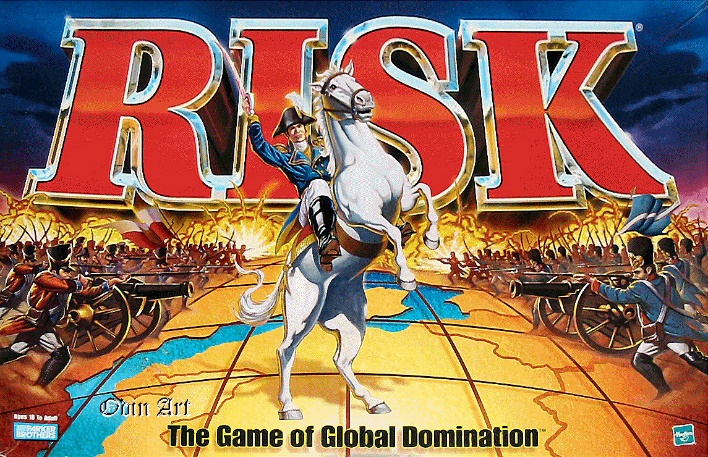
**world conquest SPRINT II design documet**

A Design Document for the Game ‘World Conquest’ for Raffle Games



March 12, 2024

Team one

University of Sussex

Contents

[Sprint II 2](#_Toc160539979)

[Design Objectives 2](#_Toc160539980)

[UI Design 2](#_Toc160539981)

[Class table 3](#_Toc160539982)

[UML Diagrams 4](#_Toc160539983)

[Skeleton Code 4](#_Toc160539984)

[Conclusion 4](#_Toc160539985)

[References 4](#_Toc160539986)

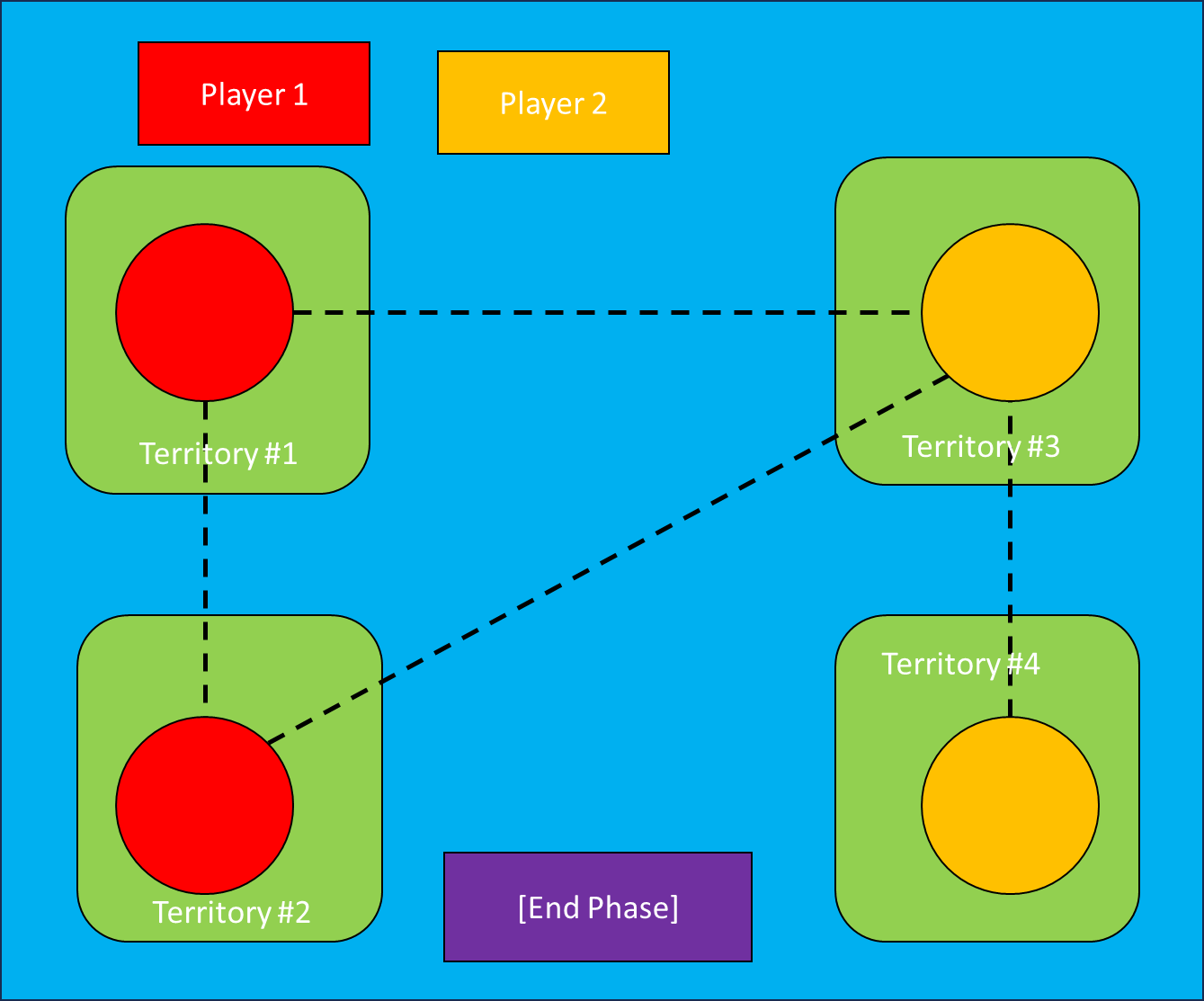
# Sprint II

## Design Objectives

For this sprint, our implementation aims are:

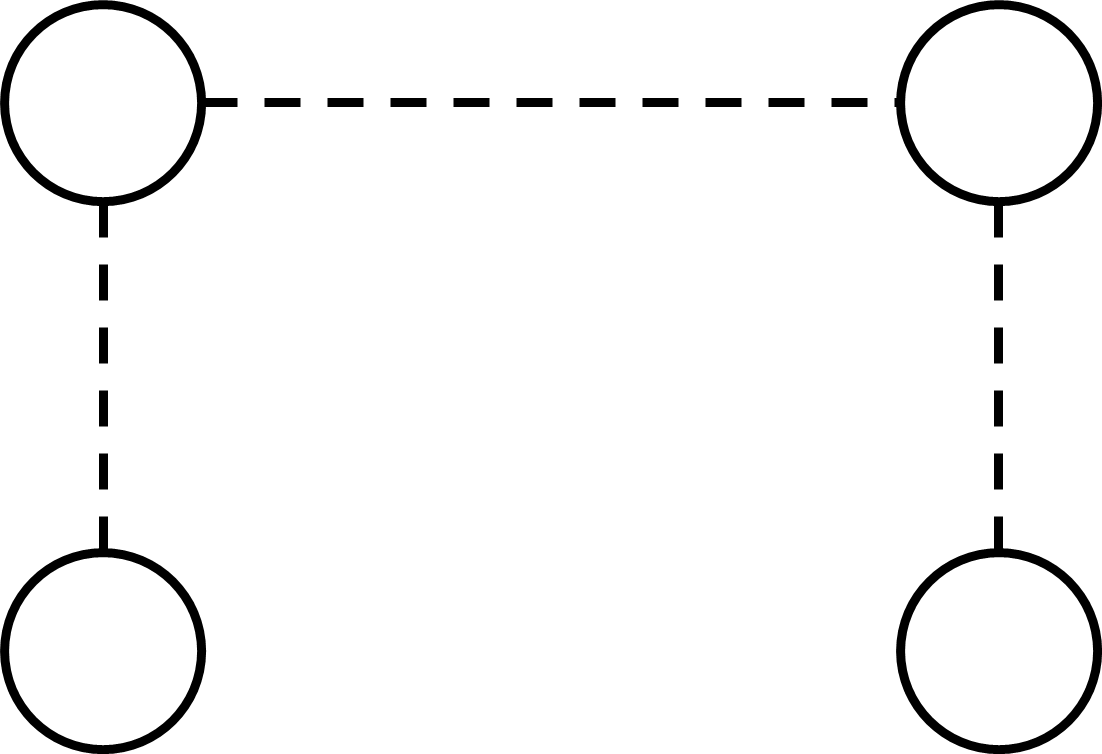
* Be able to deploy troops on a players turn
* Attack implementation with dice
* A total of 4 territories
* 2 players
* 5 troops per territory
* Territories can be conquered by another player (colour of node will change to opponent player)
* Player 1 is no longer the default winner (extension from sprint I
* Territories display their name
* Country selected with mouse

## UI Design

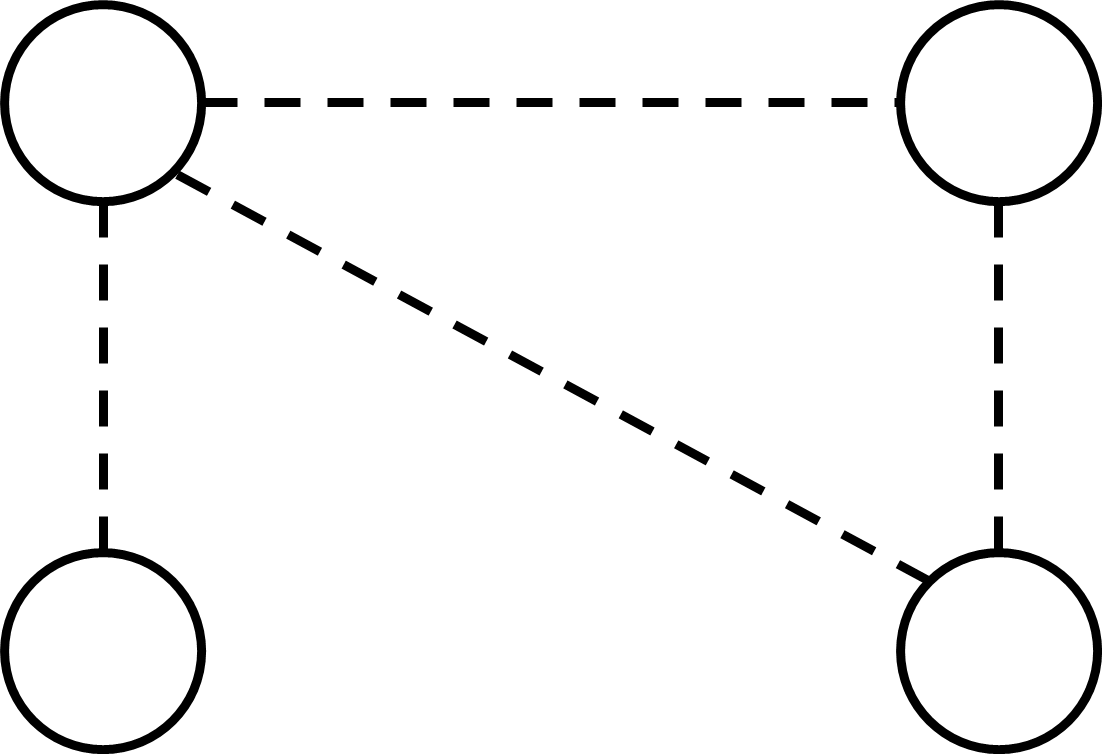


There are a variety of node connections that can be chosen when creating the map – this was one of a variety that I thought would be good for this sprint, but there is also:

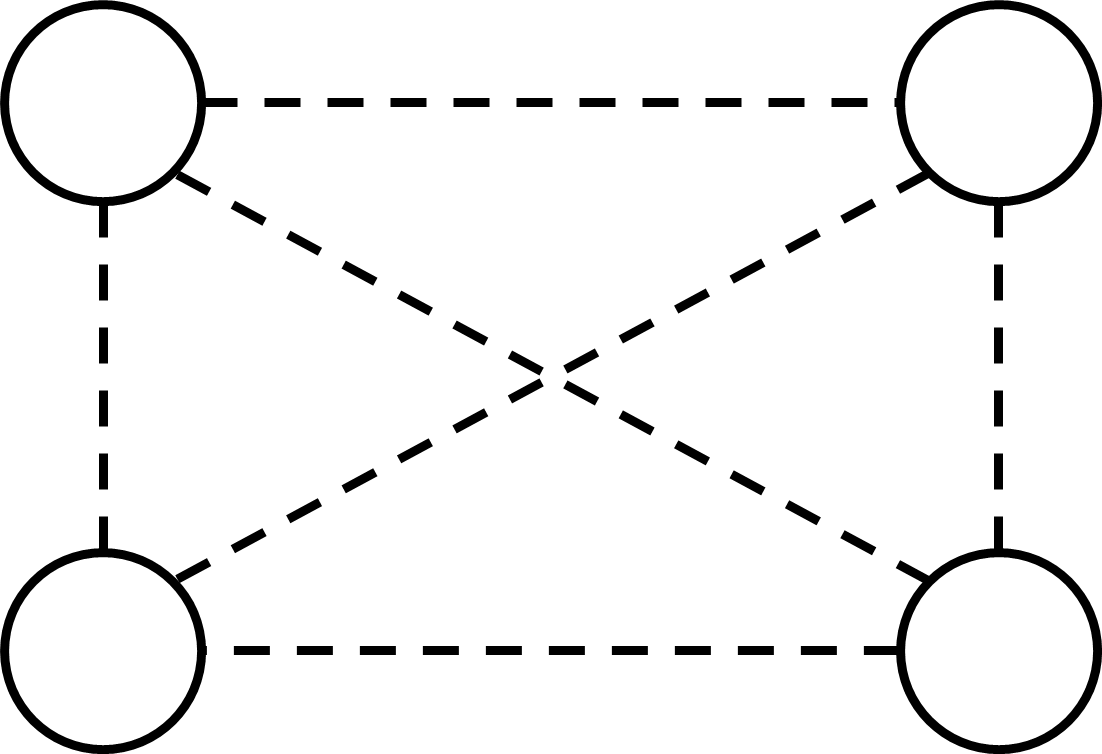
Map #1: Nodes have a connection 1 or 2 – it is a simpler implementation and it allows



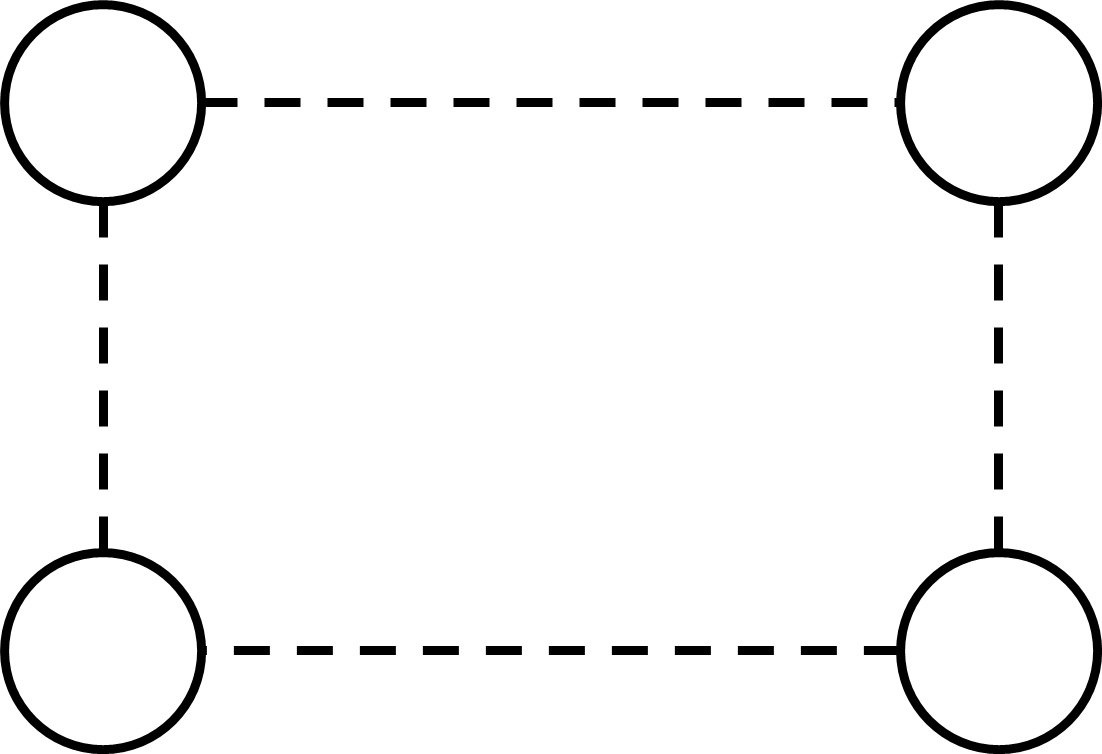
Map #2: Nodes have a connection of 1, 2 or 3 – This graph layout has a lobsided design where it is not symmetrical - which gives an advantage or disadvantage to players depending on what territories they hold



Map #3: All nodes have a connection of 3 – this is the most even graph layout – each node has the name number of connections so the difference between one territory or another is virtually negligible.



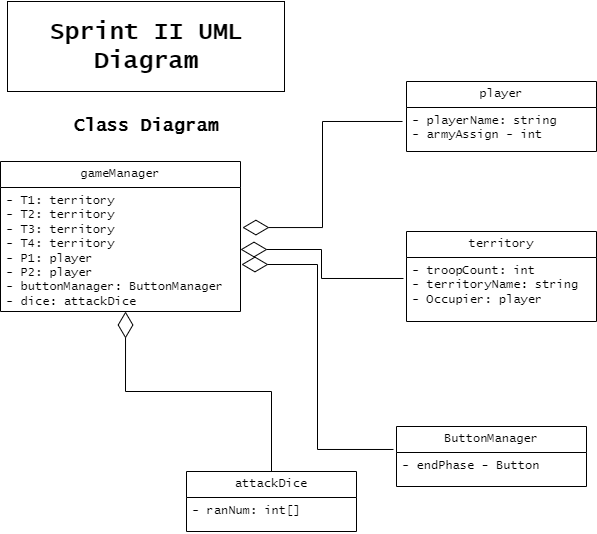
Map #4: All nodes have a connection of 2 – this has the same uniformity as map #3 but it has the has the added advantage of being a lot simpler in implementations.



## Class table

|  |  |  |  |
| --- | --- | --- | --- |
| Class No. | Class Name | Attributes | Comments |
| 1 | player | * playerName: string * armyAssign - int | Player class, contains the player information as in Sprint I |
| 2 | territory | * troopCount: int * terrirotyName: string * Occupier: player | Terrorizes are able to occupied by different players depending on the outcome of player attacks |
| 3 | ButtonManager | * End\_phase: Button | The Button should only end the phases that the player is in (such as in the event a player wants to end their attack phase prematurely) |
| 4 | GameManager | * T1: territory * T2: territory * T3: territory * T4: territory * P1: player * P2 player * buttonManager: ButtonManager * dice: attackDice | The gameManager holds all the game object information as it is an implementation of the classes above |
| 5 | attackDice | * ranNum: int[] | The Dice class should determine if the player has won or lose a battle by return the outcome of random numbers generated |

## UML Diagrams



## Skeleton Code

# Conclusion

The conclusion of this sprint to come out with a prototype with something akin

# References

Parker Brothers, 1993. *RISK - The World Conquest Game,* Beverly: Tonka Corporation.