



HAL
open science

Integrating tsunami simulations in web applications using BROWNI, an open source client side GPU powered tsunami simulation library

José Galaz, Rodrigo Cienfuegos, Alejandro Echeverria, Sebastian Pereira,
Celeste Bertin, Grazia Prato, Juan-Cristobal Karich

► **To cite this version:**

José Galaz, Rodrigo Cienfuegos, Alejandro Echeverria, Sebastian Pereira, Celeste Bertin, et al.. Integrating tsunami simulations in web applications using BROWNI, an open source client side GPU powered tsunami simulation library. *Computers and Geosciences*, 2021, 159, pp.104976. 10.1016/j.cageo.2021.104976 . hal-02112763v2

HAL Id: hal-02112763

<https://inria.hal.science/hal-02112763v2>

Submitted on 17 Mar 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 Highlights

2 **Integrating tsunami simulations in web applications using BROWNI, an open source client-** 3 **side GPU-powered tsunami simulation library**

4 J. Galaz,R. Cienfuegos,A. Echeverria,S. Pereira,C. Bertin,G. Prato,J. C. Karich

- 5 • BROWNI is a WebGL - Javascript tsunami simulation library
- 6 • Simulations run on web browsers “out of the box” making users more proficient
- 7 • Comparison with measurements and other simulators verify its accuracy
- 8 • Scientific outreach and risk communication software made with BROWNI are described
- 9 • Software and hardware limitations are discussed

Integrating tsunami simulations in web applications using BROWNI, an open source client-side GPU-powered tsunami simulation library,^{***}

J. Galaz^{a,b,*}, R. Cienfuegos^{a,c,d}, A. Echeverria^e, S. Pereira^b, C. Bertin^{b,1}, G. Prato^b and J. C. Karich^c

^aDepartamento de Ingeniería Hidráulica y Ambiental, Escuela de Ingeniería, Pontificia Universidad Católica de Chile. Vicuña Mackenna 4860, Santiago, Chile.

^bInria Chile, Avenida Apoquindo 2827 piso 12, Las Condes, Santiago, Chile.

^cCentro de Investigación para la Gestión Integrada del Riesgo de Desastres (CIGIDEN), CONICYT/FONDAP/1511007, Santiago, Chile.

^dMarine Energy Research and Innovation Center (MERIC). Av. Apoquindo 2827 piso 12, Las Condes, Santiago, Chile.

^eUniversidad de los Andes, Monseñor Álvaro del Portillo 12455, Las Condes, Santiago, Chile.

ARTICLE INFO

Keywords:

Tsunami
Simulation library
Javascript
GPU
Web
Visualization

ABSTRACT


Tsunami simulation software is a key component of state-of-the-art early warning systems but the inherent complexities in phases of installation, execution, pre and post-processing prevent their use in other areas of risk management such as communication and education. Recent advances in software and computational capacities such as the efficiency of GPU computing and the ubiquity of web browsers bring new opportunities to bridge the gap between expert and non-expert users. Here we present a Javascript library to enable a web browser to facilitate gathering and analyzing data from tsunami simulations, by means of interactive and efficient visualizations. At its core, the library uses WebGL, the browser's standard 3D graphics API, to run GPU accelerated computations of a tsunami model. A far-field tsunami model is implemented (linear shallow water equations discretized on spherical coordinates), and its implementation is validated against real tsunami observations, and benchmarked with two other tsunami software-packages. Two software platforms that use this library are presented to illustrate the powerful applications that can be developed for risk communication and education. These applications are characterized by their interactivity and fast computations, which enable users to focus on the understanding of the phenomena of tsunami propagation and iterate quickly to assess different scenarios and potential implications to tsunami risk management. Some limitations on this approach are discussed, in aspects such as scalability, performance, multi-threading and batch-processing, that can be relevant for other users. In our experience, the before mentioned benefits very well compensate the discussed limitations for this kind of applications. The library has an open source license, and is meant to be imported without modifying its source code to facilitate the creation of new applications as the ones herein presented.


1. Introduction

Tsunamis have become one of the deadliest natural hazards in the world, causing more than 200,000 casualties and billions of dollars in economic losses in the last 20 years [1], which largely justifies the important efforts in increasing tsunami preparedness and awareness that different governments have undertaken over the last decades. Tsunami early warning systems, such as those from Japan, Chile, and the USA [2, 3, 4], are examples of these efforts which have

* Source code located at <https://github.com/jgalazm/browni> and <https://github.com/jgalazm/comp-geo-paper-figures>

** **Authorship statement:** **J.G.:** Conceptualization, methodology, software (BrownI and TsunamiLab), validation, formal analysis, investigation, data curation, visualization, project administration. **R.C.:** Conceptualization, methodology, validation, resources, supervision. **A.E.:** Conceptualization, methodology, supervision. **S.P.:** Conceptualization, software, TsunamiLab web and pool. **C.B.:** Software, TsunamiLab pool. **G.P.:** Software, design of TsunamiLab web application. **J.C.K.:** Software, design of TsunamiLab pool

 jgalazm@gmail.com (J. Galaz); racienfu@ing.puc.cl (R. Cienfuegos); alejandroechev@gmail.com (A. Echeverria); sebastian.pereira@inria.cl (S. Pereira); celbertin@gmail.com (C. Bertin); grazia.pratop@gmail.com (G. Prato); crisobalkarich@gmail.com (J.C. Karich)

 [jgalazm.github.io](https://github.com/jgalazm) (J. Galaz)

ORCID(s):

¹Current address: Equifax, Isidora Goyenechea 2800, Las Condes, Santiago, Chile

52 enabled communities to react more quickly and in turn prevent casualties. These systems are based on state of the art
53 scientific knowledge and include numerical modeling as an essential tool for performing wave propagation forecast
54 [1].

55 Thanks to the recent advances in computational modeling, it is now possible to simulate complex scenarios with
56 increased resolution (e. g., GeoClaw [5], [6], COMCOT [7], CouWave [8], NeoWave [9], EasyWave [10], ANUGA
57 [11], and MOST [3]). However, gathering and analyzing data iteratively using numerical modeling is still costly owing
58 to expensive hardware requirements, complex installation processes and heavy tasks of data pre- and post processing.
59 The latter has so far limited the development of innovative applications for disaster risk management that could integrate
60 numerical simulations with geographical information systems (GIS) and visualization tools [12].

61 Indeed, new visualization and interactivity options have not been fully exploited yet, even if some interesting efforts
62 to communicate tsunami risks to emergency managers and broad audiences have been made taking advantage of such
63 capabilities. For example, Keon et al. [13] integrated inundation estimation with agent-based modeling to develop a
64 framework that allows for interactive visualization and exploration of human response. Similarly, Hsieh et al. [14]
65 showed an application that allows for efficient computation and visualization of tsunami simulations on tiled displays,
66 taking advantage of the synchronized computing power of a GPU cluster which reaches high performance while ef-
67 ficiently rendering its results. Most remarkably, the Center for Tsunami Research (NCTR) of the National Oceanic
68 and Atmospheric Administration (NOAA) [3] fosters international collaborations providing emergency managers with
69 different tools to help assessing the hazard impact visualizing demographics, evacuation routes, infrastructure, and
70 other variables that are relevant for the decision.

71 These applications can be related to a generalized need for new geovisualizations that can help in communicating
72 hazards and risks to non-expert audiences, such as scientists from other fields, emergency managers, coastal com-
73 munities, policy makers [15], as well as children, teachers, and adults in general [16]. For example, regarding flood
74 hazards, Jacquinod et al. [17] created three-dimensional (3D) visualizations using GIS to increase awareness, and [18]
75 showcased how GIS can be useful for analyzing inundation areas by integrating it with engineering software tools such
76 as HEC RAS [19].

77 GPU computing, on the other side, has allowed users to accelerate computations at a lower cost when compared
78 to traditional central processing unit (CPU) clusters; GPU also produces complex visualizations efficiently. Recent
79 examples of these are the open source software packages TsuPy [20] and Celeris [21]. TsuPy is a script-based Python
80 library that uses the NVIDIA Compute Unified Device Architecture CUDA [22] to calculate wave propagation and
81 inundation at the regional scale with a shallow water solver, obtaining high-performance computations on NVIDIA
82 GPUs. Celeris is a Boussinesq wave model that uses Microsoft Direct3D API [23] shaders to calculate and render
83 the simulations in a 3D interactive view with a graphical user interface that allows users to change parameters on the
84 fly. As reported in [24] Celeris was recently ported to the game engine Unity3D to go even further into producing an
85 immersive virtual reality user interface. These tools demonstrate that it is possible to bridge the gap between modellers
86 (expert users) and decision makers by facilitating the tasks of software and hardware setup, configuration of scenarios,
87 data preparation, post-processing, visualization and comparison with other sources of evidence.

88 The TsuPy software requires specific vendor hardware for its execution, complex installation processes and de-
89 pendency management, and the usage of server-side computations in order to be used from an interactive interface;
90 and while Celeris can take advantage of Unity3D being multiplatform, adding new interactions and sources of data
91 requires editing its source code. However, another approach not yet explored in the literature is to take advantage of
92 web-browsers to run the computations. Among other things, applications running on web browsers benefit from an
93 almost null cost of installation and plenty of tools for developers to create graphical interfaces where users could gather
94 and analyze data from simulations and other sources more efficiently. Moreover, browsers can also benefit from GPU
95 computing to speedup calculations using the WebGL application programming interface (API) [25]. WebGL is a web
96 standard with a JavaScript API designed for efficiently rendering 2D and 3D interactive graphics, and currently, the
97 only interface that can do this without any plugins or browser extensions. This gives web browsers the advantage of
98 directly mixing simulation results with other user-interface elements, handling asynchronous user interactions and data
99 updates.

100 In this study, we examine the capabilities of GPU computing from the web browser to reduce the cost of simulations
101 and take advantage of interactivity and visualization to improve tsunami risk communication and scientific outreach.
102 We introduce a tsunami-simulation library, the Browser's Numerical Interface (BROWNI), which has an API designed
103 to facilitate the creation of interactive tsunami simulations and visualizations directly on the web browser. In section 2,
104 we give an overview of the mathematical equations and numerical algorithm, to give the relevant context to understand

105 the implementation of the library. Then we explain the software implementation of BROWNI in section 3, where we
 106 also discuss some limitations of this approach. In section 4 we validate the results of BROWNI by comparing the
 107 timeseries of wave heights against two real scenarios (Tohoku, 2011 and Chile, 2010), demonstrating its capabilities
 108 to obtain accurate results, while making efficient use of the available graphics card (dedicated or integrated). Then
 109 in section 5 we give two examples of scientific-outreach software that were built with BROWNI: TsunamiLab, a web
 110 application available from www.tsunamilab.cl, and the TsunamiLab-Pool, an interactive augmented-reality version
 111 of TsunamiLab designed for public exhibitions.

112 2. Mathematical model

113 The mathematical modeling of tsunami waves is a multi-scale problem that usually requires coupling several algo-
 114 rithms to accurately cover all the physical phenomena observed from wave generation and propagation in the ocean,
 115 to wave shoaling and run-up [26, 7, 27]. Since the focus of our work is on interactive applications for tsunami-risk
 116 management and communication, we implemented in BROWNI a far-field tsunami model that has been already used
 117 by others in operational settings [27, 10, 3, 3, 28]. The implementation of more complex regimes and algorithms is left
 118 for future work. In this section we only recall these equations and its numerical resolution without further developments
 119 or improvements.

120 2.1. Model equations

121 In the ocean, tsunami waves have a characteristic wavelength $L \approx 100$ km, a characteristic amplitude $a \approx 1$ m,
 122 and a characteristic water depth of $h_0 \approx 4$ km, which means that waves are relatively long ($h_0/L \ll 1$) and of small
 123 amplitude ($a/h_0 \ll 1$) [29]. Under these characteristics, as presented in [30], a suitable approximation is given by the
 124 linear shallow water equations in spherical coordinates:

$$\begin{aligned}
 \frac{\partial \eta}{\partial t} + \frac{1}{R \cos(\theta)} \left(\frac{\partial M}{\partial \lambda} + \frac{\partial}{\partial \theta} (N \cos \theta) \right) &= 0 \\
 \frac{\partial M}{\partial t} + \frac{gh}{R \cos \theta} \frac{\partial \eta}{\partial \lambda} &= f N \\
 \frac{\partial N}{\partial t} + \frac{gh}{R} \frac{\partial \eta}{\partial \theta} &= -f M
 \end{aligned} \tag{1}$$

125 where t is the time coordinate; λ , and θ are the longitude and latitude geographical coordinates, respectively; η , M ,
 126 and N are the wave height and longitudinal and latitudinal momentum components, respectively; $R = 6378$ km is the
 127 earth's radius; $g = 9.81$ m/s² is the earth's gravitational acceleration; $h(\lambda, \theta)$ is the bathymetry at location (λ, θ) , where
 128 $h > 0$ indicates underwater floor; and $f = 2\omega \sin(\theta)$ is the Coriolis factor with $\omega = 7.29 \times 10^{-5}$ [rad/s], which is the
 129 earth's rotation frequency.

130 2.2. Numerical discretization

131 Similar to [10], [7], and [27], equation (1) is discretized using finite differences with a second order in space and
 132 time leapfrog scheme as follows:

$$\begin{aligned}
 & \frac{\eta_{i,j}^{n+1/2} - \eta_{i,j}^{n-1/2}}{\Delta t} + \frac{1}{R \cos \theta_j} \left(\frac{M_{i+1/2,j}^n - M_{i-1/2,j}^n}{\Delta \lambda} + \right. \\
 & \left. \frac{N_{i,j+1/2}^n \cos \theta_{j+1/2} - N_{i,j-1/2}^n \cos \theta_{j-1/2}}{\Delta t} \right) = 0 \\
 & \frac{M_{i+1/2,j}^{n+1} - M_{i+1/2,j}^n}{\Delta t} + \frac{gh_{i+1/2,j}}{R \cos \theta_j} \frac{\eta_{i+1,j}^{n+1/2} - \eta_{i,j}^{n+1/2}}{\Delta \lambda} = f N' \\
 & \frac{N_{i,j+1/2}^{n+1} - N_{i,j+1/2}^n}{\Delta t} + \frac{gh_{i,j+1/2}}{R} \left(\frac{\eta_{i,j+1}^{n+1/2} - \eta_{i,j}^{n+1/2}}{\Delta \theta} \right) = -f M' \\
 & N' = \frac{1}{4} \left(N_{i+1,j+1/2}^n + N_{i+1,j-1/2}^n + N_{i,j+1/2}^n + N_{i,j-1/2}^n \right) \\
 & M' = \frac{1}{4} \left(M_{i+1/2,j+1}^n + M_{i+1/2,j}^n + M_{i-1/2,j+1}^n + M_{i-1/2,j}^n \right)
 \end{aligned} \tag{2}$$

133 where $\Delta \lambda, \Delta \theta$ define the the grid size on the longitude and latitude directions respectively; Δt is the time step; and
 134 i, j , and n indicate the values of their respective variable at time $n\Delta t$ and location $(i\Delta \lambda, j\Delta \theta)$. As shown by [31], this
 135 numerical scheme is well-suited for running in the GPU.

136 Given a domain and its bathymetry h , the input required to integrate equations (1) is contained in the initial and
 137 boundary conditions. As mentioned by [7], numerical stability is ensured by selecting $\Delta t = CFL \times \Delta s_{min} / c_{max}$, with
 138 $CFL < 1$, where CFL is the Courant-Friedrichs-Lewy number, $c_{max} = \max_{ij} \sqrt{gh_{ij}}$ and Δs_{min} is the minimum
 139 geodesic distance between two vertically or horizontally adjacent grid nodes.

140 2.2.1. Initial conditions

141 Although arbitrary initial conditions can be provided through h, M , and N , there are formulas that characterize the
 142 generation of the tsunami depending on a set of parameters. Two of these are implemented in BROWNI: earthquakes
 143 and asteroids. For earthquake generation, we use a finite fault model, wherein the water is assumed to be at rest and a
 144 perturbation is added around the epicenter according to the formulas of [32]. These formulas are an explicit solution
 145 to a linear elasticity problem that assumes that the area affected by the earthquake is rectangular and that it depends
 146 on parameters such as its length, width, hypocenter, fault slip, and the 3D orientation of the fault plane. Complex
 147 earthquakes can then be represented by superposing several of these rectangles with different parameters depending
 148 on the heterogeneity of the earthquake being modeled.

149 For asteroid-generated tsunamis, [33] described several formulas that approximate the deformation of the water due
 150 to the impact of the asteroid depending on how its energy is converted into water waves. In BROWNI, one such formula
 151 is implemented. It also assumes that the initial surface has radial symmetry and can be described by a paraboloid of
 152 positive curvature whose size depends on the size, mass, density, and speed of the impactor, as well as the amount of
 153 energy that is transmitted to the water from the asteroid.

154 2.2.2. Boundary conditions

155 Three types of boundaries are considered: periodic, open, and closed. Periodic boundaries repeat the information
 156 of the opposite boundary and are used for the E-W borders whenever the domain covers a full circle of latitude. Open
 157 boundaries allow waves to leave the domain without reflections and are used whenever the domain does not cover a
 158 latitudinal circle. This boundary condition considers the trajectories of the characteristics of the Riemann invariants
 159 of the wave equations to extrapolate values in time, as explained by [27]. Closed boundaries are used to simulate
 160 a reflecting wall by imposing $\eta = M = N = 0$; this is used to model a continental shoreline as a closed internal
 161 boundary, which is a reasonable assumption since, at a large scale, run-up displacements are negligible [27].

162 3. Software implementation

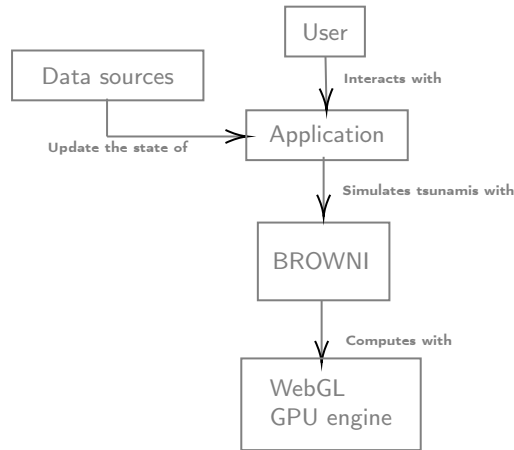


Figure 1: BROWNI architecture overview for a standard use case

3.1. Overview

163
164
165
166
167
168
169
170
171
172

BROWNI is designed to be integrated in web applications directly in the browser, as shown in Figure 1. First a user, such as an emergency manager or any person interested in visualizing and configuring tsunami simulations, interacts with a web application through a graphical user interface, on which relevant information is displayed, possibly along with interaction controls such as buttons, menus, and so on. Then, this application uses BROWNI to simulate tsunamis with the user configuration and application data, which is used to run the GPU instructions in the WebGL engine that runs the simulation efficiently. At the same time, state updates may come from other data sources, possibly producing changes in the simulator. What is important is that BROWNI is agnostic of the web-application’s purpose or structure, as also of when user-driven or data-driven changes to the simulation are applied; these occur asynchronously, i.e., independently of the main flow of the simulation, and without requiring the user to leave the web browser.

3.2. BROWNI components details

173
174
175
176
177
178
179
180
181

To explain how these features are achieved in BROWNI, a “zoomed” architecture diagram is shown in Figure 2, detailing the components of BROWNI that facilitate the configuration, interactivity, and integration of the simulations. First, the application should provide input data such as domain configuration, bathymetry, and initial conditions. Some of these data may be in different formats, such as CSV, JSON text files, or PNG and JPG images. This data is internally received by a reader component, which contains methods to parse these different formats. The reader converts this data into useful input for the shallow water model component, which can only receive input data in one format. Once the data is received, the shallow water model component is then in charge of executing the simulation sequentially. It then calls the WebGL instructions that use GPU acceleration to speed up the calculations.

182
183
184
185
186
187
188

To run several iterations until a specific timestamp is reached, a simulation controller component is implemented, which not only runs the simulation program, but also calls model functions to drive the simulation and extract results to fill an HTML5 canvas element with a pseudo-color map or colored texture, representing the values of a desired variable such as current wave heights, maximum heights, or arrival times. Then, using the controller, the application can interrupt the simulation with common playback commands (play, pause, restart, etc.) on user demand; and with the model, the application can collect the results of the simulation and populate its views, for example, by overlaying the canvas texture on a map or displaying a time series plot of a specific point of interest.

3.3. Asynchronous event-handling with life-cycle callbacks

189
190
191
192
193
194
195

Although the shallow water model and the simulation controller components help with interactivity and integration, it is important to note that they are not sufficient for handling asynchronous state changes in a predictable way. For example, one may be calling model functions before the data is available, or before the components exist, or it may be necessary to interrupt the simulation cycle under a certain specific condition, say, once the wave height is greater than a threshold or after a certain number of iterations. For this reason, we introduce the concept of “life cycle” shown in figure 3, that describes the sequence of relevant stages since the data is received until the controller has reached its

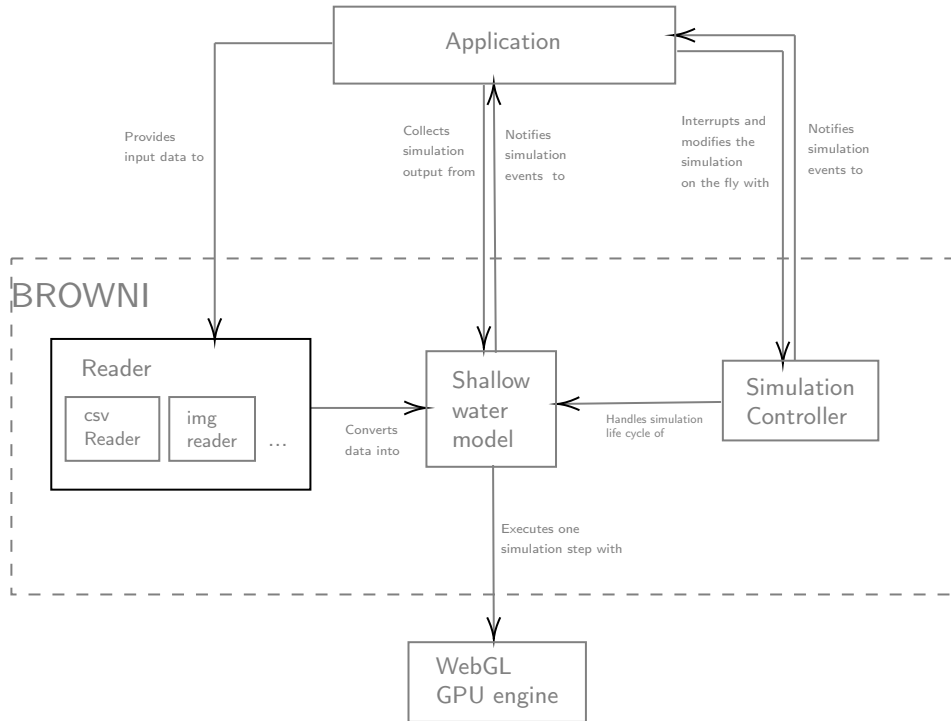


Figure 2: Architecture of BROWNI in terms of its internal components

196 final simulation time t_f . The application can then subscribe to events by providing a set of callback functions (listed
 197 on the left side of figure 3) that are internally called at specific steps of the life cycle.

198 3.4. Model implementation in WebGL

199 The shallow water model component is the software representation of the algorithm and mathematical equations
 200 presented in section 2. It uses WebGL, a web standard for GPU accelerated computer graphics on the web browser
 201 [25], and as such, it requires one to adapt the code implementation to concepts and design constraints that are very
 202 specific to computer graphics. The chosen model is well suited for this approach as every node in the numerical mesh
 203 depends only on a small number of neighboring nodes, making it analogous to image rasterization algorithms [31].

204 The simulation (or "rendering") process is performed in a special type of program called Fragment Shader, which
 205 is written in a strongly-typed language called OpenGL Shading Language (GLSL), whose most basic purpose is to fill
 206 an RGBA matrix of pixels with the desired colors to build the target image, that is, a matrix with three intensity levels
 207 for red, green, and blue and one for the transparency level (alpha). In addition, by design, a Shader program has no
 208 memory of previously rendered scenes, therefore any prior information must be provided explicitly. For this reason,
 209 and for better performance, instead of rendering to the screen, BROWNI renders to a frame buffer object (FBO), which
 210 is a WebGL object that allows storing of the pixel data of a Shader into a texture that can be used as input later in
 211 other Shader instances. In total, it uses two FBOs: FBO_1 to store the results of the previous time step and FBO_2
 212 storing new results that depend on FBO_1 . On each pixel of an FBO, the RGBA values are stored corresponding to
 213 (η, M, N, h) at each point in the simulation.

214 Figure 4 shows the flow of the simulation, in the scope of the shallow water model component. After the data is
 215 received from the reader component (top diagram), the bathymetry is rendered into a texture, the simulation timestep
 216 is calculated and the initial condition is rendered into FBO_1 . Then, when the `runSimulationStep` function of the
 217 shallow water model component is called, FBO_1 is assigned as the previous step texture and the simulation shader
 218 program is run, with its output stored into FBO_2 . The flow ends with FBO_1 and FBO_2 being swapped, such that the
 219 data is ready when the `runSimulationStep` function is called again.

Integrating tsunami simulations with Browni

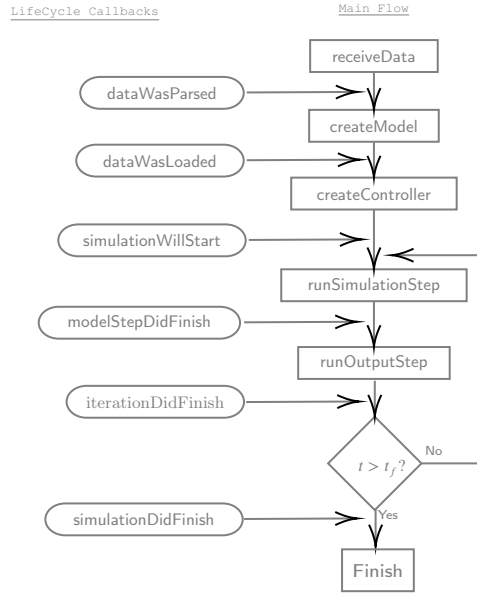


Figure 3: Life cycle diagram of a simulation in BROWNI.

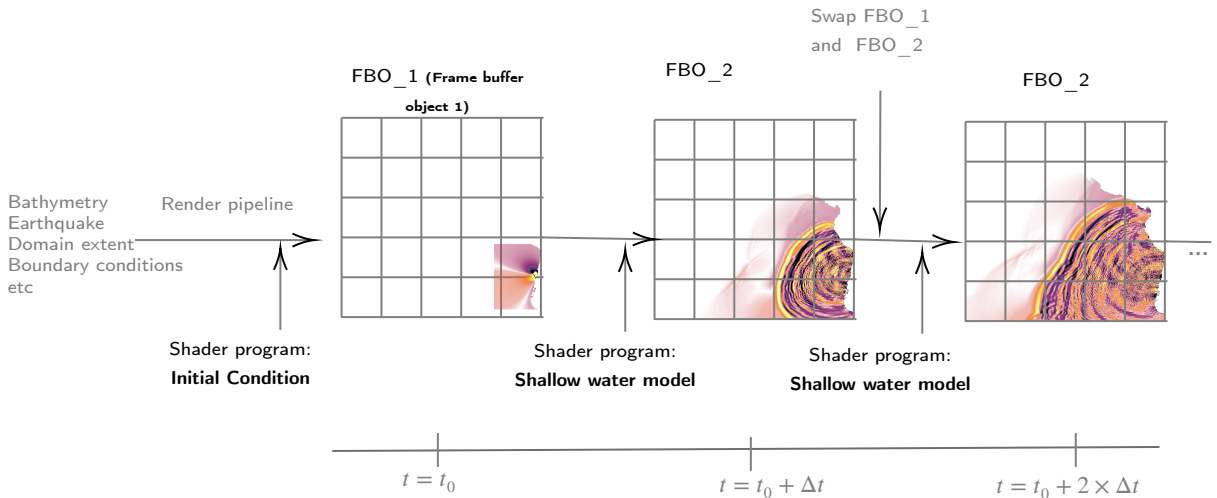


Figure 4: Diagram depicting how the timestepping scheme works with the WebGL rendering pipeline.

220 4. Validation

221 The mathematical model presented in section 2 has been already validated by other authors, for example in [34, 30]
 222 and more recently in [35, 36]. Here we examine the accuracy of the results against measurements of two different far-
 223 field tsunamis to ensure the correctness of the implemented model. This is also useful to show how BROWNI can
 224 be configured in a manual workflow where post-processing tasks are made outside of the browser context where the
 225 simulation runs.

226 To configure the simulations, it is necessary to characterize the scenario, provide output options, and optional life
 227 cycle callbacks to catch key events of the simulation. An example of a standalone HTML/JavaScript code that can be
 228 run from the web browser is shown in Figure 5. Two scenarios were studied, corresponding to the 8.8 Mw 2010 Maule
 229 earthquake and 9.1 Mw 2011 Tohoku earthquake. For each scenario, bathymetry and earthquake information were
 230 provided through external uncompressed binary files. The bathymetry source is ETOPO-1 [37], and the earthquake is

```

1 <body>
2   <script src="BROWNI.js"></script>
3   <script>
4     const scenario = {
5       xmin: 90,
6       xmax: 290,
7       ymin: -70,
8       ymax: 70,
9       discretizationWidth: 4000,
10      discretizationHeight: 2800,
11      bathymetry: 'bathymetry',
12      earthquake: 'earthquake.csv'
13    };
14    const outputOptions = {
15      displayWidth: 1000,
16      displayHeight: 700,
17      stopTime: 60 * 60 * 25
18    };
19    const lifeCycle = {
20      simulationDidFinish: (model, controller) => {
21        controller.downloadMaximumHeights();
22        controller.downloadArrivalTimes()
23      }
24    };
25    const browni = new Browni(scenario, outputOptions, lifeCycle);
26  </script>
27 </body>

```

Figure 5: Example standalone HTML file with Javascript code for configuring BROWNI to produce heat maps of figures 6 and 8

231 represented by finite fault models proposed by [38] and [39] for the scenarios of Chile and Japan, respectively, using
 232 the formulas mentioned in section 2. As shown in Figure 5, the domain covers the spherical rectangle $[90, 325.83] \times$
 233 $[-60, 70]$ and is discretized in a spherical uniform grid of $4717 \times 2600 \approx 12.26$ million nodes with a spacing of 3
 234 min. Each simulation is run until 25 h of propagation (the `stopTime` parameter) with a timestep of 2.9365 s, which is
 235 configured by default for a CFL number of 0.5 to respect the stability condition explained in section 2; all boundary
 236 conditions are open.

237 Computed results are shown in Figures 6, 7, 8, and 9, which correspond to the 8.8 Mw 2010 Maule earthquake and
 238 9.1 Mw 2011 Tohoku earthquake.

239 Since the postprocessing is made outside the context where the simulation is running, the only provided lifecycle
 240 callback is the `simulationDidFinish` function (see figure 3), where it instructs the `Controller` to export files
 241 containing gridded values of maximum amplitude and arrival times after the simulation has reached the `stopTime`.

242 “Heat maps” of maximum amplitudes and travel time isochrones were produced in Python using the results of
 243 BROWNI. Figures 6 and 8 show these results, which demonstrate how the implemented numerical model repre-
 244 sents tsunami propagation, including reflection and refraction patterns that are developed due to its interaction with
 245 bathymetry and shore lines.

246 Line plots showing the time series of wave amplitude for each scenario are shown in Figures 7 and 9. Black lines
 247 correspond to the measurements of Deep-ocean Assessment and Reporting of Tsunamis (DART) buoys after filtering
 248 out the astronomical tide using a low pass filter. DART buoys locations and identification numbers are also shown
 249 with circles in Figures 6 and 8. The other lines represent results computed by different tsunami-simulation software:
 250 Easywave (red), a software developed in the German Research Center for Geosciences, Potsdam [10] that implements
 251 the same numerical scheme described in section 2 with C++ and CUDA; and GeoClaw (green), part of the ClawPack
 252 software [5], that solves the non-linear shallow water equations using a finite-volume scheme designed to represent the
 253 propagation of tsunamis both at the global and regional scales, including inundation. All models’ results were shifted
 254 in time to match the location of the first observed peak. The computed time-shift of each model with respect to the
 255 measurements is shown in tables 1 and 2. A negative time-shift indicates an early arrival.

256 In both scenarios, BROWNI and EasyWave show almost no difference from each other, since both solve the same

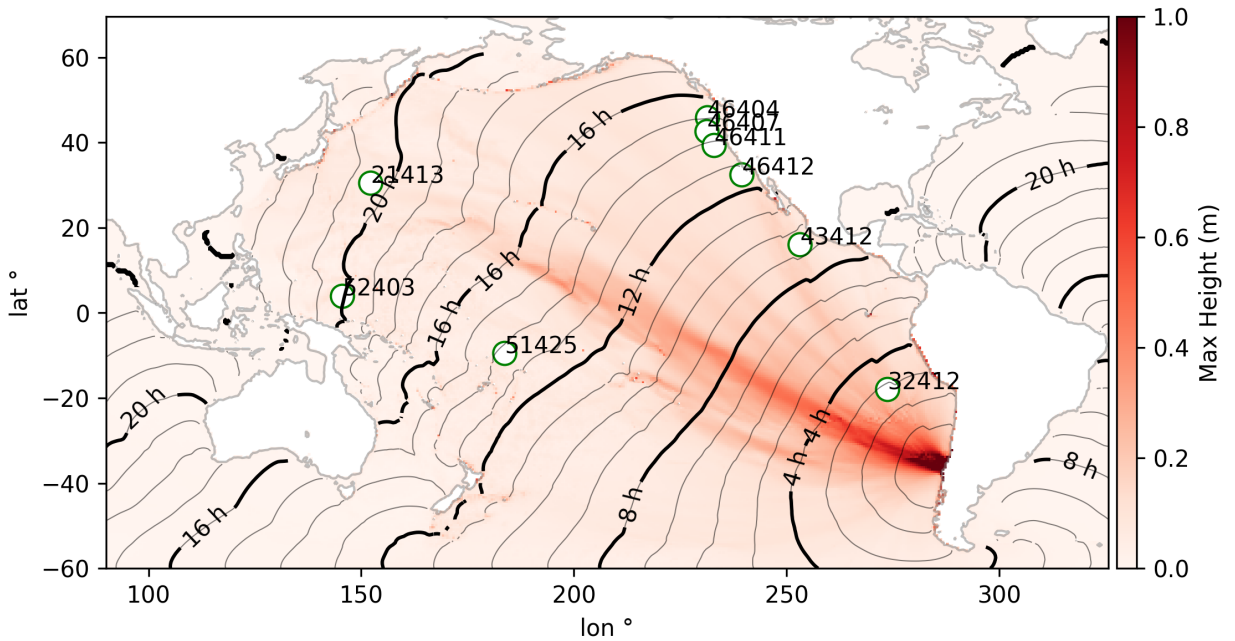


Figure 6: “Heat maps” of maximum wave amplitude and arrival time isochrones calculated by BROWNI for the 8.8 Mw scenario of Maule, Chile, 2010 and DART buoys locations.

Buoy ID	Arrival (hours)	BROWNI (minutes)	EasyWave (minutes)	GeoClaw (minutes)
32412	3.27	0.44	0.48	-1.69
43412	9.81	-2.06	-1.35	-5.85
46412	13.15	-2.87	-2.46	-8.96
51425	14.43	-13.20	-13.21	-18.14
46411	14.48	-10.06	-9.59	-15.98
46407	14.98	-4.05	-3.58	-11.04
46404	15.60	-6.57	-6.42	-13.93
21413	21.73	-23.83	-19.41	-27.24
52403	22.11	-12.14	-11.17	-15.55

Table 1

Time-shift applied to the results of each simulation-software of figure 7 sorted by arrival-time at each buoy (column "Arrival"). A negative amount indicates an earlier arrival.

257 set of equations using the same numerical scheme. GeoClaw shows an earlier arrival than BROWNI and Easywave, a
 258 behavior already observed in [35], that is explained by the different dispersion characteristics of the numerical schemes
 259 and equations. Finally, a consistent delay on the arrival of the first wave, that increases with the distance to the
 260 earthquake source, can be observed in all computed results. As reported by [40] and [41], this delay can be reduced by
 261 including additional physical features ignored in the fundamental assumptions of shallow water models such as water
 262 compressibility and earth's elasticity.

263 These results show that by using WebGL from the web browser as a computing engine, BROWNI can produce
 264 results as accurate as other tsunami simulation software packages. The next section exemplifies how additional versa-
 265 tility can be obtained, by taking advantage of WebGL as a web standard to produce interactive visualizations that are
 266 integrated with other sources of data and user-interface elements.

Integrating tsunami simulations with Browni

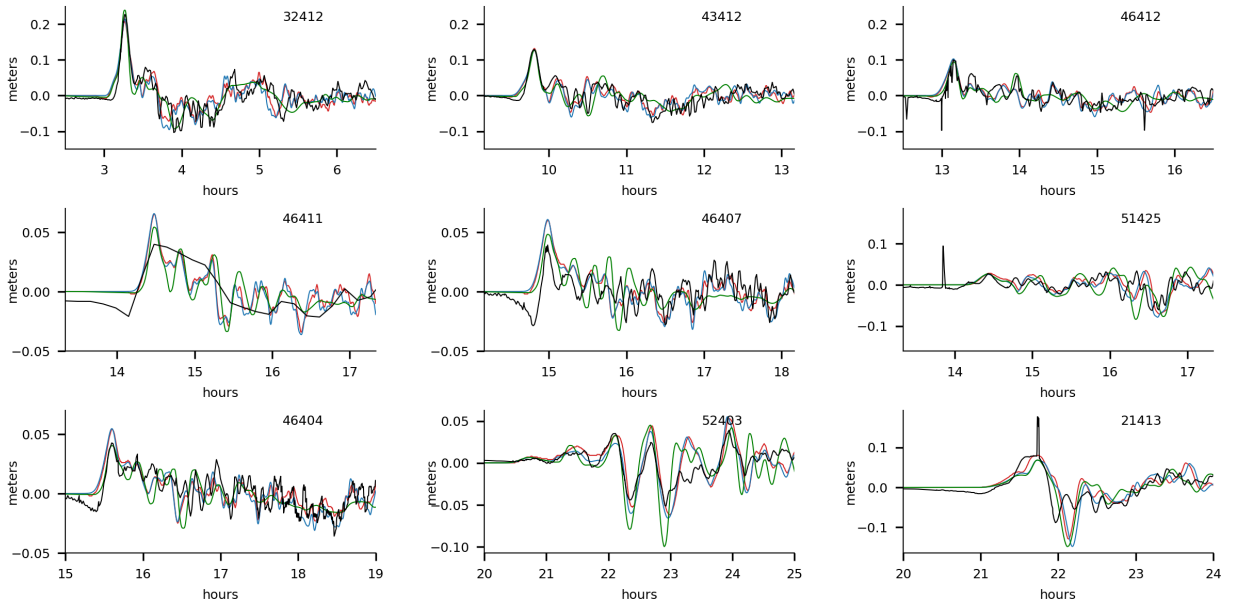


Figure 7: Time series of wave amplitude for the 8.8 Mw scenario of Maule, Chile, 2010, obtained by BROWNI (blue line), EasyWave (red line), GeoClaw (green line) and de-tided measurements (black line) of DART buoys. Time-shifts applied are shown in table 1

Buoy ID	Arrival (hours)	BROWNI (minutes)	EasyWave (minutes)	GeoClaw (minutes)
21413	1.34	-1.90	-2.00	-2.57
52403	5.18	-6.68	-6.22	0.53
51425	8.59	-4.64	-4.57	-8.25
46404	8.89	-8.33	-8.19	-10.35
46407	8.99	-9.49	-9.10	-10.71
46411	9.34	-9.83	-10.05	-13.48
46412	10.36	-8.81	-8.69	-11.83
43412	13.50	-6.91	-6.89	-11.97
32412	19.45	-13.81	-12.83	-18.37

Table 2

Time-shift applied to the results of each simulation-software of figure 9 sorted by arrival-time at each buoy (column "Arrival"). A negative amount indicates an earlier arrival.

267 5. Application to scientific outreach and communication

268 We used BROWNI to try different kinds of interactive applications on outreach activities that aim at increasing
 269 tsunami risk awareness. For example, we have participated in the international competition "Mathematics of Planet
 270 Earth 2017" organized by Imaginary [42] and the Futur.E.S. festivals organized by Cap Digital [43] (see picture (a) in
 271 figure 13). Specific details on outreach activities that we have performed are reported in [44].

272 By wrapping the simulator in a library we have separated concerns in what respects to tsunami-simulations and
 273 application development, thus simplifying the development process, making it easier and more sustainable. And once
 274 those applications were implemented, the most remarkable benefit has been the ability to reduce the total cost of
 275 gathering evidence from tsunami simulations to verify or discard previous assumptions. This makes it possible to
 276 focus the conversations on the scientific questions that are important for both the science communicators and the
 277 spectators instead of the complexities and implementation details of the simulator and the interactive visualizations.
 278 The result is a dynamic and engaging experience for all participants.

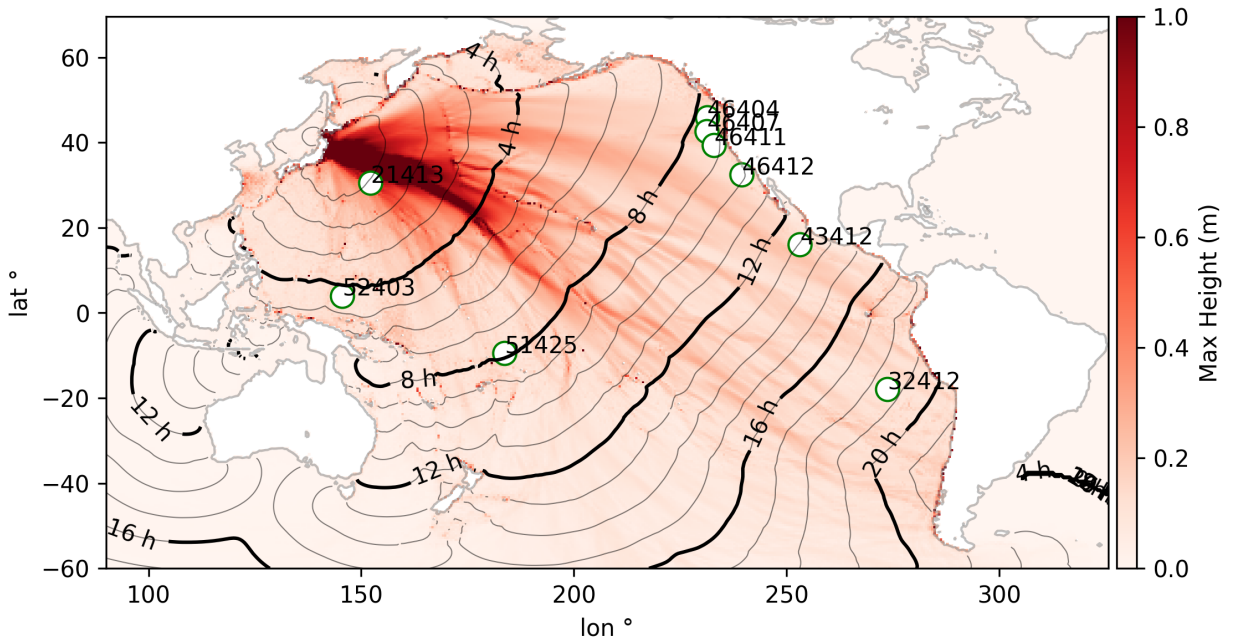


Figure 8: “Heat maps” of maximum wave amplitude and arrival time isochrones calculated by BROWNI for the 9.1 Mw scenario of Tohoku, Japan, 2011 and DART buoys locations.

279 In this section we describe two of these applications that were developed with BROWNI, namely TsunamiLab,
 280 a Single-Page Application deployed at www.tsunamilab.cl, and the TsunamiLab-Pool, an interactive augmented
 281 reality device meant for public exhibitions such as science fairs.

282 5.1. TsunamiLab: a Single-Page Application

283 TsunamiLab, available at www.tsunamilab.cl, is a Single-Page Application (SPA), i.e., a web application loaded
 284 from a single HTML document and whose content is updated using Javascript. TsunamiLab possesses several features
 285 that enable users to observe the propagation of tsunami waves around the world, and select and modify scenarios based
 286 on historical and “synthetic” data, i.e., user-defined earthquake locations and magnitudes.

287 Figure 10 shows the main view of TsunamiLab, available at www.tsunamilab.cl. Besides BROWNI, Tsunami-
 288 Lab is built using three main tools: ReactJS [45], a Javascript library maintained by Facebook that facilitates the
 289 development of complex interfaces under a reactive programming paradigm; Three.js [46] a popular general purpose
 290 3D library that uses WebGL; and the USGS Earthquake Catalog web API, an API that allows custom searches for
 291 earthquake information using a variety of parameters [47].

292 5.1.1. Frontend architecture with BROWNI

293 The software architecture is described in a component tree as shown in figure 11. On this tree, components with a
 294 common parent can access its state variables and pass them to the next component in the same branch. Components
 295 down the tree on a same branch can also receive callback functions to trigger changes in state variables of a parent
 296 component, triggering updates and data requests. For example, the magnitude and location of the current scenario
 297 are shared state-variables between the BROWNI, canvas, scene, synth. scenarios, and historical scenarios components,
 298 and thus any time there is an update (triggered by an user interaction, for example) all of these components will be
 299 notified and re-rendered.

300 TsunamiLab uses BROWNI to run the simulation and Three.js for displaying a 3D scene (in the scene component of
 301 Fig. 11) where the globe, pins, and other 3D graphics components are rendered. The integration of BROWNI with
 302 Three.js is performed by sharing an HTMLCanvasElement (the canvas component of Fig. 11) between them.

303 This canvas serves for creating instances of the WebGL context used by BROWNI to run the simulations, and also
 304 for storing the “heatmap” with the colors of the simulation at every timestep, which is then displayed in the globe of

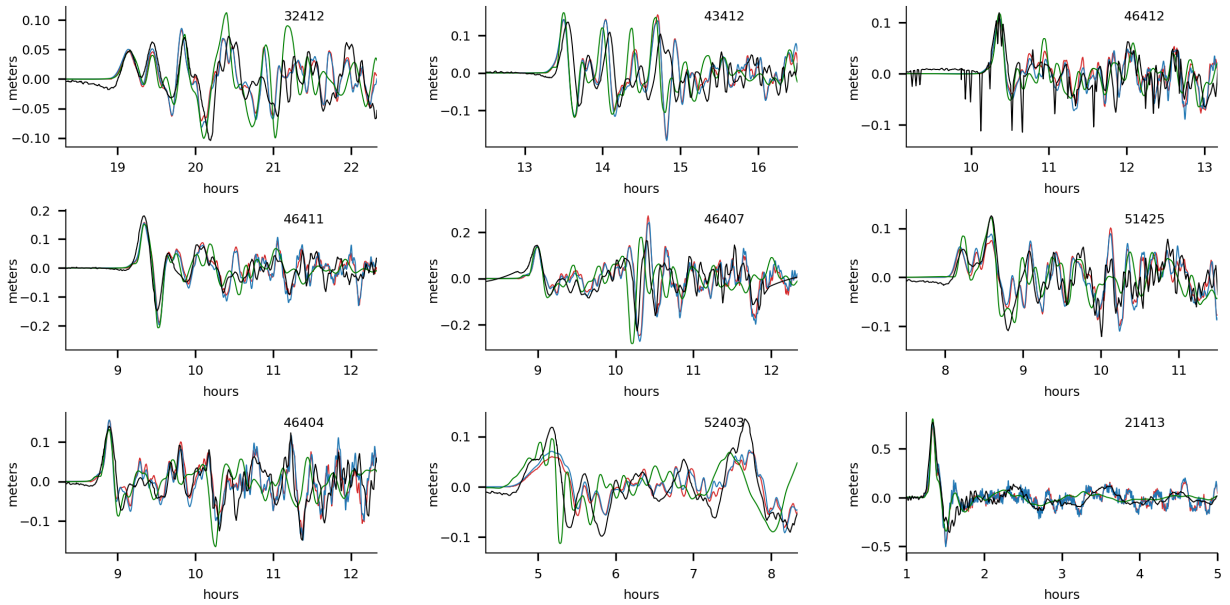


Figure 9: Time series of wave amplitude for the 9.1 Mw scenario of Tohoku, Japan, 2011, obtained by BROWNI (blue line), EasyWave (red line), GeoClaw (green line) and de-tided measurements (black line) of DART buoys. Time-shifts applied are shown in table 2

305 the scene as a texture on top of the base map.

306 5.1.2. Numerical model settings

307 The simulation domain covers the spherical rectangle $(\lambda, \theta) \in [-180, 180] \times [-70, 70]$ with a numerical grid of
 308 10 minutes resolution. Bathymetry is sampled from the ETOPO-1 dataset and compressed into a PNG image with
 309 the same resolution as the simulation to facilitate navigation. The quantization error from the PNG compression is found
 310 to be insignificant, since the only region where it becomes important is at the shores, which are excluded from the
 311 numerical domain of the simulator. The E–W boundary conditions are periodic (at $\lambda = -180, 180$), whereas the N-S
 312 borders are open (at $\theta = -70, 80$); as described on section 2, continental shorelines are simulated as inner reflecting
 313 walls.

314 5.1.3. Historical scenarios

315 Historical scenarios are represented in the globe as a collection of wave pins scattered around the globe at earth-
 316 quake locations. Earthquake location, magnitude, and focal mechanism information are obtained through the publicly
 317 available USGS Earthquake Catalog web API. Whenever the user clicks on a visible pin, an HTTP request is sent to
 318 the USGS Earthquake Catalog and the shared `magnitude` and `location` states are updated, triggering the update of
 319 BROWNI with the new scenario. At the same time, text information is updated in the “Scenario Info” component
 320 (bottom right table in figure 10) to give the user context about the current scenario.

321 5.1.4. Synthetic scenarios

322 Synthetic scenarios can be configured in a separate box in the top right corner of figure 10. The magnitude can be
 323 chosen by increasing or decreasing the radius r of the inner circle, in which case the magnitude of the new scenario is
 324 selected to be proportional to r^3 . Earthquake location can also be selected by dragging and dropping the orange pin on
 325 top of the globe. BROWNI is updated automatically whenever the `location` or the `magnitude` are changed through
 326 either of these two actions.

327 5.2. TsunamiLab-Pool: An augmented reality-device

328 With the development of TsunamiLab we noticed that people showed a greater interest after trying out their own
 329 scenarios to verify or discard their previous assumptions about tsunamis. However, from our experience in various

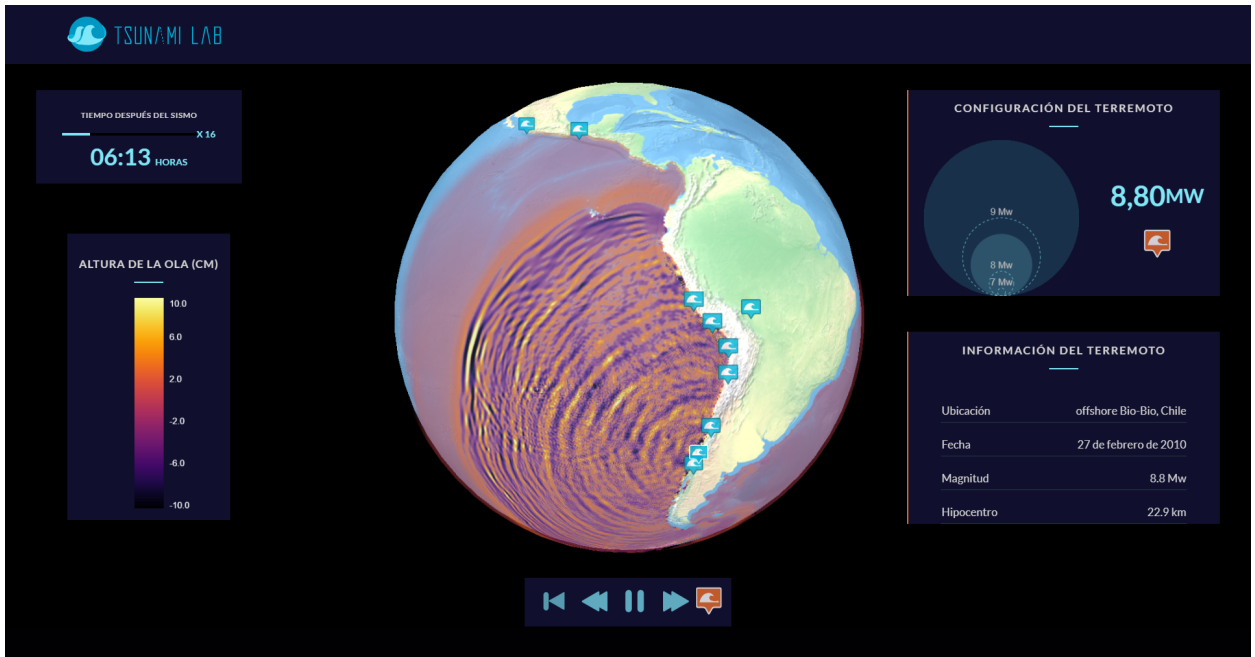


Figure 10: Screenshot of the main view of www.tsunami.cl

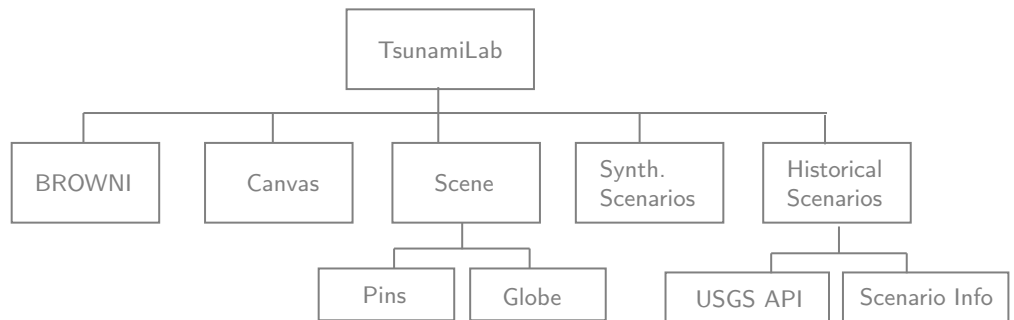


Figure 11: Component tree describing the architecture of TsunamiLab

330 exhibitions, we found that the classic mouse-keyboard-monitor system for visualization and interaction was still un-
 331 comfortable to use in contexts such as science fairs or exhibitions at local schools.

332 For this reason we created the TsunamiLab-Pool, an augmented reality device in which the visualization of the
 333 simulation is projected onto a circular surface supported by a rigid frame. The interaction is handled by an external
 334 controller, to rotate the globe and change the location and magnitude of new earthquakes. We tried two different
 335 controllers: the Sony PlayStation 3 Move (PSMove) controller and the Leap Motion controller. The former is a wireless
 336 controller developed by Sony that, in addition to traditional buttons, includes a marker whose position is tracked by a
 337 camera and mapped accordingly into the scene to help with pointing actions. The latter consists of a set of infrared
 338 lights and cameras that are used to infer the position of the user's fingers, hands, and arms. With the Leap Motion
 339 Controller the user is actually able to control the parameters of the simulation and the visualization by just moving the
 340 hands in the air (see picture (b) of figure 13).

341 The TsunamiLab-Pool software is built as in diagram 12. First the user interacts with the physical device where
 342 the simulation is projected using a controller. This controller sends the raw information of the interaction to a local
 343 controller-service. This controller-service then translates the data from the raw interactions into simulation commands
 344 that are forwarded to the simulation web app, which in turn, uses them to configure and calculate the new scenario and

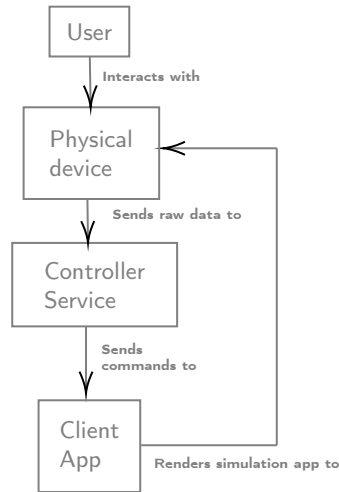


Figure 12: Diagram with the architecture of the TsunamiLab-Pool.

render the visualization back into the projection of the physical device.

The controller service is a python program that extracts the data from the actual remote controllers. It uses the PSMoveAPI python library [48] for the PSMove controller and the python API of the Leap Motion SDK version 2.3, for the Leap Motion controller. The simulation web app is a plain HTML-Javascript-CSS application, served locally with a simple http server that receives messages from the controller-service in real-time through a websockets connection. These messages change the state of the simulation, that is running with BROWNI, and also the visualization, which is rendered using Three.js.

Thanks to the API of BROWNI, no changes to the source code of the library were necessary. This made the development experience cleaner and more sustainable than, for example, duplicating the source code on each of the applications, since it allows for a better separation of concerns. And because BROWNI uses WebGL, the integration inside of the website was straight-forward and its on-site deployment only required a gaming computer to give a smooth experience to visitors and science communicators.

6. Discussion on Software and Hardware Limitations

Running the tsunami model on the browser using WebGL offers several opportunities as discussed before, but some limitations or constraints should be discussed.

6.0.1. Multi-threading

Using multiple threads or processes can be beneficial for a webpage to prevent different tasks from interrupting each other inside the browser. This is possible in recent versions of web-browsers such as those based on Chromium, by using two JavaScript API's: the `WebWorker` API to run tasks in the background, detached from the main thread; and the `OffScreenCanvas`, to run the WebGL simulation from inside a `WebWorker`, instead of an `HTMLCanvasElement`. Naturally, there are two ways on which multi-threading can improve the execution of the simulation: preventing other tasks from slowing down the computations; and preventing the simulation from slowing down the other tasks.

Although the first case is straight-forward, and is the main use-case for the `WebWorker` API, the second case must be reviewed carefully when designing a web application, since large simulations may still interrupt the webpage and other applications outside of the browser. This happens because GPUs are designed to handle only one rendering task at a time, including the rendering of the display's content.

This does not happen with CPUs since operating systems usually employ preemptive multitasking, a feature that allows to temporarily pause heavy tasks while keeping the computer responsive. This feature is not usually available in GPUs [49] and the operating system may even restart the GPU driver, assuming that the GPU was blocked, even though it was only processing a heavy task [50]. The only workaround in WebGL is to make a compromise between

375 the size of the simulation (mesh size or number of time-steps computed per second) and the responsiveness of the
376 system.

377 To overcome this limitation, native APIs may make use of a secondary GPU if available, however, this is not
378 possible in WebGL since the browser can only use the same GPU as the user's display.

379 **6.0.2. Background tabs and batch simulations**

380 Leaving a simulation running in a background tab may cause the browser to temporarily slow it to even less than
381 one frame per second, in order to alleviate resources for the currently active browsing tab. For cases where interactive
382 simulation and visualization is the priority, as is the interest on this work, this is convenient since it prevents the user
383 from missing important changes while they finish other tasks in the new tab. However, this can be limiting in other
384 cases, for example, if one seeks to run batch simulations in the background, and process their results only at the end.
385 In this case, users could disable this feature using the command-line options of the browser's executable, which is the
386 case for Chrome since version 57 [49], for example. Disabling this feature will thus compromise the setup-cost and
387 browser support, so its convenience should be evaluated on a case-by-case basis.

388 A different approach for running batch simulations is to, instead of running the simulations in the background, run
389 them interactively in the forefront while changing their parameters on the fly. For smaller scenarios, several instances
390 of BROWNI could be created in the same browser, and for more complex ones, a tiled-display visualization could be
391 built similarly to [14] and [51].

392 **6.0.3. Scalability**

393 Using CUDA to scale-up the size of the simulation has already been shown to be possible, for example in [52, 53].
394 However, in the case of WebGL this is not straight-forward, if ever possible or convenient.

395 In a single node, WebGL has some limitations, which are already listed in [54]. These limitations also exist in
396 the native APIs of OpenGL [55], OpenGL ES [56], and Direct3D [57]. Each GPU has a maximum texture size that it
397 can store and it depends not only on the available memory of the GPU but also on the specific design of the device.
398 Also, even if the information were split into several textures by decomposing the domain, there exists a maximum
399 number of textures that the GPU can store that also depends on the particular device. These factors should be taken
400 into consideration in order to give proper support to the platforms of each web application.

401 Some additional developments could be done to overcome these limitations. For example, one could try to decom-
402 pose the simulation among different computers and WebGL contexts with a real-time communication protocol such
403 as WebSockets to step through the simulation, alleviating the resources of each computer. Also, adding server-side
404 support could be done with the Node.js JavaScript runtime instead of the browser's. However, these approaches are
405 outside the scope of this work and have not been studied here.

406 **6.0.4. Browser performance**

407 Browsers such as Firefox, Edge, Google Chrome and others based on Chromium run WebGL calls using the Almost
408 Native Graphics Layer Engine (ANGLE) API [58], translating them into one of the graphics APIs available for that
409 platform; usually OpenGL on Linux and Mac, and Direct3D on Windows. This naturally means that any WebGL
410 application will be at most as fast as its native counterpart.

411 To examine the performance of BROWNI on the web browser, the elapsed time for the 8.8 Mw Maule, Chile, 2010
412 case (introduced in detail later on section 4) is shown in Table 3. Simulations were performed on a gaming laptop with
413 an Intel Core i7-7700HQ 2.8 GHz CPU and 8 GB of RAM with Intel HD Graphics 630 integrated graphics card and
414 also with an NVIDIA Geforce GTX 1060 dedicated graphics card. Two grid sizes were used with a spacing of 3 and
415 15 minutes until 25 hours of simulation passed. Although the execution times were relatively similar for the coarse
416 grid, the simulation ran 5.16 times faster with the dedicated GPU. This speedup is smaller than, for example, the 10.16
417 speedup reported in [10] for EasyWave running on CUDA, instead of on the CPU, which could be explained by the
418 previously mentioned overhead.

419 Although it can also be influenced by other factors, such as differences in the architectures of CPUs and integrated
420 graphics cards, this still illustrates that speedup differences are expected in practice. However, as was described in the
421 previous section, for applications in science education and risk communication, the interactivity and efficient visual-
422 ization can increase the performance of other tasks needed to use the simulator, making the user more proficient on
423 understanding its results.

	Integrated GPU	Dedicated GPU
15 minutes 943 × 520 grid	1.6 min	1.44 min
3 minutes 4717 × 2600 grid	43.6 min	8.44 min

Table 3

Execution times for the simulation of the 8.8 Mw, 2010 Maule earthquake for two grids of different resolution, using an integrated and dedicated GPU. Integrated GPU: INTEL(R) HD Graphics 630; Dedicated GPU: NVIDIA Geforce GTX 1060.

7. Conclusions

On this study, it has been shown how GPU computing from web browsers can allow users to gather and analyze data from simulations efficiently, without being concerned by implementation details and tasks that are not relevant for understanding the physical phenomena of tsunami propagation and its potential implications to tsunami risk management, such as setup, configuration, pre-processing and post-processing of the results.

To demonstrate this, a Javascript library called BROWNI was introduced, which has an API designed to facilitate the creation of interactive GPU-powered tsunami simulations and visualizations in the web browser. By using this API, it was possible to separate concerns when building web applications, facilitating code-reusability and maintainability across different software platforms.

Some limitations of this approach were also examined, finding that large scenarios may experience reduced performance and reduced mesh-resolution scalability support, as also necessary trade offs on browser-support to benefit from CPU multi-threading and background processing.

Results were also validated with other simulation software and measurements. These are coherent with the fact that BROWNI implements algorithms already used in operational and research settings: differences found on time-series were recalled, however these had already been discussed in the literature. This confirms the correct implementation of the algorithms.

Finally, we showed two software platforms that were developed using BROWNI, whose aim is to increase tsunami risk awareness: TsunamiLab and TsunamiLab-Pool. By creating and using these applications on different opportunities as tsunami-risk communication tools it was possible to confirm that: (1) that the benefits of the interactive simulation and visualization on the web browser can overcome its limitations, since users can focus on the scientific questions instead of the implementation details, thus producing a more engaging experience; (2) that developers can benefit from a more sustainable developer experience by using the BROWNI API as a part of their applications without worrying about the source code or other details of the simulator.

Future work may include changes to the model component to increase the accuracy in the estimation of arrival times, as proposed by [41]. In addition, improving the numerical dispersion to match the physical dispersion of linear waves in the ocean as proposed by [59] and ultimately the inclusion of higher resolution algorithms to include nonlinear effects and bottom friction is important to represent smaller scale phenomena. Other questions that remain open are in regards to the possibility of using WebGL for high performance computing in tsunami modeling, which would require adding server side support and distributed computing capabilities to extend the scalability of the simulator as discussed on section 6. Furthermore, the question of the effectiveness of interactive tsunami simulations to support educational and/or communicational methodologies for improving tsunami awareness, risk perception, and other topics has not been covered here and could also be studied with this library.

8. Acknowledgements

This project has been funded by projects CONICYT/FONDAP/15110017 "Centro de Investigación para la Gestión Integrada de Desastres Naturales" and CORFO 10CEII-9157 "Inria Chile". Support was also received by the Marine Energy Research & Innovation Center (MERIC, project CORFO 14CEI2-28228).

The authors are also grateful for the contributions of professor Felipe Cortez and the students of the course "Usabilidad y Nuevos Medios" of the School of Design of the second semester of 2018 of the Pontifical Catholic University of Chile, for their feedback and design proposals for TsunamiLab.

Name of software:	BROWNI
Maintainer:	José Galaz https://jgalazm.github.io
Year first available	2018
Hardware required	Computer with integrated or dedicated graphics card
Software required	Google Chrome 64.0.3282 or greater
Availability:	MIT license at https://github.com/jgalazm/browni
Program Language:	JavaScript, GLSL
Program Size:	179KB (source code) 2.0 MB (repository)

Table 4
Software availability of BROWNI

9. Software availability

The source code of BROWNI is available under an open-source MIT license at <https://github.com/jgalazm/browni>, and detailed information is specified in Table 4. Visualizations and data processing were performed in Python with matplotlib, and the source code for the cases presented herein is written in Jupyter Notebooks and versioned at <https://github.com/jgalazm/comp-geo-paper-figures>.

References

- [1] U. Kânoğlu, V. Titov, E. Bernard, C. Synolakis, Tsunamis: bridging science, engineering and society, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 373 (2015) 20140369.
- [2] P. Catalán, J. Cañas, C. Zúñiga, C. Zelaya, A. Gubler, L. Pizarro, C. Valdés, S. Miranda, Sistema integrado de predicción y alerta de tsunamis (sipat), in: *XXII Congreso Chileno de Ingeniería Hidráulica*, 2013.
- [3] V. Titov, U. Kânoğlu, C. Synolakis, Development of MOST for real-time tsunami forecasting, Ph.D. thesis, American Society of Civil Engineers, 2016.
- [4] O. Kamigaichi, Tsunami forecasting and warning, in: *Encyclopedia of complexity and systems science*, Springer, 2009, pp. 9592–9618.
- [5] Clawpack Development Team, Clawpack software, 2020. URL: <http://www.clawpack.org>. doi:<https://doi.org/10.5281/zenodo.4025432>, version 5.7.1.
- [6] M. J. Berger, D. L. George, R. J. LeVeque, K. T. Mandli, The GeoClaw software for depth-averaged flows with adaptive refinement, *Advances in Water Resources* 34 (2011) 1195–1206.
- [7] X. Wang, User manual for comcot version 1.7 (first draft), Cornell University 65 (2009).
- [8] P. Lynett, P. Liu, K. Sitanggang, D. Kim, Modeling wave generation, evolution, and interaction with depthintegrated, dispersive wave equations coulwave code manual, Cornell University Long and Intermediate Wave Modeling Package (2002).
- [9] Y. Yamazaki, K. F. Cheung, Z. Kowalik, T. Lay, G. Pawlak, Neowave, in: *Proceedings and results of the 2011 NTHMP model benchmarking workshop*, Boulder: US Department of Commerce/NOAA/NTHMP (NOAA Special Report), 2012, pp. 239–302.
- [10] S. Christgau, J. Spazier, B. Schnor, M. Hammitzsch, A. Babeyko, J. Waechter, A comparison of cuda and openacc: accelerating the tsunami simulation easywave, *PARS-Mitteilungen: Vol. 31, Nr. 1* (2014).
- [11] O. Nielsen, S. Roberts, D. Gray, A. McPherson, A. Hitchman, Hydrodynamic modelling of coastal inundation, 2005.
- [12] N. Merati, C. Chamberlin, C. Moore, V. Titov, T. C. Vance, Integration of tsunami analysis tools into a gis workspace—research, modeling, and hazard mitigation efforts within noaa’s center for tsunami research, in: *Geospatial Techniques in Urban Hazard and Disaster Analysis*, Springer, 2009, pp. 273–294.
- [13] D. Keon, B. Steinberg, H. Yeh, C. M. Pancake, D. Wright, Web-based spatiotemporal simulation modeling and visualization of tsunami inundation and potential human response, *International Journal of Geographical Information Science* 28 (2014) 987–1009.
- [14] T.-J. Hsieh, W.-Y. Liang, Y.-L. Chang, M. T. Satria, B. Huang, Parallel tsunami simulation and visualization on tiled display wall using opengl shading language, *Journal of the Chinese Institute of Engineers* 36 (2013) 202–211.
- [15] R. M. Teeuw, M. Leidig, C. Saunders, N. Morris, Free or low-cost geoinformatics for disaster management: Uses and availability issues, *Environmental Hazards* 12 (2013) 112–131.
- [16] M. R. Kinzel, Using educational tools and integrative experiences via geovisualizations that incorporate spatial thinking, real world science and ocean literacy standards in the classroom: a case study examined., https://ir.library.oregonstate.edu/concern/graduate_projects/2v23vz927, 2009. Last accessed on Feb. 17 2019.
- [17] F. Jacquino, F. Pedrinis, J. Edert, G. Gesquière, Automated production of interactive 3d temporal geovisualizations so as to enhance flood risk awareness, in: *UDMV 2016*, 2016.
- [18] I. Curebal, R. Efe, H. Ozdemir, A. Soykan, S. Sönmez, Gis-based approach for flood analysis: case study of keçidere flash flood event (turkey), *Geocarto International* 31 (2016) 355–366.
- [19] G. W. Brunner, HEC-RAS river analysis system: hydraulic reference manual, US Army Corps of Engineers, Institute for Water Resources, Hydrologic . . . , 2010.
- [20] A. M. Schäfer, F. Wenzel, Tsupy: computational robustness in tsunami hazard modelling, *Computers & Geosciences* 102 (2017) 148–157.

- 507 [21] S. Tavakkol, P. Lynett, Celeris: A gpu-accelerated open source software with a boussinesq-type wave solver for real-time interactive simulation
508 and visualization, *Computer Physics Communications* 217 (2017) 117–127.
- 509 [22] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with cuda, in: *ACM SIGGRAPH 2008 classes*, ACM, 2008,
510 p. 16.
- 511 [23] D. Blythe, The direct3d 10 system, *ACM Transactions on Graphics (TOG)* 25 (2006) 724–734.
- 512 [24] S. Tavakkol, P. Lynett, Celeris base: An interactive and immersive boussinesq-type nearshore wave simulation software, *Computer Physics*
513 *Communications* 248 (2020) 106966.
- 514 [25] C. Marrin, *Webgl specification*, Khronos WebGL Working Group (2011).
- 515 [26] R. J. LeVeque, D. L. George, M. J. Berger, Tsunami modelling with adaptively refined finite volume methods, *Acta Numerica* 20 (2011)
516 211–289.
- 517 [27] UNESCO, *IOC Manuals and Guides No. 35, IUGG/IOC TIME Project*, 1997.
- 518 [28] F. Imamura, A. C. Yalciner, G. Ozyurt, Tsunami modelling manual (ver-3.1.4), UNESCO IOC international training course on Tsunami
519 Numerical Modelling (2006).
- 520 [29] R. G. Dean, R. A. Dalrymple, *Water wave mechanics for engineers and scientists*, volume 2, World Scientific Publishing Company, 1991.
- 521 [30] P. L.-F. Liu, Y.-S. Cho, S. Yoon, S. Seo, Numerical simulations of the 1960 chilean tsunami propagation and inundation at hilo, hawaii, in:
522 *Tsunami: Progress in prediction, disaster prevention and warning*, Springer, 1995, pp. 99–115.
- 523 [31] A. R. Brodtkorb, M. L. Sætra, M. Altinakar, Efficient shallow water simulations on gpus: Implementation, visualization, verification, and
524 validation, *Computers & Fluids* 55 (2012) 1–12.
- 525 [32] Y. Okada, Surface deformation due to shear and tensile faults in a half-space, *Bulletin of the seismological society of America* 75 (1985)
526 1135–1154.
- 527 [33] S. N. Ward, E. Asphaug, Asteroid impact tsunami: a probabilistic hazard assessment, *Icarus* 145 (2000) 64–78.
- 528 [34] V. V. Titov, F. I. Gonzalez, Implementation and testing of the method of splitting tsunami (most) model (1997).
- 529 [35] L. M. Adams, R. J. LeVeque, *Geoclaw model tsunamis compared to tide gauge results final report*, Technical Report, 2018.
- 530 [36] D. J. Greenslade, A. Annunziato, A. Y. Babeyko, D. R. Burbidge, E. Ellguth, N. Horspool, T. S. Kumar, C. P. Kumar, C. W. Moore,
531 N. Rakowsky, et al., An assessment of the diversity in scenario-based tsunami forecasts for the indian ocean, *Continental Shelf Research* 79
532 (2014) 36–45.
- 533 [37] C. Amante, Etopo1 1 arc-minute global relief model: procedures, data sources and analysis, <http://www.ngdc.noaa.gov/mgg/global/global.html> (2009).
534
- 535 [38] B. Delouis, J.-M. Nocquet, M. Vallée, Slip distribution of the february 27, 2010 mw= 8.8 maule earthquake, central chile, from static and
536 high-rate gps, insar, and broadband teleseismic data, *Geophysical Research Letters* 37 (2010).
- 537 [39] U.S. Geological Survey, M 9.1 - near the east coast of Honshu, Japan, 2018. URL: [https://earthquake.usgs.gov/earthquakes/](https://earthquake.usgs.gov/earthquakes/eventpage/official20110311054624120_30)
538 [eventpage/official20110311054624120_30](https://earthquake.usgs.gov/earthquakes/eventpage/official20110311054624120_30), accessed July 22, 2018.
- 539 [40] A. Abdolali, J. T. Kirby, Role of compressibility on tsunami propagation, *Journal of Geophysical Research: Oceans* 122 (2017) 9780–9794.
- 540 [41] S. Watada, S. Kusumoto, K. Satake, Traveltime delay and initial phase reversal of distant tsunamis coupled with the self-gravitating elastic
541 earth, *Journal of Geophysical Research: Solid Earth* 119 (2014) 4287–4310.
- 542 [42] IMAGINARY, *Winners of the second mathematics of planet earth competition*, 2017. URL: <https://imaginary.org/es/node/1318>.
- 543 [43] A. Devillard, Les innovations à ne pas rater au festival futur.e.s en seine, ??? URL: [https://www.sciencesetavenir.fr/decouvrir/](https://www.sciencesetavenir.fr/decouvrir/les-innovations-a-ne-pas-rater-au-festival-futur-e-s-en-seine-125129)
544 [les-innovations-a-ne-pas-rater-au-festival-futur-e-s-en-seine-125129](https://www.sciencesetavenir.fr/decouvrir/les-innovations-a-ne-pas-rater-au-festival-futur-e-s-en-seine-125129).
- 545 [44] N. Zamora, A. Gubler, V. Orellana, J. León, A. Urrutia, M. Carvajal, M. Cisternas, P. Catalán, P. Winckler, R. Cienfuegos, et al., The 1730
546 great metropolitan chile earthquake and tsunami commemoration: Joint efforts to increase the country’s awareness, *Geosciences* 10 (2020)
547 246.
- 548 [45] A. Fedosejev, *React.js Essentials*, Packt Publishing Ltd, 2015.
- 549 [46] R. Cabello, et al., *Three.js*, URL: <https://github.com/mrdoob/three.js> (2010).
- 550 [47] U.S. Geological Survey, *API Documentation - Earthquake Catalog*, ??? URL: [https://earthquake.usgs.gov/fdsnws/event/1,](https://earthquake.usgs.gov/fdsnws/event/1/v1.0.18)
551 [v1.0.18](https://earthquake.usgs.gov/fdsnws/event/1/v1.0.18) 2015-01-20.
- 552 [48] T. Perl, *Cross-platform tracking of a 6dof motion controller*, Using Computer Vision and Sensor Fusion, Austria (2012).
- 553 [49] I. Tanasic, I. Gelado, J. Cabezas, A. Ramirez, N. Navarro, M. Valero, Enabling preemptive multiprogramming on gpus, *ACM SIGARCH*
554 *Computer Architecture News* 42 (2014) 193–204.
- 555 [50] Microsoft, *Resource Limits (Direct3D 11) - Windows app developer documentation*, ??? URL: [https://docs.microsoft.com/en-us/](https://docs.microsoft.com/en-us/windows-hardware/drivers/display/timeout-detection-and-recovery)
556 [windows-hardware/drivers/display/timeout-detection-and-recovery](https://docs.microsoft.com/en-us/windows-hardware/drivers/display/timeout-detection-and-recovery), last accessed on May. 05 2021.
- 557 [51] A. Kamakshidasan, J. Galaz, R. Cienfuegos, A. Rousseau, E. Pietriga, Comparative visualization of deep water asteroid impacts on ultra-
558 high-resolution wall displays with seawall, in: *2018 IEEE Scientific Visualization Conference (SciVis)*, IEEE, 2018, pp. 142–145.
- 559 [52] J. Macías, A. Mercado, J. M. González-Vida, S. Ortega, M. J. Castro, Comparison and computational performance of tsunami-hysea and most
560 models for lantex 2013 scenario: Impact assessment on puerto rico coasts, in: *Global Tsunami Science: Past and Future*, Volume I, Springer,
561 2016, pp. 3973–3997.
- 562 [53] M. de la Asunción, M. J. Castro, J. M. Mantas, S. Ortega, Numerical simulation of tsunamis generated by landslides on multiple gpus,
563 *Advances in Engineering Software* 99 (2016) 59–72.
- 564 [54] GFXFundamentals, *WebGL2 Cross Platform Issues*, ??? URL: [https://webgl2fundamentals.org/webgl/lessons/](https://webgl2fundamentals.org/webgl/lessons/webgl-cross-platform-issues.html)
565 [webgl-cross-platform-issues.html](https://webgl2fundamentals.org/webgl/lessons/webgl-cross-platform-issues.html), last accessed on May. 05 2021.
- 566 [55] K. group, *Opengl® 4.5 reference pages*, ??? URL: <https://www.khronos.org/registry/OpenGL-Refpages/gl4/>, last accessed on
567 May. 05 2021.
- 568 [56] K. group, *Opengl® es 3.2*, ??? URL: <https://www.khronos.org/registry/OpenGL-Refpages/es3/>, last accessed on May. 05 2021.
- 569 [57] Microsoft, *Resource Limits (Direct3D 11) - Windows app developer documentation*, ??? URL: <https://docs.microsoft.com/en-us/>

- 570 windows/win32/direct3d11/overviews-direct3d-11-resources-limits, last accessed on May. 05 2021.
- 571 [58] C. project, ANGLE - Almost Native Graphics Layer Engine, ??? URL: <https://chromium.googlesource.com/angle/angle/+/>
- 572 master/README.md, last accessed on May. 05 2021.
- 573 [59] T. Ha, Y.-S. Cho, Tsunami propagation over varying water depths, *Ocean Engineering* 101 (2015) 67–77.

(a)



(b)



Figure 13: Pictures of activities performed with non-experts users: (a) Interactive display that uses the SONY TMPS Move Controller to configure earthquake magnitude and location, at the 2018 Paris FUTUR.E.S. technology festival organized by Cap Digital; (b) Interactive display that uses the Leap Motion controller to configure earthquake magnitude and location using hand gestures at a local science fair in Chile.