Project Dreamer / Movement 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 Movement Module 4
- Detailed Class Design of the Movement Module 4

Process View of the Movement Module 5

Use Case View 5

Change History

Version: 0.1
Modifier: Joshua Griffis
Date: 12/03/2022
Description of Change: Design Document started

Version: 0.2
Modifier: Joshua Griffis
Date: 15/03/2022
Description of Change: Updated Mid and Detailed View

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:

- Player movement would be limited to not let players fall
- Use NavMesh to limit the movement
- No Jumping

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 movement system is designed to let players walk in a 3D environment while not allowing the player to fall down from locations. A NavMesh will be used to check if the player can move to their new location.

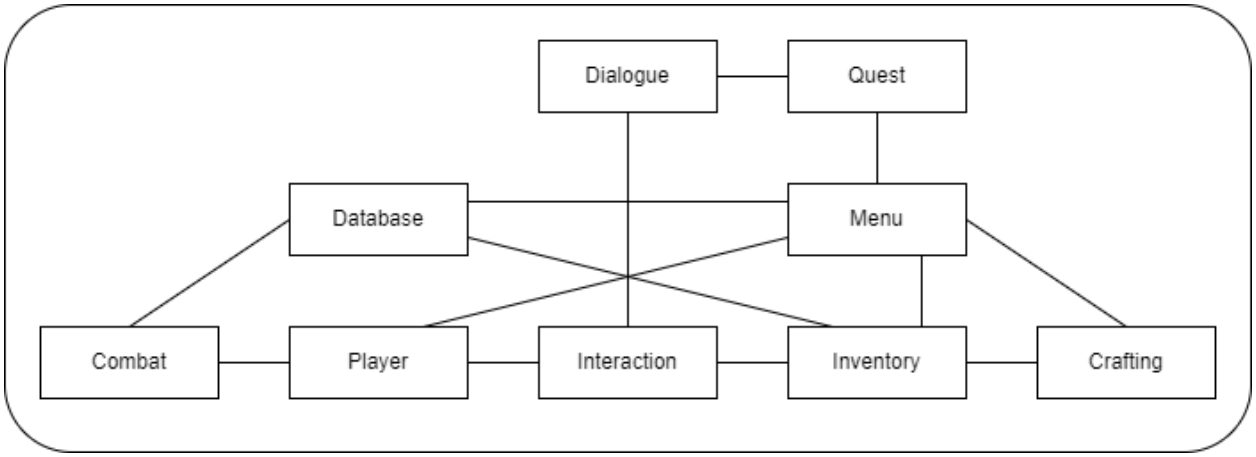
Logical View

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

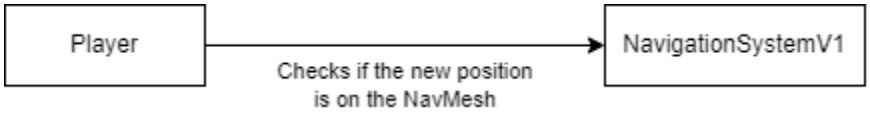
High-Level Design (Architecture of the Entire system)

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

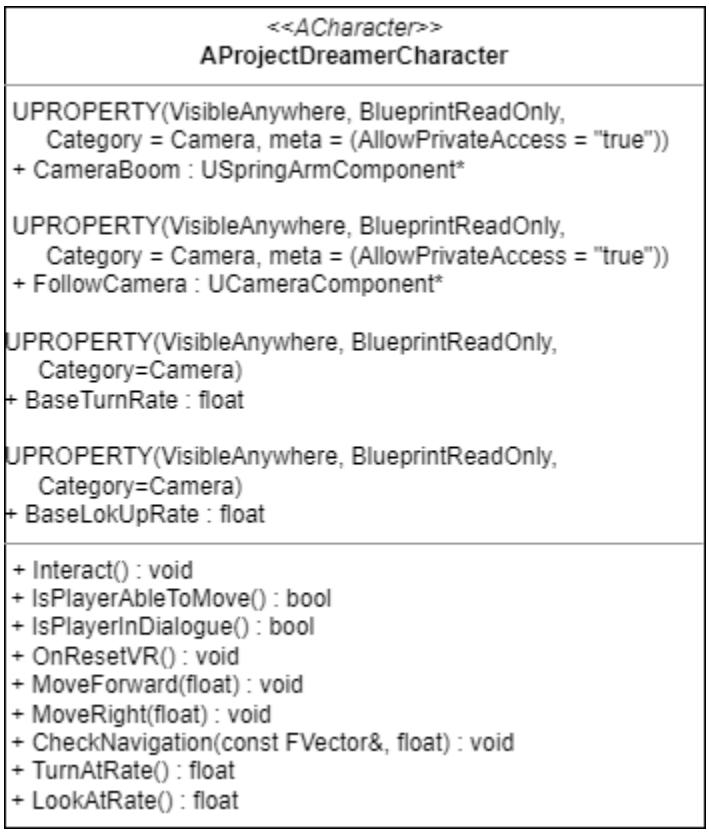


- **Player:** is the main control over the character in the world and allows the user to interact with the world.
- **Database:** stores the data needed for many features to work.
- **Interaction:** handles the objects the player can interact with like talking to characters, opening chests, opening doors, 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

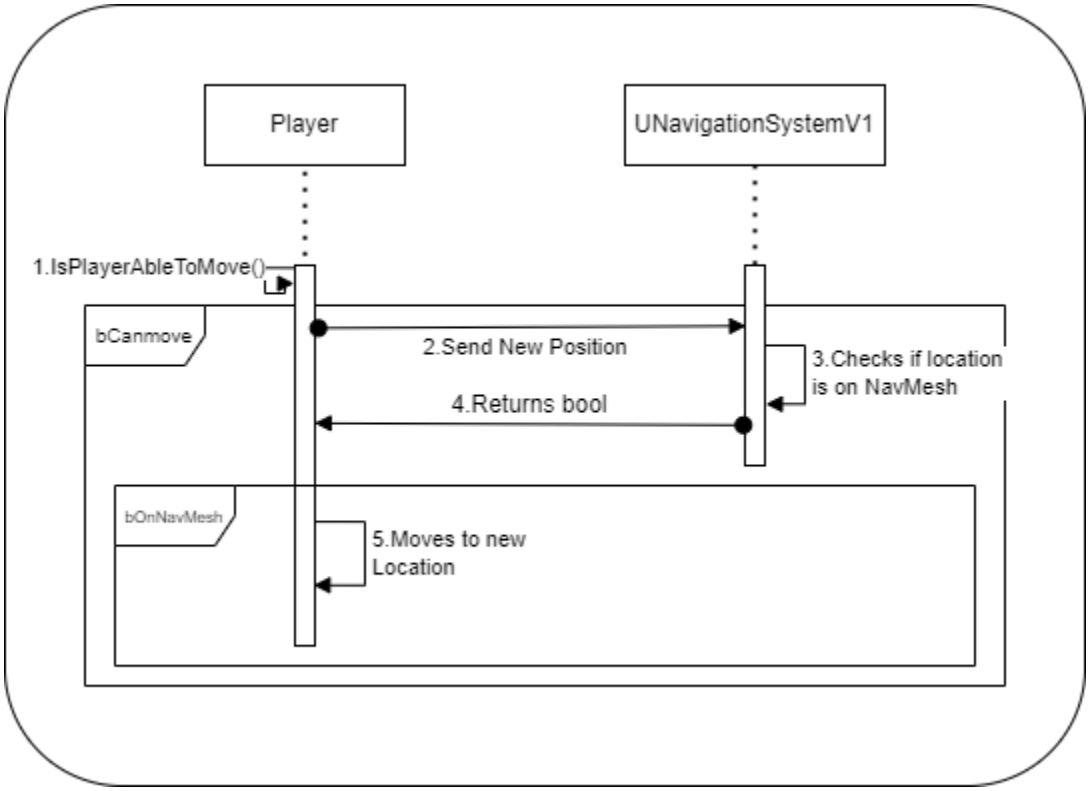
Mid-Level Design of Movement Module



Detailed Class Design of the Movement Module



Process View of the Movement Module



When the player begins movement, the player will make a check to the Navigation System to check if the player is going to still be on the NavMesh. If the new position is on the NavMesh the player will continue moving, and if it is not, the player will not move.

Use Case View

Any level you want the player to move on, you will add a NavMeshVolume object to the level and then size it to fit the whole level or just desired locations to move.