

# Semi-automatically Augmenting Virtual Environments with Plausibly Located Clutter

Russell Broughton  
Department of Computer Science  
University of Manchester  
russ@cs.man.ac.uk

Toby Howard  
Department of Computer Science  
University of Manchester  
toby@cs.man.ac.uk

## Abstract

*This paper presents an approach for semi-automatically augmenting existing virtual environments with sensibly and plausibly distributed features which model the clutter we find in real-world scenes. It focuses on the stages of identifying sensible potential clutter placement regions and then describes a novel use of parametric surfaces as a generalised means of specifying distribution patterns within these regions.*

Keywords: Clutter, Augmentation, Parametric Surfaces, Virtual Environments

## 1. Introduction

As continual advances in computer graphics hardware increases the number of polygons available to Virtual Environment (VE) designers, new possibilities become practical in both the range of VEs that can be modelled and the level of detail to which they can be represented. However, the toolset available for creating new VEs has not evolved at the same pace as the hardware, meaning that the time taken by designers to realise these new possibilities is often prohibitive. This paper looks at supplementing this toolset with a semi-automatic approach for plausibly adding large numbers of existing models to existing VEs with the minimum of user interaction to generate what might collectively be referred to as ‘clutter.’ In comparison to scenes encountered in the real world, many VEs appear incongruous due to the relatively low number of (often highly-detailed) models that comprise them. Historically, this would have been due to the hardware of the time necessitating only the predominant objects required to approximate any given scene being present, but now the restraining factor has

shifted to the time taken to populate the VE with models in what is still a largely manual process.

This work-in-progress paper does not describe a complete solution to the task of semi-automatically augmenting VEs with clutter, but rather focuses on the first two aspects of the task: identifying the potential regions within any given VE that clutter could be added to and specifying how the clutter will be distributed over these regions. The process of actually selecting a model to add and incorporating it into the VE is the subject of current work.

## 2. Clutter

It is important to first clarify what we mean by the term ‘clutter’: we mean large numbers of, often small, “inconsequential items” that individually contribute little to a scene, but when viewed as a whole help to create a sense of disorder and naturalness which may diffuse the sense of sterility usually encountered in VEs. For example: in an office scene we might expect to see pens and pencils on desks; in a warehouse scene we might expect to see stacks of boxes, and so on. Whilst being able to add clutter that conforms to this definition is a goal of the work, the term is also used to encompass anything that, when added, increases the VE’s realism by virtue of its inclusion conforming to a viewer’s expectation that the presence (but not exact position) of such an object would be expected within a real environment of the type that the VE seeks to model.

## 3. Determining Clutter Regions

The only assumption made about the properties of the existing VE to which clutter will be algorithmically added is that its model’s vertices can be placed in a winged-edge type data structure [1] (modified to allow

holes and non-manifold surfaces) for fast polygonal adjacency queries. This implies that all polygon edges should be shared by either exactly two polygons or one polygon and boundary/hole and that all of the polygons should ideally be non-convex, as Figure 1 illustrates.

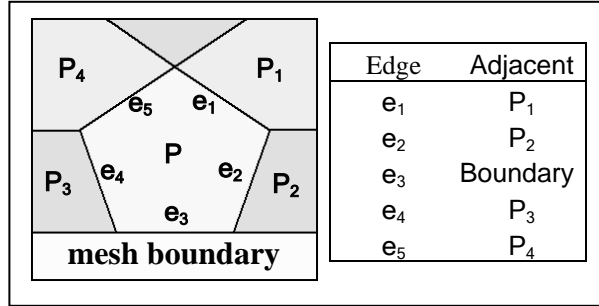


Figure 1: Adjacency structure for polygon P

Whilst clutter, by the definition above, could aptly refer to things such as posters stuck on walls, or towels draped over a thin rail, this work focuses on rigid body objects that will be inserted into the VE so as to be in a state of equilibrium (and so avoiding any incongruities of floating and obviously unstable objects). This state of equilibrium only considers gravity and normal contact forces and precludes objects held ‘artificially,’ such as a poster tacked to a wall. Hence the potential clutter regions are considered to be those that are ostensibly flat.

A flat surface is considered to be one that is composed entirely of adjacent polygons that all satisfy the constraint that they are orientated within  $\phi$  degrees of a given global up vector (specifically, the negation of the gravity vector). An algorithm that efficiently determines continuous regions on models is now presented:

1. Establish three sets: *model*: Initialised to contain all of the model’s polygons; *untested*: Stores all untested polygons that border the current region (initially empty); *region*: Eventually contains a set of polygons that define a distinct region (initially empty).
2. Take  $U$  to represent the global normalised up vector.
3. Take a random polygon from *model* and move it to *untested*.
4. Let  $A$  be a random polygon taken from *untested*. Calculate its normalised normal vector  $N$ .
5. If  $(\text{acos}(N \cdot U) < \phi)$  then move  $A$  to *region* and its adjacent polygons still present in *model* to *untested*. Else discard  $A$ .
6. Repeat steps 4 & 5 until *untested* is empty, at which point store the polygons in *region* and empty it.

7. Repeat steps 3 to 6 until *model* is empty, at which point all of the regions on the model have been found.

The important point here is that no polygon is ever considered twice making this a fast operation.

The algorithm will generate a set of regions, many of which may be too small at the scale of the clutter being added. A minimum surface area constraint is applied to all of the regions found and those which fail are discarded. Finding the areas of the polygons that comprise regions is a relatively simple operation as detailed in [2].

Figure 2 shows the surfaces identified on a couch model, taking up global vector,  $U$ , as 5 degrees .

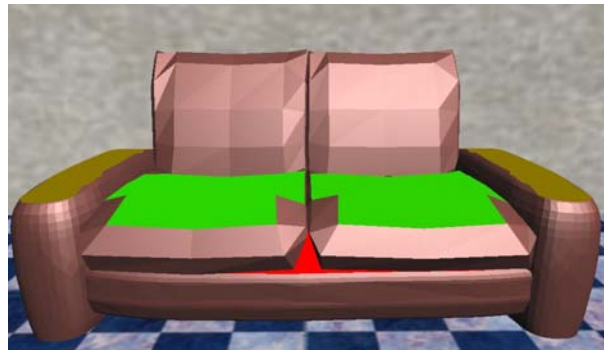


Figure 2: Clutter regions identified on a couch model

Since regions are determined at a purely local level, depending on the construction of the model, regions may be found that exist inside of models and hence may never be seen. However, it is also possible that such regions should be considered, such as the case of shelves in a closed cupboard, if it is known that it is possible to open the door and expose them. Such decisions are context-based, along with whether a region is even suitable for clutter, and so can only be made by the VE’s creator.

This form of region detection has one obvious disadvantage, in that it doesn’t cater for surfaces which are seemingly flat at the resolution of the objects being added to the scene, despite being comprised of individual polygons that would fail the orientation test; for example, imagine a car parked on a pebbledash drive. Fortunately, such surfaces rarely occur in practice, since surfaces containing fine-grain bumps are more likely to be approximated by a texture mapped single polygon, or perhaps using a more advanced rendering technique such as bump mapping.

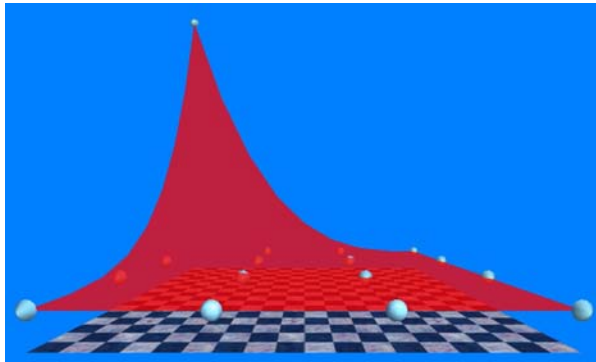
The inclusion of a precursory mesh simplification step is currently under consideration (using a basic algorithm that doesn’t preserve the original topology, such as [3]) as this would alleviate the problem of fine-

grain surfaces, as well as simplifying the task of ultimately adding clutter to the VE (perhaps by maintaining internal simplified regions that are tested against, instead of the actual geometry of the VE), albeit at the expense of accuracy.

It should be noted that the process of finding regions within any given VE is only a one-time operation which can be recorded for future use.

#### 4. Specifying Clutter Distribution

Having found the regions where clutter is to be added, the problem of specifying how it will be distributed within them now arises. The only way that every possible distribution pattern can be described is by specifying, and then somehow interpolating between, values at discrete points within the region. Irrespective of the method that is used to interpolate between them, it is very hard to visualise just what the distribution pattern that has been described is. The novel approach that this work takes to remedy this problem is to superimpose a parametric surface above each region that directly represents the distribution of clutter within it, whereby the height of the surface above each point within the region reflects the probability that clutter is added there. Figure 3 illustrates this principle with a Bézier patch being used to represent a distribution pattern, whereby the majority of clutter will be added to the far left corner of the region.



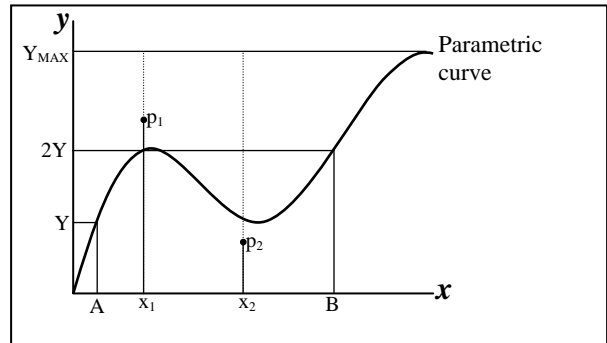
**Figure 3: Semi-transparent Bézier patch placed above target region**

This places a restriction on the parametric surface that it must be continuous, with no two points in the  $x$ - $z$  plane sharing the same  $y$  value (assuming that the  $y$  direction is taken as being the line in which gravity acts from hereon in). How this constraint is enforced depends on the type of parametric surface being used. A simple solution in the case of a Bézier patch is to restrict the motion of the control points to the  $y$  plane

(assuming that they are initially configured so as not to produce an overlap as in figure 3), although this imposes an additional limitation on the range of surfaces that can be expressed and so is the subject of current research

The main advantage of using parametric surfaces, such as Bézier patches, is that they are a well-understood modelling primitive that can be used to (in sufficient quantity) closely model any surface, meaning that any potential distribution pattern can be captured. In particular, the ability to subdivide sections of a patch means that the resolution to which distribution can be specified is variable over the region as required.

For any given point within a region, its likelihood of being selected relative to all of the other points is defined as the ratio of the height of its ‘surface point’ (used from hereon in to refer to the point on the patch directly above it) to the height of the highest other surface point within the region to which it belongs (no absolute scale is defined in this case). This can be seen, in 2D, in Figure 4:



**Figure 4: 2D parametric curve intersection**

In the Figure, point A has a  $y$  value of  $Y$  and B a value of  $2Y$ . Hence B should be considered for selection twice as often as A; the approach adopted to achieve this involves choosing a random point (herein, all references to random choices are taken to refer to fair, unbiased ones unless otherwise stated) within the volume defined under the curve. The  $x$  value of this chosen point is returned as the biased  $x$  value that has been selected. This complies with the example given in figure 4, since the probability that a point within the area directly above any given  $x$  value is selected is directly proportional to the height ( $y$  value) of its surface point. Hence, since B’s surface point’s height is twice A’s, it will be chosen twice as often over an infinite number of selections and so will appear twice as often as required.

A Monte Carlo approach is adopted to generate random points under the curve, whereby a random

value is first generated along the range of the curve in the  $x$  plane and then a corresponding  $y$  value in the range 0 to  $Y_{MAX}$  is also randomly generated. If the  $y$  value is below the curve at the  $x$  point, as is the case with  $p_2$  in figure 4, then the point is kept, whereas any points above, like  $p_1$ , are discarded. This process repeats until the required number of points has been generated.

In extending this to 3D, the 2D approach still holds true, but instead of generating a random point on a linear line, a random point within a polygonal region is generated that has a corresponding surface point  $y$  value. If this surface point's  $y$  value is less than a random  $y$  value generated between the range of the highest and lowest surface points, then it is considered to fall within the area defined by the parametric surface and is kept. Otherwise the point is discarded and the process repeats until the required number of points has been found, as before.

The calculation of the height of the parametric surface at any given  $x$ - $z$  value – the surface point  $y$  value – is performed on a polygonal rather than mathematical level, because of its applicability to any type of parametrically (or otherwise) defined surface that can be reduced to a polygonal representation.

#### 4.1 Generating a Random Point within a Polygonal Mesh

The region generation stage yields one or more polygons, where each polygon represents an infinite number of possible discrete points from which one should be chosen at random. In order to select a random point in the mesh, its polygons must first be triangulated, producing an unordered set of triangles. The robustness of the triangulation process defines the range of polygons in which a random point can be found. In this case, complex polygons of a convex nature and featuring holes may arise and so the existing 'triangle' Delaunay triangulation [4] package was adopted, as it is able to cope with these cases; although any other method of triangulation that can deal with the complex polygons produced (regardless of the quality of triangulation it produces), would also be suitable. In practice, the vast majority of polygons that are dealt with are of a simple (non-convex and without holes) nature and so employing a supplementary, simpler and quicker triangulation process in this case would yield quicker results. However, as the triangulation process need only be done once for each region, then this minor optimization isn't considered further.

Having reduced the mesh to a number of triangles, for each point to be generated, an area-weighted random selection of a triangle must be made from the set of all triangles (i.e. a triangle of area  $2X$  has twice the chance of being selected as one with area  $X$ ), reducing the problem to selecting a random point inside a triangle; a problem that has been solved by Turk [5] and involves the use of Barycentric coordinates (a coordinate system whereby points are identified within triangles by allocating the vertices weights thus defining a unique 'centre of gravity') with a random weighting being allocated to each of the three vertices, whose sum is 1. The whole process is shown in Figure 5.

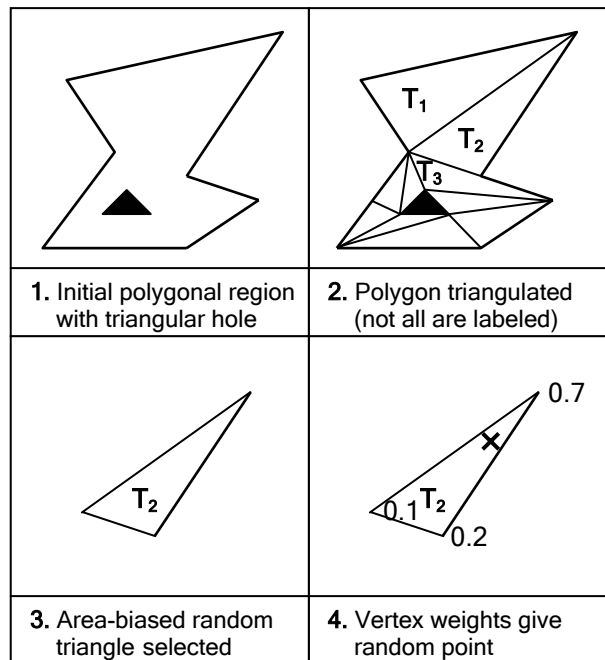


Figure 5: Random polygon point overview

#### 4.2 Results

A prototype implementation was created in MAVERIK (Manchester Virtual Environment Interface Kernel)[5][7][8], a toolkit for developing single-user VE applications. Figure 6 illustrates a patch and shows the resultant positions generated when using it to place 1000 points within the  $x$ - $z$  plane

#### 5. Future work

Other than the areas requiring future insight mentioned throughout this paper, current work is focusing on the extensive testing and evaluation of the approach described, in particular, with reference to

more traditional mathematical approaches used for specifying distribution patterns. Also, testing thus far has focused on the case of a single rectangular Bézier patch, this needs to be extended and the problem of using a number of patches to present a continuous surface above any given region considered.

The work detailed assumes that clutter distribution patterns are specified by hand. It is hoped that this process can eventually be automated, whereby an initial distribution suggestion is proposed to the user, which can then be modified if required. It is currently thought that this suggestion could perhaps be achieved with the aid of an initial ‘tagging’ stage, whereby the user manually associates a set of attributes with each model, which can then be used to reason about possible distribution patterns across it.

Ultimately, the actual selection of and addition of clutter at the positions generated needs to be considered. It is currently thought that clutter will be ‘dropped’ at these positions and then a basic mechanical model applied to bring the clutter to a state of equilibrium, using techniques similar to those described in [9]. However, such an approach would generally result in messy stacks of clutter, resembling what one might expect to find at a rubbish dump, rather than capturing the patterns found to occur in reality, like clutter commonly appearing in stacks. Hence work needs to be directed towards establishing these patterns and then looking at the possibility of adjusting the positions generated with reference to existing clutter within the VE, to create more realistic patterns.

## 6. Acknowledgements

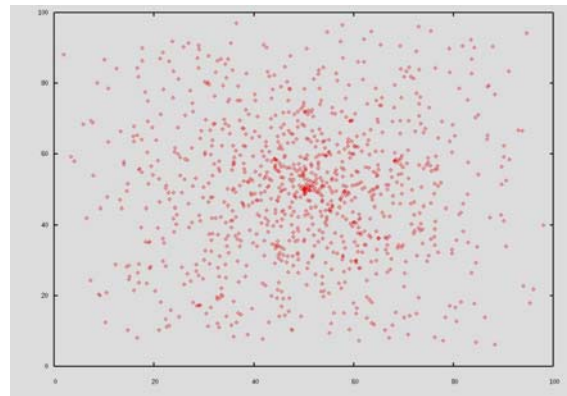
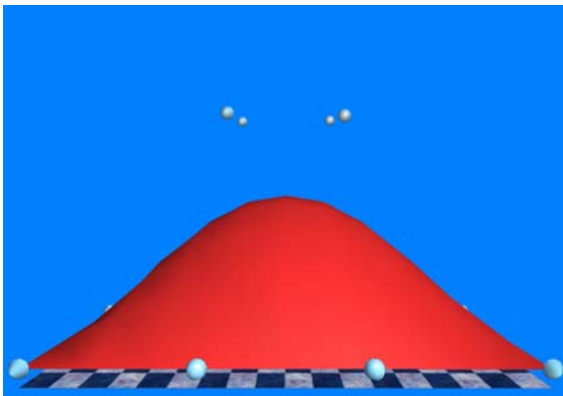


Figure 6: Example patch and the resultant distribution of 1000 points

It is a pleasure to thank all our colleagues in the Advanced Interfaces Group for their support and helpful suggestions.

## 7. References

- [1] B.G. Baumgart, “Geometric Modeling for Computer Vision, Ph.D. Dissertation, Stanford University, 1974
- [2] Ronald Goldman, "Area of Planar Polygons and Volume of Polyhedra", Graphics Gems II, Academic Press, 1991, pp. 170-171
- [3] W.J. Schroeder, J.A. Zarge, W.E. Lorensen, “Decimation of Triangle Meshes”, Computer Graphics, Volume 26, Number 2: 65-70, 1992
- [4] J.R. Shewchuk, “Triangle” Delaunay Triangulator, <http://www-2.cs.cmu.edu/~quake/triangle.html>
- [5] G. Turk, “Generating Random Points in Triangles”, Graphics Gems I, Academic Press, 1990, pp. 24-28
- [6] J. Cook, T.Howard (editors), “MAVERIK programmer's guide V5.0”, The Advanced Interfaces Group, University of Manchester, 1999
- [7] T. Howard, R. Hubbold, A. Murta, “GNU MAVERIK: A virtual reality system for research and teaching”, Proc. Computer Graphics & Visualisation in Education 99, 1999, pp. 85-91,
- [8] R. Hubbold, X. Dongbo, S. Gibson, “MAVERIK: the Manchester virtual environment interface kernel”, Virtual Environments and Scientific Visualisation, 1996
- [9] E. Guendelman, R. Bridson, R. Fedkiw, “Nonconvex Rigid Bodies with Stacking”, *Proceedings of the ACM Siggraph 2003*, Stanford University, 2003