Reviving a Tangible Interface Affording 3D Spatial Interaction

Steve Sutphen[1] Ehud Sharlin[1] Benjamin Watson[1] John Frazer[2]
[1]*University of Alberta, Canada* [2]*School of Design, Hong Kong Polytechnic University, China*
steve@cs.ualberta.ca ehud@cs.ualberta.ca watsonb@cs.ualberta.ca sdfrazer@polyu.edu.hk

Abstract

Tangible interfaces offer natural means of human computer interaction and have already been shown to simplify existing computerized applications and offer solutions for tasks that were considered to be "out of the scope" of human-computer interaction. We are currently perusing new applications for 3D geometry defining tangible interfaces. In this paper we attempt to provide the reader with introduction to the underling paradigms and research in this emerging domain. We describe in detail a tangible interface, the Segal Model, created by John Frazer and his colleagues 20 years ago. We detail our technical efforts in reviving this tangible interface and updating its communication and operating techniques. We describe our initial results of building physical 3D worlds on top of the Segal Model and then rendering them into fully active 3D worlds, using the "Half-Life"™ graphical engine.

1. Introduction

1.1 Overview

Our natural interaction with the world relies, among other abilities, on our tangible manipulation ability. We can move, manipulate, assemble and disassemble a seemingly endless variety of physical objects with very small cognitive effort. Yet, when it comes to Human-Computer Interfaces (HCI) the thirty year old Keyboard-Mouse-Monitor interface and the Window-Icon-Menu-Pointer (WIMP) interaction metaphor prevail as the major or even sole HCI. We use this interface while performing word processing, 3D graphical design and modeling, and even while manipulating a virtual avatar in a 3D-computer game. We believe that many of our natural abilities are blocked by this standard HCI, forcing complexity on what could otherwise become a simple, even natural HCI task. Especially, we believe that providing new and original tangible 3D-shape based HCI tools might change dramatically the way in which we perform 3D modeling and construction oriented HCI tasks.

We begin the paper by presenting a brief overview of the disciplines underlying these new concepts of tangible interfaces (section 2). We present a short summary of the related research performed in the field of tangible interfaces and emphasize the role that we are planning to play in this research field (section 3). We present our technical work reviving a pioneering tangible interface designed and built by John Frazer and his colleagues 20 years ago - the Segal Model [15,16] (section 4). We then present our initial results, and demonstrate some of the potential of a tangible interfaces based design of 3D worlds by using a Half-Life based graphical engine [30] (section 5). The paper concludes with a discussion of our future plans (section 6).

1.2 Physical, Natural Interaction Themes and HCI

What can we learn from the ease of building models or manipulating objects in the real world as opposed to the hardship involved in performing similar tasks in a digital environment? We briefly present three concepts that are taken from our "natural" interaction with physical tasks and which can be borrowed as themes for the design of new HCI paradigms and tangible interfaces: the affordances or transparency of the interface, the synchronization or coupling of perception space and action space, and the support of both pragmatic and epistemic tools [13].

- **Interface Affordances:** When we interact with everyday objects in the real world we usually do not have to consciously apply complex thought in order to manipulate or use them. Their "behavior" is inferred from their qualities: shape, weight, size, etc. This functionality expressed through the object's physical form is called "affordances" [13,19,26]. According to this concept, most physical objects "afford" their behavior, that is, their physical form contains a clear representation of their functionality. Most user interaction techniques, particularly in 3D modeling, include a set of rules and controls that manipulate their functionality. These rules and controls are far from affording their functionality. A simple example is the standard mouse. Although it enables us to control an endless variety of tasks, it is far from "affording" its different functionality in the diverse tasks it performs.
- **Synchronization of the Perception Space and the Action Space:** When we interact with an object or physical media in the real world, our hands and fingers (termed as parts of our action space) coincide in time and

space with the position of the object we are handling (termed as part of our perception space) [13]. This spatial and temporal "natural" synchronization of our perception space and our action space enables us to perform complex tasks. Yet, conventional computer user interfaces support concepts of separation between input and output devices, thus creating a spatial division (and in worst cases, where latency exists, even a temporal division) between the user's perception space and action space. For example, the screen - mouse standard user interface separates the objects in the perception space (the screen) from the action space (the mouse) [13].

- **Support of both pragmatic and epistemic tools:** When we build a physical 3D model, we actually perform an activity that is both cognitive, or goal related and motorized [13]. This means that we try to reach a cognitive goal - giving a 3D shape to a cognitive concept - by physical means, i.e., interaction with a certain physical media. Such physical task involves both pragmatic and epistemic actions [13,24]. Pragmatic actions can be defined as the "straightforward" maneuvers that bring our 3D shape closer to our cognitive goal. Epistemic actions, on the other hand, can be defined as the "trial and error" maneuvers that we perform while trying to progress. Some of these maneuvers will fail and would not bring us any closer to our goal, while others will reveal new information and directions leading to it. In fact, this information would have been very hard to find without trial and error. Physical 3D modeling, in most of the simpler cases, provides us with both pragmatic and epistemic tools for performing our tasks, with a low cost for mistakes made while executing an epistemic step [13]. While building models with Lego blocks we almost inattentively test the validity of certain actions, some of which do not lead directly to our "broad" task. The price of correcting an action that proves to be erroneous is minimal. On the other hand, most of the tools in a computer-user interaction environment are geared to pragmatic actions. The idea of simple, primitive epistemic tools does exist in user interaction techniques. Yet, when it comes to complex tasks such as 3D modeling, the usability of these epistemic tools is substantially inferior to their physical world parallels. For example, the concept of the 'undo' operation is linear, meaning that to 'undo' a single erroneous operation, you have to also undo all the operations that followed it.

2. Related Work

Many tangible interface technologies and applications are emerging as more natural HCI alternatives to the standard mouse and keyboard interface. To mention a few, the toy industry has several examples of such tangible interaction tools between children and computerized applications [28] or 'huggable interfaces' [3,23]). Tangible interfaces were introduced as simple and natural methods to support animation keyframing [8,9]. Tangible curve and surface inputting tools were introduced in the ShapeTape [6] and the Haptic Lens [29] respectively. Tangible 3D-shape manipulation was suggested in DO-IT [25]. Topological (though limited) tangible inputting for educational purposes was introduced in [20,31]. Planar, physical desktop tangible interaction was introduced in several studies [7,11,12,13,21,22,32]. For a thorough discussion of this emerging field and its potential see [10,11].

Several research groups developed various tangible interfaces for 3D modeling and design over the last 20 years, for a variety of applications. The impetus for the practical aspects of our current work originated in pioneering HCI tools developed by John Frazer and his collogues as early as 1980 [2,14,16,17,18]. Frazer developed working prototypes of 3D Input Devices (he also used the terms "Machine Readable Models" or "Intelligent Modeling Systems") in order to support new design approaches in architecture. The models were intelligent, in that the model was able to extract its geometry automatically. The supporting design software enabled implementation of feedback concepts such that a model (via its supporting software running on a host computer) can advise the designer on design errors or insights interactively. It is important to note that side by side with the development of 3D Input Devices by John Frazer and his colleagues, Robert Aish published and developed tools along similar lines [1,2].

Figures 1 - 4 show a sample of Frazer's 3D input devices. Figure 1 shows the 'Universal Constructor', a baseboard ('The Landscape') and a set of cells that can be stacked on top of it. The system is a tangible input and output device. A cell's position and identification are sampled in real-time, while the user stacks them on top of each other or detaches them. The user can receive feedback from the system by LEDs that are integrated into the cells (for example, the cell can output messages like 'Detach me!' or 'Stack a cell on top of me!').

Figure 2 and 3 present the 'Flexible Intelligent Modeling System' and the 'Three Dimensional Intelligent modeling System', respectively. Both were used in order to give users a 3D shape input interface. The interaction was performed by simply attaching and detaching the model's blocks. The model created was sampled in real-time by a host computer.

Another 3D input device, the Segal Model (figure 4) was built by John Frazer and his colleagues in the early 80's in order to support the work of the architect Walter Segal [15,16]. Segal had developed a timber-frame technique for self-homebuilders but had found that the users encountered difficulty when it came to self-designing their homes. The tangible input tool that was developed, The Self-Builder Model, enabled easy home

design for users without any knowledge or experience of either computers or architecture (The model was named the 'Segal Model' after Walter Segal had died). The 'Segal Model' is the main focus of this work and will be described extensively in the following sections.

Sadly, the HCI community largely did not refer to these models, and many of them were 'reinvented' by various research groups. Geometry Defining Processors (GDP) were introduced [4] along similar themes for physically defining a 3D geometry problem and solving it digitally. On a recent effort MERL's Lego-like blocks can define, in a easy-to-use manner a large scale 3D geometry and extract it to a host computer in real-time [5]. A new effort from Xerox seems to follow the same ideas, although Xerox's suggested 'Digital Clay' might enable also direct (without user's intervention) shape outputting (as well as input) [34]. Although following different paradigms (not HCI oriented), the NuMesh project [33] is worth mentioning. NuMesh suggests and examines a 3D modular network of high performance computers plugged into a 3D-lattice topology in a Lego-like modularity.

Figure 1. The Universal Constructor [15]

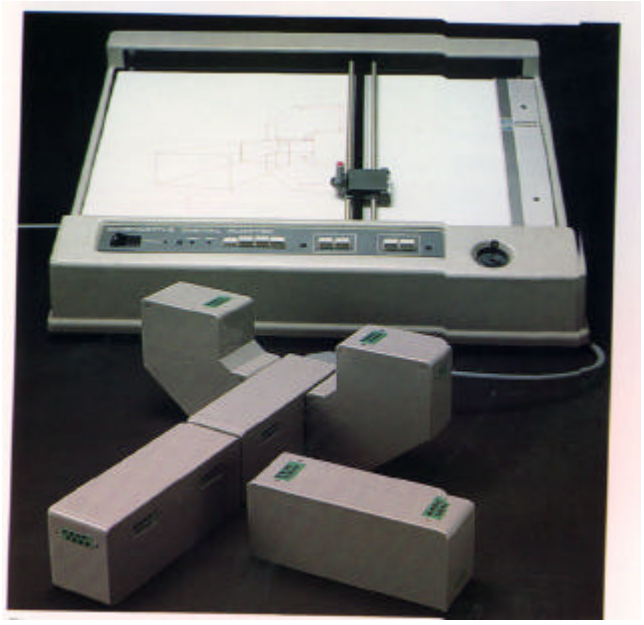

Figure 2. Flexible Intelligent Modeling System [15]

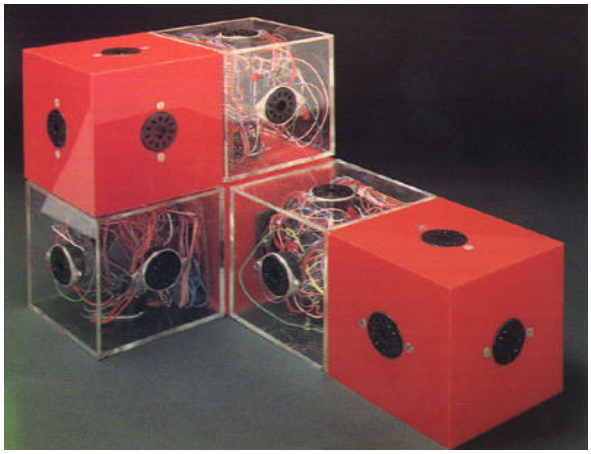

Figure 3. Three - Dimensional Intelligent Modeling System [15]

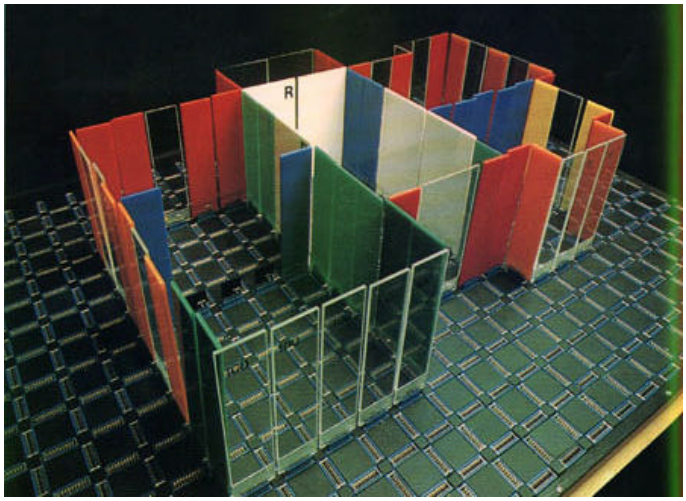

Figure 4. The Segal Model [15]

3. Research Goals

Our initial goals were to implement a 3D geometry input device, much like the Three - Dimensional Intelligent Modeling System (figure 3) or MERL's blocks mentioned in section 2. After an extensive research of the past and current efforts in this domain we came to realize that the development of such tools, as engineering demanding as it might be, would probably not yield an original contribution to the field. We decided to use an existing tangible geometry-inputting tool, and focus our efforts into the research of new application domains. We believe that new geometry inputting tangible interfaces

can not only change dramatically several interaction paradigms but also make completely new interaction paradigms feasible. We are currently researching three main applied domains for new tangible interfaces: new design tools, tools for the visually impaired and Neuropsychological assessment tools. These efforts however are beyond the scope of this paper which will detail our efforts of technically reviving one of these Machine Readable Models - the Segal Model and report our preliminary results.

The Segal model allows the user to construct various home plans, on top of a large grid, using 127 different physical entities, each with a different electronic identification. The entities can range from colored plastic panels to any other small-scale object. The board output was supported by wire-frame rendering software and a design feedback tool, which produced design advice, such as house area and cost, to interactively help the designer (and in a way, imitate some of the expertise of an experienced architect [15]). The board was scanned electronically, in real-time with very low processing demands from the hosting system (i.e., no real-time image processing, etc). The Segal model is a one-of-a kind device, a single copy was constructed for the research, and there are no known copies. The model was used with an early 80's, currently obsolete, computer [15,16]. Our immediate research goal was to 'make the Segal Model work' with a standard, modern PC using a generic as possible interface.

4. Implementation

4.1 Description of the Segal Model

The Segal model, as mentioned in section 2, is a desktop sized (102cm x 71cm) board with an array of edge connector slots covering most of the surface (figure 4). The edge connectors have 9 contacts. The center pin is used by the scanning hardware to energize the slot (a positive voltage is applied to the center pin to read out the slot contents). The panels or other tangible objects that plug into the connector slots have one to eight diodes with the anodes connected to the common (center) contact and the cathode connected to one of the eight outer contacts. This allows each tangible object to have a code ranging from 1 to 255. These values can be used to encode shape, color, size, or what ever attributes are useful for the system being designed using the model. In order to extract the orientation of the object (which way round it is – front or back) only 127 of the object codes are used while the other 127 support symmetrical orientation positions of the same objects. One way to visualize the model is as a grid formed by 24 columns of 16 vertical slots, and 16 rows of 24 horizontal slots. There are 768 slots in total on the board. The model is a passive device - it has no electronics internally - and the tangible objects only have the simple diodes for specifying attributes.

The model was constructed, along with a controller, to read out the slots occupancy values and to input them to a computer. The two units are hooked together with five 50-pin ribbon connectors. The controller is a relatively simple circuit designed to read out the 8-bit data values from the 768 slots, while still keeping the number of input/output wires manageable (there are a total of 768 x 8 = 6144 pins that must be read). The controller and board, do this by multiplexing the signals over the ribbon cables.

The basic idea is similar to a keyboard scanner, or a memory chip. Think of the board as a matrix consisting of 96 word-lines - or columns, and an orthogonal 8 bit-lines - or rows (these are actually not single lines but are an 8-bit bus, but for this part of the discussion that can be ignored for simplicity). The controller energizes one of the word lines, and then selects one of the 8 bit-lines to see whether there is a connection between the two. Diodes (on the tangible objects that are plugged into the Segal board) prevent false paths being generated as would be the case if wire or resistors were used to bridge the rows and columns. In the following simple example (figure 5) there are "sneak paths" that show a false connection at C0 (via C1,B1,B0) and another at B2 (via B1,C1,C2).

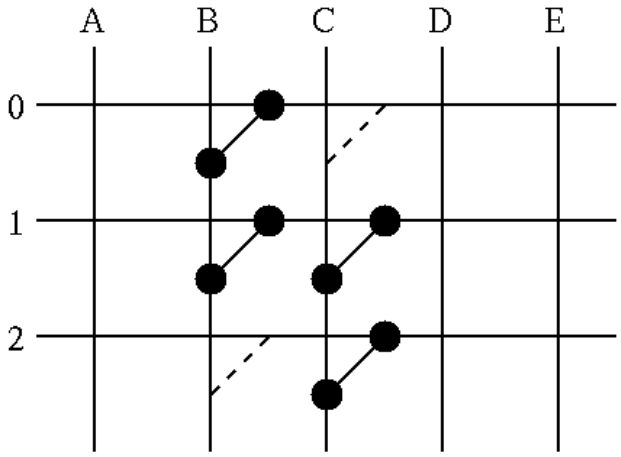

Figure 5. Matrix Scanning: Phantom connection is the dotted line

The word-lines are energized from a group of counter/decoders that form a "walking one" function. The bit-lines are read out one at a time via an 8:1 multiplexer (actually 8 of them as each output is 8-bits wide), driven by a 3-bit counter. The interface, as built, had separate signals to advance the two counters and also separate reset signals to set the two counters to an initial known state.

In the original implementation of the model the controller was attached to the I/O bus of the now obsolete BBC-B. This interface was implemented with a Rockwell R6522 (VIA - Versatile Interface Adapter). The VIA buffered the bytes from the controller and handled the handshake required by the host computer. It also allowed the host computer to control the two reset lines and two counter clocks via a control register.

4.2 The Task - Our Design Goals

For our current and future use of the Segal Model we determined that a more modern, and more widely available host would allow us more flexibility than trying to use the BBC-B even as just an interface to our modern graphical computers. There was a trade off between having to learn and reuse the BBC-B, and knowing that it would take some time to design, build, and debug a new hardware interface and software drivers. Some analysis indicated that it would add new life to the model, and make future development easier, if we devised a way to attach the controller to a commonly available computer, like a PC.

Our goal was to construct a hardware-software system that would allow a researcher to access the occupancy data from the model with relative ease. We wanted to be able to read out the 768-position occupancy matrix fast enough that a human user would not perceive a delay. We considered which operating system - Linux or Windows - to use, and decided that initially we would use which ever one was easiest, but that we wanted to do both so the model could be easily used in more environments.

These constraints dictated that we needed to design around a common "generic" interface, and that the hardware-software interface be kept as simple as possible (making it easier to develop two sets of software).

4.3 Design Considerations

Four common interfaces were evaluated to determine which would suit our requirements best. We considered using a digital I/O PCI card that could be customized to work with either the computer bus interface, or plug in as a replacement for the VIA chip. This did not fit well with the criteria for using standard interfaces, and the software to support it would have required a custom driver for both platforms. We also considered replacing the VIA chip with a microcontroller and having it send the data to an Ethernet interface (perhaps even as an HTML server), but again it was more complex, and higher performance than what we needed. A serial port (perhaps implemented with a microcontroller) was also considered. The serial transfers would have had a perceptible delay, and the hardware to implement a serial port is somewhat more than the parallel port solution that we decided to use.

Parallel ports are relatively fast, and there have been a variety of devices that had been attached up to them (from printers as they were originally designed, to scanners, cameras, floppies, and even CDRom writers) showing that they were flexible enough to accommodate this project.

Studying documents and case studies brought to light that there were several "standards", and that design mistakes could destroy the parallel port chip. The different port standards existed because the original standard (described by IBM in 1981) was not amended or formalized until the IEEE 1284-1994 standard was issued in 1994. The original parallel port had useful characteristics, but as manufacturers extended these, they did it in an ad-hoc way - yielding different "standards". The IEEE standard has several different modes - to handle devices of differing complexity and function. For our application the PS/2 bi-directional mode and the EPP (Enhanced Parallel Port) mode, were most appropriate. ECP (Extended Capabilities Port), a major part of the IEEE 1284 standard, would have worked in principal, but it was too complex for our needs. EPP is easier to support with software than the PS/2 bi-directional is, because the port hardware handles the hand-shake details. It was also capable of faster transfers (up to 1MB/second compared to the PS/2 which typically transfers at around 150KB/second). We tentatively decided to use this mode (the peripheral hardware required to implement the PS/2 and the EPP are very similar).

The parallel port solutions did require that the number of control signals be reduced from 4 in the old design (two clock signals, and two reset signals). We combined the two reset signals into one INIT signal, and added logic to advance the "walking-one" count when the 3-bit multiplexer counter overflowed - going from a value of 7 to a value of 0.

This gave us very simple software model of the interface. There was one control operation - the INIT bit which sets both counters to zero, and there were read_data operations. The read operation would be used to advance the counter to the next slot.

Since we had defined such a simple interface to this board, we were able to find a driver that would work without having to learn the details of driver writing. We obtained a "user-level" parallel port driver for Linux that was being developed by Tim Waugh [27]. This would work for our initial testing and proof of concept.

4.4 Modification of the Segal Model

Figure 6 shows a block diagram of the revised controller. The “inputs” are an INITialize signal, and the READ_Data signal. The INIT signal sets the 3-bit “row-multiplexer” counter to zero, and resets the “walking one” counter-decoder to zero—which energizes the `0’ output

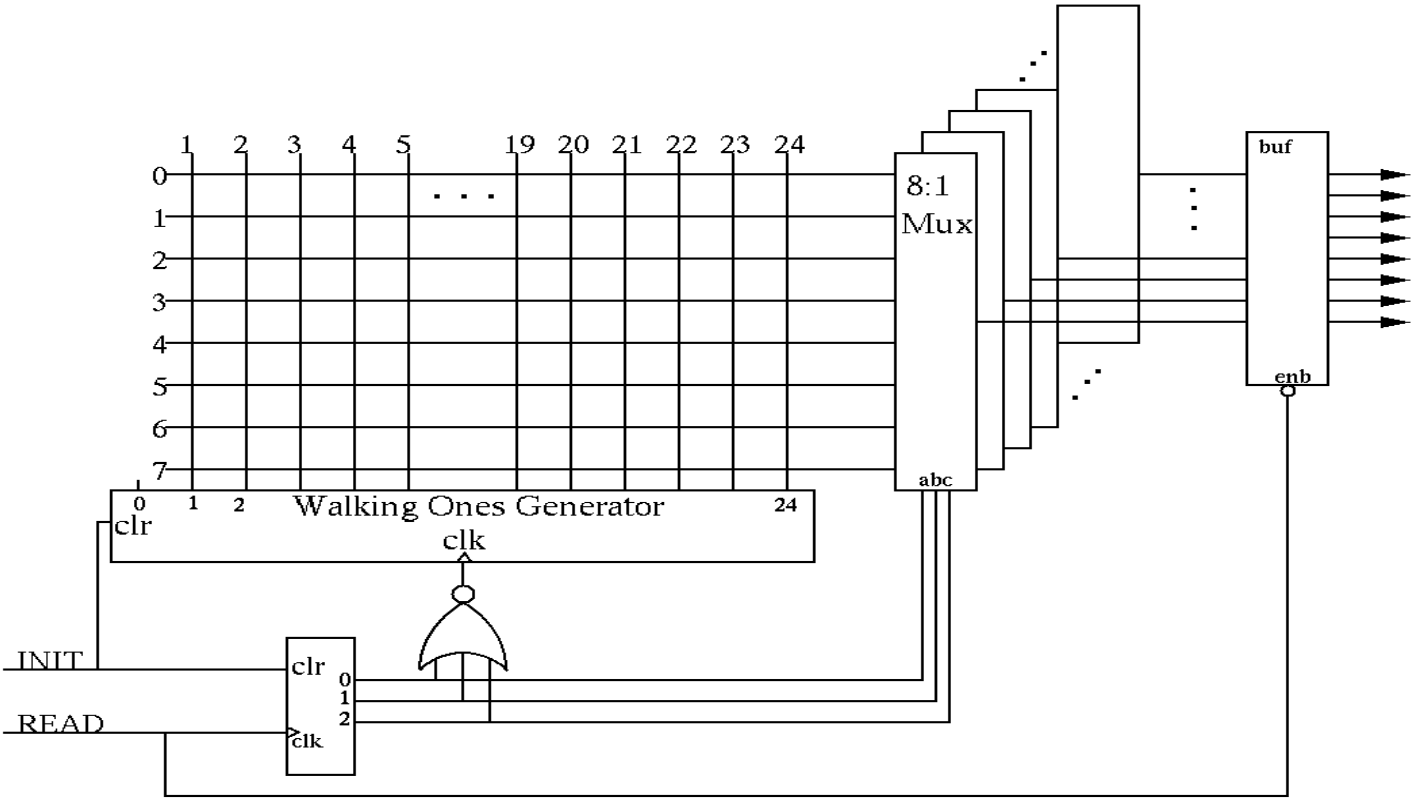

Figure 6. Controller Block Diagram: A simplified schematic of the parallel port - Segal model interface

which is not connected. The ReadData signal enables the 8-databits into the parallel port cable while it is an active `low', and when it goes high at the end of the read cycle it triggers the 3-bit counter to increment to the next row. The NOR gate output will have a low to a high transition when the counter overflows from seven to zero. This transition is used to increment the walking one counter. The walking-one circuit is implemented to wrap around at the end of the scan, so multiple scans could be implemented by simply reading more data. As there is no capability to set the two counters to a particular value, the read out is strictly sequential. If one wants to read slot N they would clear the counters with the INIT function, and then would read and discard N-1 values and then read N. In practice this is not a problem as it takes so little time to read the entire array. The remainder of this section describes the technical details of the modification of the model, and the software to make it functional. The non-technical or casual reader can skip it with no loss of continuity.

Before making the modifications to the working Segal Model controller we tested the critical logic and software with a breadboarded circuit. The test circuit implemented the handshaking and a simple 8-bit counter that was incremented each time the port was read. The driver was installed, and the system was tested with a small program. The eight input bytes that the program read were all the same, and were not related to the values in the counter. The handshake logic was working and the computer read operations were causing the counter to increment, but no data was coming back. We investigated timing issues but that was not the problem. We also changed the test circuit and program to use PS/2 mode, but found the same result. After some deeper research into the data sheets for the chip that implements the parallel port and an examination of the source code for the driver, the problem was located. When a program set the driver into a particular mode, it would save and use this state information to implement the handshake protocol (adjusting for the fact that the code to read data in PS/2 mode is quite different than EPP), but it did not tell the parallel port chip to change modes. The reason for this was that the driver had been developed for the next, experimental, version of Linux and then back-ported to the standard version. The lower level drivers differ in these two versions, generating this problem. We made small modifications to the driver and that fixed the problem.

The controller was modified with the tested design. The 240V 50Hz power supply was replaced with a universal input supply so the controller could be easily used anywhere in the world. The handshake and clock chain were tested on the bench, to assure that there were no wiring mistakes that would damage the parallel port.

When we did our initial test of the complete system we found that we could detect the presence of tangible objects, but the slot addresses did not seem logical, and more seriously, a single panel would be detected as if there were in several places.

We found that if we slowed down the scan (putting a 20-millisecond delay between reading each byte) we did not get the ghosting. Although this solved the problem it made reading the entire board (768 data points at 20ms per sample is a little over 15 seconds) much slower than the design goals. Crosstalk was also considered, as the wires from the controller to the slots are over a meter in length. An oscilloscope showed that this was not the problem, but rather one caused by residual charge on the column line. When a column is energized it charges up the wire to the group of slots. The reading of a row does not discharge this wire very much (CMOS has a very high input impedance). Because of the very low leakage path, the charged wire took on the order of 400 microseconds to have its voltage level drop below the logic `1' threshold. We had two choices - either slow the reading speed down, or add resistors to discharge the capacitance. We could not slow the transfers down with hardware, as the EPP specification requires that a port return data within 10 microseconds or the transfer will time out. This was much less than 400 microseconds. We opted for pull down resistors (20k Ohm) installed on the sixty-four input lines from the slots. That solved the ghosting problem. This had not been noticed in earlier uses of the model as it was always scanned slowly to show the user what was happening. Our new requirements were somewhat different.

The calibration of the addressing led to the discovery of a design mistake. With a single test panel we found that the lower addresses were the vertical slots near the left edge of the board, and that all the vertical slots came before the horizontal ones. The horizontal slot numbers increase from the upper left to the lower right in a row wise fashion. While doing the calibration we discovered that the first seven vertical slots were not giving us data. The test program was discarding the first eight bytes read, as the "clear" function sets the "walking-ones" counter to zero. The decoded output line `0' is activated in this state, but not used, so no slot groups are selected. Since none of the sockets was energized in this state, and since the state would not change until the 3-bit counter overflowed we discarded the first 8 values. Thinking that our understanding was wrong, we took out the "pre-read" and tried again. The first slots now worked, but the addresses still were not logical. The first column of vertical slots were numbered 8, 1, 2, 3, 4, 5, 6, 7, 16, 9, 10, 11, 12, 13, 14, 15 which did not make much sense.

An analysis of the logic showed that it was a simple problem, and is often the case, there were other pointers to the problem that we had discarded. The "missing" slot problem was not a logic error in the program, but was a hardware logic problem that was also giving a strange sequence of addresses. The difficulty was that the walking-one counter was being incremented when the 3-bit counter went from zero to one instead of from seven to zero; the wrong edge of the clock had been used. The problem had been created when the design was optimized to reduce the number of chips that would be needed to implement it. The problem was fixed with the addition of an inverter, and changing the program back to the way it had been. We had a working system that could be used on most computers world wide.

4.5 Current Technical Status and Plans

The model and its interface are basically working well. A program can scan the entire board in around 2msec (with essentially no human perception of delay). Some of the tangible objects have deteriorated because of the storage, and need to have their edge connectors cleaned. There are also a few that have a poor solder connections to the diodes on some of the panels that need to be found and fixed. We are also planning on finding or writing a MS-Windows driver, to improve the usability of the Segal module in different environments.

5. Preliminary Results

As a preliminary demonstration of the power of the Segal model as a design tool we choose to render our worlds via a “Half-Life” graphical engine [30]. The user starts by building a world using physical-tangible objects on top of the Segal model (figure 7). The physical model is sampled and transferred automatically to a virtual model in a “Half-Life” based 3D environment, with full control of all the appearance of the world (texture, lighting, etc.) and full support of virtual walkthroughs in the former physical model (figure 8). Furthermore, the virtual world can now be populated with virtual entities and characters, either by physical-tangible means or by software editing (figure 9). The physical model can then become a fully interactive, fully active virtual world (figure 10).

The preliminary use of “Half-Life” as a rendering engine prevented us from applying any interactive editing, since the process of rendering the virtual world is time consuming (less then a minute for transferring the

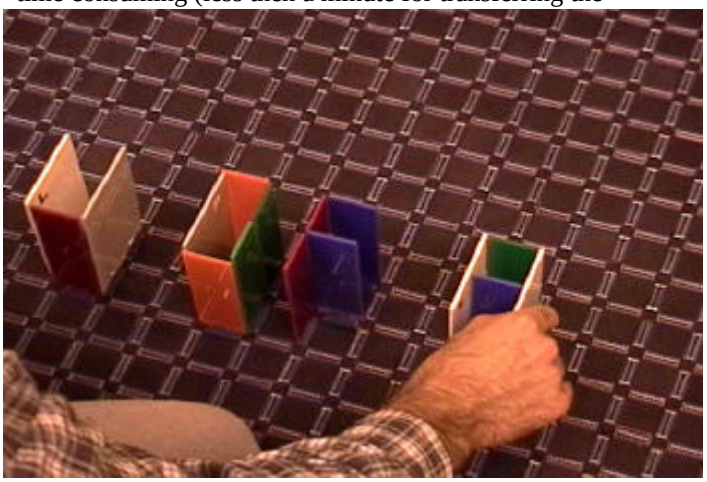

Figure 7. Building a world tangibly. The user builds the letters ‘U OF A’.

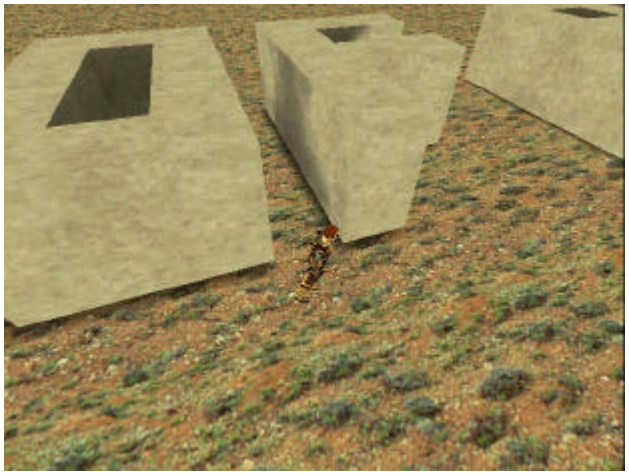

Figure 8. Gordon Freeman ("Half-Life"'s avatar) running below the virtual 'OF' structure.

Figure 9. Gordon Freeman meets two fellow scientists inside the 'U' structure.

Figure 10. A fierce battle takes place around the letters structures.

sampled model into a fully rendered 3D world on a regular Pentium PC). However, we believe that this simple implementation demonstrates the power of simple, natural and easy to use tangible input device in performing tasks that otherwise will involve a much more complex mouse-keyboard based interaction.

6. Future Work

In the short term our major tasks are solving the remaining technical issues concerning the Segal Model: temporally unstable connections and an eight undetectable slots. We then plan to create a graphical environment (in OpenGL ®) that will be editable in real-time by the tangible tool. On the long run our major tasks are to create several useful, testable interaction tools, using the Segal Model. We are planning several tools in the following fields: computer aided design, tools for the visually impaired and neuropsychological assessment tools. Our main task will be to test the tools with real users and to empirically prove their superiority over standard, existing interaction tools.

7. Conclusions

We briefly described the scope of past and current research in the tangible interfaces domain and the paradigms underlying it. We discussed geometry inputting tangible interfaces and our research motivations and goals in this field. We described a pioneering tangible interface called the Segal Model and our experience in connecting it via a standard interface to a common PC. We detailed our technical solution to this problem, using a standard parallel port connection. We then presented our initial results, rendering physical models assembled on top of the Segal model to a 3D virtual "Half-Life" worlds.

8. Acknowledgments

We would like to thank and acknowledge the contribution of John Frazer's team members who created with him the original Machine Readable Models. The models were designed and created from 1979 onwards by John and Peter Frazer. In particular, the Segal model was created for the architect Walter Segal in 1982 by John Frazer (then Professor of Computer Aided Design at the University of Ulster). Julia Frazer (then Director of Autographics Software Ltd.) wrote the original software. The help of John Potter, a research assistant at the University of Ulster is also acknowledged. Later modifications of the model to create a teaching tool (Calbuild) were by Steven Brown (also then at the University of Ulster) with David McMahon as research assistant. Funding for the project was from Autographics Software and the University of Ulster. The authors would

like to thank the current help and support of our colleague Dr. Mark Green from the University of Alberta, Computer Graphics Research Group. We also thank Lloyd White the computer graphics lab manger for his support and involvement and Matthew Olson for the "Half-Life" related advice.

9. References

[1] Aish R. and Noakes P., "Architecture without numbers - CAAD based on a 3D modeling system", Computer-Aided Design, Vol. 16 No. 6, pp. 321-328, Nov 1984.

[2] Aish R., "3D input for CAAD systems", Computer-Aided Design, Vol. 11 No..2, pp. 66-70, March 1979

[3] Alexander K. and Strommen E., "Evolution of the Talking Dinosaur: The (Not So) Natural History of a New Interface for Children", Conference proceedings on Human factors in computing systems (CHI) 98, April 18 - 23, 1998, Los Angeles, CA USA

[4] Anagnostou G., Dewey D. and Patera A. T., "Geometry Defining Processors for Engineering Design and Analysis", The Visual Computer, Vol. 5, pp. 304-315, 1989.

[5] Anderson D., Frankel J., Marks J., Leigh D., Ryall K., Sullivan E. and Yedidia J., "Building Virtual Structures With Physical Blocks (Demo Summary)", Proc. of UIST 99. Asheville, North Carolina USA, Nov. 1999.

[6] Balakrishnan R., Fitzmaurice G., Kurtenbach G., and Singh K. (1999). "Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip". Proceedings of the ACM Symposium on Interactive 3D Graphics (I3DG'99), pp. 111-118. New York: ACM.

[7] Bruns W. and Brauer V., "Bridging the Gap between Real and Virtual Modeling - A New Approach to Human-Computer Interaction" artec Paper Nr. 46. Online: http://www.artec.uni-bremen.de/field1/rugams/papers/texas/texas.html

[8] Don S. and Duncan J., *The Making of Jurassic Park*, Ballantine Books, NY, 1993.

[9] Esposito C. and paley W. B., "Of Mice and Monkeys: A Specialized Input Device for Virtual Body Animation," 1995 Symposium on Interactive 3D Graphics, Monterey CA USA (1995)

[10] Fitzmaurice G., Balakrishnan R., and Kurtenbach G, "Sampling, Synthesis, and Input Devices". Communications of the ACM, 42(8). pp. 54-63. New York: ACM, August 1999.

[11] Fitzmaurice W. G., "Graspable User Interfaces," Ph.D. Thesis, Univ. of Toronto. Online-http://www.dgp. toronto.edu/people/GeorgeFitzmaurice/thesis/Thesis.gf.html 1996

[12] Fitzmaurice W. G., Ishii H. and Buxton W., "Bricks: Laying the Foundations for Graspable User Interfaces," CHI '95 Mosaic of Creativity, 442-9, May 1995.

[13] Fjeld M., Bichsel M. and Rauterberg M., "BUILD-IT: a brick-based tool for direct interaction". In D. Harris (ed.) Engineering Psychology and Cognitive Ergonomics (EPCE), Vol. 4. Hampshire: Ashgate, pp. 205-212., 1999. Also online - http://www.iha.bepr.ethz.ch/pages/leute/fjeld/publications/EPC Ebuildit.pdf

[14] Frazer J. H., '"Three-Dimensional Data Input Devices". Computers/Graphics in the Building Process. National Academy of Sciences, Washington 1982.

[15] Frazer J. H., *An Evolutionary Architecture*, Architectural Association 1995.

[16] Frazer J. H., "Use of Simplified Three - Dimensional Computer Input Devices to Encourage Public Participation in Design", Computer Aided Design 82, Conference Proceedings, Butterworth Scientific, pp143-151, 1982.

[17] Frazer J.H., Frazer J.M. and Frazer P.A., "Intelligent Physical Three-Dimensional Modelling System", Computer Graphics 80 Conference, Conference Proceedings, Online Publications, pp. 359-70, 1980.

[18] Frazer J.H., Frazer J.M. and Frazer P.A.,, "New Developments in Intelligent Modelling", Computer Graphics 81, conference proceedings, Online Publications, pp. 139-54, 1981

[19] Gibson J. J., *The Ecological Approach to Visual Perception*, L. Erlenbaum: London, (1986).

[20] Gorbet G. M., Orth M. and Ishii H., "Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography," CHI 98, 49-56, April 1998.

[21] Ishii H. and Ullmer B., "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms," CHI 97, 234-241, March 1997.

[22] Ishii H., Wineski C., Brave S., Dahley A., Gorbet M., Ullmer B. and Yarin, P., "ambientRoom: Integrating Ambient Media with Architectural Space," Conference Summary of CHI 98, April 1998.

[23] Johnson M. P., Wilson A., Kline C., Blumberg B. and Bobick A., "Sympathetic Interfaces: Using a Plush Toy to Direct Synthetic Characters", Conference proceedings on Human factors in computing systems (CHI) 99 - Pittsburgh, May 1999.

[24] Kirsh D. and Maglio P. "On Distinguishing Epistemic from Pragmatic Action", Cognitive Science, Vol. 18, pp. 513-549, (1994).

[25] Murakami T., Hayashi K., Oikawa K. and Nakajima N.,"DO-IT: Deformable Objects as Input Tools", Conference on Human Factors in Computing Systems (CHI) 95, Conf. Companion, Denver, 1995, 87-88.

[26] Norman D. A., *The Psychology of everyday things*, BasicBooks - HarperCollins, pp. 87-104.

[27] Online: http://www.cyberelk.demon.co.uk/parport.html

[28] Online: http://www.hasbrointeractive.com/hi/product.cfm?product=99154

[29] Online: http://www.oip.gatech.edu/imtc/html/haptic_lens.html

[30] Sierra Studios, "The Official Half-Life web-site", http://www.sierrastudios. com /games/half-life/

[31] Suzuki H. and Kato H., "Interaction-Level Support for Collaborative Learning: AlgoBlock - An Open Programming Language" Online - http://www-cscl95.indiana.edu/ cscl95/suzuki.html

[32] Underkoffler J. and Ishii H., "Illuminating Light: An Optical Design Tool with A Luminous-Tangible Interface," Proceedings of CHI 98, April 1998.

[33] Ward S., Abdalla K., Dujari R., Fetterman M., Honoré F., Jenez R., Laffont P., Mackenzie K., Metcalf C., Minsky M., Nguyen J., Pezaris J., Pratt G. and Tessier R. "The NuMesh: A Modular, Scalable Communications Substrate". Proceedings of the International Conference on Supercomputing, July 1993.

[34] Xerox Park, Online: http://www.parc.xerox.com/spl/ projects/modrobots/DigitalClay/digital_clay.htm