ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,
Volume II-2/W1, ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop, 27 – 29 November 2013, Istanbul, Turkey

3D VISUALISATION OF UNDERGROUND PIPELINES: BEST STRATEGY FOR 3D SCENE CREATION

J. Guerrero, S. Zlatanova, M. Meijers

GIS Technology, OTB Research Institute for the Built Environment, Delft University of Technology, Jaffalaan 9, 2628 BX, Delft, The Netherlands (josafatisai@gmail.com, s.zlatanova@tudelft.nl, b.m.meijers@tudelft.nl)

Commission II, WG II/2

KEY WORDS: Scene creation, X3D, Web, 3D modelling

ABSTRACT:

Underground pipelines pose numerous challenges to 3D visualization. Pipes and cables are conceptually simple and narrow objects with clearly defined shapes, spanned over large geographical areas and made of multiple segments. Pipes are usually maintained as linear objects in the databases. However, the visualization of lines in 3D is difficult to perceive as such lines lack the volumetric appearance, which introduces depth perception and allows understanding the disposition and relationships between the objects on the screen. Therefore the lines should be replaced by volumetric shapes, such as parametric shapes (cylinders) or triangular meshes. The reconstruction of the 3D shape of the pipes has to be done on the fly and therefore it is important to select a 3D representation which will not degrade the performance. If a reconstruction method provides a good performance, the visualization of pipes and cables is guaranteed to provide a smooth experience to the final user, enabling richer scenes but also establishing the visualization requirements in terms of hardware and software to display underground utilities.

This paper presents our investigations on a strategy for creating a 3D pipes for 3D visualisation. It is assumed that the pipelines are stored in a database and portions of them are retrieved for 3D reconstruction and 3D visualization. Generally, the reconstruction of underground utilities can be performed in different ways and should lead to realistic appearance, produce visual continuity between segments, include objects depicting specific connections and even consider buffer volumes displaying the uncertainty and the security distance between objects. The creation of such visually pleasing reconstructions may require very detailed shapes, which will increase the complexity of the scene and degrade the performance. This research has identified four criteria to measure the complexity of the scene and conclude on a 3D reconstruction strategy: number of scene graph nodes, number of triangles and vertices on the screen, needed transformations and appearance options. On the basis of these criteria a testing framework is developed. Ten different strategies for 3D reconstruction are defined and tested for X3D, X3DOM and WebGL. The paper analyses the results of the tests and concludes on the best strategy.

1. INTRODUCTION

Many utility networks are currently managed as 2D/3D line objects with attributes in databases. This representation is sufficient for performing a variety of tasks but faces numerous challenges when these data have to be visualised in 3D environments. The visualization of 3D lines on the screen is often unclear as it lacks the volumetric appearance, required to produce depth perception, which is the key issue to understand the disposition and relationship of the objects on the screen. Lines displayed on current graphics hardware cannot be shaded as do not have any volume or surface, and as consequence occlusion and relative size are impossible to achieve. Their portrayal on 2D or even 3D displays cannot provide any depth information. With lines only, it is impossible to determine the closest object and the understanding of the scene becomes difficult if not impossible. Many projects have been initiated to investigate these issues, e.g. DeepCity3D (http://www.deepcity3d.eu) or 3DSDI (http://maasvlakte2-3dsdi.ddss.nl/).

To solve this problem, volume should be added to non-volumetric 3D lines, i.e. their 3D shape has to be *reconstructed* by creating the outer shell of the desired object and making them suitable to real-time rendering using computers equipped with Graphics Processing Units (GPUs).

Substantial research has been carried out for working with underground utilities information on different aspects. Part of it concerns the computer models and storage solutions for pipes and cables. Another relevant part for this research deals with the transformation of GIS data into visual representations, both in 2D or 3D. Research on modelling and visualization of utilities in 3D has been done by Du and Zlatanova (2006), where information is transformed into 3D objects and visualized on the fly as shapes with volumetric appearance and symbols depicting special pipeline attachments. A corresponding prototype is presented using a spatial database as storage solution, implemented with Oracle Spatial built-in spatial types. Results are visualized and transformed using MicroStation and the Java edition of MicroStation Development environment (JDML) to convert on the fly the lines into 3D shapes.

This desktop platform can be seen as a limiting factor towards the seamless distribution of information to different stakeholders. The benefits of managing centralized utilities information using DBMS and providing 3D visualization have been shown of critical importance by Zlatanova et al. (2011), revealing better relationships between pipes and objects, making easier the visual inspection while reducing the misunderstanding to the minimum.

The management and registration of utility networks in 4D (space + time) using a spatial DBMS is presented as a promising solution to maintain centralized management and a correct registration of legal rights and obligations, facilitating the analysis and comparison with the related parcels (Döner et al., 2011), as some legal aspects can only be solved using 3D information to determine its spatial relationship with utilities above or under the ground. Work on overlaying utilities information over panoramic images has been studied by Verbree et al. (2004), addressing the problem of understanding maps while translating their contents into reality and the other way around, closing the gap between geo-referenced information and augmented reality.

The process of creating 3D visualizations from 2D geographical sources on the fly has been elaborated by Schall et al. (2008) with a *transcoding pipeline*. This process separates the model content from the presentation, allowing generating temporary 3D models on demand without storing them. This transcoding pipeline requires GIS data, rules for the model generation and styles for visualization, but also a 'scene graph' specification to represent the transcoded model. This process has been applied by Schall et al. (2010) for modelling underground utilities on mobile devices for Augmented Reality applications, consuming geographical information encoded in Geographical Modeling Language (GML) and converted into 'scene graphs'. Standardization efforts towards separating content from presentation are proposed in (Basanow et al., 2008) and presented via 'Styled Layer Descriptors'. A specific description useful for visualization of underground utilities has been proposed for lines, reconstructing them based on radius and colour information.

So far, most of these works have been done on the desktop or for a web environment *using plugins* to display 3D content. With the advances on the Web, WebGL has appeared as a technology for displaying 3D content without the need of plugins. Given the novelty of this technology and the interpreted nature of the JavaScript language, research performed towards the suitability of this technology has not been carried out and moreover, the implications on the transcoding procedures on performance have not been addressed.

This paper presents a framework for 3D reconstruction of pipes and cables for visualization on the Web. The framework is tested for visualization with X3D, X3DOM and WebGL. This paper is organized as follows: The next section presents the 3D reconstruction flow, section 3 discusses the implementation and the tests, Section 4 concludes on the results of the tests.

2. RECONSTRUCTION FLOW

The proposed reconstruction flow is an abstraction created to understand and implement the reconstruction process of the studied geometrical features. This flow is basic for implementing 3D web applications and resembles the proposal of Schall et al. (2008) to convert the 2D geospatial data into 3D models and deliver them through 3D scenes. This delivery mechanism is a concept also used by Altmaier and Kolbe (2003), Heinen et al. (2005), Basanow et al. (2008), where Web 3D Service (W3DS) and other related Web Services encapsulate the server side functionality and deliver geographical scenes.

1. The reconstruction flow (Figure 1) starts with the information stored into a spatial database, from which an arbitrary user request triggers a query (spatial or not) to the data store, producing a series of results organized in tables where each row or record returned corresponds to a feature in the database, composed of a geometry definition and a set of attributes.

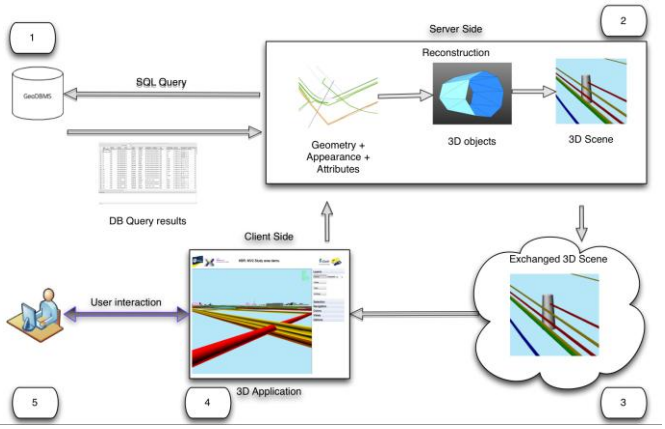


Figure 1: 3D reconstruction flow for pipe lines and cables

2. Given the geometry, the attributes and appearance mapping, the actual creation of the 3D objects starts by choosing a reconstruction approach. The chosen approach transforms the geometric object with lower dimensionality into a 3D object based on its own conversion rules. Depending on the method used, graphical primitives or custom 3D meshes can be used in replacement to construct the objects and reused along the procedure.

3. After all the pipes and cables are reconstructed; the scene is assembled by linking the produced 3D shapes to their appearance, including identifiers, actions and names for object picking (e.g. highlighting their attributes by hovering over with a pointer device).

4. When a scene is finally assembled, it is made available to the 3D engine which parses the information to produce an internal representation suitable for rendering. In this step, the declared objects, materials, identifiers, names and additional information are converted into a scene graph.

5. After creating the scene graph, the objects are displayed to the user as sequences of two-dimensional images producing the illusion of movement. If the user interacts with the scene this requires different elements from those present on the scene, the 3D application redirects that request into a database query, starting a new reconstruction flow and displaying the new elements to the user.

3. THE FRAMEWORK FOR RECONSTRUCTION

The reconstruction flow described before requires the choice of a reconstruction method to convert the 2D linear underground objects into 3D volumetric shapes suitable for visualization and interaction. The reconstruction methods set high level rules and procedures for interpreting the 2D data, but also define low level procedures to create a set of scene graph nodes representing the objects.

The high level modelling considers two approaches: the first considers that an underground object is modelled with simple independent objects, producing multiple scene graph nodes. The second approach considers that the smaller parts composing underground utilities are dependent on each other and should

produce a bigger single entity. Reconstruction approaches following the first category are termed here as *Split* methods while the others are termed *Non-split* methods.

In the low level, specific decisions for creating the scene graph are taken and the basic scene graph nodes considered for this purpose are adapted from (Strauss and Carey, 1992) and include:

Shape Nodes represent geometric or physical objects. The Shape nodes are leafs on the graph and are associated to low-level representations of triangulated models. Examples of those include Indexed Face Sets, Triangle Strips, Triangle Fans, Indexed Triangle Meshes, Line Sets, among many others. These nodes can also include definitions for common objects like Cone, Sphere, Cylinder, etc. For the reconstruction purposes, only indexed representations are considered but specifically, the Indexed Triangle Meshes, which holds a list of vertices, their normals and list of triangles representing the 3D object.

Group Nodes are to connect other nodes into graphs or sub-graphs. Examples include the Switch node, useful to implement libraries of objects and materials. The Group node is also useful to aggregate multiple nodes and share common attributes.

Property Nodes describe attributes of the objects related to the appearance of the objects, necessary to provide distinctive appearance. Examples of the used classes include *BaseColor*, *Material*, *Normal*, *Texture*, *Transform* (for affine transformation), among many more.

With these elements in consideration, four basic decisions are taken towards reconstructing the objects:

1. Geometry The first choice corresponds to the geometry used to encode the objects, which in this case can be based on primitives like the *Sphere* or *Cylinder* or arbitrarily defined using *Indexed Triangle Meshes*.

2. Transformations are a property node and one of the basic operations used to scale, rotate or translate siblings of that node. Geometric primitives and templates need to be transformed every time to place them on the desired positions.

3. Object Reuse *Scene graphs* allow creating objects and reusing their definitions, acting as templates for the creation of other similar objects. Such objects should be defined on local coordinates so transformations parameters, can be provided to transform them during runtime. An example of template could be a traffic sign used, which can be further reused by providing only its transformations parameters.

4. Material Reuse Finally, every scene graph node requires the definition of appearance in order to be rendered. The hierarchical structure of the scene graphs allows sharing appearance definitions among its siblings, but some 3D engines also allow sharing definitions across different nodes. Similarly to the object reuse, sharing material definitions reduce representation space but also can trigger optimizations in the rendering engine to avoid state changes and improve rendering performance.

Figure 2 shows all the possible reconstruction paths obtained when considering all the decisions, adding up to 12 possibilities. Paths reaching the right side are considered possible and depicted with continuous lines, otherwise denoted with dotted lines.

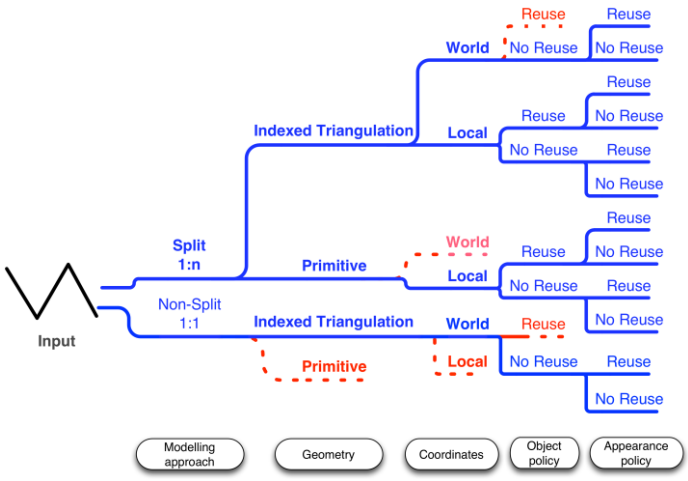


Figure 2: Possible reconstruction paths

3.1 Split reconstruction methods

Split method 1: Primitive based. This method involves the use and reuse of the Sphere and Cylinder primitives and the definition of their transformation parameters to model each element of the pipe with a primitive. The transformation parameters take the geometry of the primitives and transform them on its final configuration.

Split method 2: Custom Primitive based. The difference with the first one is the replacement of the built-in primitive with a custom geometry to control the appearance quality of the objects. Besides the use of different primitives based on the same definition, the rest of the procedures are identical to the previous method so no further details are provided to compute the transformation parameters; however the details of the actual reconstruction of the objects are important to understand the differences on final performance and the size of the objects.

Split method 3: World based primitives. In this case, each cylinder or sphere or created in their final position, i.e. expressed in world coordinates instead of local coordinates. The reason to follow this approach is to skip transformations during runtime in order to save processing time. A drawback of this method is the increase in storage to define each unique object instead of reusing primitives along the scene.

3.2 Non-split reconstruction methods

The second main branch of the reconstruction taxonomy corresponds to approaches where the reconstructed objects are modelled visually and logically as a single scene graph node. The difference imposes a stricter control on the modelling of objects involving more restrictions in order to reduce the number of scene graph nodes representing the object. A first simple and naive approach just appends all the objects modelled with triangles into a single mesh before rendering, reducing the final object count but not the triangle and vertex count. A further refinement referred here as 'stitching', requires computing the actual intersection points between the composing objects, avoiding invisible triangles, allowing vertex recycling and storage savings.

Non-Split method 1: Appending world based geometries
One of the main drawbacks of the 'split' methods is the increase of the number of scene graph nodes, leading to additional render calls or *batches* per modelled object. To deal with this situation, an important fact about the *scene graphs* can be used: every node is an explicit list of independent triangles. These independent triangle lists can be 'appended' at the end of each other and drawn instead within a single render call or 'batch', as long as they belong to the same administrative object and share appearance options.

Non Split method 2: Stitching world based geometries
The final approach is an improvement over previous methods aimed to reduce the vertex count, triangle count and scene graph node count by sharing vertices, avoiding spheres and modelling the pipe as a single object without breaks and ruptures. The basis of this method is the removal of the spheres placed between consecutive cylinders and adjusting their length so they can match in both ends. To accomplish this, the pipes are extended and cut at the bisecting plane between two consecutive centrelines.

The reconstruction methods can follow more than one path due to variations in the appearance reuse policy and if possible in the object reuse policy. Split Methods 1 and 2 follow 4 paths each, Split Method 3 follow only 2 paths, while Non Split Methods 1 and 2 follow 2 paths each one. All the possibilities add up to 14 paths to test and cover the five methods and their variations.

4. IMPLEMENTATION AND TESTS

Based on the presented decision structure, the performance of each reconstruction method can be directly evaluated. However, due to the dependency on the input objects, varying in length, span area, and other details affecting the scene complexity and the rendering performance, the high level decisions are ignored, keeping only the low level decisions for the tests.

With only four parameters, abstract tests are created and only low-level decisions are tested instead of the full reconstruction methods. With the remaining parameters, only 10 combinations are left. In practise 9 are tested since the last one, i.e. no primitives, no transformations, no object reuse and no material reuse, is not expected to bring advantages over the previous ones.

4.1 Abstract tests
The abstract tests replace specific objects, pipes in this case, with spheres, which are geometrical objects with high triangle count capable of represent objects with less or equal triangle count. GPU's do not differentiate the rendered objects; they only process triangles and their corresponding vertices (Figure 3). In addition, the sphere is a scene graph primitive available in various frameworks, making it suitable for testing the primitive cases and their optimizations.

For every test, a specific number of unitary spheres are randomly placed within a 100 x100 x 100 cube to reduce the number of occlusions while making them mostly visible within the camera viewport. Rotating the camera around the cube simulates user interaction.

For each decision path, an increasing number of nodes (50, 300, … , 2050) with an increasing triangle count per object is used (128,512,1152), running the tests for 30 seconds and collecting the number of displayed frames per second.

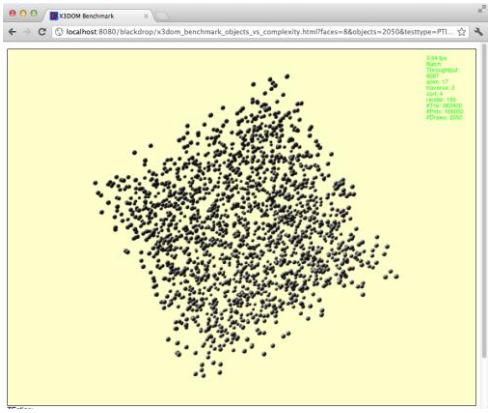


Figure 3: The spheres used for the abstract test

4.2 Tests implementation
The testing framework is implemented in a web environment using WebGL and X3D/X3DOM as the underlying technology stack. Tests are automated via custom code, simulating both the workload and user interaction. On the server side a Web server is used for dispatching files and a Servlet Container for storing results in a database (PostgreSQL). Figure 4 depicts the system architecture used for the tests.

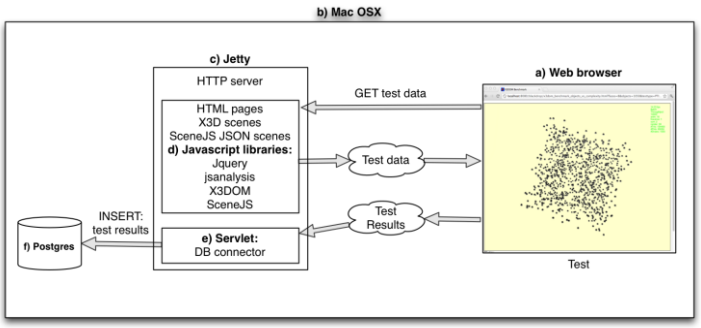
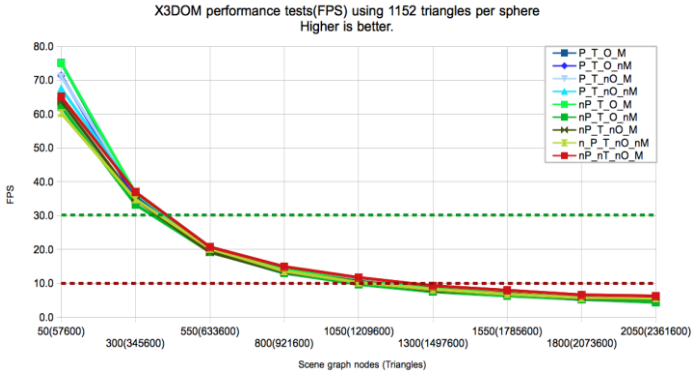


Figure 4: System architecture

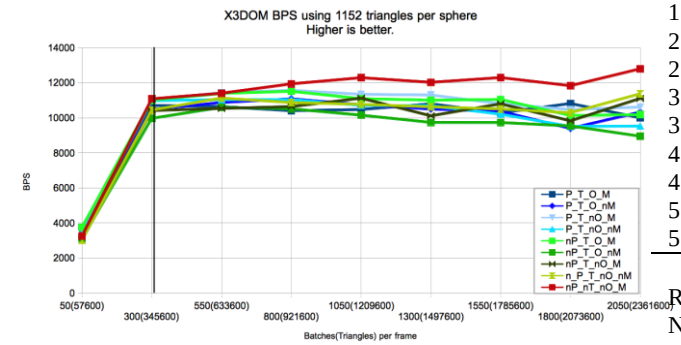
Results obtained are shown in Figures 5: a) Frames per second (FPS), b) Batches per second (BPS) and c) Triangles per second (TPS).



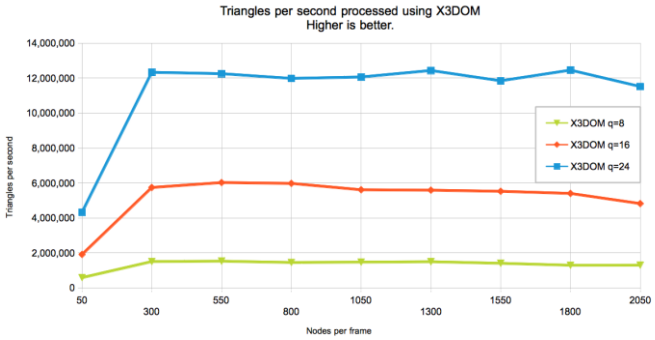
a) Frames per second for the different strategies

From the results, it was found that within the tested range an increase in triangle count, does not introduce a performance hit.

Indeed, from the TPS graph it is clear that the more triangles are given to the system, the more triangles are processed.



b) Batches per second



c) Triangles per second

Figure 5: Results of the tests

From the BPS graph, a bottleneck is found, limiting the number of batches per second and as consequence, the number of instructions that can be submitted per second. In this regard, the use of transformations and material changes require additional operations reflected as additional processing time. As consequence, it is possible to render a larger number of triangles per node without a significant performance hit, leading to the main strategy proposed in this paper: *the stitching method* (i.e. the Non-split method 2).

This strategy overcomes the bottleneck in number of batches per second of the system by reducing the number of nodes in the scene graph, grouping several geometries within the same node and avoiding transformations. This strategy can group as many geometries as possible within the limitations of the system in number of triangles, material changes and the object picking implementation.

4.3 Real datasets

To show the applicability of the abstract tests, for every reconstruction approach, a set of pipelines are reconstructed within an X3DOM scene graph, and their performance tested based on the number of frames per second (FPS). For each created scene, the complexity is also indicated by the quality of the objects, the number of nodes, the triangle count and encoding size.

The tested scene consists of 162 pipes and cables initially encoded as polylines, with an average length of 18.7 segments. Each pipe its appearance is set accordingly to the pipe category, requiring 11 different materials.

Table 1: Performance results using real datasets

Method	Quality	Node count	Triangle Count	Encoding Size (MB)	FPS
1	24	5 902	3 676 544	1.5	2.3
1	8	5 902	571 776	1.5	2.3
2	24	5 902	4 064 640	1.5	2.3
2	8	5 902	4 064 640	1.5	2.3
3	24	5 902	3 630 524	148	0.3
3	8	5 902	434 016	16.5	2.7
4	24	682	3 630 624	148	20
4	8	189	434 016	18.4	30
5	24	162	137 760	4.9	50
5	8	162	45 920	1.6	50

Results in Table 1 confirm that the same scene built using the Non-split strategies (Method 5), and in particular the stitching approach, increases the number of FPS, reduces the triangle and node count, and as a consequence the encoding size.

5. CONCLUSION

The different reconstruction approaches presented and their mapping into the corresponding scene graph showed a clear distinction between its produced complexity and the rendering performance. This difference showed that reconstruction methods with an efficient encoding not necessarily translate into an efficient rendering. It also shows that common techniques for improving the rendering performance not necessarily deliver the desired results in WebGL. In order to assemble the desired scenes and achieve real-time visualization using WebGL, the following conclusions should be considered when reconstructing underground objects:

The CPU is a critical factor. As visible from the results, applying some common optimizations like reducing the triangle count do not bring tangible benefits on the rendering speed. The reason behind it is another factor minimizing the advantages. In the studied case based on WebGL, the CPU is a bottleneck limiting the number of rendered nodes per frame and vanishing the benefits of other improvements. Therefore, improving the performance with significant margins can be achieved first by reducing the number of nodes in the resulting scene graph.

Aggregating nodes is not only a good solution but also a requirement. The reconstructed pipes are composed of multiple pieces sharing identical appearance, identifiers and administrative information, making them suitable for being aggregated or packed into a single node. Indeed, not every reconstruction approach allows packing geometries and reducing the number of nodes. Fortunately, the described family of 'non-split' methods allows such packing and in particular, the 'stitch' approach shines by reducing the triangle count while, discarding most of the spheres, reducing the number of faces per cylinder, removing the duplicated vertices on the shared boundaries, but also keeping the visual quality.

'Stitching' is an optimization over the other approaches for 3D pipes. The 'split' method' is considered an optimization compared to other approaches, reducing the size of the declared scene and increasing the performance at the rendering stage. In addition, the introduced optimizations are independent of specific scene graph implementations and rendering pipelines, being able to run on both programmable and fixed pipelines, guaranteeing its applicability on different platforms.

WebGL weakness is the performance of JavaScript. WebGL has proven to be an interesting technology to display 3D graphics

embedded in the browser. However, the inherent characteristics of reconstructed underground objects produce scenes which show one of the weaknesses of the technology: the speed at which the instructions can be submitted to the GPU. This weakness originates from the JavaScript language, which is an interpreted language producing additional overhead on the CPU and therefore reducing the processing power to submit work to the GPU. In comparison, X3D web plugins and standalone X3D players are usually developed using compiled languages and have a closer integration with the underlying hardware and graphic drivers. If X3D/X3DOM were implemented into the browser, it could produce twice to ten times more frames per second than its JavaScript counterpart. However, since this requires introducing an interface to interact with the scene (Scene Access Interface), the performance gain would remove the benefits of having direct access to the 3D scene via WebGL.

In this research the 3D shapes for the pipes needed to be reconstructed on the fly after the pipes are fetched from a DBMS, where they are managed as 3D lines. Therefore mesh optimisation approaches were not considered at all. The mesh optimisation will be very suitable if the 3DX files are stored on the server. Limper et al 2013 and Lavoue et al 2013 have shown that the performance can be significantly improved also in case of WebGL-based rendering when mesh optimisations are applied.

ACKNOWLEDGEMENTS

The authors express they gratitude to the '3DSDI' project funded by the programme Next Generation Infrastructures, Maasvlakte2, which made this research possible.

REFERENCES

Altmaier, A. and T.H. Kolbe, 2003 Applications and solutions for interoperable 3D geo-visualization, *Proceedings of the PhotogrammetricWeek.*

Apteker, R.T. J.A. Fisher, V.S. Kisimov, and H.Neishlos, 1995 Video acceptability and frame rate. MultiMedia, *IEEE*, 2(3):32–40

Basanow, J., P. Neis, S. Neubauer, A. Schilling, and A Zipf, 2008, Towards 3D Spatial Data Infrastructures (3D-SDI) based on open standards – experiences , results and future issues. *Advances in 3D Geoinformation Systems*, pages 65–86, 2008.

Behr, J., Y. Jung, T. Drevensek, and A. Aderhold, 2011 Dynamic and interactive aspects of X3DOM. *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 81–87

Bertin, J., 1983, *Semiology of graphics: diagrams, networks, maps.* Wisconsin PressWisconsin PressWisconsin Press.

Brutzman, D. and L. Daly. X3D: Extensible 3D Graphics for Web Authors. Morgan Kaufmann Series in Interactive 3D Technology. Elsevier/Morgan Kaufmann, 2007. ISBN 9780120885008. URL http://books. google.nl/books?id=yfkW80XUNqUC.

Cutting, J.E. and P.M. Vishton, 1995, Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth. *Perception of space and motion*, 5:69–117

Döner, F., R. Thompson, J. Stoter, C. Lemmen, H. Ploeger, P. van Oosterom, and S. Zlatanova, 2011, Solutions for 4D cadastre – with a case study on utility networks. *International Journal of Geographical Information Science*, 25(7):1173–1189, 2011.

Du, Y. and S. Zlatanova, 2006, An approach for 3D Visualization of Pipelines. in 3D Geo Information Systems, Lecture Notes in Geoinformation and Cartography, pages 501–517. Springer Berlin Heidelberg, 2006.

Environmental Systems Research Institute ESRI. Esri shapefile technical description, July 1998. URL http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

Fernando, E. 2004, *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education

Heinen, T., M. May, and B. Schmidt, 2005, 3D Visualisation in Spatial Data Infrastructures. *Smart Graphics Lecture Notes in Computer Science, 3638(3):222–229*

Kadaster, 2008, IMKL Informatie model voor Kabels en Leidingen (Information model for cables and pipes). *Technical Report 1.1*,Ministerie van Economishe Zaken,May 2008. (in Dutch)

Kokkevis. V., 2012, *GPU Accelerated Compositing in Chrome*, May 2012, org/developers/design-documents/gpu-accelerated-compositing-in-chrome.

KronosGroup.WebGL Specification Version 1.0, 2011. https://www.khronos.org/registry/webgl/specs/1.0/.

Kronos Group. OpenGL OpenGL ES - The Standard for Embedded Accelerated 3D Graphics, 2012. URL http://www.khronos.org/opengles/.

Lavoue, G., L. Chevalier and F. Dupont, 2013, Streaming compressed 3D data on the Web using JavaScript and WebGL, Web3D 2013, June 20 – 22, 2013, San Sebastian, Spain, 9p.

Limper, M., S. Wagner, C. Stein, Y. Jung and A. Stork, 2013, Fast delivery of 3D Web content: A case study, Web3D 2013, June 20 – 22, 2013, San Sebastian, Spain, ACM proceedings, pp. 11-18.

Meier, J.D., C, Farre, P. Bansode, S. Barber, and D. Rea, 2007 *Performance Testing Guidance for Web Applications, patterns practices*. Microsoft Corporation.

Open GIS Consortium Inc. OpenGIS Simple Features Specification for SQL, Revision 1. 1. *OpenGIS Project Document*, pages 99–049, 1999.

OpenGL.org. 2.1 OpenGL fundamentals. URL http://www.opengl.org/documentation/specs/version1.1/glspec1.1/node10.html.

OpenGL Programming/OpenGL ES Overview, January 2011. URL http://en.wikibooks.org/wiki/OpenGL_Programming/OpenGL_ES_Overview.

Ramsey, P., 2005, PostGIS 2.0 manual. Refractions Research Inc, URL http://postgis.net/stuff/postgis-2.0.pdf

Reiners, D., 2002 Scene graph rendering, March 2002. URL http://cfile25.uf.tistory.com/attach/112B9D184C51392902C354.

Schall, G., S. Junghanns, and D. Schmalstieg, 2008, The transcoding pipeline: Automatic generation of 3D models, from geospatial data sources. *Proceedings of the 1st InternationalWorkshop on Trends in Pervasive and Ubiquitous Geotechnology and Geoinformation* (TIPUGG 2008), volume 23, 2008.

Schall, G., D. Schmalstieg, and S. Junghanns, 2010 VIDENTE-3D Visualization of Underground Infrastructure using Handheld Augmented Reality, *Geohydroinformatics-Integrating GIS and Water Engineering* CRC Press/Taylor and Francis Publisher: CRC, 1, 2010.

Shilling, A. and T. H.. Kolbe, 2010, Draft for Candidate OpenGISWeb 3D Service Interface Standard, 2010. URL http://portal.opengeospatial.org/files/?artifact_id=36390.

Strauss, P.S. and R. Carey, 1992, An object-oriented 3d graphics toolkit. *SIGGRAPH Comput. Graph.*, 26(2): 341–349

Verbree, E., S. Zlatanova, and K. Smit, 2004, Interactive navigation services through value-added CycloMedia panoramic images. *Proceedings of the 6th international conference on Electronic commerce*, ICEC '04, pages 591–595.

Wei, L., 2005, A crash course on programmable graphics hardware. Paper, Microsoft Research Asia, 2005. URL http://graphics.stanford.edu/~liyiwei/courses/GPU/paper/paper.pdf.

Wolka, M., 2003, Batch, batch, batch: What does it really mean?, 2003. URL http://developer.nvidia.com/docs/IO/8230/BatchBatchBatch.pdf.

Zlatanova, S., F. Doner, and P. van Oosterom, 2011 Management and visualization of utility networks for local authorities: a 3D approach. *Electronic Government and Electronic Participation*, Schriftenreihe Informatik 37, pages 459–474.

Zlatanova, S. Y. Du and X. Liu Management and 3D visualization of pipeline networks using DBMS and AEC software. *Proceedings of the ISPRS Commission IV Symposium on Geospatial Databases for Sustainable Development.* 34:395–400, 2006.