**Abstract**—We introduce \([I/O]^3\), the next step in 3D modeling technology. Analogous to what the drawing tablet did for digital illustration, \([I/O]^3\) aims to allow intuitive manipulation of virtual 3D objects within 3D modeling software, as well as display a volumetric image of the virtual 3D objects. User hand gestures control virtual object functions, like creation, translation, rotation, and scaling, as well as control functions, like saving a copy of the virtual model or undoing or redoing a change made on the model. This model will then be displayed on a volumetric display, allowing the user to visually examine their model as if it were a tangible object, without consuming any physical material. The volumetric display is a true, 3D representation of the virtual model utilizing persistence-of- Vision, free of the characteristic distortions and limitations of common stereoscopic 3D displays.

**I. INTRODUCTION**

This document details the progress and goals of Team \([I/O]^3\) as of November 27th, 2013. \([I/O]^3\) is a solution to the existing user interface for virtual three-dimensional (3D) modeling. A user should be able to interact with virtual 3D models with simple gestures, and he or she should be able to view the model from any angle without distortions, as if it were a tangible object. The delay from user input (hand gestures) to system output (virtual model manipulation and volumetric display) should be negligible; the system should operate in real-time. The interface should be simple enough for someone who has experience with molding objects by hand to start using right away, with a minimal learning curve. In this way, while also making 3D modeling quicker and easier for current virtual 3D modelers, \([I/O]^3\) will allow anyone in traditional plastic arts to reap the benefits of virtual 3D modeling, namely being able to save iterations and revisions of a model, and correcting mistakes by using an undo feature instead of starting the entire model over from scratch (as is commonly necessary with traditional plastic arts). No fully-integrated solution to true 3D input and output from a computer exists commercially, but recent technological improvements have paved the way for this solution to become a reality. \([I/O]^3\) is our idea to combine relatively new technologies into an integrated, truly 3D input-output system.

**A. Problems with Current 3D Modeling**

3D modeling has numerous applications in a wide array of fields such as media and entertainment (movies, television, and video games), art and graphic design, architecture, and mechanical and industrial engineering. The question arises: how does one create a 3D model? One approach is to use traditional physical modeling techniques. Although being able to shape the model by hand or with the use of specialized tools gives the sculptor unparalleled control during the creation process, it can be time inefficient, wasteful, and hazardous. Mistakes made to a physical model usually require the sculptor to restart the project from scratch. This wastes time and resources, since models containing mistakes cannot be, or are typically not reused. Physical modeling also tends to use silica-based products, such as clay, as the physical modeling material. Long term exposure to silica dust has been shown to result in silicosis, a lung disease, which may result in death [1].

With the advent of 3D computer graphics technology in the past several decades, it has become possible for virtual 3D objects to be created and visualized. However, although many advances have been made in 3D modeling software paradigms in terms of how models are digitally created and stored on a low-level (from box/spline modeling to virtual sculpting), software input still relies on using a keyboard, mouse, or drawing tablet. It goes without saying that manipulating 3D objects with inherently one- or two-dimensional (1D, 2D) input devices is far from intuitive. Furthermore, all traditional computer graphics workstations rely on using an inherent 2D display to visualize renders of 3D scenes. Traditional 3D modeling software will show a model from a top-view, a front-view, a side-view, and an isometric view all at once, allowing the user to visualize what his or her model looks like from different angles. The user generally will need to rotate their model or the perspective of the virtual environment to view any other angle of the model. Naturally, displaying a 3D object on a 2D display will lose depth characteristics, and introduce visual distortions, like perspective error.

**B. Existing Solutions to Input Interface**

3D modeling is currently performed both by hand, and by a variety of different computerized platforms. Modeling in a computational environment currently has a forced feel, lacking the accuracy of physical modeling by hand. The human hand is incredibly precise. When a user sculpts or models with their hand, they are in the most control. There is no intermediate stage where the actions of the user need to be interpreted in an understandable format. A user can move their hand in three dimensions with a more intuitive nature than that of current computer hardware’s input functionality.

On a desktop computer the user must port their control through the mouse and keyboard, which limits the sculptor’s accuracy, and also has a large learning curve. The user is also limited to the amount of functions the 3D modeling software developers decide to implement. If the graphical user interface of the program is complex and confusing, these functions can be hidden behind folder sub-systems and frustrating irrelevant
button icons. The user must learn keyboard shortcuts and be bound by the 2D functionality of the mouse.

C. Existing Solutions to Output Interface

While 3D display technologies are becoming more prevalent in consumer and professional hardware, most rely on clever optical tricks to result in a pseudo-3D image. The most common technology is stereoscopic 3D, which has many implementations; red-blue 3D glasses, oppositely polarized glasses like those at 3D movie theaters, active-shutter LCD glasses like those that come with modern 3D televisions and computer displays, and autostereoscopic displays like that of the Nintendo 3DS are all examples of stereoscopic 3D displays. The general principle of stereoscopic 3D displays is that two slightly different images of a scene are transmitted to each of the viewer’s eyes. The two images represent the scene as they would be viewed from the left and right eye perspective of a person at the scene. These images are then transmitted to the viewer such that the left eye image is only seen by the viewer’s left eye, and the right eye image is only seen by the viewer’s right eye. This creates an illusion of a 3D image, but true perspective and depth of the scene are completely lost, since the images being transmitted to the viewer are fixed representations of the scene from a single point-of-view. Should the viewer move his head slightly, the image does not change as it would if the viewer was looking at the scene in person [2].

In addition to the lack of a truly 3D perspective, stereoscopic displays have many other issues. Red-blue (or other colored lens implementations) do not faithfully reproduce the captured colors of the scene. Active-shutter LCD glasses are expensive, and require active power to operate. Autostereoscopic displays only allow the viewer to be in a very narrow range in front of the display for the illusion to work, meaning realistically that only one person will see a pseudo-3D image from the display at a time. All of the display types strain the eye, causing discomfort for many people. These display types can be especially dangerous to younger children, whose eyes have not yet fully developed. The only way to faithfully display 3D objects with regards to perspective and depth is to actually display a volume that the eye can focus on naturally; this is the motivation for utilizing a volumetric display [2].

D. Specifications

General specifications for our project are detailed in Tables I and II.

TABLE I
INPUT AND SOFTWARE SPECIFICATIONS

| Gesture Recognition | ≥11 Unique Gestures |
| Gesture Resolution  | 1mm Skeletal Precision |
| Object Functions    | Creation, Selection, Deletion, Translation, Rotation, Scaling |
| Control Functions   | Save, Undo |

II. DESIGN

A. Overview

Our solution can be broken into three major subsystems, with their interactions shown in Figure 1.

Fig. 1. Block Diagram of our Solution

The user’s hand gestures are captured by a motion-tracking sensor, and sent to 3D modeling software, which will be running on a computer. The software will parse the input data and translate any recognized gestures into modeling functions. After obtaining the updated 3D model, the software will display it on the computer in traditional 2D representation, and prepare the data to be sent to the output subsystem by generating cross-sectional images of the 3D model data. The output subsystem consists of a custom-built volumetric sweep device and projector, along with an embedded system for control and synchronization. The sweep device will allow for volumetric images to be displayed, and the projector will be used to project these images onto the swept volume. The projector will receive the cross-sectional images from the software via HDMI video, and the embedded system will be used to synchronize the motion of the sweep device to the real-time cross-sectional image feed from the software to ensure the images are displayed at the correct time.

B. Block 1 - Input Hardware and Corresponding Software Interface

To track hand motions, we are using the Leap Motion sensor. This sensor was chosen for its existing software support, built-in processing, and high finger/hand tracking sensitivity. It recognizes many hand and finger gestures out-of-the-box, like taps, wipes, and circles, as well as keeping track of hand and finger positions in the XYZ coordinate planes. The Leap Motion has a skeletal precision of 1mm, meaning it can detect finger and hand locations to within 1mm [3]. The motivation behind the creation of the Leap Motion was actually to manipulate 3D models, and there already exists in-depth software support for this application [4]. Developed in Java, we are creating a Leap Motion class that interfaces with our custom 3D modeling software.
It is worth noting the Leap Motion’s limitations before describing our gesture list. This will give a better understanding of why the following gestures were chosen, and provide explanation to why some, perhaps more, intuitive gestures were not chosen. The Leap Motion’s field of view is an inverted cone with the tip of the cone at the center of the device. At its maximum field-of-view, the Leap Motion has a 14-inch radius, which can make the user feel cramped when performing certain gestures [3]. Leap Motion also cannot detect multiple fingers when they are lined up perpendicular to the device; the device will count just one finger in this situation.

![Input Hand Gesture Mapping](image)

**Fig. 2.** [I/O]³ Input Hand Gesture Mapping

Utilizing knowledge from courses like ECE242 (Data Structures and Algorithms), we will implement data structures to track coordinates sent from the Leap Motion and analyze them to form intuitive gestures. These hand gestures will be mapped to the functions in our 3D modeling software, giving the user more control over their model than traditional mouse and keyboard modeling applications. Functions to be mapped with hand gestures include object creation, deletion, selection, translation, rotation, scaling, save, load, and undo. Note the following gestures are subject to change as further testing and analytics will result in more intuitive motions. For shape creation, each shape will have a corresponding finger count. For example, creating a sphere will consist of two fingers (one hand) being held out and then tapped down (as if tapping on a table top) for confirmation that the user does indeed want the sphere to appear. When this is recognized, a sphere will spawn in its intended location. Deletion will involve a swipe gesture as if wiping the shape off the screen (and volumetric display). Selection involves holding out the pointer finger and thumb as if pinching an object to pick up. The center of the shape closest to the midpoint of the thumb and pointer finger will become selected. Translation is done by selecting a shape and then using an extended finger to drag it around the plane. Rotation is implemented by holding out a flat palm with all five fingers extended and then moving wrist up and down for x-axis rotation, tilting wrist left or right for z-axis rotation, and shifting the wrist left or right will rotate along the y-axis. Scaling is done by holding out both hands, each with one extended pointer finger. When moving the fingers apart the shape will become larger and when the fingers come closer together the shape becomes smaller. Save will take a screenshot of the user’s plane and can be loaded at a later point in time. This will be implemented by making a clockwise circle with one pointer finger at any given time. Undo will revert back to the state before the last design change was done. This will be done with a counter-clockwise circle using one pointer finger [5].

To test the gesture mapping, we will recruit inexperienced users, teach them the gestures, and let them determine if they feel the gestures are both intuitive and easy to use. How long it takes for new users to adapt to the device will also be analyzed during testing. As for the Java code, unit testing will be implemented along with regular code reviews. The input will be a certain hand gesture, which will be tested against its given output both on the computer’s traditional 2D render of the 3D scene, as well as the volumetric display. A test will be confirmed if the hand gesture is recognized with the proper input coordinates, the correct software functions are called, and the selected shape is created or manipulated appropriately.

**C. Block 2 - Software**

We will be creating our own, very basic 3D modeling software. This software will allow the user to create and manipulate virtual 3D objects, as well as save scenes formed from these objects, not unlike traditional 3D modeling software. While various robust software already exists for 3D modeling such as Maya or 3DS Max, we were unable to find one suitable for our needs. We wanted to maintain the traditional 2D representation of objects as well as have complete control of the rendering of the virtual 3D space because we require data to be output to our volumetric display. No currently existing software does this, and we deemed modifying existing open-source software far too complex. Our software will be coded in Java, using the JOGL Project for the rendering engine, to facilitate cross-platform compatibility [6].

We only expect to have basic modeling structures in place in our software, including the creation of simple shapes (cube, sphere, pyramid, cones, etc.), manipulation of these shapes (scaling, translation, rotation, combination/intersection), and the ability to save and undo progress. All of these functions will be mapped to unique and intuitive hand gestures. We do not expect to have more typical features of fully-fledged 3D modeling software implemented, like texture mapping, modifiable lighting, camera pathing, or any form of animation. We hope to implement the ability to hand-mold 3D objects, if time permits.

Unit testing will be performed for each of the software functions. We will provide the program with input through keyboard presses and mouse actions to determine if each function is working as intended. Once each of the functions is confirmed to work, we will begin integration testing with the Leap Motion. The integration tests will also be done through user input. Completion of the integration tests is confirmed once all of the 3D modeling functions respond appropriately to their designated gestures. Responding appropriately means that the intended 3D modeling function is called and the on-screen objects follow the hand or finger positions of the user.
D. Block 3 - Output Hardware and Corresponding Software Interface

Considering all of the available technologies and techniques detailed in the introduction, we settled on using a custom swept-volume display. The display involves quickly moving a projection surface, and projecting cross-sections of the 3D scene on the surface, such that persistence-of-vision (POV) effects begin to occur. The basic principle behind POV effects is that the human brain, when shown a series of images in quick succession, will blend the series of images into a single image, as if they were all displayed at once. In this way, a POV display can show many different sections of an image sequentially, and as long as they are displayed fast enough, the viewer will see all of the different sections as if they all were displayed simultaneously.

Our concept involves sweeping a planar projection surface such that it sweeps out a rectangular prism. We will be building a device that uses the same mechanical principles of a crankshaft and piston from a combustion engine to convert rotary motion into constant, reciprocating linear motion. A motor will be attached to a crankshaft and, by rotating the crankshaft, a platform attached to the crankshaft by a connecting rod will be pushed up and down along supporting rails. Provided the revolution speed of the motor remains constant, the linear motion of the platform will be constant for the majority of its sweep. A projection surface will be connected to this platform by hollow pipe supports.

Our original design focused on fixing the projector to the bottom platform, such that the distance between the projection surface and the projector remained fixed throughout the entire motion of the sweep device. This fixing of the projector would allow for us to completely bypass any focus error of the projected image over the entire length of the sweep, and would also keep the projected image the same size along the length of the sweep. However, after calculating the maximum acceleration of the platforms, along with the expected difficulty of connecting power and HDMI cables to the projector that are capable of the repetitive flexing and contracting that would be inherent with the constant motion of the projectors, we decided to fix the projector.

The maximum acceleration of the platforms was calculated as follows. The motion of the platforms is assumed to be sinusoidal with frequency 12Hz, with a peak-to-peak amplitude of 3 inches, and the force of gravity is assumed to be 9.81m/s^2:

\[ p(t) = (1.5 \text{ in})(0.0254 \text{ m/in})\cos(2\pi(12\text{Hz})t) \text{ m} \]
\[ = (0.0381\text{m})\cos(24\pi t) \text{ m} \]

\[ v(t) = \frac{d}{dt}\{ p(t) \} \]
\[ = \frac{d}{dt}\{(0.0381\text{m})\cos(24\pi t) \text{ m} \} \]
\[ = -24\pi(0.0381\text{m})\sin(24\pi t) \text{ m/s} \]
\[ = -0.9144\pi \text{m sin}(24\pi t) \text{m/s} \]

\[ a(t) = \frac{d}{dt}\{ v(t) \} \]

\[ = \frac{d}{dt}\{-0.9144\pi \text{m sin}(24\pi t) \text{m/s} \} \]
\[ = -24\pi(-0.9144\pi)\cos(24\pi t) \text{m/s}^2 \]
\[ = -21.9456\pi^2 \cos(24\pi t) \text{m/s}^2 \]

\[\max\{|a(t)|\} \approx 216.59 \text{m/s}^2 \approx 22.08G \]

As discussed later, we decided on the Texas Instruments DLP LightCrafter 3000 pico-projector. While many of Texas Instruments’ DLP products withstand stresses up to 105G without breaking [7], we decided to fix the projector above the bottom platform such that it remains stationary, in order to mitigate risk. In addition to not potentially breaking the projector by excess mechanical shock, we also eliminate the possibility of the projector detaching itself from the moving bottom platform and being launched away from the rest of the sweep device. However, fixing the projector introduces two problems in theory: 1) The projected image will only be in focus at one point during the entire sweep of the projection surface, and 2) the projected image will not remain the same size as the projection surface moves. This is a simple result of the fact that the image projected by the projector increases in size as the distance between the projector and the projection surface increases. As we received the projector ahead of time, we were able to test the focus error, and size discrepancies that result from moving the projection surface without moving the projector in tandem. We discovered that the focus error over the total throw of the mechanical device (which was decided to be 3 inches) is negligible. The change in size is significant, however, it is linear along the diagonal of the projected image, and we plan to account for this in software by scaling the images output to the projector accordingly.

The projection platform is currently a sheet of paper that is sandwiched between two quarter-inch-thick sheets of acrylic. A SolidWorks CAD model of our device as it currently stands is depicted in Figure 3.

As the projection surface moves up and down, the projector will project a planar cross-section of the 3D scene from the software. These cross-sectional images must be synchronized to the motion of the sweeping surface. We plan on controlling the motor such that it rotates at a constant speed, and using either a rotary encoder or a Hall Effect sensor to detect when the surface is at the top or bottom of its travel. A microcontroller will process the data from these sensors, and generate appropriate trigger signals to feed into the projector in order to synchronize the position of the projection surface with the display signal from the computer.

To generate a volumetric image with sufficient quality (based off of our specifications), we decided to use Texas Instruments’ DLP LightCrafter 3000, a pico-projector capable of displaying monochromatic bit planes at a refresh rate of 1,440Hz. To achieve such a high frame rate, the projector takes 24-bit RGB 60Hz HDMI video as an input, where each frame of video is an RGB image, and each pixel is made up of 24 bits of information. The projector takes the entire 24-bit image and decomposes it into 24 separate 1-bit images, which it then displays over the course of 1/60th of a second until the next frame arrives over HDMI. At 60 frames per second of...
HDMI video and 24 bit planes per frame, the projector can therefore display a total of 1,440 bit planes per second. The bit planes will be generated using planar cross sections output by the 3D modeling software. Each bit plane will represent a planar cross section of the 3D space in the software, which will continuously send the bit planes to a parallel data pipe. On the other end of the pipe, a pool of worker threads will be taking the bit planes and encoding them in the 24-bit RGB format required by the projector. A timer event will update the encoded image being displayed to lock-in a 60Hz refresh rate. Concurrent programming is especially important at this stage so as to not cause a bottleneck for the response time of the full system [8].

We originally aimed to rotate the crankshaft at 720RPM, which corresponds to a frequency of 12Hz. This would allow us an image frame-rate of 24FPS, by displaying one full image frame on the upstroke of the platform, and one full image frame on the downstroke. To keep costs down, we aim to display a monochromatic image, with a single bit depth per pixel (meaning the pixel in question will be either on or off). After choosing this projector, however, a tradeoff in our original specifications was made. The chosen projector, which refreshes at 1,440Hz, would only allot us a vertical resolution of 60 voxels, which is below our desired resolution. This maximum vertical resolution can be calculated by dividing the refresh rate of the projector by the desired frame-rate of the display output. A lower desired frame rate may prove to be a better option. For instance, an image frame rate of 12FPS would double the depth resolution to 120 crosssections per image, but we’ll have to consider whether or not the lower image frame rate is smooth enough to allow for satisfactory real-time editing of a 3D model. A more robust projection system would allow for a fullcolor, highresolution volumetric display, but proves to be very expensive (TIs DLP Discovery 4100 evaluation boards begin at $8000 and do not come with a light source, but allow for binary monochrome patterning at 32,552Hz and variable color patterning rates, depending on color depth). We are satisfied that for basic 3D modeling without textures, a binary, monochromatic display is sufficient to display a 3D model with enough detail.

### III. Project Management

Our MDR deliverables are detailed in Table III. We focused on getting the individual input and output components of the project completed for two reasons. The first reason is that in order to design and test an input to output interface, there needs to exist some working prototype to test with and design around. The second reason is that the actual interface seems extremely hard to create and could be very time consuming. Ideally, we would want to be able to focus most of our efforts into creating the interface and that is only possible so long as the other portions of the project have been completed.

### TABLE III

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Approximate Percent Completed (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstrate Scaling and Object Translation</td>
<td>100</td>
</tr>
<tr>
<td>Demonstrate Interaction between Leap Motion and Software</td>
<td>100</td>
</tr>
<tr>
<td>Create Volumetric Sweep Device</td>
<td>75</td>
</tr>
<tr>
<td>Demonstrate Software to Projector Interface</td>
<td>100</td>
</tr>
</tbody>
</table>

As shown in Table III, we completed the most of the MDR deliverables, but had some trouble with fully completing the sweep-device.

The functions for creating, deleting, translating, scaling, and rotating shapes have all been completed. We have decided to allow users to create cubes, spheres, cylinders, and square pyramids. Additionally, though not an MDR deliverable, the software is able to determine which shape the user has selected and highlights the selected shape.

A key feature of our project is the ability to interact with the 3D modeling software with hand motions. For MDR, we were able to demonstrate interaction between the Leap Motion and the software. When the user holds out two hands, each with one finger extended outwards, our software recognizes this and scales the selected shape based off of the change in distance between the two fingers. With one finger, the user is able to move the shape around, demonstrating our translation deliverable. However, our demonstration was more of a proof of concept rather than a final implementation. Currently, the shape reacts immediately when it detects fingers and there is difficulty in placing or scaling the shape as intended. Moving forward, the plan is to refine the Leap Motion to software interface so multiple shapes can be selected and manipulated in a more intuitive and smooth manner.

To create our volumetric display, we need a specialized, high-speed projector; naturally, these cost more than a typical...
pico-projector. One of the only projectors available that allows us to meet our design specifications while remaining within a reasonable budget is the Texas Instruments DLP Lightcrafter 3000. Due to the projector being a costly investment for our team, we decided to wait until we finalized the most probable design for our project to ensure that our chosen projector would still be our best choice. Around the time we were accepted as a Cornell Cup finalist, we completed what we believed to be the final design of our system, and acquired additional funds for the project. We made the purchase and luckily ended up having the projector before we originally thought we would. Even though it was not part of our originally proposed deliverables, we have begun the projector to software interface, and have a proof-of-concept encoding and display program which sends encoded image data to the projector via HDMI. From here, we can begin working on another critical path for our project: the 3D modeling software to projector interface. Since the 3D modeling software has to continuously generate bit planes, the software-projector interface will likely be at most risk of preventing us from meeting our real-time design specification. We will optimize and refine this interface to prevent such performance bottlenecks from occurring.

Unfortunately, due to complications in the initial sweep device design, machine-shop access, and shipping of parts, development of the sweep device took longer than planned. As it currently stands, the sweep device has been built, but is currently without a drive-source. As such, we did not meet our deliverable of having a complete volumetric sweep device built and tested by MDR. The device is currently using linear ball-bearings, but to meet the noise requirement, these may be replaced with sleeve bearings, which are significantly quieter (but more difficult to physically implement). A new projection surface may need to be chosen as well, as the current solution of using a piece of paper sandwiched between two quarter-inch-thick sheets of acrylic does not allow for the projected image to be sufficiently viewed from the sides. The next steps for the device will be to acquire a motor, and attach it to the device. Some iterative prototyping will be done to determine the best way to attach the motor to the device, couple it to the driveshaft, and to power and control it. After settling on a motor and integrating it with the device, the projector will then need to be physically integrated with the sweep device. Our embedded system will then need to be integrated, and tested to ensure it can properly detect the position of the sweep device and properly synchronize the projection of cross-sectional images to the position of the projection surface. We intend to use an embedded system based around the Intel Atom platform, as it is being donated to us by Intel for the Cornell Cup Competition. After all of this is done, a protective enclosure will be built, to both give the device more structural stability, and also to protect the device and users from harm.

Team [I/O]^3 is a very close-knit team. All four of the members have been friends for the majority of their time at the university, and have excelled in group projects with each other in previous courses, such as ECE353 and ECE374. The team meets once a week to update each other on progress, and once a week with our advisor, Professor Goeckel. Kevin is a CSE who feels more at home in the virtual world, rather than the physical world. As a result, he is responsible for learning how to utilize JOGL to design the 3D modeling software. Although he has never used JOGL before, Kevin is able to quickly learn new skills, something that is necessary for a project as ambitious as this one. Tom Finneran is a CSE and a hobbyist with an affinity for creating embedded systems projects, including some experience with making a persistence-of-vision display. Writing software to interface between the computer and the projector was a natural challenge for him to undertake. Christopher Pitoniak is a CSE with a strong passion for software development. He takes great satisfaction in designing and creating new programs that can make everyday tasks more efficient and enjoyable. Therefore, the role of designing and implementing the 3D user input library was the perfect deliverable for him. Shamit Som is the sole EE in the group, and has wide experience in hardware and software alike. His interest and prowess in hands-on work prove useful, if not essential, to the development of the mechanical portion of the project. Additionally, he has experience with computer 3D modeling in different contexts, ranging from CAD (exemplified by the design work done in SolidWorks on the sweep device for this project) to videogame object modeling (using 3DS Max and Milkshape). This experience allows the group to consider different approaches to implementing functions in the 3D modeling software and choose the best approaches accordingly.

IV. Conclusion

As it stands, the major interface between the input hardware and software has been tackled. This interface represented one major critical path in our project, and as such, we focused on this interface for MDR. Since we have this interface out of the way, we can move on to adding functionality in the modeling software, discerning gestures and creating the appropriate structures to map these gestures to functions within the modeling software. This path of adding functionality and testing is expected to be quite steady, with no major difficulties foreseen.

The second, and likely most significant critical path, is the design and construction of the volumetric sweep device. While the majority of it is built, there is still a non-trivial amount of work to complete. The next step is to mount a motor, and couple it to the driveshaft of the assembly. After this, sense electronics to measure and regulate the rotation speed of the assembly must be implemented, and the projector must be mounted. Finally, a final enclosure must be built to provide extra mechanical stability and protection for the user and the device. As before MDR, the unforeseen difficulties lie in designing and constructing the mechanical portions. Shamit intends to work on the mechanical aspect and projector/sweep-device integration of the project full-time during the winter break in order to account for any mechanical difficulties that may crop up during the construction process.

After the projector is mounted, we can begin working on synchronizing pattern displays from the computer to projector with the motion of the sweep device. After this synchroniza-

...
data of our 3D scene to send to the projector, and work on optimizing this data generation such that it is generated fast enough to keep up with the final display refresh rate. At this point, the project will be fully integrated.

**APPENDIX**

**CURRENT TEAM BUDGET**

Table IV outlines the current status of the Team [I/O]$^3$’s budget. The total figures estimate shipping costs, and include the extra funding from the Cornell Cup (total budget is $2000).

<table>
<thead>
<tr>
<th>Item</th>
<th>Rounded Price, in $ (incl. shipping)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leap Motion</td>
<td>87.98</td>
</tr>
<tr>
<td>DLP LightCrafter 3000, Cables</td>
<td>611</td>
</tr>
<tr>
<td>Sweep-Device Related Parts</td>
<td>540</td>
</tr>
<tr>
<td><strong>Total Spent</strong></td>
<td><strong>1238.98</strong></td>
</tr>
<tr>
<td><strong>Budget Remaining</strong></td>
<td><strong>761.02</strong></td>
</tr>
</tbody>
</table>

**ACKNOWLEDGMENT**

Team [I/O]$^3$ would like to express their gratitude to David Elentukh and Rob Tremblay, seniors in Mechanical Engineering at UMass Amherst, as well as Chuck B. Malloch, PhD, for their guidance, advice, and help in designing the volumetric sweeping device. The team would also like to thank Rick Wynn and Miles Eastman, who run the Innovation Student Machine Shop, for their advice and pointers in machining parts for the sweep device, as well as for allowing the team to use various metal stock to create the device from. Team [I/O]$^3$ would also like to thank Intel and Cornell University for their donations to our project, including an Intel Embedded System in addition to a monetary donation.

**REFERENCES**


