



Virtual World Architectures

Earlier this year, in “Next-Generation Virtual Worlds: Architecture, Status, and Directions,”¹ I described the promise of 3D virtual worlds to complement the Web with 3D models of virtual places that are fanciful or that model and mirror the real world. Marketplace evolution is one way to wait and see if and how this will come about. Another approach is to identify current limitations of virtual worlds, deconstruct and study their architectures, and consider how to evolve them to realize their promises. Here, I discuss nine articles that explore architectural issues related to virtual world evolution. Although there isn’t room in this special issue to run all the articles, they all warrant introduction as interesting examples of the state of the art in this field.

Virtual Worlds 101

Dozens of 3D virtual world implementations currently exist. Most contain notions such as regions (land); avatars that represent users who can walk, fly, chat, or speak; and objects that avatars can build, own, trade, or store in their inventory. Some virtual worlds are closed, in the sense that importing or exporting content is difficult; others are open. Some have a fixed notion

of space, with fixed-size regions and a single physics model; others can accommodate portals that take a user from one world through a door into another. Some focus on cartoonish models and support social interaction of small groups; others are used for training or simulations and can accommodate hundreds of avatars per region.

As I noted in my previous article, the real world is 3D, very high def, scalable, and diverse. If we wanted to model it, we’d have to ask what kind of database schema or object model could be used to represent the world. Without going into detail, we could take the schema of a 3D virtual world as a starting point. The kinds of entities we’d need to model include locations at a variety of scales, land use and structures, avatars, primitive and composite objects, inventory items, assets, access authorizations for places and things, and scripts.

If we deconstruct the most widely used virtual world, Second Life, we would find that it’s architected as a client viewer with servers that contain content or provide other services such as avatar authentication. We might notice that virtual worlds are built on a suite of lower-level standards – for instance, IRC for instant messaging and

Craig W. Thompson
University of Arkansas

Collada (*Collaborative Design Activity*; www.collada.org) or Second Life primitives (*prims*) for graphical content.

It wouldn't take long to notice that virtual worlds have various limitations. As similar as virtual worlds are to gaming platforms, they don't meet all the requirements for building certain kinds of games, especially fast-paced, first person shooter games. Today's virtual world implementations don't scale to a stadium of avatars or the entire earth. Virtual world implementations are heterogeneous, and most don't interoperate. Rapidly populating virtual worlds by importing content from geographic information systems or the CAD community is still uncommon, and there isn't yet a widely used way to mirror state change in the real world directly into virtual worlds or to model past, present, and possible futures in virtual worlds.

In this Issue

So, what problems must we solve to make virtual world technology widely useful?

First, we'd need to make it as seamless for any user anywhere to visit and leave one virtual world for another as it is for us to come and go to websites. Virtual worlds typically use a client-side viewer that renders content stored remotely on servers. It makes sense to integrate virtual world viewers into Web browsers. Already, virtual world URLs can access a virtual world location (for instance, the Second Life URL [http://slurl.com/secondlife/University of Arkansas/123/81/32/](http://slurl.com/secondlife/University%20of%20Arkansas/123/81/32/) accesses an x-y-z location on the University of Arkansas island). In "Extending Web Browsers with a Unity 3D-Based Virtual Worlds Viewer," Neil Katz, Thomas Cook, and Robert Smart describe an architecture for plugging the Unity 3D viewer into Web browsers. Their aim is to remove the roadblock of having separate applications for Web browsing and virtual world interaction.

Just as anyone can create a website, it makes sense for anyone to create a virtual world. But it also makes sense that an end user's avatar be able to leave one virtual world and enter others. Thus, we need a solution to avatar interoperability, so that an avatar can move between virtual worlds, and we need various ways to federate virtual worlds so individual worlds can come and go like websites do.

Cristina Lopes explores these issues in "Hypergrid: Architecture and Protocol for Virtual World Interoperability."

At a high architectural level, we can distinguish a virtual world platform from applications that are built on top of virtual worlds. Architectural questions arise: What kinds of applications can be built on virtual worlds? Where is the dividing line between the virtual world platform and the application? One way to answer the question of what capabilities a virtual world platform should support is to view this question as a red herring. Instead of a fixed virtual world platform, we'd like extensibility mechanisms for augmenting virtual worlds with additional capabilities. We can imagine virtual worlds with or without avatars, with different physics engines, with high- and low-fidelity sound, and so on. Toni Alatalo in "An Entity-Component Model for Extensible Virtual Worlds" and Jonathan Kaplan and Nicole Yankelovich in "Open Wonderland: An Extensible Virtual World Architecture," working in two different virtual worlds (OpenSimulator and Open Wonderland), have developed similar component capability extension mechanisms to accommodate the range of variation that we can predict will be needed in different virtual worlds built for different purposes.

Capabilities we'd want in a virtual world could include search engines and ways to add semantics to create "semantic worlds." In "Virtual and Real-World Ontology Services," Joshua Eno and I explore how we can use virtual world search engines to collect objects (and their labels) and then use those labels to build taxonomies that match some large-scale ontologies, like WordNet and DBpedia. We observe that virtual worlds don't generally contain a semantic layer, that such a layer might be equally important in modeling the real world, and that a smart semantic world (analogous to the Semantic Web) might result if we could extend virtual worlds (that can mirror the real world) with corresponding semantic types and rules. If virtual world architectures become extensible (as in the Alatalo and Kaplan/Yankelovich articles), then an ontology service can store and retrieve semantics about avatars, objects, and places for virtual or real-world applications that need that capability.

Some virtual worlds such as Second Life make trade-offs in their modeling capabilities.

In Second Life, it is difficult to model very large objects or very small ones, or to simulate small motor skills in workflows or object interiors. Second Life isn't the platform to use for performing remote surgery and doesn't make it easy to model stresses in bridges, heat flow, building plumbing and wiring diagrams, or similar simulation requirements. In "Accuracy in 3D Virtual Worlds Applications: Interactive 3D Modeling of the Refractory Linings of Copper Smelters," authors Anthony J. Rigby, Kenneth Rigby, and Mark Melaney identify and discuss the requirement for accurate modeling in some 3D world applications, like engineering and CAD applications and military simulations.

Two articles focus on applications built on top of virtual world platforms. In "Connecting Virtual Worlds with the Real World for Learning a Foreign Language" (to appear in a future issue of *IEEE Internet Computing*), María Ibáñez, Carlos Kloos, Derick Leony, José García Rueda, and David Maroto build an educational application on top of Open Wonderland that involves a mirror world where students interact in the real world and also in a corresponding model world, both representing an avenue in Madrid. In "I-Room: Augmenting Virtual Worlds with Intelligent Systems," Austin Tate describes a suite of collaboration tools developed at the University of Edinburgh that can be used in civilian or military command centers to gather information, understand an evolving situation, and make decisions. Several of the tools (to-do lists, planners, and so on) can be used independently of a virtual world. Interestingly, they can be tied into a virtual world (Second Life or OpenSimulator) so that, though geographically distant, the planners (that is, their avatars) can meet together, chat or talk, and see in-world representations of shared collaborative content. This virtual presence helps synchronize the team.

Although many areas of virtual world technology need further exploration, virtual worlds are evolving toward standardization. Rather than a monolithic standard, the area is moving toward a suite of loosely coupled standards that help insure interoperability: Collada is recognized as the gold standard for graphical content; the IETF Virtual World Region Agent Protocol effort (VWRAP; <http://tools.ietf.org/wg/vwrap>)²

focuses on avatar interoperability; the Web 3D Consortium (www.web3d.org) is developing 3D standards; and the IEEE Metaverse Standards working group (www.metaversestandards.org) is developing a glossary and a reference architecture for virtual worlds. Common APIs might make sense. In "Toward a Semantic Approach to Virtual World Standards" (also to appear in a future issue), David Burden considers virtual world markup languages as another area that could be standardized.

Future Directions

Where is virtual world technology going, and will virtual worlds fulfill their promise leading to pervasive use? Virtual world technology is no longer in its infancy, but it's still immature. A Gartner hype cycle graph shows virtual world technology with inflated expectations in 2006, a disillusionment trough in 2009, and the virtual world community currently slowly climbing an enlightenment slope toward a productivity plateau. While Second Life is still the dominant virtual world platform, the open source OpenSimulator platform is solidly functional, as are several other virtual world platforms such as Unity and Open Wonderland. But there is not yet a clear front-runner architecture or implementation that meets the needs of the many potential virtual worlds applications.

Early adopters in the broad education community use virtual worlds for classes and meetings. There are workshops, conferences, and journals that publish the occasional virtual world paper – and a few venues directly focus on virtual worlds. But the academic-industrial virtual world research community is splintered, heterogeneous, and distributed. The IEEE Metaverse Standards working group provides one of the best current forums for architects to meet to discuss virtual world directions.


It seems clear that virtual worlds can go well beyond being venues for social interaction to also support serious applications involving teaching, training, and simulation. Especially, it seems likely that we'll eventually have 3D models of the real world and be able to use technologies such as RFID, Kinect, and smart phones to constantly gather and update the models.

It's not yet clear what route we'll take toward a 3D Web or whether we'll get there via the efforts of a dominant player or virtual world platform or some other route coming out of left field. However we get there, it seems that a good understanding of virtual worlds' software architecture will help ensure that eventual solutions will meet a broad array of community requirements. ☐

References

1. C. Thompson, "Next-Generation Virtual Worlds: Architecture, Status, and Directions," *IEEE Internet Computing*, vol. 15, no. 1, 2011, pp. 60–65.
2. J. Bell, M. Dinova, and D. Levine, "VWRAP for Virtual Worlds Interoperability," *IEEE Internet Computing*, vol. 14, no. 1, 2010, pp. 73–77.

Craig W. Thompson is the Charles Morgan chair in the Department of Computer Science and Computer Engineering at the University of Arkansas. His research interests include artificial intelligence, databases, middleware architectures, virtual worlds, RFID, and pervasive computing. Thompson has a PhD in computer science from the University of Texas at Austin. He's an IEEE fellow. Contact him at cwt@uark.edu.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

IEEE Internet Computing: Call for Papers

Submit a manuscript on ScholarOne at <https://mc.manuscriptcentral.com:443/ic-cs>

Programmatic Interfaces for Web Applications (July/August 2012) Final submissions due 1 November 2011

Please email the guest editors a brief description of the article you plan to submit by 15 October 2011

Guest Editors: Tomas Vitvar, Cesare Pautasso, and Steve Vinoski (ic4-2012@computer.org)

The rapid growth of programmatic Web service interfaces for Web applications (open Web APIs) has revolutionized online content integration and development practices. The increasing popularity of such Web interfaces raises questions of how developers should design services and how they should maintain services' good performance and scalability. Programmatic Web interfaces typically use REST style for communication, or RESTful services implemented with HTTP, while moving away from more traditional SOAP Web services. Although they can take advantage of already existing Web architecture, many APIs that claim to be RESTful actually fail to do so. They overload the meaning of HTTP methods, ignore standard response codes, or do not well support hypermedia to represent relationships among application states. Moreover, developing a programmatic Web interface requires a tight integration with already existing back-end applications and infrastructures, and sometimes requires a new, highly dependable back-end technology.

This special issue seeks original articles on topics related to

emerging technologies and best development practices that underpin any modern programmatic Web interface. Sample topics include

- best practices, patterns, and anti-patterns of a programmatic Web interface design;
- benchmarking and evaluation of programmatic Web interface scalability and performance in large-scale Web applications;
- comparisons and empirical evaluation of various styles, protocols, and descriptions for programmatic Web interfaces;
- reports and lessons learned from developing programmatic Web interfaces for various application domains and sectors (such as social, e-commerce, video, geospatial, and so on); and
- end-to-end engineering of programmatic Web interfaces and their integration with existing back-end applications requiring the development of novel dependable and scalable technology frameworks.

All submissions must be original manuscripts of fewer than 5,000 words, focused on Internet technologies and implementations. All manuscripts are subject to peer review on both technical merit and relevance to *IC*'s international readership – primarily system and software design engineers. We do not accept white papers, and we discourage strictly theoretical or mathematical papers. To submit a manuscript, please log on to ScholarOne (<https://mc.manuscriptcentral.com:443/ic-cs>) to create or access an account, which you can use to log on to *IC*'s Author Center and upload your submission.

www.computer.org/internet/author