FreeCiv analysis

Guillermo López Alejos

March 28, 2004

Contents

1 Analysis **2**
1.1 Objectives definition 2
1.2 Requirement specification 2
1.2.1 AI Programs 2
1.2.2 AI client side interface 2
1.3 Analiced parts 3
1.3.1 Agents 3
1.3.2 *puppeteer* 3
1.4 Questions 4
1.4.1 General questions 4
1.4.2 Questions about *puppeteer* 4
1.5 Tasks 4
1.6 General subproject schema 5

1 Analysis

1.1 Objectives definition

Obligatories Create the AI client side interface.

Importants Empty

Optionals Make an open design that allows *puppeteer* to expand functionality in the future.

1.2 Requirement specification

1.2.1 AI Programs

We want the AI programs to be able to read and modify the game state (of course, according the game rules) through an interface.

1.2.2 AI client side interface

puppeteer The interface will be console-based. *puppeteer* is a console-based client-side interface that will do our dutty. Some properties are:

- *puppeteer* has to provide general game data.
- *puppeteer* has to provide agent related data.
- *puppeteer* will be used to build other stuff than client side AI.

There are manyfold functionalities to be implemented onto *puppeteer*, but this project can only handle the client side for AI players.

Language A language for input/output commands is to be defined. To acomplish the syntax check we'll use *bison*.

1.3 Analiced parts

1.3.1 Agents

The agents are programs that perfom some kind of task (move a unit from point A to point B, take care of a city's building queue, etc...). Some properties are:

- Agents are sorted hierarchically; higher the hierarchy of the agent, more complex is the task performed by it.
- Agents are event based. They can be wake-up/called back/notified/evented if either the client state changes or some other agent wake them.

1.3.2 *puppeteer*

puppeteer is a text based console that allows to "talk" directly to a client. It can be said that it's a set of two languages; the language of commands and the language of responces. Long terms goals for *puppeteer* are:

- Be a console to help debuging, logging, etc...
- Be a fullfledged textual client.
- The same for people with disabilities.
- Game demos and tutorials. If someone writes a logger which produces puppeteer commands, and if there is a way to make the civserver to behave deterministic then this log could be feed back to puppeteer and the game would be repeated again.
- Any kind of external artificial inteligence, like advisers, ai players...

puppeteer is implemented as an agent.

Why implement *puppeteer* as an agent? The main reason to implement *puppeteer* as an agent is that *puppeteer* should follow the game state changes. Thanks to the agents manager engine this can be easily done following agents rules.

1.4 Questions

1.4.1 General questions

- How is this client / server achitecture?
- How are the input / output packets handled? (see *packhand.h*).

1.4.2 Questions about *puppeteer*

- How is the lexical analysis realized?
- Where is it launched?
- What changes are to be done?

1.5 Tasks

- Define *puppeteer*'s input / output grammar.
- Integrate *puppeteer* into *FreeCiv* code.

1.6 General subproject schema

According to this analysis we can conclude that the project has two branches:

- Definition of *puppeteer*'s input / output languages.
- Integrate *puppeteer* into the *FreeCiv* code.

Index

bison, 2
puppeteer, 2
    commands language, 3
    responces language, 3