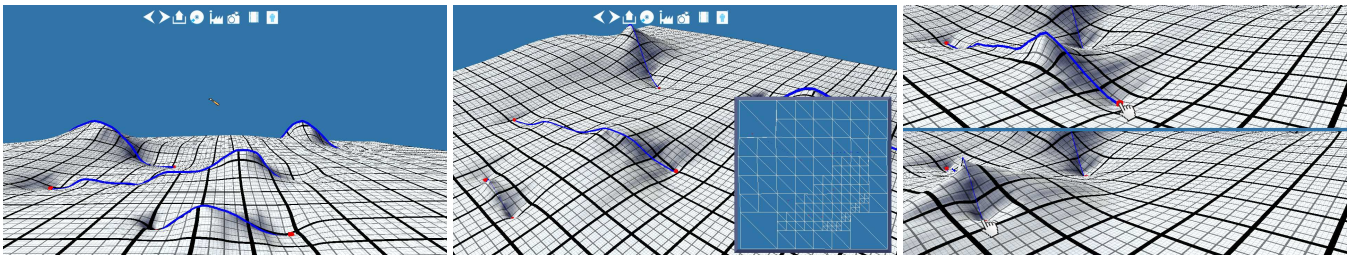
Real-time Terrain Modeling using CPU–GPU Coupled Computation

Adrien Bernhardt[1] André Maximo[2] Luiz Velho[2] Houssam Hnaidi[3] Marie-Paule Cani[1]
[1] INRIA, Grenoble Univ., France [2] IMPA, Brazil [3] LIRIS, CNRS, Univ. Lyon 1, France



(a) (b) (c)

Figure 1: *Terrain modeling example: as each stroke is drawn (a) or manipulated (c), the terrain is tessellated in the GPU to follow the stroke. In the CPU, the quadtree data structure (b) controls the quad patches sent to the GPU.*

Introduction and Related Work In many editing tools, especially sketch-based modeling, it is important to have real-time feedback to help improve the editing quality. This importance is emphasized particularly in sketch-based terrain modeling, being able to see the terrain morphing at the same time the drawing edition occurs constitutes a great user experience (see Figure 1(a)). In this work, we propose a real-time terrain modeling tool by combining a fast GPU-based terrain solver [Hnaidi et al. 2010] with a lightweight CPU-based data structure. Our tool is capable of dynamically generate multi-resolution heightmaps, enabling it to tessellate different parts of the terrain at different resolutions.

In our framework, we have two types of editing interactions: in the spirit of [Gain et al. 2009], the user can draw strokes creating elevations and crevices; and previous strokes can be interactively moved to different regions of the terrain. Differently than Gain et al.'s system, we do not extract noise from the user strokes to make the terrain more realistic, however we use a CPU–GPU coupled method to drastically improve the performance of our tool, generating terrains two orders of magnitude faster than Gain et al.'s work.

The terrain modeling in our approach is accomplished by combining the multi-grid GPU terrain solver of [Hnaidi et al. 2010] with an adaptive tessellation-based rendering shader capable of handling dynamic heightmaps. The main contribution of Hnaidi et al.'s work is to propose a GPU-based multi-grid diffusion equation solver, which interpolates not only heights but also amplitude and frequency noise. Our modeling tool uses Hnaidi et al.'s solver to allow an interactive manipulation of complex terrain primitives.

Real-time Terrain Modeling Creation of terrain models in real-time involves dealing with dynamically changing data that increases exponentially depending on the terrain resolution. In order to provide a real-time terrain modeling tool, we make use of two complementary approaches. First, a coarse version of the terrain is maintained in the CPU using a quadtree (see Figure 1(b)), where regions closer to the viewer are subdivided more than far regions. This simple and lightweight data structure fits the CPU main role of data control, while allowing it to send adaptive quad primitives to the GPU. Second, a fine version of the terrain is produced in the GPU using the tessellation control and evaluation shaders. The tessellation control shader is responsible to subdivide regularly each patch primitive, i.e. the quad leaf node sent by the CPU, while the tessellation evaluation shader reads the height values from a texture.

The first step of our algorithm is to update the quadtree data structure using a LOD-based approach. We consider the projection of the bounding box of each quad node by reading the minimum and maximum height value that falls inside the node. The projection is used to determine if the patch node needs to be sent to the GPU and to interactively adapt the quadtree in a way that each projection shape has about the same size.

The second step of our algorithm is to translate the sketch-defined terrain primitives to constraints that are used by the multi-grid GPU solver of [Hnaidi et al. 2010]. This solver provides a sequence of increasing resolution grids, up to an arbitrary size, which we store in a mipmap-pyramid texture to be read in our tessellation control and evaluation shaders. The tessellation control shader uses it to decide the subdivision level of each patch, while the tessellation evaluation shader uses it to place each generated vertex at the proper height value. The multi-resolution texture is also used by the CPU, but only a small resolution (6th mipmap level) of it since the quadtree minimum leaf size is much bigger than the texel from the highest resolution texture.

In our early experiments we discovered that by modifying several aspects of the original GPU solver, we are able to compute the entire multi-resolution heightmap texture (with $4K \times 4K$ maximum size) in $80\ ms$ using an off-the-shelf graphics card. Moreover, we use the CPU to control the solver iterations and stop at a certain resolution and then resume computing when the GPU is idle. Another interesting feature of our method is the balance between terrain generation in the CPU and in the GPU, we can control this balance by simply changing the quadtree refinement. With these features, the user can draw strokes and see at the same time the terrain morphing to the drawing. Terrain primitives, such as cliffs and mountains, are controlled seamlessly in our framework. In short, we believe this work has the potential to become an effective terrain modeling tool, creating high-quality terrain models in real-time.

References

GAIN, J., MARAIS, P., AND STRAßER, W. 2009. Terrain Sketching. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '09, 31–38.

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).