Case Study: LinkedPIXEL Wins Intel® Perceptual Computing Challenge with Gesture-based Drawing Application

By Karen Marcus

Matt Pilz, owner and sole developer at LinkedPIXEL, was one of the grand prize recipients in Phase 1 of the Intel® Perceptual Computing Challenge. His application, Magic Doodle Pad, enables users of all ages and abilities to create two-dimensional artwork without touching anything physical.

Pilz taught himself perceptual computing and how to use the Intel Perceptual Computing software development kit (SDK) while creating his application for the Challenge. Although Magic Doodle Pad does include mouse input as a backup method, the primary input method for the application is hand gestures, which Pilz kept as simple as possible.

Although he faced some challenges in design as well as a changing SDK, the process gave Pilz a solid understanding of the success factors needed when developing perceptual computing applications.

The Product: Magic Doodle Pad

Magic Doodle Pad (see Figure 1) was inspired by a previous application that Pilz developed, called Scribblify, which was a doodle application, as well. However, says Pilz, "Magic Doodle Pad was an entirely new application developed specifically for the Challenge and the perceptual camera." After reviewing information about perceptual computing, he reasoned that such an artistic application would work well with camera gestures and that it would be enjoyable for users to just sit back and draw without having to think too much about it.

During development, Pilz didn't consider a possible target market for his application other than wanting it to be accessible for different age groups and demographics. Rather, he considered the features he wanted to include. He says, "I knew that this was going to be a casual application, not a full, high-end illustration tool set. I obviously had a limited timeline, and the features were reflective of that."

Pilz reveals that when he started the Challenge, he knew nothing about the Intel Perceptual Computing SDK but knew that it seemed to be a trend. He says, "With Microsoft and Kinect* and all of these different gesture-based devices coming out, I figured it was something to look in to and was excited to give it a shot."

Figure 1. *Magic Doodle Pad title screen*

Development Using the Intel Perceptual Computing SDK

To learn about the Intel Perceptual Computing SDK, Pilz used a number of resources. He says, "My first resource was the Intel SDK documentation; I read through it and looked at the examples and code snippets that Intel provided. One particular item I used was the Intel snippet on GitHub. There I found several good, basic demonstrations of how to interact with the various sensors of the camera. So I was able to look at those and analyze them. After that, I was able to make the connection and come up with some more advanced functionality."

In addition, Pilz used the Intel developer support forums. He says, "They have a perceptual computing subforum, and it has a lot of helpful people, not just Intel staff members but other members of the community who can share their thoughts and experiences using the SDK."

Pilz has good things to say about the evolution of the SDK: “I think Intel has done a phenomenal job of improving the SDK since I used it during the beta version. Back then, it was just a simple PDF with few pages of information. Now, with the latest SDK that I downloaded recently, the core documentation is over 700 pages long, so it’s much easier to quickly review different components of the SDK.”

Deciding Which Elements to Include

Time constraints limited how many different elements Pilz could include, and he made decisions based on functionality and ease of use. Voice recognition was one element Pilz considered. He thought it could be used to switch colors or brushes without users having to move their hands way over to the side of the screen. However, he says, “With the beta SDK, it was not easy to implement, and the documentation was still incomplete, so I scratched that idea.”

Hand Gestures

Pilz knew from the beginning that hand and finger gesture recognition would be key elements, because, he says, “I wanted to create a user experience that would emulate physical paintbrush techniques. Enabling users to just wave their hands and move their fingers around in front of a camera to create art was the number one priority” (see Figure 2).

Figure 2. *Magic Doodle Pad canvas*

For the hand gestures and hand recognition, Pilz's objective was to make them as natural to users as possible. He explains: "I started mapping what gestures would correspond to what features of the app by picking up an actual brush and experimenting to see how the hand is held when users are drawing and what it should be like when they're not drawing. I programmed in the gestures so that when users extend a finger, it acts like the paintbrush. While the finger is extended or if the hand is closed as if holding on to a paintbrush, it begins drawing. As soon as users have their hands wide open, it's as if they dropped the brush, and they can navigate the menus without drawing. My goal was to create the most natural-feeling gestures possible."

Pilz conducted limited user testing, which was successful in seeing how different hand sizes and gesture styles were implemented by a variety of users.

Magic Doodle Pad incorporates several specific gestures. Pilz describes them: "For the main menu navigation and to activate options, the main hand gesture is an open hand, so a palm facing the

camera. When the app detects that the hand is open, cross-hairs appear on the screen to allow users to track where their hand is in relation to the screen. When they hover over any menu item, they see a little timer countdown. If they wait 1–2 seconds, then it activates that function. Another gesture is to enlarge or dismiss thumbnails when looking at previously created art, users flip open their hand or clap the hand back into itself; that brings the images out, and then shrinks them back" (see Figure 3).

Open Palm (Facing Camera)

- Move on-screen cross-hairs or cursor
- Stop drawing to screen if in draw mode
- Quickly snap fingers shut to close the enlarged preview image (when in Gallery view)
- Hover over a menu item until the circle indicator graphic fills to execute the action

Pointer Finger (Facing Camera)

- Start drawing on the screen if in draw mode.
- Alternatively, close all fingers as if holding a brush (this may be less accurate).

Closed Palm (Facing Camera)

- Alternate method of drawing on the screen when in draw mode.
- Quickly flick fingers open to enlarge the preview image (when in Gallery view).

Figure 3. *Magic Doodle Pad gestures*

With the Intel Perceptual Computing SDK, Pilz notes, each hand could be assigned a separate object, with its own tracking information and data that could be pulled up easily. He compared one object to the next to get the information about each hand and, from there, reused most of the code with both hands. He says, "I believe the SDK classifies hands as primary and secondary, with affiliated labels. It was just a matter of creating two objects for each one, and then tracking the information for each."

Pilz also notes that some logic is built in to the SDK. He says, "I made it so users could track both hands at once on the screen, and then swipe up to two different brushes and two different colors for each hand. The SDK made that pretty straightforward. It's just a matter of having two objects instead of one and doing conditional checks to determine which hand is performing what action."

For this application, no manual keyboard functionality is included. Pilz explains, "The entire interface was designed around the gesture concepts using the camera. So I did add basic mouse support to achieve the same objective, but the primary means of input is designed to be gestures."

The application continuously checks for hand gestures and tracks the user's hand. If the hand moves out of camera range, the application checks to see if it has come back into view. When it is in view, gesture-based controls take precedence. However, says Pilz, "If the hand is still in view and the mouse moves, then the application hides the cross-hairs and assumes that the user is now using the mouse as the primary input method. This is the case until the gesture sensor detects another movement, and then it switches back to gesture mode. At any given time, either the mouse or the gesture controls are active. I designed it to not have any negative impact based on which input method is used."

Depth Sensor

Pilz also experimented with the depth sensor of the camera. He says, "My idea with that was to make the brush size change depending on how close or far away from the camera the user's hand was. It worked, but I knew a lot of different users with different setups and different distances from the camera would be using it. It wasn't a technical issue; it was more that in that time frame, I

just couldn't seem to pinpoint the specific scale ratio to make the brush usable by the greatest number of users. So I decided not to keep that as an added feature."

Design Challenges and Achievements

Pilz's primary design challenge was with sensitivity levels in determining how open or closed a hand had to be to constitute the corresponding hand gesture. In early testing, at times users tried to draw, but the system would detect their hand as being open and wouldn't let them. He observes, "How the user was situated or the size of his or her hand affected how the application interpreted how open or closed the hand was. I had to do some fine-tuning with the sensitivity levels to determine when the hand is open and when it's closed."

Another concern was the fact that the depth and gesture sensors on the camera are 320 x 240 resolution. The application was 1024 x 768 or higher, so there was less pixel data to work with than the ideal. Pilz notes, "Once the user's hand moves toward the edge of the screen, the application would sometimes have issues trying to pick up what was going on. I solved that by hiding the cross-hairs if the hand is outside of the sensor range, assuming that the user is now using the mouse instead of hand gestures."

Pilz also had to find a way to manage simultaneous gestures from both hands. He says, "The secondary hand was added during the final stages of the development process. At first, I just created all the functionality and made it work with one gesture. Really, the only time the application uses the simultaneous hand gestures is during drawing."

Another design challenge was trying to come up with an interface that uses gesture controls. Pilz comments, "This is an entirely new type of input because you can't design interfaces as you would for a traditional mouse. To make menu options selectable only when the user really wants to select them versus accidentally as soon as they brush their hand over a certain item—that was one of the design considerations for which I struggled to come up with a good solution."

To resolve this issue, Pilz decided not to have a button toggle as soon as the user's hand hovered over it. He says, "I realized that would be a terrible user experience decision because, for example, if they're drawing and they move their hand over the exit button, suddenly it would exit the drawing. I was inspired by Nintendo Wii* and Kinect games, where they ask you to move your hand in to a specific location for a few seconds, just to verify that that's where you really mean to go. In my mind, that was the best solution at the time: As soon as the user hovers over an item, a little circle appears that starts to fill in. If they don't want to confirm that selection, they can move their hand away and the circle disappears. But if they stay there until the circle fills up, that option is selected."

In terms of achievements, the finger tracking function worked particularly well. Pilz notes, "With this app, I didn't need to get so specific as to track each individual finger. The SDK has a property of the geo-node object called `LABEL_HAND_FINGERTIP` that simply finds whatever fingertip is the

furthest out, and then it starts tracking that one. This made it easy to track an individual finger without having to concern myself with each individual fingertip."

Other Challenges

Pilz had some additional challenges with the SDK itself. He observes, "When I started development, the SDK was only on its second beta, so at that point several elements were not yet available." Another challenge was having the voice recognition component downloadable separately from the SDK. The Dragon software used for voice recognition was about 600 MB. Pilz says, "The third beta had a web-based downloader, so every time I wanted to install the SDK on a different computer for testing, I'd have to re-download the entire SDK instead of having an offline downloader."

Development Experience

Perceptual computing development and using the Intel Perceptual Computing SDK was a learning experience for Pilz and one that resulted in new innovations.

Lessons Learned

The biggest lesson Pilz learned during the development process was always to check for SDK updates. "This relates to the inherent risks of developing using beta products," he says. "With the SDK changing, I had to go in and revise my code to reflect the changes. That's just the nature of beta software, and it's important to try to find a change log to know exactly what's changed from version to version." Keeping an eye on these changes is important for developers to ensure that the application continues to work when a change happens that requires modifications.

Another lesson was the importance of becoming familiar with all the resources, documentation, code samples, and the Intel support forum. Pilz notes, "Those are the sources I stuck with to answer all the questions I had during this development process."

Pilz found programming for perceptual computing easier to learn than he expected. He says, "Some developers may be reluctant to really embrace perceptual computing because of how new and unfamiliar that concept is. That's the boat I was in when I first started; it seemed complex to me, because you're dealing with hand gestures and these different input mechanisms. All I can say is, it's not bad. In less than a week, I had a solid understanding of everything that I needed to do and even had a good chunk of my application done using these new perceptual concepts. By sticking to the SDK and reading the documentation, I would tell people not to be worried. It's a pretty straightforward process once you get in. In fact, I found it easier to use the perceptual computing SDK than some other SDKs I've used in the past."

Pilz also encourages other developers to think abstractly when coming up with perceptual applications: "You might have a menu at the top that works well with the mouse, but you should ask yourself if there are better ways to handle that through just gestures. Try to devise new and more natural-feeling ways to achieve the same results. I'm not saying that perceptual computing will ever replace traditional keyboard input, but it definitely has its benefits over mouse input for different applications."

Innovations

Pilz considers this application a success—if for no other reason than he developed it so it could be used without a traditional input device or even a touchscreen. He acknowledges that the user interface that enables users to hover over different menu items to confirm the selection isn't necessarily innovative, but he says, "To me, that's an innovative user experience approach, to allow them to navigate between all the menu items and use all of the features without ever having to use the mouse."

In addition, the gestures Pilz used for the image gallery were new, as users can just pop open the image they're hovering over by flipping their finger outwards (see Figure 4).

Figure 4. *Magic Doodle Pad gallery with the Exit button at the top*

Using the Intel Perceptual Computing SDK

Pilz appreciated the fact that the SDK included prebuilt libraries compatible with most of the mainstream frameworks, including Unity*, Processing, and openFrameworks*, as well as the raw C++ libraries. Pilz says, "By providing prebuilt libraries for these different programming environments, Intel made it that much easier to quickly get started with the SDK instead of having to reinvent the wheel by incorporating it by hand into some framework. It was easy to incorporate the SDK into Processing thanks to the work that was done to make it accessible through these different platforms."

With respect to using the Processing framework to develop Magic Doodle Pad, Pilz says, "I know a lot of entrants used Unity, but because I wasn't making a game, I went with Processing, which is built on Java* technology, and that made it easy to handle image manipulation and so forth, which I used throughout the application. GitHub libraries available through Intel for the Processing

framework made it easy for me to download and dive in without having to spend significant amounts of time reading what had already been done. After reviewing the documentation and some of the code snippets, it only took me about an hour to have my first perceptual prototype done."

"Another impressive feature," adds Pilz, "is how simple it is to track hands and fingers. You can retrieve all that data basically in one line of code to track a hand, and then another line of code to track an individual finger. It allowed me to focus more on the heart of the application rather than trying to deal with the technicalities."

Pilz also benefitted from the extensive documentation that Intel provided for the SDK as well as code snippets. "Intel actually has several different repositories online where many different code snippets are available for all of these different platforms," says Pilz. "I give Intel a lot of credit for that, because I've used other SDKs in the past, and it was much more cumbersome to get started, because the documentation was lacking."

Future Plans

With the Challenge over, Pilz continues to improve Magic Doodle Pad. He notes that, in particular, voice recognition could make the application more user-friendly: "It's a lot easier to say, 'Color blue,' or 'Brush seven,' or 'Save drawing' than it is to get your hand to the corner of the screen to access those different menu interfaces."

Another feature that could be incorporated is different gestures for each individual finger. Pilz explains, "For a painting app, it might be fun to have a different brush, a different color, or even 10 of the same brush. It would be more like finger paints at that point." Pilz would still like to incorporate the depth sensor to make the brush scale depending on how close the user is to the camera.

A three-dimensional (3D) component is another possibility. Pilz has experimented a bit with this mode, using the Oculus VR Rift*, a virtual reality headset. Adding this component to Magic Doodle Pad would enable users to essentially sculpt a 3D object. Pilz says, "I want to think that you could get much finer detail by being able to move your hands around and rotate the object in midair without having to use any controls. I believe that a higher level of detail could be achieved with that down the road. I've already seen 3D modeling applications that supported hand gestures and found them quite captivating." He adds, "I imagine if you can get to the point where you can control everything by waving your hand in front of a sensor camera, suddenly you're entirely immersed in whatever project you're working on, so it is an exciting prospect."

Pilz sees possibilities for other creative applications. He says, "It's an organic approach. Traditionally, you'll have a paintbrush in your hand. And now, you can actually simply pick up a pencil or something and hold it in midair and draw while watching the screen. I think there's a level of immersiveness with perceptual computing—to have that free experience of just waving your hands without having to touch any physical object and to be creating. We are entering what

is to me like a science fiction film, where you see all these people who make all these hand gestures and achieve great things. It's getting to that point, I think, with perceptual computing."

Having worked in the medical industry for a few years, Pilz sees applications for perceptual computing there as well: "I see this as having huge potential in the medical industry, where there might be physicians who are seeing patients and have gloves on and can't really interact with a physical device for contamination reasons. But if a program with medical records comes along that supports gesture recognition, I can see them just being able to make hand gestures without having to interfere with anything else or take off gloves to pull up different medical records. To me, that's one venue that I think has a lot of potential for perceptual cameras."

LinkedPIXEL

For several years, Pilz was a web developer for a health facility. In early 2011, he formally established LinkedPIXEL as a business; he now focuses almost exclusively on app development using different platforms. He says, "I've done a lot of Apple iOS* and Google Android* development, and I've now done some that have been released through Intel AppUp®. I've even experimented with developing apps for niche markets like the Panasonic VIERA* Connect TV."

Pilz wants to stay on top of such innovations and "hopefully come up with the next big thing." In particular, he plans to continue developing unique applications for current and next-generation platforms. He has also become interested in Intel's recently unveiled HTML5 development center, which includes many tools and resources for developing web-based apps that are cross-platform compatible. Pilz may also contribute to one or more of several ongoing Intel competitions relating to perceptual computing and application innovation.

To find out more about LinkedPIXEL go to linkedpixel.com.

About the Author

Karen Marcus, M.A., is an award-winning technology marketing writer who has 16 years of experience. She has developed case studies, brochures, white papers, data sheets, solution briefs, articles, web site copy, video scripts, and other documents for such companies as Intel, IBM, Samsung, HP, Amazon Web Services, Amazon Webstore, Microsoft, and EMC. Karen is familiar with a variety of current technologies, including cloud computing, IT outsourcing, enterprise computing, operating systems, application development, digital signage, and personal computing.