TestEnemyAttacks & TestRandomEnemies (in GarySheppardM5Tests)

TestEnemyAttacks tests for the consistent variability of the attackMonument method from each enemy class. Since each enemy class (“grunt,” “brute,” and “tactical”) attack the main monument at different damages, this test assures that different damages being passed into the attackMonument method actually deal the expected damage (even if that expected damage varies).

TestRandomEnemies tests for the random spawning of enemies over a period of time. Since the TowerDefense spawns random enemies according to a *switch* statement, this test assures that exactly one of an enemy type (“grunt,” “brute,” or “tactical”) is selected at a time by adding said type to an array given a iteration and checking for any null index (i.e. an enemy was not selected and therefore not spawned).

TestGain

TestMoneyGain checks that money gain implementation works as intended. It compared the money value after killing an enemy to the expected value.

TestEnemyKill checks that the enemy is killed off if it has taken more damage than its health. It checks that the enemy is removed from the enemy list.

TestTowerAttacks

The first test ensures that all 3 towers are unique in their attacks, comparing attack radius, damage, and cooldown/attack delay. The second test will ensure that the towers deal an appropriate amount of damage to an entity based on the tower type.

TestClearEnemyWaypoint & TestVisiblyLosesHealth

The first test checks to see if the arraylist for the enemies clears after restarting the game. This ensures that when the user will be able to restart and play the game.

The second test checks to see that the text that pops up on the screen when an enemy visibly loses health is correct depending on the type of tower it was attacked by.

TestEnemyHealthAndDamage

The first test ensures that the enemy is properly incurring damage based on the tower attacks and that the damage varies between tower's attacks.

The second test checks to see if the enemies are attacked within the proper proximity of the tower's attack radius.