Ryzom Player Achievements

GSoC overview guide

app_achievements

Idea

Achievements in Ryzom basically consist of three parts. The Achievement itself is the general container. It may hold one or many Tasks which allow step-by-step progression. Tasks consist of one or more objectives that will have to be fulfilled in order to complete a Task.
This may look like that:

```
Wealthy!
        Obtain 1,000 dappers
                [ -----------        ]
        Obtain 5,000 dappers
                [                    ]
        ...
```

Implementation

We are using the Ryzom API which provides basic rendering and database access.

The app_achievements webIG app will load the achievement structure, as well as the progress of the current player. Once loaded, there are two rendering modes for either web or ingame. Rendering will take the previously loaded structure and render it to HTML code.

Installation

In order to install app_achievements you will have to set up the Ryzom API first. The second step is to edit the file conf.php and configure the app. Set up your achievements database on your server and you should be good to go.

Achievements API

The API for achievements allows external access to the system. There are two scripts, one to query the structure and one for actual player progress.

ach_struct.php	query structure
ach_progress.php?cid=	query player progress

Facebook

The current Facebook implementation is very simple. If enabled, the app will try to connect with a Facebook account. Successfully connecting the account will allow the app to post to your wall/timeline once your character gets a new achievement done.

app_achievements_admin

The admin tools offer functions for achievement administration as well as progress manipulation which can be interesting for CSRs.

Implementation

There are admin and CSR classes that extend the structural classes from the app_achievements system. They add functions for data manipulation.

Admin

Admins can edit the menu, achievements, trigger settings and translation.
Editing the menu is quite straight forward as well as translation.

Achievement settings

The achievement settings allow editing of the actual achievements, tasks and objectives.

Achievements
Achievements (as well as tasks) may have a naming template. This template will be applied on the names of all children (in case of an achievements, this means their tasks). Naming templates use [0-n] to define where certain strings of the children's name should be inserted. Use ; to separate strings in the children's name. Eg.:

Achievement naming template: Obtain [0] dappers
Task 1 name: 1,000
Task 2 name: 5,000

Achievements may also be tied to a cult or a civilization, be sticky (on top instead of alphabetical order) and may also have a parent achievement. The latter will cause the achievement only to be displayed if the parent achievement is already complete.

Tasks
Tasks basically are very simple. In addition to the self explanatory settings there are also "condition" and "condition value". This allows to define which objectives will have to be completed to fulfill the task. Eg.:

Kill all of the mobs listed below.
Kill any of the mobs listed below.
Kill 15 of the mobs listed below.

"All" and "any" won't require a value of course.

Objectives
Objectives have four display modes: hidden, simple, value/progressbar and meta.

* Hidden will hide the objective (eg. it would be useless when having a single task with one objective to output the same string twice like "Kill Urban the Vile". Still every task needs at least one objective.
* Simple will display the objective in a list like style.
* Value/progressbar will create a progressbar. The "trigger value" is used to define the required 100%.
* Meta will tie the objective to another achievement that has to be fulfilled. This is the only way apart from trigger scripting to actually track progress. "Metalink" allows to tie the objective to the desired achievement.

The trigger condition is similar to the "condition" of tasks.

Trigger settings

Trigger settings are the most important part. They attach Atoms to the previously defined objectives which can hold scripted code for the evaluation.

Scripting

Scripting triggers uses special keywords mixed with PHP code. Each definition inside the code that starts with VALUE, ENTITY or EVENT will be put inside a PHP function.

Here are some examples:

```
VALUE _money AS $money {

        CACHE blach AS $test;

        if($money >= 10000 && $test == 0) {
                RESET;
                GRANT $money UNTIL TIMER:3600;
                FINAL;
        }
        else {
                CACHE blach SET $money;
        }

        SCRIPT wealth($money) AS $res;

        if($res == "lol") {
                DENY;
        }
}

ENTITY _pos AS $pos {
        SCRIPT inside($pos,"majestic_garden") AS $region;

        if($region == true) {
                GRANT;
        }
}
```

CSR

Gamemasters may search for characters. Once a character is selected, they will see the current achievement progress of that character. They may grant or deny achievements/tasks or objectives in case the tracked progress proves to be incorrect.

WebParser

The WebParser is the core for detecting achievement progress. Currently it is set to parse a given XML file. The XML can be generated from the PDR files for the characters. A shell script triggering the system will be needed for the parser to do anything.

Extending the parser

The parser allows for various datasources to be attached to it. By default there is only the XML driver attached to it, which will use SAX to parse the XML file.

New data drivers can be attached anytime and will have the ability to send values, entities and events to the detection core.