Programmer's Guide to Cantr II

Programming Department
programming@cantr.net

October 19, 2011

Contents

1 Introduction

The code for Cantr II is originally written by Jos Elkink around October 2001 and has been under continuous development since then. About a year after its initial creation, an additional programmer got involved, Thomas Pickert, and over the years various programmers have had a very substantial impact on the code, in particular Cosmo Harrigan, Chris Johnson, and Grzegorz

Sojka. All these programmers were or are part of the so-called Programming Department of the Cantr staff. In this department, we have had many new recruits, but most of them never get sufficiently involved, mostly thanks to training that is too erratic and a consistent lack of documentation of any kind. This document is an attempt to rectify some of that situation by adding a basic framework of documentation.

This document is intended as a manual for programmers on Cantr II and will most likely be useful for other staff members of Cantr as well. It contains information on the server setup, the code repositories and the test environment. What it does *not* include is any substantial description of the Cantr source code itself, with the exception of perhaps some very general statements. Efforts are underway to develop such documentation as well, but this will be kept separate.

2 Communication

It is very important that we keep all programmers informed about what is going on in terms of our programming. So we have a number of channels to discuss or suggest issues or to keep people posted on what is going on:

- We use the forum[1] a lot for our regular communication. Although this is not always ideal compared to email, it is easier to keep a record so that programmers who join the team later can look back at existing discussions.
- We have a mailinglist[2] which is used for various automatic emails (such as source code changes) but also for communication between programmers.
- When making changes to the source code (see below), you should always add a comment to the Subversion commit, so that other programmers can see what changes you made. All code that is used for the game should also be in a source code repository - do not upload code directly to the production server without adding it to a repository as well.
- We now have this documentation project, consisting of this and other files, that should be kept up-to-date, to inform existing and new programmers of the details of our code and procedures.

[1]http://www.cantr.net/forum
[2]programming@cantr.net

- We have a bug report and ticket tracking system, but this is not really in active use.[3] But this might change in the future.

The most important thing to keep in mind is that the programming effort is a team effort, so we need to keep others informed what changes are made, what you are working on at the moment, what you think the priorities are, etcetera. Otherwise the team cannot function properly.

3 Subversion repositories

To keep track of all changes to the source code and to allow for multiple programmers to work on the same set of files, we make use of a version control system. The program we use is a freely available, open source program called Subversion.[4] The Subversion repositories are located on the main Cantr server and clients to access the source code can be downloaded for various computer systems. Subversion works well with command line tools, but there are also more GUI-based tools available, such as TortoiseSVN for Windows. The latter program integrates with Explorer, so that keeping your local copy of the source code in sync with the repository on the server becomes very easy. When installing TortoiseSVN, do make sure that you go to View/Settings, uncheck “Ignore line endings” and check “Compare whitespaces”. For many details, there is a very good manual freely available online.[5]

cantr_web	Main source code (PHP) of the game, including most server scripts
cantr_server	C++ source code of the server process that manages the Cantr clock
cantr_doc	Repository for documentation files, including what you are reading now
cantr_animal	C# code for a Windows-only client to help manage the animal population in Cantr (including binaries)

Table 1: Overview of Subversion repositories

Table 1 lists the various source code repositories of Cantr. Who has access to the various repositories varies, but we keep passwords constant across the

[3]http://www.cantr.net/flyspray
[4]http://subversion.tigris.org/
[5]http://svnbook.red-bean.com/

repositories. The cantr_animal repository is kept particularly limited in terms of access, because it contains some sensitive material.

To start working on code from a repository, the first thing you do is to *check out* the sources from the server. This means that you download a copy of the code and you also make clear that this is a copy you are going to use to make changes, so you allow for the possibility to *commit* those changes, which means that they become part of the actual code in Cantr. To check out the code, if you do not use a graphical client, you can use:

```
svn checkout --username <username> --password <password>
         http://www.cantr.org/<repository name>
```

This will create a directory for the repository in the current directory.

Once you have the sources you can make any changes you want. You need a few more commands, however, to fruitfully work with Subversion.

The following command checks what files have changed or are new:

```
svn status
```

For example, if I do this now I see:

```
?      oldtimers.php
M      js/events.js
M      tags.inc.php
M      page.events.inc.php
```

This means that I have one file in my repository directory that is not actually part of the Subversion repository itself, namely oldtimes.php. This file is only a temporary script and not part of the real code. The other three files have modifications locally and if I would do a commit, these changes would end up in the production environment.

If you see some new files (with a ? in front of the name) and you want to add those for the next commit, you can use the following command:

```
svn add <filename>
```

This command shows you all the differences between the current files on the system and the ones in the repository on the server:

```
svn diff
```

This command will commit the changes to the server:

```
svn commit <filenames> -m "<message>"
```

Note that I suggest to make the filenames explicit. This is not absolutely necessary - if you do not do this, all files that were changes will be committed. This is not a good idea in many cases, though. You might be sure about some changes but not about others, in which case you only want to commit the former. Another reason is that you *always* want to include a message describing the changes so it is better to keep the files for one change together in one commit and use another commit for another kind of change. This makes the log files much clearer.

Note that it is always assumed that code that is in the Subversion repository is ready for the production environment. The automatic updating of the production environment does not work anymore. When we want to manually update the production environment, someone needs to log into the server and perform the following steps:

```
su
cd /home/http/www.cantr.net/www
svn export --force --username <username> --password <password>
       http://www.cantr.org/cantr_web/ .
```

Most likely, this can be found using the history command.

Some programmers who do not have the access privileges to log into the server using SSH might still be allowed to use the Upload Tool,[6] written to directly copy files into the production environment. When uploading source files directly into the production environment, they also have to be committed to the Subversion repository - it will always be assumed that when we export the source code as described above, the production environment will still be in a valid state. Abuse of the upload utility, by uploading files not relevant to the game or by uploading code that illegitimately accesses game data, will most likely result in removal from Cantr staff.

4 Source code

Almost all code in Cantr is written in PHP, with a large mySQL database as a back-end.[7] Only some limited parts, such as the program that runs the clock and a standalone application for maintaining the animal population, are written in other languages, namely C++ and C#, respectively. In future, more programming languages might be used, but PHP is the primary language for our development.

[6] http://www.cantr.net/upload

[7] http://www.php.net/ and http://www.mysql.org/ are indispensable resources when coding for Cantr.

4.1 PHP conventions

With the PHP code we have very few conventions or rules within Cantr as to how to use it. There are some minor details worth mentioning, though:

- The Cantr server is a Linux server and Linux text files have slightly different line endings from Windows files. If you use a Windows editor that happily changes all line endings in the file and then you commit the changes, all lines will be reported as changed and we have no way of keeping track of what changes were really made. So use a text editor that can handle Linux line endings.

- PHP files that are used only when included in another file end on .inc.php, those that are called as standalone scripts end on .php only. Note that by far most files are called through inclusion in index.php.

The files that need open access for the web server are located in the www subdirectory. All other files, which are included in PHP scripts, should not be in this directory, but for example under the lib tree. Subdirectories in cantr_web:

www	Public scripts
www/css	Cascading style sheets
www/js	Public JavaScript files
www/irc	IRC applet (third party)
www/map	Map viewer application
lib	Included scripts
lib/smarty	Smarty template library (third party)
lib/smarty/templates	Template directory (filled by templatemgr.php)
lib/smarty/templates_c	Compiled template directory (filled by Smarty)
lib/server	Server scripts
lib/config	Configuration include files
tpl	Page templates for Smarty engine
storage	Files / scripts not currently in use

Table 2: Overview source code folders

4.2 C++ conventions

For the C++ code of the clock script - and in future perhaps other features such as animal management - quite a few more conventions are in place.

- Hungarian notation (with p_ for parameters)[8]
- Look at the existing indention and continue in the same vain; preferably, use emacs for coding
- Each header file should include the multiple inclusion protection with '#ifndef' etc.
- Avoid unnecessary including of header files; use forward declaration when that suffices
- Use references rather than pointers whenever possible
- Be careful with memory leaks: e.g. make sure every 'new' also has a 'delete' which will be reached even if an exception is thrown halfway.
- Use the keyword 'const' whenever possible.
- Initialise variables in the same order as they are declared
- Instance variables should always be private, unless otherwise necessary
- Database usage should take this form:

  ```
  CDatabase& db = g_pApplication->GetDatabase();
  CDatabaseResult& dbResult = db.SelectQuery("SELECT ...");
  while (dbResult.PrepareNext())
    {
      // Do stuff with the data
    }
  db.RemoveResult();
  ```

  or when it does not concern a SELECT query:

  ```
  db.Query("...");
  ```

[8]**URL?**

5 Smarty

It is generally considered to be good programming practice to separate functionality from presentation. From the start, the PHP code in Cantr and the HTML code have had a very close integration. At the initiative of Grzegorz Sojka, we moved to a system whereby we use Smarty Templates[9] for managing the presentation side of things. The HTML code is located in the Smarty template file (*.tpl) and the PHP code (*.inc.php) contains the functionality needed and this then passes on the necessary information to the Smarty processor.

Cantr is a multilingual game, which always creates additional programming complications. One of them is that we keep our Smarty templates in many different languages, so that the static part of the website does not have to be dynamically translated on each page load. To manage the translation of these static parts of the site, you need this utility written by Grzegorz:

```
http://www.cantr.net/templatemgr.php
```

Here you can process any or all of the templates after some text changed or some translations were processed or a new language was added.

It is good practice to keep all new additions to the Cantr system in line with this approach - so create separate templates from your PHP code.

The generated templates are stored in:

```
/home/http/www.cantr.net/www/tpl_compiled/
```

with a reference to this directory in the Smarty.class.php file.

6 System parts

Most of this documentation will be left to other documents, but the following are the primary sections of the source code:

- Game interface: a large set of PHP scripts that form the interface between the player and the database that holds all game information. Of course, most of the scripts are designed to help the player feel as if he or she is in a large virtual world, but in fact, the person is just staring at some presentation of information in a boring database.
- Server scripts: a number of PHP scripts that are run at regular intervals to manage all the automatic parts of the game, such as animals, food, deterioration, sailing, traveling, etcetera.

[9]http://www.smarty.net/

- Administration tools: as part of the PHP scripts that form the game interface, there is a substantial subset that is written purely for the administrators of the game. In particular the Players Department has a large number of tools to help track players and the Resources Department has a set of tools to generate new content for the game, such as new objects, clothes, animals, etcetera. Whereas the Programming Department is responsible for the development of the various types of objects in the game, the Resources Department is responsible for creating the actual content, the many variations on each item, and to some extent the global distribution of resources.
- Clock server: one small application not written in PHP runs continuously on the server to update the clock every few seconds.

7 Production environment

The production environment is the code on the real game server and the server that runs the clock on the same server.

Files can be uploaded directly into the production environment using the following page:

`http://www.cantr.net/upload`

This requires password access, which is identical to that of the main game interface.

8 Test environment

All programmers on Cantr should have access to the test environment. The test environment is a copy of the web interface, including a separate database, so that one can easily make changes and test them without affecting the real production environment. Well, this is what the name "test environment" suggests. The test environment and a little web interface for uploading files to it are located at:

`http://test.cantr.net`
`http://test.cantr.net/upload`

The latter requires password access. This is the same password as for the main game interface.

The test environment is a stripped version of Cantr and programmers are free to play there, it can be reset at any moment by any of the programmers.

When this is done, all history disappears and characters are at their starting point again. Resetting the test environment is done with a script in the source code, called

```
util.duplicate_database.inc.php
```

which can be called using

```
http://www.cantr.net/util.tester.php
```

This requires a password, which is the same as to the game in general. The script copies the structure of the main database and the structural information (e.g. object types and a few locations) and it generates some extra data (e.g. new characters). It does not update the source code to the Subversion one - this requires the same procedure as that for the production environment. When the database is copied, the player information of all members of the Programming Department, the Resources Department, and the Game Administration Board are copied. This means that the password is the same in the test environment as it is in the real game, but only for those people. All these accounts are given the same three characters (that is, characters with the same names and locations).

The test environment does not have a ticking clock or any automatic processes, such as processing food or projects. The reason is that even if we were to automate this, it would use unnecessary resources when nobody is testing anything and it would be too slow when someone is. So turns are manually processed (the clock simply does not tick in the test environment). This can be done using the script at

```
http://test.cantr.net/util.tester.php
```

Using this interface, the server scripts can be called directly. For example, to run the projects processing, one can enter the filename

```
server/server.projects.inc.php
```

The login and password are the same as in the test environment in general.

9 Database

The database can be accessed using the phpMyAdmin web interface, which is available at:

```
http://mysql.cantr.net
```

This page requires a password, which is identical to that of the main game interface.

A local copy of the test database can be downloaded, if you have access, from http://test.cantr.net/dbdump/dbdump.sql.