MediaCommons: A Scalable Abstraction for Tiled Display Environments

John Mangan, David Srour, Falko Kuester
{jmangan, dsrour, fkuester}@ucsd.edu

Department of Computer Science and Engineering, University of California, San Diego (UCSD)
Center of Interdisciplinary Science for Art, Architecture and Archaeology (CISA3)
at the Qualcomm Institute, the UCSD branch of the
California Institute of Telecommunications and Information Technology (Calit2)

***Abstract*—Many disciplines need to visualize and interact with high resolution media collections for big data analytics and dissemination of their findings. Tiled display environments have addressed the hardware requirements for such systems and can be scaled up to fill entire rooms. On the other hand, enabling programmers from numerous disciplines to develop software that scales respectively with tiled display environments without introducing the complexities of networked computer clusters is not trivial. This paper explains the specific software requirements for such environments, explores related software products, and presents a media-centric solution which abstracts the complexities of tiled display environments away from application developers - the MediaCommons Framework.**

I. Introduction

With so many fields producing and required to visualize media within large data sets, research into cluster data distribution has increased. With the introduction of technology such as the OptIPortal [1], and its later evolution as shown in Figure 1, it became simpler to deploy and scale the hardware needed for high resolution, networked visualization systems. Since these tiled display environments offer large quantities of pixel real-estate, numerous middleware approaches have been produced to enable graphical applications to make use of the high pixel count. Naturally, each middleware has dealt with conflicting requirements in regards to the development, deployment, and increased visual and computational power of the applications they enabled.

Through the discussion of requirements for a tiled display environment's middleware, that are admittedly derived namely through prior experience, and the evaluation of current middlewares available for tiled display environments such as the OptIPortal, the MediaCommons Framework was designed. It aimed to enable graphical developers with minimal networked cluster experience to create applications whose display resolution and computational potential could scale up proportionally to the hardware making up the tiled display environment.

This work was supported by the National Science Foundation under IGERT Award #DGE-0966375, "Training, Research and Education in Engineering for Cultural Heritage Diagnostics." Additional support was provided by the Qualcomm Institute at UC San Diego, the Friends of CISA3 and the World Cultural Heritage Society. Opinions, findings, and conclusions from this study are those of the authors and do not necessarily reflect the opinions of the research sponsors.

II. Requirements

For a big data visualization middleware to be useful to the most disciplines, it must strive to minimize the complexities of large scale visualization systems while enabling as much of the system's visual and computational potential as possible. It is worth noting that tools should still be made available to the application developers to handle cluster complexity, even when the framework has abstracted away the details by default. The exact applications which are to be run are not known from initial development, thus the middleware must minimize assumptions made on application logic to avoid imposing upon creative design.

The OptIPortal, a planar tiled display environment, was designed at the California Institute of Telecommunications and Information Technology (Calit2). As the primary planar tiled display environments at Calit2, evolutions of the OptIPortal were the hardware systems that viable middleware would be deployed upon. The following requirements were used to describe the support desired by such middleware:

- Scalable resolution via distributed rendering
- Scalable computational potential via distributed execution
- Extensible application logic
- Abstracts networked cluster complexity away from individual applications
- Enables application specific cluster communication and synchronization
- Interaction designed for planar tiled display walls
- Supports collaboration with remote environments
- Supports collaboration with independent applications
- Numerous media data types supported natively and extensible to more
- Scene management with scene graph optimizations

Some of the requirements contradict, such as having extensible application logic along with multi-media type supports. These requirements were still accepted since they address a direct concern recognized through interdisciplinary experiences.

III. Related Work

Other middleware products were considered, both as full solutions and as modules to build atop of. While all of the middleware products enabled applications to be rendered on tiled display environments, they each targeted differing application and development needs. This section will briefly explore the advantages of each middleware, focusing on the points that were used to evaluate each middleware's compliance with the requirements mentioned previously.

A. Available Middleware

One subset of middlewares used a man-in-the-middle approach which executed the desired graphical application on the cluster's head node while acting as a proxy for a graphical API. When API calls were received from the application, the head node sent them on to the graphical API on each of the rendering nodes, thus enabling a single application to be rendered across the tiled display environment. The Chromium [2] architecture centered around OpenGL applications, allowing the environment to render generic 3D applications. DMX [3] focused on X11 environments and worked in conjunction with Xinerama. Similar to Chromium, an application would execute on a single machine and have its graphical API calls rerouted to a cluster of render nodes. By intercepting calls to an X server, DMX was able to produce a unified, tiled display windowing environment. Both Chromium and DMX used a single head node to execute an application and route rendering calls to display nodes, thus scaling up rendering beyond a single computer but not the computational potential.

SAGE [4] [5] [6] took an approach that separates the rendering and the display of pixels. Clusters of machines would render applications and then stream the pixel data via RGB buffers to the display cluster attached to the tiled display environment. Such an architecture enables both numerous applications to be rendered and numerous clusters to handle the rendering. SAGE is therefore able to scale up in terms of both computational potential of the rendering nodes and distributed display nodes, although the pixel streaming does introduce a network bandwidth bottleneck that can be saturated via dynamic, real-time applications on a large enough tiled display environment. This improves scalability beyond what is offered by Chromium and DMX, but is not able to keep up with the scalability of tiled display environments.

CGLX [7] and CalVR [8] are two middlewares that took a parallel execution approach, in which a head node and all render nodes execute the desired application. Support is also offered for intra-node communication and synchronization. Examples include communication to distribute input events and synchronization before rendered buffers are drawn on the displays. This approach mitigates the network bandwidth bottleneck between rendering and displaying by collocating the two concerns on the same nodes, therefore limiting bandwidth usage to event sharing and drawing synchronization. CGLX includes assumptions of a planar tiled display wall, and mimics a common graphics windowing library, GLUT. Standard keyboard and pointer inputs are propagated from the head node to the render nodes. This enables GLUT applications to run atop of CGLX with changes often as minimal as relinking during compilation. CalVR focused on implementing core virtual reality features, such as user tracking and rendering within CAVEs [9] [10] - 3D articulated display environments. CalVR also encapsulates OpenSceneGraph [11], which includes numerous plugins to support the loading and visualization of many common media data types. Default manipulation of such media is supported natively via the notion of SceneObjects - encapsulated data objects. Further, CalVR is designed around a plugin framework, allowing specific applications to be developed independently and executed simultaneously in a single space, such as individual plugins for differing media types.

B. Requirement Evaluation

Despite the successes and contributions made by the middleware available, none of the options fit the requirements as outlined, as shown in Figure 2. Chromium, DMX and SAGE do not scale up with respect to the tiled display environments envisioned. CGLX and CalVR both mitigate the scale concern yet lack other features.

It is paramount to recognize that the middleware options which abstracted away the complexity of cluster development also limited the potential to leverage the cluster. While they offered attractive solutions to display numerous applications within tiled display environments, they are fundamentally limited in that they cannot scale up to the current needs of a subset of applications for tiled display environments, which this paper targets. Although other options have not developed all the tools that could abstract cluster complexity away from the application developers, that is due to a lack of development, not a fundamental limitation of the middleware architecture.

IV. Solution

Although existing middleware did not fit all the needs of the growing hardware environments, two options did emerge as viable modules to build from - CGLX and CalVR. CalVR presented three major advantages over CGLX: native support for extensible media types, scene management and a collaborative plugin. CGLX supported planar interactions without the overhead of 3D virtual reality input systems. The first two advantages of CalVR, respectively, stemmed from OpenSceneGraph. The third advantage required more development if it was to hit both collaboration requirements. To minimize necessary development, CGLX was integrated with the rendering modules of OpenSceneGraph, leaving both collaboration requirements to be developed. The resulting product was titled the MediaCommons Framework.

A. Internally Developed Requirements

Although many requirements and features were natively supported via CGLX and OpenSceneGraph, a few required development. The MediaCommons Framework aimed to ease graphical application development for tiled display environments via the abstraction of scaling networked cluster complexities. While preventing the need for developers to implement the complexities themselves, the framework also aimed to enable hooks for developers to take advantage of the cluster when desired. Many basic requirements were handled trivially via CGLX, such as OpenSceneGraph rendering callbacks and scene management calls being synchronized and input events being communicated from the head node to the render nodes.

The remainder of this section will discuss specific requirements that the MediaCommons Framework has implemented to further ease development for tiled display application development.

1) Extendable Media Data Types: While OpenSceneGraph can load and spatially manipulate common media types, such as images, videos and models, it does not enable cluster specific optimizations. To this end, MediaCommons implemented RenderableDataObjects, similar to CalVR's SceneObjects, which wrap the OpenSceneGraph nodes and enable higher level media manipulation by type. Since both CGLX and OpenSceneGraph are visible within the scope of MediaCommons, a RenderableDataObject can be extended for internal cluster-centric optimizations while providing an interface to applications for manipulating the object as a whole.

The most prevalent optimizations come in the form of tiling media data, splitting it into smaller parts to allow the nodes in the cluster to focus only on the computation and rendering necessary for data which will be displayed by that node. By implementing the image tiling concepts of [12] within OpenSceneGraph nodes, numerous multi-giga-pixel images can be displayed on a tiled display wall while leveraging the scene management and rendering tools offered via scene graphs, such as position, scale and occlusion. For high resolution images, the tiles of each image make up a subgraph which dynamically adjusts itself during the update traversal of the scene graph. As far as application developers are concerned, they can position and scale a single RenderableDataObject as desired, in this case a high resolution image. The MediaCommons Framework has effectively abstracted away the complexities of such a task by transparently resolving the level of detail for the individual tiles of the image, loading the appropriate data in a second thread, and applying the data to the scene graph independently for each render node.

Dynamic media can similarly benefit from cluster optimization. For videos a similar tiled approach [13] was used which loaded tiled videos, provided textures for each tile, internally synchronized the tiles, and supported an interface to enable and disable the playback of tiles. The MediaCommons Framework applied the textures provided by the video library to OpenSceneGraph quads which were positioned respective to each other in a scene subgraph. The subgraph was encapsulated within a single RenderableDataObject which allowed applications to treat the collection of video tiles as a single video. Internally, each render node would determine which tiles it was to draw and which were culled, via hooks into the OpenSceneGraph culling traversal, and would then instruct the video library as to which video tiles would be visible. An example is shown in Figure 3.

Similar approaches were created for non-planar media as well, such as panoramic images (see Figures 1 and 4) which would apply the image data to a sphere, point clouds, and well constructed 3D models. By extending RenderableDataObjects, modules native to the MediaCommons Framework that exist outside of both CGLX and OpenSceneGraph, further features could be developed and applied to the media being displayed.

2) Collaboration: The driving project behind the collaboration of media objects was a port of the Sonnotile [14] project which was previously integrated into a CGLX application. Sonnotile enhanced both analysis and immersion of high resolution image viewing via embedded sound annotations that would spatially trigger. The goal of supporting Sonnotile via the MediaCommons Framework was dual pronged. First, it enabled sound annotations to be added to numerous media types, as opposed to the limitation of high resolution, planar images. Second, it included development towards a collaborative environment between remote applications.

A collaboration server was developed external to the MediaCommons Framework, capable of sharing object states between connecting clients. Modules were developed to provide an interface for the clients, both to connect to the server and to extend the states the server would accept. The RenderableDataObjects were refactored to be state-based, so that any information required to spatially locate them or to determine their physical size was stored in such states. Annotation states were created and applied to the RenderableDataObjects which included where and what each annotation was. The prior work from Sonnotile was refactored by Zachary Seldess, Sonnotile's creator, into a tiled display environment audio server that connected to the collaboration server. As RenderableDataObjects were interacted with, such as a change in scale or location, the Sonnotile audio server reacted by rendering spatially and visually accurate sound as triggered by the annotations.

Following the development of remote application collaboration with Sonnotile, it was trivial to add additional states to the RenderableDataObjects as well as other modules within the framework. This included states that could load media types currently supported via the framework, which allowed remote tiled display environments to manipulate media in a collaborated virtual space. This also enabled tiled display environments to alert one another of their existence, allowing for best effort locking of media objects by users as well as user cues such as color coding the RenderableDataObjects selected by differing users. While not required, tiled display applications used the head node to communicate with the collaboration server, which could then share the returned states with the render nodes in a synchronized manner, minimizing bandwidth requirements and ensuring a unified display of the virtual environment.

3) Media Manipulation: Media manipulation, such as placement and size, has been simplified via the use of RenderableDataObjects. The framework has thus far left it up to individual applications to determine how to manipulate the media shown. Two primary modes of interaction become prevalent throughout the MediaCommons Framework development: manual via users and automatic via scripting.

For data analytics and guided tours through tiled display environments and ad-hoc presentations where users wished to interact with the media directly, manual controls were used. To compliment manual interaction on the planar tiled display wall, a widget system was developed which locks widgets to the screen and renders them orthogonally after the media objects have been drawn, preventing the widget interfaces from becoming hidden or unreachable. The widget system has been extended to include menus (see Figure 4), buttons and text annotations, which have been used to change application logic, toggle application options, browse the file system for media to load, perform application specific tasks, and display metadata about currently shown media objects.

Applications that are meant to run for an extended duration, such as ambient background visualizations or exhibitions that loop media content for multiple days, such as [15], benefit from non-manual interaction. Such applications built off of the MediaCommons Framework have included script parsing of XML files and an event loop to manage the loading, placement, scale, animation, and removal of media. As the media manipulation of RenderableDataObjects is supported by all MediaCommons applications, such applications' developers need only implement the logic to connect their scripts to the manipulation, without the need to address the complexities put forth by tiled display environments.

V. Conclusion

The MediaCommons Framework, incorporating modules from both CGLX and OpenSceneGraph, has provided a means for the development of media-centric graphical applications that can scale with respect to tiled display environments while abstracting complexities of the cluster environments away from media-application developers. Application developers are able to ignore the details behind rendering the supported media objects in the distributed rendering cluster and can instead focus on developing a consistent user experience for media exploration. By abstracting the element of the tiled display environment out of the application and into the framework, application developers can do a majority of their testing, prototyping and debugging on a single desktop computer, only requiring the tiled display environment for final analysis.

The MediaCommons Framework has been used to effectively deploy a handful of applications, targeting differing goals and disciplines. Applications using the MediaCommons Framework support numerous tours and presentations throughout Calit2 on a daily basis, and served visualizations for the Designing Geopolitics 2 [16] conference, the UCSD Alumni Gala 2012, and the "Exodus: Out of Egypt" [15] exhibition. All but one of the MediaCommons applications thus far have been developed in 3 to 6 days with only 1 or 2 developers. Although specific efforts have also been taken towards supporting cultural heritage [15] [17], such advances to the framework can be effective for any discipline wishing to quickly develop media-centric applications for tiled display environments, as evidenced by its usage within non-cultural events.

VI. Future Work

The MediaCommons Framework is still maturing as stable features within applications are pulled into the framework to be leveraged by future applications; specialized media types and differing scripting functionality being key examples.

Acknowledgment

We would like to acknowledge Tom DeFanti, Larry Smarr, Joseph Keefe and our colleagues at Calit2 for support and incorporation of MediaCommons applications since the framework's conception. Special thanks to the Calit2 GRAVITY group for the development of CGLX and its applications, upon which the MediaCommons Framework was built. Thanks to the CalVR development team for insight throughout development, and to the members of CISA3 whose data acquisition and analytics helped define the requirements which led to the MediaCommons Framework.

References

[1] T. A. DeFanti, J. Leigh, L. Renambot, B. Jeong, A. Verlo, L. Long, M. Brown, D. J. Sandin, V. Vishwanath, Q. Liu, M. J. Katz, P. Papadopoulos, J. P. Keefe, G. R. Hidley, G. L. Dawe, I. Kaufman, B. Glogowski, K.-U. Doerr, R. Singh, J. Girado, J. P. Schulze, F. Kuester, and L. Smarr, "The OptIPortal, a scalable visualization, storage, and computing interface device for the optiputer," *Future Generation Computer Systems*, vol. 25, no. 2, pp. 114 – 123, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X08001040

[2] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski, "Chromium: a stream-processing framework for interactive rendering on clusters," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 693–702, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566639

[3] (2013, Oct.) Distributed multihead X project. [Online]. Available: http://dmx.sourceforge.net/

[4] L. Renambot, A. Rao, R. Singh, B. Jeong, N. Krishnaprasad, V. Vishwanath, V. Chandrasekhar, N. Schwarz, A. Spale, C. Zhang *et al.*, "Sage: the scalable adaptive graphics environment," in *Proceedings of WACE*, vol. 9, no. 23. Citeseer, 2004, pp. 2004–09.

[5] B. Jeong, R. Jagodic, L. Renambot, R. Singh, A. Johnson, and J. Leigh, "Scalable graphics architecture for high-resolution displays," in *IEEE Information Visualization Workshop*, 2005.

[6] B. Jeong, L. Renambot, R. Jagodic, R. Singhm, J. Aguilera, A. Johnson, and J. Leigh, "High-performance dynamic graphics streaming for scalable adaptive graphics environment," in *SC 2006 Conference, Proceedings of the ACM/IEEE*, 2006, pp. 24–24.

[7] K. Doerr and F. Kuester, "CGLX: a scalable, high-performance visualization framework for networked display environments," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 3, pp. 320–332, 2011.

[8] J. P. Schulze, A. Prudhomme, P. Weber, and T. A. DeFanti, "Calvr: an advanced open source virtual reality software framework," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013, pp. 864 902–864 902.

[9] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The CAVE: audio visual experience automatic virtual environment," *Commun. ACM*, vol. 35, no. 6, pp. 64–72, Jun. 1992. [Online]. Available: http://doi.acm.org/10.1145/129888.129892

[10] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen projection-based virtual reality: the design and implementation of the CAVE," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '93. New York, NY, USA: ACM, 1993, pp. 135–142. [Online]. Available: http://doi.acm.org/10.1145/166117.166134

[11] (2013, Oct.) OpenSceneGraph. [Online]. Available: http://www.openscenegraph.org/

[12] S. Yamaoka, K.-U. Doerr, and F. Kuester, "Visualization of high-resolution image collections on large tiled display walls," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 498–505, 2011.

[13] J. Kimball, K. Ponto, T. Wypych, and F. Kuester, "RSVP: Ridiculously scalable video playback on clustered tiled displays," in *Multimedia, 2013. ISM'13. 15th IEEE International Symposium on*. IEEE, in-press.

[14] Z. Seldess, S. Yamaoka, and F. Kuester, "Sonnotile: Audio annotation and sonification for large tiled audio/visual display environments," *Auditory Display, Proceeding of the 17th International Conference on (ICAD-2011)*, June 20 - 24 2011.

[15] D. Srour, J. Mangan, A. Hoff, T. Margolis, J. Block, M. L. Vincent, T. A. DeFanti, T. E. Levy, and F. Kuester, *Israel's Exodus in Transdisciplinary Perspective - Text, Archaeology, Culture and Geoscience*. New York, USA: Springer, in-press, ch. MediaCommons Framework: An Immersive Storytelling Platform and Exodus.

[16] (2013, Oct.) Designing geopolitics 2. [Online]. Available: http://www.designgeopolitics.org/dgp2012/

[17] J. Mangan, D. Srour, A. M. Richter, A. Hoff, T. E. Levy, and F. Kuester, "MediaCommons for cultural heritage: Applied mixed media visualization storytelling for high resolution collaborative cyber archaeological displays," in *Digital Heritage, International Congress 2013*. Marseille, France: IEEE, 2013 in-press.

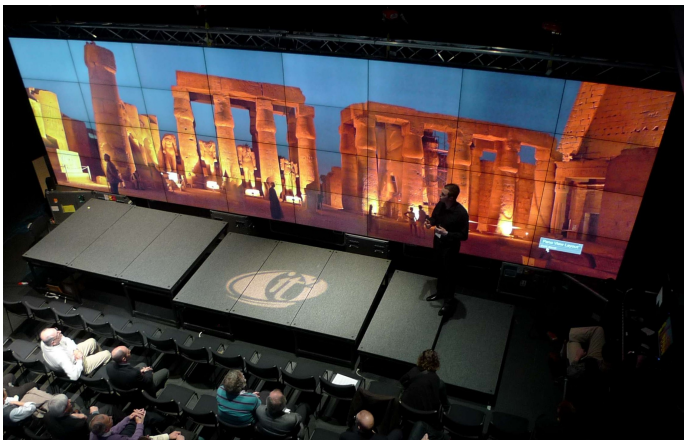

Fig. 1. A panoramic image of Luxor displayed in a MediaCommons application on a Calit2 tiled display environment made of OptIPortables.

Fig. 3. A 4K video of a tornado simulation rendered in a MediaCommons application via the tiled video playback library [13].

Requirement	Middlewares				
	Chromium	DMX	SAGE	CGLX	CALVR
Distributed Rendering	●	●	○	●	●
Distributed Execution	○	○	●	●	●
Extensible App. Logic	●	●	●	●	●
Abstracted Cluster Complexity	●	●	●	◐	◐
Communication/ Synchronization	○	○	○	●	●
Planar Interaction	◐	●	●	●	◐
Remote Environment Collaboration	◐	◐	●	◐	●
Independent Application Collaboration	○	○	◐	◐	◐
Extensible Media Types	○	○	○	◐	●
Scene Graph Optimizations & Management	○	○	○	◐	●

Fig. 2. Middleware Compliance Table: Records the level of support each middleware offers to the requirements enumerated in the Requirements section. Legend: ●Supported, ◐Undeveloped, ○Contrary to Architecture

Fig. 4. A MediaCommons application displaying 4K tiled videos, high resolution images, and panoramic images across a tiled display wall.