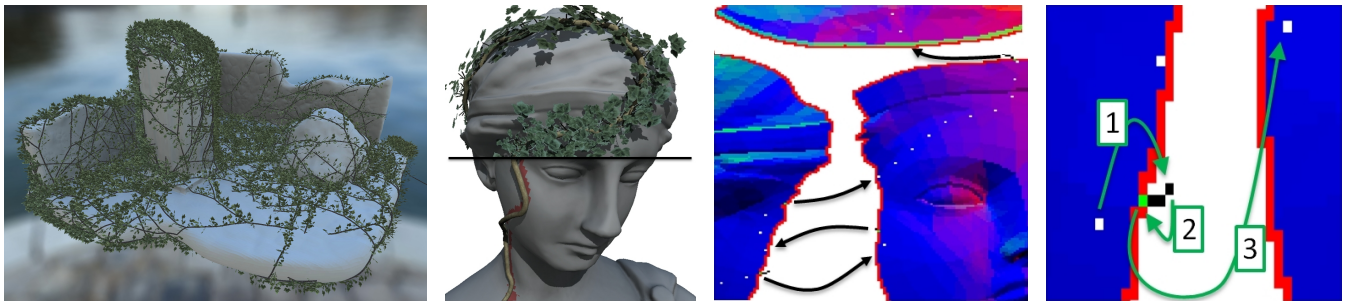
Dynamic on-mesh procedural generation control

Cyprien Buron, Jean-Eudes Marvie, Gaël Guennebaud, Xavier Granier

Dynamic On-Mesh Procedural Generation Control

Cyprien Buron Jean-Eudes Marvie Gaël Guennebaud Xavier Granier
Technicolor Inria - IOGS - Univ. Bordeaux



(a) 18 seeds generating 5.71M faces in 421ms (b) Growing area painting (c) Marching performed in (b) (d) First redirection of (c)

Figure 1: *Our texture marching approach allows to interactively control an expansion grammar over any parametrized surface (a/b). On-mesh paintings can be used to easily specify additional constraints (b). White pixels and black arrows in (c) show the marching performed for result (b), red border representing the indirection pixels. In case the destination pixel is in background (d-1), we march back (black pixels) until finding the appropriate indirection pixel (green pixel, d-2), that redirect to the next chart (d-3).*

Procedural representations are powerful to create highly detailed objects through amplification rules. Moreover, recent GPU-based approaches [Marvie et al. 2012] reach interactivity for massive scene generation. However, such fast generation methods do not consider environment contexts for controlling procedural objects. In counterpart, growth on shapes methods [Li et al. 2011] are restricted to CPU implementations, leading to limited performances.

We propose an approach, based on GPU Shape Grammars [Marvie et al. 2012], to interactively control grammar growths on the GPU using external contexts (Figure 1(a)). Our work uses textures as GPU-suitable representation encoding external contexts, and adds a texture accessor to the grammar for constraining any shape grammar rule with any external context. In addition, we introduce a texture marching method to follow the underlying surface during the grammar development. For instance, with our system one could paint restricted growth areas on the mesh, and the grammar will adapt itself to the new environment (Figure 1(b)).

Dynamic on-mesh control

Our marching algorithm is based on multi-chart geometry images encoding the surface contexts. In order to perform expansion on such surfaces, we add indirection pointers at chart borders [Lefebvre and Hoppe 2006], linking neighbor pixels of adjacent charts in the 3D space. Starting from a seed sampled on the mesh, the marching begins at the associated texture coordinates. Then, for a given direction, we fetch the destination pixel and generates a successor element between the current and destination vertices. Indirection pointers are then used to jump from one chart to another (Figure 1(c)), while reaching a background pixel is described in Figure 1(d). Moreover, to minimize the 3D direction distortions at indirection, we choose the indirection pixel in an arbitrary neighborhood for which the 3D direction is the closest compared to a base direction. In addition, as the charts may have different orientations, a rotation is applied to the marching direction according to estimated source and destination chart tangents.

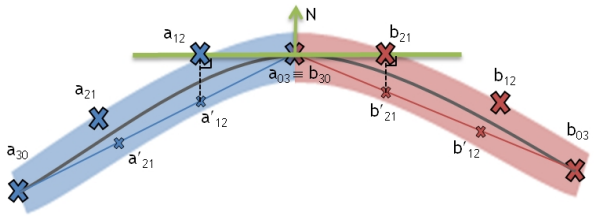


Figure 2: *Each geometry element is reduced to its two endpoints (a_{30}/a_{03}, b_{30}/b_{03}). Two additional temporary points are sampled on each segment (a'_{12}/a'_{21}, b'_{21}/b'_{12}), and projected on the tangent plane defined by the normal N at its nearest endpoint (a_{12}/b_{21} for $a_{03} \equiv b_{30}$). G_1 continuity is ensured by choosing the normal N in the plane defined by the adjacent endpoints (a_{30}, $a_{03} \equiv b_{30}$, b_{03}).*

In case geometry shapes are assigned to successor elements issued from the marching, a smooth interpolation should be performed to avoid discontinuities. To this end, we compute a smoothly varying deformation field based on point-normal interpolation through cubic Bezier curves (Figure 2). This requires a depth-first traversal of the grammar to evaluate all the marching points first. Finally, geometry shapes are instantiated in parallel and smooth deformations are achieved by interpolating a local frame along the Bezier curves.

Results

Our method is implemented on an nVidia GeForce GTX 580. The texture marching approach allows an interactive control of grammar developments, while our generic constraints increase the grammar expressiveness. Thanks to this solution one can paint procedural models, comprising 250K terminal rules and 5.71M triangles (Figure 1(a)), at interactive frame rates. While our current algorithm permits growths on pre-generated meshes, a future work will be to investigate marching on procedural objects generated on-the-fly.

Acknowlegments. We are grateful to Gaël Sourimant for the demo scenes and the video editing.

References

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. In *ACM Transactions on Graphics (TOG)*, vol. 25, 541–548.

LI, Y., BAO, F., ZHANG, E., KOBAYASHI, Y., AND WONKA, P. 2011. Geometry synthesis on surfaces using field-guided shape grammars. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2, 231–243.

MARVIE, J.-E., BURON, C., GAUTRON, P., HIRTZLIN, P., AND SOURIMANT, G. 2012. Gpu shape grammars. In *Computer Graphics Forum*, vol. 31, 2087–2095.