The Stroke of Midnight / Event Triggers

Architecture/Design Document

**Table of Contents**

1 Introduction.. 3

2 Design Goals.. 3

3 System Behavior.. 3

4 Logical View... 4

4.1 High-Level Design (Architecture) 4

4.2 Mid-Level Design. 5

4.3 Detailed Class Design. 6

5 Process View... 7

8 Use Case View... 10

Change History

**Version:** 0.1

**Modifier:** Omer Kocar and Zoe Purcell

**Date:** 03 / 09 / 2021

**Description of Change:** Module Design Document started.

**Version:** 0.2

**Modifier:** Zoe Purcell

**Date:** 03 / 14 / 2021

**Description of Change:** Slight tweaks to Design Goals and System Behaviour

**Version:** 0.3

**Modifier:** Omer Kocar

**Date:** 03 / 14 / 2021

**Description of Change:** Document finished

# **1 Introduction**

This document describes the architecture and design for The Stroke of Midnight being developed by Memento Game Studios. The Stroke of Midnight is a single-player psychological horror game where the player breaks into an art gallery on a dare, which turns into a fight for his life as he attempts to escape.

The purpose of this document is to describe the architecture and design of the Event Trigger 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 of the game logic.
* Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain into the future.

# **2 Design Goals**

The design priorities for the Event Trigger system are:

* The design should minimize complexity and development effort.
* The design should allow Level Designers to easily place certain events and jumpscare triggers around the level without having to dive deep into the code itself.
* The design should be fairly optimized so that adding sound effects, actor spawning, and other events and effects are simple for any developers who did not design the system itself.
* The design should be modular enough for ease of use in various scenarios.

# **3 System Behavior**

The Event Trigger Module is built from a single Event Trigger parent that all other Trigger Boxes inherit from. This allows all event triggers to have needed features shared. The system should allow Designers to place certain Trigger Boxes around the level, which cause certain visual or audio effects when the Player overlaps with them. Other possible effects in this system are forcing the Player to look in a specific direction, add impulses to objects, and start level sequences. The system should also allow for these Triggers to either destroy themselves upon activation or persist for a short delay before destroying.

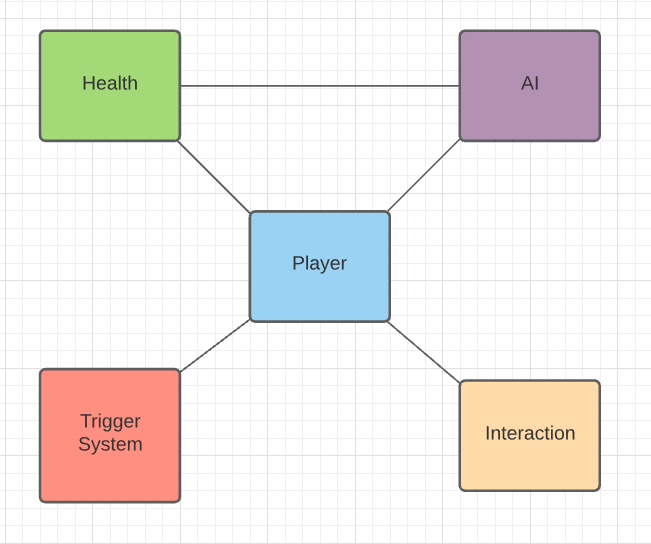
# **4 Logical View**

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.

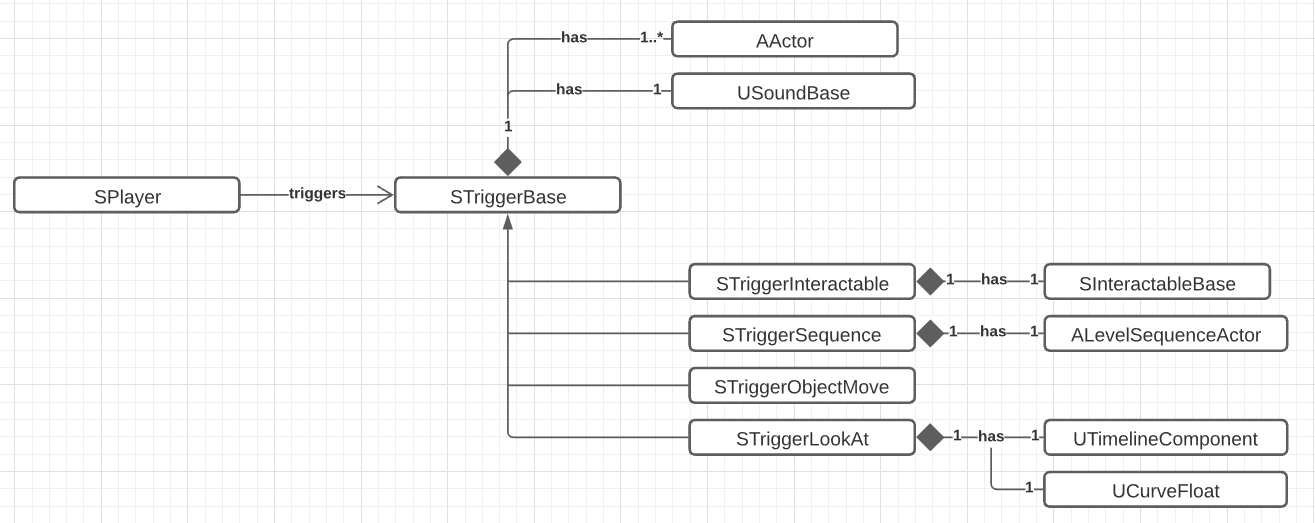
## **4.1 High-Level Design (Architecture of the Entire system)**

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



* Event Trigger Module holds all the unique event triggers and their effects on the Player character.
* Player System is the main system, which consists of a controlled character that takes in user input.
* AI System is used for all enemy behaviour.
* Health System handles how Characters take damage/receive health.
* Interaction System handles different objects that can be picked up, turned on, or interacted with by the player.

## **4.2 Mid-Level Design of the Event Trigger Module**



## 

## 

## 

## 

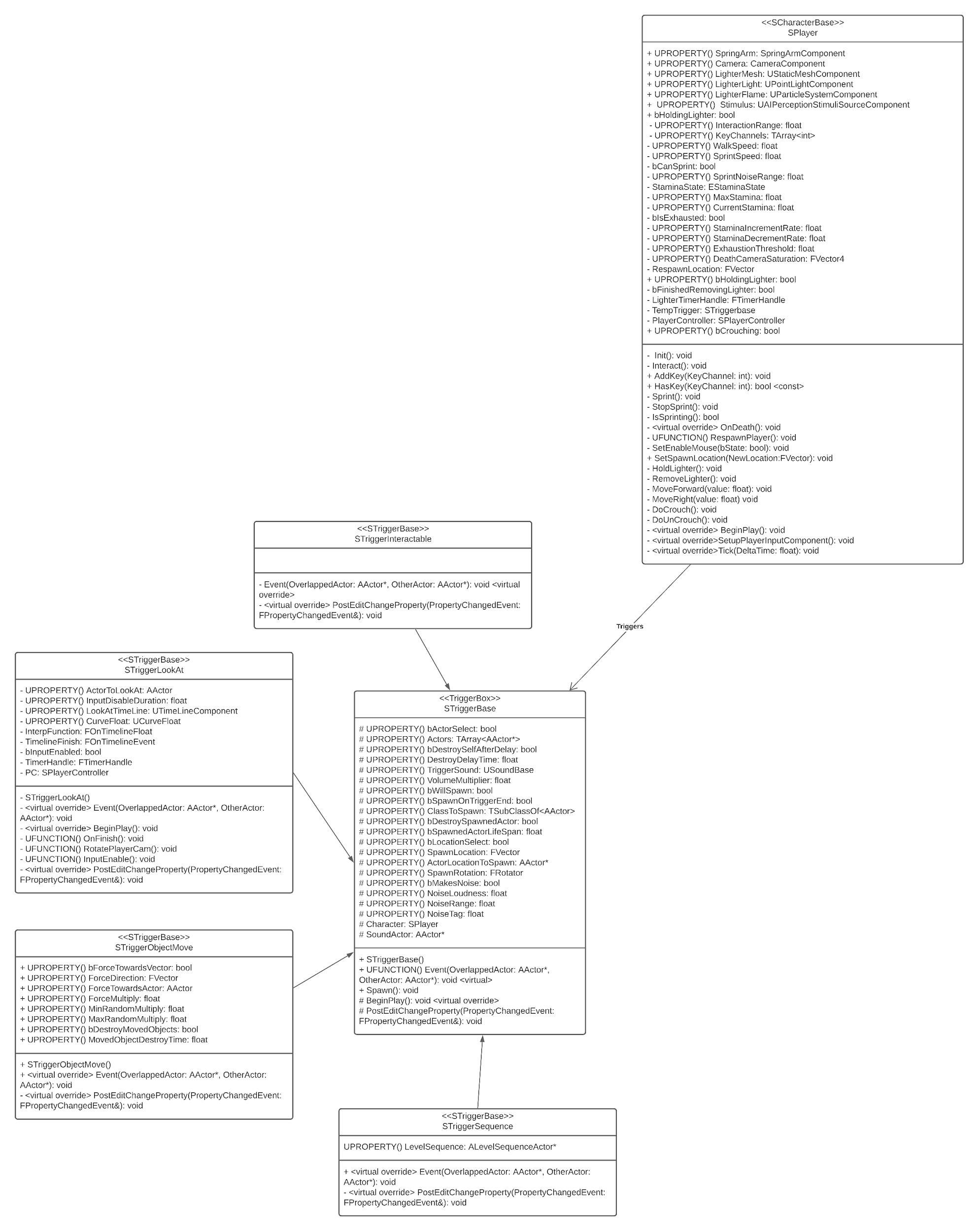
## 

## 

## 

## 

## **4.3 Detailed Class Design of the Event Trigger Module**



# 

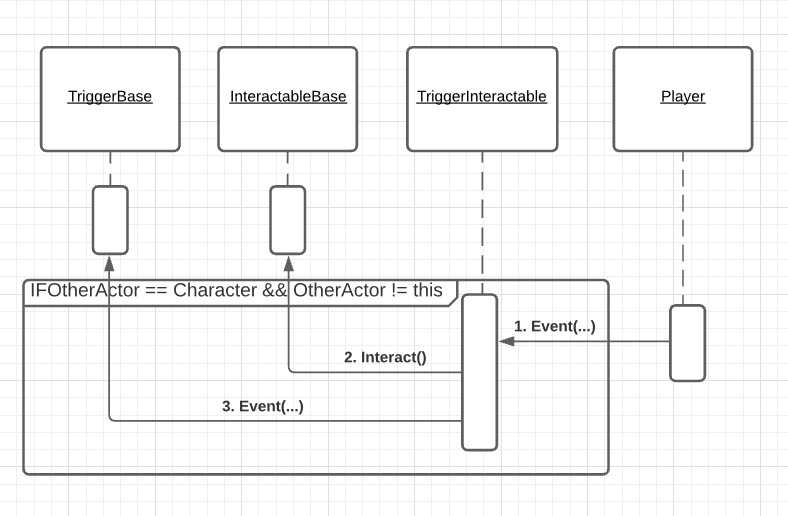
# **5 Process View of the Event Trigger Module**

**Activation**

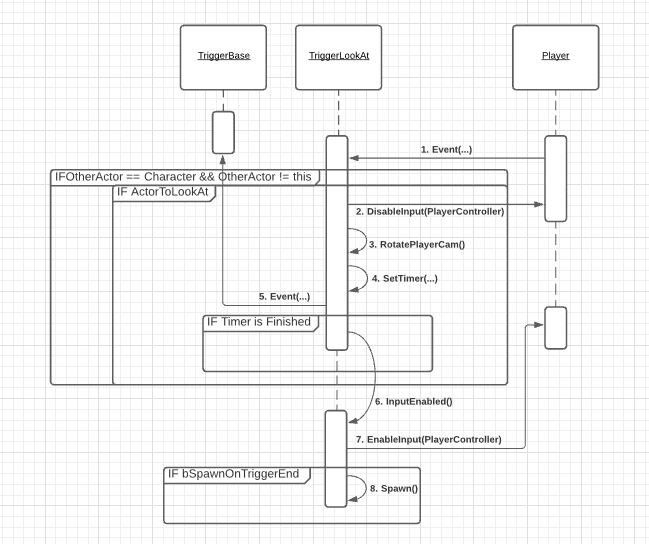
All event triggers are activated when the player overlaps with them. Upon overlap, the virtual Event function is called. Every overridden Event function calls Super::Event(), which is located in the TriggerBase class. The TriggerBase Event function can select an actor, destroy itself, spawn an object, play a sound and make noise. As a result, all the child trigger events can have these functionalities on top of each of their unique features if desired.

Any trigger box can be triggered by looking at it if LookTrace is set to Block on their CollisionComponent.

Interactable Trigger:

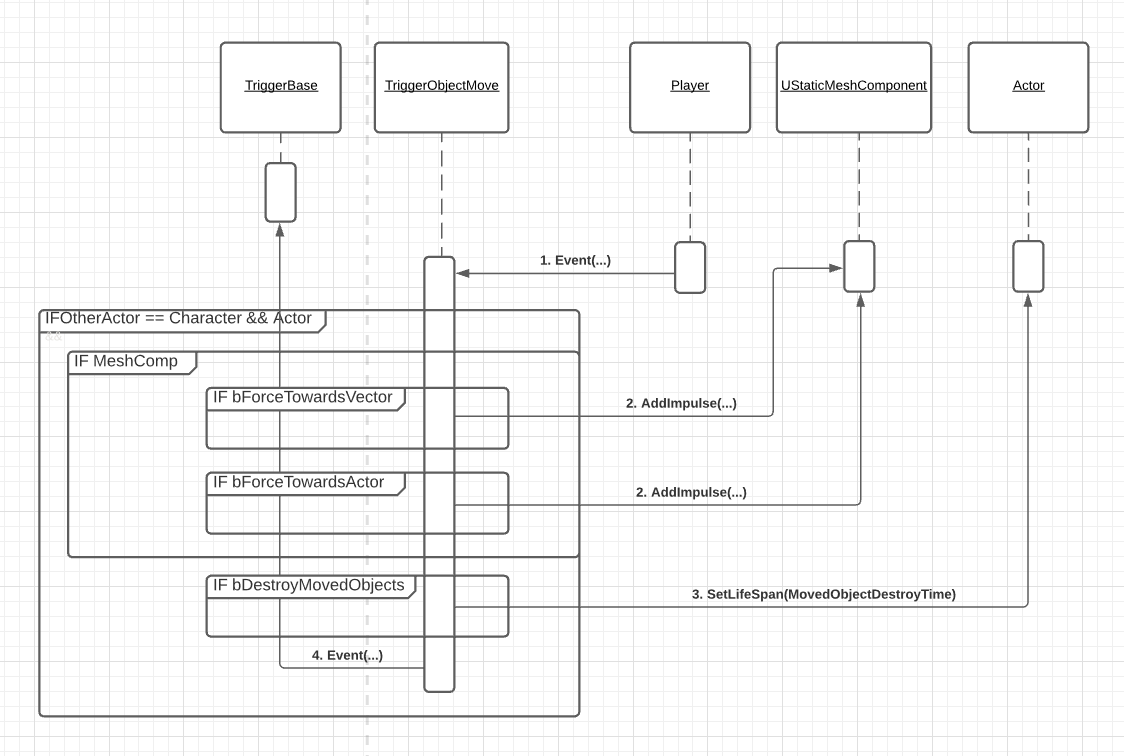


The Interactable Trigger calls the Interact function of the selected Interactable object: which is an object of any class that inherits from InteractableBase. If an actor that is not a child of InteractableBase is selected, the Trigger will skip that object and look for a valid Interactable object.

Look At Trigger

Look At Trigger is properly activated if ActorToLookAt is set. If so, the Player’s input will be disabled and the camera’s rotation timeline starts playing. The rotation is handled inside the RotatePlayerCam function.

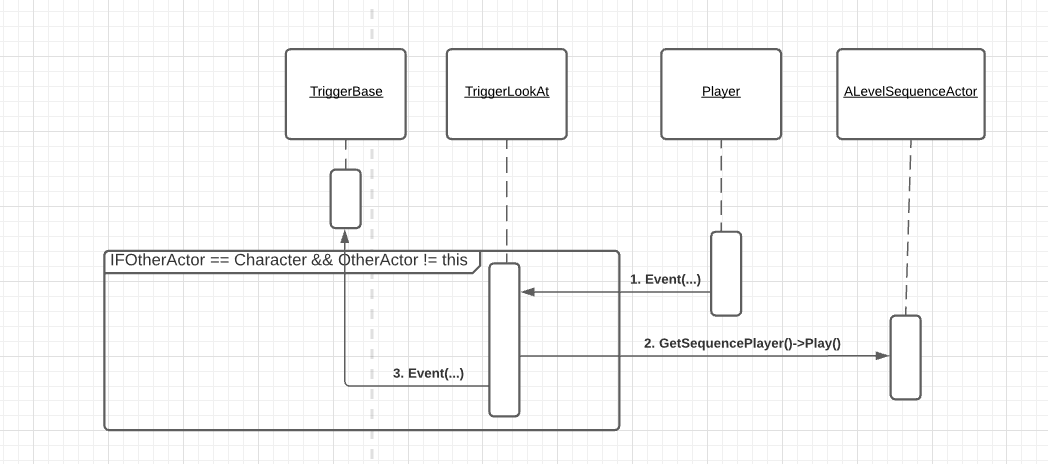
Once the Player camera rotation starts, a timer is set for an adjustable duration which calls the InputEnable function on completion. InputEnable function both re-enables player input and can also spawn an object if bSpawnOnTriggerEnd is set to true. This is useful to spawn an actor behind the player after his rotation finishes.

Object Move Trigger

Object Move Trigger adds impulse to the mesh of the selected actors if they have any. The trigger will skip any actors that do not have a mesh. If they do have one, it will activate physics on the meshes.

From there, there are two possibilities: If bForceTowardsVector is true, the actors will be forced towards a set direction vector. If bForceTowardsVector is false and ForceTowardsActor is set, the actors will be forced towards the selected actor.

Finally, if bDestroyMovedObjects is true, it is possible to set the lifespan of the actors for an adjustable duration. Any selected actors that were thrown with the impulse will destroy themselves.

Level Sequence Trigger

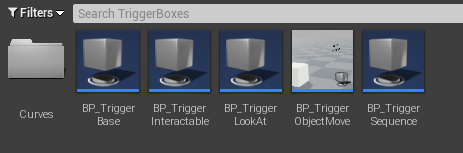
Level Sequence Trigger is the simplest trigger event. Upon trigger, it plays the pre-selected level sequence and destroys itself.

So far, the only level sequence we have is Door Shut, which slams a door.

# **6 Use Case View**

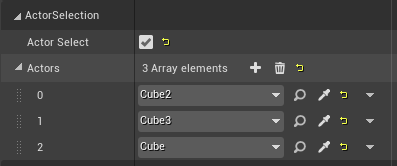
Creating an Event

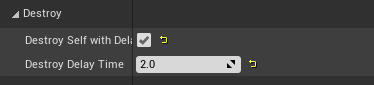
All events are preset and can be dragged from straight out of the editor into the world. All events need their default values and variables to be set after you place them into the world.

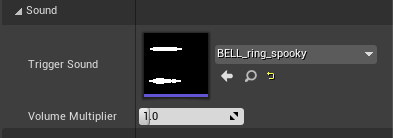


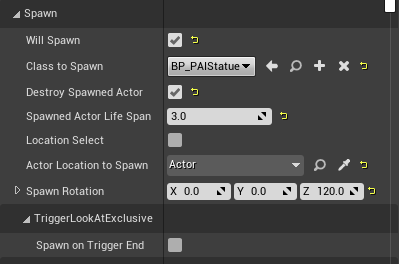
Trigger Base

Trigger Base has five different features that are also acquired by its children. The common features will be explained here:

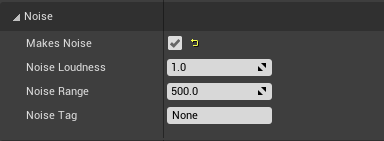
* **Actor Selection:** Trigger Events can select actors to apply their unique effect onto them if they need to. So far, ObjectMove uses this to add force to the objects and Interactable uses this to call the Interact function on the selected interactable objects.
* **Destroy:** If DestroySelfWithDelay is false, the trigger will destroy itself right after the trigger. Otherwise, it will destroy itself after DestroyDelayTime.



* **Sound:** Sound can be played on trigger with a volume multiplier.
* **Spawn:** You can spawn an actor on trigger of type ClassToSpawn. If DestroySpawnedActor is false, the spawned actor will remain in-game. Otherwise, it will be destroyed with a delay of SpawnedActorLifeSpan. If LocationSelect is true, the actor will be spawned at that location. Otherwise, it will spawn on the selected actor’s location. The rotation of the spawned actor can also be adjusted.

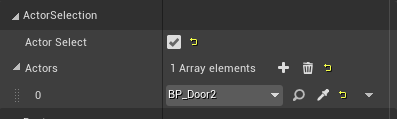


* **Noise:** Every Trigger Event can make noise if MakesNoise is true. Noise loudness and range can be adjusted. Noise Tag can also be set. So far, only AI Maynard has hearing stimuli. If he hears a “Player” tagged noise, it will trigger his Seeking state. Otherwise, his Investigate state will be triggered.



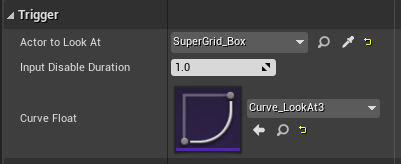
Interactable Trigger

You can set actor(s) under the ActorSelection section to trigger Interactable actors.

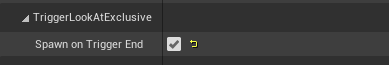


Look At Trigger

To properly make this trigger work, you need to set an actor to look at, how long the player’s input should be disabled for, and the curve float for the player camera rotation duration. There are currently three curve floats: 1, 2 and 3 seconds.

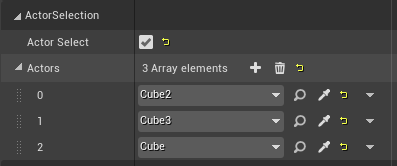


Look At Trigger, particularly, can spawn an actor at the end of the camera rotation. This can be done by setting SpawnOnTriggerEnd to true under Spawn | TriggerLookAtExclusive.



Object Move Trigger

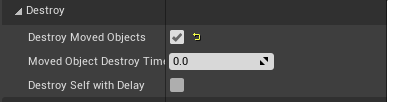
This trigger event requires actor(s) to be selected.



Once that is done, open up the Force section. If ForceTowardsVector is true, it will give you the option to select a ForceDirection vector. Otherwise, you can select ForceTowardsActor to force actors towards. Then, ForceMultiply is for how much the actors should be forced. Min and Max Random Multiply numbers are used to give each actor some randomness so that they don’t end up in the same location with the same force.

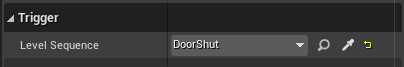


If you want to destroy the forced actors, simply set DestroyMovedObjects to true and set the destroy delay duration.



Level Sequence Trigger

To play a level sequence on trigger, simply select a level sequence from the dropdown.



Trigger Event on Sight

In order to activate any trigger on sight, all you need to do is to set LookTrace to Block on the Trigger’s CollisionComponent. Once this is done, the player can trigger the event on sight when close enough.

