Design Rationale

Both Void and Graveyard extend the Ground class. This design choice aligns with the concept of inheritance. Inheritance allows for code reuse and promotes the DRY principle. By extending the Ground class, its properties and methods are inherited, reducing code duplication and ensuring consistency.

The Void class represents a ground type denoted by the character '+'. Its primary purpose is to interact with actors on the game map, specifically rendering them unconscious when they step on it. This choice of behavior aligns with the cohesion principle, as Void has a single, well-defined responsibility related to actor interaction. In the tick method, if an actor is present on the Void, it calls the unconscious method on the actor and checks if the actor is conscious. This design follows the principles of low coupling and high cohesion, as the class only interacts with its contained actor and does not have external dependencies.

The Graveyard class represents a ground type denoted by the character 'n'. Its primary purpose is to have a chance of spawning a WanderingUndead actor when an actor is present at that location. In the spawn method, a random percentage is calculated, and if it falls below a predefined chance threshold, a WanderingUndead actor is added to the game map.