The Fixed World

A Constitutional Architecture for Persistent Virtual Worlds

Mats Heming Julner

October 6, 2026

Abstract

Persistent virtual worlds are commonly treated as software systems whose identity is inseparable from the implementations through which they are executed, rendered, stored, and operated. This paper argues that this conflation is architectural rather than necessary, and proposes a model in which a world's identity is defined constitutionally through its geography, mechanics, economy, and permissible state transitions. Once this distinction is made, implementations become replaceable, rendering becomes external, accessibility becomes architectural rather than retrofitted, and preservation shifts from maintaining software to preserving the constitution that defines the world itself. Development likewise changes direction: rather than continually redefining the world through new geography and mechanics, it improves the means through which an unchanged world is inhabited. The result is a virtual world capable of accumulating continuous shared history while remaining independent of the software through which successive generations experience it. We develop the architectural foundations of such constitutional worlds, identify which concerns the constitution must absorb and which it may leave to implementations, and examine both the consequences and the costs that follow from treating a virtual world as a place rather than as software.

1. The architectural assumption

Persistent virtual worlds are ordinarily developed as software systems. Their geography, mechanics, progression, and identity are inseparable from the servers that execute them, the applications that render them, and the institutions that operate and continually redefine them. As these implementations evolve, the world evolves with them: new regions replace old ones, mechanics are revised, expansions supersede earlier geographies, and entire worlds disappear when the software or the institution responsible for them ceases to exist.

These characteristics are usually regarded as inherent properties of persistent virtual worlds. This paper argues that they are instead consequences of a single architectural assumption: that a virtual world and the software that realizes it are the same thing. They are not. A world is not its renderer, its executable, its networking protocol, its storage engine, or the institution responsible for operating it. It is the place those systems exist to realize.

Each of those artifacts answers a different question. A renderer determines how the world is presented, an implementation determines how the world's rules are executed, a networking layer determines how information is exchanged, and a storage layer determines how state is preserved. None of them constitutes the world. A world consists of the geography, objects, inhabitants, and state transitions that together determine what can occur within it. It is the persistent place whose history is experienced by its inhabitants, regardless of the particular software through which that history is observed.

This distinction is largely absent from contemporary online games because every layer is typically produced, operated, and updated by the same institu-

tion. The world and its implementation consequently appear inseparable. They are not.

Once the two are separated, a different architectural model becomes possible. Rather than being defined through particular software, a world may be defined constitutionally: its geography, mechanics, economy, and permissible state transitions specified independently of any implementation. Software no longer determines the world's identity. It realizes an identity that already exists.

The consequences extend beyond implementation. Independent software can coexist without fragmenting the world. Rendering becomes independent of world identity. Accessibility becomes architectural rather than retrofitted. Preservation shifts from maintaining software to preserving the constitution that defines the world itself. Development changes direction: rather than continually redefining the world through new geography, mechanics, or progression systems, development improves the means through which an unchanged world is inhabited. The world therefore accumulates history rather than continual redesign. The question is no longer how software should operate a virtual world. It is how a virtual world should be founded.

1.1 Relation to existing work

That virtual worlds are places is not a new claim. Bartle (2003) treats world design as the design of a social environment rather than of a program; the legal literature (Lastowka & Hunter, 2004; Grimmelmann, 2004) describes their internal orders in the vocabulary of law; and ethnographic work records what accumulates inside them (Taylor, 2006; Nardi, 2010), including what becomes of that accumulation when an operator withdraws and the inhabitants outlast the world (Pearce & Artemesia, 2009). In all of it the world is understood as a place while the software is still understood as the thing that defines it. The preservation literature reaches the same boundary from the other side: Rothenberg (1995) established the general problem of digital longevity, and McDonough et al. (2010) and Guttenbrunner, Becker and Rauber (2010) examine games and virtual worlds specifically, converging on emulation, migration, and the maintenance of obsolete machines. Each of these is a strategy for preserving software.

Several established results are drawn upon rather than argued with. The layer separation of §4 applies information hiding (Parnas, 1972), and its treatment of concerns that age at different rates is close in spirit to Brand's (1994) account of buildings as assemblies whose components change on separate schedules. The requirement that independent implementations reproduce identical histories is the lockstep determinism long practised in game networking (Bettner & Terrano, 2001) and, more generally, the state machine approach to fault-tolerant services (Schneider, 1990). Section 4.2 stands on the classical re-

sults concerning ordering and agreement (Lamport, 1978; Lamport, Shostak & Pease, 1982) and on the necessity of choosing between consistency and availability under partition (Gilbert & Lynch, 2002). On governance, Ostrom (1990) shows that durable commons depend upon rules their participants cannot casually revise; Lessig (2006) that in software environments the rules are the architecture; and Hirschman (1970) that dissatisfaction is answered either by voice or by departure, and that the ready availability of departure tends to weaken voice, a tension §7 relies upon and does not escape.

The contribution of this paper is the constitutional model itself: that a persistent world's identity be defined by a specification of its geography, mechanics, economy, and permissible state transitions, and that every implementation, renderer, store, and operator be subordinate to that specification rather than constitutive of it. Six consequences follow which no prior arrangement makes available. Implementations become plural without the world fragmenting. Preservation becomes the preservation of a document rather than of an executable, and so ceases to depend upon the survival of the technology the world was first written for. Development inverts, improving the means of inhabiting an unchanged world instead of continually redefining the world. Agreement on history is identified as constitutional rather than infrastructural, which is where this model departs most sharply from the way distributed systems are ordinarily layered. Enforcement becomes a property rather than an activity, because an action failing to satisfy the constitution is not an illegitimate state transition but no state transition at all. And change is confined to succession rather than amendment, so that a revised rule set founds a second world instead of overwriting the first.

Prior work establishes that virtual worlds are places, that their software is difficult to preserve, and that distributed agreement is hard. None of it proposes an architecture under which the world outlives every part of the machinery that runs it. That is what is offered here.

2. The institutional world

Once a virtual world is separated conceptually from the software through which it is experienced, the architecture of contemporary online worlds becomes easier to describe. Nearly all persistent virtual worlds today are institutional worlds. Their continued existence depends upon an institution that retains ongoing authority over the world's operation. That institution develops the software, operates the servers, stores the canonical state, distributes updates, resolves disputes, and ultimately determines the world's future, including the economy its inhabitants depend upon (Castronova, 2005). Whether it is a company, a foundation, or a community project is secondary; structurally, the relationship is the same. The institution continuously governs the world.

This governance extends well beyond technical maintenance. The institution determines the world's geography by introducing or removing regions. It determines progression by changing mechanics, experience rates, and available content. It determines the economy by adjusting item sources, sinks, and production systems. It determines history by deciding which changes become permanent and which are discarded. It determines persistence itself by deciding when the world begins and when it ends. The world's continued identity therefore depends upon ongoing institutional decisions.

This arrangement has become so familiar that it is rarely questioned. Servers are expected to exist because an operator maintains them. Updates are expected because an operator changes the world. Expansions are expected because an operator creates new places. Shutdown is expected because an operator eventually withdraws support. The institution becomes inseparable from the world it operates, and the software and the world gradually become conflated. Replacing the software is assumed to replace the world; changing the implementation is assumed to change the world's identity; preserving the executable is treated as preserving the world itself.

These assumptions are understandable, but they are architectural consequences rather than necessities. They arise because contemporary virtual worlds are designed as continuously operated services, not because a persistent world inherently requires continual institutional control. The distinction matters: if the world itself can become the persistent artifact, the institution no longer needs to preserve the world's identity through continuous operation. Its role changes fundamentally. It becomes responsible only for the world's founding.

3. The constitutional world

If the world is understood as an artifact distinct from the software that executes it, the institution need not remain its continual operator. Its role may instead be limited to founding. Founding differs fundamentally from operation. To found a world is to establish the conditions under which it exists: its geography, its mechanics, its economy, its initial state, and its permissible state transitions. These become the world's constitution.

Once established, the constitution defines the world independently of any particular implementation. Software no longer determines what the world is; it determines only how faithfully the constitution is executed. This changes the role of software entirely: an implementation is no longer the world, but one means of inhabiting it. Different implementations may therefore exist simultaneously, written in different programming languages, executing on different operating systems, targeting different hardware, employing different optimization strategies, and presenting different user interfaces. If they execute the

same constitutional rules, they inhabit the same world. The constitution becomes the source of identity; the implementation becomes replaceable.

This places a constraint on what a constitution may contain. If identity depends upon independent implementations producing the same history from the same rules, those rules must be expressible in a form that admits faithful reproduction. A rule whose outcome cannot be recomputed identically by a second implementation cannot be constitutional, however desirable it might otherwise be; this is the discipline lockstep simulation has long imposed upon networked games (Bettner & Terrano, 2001). The constraint is not onerous, but it is real, and it binds before any question of how a particular world satisfies it. A constitution is therefore not merely a description of a world. It is a description written under the discipline of being reproducible.

Because the constitution defines correctness rather than software, implementations may compete in performance, accessibility, portability, maintainability, or visual presentation without fragmenting the world. Improving software no longer requires replacing the world it serves.

This relationship is familiar in other domains. A constitution is not identical to any one court. A musical composition is not identical to any one orchestra. A mathematical proof is not identical to any one textbook. Likewise, a constitutional world is not identical to any one executable. The world persists because its defining rules persist, not because a particular implementation does.

The separation also changes the nature of development. Under an operational model, improving software often requires changing the world itself. Under a constitutional model, improving software leaves the world unchanged. Performance may improve, accessibility may improve, networking may improve, and rendering may improve, while the world remains the same. The distinction is subtle, yet it fundamentally changes what is preserved: rather than preserving a particular implementation, the architecture preserves the world.

4. Separating the layers

Much of the architecture of contemporary virtual worlds follows from treating several fundamentally different concerns as a single artifact, contrary to the principle that a system should be decomposed along the lines its concerns actually divide (Parnas, 1972). The executable, renderer, networking layer, simulation, and world itself are commonly regarded as aspects of one system. In practice they often are. Architecturally they need not be. Changing how the world is presented does not necessarily change its rules. Changing the networking protocol does not necessarily change its geography. Changing the programming language does not necessarily change its history. Only changing the constitution changes the world itself.

Each layer may therefore evolve independently: renderers may improve without altering mechanics, networking without altering geography, storage without altering history, implementations without altering identity. The world becomes the stable artifact around which replaceable software evolves. This is the inverse of the conventional architecture. Traditionally the executable defines the world, and replacing the executable consequently replaces the world. Under a constitutional architecture, the executable merely realizes a world whose identity is already defined elsewhere; replacing it changes only the implementation.

The separation also clarifies an important misconception. Two implementations that differ internally are not necessarily different worlds. One may be written in Rust and another in C++; one may execute entirely on the CPU while another makes extensive use of hardware acceleration; one may be highly optimized while another prioritizes readability. If both execute the same constitutional rules, they inhabit the same world. Conversely, two implementations that appear visually identical may define different worlds if their constitutions differ. Identity cannot be determined by appearance, by implementation, or by programming language. It is determined only by the constitutional rules that define the world's permissible history.

Once these layers are separated, many assumptions surrounding persistent virtual worlds dissolve. A renderer is no longer mistaken for the world, nor an executable, nor a server. They become replaceable means through which a world is inhabited. The architecture shifts from preserving software toward preserving the world the software exists to serve. Three of these concerns deserve closer attention: rendering, because it is the layer most frequently mistaken for the world itself; agreement, because it is the layer most frequently assumed to lie beneath the world when it does not; and preservation, because it is what the whole arrangement exists to secure.

4.1 Rendering and accessibility

Once the world's identity is separated from the software that executes it, presentation no longer forms part of the world's definition. A renderer does not determine what the world is; it determines only how the world is perceived. The distinction is easily missed, because in most contemporary virtual worlds the renderer and the world are developed, distributed, and updated together, so changing the renderer appears to change the world.

Under a constitutional architecture these concerns are independent. The constitution defines the world's geography, mechanics, inhabitants, and permissible history; the renderer depicts that state. Different renderers may therefore present the same world in different ways, and two inhabitants need not perceive the world identically in order to inhabit the same world. One renderer may pursue visual realism, another may emphasize readability, another may

prioritize accessibility, and another may present the world entirely through text. These differences alter perception. They do not alter the world. Identity resides in state rather than in appearance.

One participant may accordingly experience the world through a conventional graphical client, another through a lightweight web renderer, another through a terminal designed for low-bandwidth environments, and another through software developed specifically for assistive technologies. Future implementations may employ rendering techniques or interaction models that do not yet exist. The constitutional world remains the same; only its presentation changes.

Accessibility follows directly. Once presentation is external, it becomes an architectural property rather than a secondary adaptation: the same constitutional world may be inhabited through interfaces designed for entirely different physical abilities or technological environments, without any of them holding privileged status. Accessibility becomes another implementation concern rather than a modification of the world itself.

Presentation is therefore free to evolve alongside hardware, operating systems, accessibility requirements, and interaction models without requiring the world to change. New forms of rendering may appear decades after the world's founding. Existing renderers may disappear. Others may be rewritten entirely. None of these events alter the world's identity, because continuity is established constitutionally rather than visually. The architecture permits independent experimentation in presentation while preserving a single shared world, and innovation occurs at the level of perception rather than identity.

4.2 Agreement on history

Storage and agreement are easily confused, and the distinction between them matters more than any other in this architecture.

How state is written to disk is an implementation concern. Formats may change, engines may be replaced, and a world may be stored in a dozen different ways at once without consequence for its identity. How participants come to agree on what happened is not an implementation concern, and treating it as one dissolves the claim made by the preceding sections.

The reason is direct. A constitution defines a world's permissible history. If two faithful implementations can arrive at different accounts of what actually occurred, whether through delay, partition, conflicting simultaneous actions, or a participant that reports falsely, then the world has no single history and the shared place the architecture promises does not exist. Whatever procedure resolves such disagreements therefore determines which of several possible histories is the world's own, and by the definition offered in §1 that makes it constitutional. The difficulty is not incidental: ordering and agreement among in-

dependent participants are hard problems with a long literature (Lamport, 1978; Lamport, Shostak & Pease, 1982; Schneider, 1990).

This commits the architecture to no particular procedure. A world with a single implementation and no adversaries requires almost nothing. A world inhabited by mutually distrusting participants requires a great deal. What the architecture requires is only that the procedure be named in the constitution rather than left to whichever implementation happens to be running, because an agreement rule that varies between implementations is not a rule about the world at all.

One consequence deserves stating plainly. A world that cannot determine what happened must not proceed as though it had. Where an implementation meets a disagreement it cannot resolve, halting preserves the world's identity and continuing destroys it, because a world that advances on an uncertain history has already become two worlds that do not yet know it. Stopping is recoverable. Divergence is not. The choice is the familiar one between consistency and availability when a network partitions (Gilbert & Lynch, 2002), resolved here in favour of consistency, because availability bought with a second history costs the world the only thing it was built to keep.

4.3 Preservation

The same separation changes what preservation means. The continued existence of the world no longer depends upon preserving any particular executable, server, or renderer. Implementations may disappear, programming languages may become obsolete, operating systems may cease to exist, and hardware may change: the standing difficulty of digital longevity (Rothenberg, 1995), which game and virtual-world preservation has met with emulation, migration, and the maintenance of obsolete machines (Guttenbrunner, Becker & Rauber, 2010; McDonough et al., 2010). The world remains reproducible so long as its constitution remains known.

Preservation therefore shifts from software archival to constitutional preservation. This is a considerably lighter obligation. Maintaining a decades-old executable requires maintaining the environment that executes it, and eventually the hardware beneath that environment. Maintaining a constitution requires only that its rules remain legible and that someone remains willing to implement them. The world's continuity depends not upon preserving yesterday's technology but upon preserving the rules that define its history.

5. One world

Once a virtual world is defined constitutionally rather than institutionally, there is no longer a technical requirement for multiple worlds. Persistent online games traditionally divide participants across independent servers, shards, or instances. These divisions arise from practical constraints: different servers

distribute computational load, separate databases simplify operation, regional deployments reduce latency, and independent worlds allow operators to scale by replication. These are engineering solutions to particular implementation architectures. They are not properties of the world itself.

A constitutional architecture changes this relationship. Because the world's identity is defined independently of any particular implementation, multiple implementations no longer imply multiple worlds. Implementations may increase, participants may increase, and renderers may increase; the world does not. Every participant inhabits the same geography: the same settlements, rivers, forests, roads, and landmarks.

This has consequences beyond technical simplicity. Places acquire shared meaning. A bridge becomes more than geometry; it becomes somewhere people remember crossing. An inn becomes more than a model; it becomes where people first met. A ruined tower becomes more than scenery; it becomes the place where something happened. The geography gradually transforms from content into collective memory.

Over longer periods, that memory becomes culture. Places acquire names, names acquire stories, and stories become traditions. Knowledge passes between generations of inhabitants because the places themselves endure. The world becomes familiar not through repetition of content but through continuity of place, a form of familiarity that cannot be authored, only accumulated.

This process is difficult to achieve when geography is continually fragmented or replaced. If identical towns exist across dozens of independent worlds, no single history belongs to them. If new continents continually supersede old ones, places struggle to accumulate lasting significance, and shared memory becomes attached to versions rather than locations. A constitutional world is arranged otherwise: its geography is founded once, and its inhabitants create everything that follows. History accumulates within the world rather than alongside successive revisions of it. The world becomes older without becoming different.

This changes the purpose of permanence. The objective is not to prevent change, but to ensure that change arises from the inhabitants rather than from continual redefinition of the world itself. A child entering the world decades after its founding should encounter the same mountain, the same river, and the same town that earlier inhabitants once knew. Their experience will differ because history has changed, not because the world beneath that history has been replaced.

5.1 Who executes the world

A world defined constitutionally must still be executed. Rules do not run themselves, and a constitution left unimplemented preserves a world in the sense that a score left unplayed preserves music: completely, and without anyone

hearing it. The preceding sections establish that no particular party is needed to execute a constitutional world. They do not establish that some party will.

The change the architecture makes here is narrower than it first appears, and more valuable for being narrow. It does not remove the world's dependence upon someone continuing to run it. It removes the requirement that this be a particular someone. Under an operational model a world ends when its operator withdraws, and it ends whether or not anyone still wishes to live there; the two conditions are unrelated. Under a constitutional model the world ends when nobody is willing to execute it, and willingness to execute has an evident source in willingness to inhabit. The dependency is not abolished. It is re-coupled to the only thing that ought to determine whether a world persists.

A second consequence removes a cost that institutional worlds have generally accepted as permanent. Where correctness is defined by an implementation, participants can be prevented from acting illegitimately only by an authority that detects them doing so; enforcement becomes surveillance, and surveillance becomes an indefinite contest between operator and participant. Where correctness is defined constitutionally and any party may verify it, an illegitimate action is not a violation to be caught. It is not a state transition at all. It fails to satisfy the rules by which history is computed and therefore never enters that history. Enforcement ceases to be an activity and becomes a property. The same reasoning disposes of the question of who or what is acting: an input either satisfies the constitution or it does not, and the architecture has no interest in whether a person or a program produced it.

What the architecture does not dispose of is cost. Agreement upon a single history is work, and that work grows with the number of participants whose actions must be ordered and verified. Institutional worlds meet this by partitioning: more participants are answered with more shards, more servers, more instances. The preceding section forecloses that escape deliberately, since partitioning by capacity produces precisely the fragmentation of place the architecture exists to prevent. A constitutional world therefore has a size beyond which it cannot grow while remaining one world. Where that limit falls depends upon how a given world is executed and not upon anything argued here; that a limit exists does not. A constitutional world is not merely permitted to be finite. It is required to be. Finiteness is a property of the architecture rather than a preference of its founder.

6. History replaces development

In most persistent virtual worlds, the principal mechanism by which the world evolves is development. New regions are introduced, mechanics are revised, progression systems are expanded, resources are redistributed, and entire geographies are replaced. The world's future is authored primarily through continual institutional intervention. History exists, but it develops alongside an ever-

changing foundation: places disappear, mechanics drift, progression resets, and what was once familiar gives way to successive revisions of the world itself.

A constitutional world follows a different principle. Its geography is founded once, its mechanics are founded once, its economy is founded once, and its constitution does not change. Nothing here claims that a founding can never be reconsidered; §7 takes up what happens when it must be. The claim is only that reconsidering it produces a different world rather than a later version of this one. The world does not accumulate content through continual redesign; it accumulates history through continued inhabitation. This distinction is fundamental. Content is authored, history is lived. Content is introduced deliberately, history emerges from participation. Content replaces what existed before it, history builds upon what already exists. The two are not interchangeable.

A world continually reshaped through development may remain engaging, but much of its continuity derives from the institution that repeatedly redefines it. A constitutional world instead derives continuity from the persistence of its founding. The author establishes the conditions under which history may occur; the inhabitants create the history itself.

Development therefore changes direction. The objective is no longer to extend the world's identity through continual additions, but to improve the means through which the existing world is inhabited. Implementations become faster and renderers richer, while accessibility, networking, and preservation all improve. The world itself remains unchanged.

This inversion alters the relationship between software and history. Software becomes transient and history becomes permanent. An implementation may be rewritten, a renderer may disappear, a programming language may become obsolete, and new hardware may emerge; none of these events erase what occurred within the world. The world's continuity no longer depends upon preserving the software through which it was once experienced, only upon preserving the constitution that defines its permissible history.

It also removes a property that persistent worlds have generally been assumed to possess. Such a world has no planned obsolescence. Its continued existence does not depend upon expansion cycles or continual reinvention, and its age becomes an accumulation of shared history rather than of successive software versions. The world grows older while remaining recognizably itself.

This is not an argument against development. Development remains essential; its object simply changes. Rather than continually redefining the world, development continually improves humanity's ability to inhabit it. The world grows older, not larger. Age has traditionally been difficult to preserve within virtual worlds, because a place repeatedly redesigned seldom becomes old; it becomes successive versions of itself instead. A constitutional world allows age to accu-

mulate without requiring the world beneath it to disappear. Its history deepens because its foundation endures.

7. The role of the author

A constitutional world changes not only the architecture of software but the role of its creator. Under continual operation, the author remains an administrator. The world depends upon the author's decisions: new mechanics are introduced, old mechanics removed, geography expanded, progression changed. The future of the world remains contingent upon continued intervention, and creation therefore never truly ends. The author remains responsible for determining what the world becomes.

A constitutional world follows a different philosophy. The author's responsibility is not to continually redefine the world but to found it, and founding is a finite act. The geography is established, the mechanics are established, the economy is established, the constitution is established. Once these conditions exist, the author's role changes fundamentally. The world no longer depends upon continued authorship; it depends upon continued inhabitation.

The distinction resembles that between creating a language and writing every conversation ever spoken within it. The creator establishes the structure; others create the history. Likewise, an architect designs a city, and its inhabitants determine what that city ultimately becomes (Alexander, 1979). The founding shapes possibility. It does not determine history.

This alters the relationship between authority and authorship. The author remains responsible for the world's beginning; the inhabitants become responsible for everything that follows. The world ceases to be an expression continually revised by its creator and becomes a place whose future no longer belongs to its founder. This is not an absence of authorship but its completion. The founding is not the first version of an unfinished work. It is the finished beginning of an unfinished history.

The same holds for everyone else. No participant, institution, implementation, or renderer possesses authority to redefine the world for the rest. Participation remains open, the constitution remains fixed, and history belongs equally to all who inhabit it. Authority over the world's identity is not transferred from the founder to anyone else; after the founding, it simply ceases to exist.

This arrangement has a price, and it is better stated than discovered. If the founding is final, so are its errors. A geography with a dead region, an economy with an unintended sink, a mechanic whose consequences emerge only after years of inhabitation: under continual operation each of these is a maintenance task. Under a constitutional architecture none of them can be repaired, because the authority that would repair them was deliberately dissolved. The model does not merely end institutional control at the founding. It makes the founding permanent, including wherever it was wrong.

This is a genuine cost rather than a disguised virtue, and it should be weighed against what the alternative costs in exchange: a world whose identity is perpetually contingent upon an operator's continued interest, and which ends when that interest does. Neither arrangement is free. The constitutional one trades the ability to correct for the ability to endure.

There is nevertheless a distinction between amending a world and succeeding it, and the architecture permits only the second. Because identity is defined by the constitution, a changed constitution does not produce a revised world; it defines a different one. The original is not abolished by being superseded. It remains exactly what it was, reproducible by anyone who holds its rules and inhabitable for as long as anyone cares to inhabit it. Founding a successor is therefore not an exercise of authority over the original, since it leaves the original untouched. It is simply the founding of another world, standing beside the first, between which inhabitants choose by choosing where to live.

This is a weaker mechanism than amendment, and deliberately so. Of Hirschman's (1970) two responses to dissatisfaction, the architecture leaves exit and removes not voice but its addressee: inhabitants may argue, organize, and persuade others where to live, yet no authority is competent to alter the rules of the world they occupy, and equally none is competent to alter those rules beneath them. Amendment changes the world beneath its inhabitants without asking; succession asks, by requiring them to move. It also leaves open a question the architecture does not settle: whether a successor may carry across what its inhabitants accumulated in the world it succeeds, or whether each founding begins empty. Both are coherent. The first preserves effort at the cost of making the successor partly dependent upon its predecessor; the second keeps each world wholly self-contained at the cost of discarding what its inhabitants built. The choice belongs to whoever founds the successor, and it is a constitutional choice rather than an implementation one.

Hirschman's own conclusion runs the other way, and the borrowing should not be silent about it. His concern is that available exit atrophies voice: where the dissatisfied may simply leave, an organization loses the feedback by which it would have corrected itself, and its decline accelerates. An architecture that dissolves the addressee of voice accepts that hazard deliberately. It is the trade of the preceding paragraphs in a different currency: the capacity for correction is surrendered in exchange for rules that cannot be altered beneath those living under them.

The success of such a world cannot therefore be measured solely by the ingenuity of its founding. It must ultimately be measured by whether later generations continue finding reasons to inhabit it. A constitution cannot create history. It can only make history possible.

8. Interval

The preceding sections describe an architectural model rather than a particular implementation. *Interval* is one realization of that model, and it is running: a founded and inhabited world whose constitution, implementations, and renderers are public at interval.place. Its geography, mechanics, economy, and founding conditions are defined constitutionally rather than institutionally. Independent implementations execute these rules and independent renderers present the resulting world; neither defines the world's identity.

The founding world consists of a single geography. Every inhabitant shares the same settlements, roads, rivers, forests, and landmarks, so history accumulates within one continuous place rather than across successive revisions or independent shards. The world is finite, in the manner §5.1 requires, and intentionally so. Its geography is not expected to expand, and its mechanics are not expected to evolve through continual institutional intervention. Its constitution defines the world once.

Interval is live at interval.place as a founding under test. Its constitution is still being revised, and under this architecture each revision founds a new world rather than amending its predecessor, as §7 requires; the worlds running there now are those test foundings. The endurance claimed in §6 can only be shown from a fixed point, and so it is tied to one: from the founding that will be recorded at interval.place, under the rules hash published with it, the constitution is fixed, and any later revision stands beside it as a separate world. Because identity reduces to that hash, anyone may check whether the commitment holds. From that point the demonstration asks nothing of the author except abstention.

Subsequent development instead focuses upon the software surrounding that world. Implementations may become faster, renderers richer, accessibility better, networking more capable, and preservation more robust, while the world itself remains unchanged. Interval therefore distinguishes between improving the means by which a world is inhabited and redefining the world being inhabited. The former remains open; the latter belongs to the founding. The objective is not to preserve software. It is to preserve a place.

9. Conclusion

Persistent virtual worlds have historically been treated as software systems operated through continuous institutional control. This paper has argued for a different architectural distinction. A world is not identical to the software that renders it, executes it, stores it, or transmits it. It is the constitutionally defined place whose history those systems exist to preserve.

Once the distinction is drawn, the architecture changes fundamentally. Software becomes replaceable, implementations become independent, render-

ing becomes external, and accessibility becomes architectural. Development shifts from redefining the world to improving the means by which it is inhabited. The world itself remains continuous: its geography endures, its history deepens, and its identity persists independently of the technologies through which successive generations experience it. None of this requires additional mechanisms beyond those described here. It follows simply from separating the world from the software through which it is experienced.

A constitutional world is therefore not preserved by maintaining particular software. It is preserved by maintaining the constitution that defines its permissible history. Everything else remains free to evolve. Most online games preserve software while replacing worlds. A constitutional world preserves the world while replacing software.

References

Alexander, C. (1979). *The Timeless Way of Building*. Oxford University Press.

Bartle, R. A. (2003). *Designing Virtual Worlds*. New Riders.

Bettner, P., & Terrano, M. (2001). 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond. Game Developers Conference, San Jose, CA.

Brand, S. (1994). *How Buildings Learn: What Happens After They Are Built*. Viking.

Castronova, E. (2005). *Synthetic Worlds: The Business and Culture of Online Games*. University of Chicago Press.

Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2), 51–59.

Grimmelmann, J. (2004). Virtual Worlds as Comparative Law. *New York Law School Law Review*, 49, 147–184.

Guttenbrunner, M., Becker, C., & Rauber, A. (2010). Keeping the Game Alive: Evaluating Strategies for the Preservation of Console Video Games. *International Journal of Digital Curation*, 5(1), 64–90.

Hirschman, A. O. (1970). *Exit, Voice, and Loyalty: Responses to Decline in Firms, Organizations, and States*. Harvard University Press.

Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7), 558–565.

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3), 382–401.

Lastowka, F. G., & Hunter, D. (2004). The Laws of the Virtual Worlds. *California Law Review*, 92(1), 1–73.

Lessig, L. (2006). *Code: Version 2.0*. Basic Books.

McDonough, J., Olendorf, R., Kirschenbaum, M., Kraus, K., Reside, D., Donahue, R., Phelps, A., Egert, C., Lowood, H., & Rojo, S. (2010). *Preserving Virtual Worlds: Final Report*. University of Illinois at Urbana-Champaign.

Nardi, B. A. (2010). *My Life as a Night Elf Priest: An Anthropological Account of World of Warcraft*. University of Michigan Press.

Ostrom, E. (1990). *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press.

Parnas, D. L. (1972). On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12), 1053–1058.

Pearce, C., & Artemesia (2009). *Communities of Play: Emergent Cultures in Multiplayer Games and Virtual Worlds*. MIT Press.

Rothenberg, J. (1995). Ensuring the Longevity of Digital Documents. *Scientific American*, 272(1), 42–47.

Schneider, F. B. (1990). Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Computing Surveys*, 22(4), 299–319.

Taylor, T. L. (2006). *Play Between Worlds: Exploring Online Game Culture*. MIT Press.