**ShadowStalk Shade Player Module**

Architecture/Design Document

**Table of Contents**

[**1**](#_gjdgxs) **Introduction 3**

[**2**](#_30j0zll) **Design Goals 3**

[**3**](#_1fob9te) **System Behavior 4**

[**4**](#_3znysh7) **Logical View 4**

[**4.1**](#_2et92p0) **High-Level Design (Architecture) 5**

[**4.2**](#_tyjcwt) **Mid-Level Design** 6

[**4.3**](#_3dy6vkm) **Detailed Class Design** 7

[**5**](#_1t3h5sf) **Process View** 8

**6****Use Case View** 9

Change History

**Version:** 0.1

**Modifier:** Christian Young, Hamidreza Ghasemi

**Date:** 02/15/2021

**Description of Change:** Creation of the Shade class for the player to control

**Version:** 0.2

**Modifier:** Chrisitan Young

**Date:** 04/11/2021

**Description of Change:** Networked Shade, Added Trap logic

# Introduction

This document describes the architecture and design for the ShadowStalk application being developed for Particle Interactive. ShadowStalk is a heavily stylized asymmetrical co-op horror game where the players must work together to gather keys with the goal of helping lost souls escape the confines of Limbo.

The purpose of this document is to describe the architecture and design of the Shade Player Module 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 design priorities for the Shade Player Module system are:

* The design should be easily understandable and interchangeable without ever affecting movement systems.
* The design should be network ready
* The design should be highly optimized to not affect the overall flow of the game.

# System Behavior

The Shade Module inherits from the Entity Parent Module. This allows the Shade to add systems without having to worry about dealing with the Movement systems. The Shade has 2 Spotlights for eyes which allows the player to be able to see within the darkness. This Module is able to pick up and use items such as Key and Power Ups and can also be affected by traps that are placed by the Monster. This Module is Network Ready.

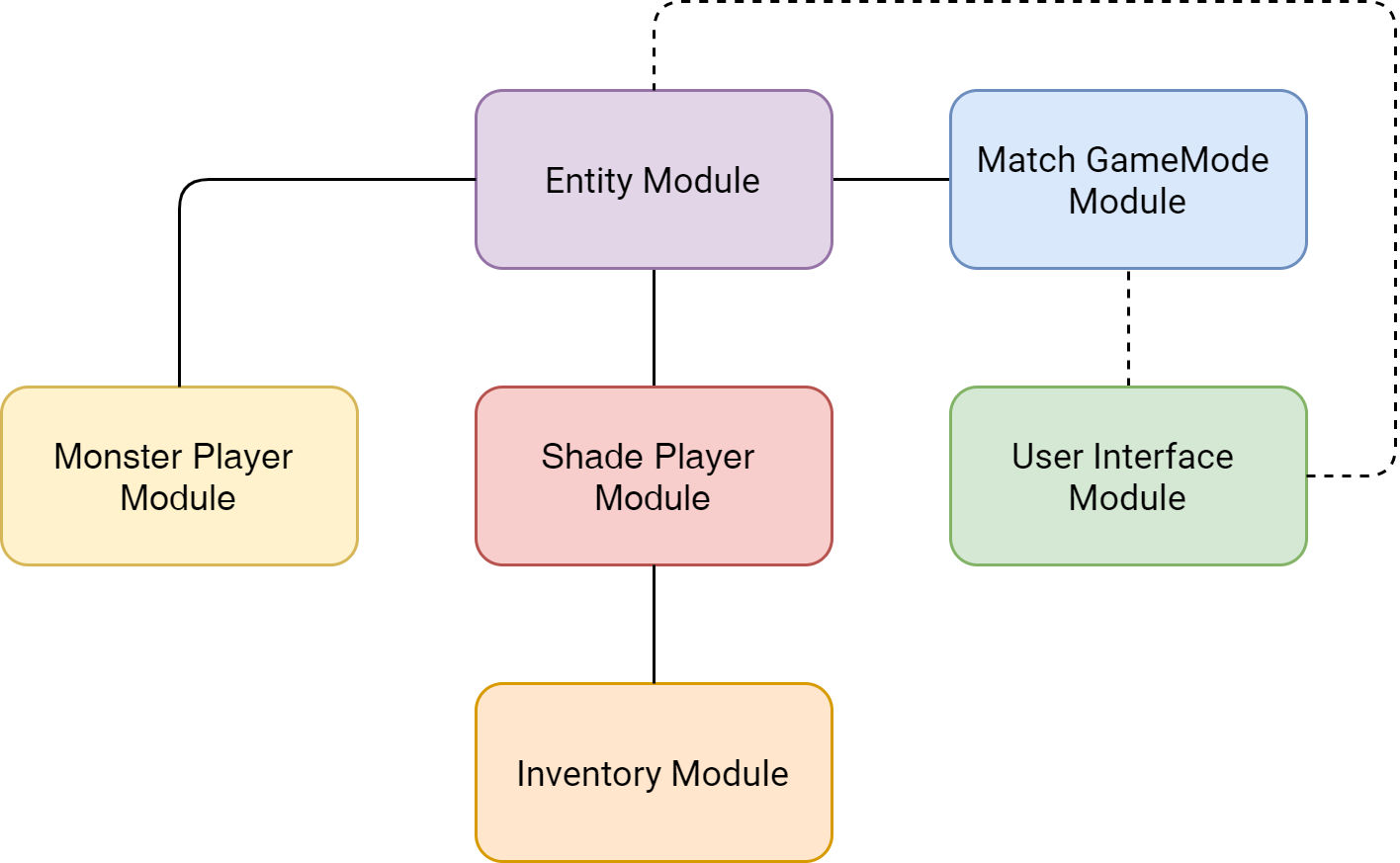
# Logical View

The Shade Module is a child of the Entity Module where it inherits all of the Entity’s Movement systems such as the Forward, Strafe, Jumping, Crouching and Mouse Look system. The Shade has two Eye components that emit light, this allows the Controlling Player to close their eyes in order to hide from the Monster Player.

The Monster Module can Interact with the Shade Module by attacking it. On the first hit the Shade will be knocked back and damaged, and on the second hit, the Shade will be knocked down for five seconds where the player will be locked out of their controls and the Monster can pick up and kill the Shade Entity.

The Monster Module can also place down a Bear Trap on the ground that is hard to notice by the Shade’s. If a Shade crosses a Trap, it will make a noise that any one who is near it can hear, along with the Monster. The trap will also lock the players movement for a short period of time.

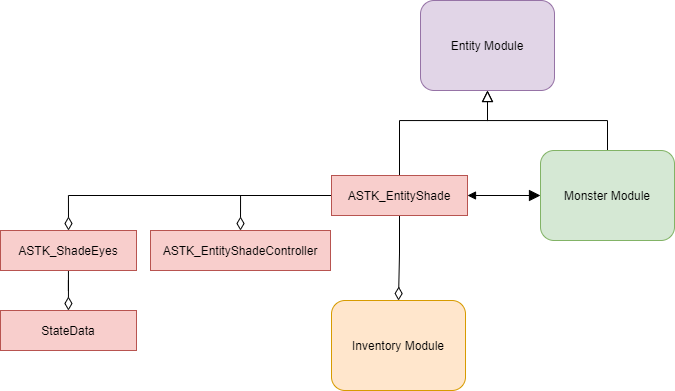
## High-Level Design (Architecture of the Entire system



The high-level view or architecture consists of **5** major components:

1. The **Entity Module** is a wrapper around Unreal Engine’s standard Pawn class. It holds the game’s custom movement component.
2. The **Shade Player Module** is the main driver of the Shade players’ character. It reacts to user input, interacts with the Inventory Module, and controls the Shade’s eyes.
3. The **Inventory Module** contains information related to what a Shade player is currently carrying (keys, items, etc.)
4. The **Match GameMode Module** is responsible for managing the game state, item spawners, pickups, and doors.
5. The **User Interface Module** is responsible for the creation of a series of interfaces and screens that allow players to interact with the game world.

## Mid-Level Design of Shade Player Module

****

**ASTK\_EntityShade:** The EntityShade class, inherits the Entity Module to allow the player to move. The EntityShade interacts with the Monster Module and is responsible for damage state. The EntityShade class also handles picking up items which then gets added to the Inventory Module.

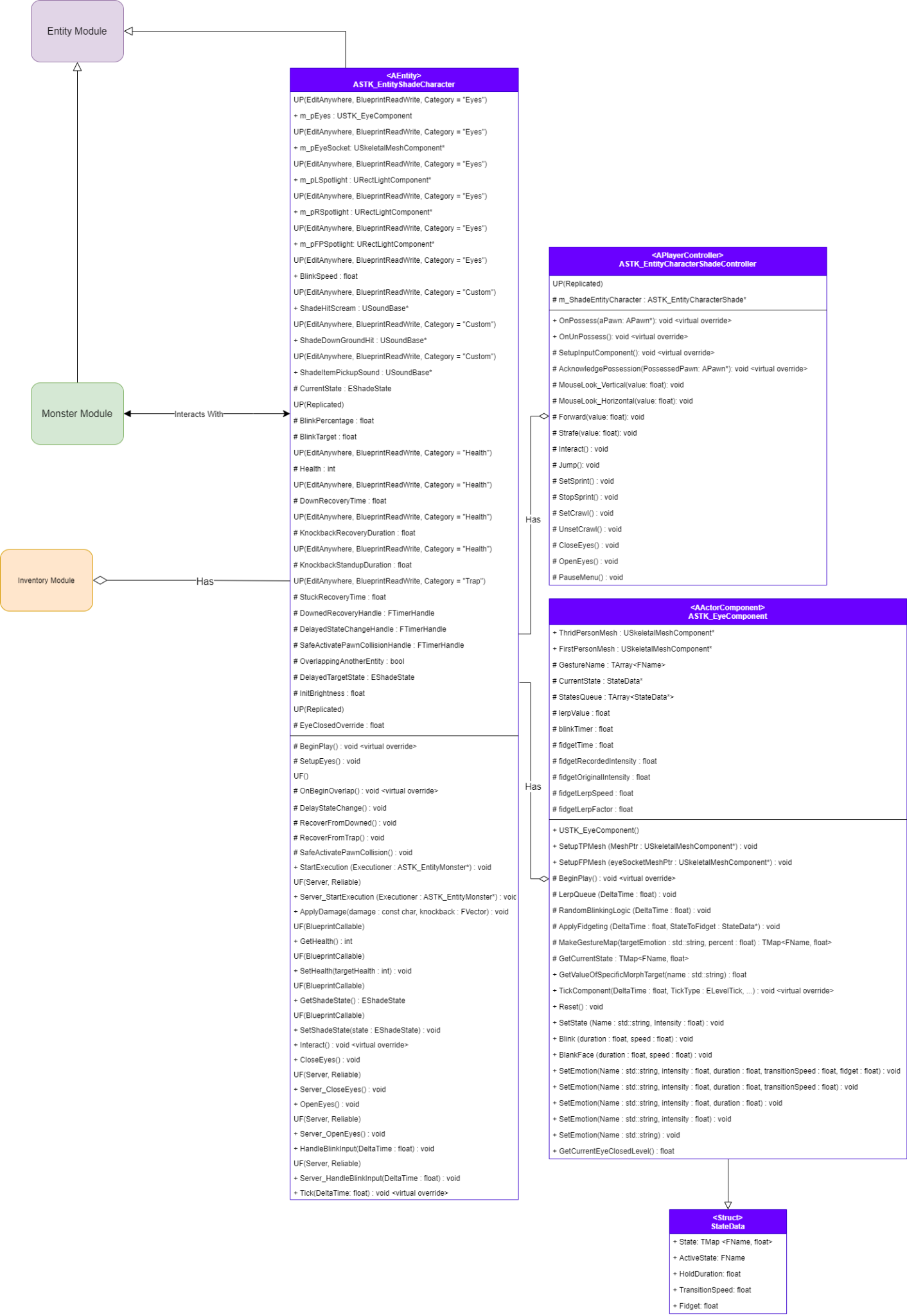
**ASTK\_EntityShadeController:** The EntityShadeController class is entirely responsible for all key inputs which then gets passed over to the EntityShade class which allows the character to move and interact with things.

**ASTK\_ShadeEyes:** The ShadeEyes class manages the EntityShade’s eyes which allows the player to be able to close their eyes to hide from the monster. This also allows the character to express different types of emotions for the face depending on what state they are in

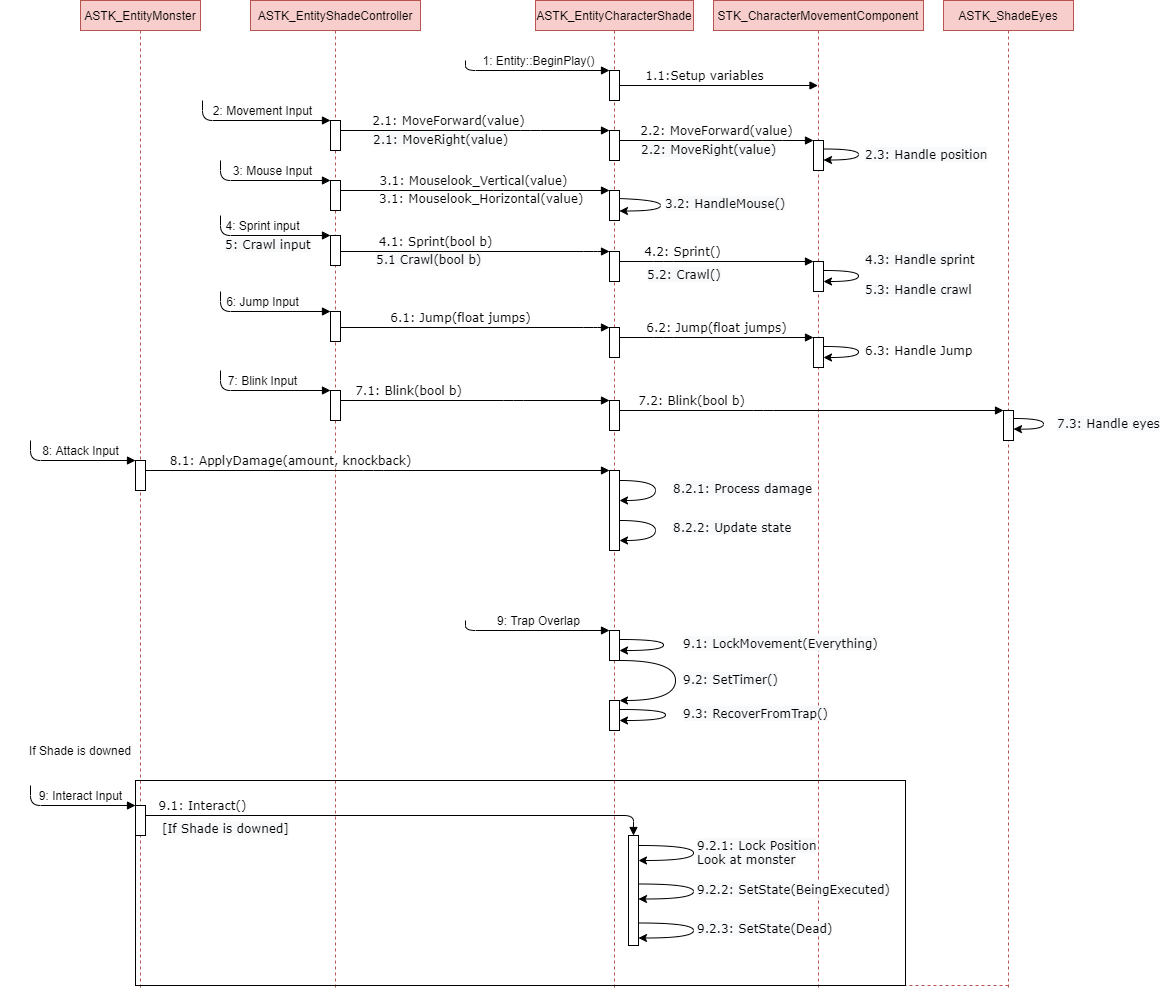
**StateData:** The StateData class holds the data for what emotions the Shade should express and when the Shade should express them.

## Detailed Class Design of Shade Player Module

Please view **UML/ShadeUML.png** for high resolution image

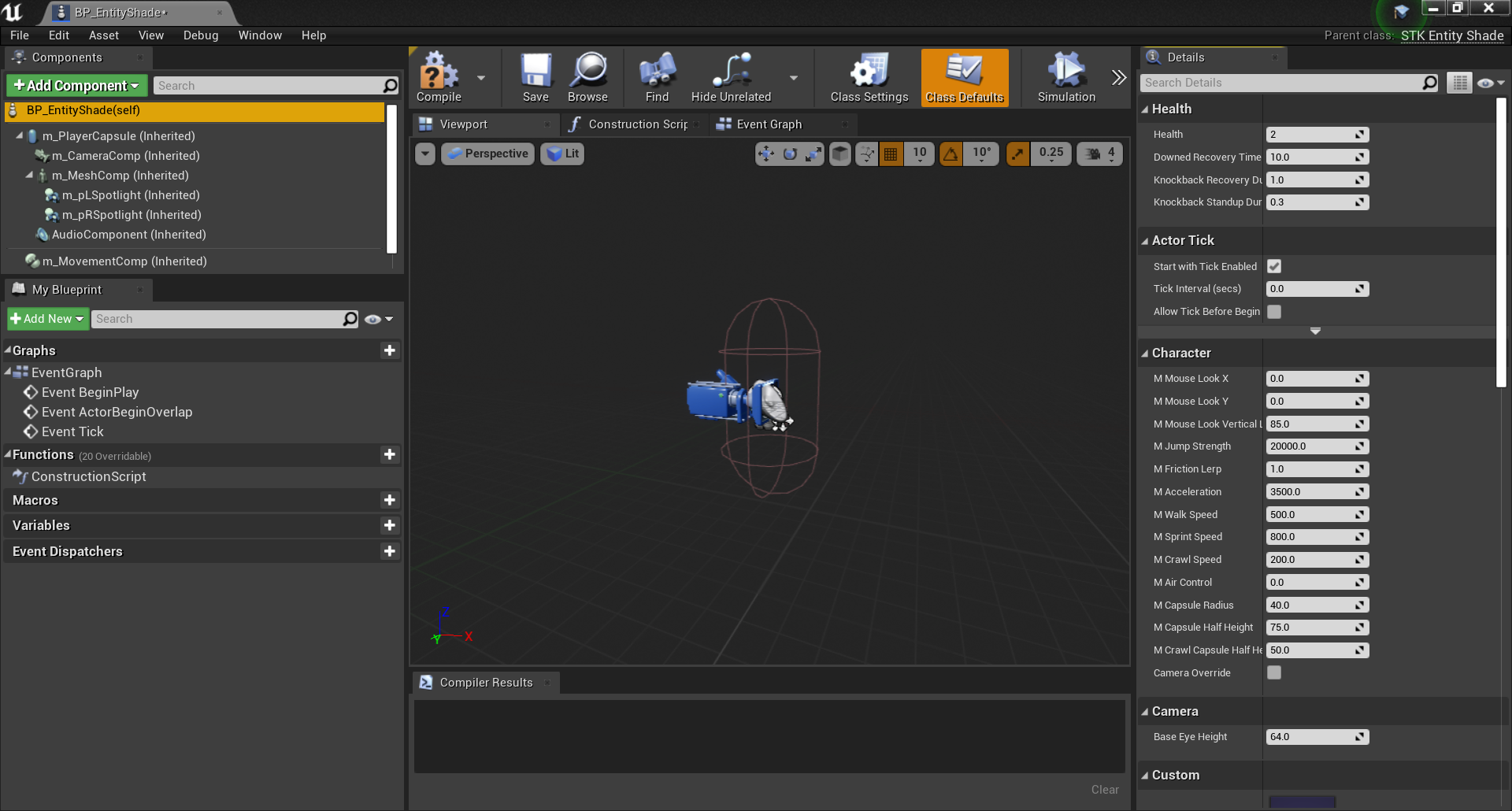


# Process View of Shade Player Module



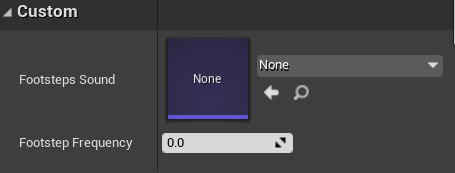
# Use Case View

**Starting from a fresh Blueprint**

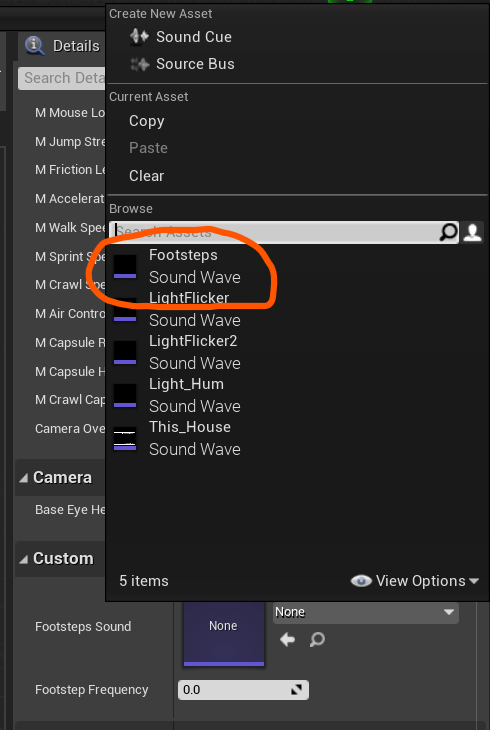
****

**Setting up Footprint Sounds:**

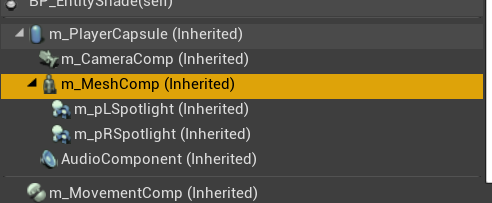
**Step 1:** Within **BP\_EntityShade(Self)** Component, Scroll down to Custom



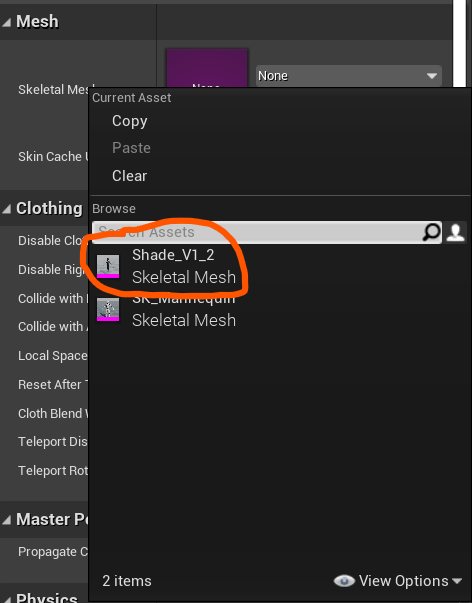
**Step 2:** Change Footstep Sound to **Footsteps Sound Wave** and Change Frequency to **1**



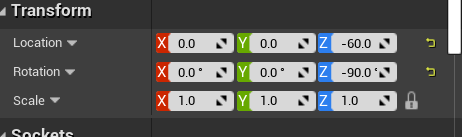
**Setting up Shade Mesh and Animations**

**Step 1:** Go to the **m\_MeshComp** Component.

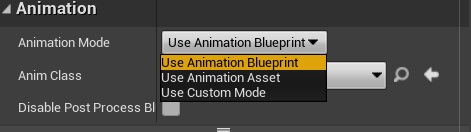
**Step 2:** Scroll down to the Mesh and Select **Shade\_V1\_2** as the Skeletal Mesh



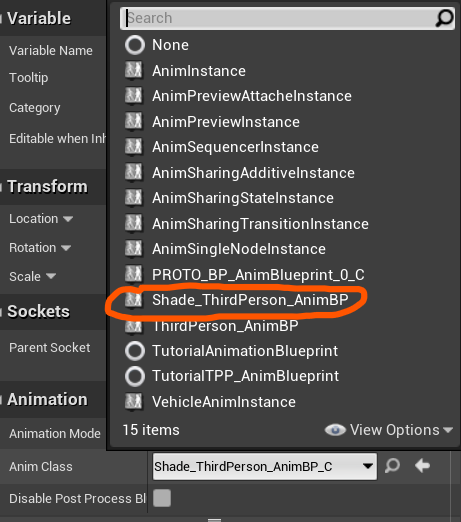
**Step 3:** Scroll to Transform and set the Z Location to **-60** and set the Z Rotation to **-90**



**Step 4:** Scroll to Animation, make sure Animation mode is set to Use Animation Blueprint

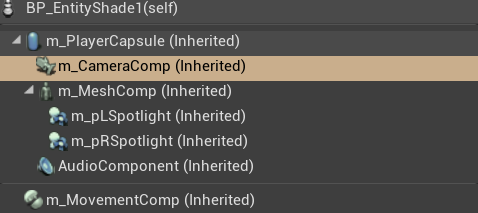


**Step 5:** Then set the Anim Class to **Shade\_ThirdPerson\_AnimBP**

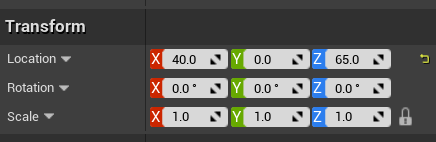
****

**Setting up the Camera**

**Step 1:** Go to the **m\_CameraComp** Component

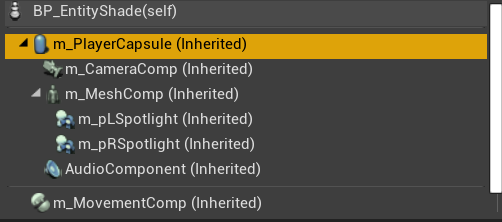


**Step 2:** Scroll to Transform and set the Location X to **40.0** and set the Location Z to **65.0**.



**Setting up the Player Capsule**

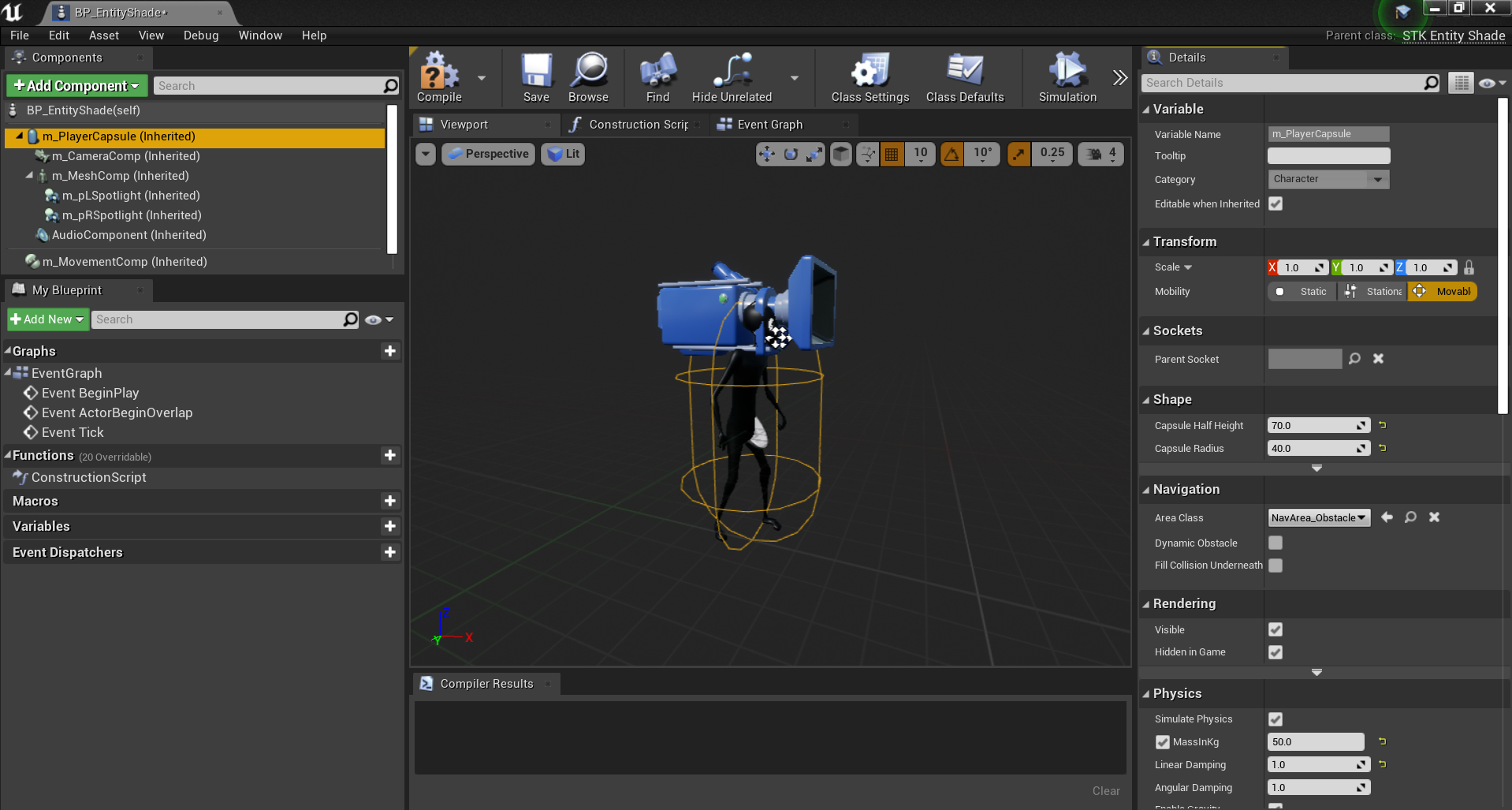
**Step 1:** Go to the **m\_PlayerCapsule** Component



**Step 2:** Scroll to Shape and set Capsule Half Height to **70.0** and set Capsule Radius to **40.0**



**This should correctly set up the Shade Blueprint**

****