FIT2099 Project Design Rationale

James Amodeo

# Leave Affordance

An Actor is associated with a Leave Action, and that leave action must be associated with an Item, so that the Leave Action can know whether or not the Item has a Leave Affordance, and the Item can be passed up to the Actor. This same design applies to the Take Action.

# Give Affordance

An Actor is associated with an Item that it may want to Give. That Item may then have a Give Affordance, with which it can be carried by an Actor.

# Dementor

A Dementor is a subclass of HPActor. It has an HPLocation attribute in order to store its “home base” so that it can check if it has reached it’s “home base” when it is returning to it through walking in the opposite direction that it travelled out in. It also has a Direction attribute that will hold a CompassBearing so that it can have consistency in it’s movement, as it is supposed to maintain travel direction between turns.

Much of the remaining design was based of the existing design of the Death Eater class. As such, it requires AttackInformation and AttackNeighbours to find and attack a neighbouring opposing actor. In addition, it uses its inherited HPWorld attribute in order to check if it can keep moving in any given direction (which uses the CompassBearing, thus the dependency).

In addition, once the Dementor has readied a move, it much use its associated Scheduler to schedule the Move action. As a final note, dependency on Team is required for initialisation of the Dementor onto the EVIL team.

# Wand

The Wand is a subclass of HPEntity, much like the Sword. When the abstract Affordance Spell is used, it will check if the user is holding a Wand, thus a dependency exists between them. Spell itself is a subclass of HPAffordance as it should be an action that will appear on the menu alongside action like move or attack, and also as such must implement HPActionInterface. Much like Attack, Spell must depend on HPEntityInterface as it is through that interface that the Spell can interact with the Entity it to be cast upon.

Expelliarmus and AvadaKedavra are both subclasses of Spell, and will be selectable actions of an HPMagic actor. This association of the spells with an HPActor specialised for the casting of magic spells is done so that Player can trigger the HPActor’s functionality for requesting player selection of a choice between actions. This getUserDecision() will happen on the HPMagic object, displaying the spells based on which spells the Player knows (according to the Spell enumeration) after the Player’s action has been selected by the player in the case that the selection was for a spell to be cast (as in, the Spell class is the Action class that was selected).

# Potions

A Potion object will be associated with an Actor object and vice versa. The actor will be able to use their access to this Potion object in order to get information about the Potion’s effects, and can change the Player’s hitpoints attribute based on that information upon consumption of the Potion object.