Mark Wolgin, Josh Davis

Professor Silber

CISC 181

4 April 2018

Project Proposal and Design Document

**Description**

We would like to design, script, and write a digital version of the board game Twilight Struggle.  The game is like risk, but more subtle, realistic, and slightly more complicated. We will describe the basics of the game below, but [a PDF of the board games’ rules can be found here.](http://www.gmtgames.com/nnts/TS_Rules-2015.pdf) We plan to do most of the original game, with only some minor tweaks, listed as follows

1. We have decided to not implement the space race. It adds more complexity to an already fairly complicated game. We felt that in the time we have it makes sense to drop this feature because it very rarely has any appreciable impact on the game.
2. We also have decided to scale back the size of the world, to just North and Central America, East and West Europe, and Asia and Southeast Asia. We are dropping the Middle East, South America and Africa, just to simplify the map and amount of information that would need to be represented on a mobile platform.
3. Finally, we will also use a reduced set of cards. In the original game there are over 150 unique events. To make matters simpler we plan to implement between 20-30 cards, cutting cards that either:
   1. No longer have a use due to removal of space race or countries that were associated with it.
   2. Are too complicated to implement for a project of this scope

Twilight Struggle is a game simulating the Cold War. The two players, the USA and the USSR, work to exert their ideological influence, represented numerically, over the countries of the world. They do this using cards from their hand that they play on their turn to either launch events which have their own specific effects, or add influence to countries of their choice. Countries have a stability number, which is the number of influence points a player needs beat the opponent's influence value by in order to control the country. Control of a country counts towards victory points for the player when scoring cards are played. The game is divided into ten turns, each with eight action rounds. Each action round contains a headline phase, during which players play a card from their hand, and a influence round, when players place influence points, or roll realignments or roll coups. Realignments and coups are ways of rolling to reduce enemy influence, and their description can be found in the rules (pg. 6-7). Victory points are a scale from +20 (US Victory) to -20 (USSR Victory), and one way of winning the game. The other way is by forcing the other player to Defcon 1 (from starting at Defcon 5) through events.

**Class Specification and Hierarchy**

Card – Interface: *This is the base for both other types of Card. It has a name, description, and a method stub runEffect. This allows for an ArrayList<Card> to exist in the Board Class and in hand, and allowing the scoring cards to coexist.*

* int cardNum
* String name
* String description
* void runEffect(int ID)

TurnCard – extends Card: *TurnCard extends Card, and as such as all of its methods and variables. It also includes a value for each card, a card side (USA, NEUTRAL, USSR), effectID, specialCard. Each card will get its data from a card.csv, to save on actual code.*

* int value
* int side
* int effectID
* boolean specialCard

ScoringCard – extends Card: *Scoring Card is a subset of card, but it has only one function, to change the Victory Points ticker. This is why it has its own class.*

* specialEffect(int ID)

Player: *This is the player, it has a hand to hold cards, and an Enum to determine the side (USA< USSR), and a method to playCard.*

* ArrayList<Card> hand
* Enum sideID
* playCard()

Country: *Mostly just a holder of values. It has a name, stability number, tallies of US and USSR influence, knowledge of whether its a battleground or not, an Enum to determine what continents it is a part of, and a method to change the US and USSR influence. Will be filled via a .csv sheet.*

* String name
* int stabilityNum
* int USInfluence
* int USSRInfluence
* boolean battleground
* modifyInfluence(int value, Enum side)
* ArrayList<Enum> continents

Board: *Holds all game elements in one class. The setup method places all the default information and initializes all of the variable. This includes the reading and setting of all card and countries. Also tracks current turn, action round, victory points, defcon, and hold the methods to place influence, roll a coup, and roll a realignment.*

* ArrayList<Card> deck
* ArrayList<Card> discard
* int turn
* int actionRound
* int victoryPoints
* int defcon
* int USMilitaryOps
* int USSRMilitaryOps
* placeInfluence(Enum side, Country cont, int value)
* rollCoup(Country cont, int value)
* rollRealignment(Country cont, int Value)
* setup()

Effects – Static: *A static method that just holds a large switch statement or if/else statement. One for cards and one for score cards. Determines the card effect to be acted by the effect id of the card.*

* getEffect(int ID)
* getScoringEffect(int ID)

Controller: *Launches the game. Creates and initializes the board. Checks to see if the game should end. Determines the winner. Updates the UI, with the UI able to change depending on need. Determines what player should play next.*

* main(String[] args)
* initialize(Board b)
* gameEnd(Board b)
* getVictor(Board b)
* updateUI(Board b)
* playerTurn(Board b)

UICore - Interface: *This is the interface for all graphical interfaces. WIll have a Board object that points to the actual board. updateBoard(Board b) will use a board to update all the information inside the UI and prepare it for output. updateUI() will output the UI.*

* Board currentBoard
* updateBoard(Board b)
* updateUI()

UIText - extends UICore: *This is the text based implementation of the game. It will only output diagnostic data and basic game functions to the terminal*. Ex:

GAME STATUS AND INFO

TURN:2, ACTION ROUND:5

REQUIRED MILITARY

OPS[0, 0]

VICTORY POINTS:-7

DEFCON:5

CURRENT PLAYER:USSR

------------------------

ISO\_ISO\_STB\_US, RA\_INFLU

WEE\_GBR\_[5]\_[5, 0]\_[USA] <- West Europe, United Kingdom, Stability of 5,

WEE\_FRA\_[4]\_[4, 0]\_[USA] USA influence 5, USSR influence 0, USA Control

EEE\_DDR\_[4]\_[0, 4]\_[SUN]

WEE\_DUE\_[4]\_[4, 0]\_[USA] <- West Europe, German Republic, Stability of 4

WEE\_BEL\_[3]\_[0, 0]\_[UNK] USA influence 4, USSR influence 0, USA Control

----------USA-----------

CRD\_CLS\_TYP\_VAL\_EID

001\_TCD\_USA\_003\_001 <- Card Number 1, It's a TypeCard, It’s a USA

030\_TCD\_SUN\_001\_021 card, It has a value of 3, Its event ID is 001

064\_SCD\_xxx\_xxx\_S01

051\_TCD\_UNK\_004\_043

---------USSR-----------

CRD\_CLS\_TYP\_VAL\_EID

011\_TCD\_USA\_003\_006

020\_TCD\_SUN\_001\_023

054\_SCD\_xxx\_xxx\_S04 <- Card Number 54, It's a Scoring Card

031\_TCD\_UNK\_004\_050

*Follows* [*ISO 3166-1 Alpha-3*](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3)*, with the exception of SUN for Soviet Union and WEE for Western Europe and EEE for Eastern Europe.*

UIGraphical - extends UICore: *This is the graphical interface of the game. It will assemble the game board elements, and redraw the image components, so when updateUI is called the ui is rendered. It also will have to handle key presses and mouse clicks.*

**Class Hierarchy**



