Visdom Mobile - Decision Support On-site Using Visual Simulation Control

Bernhard Sadransky*
VRVIS Research Center

Hrvoje Ribicic†
VRVIS Research Center

Robert Carnecky‡
Swiss Federal Institute of Technology

Jürgen Waser§
VRVIS Research Center

Figure 1: The mobile interface of the Visdom system

Abstract

The power of fast flooding simulations has not yet been used to enhance on-site decision making. In this paper, we present the case study of a mobile user interface which provides remote access to a simulation-powered decision support system. The proposed interface allows users to explore multiple alternative flooding scenarios directly on-site. Different views are used for this purpose. Scenario and time navigation is done through a temporal view, while a spatial view is used to navigate through space via a 3D rendering of a scenario. Using the touch-sensitive mobile device, the user can create alternative scenarios by sketching changes directly onto the rendering. The mobile interface acts as a thin client in a distributed environment where the server performs simulation and rendering. The approach was presented to a group of domain experts in the field of hydrology, who consider it to be a useful step forward. Further tests were done with a simulation engineer who considers the interface to be intuitive and useful.

CR Categories: I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics

Keywords: visualization, remote rendering, simulation steering

*e-mail: sadransky@vrvis.at
†e-mail: ribicic@vrvis.at
‡e-mail: crobi@inf.ethz.ch
§e-mail: waser@vrvis.at

1 Introduction

In the field of flood response-planning, decisions about evacuations and flood defenses are based on flood-risk maps. These maps describe the effects of a disaster, and are generated in advance by simulating known and expected hazards. It is also important for the on-site decision makers to have information about the consequences of a failure of already placed flood defenses, especially when this might happen in the current situation. With such information at hand, they can evaluate whether to use a second line of defence.

Present flood-management software is primarily used for communication and coordination. Advances in hardware and techniques have resulted in flooding simulations which can deliver predictions quickly enough for real-time use [Sætra and Brodtkorb 2012]. These allow an expert to simulate multiple scenarios, each modeling a different development of an ongoing situation. However, to effectively use and communicate such information, visualization and interaction are required.

Our previous work has focused on developing visualization and input techniques that allow the user to harness the power of the simulation, while being accessible enough that an expert can use them with only limited training. Sketching can be used to alter the parameters of the simulation in an intuitive way [Ribičić et al. 2012], and World Lines show the temporal evolution of scenarios and allow the introduction of alternatives [Waser et al. 2010]. Although

SCCG 2013, Smolenice, Slovakia, May 1 – 3, 2013.
© 2013 ACM 978-1-4503-2480-9/13/0005 $15.00

successful at their aims, all of these techniques were developed for a desktop client. As on-site use is important for reacting to dangerous situations, the design of a mobile client became a necessity.

In this work, we present a mobile user interface for the remote control of a simulation engine, based on the aforementioned techniques. The interface allows the user to set up different scenarios by changing the various parameters of the simulation. The user should be able to easily navigate across these scenarios and visualize their respective outcomes. To fulfill these requirements, the functionality of the desktop tool was distilled into two views adapted to touch-based interaction and a smaller screen area. The spatial view supplies tools for creating and visualizing alternative scenarios, whereas the temporal view provides the means for navigating through them.

Using these two views on a low-performance mobile device was also a challenge from an implementation point of view. We discuss the properties of the Visdom (`http://visdom.at`) framework that simplified the transition, but also describe the changes necessary to shift the computational burden from the mobile client to the server, and make the system performant.

To verify that the end result can be used in practice, we performed an evaluation by presenting the device to a group of flood-management experts. They provided oral and written feedback, answering a questionnaire designed to assess how viable they believe this approach to be.

In summary, the contributions of this paper are:

- A mobile approach for a decision-support system based on the simulation and visualization of alternative scenarios.
- Touch-based navigation of scenarios in both the spatial and the temporal domain.
- A description of the solution can be integrated into expert workflows, and an evaluation of the system with domain experts.

2 Related Work

The mobile interface presented in this paper overlaps several areas of application. The related work is presented for each of these.

Disaster Management on Mobile Devices Several systems designed to support decision makers on-site have been developed. L. Paolino et. al [Paolino et al. 2010] support evacuation operations by integrating spatial and temporal information on a mobile client. S. Kim et. al [Kim et al. 2007] introduce a mobile visual analytics system that processes sensor, location and video data for first responders. B. Betts et. al [Betts et al. 2005] describe a system that allows for report and image exchange and archiving, as well as real-time location tracking of first responders. K. Luyten et. al [Luyten et al. 2006] have developed a system to support a fire brigade on-site with a set of mobile services that offers a role-based user interface. A web-based GIS for visualization and analysis of threats for marine and coastal regions is presented by A. Stepnowski et. al [Stepnowski et al. 2010].

Flood Management FLOREON [Vondrák et al. 2008] is a web-based flood-management system, whose goal is to provide users with information about an approaching flood disaster. Given a date and an area, FLOREON computes and visualizes the flood-prediction results. Z. Zheng et. al [Zheng et al. 2011] have implemented an integrated mobile system for prediction and visual display of marine disasters. Their iPhone client provides maps overlaid with data such as water-flow velocity and direction. M. Jern et. al [Jern et al. 2003] investigate visual user interfaces for PDAs, one of the presented implementations is a flood-forecasting system. FLIWAS [de Gooijer and Reuter 2009] is a web-based flood information and warning system. It is primarily used for coordination and communication, but also gives access to precalculated flooding scenarios and flood maps.

Remote Visualization E. Mancini et. al [Mancini et al. 2012] explore the remote management of a simulation tool using an Android smartphone. Their mobile client shows the evolution of a forest-fire spread. I. Holmes and R. Kalawsky [Holmes and Kawalsky 2006] developed mobile user interfaces to access scientific applications running on the grid. The interaction capabilities include real-time simulation steering and interactive 3D visualization. MatchPad [Legg et al. 2012] provides interactive visualization for real-time sports performance-analysis. This iPad application enables the real-time visualization of actions and events during a match. G. Sörös et. al [Sörös et al. 2011] introduce an augmented visualization system which enables users to get a personal augmented view of a visualized scene using their mobile phones or tablets.

3 The mobile interface

Before discussing the mobile client in more detail, it is necessary to provide a short introduction to the Visdom framework. The framework is primarily a visualization framework, allowing dataflows to be constructed from modular components called nodes. The basic nodes necessary to create a visualization or to process data are provided as a part of the framework, and users are free to extend them by developing new plugins supplying additional nodes. Simulation- and flood management- related functionality has been introduced to the system in this way.

The development of a dataflow takes place in the Visdom client application. As nodes are added to extend functionality, the user interface increases in complexity. Every node contributes widgets used to customize its behavior, but some nodes provide more complex components such as windows. These include 3D renderings, transfer function controls, information-visualization views, and other specialized components. The end result is a very powerful yet complex interface, consisting of multiple windows used to both examine results and exert control over them.

While it is possible to recreate these exact interfaces on a mobile device, the smaller screen space and different input modalities available make a detailed interface much more difficult to interact with. To counter this problem, we decided to pursue an approach based on the reduction of complexity available to the user on a mobile device. The two most important views for a flood manager were chosen, and recreated in a format more suitable for mobile use. Additional approaches were developed to expose other options and settings in a more intuitive way, but a large part of options is simply unavailable. All of the options and the dataflow can be set up in the desktop version of the application, and exported afterwards to the mobile device. This frees the user to focus on the parameters that matter, and which can and need to be adjusted in the field. The end result is a sleek and simple interface, as demonstrated in a video available online (`http://visdom.at/media/videos/mp4/visdomMobile.mp4`).

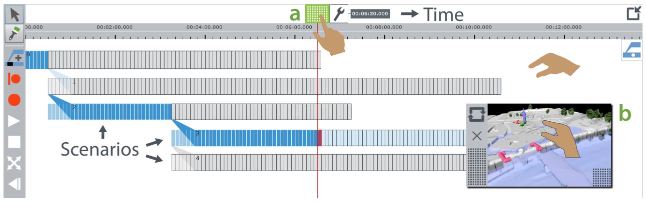

Figure 2: Temporal view in fullscreen mode (vertically cropped) visualizing multiple related scenarios as a horizontal tree of bars. (a) Time axis with time-navigation cursor. (b) View inlay showing a 3D rendering of the currently selected scenario (blue).

3.1 Temporal View

The most basic of the users' needs is navigation - both exploring the development of a single scenario and switching between alternatives. The temporal view, based on World Lines, handles these needs by visualizing multiple related scenarios in a tree-like fashion (Figure 2). A scenario is a simulation run with a specific set of parameters, represented by a horizontal World Line aligned to a time axis (Figure 2a). The parameters can be altered in the course of a simulation run to model sudden developments or actions. Each such modification creates a new scenario, initialized by the state of the parent scenario, and using new parameters. Visually, this is represented by the new World Line being linked to its parent by a branch. Thus, a tree of scenarios is formed. The user can select the time to be examined by moving a cursor placed on the time axis (Figure 2a). Tapping a World Line changes the scenario selected.

The modifications necessary to make the World Lines into the temporal view that runs on the mobile device were not extensive. The biggest problem turned out to be the input precision - on a mobile device, selection is harder to perform accurately than on the desktop. To remedy this, two changes were made. Multi-touch gestures for panning and zooming were added, so that any part of the tree can be brought into focus quickly. To minimize the number of times the user has to alter the view, the layout algorithm has also been modified to provide a more mobile-device friendly layout. Any newly created World Line is guaranteed to have a certain minimal size, and to lie in close proximity of its parent. If the existing tree must be altered to satisfy these constraints, the change is animated to allow the user to adapt to it.

3.2 Spatial View

To continue the exploration after a scenario and a time value have been selected in the temporal view, more detailed information should be made available to the user. The spatial view supplies information by showing a 3D rendering of a simulation state. The user can navigate the scene using finger-based gestures, and examine how the state changes in time by moving a special slider at the bottom of the screen.

At any time, the spatial view can also be used to change or introduce additional parameters into the simulation. By using their fingers, users can sketch directly onto the rendering of the scenario to change the parameters of the underlying simulation run. These changes can introduce new objects, such as mobile walls or sand bags. They can also alter existing parameters, e.g., increase the incoming water level, or breach a barrier (Figure 3a). An action-pool panel allows the user to choose from a list of changes that can be sketched (Figure 3c). A typical sketching session involves setting up the initial conditions by drawing incidents and possibly some

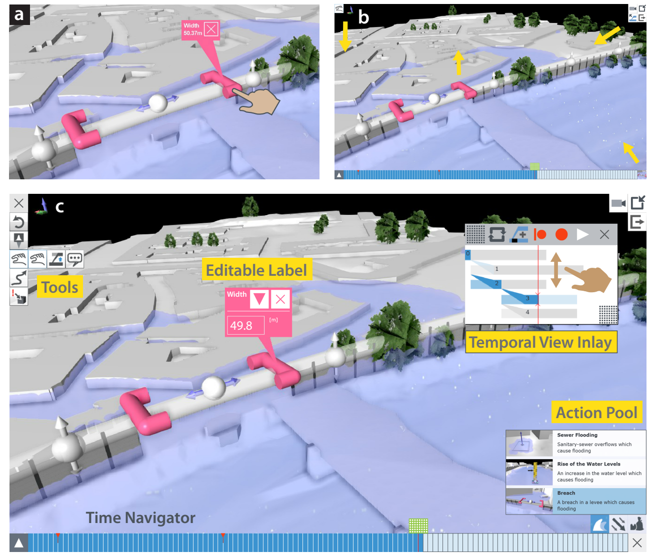

Figure 3: Spatial view in fullscreen mode showing a barrier-breach scenario. (a) Changing the width of a barrier breach using a sketch. (b) The minial interface without tools or menus. (c) The expanded interface, showing tools, an editable label, the action pool, and the temporal view inlay.

initial protection measures. To investigate different scenarios, the user sketches additional incidents as well as protection measures designed to counter them.

The spatial view contains many enhancements intended for the mobile device. Given that navigating to a good camera position is vital for sketching, we have paid special attention to navigation mechanisms suitable for mobile use. We support two modes of navigation - a first-person mode and an orbiting camera mode. Their behavior is consistent. The touch of a single finger causes a rotation of the viewpoint or the view direction, depending on the mode. Multi-finger gestures function much as they are expected to on mobile devices. Pinching causes the camera to move in the view direction, and panning causes movement in the plane perpendicular to the view direction.

3.3 Configuration views

The two main views described in the previous sections provide most of the control necessary for the configuration and examination of the scenario. However, while a sketching-based approach is useful, for some parameters other interfaces can be more expressive.

For example, a transfer function is a commonly used method of associating colors with numerical values. It can be used to map danger estimates to building colors, and thus easily show possible hazards. The representation and the way a transfer function is set up are well known to users, and it makes little sense to try and create an alternative mechanism for modifying one. Therefore a separate view is used for transfer-function setup (Figure 4b). This view is one example of the specialized control interfaces various types of simulation and analysis require. We call these configuration views.

Another example of a necessary configuration view is the settings panel (Figure 4d). Being able to configure the settings of each node in the dataflow is not useful on the mobile client, but a subset of settings may still be interesting. These can be selected on the desk-

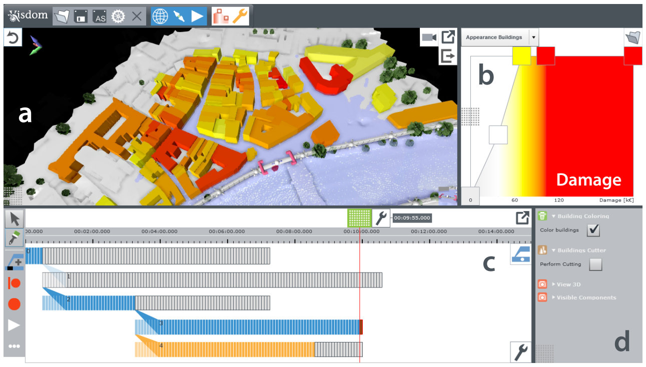

Figure 4: Using multiple views: the spatial (a), temporal (c), and two configuration views (b, d), used to color buildings according to damage using the transfer function (b).

top client, creating a less complex interface appropriate for a target user. The subset can then be configured on the mobile client.

Not all configuration views need to be persistent. We use additional views to improve the accuracy of manipulation. While sketching is intuitive, setting a desired numerical value requires a more precise approach. A partial remedy for this issue are rendered labels showing exact values of properties. These appear when manipulation is performed, and can be made persistent for properties of interest. By touching these labels, they turn into a miniature configuration view that can be used to enter the desired exact values of parameters (Figure 3c).

The configuration views are a necessary evil in the mobile client. While the spatial and temporal view suffice for almost all of the changes an expert needs to make, the configuration views need to be present for the few cases when they do not. The biggest issue with configuration views is that they consume precious screen space - a problem that will be discussed further in the next section.

3.4 Managing multiple views

The nature of the tasks commonly performed in the system often involve the use of both the spatial and the temporal view. For example, after sketching a new parameter value in the spatial view, the user wishes to advance the simulation in the temporal view, occasionally switching back to the spatial view to review the results. The simplest way to support such behavior is the placement of multiple views on-screen (Figure 4). The user is free to choose which views are visible, and also to position and resize individual views. This also allows changes introduced in the configuration views to be immediately reflected in the spatial and temporal view.

Unfortunately, having multiple views on-screen affords too little space for sketching or scenario-tree manipulation. Both of these activities require precision and detail, often possible only by using all the space provided by the mobile device screen. To this end, the overall design of the interface is geared towards a minimal number of menus and options on-screen. Every menu can be reduced to its minimal state, visible only as a button that expands it.

To allow for access to the functionality of multiple views, while focusing on only one, special mechanisms have been implemented. View inlays allow a miniature version of a view to be displayed atop another view in fullscreen mode. Depending on the inlaid view, different possibilities of use are available. The inlaid temporal view shows the scenario tree auto-centered around the current time and the selected scenario (Figure 3c). The inlay provides the basic functionality, and can be swiped to change scenarios and time steps easily. Similarly, the inlaid spatial view displays a small rendering, while supporting the same interaction gestures as the actual spatial view (Figure 2b). The time navigator appearing at the bottom of the spatial view is a special type of view inlay, that allows a single World Line to be navigated from a spatial view (Figure 3c).

4 Architecture and implementation

In the previous sections we have talked about the interface, its features, and the differences between the desktop and the mobile interface. Before continuing with the changes we have had to introduce to the system to facilitate the mobile client, another introduction to the Visdom framework needs to be made, from an architectural point of view.

Visdom is a client-server system. This choice had been made at its inception, with possible mobile and web clients in mind. The data persists on the server, while the client controls the settings which specify how the data is processed. When results need to be shown or updated, the client sends a new set of settings to the server, which performs simulation and rendering as needed, and returns results - usually an image.

Although this setup has caused some performance issues in the desktop version due to the additional costs of downloading an image from the GPU and transporting it to the client, it has worked well. The independence of server and client allowed us to develop them in two different languages: C++ for the server, and Flex for the client. C++ provides performance and access to a variety of libraries, and Flex supports many device types due to the Adobe Air platform it executes on. However, the creation of the mobile client proved that we were not able to predict all the changes that would be necessary for optimal use.

4.1 User input interpretation and the interactor stack

Before the creation of the mobile client, the user input interpretation functioned in a somewhat uneven way. For example, navigation was performed on the client side. Whenever the user wished to change the camera perspective by interacting with the mouse, the alterations of the camera settings were calculated on the client. Only the end camera perspective was forwarded to the server for rendering.

The advent of sketching introduced new kinds of user input interpretation. Both sketching and direct object manipulation require information that is not present on the client - the structure of the scene graph, the positions of various objects in the rendered image, etc. As it would be highly inefficient and impractical to forward this information alongside the image, the processing of such user input has to be left to the server. The result was a hybrid system in which the processing of user input was divided between the client and the server. The user had to select a tool to specify which type of interaction would happen in advance, so that the client would know if the input should be forwarded to the server. On the mobile client, this led to a constant switching of tools, a situation impractical for the user.

To remedy this problem, we decided to move all input processing to the server. The client presents the user with a set of tools. These include camera manipulation, sketching, object movement, etc. Each tool is a specification of entities called interactors that should be

Image size (kB)	Transfer time (ms)	Server time (ms)	Frames per second
47.5	40	15	18
112.0	51	59	10

Table 1: Rendering speed and transfer time depending on image size.

created on the server. The most basic interactor is the camera interactor - it interprets user input as camera alterations, and changes the perspective accordingly.

Other interactors such as a selection interactor, or spline-drawing interactor, can be placed atop the camera interactor in an interactor stack. The placement defines the order in which interactors are allowed to try processing the user input. For example, the drawing of splines can be constrained to surfaces designated as terrain. When user input arrives, the spline interactor examines the point where the interaction has taken place, and determines if the user has started the drawing motion on terrain. If this did not happen, the user input is passed downwards to another interactor, such as the camera interactor, who can interpret it as camera movement. By specifying an interactor stack, every tool defines what interaction functionality it allows for.

Such a setup grants two significant advantages. The first advantage is that interaction functionality can be specified separately. Each interactor is at its core a simple state machine, accepting user inputs and changing internal states and outputting data in response. By specifying an interactor stack, the logic of the individual interactors can remain simple, and easy to modify and maintain.

The second advantage is that the user input interpretation is context-sensitive. Once a tool is chosen, the user is free to use standard camera-manipulation methods if these do not have a different meaning within the tool. Zooming and pinching can be performed with every tool which does not explicitly declare that it wants to use them for another purpose - e.g. changing the size of an object on-screen. On a mobile device, this means that the user does not have to switch tools needlessly, allowing for a better user experience. The burden on the client is reduced while allowing for more complex and richer functionality to be integrated into the system.

4.2 Hardware and timings

Thanks to the interaction changes, the mobile client is a very thin and lightweight Adobe Air application deployed on a Samsung Slate tablet. The device proved sufficiently powerful to run the client, despite the overhead of the Air framework. The biggest technical issue in deploying the system was the bandwidth needed to deliver the data. The images sent from the server to the client can be relatively large. To transfer the data to someone on-site in a timely and interactive manner, we deployed a 40Mbps data connection, provided by a LTE USB mobile broadband stick. While the smoothness of interaction cannot be compared to the experience of using the desktop client, our field tests have shown that such a connection provides a comfortable experience, allowing the use of the device on-site.

We have also found that any further enhancements to the server-client architecture would be hard-pressed to improve the experience. The data in Table 4.1 shows that the limiting factor of the transfer time is latency and not bandwidth. While the transfer time could be improved by the use of better compression, the experience would not change significantly.

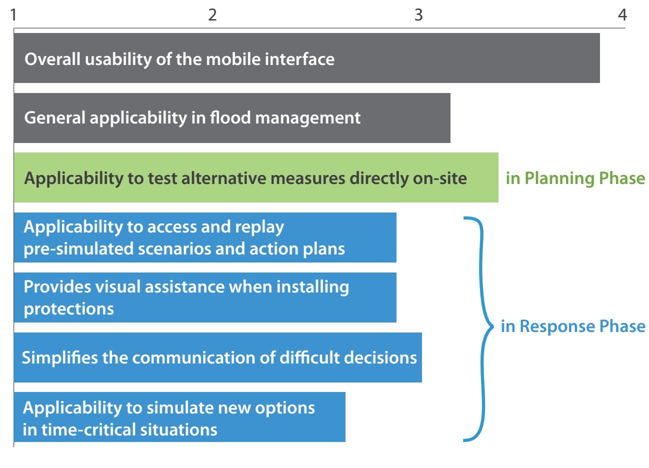

Figure 5: Questionnaire results. The answers were rated from 1 (worst) to 4 (best).

Given that the system is meant to be deployed in dire situations, it is worth discussing the possible disadvantages a user of Visdom Mobile might face. Severe weather that usually accompanies flood events could make the use of a tablet difficult. However, our domain expert partners already employ iPads for communication in the field, and have not yet had any issues with water making the devices unusable. In the case this could be a problem, rugged military tablets made especially for endurance exist and can be considered as an alternative.

A bigger issue is the question of mobile connectivity. It is entirely feasible that given a large flooding event, a mobile network might collapse due to either the desctruction of infrastructure, or sheer overload of panicked callers. The military tablets mentioned above allow for alternative means of communication based on military radio technology, supporting IP protocol-based networks. Unfortunately, we have not had the opportunity to test whether the bandwidths they provide would support the effective use of our system.

5 Domain expert feedback

In the course of our previous work, we have often collaborated with flood managers. Their work falls into two categories - preparation during the planning phase, and reaction during the response phase. The planning phase involves examining the risks of possible flooding, drafting plans to counter the risks, and finally deploying the resources that will be necessary to respond to a flood. Once the flooding begins, plans are executed, requiring coordination, and possibly improvisation if unexpected changes in conditions occur.

We wanted to ascertain for which purposes mobile access to a simulation-based system such as Visdom can be used, and gathered a group of experts to help us evaluate the usefulness of the approach. A group of eight flood-management experts were presented with a video depicting the mobile client being used in an outdoor environment and given a presentation on the possible uses of the system. Afterwards, they were asked to fill out a questionnairre rating the purposes, and to provide additional oral comments. The results of the questionnaire can be seen in Figure 5.

As the results show, all of the experts evaluated the interface as being particularly intuitive to use, resulting in a near-perfect score. The applicability to the area of flood management was also higly rated. When individual areas of application were considered, what surprised us was the importance experts assigned to support in the planning phase. Creating plans and exploring alternatives on-site were better rated than any of the response-phase uses.

However, these were also received well, with the experts giving them a positive rating. Abilities related to replaying rather than exploring alternatives were rated better. These were perceived as important in examining and executing plans, by visualizing expected developments or defenses to be built. A visual tool can also be used to communicate the possible dangers and justify decisions. All in all, even the worst-rated question received an average rating, and we consider the feedback to be positive.

To test the user interaction in more detail, a flood-protection consultant with experience in working with simulations has tested the interface. The expert was asked to perform certain tasks, and to report on his experience. He found the interface to be very intuitive, and the usability comparable to a desktop application. The minimal interfaces, window management, and view inlays were all highly rated, along with the various gesture functionality such as scenario swiping.

His assessment is that it would be possible to use the device on-site as presented. The interaction is smooth enough despite the server-based rendering, and the thin client could be useful in many situations. Apart from using the device for on-site inspections or planning, he outlined the economical implications as being particularly interesting. Given the distributed nature of the system, and the client-based session control, costs could be reduced by having a single server and many client devices, connecting as necessary.

6 Conclusion and Future Work

The ultimate design requirement of the mobile user interface presented in this paper is the simplicity of use. The range of users that could be using it, in on-site conditions that are as far from ideal as possible, require that the interaction is as streamlined as possible.

To achieve this, the simplicity and intuitiveness of the individual approaches helped. Both World Lines and the sketching-based rendering are relatively simple, and well-understood. However, they were designed with different input and output devices in mind, and the adaptations needed to make them work on a tablet were not trivial. We found that most needless user actions were the result of the two approaches used in conjunction. By designing mechanisms focusing on the system of views as a whole, we managed to build a client that can be used on a mobile device effectively.

We received positive feedback from the domain experts regarding the relevance of our mobile approach. One aspect of our future work revolves around strengthening cooperation and performing further field tests. We would also like to make use of GPS data to allow the user's position to be incorporated into the renderings, and perhaps pursue an augmented reality approach to delivering simulation results.

References

BETTS, B. J., MAH, R. W., PAPASIN, R., MUNDO, R. D., MCINTOSH, D. M., AND JORGENSEN, C. 2005. Improving situational awareness for first responders via mobile computing. Tech. rep., Smart Systems Research Laboratory, NASA Ames Research Center.

BUTSON, C., TAMM, G., JAIN, S., FOGAL, T., AND KRUGER, J. 2012. Evaluation of interactive visualization on mobile computing platforms for selection of deep brain stimulation parameters. *IEEE Trans. on Visualization and Computer Graphics (to appear)*.

DE GOOIJER, C., AND REUTER, C. 2009. FLIWAS, the right information on the right time for the right person to take the right decision. In *Proc. Final Conf. of the COST action C22 Urban Flood Management*.

HOLMES, I., AND KAWALSKY, R. 2006. The RealityGrid PDA and smartphone clients: Developing effective handheld user interfaces for e-science. In *Proc. of the UK e-Science All Hands Meeting 2006*, 502–509.

JERN, M., RICKNAS, D., STAM, F., AND TRELOAR, R. 2003. Visual user interface for PDAs. In *Proc. of the 7th Int. Conf. on Inf. Visualization*, 17–22.

KIM, S., JANG, Y., MELLEMA, A., EBERT, D., AND COLLINS, T. 2007. Visual analytics on mobile devices for emergency response. In *Proc. IEEE Symposium on Visual Analytics Science and Technology, VAST 2007*, 35–42.

LEGG, P., CHUNG, D. H. S., PARRY, M. L., JONES, M. W., LONG, R., GRIFFITHS, I. W., AND CHEN, M. 2012. MatchPad: Interactive glyph-gased visualization for real-time sports performance analysis. *Computer Graphics Forum 31*, 3, 1255–1264.

LUYTEN, K., WINTERS, F., CONINX, K., NAUDTS, D., AND MOERMAN, I. 2006. A situation-aware mobile system to support fire brigades in emergency situations. In *OTM Workshops*, vol. 4278 of *Lecture Notes in Computer Science*. 1966–1975.

MANCINI, E., WAINER, G., AL-ZOUBI, K., AND DALLE, O. 2012. Simulation in the cloud using handheld devices. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 867–872.

PAOLINO, L., ROMANO, M., SEBILLO, M., AND VITIELLO, G. 2010. Supporting the on-site emergency management through a visualisation technique for mobile devices. *Journal of Loc. Based Services 4*, 3-4, 222–239.

RIBIČIĆ, H., WASER, J., GURBAT, R., SADRANSKY, B., AND GRÖLLER, M. E. 2012. Sketching uncertainty into simulations. In *IEEE Trans. on Visualization and Computer Graphics (Proc. IEEE VIS 2012, to appear)*.

SÆTRA, M., AND BRODTKORB, A. 2012. Shallow water simulations on multiple GPUs. In *Applied Parallel and Scientific Computing*, vol. 7134 of *Lecture Notes in Computer Science*. 56–66.

SÖRÖS, G., SEICHTER, H., RAUTEK, P., AND GRÖLLER, E. 2011. Augmented visualization with natural feature tracking. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, ACM, MUM '11, 4–12.

STEPNOWSKI, A., KULAWIAK, M., CHYBICKI, A., AND RUCINSKI, A. 2010. Integrated visualization and analysis of threats for marine and coastal regions via a web-based GIS. In *Proceedings of the International Waterside Security Conference, WSS 2010*, 1–6.

Visdom Mobile Video - A demonstration of the interface. http://visdom.at/media/videos/mp4/visdomMobile.mp4, (last visited on Sept., 6^{th} 2012).

VONDRÁK, I., MARTINOVIČ, J., KOŽUSZNIK, J., UNUCKA, J., AND ŠTOLFA, S. 2008. FLOREON-system for flood prediction.

In *Proceedings of the 22nd European Conference on Modelling and Simulation, ECMS 2008*.

WASER, J., FUCHS, R., RIBIČIĆ, H., SCHINDLER, B., BLÖSCHL, G., AND GRÖLLER, E. 2010. World lines. *IEEE Transactions on Visualization and Computer Graphics 16*, 6, 1458–1467.

YUEN, D., MCARTHUR, E., WEISS, R., ZHOU, J., AND YAO, B., 2010. WebViz: A web-based collaborative interactive visualization system for large-scale data sets. AGU Fall Meeting 2010. Abstract IN23A-1354.

ZHENG, Z., HE, S., XIONG, Z., AND LIU, Z. 2011. Hydrologic prediction and visualization integrated mobile system based on service oriented architecture. In *Proceedings of the 19th International Conference on Geoinformatics*, 1–5.