

Design Rationale

Krishanu Donahar and Jovin Mathew

Introduction

The design rationale is split up so that each part represents a feature. It will explain how the feature will be implemented and what classes and/or methods will be called, created or modified.

Zombie Bite

New class: ZombieBiteAction

Called by: AttackBehaviour

Modifications: `getAction` in `AttackBehaviour` will be modified to call `ZombieBiteAction` if the attacker is a zombie and a chance variable is in the valid range.

Role/Responsibility of the class: to execute the action of a zombie biting a person

Result: if zombie bite is successful it will heal the zombie by calling `actor.heal()` and damage the human by calling `actor.hurt()`. The result will then be returned as a string.

PickupBehaviour - Class

Implements: Behaviour

Role: To allow NPCs (Non Player Character - characters/Actors that are not the controllable Player) to pick up items

Reasoning: Zombies are required to pick up weapons on the ground and Humans are to pick up food to recover health. So we thought to include a `PickupBehaviour` class in order to make the NPCs pick up items if standing on it and are able to do so.

Result: upon calling the `pickup` Behavior, if the NPC is standing on an item which it is able to pick up, it will call `addItemToInventory()` from superclass `Actor` and thus add it to the inventory of that NPC.

Zombie Pick Up Weapons

Add: `PickupBehaviour` to the `Zombie` behaviours list in constructor.

Result: the second behaviour will be the `PickupBehaviour` as the first is reserved for the `AttackBehaviour`. This means if there is no human to attack and it is standing on a weapon, it will add it to its inventory.

Zombie limb Weapons

Added the following classes: armWeapon, legWeapon, clubWeapon, maceWeapon which all implement WeaponItem.

Role: each of these classes represent the respective weapon. Arm and Leg weapons are made when a zombie loses a limb (see below in Zombie limbs and effects) and club and mace weapons are crafted from arm and leg respectively.

Description: We decided to make 4 different weapons and when one is crafted to it's upgraded version, the former is deleted and an object of the new weapon takes its place.

Zombie limbs and its effects

Modify:

- execute method in AttackAction; check if the target is a zombie and then remove limbs according to chance and limbs remaining. Also if the attacker is a zombie, check the number of arms and change the chance of bite, accordingly.
- playTurn in Zombie class to check if the zombie has one leg and movedLastTurn is true. If true then return doNothingAction() and set movedLastTurn to False.
- If Zombie has no legs remove the wanderBehaviour from the behaviours list attribute.
- Add the leg/arm weapon item on the spot the zombie fell by calling map.getLocation

Add:

- The integer attributes arms, legs and totalLimbs to track the respective limbs.
- Boolean attribute: movedLastTurn which tracks whether or not the Zombie moved last turn (this is used for when the zombie loses a leg)

Reasoning: We decided that the zombie limb should fall off in the same spot because we believe if a zombie lost a limb because it got hit, it would most likely fall off on the spot rather than fly a meter away or so.

Say Brains - sayBehaviour Class

Add: sayBehaviour is a Class with a list attribute of zombie sayings and a random integer attribute to see if it is

Role: To return a saying if the chance variable is met.

Modify: Zombie constructor to include sayBehaviour as the first in the behaviours list.

Reasoning: we decided that if a zombie decides on saying something, it will cost it one turn since it is not a very intelligent entity.

CraftAction

Implements: Action.

Role: to craft an arm into a club or a leg into a mace

How: The method will create the new weapon object (club or mace) and add it to the same spot by calling tick(Location, actor) to get the location. It then removes the old weapon by calling at(x, y).(groundFactory.newGround(groundChar)). Then calls gameMap.at(x, y).addItem() to add the weapon on the spot.

The action itself will be added to the list of allowable actions of the arm and leg weapon so that it will be an option for the player.

Corpse

New class: CorpseItem

Extends: PortableItem

Called by:

Roles/Responsibility: Corpse of humans turns into zombies in 10 turns.

Result: After a human dies, they leave a portable item on the location of their death by calling tick(location). The corpse has an age variable which increments by 1 each turn.

When the age reaches 10, the corpse turns into a zombie. The corpse item turns resets by calling at (x, y).(groundFactory.newGround(groundChar)). Then gameMap.at(x,y).addActor() to add a zombie to the previously restored location.

Farmer

New class: Farmer

Extends: Human

Modify: Map in the Application class to display a set number of farmers along with the other characters on map.

Called by:

Role/Responsibility of the class: Creates a new type of actor called farmers.

Crops

Modify:

- Tree - Changing the tree class into crops.
- Application - Removing all the trees from the game map.

Role/Responsibility of the class: Stores ripe and unripe crops, and it converts crops into food when the user or a farmer encounters a ripened crop.

Sowing Crops

New class: SowAction

Extends: Action

Called by: FarmBehaviour

Modify:

Roles/Responsibility of the class: To give farmers the ability to sow crops.

Result: When a farmer moves, it goes through the SowAction class which has a 33% probability of being executed. If the class is executed, the Farmer checks for valid dirt blocks by calling the checkForBlock() method around their location and sows an unripened Crop item on it.

Fertilizing Crops

New class: FertilizeAction

Extends: Action

Called by: FarmBehaviour

Modify:

Roles/Responsibility of the class: To give farmers the ability to fertilise unripened crops.

Result: The farmer checks the dirt block on their location by calling the checkForCrops() method in the FarmBehaviour class. If there is an unripe crop, the farmer calls the FertilizeAction class and reduces the time to ripen by 10 turns.

Farmer Behaviour

New class: FarmBehaviour

Extends: Behaviour

Called by: Farmer

Roles/Responsibility of the class: Lets the farmer to either sow or fertilize a crop.

Result: Generates a FertilizeAction if the farmer is on the block or else generates a SowAction which has a 33% probability of being executed.

Harvesting Crops

New class: HarvestAction

Extends: Action

Called by: Farmer/Player

Roles/Responsibility of the class: To give farmers and the player the ability to harvest crops into food.

Result: The farmer/player checks for ripened crops every turn by calling checkForRipe(). If the farmer/player does have a ripened crop around them, they could harvest the crop to turn into food by converting the ripened crop to dirt by using groundfactory() at the location and then to a food item by using additem in the previously restored location. If a player harvests a crop, the food (harvested crop) goes into the players inventory. If a farmer harvests a crop, the food is dropped by the farmer on the same location.

Food

New class: FoodItem

Extends: PortableItem

Called by: ConsumeAction

Roles/Responsibility: Contains the integer attribute health for the food item.

Consuming Food

New class: ConsumeAction

Extends: Action

Called by: ConsumeBehaviour

Roles/Responsibility of the class: Consume food to restore the player's health.

Result: While moving, humans constantly run checkForFood() to check if there is food dropped on their current location. After encountering food, healthCheck() calculates if their health is low enough to consume food or not. If true, NPC humans (every human except the player) consume the food right away. Whereas, the player has an option to either consume the food or store it in his inventory to be consumed later.