice events
- Simple character system
- Local-only deployment

Future Versions

- Audio support (3.0)
- Advanced scripting (4.0)
- Plugin system (5.0)

Best Practices

File Organization

- Use descriptive filenames
- Organize assets in appropriate folders
- Keep file sizes reasonable for web delivery
- Use consistent naming conventions

JSON Structure

- Format JSON for readability
- Use meaningful event and character IDs
- Comment purpose of complex events in descriptions
- Validate JSON syntax before packaging

Game Design

- Test all choice paths
- Ensure events connect properly
- Provide clear player feedback
- Design for mobile and desktop

Performance

- Optimize image file sizes
- Limit number of large assets
- Test loading times on slow connections
- Use appropriate image formats

Novellium Game Format v2.0
Last updated: October 2025