# Game Event System

For the IN Game event system, we will define an event object that will be based off of a type value. The data values it contains will differ depending on that type value. In order to create an efficient and manageable system, we must keep the amount of type values at or below a certain threshold. 7 is too many 2 is too little, so 4 or 5 will do the trick.

Events are integral to the CORE systems and subsystems of the game. In order for the game to truly function, we need these systems and subsystems to communicate in an efficient way that has as low coupling as possible.

# Game Event Class

**Event types candidates:**

* Combat Event
* Entity Event
* Sequence Event
* Player death Event
* Collision Event
* Level event
* Menu event
* Game State Event
* ActionEvent

### Event Types Final List

* Sequence Event
* Collision Event
* ~~Combat Event~~
* Game State Event
* Entity Event
* ~~Action Event~~
* Menu Event

For events have an upper level **type** ~~and a lower level~~ **~~tag~~**

Types are vague and encompass a good amount, ~~whereas tags can be really specific.~~

Types are enums ~~and tags are strings~~

The event systems purpose is for objects and systems to communicate with one another in order to perform certain actions.

In order for that to happen, we will need a mediator class for events. Here’s how it’s gonna work. An object or subsystem will need to send out an event. The mediator will create this event and notify the event handler of this event. The Event handler will then handle it by giving instructions to each system that should care.

Systems HAVE-A “GameEventListener”

The listeners notify the “GamEventHandler”

Of the “GameEvent”s