Geospatial Visualization of Large Infrastructure Projects

Klaus Wickenhauser[1], Gerd Hesina[1], Christoph Traxler[1], Michael Greiner[2]

[1] VRVis Research Center; Donau-City-Strasse 11, 1220 Vienna, Austria; Corresponding author: hesina@vrvis.at

[2] Geoconsult Wien ZT GmbH, Hütteldorfer Straße 85, 1150 Vienna, Austria

1 ABSTRACT

For most large infrastructure projects, it is mandatory to assess their impact on the urban and rural environment before they are started. Many shareholders want to involve the public or even are obliged to do so due to legal provisions. For interactive exploration, a geospatial visualization framework is needed that supports very complex scenes. They should be realistically rendered for sufficient credibility. Applied research on this topic (in close cooperation with industrial partners) resulted in GEARViewer, a geospatial rendering framework. It supports huge geospatial scenes consisting of large-scale terrain models, buildings, roads, tramways, railways, tunnels, vegetation and a skylight model. Everything is geo-referenced. An animation system augments the scene with moving vehicles and pedestrians that provides a simplified traffic simulation. This enables a realistic and vivid experience of planned infrastructure projects and allows assessing their effects on the real environment. GEARViewer supports OGC standards such WMTS to efficiently communicate with GIS. For example, basic 3D geometry can be procedurally generated from 2D GIS data. On the other hand, versatile outputs can be created mainly for presentation purposed such as high-res images, videos, even in 360° panorama formats. Furthermore, parts of the scene can be cut out and exported to be used in other 3D tools and viewers, supporting interoperability and a more flexible workflow. It was used for many planned projects in Austria and Germany.

2 INTRODUCTION

In the planning and public discussion of large infrastructure projects, many decisions need to be made that depend on a good understanding of how the project will fit into its environment. Traditionally, this has been done with 2D plans, architectural drawings or pre-rendered still and motion pictures. However, especially in the case of the general public, a more versatile, interactive approach may be helpful to provide a better understanding of the project. This allows people from different fields and potentially without knowledge about reading plans and diagrams to have a good grasp of the project. The aspects can be communicated better and ideas can be visualized and incorporated at relatively good efficiency and low time cost. From experience this can avert unnecessary misunderstandings and fears and provides a common ground for positive discussions amongst all involved stakeholders.

The essential technology for interactive visualizations and planning has been demonstrated by numerous games, content creation and other tools. They all have achieved high quality and extensive feature sets in recent years. Nonetheless, this is coupled with very specific restrictions or high costs in their production. Games have vast environments, displayed with formidable detail and realism. However, games are typically high budget productions with long development cycles, and the rendering system and 3D models are often finely tuned for the best possible quality and performance. After the development of their models and scenes the content is locked and changes can be very costly, in terms of resources and time. On the other hand, visualization tools in medicine and engineering, allow rapid prototyping and changing. This, however, often comes at a steep cost in visual quality or a restriction to very specialized aspects and a small number of objects or limited area. It should certainly be possible to build a geospatial viewer application with similar capabilities. This development style is not quite suitable for geo-visualization projects, where changes need to be incorporated on short notice, while allowing a huge number of objects and/or very expansive project areas, and manpower is typically much more limited. Furthermore, data is often created in GIS or architectural modeling applications, and with different goals than real-time rendering. This typically means that the available models may not be directly suitable – they may be much too detailed, or only available as outlines, missing surface descriptions. Often, they are also not optimized at all for the purpose of real time rendering.

Therefore, preparing models for interactive visualization requires a different workflow and other tools than typical 3D game engineering. This also applies to the functionality available in the viewing application itself, which servers a different purpose than most game playing and content creation environments. From experience, especially in planning discussions, the main benefit does not lie solely in a photorealistic display. Much more essential is the capability of being able to explain complex interrelations, enrich common 2D planning data by embedding it in a spatial 3D context (possibly together with simulation input), and being able to answer complex questions promptly with a suitable toolset, which sometimes requires a certain degree of abstraction.

The combination of these features leads to a system that already provides some basic functionality of a GIS, such as multiple layers in 3D and display of additional information. In this regard GEARViewer can be conceived as a first approach towards a 3D GIS [1]. Furthermore, it also supports more complex workflows in project planning and realization. It takes different kinds of input from different sources, processes them and outputs it either visually or in other forms. GEARViewer also allows to associate 3D objects with additional data such as links to various documents (see Figure 2). This is a first step in supporting BIM (Building Information Modeling) support as it links 3D object with meta-data important for assessments, planning and decision making.

In the following text, we will present GEARViewer, an interactive geospatial viewer developed in recent years at VRVis in collaboration with large Austrian infrastructure providers and civil engineering consultants. After an overview of related work, we will discuss the basic architecture functionality of our system and provide detail on how various design decisions were made. Finally, we will present the workflow supported by our system, and show how it was used in two practical applications.

3 SYSTEM OVERVIEW

In a typical application, the purpose of the system is to present a given infrastructure project in different variations, iterations and stages. The entire scene can on one hand be explored interactively, by means of a free fly camera, or with certain restrictions, like walking or driving along specified paths. On the other hand, camera paths can be defined for hands-off presentation purposes. In addition, it is possible to switch between project alternatives, adjust daylight and illumination parameters, take measurements to provide exact visibility and distance information, and utilize a number of other features.

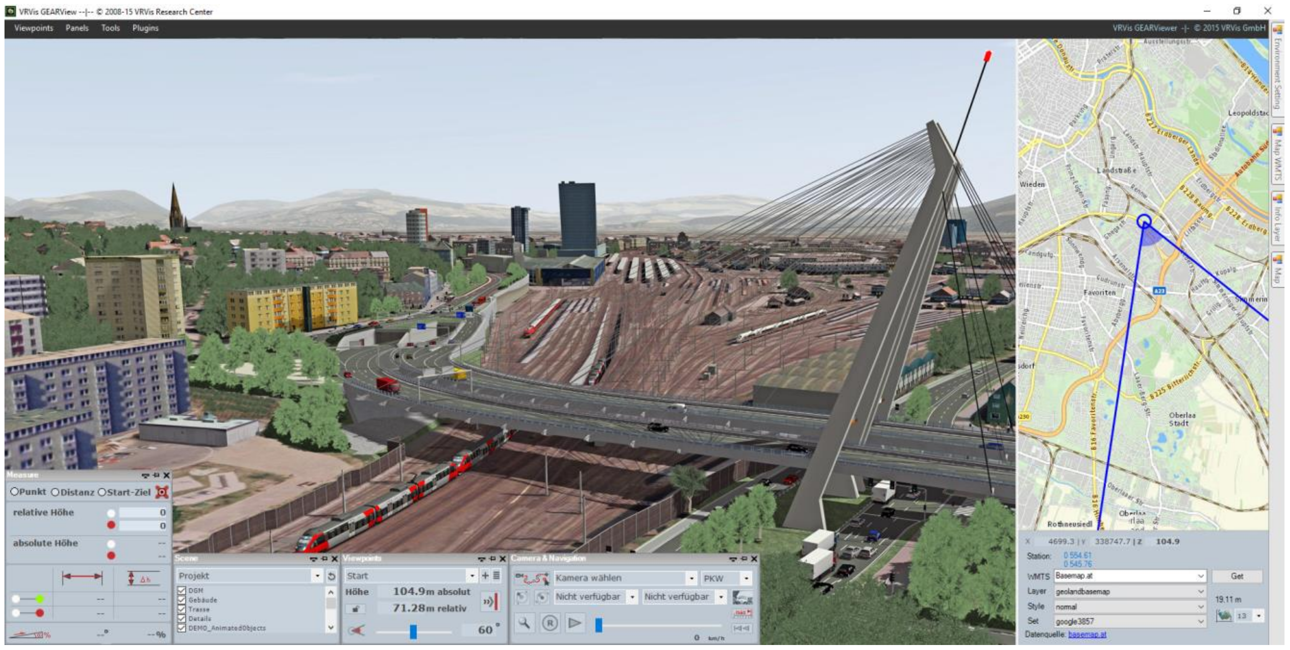

Figure 1: GEARViewer main render window with various GUI elements.

Support for switching between alternatives is provided by allowing multiple scenarios (which may be different design options, present and planned system, etc.) for the presented scene. Each scenario is made up of a number of layers that can be individually shown or hidden at runtime. In turn, each layer consists of at least one original model; individual models are combined and packed at load time for efficient rendering. Therefore, switching between layers or scenarios is very fast once data is retained in memory.

The interrelation between models, layers and scenarios is stored in an XML based scene description that is loaded at startup. This scene description also controls which parts of the scene are used for testing against collisions and which texture and shader attributes to use for each model, defines the animation networks used for dynamic content in the scene, and provides other details. This system is supported by offloading different configuration aspects to their respective XML files (e.g. Viewpoints, Controller input, etc.) definitions. This allows a plug and play behavior for different scene setups, without the need to manipulate the scene definition itself.

To support loading a variety of models, a flexible import mechanism is provided. The supported sources consist of some of the mainstream geometry file formats (VRML, OBJ) and others that are more specific to GIS workflows. This is an important aspect because real-time rendering systems normally require tedious detours to be able to handle this type of data. To this end, the GEARViewer supports the following geospatial data formats:

Geo-referenced imagery	In principle this is a normal image file, which is accompanied by a world file which describes its positioning and orientation. They can either be mapped to geometry as textures or used as a 2D Map data source[1].
ESRI shapefiles & FileGDB	These are ESRI GIS specific file formats used to export/import point, line and geometry data with various additional attributes. Their uses in the system include the positioning of objects (trees, lampposts …) and the definition of line geometry (power lines, annotation geometry …). The system supports the manipulation of read attributes through arbitrary C# expressions that are compiled on the fly (control extrusion, scaling …).
Geometry data	Used to define model data. The formats mostly used in the system are VRML (Virtual Reality Modeling Language, .wrl) and Wavefront OBJ (.obj). Many tools used in conjunction with infrastructure planning can export/import their data in these formats.
ArcGrid	A file format suitable for creating large terrain models. The raster data is used to create the actual geometry with a level of detail hierarchy. Missing data can either be ignored, resulting in holes, or be mapped to and filled with a predefined value.

Due to the modular architecture of our framework, integrating further data formats in the future is straightforward, and does not impede any existing workflow.

All data is saved in an optimized, binary cache file format. These cache files can be loaded directly and used instead of the original source data. Therefore, a closed data package can be supplied to third parties instead of having to give away the original source data. Cache files are also interchangeable between scenarios and visualization projects, which provides advantages when complex imports require significant processing time.

Various overrides are possible at load time without the need for re-importing source data. This includes offsets, scale and rotation transforms, and assigning shader programs and other rendering properties. Doing this externally to the original model and cache files allows the use of a base model that doesn't need to be touched for fine tuning or model reuse, and also works for file formats that do not support these properties.

The GEARViewer UI (Figure 1) consists mainly of a full screen sized main render window showing the real-time scene as large as possible. Navigation is supported using a mouse, keyboard, game pad or space mouse. The small GUI elements displayed at the bottom of the screen contain the most often used settings and are grouped thematically into i.e. viewpoint controls, camera paths, scenario handling, output or 2D map and can be minimized to title-bar size or closed completely if unneeded. Hidden in a roll-over sidebar are additional expert settings to gain access over the whole rich feature and parameter settings such as lighting and advanced HDR, animation or clipping plane control just to name a few. The arrangement and availability of

[1] https://www.loc.gov/preservation/digital/formats/fdd/fdd000287.shtml

https://en.wikipedia.org/wiki/World_file

GUI elements is configurable according to respective requirements and some elements appear automatically when triggering certain features such as the measure and visibility toolset or video render-output. Optionally, a full-screen stereo rendering mode (without GUI elements) is also available.

3.1 System Features

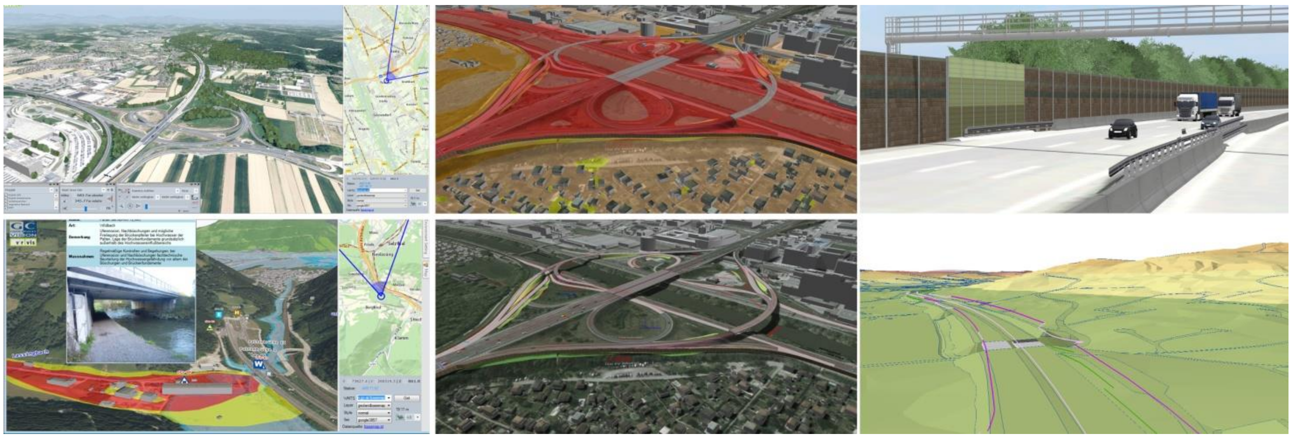
Figure 2: Feature overview – GUI, Hotlinks, Projected Textures, Actors and sketch design

3.1.1 Time System

'Real time' is a difficult concept for presenting large geospatial projects. For example, it may be desirable to present the looks of a new building over the course of a day compressed into a few minutes, or to adjust the animation time system to quickly populate animation networks on startup or to handle non-real-time output properly. Therefore, our system includes three different time systems: *animation time* is used to determine vehicle speed; *environment time* controls lighting effects such as sun position and shadows, and *output time* is typically real-time for interaction, but for rendering video output is set to a fixed rate per rendered frame. Animation and environmental time can be either stopped (frozen) or coupled to output time using varying factors. Therefore, vehicle animation may for example be slowed down to be less distracting, and environmental time may be simultaneously sped up by a large factor to present a full day/night cycle in a short period of time. Additionally, environment time can be freely set to arbitrary values to quickly view the scene at a specified date or time.

3.1.2 Actor Animation System

In many cases, enriching the presented environment with vehicle traffic or other dynamic content gives the end user a better impression of the effects of the planned project. This gives a more lifelike appearance and helps people from different fields or residents to imagine the scene in a more real and less detached way (see Figure 3).

Two variants of such animation systems are implemented within the system and can be integrated into scenes in a similar way as models are.

The first variant is geared to a partial simulation of traffic. This can on one hand either be utilized for more constrained vehicles that always need to remain on rails, like trains. On the other hand, more autonomous vehicles might be necessary, like cars, trucks and buses that sometimes need to change lanes. The basic structure boils down to: each animation network is defined by polylines in a shapefile and attributes controlling aspects like direction, maximum speed, and information concerning multiple logical connections of roads, density and other details. Each network also defines multiple source and drain nodes. To release a car a random source is chosen and a random path to a drain node is found according to the targeted traffic density. Once an actor reaches a drain node, it is removed from the system.

Vehicle position updates are constantly calculated in a separate thread, and are therefore independent of the current render frame rate. For strictly path bound vehicles, all path segments from source to drain node are pre-calculated and merged into a single path upon vehicle creation. This merged path is then used to animate the vehicle at constant speed. After each position update, front and rear axes of each vehicle are explicitly constrained to remain on the path; in the case of vehicle chains the training sub-vehicles are also aligned on the path behind the main vehicle. On the other hand, autonomous vehicles only receive their target endpoint, and are designed to automatically find a suitable route. They can exploit logical connections between paths forming multiple lanes of the same road, and therefore change lanes for more realistic behavior. These vehicles are also aware of other traffic and adjust their speed accordingly to avoid collisions. This increased realism also comes at a higher computational cost, however in our current prototype the main bottleneck is still displaying larger numbers of vehicles, and not the calculations required for updating their positions.

The other variant is a prebaked animation system. Virtually they are the same in regard to visualizing moving traffic or similar moving actors. Under the hood the prebaked animation system is a key frame animation system that trades the partially autonomous aspects with more efficiency and control. Originally the system was developed to take the output of a pedestrian simulation from our partner AIT and later got improved to include the output of the popular VISSIM simulation software [5]. In principle, the system could be further extended to take other input, as long as it is or can be converted to a key framed simulation input. The original idea was to use a GIS like approach to plot the animation paths of objects, which has the problem of only being available in 2D. This translates poorly to 3D environments because part of the necessary realism are actors adhering to their environment. The GEARViewer provides a mapping algorithm that matches the objects to the top-most geometry encountered at the position of a given key frame. This, however, leads to the problem of actors jumping on top of bridges or irrelevant geometry. To prevent this, intersection points are constrained to certain heights above actors. This allows them to go under bridges and paths that intersect themselves to have different heights, depending on the path they take (for example going first under a bridge and then over the same bridge). The system also features a manual re-cache algorithm to allow for changing of layers and geometry. The re-cache can be saved and used instead of the original data. The mapping algorithm uses the last position to select viable intersection heights and therefore has a problem with the starting position. To allow the starting position to be inside of tunnels or under bridges the system uses different ways of constraint definitions. Starting heights can be defined on a per ID basis or force all starting points in areas to have a specific height.

Vehicles are rendered using hardware instancing: the geometry and texture data of each vehicle model only exists once, and is repeated on the GPU hardware as often as required using appropriate transformations for each vehicle position. To improve diversity, colors can be adjusted for each instance through a custom pixel shader. A simple modulation of the present (texture) colors typically does not produce the desired result, so we use a separate texture color channel to modulate between the original texture and the per-instance color. In our system, this information is stored in an illumination map that accompanies most models to include self-luminance information for nighttime scenes. However it would be equally possible to use, for example, the alpha channel of an RGBA texture.

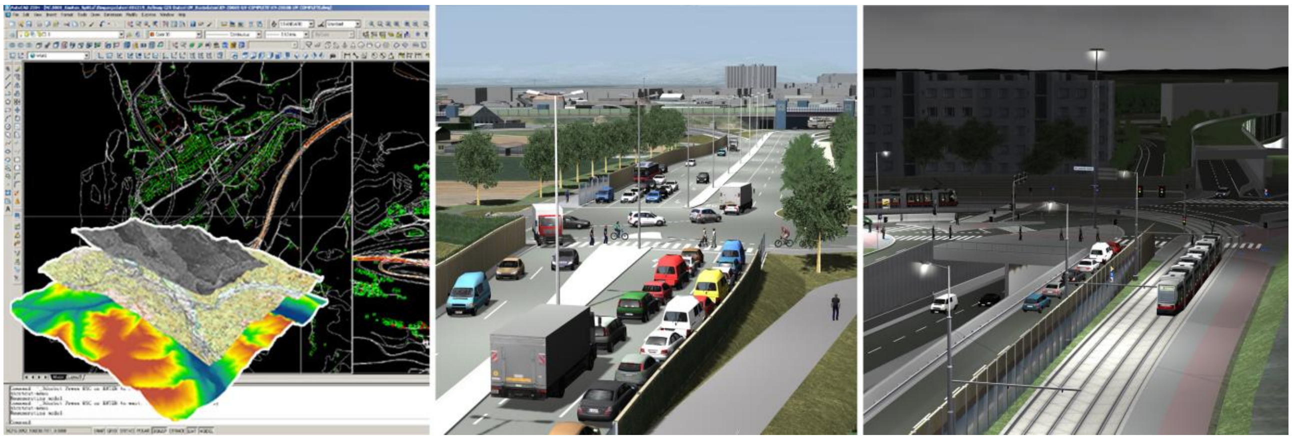
Figure 3: GIS data source, Actor system and Environment

3.1.3 Environment

Our system supports a geospatial reference location to accurately depict the sun position and lighting on a given longitude and latitude. Currently the sky is rendered using a Preetham day sky model [4] and optional cloud layers. In addition, the sun position is used for calculating dynamic illumination and shadow mapping of the scene as well as day/night changes. Many parameters of the system are user tunable, such as shadow map extents, resolution and biasing and illumination intensity and shadow contrast. Models can be augmented with an illumination, normal and specular map. They supply additional information about light emission at night, geometry information for lighting and the strength of reflections, respectively. The sources for the reflections are pre-created textures, supplied at startup time. It is possible to supply a global reflection map or specify them on a per layer basis. Furthermore the intensity of these values can be optimized globally or on a per layer basis. This enables the user to fine tune the environment for specific visualization purposes, and store and recall all parameters to quickly switch between different setups.

3.1.4 Projected Textures

An important feature of GIS tools is the possibility to display additional terrain information, by overlaying the project area with maps. This can be used to toggle the display of planning data, in the form of sketches or city maps. Furthermore a multitude of additional information can be displayed. Examples for this are heat maps, danger zones, bodies of water, forests and many more. This is easy in a 2D environment, but is a challenge in 3D when the covered areas have a huge extent. We allow the display of multiple layers on the terrain by injecting and projecting them on the geometry at startup time. The maps are stored separately from the model caches; this allows the exchange of different data layers without the need to manipulate the supplied geometry data (see Figure 2). The UI provides the tools to change the order of the displayed layers, the decision which layers to display and hide, and the transparency of the layers. There are two ways to supply these layers. First, a collection of geo-referenced tiles can be returned. They are re-projected and re-tiled to fit on the specific terrain geometry. Second, the data can be defined by shapefiles containing closed polygons and meta-information, like color. This polygonal data is used to create the tile maps for the layers.

3.1.5 Timeline

The GEARViewer also includes a timeline to consider time as an additional dimension. This is a very important dimension for almost all infrastructure projects as it enables to monitor the different stages of the realization process and to control its visualization. It makes workflows, dependencies and delays better comprehensible, which is crucial for decision making. The component is currently a prototype and only allows the definition of snapshots. These snapshots contain the scene and layer setup of specific time frames. This allows the display of project processes over time. These snapshots can be exported and are imported at the next start of the system.

The idea of a timeline allows the incorporation of many other aspects to be displayed over time. It can and will therefore be expanded in the future.

3.1.6 3D and 2D Linked Views

As mentioned the GEARViewer also includes a 2D Map. This allows different ways to correspond 3D data with 2D data. The 3D and 2D Views are geospatially linked and moving the viewpoint in one view moves the viewpoint in the other. Moving in the 3D view is done by the mentioned exploration methods, while movement in the 2D view is done by clicking on the target location or typing the target coordinates. The 2D map shows the current coordinates (in the coordinate-system of the scene) and allows different zoom levels, which can be changed by either using the mouse-wheel or directly selecting the target zoom level.
There are two kinds of 2D Views that are virtually the same but have different data sources. The first and legacy 2D View attains its data by providing a geo-referenced map or collections of tiles and is simply preprocessed and displayed in the map window.

The new 2D View is a WMTS map view. It gets data from specified WMTS services. The advantage with this approach is to provide an interface to OGC compliant data sources. While the interface is currently exclusively tested with a handful of Austrian and German WMTS providers, it can be further extended.

In addition some 3D Viewer parts can also be displayed on the 2D map to help link important features. Shapefiles showing planned power lines, for example, can also be displayed as lines overlaying the map.

3.1.7 Real-time object creation

Infrastructure projects are often large, very complex and the corresponding data creation can take a lot of resources. Furthermore, it can be very important to react to the feedback and inquiries of decision makers or residents, as the presentation is going on. The GEARViewer currently meets this requirement by an experimental feature. It is possible to place or create some objects during the visualization of a scene without the need to change the XML definition and restart the system. Models that can be imported via the scene.xml can also be placed by dragging them onto the viewer. Rudimentary placement tools allow the correct positioning in the scene.

Furthermore, it is possible to procedurally create linear geometry, like noise protection walls, by defining a line and supplying the profile information of the geometry to be created. The geometry is expanded from the profile along the defined axis. The profile information, as many other aspects, is defined via shapefiles and can therefore be easily integrated into the GIS workflow.

An example for this is the building of a new freight depot close to an inhabited area, where residents complained about the looks and noise pollution. To incorporate this feedback an earth wall with trees and noise protection walls was placed and displayed, with relatively short iteration periods and the worries of the residents were appeased. This would have been far more tedious by using just maps or static visualizations.

3.1.8 Output

One important aspect of the GEARViewer is not just to help plan and visualize the project at specific meetings but also to provide output for further applications. This can be simple presentation materials but also third-party tools that allow a multitude of different operations.

To support this, the system allows the creation of screenshots, posters and videos from within the tool, which gives more control over what to include and what to show, than using external tools. Furthermore, different subsystems, like the animation system or camera animations can be synched with the output rendering to allow a more efficient workflow. In addition, this output can also be created in form of 360° panoramas. Examples for this are websites that provide a map that allows highlights to be linked with panoramic pictures, or panoramic videos of the project area. This kind of media can also be experienced with VR systems or smart phones. It gives a more tactile feeling of the highlighted areas and scenes to interested parties.

Another output aspect is that parts of the scene can be cut out and exported as individual or concatenated .obj models. This allows the geometry to be used with other tools, light-weight applications or where the GEARViewer or remote rendering is not applicable.

An addition to path based animation systems is the possibility to record the simulation as a key frame animation for later use, where the full interactive visualization is not needed. This follows the VISSIM specification of the prebaked animation system and therefore can be used as a less expensive but more rigid input into the GEARViewer itself and as input to other tools.

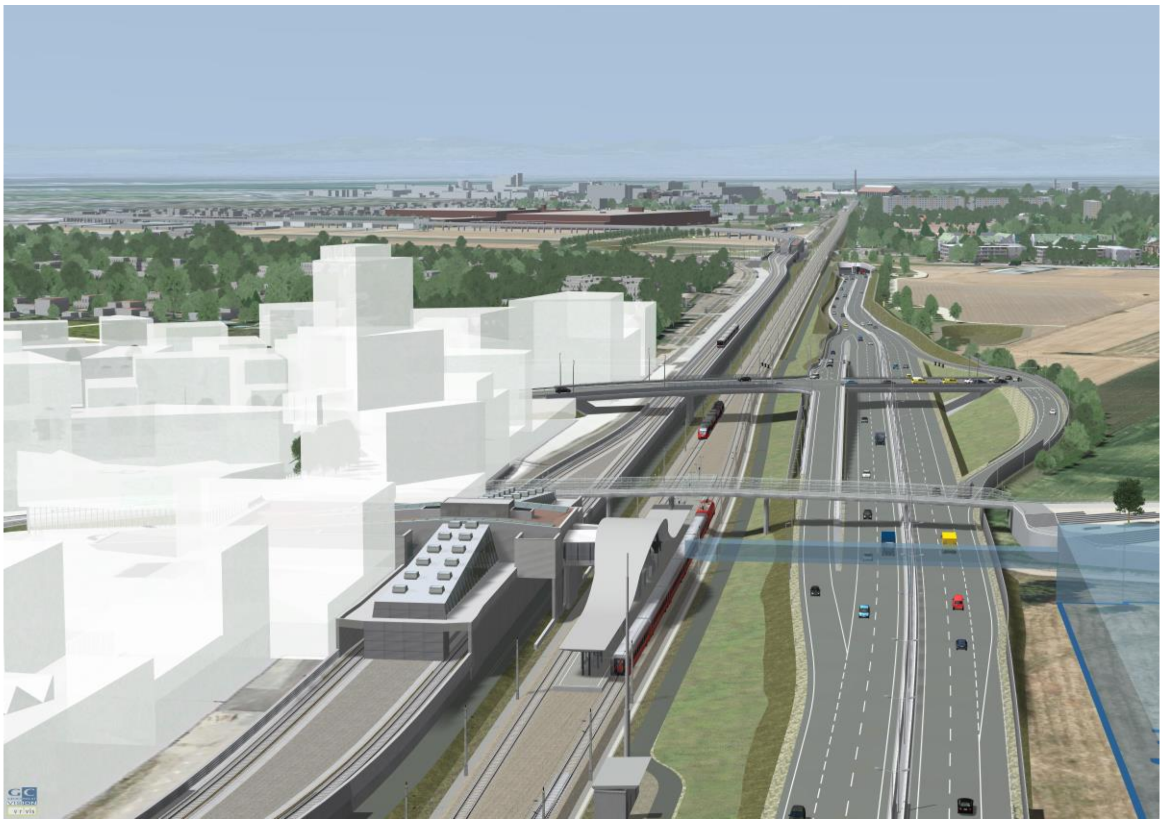

Figure 4: Project showing transparency used as a planning tool

3.2 Use Case: City development area Vienna - Aspern

While GEARViewer is constantly successfully used in multiple small to large infrastructure planning projects mostly across Austria, one particular use case was especially outstanding and complex. In the north-eastern part of the City of Vienna, a large area (previously used as an airstrip) has been designated for city development where a whole quarter named "Seestadt Aspern" with mixed residential and commercial areas is to be erected in the coming years in tie with the supporting road and public transport infrastructure. Therefore the major infrastructure carriers (Asfinag, ÖBB, Wienerlinien) and the City of Vienna with it's various municipal authorities coordinate their individual planning tasks and GEARViewer was used in service of each individual project but also combining the separate visualization models into one comprehensive view of the whole development area. The following major planning projects had to be conducted in the process (see Figure 5: Aspern project):

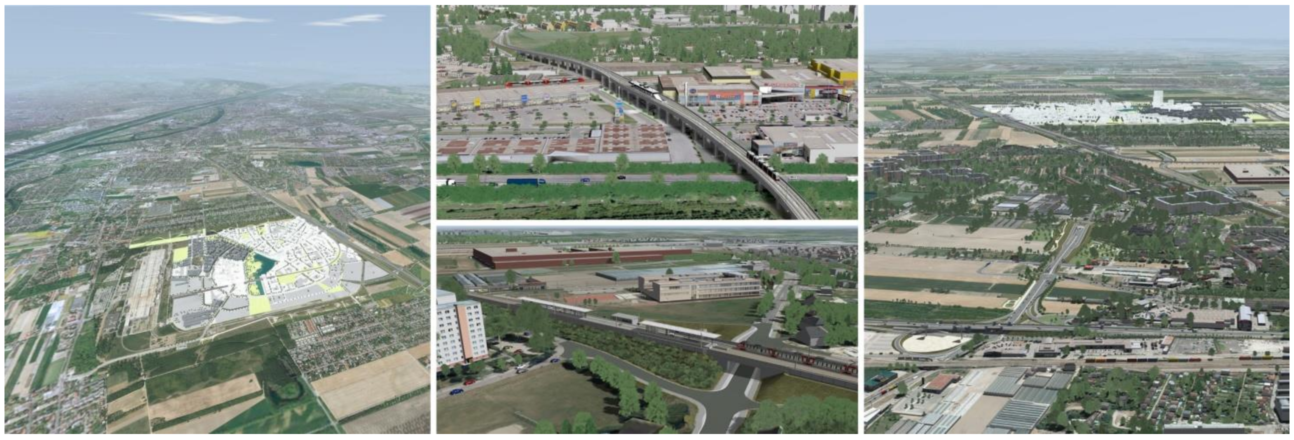

Figure 5: Aspern project

ÖBB is upgrading an existing railway to a modern electrified double tracked rail network including two new stations as part of the connection towards Bratislava. Wienerlinien have built a new metro line as public transport backbone of the new city quarter, consisiting of several stations plus new tramway lines and stops for an efficient network. The City of Vienna is developing the higher-ranking and secondary road infrastructure for the area in attunement with Asfinag who is providing the link to the recently developed motorway beltway around Vienna. All these infrastructure and traffic solutions support the ongoing architectural development and construction of the Seestadt area, which is growing according to a masterplan step by step on the designated building plots.

Each project itself is already a major effort as various planning stages need to be visualized and updated continuously in highest detail in order to inform citizens and stake holders of ongoing developments. The main challenge in addition to the already large projects was the coordination and synchronisation of the different planning levels. While some projects were already in the final stage or already under construction, others were at the beginning as concepts and in the middle of the planning and optimisation process and all of them were intertwined with more or less impact on each other and in constant adjumstment.

Based on city survey data, a very detailed basic environment model of the whole city area was modeled and used continously as base for inserting and updating all the different infrastructure projects at their different stages. For each stakeholder, a comprehensive project presentation was provided at all times, using the other projects as background context, retaining the bigger picture of each project.

GEARViewer with its modular design, the powerful render framework for huge datasets, the optimised interfaces and workflows and the large palette of features proved to be an efficient and reliable tool to provide all necessary visual support and simulations throughout the whole planning process for the involved stakeholders. It was capable of seamlessly integrating and maintaining ongoing project changes and delivering the various needed media in order to communicate the ongoing planning phases at all times.

4 CONCLUSION AND FUTURE WORK

The goal of GEARViewer and its preceding developments has been to create a versatile framework for displaying very large geospatial data in a real-time environment, with suitable handling and UI features for presentation purposes. The presented feature set supports a streamlined content creation workflow that natively supports geospatial file formats for optimal performance and customizability.

Future work will include the update of the underlying engine to a modern graphics API to improve existing features and employ modern techniques that would otherwise be impossible or increasingly impractical to implement. Furthermore, the goal will be to improve the GEARViewer's status as a 3D GIS and its interoperability as an element in a larger workflow. This includes the improvement and addition of more editor-like features which allow modifying the scene during run-time. This increases the flexibility and reduces preprocessing effort. New objects can be more tightly entwined with the scene currently explored interactively. The geographical extent of projects and the need for more details is ever increasing and therefore the employment of more and better out-of-core techniques is an avenue worth considering.

GEARViewer will fully support Building Information Modelling (BIM) [2] in the future. BIM compliancy is very important for project planning and decision support. GEARViewer already provides the basic functionality to link 3D objects with additional meta-data. Links to websites and documents can be associated with scene objects and opened in the browser. This meta-data will be BIM conform and provide the full semantic of objects in future version. Another key aspect of BIM is efficient collaboration, which is enabled by a shared repository over which data in standardized BIM formats is exchanged and all members of a project team are notified about modifications. We will make the GEARViewer BIM compliant by building an interface to exchange data with BIM repositories and that supports BIM standards such as IFC [3].

REFERENCES

[1] Brooks S. and Whalley J.L. Multilayer hybrid visualizations to support 3D GIS. In: Computers, Environment and Urban Systems vol. 32(4), pp. 278 - 292, 2008.

[2] Eastlake, Chuck; Tiecholz, Paul; Sacks, Rafael; Liston, Kathleen. BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors (2nd ed.). Hoboken, New Jersey: John Wiley. pp. 36–37, 2011.

[3] ISO - International Organization for Standardization. IFC Standard web site: https://www.iso.org/standard/51622.html. Last accessed 2017.08.31.

[4] Preetham, A.J., Shirley, Peter, Smits, Brian. A Practical Analytical Model for Daylight. Published in Proceedings of SIGGRAPH '99, pp. 91-100, New York (USA), 1999.

[5] PTV Group. VISSIM web site: http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/. Last accessed 2017.08.31.