Thoughts on Creating Better MMORPGs

By: Thomas Mainville

Paper 2: Application of Self-concepts

I. Introduction

The application of self-concepts to MMORPG systems is a concept that appears not to have been thoroughly researched. Taking this fact to mean that I have sufficient latitude to do so, I have decided in this paper to reference resources detailing self-concepts, and then to suggest ways in which they can be applied to MMORPGs. To this end I have narrowed the list of self-concepts to those I deem appropriate for application in the design of MMORPGs. This narrowed list includes the objective of being self-optimizing and the attributes self-awareness, self-monitoring, and self-adjustment.

That tomorrow's MMORPGs should be autonomic almost goes without saying. If these complex systems can self-manage, they will not require costly human intervention and maintenance. This will leave human resources free to develop new story and content. That is, it will leave the precious resource that his the human mind free to "manage the policy and not to maintain the mechanisms" (Van Roy, 2), with human minders having to interact with MMORPG systems only to introduce new content or to request implementations of new ideas and direction. In the case of new content, an MMORPG system might be designed so as to recognize the addition of a new expansion, plug-in, or module and to integrate the addition into its plan for self-maintenance. When handling external direction, the ideal MMORPG would accept simple instructions like "increase the goblin population" or "make gold more scarce in a particular region" and command appropriate agents to carry them out. In this paper I will discuss how such behavior can be handled through the application of self-concepts.

It should be noted that I am considering the application of these self-concepts within the context of a game world itself. MMORPGs might very well benefit from their application to in more (in my opinion) mundane areas, such as the allocation of computing resources such as memory, but this is not an area of particular interest to me. So, whenever I speak of self-management, the reader should remember that I am referring to self-management of the idealized virtual world itself, which is kept in balance in the same way our natural world is, through numerous natural processes (ecological, economic, environmental, etc.).

II. Pertinent self-concepts and their possible application

Out of the entire list of objects and attributes listed for autonomic systems in (Hinchley, 10), I have identified 2 objectives and 3 attributes I believe a truly self-managing MMORPG should have. In this section I explore each concept and attribute and suggest ways in which it can be applied to a MMORPG.

Attribute: Self-awareness
Self awareness, which Hinchley defines as awareness of internal state, is an essential ingredient in a truly self-managing MMORPG. If the game world is to be changed in response to emergent patterns or changes brought about by the actions of player characters (PCs) or NPCs, a controlling agent must have a view of the current game state. It is worth noting here that the internal state of a complex system like a MMORPG is probably better thought of as a collection of numerous states. Nevertheless, a monitoring agent would need to have a global view of system state and would be responsible for updating a number of state variables based on changes in that state.

Attribute: Self-monitoring
The concept of self-monitoring is defined in (Hinchley, 10) as the ability to detect changing circumstances. This ability was touched on in the previous section, and is described in more detail here. A global agent responsible for monitoring changes to the virtual world would obviously needed in order to ensure that the system's internal state (stored in a database tables, memory cache, etc.) is kept up to date. An example of how such self-monitoring would work can be derived from (Griffith, 1). In this article, Griffith proposes a global "creep" account for a particular type of enemy NPC. When I visualize such an account, the idea of an agent responsible for monitoring this account comes to mind. This agent would be sensitive to events such as an increase or decrease in the number of "creeps" as well as the defeat (and subsequent looting of) of PCs by "creeps". Such loot would be added to the amount of wealth these "creeps" would have in their global treasury. Such a monitoring agent would not take any direct action. Rather, it would rely on facilities for adjustment to respond to a given event.

Attribute: Self-adjustment
Continuing the example of self-adjustment from the previous section, facilities for self-adjustment would be necessary to ensure sensible responses to events detected by monitoring agents took place. I say sensible here because blithely reacting to such events might have deleterious effects. Taking again an example from (Griffith, 1), we can consider what adjustment might be made when a "creep" defeats a PC, taking that PCs loot. Having an event handler that simply fattened every creep's coin purse would be a simple approach, but not necessarily an appropriate one. A more intelligent approach would involve self-awareness. Before reacting to an event detected by a monitoring agent, a truly self-aware MMORPG would first consult its state (and possibly configuration) to see if the impact of such a change is consistent with current system goals. For example, assuming the game's economics-monitoring agent had detected unacceptable inflationary pressure on the game economy, a reaction to the wealth transfer event we are considering might simply be to delete the wealth.

Objective: self-optimization
Though I mentioned that more mundane details of a MMORPG's functioning were not of interest to me, this might not be true in those cases where they intersect with the functioning of the game world itself. If global behaviors emerge that dictate a reallocation of the game environment's limited resources, those resources should be

repurposed or reallocated as appropriate. Such behaviors could include certain types of NPCs are appearing or disappearing in greater numbers, certain geographical areas are experiencing heavier traffic than others, or certain subsystems, (such as that dealing with economics) being heavily taxed (due to a lot of financial activity, say). An interaction layer should be in place to request greater system resources to ensure such changes in behavior do not adversely affect the MMORPG's performance. The reader should note I am assuming here a finite number of resources, such as threads of execution or cache space, which can be allocated in effective ways to optimize game performance. That is, I am supposing that a MMORPG can be designed such that there is a correlation between virtual structures and processes and physical resources to service those structures and processes.

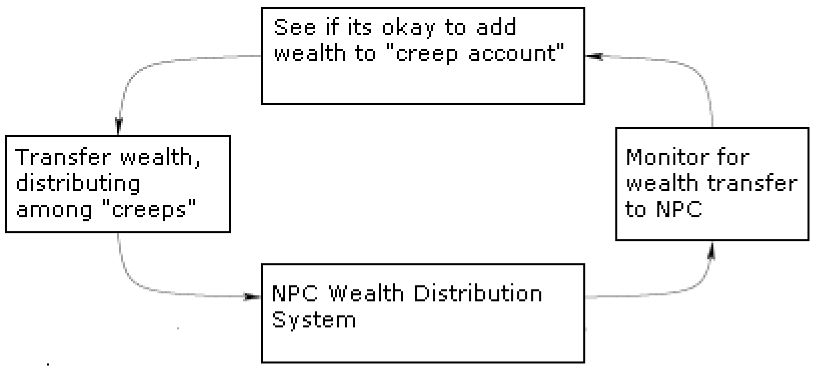
III. Implementation of self-concepts

It's interesting to consider the techniques that might be used to apply self-concepts to MMORPGs, which Van Roy identifies in (Van Roy, 2) as being among the complex, distributed systems which are good candidates for self management. In his paper, Van Roy provides some excellent insight into how to properly design self-reliant systems, focusing on three aspects of such systems: feedback loops, global properties, and general architectural framework. In this section I will visit each of these aspects, consider how each applies specifically to MMORPGs.

Feedback Loops

A feedback loop can be defined as consisting of three key elements that interact within a subsystem: "an element that monitors the state of the subsystem, an element that calculates a corrective action, and an element that applies the corrective action to the subsystem. For the purposes of [Van Roy's] paper [as well as this one], we consider these elements to be concurrent software agents that communicate by asynchronous message passing" (Van Roy, 4).

In the MMORPG setting I would define subsystems as those responsible for things like weather, ecology, economics (as I mentioned specifically in II), wealth and resource distribution, NPC generation and distribution, and so on. Accepting this definition, these subsystems would follow the pattern laid out by Van Roy in (Van Roy, 4), with the monitoring agent, calculator of appropriate action, and actuating agent mentioned in his paper being analogous to the state-monitoring agent, the agent determining "sensible action", and event handler I mentioned in II. To illustrate how the scenario from II, which deals with the transfer of wealth from vanquished PCs to NPCs might work, I've modified a feedback view from (Van Roy, 4), which is shown below.


See if its okay to add wealth to "creep account"
Transfer wealth, distributing among "creeps"
Monitor for wealth transfer to NPC
NPC Wealth Distribution System

Global Properties

Van Roy only mentions briefly global properties, but it takes no stretch of imagination to see their usefulness in a MMORPG system. Considering, for example, global weather and its effects on the NPCs that occupy a virtual world, it is possible to see how a single variable's value can have far-reaching effects. To illustrate, we can consider a Boolean global property called "isWinter", which would be set to true or false depending on whether or not it is winter in the virtual world. The simple changing of this variable from true to false could influence things like NPC migrations, availability of certain resources, the accessibility of different areas of the game world, etc. The global nature of properties such as "isWinter" would mean all subsystems (or agents) were aware of them at any time so that their actions could be synchronized based on these properties' values.

General Architectural Framework

As noted in (Van Roy, 13), an application such as a MMORPG, when consisting of several subsystems, has the potential to be very robust and fault-tolerant. This is due to the fact that each subsystem operates independently, linked perhaps only by global state and global properties as mentioned earlier in this paper. This opens up the possibility for configuration on the fly, wherein a system can activate or deactivate (or install and uninstall) portions of itself, as required. Depending on how nimble such self-configuration could be, its not inconceivable that an MMORPG could simply uninstall large parts of itself when the fell into disuse, retaining the ability to reinstall them should the come under demand again. An example would be the lair of monsters players have no interest in battling. The game could simply "dump" this area from its cache, thereby freeing up resources for more in-demand areas of the game. If a wayward player did happen to wander into this "offline" area, he might be treated with a "loading inactive area" message as the area was installed again. Given that this process was not unreasonably long, it might prove an acceptable way for a self-managed MMORPG to self-configure.

<u>IV. Conclusion</u>

As MMORPGs grow in scope, complexity, and intelligence, human intervention will become increasingly difficult, costly, and undesirable. For this reason, future MMORPGs should be developed with self-management in mind. Given that the goal of MMORPG development is to create virtual worlds which share many of the self-regulatory properties of our real world, every effort should be made to ensure they can operate harmoniously without outside intervention. After all, (with the exception of certain cases of external intervention, as claimed by our world's many religions), our world more or less functions in an autonomous way.

References

Griffith, Ken. Conservation of Objects in MMORPG Games. Game Research. 2006. <http://www.game-research.com>.

Hinchey, Michael and Sterritt, Roy 99% Biological Inspiration… Eds. Pan, Y, Rammig, F., Schmeck, H., Solar, M. Boston: Springer. 2006.

Van Roy, Self Management and the Future of Software Design. Department of Computing Science and Engineering, Université catholique de Louvain. 2006.