**Design Document**

Shamus Murray, Garrett Becker, Lucas Burns, Logan Mealy, John Will Bryan

**Test Plan Overview:**

Design Review:

Review classes and methods to be implemented when designing game.

When designing, check to make sure:

* All classes needed to properly implement game framework are present.
* Add additional classes such as Custom Game Actions for moving, attacking, etc to Risk specific classes.
* Review the methods that make up each class to make sure they will be sufficient for testing the major functionality listed below.

Functionality Test:

Test using:

-Initial Function and Crash Test: Run game on real-world device, testing for initial crashes and testing all desired functionality works using specific interaction. Fix any code needed to get all major functionality to run without crashes.

-Unit Tests: Write unit test cases that not only test major functionality but also test edge cases and any unusual circumstances. Make changes accordingly to make sure all these tests pass.

-Beta Testing: Final phase of testing will involve packaging the game and distributing the game to a group of testers who will report and bugs and crashes if they run into any. A 2nd round of beta testing may be needed if many major bugs are found to make sure they have been fixed; however this depends on time allocation.

-Functionality to test for:

* Players are alerted and allowed to take certain actions when it is/isn’t their turn. A text indicator should display turn information and actions will be disabled/enabled depending on turn status.
* Player in turn should be able to select a country, have appropriate actions be enabled, and use those actions to interact with that country or other countries (such as attack and move).
* Rolling the dice for both attackers and defenders yields the appropriate dice values and the game calculates the winner correctly based on who gets highest die value.
* Winning/losing should call the appropriate game state methods in order to deduct the appropriate troops from the losing country.
* If a country is completely overtaken, indication of new ownership should be made clear by map of countries.
* The interaction between two countries must be for adjacent countries only.
* Surrendering the game should end game for both players, declaring the other player as winner.
* Ending the turn should disable appropriate actions for player and enable appropriate actions for other player until that player ends their turn.
* All UI elements should remain static and should not move. Textviews should only update with new information as game progresses.
* Computer Player 1 should only use basic logic to play the game (random moves, attacks, surrendering).
* Computer Player 2 should use more complex logic to play the game (will use more human-like move, attack actions based off player positions and will not surrender).

**Schedule:**

* (03/29/15) Players know at all times whose turn it is, for both defending and attacking.
* (04/05/15) Players have a visual representation of what player controls each country.
* (04/05/15) Players have a visual representation of how many troops are in each country.
* (04/12/15) Players can surrender at any point during the game, giving a notification to both players that the other player has won.
* (04/12/15) Players have a representation of who wins a battle.
* (04/18/15) Smart AI wins 50% or more of games.

**Class Overview:**

RiskAttackAction:

* Contains all the methods regarding attack moves and sends that info to the gamestate.

RiskAttackRollAction:

* Contains all the methods regarding rolling the dice as an attacker and sends that info to the gamestate.

RiskDefendRollAction:

* Contains all the methods regarding rolling the dice as a defender and sends that info to the gamestate.

RiskMoveTroopAction:

* Contains all the methods regarding moving troops on the board and sends that info to the gamestate.

RiskPlaceTroopAction:

* Contains all the methods regarding placing troops at the beginning of a turn and sends that info to the gamestate.

RiskSurrenderAction:

* Contains all the methods regarding surrendering anytime throughout the game and sends that info to the gamestate.

RiskHumanPlayer:

* Contains the player’s name, the current state of the game,updating display based on actions from the human player.

RiskMainActivity:

* Contains methods regarding the default configuration of the game including the number of players and types of players allowed.

ComputerPlayer1:

* Dumb AI.
* Sends randomized computer player actions to the game.

ComputerPlayer2:

* Smart AI.
* Sends logical computer player actions to the game.
* Will be given advantage in dice rolls.

RiskGame:

* Interface that Implements Risk-Game.

RiskLocalGame:

* Class that represents the state of the game.
* Contains methods to create a new state, check if game is over, send updated information about the state to a given player.

RiskPlayer:

* Represents the RiskPlayer object.

**Class Diagram:**

