1.7 Assessing Tsunami Vulnerabilities of Geographies with Shallow Water Equations

Assessing Tsunami Vulnerabilities of Geographies with Shallow Water Equations

Rifat Aras & Yuzhong Shen
Department of Modeling, Simulation, and Visualization Engineering
Old Dominion University
raras001@odu.edu, yshen@odu.edu

Tsunami preparedness is crucial for saving human lives in case of disasters that involve massive water movement. In this work, we develop a framework for visual assessment of tsunami preparedness of geographies. Shallow water equations (also called Saint Venant equations) are a set of hyperbolic partial differential equations that are derived by depth-integrating the Navier-Stokes equations and provide a great abstraction of water masses that have lower depths compared to their free surface area. Our specific contribution in this study is to use Microsoft's XNA Game Studio to import underwater and shore line geographies, create different tsunami scenarios, and visualize the propagation of the waves and their impact on the shore line geography. Most importantly, we utilized the computational power of graphical processing units (GPUs) as HLSL based shader files and delegated all of the heavy computations to the GPU. Finally, we also conducted a validation study, in which we have tested our model against a controlled shallow water experiment. We believe that such a framework with an easy to use interface that is based on readily available software libraries, which are widely available and easily distributable, would encourage not only researchers, but also educators to showcase ideas.

1.0 INTRODUCTION

Water inundation preparedness is an important component of early response systems for disasters like tsunamis, flooding, and water inundation. Realistic inundation models play an important role in many aspects like finding coastal hazard risks, assessing vulnerable zones, and taking necessary precautions to mitigate damage from inundation.

Shallow water equations are a set of hyperbolic partial differential equations (PDEs) that were introduced 140 years ago [1]. In cases where the horizontal length of the simulation domain is much greater than the vertical length, these equations are obtained by depth integrating the infamous Navier-Stokes equations and widely used to model flows in rivers and coastal areas.

There are many strategies adopted for solving hyperbolic PDEs of shallow water systems, however our particular problem requires a solution to be (1) well-balanced for lake-at-rest problems, in which stationary steady states are preserved, (2) positivity preserving to handle dry and shore areas, and (3) able to support linear friction models. Therefore in this work, we employed a second-order well-balanced positivity preserving central-upwind scheme for solving the Saint-Venant system [2].

In order for such a model to be useful as a decision support mechanism, it has to comply with certain performance requirements. Faster-than-real time simulation of the model enables decision makers to observe the simulation, try some number of precautionary measures, and choose the optimum one. Unfortunately, it becomes impossible to simulate a sufficiently high resolution model in real time by using only the computational power of a CPU. General purpose computation on graphics processing units (GPGPU) can be described as a paradigm of utilizing high-performance many-core graphics processing units (GPUs) for computation tasks that are normally handled by CPUs. With the transition from fixed to programmable graphics pipeline, software developers gained the ability to use multiple computational cores on a GPU for non-graphics data without the explicit need of managing parallel computation elements such as threads, shared memory, and message passing interfaces [3].

In this work, our main contribution is implementing a system that uses the bathymetry, elevation, and water depth data of a geography to create the water inundation model. Our proposed system utilizes XNA Game Studio 4.0 [4] to create an interactive experimentation environment to the user. We also harnessed the computational power of GPU in every step of the simulation to be able to run the simulation faster-than-real time to allow decision makers to experiment with certain scenarios and possible outcomes at a fast pace. We have used shaders written in High Level Shader Language (HLSL) in order to delegate heavy computations to the GPU, thus taking the entire load off the CPU that can be used for other tasks.

The remainder of the paper is organized as follows. Section 2 describes the shallow water equations that form the basis of this work and explains the implementation details of the algorithm. Section 3 mentions about the validation methodology of the model and gives details about the simulation experiment we conducted using the presented model. Finally, Section 4 concludes the paper and discusses future work.

2.0 METHODOLOGY

2.1 Shallow Water Equations

The shallow water equations consist of a set of hyperbolic PDEs and can be used to model physical phenomena such as tsunamis, tidal waves, flooding, and other types of water inundation. In vector form, shallow water equations in two dimensions can be written as in Eq. (1).

$$\begin{bmatrix} w \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ \frac{(hu)^2}{w-B} + \frac{g(w-B)^2}{2} \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ \frac{(hv)^2}{(w-B)} + \frac{g(w-B)^2}{2} \end{bmatrix}_y = \begin{bmatrix} 0 \\ -g(w-B)B_x \\ -g(w-B)B_y \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{-\alpha hu}{1+\beta(w-B)} \\ \frac{-\alpha hv}{1+\beta(w-B)} \end{bmatrix} \quad (1)$$

The dependent variables in these equations are w (representing surface elevation) and hu and hv (representing horizontal and vertical discharges respectively). B represents the bottom elevation (Figure 1). In a more compact form the equation can be written as

$$Q_t = -F(Q)_x - G(Q)_y + S_b(Q,B) + S_f(Q,B), \quad (2)$$

where Q is the vector of dependent variables, F and G vectors represent fluxes along horizontal and vertical directions, and S_b and S_f represent source terms from the bathymetry slope and friction respectively.

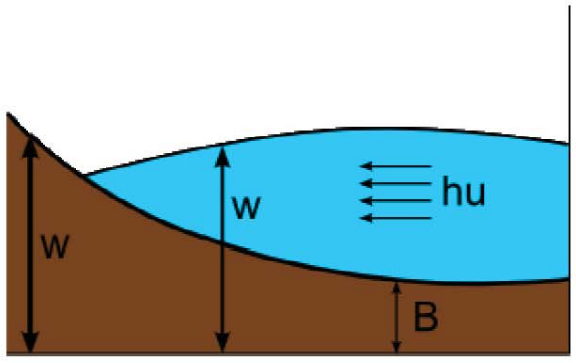


Figure 1. The visual representation of the dependent variables and the bottom topography for the 1D case.

2.2 Central-Upwind Scheme

In order to solve the presented equations, we need a well-balanced positivity preserving scheme. The central-upwind scheme proposed by Kurganov and Petrova [2] is a good candidate, because it is capable of preserving steady states (lake at rest) and also it guarantees the positivity of the fluid depth (near shore areas).

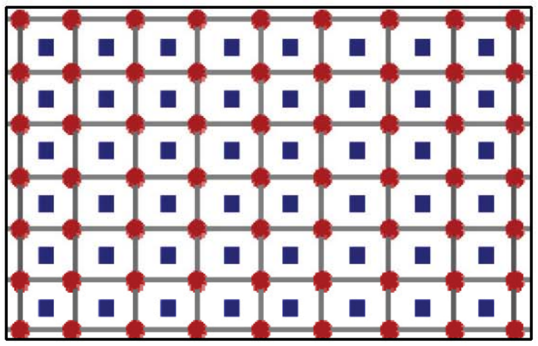


Figure 2. The staggered grid used to store dependent variables (blue squares) and the bottom topography values (red dots).

In this scheme, the dependent variables w, hu, hv, and the bathymetry data B are stored in a staggered grid (Figure 2). The bottom topography is represented as a continuous piecewise bilinear approximation. Likewise, the dependent variables at integration points are also reconstructed from non-oscillatory bilinear approximation obtained from cell averages (Figure 3).

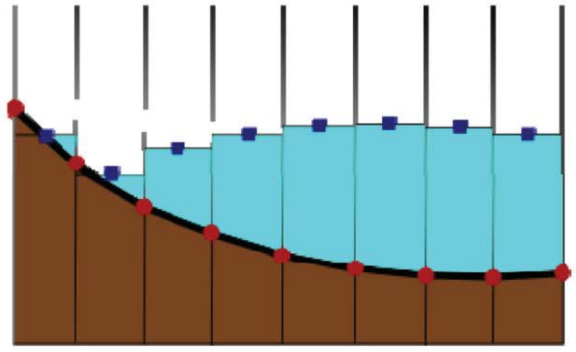


Figure 3. The slopes of the dependent variables are constructed from the cell averages.

To prevent spurious oscillations that would cause instability issues in the simulation, the slopes of the dependent variables have to be reconstructed carefully. Flux limiters, also known as slope limiters, are mechanisms that are used to prevent oscillations, thus making the system solution total variation diminishing. In this work, we used a generalized minmod flux limiter that was originally used by Kurganov and Petrova [2]. The slopes of the dependent variables in one dimension are therefore found by the function,

$$Q_x = MINMOD(\theta f, c, \theta b), \qquad (3)$$

where f, c, and b are forward $(\frac{Q^{i+1}-Q^{i}}{\Delta x})$, central $(\frac{Q^{i+1}-Q^{i-1}}{2\Delta x})$, and backward $(\frac{Q^{i}-Q^{i-1}}{\Delta x})$ slope approximations and θ is a parameter that controls how dissipative is the limiter (Figure 4). The $MINMOD$ function is defined as

$$MINMOD(a,b,c) = \begin{cases} \min(a,b,c), \{a,b,c\} > 0 \\ \max(a,b,c), \{a,b,c\} < 0 \\ 0. \end{cases} \qquad (4)$$

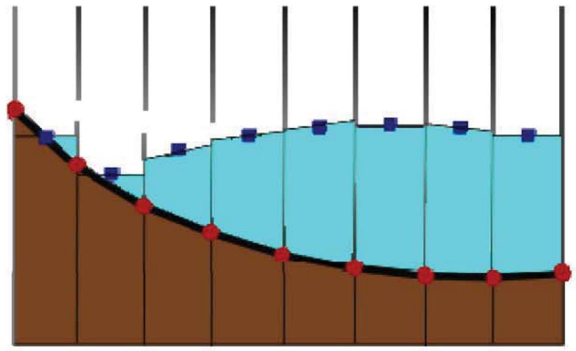


Figure 4. Slopes of the dependent variables are constructed by using a flux limiter.

After slopes are constructed for the dependent variables, one obvious problem with this construction is that we can obtain negative values for the depth component (Figure 4). As the eigen value of the Jacobian of the shallow water equations being

$$u \pm \sqrt{gh}, \qquad (5)$$

having negative values for the depth component h will break down the simulation. In order to prevent this from happening, we need to correct the water elevation slopes of the cells that have negative values for the depth component. This is simply performed by moving the problematic negative value up to the topography and recalculating the slope of the cell by using the modified value and the original average value (Figure 5).

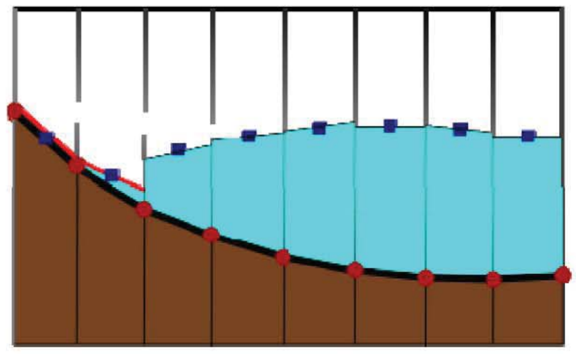


Figure 5. The slopes that result in negative depth values are corrected (red slopes).

After slopes are corrected, the flux values for the cell interfaces are calculated according to the central-upwind scheme. For the cell interface between cells *i* and *i+1*, the two neighboring values are obtained by using the corresponding slopes, and these values are used to compute the flux function (Figure 6) (see [2] for details).

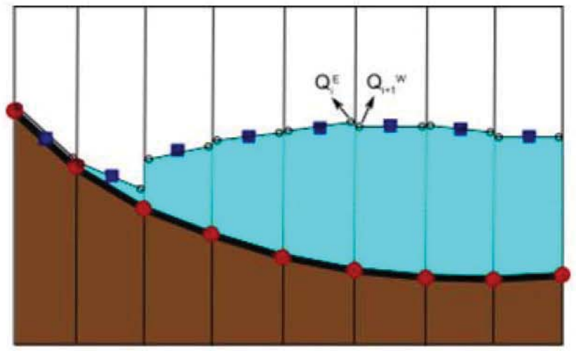


Figure 6. For each cell interface, the two neighboring values are obtained from the slopes to compute the flux value.

Finally, the computed flux values and source terms are plugged into Eq. (2) and time discretization is performed by using the following second order Runge-Kutta ODE solver,

$$Q_{ij}^{*} = Q_{ij}^{n} + \Delta t R(Q^{n})_{ij}$$
$$Q_{ij}^{n+1} = \frac{1}{2} Q_{ij}^{n} + \frac{1}{2}\left[Q_{ij}^{*} + \Delta t R(Q^{*})_{ij}\right], \qquad (6)$$

where $R(Q)$ is the right hand side of the Eq. (2).

2.3 Algorithmic Break Down and Implementation

The central-upwind scheme presented in the previous section can be realized as an algorithm that consists of several passes. A single step of the simulation is broken down into two Runge-Kutta substeps. In each of these substeps, the slopes of the dependent variables are reconstructed, flux computations are performed, and the time integration is computed.

The dependent variables and intermediate values are stored on graphical processing unit as buffers. In XNA programming, these buffers are represented as `RenderTarget2D` objects.

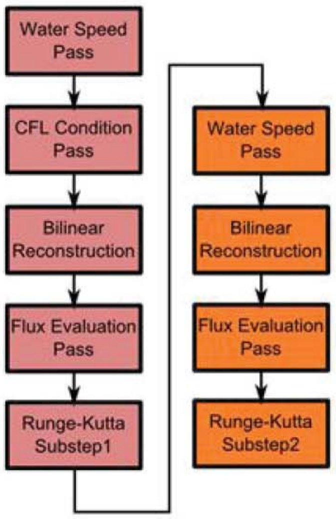


Figure 7. The flow chart of the algorithm.

2.3.1 Water Speed Pass

Water speeds in horizontal and vertical directions are obtained by dividing the dependent variables hu and hv by the water height. In shallow water simulations, fluid speeds are required to determine how large simulation time steps can be and to compute numerical flux values. As we obtain speed values by $\frac{hu}{h}$ and $\frac{hv}{h}$, one apparent problem is having very small height values. Due to floating point precision issues, when we have very small height values, we naturally end up with very high and erroneous speed quantities. This is definitely a problem that would cause very small simulation time steps and can be corrected by desingularizing the computation of speed values for small height values. In this work, we adopted the approach of Kurganov and Petrova [2] and used the following formula to compute speed values at shoal zones:

$$u^* = \frac{\sqrt{2}h(hu)}{\sqrt{h^4 + \max(h^4, \varepsilon)}}, \qquad (7)$$

where ε is a small number chosen according to the machine floating point precision. After correcting the speeds for small heights, we need to update the corresponding hu and hv values by $h * u^*$ and $h * v^*$ in order to keep the consistency of the simulation variables.

2.3.2 CFL Condition Pass

The Courant-Friedrichs-Lewy (CFL) condition [5] is a necessary condition for convergence when solving hyperbolic PDEs. For shallow water equations, this condition translates to the condition of simulation time step having to be small enough so that a wave does not pass a whole grid cell during that time step.

In order to find the global fluid velocity value for the entire simulation domain, we need a reduction operation for the water speed values that were obtained from the previous pass. Given a buffer that contains the speed values and whose size is $n \times n$, we can implement the reduction operation as a progressive down-scale operation, in which the new value of the pixel is set to the maximum of the set of pixels that this pixel is sampled from (Figure 8).

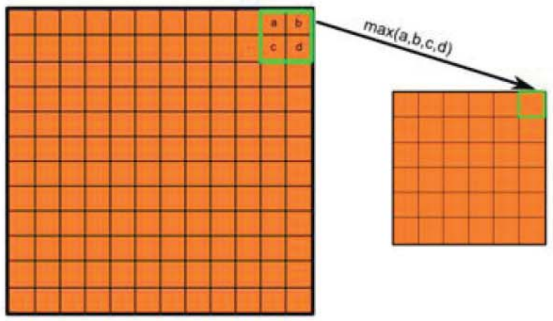
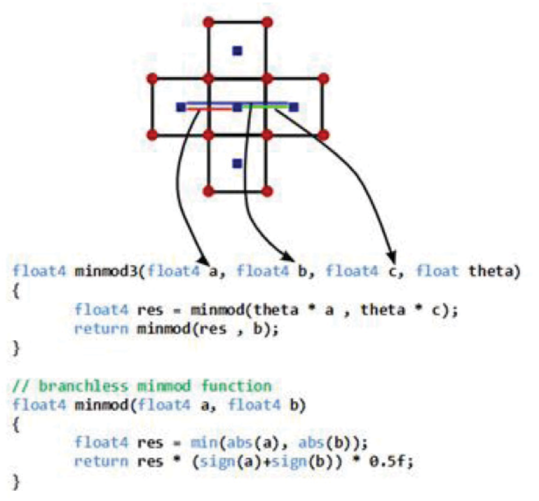


Figure 8. The down-scale operation is performed progressively until we obtain a resultant buffer that is small enough to be processed by the CPU.

2.3.3 Bilinear Reconstruction Pass

The vertical and horizontal slopes of the dependent variables w, hu, and hv are reconstructed by using a branchless implementation of the generalized minmod flux limiter [6] (Figure 9).



```
float4 minmod3(float4 a, float4 b, float4 c, float theta)
{
        float4 res = minmod(theta * a , theta * c);
        return minmod(res , b);
}

// branchless minmod function
float4 minmod(float4 a, float4 b)
{
        float4 res = min(abs(a), abs(b));
        return res * (sign(a)+sign(b)) * 0.5f;
}
```

Figure 9. Implementation of the MINMOD limiter in HLSL.

As stated before in section 2.2, special care should be taken after the slopes are reconstructed as these slopes may end up with negative depth values. To prevent this from happening, the midpoints at the cell interfaces are tested against the bathymetry information, if the slope is found to be causing negative depth value, it is corrected.

2.3.4 <u>Flux Evaluation Pass</u>

After the slopes are determined for the grid cells, these are used to approximate the values of the dependent variables at cell interfaces. According to the central-upwind scheme, we need two values for each cell interface that are approximated from the slopes of the two neighboring cells that share the interface.

The horizontal (F) and vertical (G) flux values are computed separately according to

$$F(Q) = \left(hu, \frac{(hu)^2}{w-B} + \frac{1}{2}g(w-B)^2, \frac{(hu)(hv)}{w-B}\right)^T$$
$$G(Q) = \left(hv, \frac{(hu)(hv)}{w-B}, \frac{(hv)^2}{w-B} + \frac{1}{2}g(w-B)^2\right)^T \quad (8)$$

For implementation details of the central-upwind scheme, refer to the original work of Kurganov and Petrova [2].

2.3.5 <u>Runge-Kutta Substep</u>

In this pass, the flux values computed at the previous flux evaluation pass are used to advance the simulation. The source terms that are caused by bottom topography (bed slope) and friction are also computed and included in this pass. In this work we are using a second order Runge-Kutta solver, in which the simulation is advanced in two separate substeps. In the first substep, the simulation is advanced one time step to obtain the intermediate values for the dependent variables (Eq. (6)). These intermediate values are then used as inputs to the second series of passes to complete the time integration (orange shaded steps in Figure 7).

3.0 VALIDATION & EXPERIMENT

Like any other modeling and simulation study, validity of a model plays an important role in its credibility in decision making processes. In this work, we tried to reproduce the results of a real-life experiment that was conducted in a controlled environment.

In the work of Synolakis et al. [7], the authors emphasize the necessity of verification and validation of the numerical models used in emergency planning. They present a set of analytical, laboratory, and field benchmark tests that can be used to validate numerical water inundation models. In our study, we included their conical island experiment and recreated their experiment setup in our virtual laboratory as accurately as possible (Figure 10).

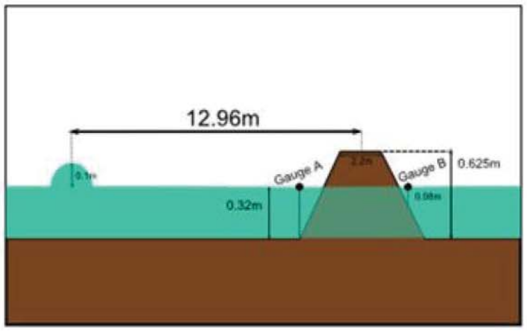


Figure 10. The recreated conical island experiment. Because of the lack of data about wave characteristic, we approximated the wave as a sine wave.

The data from the physical experiment and our virtual experiment are collected and plotted for easy comparison of the two data. Although, there was not enough data about the solitary wave other than its height, we were able to obtain very close results, capturing the essence of the physical experiment (Figure 11).

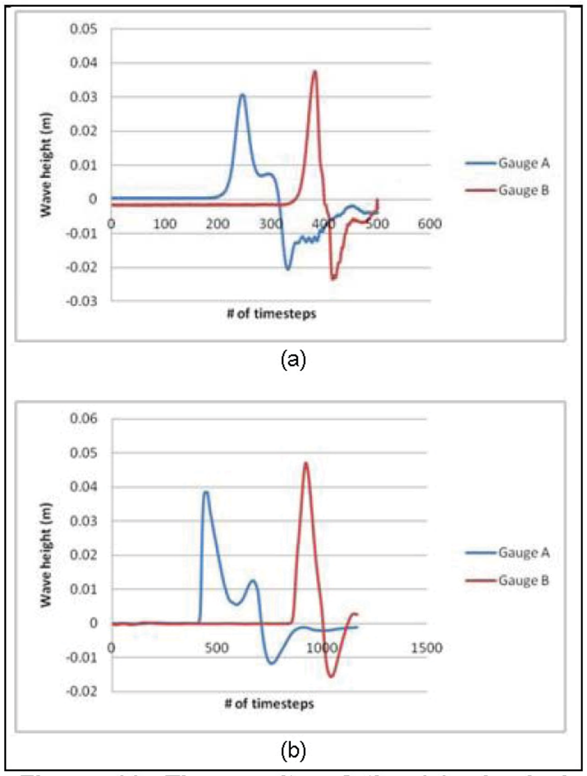


Figure 11. The results of the (a) physical experiment and (b) our simulation.

For showcasing our study, we have chosen the Virginia Beach area, particularly the area around the Virginia Beach Convention Center. The bathymetry and surface elevation data are obtained from National Oceanic and Atmospheric Administration's (NOAA) National Geophysical Data Center[8]. The data, which has a cell size of 10 meters, for the whole Virginia Beach region was preprocessed and converted to comma separated value (CSV) format to be fed to our simulation (Figure 12 and Figure 13).

4.0 CONCLUSION

Recent disasters show that water inundation readiness and emergency planning processes are vital necessities to prevent aftermath. Credible water inundation models are the important elements for emergency planning processes. Decision makers need valid, interactive, and fast models in order to respond to situations in a timely fashion.

In this work, we presented a water inundation model that is based on shallow water equations. We used a well-balanced positivity preserving scheme to solve the set of hyperbolic PDEs. By harnessing the computation power of GPUs, we were able to run a high resolution simulation faster-than-real time on a consumer grade laptop PC. To establish the credibility of the presented model, we recreated a controlled experiment and compared the physical results to our virtual results.

The current model supports static bathymetry / elevation data that is organized as a regular grid. One feature we want to add to our application is the ability to alter the terrain before running the simulation. This way, the decision makers would be able to see the possible effects of certain actions such as adding barriers to parts of the terrain.

5.0 REFERENCES

[1] A. J. C. d. Saint-Venant, "Th´eorie du mouvement non-permanent des eaux, avec application aux crues

des rivi`ere at `a l'introduction des war´ees dans leur lit," *C. R. Acad. Sci. Paris,* pp. 147-154, 1871.

[2] A. Kurganov and G. Petrova, "A Second-Order Well-Balanced Positivity Preserving Central-Upwind Scheme for the Saint-Venant System," *Communications in Mathematical Sciences,* vol. 5, pp. 133-160, 2007.

[3] NVIDIA, "NVIDIA CUDA C Programming Guide," 2010.

[4] Microsoft. (06.28.2011). *XNA Game Studio 4.0*. Available: http://msdn.microsoft.com/en-us/library/bb200104.aspx

[5] R. Courant, K. Friedrichs, and H. Lewy, "On the Partial Difference Equations of Mathematical Physics," *IBM Journal of Research and Development,* vol. 11, pp. 215-234, 1967.

[6] T. Hagen, M. Henriksen, J. Hjelmervik, and K. A. Lie, "How to solve systems of conservation laws numerically using the graphics processor as a high-performance computational engine," *Geometric Modelling, Numerical Simulation, and Optimization,* pp. 211-264, 2007.

[7] C. Synolakis, E. Bernard, V. Titov, U. Kâno lu, and F. González, "Validation and verification of tsunami numerical models," *Tsunami Science Four Years after the 2004 Indian Ocean Tsunami,* pp. 2197-2228, 2009.

[8] L.A.Taylor, B. W. Eakins, K. S. Carignan, R. R. Warnken, T. Sazonova, D. C. Schoolcraft, and G. F. Sharman, "DIGITAL ELEVATION MODEL OF VIRGINIA BEACH, VIRGINIA : PROCEDURES, DATA SOURCES AND ANALYSIS," U. S. D. o. Commerce, Ed., ed. Boulder: National Geophysical Data Center Marine Geology and Geophysics Division, 2008.

6.0 APPENDIX

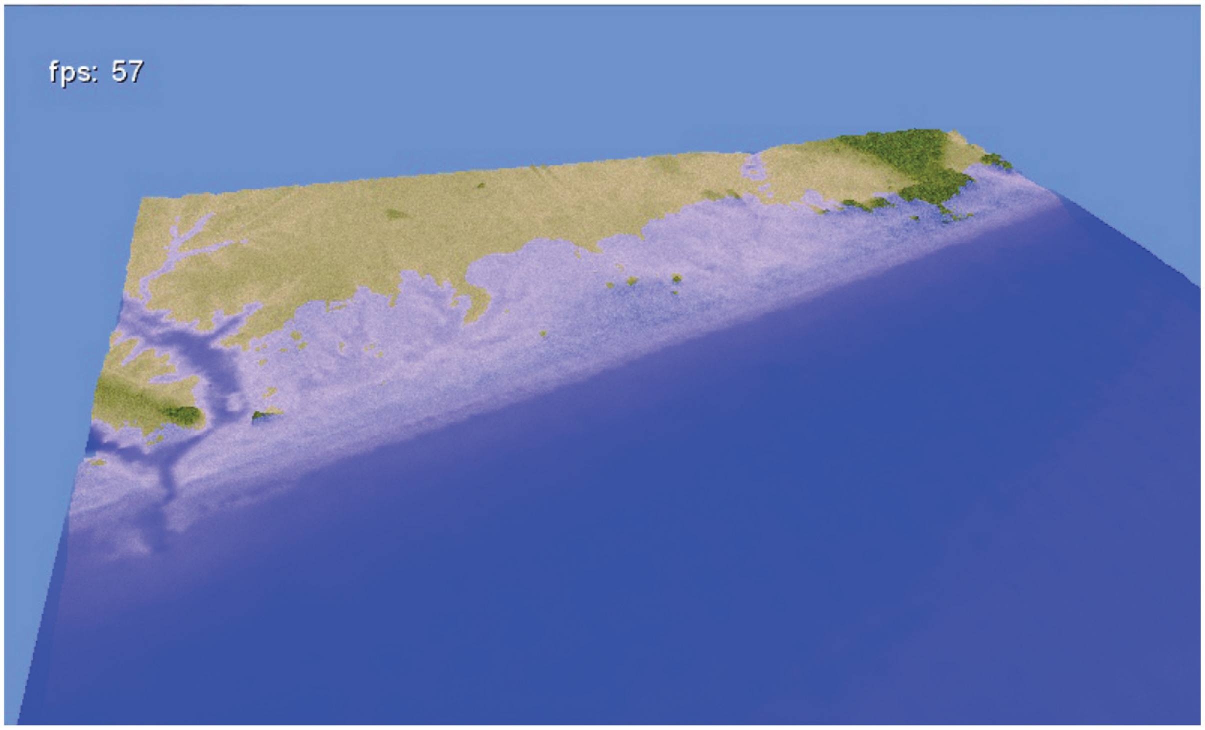


Figure 12. Visualization of the Virginia Beach area simulation as waves hit the shore line.

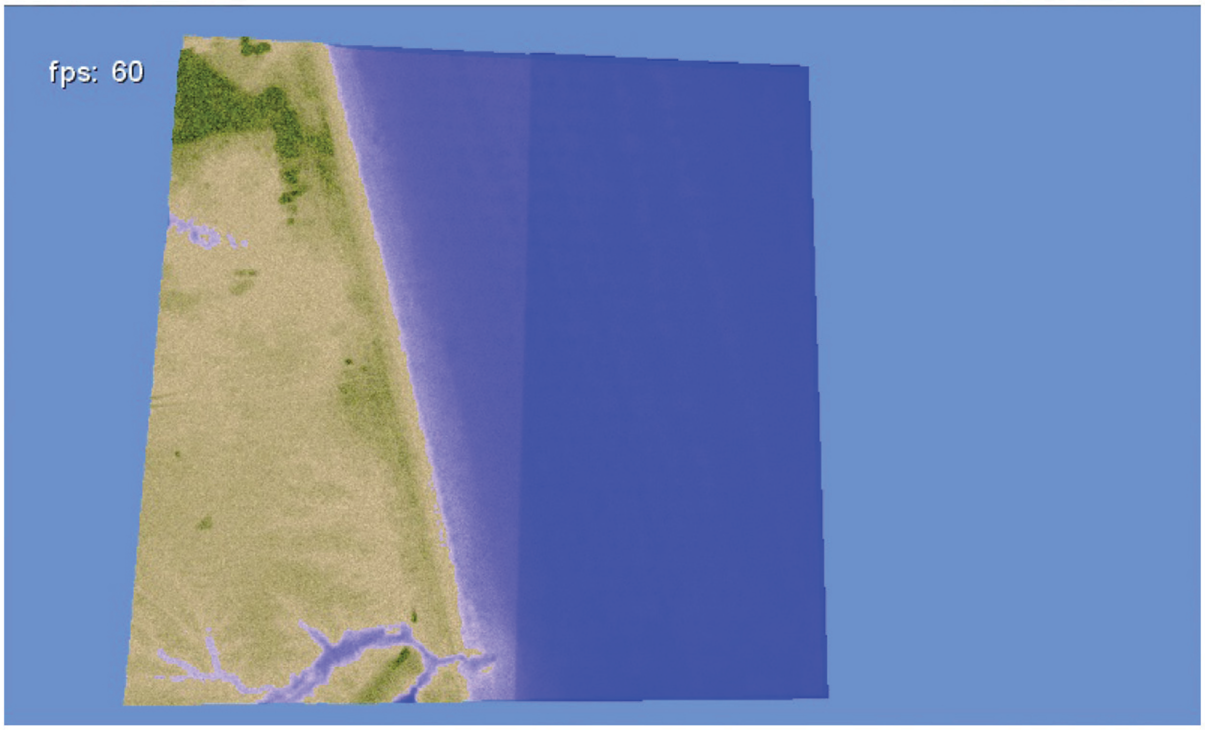


Figure 13. Visualization of Virginia Beach area simulation as waves approach the shore line.

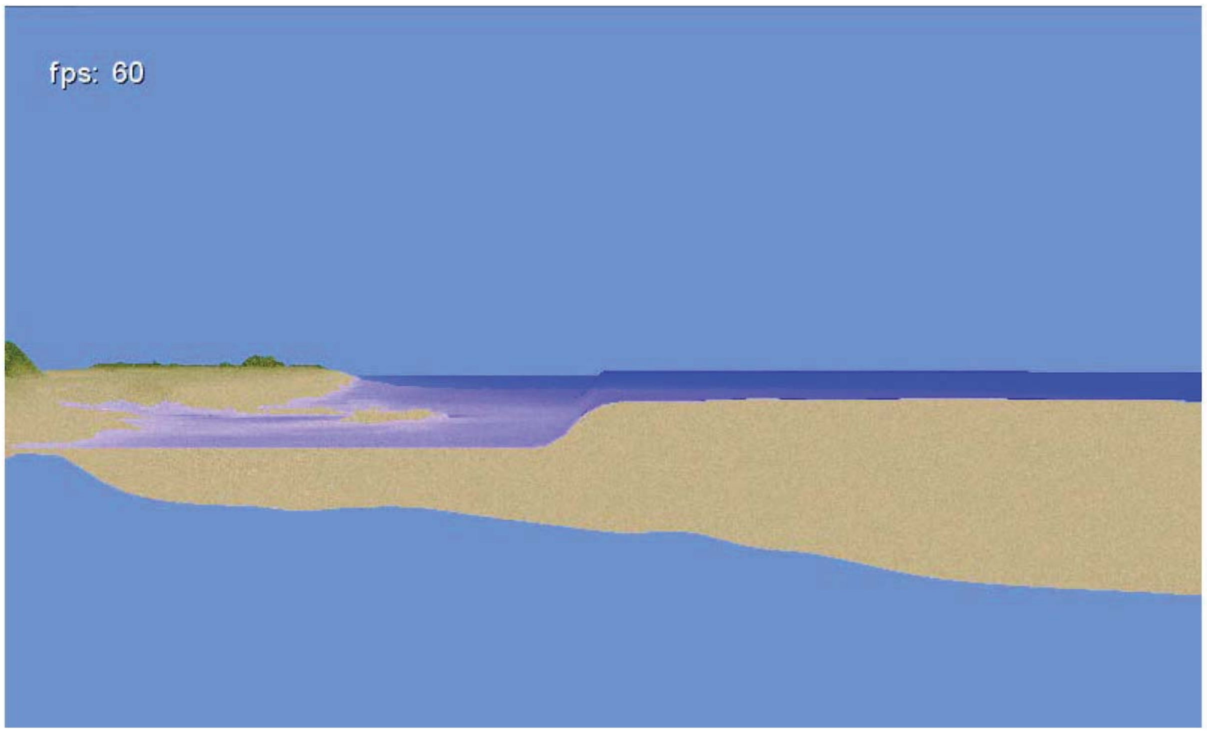


Figure 14: The side view of the wave that approaches Virginia Beach shore line.