Meaningful Touch and Gestural Interactions with Simulations
Interfacing via the Dart Programming Language

T. H. McMullen and K. A. Hawick
Department of Computer Science, University of Hull, Cottingham Road, Hull HU6 7RX, UK.
Email: t.h.mcmullen@hull.ac.nz; k.a.hawick@hull.ac.uk,
Tel: +44 01482 465181 Fax: +44 01482 466666
June 2014

ABSTRACT

Interactive technologies are improving the way in which we are able to communicate with devices. The rise in availability of products, such as the Leap Motion, Kinect and touch screen devices, means that we are able to program applications that significantly increase the potential input from human users. These devices are able to be programmed to work together allowing for situations where one device may improve the functionality of another. In this paper we discuss use of the Leap Motion and touch screens in our research into interactive simulations. We use the Dart programming language as a suitable vehicle to integrate together interface components to these two different technologies. We show how combinations of these devices can lead to new and meaningful ways to communicate with applications over and beyond the capabilities of conventional input methods for simulations.

KEY WORDS
Dart; HCI; touch-screens; LEAP; gestural interaction.

1 Introduction

Interactive simulation continues to be an important tool for exploring highly complex models and systems that are not easily explored with analytical methods. Exploring the parameter space and emergent behaviours of complex system models can benefit enormously from a rich set of user interaction capabilities.

Human-Computer Interaction (HCI) [8, 15, 32] is a relatively mature research field [9] but is one that continually benefits from the emergence of new interaction technologies [1, 4, 7, 34]. Devices that support gestures and human touch interaction are not new in principle [2] but devices such as the Leap Motion controller and desktop and larger touch sensitive screens have become commodity priced and widely available recently. Such widespread availability naturally opens up possibilities for various programs and applications that might not otherwise justify custom use of expensive domain specific tools.

Figure 1 shows ways in which a group of collaborators might interact using a large sized touch screen driven by a desktop computer.

In this article we explore the hybrid use of gestural recognition devices such as the Leap Motion controller alongside desktop (27 inch) and larger (65 inch) touch sensitive displays. We investigate how these can be used to support interaction with simulation software for studying complex systems and models.

We describe our implementation experiments in interacting with procedurally generated 3D landscapes [21]. This is an important area for interactive simulation and makes use of heavy duty computational graphics performance [18] on web platforms [22]. The touch and gestural devices allow a user to explore and navigate through such generated scenes, as well as having the potential to allow parameter adjustment and regeneration in interactive time.

Google’s Dart programming language [14, 31, 37] is a convenient platform independent [19] language and environment for this work. Dart supports device libraries [6] for interaction with both of these interaction paradigms and these devices. Work has been reported elsewhere on Dart’s
use as a simulation programming language [20] and on its library [10], interfacing [16] and performance capabilities [23, 33].

Gestural computing [5] is not a new field, but the availability of commodity devices and the software interfaces that allow their rapid uptake in applications, such as navigating bulk image sequence data [26], opens up many new aspects of HCI and also reasons to consider some of the long standing rules and conventions of HCI [24]. In particular, touch control that has recently been largely dominated by mobile devices such as tables and mobile phones [27], and gesture control [29] that is of a high enough quality and accuracy [38] to be a relied upon interaction mode [36] may give a new lease of life into the desktop computing market, that has been widely discussed as a a potentially declining one [11].

Touch computing in the context of tablet-sized displays has already been explored as a platform and vehicle for interactive simulation models [28]. Desktop devices with a similar interaction capability means that high performance processing power can be brought to bear on the field of interactive simulations.

The Leap Motion gestural controller [12] is a commodity-priced device that plugs into a conventional desktop via a USB interface. It detects hand and finger gestures in the air space immediately above it and can recognize non-touching gestures at hand and finger resolution levels. The Leap Motion has successfully been employed for a number of applications including: data navigation [3]; medical rehabilitation [13] art and sculpting [17]; speech-impaired communication via sign language [25]; and augmented reality systems [30].

The new influx of devices designed to aid with improving human computer interaction (HCI) has changed the way in which we are able to interact with an application or simulation. Through the use of various new and emerging technologies multi-touch devices are becoming commonplace, ranging in size and application. Recently the Leap Motion came to market, allowing for hand tracking in 3D space. This leads to a new area for exploration in HCI, along with the idea of combining these technologies. To create this we look at using Dart, a new programming language designed for web applications. Dart allows us to interface with multiple devices, across different platforms.

Figure 2 shows the set up we used for testing the touch input using a more conventionally sized (27 inch) Dell touch monitor along with the Leap Motion gestural device.

The rise in touch screen enabled devices has helped change how we interact with various pieces of technologies. Small devices such as smart phones and tablets have been greatly improved by the use of touch screens. Larger screens, such as TVs or monitors, are now able to be interacted with in a similar manner as to that of our smart phones. Though these larger devices are capable of such feats, very few applications are designed with that in mind. We look at the use of large multi user touch screens and how to optimize some simulations to take advantage of these new features.

Newer devices such as the Leap Motion, have allowed for more unique interactive communication with applications. The Leap Motion is a USB peripheral which is used to find out information about a user’s hand in 3D space. This data includes finger information, location, and rotations, amongst others. It is with this that we are able to experiment with new methods of interacting with simulations. These devices and approaches will continue to become more popular since they are becoming commonly embedded into other hardware, such as laptops and keyboards, and so are expected to become somewhat of an expected standard.

This research looks at the ability to use both touch screens along with the Leap Motion to create new methods to interact with simulations. To do this we must first look at each one’s strengths and weaknesses. The touch screen offers users an easy way to interact with devices. It is accurate in the data passed, and simple to design for in many applications. The Leap Motion is also easy to use, but is not as well established as touch screens. Using the Leap Motion we are able to track more than just a point on a screen, but instead a hand in 3D space. The downside of the Leap Motion is that it requires giving the user constant feedback as to the position of their hand and how that is relevant to the current screen.

In Section 2 we describe our approach to using Dart software to integrate simulation code, touch screen interaction code and Leap device interaction code with some implementation details given in Section 3. We present some selected results in Section 4. We discuss our findings and offer some conclusions and suggested areas for further work in Section 5.
2 Method

Designing for multiple inputs to work simultaneously can prove to be challenging, because of this an application needed to be created. To create this application for testing, we looked at a range of languages which support both the Leap Motion and touch screens, and decided to work using Dart. For the touch screen input this is built into Dart, while the Leap Motion is not so common needing its own specialized API for Dart. Using both these inputs can be designed to work using just one hand, leaving the other free.

For an application to be created on these devices a suitable programming language needs be selected. In this situation we chose to use Dart, as it is easy to program and can be run on a range of devices. Code written in Dart also has the advantage of being able to be converted to JavaScript, to increase the range of platforms which it can be run on. Dart’s performance is also something which needed to be looked at as we require a responsive application, which does not leave the user waiting. The performance aspect also played a role in the design of the application, as to maximize the user’s experience while minimizing the effect on the simulation. The resulting application had two separate loops, one for logic and another for rendering. The logic would occur at set intervals while the rendering would be as fast as possible. The input was taken into account during the logic step so that if rendering lagged behind the inputs would still be recorded.

With Dart, using the touch screen becomes a simple method of using an event listener like that for a keyboard or mouse. Using multi touch becomes a bit more difficult, as we need to know more than a simple input such as a key press. To manage the multi touch input Dart creates a listener which stores an array for each touch that is encountered. The problems which can occur with multi touch is that on larger screens it becomes necessary to differentiate between multiple users. To account for multiple users we take the coordinates and based on how close they are together treat them differently. Another aspect which we looked at was that of what these interactions meant, and how each one would interact with the simulation. For testing the touch screen interactions we implemented several known gestures which are discussed later on in this paper. These include: scaling; rotation; and movement within the generated simulation.

Creating an input for use with the Leap Motion proved to be a little more difficult than that of a simple touch screen. The Leap Motion requires the use of external drivers, which there is a library for. The drivers can be easily installed and implemented to work in a similar manner to that of other input devices. The Leap Motion can then be used by creating a new controller which implements a listener. The listener in turn requires being set up as to correctly interpret user input. With the listener we are able to take data for various input such as finger counts, angles and rotation of the hand relative to the Leap Motion, along with some gestures built in. To explore this we ran some simple tests in which the output from the Leap Motion was used to directly manipulate an object in three dimensional space within a simulation. These test are covered in more detail further on in this paper.

When designing an application to have meaningful inputs from both a touch screen and Leap Motion, we need to look at what use each part fulfils. In our case we have created a 3D simulation in which the user is able to interact with by using the touch screen in conjunction with the Leap Motion. Each part of the inputs were compared to find what they would be most suited for. In our situation we use the Leap Motion to control the camera and how we view the scene. The touch screen was used to aid in creating more accurate control over how we move through the scene as well as manipulating an object within. By using these together we are able to create a user experience when one is able to use one hand to move through and control 3D space. To allow for these to work together we created a camera class which is able to directly take data from the Leap Motion and, in turn, use that to change camera angles. The touch input was then passed and used based on the new view angle of the scene.

3 Implementation

Implementing a software system that integrated both the touch screen paradigm and the Leap Motion gestural paradigm was accomplished using the Dart programming language. To implement both of these we needed to look at how they interact with each other, and how to maximize their usefulness. Each device had different uses so to take advantage of the full potential we needed to design our application to use what was best for a set task. Handling multiple listeners alongside creating of custom classes to hand the input proved to be a slight challenge due to technical issues which will be discussed. To fully take advantage of such a interface a 3D environment was created. This environment needed to be designed to test the implementation of our system, along with not hindering the user’s experience.

For the more refined user input the touch screen is used. This is due to the refined accuracy and ease of understanding for the user. Using the touch screen we have the camera position move based on view angle. When the screen is swiped an event is called causing the placement of the camera to move in that direction, such as swiping left to move left. For multi touch events a loop is used to go through each touch and based on that perform an action. The action performed was also reliant on the screen size, as with larger screens it was not unusual for a user to use multiple fingers when one would work. This worked by first taking into account the screen size, then based on that finding the distance between the two inputs. Then if the inputs were of a set distance apart they would be handled as two touches or
if not they would act as a single touch event with the central value between acting as the input. For multi-touch purposes we implemented a zoom feature, which is a common gesture amongst many Apps.

The Leap Motion implementation was not as simple as using touch events. For this we required specialized drivers which have not been updated alongside Dart, causing them to slowly become outdated. This required a little bit of updating as some core functions were no longer working, causing crashes if set gestures occurred. Once the Leap Motion was functioning correctly we were able to create a class to translate between the sensor input and the simulation. This input could range from various gestures to hand location, rotation, or how many fingers were being held up. For our application we only wanted to use the rotation of the hands to apply this to our camera. Once a rotation value was read we would pass that to our custom class which would then be converted into a more relevant value for the view matrix.

As the data is in control of manipulating the view matrix, it creates an environment where the user’s hand is able to determine how and when we move around the scene.

4 Results

The resulting application partnered alongside a touch screen and Leap Motion, allowing for new and unique interactions to take place. We tested our setup on a range of screen sizes, with few limitations found. Support for smart phones unfortunately was limited in terms of input.

Algorithm 1

```dart
declare cameraMatrix, hand
Launch Leap Motion Listeners
onFrameUpdate
if hands.length != 0;
  declare Vector3 handDirection = hand.direction;
  updateCamera(handDirection)
end if

updateCamera(Vector3 handDirection)
  declare rotationVector = handDirection;
  check data, edit values to keep in a set range if needed;
  cameraMatrix.rotate(rotationVector)
```

Algorithm 1 shows the process of updating the camera Matrix based on the inputs from the Leap Motion.

Figure 3 shows the output to the console from the Leap Motion.

Figure 4 shows a user moving around a scene using the touch screen and Leap Motion

5 Discussion and Conclusions

We managed to develop a simulation and terrain navigation application which works on a range of screen sizes, from 22 inch to 65 inch, with the limitations only being those arising from the physical geometry and accuracy of the Leap Motion device. The Leap is designed - and works well - in front of a conventional desktop-sized display.

At present our software and application could not be used on smart phones and tablets, as there is as yet no support for Leap Motion on such platforms. This may change in the future and it may be that the Leap technology will be an integrated part of future tablet devices. Our Dart code base makes it likely that we would be able to port our system to any future platform, particularly using a Web platform interface to Dart.

There are some issues connected with Dart and its conversion to JavaScript code to be run on the Web client. Our application cannot be run as converted JavaScript, like some Dart code can - this is due to the need to use the Dart interfaces to the devices. Nevertheless, this work has demonstrated that a modern language like Dart can provide a very convenient integration language to incorporate input from two very different device paradigms.

HCI as a field is continually developing and there is still considerable scope for new paradigms and associated standards and terminologies to describe the sort of gestures and touch patterns that an application developer would wish to have supported for simulations or the closely related field of gaming applications [35].

In summary, we believe integration of new and emerging HCI technologies such as direct touch, with its positional accuracy, and free moving gestural interactions, with their natural fluidity of motion to be a powerful and useful addition to a simulation model such as the procedural terrain generation and navigation tool we have developed.

We anticipate that these device-level technologies will become even more widespread and that they will find uses in many other applications. Finally, we conclude that Dart is a
very promising integration language for all these operations as well as providing a good platform for implementing the simulation application itself.

Acknowledgments

One of us (THM) thanks Massey University for financial assistance in travelling to the UK, where this work was carried out.

References


