RFC - Instance

Metacosm Development Team

Revision History

Revision M1July, 15th 2001
DocBook conversion, misc typos fixes, Properties and Predicates are now taken into account.
Revision 0.3 March 2001 Revised by: Ruffy
Metacosm framework section
Revision 0.2 March 2001 Revised by: Ruffy
"Horus birthday version" (corrections, examples)
Revision 0.1 January 2001 Revised by: Janselmeer and Ruffy
First version

This document describes what is provided by the framework Metacosm and how to create your own game. It gives the great lines of the instanciation.

1. License

Copyright © 2000,2001 Metacosm Development Team

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts.

The GNU Free Documentation License is available from the FSF (http://www.fsf.org) or you can write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

2. Metacosm

For more information on Metacosm, check our site: Metacosm (http://metacosm.org).

3. Metacosm framework

You can see the Metacosm framework as a generic way to create an online multiplayer RPG in a persistent world. It is independent of any game rule.

The Metacosm framework provides:

- the server (network connections, protocol)
- the world persistence
- the administration module
- the schedule system
- the perceptions system
- the Interactions system (Actions)
- the Entities system (Properties, Influences)
- the game time
- the game space (Places)
- the random generator
- the Quests system
- the game editors: ActionsEditor, EntitiesEditor and QuestsEditor
- some generic clients (text, 2D, 3D).

4. Working list

The following list summarises the operations to do to obtain a Metacosm instance.

4.1. Design part

1. Define ideas/concepts of the new instance
2. Choose a name for your instance
3. Extend your concept: add some background, to describe the world(s), to clarify rules
4. Define your rules.

4.2. Instantiation

1. Create Properties and Entities
2. Create Predicates and Actions
3. Create Quests
4. Test, test and test
5. Write some documentation
6. Release (and enhance and release and ...)

You can rely on Basic Instance or not.

5. Design part

You must have a fairly good idea of what you want before defining your rules. ’Defining your rules’ means choosing which attributes your world(s) inhabitants, items, etc will have, which actions they could do, etc.

Example 1. Example

In the StarWars(tm) RPG, rules are based on six face dice (D6). Each player has some skills (first aid, to pilot ships, to swim, to command, to dodge, to astrogate, languages, blaster, etc), divided in six categories (dexterity, knowledge, mechanics, perception, vigour, technique). They can be sensible to the force or not. They are (or should be) heroes. Game rules also define ships characteristics, E.T. powers, the fight Empire vs Alliance, etc. To determine if an action is a success, dice are rolled and sum should be greater than the difficulty factor. The more the sum is, the more you succeed.

Example 2. Example

In the ’Mage: the Ascension’ RPG, rules are based on ten face dice. Each player has three physical attributes (strength, dexterity, vigour), three social attributes (charisma, manipulation, appearance) and three mental attributes (perception, intelligence, cunning). Skills are consciousness, intuition, vigilance, driving, meditation, technology, law, riddles, computer science, sciences, etc. There are nine magics. Entelechy and willpower are really important. To determine if a magic action is a success, dice are rolled and dice greater than the difficulty factor count as successes. The more successes you have, the better you succeed. Dice equal to one are automatic failures and remove one success. If the number of successes is zero, action failed. If the number of successes is negative, action is catastrophic (or worse).

Example 3. Dummy instance

Basic rules could be to use a physical and an intellectual attributes (values from 1 to 6, chosen with 1D6). For physical (respectively intellectual) actions, you roll 1D6 and the result should be greater than the difficulty factor. For fighting (resp. playing chess), each fighter (resp. player) adds 1D6 with its own physical (resp. intellectual) attribute; the greater wins.

After defining your rules, you should know which Properties are needed, which Actions are possible and which Quests are allowed.

You won't have to define non-Action command (i.e. commands acting outside the world, like 'help', 'quit' or 'about'). Metacosm should provide all such commands. So you just have to define Action command (like 'swim', 'kill' or 'steal').

6. Instantiation

Now it's time to make your ideas come to life.

6.1. Create Properties and Entities

With the EntityEditor, you define which Properties and Influences Entities have and which Influences are available. After defining that in general, you'll have to create real Entities (instances) to fill your world(s).

Example 4. Dummy instance

Each Entity contains two Properties, named Body and Brain (unsigned integers, range (1,6), uniform distribution (1D6)). There is only one Type: 'Human' which brings Body and Brain.

6.2. Create Predicates and Actions

With the ActionEditor, you define Predicates and choose which Actions are allowed. You'll have to code Actions if you want to use Actions not available in the Basic Instance.

Example 5. Dummy instance

The 'swim' action is coded like that:

```
perform( ..., where, ...) { if (swimming in calm water) difficulty = 3 else difficulty = 8 if ( entity.Body + 1D6 < difficulty ) entity sinks else entity swims }
```

(this is just a basic example, a 'real' swim action should take into account members tied, possible equipment, possible amputations, drugs, ...)

6.3. Create Quests

With the QuestsEditor, you add new instances for already defined Quests patterns. It means defining when a type of Quest is possible, for who, etc.

Cf Quests-RFC for details about QuestsEditor

Example 6. Dummy instance

Chess quest: requires entity.Brain >= 4, success if an Action 'to play chess' succeed against Gary Kasparov, reward is a chessboard in marble. This quest can only be done one time. Wrestle quest: requires entity.Body >= 5, success if an Action 'to wrestle' succeed against Hogar 'Hand Crusher', reward is a bodyguard job at the palace.

6.4. Test, test and test

And have fun.

6.5. Write some documentation

You'll have to write doc, both for players (background and world(s) description, how to play, etc) and developers/maintainers.

6.6. Release (and enhance and release and ...)

Now you're trapped: you'll lost your sleep, coding or playing. And please don't forget to keep contact with us.

7. Basic instance

The 'basic instance' is an instance using predefined Attributes (Endurance, Power, Coordination, Charisma, Intelligence, Will, Magic, Essence). It provides some background and a world. Many Actions and Quests are already defined, based these Attributes. The 'Basic instance' is a RFC-based world (i.e. created using examples coming from our RFC). You are free to use it to base your own instance on. It's easier to enhance the "basic instance" than to completely rewrite an instance. But you'll have to if you want to use completely different Attributes.