Volume Rendering for Deep Water Asteroid Impact Simulations with Blender and VTK

Sam Verhezen (11259566), Rebecca Davidsson (11252138)

Abstract

It is of great importance to study the effects of an asteroid hitting the ocean, as the chances are highest that such an impact occurs in the water. Understanding the consequences could aid in taking the correct measures and precautions if an incident occurs. Clear scientific visualizations are valuable to shed light on the complex data ensembles which result from asteroid collision simulations. This report utilizes ParaView in combination with the Visualization Toolkit (VTK), as well as Blender to visualize and render the data. Additional calculations were done using VTK to obtain detailed insight in the variables. At last, animations are created that prove to be informative for understanding the temporal aspect of the data. This led to the distinction of three phases in the data, namely pre-collision, collision and post-collision.

Keywords: Asteroid impact simulation, scientific volume rendering, ParaView, VTK, Blender, reflective shadowing

1. Introduction

It is nearly unavoidable that some day an asteroid will collide with the earth. The chance of an asteroid hitting the ocean is higher than it hitting the land since approximately 70% of the Earth is covered with oceans. Such a collision can have major consequences such as asteroid-generated tsunamis (AGTs) [1]. Dr. G. Gisler, head of the National Laboratory of Los Alamos (LANL), is researching which characteristics of asteroids are indicators of concerning impact on earth. His simulations include elevation of airburst, asteroid mass, angle of entry and composition. These factors determine the amount of kinetic energy that is transferred when the asteroid hits the ocean. The greater the magnitude of the energy transfer, the greater the consequences of the impact [2].

The elevation of airburst may differ significantly per asteroid. Some explode on ocean impact, whereas others explode kilometers prior to collision. The latter is denoted by *airbursting*. The kinetic energy transferred into the ocean differs on the elevation of airburst. Furthermore, this distinction is important as airbursts can result in alternative mechanisms of wave formation [2].

Creating a clear visualization is important to understand the impact of an asteroid crashing in the sea. In addition, it can provide insight in which ways the various parameters interact. This report focuses on a non-airbursting asteroid to analyse the effect it has on the water surface. In order to derive informative and detailed visualizations from the simulated data, multiple visualisation softwares are used. These include ParaView, the Visualization Toolkit (VTK) and Blender. The findings can be used in further analyses of the consequences of water asteroids impacts, e.g. in AGT research.

2. Methods

2.1 Data

The data used for this project is the Deep Water Impact Ensemble Data Set, generated using High Performance Computing (HPC) by scientists at the LANL. This data set was generated from simulations of an asteroid crashing in the ocean [2]. It consists of eleven scalar field variables; density (rho), pressure, temperature, volume fraction of water and volume fraction of the asteroid. The data included multiple time steps at a spatial resolution of 300x300x300 with four scalars. For this project, research was focused on the yA31 dataset, which contained data from an asteroid crash without airburst, with a radius of 250 m and an incoming angle of 45 degrees. The data consists of a large combined size of 109 GB. Considering the limited computing resources, the dataset was narrowed down to a selection of 15% of the data by selecting 70 of the 460 files with equal time spacing between the files.

2.2 Simulation

A top-down analysis of the data was performed. The volumes of the parameters pressure, water, asteroid, rho and temperature were rendered and inspected. In this section, the utilized visualization softwares are described in detail.

2.2.1 ParaView

The data was first explored using ParaView. Volumes of all parameters were rendered and inspected for multiple time steps to obtain insight on the data. ParaView was also used to explore data of asteroid and water volume with shadowing. The OSPRay functionality enabled adding ray tracing to the images. However, the software is limited to shader customization. Conclusively, ParaView proved most useful in automating the stages

of opening and rendering files. Therefore, the choice was made to render the animation of images over time using ParaView.

2.2.2 Blender

Blender [3] was used to overcome the limitations from ParaView and explore the visualizations with shading and reflection. In order to convert the .vti files to use in Blender, 5 preprocessing steps were required.

1. The .vti files were first rendered in ParaView with volume rendering of two scalars; water and asteroid volume.
2. The *contour* filter was applied with a linear scale and appropriate values for 10 isosurfaces of v03. The contour filter generates isosurfaces from scalar values.
3. A geometry was created, using the filter *appendgeometry*.
4. The file was saved using .ply format saving the used color scale and alpha values.
5. After opening the converted .ply file, scales were adjusted to fit the window.

Automating the opening and rendering of files was more complex in Blender. Using the *blenderpy* package, scripting the visualizations accelerated this process. Furthermore, Blender enabled the use of depth and various camera movements, making it easy to explore the data. A reflection plane was added as an object along with a Glossy BSDF node to create reflection of the water volume object. This feature is mostly used for materials such as metal and mirrors [4], however, it gives a good sense of depth when applying it other materials.

2.2.3 VTK

The spatial distributions were analyzed with volumetric visualizations, including animations over multiple time steps. The Visualisation Toolkit (VTK) in python was used for generating multiple output images in order to create the animation. Code for automation of loading and rendering was written in VTK. The volumetric visualizations were analyzed using different lookup tables, color schemes, transfer functions and thresholding. In addition, mean and max values of all scalars over time were calculated from the input matrix in 3D generated by `vtk_to_numpy`. The 3D matrix of water volume and density was used to calculate the ranges in which water splashed over time.

2.3 Pipeline

An overview of the used visualization pipeline for a single image is shown in Figure 1. This pipeline was performed for every image in order to create animations. Reading the .vti files with `vtkXMLImageDataReader` took up the most time: 2 - 3 seconds per file. For this reason, an extra preprocessing step was introduced to reduce the reading time. The simulation data was converted into HDF5 format and stored locally. Data for each scalar field and time step was loaded separately when needed. This had the advantage of only loading data that was currently needed. Furthermore, new figures could be loaded in parallel with volume rendering, which made the initial data investigation a lot faster.

After reading the files, converting the values to a flat array using `vtk_to_numpy()` took approximately 0.17 ms and was the second most time consuming function. This step made it possible to analyse the data for a given scalar value in array format. It was also used to get the range of values for a given scalar to create a fitting color scheme and detect the maximum height of water splash over time (Figure 6). Thereafter the `vtkPiecewiseFunction` was applied to map scalar values to a given opacity. Using the `vtkLookupTable`, an appropriate color scheme was set for each scalar value, based on the maximum and minimum values for a given scalar. The lookup tables should be chosen in such a way that they highlight important features and minimize less important details [5]. Color was used to improve both the visual analysis of the data as the understanding of values represented by the color schemes.

To visualize the density values in combination with pressure, `vtkThreshold` was used. This enabled the relevant values of rho to appear in the image by filtering out all the low values.

Finally, a direct volume renderer based on a GPU implementation of a ray-casting approach, `vtkGPUVolumeRayCastMapper`, was used as mapping function. This enabled relatively fast mapping with fair quality.

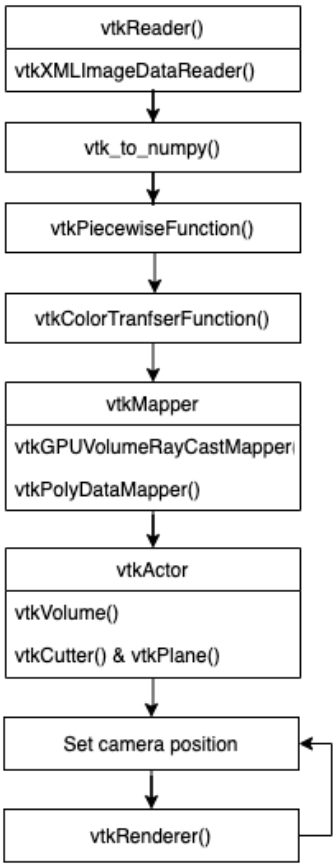

Figure 1. VTK Visualization pipeline for one single .vti file. Output of the pipeline is a converted .png image.

2.4 Animations

Several animations were constructed of which an overview can be found in Table 1. Animation 1, which is not necessarily an animation but rather an interactive visualization, has the

purpose to reflect multiple scalar values at a desired time step and camera position. This interactivity was created by using *cvlibd* [6]. The timesteps in the viewer are also connected to graphs, showing scalar values over time for the corresponding image. Adding interactivity allows for an attractive visualization that can summarize a lot of information at once. The interactive graph is divided into two groups:

- **Pressure (μbar) and density (rho) over time.** For pressure and density, the color scales were chosen to be hues of blue and yellow respectively.
- **Temperature (eV), asteroid volume and water volume over time.** Three color scales were chosen in such a way that important aspects were highlighted and the three features could be clearly separated.

The second animation shows details of temperature values for a single timestep. In comparison to all other visualizations that were based on volume rendering, this graph also includes slicing. The third animation shows a 360 degree view of the asteroid and water volume of a single timestep, made with Blender. The rotating view allows views from other positions which where not given in any other visualization. The last animation shows VTK volume rendering of temperature and pressure over time and gives a good summary of all figures presented.

Link/Reference	Animation
Animation 1	Interactive viewer of multiple images, showing water volume, asteroid volume, pressure and density over time.
Animation 2	Analysis of temperature with vtk clipping.
Animation 3	360 degree view of the asteroid and water volume after collision, where yellow represents asteroid volume and red water volume.
Animation 4	Volume rendering of temperature and pressure over time.

Table 1. Links to the referred animations.

3. Results

The animations can be accessed online via github pages (https://rebeccadavidsson.github.io/Paraview_VTK/). In this section, some snapshots of the results and a number of generated plots will be evaluated.

Blender and ParaView were used for data exploration. As an example, Figure 2 shows the post-collision density (rho) and pressure (prs), rendered with both Paraview and Blender separately. In this case, Blender was used to effectively show the size of the impact on the density by incorporating a sense of depth by reflections. Paraview was used to show differences in scalar values by using a color scheme, where darker colors represent a higher density. For the close-up in Blender, feature-preserving smoothing was applied to filter out the noisy isosurface.

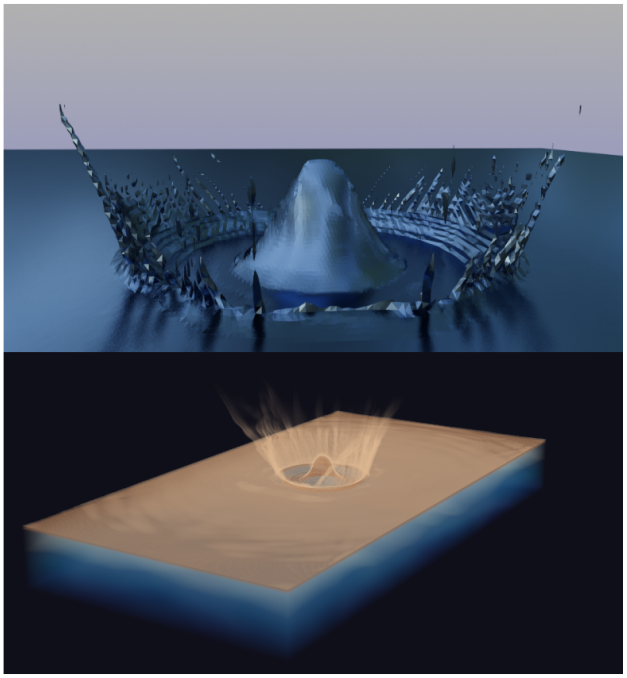

Figure 2. Comparison of parameters density and pressure (post-collision, t=3.7s) in Blender and ParaView. The upper picture is generated using Blender and the bottom picture uses ParaView, where brown and blue are the density and pressure, respectively. Darker colors represent a higher density.

Additionally, some extra visualization features such as reflection and shadowing were explored in Blender and ParaView (see Figures 3 and 5, respectively).

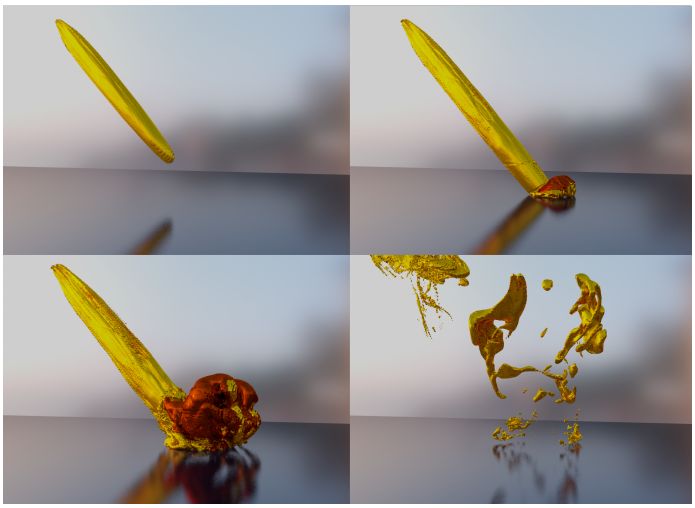

Figure 3. Blender reflective shader showing volumetric rendering over time. Yellow depicts water volume and red indicates asteroid volume. Time steps from top left to bottom right are 0.68 s., 0.97 s., 1.52 s. and 4.62 s.

The data exploration is useful for distinguishing three phases in the data over time. These phases include the period before the asteroid has hit the ocean (*pre-collision*), the moment of impact (*collision*), and the phase after the impact (*post-collision*). This

can be seen in Figure 3, where time steps 0.68 s., 0.97 s., 1.52 s. and 4.62 s were used to visualize the different stages. The collision with the ocean occurs at approximately 0.97 s.

To give a sense of the range and size of the impact before and after collision, the volume of the four time steps were rendered in Blender and can be seen in Figure 4. In line with Figure 3, a reflecting surface and shadowing create a sense of depth, allowing the comparison of impact size.

Figure 4. Comparing the range of water splash for different timesteps. Time steps from right to left are again 0.68 s., 0.97 s., 1.52 s. and 4.62 s. Here, yellow depicts water volume and blue asteroid volume.

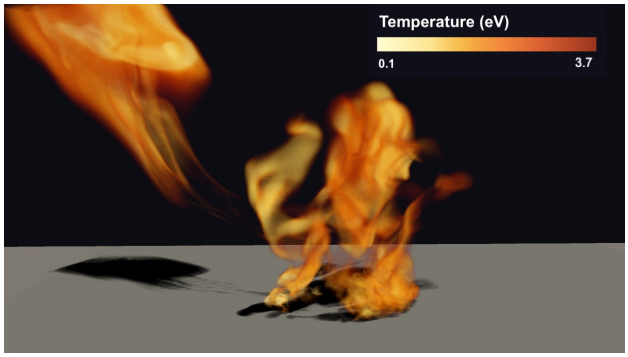

Figure 5. Shadowing in ParaView. The yellow-orange scale depicts temperature volume.

3.1 Phases of impact

3.1.1 Pre-collision phase and moment of collision

Prior to collision, the water remains still. When the asteroid hits the water, it causes a sudden burst of water in all directions. The range in which the water splashes was calculated and the result in shown in Figure 6. Water splashes as high as approximately 140 meters. Furthermore, a crash can cause high temperatures in the atmosphere, as seen in Figure 8. Here, it can be noted that the maximal temperature is reached on collision.

3.1.2 Post-collision phase

Peaks in the overall mean temperature and water volume are reached in the post-collision stage, as seen in Figure 7. Both means follow a bell-shaped curve pattern over time. Temperature reaches a maximum mean of 0.03 eV one second after the moment of impact. Shortly after, water volume reaches a maximum mean value around 0.002. It is noteworthy that the absolute maximum temperature value was reached prior the the maximum mean value, specifically at the moment of collision. This can also be seen in the interactive graph on the GitHub-results page which depicts the maximum temperature over time.

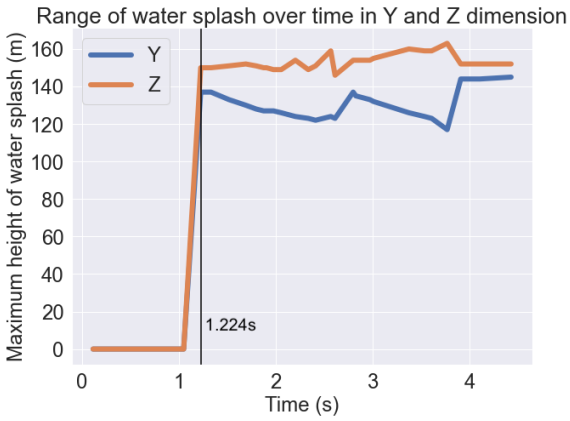

Figure 6. Splash range in meter in Y and Z dimension over time. At 1.224 s., water splash increases drastically in both Z and Y direction.

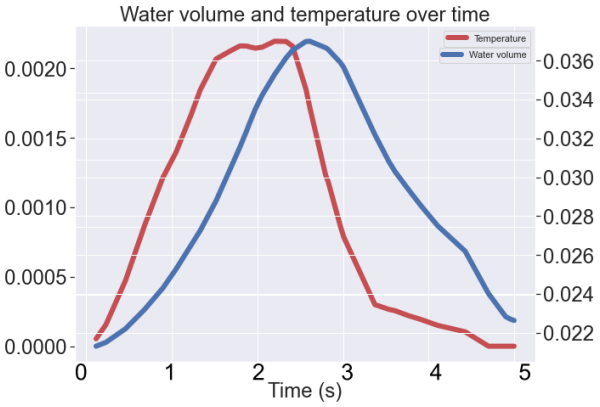

Figure 7. Mean water volume (blue) and temperature (red) over time.

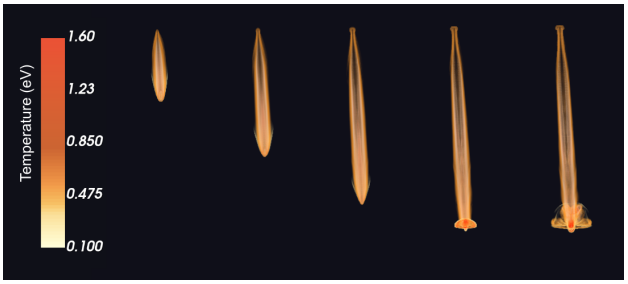

Figure 8. Temperature in atmosphere. The first three snapshots depict the pre-collision stage. The latter two show temperature volume during collision.

At last, the water evaporates into the air as well as the asteroid. This is shown by Figure 9, where water volume has a purple color and the temperature is coloured orange. It can be seen that a large amount of water has evaporated into the air and

the asteroid has disappeared. The temperature and water volume decrease back to zero, as seen in Figure 7.

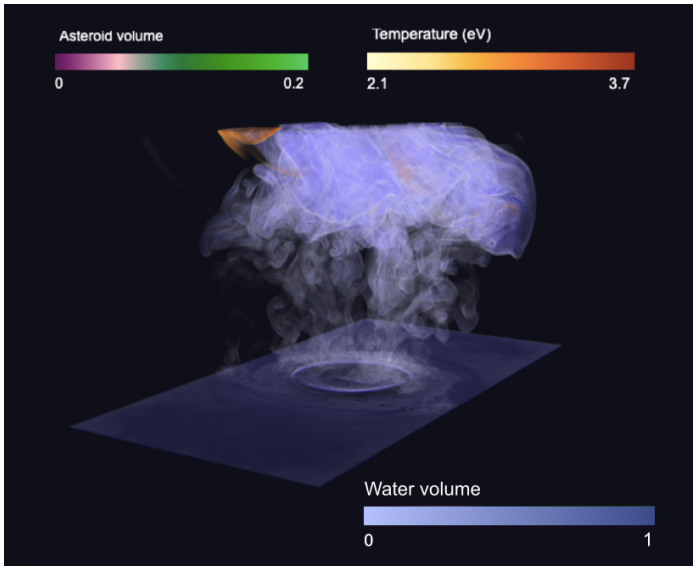

Figure 9. Volume rendering of water, asteroid and temperature. Final steps of the simulation (t = 4.8s.)

3.1.3 Animations

The majority of the aforementioned results can also be found in the animations. Furthermore, the animations show that the maximum temperature is found at the center location just where the asteroid hits the surface. In addition, the maximum temperature coincides with a peak in pressure value.

The animations give a clear view on the shapes of the rendered volumes. This allows for a more detailed inspection of the variables.

4. Discussion

This report analyses the overall impact of a non-airbursting asteroid crashing in a deep water ocean. Visualization tools ParaView (in combination with VTK) and Blender were used to visualize the data and assess the impact on several variables such as water splash and atmosphere temperature. Analysis was mainly focused on volumetric rendering.

The main challenge of this project was the large ensemble dataset. Due to limited computing resources, the amount of data was reduced to a smaller set of timesteps. More frames per second are required in the animations for a full visualization. However, the subset of timesteps was considered sufficient for the scope of the project. To allow for fast rendering of the data for exploration purposes, parallel rendering of images in VTK was utilised.

The purpose of using various colours is to allow for displaying multiple variables in one visualization. This was done in the most neutral manner possible. Colour maps which do not overlap were chosen to avoid confusion. Intuitively, temperature volume was displayed in red.

To summarize the obtained results, it was observed that the asteroid collision leads to the splashing of water in all directions, up to 140 m high. Considering the atmosphere temperature, the maximum is reached on the moment of impact, whereas the maximum mean value is reached a second after the impact. The water surface is also majorly affected by the high temperature of the asteroid. Provided the visualizations, water volume seems to increase due to evaporation of the water into the air.

Considering the use of either Blender or ParaView for volume rendering, both visualization softwares have advantages and disadvantages. Blender enabled high quality shading and glass coloring, creating a sense of depth, placement and direction. However, this had to be done with contour geometry data, which had the disadvantage of not being able to map a color scheme to scalar values. On the contrary, adding color schemes to scalar values was easy to implement in ParaView, which is shown in Figure 5. For this specific case, it can be said that using ParaView had an advantage when the main purpose is to map different scalar values to color schemes, and that Blender had an advantage when the main purpose is to create shadowing and depth.

Further research could focus on implementing volumetric rendering in virtual- and augmented reality. The beginning of this idea was already implemented by Imahorn et al. [7]. This could make it possible to interact with the data, making it easier to understand the data.

Overall, Blender and ParaView can be used effectively to show the impact of an asteroid crash in a deep water ocean. Visualizing its impact is important to understand the consequences and the possible emergence of a tsunami.

References

[1] Steven N Ward and Erik Asphaug. Asteroid impact tsunami: a probabilistic hazard assessment. *Icarus*, 145(1):64–78, 2000.

[2] John Patchett, F Samsel, K Tsai, Galen R Gisler, David H Rogers, Greg D Abram, and Terece L Turton. Visualization and analysis of threats from asteroid ocean impacts. *Los Alamos National Laboratory*, 2016.

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[4] Blender documentation. `https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/glossy.html`. Accessed: 2020-11-01.

[5] Charles D Hansen and Chris R Johnson. *Visualization handbook*. Elsevier, 2011.

[6] Cvlib-d. `https://github.com/cinemascience/cvlibd`. Accessed: 2020-11-01.

[7] Raphael Imahorn, Irene Baeza Rojo, and Tobias Günther. Visualization and analysis of deep water asteroid impacts. In *2018 IEEE Scientific Visualization Conference (SciVis)*, pages 85–96. IEEE, 2018.