

Project Dreamer / Menu Module

Architecture/Design Document

Change History	1
Introduction	2
Design Goals	2
System Behaviour	2
Logical View	3
High-Level Design (Architecture of the Entire system)	3
Mid-Level Design of Menu Module	4
Detailed Class Design of the Menu Module	4
Process View of the Menu Module	4
Use Case View	5

Change History

Version: 0.1

Modifier: Joshua Griffis

Date: 06/04/2022

Description of Change: Design Document started

Introduction

This document describes the architecture and design for the Project Dreamer application being developed by Radical Dreamers. Project Dreamer is a Third Person Role Playing Game where you explore the dream world and interact with its inhabitants and fight bad dreams.

The purpose of this document is to describe the architecture and design of the Project Dreamer application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

- Developers – they want an architecture that will minimise complexity and development effort.
- Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialise in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all game logic.
- Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain into the future.

Design Goals

The goals we decided on for the design were:

- A way to navigate and display different ways to interact with the game's systems
- The main screen will display basic information like the party's HP and SP
- Buttons to navigate between all the menus we have planned.

System Behaviour

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expected system behaviour in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document (GDD).

The menu system will be interacting with many of the game's systems, so we need to make sure it has references to the multiple managers we have in the game like the Party Manager or the Database Manager. The different screens will have different modules because of how they interact with their own systems.

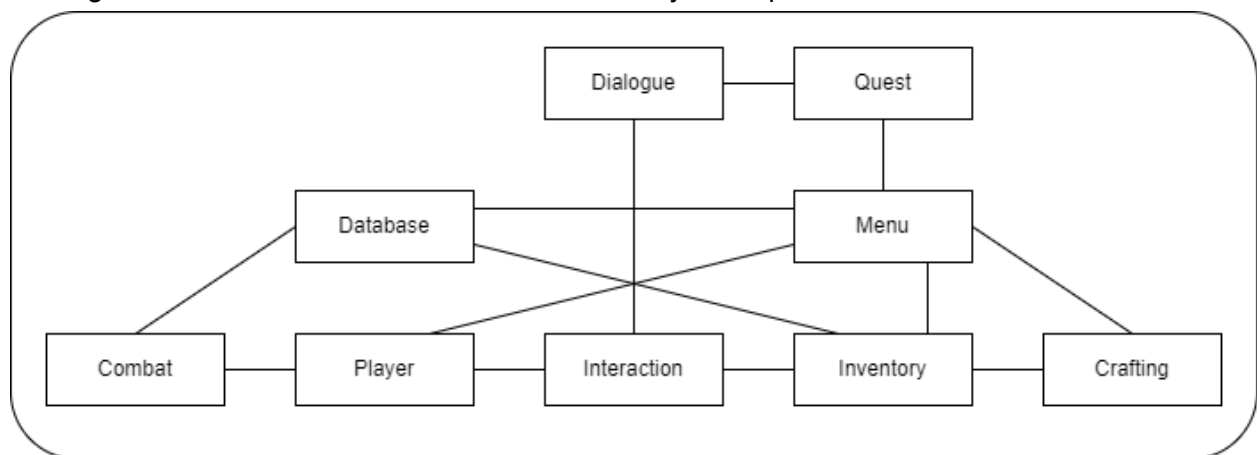
Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations

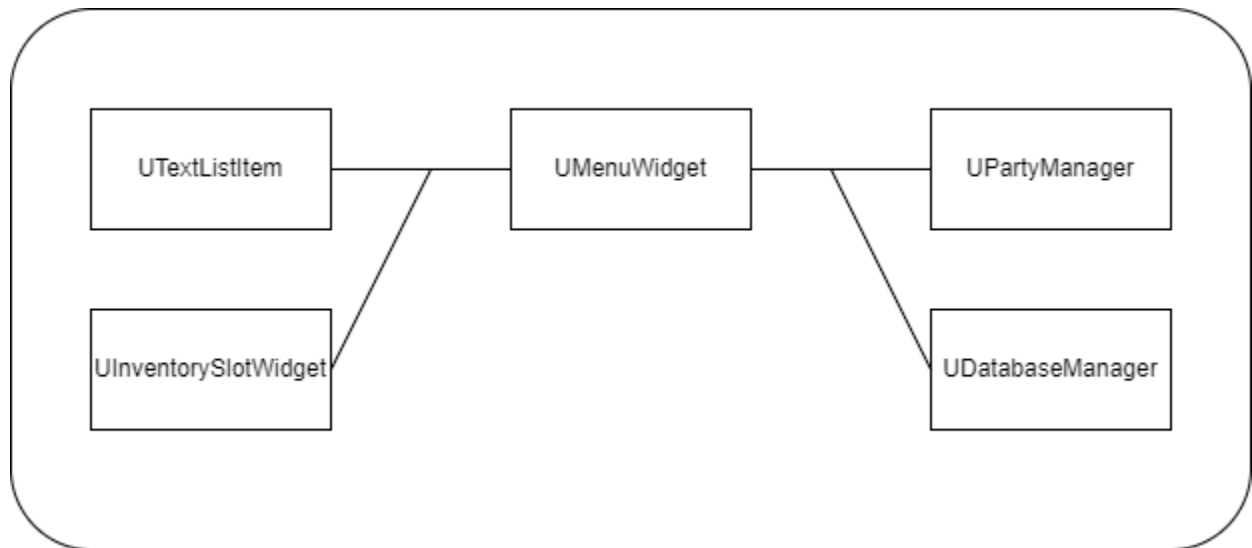
High-Level Design (Architecture of the Entire system)

The high-level view or architecture consists of 9 major components:



- **Player:** is the main control over the character in the world and allows the user to interact with the world.
- **Database:** stores the data needed for many features to work.
- **Interaction:** handles the objects the player can interact with like talking to characters, opening chests, opening doors, etc.
- **Dialogue:** is responsible for handling the flow of conversations and displaying dialogue to the user.
- **Inventory:** manages the items picked up by the player and money stored. Allowing the player to use them at a later point.
- **Menu:** allows interaction with some modules and features.
- **Combat:** allows interaction between enemies and player characters as they fight. Controls the flow of battle and which battler may act.
- **Quest:** allows the user to have stored data for quests and be able to receive rewards upon completion.
- **Crafting:** allows the player to combine multiple items together to create new items

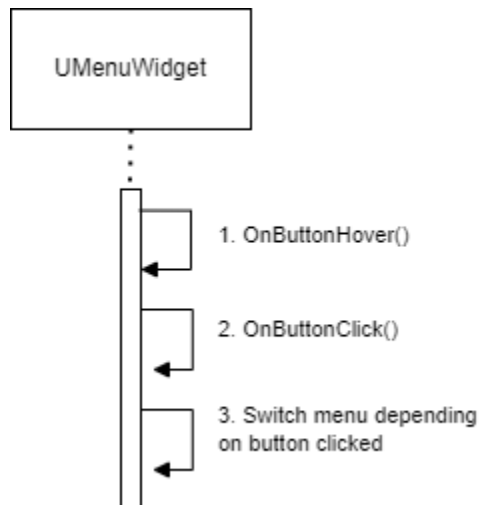
Mid-Level Design of Menu Module

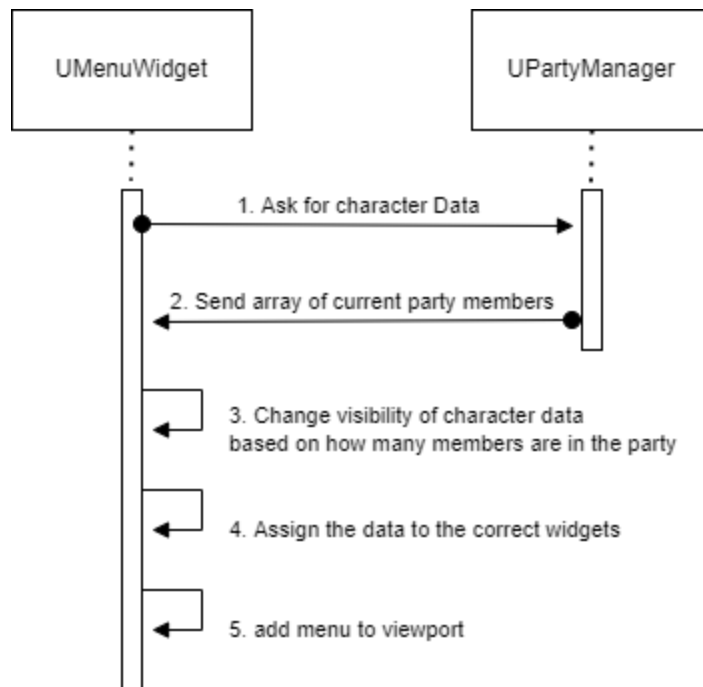


Detailed Class Design of the Menu Module

See UMenuWidgetUML.png

Process View of the Menu Module





Use Case View

Press Tab