FreeCol
Architecture, Social Context, Pull Requests/Issues

Introduction

This document describes FreeCol's as-implemented architecture, details the system's social context, and summarizes a few interesting pull requests and issues.

Architecture

Aside from a loose adherence to the client-server style, FreeCol has no official documentation or guidelines for the overall architecture of the program. In our exploration of the code, we mostly made use of top-down and systematic exploration by testing the control flow of the program against architectural styles such as MVC. We found that although observation of established architectural styles and patterns tended to be flexible, the package structure of the source code was generally a good categorization of the functionality of its associated elements.

Overview

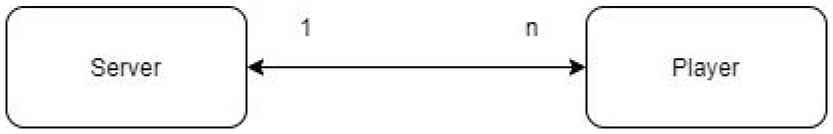

Figure 1.1 - Macro FreeCol Architecture

The two main components of the program are the client and the server. The server holds a complete model of the game state and connections to all players, as well as packages relevant to new game generation and AI control. The client holds a partial model of the game state specific to the player as well as a user interface. Whenever a new client joins the instance of the game, the new player subscribes to the game server. The game server will notify all subscribed players that a new player has joined, and the game will continue.

Client

The client can be roughly categorized into four main components: control, model, networking, and GUI, as shown in Figure 1.2 below.

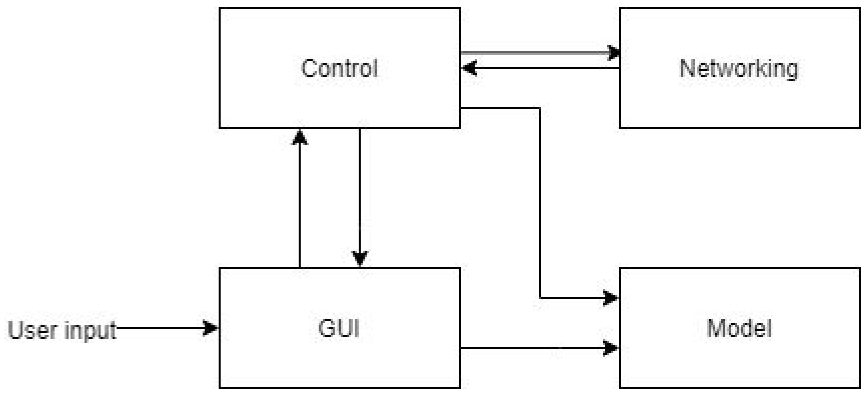

Figure 1.2 - Client-side architecture diagram

GUI

The GUI encompasses components involved in rendering and user interaction. The Canvas class holds the main GUI elements, including the map, display panels, and dialog boxes. The GUI also has associated listeners for mouse actions and changes to the window. The members of the actions package handle Interactions between the player and the GUI, and this package interfaces with both the GUI component and the control component.

Control

The control component acts as an intermediary between the networking component, the model, and the GUI. Game-relevant commands from the GUI, such as ending a turn, are handled through the control component before being passed to the server. Updates from the server also go through control before being updated to the model.

Model

The model holds data relevant to the game state, such as terrain, unit, and turn information. The client holds only a partial model of the game that is relevant to the player.

Networking

The networking component sends messages to the server. The control component handles when and what messages are sent.

Server

The server also has four main components: model, AI, networking, and generator, as shown on Figure 1.3.

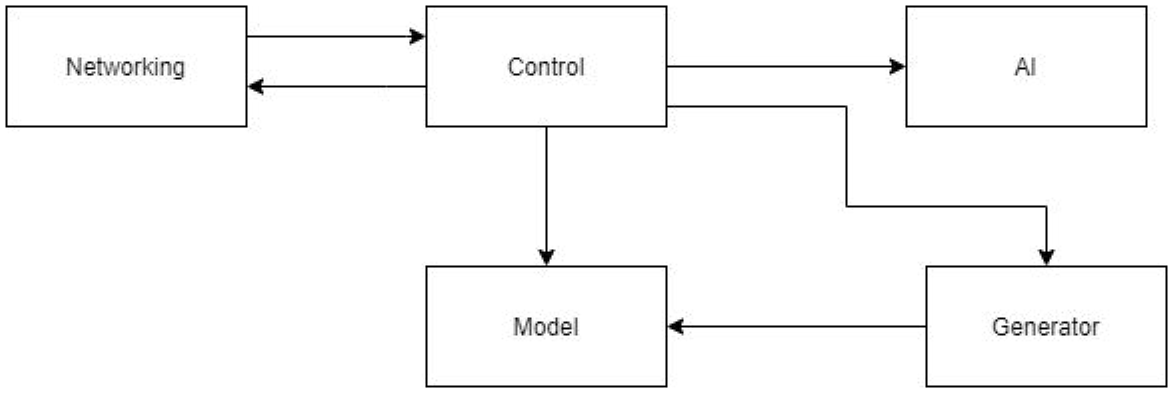

Figure 1.3 - Server-side architecture diagram

Generator

The generator component is a set of classes that are in charge of generating tiles, terrain, colonies, and maps using stored data models and files. It is only used during initialization of the game.

Model

The model component holds data relevant to the server status and game state, such as colony, buildings, trade, and loot. Unlike the client, the server holds the full model of the game.

Control

This component handles all server-side responses to clients' actions. It receives requests from the networking component, processes those requests, updates the server-side models, and sends the information to the networking component to notify players of those actions and results.

Networking

The networking component is a set of classes that sends and receives messages from clients.

AI

The AI component is a set of classes that represents the computer players play against, such as rival colonizers and natives. Although the AI component can be partitioned further into “AI Control” and “AI Model”, it is grouped into a single component as its functionality is mostly self-contained and has few notable interactions outside of itself. The AI component receives information from the control component and then calculates what moves the AI should execute.

Communication

FreeCol uses both a client-server architectural style and a publish-subscribe architectural style. Each player has their own client, which connects to the game server, but unlike a typical client-server relationship, the server also stores its connection to each player. When the current player client sends an action from the server, the server processes that action and updates its internal game model, then sends the relevant updates to all subscribed players.

Message and MessageHandlers

Message objects are the format in which data is sent between client and server. Different types of messages represent different actions, such as chatting, performing a game action, or ending a turn. Each type of message has an associated MessageHandler, which tells the receiver what to do with the message.

Social Context

FreeCol is a completely volunteer-based project run by fans of the game. It involves a robust community of not only developers but also players, artists, and translators, who communicate via mailing list and forum threads. It has been downloaded over 2 million times on SourceForge.

State of the Project

FreeCol is currently under development and has no available roadmap. However, the team first aims to create a standalone version of the game that has the same functionalities as the game Colonization, before adding new features to it.

- Last release date: Jan 20, 2020
- Recent commits: 52 since Jan 1, 2020
- No. open issues: 17
- Active developers: ~7

Project Standards

The project uses the standard Sun Java coding style. However, the team does not rigidly enforce it and tolerate some minor variants.

Contribution Process

While FreeCol has no available roadmap, there are available tasks and reported bugs in the issue section in the GitHub project repository and FreeCol's official forum. To work on an issue, fork the project, make changes, and then submit a pull request.

Before coding, an issue must be assigned to self or a comment must be posted stating that you are working on that issue. In case the issue does not exist, a new issue on either tracker must be created before a pull request is created.

Major changes to the code should be discussed on the official forum to justify them, if not, they will not be approved. This ensures that your work does not break the architectural style, and behaviour of the system.

Contribution Support Tools

TravisCI is used for building code in both the main branches and the one in pull requests.

Pull Requests

In this section, we will summarize relevant FreeCol pull requests. Because many of the more established developers on FreeCol commit directly to one of the project's feature branches, the number of pull requests is small compared to the project history and the number of commits.

Fixes bug that stops ship from sailing back to the colony

This pull request was created to fix a bug being caused by a null "map destination" variable, when a player's ship tried to return from Europe to a colony. It was initially solved by making sure the destination was properly added to the array when the ship leaves for Europe, that way the ship has a return point.

At first, one of the team members believed the issue wasn't fixed, but it turns out he was using a save file generated previous to the patch, so the contributor's changes wouldn't work since the ship had already sailed to Europe.

Fixes issue #23 whereby left clicks were ignored on goto mode

This pull request fixes a bug where if a left click event occurs, the goto mode is enabled, meaning that if a unit is currently selected and has available actions, it will move to the selected tile.

One of the core maintainers didn't understand certain sections of the code at first, so the contributor went over them, and realized that he could improve it, so he updated the pull request, before it was actually merged.

Unit will always return a FullLocationEntry

This pull request was opened to fix the pathfinding logic, since if the owner of a unit does not have a location set, one of the methods changed return null, ultimately throwing an exception.

Once one of the team members started reviewing the issue, they questioned the contributor why he was not adhering to the standards and was using warning suppressions.

Another team member pointed out that the fix is not correct, since it is changing the behavior of the initial starting point, and proposed a different approach. Ultimately, the pull request was not merged.

Preference added to flash the active unit

This pull request was added to create a preference setting that enables unit flashing when a unit is interacted with. The team's triager replies that unit flashing was actually an existing feature, but is now bugged, and doesn't occur any longer.

In the end, the pull request is not updated, because the contributor noticed that his enhancement actually causes visual glitches on his Mac, and decides he will not work on it any longer.

Update index.html

The contributor aimed to improve the main index page to make sure the project looks like it is still being worked on and not abandoned. Since the team member that replied didn't understand the purpose of one of the changes, the contributor went over his change by showing screenshots of the new starting screen and pointing out how he believed it looked better.

In the end, the pull request was closed, because the team had planned changes to add to the main page, but some of the suggested changes by the contributor were added to the patch.

Issues

In this section, we will summarize relevant issues to FreeCol's history.

Issue #9 Illegal attack move - privateer vs caravel

This issue was opened by a player who provided a filebin link of unknown content. After being asked by a contributor to open a SourceForge ticket with more information, the original poster refused, stating that “SourceForge is a dirty company and their platform is outdated, I don't want to use it.” The original poster later closed the ticket without further action.

Issue #24 General Feedback and Rant

This issue was opened by a player who was a long-time fan of the original 1995 Colonization game. It includes a long list of ways in which FreeCol deviates from the original game. A developer later submitted a pull request trying to address one of the points, which was to flash a unit upon selection. However, the pull request seemed to introduce some other bugs and was eventually closed by the submitter without merging. The original issue remains open.

Open Discussion (Source Forge) - Marcin - 2016-02-21 Ship Colony and Names

This issue was opened by a player who was bothered by the naming convention of ships and colonies for each nation. Initially, all ships and colonies were named “ship-1” and “colony-39” depending on how many you had made so far. The user then proceeded to take it upon themselves to submit pull requests to have the historical names used in the game. While doing so, an issue arose as the user wanted to implement names using Cyrylic for the ships and colonies. None of the other names and buildings used the fonts. The pull request was initially accepted but reverted so that only the Russian language pack would use Crylic, while the other languages kept their naming conventions. The issue was also brought up that Russia was not part of the colonization of the Americas at the time, making the game historically inaccurate. While controversial, the decision was to keep Russia in the game and use names of historical settlements in parts of East Asia as the names for the Russian settlements in FreeCol. The user then fully committed themselves to re-naming and correcting many of the names used in FreeCol, and their latest update was on 2020-02-13.

Bug #3146 (SourceForge) Suffered a big savegame corruption error

The issue describes a player’s game becoming slow after extended playtime. The cause was determined to be that the JVM was running out of memory. A quick fix was implemented to ask for more memory upon initialization, but the issue was left open to deal with the root causes of the memory problems.

mac app is missing executable

In this issue, it is reported that even though a JAR file is able to be generated in a Mac OSX environment, the JAR file can’t be executed, and therefore, the game can’t be run. During the discussions, one of the team members points out that this bug might have passed unnoticed since there has never been a Mac OSX since the start of development.

This is an issue that our team has faced as well, since one of our team members wasn’t able to run the game at first when using a Mac.