**"World Conquest" Low-Level Design Document**

**1. Introduction**

* **Purpose**: To elucidate the objectives of the low-level design document and its intended readership.
* **Scope**: To specify the precise components and functionalities encompassed by the document.

**2. Component and Class Design**

* **Map Generator (MapGenerator)**
  + **Responsibility**: To load and display the game map, ensuring it contains 6 continents and their respective countries.
  + **Methods**: GenerateMap(), LoadContinents(), LoadCountries()
* **Player Setup (PlayerSetup)**
  + **Responsibility**: To manage the sequence of players, color selection, army distribution, and module selection.
  + **Methods**: DetermineOrder(), SelectColor(), DistributeArmies(), SelectModule()
  + 图示

    描述已自动生成

**3. Game Mechanics**

* **Conscription Service (ConscriptionService)**
  + **Responsibility**: To calculate and allocate new army units.
  + **Methods**: CalculateReinforcements(), DeployReinforcements()
* **Attack Mechanism (AttackService)**
  + **Responsibility**: To handle the offensive and defensive actions between players.
  + **Methods**: InitiateAttack(), CalculateVictoryChance(), OccupyTerritory()
* **Card Reward System (CardRewardSystem)**
  + **Responsibility**: To assign cards to players who conquer territories.
  + **Methods**: IssueCard(), ValidateCardSet(), RedeemCardSet()
* **Territory Expansion (ExpansionService)**
  + **Responsibility**: To manage the acquisition and deployment of territories post-battle.
  + **Methods**: ExpandTerritory(), FortifyTerritory()

**4. Data Model/Database Design**

* **Player Information Table (Players)**
  + **Fields**: PlayerID, PlayerName, PlayerColor, ArmiesCount
* **Map Information Table (Maps)**
  + **Fields**: MapID, Continent, Country, OccupyingPlayerID

**5. Interface Design**

* **Interface Controller (InterfaceController)**
  + **Responsibility**: To define navigation and interaction within the game menus.
  + **Methods**: DisplayMenu(), NavigateSinglePlayer(), NavigateMultiplayer(), LoadGame(), AdjustSettings()

**6. Algorithm Design**

* **Victory Calculation Algorithm (VictoryCalculationAlgorithm)**
  + **Pseudocode**: Function CalculateVictory(player, gameState):
  + victoryPoints = 0
  + # Check if the player occupies all continents
  + if player. OccupiedContinents = = gameState. AllContinents:
  + victoryPoints += 5
  + # Add points for each occupied country
  + , for each country in the player. OccupiedCountries:
  + victoryPoints += 1
  + # Add extra points if the player holds a victory card
  + If player.HasVictoryCard():
  + victoryPoints += 2
  + # Check to see if the number of points required for victory has been reached
  + If victoryPoints &gt; = gameState.RequiredVictoryPoints:
  + Return True
  + Otherwise:
  + Return False
  + **7. Testing Plan**
* **Unit Testing**
  + **Test Cases**: Generate map
  + Purpose: To ensure that the map is correctly generated and contains the 6 continents and their countries.
  + Steps:
  + 1. Call the MapGenerator.GenerateMap() method.
  + 2. Check whether the returned map object contains six continental objects.
  + 3. Check that each continent object contains the correct number of countries.
  + Expected result: a predefined number of countries per continent.
  + Player order determination
  + • Purpose: To verify that the player order is correctly determined.
  + • Steps:
  + 1. Create an array of player objects.
  + 2. Call PlayerSetup. DetermineOrder () method.
  + 3. Check whether the order of returned players is randomized.
  + • Expected result: The player order is different from the input order.
* **Integration Testing**
  + **Testing Strategy**: Game initialization
  + Purpose: Verify that the game initialization is correct, including map generation and player Settings.
  + Steps:
  + 1. Call MapGenerator.GenerateMap() to generate the map.
  + 2. To each player to invoke PlayerSetup. DetermineOrder () to determine the order.
  + 3. To each player to invoke PlayerSetup. DistributeArmies () distribution of forces.
  + Expected results:
  + Maps are correctly generated and assigned to the game.
  + The order of players is determined at random.
  + Each player is assigned the correct number of troops.
  + •**8. Development Environment**
* **Development Engine**: Unity
* **Programming Language**: C#

**9. Appendix**

* **Glossary**:
* 1.MapGenerator: The component responsible for generating and loading the game map, ensuring that the map contains predefined continents and countries.
* 2. PlayerSetup: The component that handles the player initialization process, including player order determination, color selection, army assignment, and module selection.
* 3. ConscriptionService: A service that calculates and assigns new army units to the player.
* 4. AttackService: A service that manages attack and defense mechanics between players.
* 5. CardRewardSystem: A system that assigns bonus cards to players after capturing territory.
* 6. ExpansionService: A service responsible for acquiring and deploying territories after the war.
* 7. VictoryCalculationAlgorithm: algorithm to calculate whether the player to victory conditions.
* 8. Unit tests: Tests against individual components (such as functions, methods, or classes) to verify that they work as expected.
* 9. Integration testing: Tests the ability of multiple components to work together to ensure that they function properly when together.
* •**References**: A list of all the references and resources used.