The Bird Table

Norman Lau

Making Things Interactive

Fall 2010

Abstract

The Bird Table is an application built on an interactive multi-touch table. It allows you to playfully interact with a visualization of digital birds.

For this project, I constructed a prototype multi-touch table based on work from the open source project reacTIVision. I also developed the Bird Table software application that runs on the table.

This project acts as the first step in a larger examination of how multi-touch technology can be made useful in various contexts.

The Bird Table in use during a demo.

The Idea

Multi-touch technology has been around for a long time now, but there are still issues to explore around how it can change the way we interact with our objects and with each other. In particular, collaborative uses of multi-touch surfaces is still considered somewhat novel.

I wanted to create a small application to learn about the issues around developing for a multi-touch surface and also to begin exploring how the technology could facilitate interaction between multiple people.

The idea of the Bird Table is a simple, playful app that let's multiple people interact with a visualization of birds. Utilizing the object recognition capabilities of a multi-touch table, users can manipulate properties like color, scale, and number of birds.

Construction

A lot of information for this project was drawn from online resources created by a large DIY multi-touch community.

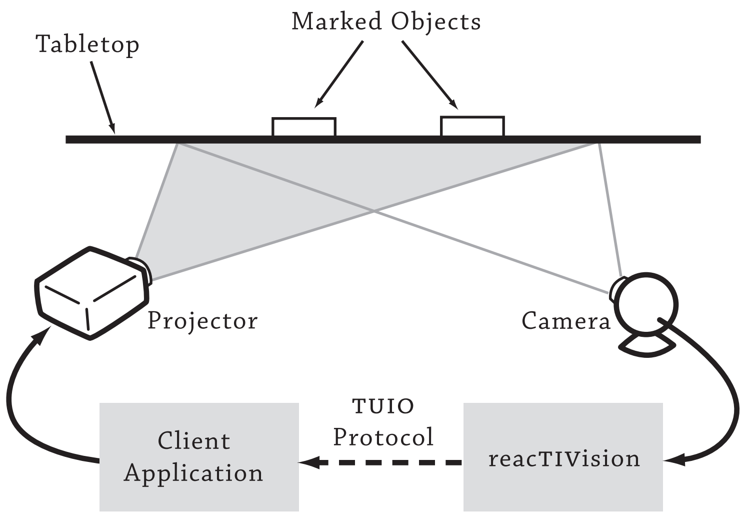
Primarily, I received guidance from the work of the open source project, reacTIVision (http://reactivision.sourceforge.net/). Coming out of the Universitat Pompeu Fabra in Barcelona, Spain, reacTIVision was developed as the computer vision technology supporting the development of the Reactable, a multi-touch tabletop music synthesizer.

The software works by using the TUIO protocol (http://www.tuio.org/), also created by the reacTIVision developers.

According to the TUIO creators:

> *"The TUIO protocol allows the transmission of an abstract description of interactive surfaces, including touch events and tangible object states. This protocol encodes control data from a tracker application (e.g. based on computer vision) and sends it to any client application that is capable of decoding the protocol."*

The following diagram shows the basic data flow of the technology:



adapted from TUIO.org

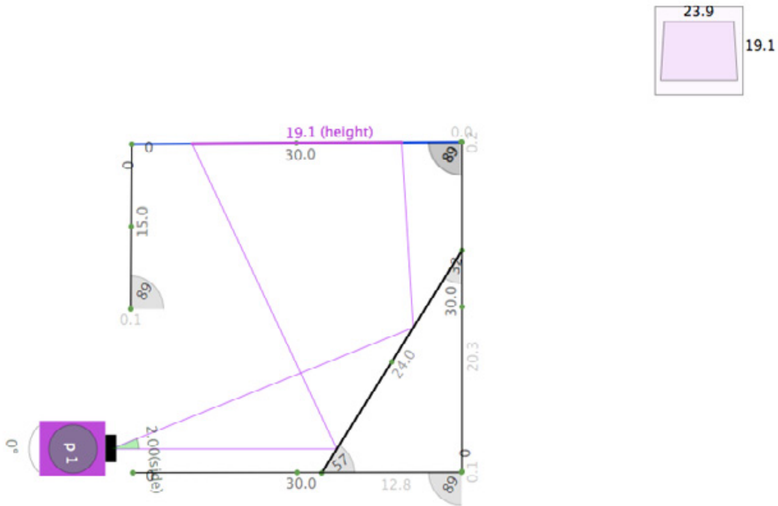
There were two main components to building my project: the hardware construction of the table itself and the development and integration of my client software application.

Hardware

The main physical components of the system are a camera, a projector, a screen for the surface display, and the frame of the table. Conceptually, the projector simply casts the video display onto surface of the table from underneath. However, consideration of each of these components is not independent which complicates the situation. The size of the screen and the dimensions of the frame depend on the projection throw length and the camera focal length. Coordinating these variables can get quite messy.

After deciding upon an appropriate size for my screen, I calculated how much distance the projector would need to project the required display dimensions. To accommodate long projector throw lengths, a mirror is placed at an angle inside the table. This allows the projection to be projected horizontally, and reflected up vertically towards the screen.

See the diagram below for a rough schematic of the projector setup from the side. The diagram was created with SimProj (http://benjamin.kuperberg.fr/lab/?p=4), an excellent simulator created specifically for the design of multi-touch table setups.



Construction of the physical table.

Some components of the system, including projector, screen, camera, and objects.

Once I had a rough idea of the dimensions for the frame (and after a few quick tests with an actual projector), I made some drawings for the frame of the table. I chose to make the box out of plywood so that I could cut exact pieces on a CNC router. The frame was designed to be pressure fit, with no need for screws or nails. I also cut small shapes that would act as the objects for interacting with the table.

The screen was simply a piece of clear acrylic, laser cut to the appropriate size. A sheet of vellum is placed over the acrylic in the final setup, creating a translucent surface. This allows the camera to see objects placed on the surface of the table, while at the same time providing a surface for the projector to display on.

The camera selected was a PS3 eye. From my research done on the web, I found that this was a low-cost camera that did relatively well in computer vision applications and was popular in the DIY multi-touch community.

Note that in the ideal table setup, I would have modified the camera to only detect infrared light and use IR light emitters to illuminate the inside of the table. This ensures that the projected image doesn't interfere with object detection by the camera. However, in the interest of keeping costs down and of staying within time constraints, I opted to see how far I could get only using visible light.

Software

The two main parts of the software system is reacTIVision and my client application, the Bird Table.

reacTIVision comes as a fully functional from the website. It allows for various settings to customize for video dimensions, lighting conditions, etc. It was relatively easy to setup and works well with the PS3 web camera. Source code is also available if any modifications need to be made.

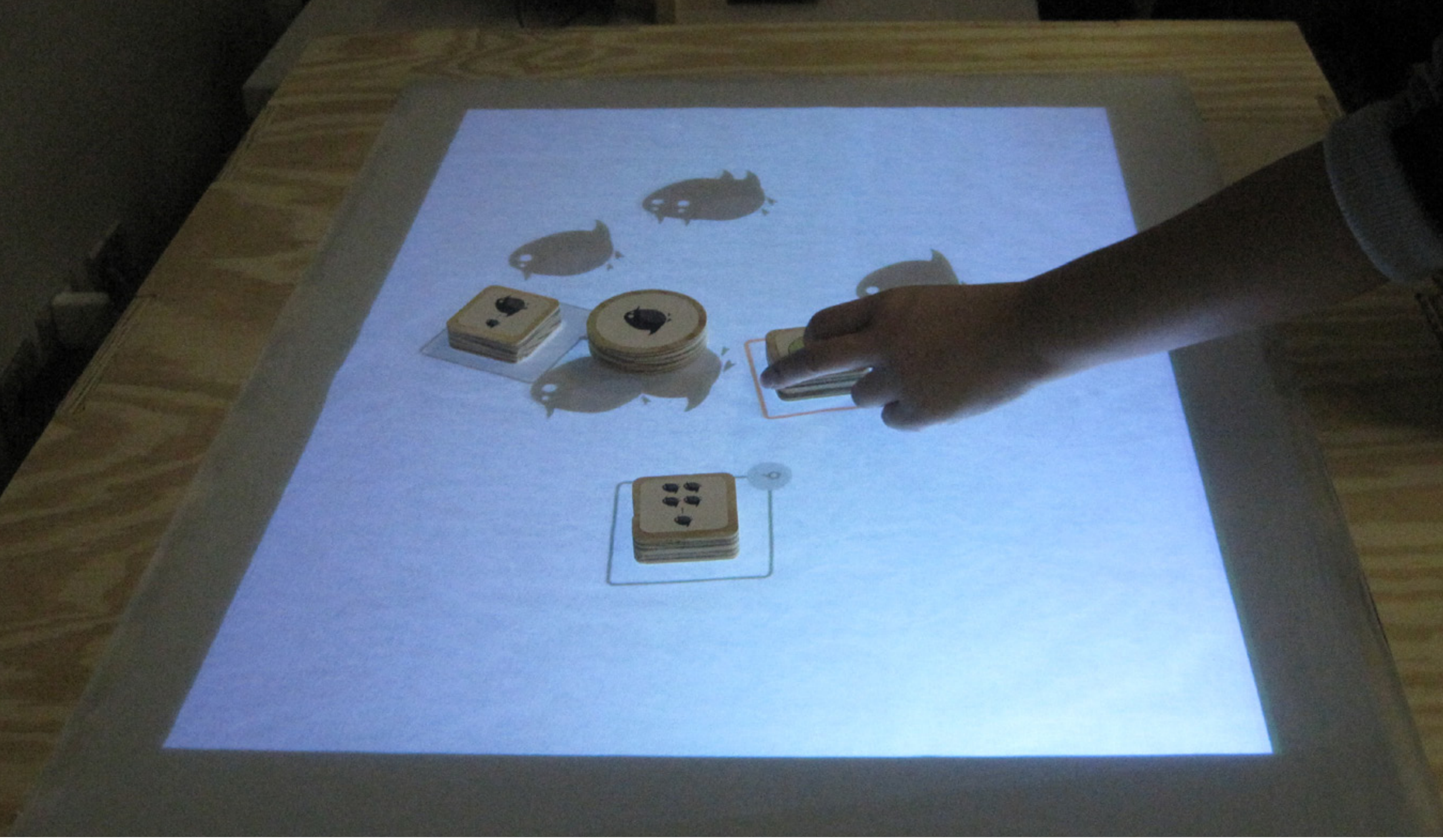
The client application was the majority of the programming effort. As explained on the reacTIVision website, the client can be programmed in a variety of languages, as long as it can understand the TUIO protocol.

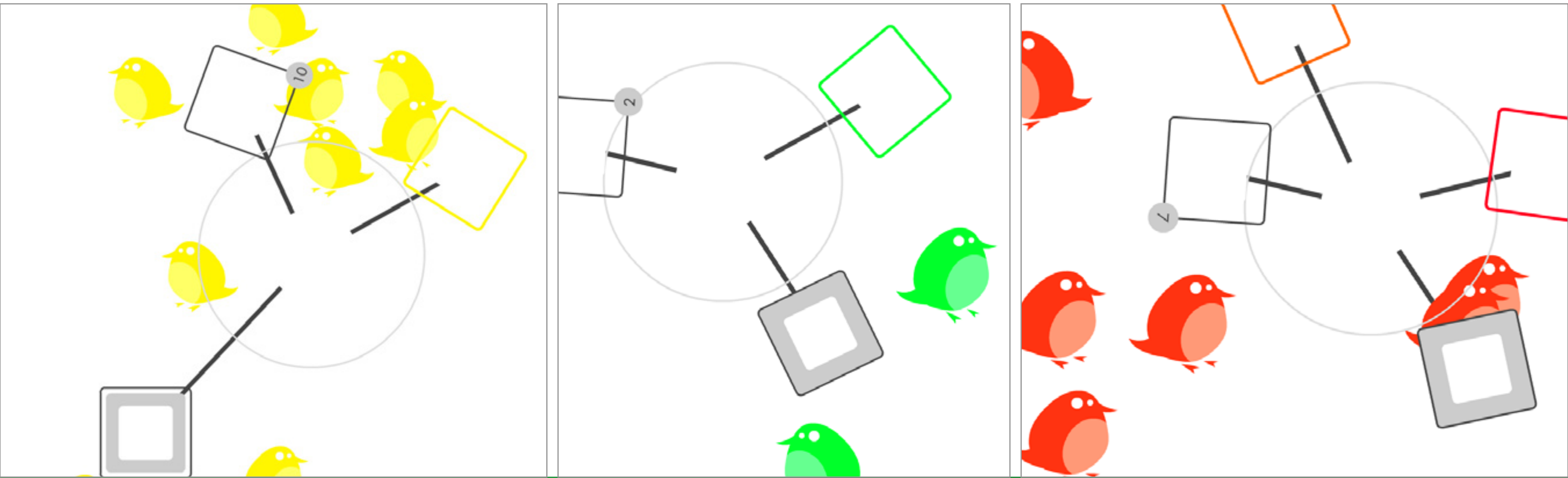
I decided to program my application in Flash so that I could take advantage of its graphics and animation capabilities. Unfortunately, Actionscript 3 doesn't natively support UDP, which is what TUIO is ultimately built upon.

To get around this, I used udpflashlc-bridge (http://gkaindl.com/software/udp-flashlc-bridge) which bridges udp input to Flash through AS3's LocalConnection functionality. Combining this with the TUIO AS3 library (http://bubblebird.at/tuioflash/tuio-as3-library/), Flash can understand the TUIO protocol and work with reacTIVision. This process is the method of communication suggested on TUIO.org.

Note that other languages have more direct means for reading UDP and might provide a different options for the client application.

With the proper software connections in place, I was free to program my actual client application, the Bird Table. In order to ease development, the TUIO website provides a TUIO simulator, that generates the same inputs that an actual table would provide. This allowed me to develop my client application before I even had a working table. To see an explanation of my program, please see the video on YouTube (http://www.youtube.com/watch?v=Id4gRwJs4zo).





Screenshots from the Bird Table software .

Conclusions

In the end, this was quite a ambitious project as I had significant obstacles to overcome in both hardware and software.

For me personally, the hardware portion provided more challenge, since I have less experience with it. The project provided me with my first experience preparing files for a CNC router and constructing an artifact of this size.

One of the greatest difficulties I encountered was related to my decision to use visible light rather than infrared. Since the camera image is affected by ambient light and the projected image, object detection can be spotty at times. In the future, it would be a worthy investment to buy the necessary IR components.

I plan to move ahead with this project as part of my master's thesis. I will examine how an interactive tabletop tool can be used in the context of service design.

Lastly, I must give thanks to Eric Brockmeyer, the TA of our Making Things Interactive class, for helping me with the physical construction of my table and to Mark Gross, our professor, for guidance on the project.