Player

Architecture/Design Document

**Table of Contents**

[**1**](#_heading=h.gjdgxs) **Introduction 3**

[**2**](#_heading=h.30j0zll) **Design Goals 4**

[**3**](#_heading=h.1fob9te) **System Behavior 4**

[**4**](#_heading=h.3znysh7) **Logical View 5**

[**4.1**](#_heading=h.2et92p0) **High-Level Design (Architecture) 5**

[**4.2**](#_heading=h.tyjcwt) **Mid-Level Design 6**

[**4.3**](#_heading=h.1t3h5sf) **Detailed Class Design 8**

[**5**](#_heading=h.4d34og8) **Process View 11**

[**6**](#_heading=h.17dp8vu) **Development View 12**

[**7**](#_heading=h.2s8eyo1) **Physical View 12**

Change History

**Version:** <1>

**Modifier:** <Nick>

**Date:** 3/6/2020

**Description of Change:** File Started

# Introduction

**Architecture and Design**

The purpose of the architecture/design document is to explain the organization of the code. A well-written architecture document will make it easier for new programmers to become familiar with the code.

The architecture/design document should identify major system components and describe their static attributes and dynamic patterns of interaction.

Software architecture and designs are typically expressed with a mix of UML models (class and sequence diagrams being the two most common) and prose. Dataflow diagrams are also helpful for understanding the interaction between components and overall flow of data through the system.

**About this Template**

This template suggests one way of documenting a software system’s architecture/design. You aren’t required to include every section in this template nor all the content in the sections you do include. However, the document you do submit should pass the following checklist:

* Are design objectives clearly stated? For example, if performance is more important than reusability, this should be made clear at the start of the design specification.
* Does the architecture partition the implementation into clearly defined subsystems or modules with well-defined interfaces?
* Does the architecture express in a clear way the main patterns of communication between subsystems and modules?
* Does the architecture satisfy the requirements?
* Is the architecture traceable to requirements?
* Any models created should either be expressed with a well-known modeling language, or if a well-known modeling language isn't used, the syntax and semantics of the symbols that are used should be defined.

This document describes the architecture and design for the Player application being developed for Tilan. The player system is how the user interacts with the game.

The purpose of this document is to describe the architecture and design of the Just Survive 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 minimize 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 specialize 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 on into the future.

# Design Goals

The goal of this system is to allow the user to interact with the game.

The design priorities for the Player system are:

* Interaction by the user with the game
* Network itself

# System Behavior

The Player system allows the user to interact with the game world. It handles input and game logic to make the game fun for the player.

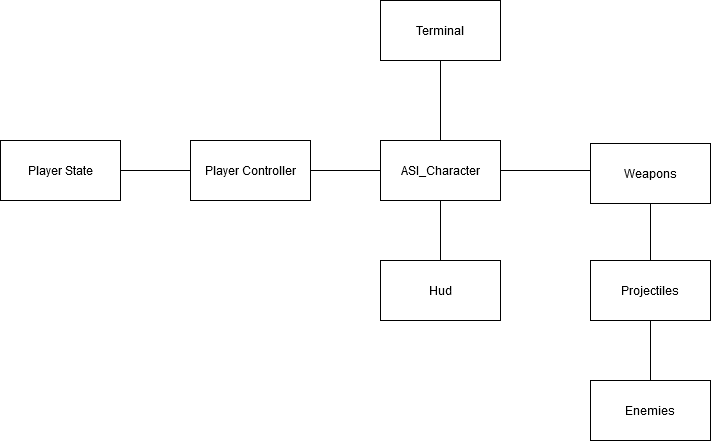
# Logical View

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 Player System

The high-level view or architecture consists of 3 major components and 5 other components it interacts with:

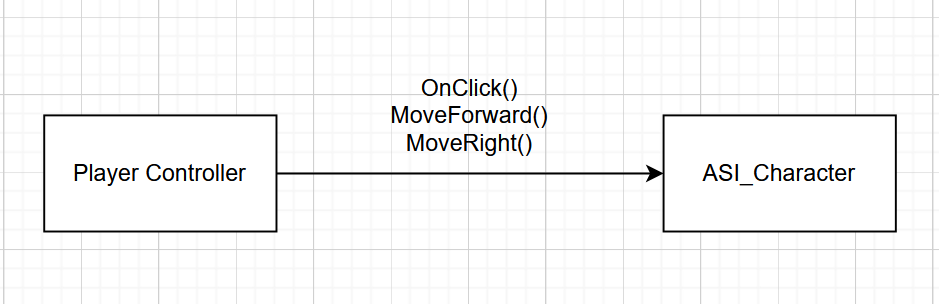
****

**System Architecture**

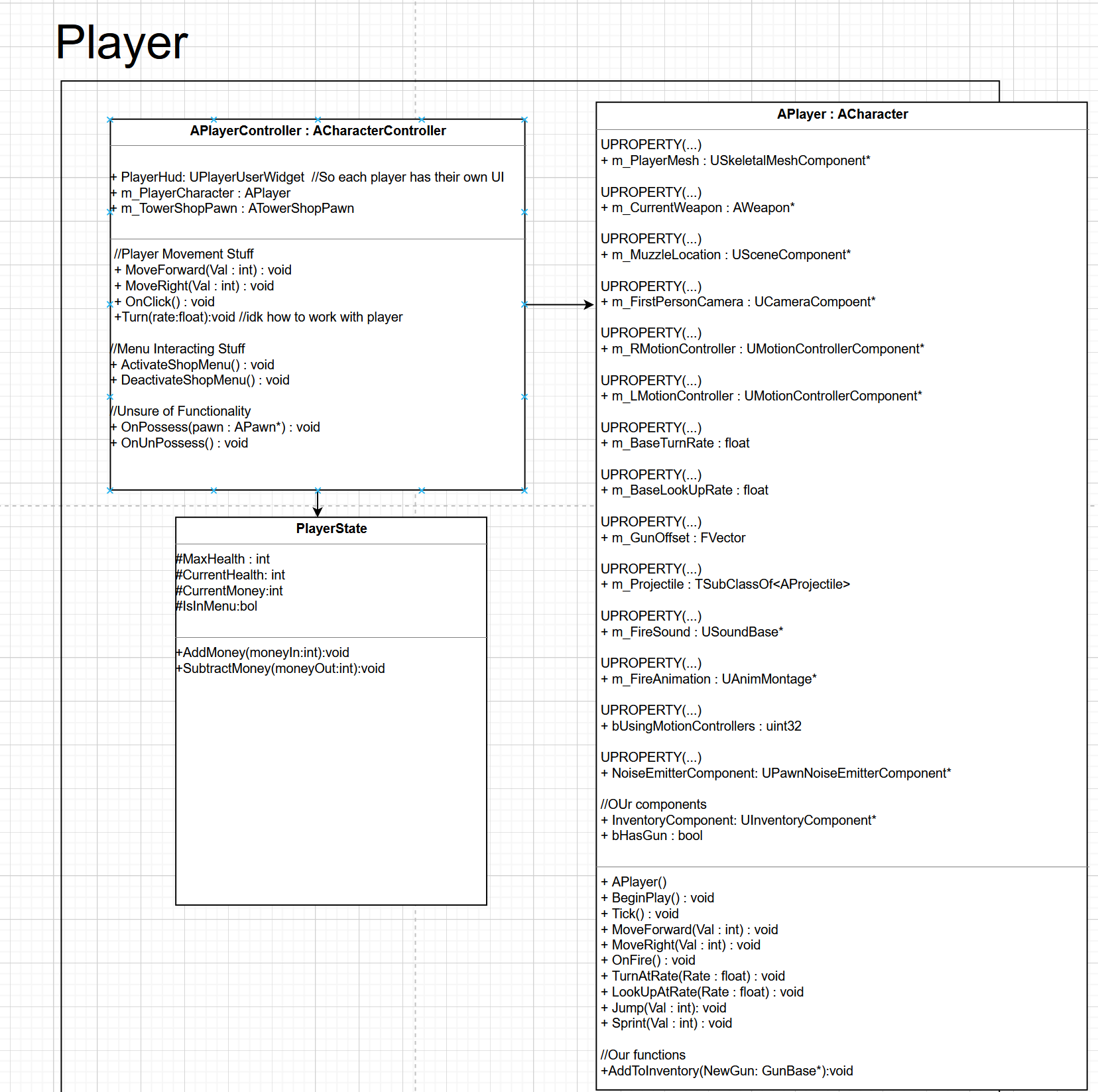
* The **Player State** has info about that player
* The **Player Controller** handles the logic for the player input
* The **ASI\_Character** is the object that interacts with the world

## Mid-Level Design of Module Player System

Player controller controls the Character

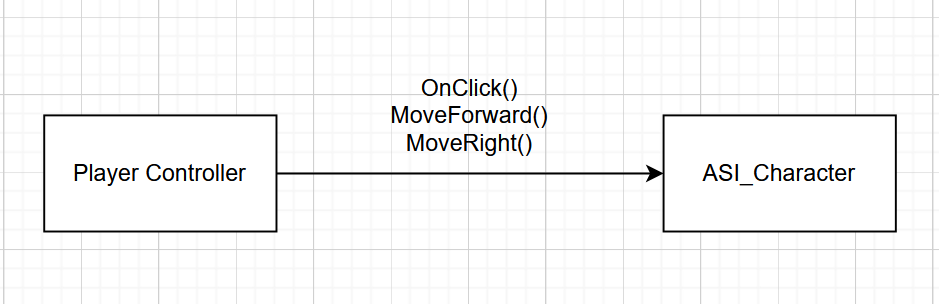


## Detailed Class Design of Module Player System



# Process View of Module Player System

The player controller controls the Character and then the Character interacts with everything in the world. As it is now, the main interaction within the system is the player controller controlling the player.



# Physical View (Applies to Multiplayer)

The player will be spawned on both the Client and the server in the different ways that the Player controller, player state and the pawns need to work.