Index

For the resubmission of our document we made the following changes:

- Structural
    - Overall formatting of the document now has a clear introduction, diagrams with explanations, and the elimination of code snippets.
- Abstraction
    - Instead of describing the features by their methods and individual lines of code we are now describing the features into higher layers of abstractions. The features are now described by their components and how they are connected.
- New Feature
    - We changed one of our features from Save Changes to Map Generator.

Feature 1: End Turn

Introduction

FreeCol is a turn-based strategy game, which means that time in the game is measured in turns. Ending the turn functions as the transition of time, and involves checking for win conditions and other game logic, and advancing or resetting the state of relevant objects to prepare for the next turn. Ending the turn is essential to gameplay - without it, the game would be unable to progress. Since FreeCol is a client-server game, ending the turn requires action on both the client and server sides.

Structure

The end turn feature involves both major components of the program: the client side, and the server side. Figure 1.1 shows a simplified UML class diagram for the client side component. Below, we will go over every client side component class and what it does.

Client Side

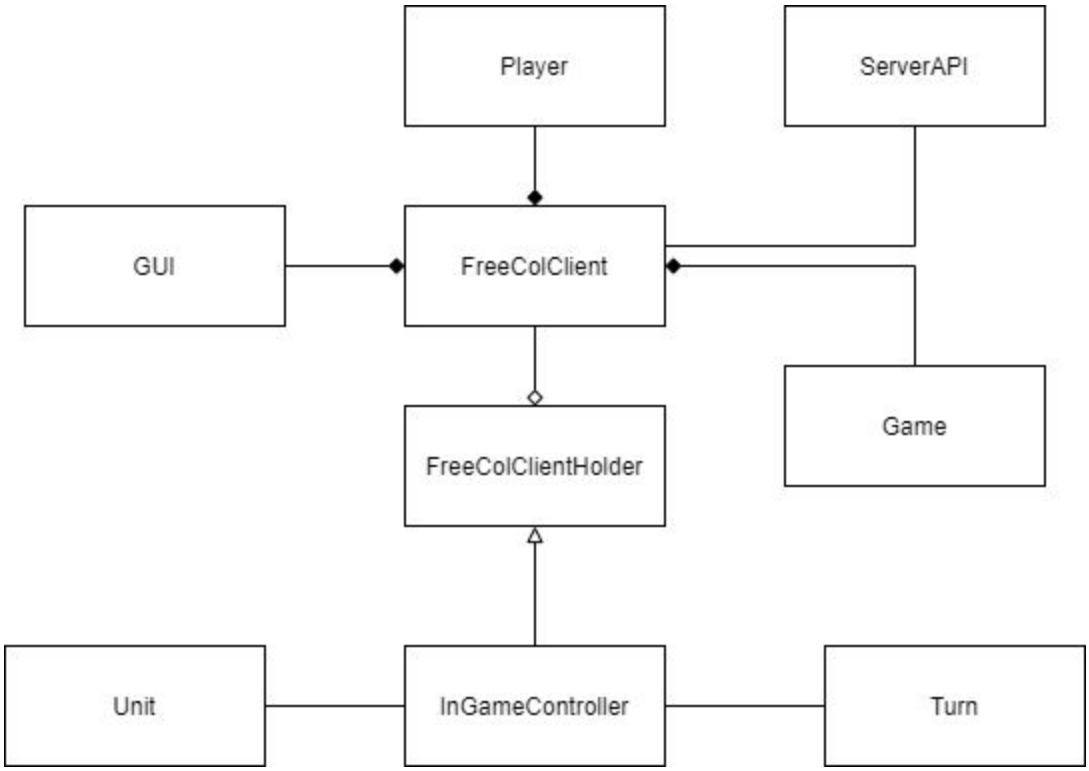

Figure 1.1 - Client side simplified class diagram

FreeColClient

This is the main control for the FreeCol client. It starts and keeps references to the GUI and other control objects in the client.

FreeColClientHolder

This class provides access to a FreeColClient's attributes and methods for subclasses. It is used as a parent for many of the client's essential controller classes.

InGameController

The client-side InGameController inherits from FreeColClientHolder and regulates general game actions such as claiming a game tile, saving the game, and updating GUI based on specific in-game events. For this feature, it initiates the client-side game logic and GUI tasks for ending a turn, and sends a message to the server.

Unit

The unit class represents a piece that can be moved around the map. A unit has an owner, a role, a state, a name, and a location.

GUI

This class encapsulates the graphical user interface for the client side application.

ServerAPI

This class controls the client/server messaging. It is in charge of connecting/disconnecting to the server side application, and sending requests to it.

Figure 1.2 shows a simplified UML class diagram for the server side component. A description for every class involved in the diagram is provided below.

Server Side

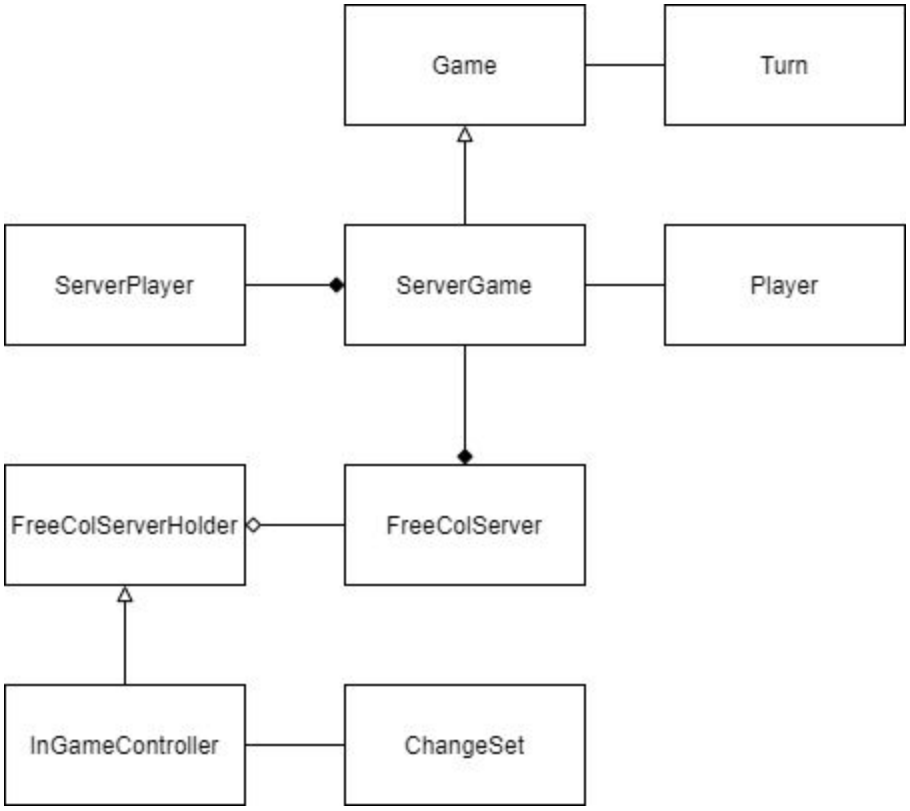

Figure 1.2 - Server side simplified class diagram

InGameController

The server-side InGameController inherits from Controller and controls the game state on the server side. One of its functions is ending the turn of a given player.

ChangeSet

This class represents the set of changes created on the server side component that are to be sent to the client side application.

FreeColServer

This is the main class for the server application. It starts and keeps references to all server objects and game models.

FreeColServerHolder

This class provides thread-safe access to FreeColServer for several subclasses.

ServerPlayer

This class uses the structure of the Player class, but also implements additional server related information, used by the server application to process player data.

ServerGame

ServerGame is the server-specific representation of the game. It inherits the general Game class that is used to store the game state in the client and implements some additional features, such as advancing the game to the next turn.

Common Models

Both the client and the server have their own copies of the game state, represented in various model classes that are common to both the client and the server. A relevant subset of these model classes include:

Player

This class represents a human or AI player. It stores the player's name, type, which nation it belongs to, and if it is alive.

Game

This class represents the main component of the game model. It controls the state of the game in the client side application.

Turn

This is a model class of a single turn in the game. It records the amount of turns that have passed since the start of the game, and can be converted to and from an in-game "date" (a.k.a. years passed since 1492).

Behavior

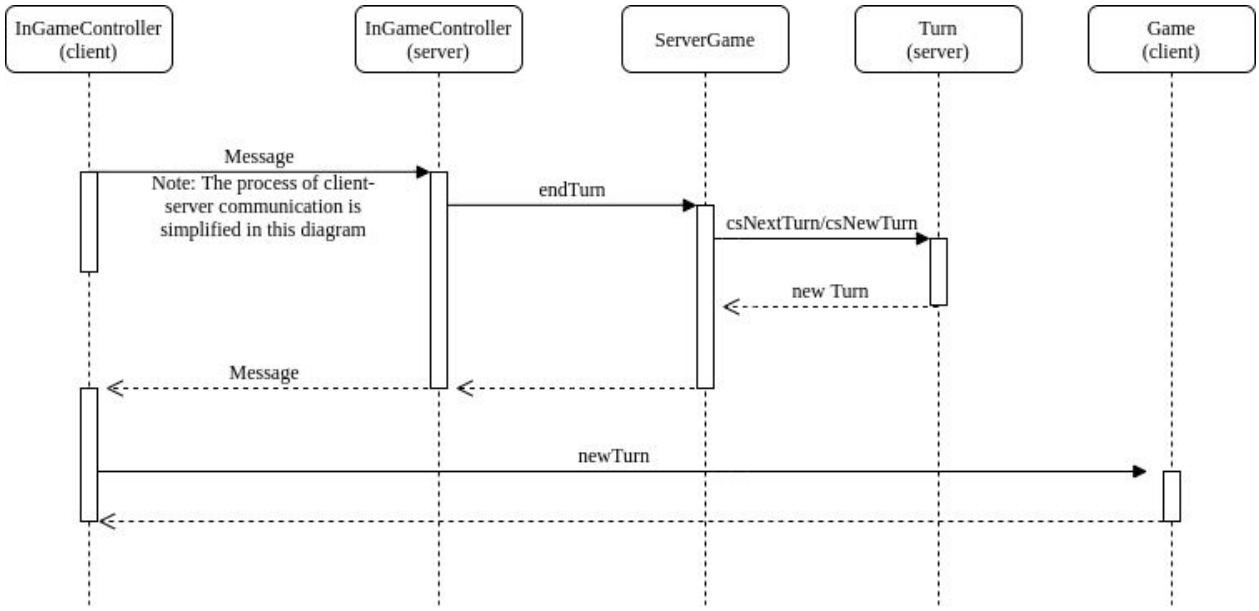

Figure 1.3 - Simplified sequence diagram showing interaction of major components

1. When the player clicks the “end turn” button, the action is passed to the client-side InGameController, which initiates the process of ending the turn. If applicable (i.e. the player has moves in this turn remaining), it shows the dialog box asking the user to confirm their action. On confirmation, it cleans outdated messages, restarts the selection cycle for all of the player’s selectable units, and notifies the server.

2. On receiving the end turn message, the server’s InGameController initiates the server-side end turn logic. It first checks for a winner, then cycles through players to find the next valid player who can start a turn. It will then call the current ServerGame object to create a new turn.

3. The current ServerGame object creates a new Turn object and sets its current Turn to the newly created one.

4. Upon successful creation of a new Turn, the server-side InGameController notifies all connected players of the new turn.

5. After the new turn message is received, the client-side InGameController will then update the client-side copy of Game with the new turn.

Feature 2: Map Generation

Introduction

The second essential feature of our project is map generation. Map generation is responsible for calculating the amount of tiles, assigning land or ocean values to each tile, and overall generating a board for the game pieces, such as players, to play on. The objective of the game is to find and conquer land regions, and declare independence. Without a map to explore, there is simply no game to play.

Structure

The map generator feature involves both major components of the program: the client side, and the server side. Below, we will go over every client side component class followed by the server-side components. Figures 2.1 and 2.2 show simplified UML class diagrams for the client and server-side components.

Server Side

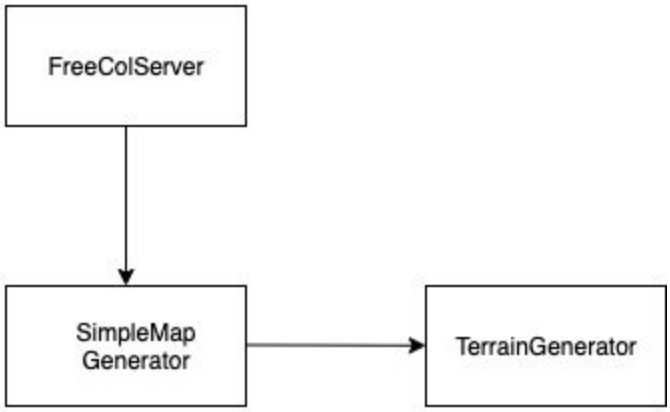

Figure 2.1 - server-side class diagram

FreeColServer

This is the main class that keeps references to all of the server objects, game model objects, generates the map, and is responsible for building and starting the game.

SimpleMapGenerator

The main feature of this class is to generate maps and the starting locations of the players before the game is ready for launch. This class preloads, nations, settlements, terrain, players, gets their respective tiles, and loads them into a map generator.

TerrainGenerator

This class loads the approximate number of land and sea tiles needed for the map, generates random ocean locations, rivers, mountain terrains, various land regions, and loads them into a Map class to be used to populate the game.

Client Side

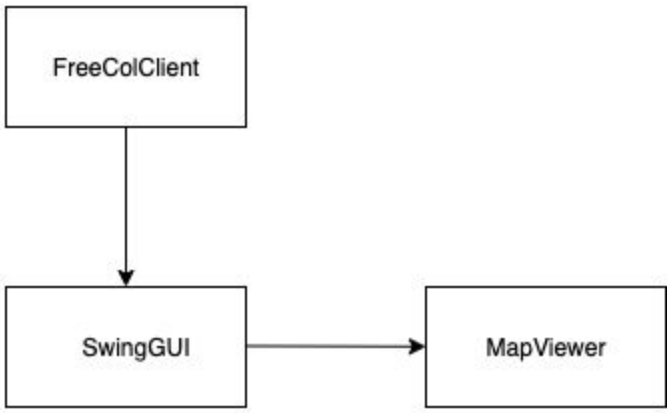

Figure 2.2 - client-side class diagram

MapViewer

MapViewer is a helper class for Swing GUI that is responsible for drawing the map. This class also displays a graphical representation of the map.

SwingGUI

This is a wrapper class that provides the overall functionality of GUI using Java Swing.

FreeColClient

This is the main class for FreeCol client and is responsible for starting the game and keeping references to the GUI and control objects. Swing GUI is instantiated under this class which utilizes Map Viewer.

Behavior

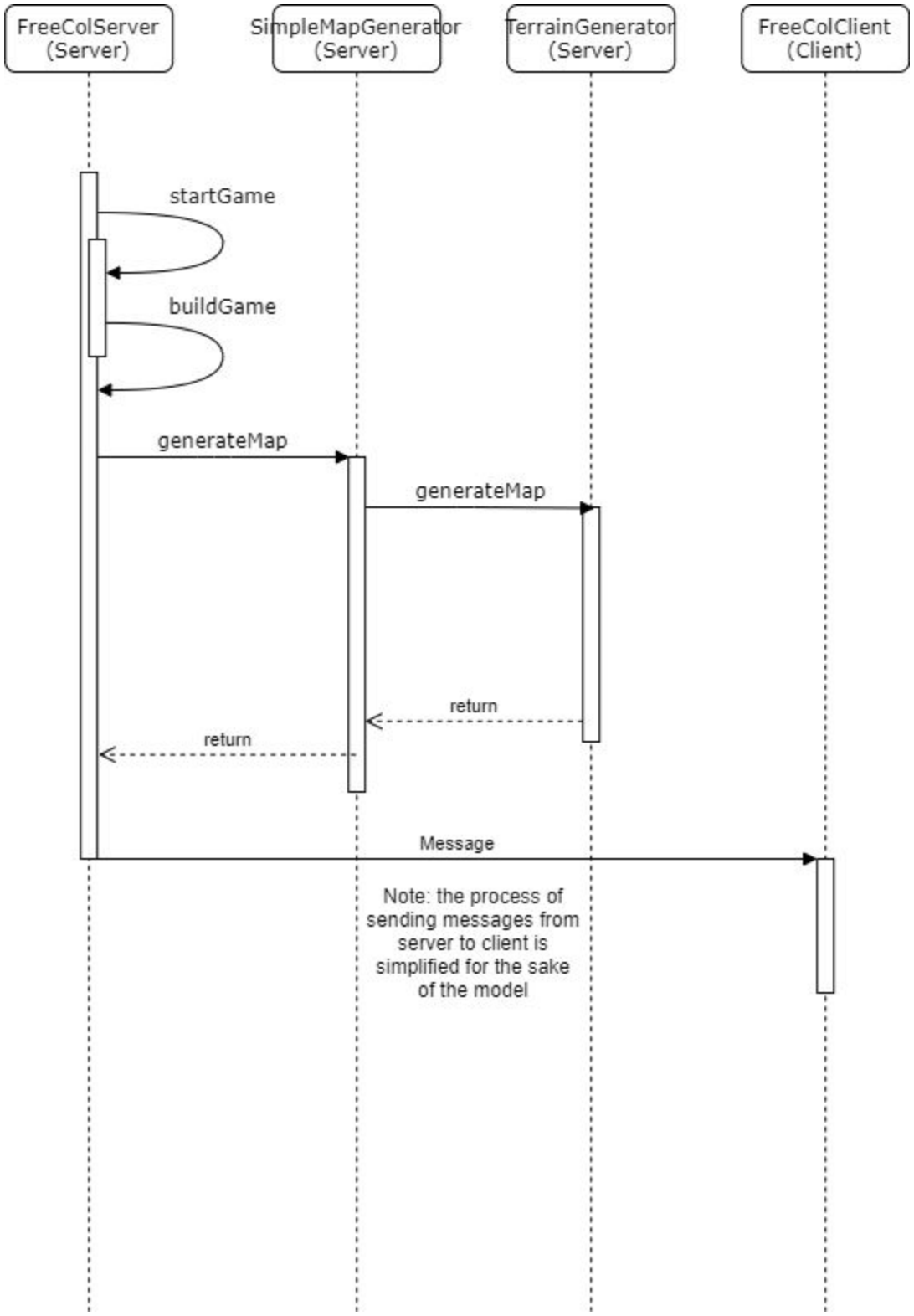

Figure 2.3 - Server-side Behavioural Diagram

- FreeColServer holds references to all the server objects and the game model objects, including information about the map. When a new game is created, FreeColServer will call startGame, which calls buildGame, which instantiates a new set of data, including a generated map through the SimpleMapGenerator class called by generateMap.

- SimpleMapGenerator class calls generateMap, which uses the TerrainGenerator class to do so. Once it generates the landscape, it populates tiles with players, Natives, settlements, and European units.

- The “generateMap” method in the TerrainGenerator class is where the terrain, such as whether a tile is land or ocean, is generated. Each individual tile’s type is based on the latitude and longitude of its location.

- Once all this information is generated, FreeColServer sends all the data to the FreeColClient class, where the map and other data can be accessed and utilized to calculate distance player has to travel, map rendering, and more.