Unit Tests

Test ID	Test Name + related requirements	Expected Result	Result	Evidence ID (Appendix)
	GameTest			
G1	CreatePlayers_AtLeastTwoPlayersAreHuman *Requirements:* N3, F1	Players 1 and 2 are returned as human when the number of players in CreatePlayer is 0	Pass	UE 1
G2	CreatePlayers_AtMostFourPlayersAreHuman *Requirements:* N3, F1	Only 4 human players are returned when the number of players is set to 5	Pass	UE 1
G3	CreatePlayers_FourPlayersAreHuman *Requirements:* N3, F1	CreatePlayers returns 4 human players	Pass	UE 1
G4	CreatePlayers_ThreePlayersAreHumanAndOneNot *Requirements:* N3, F1	Players 1, 2 and 3 are human but player 4 is not	Pass	UE 1
G5	CreatePlayers_TwoPlayersAreHumanAndTwoNot *Requirements:* N3, F1	Players 1 and 2 are human but players 3 and 4 are not	Pass	UE 1
G6	EndGame_GameEndsCorrectlyWithNoCurrentPlayerAndNoActivePlayersAndNoTurnState	When EndGame() is called: game.IsFinished == true, game.currentPlayer == null, all players are inactive and the turnState is NULL.	Pass	UE 1
G7	GetWinner_NoWinnerWhenAPlayerHasLandmarkAndAnotherHasUnits	game.GetWinner() == null when playerA owns a landmark sector and playerB owns a unit.	Pass	UE 1
G8	GetWinner_NoWinnerWhenMultiplePlayersOwningLandmarks *Tests to see if a winner is not detected when there is more than one player own landmarks. First sets up a new game and then clears all sectors and units. Two players are then given one landmark each.*	game.GetWinner() == null when playerA owns a landmark and playerB owns a landmark.	Pass	UE 1
G9	GetWinner_NoWinnerWhenMultiplePlayersWithUnits *Tests to see if a winner is not detected when there is more than one player which has units. First sets up a new game and then clears all sectors and units. Two players are then given one unit each.*	game.GetWinner() == null when two players own a unit.	Pass	UE 1
G10	GetWinner_OnePlayerWithLandmarksAndUnitsWins *Tests that GetWinner returns the correct*	game.GetWinner() != null when PlayerA is the only player that owns a landmark	Pass	UE 1

	player when that player is the only one with landmarks and units.	and a unit.		
G11	InitializeMap_OneLandmarkAllocatedWithUnitPerPlayer *Requirement:* N6	Each player has 1 sector, a landmark and owns a single unit, which is positioned in the owned sector Asserts that none of the other sectors change ownership/remain neutral	Pass	UE 1
G12	NextPlayer_CurrentPlayerChangesToNextPlayerEachTime	If the current player is playerA, the next is playerB. If the current player is playerB, the next is playerC. If the current player is playerC, the next is playerD. If the current player is playerD, the next is playerA.	Pass	UE 1
G13	NextPlayer_EliminatedPlayersAreSkipped *Requirement:* N3	If players A & B are eliminated (have no landmark sectors & units), they will be skipped and the current player will be set to playerC.	Pass	UE 1
G14	NextTurnState_TurnStateProgressesCorrectly *Requirement:* N3	Asserts that: NextTurnState() → Move2 when turnstate == move1. NextTurnState() → EndOfTurn when turnstate == move2. NextTurnState() → move1 when turnState == EndOfTurn. NextTurnState() → NULL when turnState == NULL.	Pass	UE 1
G15	NoUnitSelected_ReturnsFalseWhenUnitIsSelected *Requirement:* F5	game.NoUnitSelected() → True when no units are selected. game.NoUnitSelected() → False when a unit's selected attribute is set true.	Pass	UE 1
G16	SpawnPVC_spawnsAfter10Turns *Sets up a new game and calls game.NextPlayer 9 times to simulate passing 9 turns. The number of sectors containing a PVC are counted.* *Requirement:* F2	The number of sectors containing a PVC unit equals 1.	Pass	UE 1
G17	SpawnPVC_doesNotSpawnAfterFirstTurn *Sets up a new game then calls game.NextPlayer once and counts the number of PVC sectors.* *Requirement:* F2	The number of sectors containing a PVC unit is 0.	Pass	UE 1
G18	SpawnPVC_doesNotSpawnMultiplePVCs *Sets up a new game and changes turn 10 times. The number of PVC sectors are then*	The number of sectors containing a PVC unit remains at 1.	Pass	UE 1

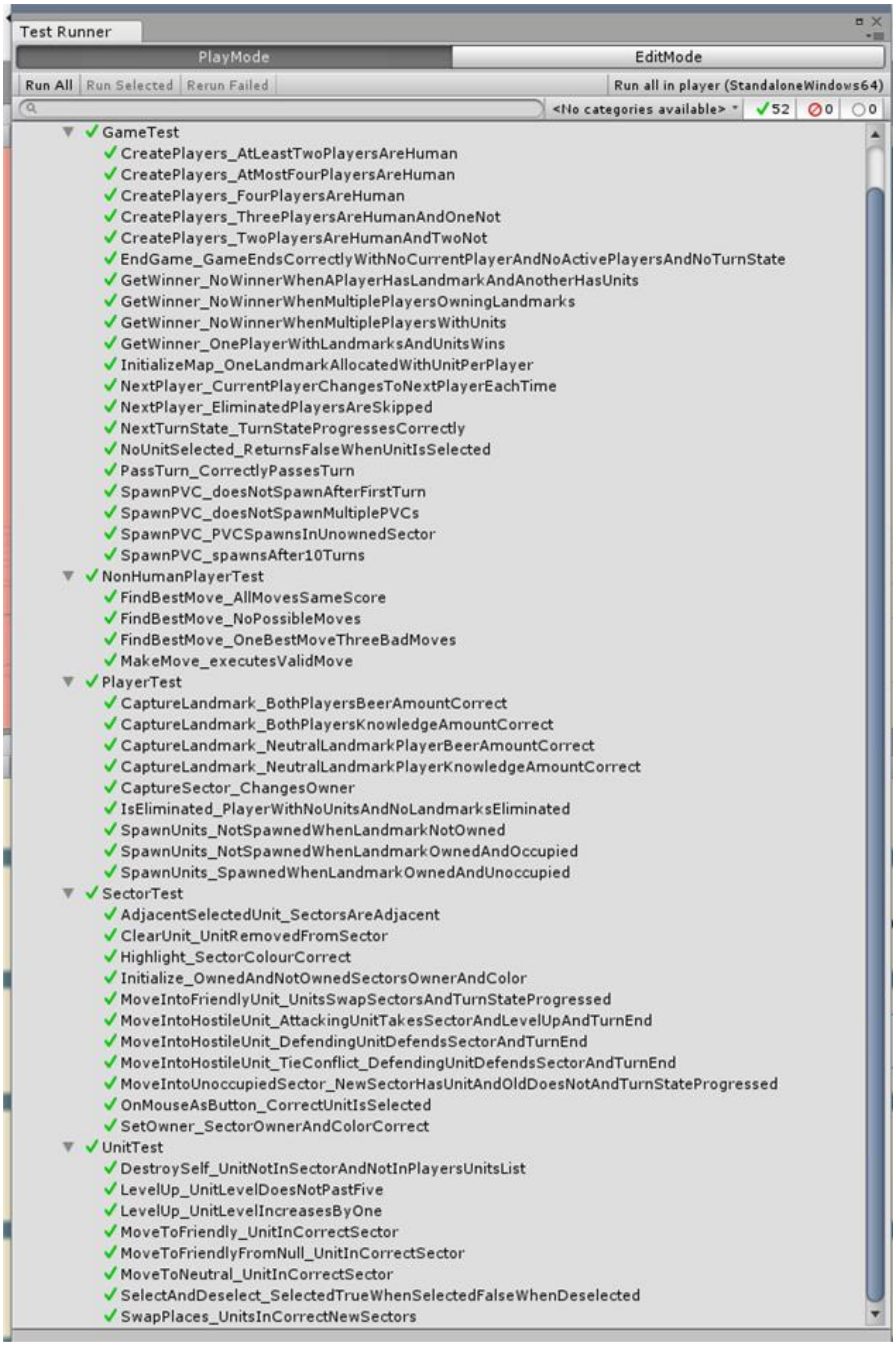
	counted. *Requirements:* F2			
G19	SpawnPVC_PVCSpawnsInUnownedSector *Sets up a new game and makes a PVC spawn by calling game.NextPlayer 9 times. The sector containing the PVC is then found.* *Requirements:* F2	When the PVC spawns, the owner of the sector containing the PVC unit is null.	Pass	UE 1
G20	PassTurn_CorrectlyPassesTurn *Tests that the PassTurn method correctly ends the current player's turn and passes it to the next player. To do this the test sets up a new game and then calls PassTurn, followed by UpdateAccessible to make the system detects the change.*	The current player changes from players[0] to players[1].	Pass	UE 1
	PlayerTest			
P1	CaptureLandmark_BothPlayersBeerAmountCorrect *Requirements:* N6, F5	When playerA captures a beer landmark sector: The owner is the same as playerA. The sector is added to playerA's ownedSector list. playerA's beer value increases by the landmark.GetAmount(), and the previous owner's beer value decreases by landmark.GetAmount().	Pass	UE 1
P2	CaptureLandmark_BothPlayersKnowledgeAmountCorrect *Requirements:* N6, F5	When playerA captures a knowledge landmark sector: The owner is the same as playerA. The sector is added to playerA's ownedSector list. playerA's knowledge value increases by the landmark.GetAmount(), and the previous owner's knowledge value decreases by landmark.GetAmount().	Pass	UE 1
P3	CaptureLandmark_NeutralLandmarkPlayerBeerAmountCorrect *Requirements:* N6, F5	When playerA captures a neutral beer landmark, the difference between their new beer score and old beer score should equal landmark.GetAmount().	Pass	UE 1
P4	CaptureLandmark_NeutralLandmarkPlayerKnowledgeAmountCorrect *Requirements:* N6, F5	When playerA captures a neutral knowledge landmark, the difference between their new and old knowledge values should equal landmark.GetAmount().	Pass	UE 1

P5	CaptureSector_ChangesOwner *Requirements:* N9, F5	When a player captures a sector, the sectors owner changes to the player. The sector is added to the player's ownedSectors list. The sector is removed from the previous owner's ownedSectors list.	Pass	UE 1
P6	IsEliminated_PlayerWithNoUnitsAndNoLandmarksEliminated	playerA.IsEliminated() == false when playerA has units. playerA.IsEliminated() == false when playerA owns landmarks. playerA.IsEliminated() == true when playerA doesn't own any units and landmarks.	Pass	UE 1
P7	SpawnUnits_NotSpawnedWhenLandmarkOwnedAndOccupied *Requirement:* F6	If a landmark sector owned by a player contains a unit, spawnUnits() does not place another unit in the landmark sector.	Pass	UE 1
P8	SpawnUnits_NotSpawnedWhenLandmarkNotOwned *Requirement:* F6	If a landmark sector is not owned by playerA, playerA.spawnUnits() does not place a unit in the landmark sector. (landmarkedSector.GetUnit is null.)	Pass	UE 1
P9	SpawnUnits_SpawnedWhenLandmarkOwnedAndUnoccupied *Requirement:* F6	If a landmark is owned by playerA and doesn't contain a unit, playerA.SpawnUnits() will spawn a unit into the landmark sector. (landmarkSector.GetUnit() is in playerA.units).	Pass	UE 1
	NonHumanPlayerTest			
NP1	MakeMove_executesValidMove *Requirements:* N3, F1	NonHumanPlayer gains 2 sectors and the unit has been moved from its origin	Pass	UE 1
NP2	FindBestMove_OneBestMoveThreeBadMoves *Requirements:* N3, F1	FindBestMove returns the best move and the sector out of 4 possible moves	Pass	UE 1
NP3	FindBestMove_AllMovesSameScore *Requirements:* N3, F1	FindBestMove returns a valid move and sector	Pass	UE 1
NP4	FindBestMove_NoPossibleMoves *Requirements:* N3, F1	FindBestMove is null and the score of the best move is -1	Pass	UE 1

	SectorTest			
S1	AdjacentSelectedUnit_SectorsAreAdjacent *Requirement:* F5	If 2 sectors, sectorA & sectorB, are adjacent, sectorA appears in sectorB.GetAdjacentSectors and sectorB is in sectorA.GetAdjacentSectors. If sectorA contains a unit which is unselected, sectorB.AdjacentSelectedUnit returns null. If sectorA contains a unit which is selected, sectorB.AdjacentSelectedUnit is not null.	Pass	UE 1
S2	ClearUnit_UnitRemovedFromSector	If a sector does not contain a unit and sector.ClearUnit is called, sector.GetUnit() is null. If a sector contains a unit and sector.ClearUnit is called, sector.GetUnit() should also return null.	Pass	UE 1
S3	Highlight_SectorColourCorrect *Requirement:* N8	When sector.ApplyHighlight(x) is called, the materials color should be equal to the base colour + (x,x,x). When sector.RevertHighlight(x) is called, the materials color should return to the base colour.	Pass	UE 1
S4	Initialize_OwnedAndNotOwnedSectorsOwnerAndColor *Requirement:* N8	Unowned sectors owner is null, the material's colour is grey, GetUnit() is null. For landmark sectors, GetLandmark() is not null, but for normal sectors GetLandmark() returns null.	Pass	UE 1
S5	MoveIntoFriendlyUnit_UnitsSwapSectorsAndTurnStateProgressed *Requirement:* F5	When a player moves a unit into a sector already containing one of their units, the units switch sectors.	Pass	UE 1
S6	MoveIntoHostileUnit_AttackingUnitTakesSectorAndLevelUpAndTurnEnd *Requirements:* N9, F5	When a player attacks another unit and wins, the unit is moved to the defending sector and its level is increased by 1. The turnstate then changes to EndOfTurn.	Pass	UE 1
S7	MoveIntoHostileUnit_DefendingUnitDefendsSectorAndTurnEnd	When a player attacks another unit and loses, the unit is destroyed and the	Pass	UE 1

	Requirements: N9, F5	defending unit's level does not change. The turnstate then changes to EndOfTurn.		
S8	MoveIntoHostileUnit_TieConflict_DefendingUnitDefendsSectorAndTurnEnd *Requirements:* N9, F5	Attacking unit is destroyed, defending unit does not level up and turn state progresses to end of turn	Pass	UE 1
S9	MoveIntoUnoccupiedSector_NewSectorHasUnitAndOldDoesNotAndTurnStateProgressed *Requirement:* F5	When a unit is moved into an unoccupied sector, the unit moves out of one sector into the other sector and turn state progresses to move2.	Pass	UE 1
S10	OnMouseAsButton_CorrectUnitIsSelected *Requirement:* N8	Tests that a unit is only selected if it is owned by the current player and there isn't already a selected unit.	Pass	UE 1
S11	SetOwner_SectorOwnerAndColorCorrect *Requirement:* N8	Colour of the owned sector matches the colour of the player.	Pass	UE 1
	UnitTest			
U1	DestroySelf_UnitNotInSectorAndNotInPlayersUnitsList	When DestroySelf is called, the unit is removed from its sector and from its owner's unit list.	Pass	UE 1
U2	LevelUp_UnitLevelDoesNotPastFive *Requirement:* N9	If a unit's level equals 5 and LevelUp is called, the level remains equal to 5.	Pass	UE 1
U3	LevelUp_UnitLevelIncreasesByOne *Requirement:* N9	If a unit's level is < 5, calling LevelUp will increase its level by exactly 1.	Pass	UE 1
U4	MoveToFriendly_UnitInCorrectSector *Requirement:* F5	The Unit moves to the new sector, and is no longer present in the old sector.	Pass	UE 1
U5	MoveToFriendlyFromNull_UnitInCorrectSector *Requirement:* F5	Unit moves to the sector, checks to see if the unit is in sector.	Pass	UE 1
U6	MoveToNeutral_UnitInCorrectSector *Requirement:* F5	Unit moves to sector. Tests that the unit is in the new sector and no longer in the old sector.	Pass	UE 1
U7	MoveToHostile_UnitInCorrectSectorAndLevelUp *Requirement:* F5	Tests that the Unit successfully moves to the sector, levels up and is no longer in the old sector.	Pass	UE 1
U8	SelectAndDeselect_SelectedTrueWhenSelectedFalseWhenDeselected	Tests that when a Unit is selected, selected == true, and when it is deselected	Pass	UE 1

		selected == false.		
U9	SwapPlaces_UnitsInCorrectNewSectors *Requirement:* F5	Tests that when SpawnPlacesWith is called, the two Units move from their old sector to their new one, swapping places.	Pass	UE 1

Evidence ID	Evidence Of Testing
UE 1	

```
Test Runner
PlayMode | EditMode
Run All | Run Selected | Rerun Failed                Run all in player (StandaloneWindows64)
<No categories available>  ✓52  ⊘0  ○0
▼ ✓ GameTest
    ✓ CreatePlayers_AtLeastTwoPlayersAreHuman
    ✓ CreatePlayers_AtMostFourPlayersAreHuman
    ✓ CreatePlayers_FourPlayersAreHuman
    ✓ CreatePlayers_ThreePlayersAreHumanAndOneNot
    ✓ CreatePlayers_TwoPlayersAreHumanAndTwoNot
    ✓ EndGame_GameEndsCorrectlyWithNoCurrentPlayerAndNoActivePlayersAndNoTurnState
    ✓ GetWinner_NoWinnerWhenAPlayerHasLandmarkAndAnotherHasUnits
    ✓ GetWinner_NoWinnerWhenMultiplePlayersOwningLandmarks
    ✓ GetWinner_NoWinnerWhenMultiplePlayersWithUnits
    ✓ GetWinner_OnePlayerWithLandmarksAndUnitsWins
    ✓ InitializeMap_OneLandmarkAllocatedWithUnitPerPlayer
    ✓ NextPlayer_CurrentPlayerChangesToNextPlayerEachTime
    ✓ NextPlayer_EliminatedPlayersAreSkipped
    ✓ NextTurnState_TurnStateProgressesCorrectly
    ✓ NoUnitSelected_ReturnsFalseWhenUnitIsSelected
    ✓ PassTurn_CorrectlyPassesTurn
    ✓ SpawnPVC_doesNotSpawnAfterFirstTurn
    ✓ SpawnPVC_doesNotSpawnMultiplePVCs
    ✓ SpawnPVC_PVCSpawnsInUnownedSector
    ✓ SpawnPVC_spawnsAfter10Turns
▼ ✓ NonHumanPlayerTest
    ✓ FindBestMove_AllMovesSameScore
    ✓ FindBestMove_NoPossibleMoves
    ✓ FindBestMove_OneBestMoveThreeBadMoves
    ✓ MakeMove_executesValidMove
▼ ✓ PlayerTest
    ✓ CaptureLandmark_BothPlayersBeerAmountCorrect
    ✓ CaptureLandmark_BothPlayersKnowledgeAmountCorrect
    ✓ CaptureLandmark_NeutralLandmarkPlayerBeerAmountCorrect
    ✓ CaptureLandmark_NeutralLandmarkPlayerKnowledgeAmountCorrect
    ✓ CaptureSector_ChangesOwner
    ✓ IsEliminated_PlayerWithNoUnitsAndNoLandmarksEliminated
    ✓ SpawnUnits_NotSpawnedWhenLandmarkNotOwned
    ✓ SpawnUnits_NotSpawnedWhenLandmarkOwnedAndOccupied
    ✓ SpawnUnits_SpawnedWhenLandmarkOwnedAndUnoccupied
▼ ✓ SectorTest
    ✓ AdjacentSelectedUnit_SectorsAreAdjacent
    ✓ ClearUnit_UnitRemovedFromSector
    ✓ Highlight_SectorColourCorrect
    ✓ Initialize_OwnedAndNotOwnedSectorsOwnerAndColor
    ✓ MoveIntoFriendlyUnit_UnitsSwapSectorsAndTurnStateProgressed
    ✓ MoveIntoHostileUnit_AttackingUnitTakesSectorAndLevelUpAndTurnEnd
    ✓ MoveIntoHostileUnit_DefendingUnitDefendsSectorAndTurnEnd
    ✓ MoveIntoHostileUnit_TieConflict_DefendingUnitDefendsSectorAndTurnEnd
    ✓ MoveIntoUnoccupiedSector_NewSectorHasUnitAndOldDoesNotAndTurnStateProgressed
    ✓ OnMouseAsButton_CorrectUnitIsSelected
    ✓ SetOwner_SectorOwnerAndColorCorrect
▼ ✓ UnitTest
    ✓ DestroySelf_UnitNotInSectorAndNotInPlayersUnitsList
    ✓ LevelUp_UnitLevelDoesNotPastFive
    ✓ LevelUp_UnitLevelIncreasesByOne
    ✓ MoveToFriendly_UnitInCorrectSector
    ✓ MoveToFriendlyFromNull_UnitInCorrectSector
    ✓ MoveToNeutral_UnitInCorrectSector
    ✓ SelectAndDeselect_SelectedTrueWhenSelectedFalseWhenDeselected
    ✓ SwapPlaces_UnitsInCorrectNewSectors
```