Kinect Programming

Andrew Hassan

September 27, 2012

Contents

Introduction

For the "Terminator Hand" project, we will be using the Microsoft Kinect to allow the robotic hand to mimic users' hand movements. This is done by sampling three-dimensional image data from the Kinect and using it to move mechanical parts in the robotic hand. To do this, we must have a way to effectively interface with the Kinect to capture its data.

There are two main libraries that are used to provide this functionality: OpenKinect (OK) and Microsoft's Kinect SDK (MSSDK).

While both of the libraries share a core set of features, there are certain aspects that differ between them that may make one more suitable for this project. This report will outline some of the aspects that can be considered, and will make a conclusion about which is better for this project.

Platform Support

An important characteristic to look at is platform support. Right now, the requirements for this project aren't well defined; this means that the computer using these libraries may have to interface with hardware at some point, and that could be platform-dependent. Because of this, it's vital that we consider the various platforms that the libraries run on.

Operating System Support

In terms of operating system support, OK is more flexible than MSSDK. OK supports Windows, Linux, and Mac whereas MSSDK supports only Windows 7.

Programming Language Support

Both OK and MSSDK support "high-level" and "low-level" APIs. Low-level functions provide more direct access to the hardware while the high-level calls provide more abstract data from the Kinect.

OK has high-level API wrappers for most common languages such as Python, C++, C#, Java, JavaScript, and others. MSSDK has high-level support only for C# and Visual Basic.

OK's low-level calls are done in C, while MSSDKs low-level calls are done with C++.

Obviously with OK, there are a lot more options for high-level functions, which are probably more useful than low-level calls for this project.

Hardware Support

In terms of hardware, there are two versions of the Kinect: the standard XBOX version and a "Kinect for Windows" version. The limitation with choosing one version over the other however

is that OK supports only the regular Kinect while MSSDK supports both versions. According to Microsoft, the Kinect for Windows is fully supported and should be used in Kinect development.

Being supported by both libraries and being cheaper (about $100 as opposed to $150[1] for the Kinect for Windows), it seems that the normal Kinect would be the more optimal choice. However, upon researching the specifications of the two, the Kinect for Windows has a higher resolution and slightly better hardware. This may be worth the extra $50, especially if it will be used to track smaller objects such as hands.

Community Support and Documentation

In terms of community support, both MSSDK and OK have public forums and help topics available; there seems to be an abundance of help for both libraries. In terms of documentation, however, there are slight differences between them.

MSSDK's documentation is hosted on Microsoft's Developer Network (MSDN) site. There are overviews, API reference documents, a programming guide, and samples.

Similarly, OK's documentation has the same categories. But because of the numerous supported wrappers, the quality of information for each seems to be lower than MSDN. That doesn't mean that it is bad though. Simply reading the OK API reference and then looking at the samples for each language, it isn't difficult to see how to make each call.

Functionality

To judge the two APIs, the degree of control over the Kinect's inputs and outputs must be compared. The Kinect has three main sources of input: an IR depth sensor, a colour sensor, and a microphone array. It also has three main sources of output: a tilt motor, an LED and an IR emitter.

API Calls

As stated before, both MSSDK and OK have high- and low-level APIs. However, in the case of this project, the high-level APIs may be of more value to us, as it doesn't require us to write tedious low-level code by hand.

The high-level APIs in MSSDK are a lot more feature-complete than those available in OK. MSSDK has C# libraries for all the inputs and outputs, whereas in OK, C code would have to be written for certain functions.

[1] $150 after a $100 student discount

Conclusions

There are many things in common with OK and MSSDK; for essentially everything except platform options, MSSDK is either on par or better than OK. However, there are a few things that set them apart such as the hardware limitations of OK.

Recommendations

In general, it seems that MSSDK is more complete in terms of features and support for the hardware. It also has better documentation than OK. And it includes more in-depth samples for both high- and low-level programs. It also has better high-level APIs. And I believe that despite its limitations in platform options, it is the better choice for development.

In terms of actual hardware, this means that we have the option of getting the XBOX Kinect or the Kinect for Windows. I would suggest getting the XBOX Kinect because it is cheaper, more available, and works with OK in case we have to switch libraries for any reason.

Glossary

- MSSDK — Microsoft's Kinect SDK
- OK — The OpenKinect library