

Figure 6: Collide-and-extrude feature to achieve intriguing interlocking more intuitively and rapidly.

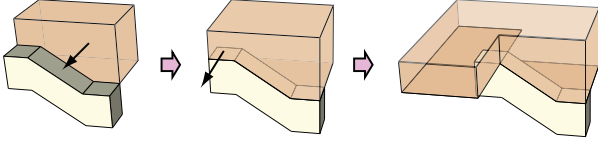


Figure 7: Collide-and-extrude feature used to create a unique double storey unit that utilizes an existing staircase's roof as its stair base to the second level.

conservative estimate derived from potential growths of other units in the proximity. With this, as in our example, the unit can better decide whether to venture into occupying the whole of the top of the staircase or to lock the colliding face.

## 6 Form Completion Engine

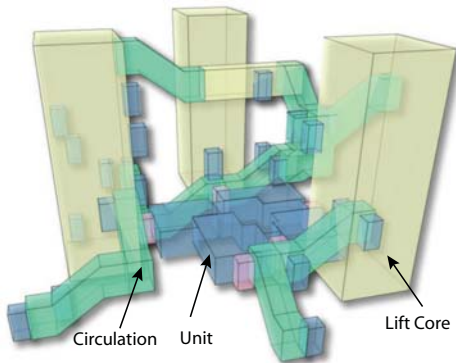


Figure 8: An illustration of five completed units with entrances colored in green.

The form completion engine provides an interactive tool that aids the user in synthesizing various completed forms corresponding to a selected number of partially grown units or seed units. It can be used as a stand-alone tool for rapid design synthesis, or can also serve to complement the growth process to better claim gaps (voids) in space for units; see Figure 8 for a simple illustration. Before presenting the working of the completion procedure in Section 6.3, we first discuss our design representation in Section 6.1 to support the completion procedure, and a form complexity measure in Section 6.2 to categorize designs.

### 6.1 Design Representation

For the purposes of intuitive manipulation and ease of conceptually visualizing designs mentally by the user, we adopt a representation based on 2D floorplans, which will eventually be synthesized as 3D volumes. This is entirely adequate as a form design work spans at most 2 to 3 levels at a time.

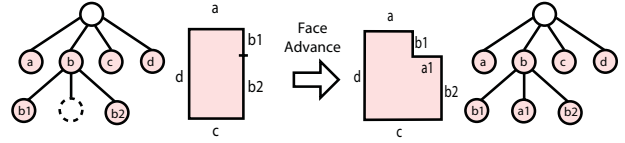


Figure 9: Modifications to the shapetree structure due to face split (left) to generate a virtual node (shown in dashed line), and face advance (right).

The *ShapeTree* is a tree-based representation that provides an implicit encoding of the shape of a given rectilinear unit in 2D. For any shape, its representation is a tree with a set of nodes  $V = (v_1, v_2, \dots, v_i)$  and a set of corresponding weights  $W = (w(v_1), w(v_2), \dots, w(v_i))$  to keep for example, the lengths of edges. This tree-based representation of unit shapes begins with a rooted tree with four child nodes each representing the four faces of the unit square with their respective orientations. The leaf nodes of the tree at any point in time correspond to edges of the current 2D geometric shape, while internal nodes implicitly record a history of unit shape variation, as well as providing an immediate decomposition into constituent boxes. Subtrees also represent a decomposition into sub-shapes. The face advance and face split operations in Section 5.1 translate straightforwardly to the 2D case to work on the shapetree (See Figure 9). To represent a particular configuration of floorplan with  $n$  units, we use a set of  $n$  shapetrees  $S = \{T_1, T_2, \dots, T_n\}$  with additional graph edges linking some node of  $T_i$  to another node of  $T_j$  for any incidence relation between the two faces in the floorplan. We have considered other planar graph representations for our floorplan designs, but found this simple representation by a set of shapetrees sufficient (and efficient) in manipulating topological designs in our process.

### 6.2 A Measure of Form Complexity

For purposes of generating a spectrum of shape configurations from simple to complex for a variety of applications, we introduce a form complexity measure that characterizes intriguing interlockings of two or more units. Our measure captures the proportion of the total number of faces that are interlock (incident) to other units. Specifically, for a set of shapes  $S = \{s_1, s_2, \dots, s_k\}$ , with  $|s_i|$  denoting the number of faces of  $s_i$ , the complexity measure  $F_q$  of  $S$  is determined as

$$F_q(S) = \frac{1}{\sum |s_i|} \sum_{i=1}^k \sum_{j=i+1}^k C_{ij}$$

where  $C_{ij}$  is the total number of incident faces of  $s_i$  and  $s_j$ . We have  $F_q(S)$  being trivially 0 for a set of shapes that do not interlock. In Figure 10 we show some examples of various sets of shapes and their corresponding form complexity measure. This formalism is inspired by the work of [Gero and Kazakov 2004] on qualitative symbolic modeling where the authors also classify configurations of shape without considering the actual dimensions of the shapes.

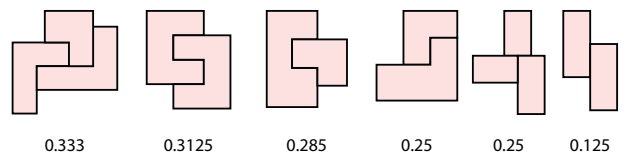


Figure 10: Various set of shapes and their form complexity measure.

### 6.3 The Completion Procedure

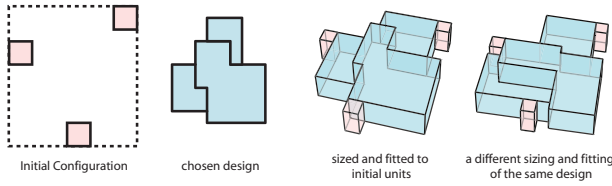


Figure 11: An illustration of a chosen design and its different completed forms. Pink squares represent entry points to units.

Refer to Figure 11. The input to the completion procedure is a set of  $n$  partially grown units or initial seeds, as selected by the user. There are three steps to complete a form. First, user selects a required form complexity to generate forms (Section 6.3.1). Second, a form is selected to link up with the initial seeds (Section 6.3.2). Third, the form is sized into useful units (Section 6.3.3). The resulting form of these steps can further be modified (such as adding some balcony features) by users depending on their needs.

#### 6.3.1 Topology Generation

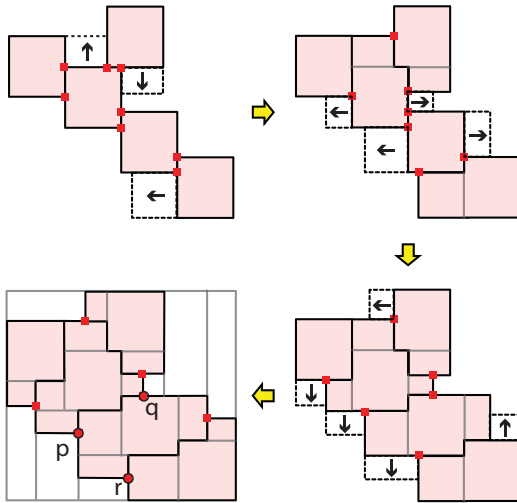


Figure 12: The starting configuration of 5 canonical units, and the steps in generating form with complexity in the range of 0.36 to 0.38. Red points represent possible split points to increase form complexity.

With  $n$  input seeds and an input form complexity, the system (by default) starts with  $n$  canonical units to start enumerating forms of the required form complexity; see Figure 12. To do this, the system first identifies split positions (shown as red points in Figure 12) that can increase form complexity, then select one or more among them to perform the actual split and face advance operations. The selection takes into account considerations such as equal distribution of unit shape complexity. With different initial arrangement of canonical units or different choices in the split and face advance operations, the system generates a pre-defined number of forms for selection to proceed onto the next step.

For the next two steps, the system needs to maintain the form chosen by the user. To this end, we define the *topology* of a form in

the following discussion. A form (as shown in Figure 12) is lying within a smallest bounding box. The form of each unit (from the starting  $n$  canonical units) consists of sub-units (that are boxes) that resulted from split and face advance operations as captured in the shapetree. The empty space outside of all sub-units within the bounding box is considered as the *sea region*. We can conceptually partition the sea region by sweeping a vertical plane from left to right to generate a unique set of (smallest number of) rectangular boxes (as shown in Figure 12(bottom left)). This decomposition of the sea region can be realized easily with the set of shapetrees by tracing along the boundary of the form. Analogously, we can do a unique partitioning of the sea region with horizontal plane sweeping from top to bottom.

Let  $E$  be the set of all the edges of the sub-units and rectangular boxes of the sea region, resulted from both the vertical and horizontal sweep planes. Edges in  $E$  are used to construct paths from some corner (vertex) of a sub-unit to another corner (vertex). In the following, for simplicity, we ignore the case where corners  $p$  and  $q$  are endpoints of an edge or are connected by a chain of vertical edges or a chain of horizontal edges - for this special case, we just need to modify the consideration accordingly. Two corners  $p$  and  $q$  of possibly different sub-units are such that  $p \prec_x q$  if, and only if, there exists a path in  $E$  from  $p$  to  $q$  with no horizontal edge oriented towards the negative  $x$  axis. Similarly,  $q \prec_x p$  if, and only if, there exists a path in  $E$  from  $p$  to  $q$  with no horizontal edge oriented towards positive  $x$  axis. If both of the above do not exist, then  $p$  and  $q$  are not comparable under  $\prec_x$ . Analogously, we can define the relationship  $\prec_z$  in the  $z$  direction between two corners. See the example in lower left of Figure 12 where we have  $p \prec_x q$ , and  $r$  is not comparable to  $q$  under  $\prec_x$ . To maintain the topology of a form in the next two steps, we maintain the relationships  $\prec_x$  and  $\prec_z$  of all the corners.

#### 6.3.2 Incidence Assignment and Realization

This step is to take a chosen form to assign correspondences of some of its edges (or corners) on the boundary to some edges (or corners) in the initial configuration. It stretches the form to fit into the initial configuration obeying the mentioned correspondences of edges or corners. The correspondences can be interactively assigned or autonomously performed by the system. Naïve methods of automatically assignment may not result in a physically realizable form. On the other hand, it can also be the case that multiple choices of realizable correspondences can exist; see Figure 11.

For a boundary path starting at corner  $p$  and ending at corner  $q$  with  $p \prec_x q$  and  $p \prec_z q$ , a realizable correspondence of  $p$  to  $p' = (p'_x, p'_z)$  and  $q$  to  $q' = (q'_x, q'_z)$  is one such that  $p'_x < q'_x$  and  $p'_z < q'_z$ . Analogously, we can define realizable correspondences for all the other configuration of  $\prec_x$  and  $\prec_z$ . When  $p$  and  $q$  are not comparable under  $\prec_x$  ( $\prec_z$ , respectively), then there is no constraint on the relationship of  $p'_x$  and  $q'_x$  ( $p'_z$  and  $q'_z$ , respectively). With this, it is easy to validate manual correspondences specified by the user, or to generate valid correspondences automatically.

Following realizable correspondences, the system computes a physical realization iteratively by moving one by one edges of sub-units to their constrained locations. The edge moving process uses two operations to preserve the topology of the form. In *cascading edge move* (Figure 13), the movement of an edge can result in movement of other edges so as to maintain non-zero area for each sub-unit and rectangle in the sea region. In *staircase edge move* (Figure 14), the movement of an edge can bring along a movement of another parallel edge (separated by just one orthogonal edge) when the separation between the edges violates some minimum separation requirement. These edge moves can propagate to other affected sub-units.

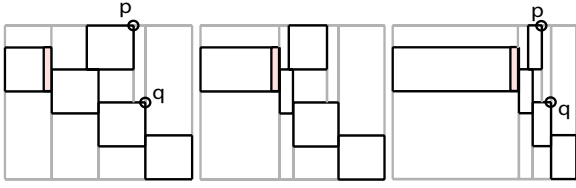


Figure 13: An illustration of effect of an cascading edge move on the bold edge while maintaining the relative positioning of  $p$  and  $q$ .

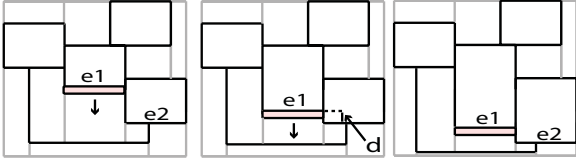


Figure 14: An illustration of effect of a staircase edge move on  $e1$  that induces a move of  $e2$  and other part of the design.

### 6.3.3 Geometry Sizing

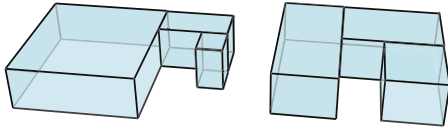


Figure 15: Comparison of good and bad unit sizings.

This step attempts to size each sub-unit to one that is sufficiently large enough to be a room of some architectural use. With the mentioned edge moving operations, the user can manually size each sub-unit. As for automatic sizing, we aim for equally-size sub-units (see Figure 15). The algorithmic problem is as follows. For an input set of rectangles  $R = \{r_1, r_2, \dots, r_k\}$  with initial areas  $A_0, A_1, \dots, A_k$ , we want to change the areas of each  $r_i$  into a targeted value  $A_i'$ . For our problem, we assign to those rectangles belonging to sub-units a large target value  $M$  (which can be the total area of the bounding box of the design over the number of sub-units in the design) whereas those to the sea region a small positive value  $m$ . Then, an iterative process, adapted from [Kreveld and Speckmann 2004], with some bound on the number of iterations, is performed to size each sub-unit while maintaining the topology of the form. The process gives priority to size sub-unit with the largest difference to its target size.

## 7 Experimental Results

Our system is implemented with the CGAL library on a Pentium IV 3.0GHz, 1GB DDR2 RAM and nVidia GeForce 6600 GT with 128M DDR3 video memory. Alpha and Beta (Figure 1(left) and Figure 16) are two large scale models, generated using our system with their input cores as shown too. These models are generated in less than 5 minutes in an interactive session. See the accompanying video for an animation of the generation process of the Beta model. A partial example of the form completion method is shown as Figure 8. See the accompanying video for a more comprehensive illustration of the process.

Table 1 summarizes the statistics on the use of space by the

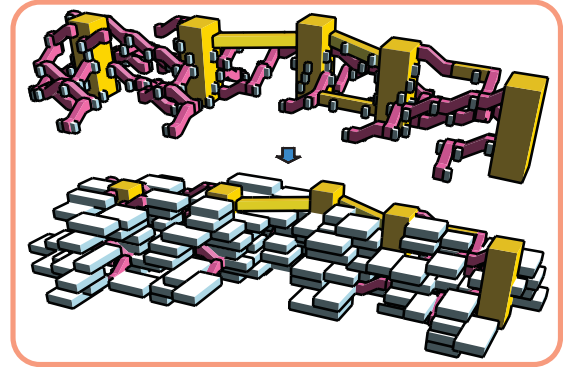


Figure 16: Initial and final generated form of our Beta model.

Model	Site Area	No. of Units	Total No. of Occupants	Occupancy Ratio
Alpha	90m x 50 m	71	213	0.0095
Beta	100m x 60m	134	364	0.0101
		Type A Units	Type B Units	Type C Units
Alpha		28	15	28
Beta		42	31	61
		Multiple View Units		
		Average View	Two Views	Three Views
Alpha		28m <sup>2</sup>	17	29
Beta		25m <sup>2</sup>	34	54
		Average Floor Area	Units with Double Volume	Units with Outdoor Space
Alpha		62.2m <sup>2</sup>	25	27
Beta		60.5m <sup>2</sup>	47	57

Table 1: Table of statistics on space use for generated models.

Alpha and Beta models. We note that Swedish housing standards [Swedish Regulation] specify that the minimum comfortable dwelling area is at 30 to 50m<sup>2</sup> for one person (Type C), at 50 to 85m<sup>2</sup> for three (Type B), and at greater than 85m<sup>2</sup> for a family of five (Type A). With these, we can calculate the number of total occupants for Alpha and Beta. To appreciate the occupancy ratio, we note a typical cookie-cutter building of about 32m x 27m has 4 units per storey. Assuming each unit houses 4 persons, we then have 16 persons in one storey, which gives an occupancy ratio of 0.0185 person per square meter. Our (5 storey) Alpha and (6 storey) Beta forms can achieve about half the efficiency in usage of space compared to a typical conventional cookie-cutter building.

Apparently, the efficiency in space usage in the conventional way is traded to create outdoor space of good attribute values in Alpha and Beta; see for example the insert to Figure 1 on the possible usage of suspended open spaces. Table 1 also summaries the statistics on the view and shape variation of Alpha and Beta. Indeed, we observe that about half the number of units have good suspended open spaces. In addition, the average view of all the units are considered high as it is close to half the size of the average floor area. As for shape variation, we also observe good percentage of units having attractive double volume and multiple views. Due to the non-regular units, there are many ventilation corridors in both Alpha and Beta from our visual inspection.

On the whole, we demonstrate the possibility of creating forms that look beyond the usual occupancy ratio but a premium type of housing units with good views and variation of shapes to provide for quality living in modern housing.