Real-Time Terrain Modeling using CPU–GPU Coupled Computation

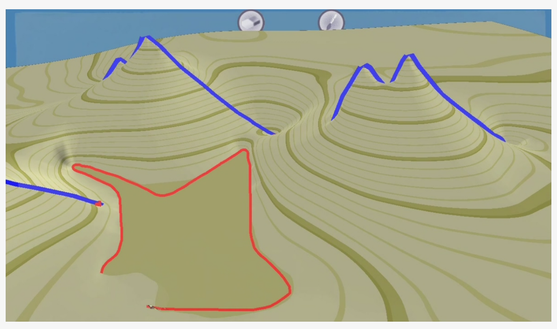
Adrien Bernhardt [1] adrien.bernhardt@inrialpes.fr
Andre Maximo [2] andmax@impa.br
Luiz Velho [2] lvelho@impa.br
Houssam Hnaidi [3] houssam.hnaidi@liris.cnrs.fr
Marie-Paulie Cani [1] marie-paule.cani@inrialpes.fr

[1] INRIA, Grenoble Univ., France [2] IMPA, Brazil [3] LIRIS, CNRS, Univ. Lyon 1, France

Abstract—Motivated by the importance of having real-time feedback in sketch-based modeling tools, we present a framework for terrain edition capable of generating and displaying complex and high-resolution terrains. Our system is efficient and fast enough to allow the user to see the terrain morphing at the same time the drawing editing occurs. We have two types of editing interactions: the user can draw strokes creating elevations and crevices; and previous strokes can be interactively moved to different regions of the terrain. One interesting feature of our tool is that terrain primitives can be interactively manipulated similarly to primitives in vector-graphics tools. We achieve realtime performance in both modeling and rendering using a hybrid CPU–GPU coupled solution. We maintain a coarse version of the terrain geometry in the CPU by using a quadtree, while a fine version is produced in the GPU using tessellation shaders.

Sketch-based Terrain Modeling

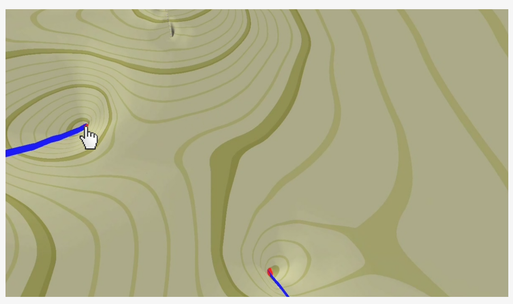
Mountain Strokes



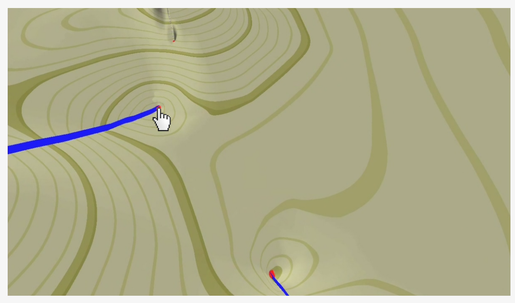
Plateaux Strokes

The majority of procedural editing and feature extraction tools for terrain modeling are based on a top-view interaction. Although interesting, this approach limits the artistic freedom of the tool. Our modeling system is based on a 3D camera with a first-person perspective and sketch-based drawing interaction over the moving view plane. This method allows the user to draw landscape silhouettes at any distance and with more freedom.

Terrain Primitives as 3D Curves

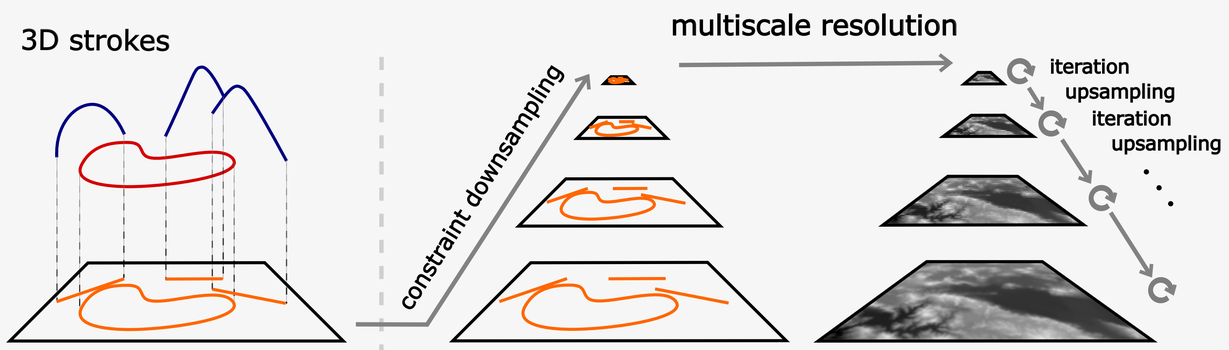


Moving Strokes Example



Our terrain modeling system allows intuitive and natural interaction through simple sketches drawn over the current camera viewing plane. The sketches can be created, moved or deleted; effectively changing the terrain. Depending on the desired primitive to create or modify, the user can move the camera to choose the best view of the terrain. The camera zoom can also be used to interact with terrain primitives at different scales.

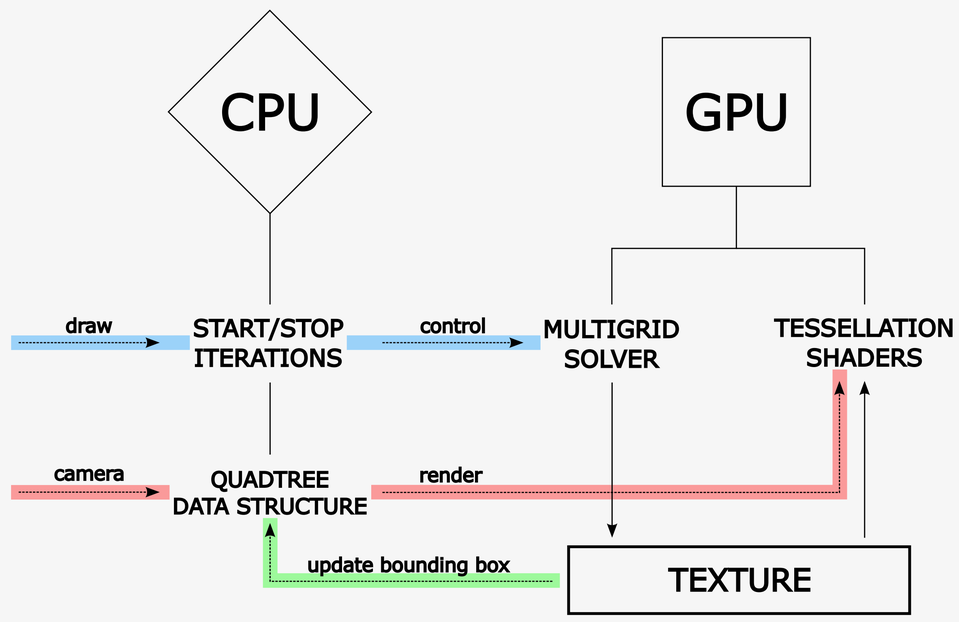
3D Curves and the Multigrid Solver



The 3D curves before rasterization represent terrain primitives and are similar to curves in a vector-graphics tool

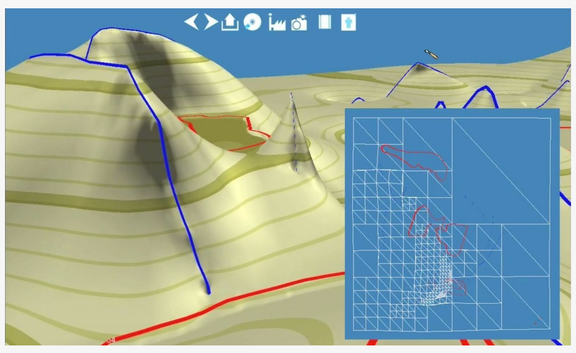
The landscape primitives specified by 3D strokes are converted to constraints through rasterization and stored on a 2D constraint texture. This texture is downsampled to be used in a solve-then-upsample process, i.e. the multigrid solver, where it first solves for the small resolutions before extrapolating the solution for finer resolution systems. The final solution is the heightmap of the terrain in a multiresolution pyramid stored as a mipmap texture.

CPU–GPU Coupled Solution

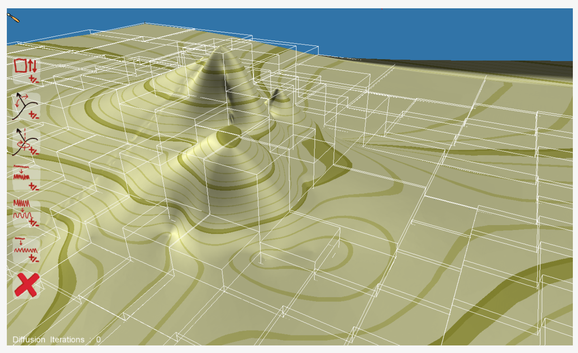


The terrain modeling framework of our tool. While the CPU controls the multigrid solver depending on drawings and sends terrain patches to be rendered, the GPU generates the terrain in texture memory and tessellates its own produced heightmap. This setting requires three different data paths: adaptive rendering using a quadtree (red); solver iterations control (blue); and updating coarse-resolution details on the quadtree (green).

CPU Quadtree Adaptation



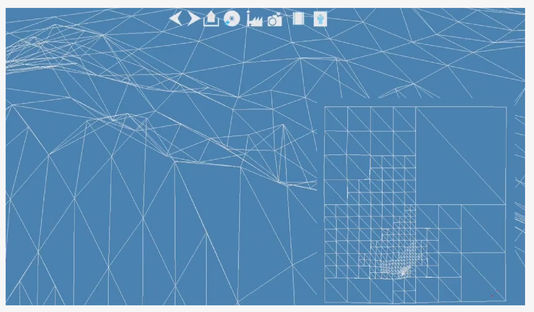
Quadtree Data Structure



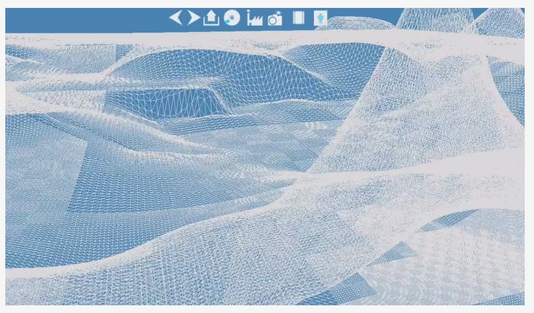
Quad Patches Bounding Boxes

Our approach partitions the terrain with a view- and heightmap-dependent quadtree. This lightweight data structure controls the quadrilateral patches to be rendered, serving two main purposes: it allows a first coarse LOD analysis of the terrain; and it frees the CPU from the costly task of generating the entire heightmap data.

GPU Tessellation



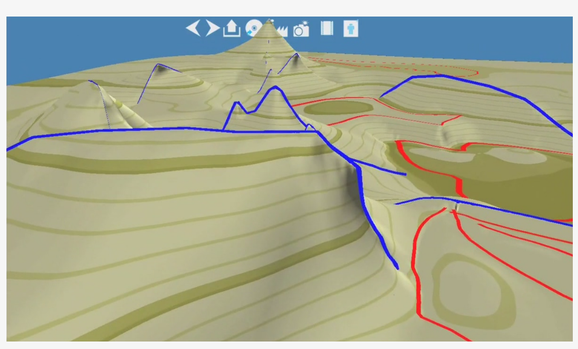
Minimum Quadtree Refinement

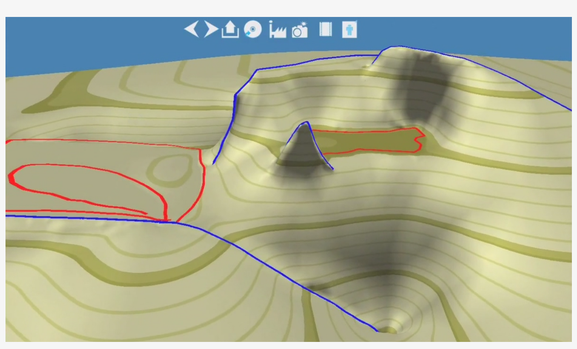


Terrain Tessellation

We employ the GPU to solely generate and maintain the terrain data on its own texture memory. The generation is accomplished by a fast multigrid solver specifically designed for terrains. The generated data is constantly used in our approach to morph the original rendering of low-resolution quadtree patches in smooth high-resolution triangles through the graphics-card tessellation shaders. The tessellator allows for a second fine LOD analysis of the terrain, balancing the decision between the units. Although the whole data is updated and used by the GPU, the management is done by the CPU.

Modeled Terrain Results





Terrain example generated by our modeling system (north and south views). The blue strokes were drawn to create mountains and crevices, and the red strokes were drawn to create plateaux in the midst.

We have tested our modeling system with different heightmap resolutions and number of solver iterations. Table I presents the terrain creation solver and tessellation-based rendering timings and GPU memory consumption for the tested terrain size and respective number of iterations. The timings are given using a 1024 × 768 pixel viewport and considering the camera constantly moving. All timings were performed in an Intel Core2Quad CPU and an nVidia 480 GTS GPU using OpenGL 4.1.

Terrain Size	Iterations	Creation (ms)	Tess. (ms)	GPU (MB)
512×512	45	23	4.8	16.9
$1K \times 1K$	49	28	5.2	57.3
$2K \times 2K$	53	35	5.9	141.8
$4K \times 4K$	56	44	6.7	716.7

TABLE I
TERRAIN MODELING COMPUTATIONAL TIMINGS AND GPU MEMORY CONSUMPTION AT DIFFERENT RESOLUTIONS.

References

HNAIDI, H., GUÉRIN, E., AKKOUCHE, S., PEYTAVIE, A., AND GALIN, E. 2010. Feature based terrain generation using diffusion equation. Computer Graphics Forum 29, 7 (September).

BERNHARDT, A., MAXIMO, A., VELHO, L., HNAIDI, H., AND CANI, M. 2011. Real-time Terrain Modeling using CPU-GPU coupled computation. To appear in Proceedings of SIBGRAPI (August).