Infrastructure Proposal
Display Specifications
Red Team

Shivashankar Balu Ding Liu Ross Messing
Matt Post Arrvindh Shriraman

Last updated 10/22/2004 14:11

The display system will show key information about the current state of the game. This document describes the content and format of the information that the system requires, and when that information should be sent.

1 Display

The display will indicate each player's current status (free, in jail, or removed from the game). In addition, we may include other information, such as their in-game performance. Team affiliation for each player will also be clearly indicated by the display.

The display will indicate each team's "last known" current score and the number of objects in its possession. Any time the team's executive learns this information, it must send it to us in the format described below. We will also include information about triggered alarms, including where they occurred, who triggered them, and specifically whether a team member triggered his own team's alarm.

We will also display messages received from all executive machines and the current turn, clock, and period.

The manner in which this information will be laid out on the screen will be decided once we have more information about the infrastructure from the other teams. We plan to implement a graphical display in Java or the C++ QT library, which will present an intuitive and useful display.

2 Information requirements

The display system will include a multi-threaded server that sits on a TCP socket and waits for remote connections. All information will be communicated in short, descriptive messages over TCP/IP. These messages consist of a number of colon-delimited fields, and are case-insensitive but whitespace-sensitive. The

Requirement	Message Format
Clock	TICK:⟨round⟩:⟨period⟩:⟨clock⟩:⟨pause⟩
Player status	STATUS:⟨player-name⟩:⟨status-string⟩
Alarms triggered	ALARM:⟨area-name⟩:⟨player-name⟩
Dialog messages	DIALOG:⟨team-name⟩:⟨short-message⟩
Treasure count	TREASURE:⟨team-name⟩:⟨count⟩

first field indicates the type of message update, and the following fields are determined by it. Also, strings should be sent in UTF format. In Java, this can be done by calling the writeUTF() method on a DataOutputStream. Example code follows in the appendix.

We need the following information from each team's executive. Information should be sent immediately when it changes. The basic ideas and some of the formatting is borrowed from last year's Red team's specifications, although we believe that it is presented here more clearly.

All messages are case-insensitive, except for ⟨short-message⟩, which is displayed as-is. *All information should be sent immediately when it becomes available.* This applies especially player status and treasure information.

2.1 Clock ticks

As described by the green team, the clock server sends its information every second in the format described above. All of the arguments are integers. ⟨round⟩ is the current round, ⟨period⟩ is the current period, where 0 means "red", 1 "blue", 2 "green", and 3 "yellow". ⟨clock⟩ is an integer conveying the number of seconds remaining in the current period. ⟨pause⟩ is a 1 if the game has been paused, and a 0 otherwise. When the game is paused, the "PAUSED" is displayed for both the round and the clock.

This information is sent every second.

2.2 Player status

⟨player-name⟩ corresponds to the names described in the Blue team's grammar. Possible values are "anustup", "arrvindh", "ben", "piotr", "mike", "lior", "tanu", "pin", "ding", "wenzhao", "shiva", "matt", "jacob", and "ross". As mentioned earlier, values are case-insensitive. ⟨status-string⟩ is one of "registered", or "idle".

2.3 Triggered alarms

Alarm messages should be sent from each team's alarm system immediately after an alarm is triggered. ⟨area-name⟩ and ⟨player-name⟩ follow the syntactic conventions outline above. Note that there is an implicit synthesis requirement here: the alarm system knows when an alarm is triggered, but does not know who triggered it. Thus, in order to assemble this message, each team's executive

will have to collect the information from the alarm system and then send the alarm message.

The information must be sent as soon as it has been assembled.

2.4 Dialog messages

Dialog messages consist of all communication generated by any of your systems. ⟨team-name⟩ is one of *red, green*, or *blue*. The message is an arbitrary natural language string of no more than 40 characters (this constant may have to be adjusted). For example, "matt will steal from the blue area" or "anustup will self-destruct". Note that the ⟨short-message⟩ field is the only piece of the update strings that is case-sensitive.

Following the precedent of last year's red team, we plan to listen on port 9990 for incoming messages. Note that this has changed from the previous value of 4123 in order to fit with broadcast's from the Green team's time server.

2.5 Treasure information

We will also attempt to maintain information about each team's current score by requiring that updates be sent immediately when the information is known. Whenever any executive asks the "RoomQuery" question from the Blue team's grammar, that executive, upon receiving a response, must immediately send it to us in the format described above. ⟨team-name⟩, as before, is one of "red", "green", or "blue".

3 Database

In collaboration with the green team, we will be constructing a minimal (non-relational) database for storing information across sessions, reboots, etc. All information stored in and retrieved from the database is handled by our server, so that no outside access is permitted except through our interface. This ensures fairness and consistency, since the interface defines only a means of receiving information, not sending it.

4 Acknowledgments

We are heavily indebted to the excellent work prepared by the 2003 Kleptomania Red team, which consited of Madhur Ambastha, Jonathan Shor, Wei Jiang, and Viendra Maratha. Their work can be found at http://www.cs.rochester.edu/u/nelson/courses/csc_400/assignments/kleptomania/display_specs_2003.html .

Appendix – sample code

(You'll have to insert the appropriate exception handling).

```
import java.io.*;
import java.net.*;
...
Socket sock = new Socket(displayServer, 9990);
DataOutputStream out = sock.getOutputStream();
out.writeUTF(update);
```