What you Sculpt is What you Get: Modeling Physical Interactive Devices with Clay and 3D Printed Widgets

Michael D. Jones
Dept. of Computer Science,
Brigham Young U.
3328 TMCB, Provo, UT 84602
Mike.jones@byu.edu

Kevin Seppi
Dept. of Computer Science
Brigham Young U.
3328 TMCB, Provo, UT 84602
Kevin_seppi@byu.edu

Dan R. Olsen
Dept. of Computer Science
Brigham Young U.
3328 TMCB, Provo, UT 84602
olsen@sparxteq.com

ABSTRACT

We present a method for fabricating prototypes of interactive computing devices from clay sculptures without requiring the designer to be skilled in CAD software. The method creates a "what you *sculpt* is what you get" process that mimics the "what you *see* is what you get" processes used in interface design for 2D screens. Our approach uses clay for modeling the basic shape of the device around 3D printed representations, which we call "blanks", of physical interaction widgets such as buttons, sliders, knobs and other electronics. Each blank includes 4 fiducial markers uniquely arranged on a visible surface. After scanning the sculpture, these fiducial marks allow our software to identify widget types and locations in the scanned model. The software then converts the scan into a printable prototype by positioning mounting surfaces, openings for the controls and a splitting plane for assembly. Because the blanks fit in the sculpted shape, they will reliably fit in the interactive prototype. Creating an interactive prototype requires about 30 minutes of human effort for sculpting, and after scanning, involves a single button click to use the process.

Author Keywords
Physical computing; 3D printing; prototypes

ACM Classification Keywords
D.2.2 Design Tools and Techniques, H.5.2 User Interfaces.

INTRODUCTION

We envision deployable prototypes of physical computing devices similar to that shown in Figure 1. These are single purpose devices that must physically fit into a specific context. The shape of the devices must fit human form and motion. For example, the bicycle mounted music player controller in Figure 1a must be easily accessible by a person on a bicycle. The steering wheel mounted stereo controls in Figure 1b must be carefully placed to lie under the thumb of a hand holding the steering wheel. Similarly, widgets on the camera controller in Figure 1c should be placed to allow operation without looking at the device to locate the controls.

An interactive prototype allows the designer to determine if the shape and interaction will work with human form and motion. It is difficult to make that assessment when looking at a rendering on a computer screen. Finally, it is also important to be able to rapidly iterate the design. The designer needs to be able to go from idea to working prototype quickly so that she can work through many iterations in a short amount of time. Shape alone, is not sufficient. Interactive devices must integrate with the shape and the resulting prototypes must be interactively functional to test usability in context. Finally, it is critical that the functional prototype preserves the shape and widget placements as laid out in the sculpture. Otherwise, widgets may not be correctly placed relative to a user's finger shape, position and movement.

A diverse set of prior approaches to prototyping physical interactive devices share many of these goals. Many of these projects, such as Gadgeteer [15], Phidgets [5] and d.tools [6] and many others, present novel electronics and software systems for building physical interactive devices. We see a need for better methods for integrating the shape of the device with the interactive widgets. The Switcharoo [2], Calder [9], and BOXES [7] projects also describe new methods for creating the device shape but we aim for more fidelity and fluidity in design of the shape than found in these projects.

Savage in Makers' Marks [11] also merges form and function for interactive 3D printed objects using a process based on scanning physical representations of sculpted objects. The maker annotates the object with stickers that represent the approximate positions of input or output elements such as buttons or lights. Sticker positions are recovered after scanning and geometry is added before printing to support assembly using the required components. Stickers provide fluidity but not fidelity in the design process. Stickers do not represent the shape of interactive elements above the surface, do not represent the volume of components beneath the

CHI'16, May 07-12, 2016, San Jose, CA, USA
 ISBN 978-1-4503-3362-7/16/05 $15.00
DOI: http://dx.doi.org/10.1145/2858036.2858493

a b c

Figure 1 Three examples of physical interactive devices.

surface, and may not accurately represent widget locations on curved surfaces. In some cases the captured shape of the housing is extended with a boxy protrusion to make room for components beneath the surface.

Like Makers' Marks, we see value in scanning physical representations of interactive widgets but we envision a process in which the designer works with the actual widget shapes knowing that any sculpted form can be converted into a functional prototype exactly as it was sculpted. Precisely preserving widget placement and device shape is particularly important when designing a device intended to fit comfortably in the hand or to have buttons placed under the fingers when held in a certain way.

Our process begins with sculpting in clay around 3D printed blanks which represent interactive widgets such buttons, sliders and knobs. The blanks are partially embedded in the clay to represent electronics that must fit below the surface and to expose interactive widgets that lie above the surface. Blanks can be easily repositioned in the wet clay. The sculpture is scanned and the scanned mesh is modified to create an identically shaped 3D printable housing with the intended widget placements. After assembling widgets in the housing, the designer has an interactive prototype.

One problem with this process is that there is a significant amount of geometry processing that happens between scanning and printing. The triangular mesh created by the scanner needs to be augmented to support the interactive widgets and prototype assembly. This processing includes adding mounting geometry for the widget parts, cutting holes in the surface for interactive widgets that lie on the surface and making it possible to assemble widgets in the housing. Of these steps, adding mounting geometry is particularly difficult because the geometry must be positioned and oriented precisely to match widget placement in the sculpture.

We present an algorithm that searches through the scanned mesh to find fiducial markers that were 3D printed into the widget blanks. After finding markers, the algorithm searches for groups of markers that match marker placement on the different widget types. Widget type, position and orientation can be recovered from matching groups of markers and then the additional structure can be added in the right places. The process depends on creating several models, included in the model for the blank, derived from a 3D model of the widget and its electronics but this only needs to be done once per widget type. This algorithm allows a designer to convert a scanned mesh into a prototype housing with one button click. The prototypes shown in Figure 1were created using this process. Each prototype required less than 30 minutes of scanning and was correctly processed by our algorithm.

DESIGN PROCESS

Our approach supports a process in which the design is created by physical means. In this section we describe this process and highlight key problems that must be solved to support that process.

The designer sits at a table with clay and 3D printed blanks that represent widgets as shown in Figure 2a. In the photograph, trays of blanks sit to the right of the widgets they represent. There are also 3D printed blanks to represent

a b c d

Figure 2 Sculpting around printed blanks that represent widgets to make an interactive physical device.

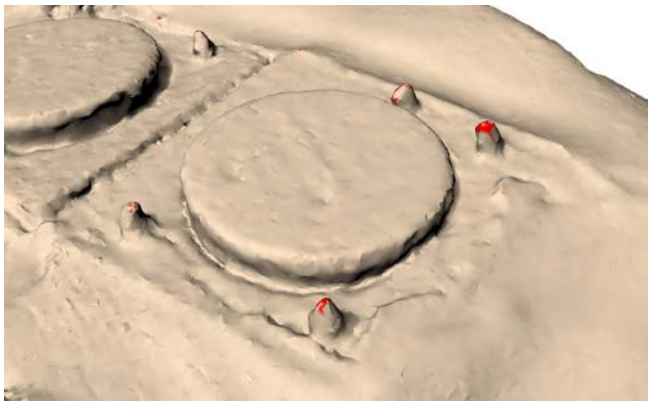

Figure 3 In this scanned model of a sculpted prototype, vertices on fiducial markers are colored red.

batteries, processors and other elements that might be needed in the functional prototype.

The designer molds clay around the blanks while leaving interaction surfaces exposed to create a clay model of the prototype shown in Figure 2b. This clay model includes the interactive widgets embedded in the right places for interaction. In this case, the steering wheel mounted controls are placed comfortably under the thumb when the hand grasps the steering wheel.

Sculpting around blanks is a key part of the process. The resulting prototype will be functional only if the actual widgets can be implemented in their location as specified in the sculpted form. It is not useful to allow the designer to sculpt a housing with a specific placement of widgets only to later discover that that placement is not feasible in the sculpted shape. The printed blanks include volume below the surface to represent the electronics and other parts needed to implement that element above the surface. Later we discuss the use of offset surfaces relative to models of complete widget assemblies as a solution to this problem.

After creating the clay model, the designer scans the sculpture. We used a scanner with resolution of 0.13 mm. The locations of the blanks and the widgets they represent must be recovered from the scanned model and this is done by first finding fiducial markers in the scan. Markers are grouped into predefined arrangements printed into the blank. Blanks were 3D printed using a stereolithography (SLA) process that has a layer thickness of 0.028 mm and a resolution of 600 dpi. Printing blanks at lower resolution using a fusion deposition process (FDM) resulted in misshaped markers that were difficult to find in the scanned model. From these groupings the algorithm recovers widget type, placement and orientation.

The scanned sculpture is processed and printed. The housing and electronics are shown in Figure 2c. Processing the scanned mesh involves adding mount points for electronics and other parts needed to implement a widget, splitting the housing to provide access to the interior, and adding snaps so the housing can be closed after assembly.

The designer can then use the prototype in the intended context as shown in Figure 2d. This allows for evaluation of shape and functionality in ways that are difficult to do when looking at a shape on a computer screen or when using a prototype connected to a workstation by a cable. The designer can repeat the process either by starting over with a new sculpture, modifying the existing sculpture or modifying the scanned model in software.

This design process and the process in Makers' Marks [11] share four key elements. Both involve tangible representation of the device shape, scanning the representation to recover the shape, recovering interactive widgets from the scanned model, and the automatic placement of mounting geometry to support interactive widgets.

The key difference between the Makers' Marks design process and ours is that we represent the physical constraints of widgets to the designer in a natural way using a 3D printed blank. Blanks accurately represent a widgets shape above and below the object surface. Makers' Marks indicate only the widgets position using a stickers as markup on the object surface. In Makers' Marks the design may have to be perturbed to accommodate constraints not represented by stickers.

These two methods offer different design experiences. Stickers are easy to reposition on the surface while moving a 3D printed blank embedded in clay involves moving the clay, repositioning the blank, and replacing the clay. Furthermore 3D printed blanks must be placed inside the object. In order to properly represent the spacing inside of the object we do not allow approaches like the taping together of cardboard tubes as in Makers' Marks' box example [11]. However this approach could be replicated with more effort by cutting holes in the tubes to place the printed blanks inside.

The power of our approach lies in the fact that the designer can manipulate the design without the extra cognitive load of assessing the internals of the design. Thus if the desired parts fit in the design, then it will be possible to assemble the object without further adjustment. Several problems need to be solved to full enable this "what you sculpt is what you get" design process based on clay and 3D printed blanks. These problems are:

- ensuring that widget placements in the sculpture can be implemented as placed,
- finding fiducial markers in the scanned mesh,
- grouping fiducial markers into individual widget placements,
- recovering widget type, position and orientation from widget placement,
- creating the housing by adding mounting points and

- splitting the scanned model for easy assembly and adding snaps to hold it closed after assembly.

In the following sections we describe how we solved these problems and share insights gained along the way.

FINDING FIDUCIAL MARKERS

Our process depends on being able to automatically recover the type, position and orientation of each widget using only a scanned model (mesh) of the design. To do this we need to recover the locations of the fiducial markers that identify each widget type and placement. This involves traversing the vertices in the mesh and labeling vertices that lie on one of these markers. Figure 3 shows markers colored red in a rendering of a scanned model of the sculpture from Figure 2b.

Cones with height 3 mm and a base diameter of 2 mm serve as markers. We chose cones because they have a single point with unique curvature. Cylinders also have unique curvatures, but the unique curvature is found on every point of the rim rather than just one on the tip. We experimented with several cone heights and base diameters before using the 3 mm tall cones with 2 mm bases. Smaller, shorter cones were difficult to locate in scanned meshes. Larger, taller cones poked into the designer's fingers and were more distracting than smaller cones.

We find the tips of cones in the scanned mesh by analyzing the slope at each vertex. The scanned mesh consists of vertices, or points, and edges connected into triangles. At the tip of a cone, the surface slopes downward in every direction. The algorithm computes the slope in the direction of the neighboring vertex. If the slope is downward in every direction, then that vertex must lie on a cone.

We compute the slope to a vertex as shown in Figure 4. To find the slope at A in the direction of vertex B we first compute the vector n which is the normal at A. The vector n is perpendicular to the surface at A and is computed by averaging the normal for the triangles that share A as a vertex. Given n and B we compute a coordinate system at A in which n points in the positive z direction and x is perpendicular to z but parallel to the line that connects A and B. In Figure 4 the x axes from A to each of the other vertices are shown in red. We project the position of B into this coordinate system and then compute dx, the change in x, and dz, the change in z, from A to B. The slope from A to B is dz/dx.

We repeat the process for neighboring vertices B, C, D and E (and for vertices not shown in the illustration). For each neighbor we compute a new coordinate system in which x is parallel to the line from A to the neighbor, and then compute the slope dz/dx in that coordinate system.

After computing the slope in each direction we check that each slope is within the range -0.6 to -1.5. If any slope is outside this range then there is a direction on the surface in which the slope is too flat or too steep for that vertex to lie on the tip of a scanned fiducial marker.

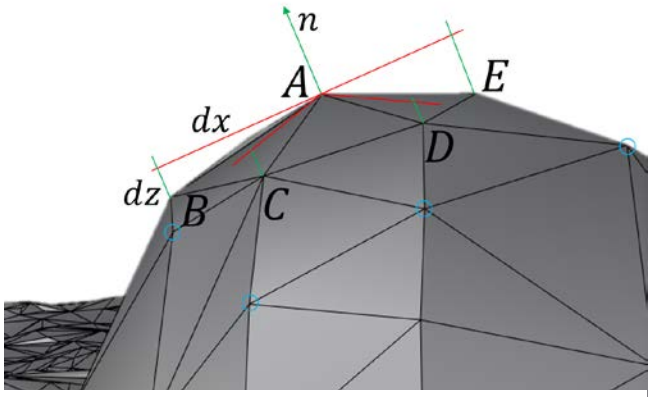

Figure 4 Close up view of mesh vertices on a scanned fiducial marker. We locate vertices on the marker by computing the slope from a vertex to neighboring vertices.

We tried classifying vertices using an estimate of signed mean curvature. The signed mean curvature is the average of the principle curvatures at a vertex. The principle's curvatures are the largest and smallest local curvatures, and the local curvatures are precisely the slopes computed using dz/dx as described above. But signed mean curvature loses too much information and misclassifies more vertices than the threshold method described above. A key feature of a vertex on a bump is that *all* normal curvatures lie within a given range not just that the average of the minimal and maximal normal curvature.

We compute the slope to the neighbors two edges away from a vertex rather than using vertices only one edge away. In Figure 4 this means we use the slope to the vertices circled in blue rather than vertices B, C, D and E. This gives a larger span over which to estimate slope, and makes the algorithm less sensitive to noise. This decision depends on the mesh being at a certain resolution relative to the size of the bumps in a blank. It works well when a vertex on a bump top has at least 2 generations of neighbors that also lie on the bump.

This method for finding vertices on fiducial markers is insensitive to changes in mesh resolution. Reducing the resolution of the mesh matters because processing a mesh with fewer vertices takes less computational time and memory. The mesh resolution reduction algorithm we used preserves a dense collection of vertices on curvy parts of the mesh and leaves a sparse collection of vertices in flat areas. This is because one flat triangle approximates a flat plane just as well as many flat triangles. Because the fiducial markers are curvy, they are left with many vertices which simplifies labeling vertices that lie on them.

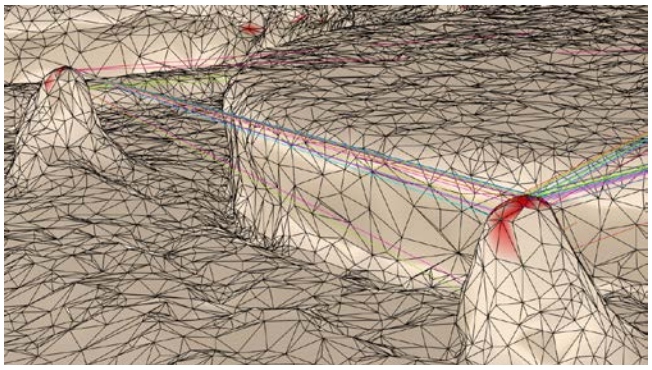

Figure 5 Each colored line connects a marked vertex to 3 other marked vertices with together represent a widget placement. Duplication is due multiple labeled vertices on each marker.

The algorithm often finds several vertices per bump in the scanned mesh. For example in Figure 4 vertices B, C, D and E might also be labeled as lying on bumps. In our approach it is better to identify several vertices on bumps than to miss all of the vertices on a bump. Finding several vertices per bump causes the widget location algorithm to find many widget placements (15 is average) per actual widget blank embedded in the mesh. This happens because each vertex on each bump can be grouped with several vertices on other bumps to create a valid placement. Figure 5 shows vertices labeled on a marker and the connections found on neighboring markers. Each colored line is part of a different widget placement found by the algorithm. We handle these duplicate placements by merging placements that are in close proximity as described in the next section.

Cones used as fiducial markers can be reliably located in the scanned model and this supports recovery of widget type, placement and orientation. Recovering widget placement information is a necessary step in automatically converting a scanned sculpture into an interactive prototype with working widgets.

FINDING WIDGET PLACEMENTS

Finding fiducial markers alone does not identify the type or placement of widgets. What we need to know is which groups of markers represent a widget placement. The algorithm that groups markers into placements should ignore mislabeled vertices that do not actually lie on markers and should find only the valid widget placements.

Figure 6 shows widget positions and orientations recovered from vertices marked on the scanned mesh. Markers are highlighted with colored balls, the plane of the widget is shown with a blue triangle and a vector perpendicular to the plane is shown in purple. Other colors in the image correspond to the vertex order and will be explained in the next section.

The grouping algorithm searches through combinations of labeled vertices to find groups that match the relative positions of markers printed into widgets. The markers on a widget are always arranged into unique asymmetric groups of 4 markers.

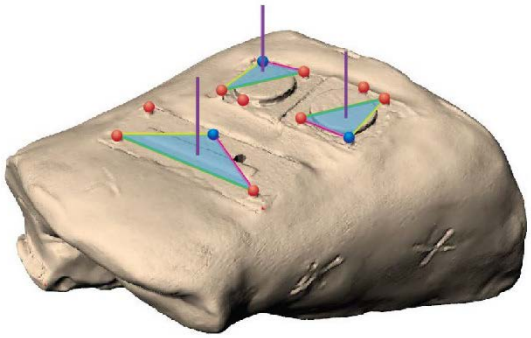

Figure 6 Widget placements recovered by searching for properly spaced clusters of 4 markers and then merging duplicate cluster that lie in close proximity.

Although 3 markers are sufficient to recover position and orientation, we use 4 markers per widget. Using 4 markers rather than 3 eliminates many false positives when finding widget placement. With 3 bumps there are 3 distances to check. With 4 bumps there are 6 distances to check between every pair of bumps. Checking more distances allows the algorithm to exclude more spurious placements.

When checking distances between markers, the algorithm has a tolerance of 0.7 mm. In our tests, we found that about 75% of the distances between vertices on bumps in a widget placement were within 0.5 mm of the expected distance. The other 25% of the distances were between 0.5 and 0.7 mm of the actual distance with larger errors over larger distances.

The algorithm searches through every feasible ordered combination of up to 4 vertices starting with ordered pairs of 2 labeled vertices. We reduce the search space by only searching through labeled vertices that are close enough to the starting vertex. If the greatest distance between two bumps on a widget blank is 31 mm then we only search through labeled vertices within 31.7 mm of a given vertex. The additional 0.7 mm accounts for tolerance in distance matching.

After finding all widget placements the algorithm merges placements in which the center of the placement is closer than 2 mm to the center of another placement. In most cases the actual distance between any two placement centroids is either less than 1 mm or greater than the width of a widget blank on the surface. Merging nearby placements smooths out noise in the widget placements found when several vertices are marked on each marker location.

Finding the widgets in the scanned sculpture is the key step in supporting a design process that includes widgets embedded into clay models. With the bump grouping and

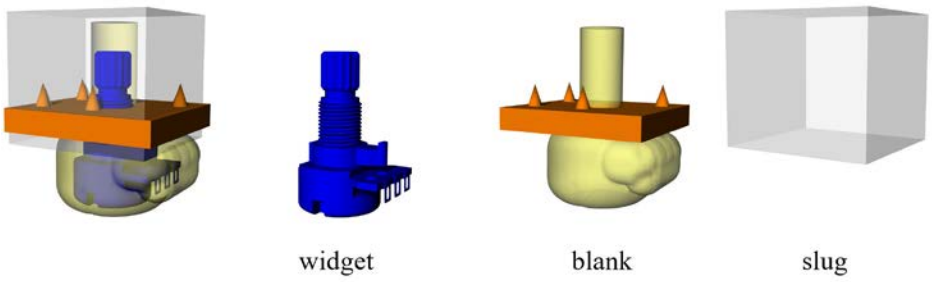

Figure 7 Widget, blank, and slug models for a potentiometer knob.

placement information, we now have what we need to compute widget placement.

RECOVERING WIDGET TYPE AND ORIENTATION

In order to save designer time and effort by automatically placing widget mounts, the algorithm needs to know the type and orientation of the widgets. This information needs to be recovered from the 4 fiducial markers that represent the widget placement.

The widget type is encoded by a unique arrangement of the 4 markers. In this scheme the 4 markers printed into any given widget are positioned at different distances from each other than the 4 markers on any other widget.

After recovering the widget type, the algorithm keeps only the first 3 marker locations rather than all 4. Each marker is numbered and the number is used to compute orientation. We recover vertex numbers from the distances between fiducial markers. The distance between every pair of markers is different for a given widget type. Once the four markers are found, it is simple to match the distances and recover the marker numbers. The plane of the widget placement is defined by markers 1 through 3. In Figure 6 each blue ball gives the location of marker 1 for each widget placement. The yellow line connects marker 1 to 2 and the magenta line connects marker 1 to 3. The plane is shown by the blue triangle. The widgets at the top of the model are both buttons but one is rotated 180 degrees compared to the other. The bottom widget is a slider

The markers used to determine the plane of the placement should be as far apart as possible, should not lie in a single line and should form a right triangle if possible. This gives a larger basis for defining the plane and is less sensitive to noise introduced in scanning.

Given widget placement, orientation and type, the algorithm can now modify the scanned mesh to create a 3D printable housing for the prototype. This will save designer time and effort as the designer does not need to make these modifications by hand.

MODELING WIDGETS

Prototyping with printed widget blanks embedded in clay requires building two models based on the widget geometry. These models ensure that the widget can be implemented in the space allotted and provide fiducial markers for recovery of placements.

Modeling the widgets is not part of the prototype construction process but needs to be done before the process begins. It needs to be done once per widget type and requires CAD skills. Widget models are based on an accurate 3D representation of the widget which includes everything needed to interact with the widget and to implement the widget.

Several additional models are derived from the base widget model. These are shown in Figure 7 with a composite view on the left. The base widget model is shown in blue. The base model is in this figure is a knob implemented using a potentiometer. The mounting bracket is a plate shown in solid orange. This widget attaches to the housing by attaching it to that plate. Other widgets have more complex mounting geometry and this would be included as part of the mounting bracket. The grey box is called the “slug” and used to cut a hole in the housing to accommodate the mounting bracket. The printed blank that represents this widget consists of the mounting bracket and the yellow offset mesh.

The yellow geometry below the plate is a mesh offset by 3 mm from the widget in order to allow space for the widget in the sculpture interior. In this context, offsetting the mesh means moving every vertex in the mesh 3 mm along that vertex’s normal. The offset vertices are remeshed to create a mesh that is 3 mm from the original mesh. Adding this offset to the printed blank allows the designer to sculpt down to the printed widget model while still leaving room for the housing wall to enclose the widget. Without this offset the designer must guess where the widget is inside the opaque modeling clay.

The four fiducial markers need to be added to the visible parts of the orange plate or the visible parts of the blue model above the orange plate. If the 4 markers do not lie on the plane of the orange plate then the widget model must also include the angles between the plane of the first 3 markers and the plane of the orange plate.

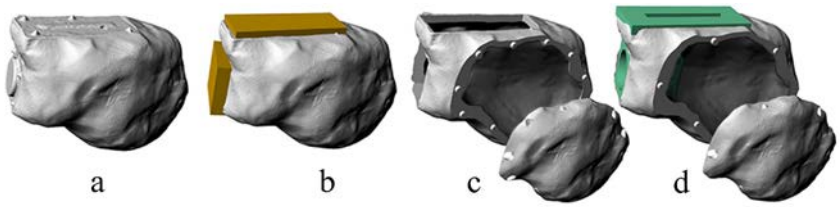

Figure 8 Creating a 3D printable housing from a scanned mesh of sculpted prototype.

We also include 3D printed blanks to represent additional electronic components, such as a processor or a battery, that are completely enclosed in the housing. These blanks are included in the sculpture to make sure that there will be enough space for those parts in the printed prototype without modifying the housing shape. For the devices shown in this paper, we used blanks to represent the volume of an Arduino processor and a battery and embedded those in the sculpture.

We do not locate the blanks that represent embedded electronic components that lie completely within the housing because they are hidden beneath the surface. We also do not add mounting brackets for these elements. Instead, the designer must remember which enclosed electronics belong in the housing and place them by hand.

The blank, slug, bracket and parts should be stored in the same coordinate space in the positions relative to each other. By convention, we stored the widget with the tip of marker 1 on the origin and the tip of marker 2 on the positive x axis. Markers 1, 2, and 3 all lie on the xy plane. The vector created by taking the cross product of the vectors connecting marker 1 with 2 and 1 with 3 points in the positive z direction. This relative placement and alignment is important because it means we know the position and orientation of the parts when they are read into the scanned model space. Any alignment with the model coordinate system will suffice, but it must be unambiguous.

The widget and associated models can now be used in their printed form in sculpting and can be loaded in digital form to prepare the prototype housing for printing and assembly. The investment in modeling time and effort needed to create the widget models pays of in time and effort saved later during prototyping.

CREATING A HOUSING FOR THE PROTOTYPE

Creating a housing involves 4 steps: cutting holes for the interactive elements that support widgets, splitting the housing to allow assembly, adding snaps so the housing can be closed after assembly, and adding mounting brackets and other parts for each widget. These steps are shown in Figure 8, as will be explained in the text below, beginning with the scanned sculpture in Figure 8a and ending with the final printable housing in Figure 8d.

Cutting holes for widgets. Holes need to be cut so that interactive widgets can be exposed on the prototype surface. For each widget placement, the algorithm loads the slug, this is the grey box in Figure 7, which encloses all of the volume needed for interactive elements and for mounting brackets. This slug is shown in brown in Figure 8b. The holes are created by a Boolean subtraction operation which subtracts the slug from the scanned mesh. The holes are visible in Figure 8c (which also shows the housing split into two pieces for assembly).

Splitting the housing. Assembling widgets for the interactive prototype requires access to the interior of the housing. We provide that access by splitting the housing into two pieces. The split must provide a large opening through which electronics can be moved into position. We find a plane for splitting the housing by evaluating 1600 different planes that intersect the housing. Figure 9 shows the feasible planes explored for a mesh with the final selection shown in red. A splitting plane is feasible if it passes through the mesh inner and outer walls and does not intersect a hole cut for widget placement. The optimal splitting plane creates the largest opening in the housing. The area of the opening is the area of the polygon created on the plane when the mesh is intersected with the plane. The plane with the largest area is selected and used to split the housing mesh.

Adding snaps. The housing pieces need to be fastened back together after assembly. The algorithm adds between 6 and 12 post and hole pairs to the matching faces created by splitting the housing. The posts and holes can be seen Figures 8c and 8d. Posts with radius 1.3 mm proved durable in assembly and reassembly of the printed housing. Larger

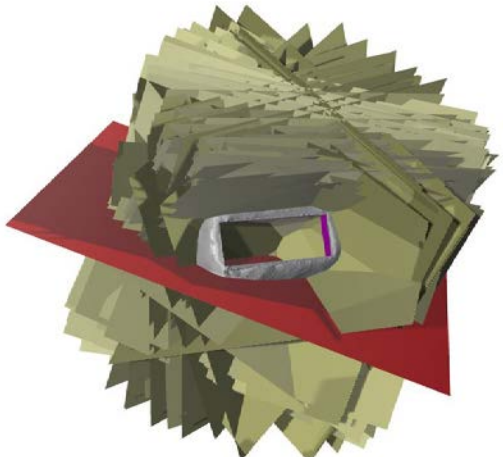

Figure 9 Finding a plane to split the housing. The red plane was selected from the feasible candidates shown in yellow.

posts would have been more durable but posts and holes need to be small enough to fit on the rim created in the mesh wall.

The hole radius is 0.35 mm larger than the post radius so that the printed post makes a friction fit into the printed hole[1]. The friction fit was tightest when the housing was printed with the plane of the rim parallel to the plane of the print head. Small ridges created in the layer by layer deposition process left small ridges in the post and hole and these ridges fit snugly together with the 0.35 mm tolerance. Other 3D printing process may require different approaches to adding snaps.

Placing brackets. Brackets and other printed pieces needed to implement widgets are added next. These are shown in green in Figure 8d and correspond to the orange bracket in Figure 7. The geometry for brackets and other pieces for each widget placement are loaded, rotated and translated into position. Brackets are joined to either the top or bottom of the split housing using a Boolean union operation. Moving parts that are not attached to the housing are simply left in place to be printed in place.

These modifications to the mesh yield a housing for a functional prototype that is ready for printing and assembly.

EXAMPLES

We have constructed several interactive physical computing prototypes using this process. Prototypes were created by two undergraduate university design students who are part of the research team but who did not write any of the software described above. In each case sculpting required less than 30 minutes of the designer's time, converting the scanned mesh into a printable housing required pressing a button to start the algorithm. The algorithm correctly found widget placement, orientation and type and then prepared the housing for printing and assembly. All sculptures were assembled into functional prototypes. These prototypes are each shown in Figure 1. Four other devices were sculpted by different users and correctly processed by the algorithm to create housings that could be printed and assembled.

Each prototype involves a different hand position relative to the interactive widgets. This means that each prototype shape accommodates different motion of the fingers, hand or arm. Sculpting gives the designer tactile feedback on position and shape needed to support each motion in each context. The camera remote is hand-held. The bike mounted mp3 player control requires moving the arm to access the controls. The steering wheel mounted controller only requires moving the thumb.

Bike mounted mp3 player controls. Using this device involves moving the entire arm while seated on a bicycle.

Figure 10 A widget set together with printed blanks used in sculpting. The widgets are a button on the top left, a knob on the top right and a slider on bottom. The button includes a complex mounting bracket printed in white and a moving plunger.

During sculpting the designer sat on a bicycle with the clay wrapped around the bicycle tube that connects the handle bars to the seat (this tube is called the "top tube"). This let the designer check the required arm motion in context. The sculpture wrapped just less than half way around the top tube so that it could be removed for scanning without cutting or moving the clay. The printed controller was attached to the top tube using a thin layer of wet clay. This was sufficient for short test rides (as seen in Figure 1) but double sided foam tape or a stronger adhesive might be needed for longer term testing.

Placing the controller on the top tube is reminiscent of placement of bike gear shifters on the tube connecting the handlebars to the crank arm assembly (this is called the "down tube"). More recent bicycle models put shifters on the handlebars. After testing the interactive prototype on a bicycle ride the designer might decide to place the controls on the handlebars, the down tube, or somewhere else. In either case, designing a new prototype likely involves another 30 minutes of sculpting and pressing a button to create the housing.

Steering wheel mounted mp3 player controls. Unlike the bike mp3 player controller, the steering wheel mounted version does not require moving the entire arm. Placing the controller in the right position on the steering wheel itself allows it to be used with the thumb without changing the hand's grip on steering wheel.

Sculpting the controller shape saved time in two ways in this example. First, the clay and widget models could be pressed into a hole cut in the steering wheel. This gave the prototype housing a shape that would fit in the cut out. Achieving this fit in a CAD tool would require more time. Second, it was easy for the designer to hold the steering wheel with the

[1] When printed on a Stratasys Dimension Elite at a layer resolution of 0.25 mm.

sculpture in place and test the thumb motion needed to reach the controls.

An interactive prototype of this controller intended to fit a different steering wheel could also be constructed by forming clay on that steering wheel and would likely involve only 30 minutes of sculpting time.

Camera remote. This prototype is a handheld camera remote control. The intended use is to be held in the photographer's hand while they move away from a camera to compose a picture or to be in a picture. The prototype includes a slider under the thumb for controlling zoom and a button under the index finger for taking a picture. It should be usable while looking at the composition or at the camera itself. This means that the interactive widgets should lie under the fingers for easy use without looking at those widgets.

Sculpting around the printed widget models let the designer quickly and accurately place controls under fingers when the device held in the hand. The sculpture required less than 20 minutes to create which was the shortest amount of time across all examples. The final shape was rough but could be refined in either clay or software if testing the functional prototype confirmed that this was a useful application.

A WIDGET SET

Constructing interactive prototypes required a widget set. We need a widget set that keeps the prototype design cycle fast and simple while preserving the feel of interaction with physical objects. This meant using widgets that are simple to assemble, can be printed in any orientation, and which provide good haptic feedback.

After reviewing several widget sets for 3D printed prototypes (see [8, 9, 11] for recent examples) we designed our own widget set based heavily on using existing electronics to convert interaction into electrical signals. Using existing electronics gave up some flexibility in design options but led to simple assembly while providing good haptic feedback. In principle, 3D printable blanks, brackets and slugs as described earlier could be designed and for any of the widgets in [10, 12, 14]. Having a collection of blanks for these widgets on hand would allow the designer to quickly switch between widget types depending on the needs of a project.

The widget set contains a button, a slider, and a knob. Figure 10 shows the widgets together with the printed form used in sculpting. The knob is a potentiometer knob, the slider is a potentiometer slider and the button is a clickable button on a breakout board[2]. The button includes a mounting bracket to hold the breakout board in place, a moving plunger and an enclosure. Choosing a pre-made potentiometer slider limited our designs to straight run sliders with limited lengths. Custom printed sliders would have allowed for curved paths and more flexibility in length. We added additional haptic feedback to the button by placing two ring magnets with polarity mismatched on the button plunger shaft. The magnets are placed in the enclosure after printing. This creates a force that restores the button to its idle position after a button press. We also explored several print-in-place options for providing haptic feedback with the button but the motion varied widely with the print orientation relative to the plane of the 3D printer tray.

All widgets are connected to an Arduino board[3] with Bluetooth. The Arduino board provides power, ground and processes signals from each widget. The Arduino board sends events over Bluetooth to a host for processing.

RELATED WORK

Our work builds on ideas from Switcheroo [1] and Calder [9]. Each of these projects was focus on how to integrate interactive devices with physical shape design. In Switcheroo, passive RF tags could be pinned to a foam surface to place interactive widgets. The Calder project took a similar approach, but involved pinning actual electronic elements to foam. This meant that interactive widgets could be quickly repositioned as the designer explored the feel of the placement of each element. In Calder however, interactive elements protrude from the surface unless the designer creates a cavity in which to embed electronics. This cavity fixes the widget placement and eliminates fluid repositioning. Working with clay, and an oil-based clay in particular, allows the widget to be embedded in the object and to be repositioned as needed. Printing a representation of the widget ensures that elements beneath the surface will fit in the prototype.

Hartmann and Klemmer [6, 8] frame physical design as conversations with the design medium such as clay, paper or foam core, and point to Schoen's reflective practice [13] as a foundation. They were focused on device integration and software and argue that iteration through prototypes is important for designers to understand a problem. Our work shares this foundation but we focus more on integrating the physical shape with the interactive electronics.

The BOXES project merges form and function early in prototyping physical interactive objects [7]. Clever use of cardboard, foil, thumbtacks and masking tape allows a designer to quickly iterate through different shapes and place buttons on those shapes. Our process is a bit less fluid than cutting cardboard, but supports a richer set of shapes and interaction methods beyond buttons. As with some of the other work, the prototype was a mass of tethered wires that was impossible to deploy in a realistic context.

---

[2] We used a 10k Ohm rotary potentiometer knob (Sparkfun part 09939), small 10k linear taper potentiometer slides (11602), and a mini push button switch (97).

[3] A RedBearLab Blend Micro Arduino board.

Savage et al. in Makers' Marks [11] merge form and functionality in the context of 3D scanning and printing. Approximate widget positions and widget types are recovered by using SIFT and RANSAC to recognize fiducial marks printed on stickers. Our process uses only the sculpted object geometry and does not require recovering the sculpture's surface color and does not use image processing. Makers' Marks supports mechanical elements such as hinges. In principle our process could as well by embedding 3D printed representations of hinges in the clay.

deForm [4] combines the expressiveness of clay with digital interaction using a pair of structured light scanners. While the process is limited to 2.5D deformed surfaces rather than arbitrary 3D shapes, the process may support a fluid environment for creating and modifying interactive physical devices.

In DisplayObjects Akaoka et al. [1] also argue for early merger of prototype form and interaction, but take a different approach based on projecting interfaces onto solid objects. This leads to interesting simulations but is not deployable into usable scenarios.

Many projects have constructed sets of interactive widgets for physical computing and prototypes. They differ primarily in the mechanism used to detect interaction. Interaction has been created using sound [12], pneumatics [14], light [10] and of course electricity [15, 4]. Our focus is different however. Our focus is on supporting the creation of shapes to house the interactive elements, and to do so in such that the shape fits human form and motion. In principle, any widget set could be used with our process given appropriate modeling of each widget. Voodoosketch [3] merges physical widgets with sketched interfaces on interactive surfaces but does not contemplate 3D interactive physical devices.

SUMMARY AND FUTURE WORK

Sculpting keeps physicality in prototype design. We presented a fast and simple method for converting a scan of the sculpture into a printable interactive prototype. The method eliminates CAD skills from the process. The algorithms described were implemented in C# to run in the RhinoCommon API in Rhinoceros 5[4].

This method depends on printing 4 cone shaped fiducial markers into models of widgets embedded into the sculpture. Vertices with negative local slope in every direction are labeled as lying on fiducial markers because cone tips have negative local slope in every direction. We group labeled vertices into widget placements by matching the distance between groups of markers against marker locations on each widget. Widget type and orientation are recovered from the groups of 4 markers. Finally, the scanned mesh is modified based on the widget types, locations and orientations. The printed housing can be combined with electronics to make an interactive prototype.

A key factor in saving designer time and effort is that if widget blanks fit in the sculpture then widget electronics and parts will fit in a housing with the same shape and size. This prevents sculpting a shape and placing widgets that later prove to be infeasible for the given shape.

Future advances in 3D scanning and printing technology will significantly improve this prototyping process. Scanning and printing sculpted prototypes require significant time and effort. Preparing hollowed out scans of sculptures in our example required about 30 minutes of focused effort over the course of an hour by an experienced user[5]. The housings in our examples required 8 to 12 hours each to print. The designer is free to do something else while they wait, but this adds time to the prototype iteration cycle.

There is no guarantee that the splitting plane found by our algorithm will result in an opening that allows electronics to be inserted into the housing. The algorithm worked well in all of our examples—even though designers were not told to consider the feasibility of splitting the housing for assembly. Several methods could be explored to ensure that the split allows assembly. The housing could be split with a curved surface that would allow more possibilities. The function for scoring candidate splits consider the size of the largest object that will through an opening rather than just the area of the opening. In some cases it may be necessary to split the housing into more than 2 pieces.

ACKNOWLEDGEMENTS

This work was supported by National Science Foundation grant IIS-1406578. David Brandt designed the widget set used in this paper.

REFERENCES

1. Akaoka, E., Ginn, T., Vertegaal, R., 2010. DisplayObjects: prototyping functional physical interfaces on 3d styrofoam, paper or cardboard models. In: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction. ACM, pp. 49–56.
2. Avrahami, D., Hudson, S. E., 2002. Forming interactivity: a tool for rapid prototyping of physical interactive products. In: Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques. ACM, pp. 141–146.
3. Block, F., Haller, M., Gellerson, H., Gutwin, C., and Billinghurst, M. 2008. VoodooSketch: Extending

[4] See www.rhino3d.com

[5] Using a NextEngine 3D Scanner HD with ScanStudio HD Pro software and netfabb Professional for repairing and hollowing out the mesh.

interactive surfaces with adaptable interface palettes. In Proceedings of the 2nd International Conference on Tangible and Embedded Interaction (TEI'08). ACM. pp. 55-58.

4. Follmer, S., Johnson, M., Adelson, E., and Ishii, H., 2011. deForm: an interactive malleable surface for capturing 2.5D arbitrary objects, tools and touch. In Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11). ACM, New York, NY, USA, pp. 527-536.
5. Greenberg, S., Fitchett, C., 2001. Phidgets: easy development of physical interfaces through physical widgets. In: Proceedings of the 14th annual ACM symposium on User interface software and technology. ACM, pp. 209–218.
6. Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., Gee, J., 2006. Reflective physical prototyping through integrated design, test, and analysis. In: Proceedings of the 19th annual ACM symposium on User interface software and technology. ACM, pp. 299–308.
7. Hudson, S. E., Mankoff, J., 2006. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. In: Proceedings of the 19th annual ACM symposium on User interface software and technology. ACM, pp. 289–298.
8. Klemmer, S. Hartmann, B., and Takayama, L. 2006. How bodies matter: five themes for interaction design. In *Proceedings of the 6th conference on Designing Interactive systems* (DIS '06). ACM, New York, NY, USA, 140-149.
9. Lee, J., Avrahami, D., Hudson, S., Forlizzi, J., Dietz, P. and Leigh, D. "The Calder toolkit: wired and wireless components for rapidly prototyping interactive devices" in Processings of the 5th Conference of Designing Interactive Systems: Processes, Practices, Methods and Techniques. 2004. pp. 167-175.
10. Savage, V., Chang, C. and Hartmann, B. "Sauron: Embedded single-camera sensing of printed physical user interfaces" ACM Symposium on User Interface Software and Technology (UIST '13). 2013.
11. Savage, V., Follmer, S., Li, J., and Hargmann, B. 2015. Makers' Marks: Physical markup for designing and fabricating functional objects. In: Proceedings of the 28th Annual Symposium on User Interface Software and Technology (UIST'15). ACM. pp. 103-108.
12. Savage, V., Head, A., Hartmann, B., Goldman, D. B., Mysore, G., Li, W., 2015. Lamello: Passive acoustic sensing for tangible input components. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 1277–1280.
13. Shön, E. 1984. The Reflective Practitioner: How Professionals Think in Action. Basic Books.
14. Vázquez, M., Brockmeyer, E., Desai, R., Harrison, C., Hudson, S. E., 2015. 3d printing pneumatic device controls with variable activation force capabilities. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 1295–1304.
15. Villar, N., Scott, J., Hodges, S., Hammil, K., Miller, C., 2012. .net gadgeteer: a platform for custom devices. In: Pervasive Computing. Springer, pp. 216–233.