Frontier: A Digital Terrain Data Generator

Remy Oukaour, 107122849
CSE 528, Stony Brook University

1. Introduction

Representing terrain in a digital form has many applications: movies, computer games, scientific simulations, architectural models, and others all work with digital terrain. Some of this data is sourced from the real world, such as the U.S. Geological Survey's effort to map the Earth via satellite radar [1]. Other terrain is designed by hand or algorithmically generated.

In all cases, it is desirable to have large amounts of sufficiently realistic terrain. Surveys to gather real-world data are expensive and may create incomplete datasets, while human and algorithmic designers may not capture plausible terrain features. The goal of Frontier is to address these problems by filling in missing data, extending existing data to cover more area, and using physical simulations to recreate landscape features in artificial data.

2. Compilation and Execution

The files used by Frontier are organized into the following subdirectories:

- `bin`: Contains the compiled `frontier.exe` binary executable.
- `doc`: Contains this documentation.
- `ide`: Contains the Visual Studio project files needed to build Frontier.
- `include`: Contains C++ header files used by external libraries.
- `input`: Contains DTED files to be opened and processed.
- `lib`: Contains compiled library files.
- `output`: Intended to contain saved images.
- `res`: Contains resources (specifically icon images) used by the user interface.
- `src`: Contains the source code.
- `tmp`: Contains temporary files created during compilation.

Frontier depends on the Fast Light Toolkit (FLTK) library, available at http://www.fltk.org.

To compile Frontier, open the Visual Studio solution file `ide\Procedural Terrain.sln` and build it in Release mode for the x64 architecture.

To run Frontier after compiling it, start `bin\x64\Release\frontier.exe`.

3. Input File Format

Frontier reads digital terrain elevation data (DTED) as PNG images. The red channel stores the elevation, the green channel stores the hardness, and the blue channel stores the solubility. (Hardness and solubility are two parameters used by the erosion algorithms.) If the alpha channel is completely transparent, the point is treated as missing. Each channel varies from 0 to 255, but these are converted to floating-point values from 0 to 1 for better precision during processing.

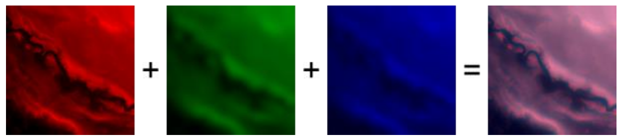


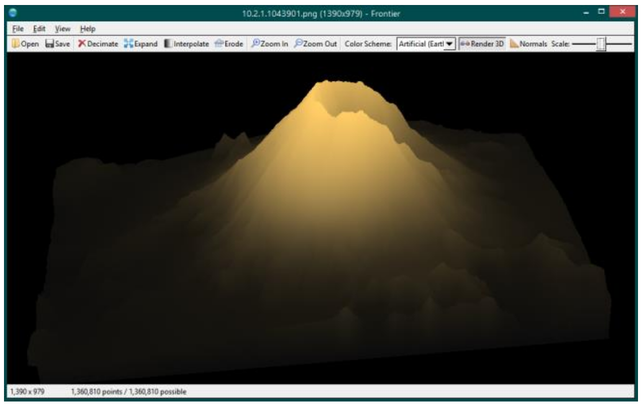
Figure 1: Combining elevation (red), hardness (green), and solubility (blue) channels to get a single DTED image.

DTED images are commonly found in black and white, making hardness and solubility identical to elevation. If this is not desired, the images can be edited beforehand.

3.1. Digital Terrain Elevation Data (DTED) Files

DTED files come in many different formats. Some common ones are USGS Digital Elevation Model files (.dem), USGS Spatial Data Transfer Standard files (.sdts), and Shuttle Radar Topography Mission files (.hgt). Writing parsers for all of these formats would be beyond the scope of this project, so the Geospatial Data Abstraction Library (GDAL) is used to convert these formats into .png files [2].

4. User Interface



10.2.1.1043901.png (1390x979) - Frontier

Color Scheme: Artificial (Eartl

1,390 x 979 1,360,810 points / 1,360,810 possible

Figure 2: The Frontier user interface.

Frontier uses a Windows-standard GUI with a menu bar, a toolbar, a workspace, and a status bar.

When a DTED file is opened, it appears in the workspace. The status bar shows its dimensions, as well as how many of its points have known elevations. If a new file is created, it starts out empty but can be filled by the interpolation function.

In the default 2D rendering mode, dragging the terrain with the mouse pans, and scrolling the wheel zooms. In 3D mode, dragging with the left mouse button rotates, so using the middle or right button pans. The 3D mode also has a Scale slider on the toolbar to control the elevation appearance. In some cases, the normal vectors need to be refreshed manually with the Normals button or menu item.

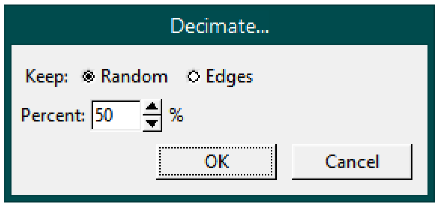
Six color schemes are available for the terrain. Elevation can be viewed in grayscale (the default); each color channel can be viewed separately to visualize the properties they encode; the three channels can be viewed together to resemble the source image; or an artificial earth-tone color scheme can be applied.

5. Program Functions

Frontier has functions to generate artificial terrain and process it to better resemble real terrain, or to fill in and extend incomplete real data.

5.1. Decimation

Decimation removes a fraction of points from the terrain, emulating the occurrence of “holes” in DTED recorded from the real world. The removed data can then be filled in via interpolation.



Decimate...

Keep: [✓ Random] Edges

Percent: 50 %

Figure 3: A dialog showing the user-adjustable decimation parameters. Higher percentages remove more points.

Points can be picked randomly, or edges can be preferentially kept. The percentage of points to remove is chosen by the user.

Edge detection is done using the Sobel operator, convolving the elevation data with two kernels:

$$\boldsymbol{G_x} = \boldsymbol{E}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad \boldsymbol{G_y} = \boldsymbol{E}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\boldsymbol{G} = \sqrt{\boldsymbol{G}_x^2 + \boldsymbol{G}_y^2}$$

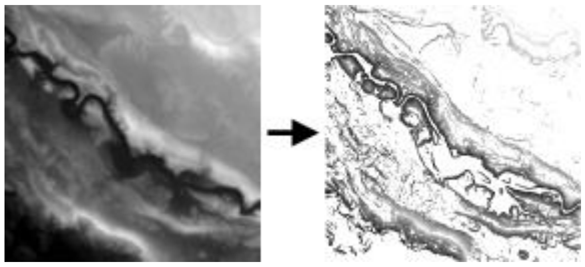
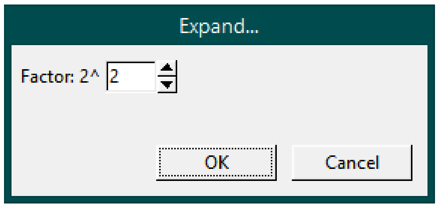


Figure 4: The result of decimating some terrain, keeping its edges.

5.2. Expansion

Small pieces of terrain can be expanded to cover a greater area. The expanded area can then be filled in via interpolation.



Factor: 2^ 2

Figure 5: A dialog showing the user-adjustable expansion parameters. Higher factors expand more.

Expansion spreads points apart by a factor of 2^n, where n is a parameter chosen by the user.

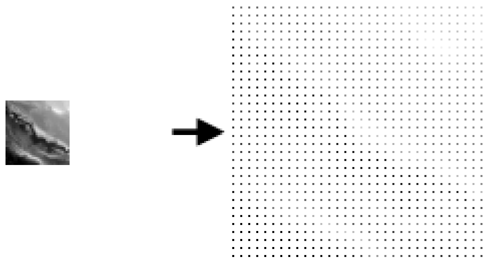
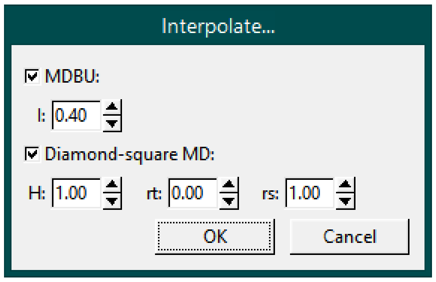


Figure 6: The result of expanding a piece of terrain by a factor of $2^2 = 4$.

5.3. Interpolation

Interpolation fills in gaps left by decimating or expanding terrain, or by incomplete recording of real-world data, using the Morphologically Constrained Midpoint Displacement algorithm [4], an extension of diamond-square MD with a constraining bottom-up (MDBU) process.



☑ MDBU:

I: 0.40

☑ Diamond-square MD:

H: 1.00 rt: 0.00 rs: 1.00

Figure 7: A dialog showing the user-adjustable interpolation parameters. *I* affects MDBU; *H*, *rt*, and *rs* affect MD.

Diamond-square MD [3] starts with four corner points, with elevations already known or assigned randomly. The average of all four, plus a random displacement, is used as the elevation of the central point (the "diamond" step). Then the average of each corner pair and with the central point, plus a random displacement, is used as the elevation of each edge point (the "square" step). (This implementation is capable of handling arbitrary rectangles, not just squares, at the cost of performance.) The process then recurses along each of the four resulting squares until all points have their elevations known.

Displacements are based on the recursion depth n and on the parameters chosen by the user:

$$\delta = (\mathcal{U}(0,1) + rt) \times rs \times 2^{-nH}$$

The parameters *rt* and *rs* translate and scale the random quantity, respectively. *H* affects the noisiness, with values closer to 0 being noisier.

By itself, diamond-square MD can generate elevations that do not match with known points when those points are only reached after deep recursion. The MDBU process propagates the deep points' elevations to points that are reached sooner in the MD process. MDBU starts by labeling all the points with known elevations as "active;" all the rest are "idle." For every idle point p that is a parent of at least one active point, its elevation is calculated based on its children $C(p)$:

$$e_p = \frac{1}{|C(p)|} \sum_{q \in C(p)} e_q + \Delta(e_q, \text{distance}(p,q))$$

Here e_p is the elevation of p. The Δ function interpolates the elevations of each child point q by their distance from p:

$$\Delta(e,d) = e\left(1 - \text{sgn}(I)\left(1 - \left(1 - \frac{d}{d_{max}}\right)^{|I|}\right)\right)$$

The parameter I tunes the interpolation. Negative values are steeper towards the child points, positive values towards the parent point, with values further from 0 increasing the suddenness of the disparity.

Active points are labeled "processed" after being used to interpolate their parents, while idle points are labeled "active" after being assigned an elevation. Then MDBU repeats until no points are active.

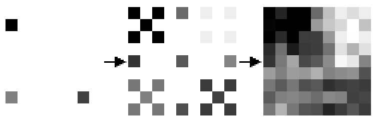
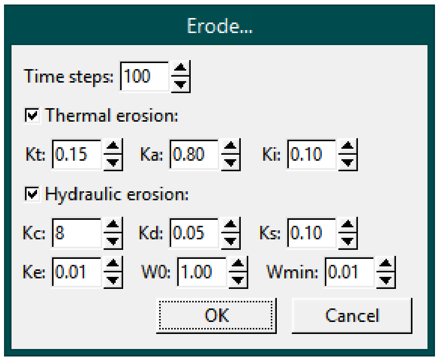


Figure 8: The result of interpolating four known elevations in a 9×9 grid with MCMD: first MDBU, then MD.

5.4. Erosion

Erosion simulates physical processes on the surface of Earth that alter terrain, generally by reducing peaks and smoothing rough areas. It can make artificial terrain better resemble real landscape formations, or it can be used to extrapolate how real terrain will be shaped by erosion forces.



Erode...

Time steps: 100

☑ Thermal erosion:

Kt: 0.15 Ka: 0.80 Ki: 0.10

☑ Hydraulic erosion:

Kc: 8 Kd: 0.05 Ks: 0.10

Ke: 0.01 W0: 1.00 Wmin: 0.01

Figure 9: A dialog showing the user-adjustable erosion parameters. K_t, K_a, and K_i affect thermal erosion; K_c, K_d, K_s, K_e, W_0, and W_{min} affect hydraulic erosion.

Erosion processes are calculated for every elevation point, and repeated for a number of time steps chosen by the user.

5.4.1. *Thermal Erosion*

Thermal erosion involves any physical phenomena other than the water cycle. The original algorithm [5] models the process of steeply sloped rocks cracking due to thermal expansion and crumbling at their sides. An extension to it [8] allows for more precise tuning.

Each point p has a talus angle T, calculated as $\tan^{-1}(K_a h_p + K_i)$. Here h_p is the hardness of p, and K_a and K_i are the talus angle coefficient and bias, respectively. If the elevation difference d between p and a neighbor q forms an angle α steeper than T, then an amount $K_t(1 - h_p)d_{max}/2$ of material is distributed from p to q, where K_t is the thermal erosion rate and d_{max} is the maximum elevation difference between p and its neighbors. This is done proportionally for all the neighbors of p below the talus angle.

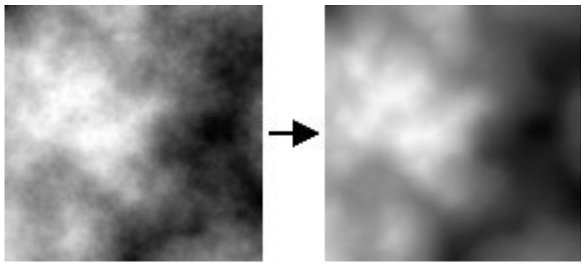


Figure 10: The result of running thermal erosion for 1,000 time steps on diamond-square–generated terrain.

5.4.2. *Hydraulic Erosion*

Hydraulic erosion involves the cycle of water as it rains onto terrain, flows downhill, and gradually evaporates. The original algorithm [5] models the behaviors of flowing water to dissolve terrain into sediment and to deposit sediment at lower areas. Extensions to this algorithm improve its accuracy and physical basis [6][7].

In addition to elevation, each point stores amounts of water and sediment. To begin the erosion process, rain is distributed on each point p as $W_{min} + W_0 e_p$, where e_p is the elevation of p and W_{min} and W_0 are the rainwater minimum and coefficient, respectively.

Each point p calculates the elevation difference d and the water amount difference f between itself and each neighbor q. If no neighbors are below p, any sediment suspended at p is deposited as new terrain material. For any neighbors below itself, the water at p flows proportionally to each one.

An amount of water Δw flowing from p to q has a sediment capacity c of $K_c \Delta w$, where K_c is the sediment capacity coefficient. If there is more sediment at p than the water can carry, it transports as much as it can to q, and the excess sediment is deposited on p proportional to K_d, the deposition coefficient. If the water has excess capacity, some of the terrain material at p is dissolved to fill the excess, proportional to $K_s s_p$, where s_p is the solubility of p and K_s is the dissolving coefficient.

After each time step of hydraulic erosion, water at each point is evaporated proportional to K_e, the evaporation coefficient.

6. Conclusion

Frontier is capable of stretching small DTED over a larger area, filling in gaps or generating entirely new DTED with a constrained midpoint displacement algorithm, and eroding the generated terrain for increased realism. It allows results to be visualized in both 2D and 3D, with color schemes that highlight different properties of the terrain.

A potential area for future work is to implement its more time-consuming algorithms on the GPU, as was done by some of the referred authors [7][8]. Increased physical realism is also desirable, although a higher-fidelity erosion simulation may not be the best way to achieve this.

7. References

[1] United States Geologic Survey. 2013. "The National Map." Retrieved on November 15, 2013 from http://www.nationalmap.gov/.

[2] Open Source Geospatial Foundation. 2013. "GDAL Utilities." Retrieved on November 15, 2013 from http://www.gdal.org/gdal_utilities.html.

[3] M. A. Pumar. 1996. "Zooming of Terrain Imagery Using Fractal-Based Interpolation." In *Computers & Graphics, Vol. 20*.

[4] F. Belhadj. 2007. "Terrain Modeling: A Constrained Fractal Model." In *AFRIGRAPH '07 Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*.

[5] F. K. Musgrave, C. E. Kolb, and R. S. Mace. 1989. "The Synthesis and Rendering of Eroded Fractal Terrains." In *Computer Graphics, Vol. 23*.

[6] B. Benes and R. Forsbach. 2001. "Layered Data Representation for Visual Simulation of Terrain Erosion." In *Proceedings of 17th Spring Conference on Computer Graphics*.

[7] N. H. Ahn, A. Sourin, and P. Aswani. 2007. "Physically based hydraulic erosion simulation on graphics processing unit." In *GRAPHITE '07 Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*.

[8] B. Jákó. 2011. "Fast Hydraulic and Thermal Erosion on the GPU." In *Proceedings of CESCG 2011: The 15th Central European Seminar on Computer Graphics*.