Integrated Simulation and Visualization for Flood Management

Daniel Cornel
cornel@vrvis.at
VRVis Forschungs-GmbH
Vienna, Austria

Andreas Buttinger-Kreuzhuber
buttinger@waterresources.at
TU Wien
Vienna, Austria

Jürgen Waser
jwaser@vrvis.at
VRVis Forschungs-GmbH
Vienna, Austria

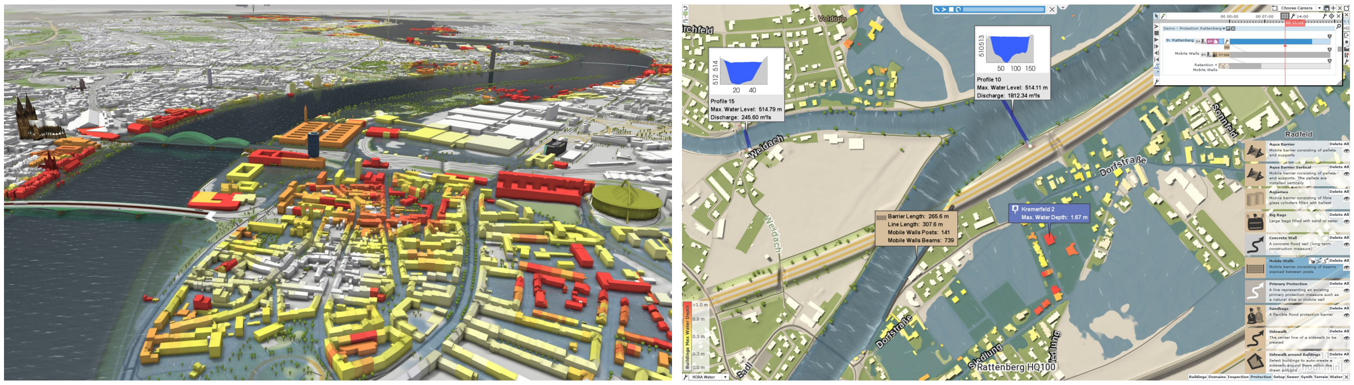

Figure 1: Interactive decision support in Visdom. (Left) Playback of river flooding. (Right) Planning of flood protection.

CCS CONCEPTS

• **Information systems** → **Decision support systems**; • **Computing methodologies** → *Interactive simulation*; • **Human-centered computing** → *Geographic visualization.*

KEYWORDS

interactive, decision support, flood simulation, water rendering

ACM Reference Format:
Daniel Cornel, Andreas Buttinger-Kreuzhuber, and Jürgen Waser. 2020. Integrated Simulation and Visualization for Flood Management. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '20 Talks), August 17, 2020.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3388767.3408335

1 INTRODUCTION

Floods are the most frequent natural disasters worldwide, and great efforts are being made in flood management to reduce the damages they cause. Numerical flood simulations allow flood managers to simulate flooding incidents before they happen to safely explore the best ways of mitigation. To ease this workflow of simulation, 3D visualization, exploration, and planning, we integrate all tasks in our interactive decision-support system Visdom. This scenario-based system allows the user to maintain many different flooding scenarios at once for analysis and comparison. For navigation through scenarios and through time, we use a track-based user interface similar to that of video editing software. For spatial navigation and interactive visualization, we take a lot of inspiration from video games, in particular city-building games. Many of the requirements for video games, such as performance, aesthetic presentation, and intuitive interaction, are similar for serious applications. However, there are also differences. Our input data are not the output of an art pipeline, but raw and sometimes erroneous real-world data in the range of several gigabytes. The visual output of our system has to be, above all, reliable and faithful to the input and simulation data, which in games is often not as crucial as runtime performance. Yet, our generalized techniques for real-world data without artistic control might be interesting for video games, for example when addressing topics such as dynamic inundation of procedurally generated environments.

2 INPUT DATA

Our system is a 3D geographic information system that reads georeferenced data acquired from authorities and open data services. For the setup of these data, we use a modular dataflow system with a visual programming interface similar to the Unreal Engine's Blueprints system. A job system resolves dependencies between modules and executes data acquisition and processing jobs in parallel. The data themselves can be separated into data visualized to provide a geospatial context for orientation and navigation, such as vegetation or high-resolution building meshes, and data required as boundary conditions for the flood simulation. Our simulation is grid-based, so it operates on a discrete digital elevation model (DEM) defined on a Cartesian grid. The DEM resolution in areas of interest usually ranges from 0.5 to 5 meters and is a tradeoff between precision and simulation time. Additionally to the elevation data, the roughness of the terrain, which influences the flow velocity, is derived from land use vector data. Wall boundary conditions by buildings and flood protection structures are created from vector data rasterized on the simulation domain. Finally, flooding parameters such as inflow conditions or precipitation rates are specified for the simulation, which in our system can either be set manually by the user, imported from historic floodings, or streamed in real time from sensors or prognosis services.

3 SIMULATION

The simulation we use for river floods and heavy rains is a validated, mass-conserving, state-of-the-art shallow water simulation using the finite-volume method and is one of the fastest flood simulations available. We use our own second-order numerical scheme in time and space [Buttinger-Kreuzhuber et al. 2019] to solve the shallow water equations with high accuracy, although faster first-order schemes would already be sufficient to achieve visually plausible results. The grid structure of the input data is well suited for parallelization, which makes it much more efficient than simulation on triangular meshes. With a CUDA implementation tailored to hardware features of Nvidia's consumer-grade cards [Horváth et al. 2016], we can simulate typical urban flooding scenarios many times faster than real time. We also include bidirectionally coupled sewer network and infiltration models to provide a more comprehensive model of the water flows. This is important for urban heavy rain scenarios, where an overburdened sewer network can cause new floods by sewer overflows. For flood and storm water management, accuracy and robustness of the simulation are crucial, because flood mitigation measures are based on the simulated water propagation. This is why we usually favor accuracy over simulation time. For demonstration or entertainment purposes, however, it is easily possible to increase the performance by decreasing the resolution of the simulation domain. The resulting data consisting of absolute water levels, relative depths, and 2D velocities are optionally resampled on a quadtree to reduce their size for visualization.

4 VISUALIZATION

Traditional geographic information systems only offer 2D visualization that domain experts are used to, but multiple user studies suggest that understanding the severity of flooding scenarios and orienting in the virtual world is easier in three dimensions. This is why we offer seamless switching between 2D and 3D visualization. Most of the visualization challenges naturally arise in three dimensions, where we have to deal with occlusion, perspective, and proper tessellation. The large scale of the data also forces us to frequently update dynamic data in the order of several gigabytes at runtime. The main offenders here are the water heightfield output by the simulation, the terrain heightfield sampled from the DEM, and arbitrarily complex vector data for land use polygons, building footprints, street and railway lines, and many more.

To render heightfields, a continuous surface has to be reconstructed from the discrete data by interpolation, which is a costly process that has to be repeated every frame. For water surfaces, we use third-order interpolation to obtain natural shorelines. However, for data defined on a quadtree, all implicit neighborhood information is lost, which makes most C^1-continuous reconstruction techniques too expensive for real-time application. We developed a real-time approach that replaces third-order interpolation in a quadtree by two bicubic interpolations in sparse regular grids of two consecutive quadtree levels and a linear combination of the two values to a C^1-continuous result, which we made available online [Cornel 2019]. Once we can reconstruct a closed surface, we sample the height of triangle vertices for mesh generation. To distribute triangles evenly in screen space for all possible zoom levels, we rely on recursive hardware tessellation [Lee et al. 2013], which allows us to subdivide already subdivided triangles once more. In games, surface reconstruction and triangulation at runtime can sometimes be avoided completely by confining the water surface to a control mesh. But if dynamic inundation at runtime is required, these steps can still be accelerated by reducing the size and resolution of the simulation domain or by using a lower-degree polynomial as interpolant.

The final water surface is shaded with various effects known from games to approximate the complex light interactions of water in real time, such as screen-space reflections and distance-dependent blurring of refractions. However, instead of achieving an appearance as realistic as possible, we aim for an aesthetic, but insightful visualization of important flow properties, for which we use exaggerated waves and foam synthesized from the water's velocity field. For the waves, we combine the superposition of individual wave functions commonly used for open waters with tile-based approaches commonly used for local flows such as rivers. By using continuous wave functions within tiles rather than static textures with wave patterns, we avoid repetitive wave arrangements and eliminate pulsing artifacts. Different tile sizes for waves of different wavelengths allow us to visualize small-scale local flows on top of large-scale principal flows with arbitrary wavelengths. On top of the water surface displaced by waves, we display foam with a cellular noise function at locations of high velocity or turbulence.

We render our terrain heightfield with the same reconstruction and tessellation as the water, which in sum takes between 10 and 20 ms, depending on the scene. On top of the terrain, we display geographic information data in the form of polygons and lines to provide a geographic context. We triangulate and extrude these shapes in a preprocessing step and intersect the volumes with the terrain mesh by using a stencil buffer pass to determine per-fragment containment similar to the shadow volume algorithm. This approach allows us to display thousands of shapes pixel-accurately and without z-fighting at interactive framerates.

ACKNOWLEDGMENTS

VRVis is funded by BMK, BMDW, Styria, SFG and Vienna Business Agency in the scope of COMET – Competence Centers for Excellent Technologies (854174) which is managed by FFG.

REFERENCES

Andreas Buttinger-Kreuzhuber, Zsolt Horváth, Sebastian Noelle, Günter Blöschl, and Jürgen Waser. 2019. A Fast Second-Order Shallow Water Scheme on Two-Dimensional Structured Grids over Abrupt Topography. *Advances in Water Resources* 127 (2019), 89–108.

Daniel Cornel. 2019. Shadertoy - Adaptive Grid Interpolation. http://shadertoy.com/view/WsXXRf (last visited on June, 30th 2020).

Zsolt Horváth, Rui A.P. Perdigao, Jürgen Waser, Daniel Cornel, Artem Konev, and Günter Blöschl. 2016. Kepler Shuffle for Real-World Flood Simulations on GPUs. *The International Journal of High Performance Computing Applications* 30, 4 (2016), 379–395.

Hyunjin Lee, Yuna Jeong, and Sungkil Lee. 2013. Recursive Tessellation. In *SIGGRAPH Asia Posters.* ACM, New York, 16:1.