Workshop on Visualisation in Environmental Sciences (EnvirVis) (2015)
A. Middel, G. Weber, and K. Rink (Editors)

Real-Time Visualization of Urban Flood Simulation Data for Non-Professionals

S. Grottel[1], J. Staib[1], T. Heyer[2], B. Vetter[1] and S. Gumhold[1]

[1]Computer Graphics and Visualization, TU Dresden, Germany
[2]Institute of Hydraulic Engineering and Technical Hydromechanics, TU Dresden, Germany

Figure 1: *Visualization of a flood simulation; three different points in time show the progression of the flood into urban living areas. Realistic rendering of the flood water increases the understanding and awareness of danger.*

Abstract

Perception and understanding of risk in predictive flood simulation data is a key factor in flood risk management. However, scientific visualizations are often hard to grasp for non-professionals. We present a visualization tool for interactive, realistic flood water rendering, incorporating a variety of optical effects. Our implementation is based on state-of-the-art algorithms of interactive computer graphics using animated texture coordinates. The flood simulation data contains water depth level and a flow field. Together with terrain elevation and buildings our tool allows for natural understanding of the data. We maintain interactive frame rates on standard desktop computers. Our results were judged useful by average citizens and political decision makers.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.8 [Computer Graphics]: Applications—Flood visualization

1. Introduction

Flood disasters in urban environments are critical concerning the danger to human lives and economical damage. Due to the predictions of the Intergovernmental Panel on Climate Change (IPCC) and the numerous devastating events in the past few years, flood risk management has been one of the key issues in environmental engineering. Great efforts have been undertaken in order to improve the tools for hydrologic and hydraulic modeling and simulation. Most of the research focused on the increase in computational speed for real-time analysis and on increasing the reliability of the model predictions. Less attention was paid regarding the aspect of risk perception, although being one key element in flood risk management. In particular, state-of-the-art rendering techniques and visualization possibilities are currently only be-

DOI: 10.2312/envirvis.20151089

Figure 2: *Visualization of flood simulation data with QGIS (http://qgis.org/). Although all simulation data is visualized, the information, especially the risk and danger, is hard to grasp for non-experts, even when choosing appropriate color maps.*

ing utilized to small extend. The current form of simulation data visualization (cf. Fig. 2) is useful for experts but not intuitively graspable by non-professionals, like interested average citizens and decision-makers from politics.

Our goal was to create a more realistic visualization of the flood water. We compromise for less accurate visual representations to increase the understanding of the danger resulting from the flood. Based on state-of-the-art algorithms from computer graphics for interactive water rendering, we designed an application to present our simulation data to a wide range of users (cf. Fig. 1). Our hydraulic simulations for these scenarios follow a shallow water model, storing water levels and flow velocities on a 2D unstructured grid.

In this paper we present the resulting visualization software. It incorporates realistic, interactive rendering of flood water based on time-dependent simulation data sets. The focus of the implementation is to be executable on a wide range of devices to reach many potentially interested users.

2. Related Work

Flood visualization usually follows one of two approaches: either for data analysis by experts or information presentation, which also aims at the general public. For example, [RWF*13] visualize the ensemble flood simulation data for exploration and simulation steering. Tutenel et. al [TKE13] present flood simuluation data as classical scientific visualization. In [WKS*14] a visual tool for flood management response planning is presented. An example for a commercial water management software is 3Di [3Di15]. Presentations to decision makers and the general public are usually much more designed and tailored for easier understanding, e.g. [Bai12]. We also aim at near photo-realistic but interactive rendering of flood water, to achieve a representation intuitively understandable for non-professionals.

The rendering of water plays an important role in computer graphics. The core concepts of waves and optics in open waters can be found in [Tes99]. Small scale water representations can be achieved by surface rendering [EMF02] of data simulated on a grid or utilizing particles. If the flow data is given or computed on a grid, interactive applications usually use a texture representation for rendering. This is known in the computer graphics and visualization communities as advecting [Ney03] or moving textures [MB96]. The core idea is to warp the texture coordinates or advect the texel's values along the directions given by a flow field. To counter undesired strong distortions of the original textures, multiple textures (usually two or three) are used and periodically reset. Blending between these time-shifted textures generates a smooth continuous visualization. Such techniques are used for scientific flow field visualization as well [vW02]. An overview of texture based techniques for flow visualization is given in [LHD*04].

Rendering can be directly integrated with a simulation. In a scientific context, [GKT02] simulated a dam failure and visualized the results in non-photorealistic fashion. As fully 3D simulations are oftentimes too expensive, alternative simulation methods can be used for limited scenarios, like 2D shallow water [HHL*05, BSA12] only storing water levels or 2.5D simulations of stacked water columns [MFC06]. Some applications extend continuous simulations with particles to describe additional effects like splashes, breaking waves [DHBF11], or bubbles and foam [TSS*07]. Utilizing the possibilities of GPGPU, simulations reach complex scenarios while maintaining interactive computation times [BHLN10]. Combinations of different approaches allow for large simulation data sizes and small scale wave features at the same time [CM10].

Interactive, realistic rendering of water is discussed in [Bel03], implementing advanced optics previously described (cf. [Tes99]), like refraction and reflection. [YYM05] extend reflection from flat mirrors or environment map reflections, using light-fields. The state-of-the-art approach for water rendering in interactive computer graphics is mesh displacement for large scale waves [Kry05], texture-based rendering of small scale waves [YNBH09b], and the use of particles for details [CC06, YNBH09a] not representable by a surface height map, like splashes and breaking waves [TMFSG07]. Purely particle-based renderings, like in [CLCC07], are in many cases too expensive to be used on large scales. Wave textures can be organized in tiles, for which the texture coordinates are advected, and which are blended together to obtain a continuous overall texture [vH11]. Particles can, in addition to optical small scale effects, be used to describe local details of interactions between water and solid objects [YHK07]. Small scale waves can also be generated parametrically based on water borders [YNS11]. To optimize rendering speed, the resolution of the water surface mesh can be adapted in a level-of-detail approach [Joh04], allowing to work with large data and to incorporate many vi-

sual effects at close visual range [SYDZ07]. This approach is applied in many modern games with high-quality graphics in which water plays an important role [Vla10, Hol12].

Our approach is similar to [vH11], as we use an uv map to animate the texture mapping and use tiles of constant flow, blended together to achieve a continuous representation. Several optical effects are added to increase realism.

3. Concept and Data

We did not incorporate water simulation in our tool, but only visualize data sets created by scientific simulations. To optimize the rendering process we convert the simulation data as part of a preprocessing step. Within this step, we also integrate and unify data from different additional sources.

The simulation itself uses an unstructured grid. Each grid point stores its position, water depth level and a flow vector. Obstacles, like buildings, are explicitly modeled by polygonal holes in the simulation grid. The current simulation does not address damage effects, like collapsing buildings, but focuses on prediction on water movement, like the time required for the flood water to flow off. The simulation data exported only stores unconnected points. In the preprocessing, we compute a Delaunay triangulation to reconstruct connectivity. We render the simulation grid into two 2D textures for each time step, one storing the water depth and one storing flow velocity vectors. We pack float values into multiple color channels to increase storage precision. For example, the red and green channel encode the flow direction components, while the magnitude is stored in the blue channel.

To provide an extended context for the simulation data we add terrain and building geometry data. The terrain elevation data is register with the simulation data based on RD83 geodetic reference coordinates. We use satellite/overflight ortho-images as terrain texture. Geometry data of buildings were provided by the Geo-Information Office of the state of Saxony, Germany. As this data did not include textures or color information, we could not achieve realistic looks for these parts. We reused the texture from the ortho-images. This produces good results for wide overviews, but in close-ups alignment problems between the texture and the models become apparent (cf. Fig. 1). Finally, the polygonal meshes describing buildings, are merged with the terrain geometry.

4. Implementation Details

Our implementation uses Java and OpenGL ES2. This reduces the available GPU capacities, e.g. no floating point GPU buffers are available. However, this also allows for platform independent execution on a wide range of systems. The relatedness between OpenGL ES2 and WebGL, allows for simple porting of our visualization as web application.

The terrain and skybox are rendered as static, textured geometry. The main aspect of our work is the rendering of the water data. The different aspects are shown in Fig. 3. The components of our water rendering are (a) the moving normal map for small scale waves, (b) reflection based on environment mapping, (c) additional foam to highlight turbulent regions, and (d) the water color based on the water depth.

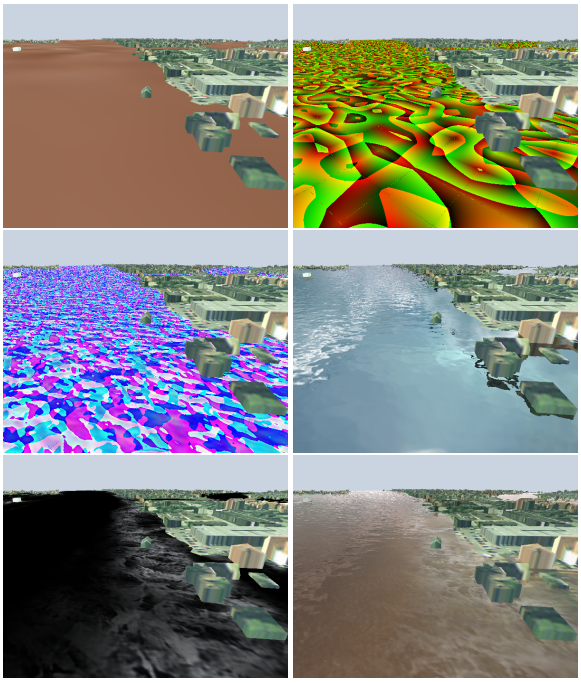

Figure 3: *The different optical elements of our water rendering: (from top left, to right and down) the base color of our muddy water, the coordinates from the moving uv map, the combined normal map, the reflection containing the sky box and buildings, the additional foam effect based on water-obstacle interaction, and the final result.*

The core idea is similar to [Vla10] but incorporates improvements mentioned in [vH11]: We did not incorporate geometric large scale waves, as we wanted to stay true to the simulated water levels. Instead we use color and normal maps to render small scale waves on the water surface to show the movement given by the flow data from the simulation. The texture coordinates are fetched from a time-dependent uv map. This map is organized in tiles storing the current distortion (advection) of the surface based on constant velocity. These tiles are stored in a low resolution buffer. In each update operation, the position and orientation of all tiles are updated by the corresponding values from the velocity map, resulting in the advection of the texture coordinates. Without additional measures, this would lead to massive distortions after only a few time steps. To counter this effect, we only distort each tile for a few time steps and reset it periodically. The hide this resetting, we blend two tile maps in an interleaved manner (cf. [Ney03]). To avoid the negative effects described in [vH11], the tile maps are not

reset to unit box coordinates but noise is added. For the final shading of the water surface a single normal map value is computed: the four tiles overlapping in the pixel are fetched, providing different uv coordinates each. Normal map values fetched with these coordinates are averaged for the final value. The top-right image in Fig. 3 shows the distorted texture coordinates resulting from one tile map. The middle-left images shows the resulting normal map values after the combination of both tile maps.

The shading of the water surface includes reflection and refraction values which are combined based on Snell's Law, similar to [Tes99]. Refraction is approximated by accessing the terrain texture. Reflection uses a skybox environment map in addition to a framebuffer storing the images of terrain and buildings mirrored at an approximate flat water surface. Access to both textures are deviated with noise modulated by normal map strength.

The additional foam buffer stores the strength of turbulence in the flow data. During each update operation the stored values are decreased by a fix amount and turbulence from the current flow data is added. This turbulence is derived from the gradient in the flow data texture, i.e. strong local variations in the flow vectors. This dissipates previous turbulence and introduces new flow smoothly (cf. Fig. 3 lower left). These values are used in the water surface shading to blend between two color textures, one only showing small ripples, and one with strong foam, almost white water.

Finally, for the water base color a pseudo-volumetric shading effect to the water is used to show the water depth: We blend the color of the sub-water terrain with the water color. Similarly, we adjust the color of the water surface, including it's transparency. Thus, deep waters appear opaque and shallow areas show the terrain beneath (cf. Fig. 1).

5. Results and Discussion

For testing, we used small synthetic data sets during development and one real-world data set for the final evaluation. We chose the region of Dresden-Laubegast (Germany), traditionally being one of the hot spots during flood events in the river Elbe. The geometry data consisted of 222,972 vertices (34,637 for terrain and 188,335 for buildings) and 327,976 triangles (68,856 for terrain and 259,120 for buildings). The simulation was performed on a grid with 34,637 data points over 95 time frames. Our preprocessing converted each time step into two 512×512 textures, one for the water depth values and one for the velocity field. Stored as PNG files, this data has a size of 39.4 MB.

The overall rendering perfomance was tested on two computers: System 1 had an Nvidia GeForce GTX 680 GPU with 2 GB DDR5 VRAM, an Intel Core i7-3770K@3.5GHz CPU with 16 GB main memory, and used the Windows 8.1 operating system. System 2 had an Nvidia GeForce GTX 660 GPU, the same CPU and the same amount of memory, and used Ubuntu Linux 14.04 as operating system. The test render setup showed the complete data set filling roughly the whole window at 1920×1080 pixels. The average frame rate on system 1 was 156 FPS and on system 2 was 136 FPS. We believe these are good performance values given the amount of data, the complexity of the shaders, the amount of texture accesses, and the multiple rendering passes required.

We consider the image quality of our approach very high for interactive rendering. The quality of the water rendering can be judged from our images and the supplemental video. The chosen textures and noise values nicely create the look of flood water, comparable to photos of actual events. Some issues concerning the quality arise from the blending between patches, especially in border regions, where the flow field varies strongly between neighboring data points.

We used a public presentation and discussion with experts and non-professionals to evaluate the effectiveness of our visualization. The results generated by the hydraulic simulation data, digital terrain data, digital maps and 3d city model data were presented at 35th Conference on Hydraulic Engineering in Dresden in March 2014. We received very positive feedback, among others from decision makers, such as the environment office of the city of Dresden and the Dam Authority of the Free State of Saxony. The visualization was considered intuitive and useful for presentation and discussion, especially in combination with classical scientific visualization. Similar visualizations were requested for different areas as well. Preparations to extend our application for these scenarios are currently underway.

6. Conclusion and Future Work

We presented a visualization application of realistically rendered water in flood simulation data. The motivation is to provide an intuitive view which conveys the disaster and danger. To be able to reach a wider user base, including non-professionals, we implemented our approach using portable technology and preprocessed the simulation data for easy access and handling. Our renderer is based on state-of-the-art algorithms and incorporates many optical effects to increase image quality and realism, while maintaining interactive frame rates. Our results were assessed useful by decision makers and we plan to continue our efforts.

Currently, we are extending our work in several aspects: As several additional scenarios were requested by political decision makers, we stream line our data conversion and visualization process towards an easy-to-use tool. This includes the incorporation of enhanced data on landscape and building models from additional sources. We further simplify the user interface towards an application for a non-professionals for exploration and information gathering about flood risks. We also work on integrating visual elements from scientific visualization, like glyphs and iso-lines highlighting numeric values. Based on our current implementation we want to set up a web-based information portal.

Acknowledgements

We thank Ludwig Schmutzler (TU Dresden) for his help with the supplemental video. This work was partially funded by BMBF Project ScaDS.

References

[3Di15] 3di waterbeheer, 2015. http://www.3di.nu. 2

[Bai12] BAINES I.: Exeter - flood visualisation. https://www.youtube.com/watch?v=0QL0hYIURyk, Environment Agency UK, 2012. 2

[Bel03] BELYAEV V.: Real-time simulation of water surface. In *In GraphiCon-2003* (2003), Press, pp. 131–138. 2

[BHLN10] BRODTKORB A. R., HAGEN T. R., LIE K.-A., NATVIG J. R.: Simulation and visualization of the saint-venant system using gpus. *Comp. Vis. Sci. 13*, 7 (2010), 341–353. 2

[BSA12] BRODTKORB A. R., SÃĘTRA M. L., ALTINAKAR M.: Efficient shallow water simulations on gpus: Implementation, visualization, verification, and validation. *Computers and Fluids 55*, 0 (2012), 1 – 12. 2

[CC06] CHIU Y.-F., CHANG C.-F.: Gpu-based ocean rendering. In *Multimedia and Expo, 2006 IEEE International Conference on* (July 2006), pp. 2125–2128. 2

[CLCC07] CHANG J.-W., LEI S., CHANG C.-F., CHENG Y.-J.: Real-time rendering of splashing stream water. In *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on* (Nov 2007), vol. 1, pp. 337–340. 2

[CM10] CHENTANEZ N., MÜLLER M.: Real-time simulation of large bodies of water with small scale details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2010), SCA '10, Eurographics Association, pp. 197–206. 2

[DHBF11] DALRYMPLE R., HERAULT A., BILOTTA G., FARAHANI R.: Gpu-accelerated sph model for water waves and free surface flows. *Coastal Engineering Proceed. 1*, 32 (2011), 9. 2

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (July 2002), 736–744. 2

[GKT02] GOUDA M., KARNER K., TATSCHL R.: Dam flooding simulation using advanced cfd methods, 2002. 2

[HHL*05] HAGEN T., HJELMERVIK J., LIE K.-A., NATVIG J., OFSTAD HENRIKSEN M.: Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory 13*, 8 (2005), 716–726. 2

[Hol12] HOLDER D.: Water Technology of Uncharted. In *Game Developers Conference* (2012). 3

[Joh04] JOHANSON C.: *Real-time water rendering - Introducing the projected grid concept*. Master's thesis, Lund University, 2004. 2

[Kry05] KRYACHKO Y.: *Using VertexTexture Displacement for Realistic Water Rendering*. No. 2 in GPU Gems. NVIDIA Corporation, 2005, ch. 18. 2

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum 23*, 2 (2004), 203–221. 2

[MB96] MAX N., BECKER B.: Flow Visualization Using Moving Textures. In *Proceedings of the ICAS/LaRC Symposium on Visualizing Time-Varying Data* (1996), Banks D. C., Crockett T. W., Stacy K., (Eds.), NASA Conf. Pub. 3321, pp. 77–87. 2

[MFC06] MAES M. M., FUJIMOTO T., CHIBA N.: Efficient animation of water flow on irregular terrains. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia* (New York, NY, USA, 2006), GRAPHITE '06, ACM, pp. 107–115. 2

[Ney03] NEYRET F.: Advected Textures. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)* (july 2003). 2, 3

[RWF*13] RIBIČIĆ H., WASER J., FUCHS R., BLÖSCHL G., GRÖLLER E.: Visual analysis and steering of flooding simulations. *IEEE Transactions on Visualization and Computer Graphics 19*, 6 (2013), 1062–1075. 2

[SYDZ07] SHI S., YE X., DONG Z., ZHANG Y.: Real-time simulation of large-scale dynamic river water. *Simulation Modelling Practice and Theory 15*, 6 (2007), 635 – 646. 3

[Tes99] TESSENDORF J.: Simulating ocean water, 1999. Copyright Jerry Tessendorf. 2, 4

[TKE13] TUTENEL T., KEHL C., EISEMANN E.: Interactive visual analysis of flood scenarios using large-scale lidar point clouds. poster, May 2013. Geospatial World Forum 2013. 2

[TMFSG07] THÜREY N., MÜLLER-FISCHER M., SCHIRM S., GROSS M.: Real-time breaking waves for shallow water simulations. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), PG '07, IEEE Computer Society, pp. 39–46. 2

[TSS*07] THÜREY N., SADLO F., SCHIRM S., MÜLLER-FISCHER M., GROSS M.: Real-time simulations of bubbles and foam within a shallow water framework. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 191–198. 2

[vH11] VAN HOESEL F.: Tiled directional flow. In *SIGGRAPH Posters* (2011), ACM, p. 19. 2, 3

[Vla10] VLACHOS A.: Water Flow in Portal 2. In *SIGGRAPH Course on Advances in Real-Time Rendering in 3D Graphics and Games* (2010). 3

[vW02] VAN WIJK J. J.: Image based flow visualization. *ACM Trans. Graph. 21*, 3 (July 2002), 745–754. 2

[WKS*14] WASER J., KONEV A., SADRANSKY B., HORVATH Z., RIBIČIĆ H., CARNECKY R., KLUDING P., SCHINDLER B.: Many plans: Multidimensional ensembles for visual decision support in flood management. *Computer Graphics Forum / Proceedings of Eurovis 2014 33*, 3 (2014), 281–290. 2

[YHK07] YUKSEL C., HOUSE D. H., KEYSER J.: Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007). 2

[YNBH09a] YU Q., NEYRET F., BRUNETON E., HOLZSCHUCH N.: Scalable real-time animation of rivers. *Computer Graphics Forum (Proceedings of Eurographics 2009) 28*, 2 (mar 2009). 2

[YNBH09b] YU Q., NEYRET F., BRUNETON E., HOLZSCHUCH N.: *Spectrum-preserving texture advection for animated fluids*. Tech. Rep. RR-6810, INRIA, Grenoble, France, 2009. 2

[YNS11] YU Q., NEYRET F., STEED A.: Feature-based vector simulation of water waves. *Comput. Animat. Virtual Worlds 22*, 2-3 (Apr. 2011), 91–98. 2

[YYM05] YU J., YANG J., MCMILLAN L.: Real-time reflection mapping with parallax. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 133–138. 2