Computational Aesthetics in Graphics, Visualization, and Imaging (2010)
O. Deussen and P. Jepp (Editors)

Procedural generation of surface detail for science fiction spaceships

Kate Kinnear and Craig S. Kaplan

University of Waterloo, Canada

Abstract

Procedural modelling can be used to generate digital content such as 3D digital models programmatically. In computer graphics, Procedural modelling has focused primarily on natural scenery and cityscapes. This paper considers the use of procedural modelling in a new domain: science fiction spaceships. We examine aesthetic principles as they relate to the beauty and visual interest of spaceships, especially surface details, and determine how these principles can be applied in a practical procedural modelling algorithm. We describe a prototype system that synthesizes and distributes surface details on large-scale spaceships. Given a surface representing the frame of a spaceship, we distribute geometry automatically in a coherent manner to achieve a characteristic science fiction aesthetic.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Spaceships permeate popular culture. The real-life vessels that have carried us to the moon and that we have flung into deep space are some of the ultimate symbols of humanity's technological progress. Perhaps even more so, the fictional craft of literature, movies, television and games give us an opportunity to dream of a grand future (or of a time Long Ago). Iconic, unforgettable ships such as the USS Enterprise and the Millennium Falcon are instantly recognizable by millions around the world.

The success of a science fiction franchise depends in part on the visual appeal of its technology. Ships must reflect the overall art direction of the franchise. Moreover, individual ships or fleets must fall into visually distinct styles that are tied aesthetically to the cultures or factions that use them. And of course, they must capture the imagination and paint a believable (or at least self-consistent) picture of space travel.

Prior to the almost universal adoption of computer graphics in this industry, there was a venerable tradition of physical model building associated with fictional spaceships. The masters of this domain were undoubtedly the pioneers at Industrial Light and Magic who created the ships for the original *Star Wars* movies [Pet06]. One notable feature of their process was *kitbashing*: the use of miscellaneous parts scavenged from commercial plastic models. The unplanned juxtaposition of pre-fabricated components lent their ships a deeply technological look. The high level of detail established a clear sense of scale.

Most modelling work is now done digitally. While there are obvious benefits associated with a digital pipeline, we feel that some of the visual appeal of the earlier physical models has been lost in translation. Designers are lacking the craft and serendipity of a collaborative, tactile process. Furthermore, in many cases the need for efficient rendering relegates detail to painted textures instead of geometry.

In this work we investigate this gap, and search for opportunities to recapture some of the visual style of physical spaceship models in the digital age. In particular, we focus on a seemingly tractable sub-problem: the generation of low-level surface details on large-scale ships (what model builders call *nurnies* or *greebles*). Our most direct sources of inspiration are the large models from *Star Wars*, particularly the Imperial Star Destroyer and the close-up model of the Death Star trench. The nurnies are the many small protusions and mechanical parts that adorn those surfaces.

DOI: 10.2312/COMPAESTH/COMPAESTH10/083-090

While these models represent just one point in a large design space, they have also had a profound, lasting influence on subsequent work.

1.1. Procedural modelling

We choose to consider the generation of spaceships as a problem of procedural modelling. Procedural modelling has enjoyed a long history in computer graphics, primarily as a means to create naturalistic structures like mountains and trees [EMP*02, PL90]. More recently there has been an increasing use of procedural modelling for architecture, ranging from individual buildings [WWSR03, MWH*06, LWW08, Teo09] to city plans and road layouts [PM01].

We believe that applying procedural modelling techniques to spaceship design could yield interesting insights and results. Despite the many successes of procedural modelling, it has not been applied to many different classes of object; this work can provide a new case to study. The potential benefits to spaceship designers are considerable, especially if procedural techniques could be used to create whole fleets of thematically related ships.

Spaceships represent an especially interesting domain for studying procedural modelling. Because they are fictional, they are not bound by any real constraints of physics or engineering; they can assume any form whatsoever. Of course, designs still adhere to many established conventions, but those conventions are more about conforming to (or departing from) human expectations and evoking associations with objects in our world. Furthermore, we can look to an extensive tradition of ship design for inspiration and examples.

This paper presents a procedural modelling technique that distributes small-scale details on the surfaces of large-scale spaceship surfaces. Our hope is to contribute a new domain to procedural modelling while sparking interest in recapturing the appeal of physical, kitbashed models.

2. Spaceship design and aesthetics

As part of this work, we reviewed the history of real and fictional spaceship design, and depictions of spaceships in visual and literary media. We conducted an extensive survey of spaceships, which cannot be reproduced here but appears in the thesis by Kinnear [Kin10]. In this section we analyze common features of spaceships and articulate some aesthetic criteria that will drive the development of the procedural modelling technique of Section 3.

2.1. Spaceship structure

We discuss the structure of spaceships in coarse-to-fine order, talking about the overall shape first and moving to progressively smaller details.

2.1.1. The frame

At the topmost level we speak of a spaceship's *frame*, its gross shape disregarding any small details. Although there are no universal rules for frame shape, there are conventions built upon our shared expectations. Many frames are clearly inspired by animal forms (particularly birds, fish and insects) and human vehicles (cars, planes, rockets). The connection to earthbound vehicles is a curious one. Many ships have streamlined, aerodynamic forms (sometimes including wings), even though one would assume this concern to be irrelevant in space. However, our everyday experience of speed is irrevocably tied to aerodynamics, and hence aerodynamic ships evoke speed in a way that others would not.

An important consideration in frame design is that a spaceship's frame should be clearly comprehensible in silhouette [Chi08]. This rule is well known in animation as the principle of staging used first by Disney animators and later re-expressed for computer animation at Pixar [Las87]. Some of the design rules used for Federation Starships in *Star Trek* can be seen as arising directly from the need to have strong silhouettes [Sch09].

2.1.2. Functional components

A *functional component* is a large-scale division of a spaceship's frame the fulfills a clear (though potentially fictional) purpose. Most spaceships exhibit some decomposition into functional components; the correspondence to our expectations of human-designed objects lends such ships a greater sense of plausibility. The most common functional components are the cockpit, wings, weapons systems and propulsion. Although functional components help lend a ship a sense of design, there are notable exceptions; the iconic Borg Cube from *Star Trek* made a profound mark on science fiction by deliberately flouting traditional conventions of frame shape and functional components.

2.1.3. Surface details

The look of a ship is greatly enhanced by the small details on its surface. Many fictional ship surfaces are inscribed with a network of *panels* that give them visual interest with relatively little effort. *Windows* can help imbue a ship with an overall sense of scale. *Pipes* help connect what may otherwise seem like random details on a ship surface, endowing those details with a greater sense of intent. *Nurnies* and *greebles* populate a surface with numerous fine details, and impart an overall sense of machinery and advanced technology.

When a spaceship's frame is symmetric, we have found that surface details tend to obey the frame's symmetries in proportion to their size and semantic importance. That is, larger details are more likely to be symmetric, whereas smaller details might be specialized on either side of the ship.

2.2. Spaceship aesthetics

Many theories of aesthetics might be brought to bear in studying spaceships. Because our goal to is distribute surface details, we discuss only those theories that we have found useful in that context. An extended discussion of the relationship between aesthetics and spaceship design appears in the thesis by Kinnear [Kin10].

In decorative arts, we seek to convey meaningful information to the viewer without confusing or overwhelming them. Eighteenth century philospher Francis Hutcheson offered a theory of *uniformity amidst variety* [Hut26], stressing the need to find a harmonious balance in design between order and complexity. Uniformity helps to organize a design, while variety provides a steady supply of visual interest. However, too much order can be dull, and too much complexity overwhelming. Gombrich echoed this sentiment: “Delight lies somewhere between boredom and confusion” [Gom75].

We can find the notion of uniformity amidst variety re-expressed in various guises over the centuries since Hutcheson articulated it. In the twentieth century, the mathematician Birkhoff developed a theory of aesthetics in terms of the relationship between order and complexity [Bir03]. In the language of information theory, we can see Gombrich’s “delight” as the sweet spot in a function that relates information content to aesthetic value [Bum05]. The grouping principles of Gestalt psychology fit naturally here as well, as an explanation of how the brain discovers the order that underlies a complex stimulus.

We use the principle of uniformity amidst variety as our guide in developing the procedural modelling algorithm of the following section. When a step in our algorithm introduces too much order to the result, we seek to counteract that order with a source of complexity, and vice-versa. When the algorithm exposes user-tunable parameters, we seek values for those parameters that balance between order and chaos.

3. Procedural synthesis

We have explored the foregoing aesthetic principles in the context of a proof-of-concept technique for placement of small-scale details on the surfaces of science-fiction spaceships. Given a description of the gross shape of the shaceship’s frame, our algorithm subdivides the frame into small rectangular panels, and decorates the panels with a combination of geometric primitives and kitbashed models. Our algorithm also constructs longer pipes that unite more distant parts of the spaceship, increasing the overall sense of uniformity.

For the purposes of our prototype algorithm, we assume a very simple description of a spaceship frame, one that is provided explicitly by the user. The frame is a collection of rectangles that we refer to as “nodes”. They can be positioned and oriented arbitrarily in 3D space. Frames constructed this way are limited in their ability to express the shapes of many popular spaceships, but they are adequate for the purposes of exploring the sense of order in placement of small surface details. Rectangles also suffice for depicting models like the Death Star trench [Pet06, Page 152], one of our main sources of inspiration in this work.

We also know in advance that we are interested in studying the use of symmetry in spaceship design. Thus, we designate one of our coordinate planes in advance as a plane of bilateral mirror symmetry. Any pairs of rectangles that are evenly placed on either side of that plane will be bound together during processing, so that the layout of detail on them may be approximately symmetric. If a rectangle is bisected by the plane of symmetry, it is divided in two and the two halves are treated as a symmetric pair.

3.1. Grid cells and sub-panels

Given a frame, our first goal is to split each node into smaller irregular panels that will define the layout of geometric detail. We accomplish this goal in two phases.

We do not want the panels to be too irregular, and so we first subdivide the node into a regular grid of *cells*. Nodes that are related by mirror symmetry are guaranteed to receive symmetric cell grids.

We would like our spaceship to contain a few large-scale structures that stick out from the general population of nurnies, structures that are guaranteeed to respect the symmetry of the frame. We immediately set aside some of the grid cells to serve as locations for these objects. In particular, we choose a random n-omino for a user-supplied value of n (we use $n = 4$ in our implementation), and mark n-ominoes at random locations in the grid of cells for each node. Symmetric nodes have their n-ominoes placed symmetrically across the mirror plane. We refer to these cells as “pattern cells”. The placement of geometry within pattern cells will be handled in Section 3.2. Figure 1(a) shows a node divided into grid cells, with pattern cells highlighted.

The remaining cells in the grid are too regular to enable the kind of variety we seek in the placement of nurnies. Therefore, we apply a randomized splitting algorithm to each cell to obfuscate cell edges and create a more irregular layout of smaller rectangles.

The core of this splitting algorithm consists of a step that splits a single rectangle into a few smaller ones. Let R be an arbitrary, axis-aligned rectangle. We choose a random point inside R, and draw any three of the four horizontal and vertical lines connecting this point to the edges of R. The choice of three lines yields a T junction inside the rectangle that strikes a visually appealing balance; we find that using all four lines leads to too much regularity.

Note that a point too close to the edges of R might produce a very narrow child rectangle. We mitigate this problem by

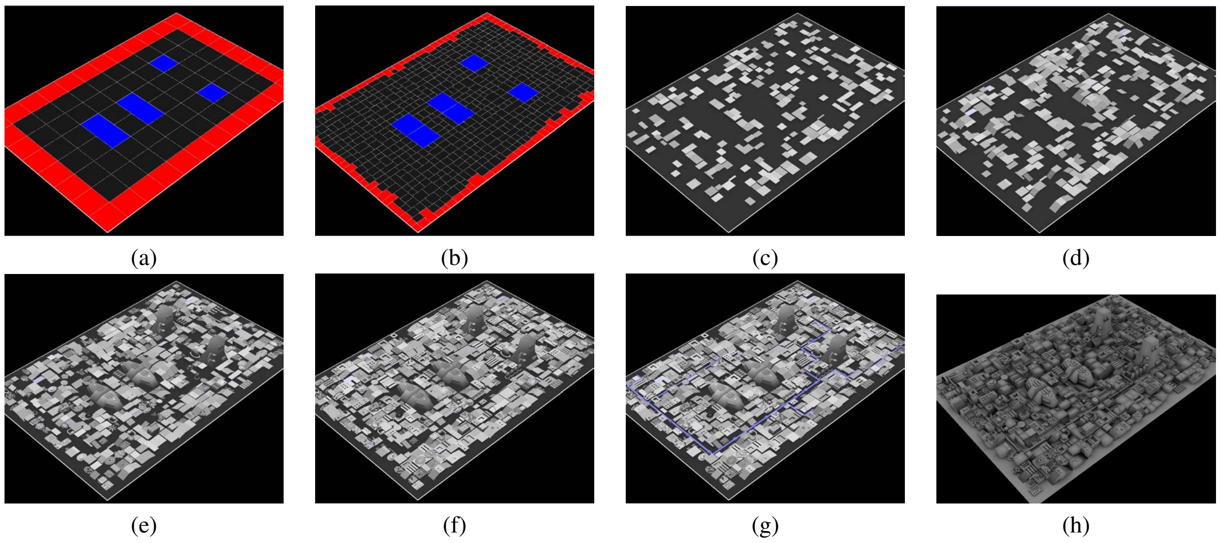


Figure 1: *A sequence of snapshots showing the phases in our procedural detail synthesis algorithm. An initial rectangular node is divided into grid cells (a), with reserved pattern cells coloured blue and edge cells coloured red. The grid cells are divided into sub-panels (b). Cuboids are added to some of the sub-panels (c), and some neighbouring sub-panels receive complementary geometry (wedges) (d). Geometry is assigned to all remaining sub-panels and the pattern cells (e). Secondary geometry is added to enhance detail (f). Finally, pipes are routed along edges between sub-panels (g). The finished surface is rendered with ambient occlusion in Houdini (h).*

constraining the split point to a scaled, centred copy of the rectangle. Conversely, if we happen to be given a narrow rectangle, we attempt to produce more squarish offspring. If the height of R is more than 2.5 times its width, we split using only a horizontal line instead of a T junction (and likewise if R is much wider than it is tall).

The preceding step splits a single rectangle into two or three children. We can split a grid cell any number of times by maintaining a list of the current rectangles within a cell, sorted in decreasing order of area. Each time we wish to split, we pick a random panel in the kth percentile of area or higher for this cell (where k is a user-defined constant, 20% in our implementation) and split it. The process can be repeated for any number of splits or until a desired number of cells is reached. In further processing steps, we will refer to the resulting rectangles as "sub-panels".

When creating sub-panels, we process each grid cell in conjunction with its symmetric partner, if any. When splitting a rectangle, there is a user-supplied probability that the split will be carried out symmetrically, increasing the ship's global regularity. If the split is symmetric, its children will then be considered for symmetric splitting, and so on.

Figure 1(b) shows the result of dividing grid cells into sub-panels.

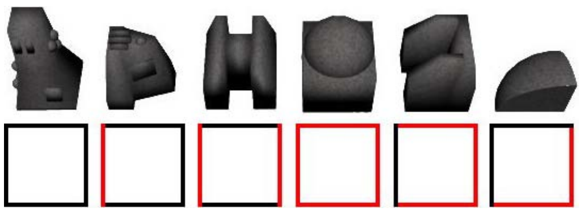


Figure 2: *The six models we use to fill pattern cells. In the square below each model, a black edge indicates a border with a non-pattern cell and a red edge indicates a border with a neighbouring pattern cell. The appropriate model is chosen based on the labelling of its neighbouring cells.*

3.2. Pattern cells

As stated previously, we use pattern cells to create large-scale regularities across the spaceship frame. To that end, we define special-purpose geometry that is used only in these cells. Every pattern cell is bounded on its four sides by pattern cells and ordinary grid cells. Taking into account symmetries, there are six distinct ways that these neighbours can be arranged. We define a family of six interrelated 3D models, each designed for one of those neighbour arrangements. The models we use in this paper are shown in Figure 2. They assemble into clusters in a manner reminiscent of Wang tiles [CSHD03], though unlike most uses of Wang tiles in computer graphics we do not incorporate any randomness into our configurations.

3.3. Nurnies

The irregular arrangement of sub-panels created above now serves as a guide for the placement of small detail geometry, or nurnies. Our nurnies are a combination of parameterized geometric primitives and instances of polygonal models. They are placed within sub-panels in a number of phases, described below.

For the most part, every sub-panel will eventually be augmented with geometry. However, we must treat sub-panels differently if they happen to have edges that lie on the boundaries of two different nodes (that is, a sub-panel adjacent to a seam between nodes). If the dihedral angle between two connected nodes is less than 180 degrees, we add geometry to the sub-panels only on one side of the seam. With this restriction, we avoid creating interpenetrating geometry on adjacent sub-panels.

In the first phase of nurnie placement, we add cuboids (rectangular prisms) randomly to a user-supplied fraction of the panels (30% in our prototype). Each cuboid is scaled to a random height within a user-specified range. Cuboids are the most flexible of our primitives, and the most compatible with other geometry. We add them first and use their placement as a guide for future phases. Figure 1(c) shows a node with cuboids added.

Next, we decide whether any cuboid faces might be candidates for "complementary geometry". Complementary geometry is any geometry that can be attached to a rectangular side of a cuboid to extend it naturally into a neighbouring sub-panel, adding to the overall order in the layout. In our implementation we experiment with wedges that start at the face of a cuboid and drop to the node surface over the space of a sub-panel (these wedges serve as a prototype for a broader range of connective elements that could be implemented in a similar manner). For each placed cuboid, we check which of its four edges are shared in their entirety with neighbouring sub-panels. For those edges we place a wedge in the neighbouring sub-panel with a user-specified probability (50% in our prototype). The wedge lines up with the corresponding face of the cuboid. In Figure 1(d), wedges have been added adjacent to some cuboids.

In the final placement step, the empty sub-panels that remain are given geometry. It is in this step that we wish to capture some of the fascinating aesthetic quality achieved by kitbashing in physical model building. Using Side Effects' Houdini,[†] we tore parts out of a variety of pre-existing 3D models obtained online.[‡] We arrived at a final kitbashing library of 70 models, shown in Figure 3.

[†] www.sidefx.com

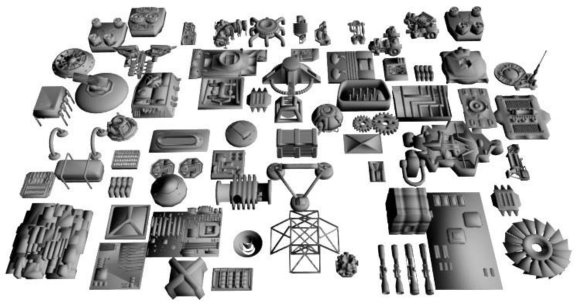


Figure 3: *The 70 models in our kitbashing library.*

The choice of models for the kitbashing library is necessarily subjective, but over time we found that some rules of thumb emerged. Tanks are the best source of parts, as they tend to be covered in complex weapons and esoteric mechanical components. Other useful models included four-wheelers, trains, helicopters, robots, internal car parts, weapons, computer components, electronics and spaceships. Car and plane exteriors were too smooth and featureless to be useful. We avoided shapes that are too immediately recognizable, since these can be distracting. We avoided overly complex geometry, which would clash with the overall level of detail in our ships and degrade under non-uniform scaling. Finally, we aimed for models with roughly flat bottoms and square footprints.

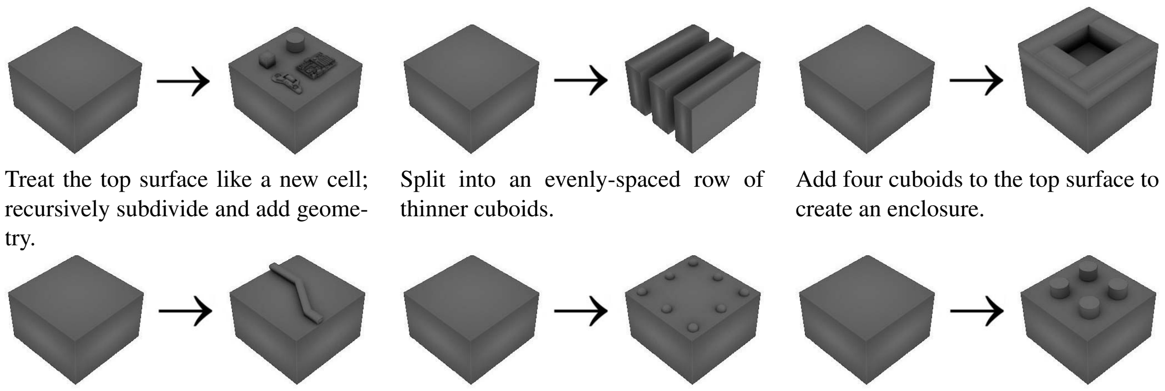
Each empty sub-panel is filled with either a geometric primitive (a cylinder, pyramid or cone) or a kitbashed model. All geometry is scaled non-uniformly to fit the sub-panel. However, in the case of kitbashed parts, if the x and y scaling factors differ by a factor of more than two, we opt instead to place multiple copies of the same part in a row. This change mitigates the distortion of non-uniform scaling, and creates clusters of repeated elements. Figure 1(e) shows the result of adding geometry for the pattern cells and all remaining sub-panels.

3.4. Secondary geometry

Once all of the initial geometry has been placed, we make a second pass over the node to refine the geometry and create a greater sense of stylistic unity.

First, inspired by the use of shape grammars in procedural modelling [SG72, WWSR03], we apply a set of substitution rules to eligible primitives in order to add low-level

[‡] See TurboSquid (www.turbosquid.com), 3DXtras (www.3dxtras.com) and Scifi-Meshes (www.scifi-meshes.com)



Treat the top surface like a new cell; recursively subdivide and add geometry.

Split into an evenly-spaced row of thinner cuboids.

Add four cuboids to the top surface to create an enclosure.

Add a division connecting opposite edges of the top surface.

Add hemispherical "bolts" near the outer edges of the top surface.

Add a regular grid of cylinders to the top surface.

Figure 4: *A set of six substitution rules for rectangular cuboid nurnies.*

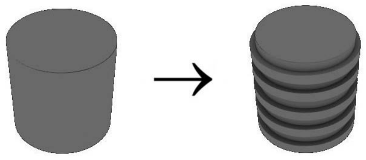


Figure 5: *A replacement rule for cylindrical nurnies: a cylinder is split into a stack of smaller, tiered cylinders.*

details. Our implementation uses a set of seven hard-coded rules, Illustrated in Figures 4 and 5. Most of the rules apply to arbitrary cuboids that were placed in the previous step (Figure 4). They add geometry to the cuboid, replace it with a set of related shapes, or recursively subdivide and decorate its top surface. One additional rule applies to cylindrical nurnies (Figure 5); it replaces a cylinder with a stack of cylinders of alternating radii. These substitution rules help give the sense that the different components of a ship are related to one another without resorting to overt repetition. The fact that they can be applied at different scales and at multiple recursive levels ensures that the final model exhibits detail at a range of scales.

Second, we search for opportunities to establish additional geometric connections between nurnies, particularly nurnies that come from the kitbashing library. A physical model builder would build such connections by adding additional parts that connect a nurnie to its neighbours. In our approach, we connect some nurnies to adjacent geometry using simple pipes with square cross sections. We identify candidate nurnies by searching for instances of kitbashed geometry containing flat regions large enough to accommodate such a pipe. We build vertical pipes upward from two nurnies beyond their maximum heights, and connect the vertical pipes with horizontal components. An example of such a connection is shown in Figure 6. Figure 1(f) adds secondary geometry to previously placed nurnies.

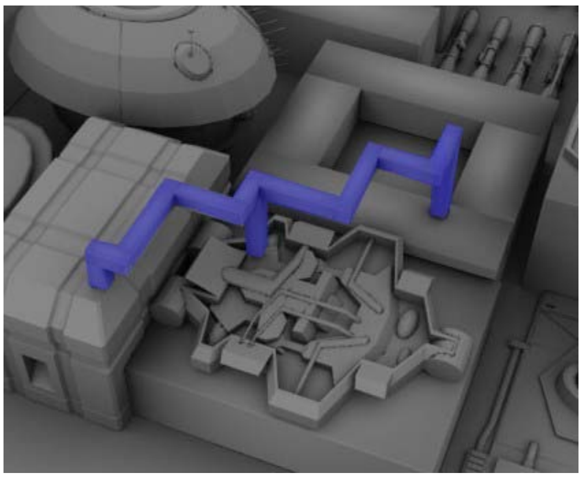


Figure 6: *A small square pipe (shown in blue), used to connect adjacent nurnies.*

3.5. Pipes

In the final step of our synthesis algorithm, we add pipes to each node in the model. Pipes help establish larger-scale connections across distant parts of the ship, lending an added sense of plausibility to the design by suggesting that there are functional relationships between widely scattered components.

To reinforce plausibility, these pipes should conform to our everyday experience of the placement of pipes in an industrial setting. They should be as short as possible, and bend as few times as possible. Pipes feeding nearby destinations should be merged into a single physical pipe that branches as late as possible. (It may also be possible to have multiple pipes running together in a parallel bundle.) In our implementation, we also make the decision that for each node, our pipe system will be modelled as a single source that routes to multiple destinations; more complex routing schemes could follow as a simple extension.

We route our pipes along the edges between sub-panels. Therefore, the first step in pipe routing is to construct a doubly-connected edge list representing the planar map induced by the sub-panels. We exclude edges separating connected pieces of geometry (such as cuboids and wedges), in order to avoid breaking these connections. A standard map overlay algorithm is certainly sufficient for constructing the graph [dBvKOS00]. However, the construction can be simpler in our case, because the map is formed from a collection of non-overlapping rectangles. We can reverse the process of panel subdivision to merge sub-maps back up into one planar map for each grid cell, then merge the grid cells into a master map. The merging operations can be ordered so that we need only ever consider a pair of maps that interface along a single line segment.

For each node in the frame, we now pick one source location for the pipe system and some number of destination locations. Each location is chosen as a random edge in the planar map.

To route pipes from the source to the multiple destinations, we construct a minimum-weight spanning tree of the node's planar map using a modified form of Dijkstra's algorithm. The cost of each edge is initially taken to be its length. Suppose now that a new edge (v,w) has just been added to the spanning tree, with v previously in the tree and w newly added. We consider the other edges adjacent to w (of which there are at most three in our planar map). We add a small penalty to the cost of the edges that are perpendicular to (v,w), in order to favour straight pipes.

When Dijkstra's algorithm is complete, we can use the spanning tree to trace pipes from each of the destinations back to the source. We then construct cylindrical geometry for these pipes, randomly adding additional cylinders to suggest pipe segments bolted together.

Of course, the nurnies were initially placed with no regard for the eventual paths of the pipes. To avoid having pipes interpenetrate with pre-existing geometry, we non-uniformly scale any nurnies adjacent to pipes in order to make additional room.

In Figure 1(g), pipes are added and the geometry for one frame node is completed.

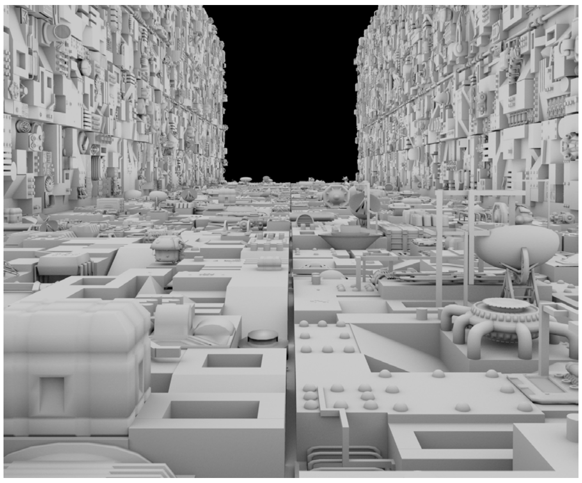


Figure 7: *A close-up of a space station trench, rendered using the Sunflow render system (sunflow.sourceforge.net).*

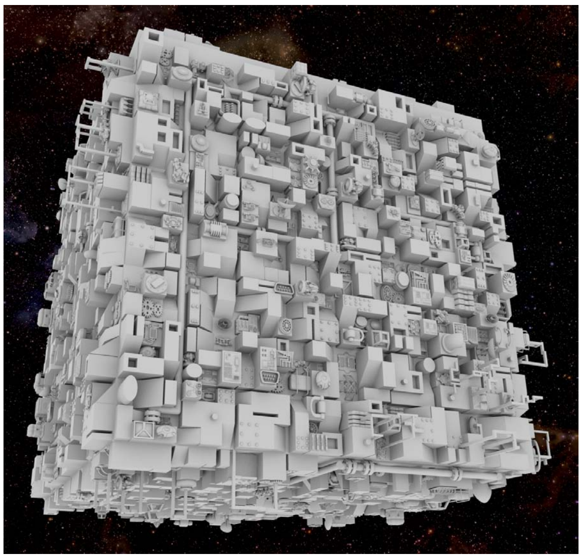


Figure 8: *A menacing cube-shaped spaceship. Resistance is futile.*

4. Results

Our implementation is written in C++. It uses OpenGL for 3D graphics and GLUI[§] for interactive widgets. Our application constructs the geometry as described above, allows the user to preview it, and outputs an OBJ file for final rendering. Figure 1 shows a sequence of snapshots of detail synthesis in action. Figure 7 shows a close-up of one node, making it easy to pick out the primary and secondary geometry, the

§ glui.sourceforge.net

primitives and kitbashed models, and the network of pipes. One final ship is shown in Figure 8.

5. Conclusions

This paper presents a first attempt to apply techniques from procedural modelling to a new domain, that of spaceships. Rather than attempting to solve the entire problem of procedural spaceship generation, we deliberately focus on the question of placing surface detail on large ships, a question to which we believe procedural techniques are well suited, and one that might be informed by existing aesthetic theories.

We believe that our results capture some of the technological look and visual appeal of science fiction ships such as those in the original Star Wars trilogy, and that our method strikes a worthwhile balance between regularity and chaos. While we feel we have satisfied our aesthetic goals for this project, we recognize that much more work would be necessary to expand this prototype into a practical, expressive system in a production environment.

A natural next step would be to generalize our system to work on a wider variety of frames. Any flat surface can of course be embedded in a rectangle and thereby given a set of grid cells and sub-panels, but it would be interesting to adapt the division process to operate on non-rectangular, or even non-flat surfaces. Beyond that, we would naturally like to investigate procedural generation of the frames themselves, based on the constraints and conventions of the established tradition of ship design.

The idea of studying design conventions in order to derive procedural modelling algorithms lends itself readily to other classes of object. We believe that mecha robots are another science fiction domain that could benefit greatly from procedural modelling. In a practical setting, we can imagine similar techniques being used in automotive design and industrial design.

We would also like to explore whether our system could be tailored to specific visual styles or genres. For example, the increasingly popular aesthetic of Steampunk, which celebrates Victorian-era technology, might be accessible as a special case of our system with the right choices of primitives and kitbashed models.

Acknowledgments

Thanks to Todd Boyce for many ideas and inspiration, to Bill Cowan, Dan Goldman and Stephen Mann for helpful discussions and feedback, to Paul Salvini of Side Effects Software for providing Houdini, and to Casper Del Blanco, Mark Kuykendall and Carl Fredsberg for their 3D models.

References

[Bir03] BIRKHOFF G. D.: *Aesthetic Measure*. Kessinger Publishing, 2003. 3

[Bum05] BUMGARDNER J.: Information theory and art. *Mung Being*, 3 (2005). 3

[Chi08] CHIANG D.: *Mechanika: Creating the Art of Science Fiction with Doug Chiang*. IMPACT books, New York, NY, 2008. 2

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 287–294. 4

[dBvKOS00] DE BERG M., VAN KREVELD M., OVERMARS M., SCHWARZKOPF O.: *Computational Geometry: Algorithms and Applications*, second ed. Springer-Verlag, 2000. 7

[EMP*02] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling*, third ed. Morgan Kauffmann, 2002. 2

[Gom75] GOMBRICH E.: *The Sense of Order: A Study in the Psychology of Decorative Art*, third ed. John Wiley & Sons, New York, 1975. 3

[Hut26] HUTCHESON F.: *An Inquiry into the Original of Our Ideas of Beauty and Virtue*. Liberty Fund Inc., 1726. 3

[Kin10] KINNEAR K.: *The aesthetics of science fiction spaceship design*. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, January 2010. Available online at http://hdl.handle.net/10012/4935. 2, 3

[Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), pp. 35–44. 2

[LWW08] LIPP M., WONKA P., WIMMER M.: Interactive visual editing of grammars for procedural architecture. *ACM Transacations on Graphics 21*, 3 (2008). 2

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. *ACM Transactions on Graphics 25*, 3 (7 2006), 614–623. 2

[Pet06] PETERSON L.: *Sculpting a Galaxy: Inside the Star Wars Model Shop*. Insight Editions, San Rafael, CA, 2006. 1, 3

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1990. 2

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 301–308. 2

[Sch09] SCHNEIDER B.: Ex Astris Scientia. http://www.ex-astris-scientia.org/articles/design.htm, 2009. 2

[SG72] STINY G., GIPS J.: Shape grammars and the generative specification of painting and sculpture. *In Proceedings of IFIP Congress71* (1972), 1460–1465. 5

[Teo09] TEOH S. T.: Generalized Descriptions for the Procedural Modeling of Ancient East Asian Buildings. In *Workshop on Computational Aesthetics* (Victoria, British Columbia, Canada, 2009), Deussen O., Hall P., (Eds.), Eurographics Association, pp. 17–24. 2

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. *ACM Transactions on Graphics 22*, 3 (7 2003), 669–677. 2, 5