

## **Project 7**

### **CSCI 5448**

DENNIS WINDHAM

NEILL SHIKADA

## **CONTENTS**

<b>Final Project Report</b>	<b>2</b>
Title	2
Team Members	2
Final State of System Statement	2
Final Class Diagram and Comparison Statement	2
Third-Party code vs. Original code Statement	2
Statement on the OOAD process for your overall Semester Project	2
Appendix A: Demonstration Class Diagram	4

**FINAL PROJECT REPORT****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

**Final State of System Statement.****Final Class Diagram and Comparison Statement.**

See [Appendix A: Demonstration Class Diagram](#) for our class diagram.

**Third-Party code vs. Original code Statement.**

We are using the Unity3D game engine for this project. However, we purposely limited involvement of Unity's bloated MonoBehaviour class. It is conventional for Unity developers to have every class in the program inherit from MonoBehaviour, but we are keeping the overwhelming majority of our code pure C#, and where possible use the adapter pattern to interact with MonoBehaviour.

**Statement on the OOAD process for your overall Semester Project.**

- (1) Throughout the development of the project, we held biweekly meetings lasting anywhere from 18 to 24 hours. During our meetings, we primarily discussed the state of our project, design approaches to consider moving forward and how OO patterns could solve the problems we envisioned. This level of dedication secured a robust design for our project early on and helped us stay on track during development.
- (2) A major part of our design process was sticking at least to the following OO principles:
  - Program to an interface, not implementation
  - Encapsulate what varies
  - Favor composition over inheritance
  - Aim for loose coupling, strong cohesion
  - Single Responsibility Principle
  - Open-Closed Principle

- Liskov Substitution Principle

Generally speaking, it worked well for us, although we cannot state with absolute certainty that none of our code violates the said principles. However, we feel that we still managed to write a rather clean and maintainable codebase.

- (3) We felt that as our project grew, the readability of its UML class diagram greatly deteriorated. In fact, at one point, after an intense coding session which involved inclusion of a number of unforeseen classes, we were flabbergasted to find out that the new class diagram looked twice as convoluted as before.

