Paper3D: Bringing Casual 3D Modeling to a Multi-Touch Interface

Patrick Paczkowski* Julie Dorsey* Holly Rushmeier*
Yale University
Min H. Kim†
KAIST

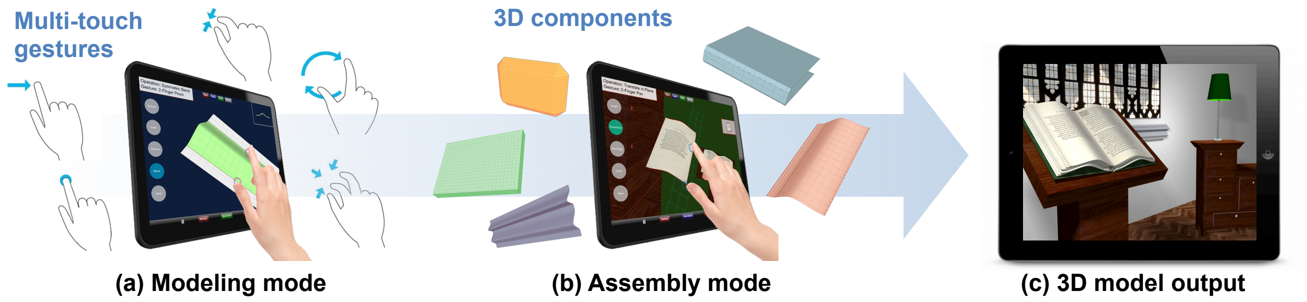

Figure 1. An overview of our novel 3D modeling technique, designed for a multi-touch interface to create 3D models on a tablet device. (a) Users fold 2D sheets of paper in the modeling mode, using gesture-based modeling tools such as folding, bending, extending, and cutting. (b) A set of modeled 3D components is assembled together through pinning and taping, resulting in (c) a complex 3D scene.

ABSTRACT

A 3D modeling system that provides all-inclusive functionality is generally too demanding for a casual 3D modeler to learn. In recent years, there has been a shift towards developing more approachable systems, with easy-to-learn, intuitive interfaces. However, most modeling systems still employ mouse and keyboard interfaces, despite the ubiquity of tablet devices, and the benefits of multi-touch interfaces applied to 3D modeling. In this paper, we introduce an alternative 3D modeling paradigm for creating developable surfaces, inspired by traditional papercrafting, and implemented as a system designed from the start for a multi-touch tablet. We demonstrate the process of assembling complex 3D scenes from a collection of simpler models, in turn shaped through operations applied to sheets of virtual paper. The modeling and assembling operations mimic familiar, real-world operations performed on paper, allowing users to quickly learn our system with very little guidance. We outline key design decisions made throughout the development process, based on feedback obtained through collaboration with target users. Finally, we include a range of models created in our system.

Author Keywords
3D Modeling; Multi-Touch Interface; Papercraft; Folding

ACM Classification Keywords
I.3.5 Computational Geometry and Object Modeling

INTRODUCTION

Creating shapes is an essential operation in 3D computer graphics. Packages such as Blender or Maya provide many modeling techniques, e.g., polygonal modeling, modeling with NURBS or subdivision surfaces. However, learning how to use the majority of existing 3D modeling packages is challenging, hindering widespread use. An additional difficulty commonly experienced by users is the communication of 3D modeling coordinates through the standard user interface. Typical input devices (such as mice) and displays provide 2D input and output. These devices are inadequate for specifying 6 degree-of-freedom (DoF) position and orientation. Conversely, haptic devices with a full 6-DoF (e.g., Geomagic Touch) are inaccurate and difficult for novice users to operate due to a lack of haptic feedback on the tip of the pointer.

Recently, motion sensing interfaces with higher DoF have been commercialized, such as LeapMotion and Microsoft's Kinect, but these still suffer from lack of accuracy, underconstrained input and lack of haptic feedback. By contrast, input devices with multi-touch interfaces have evolved in recent years, with products like the Apple iPad and the Samsung Galaxy Tab. Though each input touch is still two-dimensional, the combination of two or more touches has resulted in gestures with more information than the input of a traditional, single-point device. Due to their ubiquity and intuitiveness, these devices are particularly powerful for casual users.

In the field of 3D modeling, research problems such as 3D transformations on a multi-touch device have been recently studied, and there are domain-specific modeling systems for a multi-touch interface. However, systems specifically focused on casual 3D modeling by users with little modeling experience (e.g., sketch-based interfaces) are still fairly limited either in scope, learnability or modeling representation. This paper

* e-mail: {patrick.paczkowski;julie.dorsey;holly.rushmeier}@yale.edu
† e-mail (corresponding author): minhkim@kaist.ac.kr

presents a novel 3D modeling paradigm tailored for such use, and fully leverages the benefits of a multi-touch interface.

Inspired by traditional papercraft art forms such as origami, we devised a modeling technique in which deformations mimic physically-based operations commonly performed on sheets of paper, such as folding, bending, cutting, pinning and taping. We used the simple, yet powerful principles of these ancient art techniques as a foundation, and extended it into a broader, practical technique for modeling developable surfaces. We designed and integrated this technique from the ground up for a multi-touch device. We evolved the system design in collaboration with both casual and experienced users. Figure 2 demonstrates a range of 3D models created using our system. Our contributions are as follows:

- A new modeling paradigm inspired by papercrafting, and extended by operations related to other physical actions;
- An implementation of Paper3D, based on this new paradigm, designed from the ground up to use multi-touch gestures;
- A discussion of the user-guided design decisions made throughout the evolution of our system;
- A validation of the effectiveness and usability of our system through extensive user feedback;
- A demonstration of a range of models and scenes designed by users of our system.

PREVIOUS WORK

This section surveys previous work in the fields of computing interfaces, 3D modeling and papercraft simulators.

Multi-Touch Interfaces

It is difficult to specify 3D coordinates using a 2D graphical interface (e.g., trackball, stylus, mouse), a limiting factor for intuitive graphical and modeling software. In recent years, indirect 3D graphical interfaces have gained popularity (e.g., Microsoft Kinect, LeapMotion, Geomagic Touch, etc.). Though these devices enable 6-DoF tracking, the lack of haptic feedback still limits their intuitiveness [9].

From their inception, it was clear that multi-touch interfaces, pioneered by Lee et al. [11], offer significant advantages with their wide range of natural inputs, such as pinch gestures. Multi-touch devices (Apple iPad, Microsoft Surface, etc.) are now widespread. In this work, we used a multi-touch graphical interface for a 3D modeling system. The *intuitiveness* of such devices does not fully eliminate the ambiguity of mapping 2D screen inputs into a 3D coordinate space, but we have found that users, particularly casual modelers, find these devices more natural.

3D Modeling Systems

This section surveys available 3D modeling alternatives.

Desktop Modeling Systems.

Commercial desktop systems such as AutoCAD, Maya or SolidWorks provide extensive modeling capabilities suited for various industries. These tools are geared towards professionals who can go through extensive periods of training. More accessible, alternative modeling systems have gained in popularity, such as SketchUp (an easy to use system for architects) and ZBrush (a digital sculpting tool, relatable to traditional sculpting). It would be interesting to study how these systems might benefit from a multi-touch interface. For creating complex curved surfaces, desktop systems often rely on the specification of indirect control points (e.g., NURBS), which can take time to learn to use effectively.

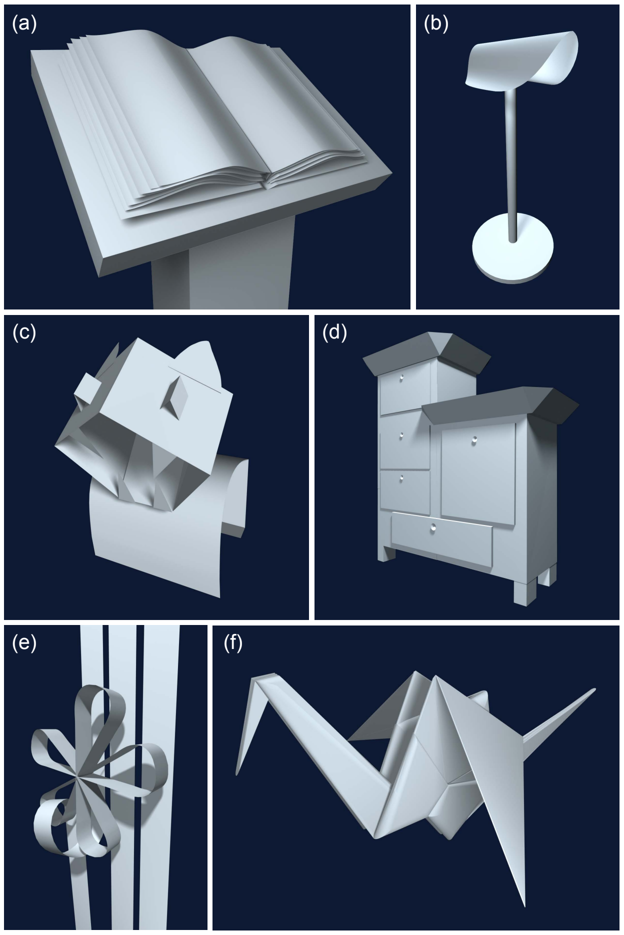

Figure 2. Six models created in our system (3D geometry shown only): (a) a book on a podium, (b) a lamp, (c) a papercraft tiger, (d) an armoire, (e) ribbons, and (f) an origami paper crane.

Gesture-Based Modeling Systems.

Most modeling techniques have one-point-based interaction – e.g., dragging a point or set of points to squash or stretch, grabbing something and moving it. There have been various papers describing and evaluating methods for transforming objects with multi-touch, e.g., [1, 3, 6, 12, 13]. However, this is just one small aspect of a modeling system. Some researchers have begun to explore how a multi-touch interface could be used to model primitive and abstract shapes [2, 8] or to interact with 3D objects [6]. De Araùjo et al. [4] introduced a semi-immersive environment for preliminary conceptual modeling without haptic feedback. While their experience was satisfactory, the modeling system itself proved insufficient for precise control of 3D geometry. Walther-Franks et al. [21]

performed a preliminary study by augmenting Blender, the 3D modeling tool, with a multi-touch interface. The multi-touch operations received positive feedback, but are mainly limited to object *animation* functions, rather than geometric modeling tools.

Recently, Autodesk launched a set of mobile products for modeling and design, including 123D Design and 123D Sculpt – notably simplified tools compared to their desktop counterparts. However, the modeling functions and interaction in these systems are still typical of *one-point-based* desktop modeling/sculpting systems – beyond view manipulation, little is done to take advantage of a multi-touch interface. Several domain-specific modeling systems have been created for multi-touch interfaces, such as Eden (for constructing organic sets) [10], Sun et al.'s system for architectural design [19], and Wilson et al.'s physics simulator [22]. However, these systems are not suitable for casual, general-purpose modeling, and are primarily targeted at domain-specific professional users.

Papercraft Simulators

Papercrafting is a set of art forms that use paper as an artistic medium to create physical 3D objects. In particular, origami, the traditional Japanese art of paper folding, has been extensively studied from a computer science and applied math perspective. Origami simulators have been implemented in various forms, including Maya plugins [17], stand-alone applications employing desktop interfaces [20], and others [14, 15]. Rather than producing a faithful simulation of origami, we use the time-honored principles of origami as a stepping stone to create a novel 3D modeling paradigm and system to design freeform 3D objects and scenes. Similar to the way sketching inspired the Teddy system by Igarashi et al. [7], we take inspiration from simple interactions with physical paper.

To summarize, our main goal is to propose a novel 3D modeling technique inspired by papercrafting and implemented as an intuitive system designed for a multi-touch interface. The following sections present details of Paper3D.

THE PAPER3D MODELING SYSTEM

To create an effective multi-touch modeling system, we observed that paper folding – a simple, physical process virtually anyone can identify with – has strong synergy with a multi-touch interface on a flat-surface device. We built on folding with additional physically inspired operations including cutting, bending, pinning and taping. We then further expanded the system to allow the assembly of sets of individually defined component objects into complex scenes. This form of inspiration allows us to focus purely on developable surfaces.

In the traditional papercraft art form of origami, a person begins with a single, square sheet of paper. Only three types of folds are permitted: mountain folds (forming a ridge), valley folds (forming a trough), and creases. Many combination folds exist (e.g., reverse, rabbit-ear, squash), but these are simply compositions of the three basic folds. Origami, and particularly the work of [15] inspired several initial modeling functions in our system. However, our work evolved into a modeling technique with broader functionality, extending

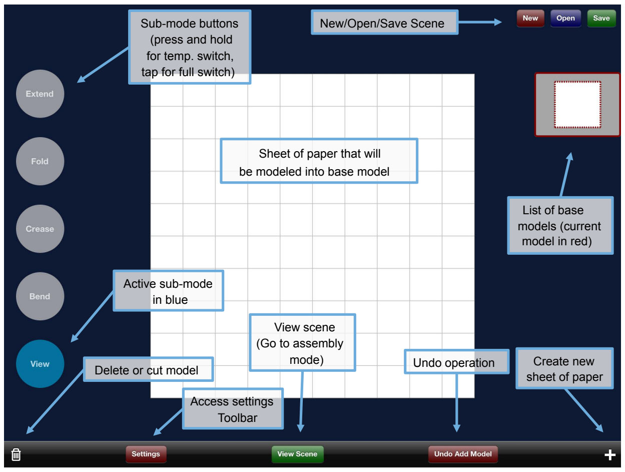
Sub-mode buttons (press and hold for temp. switch, tap for full switch)
New/Open/Save Scene
New
Open
Save
Extend
Fold
Crease
Bend
View
Sheet of paper that will be modeled into base model
List of base models (current model in red)
Active sub-mode in blue
View scene (Go to assembly mode)
Delete or cut model
Undo operation
Create new sheet of paper
Access settings Toolbar
Settings
View Scene
Undo Add Model

Figure 3. Interface for the *modeling* mode of Paper3D. Radio buttons (left) allow switching between submodes; the view submode is active. Model thumbnails (right) are used to select/edit a different model. The bottom toolbar offers additional functionality such as undoing. The *assembly* mode interface has different submodes but an otherwise similar interface.

far beyond the constraints of traditional origami or origami simulators. We give an overview of our modeling technique.

This section gives a high-level overview of Paper3D (interface shown in Figure 3), followed by an in-depth description of the system's modeling and assembly tools in the next section.

Intuitive, Gesture-Based Interactions

Paper3D is created from the ground up for a tablet device with a multi-touch display (a third-gen. iPad). As such, the majority of user interaction with the system occurs through *intuitive* single- and multi-finger gestures. An intuitive interaction is one that has an easily understandable gesture mapping, is predictable and straightforward to replicate, and results in immediate visual feedback to the user. In the context of modeling, we focus on creating interactions that have a tangible, direct effect on a model. All user gestures requires a specific number of inputs – e.g., a 1-finger pan or a 2-finger pinch. We use the notation t_i to represent the normalized screen coordinate of the $(i+1)^{th}$ gesture input, and define t_{i_start} and t_{i_end} as the initial and new/final screen coordinates of the user's finger. The corresponding points projected along the camera view direction onto the plane of current face f are p_{i_start} and p_{i_end}. The primary gestures we used are illustrated in Figure 4.

Scene Objects

Similar to a standard polygonal mesh, each individual object, or *3D component* in our system consists of interconnected *faces*, *edges* and *vertices*. Each face is a convex, planar area of the component model; its closed outline is composed of

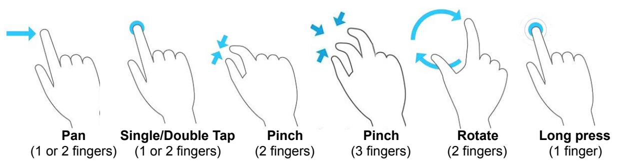

Figure 4. Examples of multi-touch gestures used in Paper3D.

n edges and vertices ($n \geq 3$). Every edge has exactly two vertices, and is classified as a *boundary edge* (belongs to a single face, or two folded-over faces), a *fold* (between two non-coplanar faces), or a *crease* (between two coplanar faces). Two or more edges can have the same vertex; we refer to a vertex belonging to two boundary edges as a *corner*.

System Modes

In Paper3D, there are two main stages of design (and corresponding modes): a *modeling* stage and an *assembly* stage. Figure 3 shows the interface of the modeling mode. Users can transition between the modes at any point. The *modeling mode* allows users to model sheets of source paper (one at a time) into 3D components through creasing, folding, bending, extending and cutting operations, as outlined in the previous section. In the assembly mode, users can insert models, transform them, and then intuitively group them together through pinning and taping operations.

Owing to the range of operations in each mode, certain functions require the same gesture. To remove ambiguity, the interface of each mode has five radio buttons to the left of the screen, grouping sets of operations into submodes. In the modeling mode, the five submodes are *extend*, *fold*, *bend*, *crease/cut*, and *view* (Figure 3). In the assembly mode, the submodes are *insert*, *transform*, *pin/tape*, *color*, and *view*. If desired, a user can rest a finger of their non-dominant hand on a submode button, activating the submode only for as long as they are holding the button. This can streamline operations – e.g., by allowing quick viewing of a model between modeling operations. In the modeling mode, thumbnails of existing models can be used to select a different model for editing. In the insert submode of assembly mode, these thumbnails can be used to drag in copies of each model into the scene.

3D MODELING AND ASSEMBLY TOOLS

This section describes user operations available to model paper in our system into 3D components, and subsequently assemble and texture them to create 3D scenes. Implementations of our modeling operations typically involve uses of functions $\text{SPLIT}(f, l)$ (divides a face f into two new faces along line l), $\text{ROTATE}(f, l, \theta)$ (rotates f about line l by an angle θ), and $\text{TRANSLATE}(f, l, d)$ (translates f a distance d along line l).

Gesture-Based 3D Modeling Tools

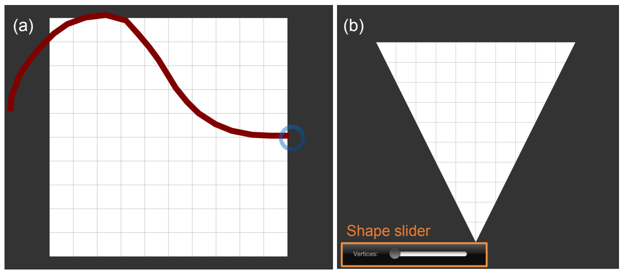

Figure 5. Creating a new sheet of virtual paper: (a) Users can define a freeform outline for the paper with a single finger pan. (b) Alternatively, a regular polygon can be selected using a shape slider to control the number of vertices (see inset).

Creating a 3D component begins with an initially flat sheet of paper that can be resized and stretched using a pinch gesture, which maps the two gesture inputs to the two opposite corners of the sheet. Its shape may also be defined with a freeform outline tool, or by choosing a predefined polygonal shape (e.g., triangle, circle). (See Figure 5.)

Planar Operations.
Three fundamental operations available to the user are *creasing* (dividing one or more faces into two), *extending* (creating new, connected, coplanar faces), and *cutting* (dividing part of a model along a crease). The user defines a crease through two finger inputs p_0 and p_1, panning to adjust its position and orientation. (See Figure 6a.) Upon releasing, the face is divided through a $\text{SPLIT}()$ operation into two new faces connected along the crease. A outward pinch with fingers over a selected crease and one of its connected faces cuts that side of the model, either removing it or making it a separate component. (See Figure 6b.) To extend the current sheet, as seen in Figure 6c, the user can drag two fingers along the outward normal of any boundary edge, creating a new face. Two of its vertices are of the crossed boundary edge; the remaining ones are defined by the two projected gestural inputs p_0 and p_1.

Extending.
In addition to planar extending, we provide additional extend tools. Users can bridge two edges using an *edge-to-edge extend*, by placing a finger over one edge and swiping the other edge towards the first (Figure 6d). A three-finger pinch on a face will extend the paper along all the boundary edges of a face simultaneously, along the normal to the original face. The distance the fingers are spread apart controls the width of the new faces, and the angle of the new faces can be subsequently adjusted (Figures 6e and 6f). A four-finger pinch performs an extend operation on all the faces of the component simultaneously, allowing users to quickly add thickness to their model. Any curved face is extended along its original normal prior to the added curvature.

Folding.
A one-finger drag over one of the adjoining faces of a selected crease adjusts the angle of the crease (through a call to ROTATE()), turning it into a folded edge, with the angle defined by t_0 projected onto the normal of the edge. All faces connected to the rotating face (on the same side of the crease/fold) are rotated together with it. The angle of any existing crease or fold of the 3D component may be adjusted in this way. In a *corner fold* operation, a user's finger is dragged across the screen over a corner of a face (selecting it), and the user starts to fold over the face. (See Figure 6g.) The folding line l_{fold} is defined as the line perpendicular to $l(p_{start}, p_{end})$ and passing through the midpoint of p_{start} and p_{end}. The new, folded-over face f_{folded} is rotated 180° about l_{fold}. Similarly, *edge folding* lets the user fold over the paper by grabbing any boundary edge of the component (Figure 6h), with its final position unambiguously defined through a two-finger pan gesture. The midpoints of the two start and end touches, p_{0_mid} and p_{1_mid}, define the folding line l_{fold}, and the new face f_{folded} is rotated 180° about l_{fold}. (See Figure 7a and 7b.)

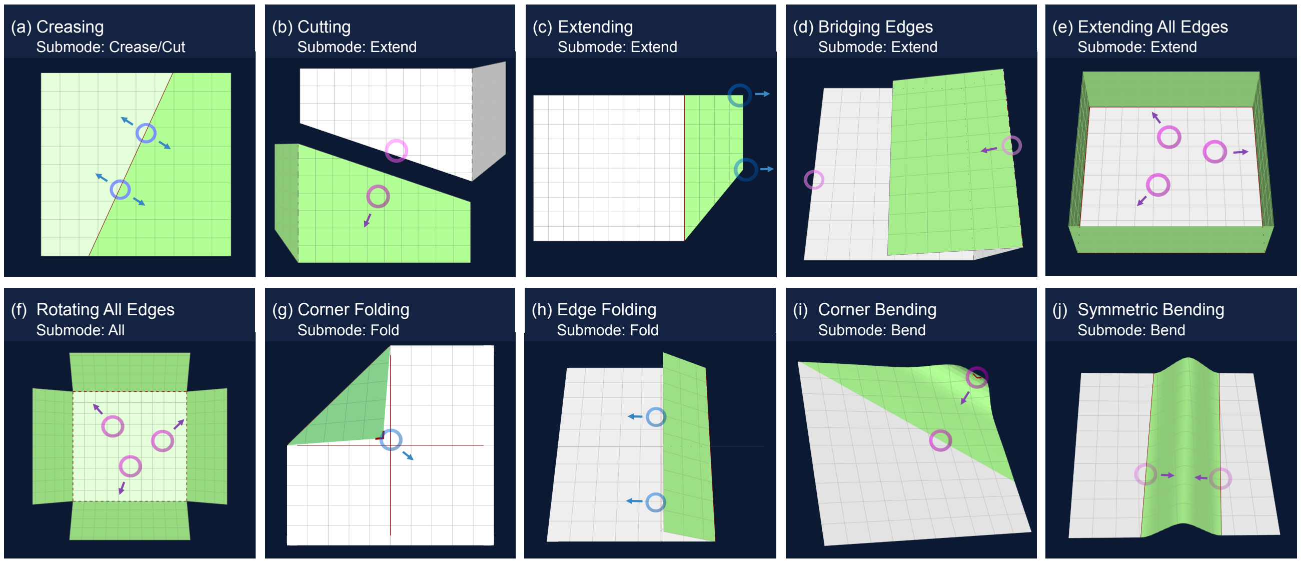

Figure 6. A selection of modeling tools in Paper3D: (a) creasing a sheet of paper, (b) cutting along an edge, (c) extending along an edge of a face, (d) bridging edges of two faces, (e) extending all edges of a face, (f) rotating all faces generated in (e), (g) corner folding, (h) edge folding, (i) corner bending, and (j) symmetric bending. Circles and arrows indicate the position and motion of user touches.

Algorithm 1 Corner bending

```
1: procedure CORNERBEND($f_{orig}, t_0, t_1$)
2:     $p_{1\_end} \leftarrow$ PROJ($f_{orig}, t_1$)
3:     $w_{orig} \leftarrow$ LENGTH($p_{0\_start} - p_{1\_end}$)
4:     $l_n \leftarrow$ LINE($p_{0\_start}, p_{1\_end}$)
5:     $l_{fold} \leftarrow$ LINE($p_{1\_end}$, NORMAL($l_n$))
6:     if ISBENDABLE($f_{orig}, l_{fold}$) then
7:         $f_{fixed}, f_{folded} \leftarrow$ SPLIT($f_{orig}, l_{fold}$)
8:         $f_{temp} \leftarrow$ FACE(VECTOR($l_n$), NORMAL($f_{orig}$), $p_{1\_end}$)
9:         $p_{0\_end} \leftarrow$ PROJ($f_{temp}, t_0$)
10:        BEND($f_{folded}, p_{0\_end}, p_{1\_end}, w_{orig}$)
11:    end if
12: end procedure
13:
14: procedure BEND($f, p_0, p_1, w_{orig}$)
15:     $f_{bend\_prev} \leftarrow f$
16:     $l_{parts}, \theta_{folds} \leftarrow$ COMPUTEBENDSTRIPS($p_0, p_1, w_{orig}$)
17:     for $i \leftarrow 0, n_{parts}$ do
18:         $f_{bend\_prev}, f_{bend\_curr} \leftarrow$ SPLIT($f_{bend\_prev}, l_{part\_i}$)
19:         ROTATE($f_{bend\_curr}, l_{part\_i}, \theta_{fold\_i}$)
20:         ADDTOSUBFACES($f, f_{bend\_curr}$)
21:         $f_{bend\_prev} \leftarrow f_{bend\_curr}$
22:     end for
23: end procedure
```

Angled Folding and Bending.

A pinch gesture on a multi-touch device lets us define an *angled corner folding* tool, allowing a user to simultaneously fold over a corner of the sheet of paper, while also controlling the folding angle between the fixed and folded-over parts. The folding line l_{fold} is defined as the line perpendicular to LINE(p_{0_start}, p_{1_end}) and passing through p_{1_end}. (See Figure 7c.) The angle of rotation of face f_{folded} is found using p_{0_end}, the 3D point found by projecting t_{0_end} onto the plane passing through p_{0_start} with normal parallel to l_{fold}. *Angled edge folding* is analogous, except it starts on a boundary edge instead of a corner. In *corner bending*, we extend angled folding to allow users to curve parts of their model (Figure 6i). The angled face f_{folded} is curved into n bend strips, through a recursive sequence of n SPLIT() and ROTATE() operations. Dividing lines l_1 to l_n are parallel to the initial dividing line l_{fold}. Following [15], we determine the line spacing and the angle of each rotation by minimizing the energy function

$$E = \alpha\Sigma_i(d_i - L)^2 + \beta\Sigma_i(a_i - a_{i+1})^2,$$

where L is a constant equal to $1/k$ multiplied by the distance between the line l and the selected corner at p_{0_start}, and d_i and a_i are the widths of and angles between the bend strips. Constants $\alpha = 0.6$ and $\beta = 0.4$ were determined experimentally. (See Figure 7d and Algorithm 1.) If the user's fingers are close together, the paper is curved into a cylindrical shape (instead of using the energy minimization function), connecting points p_{0_end} and p_{1_end}. *Edge bending* instead bends over an edge of the model, but is otherwise identical.

Symmetric Folding and Bending.

A *symmetric bend* is activated when a user pinches an inner region of the paper. As the user's two fingers are drawn together, the paper bends upwards, forming a peak at the midpoint (Figure 6j). In the implementation, we define w_{orig} as half the original distance between the user's initial, projected contact points p_{0_start} and p_{1_start}. w_{bend} and h_{bend} are defined as the width and height of half the bent portion of the paper, computed based on the new/final contact points p_{0_end} and p_{1_end}. The original face f_{orig} is first divided along the line passing through the midpoint of p_{0_end} and p_{1_end}, and perpendicular to the line through those points. The two resulting faces are translated inwards by a distance equal to $w_{orig} - w_{bend}$, and then each is split into two new faces along the fold lines through points p_{0_end} and p_{1_end}. Finally, the inner two faces f_{0_folded} and f_{1_folded} are curved, such that they meet symmetrically at the peak of the bend p_{peak}, and maintain their original dimensions. (See Figure 7e and Algorithm 2.) The orientation of the bend and the thickness and height of the bend are readjusted in real-time in response to user input, until

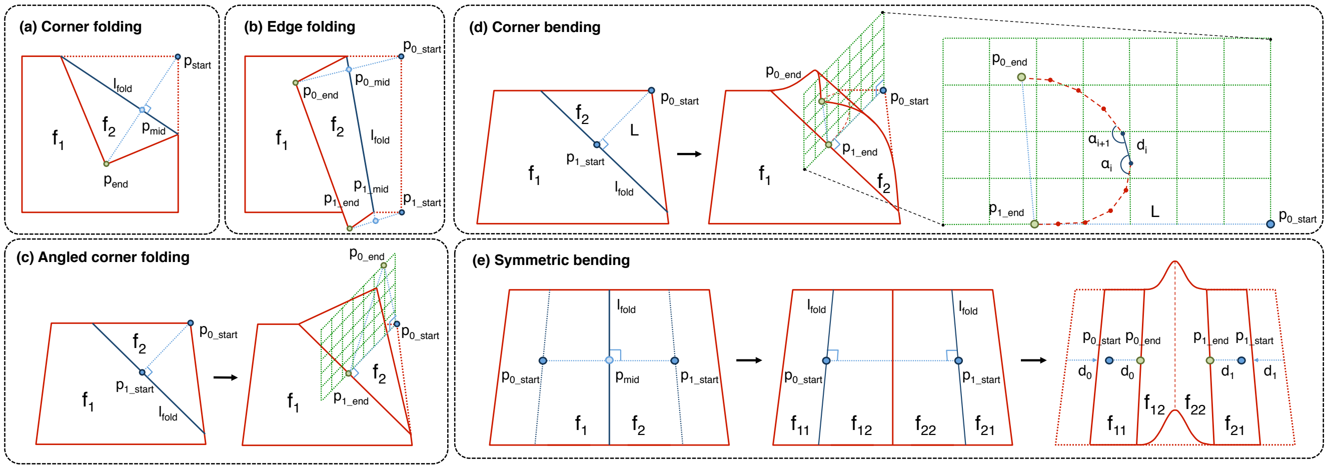

Figure 7. Technical details for a selection of tools in Paper3D. (a) and (b) show how the folding line l_{fold} is calculated for corner and edge folds. (c) illustrates how the two user inputs are projected onto a plane perpendicular to the current face, allowing the user to specify both the fold angle and orientation. (d) shows how a curve is fit between the two projected input points, resulting in the curved surface. (e) shows the effect a symmetric bend has on a face of the model, translating each half inwards as the bend is increased.

Algorithm 2 Symmetric bending

```
1: procedure SymmetricBend($f_{orig}, t_0, t_1$)
2:     $w_{orig} \leftarrow 0.5 * \text{LENGTH}(p_{0\_start} - p_{1\_start})$
3:     $w_{bend} \leftarrow \text{MIN}(0.5 * \text{LENGTH}(p_{0\_end} - p_{1\_end}), w_{orig})$
4:     $h_{bend} \leftarrow \text{SQRT}({w_{orig}}^2 - {w_{new}}^2)$
5:     $p_{0\_end} \leftarrow \text{PROJ}(f_{orig}, t_0)$
6:     $p_{1\_end} \leftarrow \text{PROJ}(f_{orig}, t_1)$
7:     $p_{mid} \leftarrow 0.5 * (p_{0\_end} + p_{1\_end})$
8:     $p_{peak} \leftarrow p_{mid} + h_{bend} * \text{NORMAL}(f_{orig})$
9:     $l_n \leftarrow \text{LINE}(p_{0\_end}, p_{1\_end})$
10:    $l_{0\_fold} \leftarrow \text{LINE}(p_{0\_end}, \text{NORMAL}(l_n))$
11:    $l_{1\_fold} \leftarrow \text{LINE}(p_{1\_end}, \text{NORMAL}(l_n))$
12:    $l_{mid\_fold} \leftarrow \text{LINE}(p_{mid}, \text{NORMAL}(l_n))$
13:    if IsBendableRegion($f_{orig}, l_{0\_fold}, l_{1\_fold}$) then
14:        $f_0, f_1 \leftarrow \text{SPLIT}(f_{orig}, l_{mid\_fold})$
15:        $d_{trans} \leftarrow (w_{orig} - w_{bend})$
16:        $\text{TRANSLATE}(f_0, l_n, d_{trans})$
17:        $\text{TRANSLATE}(f_1, l_n, -d_{trans})$
18:        $f_{0\_fixed}, f_{0\_folded} \leftarrow \text{SPLIT}(f_0, l_{0\_fold})$
19:        $f_{1\_fixed}, f_{1\_folded} \leftarrow \text{SPLIT}(f_1, l_{1\_fold})$
20:        $\text{BEND}(f_{0\_folded}, p_{0\_end}, p_{peak}, w_{orig})$
21:        $\text{BEND}(f_{1\_folded}, p_{1\_end}, p_{peak}, w_{orig})$
22:    end if
23: end procedure
```

the user's fingers are lifted off the screen. Subsequently, the position of the peak can be adjusted horizontally and vertically. Analogous to symmetric bending, a *symmetric fold* forms a peak at the midpoint of the two points of contact, where the portion of paper between the two pinched fingers is folded upwards, creating a folded peak in the middle.

Transforming, Grouping and Texturing Tools

Once a selection of single-sheet components has been constructed using our system's modeling operations and inserted into the scene, they can be transformed and assembled into more complex models using pinning and taping tools.

Pinning and Taping.

A user pins two objects together by first tapping on a face f_0 of a model m_0. This creates a pin at projected location p_0, with normal equal to f_0's normal. A subsequent tap of face f_1 of a second model m_1 indicates a desired location and alignment of the first model. The two faces are pinned together; this is highlighted through animation. If m_0 has been pinned to the wrong side of f_1, it can be flipped over using a rotate gesture. Adjustments can subsequently be made by translating within the plane of f_1 or rotating about the pin axis. (See Figures 8a and 8b.) Taping is similar to pinning, except a user first taps an edge e_0 of m_0 (with two fingers, to distinguish from pinning), and then taps an edge e_1 of model m_1. The same transformation occurs as above, except now, e_1 is constrained to lie on e_0. The user can rotate m_0 about the axis of the tape (the line passing through e_1), or translate along this axis (Figures 8c and 8d).

Transforming and Duplicating.

Individual and assembled models can both be transformed through uniform scaling (pinch gesture), translation within the plane of any face of the model (two-finger pan), and rotation about any edge of the model (one-finger pan). In addition, users can temporarily drop a pin or tape onto a model (without grouping), and translate and rotate about the pin/tape. Both individual models and groups can be duplicated using a one-finger drag operation.

Coloring and Texturing.

As part of the assembly mode, we provide coloring and texturing tools to embellish the resulting models. In the coloring/texturing submode, a color picker allows users to select any desired color. Subsequently, a single one/two/three-finger tap on a face of a model will change the color of the face/model/group to the selected color. (See Figure 8e.) A long press over a model sets the current color to that of the indicated face. For texturing, the list of model thumbnails is replaced by a list of texture thumbnails. Once a texture is selected, it can be applied to a face/model/group in the same way as a color. The texture mapping can be adjusted using a pinch gesture. (See Figure 8f.)

IMPLEMENTATION DETAILS

Paper3D was implemented from the ground up for a tablet device with a multi-touch display (a third generation iPad). As

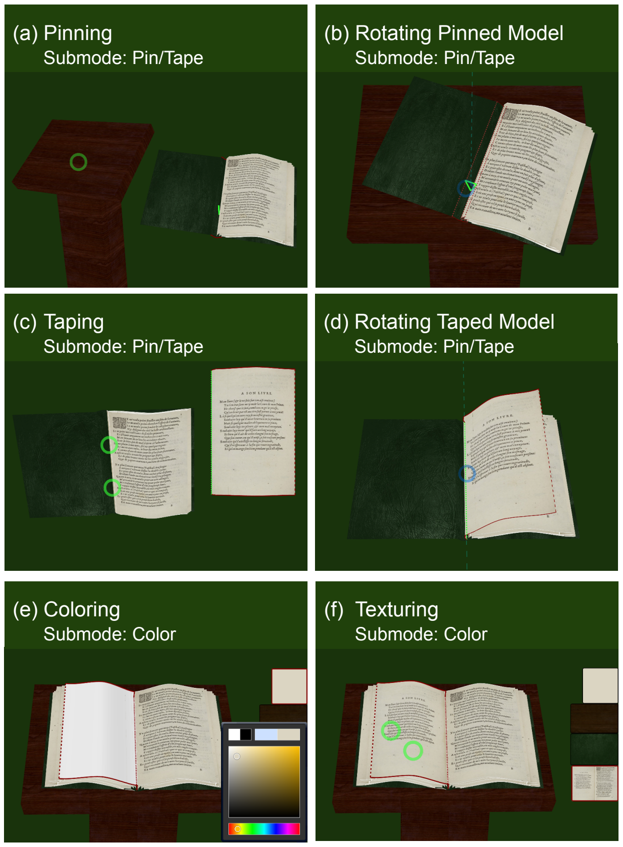

Figure 8. A selection of assembly operations. (a) The user adds a pin to the spine of the book. Tapping over the top of the bookstand attaches the two together at the pin location. (b) Users can then rotate the book about the pin, or translate the book in the plane of the stand. (c) The user has added tape to the edge of the loose book page; a two-finger tap over the spine of the book then attaches the page to the spine, aligning the two edges. (d) The alignment is adjusted. (e) The user can color an object using the color palette and a subsequent tap gesture. (f) Textures can be mapped onto any surface using a pinch gesture.

such, the majority of user interaction with the system occurs through single- and multi-finger gestures. Our system is implemented in a mix of C++ and Objective C, with OpenGL ES as the rendering API.

Object Storage

Each object instance in our system (i.e., vertices; edges; faces) is stored dynamically, and has a unique identifier. Standard map containers store pointers to these objects for easy access. Each model stores three separate maps for its faces, edges and vertices, respectively. As shown in Figure 9, each face stores a map of its edges and an ordered list of its vertices, while edges store pointers to their associated faces and vertices.

Modeling Updates

Figure 9 shows an example of how the internal data structures of the system are updated after a crease operation. Original face f_0 is divided into new faces, f_1 and f_2. The outline of f_0 is split: vertices v_0, v_5, v_6, v_2, and v_3 now make up the outline of f_1, while v_5, v_1 and v_6 comprise the outline of f_2. We maintain a history of operations performed on the models, as shown in the tree structures in the same figure. Each face has

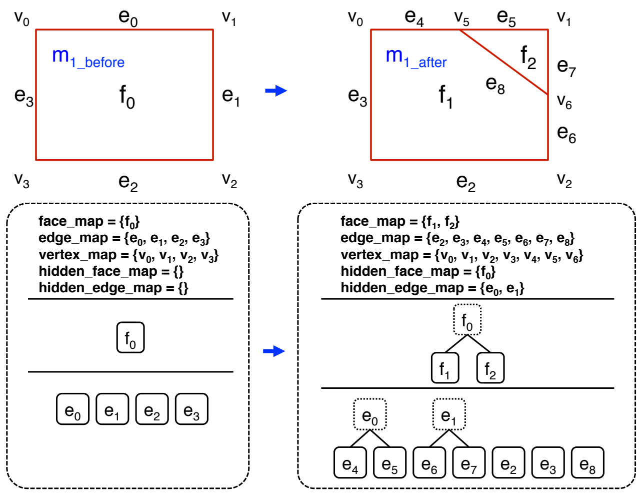
v_0
e_0
v_1
m_{1_before}
e_3
f_0
e_1
v_3
e_2
v_2
v_0
e_4
v_5
e_5
v_1
m_{1_after}
f_2
e_7
e_8
e_3
f_1
v_6
e_6
v_3
e_2
v_2
face_map = {f_0}
edge_map = {e_0, e_1, e_2, e_3}
vertex_map = {v_0, v_1, v_2, v_3}
hidden_face_map = {}
hidden_edge_map = {}
f_0
e_0 e_1 e_2 e_3
face_map = {f_1, f_2}
edge_map = {e_2, e_3, e_4, e_5, e_6, e_7, e_8}
vertex_map = {v_0, v_1, v_2, v_3, v_4, v_5, v_6}
hidden_face_map = {f_0}
hidden_edge_map = {e_0, e_1}
f_0
f_1 f_2
e_0 e_1
e_4 e_5 e_6 e_7 e_2 e_3 e_8

Figure 9. Overview of the object data structure in our system. The right-hand side shows how the data structure is updated when a face of the model is split (e.g., through folding).

a parent face and two children, all initially set to null. In this example, the parent face f_0 is labeled as hidden, and its left and right children are set to f_1 and f_2, respectively, while their parent is set to f_0. Edges are divided and updated similarly: e_0 is split into e_4 and e_5, e_1 is split into e_6 and e_7, while a new edge e_8 is created between f_1 and f_2. Parent and child edges are updated, as are edge map containers. If Figure 9 instead showed a corner bend operation, f_2 would subsequently be replaced by a curved surface, by recursively dividing it into a set of bend strips of near-equal length and parallel to edge e_8. Undoing an operation simply involves accessing the tree structures to determine the original objects.

RESULTS AND DISCUSSION

Paper3D went through a natural, user-guided evolution. Once we had a stable, preliminary version of our system, we showed our system to four modelers: three casual 3D modelers with only about three months of modeling experience, and a professional designer with extensive modeling experience. These four modelers learned to use our system (5–10 hours of practice), and each created a selection of 3D models using Paper3D, while providing us with feedback. This feedback resulted in subsequent modifications to the system.

3D Modeling Results

Figure 10 displays a selection of models and scenes created by the independent modelers using our system. Our best estimate is they spent an average of 4–5 hours conceptualizing and creating each scene. We attribute some of this to the fact that throughout the modelers' usage, we were making periodic changes to the system in response to user feedback. Figure 10a shows a papercraft-like jungle scene, with a collection of animals surrounded by plants and trees, easily created with the modeling operations of our system. The modern architectural interior in Figure 10b is composed of curved surfaces that would have been difficult to create in an existing modeling system. Figure 10c shows a more traditional interior of a library; again, note the nontrivial curvature of the lounge chair and the lamps. Figure 10d displays an artsy design made out of intertwined, twisting ribbons, and emphasizes

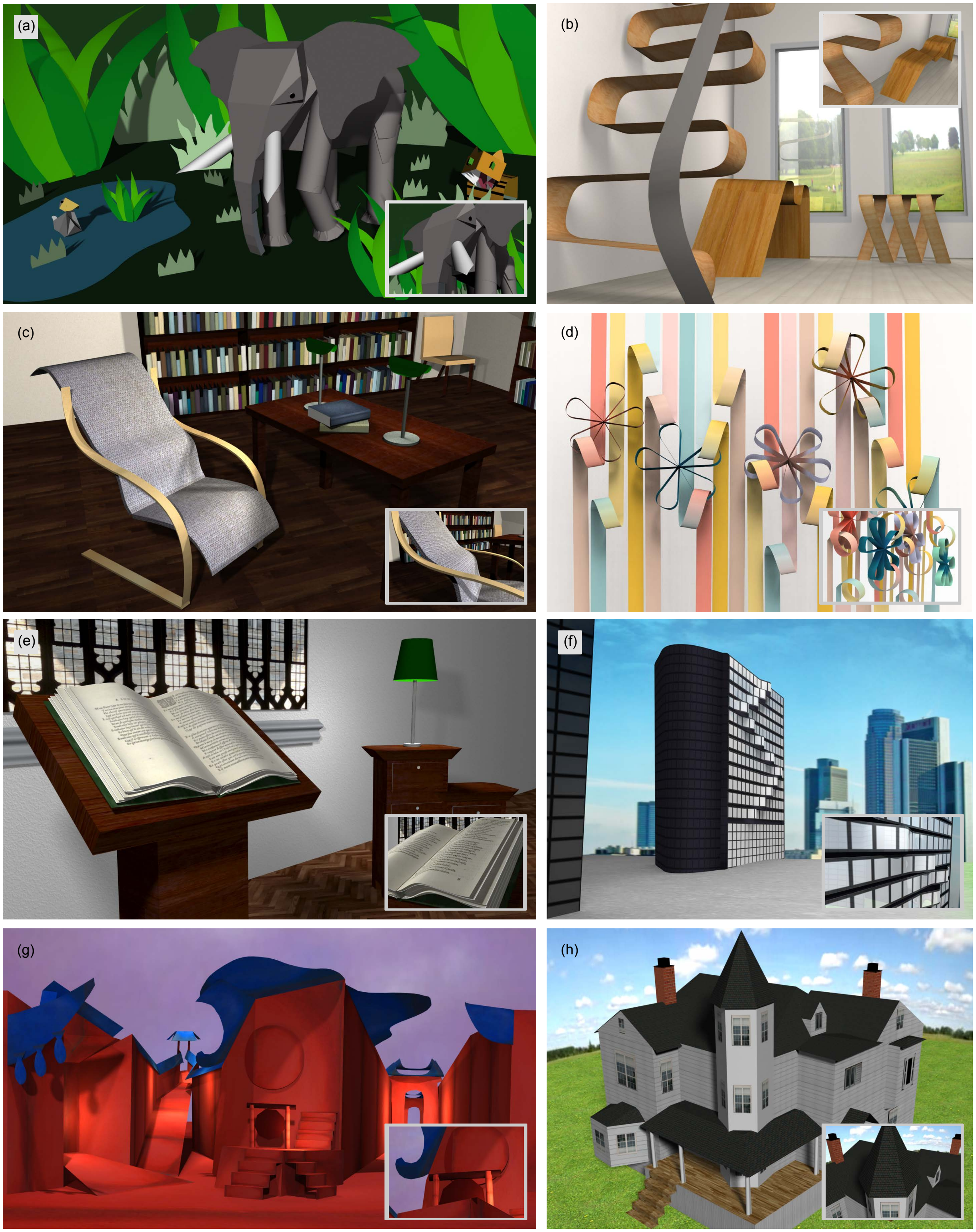

Figure 10. A selection of 3D models and scenes created by users of our system. (a) A papercraft-like jungle scene; (b) a modern architectural interior; (c) a library scene with a curved lounge chair; (d) an artsy design made out of intertwined, twisting ribbons; (e) a manuscript on a podium, with a lamp lighting an armoire in the background; (f) a modern, curved skyscraper; (g) a David Hockney inspired stage set; (h) a Queen Anne style house.

the simplicity with which natural forms of curvature can be modeled in Paper3D. Another such example is illustrated by the manuscript lying on the podium (Figure 10e), though the other elements of this scene were also created with little effort in Paper3D. The cross-section of the skyscraper in Figure 10f contains an array of curves which were easily modeled using the bending functions in our system. Figure 10g shows a David Hockney-like stage-set design; the paper-modeling operations of our system are ideally suited for a model like this one. Finally, though a system like SketchUp is tailored to creating models such as the Queen Anne style house (Figure 10h), we demonstrate that users of Paper3D can also successfully produce these types of models.

Discussion of System Evolution

We discuss a range of key design decisions that we made throughout development of the system, both a result of user feedback, and our own observations and insights.

Gesture Mapping.

Though we used papercrafting as an inspiration, it was often not appropriate to directly simulate certain interactions, but rather to use gestures appropriate in the context of 3D modeling with a multitouch device. For example, the crease gesture in Paper3D uses a pinch gesture to define the orientation of the crease, even though this does not mimic physically creasing paper. Similarly, a corner fold uses a one finger pan – more natural on an iPad than grabbing the corner by pinching. Other operations, however, such as a symmetric bend, precisely mimic the physical interaction. In some cases, gestures with three or more fingers (and in some cases, only two fingers) led to significant screen occlusion, affecting our choices.

Curved Surfaces.

Bending paper is not only a very natural, physically-realizable operation, but it can also be effectively simulated on a multi-touch interface. Users can control two degrees of freedom simultaneously, streamlining the curve creation process. Paper3D is limited to creating developable surfaces, by virtue of using sheets of paper as a starting point – a conscious design decision, as doubly curved surfaces can be intimidating for inexperienced modelers. Instead, a strength of our system is making curved surface creation more accessible to casual 3D modelers.

Rendering Folded Edges.

It was important to us to give users a visual sense that they are manipulating paper (as long as the rendering quality does not affect performance). Narain et al. [16] introduced a sophisticated simulation technique for rendering folded and crumpled paper; however, this algorithm could not run in realtime. Furuta et al. [5] provided a rendering algorithm, but only for flat, not curved edges. In Paper3D, we decided to render all folds as thin, curved areas between the two adjoined faces. The radius of the curve is determined in part by the relative ordering and computed offsets of layers of folded-over faces. Folding a face over itself multiple times results in increasingly thicker, more curved edges (Figure 11), giving models a more volumetric appearance.

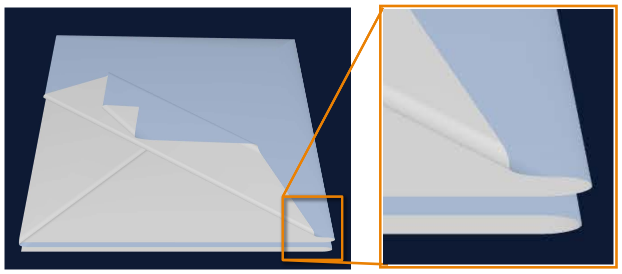
Figure 11. The representation of non-boundary edges in our system, i.e., creases, mountain folds and valley folds.

USER EVALUATION

The user-guided evolution of Paper3D afforded a running evaluation of our modeling system, following the long-term evaluation method of Shneiderman and Plaisant [18]. Once Paper3D evolved into its current form, we asked each of our testers for a summary review of the system, detailing both their experience with the tool, and how the system compares to previously used modeling systems. We present their overall impression of Paper3D, along with other notable feedback.

Overall Experience.

All four modelers found their experience with the system to be very positive. One of the modelers thought Paper3D was more enjoyable than Autodesk Maya, since she was immediately able to learn all its functions and features, whereas she found Maya to be overcomplicated, with many features she had no idea how to use. The gesture-based system was singled out as more intuitive than using a mouse. The modelers found our system easier to learn, and objects within it easier to manipulate. All modelers thought Paper3D was enjoyable to use, and saw it as very accessible to people with little or no prior modeling experience. Users considered our system a competitive alternative to 3D modelers.

Mobility.

The mobility of the iPad is an advantage over larger display devices (e.g., an Eizo Multitouch Radiforce device). The designer supported this, pointing out that Paper3D's mobility makes the modeling experience more enjoyable. Additionally, she observed that Paper3D could be useful as a casual, portable modeling tool for quick prototyping in the early phase of design.

Precision.

Our users observed that using a mouse provides better precision than hand gestures, though in general, this did not negatively affect their modeling process. This agreed with our assumption that while the precision of a tablet device may not be sufficient for a professional modeler, it should suffice for a casual one. For enhanced precision, we provided a range of snapping features (e.g., corner-to-corner snapping during folding), and a secondary, orthogonal view for more complex operations. In the future, we plan to add guides for more accurate model selection, placement and grouping.

Number of Submodes.

Though users did not mind having too many submodes, optimizing their number is an area of future work. Additionally, since more emphasis was placed on developing the system's gesture-based interface and modeling paradigm, improving the general button and toolbar layout of the present interface will be another future area of focus.

CONCLUSION AND FUTURE WORK

We have presented Paper3D, a novel 3D modeling system designed specifically for multi-touch interfaces. Our iPad application introduces a new style of modeling inspired by papercrafting, with intuitive, familiar operations, such as folding, bending, cutting, pinning, and taping. Our system is easily learned by experienced and casual modelers alike, allowing users to create a wide range of developable surfaces. We validated our claims by asking modelers to test Paper3D and use it to create a variety of models, while providing valuable feedback for iterative development. Their final reviews of the system were uniformly positive and encouraging. Several interesting future directions of Paper3D remain, such as improving input precision, experimenting with additional forms of curvature, improving rendering efficiency and quality, extensions to support animation, and hardcopy output.

ACKNOWLEDGMENTS

Julie Dorsey acknowledges support from the National Science Foundation under Award 1018470. Min H. Kim acknowledges partial support from Korea NRF grants (2013R1A1A10-10165 and 2013M3A6A6073718) and an ICT R&D program of MSIP/IITP (10041313). We thank Yumiko Nakamura, Victoria Nielsen, Marko Micic (Yale University), and Seijin Cha (KAIST) for their invaluable feedback and modeled designs.

REFERENCES

1. Au, O. K.-C., Tai, C.-L., and Fu, H. Multitouch gestures for constrained transformation of 3D objects. *Comput. Graph. Forum 31*, 2 (2012), 651–660.

2. Chang, S. H.-H., Stuart, L., Plimmer, B., and Wünsche, B. Origami simulator: a multi-touch experience. In *CHI Extended Abstracts*, ACM (2009), 3889–3894.

3. Cohé, A., and Hachet, M. Beyond the mouse: Understanding user gestures for manipulating 3d objects from touchscreen inputs. *Comput. Graph. 36*, 8 (Dec. 2012), 1119–1131.

4. De Araùjo, B. R., Casiez, G., and Jorge, J. A. Mockup Builder: Direct 3D modeling on and above the surface in a continuous interaction space. In *Proc. Graphics Interface* (2012), 173–180.

5. Furuta, Y., Mitani, J., and Fukui, Y. A rendering method for 3D origami models using face overlapping relations. In *Proc. Smart Graphics 2009*, vol. 5531, Springer (2009), 193–202.

6. Hancock, M. S., Carpendale, M. S. T., and Cockburn, A. Shallow-depth 3D interaction: design and evaluation of one-, two- and three-touch techniques. In *Proc. ACM CHI* (2007), 1147–1156.

7. Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: A sketching interface for 3D freeform design. In *Proc. SIGGRAPH* (1999), 409–416.

8. Joshi, A., Robertson, G., Wünsche, B., and Plimmer, B. Bubbleworld builder - 3D modeling using two-touch and sketch interaction. In *Proc. GRAPP* (2010), 116–122.

9. Keijser, J., Carpendale, S., Hancock, M., and Isenberg, T. Exploring 3D interaction in alternate control-display space mappings. In *Proc. Symp. on 3D User Interfaces* (2007), 526–531.

10. Kin, K., Miller, T., Bollensdorff, B., DeRose, T., Hartmann, B., and Agrawala, M. Eden: A professional multitouch tool for constructing virtual organic environments. In *Proc. ACM CHI*, ACM (New York, NY, USA, 2011), 1343–1352.

11. Lee, S. K., Buxton, W., and Smith, K. C. A multi–touch three dimensional touch–sensitive tablet. In *Proc. ACM CHI* (1985), 21–26.

12. Liu, J., Au, O. K.-C., Fu, H., and Tai, C.-L. Two-finger gestures for 6DOF manipulation of 3D objects. *Comput. Graph. Forum 31*, 7-1 (2012), 2047–2055.

13. Martinet, A., Casiez, G., and Grisoni, L. The design and evaluation of 3D positioning techniques for multi-touch displays. In *Proc. the Symposium on 3D User Interfaces* (2010), 115–118.

14. Mitani, J. The folded shape restoration and the rendering method of origami from the crease pattern. In *Proc. Int. Conf. on Geometry and Graphics* (2008), 1–7.

15. Miyazaki, S.-Y., Yasuda, T., Yokoi, S., and Toriwaki, J.-I. An origami playing simulator in the virtual space. *J. of Vision and Computer Animation 7*, 1 (1996), 25–42.

16. Narain, R., Pfaff, T., and O'Brien, J. F. Folding and crumpling adaptive sheets. *ACM Trans. on Graph. (TOG) 32*, 4 (2013), 51.

17. Nitsch, E. J. *When pigs fly: a study of computer generated paper folding*. M.S. thesis, Texas A&M University, 2008.

18. Shneiderman, B., and Plaisant, C. Strategies for evaluating information visualization tools: multidimensional in-depth long-term case studies. In *BELIV 06: Proc. the 2006 AVI workshop on Beyond* (2006).

19. Sun, Q., Lin, J., Fu, C.-W., Kaijima, S., and He, Y. A multi-touch interface for fast architectural sketching and massing. In *Proc. ACM CHI*, ACM (New York, NY, USA, 2013), 247–256.

20. Tachi, T. Rigid-foldable thick origami. *Origami 5* (2011), 253–264.

21. Walther-Franks, B., Herrlich, M., and Malaka, R. A multi-touch system for 3d modelling and animation. In *Proc. Smart Graphics*, Springer-Verlag (Berlin, Heidelberg, 2011), 48–59.

22. Wilson, A. D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., and Kirk, D. S. Bringing physics to the surface. In *Proc. UIST 2008*, ACM (2008), 67–76.