Temporally Coherent Sculpture of Composite Objects

Artur P. Sampaio, Raphaëlle Chaine, Creto A.Vidal, Joaquim B. Cavalcante-Neto

Temporally Coherent Sculpture of Composite Objects

Artur P. Sampaio[a,b], Raphaëlle Chaine[a], Creto A.Vidal[b], Joaquim B. Cavalcante-Neto[b]

[a]*Université Lyon 1, CNRS, LIRIS, UMR5205, F-69622, France*
[b]*Department of Computing, Universidade Federal do Ceará, 60455-760 Fortaleza, Brazil*

Abstract

We address the problem of virtual sculpting and deformation of shapes composed of small, randomly placed objects. Objects may be tightly packed - such as pebbles, pills, seeds and grains, or be sparsely distributed with an overarching shape - such as flocks of birds or schools of fish. Virtual sculpture has rapidly become a standard in the entertainment industry. Composites, though, are still usually created in a static way by individually placing each object or by sculpting a support surface and procedurally populating the final shape. That raises problems for the generalisation to evolving shapes with visual continuity of the components. Large amounts of geometrical data are generated, and must be maintained and processed, both by the CPU and by the GPU. Whenever the shape is deformed, one has to define how these compositing objects should turn, displace or disappear inside the volume, as well as how new instances should become visible to the outside. It is difficult to rely on a physical system to perform that task in real time. The system we suggest can be constructed upon any uniform mesh-based representation that can be deformed and whose connectivity can be updated by operations such as edge splits, collapses, and flips. The mesh remains populated with an aperiodic distribution of composing elements that are automatically updated under deformation. The idea is to sculpt the shape as if it were filled with little objects, without handling the complexity of manipulating volumetric shapes. For this purpose, we suggest exploiting the properties of the uniform sampling of the surface. We show that we are able to properly handle virtual sculpting of composites in real-time and maintaining temporal continuity. This system also uses GPU optimisations to render individual elements efficiently. To our knowledge, no previous sculpting system allows the user to simultaneously see and sculpt agglomerates in such a fast and reliable fashion.

Keywords:
Sculpting, Interactive techniques, Agglomerates
2000 MSC: 10.050, 30.010

1. Introduction

Modeling virtual objects is essential in the entertainment industry. Far from computer-aided design (CAD) and tedious static parameterisation systems, artists expect to use abilities developed from real life sculpting in digital environments. Many systems now allow users to sculpt objects as they would do with virtual clay, but little effort has been made in the direction of materials composed of smaller entities that are glued together or that roll around each other. Artists should feel free to deform such global composites, as the elements change position, and appear or disappear automatically around each other. Likewise, we would like to give artists the opportunity to sculpt a cloud of little objects that are not joined to each other or to deform such composites through other means, such as scripted deformations.

In this paper, we consider the sculpting and rendering of surfaces represented as agglomerates of smaller, composing 3D elements. These agglomerate materials are common in nature: rock piles, armies of ants, castles of sand, beautiful works of tiled art - basically any group of similar, randomly oriented elements that, when viewed together, form a larger scale shape (see Figure 2). The 3D elements we use, henceforth denoted CompEls, can be seen as 3D models anchored at the faces of a support mesh. Representing and updating those kinds of assemblies efficiently remains an open problem. The sheer quantity of elements on a surface and the polygons required to represent them can cripple even powerful GPUs.

As stated in [3], the biggest challenge in creating an agglomerate is controlling the position of individual objects. Traditionally, agglomerates were created by placing each composing element in its final position, one by one. This can be cumbersome due to sheer object count. The other common approach is to design an intermediate representation and to populate it afterwards with smaller elements, using for example systems of particles in the case of a regular distribution or sample-based distributions [4]. This *a posteriori* element sampling can either be limited to the object's surface or encompass the entire volume [5]. Extensive research has been done on how to generate and render such models correctly. The representation of the proxy object generally does not need to be exact or high quality. It can result from the rough meshing of an implicit surface, or from any other meshing design that does not involve constraints on the quality of the facets. The final rendered composite will display very high quality results and sampling properties. Those optimisation techniques are clearly not intended for real-time applications or for dynamic surfaces. The corollary is that the user cannot see the desired agglomerate pattern

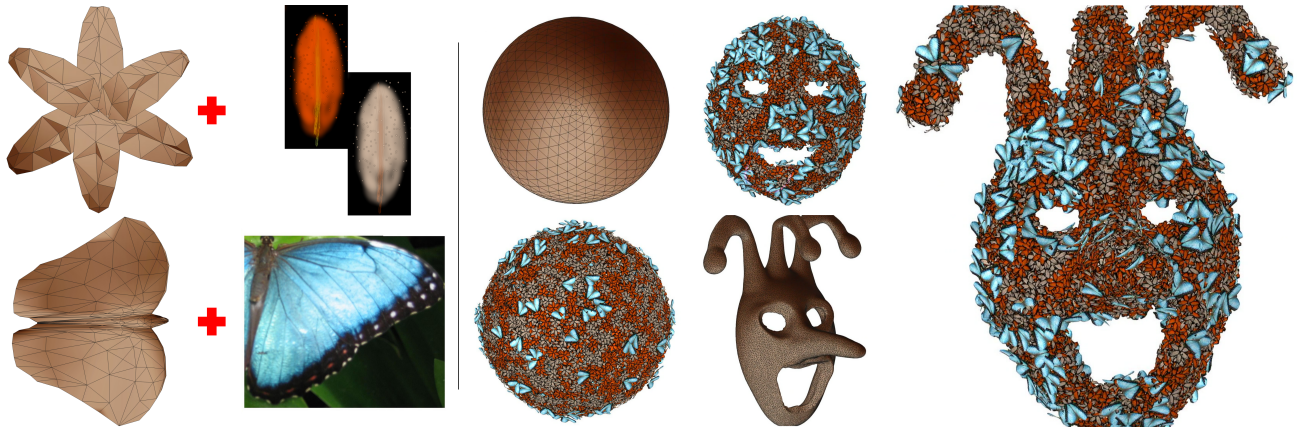


Figure 1: Carnival mask sculpted using our framework. Left: input CompEls and accompanying textures. Center: initial sphere mesh and intermediate steps of the sculpting process, with and without CompEls on the outer surface. Right: final rendering.

before the end of the design process of the proxy shape. This can be a problem for the design of object clouds, where it becomes very difficult to have an idea of the final shape's outer silhouette without the presence of the actual objects. The problem is compounded on the representation of animations, where temporal continuity or coherence is also of concern. Merely resampling elements on updated regions of an animated surface can cause significant visual artifacts, due to sudden changes associated with rapid object insertion and removal, as described in [6]. The temporal continuity that we aim to obtain is similar to the one defined by Medeiros et al [7], which means that objects should be inserted, displaced and removed without popping effects or loss of visual continuity between frames.

Another approach to minimize the cost of positioning the elements over the surface is to delay their involvement until the rendering pipeline, using works on spatial adaptive sampling of density function and projections. A number of publications focus on the process of rendering images using dots, called stipples. A stipple can later be transformed into the anchor of an orientable texture unit. For that reason, a good stippling can be the basis for agglomerate generation. One usually aims at obtaining blue noise samplings [8]. Hierarchical tiles with different levels of resolution [9], [10] can be used to generate adaptive samplings with fast timings (except for preprocessing costs). More focused on meshed surfaces to be sampled, Pastor et al [11] use a dynamic stippling generation system that performs directly on a mesh and that relies on adaptive subdivision meshes. However, although they use a hierarchical framework and the mesh could be edited for nearly isometric deformation, their work relies on fixed topology and overall connectivity. Finally, there is a tight coupling between vertices and stipples. For the same kind of applications, a hierarchical Poisson disk sampling was precomputed on the surface [7]. It can be later evaluated and rendered according to an importance function. Their system, however, uses fixed connectivity meshes. Singh et al [12] employ a sketch-based interface to interactively create 2D mosaics. More recently, research has been done on detail-preserving deformation [13] that does not drastically affect the overall aspect of a shape, like the stretching of one part. The method is capable of maintaining temporally coherent details on meshed surface stretch with local updates performed on the geometry and connectivity. The number of details evolve on an enlarging area. Though capable of decoupling mesh and detail resolutions, those methods are not intended for agglomerate material representation. Details are represented as high frequency changes applied on top of objects' surfaces.

Textures are also a fast and reliable way of adding information to meshed surfaces without requiring any additional geometry description. For that reason, they are natural candidates for representing composites. Tiled composite textures can be mapped to large surfaces by reusing texture cells. A composite could be created by aperiodic tile packing and good surface parameterization. This precludes the need for generating individual compositing elements on the surface. The resulting textured surface can also be made non-periodic, and retains most of the attributes of the tile packing. Most of the related research follows this strategy [14], [15], [16]. The tile packing is usually created from methods such as Wang Tiles and Corner tiles, which have an expensive preprocessing step: the construction of a good distribution, such as Poisson Disk Distribution, of the initial tile packing, and surface parameterization. The resulting tiled texture can be mapped on the fly to the mesh by means of Direct Stochastic algorithms implemented by hashing functions, as discussed in [17]. In order to generate the most photorealistic agglomerate materials, recent research has given attention to 3D texture mapping to represent complex surface detail on parameterised surfaces [18], [19]. The main restriction to this kind of strategy to our intended application on dynamic meshes is that reparameterization after deformation can be time consuming or non-temporal preserving. Surface tracking methods can ensure consistent parameterization [20] and insertion of details [21] through animations even on large-scale surface deformations, but so far have not been able to do it in real-time.

A more direct approach, called texture bombing, involves sampling texture or mesh elements directly on the target surface [22]. Sampling variation and rendering efficiency can be

(a) An ant bridge reveals nature sculpting through agglomerates [1].

(b) An ant bridge sculpted in our framework.

(c) Deus Ex Machina [2]. The swarm of bugs forming the god's face illustrates an industry inspiration for our work.

Figure 2: Agglomerates.

improved by adding noise and by precomputing levels-of-detail [23]. A common limitation is the requirement of static connectivity and geometry. Changing the underlying mesh connectivity would require recalculating all texture and element particle maps. [24] uses 2D texture decals placed on the surface and performs local parameterization only, which allows for small deformations on the surface with consistent texturing and real-time rendering. It is not, however, targeted at accumulating or large scale deformations.

In the domain of surface sculpting and deformation, many techniques have been proposed. As discussed in [25], a number of sculpting frameworks use volumetric models with adaptive topology. These systems operate by locally carving and merging materials under the action of the user. Though they are straightforward to implement, they usually have large storage requirements, as well as sampling and continuation issues, as the surface needs to be constantly re-extracted.

Concerning boundary representations, a number of works deal with the editing of point-based surfaces [26], while others deal with meshes. First spatial warping techniques [27] that were developed to determine displacement vectors in adaptive meshes, mainly depended on the use of haptic interfaces in order to become intuitive. Since then, several applications managed to deform and refine a mesh in real time, by embedding it into a deformation field [28]. A deformation field assigns a displacement to each point of the evolving surface to produce local perturbations [27]. Those deformation fields may be defined in the ambient space, while still holding properties such as volume preservation [29], [30] or detail preservation [31] of the shape being deformed. They can also depend on differential properties of the surface or even on features to be preserved [32]. Finally, Laplacian editing [33] can be used to extend the deformation field between a fixed part of the shape being deformed and a handle. An extension of Laplacian editing can be used to preserve details when stretching adaptive meshes [34]. Although such systems can convey a wider range of operations, the performance cost related to updating and solving sparse linear systems usually restricts their use, in the context of solid sculpting, to mesh posing and detail-preserving deformations. Furthermore, the fast intersection detection and prevention between multiple rigid objects has been vastly studied, but solutions tend to be too slow for real-time [35].

2. Overview

In this research, we address the problem of sculpting shapes composed of small volumetric elements. For now, we exclude the use of shapes considered "too elongated", such as hair fibers, since they can be seen as lower dimensional objects. Furthermore, we are interested in shapes whose component elements can be distant or loosely glued to each other. Under the action of a deformation field, two touching elements should be able to separate, if so required. In Section 3, we will suggest how these shapes can be empirically described by a system of elements referred to as CompEls. This system relies on a uniform sampling of the evolving shape to describe the relative positions of CompEls to one another, without implementing interaction forces, but only by identifying local area variations on the shapes envelope. Section 4 then describes how to initialise the CompEls coverage of the shape, and how to update it under the action of a deformation field that controls surface vertices' and CompEls' positions alike, even when it does not correspond to an isometric deformation. Intuitively, CompEls undergo the same deformation of the faces of the support mesh, whose vertices present uniform sampling, except when a triangle disappears or is split. We suggest fast redistribution schemes aiming at little to no CompEls displacement in Section 4.2. While splitting a triangle is quite natural, the transfer of CompEls from a destroyed triangle imposes to make room in neighboring triangles by stretching sub-regions. The heuristics proposed only use simple barycentric coordinates tests and computation of pairs of closest points between two edges. Finally, we discuss in Section 5 how to use the graphic card to perform CompEl variation, visual intersection prevention and to speed up the CompEl-made

shapes' display, before presenting numerical results in Section 6 and concluding in Section 7.

3. A new system for empirical description of composite shapes

3.1. Avoiding management of interactions between components?

Whenever a composite shape is deformed, its elements are displaced under the action of the deformation. The nature of the interactions between components that collide can be very complex to model, leading to numerical systems that are difficult to solve in real time. Even if not physically accurate, we decided to make a compromise by basing our approach on empirical descriptions instead. We stress that our goal is to offer the user a magnified experience of sculpture, still in real time, and we do not necessarily adhere faithfully to the physics of the real world. The observations we have decided to focus on are the followings:

- Elements that move toward and touch each other are clamped in their movements, they cannot get closer. This implies a change in their position at this time step, compared to the expected position that would have been reached in absence of collision (position induced by the local deformation of the shape exclusively). Among the elements visible from the outside, some only undergo tangential additional movement on the surface, while others are pushed inside the shape and disappear behind other elements.
- Conversely, when the deformation field separates elements that were previously in contact, new elements appear between the distanced CompEls, emerging from the interior of the shape.

3.2. Maintaining a uniform set of anchors on the shape

Since we are interested in empirically simplifying the deformation of a compound shape, it may be worthwhile to get rid of internal components, and to focus on visible elements only. These are the elements of the outer layers of the evolving shape. We also need descriptors in order to locally identify the effects of deformation on the surface of the shape. Indeed, we aim to know wherever the surface is contracting and wherever it expands. This will be useful to avoid computing exact collisions between components. To this end, we propose to maintain a uniform sampling of a rough envelope of the shape with a set of anchors. Wherever the shape is deformed isometrically, the sampling should remain stable. In contrast, the emergence of new samples is an indicator of the areas of expansion, where new components should become visible. Finally, the disappearance of samples will identify areas where the shape locally contracts.

It is important, therefore, to locate the components that we track with respect to the uniform sampling. We do not aim to couple each component with a sample, since we do not want the elements to be distributed regularly over the surface. Furthermore, we want the number of samples to be significantly smaller than the number of components. We suggest connecting the samples into a triangulated mesh to anchor the components therein. For that aim, we focused on the quasi-uniform meshes proposed by [32] to maintain a nearly uniform sampling of a shape that is sculpted under the action of deformation fields but other systems are also possible. Quasi-uniform meshes are characterised by a level of detail d, which technically corresponds to the maximum distance between two adjacent samples, as well as a minimum thickness t, below which the shape breaks locally, as described in [32]. Unlike similar systems [36] previously proposed in image processing, quasi-uniform meshes do not guarantee a minimum length for the edges but still promote the elimination of edges that are too short. Whenever the surface is deformed, edges that have become too long are broken by a sequence of flips or splits. This means that samples becoming distant are not connected anymore, and that new samples are inserted in between whenever a split is performed. That sequence of long edges destruction is followed by a sequence of merging short edge extremities (those with length smaller than $d/2$). When this step is over, all the edges that once again grew larger than d are flipped or split. The proposed algorithm optimises the uniformity of the edges, and does not require any additional step. It also does not require a relaxation loop. The resulting mesh is only consistent with the constraint regarding the level of detail d. The uniformity obtained in practice turns out to be sufficient for our purpose. Finally, as described in [32], in order to ensure an accurate tracking of the deformation (at the precision d) and of the changes in topological genus, we recall that deformations have to be subdivided into smaller steps d_{move} such that $4d_{move}^2 \leq t^2 - d^2/3$.

3.3. Description of outer layers of a shape by a system of CompEls.

We suggest to use quasi-uniform meshes as an anchor for the component elements of the shape. Given a quasi-uniform mesh of the input shape, we propose to populate it with a set of elements that we shall refer to individually as CompEl (Composing Element) hereafter. Each CompEl is anchored into a triangle and is characterised by its barycentric coordinates within that triangle, by its depth level with respect to the envelope of the object and by its orientation (rotation) relative to its support triangle.

4. Generation and update of CompEls under deformation

4.1. Initialisation

The initial distribution of CompEls is obtained using an accelerated dart throwing technique on an input quasi-uniform mesh (it is easy to transform any triangular manifold mesh into a quasi-uniform mesh using a sequence of edge flips, edge splits and edge collapses). The user should only provide the target density e corresponding to the number of CompEls that are visible or partially visible per area unit and the number of layers n over which the CompEls are distributed. CompEls are then inserted in the faces by randomly picking their barycentric coordinates, their orientation and their level of deepness. The method

is fully parallelizable, as it only requires information from the given face.

Considering that the faces may not exactly have the same area, each face receives a number of CompEls proportional to its area. The depth level is just an artificial information to be used by the graphic card during the rendering step, as described in section 5. If one wishes to sculpt a shape with the appearance of a plain object, it is important to choose e sufficiently large such that e*(projected area of a CompEl) is greater than the area of an equilateral triangle with edge length d. In order to ensure small area triangles to contain CompEls, we also set a minimum threshold on the number of *CompEls*. This naive initialisation is simple and powerful enough to create a good jittered grid-like distribution, provided that the triangles are approximately regular, with a controlled area.

4.2. *Temporal coherence and continuity under freestyle sculpting*

Whenever the shape is changing under the action of a deformation field, the vertices of the quasi-uniform mesh move accordingly and the triangles are deformed. As described above, the outgrown triangles are automatically cut into smaller triangles and small triangles collapse in turn. It is important to set rules for internal evolution and transfer of the CompEls between those triangles. Wherever the shape undergoes a rigid transformation locally, the triangles keep their initial dimension, and the CompEls do not change their position with respect to their support faces. Anywhere else, we suggest the evolution of the CompEls to be directly coupled with the elementary operations performed on the triangles: flip, split and collapse of the edges. The following rules should be interpreted as CompEls migration rules, from the triangles being destroyed towards the triangles being created and towards the neighboring triangles whose dimensions are impacted by the operation. Those rules remain local and simple, while still ensuring temporal continuity of CompEls. Thus, the evolution of the resulting system does not relate any more to some force equalizing system as is the case with a physical system. The update can be performed in real-time, possibly involving parallelism.

After each operation, the proportion of CompEls per unit of area is maintained by creating new CompEls in any triangle with a shortage in the number of CompEls. These are created in deeper layers. When a layer of depth is not sufficiently represented in a triangle, the update also picks a few CompEls randomly from deeper layers and promotes them to upper layers. Similarly, CompEls can be destroyed in deeper layers whenever a triangle is too crowded. With this maintenance strategy, visual artifacts related to the sudden insertion or removal of CompEls are minimised or overcome entirely, as those operations are performed behind the elements on the surface.

4.2.1. *Edge split*

Splitting an edge that has grown beyond d after applying a deformation results into replacing its two incident triangles by four triangles. We first address the case where the edge is cut in the middle. We suggest a very simple CompEls coordinate update within these triangles that ensures that CompEls remain where they were positioned before the division step. Figure 3 illustrates the technique on a face f affected by an edge split.

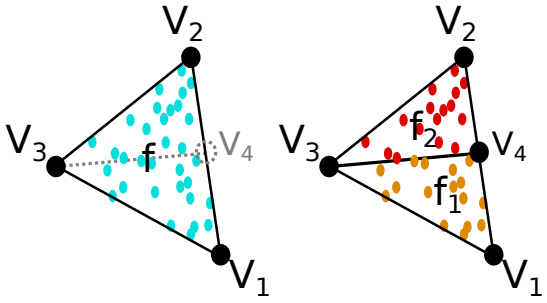


Figure 3: Splitting an edge at its center. Left: CompEls in f before split. Right: Distribution of CompEls between f_1 and f_2 after V_4 splits the edge V_1V_2 at its center.

Split update. Let (α, β, γ) be the barycentric coordinates of a CompEl C with respect to the vertices V_1, V_2 and V_3 of a face f to be split on V_1V_2 ($C = \alpha V_1 + \beta V_2 + \gamma V_3$), a straightforward implementation to the CompEls distribution between each of the two resulting faces f_1 and f_2 is simply to transmit to f_1 the CompEls whose barycentric coordinate with respect to V_1 is bigger than that to V_2, and the remaining to f_2. The barycentric coordinates of the CompEls attached to f_1 ($\alpha > \beta$) thus evolve into $(\alpha - \beta, 2\beta, \gamma)$ with respect to V_1, V_4 and V_3.

Proof. As we know that the triangle is split in half with the insertion of a vertex V_4 in V_1V_2 ($V_1 + V_2 = 2V_4$), we can update the barycentric coordinates straight away.

$$C = \alpha V_1 + \beta(2V_4 - V_1) + \gamma V_3 = (\alpha - \beta)V_1 + 2\beta V_4 + \gamma V_3$$

The barycentric coordinates of the CompEls attached to f_1 ($\alpha > \beta$) thus evolve into $(\alpha - \beta, 2\beta, \gamma)$ with respect to V_1, V_4 and V_3.

In the case where a deformation is decomposed into steps that are sufficiently small with respect to d and the minimum thickness t, the split of an edge using its middle is sufficient to track the deformation with precision d [32]. However, if we want to improve the quality of the resulting anchor mesh, we propose choosing another position for the added vertex that better corresponds to the sampling strategy proposed by [30]. Whenever an edge is split, the new vertex is positioned at the location corresponding to the deformation of the middle of the initial edge (the one before deformation) (see left Figure 4). This position may not be equidistant from the two extremities of the deformed edge. We adapt the update of the CompEls so as to take these relative distances into account, while still providing a fast scheme. We calculate the ratio between the distances $r_1 = \|V_1V_4\|$ and $r_2 = \|V_2V_4\|$, and use that ratio $r = r_2/r_1$ as the new threshold for CompEls placement in f_1 or f_2. An illustration of the modified procedure is shown in Figure 4. We therefore avoid overly sparse regions in f_1 and overly crowded regions in f_2.

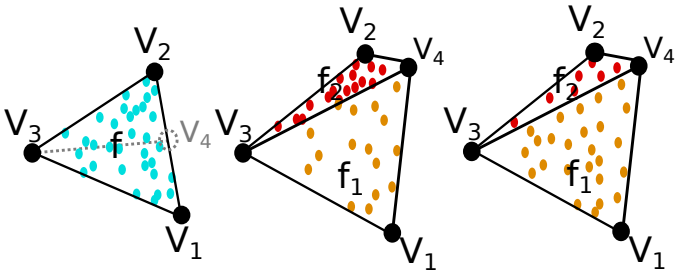


Figure 4: Modified edge split algorithm. Left: f before split of V_1V_2, and before shifting the initial midpoint V_4 to its deformed position. Center: naive approach, that distributes the CompEls with respect to the initial midpoint of V_1 and V_2 and that does not adapt to the difference between the areas being deformed. Right: modified approach, taking into account the area difference.

4.2.2. Edge collapse

The collapse of an edge VV' affects all the triangles that are incident to V or V': the two triangles that are incident to the edge VV' disappear and the remaining are distorted. In the case where the surface is nearly planar, the migration of CompEls could be obtained by superposing the configurations of the triangles before and after the edge collapse, with the initial CompEls being projected on the remaining triangles. This amounts to not moving CompEls during redistribution (see top Figure 5). This strategy has already been proposed by [37] for traditional texture update on edge collapse algorithms. The overlay of the original and modified projected connectivities results in a set of convex polygonal cells, whose determination imposes the calculation a number of points that, in the worst case, is proportional to the square of the number of neighbouring edges [38]. Time requirement to correctly place CompEls during the edge collapses at a given frame can therefore increase. Furthermore, if the surface is not flat, the extension of this approach to the general case requires the determination of a projection plane that is suitable for all the participant triangles, before and after deformation. Especially, it is necessary to avoid the folding of the projected surface. The choice of such a plane is not always obvious or legitimate. We propose a simple heuristic that is an order of complexity faster and that does not require explicit projection calculations (see bottom Figure 5).

Among the faces that are affected by the collapse of an edge VW, the two faces VWX and WVY are destroyed. The other affected faces are maintained, but their vertex incident to the collapsed edge (V or W) is moved to the midpoint V' of VW. We suggest only redistributing CompEls on destroyed faces to the remaining ones. In our heuristic, the surviving faces also keep their own original CompEls, but those are redistributed over a sub-region (of triangular or quadrangular shape), in order to make room for the CompEls of the triangles being destroyed.

Collapse update. Let $V_1, .., V_n$ denote the vertices incident to the new vertex V' that are located counterclockwise between X and Y, and $W_1, .., W_m$ the vertices that are located counterclockwise between Y and X. We execute our Compel distribution by visiting the faces around V' in a specified order.

In order to determine the CompEls that migrate from VWX to $V'XV_1$, our approach computes the pair of closest points P_1 and P'_1 that are located on the straight lines (VX) and $(V'V_1)$ respectively (as illustrated in Figure 6). If P_1 and P'_1 are located within the interior of the segments VX and $V'V_1$ respectively, these two points can be used to define a kind of local projection of the CompEls (see illustration in Figure 5 and 6):

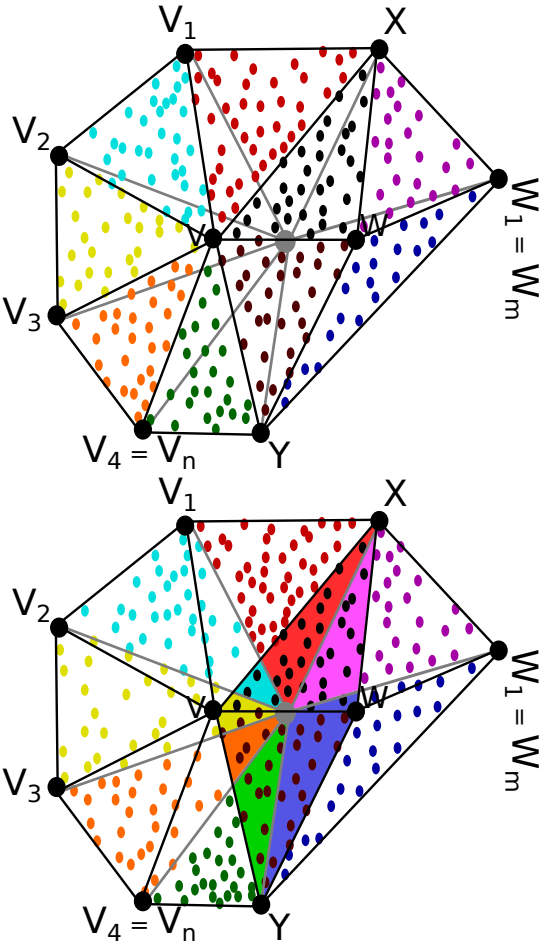


Figure 5: CompEls redistribution on edge collapse performed on a planar surface. In both images, the original (in grey) and modified (black) connectivities are shown overlaid. CompEls are colored according to their originating faces. Top: CompEls are shown in their initial positions (that also correspond to their projections on the new faces, without moving: in that case, the determination of the resulting barycentric coordinates would requires search for the adequate face). Bottom: Compel distribution on edge collapse by using the proposed heuristic. CompEls are shown in their final position: they are pushed within their originating faces being deformed to make room for the CompEls of the faces being destroyed (colored areas illustrate the reassignment).

- CompEls initially located in VXV_1 are pushed above segment XP'_1 on the resulting face $V'XV_1$. Everything happens as if the point V were pushed towards P'_1, which means that the barycentric coordinates (α, β, γ) of a CompEl C expressed in P'_1XV_1 are the same as its original coordinates in VXV_1. We say that the triangle VXV_1 is lifted to P'_1XV_1. Since we know the barycentric coordinates (μ'_1, ν'_1) of P'_1 with respect to V_1 and V', the new coordinates of C in the triangle $V'XV_1$ are $(\alpha\nu'_1, \beta, \alpha\mu'_1 + \gamma)$.
- CompEls of VWX that are initially located within $V'XP_1$ are lifted to the triangle $V'XP'_1$ by keeping their barycen-

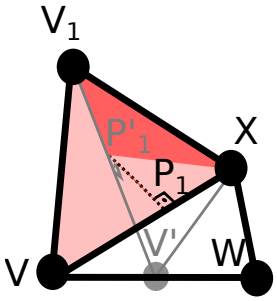


Figure 6: 3D illustration of region lifting, in the general case of non-planar surfaces. CompEls belonging to the triangle V_1VX (in light-red) are lifted to the subregion $V_1P'_1X$ (darker red) of triangle $V_1V'X$. The dotted line reveals the shortest segment between edges (VX) and and $(V'V_1)$.

tric coordinates unchanged within those two triangles. Therefore, their new coordinates within $V'XV_1$ can be updated using the coordinates (μ'_1, ν'_1) of P'_1 and the coordinates (μ_1, ν_1) of P_1 within VX.

We iteratively carry a similar treatment with CompEls contained in the triangles VV_iV_{i+1} for increasing values of i ranging from 1 to a stop condition defined below. In order to determine which of the triangle VWX's CompEls are reassigned to the triangle VV_iV_{i+1}, we compute the pair of closest points P_{i+1} and P'_{i+1} that are located on the straight lines (VX) and $(V'V_{i+1})$ respectively. If P_{i+1} and P'_{i+1} are located in the interior of the segments VX and $V'V_{i+1}$, the following migrations are performed:

- CompEls initially located in VV_iV_{i+1} are lifted above the segment $P'_iP'_{i+1}$ on the resulting face $V'V_iV_{i+1}$. This amounts to lifting the initial triangle VV_iV_{i+1} into the quadrangle $P'_iV_iV_{i+1}P'_{i+1}$. Let C be a CompEl (α, β, γ) in VV_iV_{i+1} and let C' be the position corresponding to the same coordinates (α, β, γ) within $V'V_iV_{i+1}$. The target position C" of the CompEl C will be placed along the line $(V'C')$: one determines the intersection $P'_{(i,C)}$ (resp. $V'_{(i,C)}$) of this line with $P'_iP'_{i+1}$ (resp. with V_iV_{i+1}) and we consider the linear mapping of $V'P'_{(i,C)}$ into $V'V_{(i,C)}$. C" is obtained as the image of C' by this linear mapping. This defines the lifting of C into C" (see illustration in Figure 5 for $i = 1$).
- CompEls of VWX that are initially located within $V'P_iP_{i+1}$ are lifted to the triangle $V'P'_iP'_{i+1}$ by keeping their barycentric coordinates unchanged within these two triangles.

Stop condition. The index i is incremented until we reach the value I such that P_{I+1} and P'_{I+1} are not located in the interior of the segments VX and $V'V_{I+1}$ (stop condition). In that case, V is projected onto P' within the triangle VV_IV_{I+1} (this is illustrated in triangle VV_2V_3 of Figure 5). The CompEls of VWX initially included in VP_IV are lifted to the triangle VP'_IP. It is also necessary to compute P" as the intersection of $V'P'$ with V_IV_{I+1}. CompEls located inside the triangle VV_IP", are lifted within the quadrangle $P'P'_IV_IP$". When the stop condition is met, we do not move to the next triangle and the index i is not incremented.

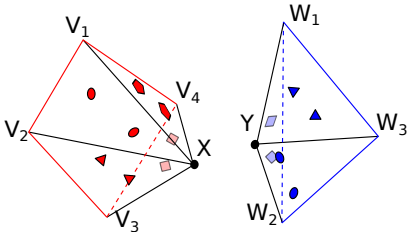
In the same way, if P_1 and P'_1 do not belong to the segments VX and $V'V_1$ respectively, we project V on the triangle $V'XV_1$ at point P' (and compute P" as described above). The subtriangle $V'XV$ (within VWX) is then lifted to $V'XP'$ (within $V'XV_1$). CompEls located inside the triangle VXP" are lifted to the triangle $P'XP$", and no iteration is performed on i.

The same process is performed starting from the edge VY in the direction of V_n, from WX in the direction of W_m and from the edge WY in the direction of W_1. If any triangle incident to vertex V' is not reached by these four iterations, it keeps its CompEls unchanged. Figure 5 illustrates the strategy's results and the CompEls' barycentric coordinates update.

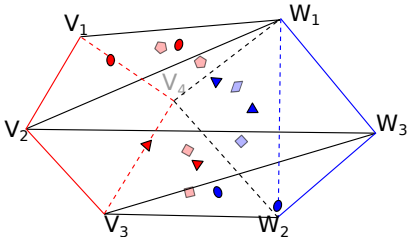
4.2.3. *Edge flip*

In nearly planar areas, edge flip is a preferred alternative to edge split and edge collapse, whenever it is sufficient to restore the level of detail d or the quality of a triangle. This favours a better stability of the quasi-uniform sampling since it prevents creation and destruction of vertices.

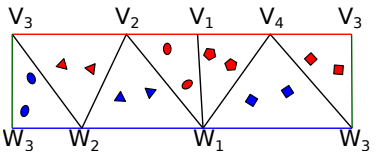
Whenever an edge V_1V_2 is replaced by its flipped counterpart V_3V_4, one first calculates the pair of points P and P' on the edges V_1V_2 and V_3V_4 that minimize the distance between the lines supported by these edges. Compel barycentric coordinates are first updated as if V_1V_2 was to be split in P. CompEls of the triangle PV_3V_1 (resp. PV_1V_4) are then lifted to $P'V_3V_1$ (resp. $P'V_1V_4$). Since P' is on V_3V_4 we then proceed the inverse of a split barycentric coordinate update to merge $P'V_3V_1$ and $P'V_1V_4$ into $V_4V_3V_1$. The same thing is done on the other side of V_4V_3.



(a) The original neighborhoods of two vertices before ring join.



(b) The two vertices are removed and the two neighborhoods are stitched together.



(c) Resulting connectivity unwind at the edge V_3W_3 (shown in green, duplicated)

Figure 7: CompEl redistribution on ring join operations.

4.2.4. Change in topological genus

Whenever the distance between two vertices located in different sides of the quasi-uniform mesh becomes smaller than the minimum thickness t, the shape locally flattens and breaks. This amounts anticipating intersections and changes in topological genus. This change is performed through vertex ring join of the two vertices [32]. It can cause major topological and visual changes in the mesh. In order to maintain a minimum temporal continuity, we propose to reuse destroyed facets to join the vertices. The faces and their corresponding CompEls are maintained by merely replacing the joining vertex with the selected one in the other ring. This effectively stitches both 1-rings together, as depicted in Figure 7.

5. Rendering

Storing and rendering, in real time, a large number of CompEls can be a considerable technical challenge. A single model can have tens of thousands of CompEl instances, which can hinder performances due to a sheer triangle count. Using the capabilities of the graphic card can be useful for that challenge. We also use the programable rendering pipeline to lower visual problems related to CompEls intersecting one another, which is a problem not commonly present in traditional 2D texture bombing. Finally, we will explain how our system is compatible with the introduction of variability in the models of CompEls being used without loosing the benefits of instancing. In this section, we outline the algorithms employed to improve the geometric quality, variability and rendering time of a large number of CompEls.

5.1. Variability

CompEl models are displayed using traditional instance-based rendering. This means that we use a single CompEl model, and that the graphic card performs the numerous mesh instantiations of this model. This alleviates memory requirements in the graphic card and allows us to render a large number of objects in real time. Variability can be introduced directly during instancing. Given a very small number of input models, we are able to synthesize a possibly large number of different CompEls. We control variations between CompEls originated from the same model by means of a randomisation seed, assigned to each element, that is used by shader operations.

The first source of diversification are textures. Increasing the number of different textures applied to CompEls can increase variability without increasing connectivity complexity. Different textures are stored and made available in an array. The one used can be picked on the fly using the randomisation seed. Figure 8 illustrates the concept.

We compound differentiation by adding low frequency noise to the positions of CompEl vertices. A low frequency noise can be efficiently generated in the graphic card by a number of algorithms. We opted for a Perlin Noise-like function, as defined in [39]. Noise intensity is evaluated using vertex normals offset by the randomisation seed. The resulting noise strength is applied as a vertex displacement along its normal direction (Figure 9).

Figure 8: The same rock mesh can be used with different textures to generate different CompEls.

Figure 9: Adding noise to a rock mesh can generate different pebbles.

5.2. "Back CompEl" Culling

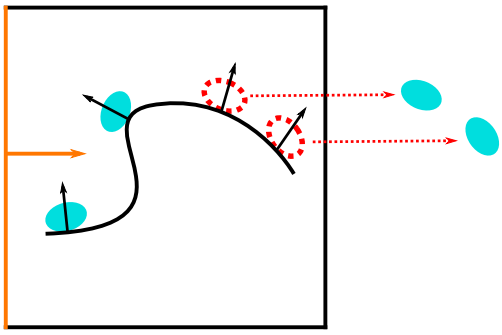
Backface culling and clipping algorithms are common place in modern graphic cards. CompEl meshes, after vertex processing in the rendering pipeline, can already be automatically clipped out. If they are not, however, there will usually be triangles on their surface facing the camera (always, if they are closed manifold inside the viewing frustum). In crowded composite objects, this may result in several hidden CompEls being passed on to primitive assembly and fragment processing.

We can alleviate the problem by discarding all the CompEls that face away from the camera. At the level of each vertex (in Normalized Device Coordinates (NDC)), we compare the quasi-uniform mesh normal closest to the CompEl's barycentre and the eye vector. If the CompEl is deemed backfacing, we displace the vertex outside the viewing frustum, resulting in the entirety of the CompEl being clipped. Figure 10 illustrates this technique.

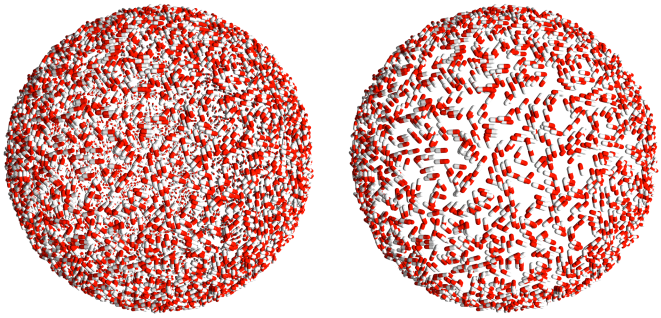
5.3. Visual intersection prevention

Intersections in a scene are a constant concern in computer graphics [40]. Applications that work with 3D objects clumped together usually solve this problem by making use of physical simulations, polygonal collision tests or other indirect methods, such as bounding boxes, to prevent pairs of rigid objects from intersecting each other. Collisions can not only be costly to identify, but also troublesome to solve when more than two objects are involved. Our framework aims at being fully local in order to better support interactive sculpting times, so we cannot resort to costly collision detection calculations.

To tackle the issue, we propose a CompEl squish in the rendering pipeline. After transformation to the Normalized Device Coordinates, we add a displacement to CompEl vertices perpendicular to the canvas. We move the vertex so that its depth approaches the same depth as its CompEl's projected center of mass. As per [40], this deformation can be represented as a simple linear scale in a cubic set of hyperpatches around the CompEl. This squish operator cannot cause any new self-intersections in a mesh, as long as the scale factor s is less



(a) CompEls distributed along a surface in NDC. The screen and the viewing vector are shown on the left. CompEls placed where the surface normal forms an angle smaller than 90° with the viewing vector, shown dotted in red, are sent outside the NDC to be clipped.



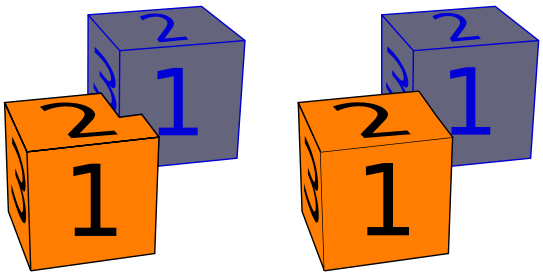
(b) Model rendered without culling. (c) Model rendered with culling.

Figure 10: Back CompEl culling.

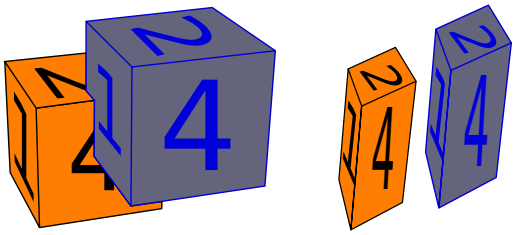
than 100%. Also, since the displacement is perpendicular to the canvas, only the intersection of objects with different centers of mass is altered. The rendering of individual CompEls is not affected. In the limit case, when $s = 100\%-$, all CompEls become equivalent to parallel 2D images, which cannot intersect (Figure 11). Though arguably ideal for static viewports, applying the squish operator with $s = 100\%-$ may generate animation artifacts. As the camera rotates, CompEls that were behind other CompEls may suddenly pop to the front. This is a direct consequence of CompEls whose centers of mass become closer to the canvas than the other CompEls' center of mass. It is also theoretically possible, though not observed in our tests, to experience z-fighting between squished CompEls. Both of these problems can be minimized by allowing some small intersection, through the reduction of s. This dilutes the popping effect for several frames. Notice also that it is possible for the squish to change the rendering of a CompEl that is only partly contained by the volume.

6. Results

Our CompEl redistribution algorithm greatly limits the number of CompEl insertions and deletions on all basic topological operations performed on the anchor mesh, as can be seen in Table 1 with respect to Figure 12. In our tests, the number of CompEls created and removed on deformation was reduced by more than 70% when compared to simple populating of newly



(a) View from the camera before (left) and after (right) squish.



(b) A hypothetical side view before (left) and after (right) squish reveals that the two intersecting CompEls had their vertices moved towards the depth of their respective centers of mass in the NDC.

Figure 11: Particle Squish.

inserted faces. Their numbers oscillate according to the desired CompEl frequency and the evolution of the surface's area, but are otherwise largely independent from the insertion or removal of faces and vertices.

Figure 12: Example sculpture created from a sphere. Left: pills used as CompEls show their relative frequency and texture distribution. Right: the final result. Time taken: 10-15 min.

Our local distributing algorithm, coupled with the good quality triangles favored by the underlying sculpting framework, ensure compliance with the desired CompEl density throughout the sculpting process.

The framework is capable of conveying several different types of composite materials. Figure 13 shows the resulting sculpture of dissociated elements. We also show that it can create significant element variation from a limited initial set.

The system allows for very smooth and intuitive sculpture at interactive rates, being suitable for the use of digital artists of all backgrounds, as shown in Figure 14.

Our frames per second are on par or more efficient than the

Figure 13: "Flight of the hummingbirds". CompEls sampled from simple quasi-uniform cloud meshes.

Table 1: CompEl creation and destruction for the sculpture in Figure 12. The number of CompEls is shown as a function of the required density e.

Variable	# Initial	# Created	# Removed	# Final
Edge Splits				41263
Edge Flips				13573
Edge Collapses				30122
Vertices Ring join				22
Area	12.52	N/A	N/A	8.78
Vertices	1026	41270	30197	12099
Faces	2048	82526	60384	24190
CompEls				
e =3986	48882	27469	41844	34507
e =797	8911	66	3855	5122
e =341	3328	14	1128	2214

rendering of a equivalent number of polygons due to our back CompEl culling algorithm. The performance, however, can be significantly diminished if CompEl meshes are dense.

Visual continuity is ensured by the seamless creation and destruction of CompEls on surface stretch and by their redistribution between faces affected by changes in connectivity.

Newly inserted elements sprout seamlessly from inside the sculpted object, giving the impression they were always there to begin with. Similarly, excess elements are first pushed from sight towards inside the shape before disappearing.

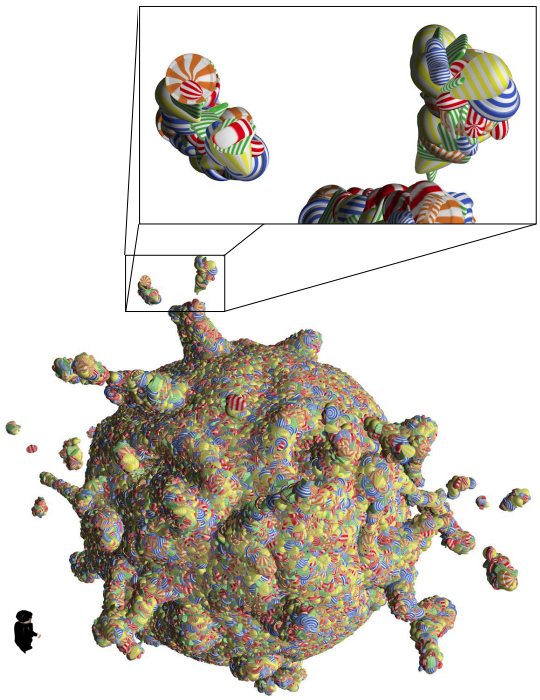


Figure 14: "My World". Example sculpture by a 6-year-old from candy meshes reveal how simple it is to sculpt with the framework. A zoomed view shows individual CompEls. The artist is shown on the left, looking at his creation.

7. Conclusion

Interactive virtual sculpting of composite shapes remains a challenge in Computer Graphics. We proposed an empirical approach that helps deforming an agglomeration of component elements in real time. We believe that a real-time approximate solution for these problems, such as ours, is of interest to the virtual creation of shapes. Sculpting is a creating process rather than a mere reproduction of a preconceived model. Therefore, the artist needs to assess the tentative result of his work on the fly, even if at an approximative scale, before the final quality rendering is performed. For this to be possible, we concentrate on the elements that are present on the most superficial layers of the shape being sculpted. Secondly, rather than modeling the interactions between the component elements, we only maintain the properties of their relative placement. This can be performed by setting and tracking a dynamic uniform sampling of the shape that is useful to characterize the local evolution of the surface. A CompEl results from the coupling between a com-

ponent element and a triangulation of the uniform sampling. Of course, such a system does not completely cover the problems of intersection between CompEls since we do not focus on those interactions. We do, however, employ visual "tricks" into the graphics pipeline to display elements as if they were entirely separate and non-overlapping. The results obtained so far are very satisfying and allow the user to create and deform composite shapes fluently. Future research could focus on the efficient use of CompEls with high-density meshes. Research could also be done to allow for the simultaneous sculpting of regions with different densities and other, non-uniform, CompEl distribution. Our perspective is now to extend this approach to more complex components, such as hair, to better use surface curvature information and to further exploit the possibilities offered by the graphics card.

Acknowledgements

Artur P. Sampaio acknowledges CAPES for the fellowship (PDSE Proc. 14495/13-9). This work was also funded by the Programme Avenir Lyon Saint-Etienne (PALSE) at the Université Claude Bernard Lyon 1.

[1] Land MD. Bridging environmental conservation and animal rights. 2016.

[2] Robertson B. The matrix resolution. Computer Graphics World 2003;.

[3] Ma C, Wei LY, Lefebvre S, Tong X. Dynamic element textures. ACM Trans Graph 2013;32(4):90:1–90:10. URL: http://doi.acm.org/10.1145/2461912.2461921. doi:10.1145/2461912.2461921.

[4] Emilien A, Vimont U, Cani MP, Poulin P, Benes B. Worldbrush: Interactive example-based synthesis of procedural virtual worlds. Proceedings of SIGGRAPH 2015;.

[5] Peytavie A, Galin E, Grosjean J, Mérillou S. Procedural Generation of Rock Piles Using Aperiodic Tiling. Computer Graphics Forum 2009;URL: https://hal-unilim.archives-ouvertes.fr/hal-01250531.

[6] Schwarz M, Stamminger M. On predicting visual popping in dynamic scenes. In: Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization. APGV '09; New York, NY, USA: ACM. ISBN 978-1-60558-743-1; 2009, p. 93–100. URL: http://doi.acm.org/10.1145/1620993.1621012. doi:10.1145/1620993.1621012.

[7] Medeiros E, Ingrid L, Pesco S, Silva C. Fast adaptive blue noise on polygonal surfaces. Graph Models 2014;76(1):17–29. URL: http://dx.doi.org/10.1016/j.gmod.2013.10.004. doi:10.1016/j.gmod.2013.10.004.

[8] Vanderhaeghe D, Barla P, Thollot J, Sillion FX. Dynamic point distribution for stroke-based rendering. In: Proceedings of the 18th Eurographics Conference on Rendering Techniques. EGSR'07; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 978-3-905673-52-4; 2007, p. 139–46. URL: http://dx.doi.org/10.2312/EGWR/EGSR07/139-146. doi:10.2312/EGWR/EGSR07/139-146.

[9] Ostromoukhov V, Donohue C, Jodoin PM. Fast hierarchical importance sampling with blue noise properties. In: ACM SIGGRAPH 2004 Papers. SIGGRAPH '04; New York, NY, USA: ACM; 2004, p. 488–95. URL: http://doi.acm.org/10.1145/1186562.1015750. doi:10.1145/1186562.1015750.

[10] Wachtel F, Pilleboue A, Coeurjolly D, Breeden K, Singh G, Cathelin G, et al. Fast tile-based adaptive sampling with user-specified fourier spectra. ACM Trans Graph 2014;33(4):56:1–56:11. URL: http://doi.acm.org/10.1145/2601097.2601107. doi:10.1145/2601097.2601107.

[11] Pastor O, Freudenberg B, Strothotte T. Real-time animated stippling. Computer Graphics and Applications, IEEE 2003;23(4):62–8. doi:10.1109/MCG.2003.1210866.

[12] Abdrashitov R, Guy E, Yao J, Singh K. Mosaic: Sketch-based interface for creating digital decorative mosaics. In: Proceedings of the 4th Joint Symposium on Computational Aesthetics, Non-Photorealistic Animation and Rendering, and Sketch-Based Interfaces and Modeling. SBIM '14; New York, NY, USA: ACM. ISBN 978-1-4503-3018-3; 2014, p. 5–10. URL: http://doi.acm.org/10.1145/2630407.2630409. doi:10.1145/2630407.2630409.

[13] Rohmer D, Hahmann S, Cani MP. Real-Time Continuous Self Replicating Details for Shape Deformation. Computers and Graphics 2015;URL: https://hal.inria.fr/hal-01152928.

[14] Wei LY. Tile-based texture mapping on graphics hardware. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. HWWS '04; New York, NY, USA: ACM. ISBN 3-905673-15-0; 2004, p. 55–63. URL: http://doi.acm.org/10.1145/1058129.1058138. doi:10.1145/1058129.1058138.

[15] Li H, Wei LY, Sander PV, Fu CW. Anisotropic blue noise sampling. In: ACM SIGGRAPH Asia 2010 Papers. SIGGRAPH ASIA '10; New York, NY, USA: ACM. ISBN 978-1-4503-0439-9; 2010, p. 167:1–167:12. URL: http://doi.acm.org/10.1145/1866158.1866189. doi:10.1145/1866158.1866189.

[16] Vanhoey K, Sauvage B, Larue F, Dischler JM. On-the-fly multi-scale infinite texturing from example. ACM Trans Graph 2013;32(6):208:1–208:10. URL: http://doi.acm.org/10.1145/2508363.2508383. doi:10.1145/2508363.2508383.

[17] Lagae A, Kaplan CS, Fu CW, Ostromoukhov V, Deussen O. Tile-based methods for interactive applications. In: ACM SIGGRAPH 2008 Classes. SIGGRAPH '08; New York, NY, USA: ACM; 2008, p. 93:1–93:267. URL: http://doi.acm.org/10.1145/1401132.1401254. doi:10.1145/1401132.1401254.

[18] Koniaris C, Cosker D, Yang X, Mitchell K. Texture mapping techniques for volumetric mesostructure. Journal of Computer Graphics Techniques (JCGT) 2014;3(1):18–59. URL: http://jcgt.org/published/0003/01/02/.

[19] Meng J, Papas M, Habel R, Dachsbacher C, Marschner S, Gross M, et al. Multi-scale modeling and rendering of granular materials. ACM Trans Graph 2015;34(4):49:1–49:13. URL: http://doi.acm.org/10.1145/2766949. doi:10.1145/2766949.

[20] Bojsen-Hansen M, Li H, Wojtan C. Tracking surfaces with evolving topology. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 2012;31(4):53:1–53:10.

[21] Mercier O, Beauchemin C, Thuerey N, Kim T, Nowrouzezahrai D. Surface turbulence for particle-based liquid simulations. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2015) 2015;34(6).

[22] Dischler JM, Maritaud K, Lvy B, Ghazanfarpour D. Texture particles. Computer Graphics Forum 2002;21(3):401–10. URL: http://dx.doi.org/10.1111/1467-8659.t01-1-00600. doi:10.1111/1467-8659.t01-1-00600.

[23] Wang L, Shi Y, Chen Y, Popescu V. Just-in-time texture synthesis. Computer Graphics Forum 2013;32(1):126–38. URL: http://dx.doi.org/10.1111/cgf.12003. doi:10.1111/cgf.12003.

[24] de Groot E, Wyvill B, Barthe L, Nasri A, Lalonde P. Implicit decals: Interactive editing of repetitive patterns on surfaces. Computer Graphics Forum 2014;33:141–51.

[25] Aghdaii N, Younesy H, Zhang H. 5-6-7 Meshes. Proc of Graphics Interface 2012;.

[26] Zwicker M, Pauly M, Knoll O, Gross M. Pointshop 3d: An interactive system for point-based surface editing. SIGGRAPH 2002;.

[27] Gain J, Marais P. Warp sculpting. IEEE Transactions On Visualization And Computer Graphics 2005;11(2):217–227. doi:10.1109/TVCG.2005.36.

[28] Stanculescu L, Chaine R, Cani MP, Singh K. Sculpting multi-dimensional nested structures. Computers & Graphics 2013;37(6):753–63. doi:http://dx.doi.org/10.1016/j.cag.2013.05.010; shape Modeling International (SMI) Conference 2013.

[29] Angelidis A, Cani MP, Wyvill G, King S. Swirling-sweepers: Constant volume modeling. Graph Models 2006;68(4):324–32. Special Issue: Pacific Graphics 2004.

[30] von Funck W, Theisel H, Seidel HP. Vector field based shape deformations. In: ACM SIGGRAPH 2006 Papers. SIGGRAPH '06; New York, NY, USA: ACM. ISBN 1-59593-364-6; 2006, p.

1118–25. URL: http://doi.acm.org/10.1145/1179352.1142002. doi:10.1145/1179352.1142002.

[31] Sumner RW, Schmid J, Pauly M. Embedded deformation for shape manipulation. In: ACM SIGGRAPH 2007 Papers. SIGGRAPH '07; New York, NY, USA: ACM; 2007,URL: http://doi.acm.org/10.1145/1275808.1276478. doi:10.1145/1275808.1276478.

[32] Stanculescu L, Chaine R, Cani MP. Freestyle: Sculpting meshes with self-adaptive topology. Computers & Graphics-UK 2011;35(3, SI):614–622. doi:10.1016/j.cag.2011.03.033.

[33] Sorkine O, Cohen-Or D, Lipman Y, Alexa M, Rössl C, Seidel HP. Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. SGP '04; New York, NY, USA: ACM. ISBN 3-905673-13-4; 2004, p. 175–84. URL: http://doi.acm.org/10.1145/1057432.1057456. doi:10.1145/1057432.1057456.

[34] Dekkers E, Kobbelt L. Geometry seam carving. Comput Aided Des 2014;46:120–8. URL: http://dx.doi.org/10.1016/j.cad.2013.08.024. doi:10.1016/j.cad.2013.08.024.

[35] Tang M, Manocha D, Tong R. Fast continuous collision detection using deforming non-penetration filters. In: I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. New York, NY, USA: ACM. ISBN 978-1-60558-939-8; 2010, p. 7–13. doi:http://doi.acm.org/10.1145/1730804.1730806.

[36] Lachaud JO, Montanvert A. Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction. Medical Image Analysis 1999;3(2):187 – 207. doi:http://dx.doi.org/10.1016/S1361-8415(99)80006-1.

[37] Hale JG. Texture re-mapping for decimated polygonal meshes. Tech. Rep.; Edinburgh University; 1998.

[38] Cohen J, Manocha D, Olano M. Simplifying polygonal models using successive mappings. In: Proceedings of the 8th Conference on Visualization '97. VIS '97; Los Alamitos, CA, USA: IEEE Computer Society Press. ISBN 1-58113-011-2; 1997, p. 395–ff. URL: http://dl.acm.org/citation.cfm?id=266989.267108.

[39] Gustavson S. Simplex noise demystified. 2005.

[40] Gain JE, Dodgson NA. Preventing self-intersection under free-form deformation. IEEE Transactions On Visualization And Computer Graphics 2001;7:289–98.