Table of Contents

[Intro/Overview 1](#_Toc167011275)

[Full Gameplay Flow 3](#_Toc167011276)

[Core Game Updates Needed 4](#_Toc167011277)

[Adaptation Requirements 4](#_Toc167011278)

[Adaptation To Do: 6](#_Toc167011279)

[Use XML File Source *[Done, 5/5/24]* 6](#_Toc167011280)

[Player Character Creation 7](#_Toc167011281)

[Verify Gameplay Works Via Web Browser 8](#_Toc167011282)

[Add Abstraction Layer Temp with Good OOP 8](#_Toc167011283)

[Implement Combat Functionality 8](#_Toc167011284)

[Ensure BCI Friendliness 8](#_Toc167011285)

[Abstraction Layer Update 8](#_Toc167011286)

[Add Server Functionality 9](#_Toc167011287)

[Additional Missing Components 9](#_Toc167011288)

# Intro/Overview

1/21/2024 (1351): Very casual doc format, just to get a clear handle on what’s needed for adapting the demo game to the MVP game that works with toolset.

As a general overview, we have essentially three deliverables for this project (for MVP, or rather initial testing):

* Core Game:
  + Is entirely BCI-friendly
  + Executes the game, using:
    - XML delivered module
    - CSV character sheets
    - Libraries/dictionaries
  + Ideally uncoupled from the Default Core Binary Percentage system where possible
  + With GSAL in mind, if not yet built
  + Looks up available genres through libraries and allows players to select one for a given module
  + Allows for served game, run by a GM with players
* Libraries/Dictionaries:
  + Maybe delivered by XML but living in final form as a database
    - At this stage PostgreSQL most likely?
  + Provides options for available genres
  + Provides generics by genre lookup
  + Will also have to provide GSAL to GSP lookups for the game system.
* Modgodtoolset:
  + Allows for the creation of XML modules to be used by the core game
  + Goal is to use Aurora-inspired GUI
  + Allows creator to use generics for words to be replaced by genre lookup, so the module is genre-agnostic (selectable at the game level).
  + Can set difficulty modifiers for checks and tasks.
    - Think from the role-master system.
      * Should also have ‘impossible’ or ‘automatic’ for checks’
      * Also astronomical odds, for example. Try to keep each game system’s probability curve.
  + This is also game-system agnostic, but can be forced/required.
  + NPC creation should be more into our system
    - For 1.0 be as generic as possible, it will be flawed

We have all (or almost all) of this documented elsewhere, but as it’s substantially evolved over time, I’m using this to clarify adaptation thinking.

## Full Gameplay Flow

So now, the flow for playing a game session could be:

1. Player (or GM) initiates a game.
   1. They provide the module XML (this can be stored locally, on self-hosted server, or central server).
   2. All players have access to the game and start it on their own machines and connect via server. Can be:
      1. This can be application running locally.
      2. This can be accessed through web browser.
   3. All players have a player character CSV.
      1. This can be created in-game.
      2. This is communicated across the server connection.
2. The GM (or lead player) selects the game system and the genre.
   1. This is communicated among the players, or served to them. Right now there should only be one game and genre defined by the GM per session.
3. The game:
   1. Draws all game information from the XML, including:
      1. Locations (in nested structure)
      2. Map connections using above
      3. Descriptions and options
      4. NPCs/monsters
      5. Treasures and rewards/penalties
      6. Relevant checks
      7. All dialogue
   2. Uses libraries/dictionaries to update all text for the genre and game type
      1. Via connection if not bundled with game
   3. Throughout, the game’s GSAL translates core game mechanics through GSP to provide modified outputs.
   4. Allows for the creation or loading of a player character. The display for this will use GSP/genre as well.
   5. Will have a gamemaster role, like Neverwinter Nights, which allows for heighted powers during game play.
      1. This is not be necessary to be functional for MVP.

# Core Game Updates Needed

Working from our demo game, there are a lot of modifications to be made.

## Adaptation Requirements

1/28/24 (1324): Adapting from the current Demo version, here are the requirements needed (in a highly casual format):

* Most important piece is game can be run by BCI (so simple, turn-based). Updates cannot break this setup. Core purpose remains: **play a turn-based RPG with your thoughts.**
* Player Characters (current version was never fully in place):
  + Create character:
    - Can take the game system choice for the character display
    - Will save locally in a CSV
    - Will actually save modifiers to our percentile system that are adaptable
  + Load character:
    - Will load a CSV with the appropriate format
    - Will display per the chosen game system
* For demo/MVP, consider replacing the library/dictionary element with a local testing example, such as JSON. This can then be replaced by connecting to external DB, or whatever final solution is.
* Loads module from XML file (change from current, using JSON).
  + All game elements from XML must be adaptable:
    - For varying genre options (starting with just 3 default but ability to add more)
      * Sci fi
      * Current day
      * Historical fantasy (swords and sorcery?)
    - For varying RPG systems (starting with 2, and maybe the underlaying system):
      * Underlaying percentile system (uses BCI RPG temp Hawke PDF)
      * BFRPG
      * OpenD6
  + This XML must include game map with location format as agreed upon.
    - Has hierarchy structure:
      * Region (name, description, folder)
        + Location (name, description, folder)

Space (name, description, folder)

Scene (name, description, also contains game elements)

* + XML contains game text, and all options.
  + XML contains connections between options and next game text
  + XML contains all statistics required in-game
    - NPCs and monsters
    - Checks and challenges
    - Treasures
    - Rewards/penalties and other
* Adds server functionality (never in demo game)
  + Allows running a game
  + Allows for joining an existing session
  + Turn-based, using our predefined logic for option selection
    - \*\*For demo if not MVP, consider a simplification of this
  + Handles communication and multiple-player aspects
* Adds combat functionality (was possible in demo game, with rolls and the like coming from JSON, but not implemented)
* Uses our library/dictionary option for:
  + Replacing flagged generics
  + Setting character sheet display (will only have modifiers on the character sheet)
  + All game system/rules output
* Is as OOP as possible, allowing for various updates to be made

## Adaptation To Do:

2/4/24 (1312): Specific steps here needed to adapt demo game into the MVP game per the requirements above.

### Use XML File Source *[Done, 5/5/24]*

2/4/24 (1325): This is the simplest conversation initially, so starting here.

Planned steps:

1. ~~Replace the JSON (as is) with XML file version. Run and test.~~
   1. Completed 4/21/24. Required mod to the Dice Roller, to type-safe the strings coming through. Not sure why this was functioning in the JSON version, which was also sending strings into die type array.
2. ~~Replace the current location to utilize location hierarchy structure~~
   1. Completed 5/5/24. Adds a <Start>True</Start> to the <Space>. It can ignore or process a <Start>False</Start>, but expects just one—if there are multiple true starts it will work from the first one it finds (ala Aurora Toolset).

### Player Character Creation

2/4/24 (1341): Update/replace the very roughed demo game character creation for the requirements.

Planned steps:

1. Structure for GUI replacement via GAL (can show different labels, statistics and apply varying logic to translate from #2)
2. Build default binary/percentile system (underlying) character creation
3. Output to CSV
4. Input/Load from CSV

### Verify Gameplay Works Via Web Browser

Revisit game at this level to be sure it can run as both application (default) and from browser (served via our server, current temporary method OK for this initial step).

### Add Abstraction Layer Temp with Good OOP

Using a temporary Dictionary/Library (local JSON, for example), build in the abstraction layer to demo game in a way that’s expandable for future needs.

Planned steps:

1. Load aspects from the temp source
2. Update menu to allow for loading genre taken from library (#1)
3. Update menu to allow for loading game system take from library (#1)
4. Update player character creation output to display player character taken from #3 (via #1)
5. Update player character creation to allow for creating character using chosen game system (#4)
6. Update game to replace generics in XML per genre choice (#2 via #1)
7. Update game to replace die/game outputs per game choice (#3 via #1)

### Implement Combat Functionality

Add combat functionality. Demo went no further than implementing die rolling and test ability checks. This explodes out into a number of subcategories (like initiative, special abilities, and more).

### Ensure BCI Friendliness

Updates the demo game to ensure all navigation and gameplay aspects are runnable by BCI-capable inputs.

### Abstraction Layer Update

Update the temporary library/dictionary system to use our server/DB as required, in a secure fashion.

### Add Server Functionality

Build the required server functionality into the game. Note that this was never in the demo game.

### Additional Missing Components

Add missing/broken aspects from the original requirements not yet in place in the Demo, as needed for the MVP.

## 