RiverLand: An Efficient Procedural Modeling System for Creating Realistic-Looking Terrains

Soon Tee Teoh

Department of Computer Science, San Jose State University

Abstract. Generating realistic-looking but interesting terrains quickly is a great challenge. We present RiverLand, an efficient system for terrain synthesis. RiverLand creates a realistic-looking terrain by first generating river networks over the land. Then, the terrain is created to be consistent with the river networks. In this way, the terrains created have a proper drainage basin, an important feature lacking in many existing procedural terrain methods. The terrains generated by RiverLand are also widely varied, with rolling hills, river valleys, alpine mountains, and rocky cliffs, all seamlessly connected in the same terrain. Since RiverLand does not use complex fluid simulations, it is fast, and yet is able to produce many of the erosion features generated by the simulation methods.

1 Introduction

Computer-generated terrains are in high demand due to the popularity of 3D games, online 3D fantasy worlds, and CGI movies. These virtual 3D terrains are desirable because they can be aesthetically pleasing and visually dramatic. However, efficiently generating realistic-looking terrains remains a huge challenge. The attractiveness of virtual terrains is that they can be creatively designed, so that they are not identical to real-world terrains. Yet they have to bear similarities to real-world terrains because users desire realism in virtual landscapes.

Creating realistic terrains is challenging because terrains on earth are sculpted by various complex forces. The two most influential physical processes are *tectonics* and *water erosion*. Plate tectonic movements create large features such as volcanoes, fold mountains and block mountains. River or glacial erosion then modify the land to create canyons, plains and river valleys. Other forces such as wind, rain and freeze-thaw action also alter the physical landscape. The simplest and fastest procedural terrain modeling algorithms make use of fractals. Although these algorithms can produce terrains with mountains and valleys with interesting shapes, they are not realistic, because there is no river network in the resulting terrain. In fact, creating a river network on a fractal terrain is difficult because rivers must flow from higher ground to lower ground, while a fractal terrain often does not have a natural path for a river to flow. Therefore, complex river simulation methods have been proposed. However, these methods tend to be significantly slower.

We have developed a new system, RiverLand, that generates a procedural terrain by first generating the river network. The method used by RiverLand generates rivers with tributaries, and then creates mountains and other physical features on the land. In this way, a realistic-looking terrain is created. RiverLand is versatile; it allows users the option of controlling the terrain by specifying ridges and mountains, and it can produce many interesting features including alpine mountains, rolling hills, cliffs, plains and valleys. Furthermore, these different features compose a single terrain in a seamless manner.

In this paper, we examine the existing procedural terrain methods, and describe the advantage of RiverLand compared to these methods. We also describe how RiverLand accomplishes the challenges of terrain modeling mentioned in these papers. RiverLand is also a fast system, generating a 256 $\times$ 256 height-map in 0.2 seconds.

2 Related Works

Early works [1–4] in procedural terrain synthesis were based on fractals and fractional Brownian motion. These foundational algorithms can efficiently create landscapes with mountains and valleys. However, as pointed out by later works, a major weakness of these fractal terrains is that they have no global erosion features, and are therefore limited in their realism.

Noticing this defect in fractal approaches, Kelley et al. [5] presented a method for simulating stream networks. Musgrave et. al [6] then added local control to fractal terrains, and then simulated various erosion processes on these terrains. Other simulation-based methods [7] [8] [9] are also able to produce realistic-looking terrain. Improving on speed, Neidhold et al. [10] are able to run fluid dynamics simulations on an initial terrain, and generate eroded terrain in near real-time (a 256x256 terrain in 0.25 seconds on a 2.4 GHz machine). Recent works [11] [12] implemented fluid dynamics computations on the GPU, also improving on speed. They are able to simulate a 1K $\times$ 1K terrain with 50 iterations in 11.2 seconds (compared to 67 seconds using the CPU). These simulation-based methods are limited because they require an initial starting terrain, and they also tend to generate uniform-looking terrains because the same erosion processes and parameters are used throughout the entire terrain.

Prusinkiewicz and Hammel [13] proposed a fractal algorithm to generate a river, and corresponding mountains. A weakness of this method is that it produces rivers with assymetric valleys and no tributaries.

Belhadj and Audiber [14] proposed first generating ridge lines using *Ridge Particles* and then growing river networks by dropping *River Particles* on top of the ridges. A midpoint displacement method is then used to fill in the rest of the terrain. This method is relatively fast, producing a 512 $\times$ 512 terrain in 0.45 seconds on a 3.0 GHz machine. In comparison, our system starts with only the river network, and is able to generate more varied terrains. We also allow users to optionally influence the terrain by specifying desired ridges and mountains.

More recently, Schneider et al. [15] presented a system for real-time editing, synthesis, and rendering of infinite landscapes using the GPU. However, like earlier methods, this system produces no river network and is limited in realism.

Recently, several works [16] [17] [18] have taken a new approach. Observing that it is difficult for most existing systems to generate terrain with a desired style, these methods use actual real-world Digital Elevation Maps (DEM) as examples, to synthesize new terrain. Brosz et al. [17] extracts high-resolution details from existing terrain to add details to a low-resolution terrain. Zhou et al. [18] allows the user to sketch a path on an empty canvas, and provide a sample terrain style DEM dataset, for example the DEM of the Grand Canyon. They then produce a terrain with a canyon traced around the sketch provided by the user. The images produced look realistic, compared to previous methods. However, these example-based methods have several important limitations. First, they have difficulty in producing realistic heterogeneous terrain, merging different terrain styles seamlessly. Second, they are unable to generate novel terrain styles that are not identical to any existing style in their database. Third, the resulting terrain may not be physically realistic, since the river networks they produce are arbitrary, and not based on physical simulation.

There are also several excellent commercial and freely available procedural terrain systems. Mojoworld (www.pandromeda.com), for example, allows the user to create entire planets, including terrain, vegetation and atmosphere. Terragen (www.planetside.co.uk) also provides photorealistic scenery rendering, with tools available to create terrains. It allows the user to “paint” the shape of a landscape to position mountains and valleys; and automatically shapes the rest of the terrain with random fractal methods and physical-based methods such as simulation of glaciation. Bryce (www.Daz3D.com) allows the user to create virtual worlds, including both natural and urban landscapes.

Gain et al. [19] presented a novel system that allows the user to sketch the silhouette of a mountain on a 3D view of the terrain, and optionally modify shadow and boundary lines. The program then automatically generates a landscape that fits the user sketch.

3 RiverLand Overview

RiverLand begins with an empty 2D canvas which represents the top view of the land. The user is allowed to paint regions on the canvas. Within each region, the user is allowed to specify ridge lines, and the heights and widths of various points on the ridge lines.

The procedural terrain synthesis method begins by creating random river seed points on the region boundary. From these seed points, meandering rivers are grown into the region. Tributaries are also spawned from these rivers. Rivers are not allowed to cross user-specified ridges. Once inside the influence of a user-defined ridge, a river grows up in the direction towards the ridge line. Once the river network is established, mountains are grown on the rest of the terrain. Fractal methods are used to introduce interesting variations into the terrain. Figure 1 shows an overview of this process.

Compared to previous methods, RiverLand has the following advantages: (1) It produces more heterogenous landscapes, with steep ravines, alpine mountains, rolling hills, flat river valleys and cliffs within the same terrain; (2) it produces a globally physically consistent river network, a feature only present in a few of the existing algorithms; (3) it is near real-time, faster than most of the existing algorithms; and (4) it allows users to influence the terrain, a feature found only in some existing methods.

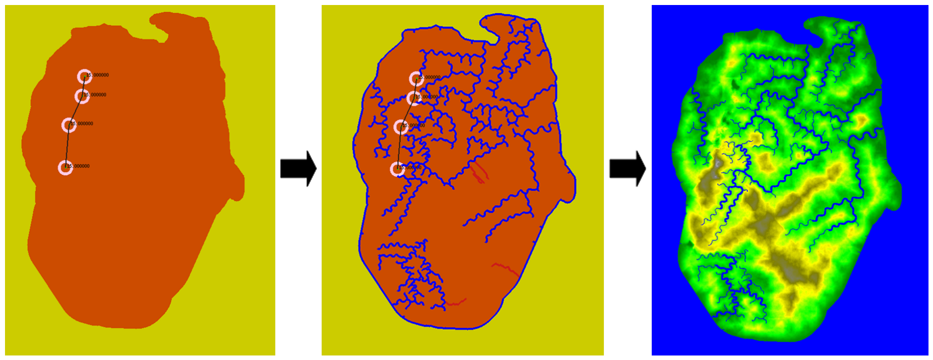

Fig. 1. RiverLand Terrain Synthesis Process. Left: User optionally paints region and ridges. Middle: Algorithm generates fractal ridges and rivers. Right: Terrain topography is fitted onto map.

4 Creating the River Network

For each region painted by the user, RiverLand assumes that its boundary is flat land. Terrain synthesis begins by setting n random boundary cells to be river mouths, each mouth being at least d cells apart. By adjusting the parameters n and d, the user can influence the eventual appearance of the terrain.

From each river mouth, a river network is grown. The initial direction vector of the river is the vector perpendicular to the boundary. Going forwards on this vector for distance $SegmentLength + r$ (where r is a random number), set a *SegmentPoint*. From this new SegmentPoint, repeatedly rotate the vector by a slight random angle, go forwards by a distance, and set a new SegmentPoint, until one of the following conditions is reached: (1) It goes with a distance d from the river or region boundary, or (2) The target t number of SegmentPoints has been set, where d and t are set by the user. By adjusting the limits of angle rotation of the river vector, and parameters such as SegmentLength, each river can have different characteristics. For example, large straight rivers can be made by increasing SegmentLength and decreasing the rotation angle, and also increasing the target number of SegmentPoints. By generating random rivers with different parameters, rivers of different characteristics can be created in the same region.

Next, a meandering river is fit through the SegmentPoints. In nature, most rivers do not travel in a straight path; instead they alternately curve left and right, forming the familiar meanders. This is done by fitting a curve between each SegmentPoint. A random curvature is set for each segment. Each river has a *MeanderCurvature* parameter that controls the mean meander curvature of the river. Figure 2 shows how different parameters for *SegmentLength* and *MeanderCurvature* influence the characteristics of a river. RiverLand generates rivers with different parameters to create a more varied terrain.

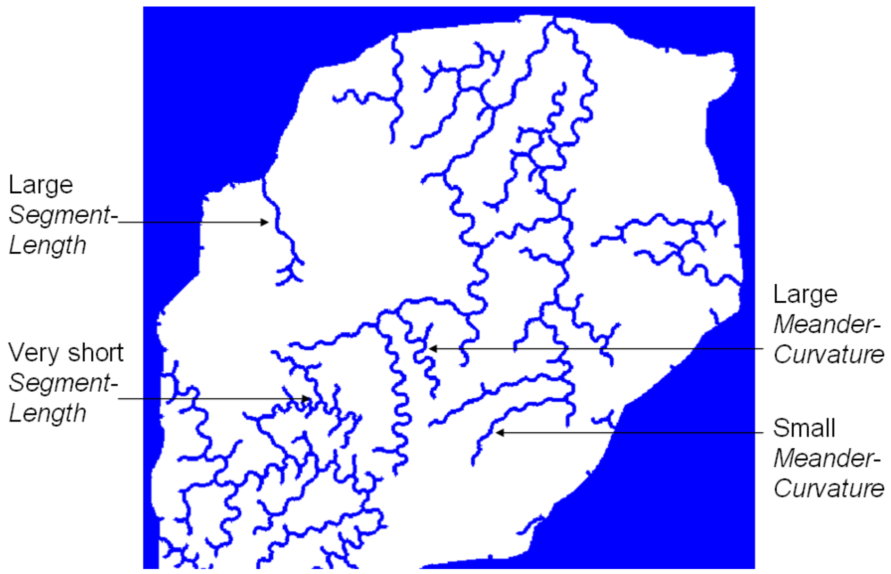
Large *Segment-Length*
Very short *Segment-Length*
Large *Meander-Curvature*
Small *Meander-Curvature*

Fig. 2. Different values of parameters SegmentLength and MeanderCurvature create rivers with different characteristics, producing a more varied terrain.

Once the meandering river has been fit, new seed points are placed randomly on the concave banks of the river. Tributaries can be grown outwards from these new seed points. These new seed points are inserted into the list of all seed points. At each iteration, a seed point is randomly selected, and a river or tributary is grown from it, until a target number of rivers or tributaries have been created, or until there are no seed points left. This completes the creation of the river networks in the region.

After the river network is created, a height is assigned to each river cell. At all river mouths, the height is assumed to be zero. For each river, a parameter *AverageSlope* is set. Going from one river cell to the next, the height is increased by AverageSlope, plus some random offset. When the algorithm reaches a river cell that is a distance d (set by the user) from the source, the river is allowed to be steeper. This is because real-world rivers tend to be steeper near the source. Due to the deposit of eroded sediments, rivers tend to form large, flat river plains downstream. RiverLand can also simulate water-flow so that rivers get wider as they flow downstream, and as more tributaries join them.

If the user has defined some ridges, the river network algorithm has to take them into consideration. When a river SegmentPoint is set within the influence of a user-defined ridge, the river vector is constrained to go straight up the user-defined ridge, within a narrow angle. Also, the river is not allowed to grow past the user-defined ridge line. This ensures that the generated river network is consistent with the user-defined ridges.

5 Generating Terrain Height

Once the height of each river cell is set, the next step is to generate the heights of all the cells in the terrain. First, the medial axis (skeleton) of the river and boundary cells is found. This medial axis consists of all the cells that are of maximal distance from a river or boundary cell. These medial axis cells are called "Maximal Cells". To find the Maximal Cells, first add all the river and boundary cells into a list, and set their distance d to 0. For each cell C_1 in the list, consider all its adjacent cells to the list. Let C_2 be an adjacent cell to C_1. Then, the candidate distance of C_2 from C_1 is $d_{C2} = d_{C1} + \mathit{slope} \times \mathit{dist}$, where *dist* is the horizontal distance between C_1 and C_2, and *slope* is a parameter set by the user. For each river or region boundary cell, a *slope* parameter is set to control the slope of the terrain from this cell. To create gentle slopes, set the value to less than 1, say 0.75 or 0.5. Allowing the *slope* variable allows the system to generate assymetric ridges, where one side is steeper than the other. If this candidate distance d_{C2} is less than the existing distance $d_{C2existing}$ from C_2 to the river, set $d_{C2existing}$ to d_{C2}, and set the *PathToRiver* pointer of C_2 to point to C_1. Then, remove C_1 from the list.

When the list becomes empty, all the cells that have no incoming *PathToRiver* pointers are the Maximal Cells. Each Maximal Cell has an associated River Cell or Boundary Cell from which it came. Next, a $\widehat{\mathrm{M}}$inimumHeight and *IdealHeight* of each Maximal Cell is assigned. The MinimumHeight of a Maximal Cell is the height of its associated River Cell, since the path from the Maximal Cell to the River Cell is considered the drainage basin of the River Cell, and so cannot be lower than the River Cell. The *IdealHeight* of the Maximal Cell is set to $MinimumHeight + \mathit{slope} \times \mathit{distancetoriver}$. Next, the *IdealHeights* of all Maximal Cells are smoothed by taking the average *IdealHeights* of all neighboring Maximal Cells. The actual height of each Maximal Cell is set to its *IdealHeight* unless it is less than the *MinimumHeight*, which happens only rarely. Finally, the heights of all the cells along the path from the Maximal Cell to the River Cell are linearly interpolated. Other adjustments are later made to vary the heights, as well as to add the influence of user-defined ridges.

The algorithm described can produce many cliffs along maximal cells. This is because maximal cells mark the boundary between the drainage basins of different rivers. Since the rivers have different heights, the maximal cells associated with each river would also have different heights. The solution is to use a cliff-removal step to produce a more natural-looking terrain. First, a cliff threshold c is defined. Any cell that has a height difference greater than c with any adjacent cell is considered a cliff. The cliff-removal algorithm adds all the lower cliff cells into a list L, and raises each lower cliff cell to within c of its tallest adjacent cell, to remove the cliff. Because raising this cell may create new cliffs, the iterative step examines every neighbor N of every cell C in list L, and if that neighbor N's height $h_N < h_C - c$ (where h_C is `C`'s height), N is added to list L, and h_N is raised to $h_C - c$. This is repeated until list L is empty. After this step, all the cliffs created on the river drainage basin boundaries are removed.

So far, all the heights increase linearly from the river/coast to the top of the hills. To generate a more interesting terrain, the slope profile can be changed. Using the default linear function, the height h of a cell is set to $h = sd$, where s is the slope, and d is the distance of the cell to the river or coast. To make the terrain more interesting, different functions can be used, so that $h = f(d)$. Currently, a few hard-coded functions are available in RiverLand. Figure 3 shows an example of creating cliffs by using a non-linear function for the slope profile.

6 User-Defined Ridges

The user is allowed to define and edit ridge lines to influence the terrain, before the algorithm computes the river network. The user clicks with the left mouse button to place control points for ridges, and has the option of selecting and moving control points with the right mouse button. For each control point, the user can enter the desired height and width. RiverLand scan-converts the ridge lines to find all the cells along the lines. For each cell between the control points, RiverLand uses the random midpoint offset method to assign its width and height. According to the random midpoint offset method, the height h of a point midway between two other points P_{left} and P_{right} is given by $h = avg(h_{Left}, h_{Right}) + srd$, where s is a user-defined roughness factor, r is a Gaussian random variable, and d is the horizontal distance between P_{left} and P_{right}. The same formula to used to calculate the width. Starting with a pair of control points, the midpoint height and width are calculated, and then the rest of the points are recursively calculated.

Once the height of each cell on the ridge line is calculated, the height of all the cells within the influence of the ridge is set by linearly interpolating between the ridge cells and the boundary cells, whose height is set to 0. This is done by finding the path from each ridge line cell to the ridge boundary, using a method similar to the method described in Section 5 used to set terrain heights from river cells.

After that, a random fractal method is used to roughen the terrain. Then, the user-defined terrain is merged with the rest of the terrain height-map, which is created by setting the heights of cells from the algorithm-generated river network (see Section 5). For each cell, take the maximum of the height from the river network and the user-defined ridge. The result is shown in Figure 4.

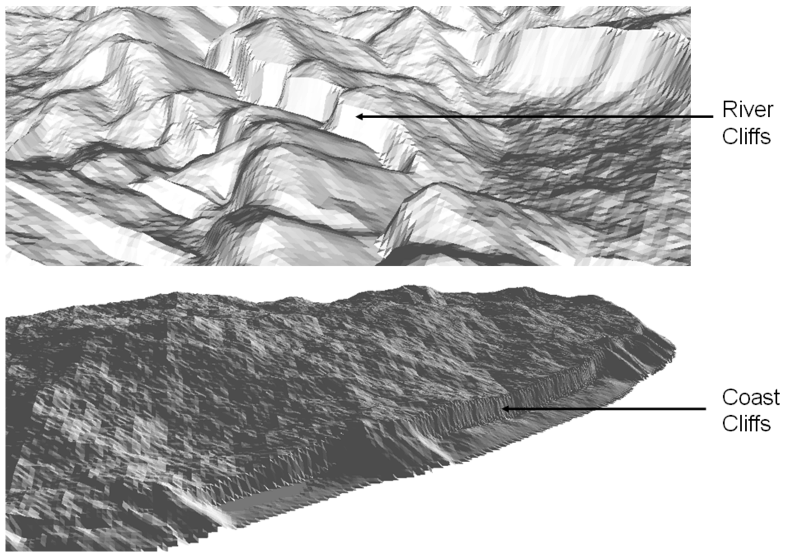

Fig. 3. Changing the slope profile to form river cliffs and coast cliffs

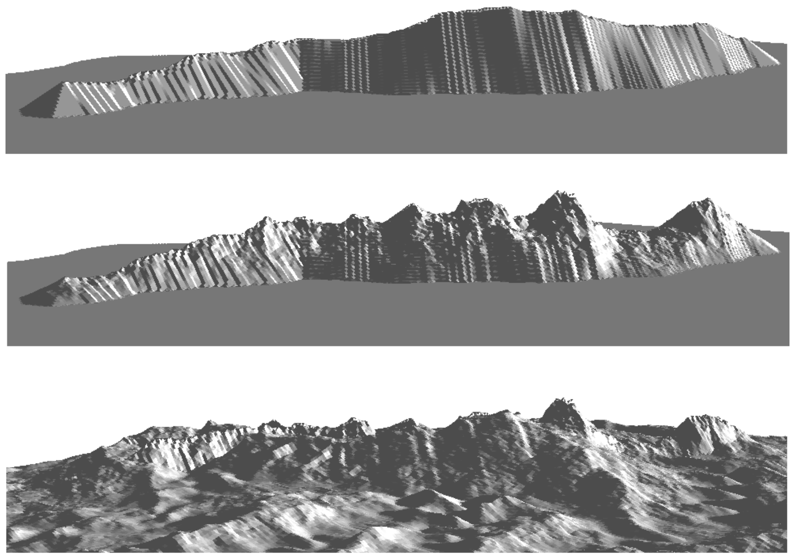

Fig. 4. Top: 3D view of user-defined ridge. Middle: Heights adjusted by random offsets to roughen terrain. Bottom: User-defined ridge shown with the rest of the terrain.

7 Randomized Fractal Overlay

The random midpoint offset algorithm was introduced by Miller [4] to produce fractal terrains. Although this is an efficient method to quickly produce large terrains which look realistic in small patches, the resulting terrains are uniform and do not have realistic erosion features found in real-world terrains. However, this method is useful for adding roughness to the terrain.

In RiverLand, we use the random midpoint displacement algorithm in several ways already mentioned. First, it is used to vary the height along the user-defined ridge lines. It is also used to vary the width along the user-defined ridges.

Next, it is also used to vary the height on all the cells covered by the user-defined ridges. It is also used to vary the height of all the cells created from the river network. This is done by first creating a fractal terrain "overlay". For example, if the terrain is $n \times n$ cells, we create another fractal terrain that is at least $n \times n$ cells. The fractal terrain is created this way. First a coarse $k \times k$ grid of random height values is created. Each cell of the coarse grid is then subdivided using the random midpoint displacement mehtod into $m \times m$ fine grid points, where m is a power of 2. Finally, there would be $(k-1)m+1 \times (k-1)m+1$ cells. The roughness of this fractal terrain can be controlled by the user by controlling the range of height values of the initial coarse grid, and also by controlling the roughness parameter in the random midpoint displacement algorithm.

To prevent the fractal terrain from altering the correctness of the river network, river cells' heights are not adjusted by the fractal terrain overlay. This is to preserve the rule that rivers' heights are non-increasing as they flow towards the mouth. The height change of a cell is therefore $dh = f \times d_r$ where f is a constant set in the program, and d_r is the distance of a cell from a river or region boundary cell.

In addition, on the 2D map, RiverLand also allows the user to paint over an area called a "Modifier Patch", to modify the terrain to add rocky outcrops. Many examples of famous scenery in the real world, such as Guilin (China), the Three Gorges (China), Yosemite Valley (USA) and Rio De Janeiro (Brazil), are rocky outcrops, because they make the landscape interesting. Rocky outcrops can take different shapes, and this is allowed in RiverLand by setting different parameters for a Modifier Patch: r, the average radius of a rock, h the average height, d the average distance between outcrops, and s the steepness. Rocky outcrops are then generated within the Modifier Patch and added to the fractal overlay. Since the rocky outcrop modifies the fractal terrain overlay, it does not modify the final terrain directly. In that way, it will not change the course and height of a river, so that the river networks remain correctly downward flowing.

8 Performance

RiverLand is a fast system; it is able to generate a 256×256 terrain height-map in 0.2 seconds, a 512×512 terrain in 0.9 seconds, and a 1024×1024 terrain in 4 seconds on a 2 GHz machine with 2 GB RAM. This compares well with other recent methods. This does not include the time for generating the fractal overlay, which is done only once during set-up, and can be re-generated by user request.

9 Conclusions

We have presented RiverLand, a system for the fast procedural modeling of terrains. Existing terrain synthesis methods can be generalized into two main types: (1) fractal-based, and (2) simulation-based. In general, fractal-based terrains are faster to generate, but are limited in realism, whereas simulation-based methods are much slower, and their results vary: some terrains look more realistic than others. RiverLand is fundamentally a fractal-based procedural modeling method, but the process is designed to produce terrains consistent with river networks, an important feature lacking in most existing faster fractal-based methods.

One attractive feature of RiverLand is that it allows the user to influence the terrain if the user has preferences. First, by choosing different parameters, the user can influence the style of the terrain. For example, increasing the number and length of rivers makes the river network cover more of the terrain. Decreasing the minimum distance between rivers results in higher river density. Users can also control the roughness and slope of the terrain. The user can also draw ridge lines, and for each control point on the ridge line, the user specifies the desired height and width. RiverLand produces a river network and terrain height-map consistent with the ridge lines defined with the user. The user can paint Modifier Patches to create rocky outcrops, to make the terrain more interesting. Most of the existing terrain synthesis methods do not allow the user such extensive control of the terrain, and yet generate a terrain that is consistent over different areas. RiverLand can create a terrain without any user input at all, but can also create a terrain with a significant amount of user input.

Another advantage of RiverLand is that the terrains produced contain a wide variety of interesting physical features found in real-world terrains, such as cliffs, gentle river valleys, deeper ravines, alpine mountains, rocky outcrops, and rolling hills. All these features can be found in the various example images shown in this paper. Figures 5 shows more examples. This figure also shows an example of automatic adjustment of river width according to water flow simulation by RiverLand.

One important desirable physical feature mentioned in other papers is ridges, which are not generated by the basic fractal methods, but occur in real-world terrains. The terrains produced by RiverLand produce ridges, even if no ridge lines are specified by the user. The terrains generated by RiverLand contain a diversity of features fitting naturally in the same terrain. Also, since RiverLand is not an example-based terrain synthesis program, it can create terrains which are not replicas of real-world examples, and so is less limited in range. By adjusting different parameters, users can also create a terrains in a variety of different styles, and so RiverLand is less rigid, compared to example-based methods.

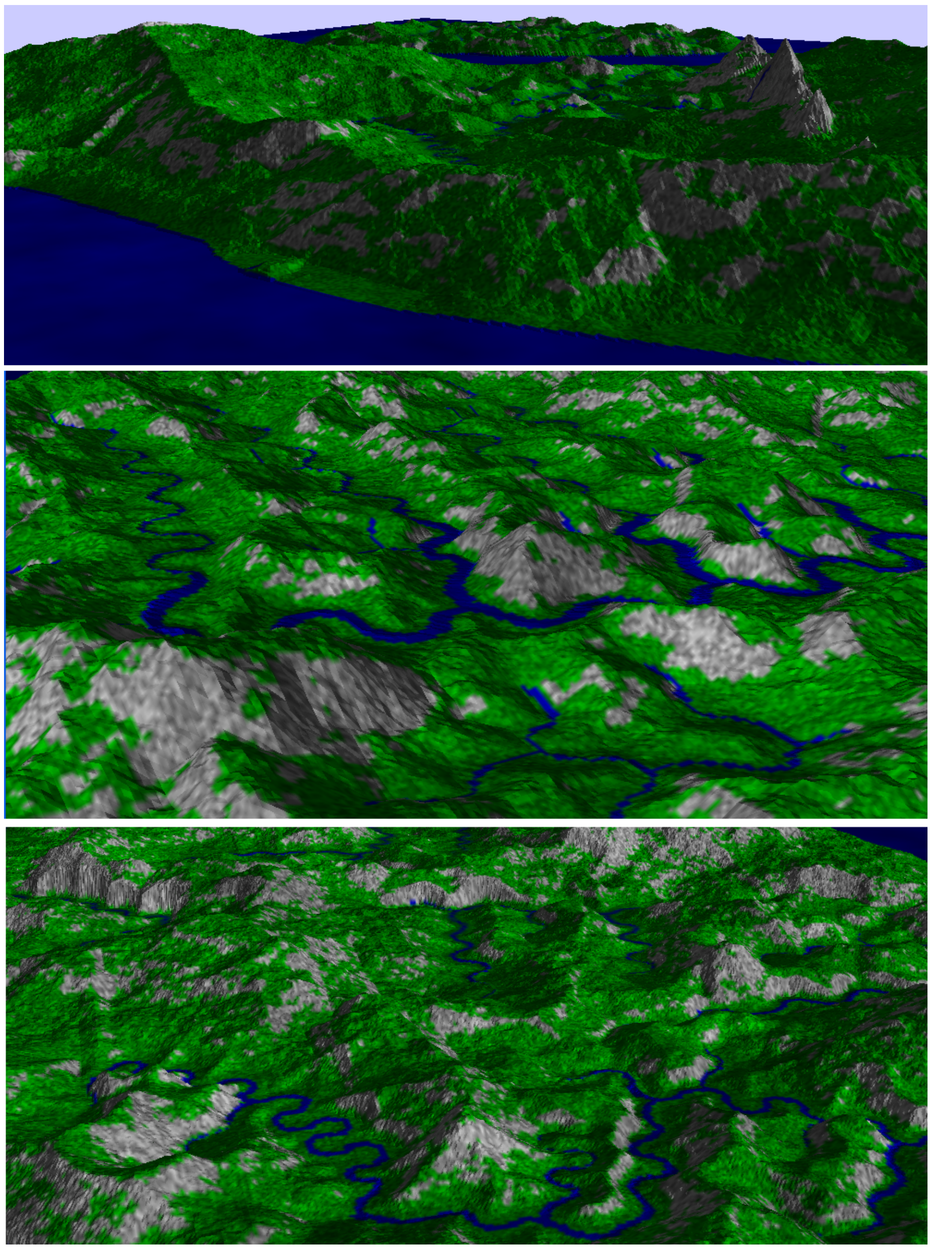

Fig. 5. 3D view of terrains enhanced by color, showing terrain in green, river cells in blue, steep cliffs in grey, and high elevation in white. RiverLand allows automatic adjustment of river width according to water flow simulation, shown in the bottom terrain.

References

1. Carpenter, L.: Computer rendering of fractal curves and surfaces. In: Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques. (1980) 109
2. Mandelbrot, B.: The Fractal Geometry of Nature. W.H. Freeman and Co. (1982)
3. Fournier, A., Fussell, D., Carpenter, L.: Computer rendering of stochastic models. Communications of the ACM **25** (1982) 371–384
4. Miller, G.: The definition and rendering of terrain maps. In: Proceedings of the 13th Annual Conference of Computer Graphics and Interactive Techniques (SIGGRAPH 1986). (1986) 39–48
5. Kelley, A., Malin, M., Nielson, G.: Terrain simulation using a model of stream erosion. Computer Graphics **22** (1988) 363–268
6. Musgrave, F., Kolb, C., Mace, R.: The synthesis and rendering of eroded fractal terrains. Computer Graphics **23** (1989) 41–50
7. Roudier, P., Perrin, B.: Landscapes synthesis achieved through erosion and deposition process simulation. Computer Graphics Forum **12** (1993) 375–383
8. Nagashima, K.: Computer generation of eroded valley and mountain terrains. The Visual Computer **13** (1997) 456–464
9. Chiba, N., Muraoka, K., Fujita, K.: An erosion model based on velocity fields for the visual simulation of mountain scenery. The Journal of Visualization and Computer ANimation **9** (1998) 185–194
10. Neidhold, B., Wacker, M., Deussen, O.: Interactive physically based fluid and erosion simulation. In: Eurographics Workshop on Natural Phenomena. (2005)
11. Nguyen, H., Sourin, A., Aswani, P.: Physically based hydraulic erosion simulation on graphics processing unit. In: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia. (2007) 257–264
12. Krištof, P., Beneš, B., Křivánek, J., Šťava, O.: Hydraulic erosion using smoothed particle hydrodynamics. Computer Graphics Forum (Proceedings of Eurographics 2009) **28** (2009)
13. Prusinkiewicz, P., Hammel, M.: A fractal model of mountains with rivers. In: Proceedings of Graphics Interface. (1993) 174–180
14. Belhadj, F., Audiber, P.: Modeling landscapes with ridges and rivers. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST 2005). (2005) 151–154
15. Schneider, J., Boldte, T., Westermann, R.: Real-time editing, synthesis, and rendering of infinite landscapes on GPUs. In: Vision, Modeling and Visualization 2006. (2006)
16. Chiang, M.Y., Huang, J.Y., Tai, W.K., Liu, C.D., Chiang, C.C.: Terrain synthesis: An interactive approach. In: Proceedings of the International Workshop on Advancaed Image Tehnology. (2005) 103–106
17. Brosz, J., Samavati, F., Sousa, M.: Terrain synthesis by-example. Advances in Computer Graphics and Computer Vision International Conferences VISAPP and GRAPP 2006, Revised Selected Papers **4** (2006) 58–77
18. Zhou, H., Sun, J., Turk, G., Rehg, J.: Terrain synthesis from digital elevation models. IEEE Transactions on Visualization and Computer Graphics **13** (2007) 834–848
19. Gain, J., Marais, P., Strasser, W.: Terrain sketching. In: Proceedings of the ACM Symposium on Interactive 3D Graphics and Games. (2009)