****

**COLD NITES**

Alpha-2

**Camera Manager Design Document**

Designed & Implemented by

Yash Chamria

**Change Log**

**Version** – 0.2

**Modifier** – Yash Chamria

**Date** – 28th March, 2021

**Description** – Created the Design Document Body.

**Table of Contents**

1. **Introduction 4**
2. **Design Goals 5**
3. **System Overview and Behaviour 6**
4. **Logical View 7**

**A. High-Level Design Architecture of the Entire System**

**B. Mid-Level Design of Camera Manager**

**C. Detailed Design of Camera Manager**

1. **Process View 11**

**A.**

**B.**

1. **Use Case View and Practice 18**
2. **In Code**
3. **In Editor**

**1. Introduction**

Cold Nites is a Grid-style turn-based game. The player must strategically navigate through the level to survive the cold night, protecting the boy from all the mischievous elements of the city. And, there are always multiple ways to solve the puzzles along the way.

This design module will focus on the construction(architecture) and implementation of the Camera Manager. Camera Manager allows the designer to spawn any number of camera and seamlessly blend between them throughout the level.

This document will describe the architecture and design choices that make the Camera Manager implementation easy to use, understand and reusable for all the fellow programmers, level designers, and major stakeholders.

Below are interest points for the mentioned parties:

**Programmers** – Camera Manager along with Event Manager, will handle all the camera events.

**Level Designer** - One of the main targets of the Camera Manager is to achieve ease of level designing. It will provide nifty functionality of adding any number of cameras in the scene with just a click. And just adding the name of the camera in the tile is enough and Camera Manager will manage the blend between them.

**Project Manager (and the Team)** - All the tasks during the group meetings were assigned with everyone's and the Project Manager's agreement. This, Camera Manager design module and the code implementation will address all the concerns and will fulfill all the requirements in the game's and team's best interest.

**2. Design Goals**

The design priorities for the Camera Manager are mentioned below:

* The design should minimize the complexity of the camera switching and blending throughout the level.
* The design should allow the level designer to experiment on the fly and allow easy level creation.

**3. System Overview and Behaviour**

The TileMap actor is responsible for spawning the 'Tile' actors in a grid-based fashion in the world.

**Base Camera Actor -**

Tiles are the building block for the TileMap class. Tile inherits from an actor and is spawned by the TileMap.

Individual tiles hold a number of properties (Walkability, Destructibility, WinTile, etc.), their coordinates based on the 2D grid system, their world position, and most importantly an array of actors currently present on the tile. For this to work, all the actors need to register and unregister themselves to the Tilemap, on a particular Tile.

**Camera Manager Actor -**

TileMap will act as an intermediary manager between the world actors and the Tiles, which will provide ease of coding.

TileMap stores an array of Tile actors spawned in the world. It keeps track of all the TileCoordinates and assigns those to the individual Tiles.

With TileMap, any actor can get its coordinates on the grid, its neighbouring Tile or any desired Tile. This makes spawning and grid movement possible.

\*TileMap can also return the array of current actors on the particular Tile making the event base system possible for the game. (Alpha 2)

TileMap is also responsible to make level designing simple.

A Designer can change the TileMap Size from 5x5 to 15x10 on the fly without diving into the code.

Also, a designer can toggle TileProperties on or off from the editor itself. Tile also comes with some aesthetic customization settings.

**4. Logical View**

The logical view describes the high-level architecture for the entire game from all the core classes to high-level relations and interactions between them with a flow chart making it easy to read and understand.

Later, it will dive deep into the high-level and detailed design for the Camera Manager Module, using a UML Class Diagram.

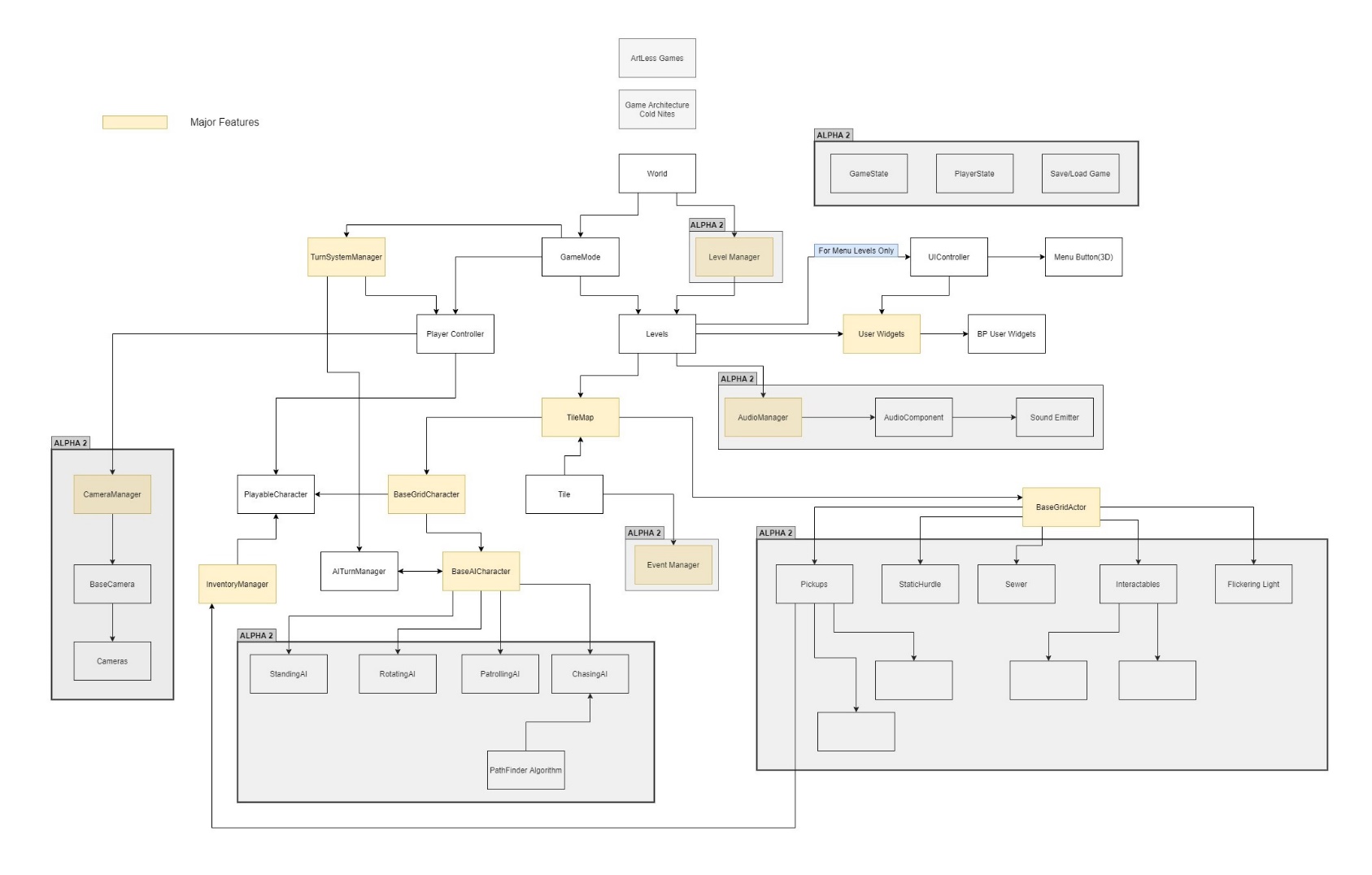
**A. High-Level Design Architecture of the Entire System**

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.

The primary features for the Alpha 2 release:

1. **AI System** -
2. **Event System** -
3. **Camera Manager** -



**B. Mid-Level Design of TileMap System**

**C. Detailed Design of TileMap System**

**5. Process View**

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

**6. Use Case View and Practice (Application)**

The use case will focus on showing the uses of TileMap at different stages in the game and will explain its application so that the TileMap module can act as a guide/reference for someone not quite familiar with the TileMap codebase.

**A. In Code:**

**B. In Editor:**