CSCI 3308 - Lab 105 Team 2 Due: 2/16/20
Project Milestone 2

**Team Name:** CU Dream Team

**Team Members:**
Michal Bodzianowski, Gunnar Enserro,
Jacob Klus, Tommy Kopala, Isaac Vance

## Warcrimes.io Milestone: 2

Project Features:

1. Multiplayer
    a. The game will be played in sessions filled with 20 players. The goal will be to eliminate all other players.
    b. Non-Functional Requirements:
        i.
    c. Functional Requirements:
        i. Architecture
            1. Backend Redis Server
            2. Web Server
            3. Backwards Proxy Server
            4. Session Proxy Server
    d.
2. 4x Model
    a. The 4x Model is expand, explore, exploit and exterminate. Players will need to explore the map and exploit available resources to build up armies and be the last player.
    b. Functional
        i. Unit functionality
            1. What: The main feature of 4x games is unit interactions. Units will be the driving force behind the game.
            2. Who: Any players
            3. Why: It is one of the core components of the game
            4. When: During the game
        ii. Win/Loss conditions
            1. What: Informs the players when they they have won or lost
            2. Who: Players who meet either condition.
            3. Why: Gives players a goal to obtain and keep them invested

            4. When: At the end of the player's game
    c. Non-Functional Requirements
        i. Control Units
            1. What: Players need to be able to send commands to their units to execute plans.
            2. Who: Players
            3. Why: To add strategy to the game, players will need to be able to send commands to their units.
            4. When: When the player wants to control units
        ii. Build bases/cities
            1. What: Players will be able to build buildings to control points and get more resources
            2. Who: Players
            3. Why: Allowing players to build buildings, players will be able to create more detailed plans and get more invested in the game.
            4. When: When players want to build buildings

3. Leaderboard
    a. Tracks statistics of top players and displays on the Leaderboard page. Updates regularly.
    b. Functional Requirements
        i. Stores information
            1. What: Database stores player information from games
            2. Who: Players
            3. Why: Allows us to fill in leaderboard and get information to balance units
            4. When: After a game is complete
    c. Non-Functional Requirements
        i. Shows information
            1. What: Webpage that lets player see their stats compared to other players
            2. Who: Players
            3. Why: Allows players to see how good they are and compete with friends
            4. When: When players want to see their stats.
4. Terrain Generation
    a. Allows for strategy so players will have to adjust and overcome issues involved in how the land around them is made.
    b. Functional
        i. Wave function collapse

            1. Takes in seed and generates very well designed maps.
            2. Uses pre-made chunks to generate even cooler chunks.
            3.
    c. Non-Functional Requirements
        i. Good design and patterns in land that are strategic to use.
5. Base generator
    a. Kind of like most city builders the land is zoned for certain buildings and structures that will aid in resource gathering and unit building.
    b. Functionality
        i. Users will zone the land in the game and communicate that with the servers.
        ii. The server will describe a timeline for the buildings to complete the build and communicate that back to the users.
        iii. Wave Function collapse will work with on top of terrain to build the bases
    c. Non-Functional Requirements
        i. The users see a satisfying building that generates in layers and conjoins with surrounding paths on the ground.
6. Server side statistics
    a. Since it would be a shame to let the players tell the server if they won the battle, instead the server before interaction or battle will calculate the statistics and prepare ahead. On top of this the server is also scheduling events in the game and announcing them as they occur.
    b. Functionality
        i. State storage
        ii. State backup database incase game server crash
        iii. Statistics engine calculating probabilities and actions.
    c. Non-Functional Requirements
        i. Users will be alerted to enemies and can watch the battles
        ii. Battles will be realistics in states aka the foot troops should have died from the tank.

## **Project Plan:**

Our chosen management tool is a kanban board in trello. The features that are to be developed are all the features listed above.

## **Sprint Sequence:**

Since we are using agile development method, we are focusing on multiple features at once with each of our team members working on their areas of expertise. Our sprint structure will be as follows with the following sub teams:

### Sub Teams:

Main Backend: Gunnar and Isaac
FrontEnd Architecture: Tommy and Michal
FrontEnd Design: Jacob and Michal
Vestibular Feature Backends: Jacob and Isaac
Integration: Full Team led by Gunnar

### Sprint Sequence:

Sprint 1: 2/17 - 3/1:

Main Backend - Establish backend architecture for single session use.
FrontEnd Architecture - Get 4x prototype up and running.
FrontEnd Design - Complete simple layout of website.

Sprint 2: 3/2 - 3/15

Integration - Full team collaboration to get basic application integrated and functional. At the end of this all team members test working application functionality. Once functional, launch website.

Sprint 3: 3/16 - 3/29

Main Backend - Expand backend architecture to host multiple sessions.
FrontEnd Architecture - Work on expanding the generate terrain abilities using the wave collapse feature.
Frontend Design - Improvements on the design and layout of the website.
Integration - Test generate terrain feature after wave collapse feature is implemented to develop this feature.

Sprint 4: 3/30 - 4/12

Main Backend - Continue to scale backend architecture capabilities to support new base generation feature.
FrontEnd Architecture - Expand use of wave collapse feature to implement base generation feature.
FrontEnd Design - Work with Vestibular Database team to begin design of leaderboard feature.
Vestibular Feature Backend - Begin developing architecture for leaderboard feature.
Integration - All team members continue to work together to integrate new features during this sprint, and test base generation feature at the end of this sprint.

Sprint 5: 4/13 - 4/22

Main Backend - Link architecture to leaderboard feature.
FrontEnd Architecture - Finish the design of the game and test for any game and test for bugs.
FrontEnd Design - Finish web design and layout, launch accessibility to leaderboard feature.
Vestibular Feature Backend- Launch the leaderboard feature.
Integration- Test leaderboard feature. Everyone searches for bugs anywhere in the application. Application should include all planned features by this point and achieve state of awesomeness.