Project 5

CSCI 5448

DENNIS WINDHAM

NEILL SHIKADA

CONTENTS

Date: October 31, 2022.

PROJECT SUMMARY

Title.

Object Oriented Civilization Game Clone

Team Members.

Name	Role	Email
Neill Shikada	Graduate Student	Neill.Shikada@colorado.edu
Dennis Windham	Graduate Student	dene5275@colorado.edu
Bruce Montgomery	Instructor	bruce.r.montgomery@colorado.edu

Overview.

Our project is a small Unity Engine video game based on Sid Meier's Civilization. Our version of the game will be based on a square game board as opposed to Civilization's hexagonal board. There will be one human player and several AIs. Each player will have a starting city that can generate one of three rock-paper-scissors-like unit types every 3 turns, and control their movement in the game board. Units and cities have individual health bars and attack strengths, depletion of the health bar leads to removal of the respective entity from the game. A player loses when their city is removed from the game, and the last remaining player wins.

Project Requirements.

- System Requirements
  - The game needs to be fully portable across two major desktop operating systems: Windows (10+) and Linux (kernel 5.15+).
  - Hardware requirements need to be modest, not exceeding the capabilities of entry-level consumer chips.
- Game Board
  - The game board needs to be a square grid of a size that facilitates good gameplay for 2-3 players, likely between 20×20 and 30×30.
  - The game board needs to be able to fit the entire game screen as we do not intend to support camera controls.
  - The game board needs to be interactive to the human user: they will be able to use their mouse to select a unit, get a prompt for which cells the unit may move into, and move the unit by selecting an available cell.
  - No two units should be able to share the same cell in the game board.
- Gameplay
  - Players take turns sequentially. Each turn, the player can use all of their units once, choosing to either move or attack, and produce a new unit at the City. The player needs to be able to skip the turn if they are happy with their decisions and do not to finish the turn.
  - By default, each player has one starting city that can create a unit every 3 turns. The starting city has its own attack power, meaning that units that siege it take damage. A city has 500 base HP.
  - The human player must be able to select one of three base civilization types that determine gameplay changes:
    - Conquerors
      - Damage dealt to enemy cities is increased by $\times 1.5$.
      - Civilization's own city has 25% less health.
    - Barbarians
      - City can produce units every 2 turns instead of 3.
      - Every unit has 30% less health.

        * Defenders
            · City has $\times 2$ health and attack power.
            · City can produce units every 4 turns instead of 3.
    - There will be 3 unit types available to every player. To attack, a unit always needs to be in a cell adjacent to their target. Every unit has the base attack damage of 30 and a total of 100 HP.
        * Melee
            · Deals $\times 1.5$ damage to Ranged units.
            · Deals $\times 0.5$ damage to Airborne units.
        * Ranged
            · Deals $\times 1.5$ damage to Airborne units.
            · Deals $\times 0.5$ damage to Melee units.
        * Airborne
            · Deals $\times 1.5$ damage to Melee units.
            · Deals $\times 0.5$ damage to Ranged units.
    - When a city or a unit lose all their HP, they are removed from the game. If a city is removed, the player who owns the city instantly loses and all their units are removed from the game. If the human player loses, the game is over.
    - Total damage done by units of each player needs to be tracked and displayed at the end of the game.
    - The player may choose to quit the game early, at which point they automatically lose and the game concludes.
- AI
    - The game needs to have a basic AI for the human player to play against.
    - The AI will follow simple heuristics:
        * Keep unit type split close to $1 : 1 : 1$ when producing new units.
        * For every available unit, take the greedy action: if there is a nearby enemy unit or a city, attack them, if not, move towards the closest enemy city.
- UI
    - The UI needs to initially display a civilization type choice to the player, along with respective bonuses.

- During the actual gameplay the UI needs to communicate to the player when they may produce new units.
- The UI needs to tell the player what units may be selected, and what their available actions are after selection.
- After the game is over, the human player will be presented with a scoreboard showing whether they won or lost, and how much damage their units have done in total.
- The player needs to have an button to quit the running game.

Users and Tasks: Use Cases. The system will have a single human user. See Figure 1 for the Use Case diagram.

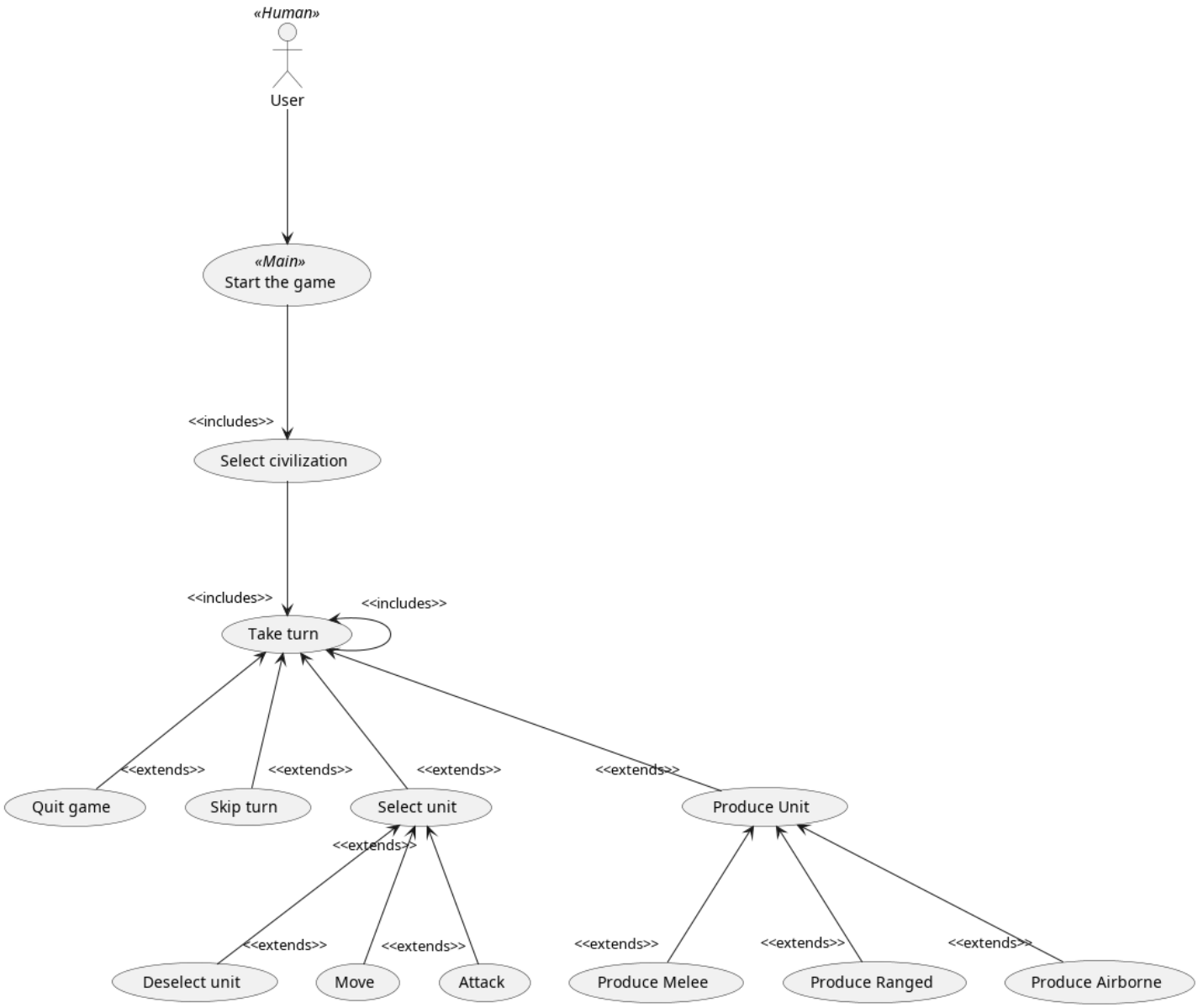

FIGURE 1. Use case diagram.

UML Activity or State Diagram (N).

Architecture Diagram (N).

Data Storage.

The game will keep track of its state in memory. After the an ending condition is reached, the game will save some statistics to a text file:

- Turns taken to finish the game.
- Endgame condition for the human player.
- Total damage done by each player's units.
- How many units of each type were created throughout the game by each player.

UI Mockups / Sketches (N).

UML Class Diagram & Pattern Use (D).

User Interactions / UML Sequence Diagrams (N).