CEN 4020: Software Engineering I, Spring 2020
Florida State University
- Group Project Proposal–

1. **Project title**
   Riskier (TBD)

2. **Brief overview of what you are proposing**
   Riskier is a 2-D command & control game. The goal of the game is to control the entire map or eliminate all other players. The game is a standalone game that implements a game board with sectioned into region that the player can play against other players or AI and control resources (soldiers) to take over other regions. When players compete for regions the game enters a turned based combat system that allows the player to use their skill and resources to compete and win.

3. **Motivation**
   We are most interested in developing a game as we all have experience, have an abundance of developing resources, and have the ability to expand on the project in the future. In addition to this, we all look forward to learning different technology stacks than those used in our programming classes.

4. **Features to be implemented and types of user**

Game Features (See requirements/Project_DesignDoc for design mockups)

Blue - Critical Feature

Black – Preferred Feature

Red - Optional Features

1. Title Screen
    a. Start Game
        i. New Game
        ii. Saved Game
        iii. Network
    b. Options
    c. Online Leaderboard
        i. Google Firebase Users
        ii. Rankings based on wins/power
    d. Exit
2. Game Setup
    a. Map Selection (if multiple)
    b. Character Selection
        i. Character Special Ability
    c. Opponent Settings
        i. Set Difficulty
        ii. Set Play-Style

d. Additional Options
    i. Turn Settings
    ii. Battle Settings

3. Game
    a. Game Board
        i. Presentation of Map
        ii. Starting Position/Moves
        iii. Map Control
            1. Territory Control & Power
            2. Movement of Power

    b. Menu Options
    c. Turn System
        i. Staged Turn
            1. Allocate
                a. Introduce new power
            2. Move
                a. Move power around gameboard
            3. Battle
                a. Use power to battle against opponents
    d. Player Information/Inventory
        i. Power Available
        ii. Statistics
            1. Power gain/loss
            2. Battles won/lost
    e. Opponent Information
        i. Power Available
        ii. Statistics
            1. Power gain/loss
            2. Battles won/lost
    f. Power
        i. Units
            1. Unit 1 - Attack Oriented
            2. Unit 2 – Defense Oriented
        ii. Player Modifiers
    g. Game AI
        i. Opponent
4. Battle
    a. Menu Options
    b. Player Board
        i. Power Position
            1. Attack Position
            2. Support Position
            3. Retreat Position

c. Opponent Board
    i. Power Position
        1. Attack Position
        2. Support Position
        3. Retreat Position
d. Turn System
    i. To be determined by rules
e. Player Actions
    i. To be determined by rules
f. Battle Information
    i. To be determined by rules

5. **Risk / Challenges**
    - Time Commitment (5 upper-level full schedules)
    - Unfamiliar game development software (Unity)
    - Developing a Game AI
    - Creating a **balanced** game
    - Analytical-based group (creativity portions may suffer)

6. **Existing related project**.
    - Risk (Board Game) - Map Control
        - https://shop.hasbro.com/en-us/product/risk-game:2C7C6F52-5056-9047-F5DD-EB8AC273BA4C
        - https://store.steampowered.com/app/1128810/RISK_Global_Domination/
        - Riskier will feature improvement to the combat system that moves from a number-based combat system to a more advanced battle system that allow for players with less power still have a chance at winning in combat
    - Sid Meyer's Civilization Games - Map Control
        - https://civilization.com/civilization-5/
        - We plan on using aspects such as a hex system that allow for clear borders between territories. Along with this aspect such as various power units that allow for different strategies in play. We plan to improve on the simplicity of the tile system that allows you to see the territory you control and moving your units around the field.
    - Final Fantasy Games - Combat
        - https://finalfantasy.fandom.com/wiki/Battle_system
        - The final fantasy combat system provides a good basis for how different units can interact and be balanced. We plan to improve by making the system simpler but still robust.
    - Fossil Fighter DS Game - Combat
        - https://fossilfighters.fandom.com/wiki/Fossil_Battles
        - We on incorporating the Zone system that Fossil Fighters introduced. We plan on improving on this system by making it straight-forward and approachable for a new player.

7. **Intended platform / programming language**
    - Windows 64-bit
    - Unity Game Engine
        - C#

8. **Third-party libraries / APIs to be used**
    - GitHub - version control and document management
    - Unity Game Engine
    - Unity
    - Google Firebase (for possible online stats saving)
    - Google Cloud Platform (server) (if multiplayer implementation)

9. **Team members, expertise, project responsibilities, and team organization**

Member	**FSUID**	**Expertise**	**Project Responsibilities**
Wesley Watkins	wjw16	• C++ • Python • AI	• Game AI
Antonio Vidal	aa17t	• Android App Dev • Google Firebase • Java	• 1v1 Combat Turn • Rules
Bradley Hodson	bmh17b	• C++ • Java • Game experience	• Rules • Game Board
Michael Styron	mts17b	• C++ • Python	• UI • Menu
Grayson Wagstaff	gaw17d	• C++ • Python • Sysadmin	• Management • System Administration • Fill

We plan to use the Agile method of code sprints as it lines up with the continuous due dates set by the assignment. At the end of each sprint we will meet to determine what went wrong, what went right, and to see where we can improve the next sprint.

The team decided in our first meeting to have each person “manage” a part of the codebase a main developer. Along with specified code owners we plan to have rotating developers verify a different piece of the codebase via manual review before checking it in to the master branch at the end of the sprint. Due to the nature of a game, automated testing beyond testing compilation may not be possible, relying on manual review of other developers to catch mistakes. Each sprint the developer

must have their manual reviewer review the code **at least once** if not multiple times to confirm the code performs to the expected requirements.

For management of code check-ins, developer rotation, and task assignment the group has elected Grayson Wagstaff. Along with these duties, Grayson will fill in on tasks that fall behind on sprints or assist developers that need it.

For communication, we will use a Discord server that provides multiple channels to discuss core features and allow other developers to catch up what it is happening within a branch. Along with this, we have a group message for important messages that go outside of the normal developing process. Meetings are also scheduled at the end of sprints to review and plan for the next sprint.