Testing Artifacts

Unit Testing

Unit testing will be conducted for every object and system independently of each other. This will be done to ensure there is no issue with the algorithm or methodology used.

Entities

* Tiles
  + Have valid terrain
  + Rotations are valid
  + Can accurately check neighbors
* Farmland (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
  + Correctly update neighboring cities
* Roads (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
  + Correctly end road
* Monasteries (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
* Cities (Terrain)
  + Have correct ID
  + Valid connections
  + Correctly add/remove meeples
  + Accurate shield status
  + Correct completion
* Board
  + Adds valid tiles
  + Correct storage of tiles
  + Correct retrieval of tiles
* Meeple
  + Correct placement
  + Correct player
* Players
  + Correct score
  + Correct meeple count
* Tile stack
  + Correct set of tiles
  + Acceptable randomization
  + Removal of invalid tiles
  + Correct alert when out

Scoring System

* forfeit case:
  + If player makes illegal move or runs out of time, system should declare the other player as winner.
* endgame cases:
  + if both players scores are same, both should be declared winner
  + if one player has greater score, they should be declared winner
* score during gameplay case:
  + if no meeples, it should not score terrain even if it is completed.
  + check if 'scorability' is defined right for each terrain type
  + check that farmer meeples don't get removed
  + check that calculated Points for terrain are correct

Action System

* Test if the system is able to set one player and one AI, two AIs, or two players
* Test if the system is able to set a player name
* Test if the system is able to place a random tile in the center of the board
* Test if the system is able to place a desired tile in a legal position of the board
* Test if the system is able to score the game properly
* Test if the system is able to move seamlessly to and from each action

Acceptance Testing

This will use a testing framework such as fitness to test use cases. This helps to ensure that integration is correctly done and that the different components interact correctly.