Design Rationale REQ5: The inhabitants of the burial ground

Introduction

In this section, I will provide a detailed explanation of the design and implementation of the classes: Hollowsoilder, HealingVial and RefreshingFlask. Class Wandering Undead was also refactored.

Design Implementation

1. Creating Hollowsoilder class

   This class is inherited from the actor class and given the wander behavior and attack behavior, the code in the wanderbehaviour class and attackbehavior class is reused.
   This actor overwrites to only allow to attack the player, this is why the status "HOSTILE_TO_ENEMY" is checked before allowing attack action.
   This actor also overwrites to its intrinsic weapon specifications.
   This actor implements the deathrewarder interface to drop a healingVial with a 20% probability and a refreshingflask with 30% probability.

2. Creating HealingVial class

   This class is inherited from the item class and has the display char 'a'
   The actor is limited to having one healingVial in the inventory.
   If HealingVial which dosent have the status consumed(unconsumed) and the player dosent have a healingVial, when a HealingVIal is found it can be picked up. This is how the pickupaction is tailored to this item
   The dropaction make sures to removeitems that has a consumed status from the inventory.
   This item allows the players to be healed when the health is less than the maximum amount.

3. Creating RefrreshingFlask class

   This class is inherited from the item class and has the display char 'u'
   The actor is limited to having one RefreshingFlask in the inventory.
   If RefreshingFlask which dosent have the status consumed(unconsumed) and the player dosent have a RefreshingFlask, when a RefreshingFlask is found it can be picked up. This is how the pickupaction is tailored to this item
   The dropaction make sures to removeitems that has a consumed status from the inventory.
   This item allows the players to increase stamina when the stamina is less than the maximum amount.

4. Refactored WanderingUndead class

   The method "onDeath" is refactored to add the possibility of this actor dropping a HealingVial with a 20% probability.

Summary

In summary, the detailed design rationale encompasses various object-oriented principles, including encapsulation, the SRP, the Open-Closed Principle, the Interface Segregation Principle, and modular and flexible design practices. The implementation details considered in each design decision ensure code maintainability, code reusability, and overall system flexibility.