

# Adding Game Logic

Now that you have a basic understanding of how the event system works, along with a base to work off of. It's time to implement some game logic!

Game logic can be implemented in two places: the Arena, or your Competition class. While we do not have a Competition class yet, it is typically recommended to leave most code inside of your Arena class, with a custom Competition class being responsible for storing values that change throughout the game (i.e. the number of blocks broken). We will get to this later.

## Using Events

Going from our previous example of a grace period, let's add some listeners to make this functional!

```
public class MyArena extends Arena {
    private static final String INFECTED_METADATA = "infected";

    @ArenaOption(name = "infection-time", description = "How long a player should be infected once hit.")
    private Duration infectionTime = Duration.ofSeconds(5);

    @ArenaEventHandler
    public void onDamageEntity(EntityDamageByEntityEvent event) {
        if (event.getDamager() instanceof Player damager && event.getEntity() instanceof Player player) {
            // Player is not infected, let's infect them :)
            if (!player.hasMetadata(INFECTED_METADATA)) {
                player.setMetadata(INFECTED_METADATA, new FixedMetadataValue(MyPlugin.getInstance(), true));
                player.sendMessage("You have been infected!");
                damager.sendMessage("You have infected " + player.getName() + "!");

                // Infect the player for the given duration
                Bukkit.getScheduler().runTaskLater(MyPlugin.getInstance(), () -> {
                    player.removeMetadata(INFECTED_METADATA, MyPlugin.getInstance());
                    player.sendMessage("You are no longer infected!");
                }, this.infectionTime.toMillis() / 50);
            }
        }
    }
}
```

```

}

@ArenaEventHandler
public void onMove(PlayerMoveEvent event) {
    if (event.getPlayer().hasMetadata(INFECTED_METADATA)) {
        event.getPlayer().sendMessage("You are infected! You cannot move!");
        event.setCancelled(true);
    }
}
}
}

```

In this example, if a player is hit by another player while in a MyArena, they will be "infected" for a short period of time. The duration for which they are infected is pulled from the **infectionTime** variable specified in your arena YML. While this is a very simple example, it demonstrates how to use the event system in BattleArena.

As a reminder, **no game specific variables** should be stored in this Arena class. As an example, this would be **wrong**:

```

// Do NOT do this - MyArena only exists ONCE, and this map will leak
// across ALL competitions of type MyArena
private final Set<UUID> infectedPlayers = new HashSet<>();

@ArenaEventHandler
public void onDamageEntity(EntityDamageByEntityEvent event) {
    if (event.getDamager() instanceof Player damager && event.getEntity() instanceof Player player) {
        // Player is not infected, let's infect them :)
        if (!this.infectedPlayers.contains(player.getUniqueId())) {
            this.infectedPlayers.add(player.getUniqueId());
            player.sendMessage("You have been infected!");
            damager.sendMessage("You have infected " + player.getName() + "!");

            // Infect the player for the given duration
            Bukkit.getScheduler().runTaskLater(MyPlugin.getInstance(), () -> {
                this.infectedPlayers.remove(player.getUniqueId());
                player.sendMessage("You are no longer infected!");
            }, this.infectionTime.toMillis() / 50);
        }
    }
}
}
}

```

```
@ArenaEventHandler
public void onMove(PlayerMoveEvent event) {
    if (this.infectedPlayers.contains(event.getPlayer().getUniqueId())) {
        event.getPlayer().sendMessage("You are infected! You cannot move!");
        event.setCancelled(true);
    }
}
```

Up next is creating a custom map and competition class, which will further extend the functionality above and demonstrate how to store per-game logic the correct way for **each** game individually.

---

Revision #4

Created 5 December 2024 18:25:13 by Redned

Updated 5 December 2024 18:41:30 by Redned