Fire propagation visualization in real time

Sigfrido Waidelich[1], Karina Laneri[2,3], and Mónica Denham[1,2]

[1]*Laboratorio de Procesamiento de Señales Aplicadas y Computación de Alto Rendimiento. Sede Andina, Universidad Nacional de Río Negro, Argentina*
[2]*Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina*
[3]*Física Estadística e Interdisciplinaria, Centro Atómico Bariloche, Río Negro, Argentina*
sigfri2wai@gmail.com, laneri@cab.cnea.gov.ar, mdenham@unrn.edu.ar

Abstract

Our motivation comes from the need of a tailored computational tool for simulation and prediction of forest fire propagation, to be used by firefighters in Patagonia, Argentina. Based on previous works on Graphic Processing Units (GPU) for fitting and simulating fires in our region, we developed a visualization interface for real time computing, simulation and prediction of fire propagation. We have the possibility of changing the ensemble of raster maps layers to change the region in which fire will propagate. The visualization platform runs on GPUs and the user can rotate and zoom the landscape to select the optimal view of fire propagation. Opacity of different layers can be regulated by the user, allowing to see fire propagation at the same time that underlying vegetation, wind direction and intensity. The ignition point can also be selected by the user, and firebreaks can be plotted while simulation is going on. After the performance of a high number of stochastic simulations in parallel in GPUs, the application shows a map of the final fire surface colored according to the probability that a given cell burns. In this way the user can visually identify the most critical direction for fire propagation, a useful information to stop fire optimizing resources, which is specially important when they are scarce like is the case of our Patagonia region.

Keywords: Forest Fire Simulation, GPGPU, High Performance Computing

1 Introduction

Every year, Argentina is the scenery of several and sometimes huge forest fires. Our country has a big extension of wild vegetation and firefighters unfortunately don't have reliable computational tools for management purposes.

In addition, our region still doesn't have fuel models for local vegetation to use as input for propagation and instead similar vegetation types from countries like Canada, Australia and USA are used [1, 2, 3].

Our team is working in collaboration with the fire brigade of "*Fire, Communications and Emergency Department*" (ICE) of the Nahuel Huapi National Park, in San Carlos de Bariloche. When a fire occurs, they mainly use their experience in the field to build the strategy to stop the fire. In some of the cases they used the outcome of FARSITE [4] but they noticed that simulations were very inaccurate depending on the vegetation that burns. Therefore we based our development on the needs of the ICE, a simulation tool adapted to our region with a friendly interface to be used to have more information to make decisions when fire is occurring.

Several simulators were developed to reproduce forest fire behavior, which need accurate inputs to give high quality results. Typical inputs are: topography (terrain elevation, slope and aspect), type of fuel, wind intensity and direction, and in several cases information from different weather stations. Simulators can be classified in to mainly two types: vector based and raster based [5]. Vector based approaches rely on elliptical waves propagation, they are more accurate for reproducing fire perimeter but they have greater time requirements. The most popular of this type of simulators is FARSITE [4]. Raster based approaches are based on a uniform grid in which fire propagates according to certain rules.

Some authors ([6], [7]) claim that the raster approach (including Cellular Automaton (CA) model) is more efficient than the vectorial approach. Vector implementation treats the fire perimeter as a closed curve, discretized through a number of points, each point spreads fire according its local conditions and fire propagation model. Then, the new fire perimeter is formed by the union of the outer shape of all individual fires. On the other hand the raster approach spread fire over a mesh of contiguous cells that can be inactive (not burning or burnt) or active (burning). Each cell has its own state and fire spreads from a cell to its neighbours based on a set of rules. It is interesting to note that raster approach matches with GPU execution model that uses a large amount of threads in a grid of threads in a SIMD (Single Instruction Multiple Data) model.

The use of GPUs shortened tremendously simulation times but also offers the capability of rendering the simulation outputs in simulation time. In this sense they are optimal to visualize fire spread in reasonable times, which allow the use of these tools for decision making and management purposes.

Visualization of fire spread offers a number of benefits, that includes the comparison of model output with fires occurred in the past, but also the evaluation of the consequences of implementing preventing measures like firebreaks, prescribed burns or other changes in the environment. Using visualization is also possible to train firefighters and communicate results to the community in an easier way. If the visualization is also interactive it allows to test fire behavior after thinking strategies that could be very dangerous to test directly in the field. FARSITE has a visual interface that shows the simulation in 2D [4] but more attractive and interactive spatial visualization tools were built in GPUs [5].

Fire propagation models, on one extreme, can rely only in physical principles, in which case we have the advantage of understand the mechanisms but the disadvantage that very detailed and difficult to obtain input information is needed [5]. On the other extreme there are pure statistical models based on datasets to predict fire propagation for a particular situation but these models are very specific. And in between both approaches we can design semi-physical models that rely in some physical principles but are fitted to some data sets that allow to validate the model [5].

In previous works [8], a High Performance Computing forest fire simulator was developed as a parallel cellular automata implemented in CUDA-C ([9, 10, 11]). Our simulator spreads the fire over a landscape taking into account several layers of topography, vegetation type and wind ([12, 8]). The model for fire propagation is a semi-physical model and takes into account as input the actual vegetation, slope, average wind intensity and direction. The simulator is calibrated for a given region comparing thousand of simulations with the real fire final perimeter. The search in the parameter space was done using a Genetic Algorithm (GA), a previously implemented methodology [13, 14, 15] that we programmed in parallel using CUDA-C. A Monte Carlo method was also used to find the best parameters but GA was proved to be more efficient. The results of this calibration step are exposed in [8].

We now present the development of a visualization tool using OpenGL ([16, 17]), built on top of the simulator. Once the calibration was performed and the propagation parameters were estimated, we can simulate and visualize 10 stochastic simulations in parallel in the order of fraction of seconds which is acceptable for the time needed to make decisions in real time [18]. In this contribution we will not discuss in detail the model fitting stage that was previously explained elsewhere [8]. The main contribution of this work is the design and implementation of a powerful visualization tool for fire propagation, based on the needs of firefighters in Patagonia, Argentina.

2 Forest fire spread model

A Cellular Automaton was developed to simulate fire spread. The landscape is represented as a grid of cells and fire spreads over this grid. Each cell has its own state: burnt, unburnt and burning (cells that can spread fire to their neighborhood). Fire spreads to neighboring cells according to the following probability:

$$p = \frac{1}{1+\exp(-(\beta_0+\beta_1 I_f+\beta_2\psi+\beta_3\omega+\beta_4\sigma))} \tag{1}$$

Where I_f , ψ, ω and σ, are related with the vegetation type, aspect, wind direction and slope respectively, as explained in [8, 12]. Fuel type coefficient β_0 is the baseline fire propagation probability for shrubland cells and β_1 is the difference between shrubland and forest. Parameters β_2, β_3 and β_4 modify propagation probabilities according to aspect (ψ), wind direction (ω), and slope (σ) respectively [12].

In order to determine if a target cell is reached by fire, the ignition probability is calculated taking into account the state of the 8-cells neighborhood using Equation 1 as explained in [8]. The following pseudocodes illustrate the main characteristics of algorithms (in order to improve clarity these algorithms does not show visual interface details):

Algorithm 1 Main loop

```
1: init all maps                         ▷ Read maps from hard disk
2: init input simulator parameters       ▷ $\beta_i$ values
3: while fire spreads do                 ▷ CA simulation steps
4:     parallel CUDA thread grid execution
5:     fire map update
6:     fire spread flag setting          ▷ If fire stops
7: end while
```

Algorithm 1 presents the main fire spread simulation loop, which is executed to perform a complete simulation. Firstly, input simulator parameters are initialized (Algorithm 1 lines 1 and 2). Then, while fire spreads, the loop block is executed: a CUDA thread grid is launched to perform each CA step. The thread grid is a 2D matrix of threads, with the same dimension of the raster maps. Then, one thread per map cell is launched and all cells of the new fire map are calculated concurrently (or parallel execution depending on the number of CUDA cores available on the graphic card). All threads execute the same function (CUDA kernel) in a SIMD way. In a more accurate analysis, when CUDA application is analyzed the model becomes to SIMT: Single Instruction Multiple Thread since each thread can take its own instruction trace.

Once the kernel function ends, the fire progress map is updated with the new fire state. Furthermore, if no cell changes to the burning state, then fire does not spread (if borders are reached or the fire can't propagate any more), and a flag is set *false* to stop the simulation.

Algorithm 1 presents a complete simulation, that is one of the options of our simulator. When user needs, the CA advances just 10 time steps. Section 4.2 will present this propagation possibility.

Algorithm 2 Thread$_{i,j}$ algorithm

```
1: thread position ← grid row i and column j
2: if $cell_{i,j}$ is burnable then
3:     for each neighbor $k$ of $cell_{i,j}$ do
4:         $PoI = f(p_k)$
5:     end for
6:     if $PoI >$ random value then
7:         $cell_{i,j} = burning$
8:     end if
9: end if
```

The Algorithm 2 illustrates the inner loop that is executed to spread the fire (line 4 Algorithm 1). First of all we verify that the target cell is burnable (Algorithm 2 line 2) and if this is the case, we proceed to evaluate the Probability of Ignition (*PoI* in line 4 of Algorithm 2) of the target cell produced by at least one of its neighbors. This probability of ignition (*PoI*) is computed evaluating Equation 1 for each of the k neighbors and combining all the contributions in the function represented by f in line 4 Algorithm 2, as explained in [8].

The target cell will be burned according to that probability (*PoI*) by throwing a random number as in line 6 of Algorithm 2. All is done in parallel and once all burning probabilities are computed by each of the threads the whole fire map is updated at once.

The above explained simulation procedure is done millions of times to fit the parameters of the model to the real fire data. This fitting step is done in order to find the best set of simulator parameters (β_i values in Equation 1) [8].

The fitting procedure is done by minimizing the distance, or fitness, between real fire and simulation. The fitness is computed as the number of burned cells that have in common both, simulation and real fire. Fitness equal to zero indicates that the fire after simulation is identical to the real fire. The search of the ensemble of parameters β_i that best fit the model to data, was done using a Genetic Algorithm (GA). This search strategy consists in changing randomly some β_i, perform a simulation and compute the corresponding fitness. The β_i values are changed by performing selection, crossover and mutation between different ensembles, after every change a simulation is performed and the fitness of the simulation is calculated comparing with real data.

In addition to GA, a Monte Carlo method was implemented, where β_i values where chosen from the valid range of parameter values simulating a brute force method. The GA converges quickly to good results [8].

The fitness values are ranked in decreasing order and the corresponding β_i are therefore ranked together with their fitnesses. A histogram of the best ranked individuals allows to determine the best values of β_i and the associated errors, as explained in more detail in [8].

After the fitting procedure we obtain a best set of β_i parameters for each of the Patagonia regions presented, i. e. *Falso granítico* and *Laguna Seca*. Is with those parameters that we will simulate and visualize several simulations to have a measure of fire extension and propagation. As the simulations are stochastic, the use of the best set of parameters will give rise to different fire scars.

3 Simulator inputs and outputs

Simulator inputs are several maps that describe the landscape (terrain vegetation, slope, aspect and elevation), as well as wind speed and direction.

Fire environment is described by several raster maps that represent the area of interest as a grid of cells. Each raster map has the description of the area that represents: number of rows and columns, cell sizes, and coordinates of one corner. Using this information, each map can be georeferenced using GIS software (including Google Earth).

For topography description DEM (Digital Elevation Model) maps were used. Nowadays several satellite and radar information is available from different data sets on Internet. For example, USGS Earth Explorer offers maps of several sources and types: LandSat, Sentinel, ASTER global DEM collections for example, with high temporal and spatial resolution [19]. This information is freely available and we are using DEM raster maps from this site. Therefore using GIS tools, slope and aspect maps can be obtained from the elevation raster map. The elevation map is used to visualize the topology of the area of interest. Slope and aspect maps are inputs for the fire spread model.

Fuel type information, as described on [1, 2, 3], are specific combinations of vegetation types that together define the fire behavior. For Argentina, fuel types are still not characterized as in [1] and [2]. As a surrogate of fuel type we will use vegetation type. Combining vegetation type with fire behavior is a very important task to be accomplished, because fuel type often determines fire behavior (specially when nor slope neither wind are strong enough to drive fire propagation).

Our simulator uses wind velocity and direction for fire spread computation. However for the moment we are using average quantities and it would be a future task to include wind variability and ultimately coupled wind models [18] to allow a more accurate real-time prediction of fire propagation.

At the moment, we are working with 2 real fires occurred at the Northwestern Patagonia Andean region (Figure 1). This Figure shows estimated ignition points of both real fires. The first test case is called *Falso Granítico* fire, that occurred in 1999, in the *Falso Granítico* Hill proximity (at 41°21′59″ S, 71°38′46″ O). Raster maps of 801x801 cells were used for this case. The second test case corresponds to a fire occurred in 2012 near to *Laguna Seca* (at 41°02′35″ S, 71°16′36″ O). Raster maps of 181x423 cells were used for this case. We constructed slope and aspect raster maps from DEM information. In both cases, map resolution is 30x30m.

Figure 1 shows different vegetation types. More detailed vegetation cover classification for Patagonia Andean region is available in [20] (updated in 2016).

In Figure 1 vegetation classification corresponds to: forest type A where predominates native forest (lenga, coihue, cypress), forest type B where predominates low vegetation (lenga, ñire, and mixed woods) and forest type I to indicate exotic forest. Grassland includes steppe, marshland and wetland. Finally, shrubland includes shrubland (native and exotic) and infrastructure areas (to avoid more vegetation type divisions). No fuel includes rocks, lakes, bare ground, ice and snow.

In order to feed our simulator we transform this data to three vegetation types. At the moment our simulator considers three fuel types: shrub, forest and no fuel. Including more accurate and real vegetation types is one of the most important open lines in a near future.

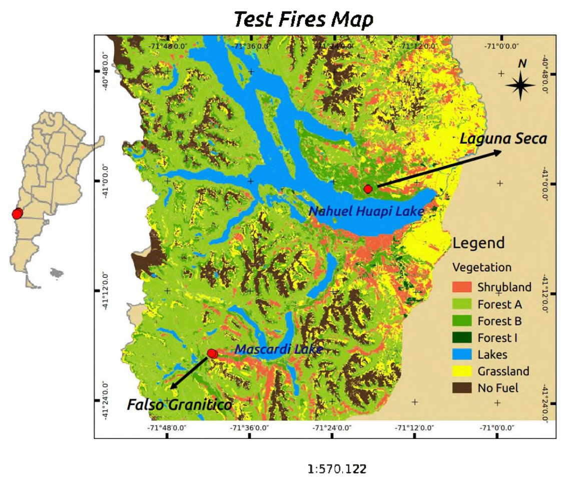


Figure 1: Area of interest situated at the Nahuel Huapi National Park in the North West Patagonia. Arrows indicate the two ignition points of *Laguna Seca* and *Falso Granítico* fires.

The simulator output is a raster map where each cell is labeled according to the resulting probability of ignition after the execution of 10 simulations (this number can be changed by the user and is limited by the memory of the graphic card). The output map can be georeferenced using GIS, given that meta data is copied from the input raster maps.

4 Forest fire simulator

Our application has the ability to perform high performance simulations of the spread of wild fire in a very complex environment. In addition it incorporates a powerful visualization platform, including an intuitive interface for user interaction, to command fire progress.

Most of the times, the visualization of a problem helps to understand the behavior of a complex phenomena. Our simulator shows fire progress over the landscape, where the action of topography, vegetation and wind can be tested and modify. For instance, we provide a helpful user interface, where firebreaks can be easily set, fire can move forward and backward, the ignition point can be changed by the user, etc.

OpenGL ([16, 17]) was chosen to program the simulator visualization and user interaction. Therefore, fire environment visualization and fire progress are rendered by graphic cards, as well as user interaction (with mouse and keyboard). Finally as previously mentioned, CA for fire spread simulation is also executed in graphic cards. In summary, simulation fire progress, visualization in real time and user interaction are all solved by GPUs, that deal with all these high performance requirements in a successful way.

Next paragraphs will expose visualization and afterwards the most relevant tools of our simulator. We will explain important simulator characteristics, as well as design details and some implementation decisions.

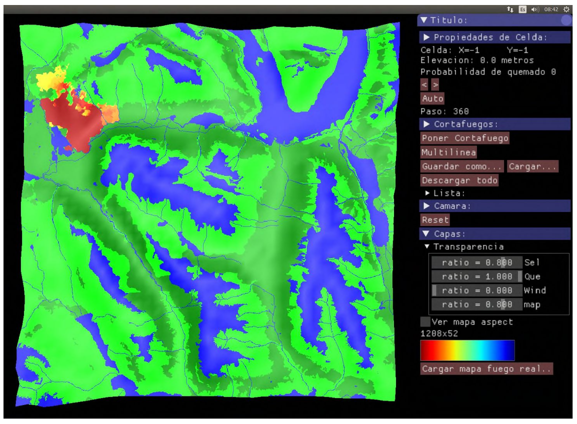
4.1 Forest fire progress visualization

Figures 2 and 4 show our application main window. The left panel shows the fire progress and the ignition probability of cells. The ignition point was arbitrarily set by the user. Ten simulations are performed in parallel starting from exactly the same ignition point. Given that the model is stochastic every realization is different from the previous one. While performing 10 simulations in parallel the burning probability of each cell is determined. If a given cell was burnt in all of the simulations, the burning probability is one, but if only some of the times the cell with coordinates (i, j) burns, the probability will be:

$$bp_{i,j} = \sum_{1}^{10} \frac{Times\ burns}{10} \tag{2}$$

According to Equation 2 if a given cell burns in all the simulations then bp=1. We colored the final simulation according to this probability, being red for bp=1 and more yellowish when this probability diminishes.

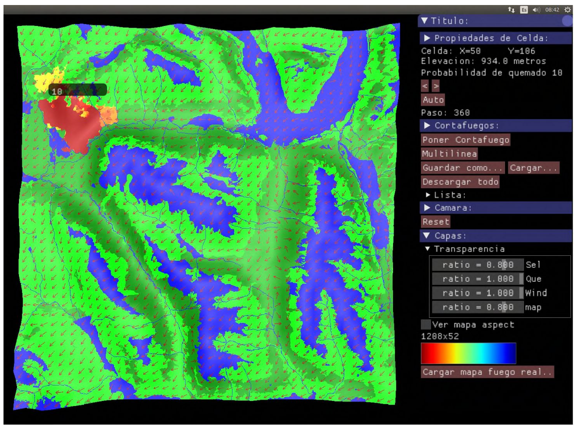
When the mouse pointer moves over the map cells, a label with the ignition probability information is shown. This probability is the one displayed in Equation 2, i. e. the number of times that this cell burned through the 10 simulations. As the model is stochastic, the simulations are different from each other. Figure 3



```
▼ Titulo:
▶ Propiedades de Celda:
Celda: X=-1      Y=-1
Elevacion: 0.0 metros
Probabilidad de quemado 0
< >
Auto
Paso: 360
▶ Cortafuegos:
Poner Cortafuego
Multilinea
Guardar como... Cargar...
Descargar todo
 ▸ Lista:
▶ Camara:
Reset
▼ Capas:
 ▾ Transparencia
   ratio = 0.800  Sel
   ratio = 1.000  Que
   ratio = 0.000  Wind
   ratio = 0.800  map
Ver mapa aspect
1208x52
Cargar mapa fuego real..
```

Figure 2: Top view of the landscape: burnable cells (green) correspond to different types of vegetation, unburnable cells (blue) are lakes, rocks, etc. Fire is showed in red (high burning probability) and yellow (lower burning probability).

shows the same simulator top view where the mouse pointer is labeled with cell fire probability. This Figure includes the wind layer (arrows above the terrain). This simulator feature will be explained later.



```
▼ Titulo:
▶ Propiedades de Celda:
Celda: X=50      Y=106
Elevacion: 934.0 metros
Probabilidad de quemado 10
< >
Auto
Paso: 360
▶ Cortafuegos:
Poner Cortafuego
Multilinea
Guardar como... Cargar...
Descargar todo
 ▸ Lista:
▶ Camara:
Reset
▼ Capas:
 ▾ Transparencia
   ratio = 0.800  Sel
   ratio = 1.000  Que
   ratio = 1.000  Wind
   ratio = 0.800  map
Ver mapa aspect
1208x52
Cargar mapa fuego real..
```

Figure 3: Top view of the landscape: arrows represent average wind direction and arrow colors represent wind intensity. More wind (red) and less wind (blue). Fire scar is shown in red/orange/yellow according to the probability that cell burns after ten simulations. Higher probability (red) lower probability (yellow).

The number of parallel simulations to be performed can be changed. Input maps are loaded once in GPU global memory. These maps are read by kernel threads through each of the simulations. However each of the parallel simulations need current fire map and a new fire map for labeling each cell that is reached by fire. Therefore, these two maps (with fire spread state) are allocated for each of the 10 simulations since vegetation, slope, aspect, wind maps are allocated just once (they are read only input maps). Our application takes care of the use of GPU memory.

The visualization of fire progress is updated every 10 cellular automaton steps. This value can be changed, but we could see that this value is useful for clarity. Furthermore, every 10 steps fire maps are saved in order to go back and reproduce past states of the fire. For improving application efficiency just the coordinates of the ignited cells are stored. In this way past fires can be easily recovered. Then, fire can advance or rewind as the user needs for simulation observation. This simulator feature in combination with firebreak setting are very important and powerful tools for firefighters and other users. Two buttons on the main menu and two keys allow to command fire progress forward and backward.

Map view (left panel on the simulator Figures) can be changed as user need: top view is the default view (Figure 2 and Figure 3), then user can zoom in and out the map, and drag it to view adjacent areas. Furthermore, the map can also be rotated to modify this view.

This actions are performed on the graphic card. Figure 5 and Figure 6 show different views of the same map (*Falso Granítico* fire area). Figures5 and Figure 6 show the left top map corner zoomed in and rotated. Terrain elevation is accentuated in these last figures. Each time, the default top view can be set with the Camera Reset button.

The performance of visualization and user interaction is excellent. For example due to design and implementation characteristics, setting firebreaks and changing map view (zoom, drag and rotation) needs lot of computation for updating fire map. We can see that GPUs solve this requirements in a very successful way. We tested our simulator in different computers and graphic cards and it executes with no delays both computation and interaction.

Next paragraphs present the simulator tools implemented until now and in the section open lines we will explain the additional functionality that is planned to be added to our simulator.

4.2 Interaction tools

Most of the simulator options and functionality is controlled with the right panel menu. Figure 4 shows this main panel with the most important interaction tools classified in different sections.

First menu section shows cell features: the user can move the mouse pointer over the map and the most important cell characteristics are shown: cell coordinates, terrain elevation and burning probability (if cell was reached by fire).

Then, two buttons appear with $<$ and $>$ symbols. This buttons allow the user to select to go forward or backward the fire progress. This functionality is available with $\leftarrow$ and $\rightarrow$ keyboard buttons also. As mentioned, 10 cellular automaton steps are performed each time the $>$ or $\rightarrow$ buttons are pressed. The Auto button executes CA steps until the simulation is completed (process showed on Algorithm 1).

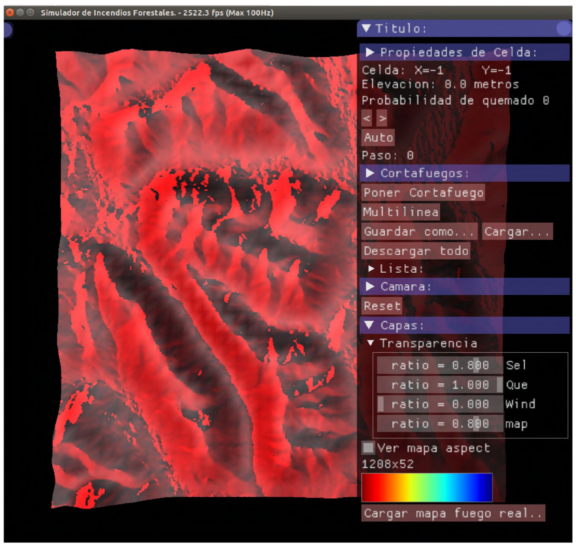


Figure 4: Simulator main menu. In this case the aspect map view is shown: this view emphasize slope orientation.

Next menu section defines firebreak functionality. This panel section allows the user to set firebreaks over the landscape. That means that these parts of the terrain are set as nonburnable cells. Once the firebreak button is selected, the mouse is used to draw the firewall.

The user can set line firewalls as well as multiline firewalls (freehand drawing). Figures 5 and 6 show the same time step of the CA without firebreak and when a firebreak was set respectively. When a firebreak is set, fire doesn't spread to the north-west (in this case) of the terrain.

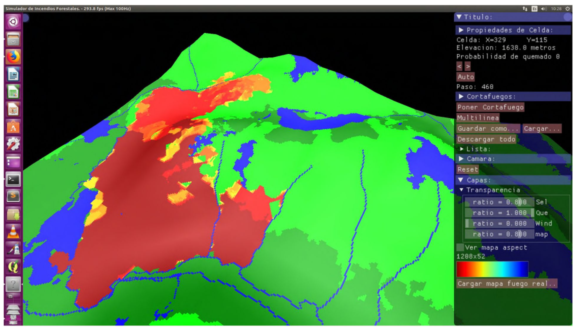


Figure 5: Top left corner of previous figure zoomed and rotated. Cellular Automaton was executed 460 times. No firebreak was set.

Firewalls can be stored in files and they can be loaded at any time. The combination of setting firewalls and choosing when fire goes forward or backward allows the user to understand where firewalls are more efficient, in combination with topography, vegetation, etc. This is a powerful capability of our simulator.

Firebreaks are limited by raster cells shape. Fire

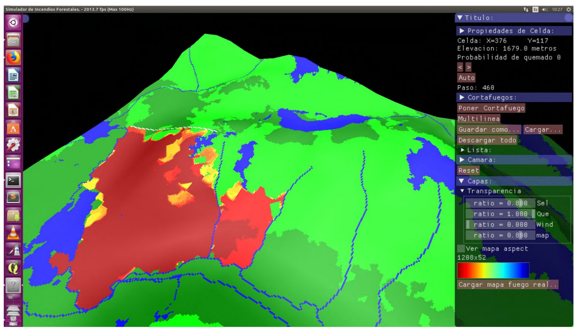


Figure 6: Same simulation step than the previous figure but with a firebreak (white line) at the top left corner. The fire does not spread to the north of the terrain. Cellular Automaton was executed 460 times.

propagation through a diagonal firebreak is avoided calculating when fire has to spread or not when target cell is adjacent to a firebreak. For example, authors of [5] mention the same problem and they decided to modify the thickness of the firebreak line. Firebreak thickness was set to twice the size of a single cell. Our approach maintains the thickness of firewall but calculating the interaction of the fire with the firebreaks diagonal cells.

The next section manages camera options. As already mentioned, fire area can be zoomed, rotated, relocated and simulator window can be reshaped. Camera Reset button allows the user to set the default top view (Figure 2).

The last menu section allow to manage several raster maps as information layers that can be visible or not, depending on their transparency. Wind layer, fire layer and firebreaks layer can be turned on and off by changing layer level of transparency. Using transparency sliders, the users can change which information to visualize, depending on their needs. If aspect terrain information is useful then the user can check this option (this check box is selected in Figure 4).

More layers are planned to be included. For example, wild fires final burnt area can be obtained from aerial images. Then, using GIS tools (digitalization process) we can obtain vectorial or raster maps with the final burned area. At the moment our team is working in order to include this final fire map as a simulator layer. As other layers, user will be able to set its transparency in order to visualize this map in combination with other layers. This will be a powerful tool for testing firebreaks and simulations accuracy.

Wind speed and direction is showed as a wind layer. User can manage layer transparency to see wind features. This information is presented by colored arrows. Each arrow shows the direction and intensity of wind.

Wind intensity is calculated using a color ramp: blue are minimum velocities and red are higher velocities. The color ramp is included in the main menu. Arrow color is calculated for each map using an equalized histogram for wind speeds. Therefore, less frequent values are ignored and most frequent values are used for speed values discretization.

Wind direction is represented by arrows. Each arrow represents the average of a group of cells values. Depending on the map resolution, if each of the wind cell is represented by a unique arrow, the visualization gets confused specially when combined with other layers. Therefore, we choose to divide the number of cells proportionally to the map dimension size.

Border cells were a particular difficulty for visualization of wind arrows. As they have less than 8 neighbors, the average was calculated taking into account the available neighbors.

4.3 OpenGL programming

When the simulator is launched all fire environment maps are read from the hard disk. The OpenGL functionality is also initialized: several functions (callbacks) are registered to OpenGL in order to solve the simulator window display, keyboard button pressing (down or up), mouse motion (active or passive) and simulator window reshape.

To implement the simulator main menu, the library ImGUI (Immediate Modal Graphical User Interface [21]) was used. ImGUI is a graphical user interface library for C++. This library is particularly suited for integration in game engines (for tooling), real-time 3D applications, full screen applications, embedded applications, etc. This library offers the implementation of menu buttons, panels, lists, checkboxes, radio button, labels, sliders, etc.

The simulator main window shows fire progress map. This map is an OpenGL texture calculated from different bitmaps. Our simulator manages 4 bitmaps: vegetation map, fire breaks map and fire map. These bitmaps have the same fire maps dimensions (rows x columns). Each bitmap pixel corresponds to a map cell. Each pixel is a (r,g,b,alpha) tuple, where r, g and b form pixel color and alpha is the transparency of the pixel. Using these 3 bitmaps a final bitmap is formed (this is the texture that the simulator passes to OpenGL to draw each frame).

There are some events that cause final texture update. This texture update will be used during next window frames actualization. Different operations cause texture regeneration: fire spread changes (moving forward or backwards), line or multiline firebreak setting, saved firebreak loading, layer slider controls settings, etc.

When a texture update occurs, bitmaps are copied to GPU, then, a CUDA kernel is launched in order to compute final texture, pixel colors and transparencies, and this new texture is used to update the fire progress map on the following frames.

In a low level of implementation all the visualization is solved by OpenGL vertexes, triangles and indexes. Indexes are converted to bitmap UV coordinates to form textures. Camera and light position (ambiance light and spectral light), near and far planes, zooms, dragging and rotation of the figure are solved by low level matrices that multiply vertexes or are used to define colors and transparencies.

In order to calculate map zooms, dragging, rotations, etc, some transformation matrices are used and joined with the map model. For example, if user has zoomed in and rotated the image, 4x4 matrices for this operations are multiplied and the result is saved in the transformation matrix. Then, this transformation matrix is used to multiply each vertex to obtain the new vertex values.

These operations are optimized in GPGPU. The simulator response time and user interaction are solved in a very successful way in CUDA-C and OpenGl.

5 Conclusions and Open Lines

A visualization tool for fire propagation was developed in OpenGL on top of a forest fire simulator previously developed in CUDA-C. We presented a useful computational tool to be used for fire management, training and communication. It was developed on graphic processing units to meet the requirements of a high performance real time application, with a friendly graphical interface to be use for firefighters.

Models based more on the physical mechanisms of fire propagation, such as the physical-statistic model defined by Rothermel will be implemented in the near future. This will help to better understand the mechanisms involved in fire propagation in our region. To this aim it's necessary to characterize the fuel types for Patagonia, a task that should be in the agenda of natural resources management.

To improve prediction, it will be necessary to include the wind variability both in direction and intensity in real time. Another possibility will be to couple a wind simulator to the model simulator. This would be a very challenging task because the whole simulator will have to accomplish with the real time requirements.

Our model includes three vegetation types, but we have access to more detailed vegetation maps that should be conveniently transformed to raster maps, to be included in the model.

Another open line is to fit the advance of the fire front. In previous works we fitted our model to the final fire scar but intermediate fires scars can be obtained using satellite images or will be provided by our collaborators at ICE.

One of the useful features of our previous work was the possibility to determine the ignition point with its corresponding uncertainty. The ignition point could be caused by natural factors, like thunders or by anthropic causes which is mostly the case in our region. The possibility to visualize the ignition point if its unknown, will probably help to get clues about the fire origin.

Other feature to be included into the visualization interface will be a sensitivity analysis of the parameters.

Until now we are visualizing the output of 10 simulations performed with the best set of parameters. However with our fitting procedure we can determine an error associated with those parameters. Changing one of the parameters in its error range, but fixing the rest in the best values, it is possible to measure how sensible are the simulations of fire propagation to a change of that parameter. In this way we can add to the visualization panel interface, some slider bars representing the possible variability range of each parameter. With this tool, the user can easily explore the sensitivity of simulations when parameters are changed, generating less probable but also possible fire scenarios.

6 Acknowledgements

We acknowledge Marcelo Bari and collaborators from ICE (Departamento de Incendios, Comunicaciones y Emergencias del Parque Nacional Nahuel Huapi, APN: Administración de Parques Nacionales). For helpful comments on maps and GIS we thank Gabriela Denham. M. Denham and K. Laneri are members of Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina. M. Denham and K. Laneri are part of the project TIN2014-53234-C2-1-R of Ministerio de Ciencia e Innovación (MICINN-Spain). M. Denham, K. Laneri and S. Waidelich are part of the project PI-UNRN "Aplicación de Alto Rendimiento para Predicción del avance del fuego en Incendios Forestales".

References

[1] J. H. Scott and R. E. Burgan, "Standard fire behavior fuel models: A comprehensive set for use with Rothermel's surface fire spread model," tech. rep., United States Department of Agriculture. Forest Service. Rocky Mountain. Research Station, 2005.

[2] E. H. Anderson, "Aids to determining fuel models for estimating fire hehavior," tech. rep., United States Department of Agriculture. Forest Service, 1982.

[3] S. Taylor, R. G. Pike, and M. E. Alexander, *Field Guide to the Canadian Forest Behaviour Prediction (FBP) System*. Canadian Forest Service, 1996.

[4] M. A. Finney, "Farsite: Fire area simulator - model development and evaluation," tech. rep., USDA Forest Service, 2004.

[5] R. V. Hoang, M. R. Sgambati, T. J. Brown, D. S. Coming, and F. C. Harris, "VFire: Immersive wildfire simulation and visualization," *Computers & Graphics*, vol. 34, no. 6, pp. 655 – 664, 2010.

[6] T. Ghisu, B. Arca, G. Pellizzaro, and P. Duce, "A level-set algorithm for simulating wildfire spread," *CMES Computer Modeling in Engineering and Sciences*, vol. 102, no. 1, pp. 83 – 102, 2014.

[7] T. Ghisu, B. Arca, G. Pellizzaro, and P. Duce, "An improved cellular automata for wildfire spread," *Procedia Computer Science*, vol. 51, no. Supplement C, pp. 2287 – 2296, 2015. International Conference On Computational Science, ICCS 2015.

[8] M. Denham and K. Laneri, "Using efficient parallelization in graphic processing units to parameterize stochastic fire propagation models," *Journal of Computational Science*, vol. 25, pp. 76 – 88, 2018.

[9] T. M. John Cheng, Max Grossman, *Professional CUDA C Programming*. Wrox, 2014.

[10] S. Cook, *CUDA Programming. A developer's guide to parallel computing with GPUs*. Morgan Kaufmann Publishers Inc., 2013.

[11] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2010.

[12] J. M. Morales, M. Mermoz, J. H. Gowda, and T. Kitzberger, "A stochastic fire spread model for north patagonia based on fire occurrence maps," *Ecological Modelling*, vol. 300, no. 0, pp. 73 – 80, 2015.

[13] M. Méndez Garabetti, G. Bianchini, M. L. Tardivo, P. Caymes Scutari, and G. V. Gil Costa, "Hybrid-parallel uncertainty reduction method applied to forest fire spread prediction," *Journal of Computer Science and Technology*, vol. 17, pp. p. 12–19, Apr. 2017.

[14] M. Denham, "Dynamic data driven application for forest fire spread prediction," *Journal of Computer Science and Technology*, vol. 12, pp. p. 84–86, Aug. 2012.

[15] M. Denham, *Predicción de la Evolución de los Incendios Forestales Guiada Dinámicamente por los Datos*. PhD thesis, Universidad Autónoma de Barcelona, 2009.

[16] E. Meiri, "OGL. Modern OpenGL Tutorials."

[17] E. Luten, "OpenGL Book."

[18] G. Sanjuan, T. Margalef, and A. Cortés, "Applying domain decomposition to wind field calculation," *Parallel Computing*, vol. 57, pp. 185 – 196, 2016.

[19] U.S. Department of the Interior, "USGS: U.S. Geological Survey. Earth Explorer."

[20] CIEFAP-MAyDS, "Actualización de la Clasificación de Tipos Forestales y Cobertura del Suelo de la Región Bosque Andino Patagónico. Informe Final," tech. rep., Centro de Investigación y Extensión Forestal Andino Patagónico, 2016.

[21] O. Cornut, "Immediate mode graphical user interface."