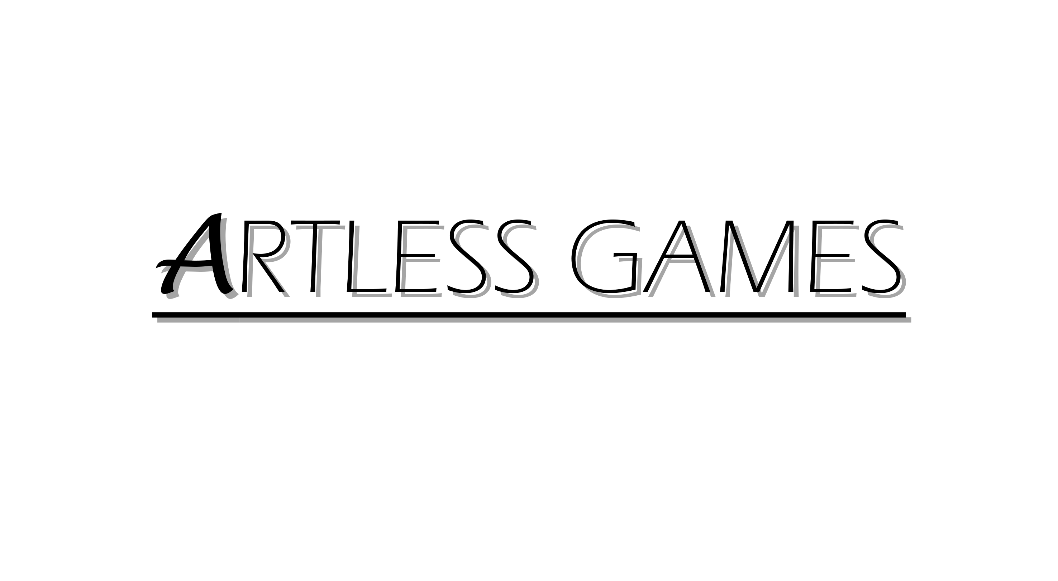
****

**COLD NITES**

Alpha-1

**Inventory Module Design Document**

Designed & Implemented by

Xiyang Chen

040995767

**Change History**

**Version:** 0.1

**Modifier:** Xiyang Chen

**Date:** 02/04/2021

**Description of Change:** Pick up function added to the game.

**Version:** 0.2

**Modifier:** Xiyang Chen

**Date:** 02/18/2021

**Description of Change:** Inventory function added to the game.

**Version:** 0.3

**Modifier:** Xiyang Chen

**Date:** 03/11/2021

**Description of Change:** Improvement of Pickup Debug and Improvement

**Table of Contents**

[1 Introduction 4](#_Toc131876418)

[2 Design Goals 5](#_Toc131876419)

[3 System Behavior 6](#_Toc131876420)

[4 Logical View 7](#_Toc131876421)

4.1 High-Level Design 7

[4.2 Mid-Level Design 9](#_Toc131876424)

[4.3 Detailed Class Design 9](#_Toc131876425)

[5 Process View 10](#_Toc131876429)

**6 Use Case View 13**

# 1. Introduction

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

This document describes the Pickup and Inventory module's architecture and design for the Cold Nites from the ArtLess Studio application being developed for GAM1528 project. The Cold Nites is a turn-based puzzle game.

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

* **Programmers** – The pickup function is essential to both gameplay and game mechanism.
* **Project Manager** – All the tasks during the group meetings were assigned with everyone's and the Project Manager's agreement. This module and the code implementation are highly related to the other modules. The PM (Project Manager) is in charge of the connection and potential influence among the programmers in the team.

# 2. Design Goals

The design priorities for the Pickup & Inventory system are:

* The design should minimize complexity and development effort.
* The design should help player grab and equip items during the game.

# 3. System Behavior

The system allows the player to pickup the useful tools and buffs in the game. The inventory list allows the player to check and equip the tools been collected.

# 4. Logical View

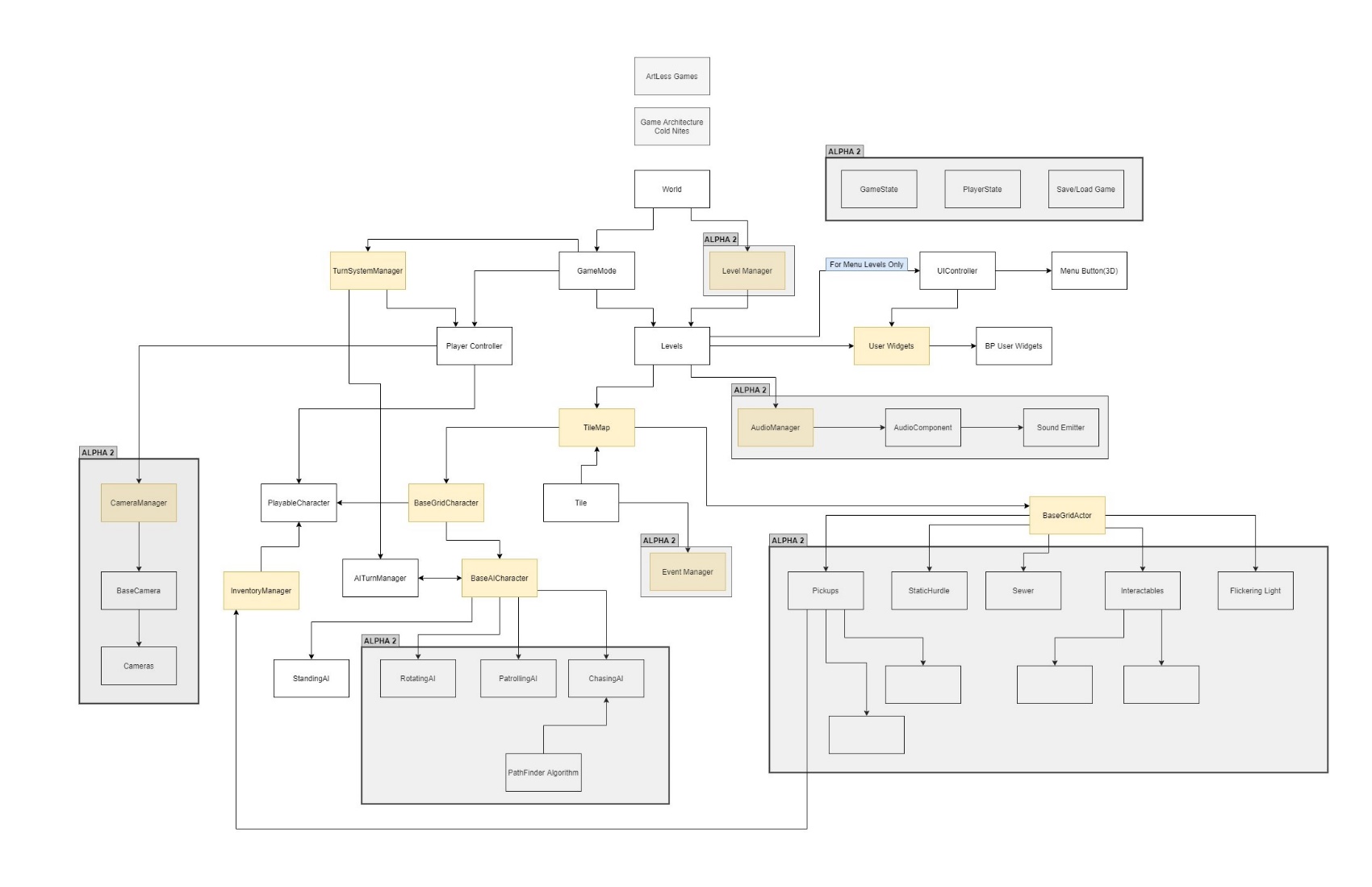
The logical view describes the high-level architecture for the entire game from all the core classes to high-level relations and interactions between them with a flow chart making it easy to read and understand.

The detail will be demonstrated in UML diagram.

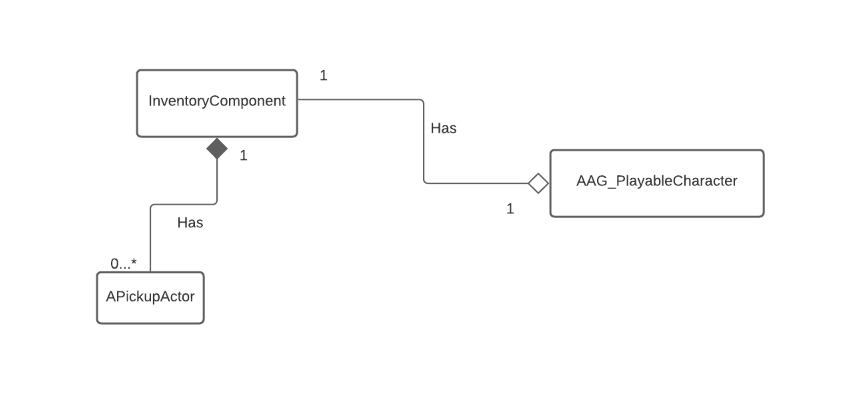
## 4.1 High-Level Design (Architecture of the Entire system)

The primary features for the Alpha 1 release:

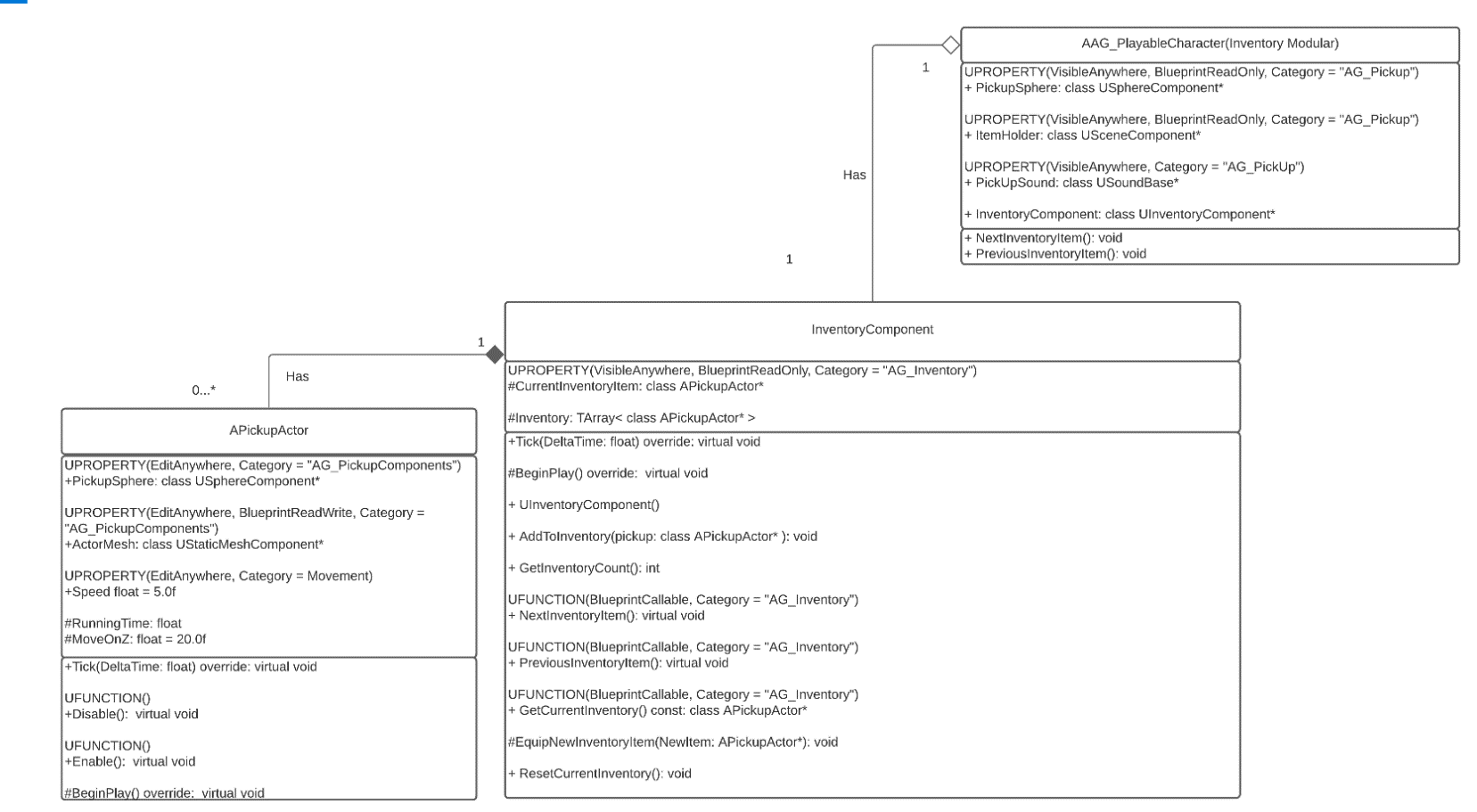
1. **TileMap** - TileMap provides the grid-based behaviour for the game and will facilitate the event system, based on the actor present on the Tiles.
2. **Turn-Based System** - This provides the turn-based aspect for the game. It is responsible for maintaining the turn order for all the world elements(actors) 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.



## 4.2 Mid-Level Design of Inventory



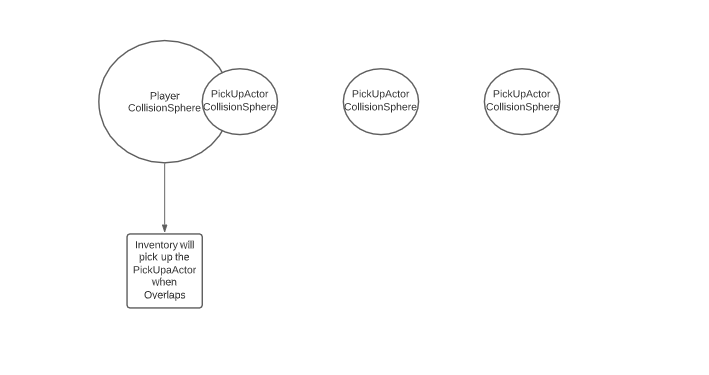
## 4.3 Detailed Class Design of Inventory

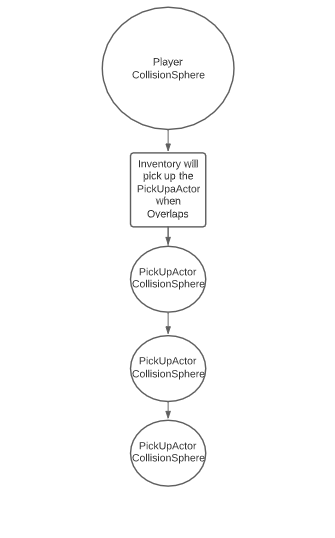


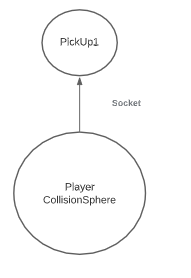
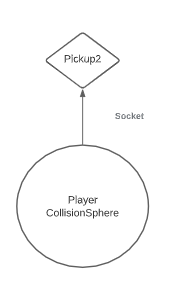
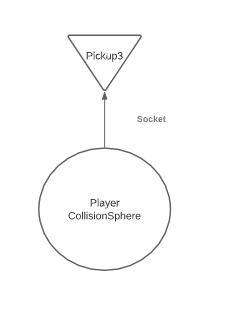
# 5. Process View

# 

When the collision sphere overlaps a pickup actor, the inventory component will pickup the actor, and stored it on the list.





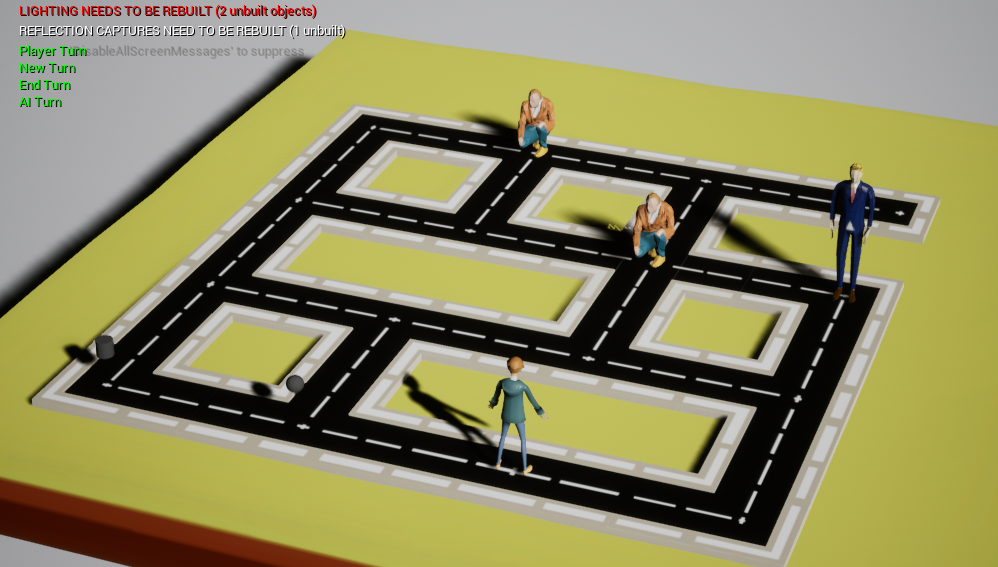
  

The inventory will equip the pickup actor by attaching it to a socket. Using PageUp and PageDown key to switch the pickups.

# 6. Use Case View

When the player overlaps with an item on the track, the player should automatically pickup the item.





Picks up when overlapping.





Inventory list will be controlled by Page-up and Page-down keys to go through the items been collected and equip the item.



