

**COLD NITES**

**Alpha-1**

**Player Character Design Document**

Designed & Implemented by

Jonathan Sime

**Change History**

**Version:** 0.1

**Modifier:** Jonathan Sime

**Date:** 03/01/2021

**Description of Change:** Created Design document and modified contents of items 1 and 2

**Version:** 0.2

**Modifier:** Jonathan Sime

**Date:** 03/08/2021

**Description of Change:** Modified contents of items 3 and 4

**Version:** 0.3

**Modifier:** Jonathan Sime

**Date:** 03/13/2021

**Description of Change:** Finished modifying contents

**Version:** 1.0

**Modifier:** Jonathan Sime

**Date:** 03/15/2021

**Description of Change:** Formatted to be consistent with other team documents

**Table of Contents**

**1 Introduction 4**

**2 Design Goals 5**

**3 System Behavior 6**

**4 Logical View 7**

**4.1 High-Level Design 7**

**4.2 Mid-Level Design 9**

**4.3 Detailed Class Design 9**

**5 Process View 10**

**6 Use Case View 12**

**1. Introduction**

This document describes the architecture and design for the Cold Nites application being developed for GAM1528 High Level Development. Cold Nites is a turn-based puzzle game in a sleeping/dreaming setting.

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

* Developers - For an architecture that minimizes complexity and development effort
* Project Manager - For an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies.
* Maintenance Programmers - For assurance that the system will be easy to evolve and maintain in the future.

**2. Design Goals**

The main design priorities for the Player system are shown below:

* A design that should minimize complexity and development effort.
* A design that allows the player to use the character simply and without issues.

**3. System Behavior**

This system will allow the player to move the playable character around the level to pick up mission items or collectibles for points. The player will be able to move while it is their turn and may not move during AI turns. The player will also be able to forgo moving to activate interactable inside the level to continue the levels.

**4. Logical View**

The logical view describes the module from a high concept to a detailed analysis of the interactions and possible use cases.

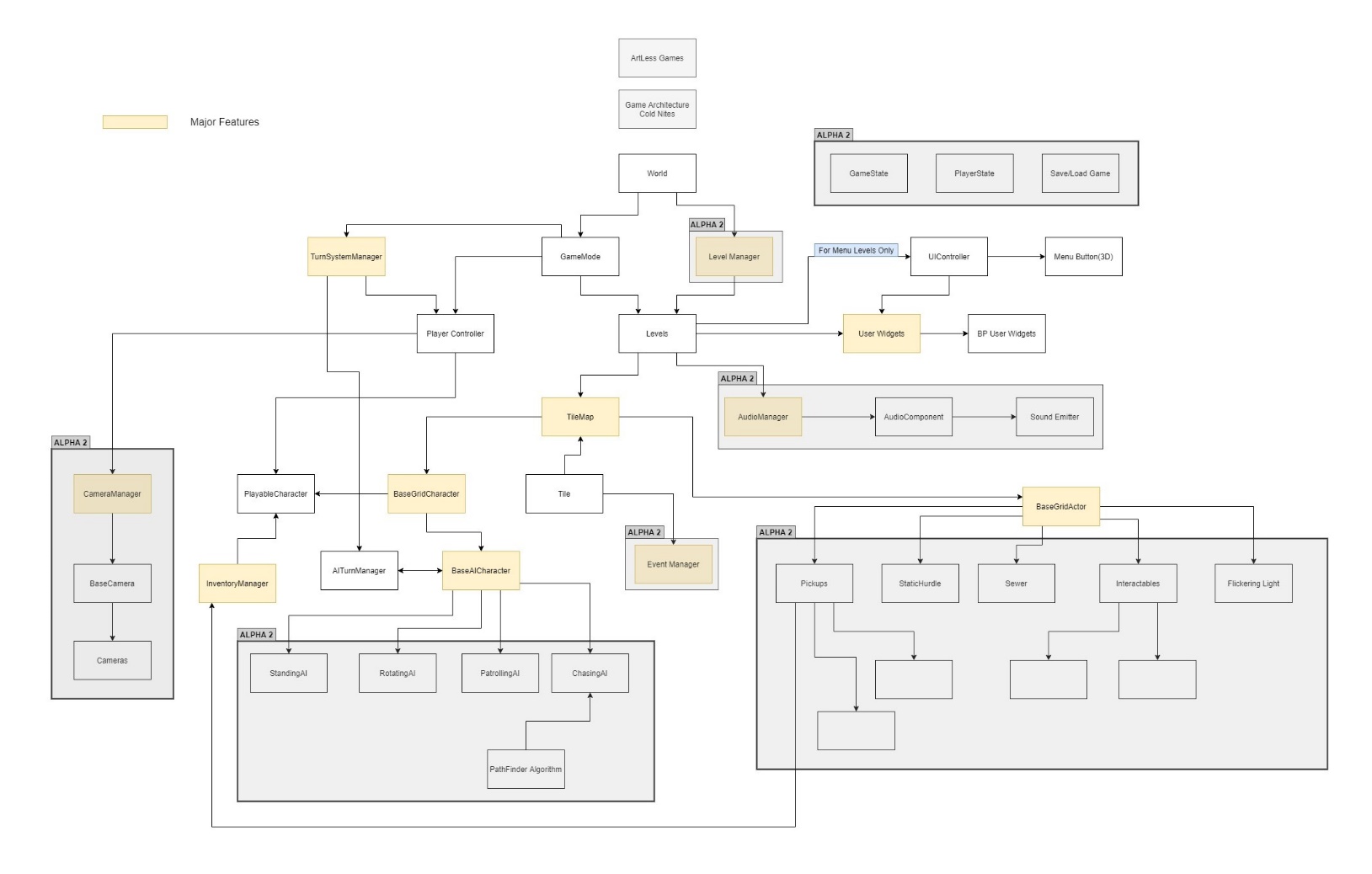
**4.1High-Level Design**

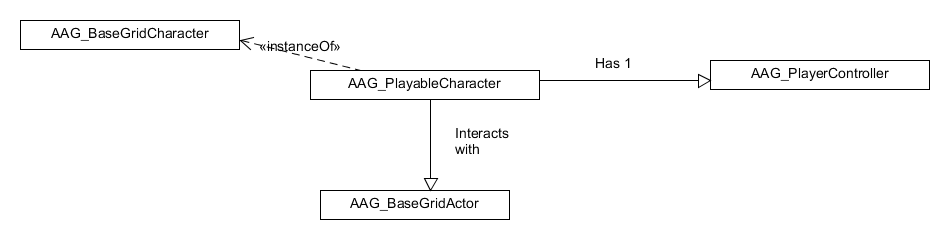
The high-level view consists of 4 major components:

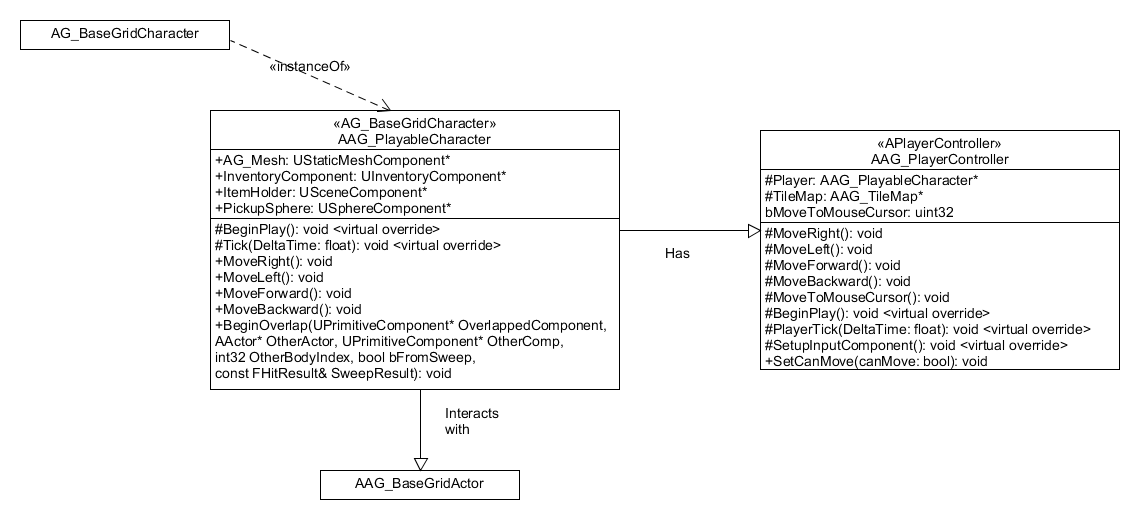
* Player system consists of a controllable character that inherits from BaseGridCharacter, which takes user inputs to perform appropriate moves.
* TileMap system provides the grid-based behavior for the game and will facilitate the event system, based on the actor present on the Tiles.
* Inventory system stores the item for the corresponding actor and will allow the player easy access to any collectible throughout the game.
* User Interface is responsible for Main Menu and any in-game HUD (or User Widget) with which the player can interact.

The primary features for the Alpha 1 release:

1. **TileMap** - TileMap provides the grid-based behaviour for the game and will facilitate the event system, based on the actor present on the Tiles.
2. **Turn-Based System** - This provides the turn-based aspect for the game. It is responsible for maintaining the turn order for all the world elements(actors) and the player.
3. **Base Grid Classes** - These classes work as a foundation class for all the actors/characters spawned in the game. These classes are closely integrated with handling the TileMap(Grid-Base) Behaviour of the game.
4. **Player Character** - Player is a controllable character that inherits from BaseGridCharacter, which takes user inputs to perform appropriate moves.
5. **Inventory System** - The pickup function helps the player grab the items on the map. Inventory stores the items for the corresponding actor and will allow the player easy access to any collectible throughout the game, and it also assists in equipping the stored items.
6. **Menu Interface** - The Menu Interface will be responsible for Main Menu and Pause Menu with which the player can interact.

******

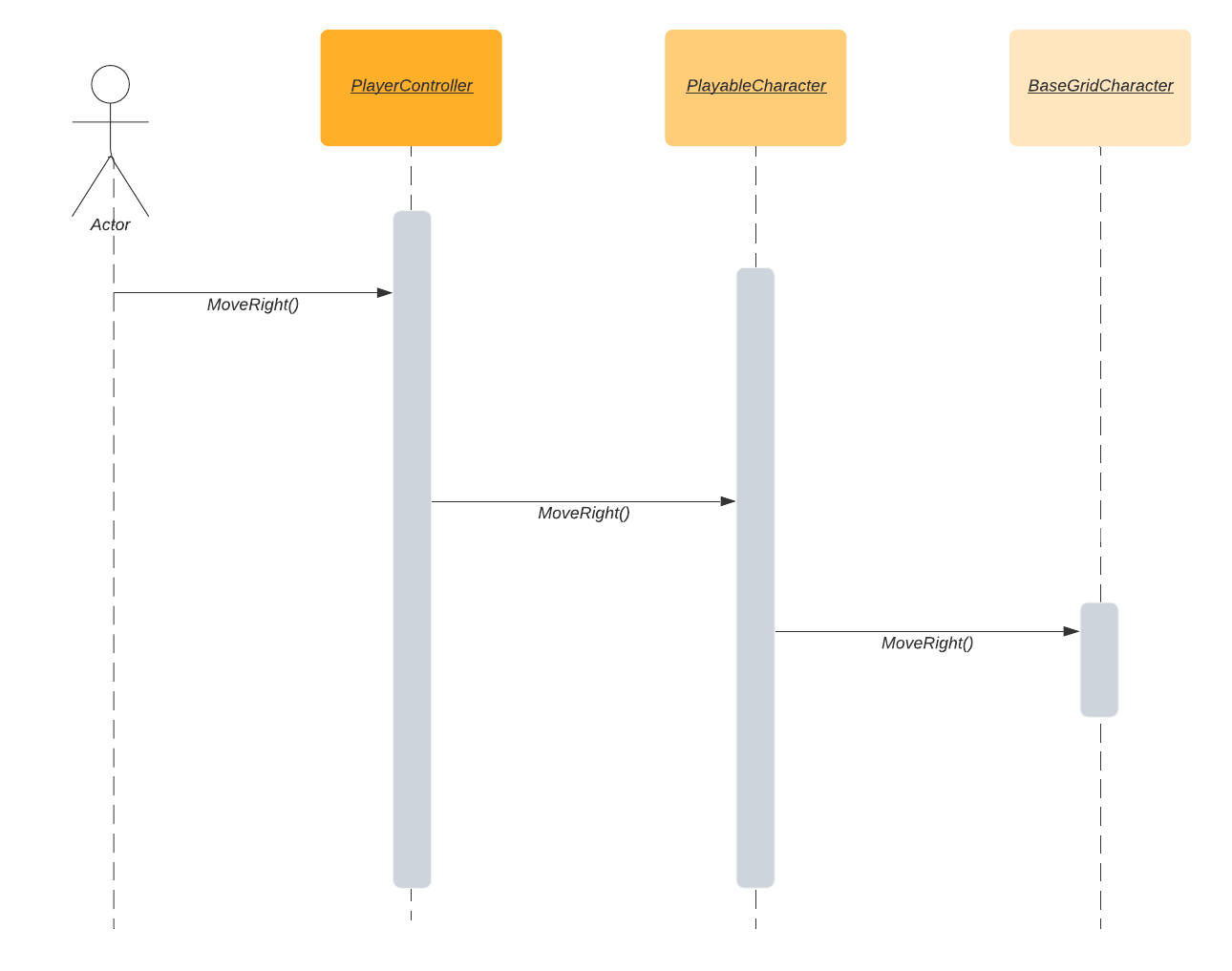
**4.2 Mid-Level Design**

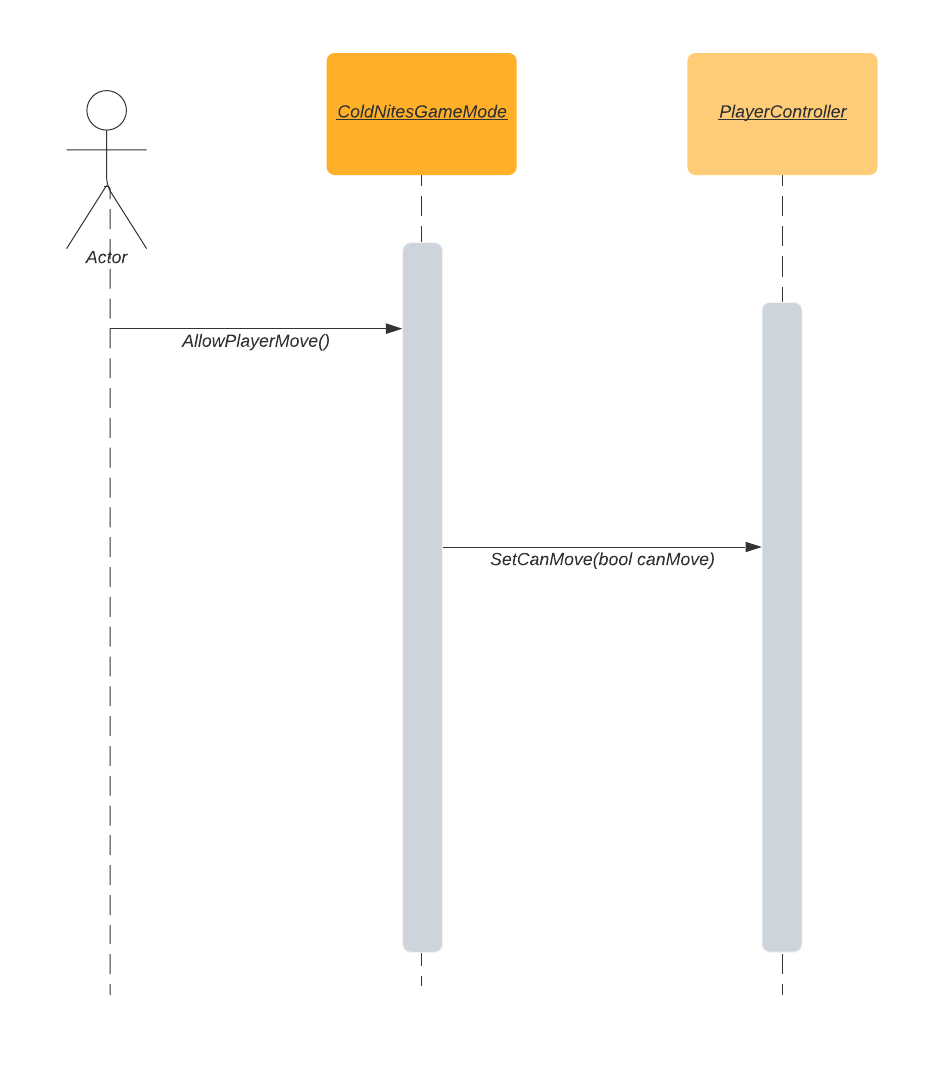
**4.3 Detailed Class Design**

**5. Process View**

The process view will explain the relation and interaction between various cases using Sequence and Collaboration Diagrams.

When it is the player’s turn and he wishes to move right, the input will go through the PlayerController->MoveRight() which in turn will call the PlayableCharacter->MoveRight(), which will call its parent’s BaseGridCharacter->MoveRight(). This same sequence applies for moving left, backwards, and forwards.

****When a new turn begins and the player gets to move. The GameMode will call AllowPlayerMove which will call PlayerController->SetCanMove(true). After the call the player will be able to make inputs to continue the game.



**6. Use Case View**

**Movement**

The player puts an input with either the WASD keys or the Left Mouse Button, which will then be processed by the PlayerController to find out which direction the player wants to head, and will call the function for that accordingly. The player will continue to move until the middle of the target tile. Once there, the player's turn will end.

**Interactions**

**This has not been implemented yet as buttons or interactables haven’t been added yet.**

If the player wishes to interact with a button of level, or any interactable, the player will lose their movement for the turn. This forces the player to think about when to interact and when to move.