Holly: A Drawing Editor for Designing Stencils

Yuki Igarashi
University of Tsukuba

Takeo Igarashi
University of Tokyo

Stenciling is a form of artistic expression in which you print images on target objects (for example, fabric or a postcard) by applying pigment over a plate with holes. A stencil contains two types of regions. A negative region is an empty space through which paint can pass; a positive region is the surface surrounding the negative region.

Creating stencils can be difficult because they must satisfy specific physical constraints. In a valid stencil, the template is a single, connected piece of material (see Figure 1). Sometimes islands—isolated unconnected positive regions—are inadvertently created and must be connected by bridges to the main part of the stencil. So, stencil design normally requires knowledge, experience, and skill. Consequently, most people simply buy ready-made stencil plates rather than make their own.

Our Holly system lets you easily design valid stencils from scratch (see Figure 2). You interactively draw freeform strokes on the digital canvas, and Holly automatically generates the stencil. You can print out the final image using a cutting plotter—for example, a Craft ROBO (www.graphteccorp.com/craftrobo). Holly is so simple and straightforward that children can use it. See our accompanying video "Holly," at www.computer.org/portal/web/computingnow/cga/videos. (For a look at some of the research that inspired us, see the sidebar.)

User Interface

You design a stencil using a combination of strokes, primarily employing the brush tool (see Figure 3a) and fill tool (see Figure 3b). Each time

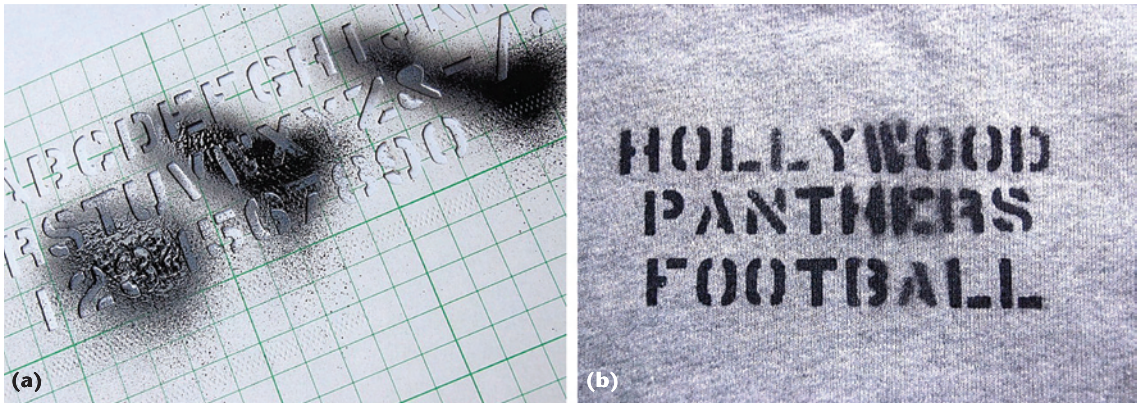

Figure 1. Stencils. (a) In a valid stencil such as this, the template is a single connected piece of material. All islands are connected by a bridge to the main part of the stencil. (b) Using the depicted stencil produces this result. (Source: www.mintclothes.com; used with permission.)

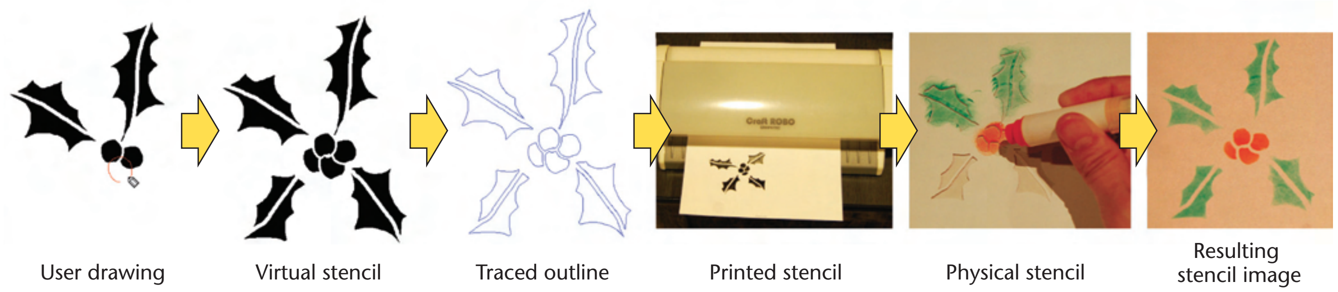

Figure 2. The Holly stencil-drawing system. First, you draw an image using freeform strokes. Holly then automatically generates a virtual stencil. Next, you create a physical stencil using a cutting plotter. Using that stencil, you add pigment to a surface.

Related Work in Interactive Drawing and Stencils

Our research builds on previous research in interactive drawing editors.

Raster Graphics

Commercial systems include Adobe Photoshop and Microsoft PaintBrush. Yingge Qu and her colleagues proposed a technique that propagates color over regions exhibiting both pattern and intensity continuity.[1] Their method effectively colorizes black-and-white manga, which contains intensive amounts of strokes, hatching, half-toning, and screening. James McCann and Nancy Pollard presented an image-editing program that lets artists paint in the gradient domain with real-time feedback on megapixel-sized images.[2]

Vector Graphics

Commercial systems include Adobe Flash and Adobe Illustrator. TicTacToon is a system for professional 2D animation studios that replaces the traditional paper-based production process.[3] This animation system is one of the first to use vector-based sketching and painting. Michel Gangnet and his colleagues solved the gap detection and closing problem in TicTacToon's cel-coloring application.[4] Paul Asente and his colleagues introduced a metaphor that allows planar-map-based coloring without changing the original paths.[5] This results in greater editing flexibility and ease of use.

Creating Stencils

Jonathan Bronson and his colleagues presented a method for generating expressive stencils from 3D polygonal meshes or images.[6] In their system, the user provides the input geometry and can adjust the view, lighting conditions, line thickness, and bridge preferences to achieve the desired stencil. (Bridges connect islands—otherwise disconnected positive regions—to the rest of the stencil.) The stencil-creation algorithm employs multiple metrics to measure the appropriateness of connections between unstable stencil regions. These metrics describe local features to help minimize the distortion of the abstracted image caused by stabilizing bridges. Bronson and his colleagues aim to create a stencil from a given 3D model or image. In contrast, we aim to design a stencil from scratch using drawing operations (see the main article).

References

1. Y. Qu, T.-T. Wong, and P.-A. Heng, "Manga Colorization," *ACM Trans. Graphics,* vol. 25, no. 3, 2006, pp. 1214–1220.
2. J. McCann and N.S. Pollard, "Real-Time Gradient-Domain Painting," *ACM Trans. Graphics*, vol. 27, no. 3, article 93; http://graphics.cs.cmu.edu/projects/gradient-paint/grad.light.r2226.pdf.
3. J.-D. Fekete et al., "TicTacToon: A Paperless System for Professional 2D Animation," *Proc. Siggraph,* ACM Press, 1995, pp. 79–90.
4. M. Gangnet, J.V. Thong, and J.-D. Fekete, "Automatic Gap Closing for Freehand Drawing," 1994; http://insitu.lri.fr/~fekete/ps/siggraph94.pdf.
5. P. Asente, M. Schuster, and T. Pettit, "Dynamic Planar Map Illustration," *ACM Trans. Graphics,* vol. 26, no. 3, 2007, article 30.
6. J. Bronson, P. Rheingans, and M. Olano, "Semiautomatic Stencil Creation through Error Minimization," *Proc. 6th Symp. Nonphotorealistic Animation and Rendering*, ACM Press, 2008; www.cs.umbc.edu/~jonbron1/stencil.pdf.

you complete a stroke, the system automatically generates a stencil image in which all positive regions are connected.

Holly provides two design modes: underwriting (the default) and overwriting. In underwriting mode, the system renders the last drawn stroke beneath the existing strokes. In overwriting mode, it renders the last drawn stroke on top of the existing strokes. You can change the mode while designing the stencil.

You can erase a portion of the existing strokes using the brush-eraser tool (see Figure 4a) or fill-eraser tool (see Figure 4b).

Internally, Holly stores every drawing and eraser stroke as a primitive, which you can easily modify. You can move a stroke using a standard dragging operation. Right-clicking on a primitive opens a pop-up menu that lets you apply various commands to the primitive—for example, to delete it or change its position (see Figure 5a). You can select a set of primitives by rubber-banding them, or you can duplicate them by using a rubber-stamp tool (see Figure 5b).

Each stroke is oriented in a clockwise direction, and Holly appropriately renders self-intersecting strokes (see Figure 5c). You can reverse the orientation using a pop-up menu. The default setting automatically adds a margin around the stroke, but you can connect strokes without margins (see Figure 5d). This operation combines the underlying stroke and the newly drawn stroke as a group.

Holly automatically detects any islands and adds a bridge connecting them to the main sheet. For example, suppose you add a fill primitive in underwriting mode that creates an island (see Figure 6a). Holly then adds a bridge (see Figure 6b). When you move a stroke, Holly automatically updates the

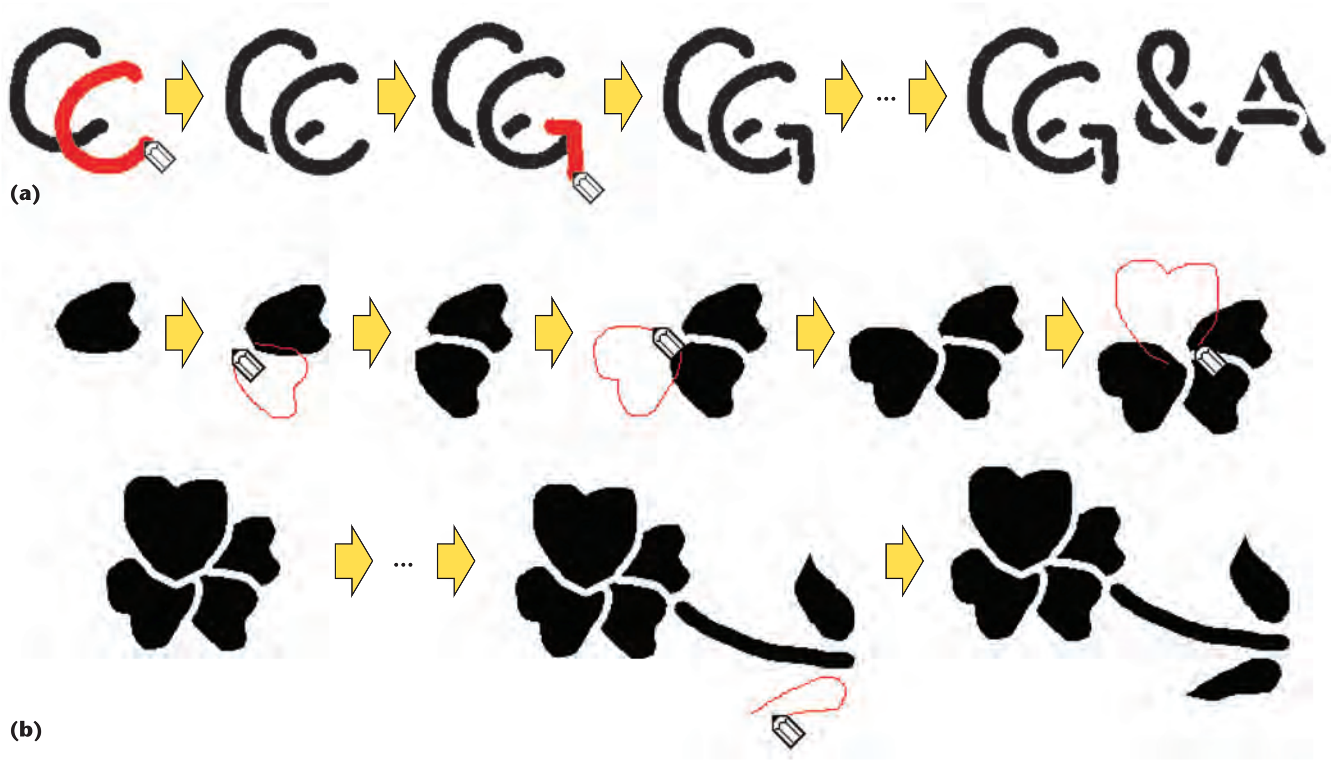
Figure 3. You design a stencil using a combination of brush and fill strokes. (a) When you use the brush tool, Holly generates a negative region along your input strokes. (b) When you use the fill tool, the system connects the first and last stroke points and generates a negative region in it.

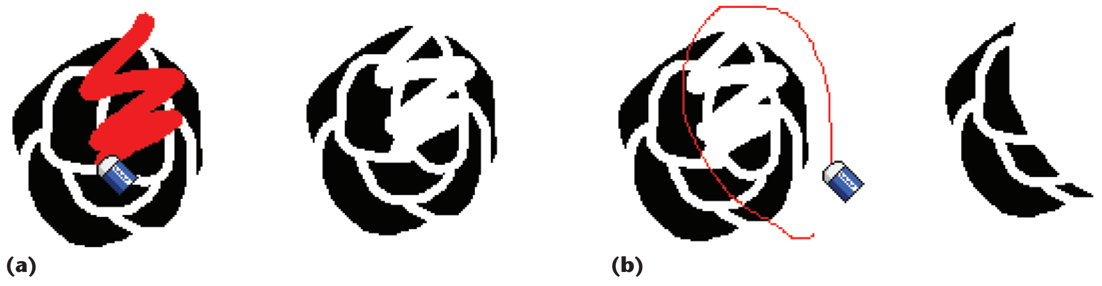
Figure 4. You can erase strokes using the (a) brush-eraser tool and (b) fill-eraser tool. When you use the brush-eraser tool, Holly generates a positive region along the user input stroke. When you use the fill-eraser tool, Holly connects the first and last points of the stroke and generates a positive region within.

bridge's stroke position (see Figure 6c) or deletes the bridge if it's no longer necessary (see Figure 6d). If you dislike the automatically created bridge, you can manually add one by using one of the eraser tools (see Figures 6e and 6f). You can change the bridge's width to account for the stencil material's rigidity.

You can turn off automatic bridge creation. In this mode, Holly warns you by highlighting the isolated region with a green boundary (see Figure 6g). The system removes the isolated region when you export the design as a vector file when you're done designing the stencil. Figure 6h shows the resulting outline from this mode.

Holly exports the final stencil image as an SVG (Scalable Vector Graphics) file or a DXF (Drawing Exchange File). As we mentioned before, you can obtain the physical stencil using a cutting plotter. The total cutting time depends on the design, and it's helpful to know during the design phase how long it will take to cut the stencil. So, Holly estimates the cutting time and presents this information to you while you're designing the stencil. (We discuss this in more detail in the next section.)

Implementation

We implemented our prototype system as a Java program that internally combines vector and raster representations. Holly represents a brush stroke as a polyline and a fill stroke as a polygon. It represents normal strokes as negative primitives and eraser strokes as positive primitives, all stored

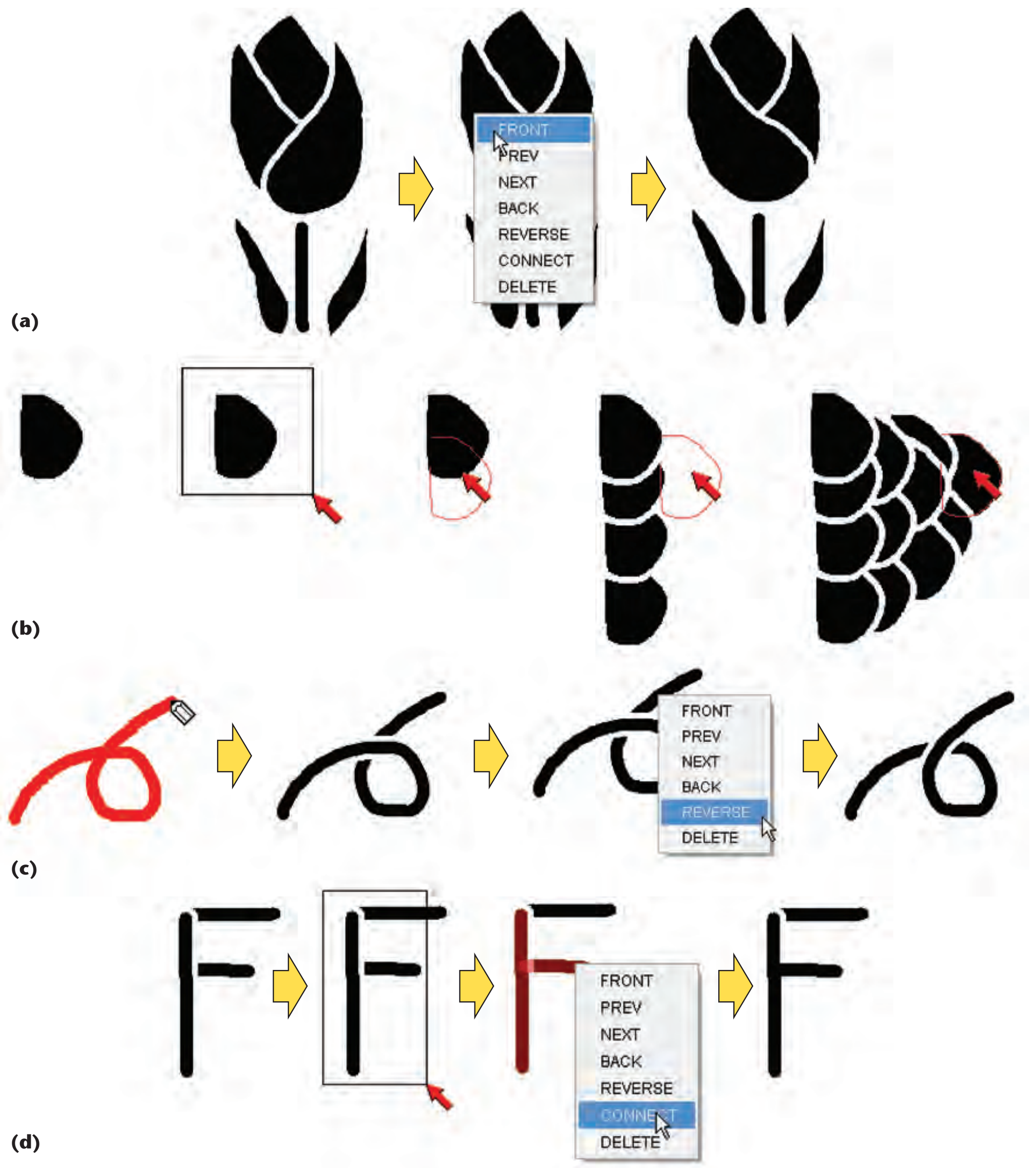

Figure 5. Various tools let you (a) change the order of strokes, (b) copy and paste them, (c) create self-intersecting strokes, and (d) connect stroke lines without a margin.

in an ordered list. The system represents the stencil image as a Boolean raster image (see Figure 7), with true (positive) and false (negative) regions. Holly renders the strokes in the stroke list on the image, one by one.

Figure 8 shows Holly's data structure. The system holds the primitive list (strokes and groups) and the list of automatically generated bridges (strokes). When you change the primitives' order or remove or move a primitive, Holly re-renders all the primitives. A primitive contains an array of points and types (brush, fill, brush-eraser, or fill-eraser). The system also re-renders all the primitives when it loads them. However, in overwrite mode, Holly only updates the last drawn primitive's bounding box.

At the start of the design process, the stencil image is entirely positive. When you use the normal brush tool to draw a stroke over an existing stroke, Holly samples it before drawing a positive stroke with an extra margin on the stencil raster image, based on your input stroke. It then draws a negative stroke with the original brush radius (see Figure 9). This generates a margin around the strokes. When you use the fill tool, the system connects the stroke's first and last points and generates a negative region in it. Holly then draws a wide positive region with a margin and draws in the negative region.

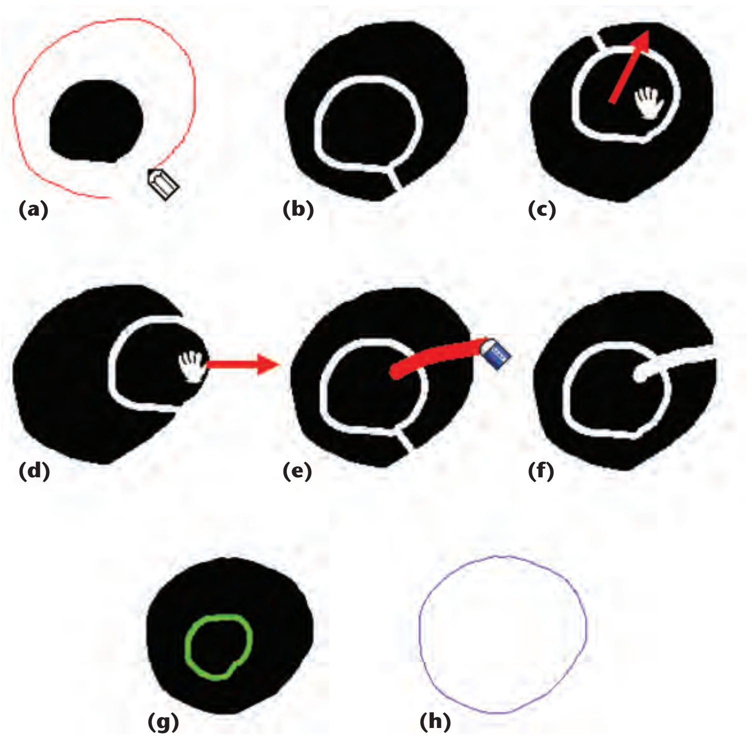

Figure 6. Bridge creation. The user (a) adds a fill primitive in underwriting mode, thus creating an island (an isolated positive region). In automatic mode, Holly (b) detects the island and connects it to the main sheet. If the user moves a stroke, the system either (c) updates or (d) deletes the bridge. Users can also (e) use one of the eraser tools to (f) manually add a bridge. In warning mode, Holly (g) highlights the isolated region with a green boundary. When you proceed to printing, Holly (h) automatically removes the isolated region from the final stencil image.

When you draw a new stroke on the canvas, the system adds the stroke to the ordered list. In underwriting mode, Holly adds the stroke to the beginning of the list and re-renders the whole stencil image from scratch. In overwriting mode, the system adds the stroke to the end of the list. In this case, Holly simply renders the new stroke on the current stencil image.

A self-intersecting stroke normally causes an isolated region. To address this problem, we use a more detailed algorithm. When you choose the normal drawing mode, Holly interactively redraws the positive wide strokes and negative strokes for each mouse-dragging event. Internally, the system first draws a positive wide stroke from p_{n-1} to p_n. It then draws a negative normal-width stroke from p_{n-2} to p_n. In this way, you can draw a self-intersecting stroke (see Figure 5c). When you reverse an overlapping area, Holly reverses the points in the target stroke's list and re-renders the stroke.

For the brush-eraser tool, Holly renders a positive stroke with a normal brush radius on the basis of your input stroke. For the fill-eraser tool, the system connects the input stroke's first and last points and then generates a positive region in it.

When you connect strokes, Holly creates a group primitive using the selected strokes. In the primitive rendering, the system first draws the positive strokes with an extra margin on the stencil raster image. It then draws the negative strokes with the original brush radius.

To deal with islands, Holly first repeatedly applies a flood-fill algorithm to the stencil image to decompose the positive pixels into connected regions. It then computes the distances between each pair of connected regions. (The distance between two regions is the minimum distance between the pixels in separate regions.) This generates a complete graph whose vertices are regions; the cost of an edge is the computed distance. The system then computes the graph's minimum spanning tree and places bridges at the tree's edges. The bridges are represented as ephemeral primitives and are deleted and recomputed every time you modify the image.

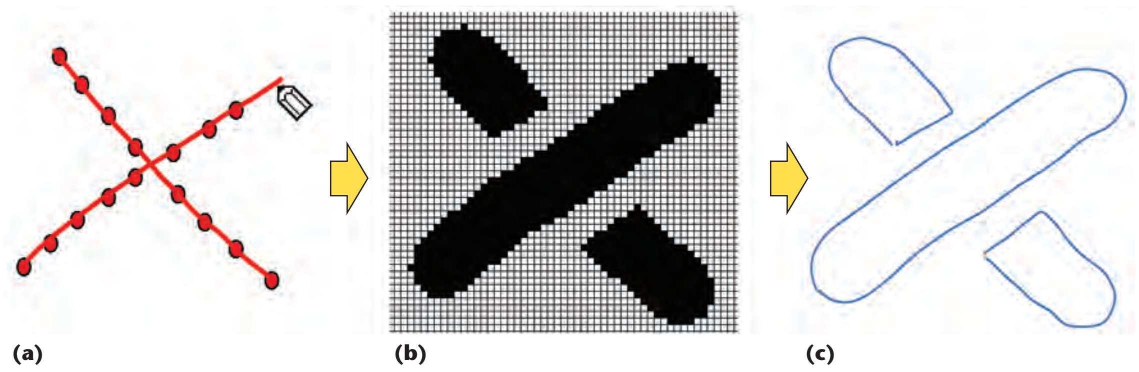

Figure 7. Holly represents the stencil image as a Boolean raster image. (a) Holly represents the user-input stroke as polylines. (b) It then converts it to a Boolean raster image. (c) Finally, Holly exports the resulting outline in a vector format.

To produce the output, Holly first removes any isolated regions that appeared in the warning mode and applies the marching-squares algorithm.[1] The system traces the resulting edges, applies smoothing, and exports a vector file for use with a cutting plotter. The total cutting time increases proportionally to the total stroke length, which is the sum of the circumference of each traced edge. The total time (T_{total}) to cut a real stencil plate is

$$T_{\text{total}} = W_{\text{cut}} L_{\text{stroke}},$$

where L_{stroke} is the stroke line's length and W_{cut} is the weight per unit L_{stroke}. Holly presents you with both the manual cutting time and the automatic cutting time using the cutting plotter. For manual cutting, we use $W_{\text{cut}} = 1.1$ s/cm, on the basis of Jun Mitani's research.[2] For automatic cutting, we use $W_{\text{cut}} = 0.5$ s/cm, on the basis of information from Craft ROBO's manufacturers.

Results

Holly runs in real time on a 1.1-GHz Pentium-M PC. Using Holly, we designed and made several stencils and used them to decorate paper and fabric (see Figure 10).

We also ran a small workshop to have novice users try Holly. Ten children, ages 8 to 11, joined the workshop, accompanied by their parents. We gave a brief tutorial at the beginning. The children took about an hour to create their designs (see Figure 11a). They then printed the designed stencils using a cutting plotter and used them to decorate fabric bags, which also took them roughly one hour (see Figure 11b).

Figure 11c shows some stencil designs and finished bags that the children created. They quickly learned how to use Holly. Furthermore, they enjoyed the process. They also gave us valuable feedback for future improvement. For example, they wanted to use stencil templates, and they wanted an easy-to-use input method to name files because they used a keyboard-free tablet PC.

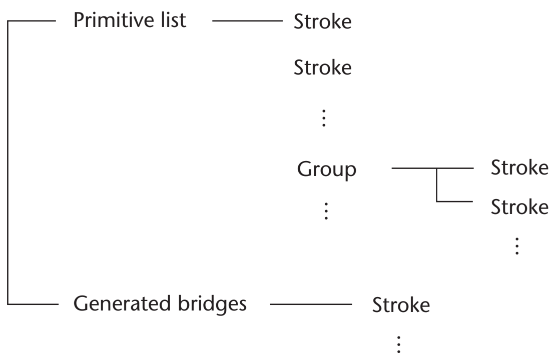

Figure 8. Holly's data structure. The system holds the primitive list (strokes and groups) and the list of automatically generated bridges (strokes).

The current implementation doesn't consider the physical-stencil material's rigidity. If the stencil is hard (for example, wood), the bridges can be thin. If the stencil is soft (for example, paper), bridges should be wider. In addition, bridges that support big islands should be wider. We plan to program the system so that it automatically adjusts the width of the bridge according to the source material's rigidity.

Also, Holly currently only shows you a black-and-white image. We plan to support multiple colors in the future and to run another workshop to have more test users try our system.

This research demonstrates the effectiveness of designing physical objects by considering the material's physical properties in the construction process. We'd like to apply this approach to aid in designing other physical objects. For example, we plan to analyze liquid flow to help design a better teapot or predict possible ways that leather wears to help design a sturdy briefcase.

Acknowledgments

We thank Jun Mitani for providing source code for exporting Drawing Exchange File. We also thank the National Museum of Emerging Science and Innovation in Japan for providing the workshop

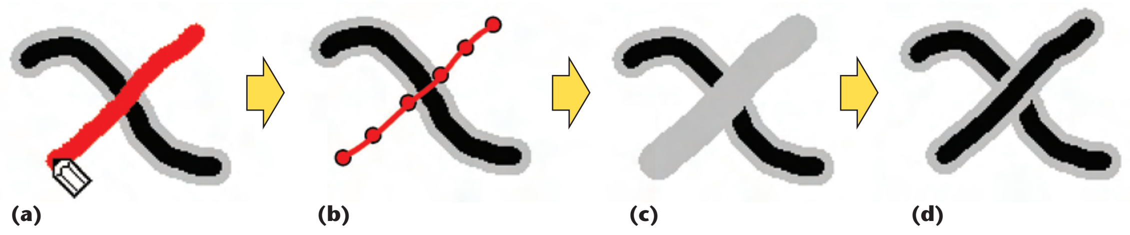

Figure 9. The algorithm for drawing a stroke on a raster image. (a) When you draw a stroke over an existing stroke, (b) the stroke is sampled, then (c) Holly draws a positive stroke with an extra margin on the stencil raster image, and finally (d) Holly draws a negative stroke with the original brush radius to create the final image.

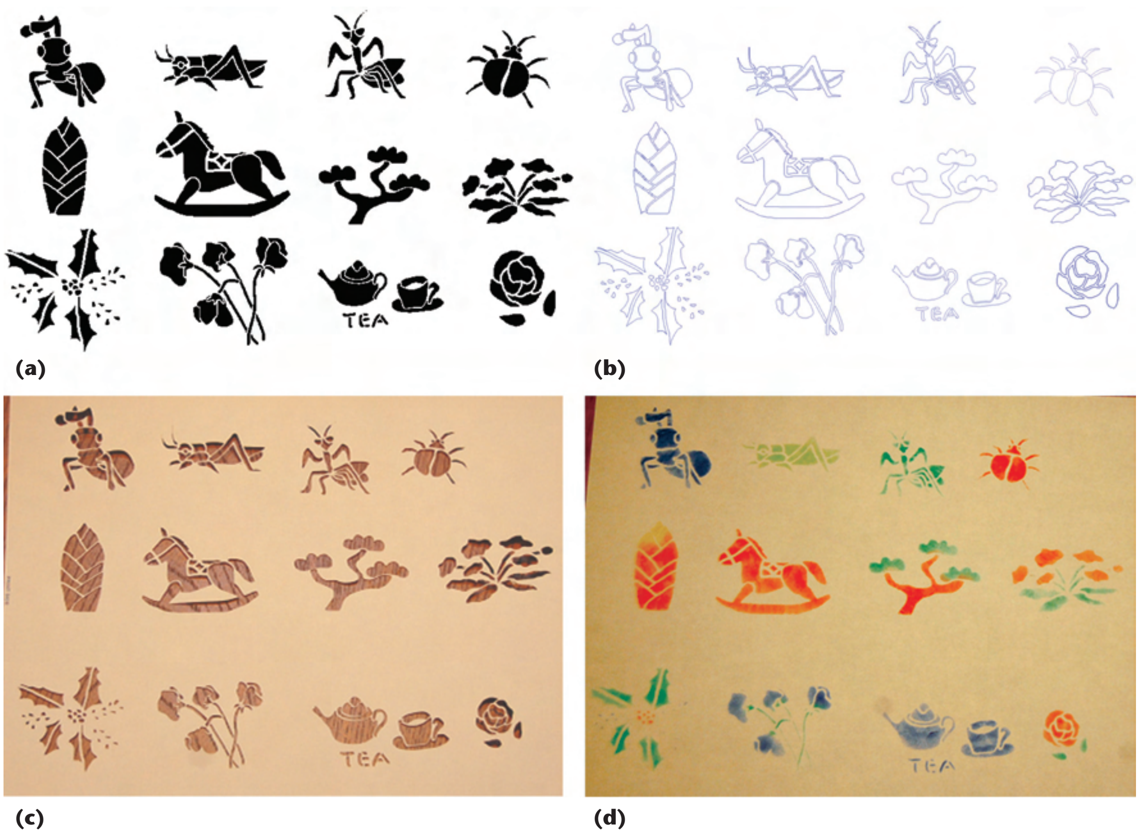

Figure 10. Results using Holly: (a) the result of Holly's painting operations, (b) the resulting outline data set using the cutting plotter, (c) the physical stencil plate created using the cutting plotter, and (d) the decorated paper using the stencil.

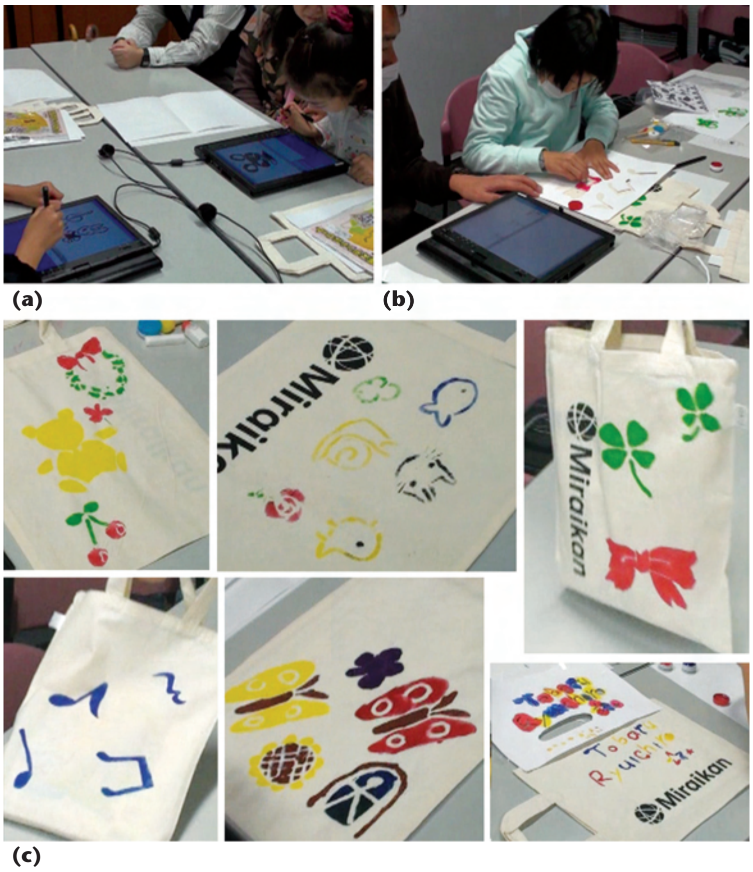

Figure 11. Photos from a workshop in which 10 children used Holly: (a) designing stencils, (b) decorating fabric bags, and (c) some of the stencils and bags. The entire process took approximately two hours.

location. This work was partly supported by grant-in-aid from the Japan Society for the Promotion of Science Fellows.

References

1. W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *ACM Siggraph Computer Graphics*, vol. 21, no. 4, 1987, pp. 163–169.
2. J. Mitani, "Research for a Design Support System for Three-Dimensional Paper Models Using Computer," doctoral dissertation, Dept. of Eng., Univ. of Tokyo, 2004 (in Japanese).

***Yuki Igarashi** is a Research Fellow of the Japan Society for the Promotion of Science at the University of Tsukuba. Contact her at yukim@acm.org; www.geocities.jp/igarashi_lab.*

***Takeo Igarashi** is an associate professor of computer science at the University of Tokyo and the research director of the User Interfaces for Design Project, part of the Japan Science and Technology Agency's Exploratory Research for Advanced Technology (Erato) program. Contact him at takeo@acm.org; www-ui.is.s.u-tokyo.ac.jp/~takeo.*

Contact department editor Mike Potel at potel@wildcrest.com.