

PerlNomic: Rule Making and Enforcement in Digital Shared Spaces

Mark E. Phair¹ and Adam Bliss²

¹Dept. of Electrical Eng. and Computer Science
University of California, Irvine, CA 92697

²Dept. of Mathematics
University of California, Berkeley, CA 94720
mphair@uci.edu, abliss@math.berkeley.edu

May 1, 2005

ABSTRACT

In the United States and countries with similar legal systems, it is the role of judges to interpret the laws created by the legislature, thereby deciding individual cases and creating a body of case law that is considered on par with the original legislation. One need for judges arises out of the possible ambiguity of language in the wording of laws, leading to situations like the question of what constitutes “use of a firearm” in criminal law.

These kinds of ambiguities do not arise as often in programming languages as they do in natural languages, both by design and by necessity. Programming languages are designed to allow programmers to avoid ambiguous situations, although some issues like uninitialized memory and multithreaded systems can cause them to arise; neither of these situations will cause ambiguity, however, if the system is properly designed. In addition to the desire of language designers to discourage ambiguity, the deterministic nature of computers also works towards the same goal.

PerlNomic is a computer-based version of the rule-making game Nomic, and it takes advantage of this unambiguity to allow the Perl interpreter to stand as the only required judge. While it is clear where this approach stands on the long-standing conflict in legal philosophy between “the letter of the law” and “the intent of the legislature,” this lack of ambiguity can lead to a more transparent legal process. We present here some case studies of games of PerlNomic played over the last several years, specifically focusing on the interesting cases that have arisen, such as the development of economic controls. We then discuss scenarios in which a similar system could be applied to more complex virtual environments and communities and serve as the core of a multiuser environment without being its focus.

KEY WORDS

E-rulemaking, Virtual Environments, Distributed design, Online facilitation and community-building

1 Introduction

In *Smith v. United States*¹, the defendant attempted to sell a firearm to an undercover law enforcement officer. Instead of being paid in cash, however, he requested to be paid with drugs. Because the sale involved drugs, the prosecutors attempted to invoke a statute that stated, “Whoever, during and in relation to any crime of violence or drug trafficking crime . . . uses or carries a firearm, shall, in addition to the punishment provided for such crime of violence or drug trafficking crime, be sentenced to imprisonment for five years....” Did the legislation mean “use” as a weapon, or did it simply mean “to carry?” It was up to the court to decide.

Interpretation of law is the primary duty of judges, and ambiguity in statutes is a common occurrence. Different judges will interpret the statute in different ways, and the appeal system in the United States allows subsequent judges that hear a case to overturn previous judges’ findings. So, for many cases, the outcome depends on the judge that is assigned to it. In a truly democratic society, it could be argued, the laws approved by the citizenry would be applied in a consistent manner and any particular judge hearing a case should come to the same conclusion as any other; ideally, that decision would be in line with the intent of the lawmakers, but if consistency is desired, then the “letter of the law” should be the overriding philosophy. In a truly democratic society, every individual would be able to determine exactly what the consequences would be for any action, and they could make decisions knowing that the *stated* will of the populace will be carried out.

In digital shared spaces, such as message boards, massively multiplayer online role-playing games (MMORPGs), or chat rooms, governing is typically performed by one or more administrators of the community. Those administrators have the power to reward and punish users based on their perceived behavior, and, in most cases, they use that power to maintain order in the community. Administrators are not always benev-

¹113 S.Ct.1050 (1993)

olent, however, and it often happens that the only recourse of users that feel abused or slighted is to simply leave the community.

Other types of government do exist online. There are more libertarian-type communities in which any user can do as she pleases, but other users will typically have to take personal action to avoid being harmed by malicious or annoying members of the community. An example of this kind of community would be the USENET news system, where users must simply filter out undesired messages on their own. Some communities have more democratic systems in which users can vote on issues either directly or indirectly; for example, Google's [Google, Inc.] well-known PageRank system treats a link to a page as a "vote" towards that page's popularity.

In nearly all online communities, however, certain aspects are unchangeable by the typical user. Even if users can vote on attributes or request that the system administrator change a few rules, very rarely do users have direct access to the "laws" of the community. These laws are built into the computer programs that make up the system, and controlled only by the system's owners or administrators—creating what Shirky calls an "owned system" Shirky [2004]. In a true online democracy of the strictest definition, users would be able to vote on individual lines of computer code and thereby make changes to the laws that govern their community. Offline ("real") governments typically provide in their laws a means of modifying the laws themselves; Article V of the Constitution of the United States [Con, 1787] exemplifies this.

This paper explores the application of these principles to digital shared spaces. The presented system, called PerlNomic, allows its users (called players) to modify the very core of the system itself by making proposals in the form of computer code on which the online community can vote—if the proposal is accepted then the proposal is executed and the rules changed.

The system is currently based on the Web, and all transactions are currently carried out over the hypertext transfer protocol (HTTP) with all content either being plain text or hypertext markup language (HTML). It uses no images, sounds, animations, or any other non-text media. The system could be extended to include all of these and more, but that has not yet come to pass. The authors hope that this initial work will lay the foundation for richer online communities with the same governing structure, and, perhaps, that a similar philosophy might find its way into something like a MMORPG.

1.1 A Brief History of Nomic

In 1990 Peter Suber published a book titled *The Paradox of Self-Amendment*, whose aim was to explore a thesis of Alf Ross. Ross held (1) that a rule or law which authorizes its own amendment constitutes a logical contradiction (Suber agreed), and (2) that therefore no such authorization could be

legally sound (Suber disagreed). The book went out of print in 1997, and an online edition was released as "open-access" in 1998 [Suber, 1998b].

If the rankings of a Google search are any indication, the lasting effects this 21-chapter book on the fields of legal theory, formal logic, and philosophy are matched—if not exceeded—by the legacy of the last of its three appendices. Entitled "Nomic: A Game of Self-Amendment," this appendix sketched the rules for a game which was "intended to illustrate and embody the thesis of [the] book" [Suber, 1998a]. Suber had invented the game years earlier and a preliminary version of its rules and commentary appeared in Douglas Hofstadter's column "Metamagical Themas" in a 1982 issue of *Scientific American* [Hofstadter, 1982].

The essence of Nomic is captured in the motto that "to play the game is to change the rules." There is an Initial Ruleset which outlines some mechanics for play, but these rules are quickly changed by play itself. Under the initial rules, each turn consists of (1) proposing an enactment, repeal, or amendment of a rule (or of a previous amendment to a rule), (2) voting by all eligible players on the proposed rule-change, (3) applying the rule-change if it achieves unanimous approval, and (4) awarding the proposer a random number of points.

Its original context all but forgotten, Nomic has enjoyed immense staying power for a simple reason: whatever its effectiveness as a model of a constitutional government, it turned out to be *fun to play*. In the past 23 years thousands of players have run hundreds of games of Nomic all across the globe.

1.2 Overview

The present paper discusses the application of the principles of Nomic to virtual environments (VEs). In this vein, it first covers previous work dealing with legislation of VEs, then explains the core rules and ideas of Nomic, and finally covers the combination of the two. The primary focus of the paper is the particular amalgam of Nomic and computer technology known as PerlNomic. After a thorough discussion of the various games of PerlNomic that have been played, we conclude with some musings on the application of these same principles to more advanced VEs.

2 Related Work

This section covers the governing of VEs, discusses the original game of nomic and takes a brief survey of the previous work involving the realization of nomic in digital shared spaces.

2.1 Virtual Environment Rulemaking

Borders and Bryan [2001] argue that order and rulemaking will arise spontaneously out of online communities, but also

suggest that initial rulesets can lead to interesting experiments. Prominent among their suggestions for the properties of initial rulesets is the proscription against complicated initial rules; amazingly complex situations can arise from simple initial conditions, so there is no need to make the initial conditions any more complex than the absolute minimum.

Muramatsu [2004] explored the social regulation of so-called multiple user domains² (MUDs) from the points of view of both the players and the administrators. The particular types of regulation covered were direct action by administrators and automatic regulation built into the system by those same administrators. In the case of MUDs, administrators typically implement automatic regulation in order to reduce their workload, rather than to more transparently enforce the rules—although they will often use the rules as a way to “depersonalize” player regulation.

Muramatsu also notes the important fact that a behavior must be detectable to be regulated. A good deal of the work that goes into automatic regulation in MUDs is the design and implementation of detection systems. An example of a difficult-to-detect situation is what is known as “indirect player-killing” (indirect PK). PK of any type is the action of one player controlled by a human killing another player controlled by a human. Indirect PK is when a player causes another player to die by indirect means; for example, if a player uses a magic spell to cause another player to sleep until he starves to death, the direct action (sleep spell) did not directly cause the death, but it did begin a chain of events that led to the player dying. The MUDs studied by Muramatsu supported adversarial play (specifically, both direct and indirect PK), which requires an even greater degree of regulation to maintain an appearance of fairness for the players. Indirect PK is much more difficult to detect and disallow than direct PK, because the question of intent becomes important. It is clear that a player intends to kill another when the former begins attacking the latter with a samurai sword, and this kind of direct PK event is therefore much easier to detect and regulate than say, an act as subtle giving another player a poisoned potion (the giver might be intending the player to use the potion for some other purpose).

Many regulatory issues were left unresolved by administrators in the MUDs that Muramatsu studied, and the overwhelming reason was scarcity of programmers allowed to make changes. There were plenty of players who might have been able to write the regulatory code, but only administrators were allowed to edit the codebase, so the only ways to program more regulations would be to spend more of their own (volunteer) time, or to appoint more players to administrative positions.

Lessig [1999] explains how the Internet is already regulated programmatically, and explores the problems that can arise because of this. He also goes further, to show that such a large

piece of our lives is already regulated by code, and provides the (then upcoming) Y2K debacle as evidence. Lessig writes, “code is not elsewhere, and we are not elsewhere when we feel its effects.” Although repercussions of the Y2K bug never seemed to amount to much, there was a real threat that was stemmed by hours upon hours of re-programming or replacing outdated systems; if those steps had not been taken, it is probable that some of the predicted doomsday (if only the milder ones) would have come to pass. “Just as we should worry about the bad regulations of law, so too should we worry about the bad regulations of code,” Lessig warns, arguing for a balanced approach in which citizens keep an eye both on the code that regulates their lives and on the government that regulates that code.

2.2 Nomic: An exercise in ambiguity

Before examining PerlNomic in detail, it will be instructive to consider several interesting features of Suber’s original ruleset for Nomic. Suber included some meditations on these subtleties—from which we will borrow liberally—along with the ruleset in the appendix to his book.

The Initial Ruleset centrally features a two-tiered system of rules. It designates each rule as either *mutable* or *immutable*. Immutable rules cannot be directly amended, and they take precedence over mutable rules in all conflicts (and the rule that provides for this precedence is itself immutable). However, even the immutable rules are not really immutable in the literal sense, for they can be *transmuted* into mutable rules, amended, and transmuted back to immutable status. Transmutation takes a turn all to itself, and is intentionally difficult to achieve. Unlike an ordinary rule-change, whose voting requirement drops from unanimity to simple majority after the second round of turns (or sooner if, as is often the case, the players amend the mutable rule that governs this), transmutation continues to require a unanimous vote—and the rule that provides for this is immutable. Thus immutable rules, while not truly inviolate, take more time and concerted effort to alter. Suber sees this division as an essential feature of a self-amending system such as a federal government. A separate class of laws which are (1) logically superior to and (2) more difficult to amend than all other laws functions “to prevent a brief wave of fanaticism from undoing decades or centuries of refined structure. It is self-paternalism, our chosen insurance against our anticipated weak moments” [Suber, 1998a].

Immutable rule 101 of the Initial Ruleset³ provides that “All players must always abide by all the rules then in effect.” This strange rule presages a fascinating aspect that often appears in actual Nomic games, especially ones that have been running for a while: the game begins to reach out into “real life” and to have effects beyond the confines of what one would ordi-

²AKA multiple user dimensions, multiple user dungeons

³The Initial Ruleset contains 29 rules: immutable rules 101-116, and mutable rules 201-213.

narily consider “the game.” Many have put forth that rule 101 should be eliminated from the Initial Ruleset and made part of some “outside” body of rules or codes of conduct. Suber counters, though, that he included this rule *specifically* so that it could be amended. Its presence serves to remind the players that *they have control over the extent and jurisdiction of the rules* and that they can extend (or limit) them whenever they choose. For example, several games have involved rules which regulated the ability of their players to participate in other games of Nomic [Ryan, 2005]. More fanciful or arbitrary “real-life” rules are not uncommon, such as a requirement that each player must eat a hot dog once a week. In this way, Suber writes, the “line between play and non-play may shift at each turn, or it may apparently be eliminated.”

Mutable rule 212 outlines a process called “Judgment.” Any time the players disagree about the legality of a move or an interpretation of the rules, they may request that a *judge* be selected to settle the issue. The judge is simply another player of the game, and therefore no guarantee is made to his or her impartiality. The rules do not require that judges render rulings consistent with previous decisions, only that they “consider game-custom and the spirit of the game” foremost in reaching their opinion. A Judgment may be overruled by a unanimous vote among the other players; in this case another player is selected to be judge. Suber admits the necessity of such a provision, both in Nomic and in political legislature:

[R]ules will inevitably be adopted that are ambiguous, inconsistent, or incomplete, or that require application to individual circumstances not specified in the rule or not anticipated by the framers. “Play” must not be interrupted; some agency must be empowered to make an authoritative determination so that “play” may continue.

(We will soon see that PerlNomic does not suffer from this characteristic, but this advantage—if indeed it is one—comes at a price.)

Finally, the last mutable rule is

213. If the rules are changed so that further play is impossible, or if the legality of a move cannot be determined with finality, or if by the Judge’s best reasoning, not overruled, a move appears equally legal and illegal, then the first player unable to complete a turn is the winner.

The initial ruleset provides only one other way to win the game: by acquiring 100 points. But because the rule that provides for this is mutable, it is invoked very rarely. If one player begins to approach a score of 100, the other players will quickly unify and vote to increase the winning threshold. Thus, cunning players may attempt to introduce a paradoxical but seemingly innocuous rule-change, and—if it is approved—win by being unable to complete the turn.

2.3 Ideas on Mechanized Nomics

Shirky [2004] provides an enlightening discussion of the contrasts between governance in the offline world and the online world⁴, and he notes that as more of our discourse appears in “owned” environments (i.e., on web servers), citizens should develop a better understanding of how they can bring these environments more under their own control. He suggests Nomic-style systems as grounds for political experiments, and discusses various existing and possible Nomics with varying degrees of mechanization. Nomics based on Wikis and Blog-Nomic [Various, 2005] are at the less-mechanized end of the spectrum. These Nomics automate somewhat the process of rule-making, but neither do they provide for the automatic interpretation of those rules, nor do they allow the underlying process itself to be changed. At the other end of the spectrum are versions of Nomic in which the rules are interpreted completely mechanically, so that no human judgement is required. Shirky also discusses the possibility of versions of Nomic that provide a specialized, simplified control language that allows less programming-savvy players to participate as well.

3 PerlNomic

The authors have implemented a novel game called PerlNomic. In this section we will detail the nature of the game and present four case studies from previous rounds of play.

3.1 About the Game

PerlNomic runs on a web server using the Common Gateway Interface. Each web page comprises a Perl script which takes certain actions when requested by players. For example, one script allows players to submit proposals (which are arbitrary pieces of code); a second script allows players to vote on pending proposals; a third allows a player to activate (run) a proposal which has sufficient votes in its favor. PerlNomic is not turn-based; any player can submit a proposal at any time and players vote on whatever is pending. When proposals are activated, points are awarded to the player who wrote the proposal and to all the players who voted on it.

PerlNomic inherits many common characteristics from its namesake and progenitor, but also differs from it in several important aspects.

Like Nomic, PerlNomic was implemented with a kind of two-tier system of rules—but in a distinct way. The concepts of mutable rules, immutable rules, and transmutation could be realized literally on a web server filesystem by making certain scripts unwritable; then a `chmod` command would be needed before any proposal could alter them. However, we could find no simple and effective way to enforce programmatically any

⁴Additionally, he also asks if online governance can actually be fun, a question that is addressed later in the present paper.

restrictions on proposals that include `chmod` commands—such as requiring that `chmod` be the only command in the proposal, or that the proposal be passed by unanimous vote. Such restrictions would be necessary to carry Suber’s idea of self-paternalism into PerlNomic. Therefore, a PerlNomic proposal can patch any of the system scripts as easily as any other.

However, PerlNomic does include a separate set of *metarules*, which are not strictly rules of the game itself. They should be seen as *guidelines* for the behavior of players outside the game. This enables the metarules to regulate so-called *out-of-game* behavior, and their enforcement is left to the nebulous world of human judgment. The metarules themselves, however, are stored in a plain text file on the web server along with all the PerlNomic scripts—so they are no harder to modify than the rest of the game.

The script-based nature of PerlNomic imposes a fundamental limit on the scope and extent of play. There can be no rules about hot dogs in *this* game. The only objects and actions subject to legislation are those on the web server. A thin gray area does exist, though, between *in-game* and *out-of-game* actions. This is where the metarules are crucial: they sharpen the line between the game and the “real world.” Out-of-game actions explicitly forbidden by the metarules include attempts to hack the infrastructure: to obtain access to the web server, to packet-sniff for passwords, to engage in denial-of-service attacks, or to extort votes out of other players.

In one respect, then, PerlNomic actually improves upon the analogy Suber attempted to create between Nomic and the self-amending legislation of a country. The possibility of a military coup always exists in the governing of a people, and this eventuality is not modelled by Nomic. In PerlNomic, it is conceivable—though of course forbidden—that a player could storm the co-location facility where the web server is located, and through physical force apply a change to the code. Of course, by its very nature, PerlNomic fails to simulate the intricate machinations of a human-based justice system. While this means there are messy reinterpretations of ambiguous statutes, it does rob the game of *wiggle room*. In Nomic, a judge can talk her way out of a sticky paradox in the rules, but in PerlNomic, a bug in the code that crashes the system cannot be fixed.

3.2 Summaries of Completed Games of PerlNomic

The authors have initiated five games of PerlNomic in the past three years. In this section we summarize those games and present some of the interesting events that occurred and the lessons that we learned.

3.2.1 Game 1

PerlNomic 1.0 was stillborn; as soon as two players joined the game, a bug in the code caused the entire system to crash. This bug was fixed manually, by accessing the web server’s filesystem directly, and the result was called PerlNomic 1.1. The game was advertised to select undergraduate friends of the authors, and a total of 13 players joined. The players communicated using a Wiki (a web-based content-management system where any user may alter any page). The game began on October 13, 2002 and was hosted on a college academic computing server.

The game experienced some growing pains as certain bugs were discovered in the initial codebase that would allow for arbitrary control over the game. Two separate (but related) exploits arose in this way. The bugs themselves were discovered by two different players of the game, and proposals were put through to fix the bugs in due course. While each of the proposals were pending, however, one devious player (an author of both the current paper and the initial codebase) discovered a way to apply them to gain control over the game. He used the exploits to push through the pending proposals without waiting for them to be voted on. This action could be considered akin to temporarily declaring martial law when a country is being invaded: the player assumed emergency powers which were beyond those granted by the rules of the game, and used them to assure that nobody else could assume emergency powers for some worse reason.

The initial codebase featured an extremely rough user interface. The players quickly became dissatisfied with this and began to propose changes which would make the system easier to use. This provided an easy point of focus for early proposals; since the interface was so bad, it was easy to make better. Any proposal that improved the interface was likely to gain large support among the other players, and proposals which passed rewarded the author with points.

By the third week of the game, it became very difficult to pass any proposals. Several of the players had lost interest in the game, or were too busy to come vote on the pending proposals, or indeed had forgotten their passwords. Thus the simple majority of seven players was nearly impossible to achieve. The remaining active players drafted a mechanism to mark players as *inactive* if they had not voted in several days; then the number of votes needed to pass would be half the number of *active* players. An email was sent out to the inactive players requesting that they return long enough to vote in this activity mechanism. When it finally passed, the pace of the game picked up again.

A month later, as the school semester was winding down and the novelty of the game was wearing off, all the players but one became inactive. Safeguards had been put into place to prevent the “last player standing” from being able to win in such a scenario; however, the safeguards were incomplete, and he found a way to give himself 101 points and secure the win.

The game ended on December 30, 2002, after running eighty days.

3.2.2 Game 2

PerlNomic 2.0 began on June 7, 2003, using an updated codebase that incorporated the user-interface enhancements and bug-fixes from PerlNomic 1.1. The game was hosted on a free web-space provider, in what was probably a flagrant violation of their terms of service. Again, the game was advertised to select friends of the authors, in addition to all the players of PerlNomic 1.1. Communication among the players was carried out mostly over email, and also (in a few cases) through face-to-face conversations.

With a mind towards giving the game more appeal to a broader user-base of people less well-versed in Perl and the art of patching code, PerlNomic 2.0 was supplemented with a web application called the PatchMaker. The PatchMaker would allow any user to download a local copy of the then-current PerlNomic codebase; make changes to that codebase in his or her *sandbox*; make some changes to the files by editing them through a web form; create a file summarizing the differences thus induced between the sandbox and the live codebase; and craft a proposal which would use the `patch` program to implement these changes on the live codebase. However, for security reasons the PatchMaker could not let users *test* their changes, so players still needed to be able to download and test the alterations it generated to be sure that they worked.

PerlNomic 2.0 passed quickly into severe inactivity and wallowed in this state for several months. In October of 2003, two new players (also personal friends of the authors) joined the game and injected a bit more activity into it. However, the play at this point was all political. Very few proposals had much merit or content; the players were mostly trying to out-scheme each other to get the most points by proposing and voting. To liven things up, one player created a lottery system where players could enter points and have a chance at winning more than they risked. Ultimately this was not entertaining enough, and the game lapsed back into major inactivity in December of 2003. Sometime in January of 2003, the free web-space provider disappeared, taking the game with it, and no winner was declared.

3.2.3 Game 3

PerlNomic 3.0 began on September 13, 2004, hosted on a different free web-space provider and advertised to the players of the previous games. Unfortunately, the web-server was in Florida and the following series of hurricanes caused the game to suffer much downtime. When it came back, the game progressed in fits and starts, with much inactivity and forgetting of passwords. A total of six players joined, including one person whom the authors did not know personally (he found the web page through the Google search engine).

At one point, when all but one of the players were inactive, the last player decided to add a second account for himself. Because he now controlled two votes, he had much more power when the other players returned. Eventually, they managed to remove the extra player from the game and punish the offending player by removing some of his points.

PerlNomic 3.0 was the first game to see transferable points. This created an economy wherein points could be traded for votes or other actions (in- or out-of-game). However, as one player (through clever voting, point trading, and playing the lottery) approached a score of 100 points, the other players passed an “inflation” proposal which increased the winning condition from 100 point to 1000 points. When the game later began to die from inactivity, the last remaining player voluntarily reversed the inflation proposal, establishing the 100-point winning condition and allowing the 101-point player to win.

3.2.4 Game 4

PerlNomic 4.0 (PN4) began April 11, 2005. This incarnation was hosted on a commercial web hosting service and advertised heavily on USENET, Slashdot, Google Adwords, and several Perl and Nomic enthusiast websites. Accordingly, it was *extremely* popular, peaking at 81 total players (and another 218 requests for player accounts which never activated). Much inter-player communication occurred using the in-game comment system (which had been written in PerlNomic 2.0, but not used much since then); eventually one player voluntarily hosted a public web forum for further communication.

One of the interesting issues raised in PN4 was a dispute over the problems caused by the point reward system. The rules were initially set up to award extra points to people who voted against proposals that passed and to people who voted for proposals that failed, and, as a consequence, certain players would vote against good proposals simply to receive the extra points. This initially caused a major problem for “new user” proposals, because it would often happen that a new player could not join because a few people would refuse to let go of the extra points they would receive for voting against the new user. The players quickly decided to remove the extra points awarded when the subject of the vote was a “new user” proposal, but to still award the extra points in all other cases.

There was a small but vocal faction that argued for the continuation of extra point awards for dissent. They thought that this feature had been included in Suber’s version of Nomic for some good reasons. It added a little conservatism to the game, they argued, and made the voting process more interesting than would a flat award that ignored which vote was cast.

Out of the discussion came two proposals. The first simply proposed to remove the extra awarded points, thus removing all point-based incentive for voting against a proposal that was going to pass anyway. The second proposed that an attribute be added to players that tracked their so-called “ethos,” a measure

of how inclined the player is to support order. A player would receive positive ethos points for voting with the crowd (more order) and negative ethos points for voting against the crowd. The first proposal became too outdated to function before it received enough votes to pass (an indication that it did not have much support), but the second (called *Ethos*) passed quickly.

Soon after *Ethos* passed, a new proposal appeared, called *smite*, that would allow players with extremely negative ethos to be punished. The idea behind this proposal was that there is nothing wrong with voting against the crowd, but that someone who votes against the crowd much more often than they vote with the crowd is very likely just doing it for the points, and should therefore be reigned in a bit. *smite* received considerable support and passed. Even once the ability to *smite* players was in place, no one was ever *smited*. It is possible that the fear of being *smited* was sufficient to keep players from voting against the crowd too often. The winner of the game even had to keep this in mind when rushing for the finish.

PerlNomic 4.0 was won on April 26, 2005. An overlooked bug in a certain proposal allowed it to be activated without being cleared away; thus its proposer could activate it again and again and again. The win condition had been adjusted to 1000 points, just as in PerlNomic 3.0, so he had to activate the same proposal over a hundred times. Fearful of being *smited*, he arranged to switch his vote from “no” to “yes” and back again every few activations. This kept his Ethos relatively stable during the twelve-minute mad dash. Two other players realized what he was trying to do and switched their votes on the bugged proposal from “yes” to “no”—but this still left just enough votes for it to be activated the last few times.

3.2.5 Game 5

When PN4 was over, several players amiably congratulated the winner, then immediately began asking about a new round. Some complained on the forum [Various] that they were “going through Nomic withdrawal.” A day later, PerlNomic 5.0 was uploaded to an undisclosed location on the same web-server. It was advertised to *no-one*, yet one of the players from PN4 discovered it by guessing its URL on April 27, 2005. By the time of this writing, April 30, there were 19 players in the game.

4 More Elaborate Virtual Environments

It is theoretically possible to extend PerlNomic such that it resembles anything currently available on the Internet. Although the current interpreter is written in Perl, a player could very easily propose that other interpreters be used as well, or even write a proposal that creates a system that provides completely

unrelated services like an internet relay chat (IRC) server or even an e-mail server. A particularly excited player could, with a considerable amount of work, write a proposal that creates a MMORPG server that allows players to connect and interact in a fully 3D immersive environment.

A leap from the current system to a 3D graphical engine would be rather severe, and this change probably could not realistically happen in any reasonable amount of time. If a community using PerlNomic chose to extend it in this direction, the progress would most likely be incremental, with the primary initial focus being the integration of development tools. A sophisticated project like a server for a 3D environment requires much more sophisticated tools than the current system provides. These tools could, however, be developed or integrated from other existing open source systems, and a framework could be built that would enable much more lofty aspirations.

Although making these changes within the current framework of PerlNomic would be feasible, it is likely that the democratic nature of the community would slow development to the point that such a massive undertaking would be difficult. Committees are wonderful for small, incremental changes, but they are non-ideal for the crafting of large works. From a practical standpoint, any system that hoped to use 3D animated graphics should start with the basic parts needed for a minimally functional system as the base ruleset and should then allow a community to build on from there.

It might very well be asked why a community would even care to modify itself into a 3D environment from the simple and easy-to-understand text-based environment in which PerlNomic now resides. Imagine an online, immersive environment that was a simulation of the offline world. Inhabitants are allowed to make changes to the world by moving around and moving objects, just as we are used to doing in our day-to-day lives. One day, a kind of anti-Newton rises up and suggests that she never really liked gravity in the first place, and she proposes that the community just rid itself of the concept. Her proposal comes in the form of computer code that actually changes the way the simulation functions. People debate the advantages and disadvantages of the proposal, pick it apart line by line and even test it on their own, personal copies of the simulation. Eventually, a majority of the world decides that the proposal has enough merit to warrant the change, and it is activated. Suddenly, as if it never existed, gravity just stops. There would most likely be chaos, but everyone who took any interest in the proposal probably knew that there would be, and they might all simply welcome the experiment as a way of bringing a little surreal excitement into their lives. Regardless of their motivations for choosing to alter the world, it was *their* choice, and that is what a democratic world is all about.

5 Summary

The ambiguity of laws in the “real” world leads to uneven and often unpredictable application of those laws, which, it has been argued, impedes a truly democratic society. By applying the concept of a consistently enforced legal corpus to the rule-making game Nomic, a much more democratic system can be achieved, especially in the context of digital shared spaces. This paper has introduced PerlNomic, a novel realization of these concepts.

Shirky asked whether or not a game of this nature could be fun. PerlNomic has had international appeal, with over 3,000 visitors from at least six continents, and the authors hope that the ongoing interest in the game has answered Shirky’s question with a resounding “Yes!”

References

- U.S. constitution, 1787. Article V.
- Anonymous. PerlNomic - an experiment in cooperative coding. Available at <http://developers.slashdot.org/article.pl?sid=05/04/14/1643251>. As of April 2005.
- R. Max Borders and Doug Bryan. Experimental politics: Ways of virtual worldmaking. In *Fourth International Conference on Cognitive Technology (CT 2001)*, 2001.
- Google, Inc. Google technology. Available at <http://www.google.com/technology/>. As of April 2005.
- Douglas Hofstadter. About nomic: A heroic game that explores the reflexivity of the law. *Scientific American*, 246 (6):16–28, June 1982.
- Lawrence Lessig. *Code and Other Laws of Cyberspace*. Basic Books, October 1999.
- Jack Isamu Muramatsu. *Social Regulation of Online Multi-player Games*. PhD thesis, University of California, Irvine, 2004.
- Malcolm Ryan. Nomic history. Available at <http://www.nomic.net/~nomicwiki/index.php/NomicHistory>, 2005. As of April 2005.
- Clay Shirky. Nomic world: By the players, for the players. Available at <http://http://www.shirky.com/writings/nomic.html>, 2004. As of April 2005. Published first on the Networks, Economics, and Culture Mailing List.
- Peter Suber. Nomic. Available at <http://www.earlham.edu/~peters/nomic.htm>, 1998a. As of April 2005.
- Peter Suber. The paradox of self-amendment. Available at <http://www.earlham.edu/~peters/writing/psa/>, 1998b. As of April 2005.
- Various. PerlNomic fourm. Available at <http://pn.jgb.ca/forum/viewtopic.php?t=10>. As of April 2005.
- Various. BlogNomic. Available at <http://blognomic.blogspot.com/>, 2005. As of April 2005.