An Accessible CAD Workflow Using Programming of 3D Models and Preview Rendering in A 2.5D Shape Display

Alexa F. Siu
Stanford University
Stanford, CA, USA
afsiu@stanford.edu

Joshua Miele
The Smith-Kettlewell Eye
Research Institute
San Francisco, CA, USA
jam@ski.org

Sean Follmer
Stanford University
Stanford, CA, USA
sfollmer@stanford.edu

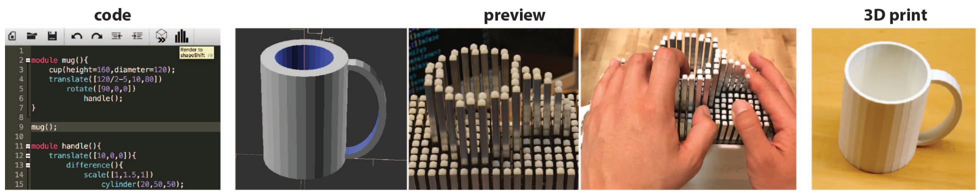

Figure 1. We propose an accessible CAD workflow where 3D models are generated through OpenSCAD, a script-based 3D modeling tool, and rendered at interactive speeds in shapeShift, a 2.5D shape display consisting of a grid of 12×24 actuated pins.

ABSTRACT

Affordable rapid 3D printing technologies have become a key enabler in the maker movement by giving individuals the ability to create physical finished products. However, existing computer-aided design (CAD) tools that allow authoring and editing of 3D models are mostly visually reliant and limit access for people who are blind or visually impaired. We propose an accessible CAD workflow where 3D models are generated through OpenSCAD, a script-based 3D modeling tool, and rendered at interactive speeds in an actuated 2.5D shape display. We report preliminary findings on a case study with one blind user. Based on our observations, we frame design imperatives on interactions that might be important in future accessible CAD systems with tactile output.

ACM Classification Keywords

K.4.2 Social Issues: Assistive technologies

Author Keywords

Visually Impaired; Computer-Aided Design, CAD; 3D Printing; Rapid Prototyping; Tactile Aids, 2.5D Displays

INTRODUCTION

Maker culture and tools have the potential to empower individuals in shaping their world through building, hacking, tinkering, or designing. Affordable rapid 3D printing technologies have become a key enabler in the movement by giving individuals the ability to sketch an idea and create it into a physical finished product. However, current computer-aided design (CAD) tools that allow for authoring and editing of 3D models are mostly visually reliant and limit access for people who are blind or visually impaired. Script-based 3D modeling tools, such as OpenSCAD, have more accessible input methods but no accessible way to view the generated models at interactive rates. One way for blind makers to explore the model they have created through scripting is by 3D printing it, but the turnaround takes a few hours.

With increasing work in the development of shape displays [7, 1, 8] and tactile arrays [2, 4, 9], we believe there is an opportunity for these displays in increasing output accessibility of existing CAD tools. In this paper we explore an accessible CAD workflow where 3D models are generated through OpenSCAD and rendered at interactive speeds in an actuated shape display. We chose text based procedural programming as input for its accessibility and compatibility with existing assistive technology (i.e. screen readers and braille displays). Moreover, we see an opportunity in using the shape display tactile output as not only a means to teach CAD but also to entice young programmers through an accessible programming task. Past work in programming accessibility have often focused on audio feedback [6, 5] and while some have proposed using 3D printers with the appeal of producing something "real", the slow turnaround has typically been a limitation [3].

In the last part of this work, we report preliminary findings on a case study with one blind user. Our goal with the study was to frame design imperatives for interactions and features that accessible CAD systems with tactile output might need to enable. Additionally, we wanted to test if a user might be able to form a mental model of an object through only partial views of the object i.e. with a 2.5D display the user is only able to explore one face of the object at a time.

3D MODELING WORKFLOW

Our tool uses OpenSCAD, a script-based 3D modeling open-source software, to create and generate 3D models. To obtain a quick preview of the generated model, we use shapeShift [7], a 2.5D shape display consisting of a grid of 12×24 actuated

ASSETS 2018, October 22–24, 2018, Galway, Ireland

ACM ISBN 978-1-4503-5650-3/18/10.
http://dx.doi.org/10.1145/3234695.3240996

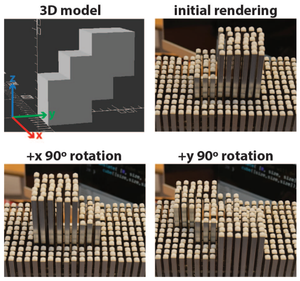

Figure 2. Top: digital rendering of the 3D model (left) and model when first rendered on the display (right). Bottom: the model rendered on the display and rotated 90° around the positive x-axis (left), and around the positive y-axis (right)

square pins (4.8 mm) with an inter-pin spacing of 2 mm. The pins move at an average speed of 70mm/s, thus shapes can be rendered within a second.

We modify the OpenSCAD source code to add an additional menu option, '*Render to shapeShift*'. The user writes code to generate a 3D model in the OpenSCAD text editor and then clicks a button to render the model in the shape display. While OpenSCAD is unitless, the shapeShift application by default assumes units of mm so objects are rendered in a 1:1 scale.

A numeric keypad is used to manipulate the 3D model on the display and there are four possible commands: rotation, translation, scaling, and view resetting. Keys map to discrete increments of 90° for rotation and translation.

EVALUATION

We obtained feedback from one blind user to find what interactions with the system helped the user navigate and understand a 3D model and what interactions were lacking. Our goal was to frame design imperatives on interactions that might be important in future accessible CAD systems with tactile output. Our user was a male experienced blind programmer. One day before presenting the system, the user was given a document containing a high level overview of the system, an introduction to essential functions in OpenSCAD, and examples demonstrating how to create simple shapes. The user was given this information ahead of time in order to allow sufficient time to familiarize with the language syntax. The second day, the user had the opportunity to try out the entire 3D modeling pipeline with the shape display. This session lasted one hour.

The first 5 minutes of the session were focused on the user familiarizing with the shape display rendering limits through exploration of an assortment of shapes (e.g. a pyramid, a car). Next, the user was introduced to the navigation commands. The user was given several sample scripts and encouraged to explore the models and modify the scripts. To verify the user's understanding, two 3D models were rendered and the user was asked (without access to the code) to describe the object and in what plane there might be any unique features, and to explain how might the model have been created. Last, the user was presented with a mug printed in a Makerbot 3D printer (Figure 1) and asked how he might be able to replicate it.

Design Imperatives

Based on our observations throughout our informal study we formulate the following design imperatives:

1. Use Predictable Navigation Commands. After taking a few minutes to freely explore a model, the user was able to accurately describe the different faces of the object. When asked to show where a unique feature was, the user quickly navigated to it. For small features, such as a button in a camera, the user even scaled the model so the feature of interest was shown more clearly. The user referred to this as *"bringing the part into focus"*. When commands are predictable, the user is able to clearly track the state of the object. We found 90° rotation increments were predictable by the user in contrast to very fine resolution of $10-30^\circ$. Overall, this showed us that despite only being able to feel one plane of the object at a time, the user was still able to form an accurate mental model of the object.

2. Provide Feedback about Display Size Limits. Displays don't have infinite workspace, thus rendered models may be clipped and the user may be unaware of this. In one scenario, our user wanted to compare the relative scale of two objects he had explored so he put them in the same script to render at once. One of the objects was four times the size of the other so when initially rendered, the large object resulted in the display's pins moving to their max height. The user commented that from this he understood that one object was clearly much larger but since the pin heights were saturated, he couldn't tell how much more. Because shape and tactile displays will likely always have workspace constraints, systems need to provide feedback in places where models exceed the active workspace. One way to indicate this could be to have the pins vibrate.

3. Initial Rendering of the Model Should Try to Fit The Entire Model Within the Workspace. We noticed over time, the user adopted an exploration strategy. The user would start exploring the model by first scaling it down as much as possible so it all fit within the display boundaries and then start exploring each face and scale up. When asked about this, the user commented that first having a view of the entire model, even if it was very low resolution, helped him get a rough idea of the spatial layout and then as he scaled up he could better understand where the finer details where. Based on this we would suggest optimizing the initial view to show the entire model even if this breaks the 1:1 scale. Other methods should be investigated to allow the user to understand the current scale, and to easily switch between scales.

4. Good Mental Models Allow Low Resolution Representations. For objects the user understood well, he would explore them at a very low scale and commented that even though it was very low resolution, *"once I know what things are, it doesn't matter if they look really square. Now I already know them and don't mind they're very small"*. Having an understanding of how a model is created helps the user form a mental model of the object and compensates for low resolution output. We believe this will be the case often when users are authors, however further work on helping users understand the "gestalt" of the object or labels might be necessary for objects that are complex or that they have not authored.

ACKNOWLEDGMENTS

We thank all the users for their feedback. This work was supported by the NSF GRFP (#DGE-114747).

REFERENCES

1. Sean Follmer, Daniel Leithinger, Alex Olwal, Akimitsu Hogge, and Hiroshi Ishii. 2013. inFORM: dynamic physical affordances and constraints through shape and object actuation. *UIST '13 Proceedings of the 26th annual ACM symposium on User interface software and technology* 13 (2013), 417–426.

2. American Printing House for the Blind. 2018. Introducing Graphiti - A Revolution in Accessing Digital Tactile Graphics. (2018). `https://www.aph.org/graphiti/`

3. Shaun K Kane and Jeffrey P Bigham. 2014. Tracking@ stemxcomet: teaching programming to blind students via 3D printing, crisis management, and twitter. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 247–252.

4. Alexander Russomanno, Sile OâĂŹModhrain, R Brent Gillespie, and Matthew WM Rodger. 2015. Refreshing refreshable braille displays. *IEEE transactions on haptics* 8, 3 (2015), 287–297.

5. Jaime Sánchez and Fernando Aguayo. 2005. Blind learners programming through audio. In *CHI'05 Extended abstracts on Human factors in Computing Systems*. ACM, 1769–1772.

6. Robert M Siegfried. 2006. Visual programming and the blind: the challenge and the opportunity. *ACM SIGCSE Bulletin* 38, 1 (2006), 275–278.

7. Alexa F Siu, Eric J Gonzalez, Shenli Yuan, Jason B Ginsberg, and Sean Follmer. 2018. shapeShift: 2D Spatial Manipulation and Self-Actuation of Tabletop Shape Displays for Tangible and Haptic Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 291.

8. Faisal Taher, John Hardy, Abhijit Karnik, Christian Weichel, Yvonne Jansen, Kasper Hornbæk, and Jason Alexander. 2015. Exploring interactions with physically dynamic bar charts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3237–3246.

9. Kai Zhang and Sean Follmer. 2018. Electrostatic Adhesive Brakes for High Spatial Resolution Refreshable 2.5 D Tactile Shape Displays. In *Haptics Symposium (HAPTICS), 2018 IEEE*. IEEE, 319–326.