Workshop on Visualisation in Environmental Sciences (EnvirVis) (2017)
R. Bujack and A. Middel and K. Rink and D. Zeckzer (Editors)

Setting up Virtual Geographic Environments in Unity

K. Rink[1] and L. Bilke[1] and O. Kolditz[1,2]

[1]Department of Environmental Informatics, Helmholtz Centre for Environmental Research, Leipzig, Germany
[2]Chair of Applied Environmental System Analysis, TU Dresden, Germany

Abstract

We propose a setup for presenting complex heterogenous environmental data in a Virtual Reality environment. After a number of pre-processing steps required to integrate various data sets in a unified context, our workflow uses the Unity engine in combination with MiddleVR to create interactive applications for Virtual Reality environments to support research and evaluation for case studies containing domain-specific data from environmental monitoring and modelling. Typical challenges are described when handling such data in a graphics environment and solutions are proposed where possible. We give an overview of project structure and functionality added to Unity for visualisation of and interaction with the data and present an example application.

1. Motivation

Large-scale projects in environmental sciences, such as the development of water management schemes in arid regions [HMR09, GSSAS12], reduction of groundwater contamination [WDG*12, LWZ*15] or evaluation of possible effects of climate change [VMG*10, Int14] often rely on a large collection of data for a comprehensive evaluation of the current situation and are also needed for setting up models to predict future developments. The individual data sets often have vastly different properties. Examples include the structure, format, extents or geographic coordinate system. This complicates the integration into a unified context and the derivation of a numerical model for a subsequent simulation of environmental phenomena. In addition, data sets are often not consistent with each other due to differences in monitoring techniques, measurement procedures or post-processing methods applied by measurement devices or analysts [RBK14]. Finally, data sets may only be of limited use to a researcher or stakeholder outside of the scientific domain (e.g. a hydrogeologists might not be able to interpret raw climate data even though they may need to incorporate such data into their models). Therefore, it is necessary to find adequate methods for post-processing and visualising all data sets and finding useful visual metaphors for abstract or complex data.

2. State of the Art

Presenting complex data in a Virtual Reality (VR) environment is an excellent way for facilitating detailed discussions with larger groups [vDLS02, LJPF*08, Dem14]. If data sets are adequately prepared, the relation between data sets, the significance of certain parameters or the temporal evolution of phenomena can be visualised in a descriptive way, allowing even non-experts to understand consequences and complex interrelations of changes to an environment (e.g. increase/reduction or change of certain types of fertiliser or other chemicals, changes to an irrigation system or waste water management system, etc.).

A number of frameworks for VR environments have been presented in recent years. While some solutions focus on specific types of data [SLCI*13, Hod], others (incl. OpenSceneGraph [WQ12], the Vrui VR Toolkit [Kre08], OmegaLib [FNM*14] or COVISE [ASDL*16]) provide functionality and interfaces for developing monolithic, stand-alone applications. Missing functionality has to be programmed into the system, requiring an intimate understanding of the existing framework. Vrui is the only system offering an interactive generic application, the *3D Visualizer*. But due to slow development, few examples and a small user base it is tedious to develop applications for large, heterogenous environmental case studies.

After previously working with OpenSceneGraph and testing VRUI, we have opted to use Unity [Gol11, Uni] in combination with MiddleVR [Mid] for the construction of virtual geographic environments (VGEs). Similar to the other examples, Unity is platform-independent and available for free. It has a very large user base and many tutorials, extensions and guides are available. By using MiddleVR as a plug-in for Unity, a large range of rendering backends are instantly supported for compiling Unity-based applications for PC, web, VR environments or head-mounted displays (HMDs). This includes the automatic solution of typical VR challenges, such as clustered rendering (i.e. synchronisation of multiple applications running in parallel), the support of special input devices, incl. XBox-controller or Flystick, as well as tracking-functionality and various stereoscopic display modes [BFH*14]. Unity provides an interactive and extendable GUI for creating applications, allowing a 3D view of data sets right after import. Like-

DOI: 10.2312/envirvis.20171096

wise, animation and interactive functionality can be instantly tested within the software. Extensions are added by implementing the required methods in C#. Once functionality has been integrated, it is available within the Unity GUI and no further programming is necessary to create new applications with that same set of methods.

3. Environmental Data in Unity

In our Virtual Geographic Environments a large range of heterogeneous geoscientific data sets (e.g. remote sensing data, river networks, land use classes, climate data, etc.) can be explored in an interactive 3D scene. In addition to monitoring data, these applications also include results from the simulation of (hydro-)geological processes and phenomena. Such models are created based on the properties of the surface, stratigraphic information, location of water bodies, etc., so their representation in the same context as the base data sets is helpful for the evaluation of the characteristics of the model. Boundary conditions and source terms, namely precipitation, in- and outflow of water bodies, distribution and concentration of chemicals, etc. can be visualised in an intuitive, geospatial context. A suitable visualisation of simulation results allows to better assess plausibility and estimate implications for the environment.

Environmental data is usually provided in a multitude of formats. Examples include formats used in Geographic Information Systems (GIS) such as shape-files or GeoTIFF-rasters, formats used by modelling software like FEFLOW [DHI], Petrel [Sch], GOCAD [Par], etc. as well as multipurpose formats such as VTK [SML06] or NetCDF [RD90]. Pre-processing tools are needed for matching and mapping all data sets, removing artifacts, and creating a 3D representation of the data. We recommend doing adjustments as much as possible in the original software. Specifically GIS usually have a large set of tools for simplifying, filtering or even restructuring data sets. Simulation software mostly does not offer such functionality. Therefore, we employ two VTK-based open-source applications for processing and visualising data sets:

- **Paraview** [AGL05] offers a large range of established data analysis and visualisation algorithms for visualising large and complex data sets. Unfortunately, file formats from GIS or modelling software are mostly not supported.
- The **OpenGeoSys Data Explorer** [RBK14] is the graphical user interface for the numerical simulation software OpenGeoSys [KBB*12]. It can import a large range of file formats used for GIS- and modelling software and offers some specific functionality for restructuring and analysing such data.

Both frameworks can handle VTK-data structures and file formats and complement each other when handling environmental data. Once preprocessing is finished, all relevant data is written into Autodesk FBX format. We chose this particular format because it is supported by a large range of software frameworks, including Unity, 3D Studio MAX, or Maya. We used FBX SDK [Aut] to implement a converter from VTK to FBX that already automatically solves a number of potential problems when introducing this kind of data to the Unity framework.

In addition to technical aspects such as data conversion, a number of domain-specific and computer graphic related challenges have to be considered when introducing geoscientific data to Unity:

Coordinate Systems: Geoscientific data is usually referenced in a geographic coordinate system. There are hundreds of such systems, either employing a spheroid using latitude and longitude (e.g. the World Geodetic System WGS1984) or a map projection measuring the distance in x and y from a preassigned origin, usually in metres (e.g. the Universal Transverse Mercator (UTM) system). Both types are not *per se* suitable for use in Unity (or 3D computer graphics in general). The spheroid approach will cause vast distortions in z-direction which is usually given in metres. But there will also be distortions in x- and y-directions (similar to a fisheye-lens effect) if the region of interest is reasonably large. The map projection approach is in theoretically feasible but will result in very large coordinates for all objects in the scene (usually somewhere in the 10^5–10^7 range). This results in problems, especially with smaller objects, when exact coordinates are of vital importance but cannot be assigned correctly anymore due to floating point precision errors.

To solve this issue, we implemented functionality that will calculate the bounding box B of all data sets in the scene and then translate and scale B around the coordinate origin. This results in smaller coordinates and a more viable 3D effect in the VR environment. We assume that all data sets use the same projection. This is rarely the case when acquiring the data but can be solved using map projection methods implemented in GIS or other specialised software. If this requirement holds, all data sets will be located correctly relative to each other and can be translated and scaled without any changes to that relative position.

Large Data Sets and Accuracy: Computer generated scenes, particularly in games, often make use of techniques to increase performance and reduce memory by simplifying objects in a statistically reasonable way, using techniques such as bump mapping (for surfaces) or mipmapping (for textures) to make scenes appear more detailed than they actually are. For geoscientific data these approaches are not applicable. Detailed data is often difficult to obtain and while they occasionally can be simplified by interpolation, objects will still often consist of millions of triangles to represent all details necessary for a professional assessment. Likewise, textures are created based on remote sensing imagery. Contrary to gaming applications, textures are neither clamped nor repeated but instead need to have an extent as large as the object they are mapped on. Unity has predefined limits for both the triangle count as well as for texture resolution. Specifically, objects consisting of more than 2^{16} vertices will be automatically split into multiple "sub-meshes", such that the number of vertices in each sub-mesh is below that threshold. However, this automatic splitting often results in rather random patches (see Fig. 1a) that will cause problems for post-processing and result in a high rendering workload. Therefore, we implemented an algorithm that will split surfaces into roughly square patches (Fig. 1). This increases performance due to view-frustum culling as not all parts of a large surface will need to be rendered when the user is located within the scene. It also allows to subdivide very large textures (i.e. remote sensing imagery) into smaller images and use those on the parts of the surface. Since the

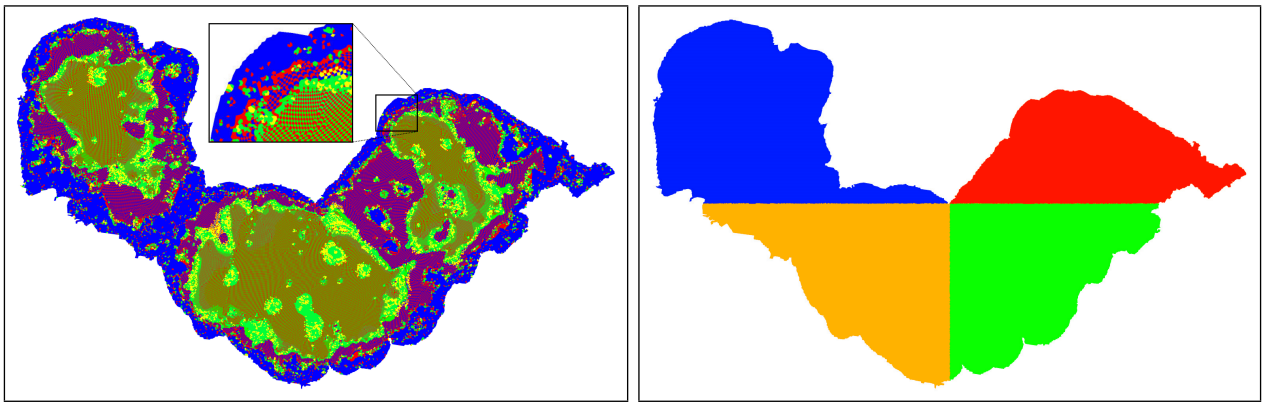

Figure 1: *Subdivision of meshes: Example of a Finite Element Mesh consisting of* 330 000 *triangles. The left image shows the automatic partition of the object by Unity. The right image shows the result of our algorithm. Both methods subdivide the mesh into four parts but our algorithm provides a tiled partition of roughly rectangular submeshes better suited for post-processing and rendering.*

maximum allowed texture size is 4096×4096 pixels, this additional step is occasionally necessary, for instance when visualising the surface of a research area.

Object Extents: Another challenge is the handling of objects with varying extents. This is a more familiar problem in visualisation although usually not as extreme as when handling geoscientific data. It is not unusual for case studies with region of interest (ROI) of hundreds or thousands of km^2 to include data sets that have an extent of $1\,km^2$ or are even smaller. One of the reasons is that small *intensive test sites* are used to get a detailed picture of processes in a small region with typical region-specific parameters (e.g. the groundwater regime on a hill slope consisting of a certain hydrostratigraphic composition) and extrapolate the results to the ROI. Further examples include data acquired from close-meshed sensor networks (e.g. for measuring soil moisture), natural formations (e.g. lakes,) or structures (e.g. buildings such as dams or landmarks). Since Unity uses coordinates with float-precision, small objects in the scene might not be represented as exactly as necessary, resulting in a jittering effect during movement. This can be partly improved by adjusting the bounding box of the scene but may remain a problem still.

Game engine limitations: Since Unity is a game engine, a number of features supported by a scientific visualisation library such as VTK are missing, as they are not usually needed for game design. Necessary additions to the existing functionality required for using geoscientific- and modelling data include:

- **Creating Materials:** Objects in Unity are assigned a material (i.e. shader with a certain parameterisation). Because of Unity's focus on game development, the predefined materials were not sufficient for the way we need to represent geoscientific data. Besides basic properties such as diffuse/ambient colour or texture, particularly the lack of vertex colours required the definition of new materials. In addition, some of the data sets we integrate require backface culling while others do not. Also, data sets are sometimes rendered in a semi-transparent state (e.g. the model domain, to allow the display of processes inside while still maintaining a view of the overall extent and stratigraphic partition). To display all data sets correctly, materials have to be defined for a number of different render pipelines due to the combination of the above properties.
- **Cell scalars:** VTK allows the allocation of data (scalars, vectors, etc.) to either vertices or cells of a mesh. In many numerical models cells represent a discrete part of the domain with certain properties such as permeability, conductivity or the concentration of a given chemical compound. Even after introducing our own material-definitions, Unity only allows additional data to be assigned to vertices of the mesh, thus requiring the duplication of vertices to faithfully represent properties of the model.
- The standard FBX-import of Unity ignores objects consisting exclusively of points or line elements. However, such **point- or line meshes** are required to represent a number of typical data sets in geoscientific studies, including boreholes, river- or sewer networks or streamlines. Fortunately, in recent years Unity has added the capability to display these types of data so writing our own interpreter to build line meshes from imported FBX files solved the issue.

The resulting 3D scene (Fig. 2) allows free movement of the user using a number of established devices, including keyboard, mouse, gamepad, HMD-controllers or Flystick. To simplify navigation and data access, we implemented a menu that can be toggled by pressing a key, either on the keyboard or the corresponding device. It will always appear in correct orientation to the user and can be moved to any position by picking it with the input device so it does not occlude important information. A user can select between the following options:

- **User-controlled data visibility:** The visibility of objects or groups of objects can be switched on or off. In addition, a slider allows setting a specific transparency.
- **Viewpoints:** A scene can contain a number of named viewpoints, i.e. pre-defined positions of the camera. By accessing one of these in the menu, the camera will "fly" to the respective position via automatically interpolated or pre-defined fly-paths.
- **Animations:** If a scene contains animations such as time steps

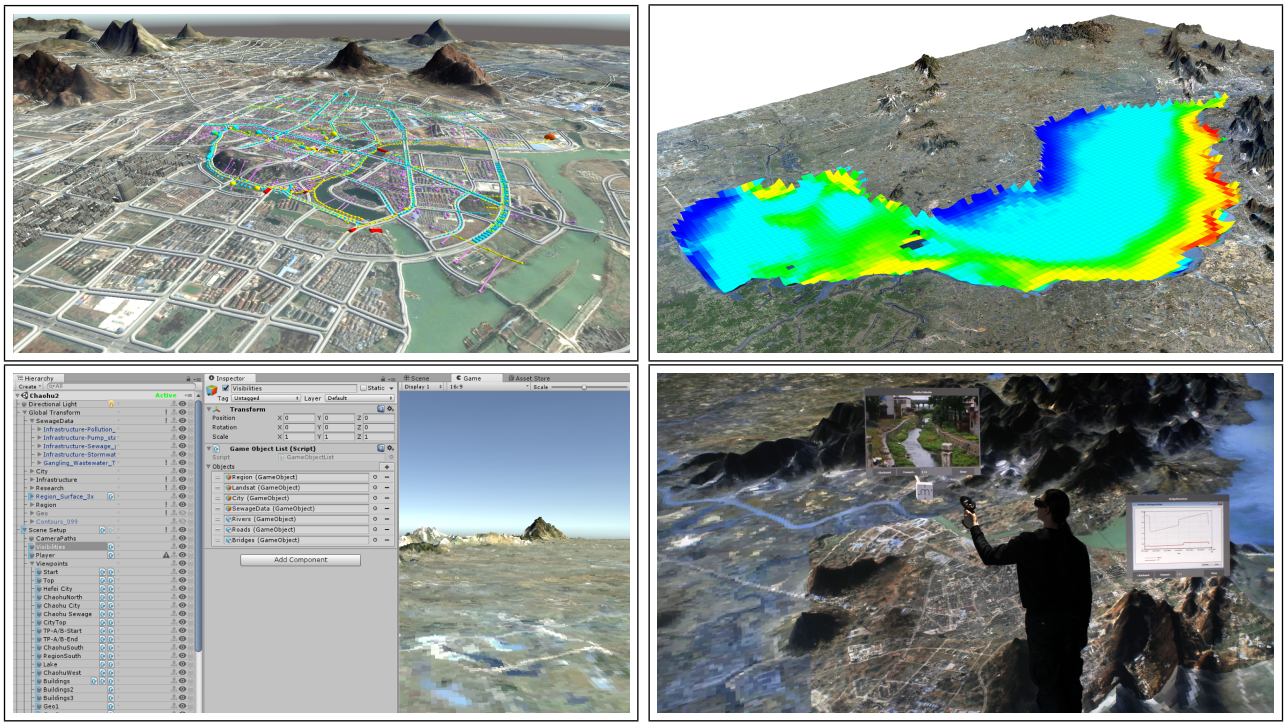

Figure 2: *Example of a virtual geographic environment:* ***(a)*** *screenshot of monitoring and administrative data (3D surface with aerial image, streets and sewage network)* ***(b)*** *time-step from a transient hydrological simulation* ***(c)*** *Unity project structure incl. self-implemented functionality* ***(d)*** *application in VR environment with open overlays.*

from climate measurements or the transient simulation of certain processes, the playback can be started, stopped or paused from the menu and individual time steps can be selected. Note that "animation" does not refer to actual changes in scene objects but instead the sequent display of discrete time steps of 4D data sets. In the same way, it is possible to switch between different **simulation scenarios.**

In addition, we added functionality for creating a 3D scene. This includes:

- **Fly paths:** While selecting viewpoints will automatically calculate a path to the destination by interpolation position and camera orientation, paths can also be pre-defined when creating the scene. Here, the exact route and orientation of the camera can be adjusted.
- **2D Overlays:** Additional information can be accessed by picking objects in the scene. This will open up an overlay window (similar to the menu) where photos, diagrams, websites, videos, etc. related to the picked object can be displayed. This functionality is embedded via MiddleVR and includes its own wrapper for interaction.
- **Automatic data visibility:** The opacity of data sets can be changed automatically when entering certain regions within the scene (i.e. predefined bounding boxes that trigger the execution of a script) or selecting predefined viewpoints. This is equivalent to user-controlled changes from the menu but happens automatically and is thus more useful during presentations.

Finally, we have added a template for scenes to Unity that allows making all of the functionality described above available in any Unity project with just a few mouse clicks. Unity stores projects in a tree-structure. For the purpose of creating virtual geographic environments, we added certain constraints to that tree such that all projects have the same layout. For instance, one child of the root node will always contain a subtree with all the data sets, another child will contain all predefined viewpoints, yet another child will contain scripts for all pickable objects, etc. (Fig. 2c).

The resulting package of pre-processing tools and Unity extensions allows to set up new scenes in a short amount of time, without additional programming and without considering technical aspects of the subsequent architecture used for presentation. An extensive overview of the application side of our workflow has been presented in a separate article [RCB*16] with a focus on the hydrogeological application of virtual geographic environments.

4. Conclusions & Outlook

We presented a workflow for creating virtual geographic environments using the Unity game engine. Starting with a representation of the data suitable for visualisation, we provide methods for converting and importing data sets into Unity. While a number of small adjustments have been necessary to handle the particular structure and properties of geoscientific data sets, we have been largely able to profit from the functionality already provided by Unity and MiddleVR. Templates and integrated extensions for Unity ensure a fast

setup of new scenes while still keeping the system open for adding more functionality. In the future we plan the integration of the VTK visualisation pipeline in Unity, allowing for the application and adjustment of visualisation algorithms during runtime of the application. The resulting presentations will offer more possibilities for interaction by allowing changes to the parameterisation of presentation such as selection from multiple scalar arrays associated with data sets, changes of colours, cutting planes, glyph parameters or seed points for random selections within large data sets.

Acknowledgements: Funding for this research work was provided by the German Federal Ministry of Education and Research (BMBF) CLIENT program "International Partnerships for Sustainable Innovations" for the project "Managing Water Resources for Urban Catchments" (grant number 02WCL1337A) and the Helmholtz Research Network "Research Centre for Environmental Information Science" (grant number HIRN0001).

References

[AGL05] AHRENS J., GEVECI B., LAW C.: ParaView: An End-User Tool for Large Data Visualization. In *The Visualization Handbook*, Hansen C., Johnson C., (Eds.). Elsevier, 2005. 2

[ASDL*16] AUMUELLER M., SCHULZE-DOEBOLD J., LANG R., ET AL.: *COVISE User's Guide*. Tech. rep., High-Performance Computing Center, Stuttgart, 2016. 1

[Aut] AUTODESK: FBX SDK. http://www.autodesk.com/products/fbx. Accessed: 2017-02-15. 2

[BFH*14] BILKE L., FISCHER T., HELBIG C., ET AL.: TESSIN VISLab – Laboratory for Scientific Visualization. *Environ Earth Sci 72*, 10 (2014), 3881–3899. doi:10.1007/s12665-014-3785-5. 1

[Dem14] DEMIR I.: Interactive Web-based Hydrological Simulation System as an Education Platform. In *Proc of 7th Intl. Congress on Env. Modelling and Software* (2014), iEMSs. 1

[DHI] DHI GROUP: FEFLOW. https://www.mikepoweredbydhi.com/products/feflow. Accessed: 2017-02-15. 2

[FNM*14] FEBRETTI A., NISHIMOTO A., MATEEVITSI V., ET AL.: Omegalib: a Multi- View Application Framework for Hybrid Reality Environments. In *Proc of IEEE Virtual Reality* (2014), IEEE, pp. 9–14. doi:10.1109/VR.2014.6802043. 1

[Gol11] GOLDSTONE W.: *Unity 3.x Game Development Essentials (2nd Edition)*. Packt Publishing, 2011. 1

[GSSAS12] GRUNDMANN J., SCHÜTZE N., SCHMITZ G. H., AL-SHAQSI S.: Towards an integrated arid zone water management using simulation-based optimisation. *Environ Earth Sci 65*, 5 (2012), 1381–1394. 1

[HMR09] HÖTZL H., MÖLLER P., ROSENTHAL E.: *The Water of the Jordan Valley*. Springer, 2009. 1

[Hod] HODGETTS D.: VRGeoscience – Virtual Reality Geological Studio. http://www.vrgeoscience.com. Accessed: 2017-02-15. 1

[Int14] INTERGOVERNMENTAL PANEL ON CLIMATE CHANGE: *Climate Change 2014 – Impacts, Adaptation and Vulnerability: Regional Aspects*. Cambridge University Press, 2014. 1

[KBB*12] KOLDITZ O., BAUER S., BILKE L., ET AL.: OpenGeoSys: An open source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media. *Environ Earth Sci 67*, 2 (2012), 589–599. 2

[Kre08] KREYLOS O.: Environment-Independent VR Development. In *Advances in Visual Computing*, vol. 5358 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 901–912. doi:10.1007/978-3-540-89639-5_86. 1

[LJPF*08] LAVIOLA JR J., PRABHAT, FORSBERG A., ET AL.: Virtual Reality-Based Interactive Scientific Visualization Environments. In *Trends in Interactive Visualization* (2008), Springer, pp. 225–250. 1

[LWZ*15] LIU C., WANG Q., ZOU C., ET AL.: Recent trends in nitrogen flows with urbanization in the Shanghai megacity and the effects on the water environment. *Environ Sci Pollut Res 22* (2015), 3431–3440. 1

[Mid] MIDDLEVR DEVELOPERS: MiddleVR SDK – a generic immersive virtual reality plugin. http://www.middlevr.com/middlevr-sdk/. Accessed: 2017-02-15. 1

[Par] PARADIGM: GOCAD. http://www.pdgm.com/products/gocad/. Accessed: 2017-02-15. 2

[RBK14] RINK K., BILKE L., KOLDITZ O.: Visualisation Strategies for Environmental Modelling Data. *Env Earth Sci 72*, 10 (2014), 3857–3868. doi:10.1007/s12665-013-2970-2. 1, 2

[RCB*16] RINK K., CHEN C., BILKE L., ET AL.: Virtual Geographic Environments for Water Pollution Control. *Int J Dig Earth* (2016). doi:10.1080/17538947.2016.1265016. 4

[RD90] R. R., DAVIS G.: NetCDF: an interface for scienti?c data access. *IEEE Comp Graph and App 10* (1990), 4. 2

[Sch] SCHLUMBERGER: Petrel E&P Software Platform. https://www.software.slb.com/products/petrel. Accessed: 2017-02-15. 2

[SLCI*13] SUBBARAO M., LINDBERG CHRISTENSEN L., ITOH S., ET AL.: Bringing Data to the Dome: Scientific Visualization for all Planetariums. *The Planetarian 42*, 4 (2013), 20–22. 1

[SML06] SCHROEDER W., MARTIN K., LORENSEN B.: *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics (4th Edition)*. Kitware, Inc., 2006. 2

[Uni] UNITY TECHNOLOGIES: *Unity User Manual*. https://docs.unity3d.com/Manual/UnityManual.html, Accessed: 2017-01-20. 1

[vDLS02] VAN DAM A., LAIDLAW D., SIMPSON R.: Experiments in Immersive Virtual Reality for Scientific Visualization. *Comput Graph 26* (2002), 535–555. 1

[VMG*10] VÖRÖSMARTY C. J., MCINTYRE P. B., GESSNER M. O., ET AL.: Global threats to human water security and river biodiversity. *Nature 467* (2010), 555–561. 1

[WDG*12] WALTHER M., DELFS J. O., GRUNDMANN J., ET AL.: Saltwater intrusion modeling: Verification and application to an agricultural coastal arid region in Oman. *J Comput Appl Math 236*, 18 (2012), 4798–4809. 1

[WQ12] WANG R., QIAN X.: *OpenSceneGraph 3 Cookbook*. Packt Publishing, 2012. 1