Conference and Exhibition of the European Association of Virtual and Augmented Reality (2014)
G. Zachmann, J. Perret, and A. Amditis (Editors)

Towards a new platform paradigm for synergetic Virtual Environments

M. Hocke, S. Pena Serna & J. Wurster

ESI Software Germany GmbH

Abstract

*Most of today's Virtual Reality system architectures follow visualization-centric data paradigms, using hierarchical data structures typically containing static content. As one of the future's - and even today's - key communication and collaboration tools, Virtual Reality is quickly developing into an integrator of heterogeneous technologies and content. For instance, highly sophisticated software simulation technology exists today to enable industrial users [KNC*10] to apply the benefits of virtual prototyping throughout the challenging steps of the product development process:* ***Immersive Virtual Engineering*** *has an enormous potential to deliver the engaging work environment for today's and future technical users delivering a fast, accessible and pertinent virtual model that drives day to day engineering decisions - individually and/or collaboratively, from concept design to process engineering and serviceability. Within this paper, we would like to point out some of the new challenges and their consequences, suggesting new paradigms in system architecture design that will enable next generation software platforms to handle the industry's demands towards synergetic Virtual Environments.*

1. Introduction

Most current Virtual Reality system architectures follow visualization-centric data paradigms, using hierarchical data structures typically containing static content. Years of expertise invested both by research groups and commercial vendors have resulted in a multitude of highly efficient, optimized approaches available to solution providers and industry users (e.g. [RS01]). Virtual Reality use cases such as exploration, simple data interaction and even multi-site collaboration therefore are handled well on reasonably complex data. Requirements for synergetic virtual environments supporting **Immersive Virtual Engineering** are emerging from design and functionality prototyping, process planning, asset management, commissioning, quality management to production issue tracking. Synergetic virtual environments have to handle the combined complexities of process engineering, highly interactive immersive workflows, computationally intensive simulation results with heterogeneous, arbitrary added content in a collaborative virtual experience.

As an example of a visionary use case, consider an isolated aspect of the product development cycle for a scooter. During initial test production of an updated model, the first series of completed products show engine peak torque performance values below expectations, resulting in unacceptable product characteristics. Within an immersive environment, a group of engineers analyzes the problem using live data streamed in from a parallel dyno testing run. Comparing the actual performance values with data obtained from a previously run flow simulation of the engine based on the original construction data, the system highlights recent changes in specification for all engine parts. As associated process metadata is analyzed, a recent design change in a third party part is displayed. The team of engineers reruns the flow simulation based on the updated data, identifying a gas flow change caused by the redesign as a result. Based on the data and evidence at hand, the team issues a quality control defect notice containing complete documentation of the change, the performance data obtained during the physical product engine dyno run, the original revision specifications of the third party part and a replacement request order with the manufacturer of the third party part. The defect could ultimately be found, analyzed, identified - and in a follow up session even resolved and verified using one virtual work environment and integrative process. Adding to the proven benefits of Virtual Environments through improvements of communication and workflow, this promises significant ad-

DOI: 10.2312/eurovr.20141334

ditional savings in time and cost through shorter, optimized processes and workflows.

Within the afforded workflows, immersive Virtual Environments have enormous potential to form the missing link throughout the engineering process by leveraging their established virtues in communication and collaboration, helping to understand the complexities of process and workflow interactions. Existing systems such as VRML [CB97] and its successor X3D [Con13] focus on static, visualization data-optimized structures and limit dynamics to predefined behaviors; thus the key aspects of dynamic behaviors - a flexible concept of time, process and parallelism - are either not handled at all or are handled in an inefficient way. Aside of these principal limitations, complexities and data sizes associated with simulation results typically exceed the assumptions for real time systems. While some visualization systems specifically handle almost arbitrary amounts of data using out of core techniques, these approaches might need to be rethought to broaden the scope to other types of data. Out of core techniques also lack when considering the dynamic aspects of data and behavior required within an integration platform design.

Contributions to integrated data will typically originate from a multitude of foreign data models in multi-core and multi-node systems, affording data handling that interprets and continuously evaluates in parallel rather than traditional approaches that just read any incoming data once, following the legacy system process of reading in raw data, conversion to a local model and visualization. Crossing the boundaries of process- and machine-local data models is therefore an important aspect of rethinking the Virtual Reality platform.

2. Multi-tiered dynamic data behavior

Following these new requirements for integrative software systems, the main challenge is to enable and maintain parallel, dynamic behavior throughout the system. Therefore, we introduce a new platform approach that adapts to the described trend ranging from the generic requirements for software architectures like maintainability and evolvability to concrete concepts for an architecture fulfilling today's industrial needs in data handling. Additionally, our proposed system approach uses parallelism to handle an arbitrary number of data structures, aiming to achieve full interactivity and responsiveness for the virtual environment regardless of system load and process step. It exploits the idea of balanced decentralization of asynchronous processing running in parallel between multiple instances, while minimizing the latency and performance issues. Complementing the holistic architectural considerations for dynamic behaviors in structures, the underlying representation of data needs to sustain and support responsiveness of the system for arbitrary mass change of any element property, e.g. through animation of rigid body or individual vertex transformation. Besides simple parameter changes, we also need to consider complex multi-property change such as topology-changing deformations of 3D objects through the result of real-time physics, virtual human simulation or live exploration of numerical simulation results (e.g. airflow or crash simulation). In this paper we will primarily focus on the presentation of the two main concepts: a) domain based data flow concept and b) modular object model.

2.1. Domain based data flow concept

Software system architectures have been proposed that focus on basic functionality and introduce flexibility regarding the rendering system and the user interface toolkit [SRH05] or demonstrate how a Virtual Reality engine integrates VR hardware and software within a graphics API [FK05]. These systems, however, define a more centralized view on data models especially in terms of scene representation. Of course, virtual environments have a certain set of data models that have to be processed from input to the user's visual perception as directly as possible. In [SRH05], the layered system architecture introduces a so called Graphics Layer that includes the mentioned data models but data integration is not covered. In general such concepts are lacking interfaces for dynamically acquiring data as our visionary example is showing. Other publications focus on creating a model for data flow concepts. In [RL10] a classification of parallelization methods and techniques is done. It defines implicit methods as software development requirements that are often given by the programming layer as for example compilers or programming languages. Explicit methods are based on shared data while using concurrency. Consequently this needs a deeper analysis of the data flow to manage synchronization of threads. Therefore, data flow networks are introduced in which nodes are containers of value holders and the edges represent the flow of data. In [RL10] they show how networks for data flow regarding an explicit method are created and which heuristics are used to traverse the network efficiently. The performance of such traversing techniques including multi-threaded processing seems strongly dependent on the quality of the selected network. Mapping data flow to an equivalent graph was also investigated by [RS01]. Their method is, however, intended to handle data of tracking and multi-modal input. It is also about message passing between nodes, parallelization of event handling and handling synchronization transparently. The idea of parallelization has also been covered early on, e.g. by [SGLS93], describing a decoupled simulation model to improve responsiveness in terms of user interaction. Decoupling is achieved by identifying four kinds of separate processes: the master process, responsible for interaction and geometric updates, a computation process for the actual simulation and dedicated client and server processes for device data exchange. The concept to separate some parts of data flow by the principle of client/server is also described in [MC13]. Although the scene graph model is still a central part of this system, flexible system design is realized by strictly separating producers and consumers by communicating data changes over a TCP/IP network. Decoupled data transfer inside the components is not solved however. Handling multi-threading transparently and exchanging data decoupled by implementing the actor model is the concept in [FL08] to realize a novel simulation core for intelligent virtual environments (SCIVE). In addition the data flow is modeled on an abstraction level by semantic networks. This has the advantage to be independent of any underlying programming language. This system allows dynamic behavior but the corresponding networks can get very complicated for large, complex systems. Our proposed design paradim is similar to the actor model idea in [FL08], has decoupled communication and its data flow is easy to design by connecting components of value holders explicitly. Possible common multi-threading issues should be minimized even if shared data models are involved. The following sections introduce the concepts of domains and data flow to meet the mentioned requirements.

2.1.1. Domains

There are considerable numbers of data models to be managed when designing today's Virtual Reality software. For example, scene data, rendering specifics, simulation data and application oriented parts as well as traditional and immersive user interfaces. Concerning our visionary example, we have a simulation acquisition thread for the engine dyno testing that continuously tracks results. For comparison reasons, a second dataset from a previous run is acquired in parallel through a data access process. Following this work-flow, one finds further possible data models, such as a metadata handling process and potentially different scene representations that can be switched in the current view. Those structures have to be protected against corruption in a highly multi-threaded environment, due to possible concurrent access. Therefore, the aim of the architectural design is to minimize the coupling and to increase the concurrency. In general, data models have very complex structures, so that additional locking mechanisms would make them more complicated and almost unmanageable. Multi-threading problems like deadlocks, race conditions etc. would have to be solved and would be a recurring issue. The purpose of domains is to deliver a simple solution and make the domain responsible for guarding the data model instead of using thread-safe objects. A domain is a well defined part of the software architecture which has its own thread, data objects and module instances. Domains with a shared data model consist of generic objects, so-called operands, that they manage. No other thread may change them directly. While every operand has its own implementation, a coherent directed graph at least implicitly remains through referencing other operands. Instead of direct access, operands are handled decoupled with the help of the data flow concept as described in the following section.

2.1.2. Data flow

Concerning the domain concept, data changes are triggered by queuing jobs, so-called operations, that are processed inside the domain thread. Writing to a data model is a time consuming process and in previous more centralized approaches the resulting behavior is that threads have to wait for accessing the data model because resources are already locked by another thread. This also means waiting times are not easily predictable as they strongly depend on arbitrary long resource locking. The data flow concept does not suffer from classical locking problems because synchronization points are only happening inside a single queuing mechanism where operations are entered for later processing. Adding an operation to the queue only takes very little effort as it only copies the reference. However, synchronization still needs to be done and cannot be avoided.

Further optimizations can be achieved through managing more than one queue and swapping them to reduce the collision times of operation producers and their consumer to a minimum. The result is a highly decoupled data model environment that keeps typical multi-threading topics transparent in most of the cases. Any observed domain should inform other listening domains about its changed operands if needed. This introduces a difficulty in maintaining decoupled domains, as handling data modifications usually causes reading the data that has changed in the observed model. In our proposed data flow concept, domains broadcast events to the event loops of the observing domains with the corresponding data that has actually changed. This reduces the synchronization overhead because event handling does not need to acquire any lock for reading from the source domain and decoupling can be maintained without threatening data integrity.

Consider again our example, where the team is inspecting the analyzed data with further information like metadata associated and simulation results from a previous run loaded in parallel. This is a more complex process and costs a lot of computation time. Additionally the flow simulation results need to be frequently published to the scene and some rendering instance as well. Undoubtedly this leads to latency in showing the result. At the same time however, users want to retain control e.g. by the user interface without being dependent on this process of high effort. The aim is to reduce the latency for the user with respect to the responsiveness of the system. To lay the foundation for this, the domain concept offers decoupling in all kinds of message passing, and therefore we have the building-blocks to construct a highly dynamic, responsive system.

Figure 1 shows an example of a typical basic setup of a system for Virtual Reality that consist of a number of domains interacting with each other. For simplicity, more advanced domains like the distribution domain are only outlined, and would also have all components shown in the other domains.

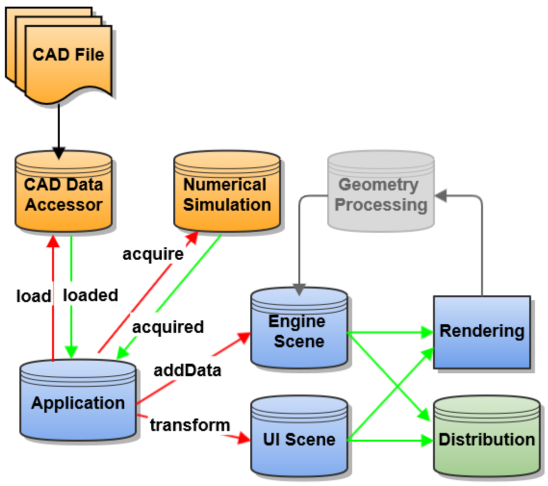

Figure 1: *Domain System*

2.1.3. Flow Control

The concept of domains is to have decoupled data processing by operations and events in domain threads. Without flow control, the flow of data depends entirely on OS schedulers which balance domain threads. However, uneven long periods of data processing in domains are natural behavior in OS schedulers, which from the developer's point of view are non-deterministic. Consequently there is a risk of growing numbers of entries in message queues of domains. This means that an unbalanced data flow causes buffer overflows, with "Overflow" in this case meaning that this unbalanced data processing is a continuous process and the number of operations/events queued is increasing unceasingly. We will need strategies to either reduce the likelihood of these situations or to avoid them entirely.

2.1.4. Flow control strategies

We want to focus on 3 basic strategies to cover data flow control respectively reducing possible high loads of redundant messages in the intended system. The strategies are a) avoid queuing, b) reduce load of data changes and c) delay queuing of data changes. The solution for controlling data flow is a mixture of all strategies that will minimize the problem of growing memory usage in terms of increasing number of operations/events in domain threads. Figure 2 shows a thoroughly realistic scenario where a tracked input device samples with high frequency. The system is in a state where the tracked positions are mapped to the 3D user interface. Creating the operations and adding them to the corresponding scene domain with the given frequency would produce a load that cannot be handled. The situation would be that the scene domain is processing the current transformation but does not finish before the next change arrives. The first strategy follows the pattern that it should only be queued when it is possible to process the current job, otherwise the current thread proceeds its run. Therefore, from the application's point of view, it makes sense to identify a sub-system of the whole setup that is relevant for interactivity. An instance within the distribution component that transports changes over a slow network should not belong to it. In terms of the example in Figure 2, the tracking device input thread would immediately go on sampling without queuing the transformation, preferring concurrent and consistent results for as little as possible latency and high responsiveness.

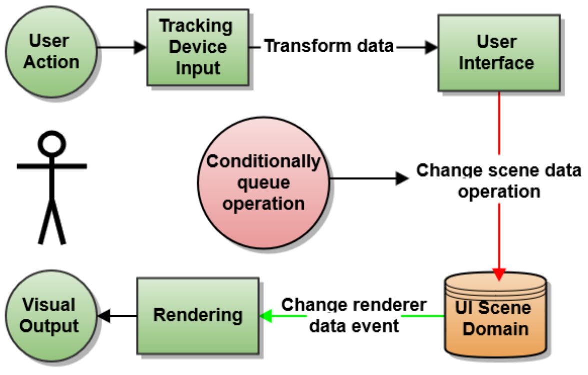

Figure 2: *Avoiding queueing*

With strategy a) the interactive sub-system is in a balanced state, but the complete system contains domains that are delayed because of high effort. To increase control, further reduction of load is obtained as redundant packets can be omitted and only the up-to-date packet reflects the current state. This introduces an important precondition in particular on how data is represented in the domain concept. All data of operands should be absolute and not representing relative change to the previous value. Eventually for our example regarding distribution, the domain can reduce the number of elements in its queues e.g. by skipping transformation data for an immersive user interface widget except for the most recent entry.

The last presented strategy is a more coupled solution because it introduces some synchronization points in terms of waiting situations. If a domain signals producers of high loads that a certain threshold of queue entries has been reached, producers will have to wait until there is a free slot to proceed. Through concrete implementation this can be anything from a smooth transition to a full stop to get rid of the current load very fast. Strategy c) fits well for e.g. the distribution scenario since it postpones the steady delay that would occur in combination with the first strategy to a later time, and with strategy b) it will even be a rare situation to cope with.

2.2. Modular Object Model

In order to delimit the context, this section does not cover the whole data model of a VR application, which is also referred to as the semantic model or the semantic description (see Chevaillier et al. [CTB*12], and Flotynski and Walczak [FW13]). This section focuses on the part of the data model related to the 3D object. For the sake of clarification, we understand semantics as the meaning, which is inferred from information and knowledge being available within a specific context, and semantic enrichment as the mechanism for enriching 3D objects with semantics ([PSSD*11]). Consequently, existent information and knowledge related to a 3D object ([SF09]), sometimes referred to as metadata, include: i) information related to its intrinsic structure, ii) information related to the meaning of the physical object represented by the 3D object, iii) information related to the digital provenance, and iv) knowledge related to the application domain.

3D objects, seen as the digital representation of existing or of to be potentially created physical objects, are composed by 4 different characteristics: i) dimensions, ii) structure; iii) functionality, and iv) context. All these characteristics are represented by different models: a) geometric models, b) material models, c), simulation models, and d) semantic models, respectively. Every single model deals with dedicated properties of the 3D object, for which a specific representation and a corresponding interface are needed. These properties cannot easily be considered in a synergetic manner and their uncorrelated handling prevents the definition of rich relationships. A Smart 3D Object needs to be aware of and to correlate all properties (models), in order to smartly interact within a given environment. To support the concepts behind Smart 3D Objects, the traditional VR systems need to be reevaluated (e.g. Whyte et al. [WBTM00]). Dynamic behavior (e.g. Pena Serna et al. [PSSF11b]) such as the deformation of and object caused by a collision, the direct manipulation of an object by morphing its shape or the dynamic streaming of geometric information demand the flexibility to locally perform modifications and adaptations at geometric and topological level (see Morse et al. [MBS11] and Brunton et al. [BCM*11]). The main categories behind the modular object model are: a) geometry data, b) topology data, c) visualization data, and d) partition data (Figure 3).

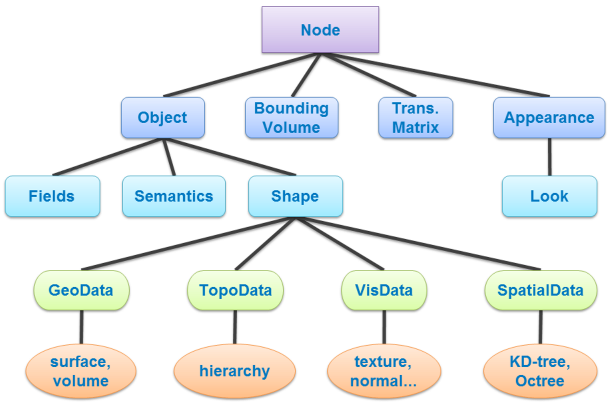

Figure 3: *Modular Object Model*

2.2.1. Geometry Data

A 3D object might be described in different ways according to the expected purpose. Complex 3D objects are difficult to describe with a simple formulation, because of that the description of a 3D object is usually a collection of simple formulations for individual sets. In the engineering domain, this is commonly achieved by means of combining geometric and topological descriptions. On the one hand, the geometric description might be smooth or discrete, and it governs the form of the 3D object. On the other hand, the topological description deals with the existing relationship and connectivity between the different sets. Although a 3D object is only a digital representation of a physically or a digitally born object, it finally embodies physical properties. The geometry enables the computation of such physical properties within a set and the topology allows for assembling the properties of the individual sets within the whole 3D object (e.g. Pena Serna et el. [PSSSM09]).

In the context of virtual manufacturing and virtual engineering, the geometry is given and it is therefore not created; instead it is transformed, in order to meet the requirements behind the mathematical and numerical models of the forthcoming process. In the engineering field, modeling has a wider acceptance based on smooth representations (e.g. B-Reps); other fields, for instance entertainment, are used to modeling with smooth (e.g. subdivision surfaces) or also discrete representations (polygonal meshes). In both cases, only the surface of the 3D object is represented. In terms of simulation, discrete representations are currently more suitable for numerical computations. In this context, a description of the volume of the 3D object is required (e.g. tetrahedral / hexahedral meshes), in order to better associate the material properties and the expected functionality of the 3D object represented through physical laws (refer to Pena Serna et al. [PSSF10]). The machining operations are characterized by employing a discrete surface representation in the STL format (StereoLithography). The most common representation for geometry data in Virtual Reality are optimized for static behavior and fast rendering (e.g. Rossignac [Ros96] and Zachmann and Langetepe [ZL03]). Nevertheless, given the requirement to enable dynamic behaviors, the geometry data requires the capabilities to flexibly deal with the addition and removal of geometric definitions. Our current ideas foresee the use of dynamic buffers with flags (Figure 4), indicated unreferenced data and avoiding memory handling (see Pena Serna et al. [PSSF11a]).

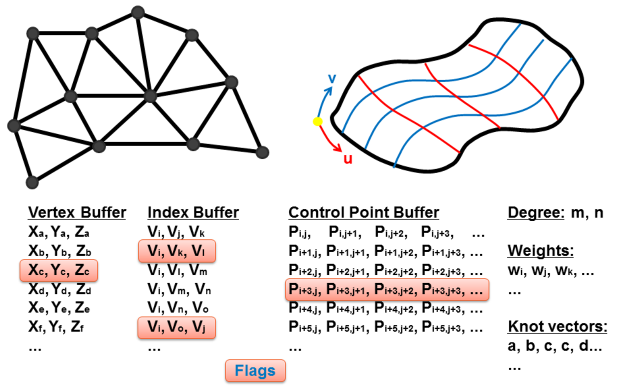

Figure 4: *Geometry Data*

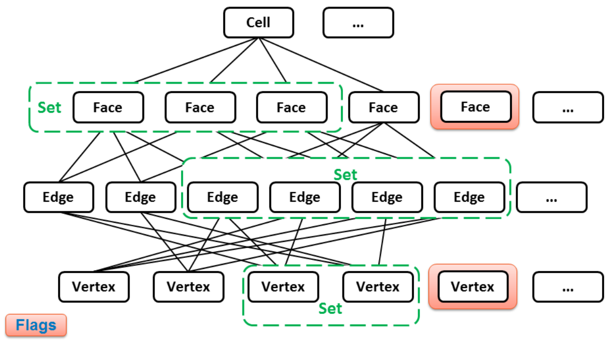

Figure 5: *Topology Data*

2.2.2. Topology Data

In order to avoid the traversal of the whole 3D object and to streamline the computation process for local properties on demand, a querying interface is desired (refer to Kremer et al. [KBK12] and Canino and De Floriani [CF13]). The following concepts describe a topological data structure, which could be used for different representation schemes (e.g. B-Reps, meshes, subdivision). The main characteristics are (see Pena Serna et al. [PSSF11a]): i) rule-based hierarchical topology, ii) parental relationship graph, and iii) indirect sibling loops. Rule-based Hierarchical Topology: the topology of a 3D object consists of a hierarchical description of entities in different dimensions. Given that our 3D objects live in R^3, we describe the hierarchies of entities from R^3 to R^0 as: Cell, Face, Edge, and Vertex; thus decomposing the 3D object into orientable entities in every dimension. Parental Relationship Graph: the decomposition rules enable the establishment of intrinsic relationships, easing the development of the querying interface. These hierarchies are structured, by means of creating a parental relationship graph, which indicates the relationship from the parent to the direct children. Indirect Sibling Loops: in order to store the relationships without increasing the use of memory while still achieving fast access to the information, the oriented loop formed by the children of the parent is indirectly defined by the siblings within the loop. In a similar way as in the geometry data, the topological entities can be flagged to indicate that these are no referenced and to avoid performance penalties. Moreover, topological entities of the same dimension can horizontally be clustered into sets, which can represent application-driven semantics (Figure 5).

2.2.3. Visualization Data

We understand visualization data as the set of information, which is needed to generate a visually pleasing image, but which is independent from the geometric representation. In this context, we consider normal and tangent directions, colors, texture, and texture coordinates, among others. The traditional approach to store and access this information is tightly associated with the geometry representation itself. Nevertheless, the dynamic behaviors that current use cases require demand for a more flexible storage and accessing, which can be selected according to the application needs (Figure 6). For very large data sets, the rapid exploration instead of the dynamic manipulation is expected. Thus, a single-indexing approach with split vertices is ideal. A highly dynamic scenario is best satisfied with a multi-indexing strategy, which allows for manipulating geometric and visualization data without handling sorting and splitting operations. In case of an application with large and highly dynamic conditions and with very strict requirements toward memory and performance; the multi-indexing strategy can be restricted to only integrated data.

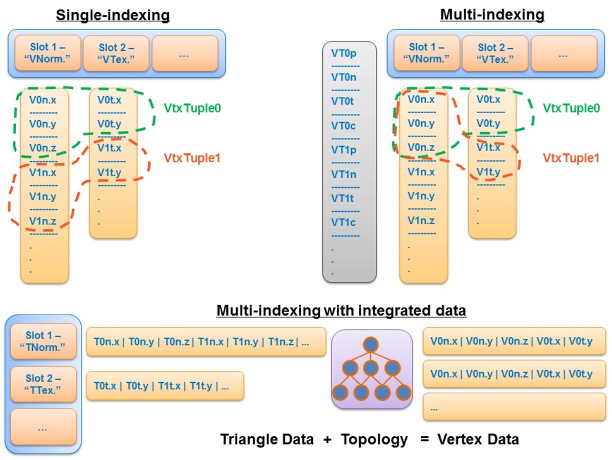

Figure 6: *Visualization Data*

2.2.4. Spatial Data

Spatial querying is a procedure dedicated to searching within a bounded collection of items being defined in R^3. In our context, the items within the collection are 3D objects, which are hierarchically structured by means of a scene graph. On the one hand, the acceleration structure underneath the scene graph could be a bounding volume hierarchy, an object oriented tree, etc. On the other hand, the acceleration structure for the geometric representation could be a k-d-tree, a b-tree, an octree, the direct geometry, etc. Hence, the spatial query component should be flexible enough, in order to handle different acceleration structures. The spatial query component is designed to independently handle the scene and the geometry structure, according to the needs: a) highlighting or selecting 3D objects, or b) highlighting or selecting elements. Motivated by the principles of Open VDB [Mus13] and our requirements for a dynamic behavior, we are currently conceiving an acceleration structure based on shallow and wide trees, which will allow us to rebuild individual branches without affecting a big portion of it (Figure 7. These characteristics will be complement by referring the leaves of the tree to topological properties instead of geometric ones, in order to minimize the amount of needed updates.

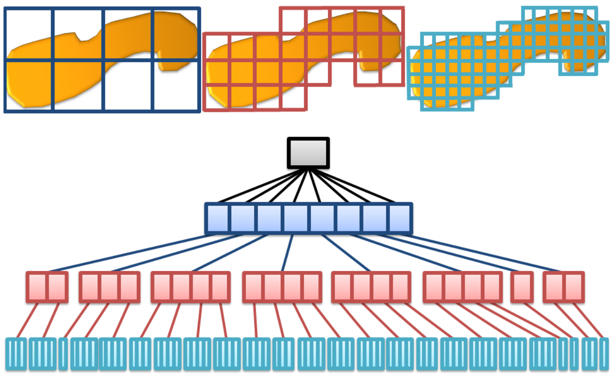

Figure 7: *Spatial Data*

3. Conclusion and Future Work

Early validation, review and testing throughout an integrative product development process forms the key benefit of next-generation synergetic virtual environments. While providing exciting new possibilities, fully dynamic data handling - interpreting heterogeneous data from arbitrary sources in parallel - imposes a potentially disruptive requirement to existing virtual reality platform architectures. Within this paper, we have proposed a new system modelling approach as groundwork for handling dynamic data behavior throughout highly interactive immersive simulation applications. A fully furnished system based on the work presented should be able to maintain and possibly improve typical system responsiveness through specific high performance, low latency paths. We have continued to illustrate considerations towards a fully dynamic, extensible model for typical 3D data. Accompanying the implementation of a fully-featured software platform based on the synergetic platform paradigm, we would like to further explore the aspects of crossing the system boundaries - dedicating further work to interpretation of runtime remote data as well as synchronization and distribution for interactive collaboration and remote processing. Other topics to explore in future work would consider approaches to reduce total system latency for varying domain input to output interactions.

References

[BCM*11] BRUNTON R., COOLAHAN J., MORSE K. L., SCHLOMAN J., RIGGS B., SCRUDDER R.: *LVC Architecture Roadmap Implementation, Common Capabilities – Common Data Storage Formats*. Progress Report NSAD-R-2011-022, Johns Hopkins University, February 2011. 5

[CB97] CAREY R., BELL G.: *The annotated VRML 2.0 reference manual*. Addison-Wesley Longman Ltd., 1997. 2

[CF13] CANINO D., FLORIANI L. D.: Representing simplicial complexes with mangroves. In *IMR* (2013), pp. 465–483. 6

[Con13] CONSORTIUM W.: X3d standards for version v3.3. ISO Standard ISO/IEC IS 19775-1:2013, Nov 2013. 2

[CTB*12] CHEVAILLIER P., TRINH T., BARANGE M., LOOR P. D., DEVILLERS F., SOLER J., QUERREC R.: Semantic modeling of virtual environments using MASCARET. In *5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2012, Costa Mesa, CA, USA, March 5, 2012* (2012), pp. 1–8. 5

[FK05] FELLMANN T., KAVAKLI M.: Vair: System architecture of a generic virtual reality engine. In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on* (2005), vol. 2, IEEE, pp. 501–506. 2

[FL08] FRÖHLICH C., LATOSCHIK M. E.: Incorporating the actor model into scive on an abstract semantic level. In *Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), proceedings of the IEEE Virtual Reality 2008 workshop* (2008), pp. 61–64. 3

[FW13] FLOTYNSKI J., WALCZAK K.: Semantic multi-layered design of interactive 3d presentations. In *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, Kraków, Poland, September 8-11, 2013.* (2013), pp. 541–548. 5

[KBK12] KREMER M., BOMMES D., KOBBELT L.: Openvolumemesh - a versatile index-based data structure for 3d polytopal complexes. In *Proceedings of the 21st International Meshing Roundtable* (Berlin, 2012), Jiao X., Weill J.-C., (Eds.), Springer-Verlag, pp. 531–548. 6

[KNC*10] KHALDI F. E., NI R., CULIERE P., ULLRICH P., ABOITIZ C. T.: *Recent Integration Achievements in Virtual Prototyping for the Automobile Industry*. Tech. rep., ESI Group, May 2010. 1

[MBS11] MORSE K. L., BRUNTON R., SCHLOMAN J.: X3d – 3d manmade feature common data storage format. In *Fall Simulation Interoperability Workshop 2011* (Orlando, Florida, USA, September 19–23 2011), 2011 Fall SIW, SISO, Curran Associates, Inc., pp. 22–31. 5

[MC13] MALESHKOV S., CHOTROV D.: Affordable virtual reality system architecture for representation of implicit object properties. *CoRR abs/1308.5843* (2013). 2

[Mus13] MUSETH K.: Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph. 32*, 3 (July 2013), 27:1–27:22. 7

[PSSD*11] PENA SERNA S., SCOPIGNO R., DOERR M., THEODORIDOU M., GEORGIS C., PONCHIO F., STORK A.: 3d-centered media linking and semantic enrichment through integrated searching, browsing, viewing and annotating. In *Proceedings of the 12th International Conference on Virtual Reality, Archaeology and Cultural Heritage* (Aire-la-Ville, Switzerland, Switzerland, 2011), VAST'11, Eurographics Association, pp. 89–96. 5

[PSSF10] PENA SERNA S., STORK A., FELLNER D. W.: Tetrahedral mesh-based embodiment design. In *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference* (New York, NY, USA, August 2010), vol. 3 of *IDETC CIE 2010*, ASME, pp. 131–140. 5

[PSSF11a] PENA SERNA S., STORK A., FELLNER D. W.: Considerations toward a dynamic mesh data structure. In *Proceedings of the Eurographics Swedish Chapter Conference* (Goslar, Germany, November 2011), SIGRAD 2011, Eurographics Association, pp. 83–90. 5, 6

[PSSF11b] PENA SERNA S., STORK A., FELLNER D. W.: Interactive exploration of design variations. In *Proceedings of the NAFEMS World Congress: A World of Engineering Simulation: Industrial Needs, Best Practice, Visions for the Future* (Glasgow, UK, May 2011), NWC 2011, NAFEMS, pp. 1–18. 5

[PSSSM09] PENA SERNA S., SILVA J., STORK A., MARCOS A.: Neighboring-based linear system for dynamic meshes. In *Proceedings of the 6th Workshop in Virtual Reality Interactions and Physical Simulations* (Aire-la-Ville, Switzerland, Switzerland, 2009), VRIPHYS '09, Eurographics Association, pp. 95–103. 5

[RL10] REHFELD S., LATOSCHIK M. E.: A comparison of parallelization methods for data flow networks. In *Software Engineering and Architectures for Realtime Interactive Systems SEARIS, proceedings of the IEEE Virtual Reality 2010 workshop* (2010). 2

[Ros96] ROSSIGNAC J. R.: Computational representations of geometry. SIGGRAPH 96 Course, May 1996. 5

[RS01] REITMAYR G., SCHMALSTIEG D.: An open software architecture for virtual reality interaction. In *Proceedings of the ACM symposium on Virtual reality software and technology* (2001), ACM, pp. 47–54. 1, 2

[SF09] SPAGNUOLO M., FALCIDIENO B.: 3d media and the semantic web. *IEEE Intelligent Systems 24*, 2 (Mar. 2009), 90–96.

[SGLS93] SHAW C., GREEN M., LIANG J., SUN Y.: Decoupled simulation in virtual reality with the mr toolkit. *ACM Trans. Inf. Syst. 11*, 3 (July 1993), 287–317. 2

[SRH05] STEINICKE F., ROPINSKI T., HINRICHS K.: A generic virtual reality software system's architecture and application. In *Proceedings of the 2005 international conference on Augmented tele-existence* (2005), ACM, pp. 220–227. 2

[WBTM00] WHYTE J., BOUCHLAGHEM N., THORPE A., MCCAFFER R.: From cad to virtual reality: modelling approaches, data exchange and interactive 3d building design tools. *Journal of Automation in Construction 10* (2000), 43–55. 5

[ZL03] ZACHMANN G., LANGETEPE E.: Geometric data structures for computer graphics. SIGGRAPH 03 Course, August 2003. 5