8-2010

CREATING TOUCHPANEL GRAPHICS FOR CONTROL SYSTEMS

Lucas McDaniel
Clemson University, lucasmcd@gmail.com

CREATING TOUCHSCREEN GRAPHICS FOR CONTROL SYSTEMS

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Fine Arts
Digital Production Arts

by
Lucas Ramsay McDaniel
August 2010

ABSTRACT

More often than system designers would like to admit, a discrepancy lies between the implementation of audiovisual control systems and their apparent ease of use to a novice or casual user. System designers and programmers are often hampered by the software tools provided by industry manufacturers and cannot reliably create desirable graphical interfaces that match the level of systems they are asked to program and install.

Popular consumer trends in portable touchscreen devices, pioneered on devices such as the Apple iPhone, light a way forward into a new era of elegantly solving the audiovisual control system graphical user interface problem. Since expensive specialized hardware can be replaced by readily available consumer devices and a wide variety of tools exists with which to create content, possible alternatives to the current methods of designing the graphical user interface for the audiovisual system are ripe for discovery.

Using the latest release of Autodesk Maya 2011, with features such as Python and Pymel, we have developed scripts to generate graphical user interface content for use with audiovisual control systems hardware. Also explored is the potential for a standalone development environment such that audiovisual designers and programmers are not required to operate Maya or adjust scripts to generate content. Given this new level of control over the graphical user interface, coupled with the flexibility of the control system central processor programming, a truly powerful, intuitive, and groundbreaking control interface can finally be realized.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF FIGURES

List of Figures (Continued)

Figure Page

CHAPTER ONE

INTRODUCTION

The Advent of an Interoperable Touchscreen

Control systems brought to market by worldwide leaders AMX and Crestron [HLC01] have dominated the professional audiovisual, broadcast, presentation, video-teleconferencing, and film industries for nearly thirty years. As early adopters of touchscreen technology, these two companies have strived to develop hardware and software that make complicated electronic systems more manageable and easier to use. As computer technology has evolved, we now see such technology affordable by a much broader portion of consumers in the United States and across the world.

In present day 2010, Apple Inc. has brought a series of products to market providing an elegant and affordable way to expand and improve on this two decade old concept of tightly integrated remote control, paving the way for newly developed applications to offer alternative and possibly groundbreaking approaches to the design and programming of control systems. Previously, the programmer had no choice but to rely on the in-house graphics development software developed by control systems manufacturers. Though not much more than Microsoft Paint [wiki01], they were the only way to develop and upload the graphical content for the touchscreens. Such programs, it should be noted, were also exclusive to their product lines. This situation prevented cross-platform support of the developer's interface files, and it has remained that way through the years to discourage the use of the competing product.

The fact that Apple has presented a situation that has forced compatibility in the audiovisual industry is not without a certain amount of irony, seeing as how they are often criticized for exerting too much control over their hardware and software products. It should be noted, however, that similar to what the personal computer did to computing in the 1990s, little known applications of technology have suddenly become much more viable in the current consumer market. Could such easily obtainable and popular products actually cause interoperability between competing hardware manufacturers AMX [amx02] and Crestron in the years to come [rave01]? Given the direction technology is heading, it appears as if this is one of the resulting eventualities [nsca01].

Suffice to say that such camps have been entrenched for a number of years and have developed software tools that still get the work done day in and day out, and many designers will continue using those tools for years to come. However, presented before us now is an entirely new area of graphics design that has a great deal of potential for growth, particularly among more technologically savvy consumers as they rise in numbers, age, and affluence. This research is preparation for what could potentially be demanded: an independent group of control systems graphics tools for independent programmers that serve not only the market giants, but many other smaller or more specialized niche brands of hardware products available on the market.

Graphics Design Tools and Pipeline

Concepts

In subsequent chapters it will be argued that the time spent on implementing content for particular brands' platforms should be replaced with time spent utilizing the incredibly powerful graphics tools available today. Furthermore, content should be designed in an interoperable fashion such that many control system user interfaces can be generated from a central repository of graphics material. Not only would this allow for interoperability, but would also aid in the support of new products and other emerging markets by supporting a larger base of hardware with a minimum of implementation time [MLC01]. With the demand of such hardware and software growing every day, this area could very well be worth exploring, quite possibly for more than purely intellectual pursuits.

For interoperability and a central repository system to function, one-off and piecemeal graphical development, at least in the sense that the industry knows it now, would be a methodology that would need revamping. The core of our study on the topic will rely on relatively simple graphical concepts like icons, text, submenus, short animation cycles, etc. Such attributes are well suited for initial proof-of-concept, bearing in mind that the goal of this work is to focus on original concepts without concern over providing fully operational systems. Given all of these factors, the method most useful for an undertaking of this sort appears to be centered on creating individual scripts to ultimately function as a set of design tools.

The proposed tools would be specially tailored for audiovisual control system graphical user interface requirements, and in a later version, would be accessed by a standalone program that would be used to add and edit existing assets to formulate content for the control system. Though the assets themselves may not vary much from project to project, the precise configurations thereof almost always do, and such a program would be quite useful, particularly if it provided a way to reduce the amount of re-entering of text data so typical in the system design, documentation, and programming process. Examples of possible tools might include those which create and modify geometry, materials, and animated sequences, as well as provide a way to view, frame, and render the sequences while performing basic file I/O.

Implementing a system to work flawlessly and with which the software has the capability of making more robust use of positional data (example: multi-touch gestures for translation, pan, and zoom, as well as accelerometer and compass data for rotation and azimuth) from the hardware device is not entirely out of reach, but it is out of scope of the work presented. Such a prototype would be an important step to a more immersive, higher level 3D environment, and represents an important future goal of this project should it produce favorable results.

Implementation

With the inclusion of Maya's python platform, Mayapy, and the recent addition of Pymel, scripts will be developed to generate graphical user interface files for use with audiovisual control systems hardware. In addition, groundwork will be laid so that with further Python scripting, a fledgling standalone development environment can be deployed. Such an addition would expand the usability of the tool set beyond the developer to other audiovisual designers and programmers with little to no skills in these areas. The result of this work in the standalone application would generate a script that can be run on any installation of Maya 2011 or later, which would process the script's instructions and create the graphics for use in a GUI designer program. Later revisions of this application could attempt to replace the use of third-party GUI designers altogether, with output plugins for common control system formats developed and included along the way.

Once the appropriate tie-in, or "hook" assignments, have been made with the control system's run-time code, a standard procedure in the developmental phase, the graphics created can be run on a multi-touch platform such as the Apple iPad. Given the control systems programmer's central processor code and the feedback to/from the newly developed custom graphical user interface design files, a truly powerful and intuitive custom control interface can finally be realized.

Results

The technical achievements of this research will focus on the mechanics, supporting code, and development environment necessary to allow primitive models, textures, lighting, and camera arrangements to behave in a framework built with the express intent of developing 3D graphic sequences for user interface elements. The artistic achievements will focus on updating and granting greater flexibility during the design phase of software development for a control system programmer, while simultaneously enhancing human interaction and providing a more intuitive look, feel, and curb appeal to the controls of an audiovisual control system. Furthermore, the work will be supported by comparison with the standard approaches: (1) low-level design within the proprietary software applications provided by the hardware manufacturer, and (2) high-level design by method of a third party implementation using standard 2D graphics methods.

Results, including script snippets, screenshots, rendered stills, animated sequences, and their limitations will also be included. Python, once feared and unknown, was found to be both wieldable and powerful in both the Maya script editor, expression editor, and open source integrated development environment (IDE). Finally, for accomplishments, setting up all the dependencies, compilers, interpreters, and syntax highlighters for use with an external IDE proved to be an accomplishment in its own right.

CHAPTER TWO

BACKGROUND

Background information will first be provided on control systems in general and then coupled with automation concepts to show how it applies to the industrial, commercial, and residential market sectors. Before doing so it is also important to take a look ahead, as it appears that some very large changes in this area are coming, some of which might bolster the stance of high quality graphics design for use with user interfaces.

2.1: Technological Trends

AV/IT Convergence

One way to get a glimpse of what is to come by taking a look at recent acquisitions by one of the largest technology corporations in existence, Cisco systems:

Exerpts from: List of Acquisitions [cisc03]:

Scientific-Atlanta, Inc. - November 18, 2005: …a leading global provider of set-top boxes, end-to-end video distribution networks and video system integration…

SyPixx Networks, Inc. - March 7, 2006: …This acquisition will enable Cisco to deliver video surveillance as part of an Intelligent Converged Environment.

Arroyo Video Solutions, Inc. - August 21, 2006: ...network delivered entertainment … across the growing portfolio of televisions, personal computers, mobile handsets, and emerging media capable devices in our increasingly connected lives.

Tivella, Inc. - December 15, 2006: …provider of digital signage software and systems. Digital signage is an emerging technology that has the potential to transform the customer experience and to promote richer communications…

BroadWare Technologies, Inc. - May 21, 2007: …provides a smooth migration path from analog surveillance video to a digital network solution.

DiviTech A/S - June 10, 2008: …Cisco plans to … create an end-to-end platform that offers all layers of digital video management (element, network and service) in a single modular product.

Pure Networks, Inc. - July 23, 2008: … foundation for the development of new applications, tools and capabilities for consumers to use in an increasingly "connected life" at home, at work and on the go.

Richards-Zeta Building Intelligence, Inc. - January 27, 2009: …intelligent middleware technology that enables businesses to integrate building infrastructure … improved efficiencies, greater energy savings and a reduced carbon footprint.

Pure Digital Technologies Inc. - March 19, 2009: …key to Cisco's strategy to expand our momentum in the media-enabled home and to capture the consumer market transition to visual networking.

Tandberg - October 1, 2009: …a global leader in video communications, including a broad range of world-class video endpoint and network infrastructure solutions with intercompany and multi-vendor interoperability.

Starent Networks, Corp. - October 13, 2009: … a leading supplier of IP-based mobile infrastructure solutions targeting mobile and converged carriers … as global mobile data traffic is expected to more than double every year through 2013…

Set-Top Box Business of DVN (Holdings) Ltd. - November 2, 2009: …shares Cisco's vision of a high-performance, scalable and services-rich cable interactive platform extending into every home.

MOTO Development Group, Inc. - May 18, 2010: …develops products and product strategies for the consumer industry…

CoreOptics Inc. - May 20, 2010: … will enable Cisco to equip service provider customers with highly advanced 100 Gigabits per second (Gbps) transmission technology…

The companies listed above comprise over a third of Cisco's acquisitions over the last five years, and over two-thirds of acquisitions in the last three. As audiovisual technology converges with information technology and information technology converges with the management of building systems, graphical user interfaces will become more important to realize the resultant functionality of their interconnection.

2.2: Industrial Control Systems

A control system is defined as "a device or set of devices to manage, command, direct, or regulate the behavior of other devices or systems" [wiki03]. There are many types of control systems, most commonly derived from the mechanical and electrical engineering fields, though currently most of the control techniques themselves are implemented through computers. Logic control systems are Boolean by nature, and were first elaborate electrical relay banks. They were replaced by the programmable logic controller (PLC) [wiki05] and the programmable automation controller (PAC) [wiki06], which are much more rugged, immune to electrical noise, and built to operate in extensive temperature ranges (Fig. 2.1).

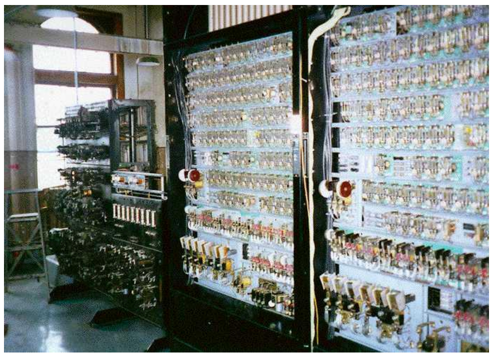


Figure 2.1: Left: Electrical Relay Bank, and Right: PAC with Ethernet and Serial Ports [wiki03] [wiki06]

Another simple control system is the linear negative feedback loop. Such devices use "linear negative feedback to produce control based on other variables with the goal of keeping process within a particular range" [wiki07] (Fig 2.2). A common example of this technique is an automobile's cruise control.

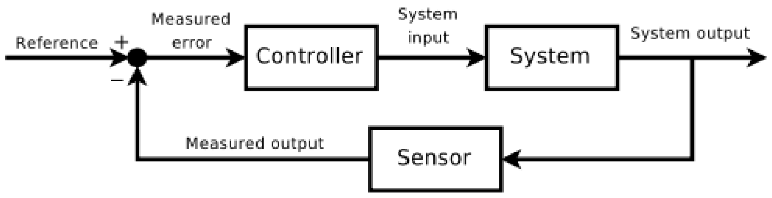


Figure 2.2: Linear Negative Feedback Loop [wiki07]

One general category of control systems is the open-loop control system. Open-loop control systems do not directly make use of feedback, where the process of control is sequenced or prearranged. An alternative to this is the closed-loop method. Closed-loop control systems use feedback in order to regulate certain set values. Such systems are of particular relevance to audiovisual control systems designers and programmers, and would make optimal use of the techniques developed in the results section below.

Another important aspect to control systems originates from a field called control theory. Control theory is “an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems” [wiki07]. Classical control theory deals with single input, single output time domain differential equations, with the most common controller of this type being a proportional-integral-derivative (PID) controller [wiki09] (Fig 2.3). Though most often set up and managed beforehand by means of mathematical calculations, research is currently ongoing in applying graphical user interface technology to the technique [MMM01].

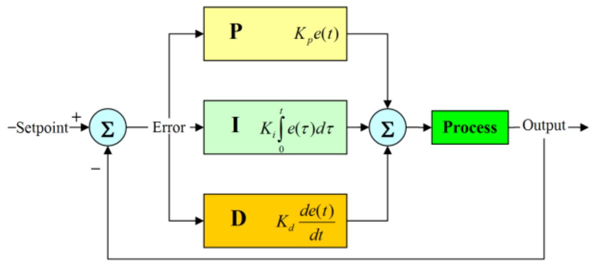


Figure 2.3: Block Diagram of PID Controller [wiki03]

What has become categorized as modern control theory uses multiple input *and* multiple output first-order differential equations defined using state variables, one of the sets of variables that describe the state of a dynamical system. Intuitively, “the state of a system describes enough about the system to determine its future behavior” [wiki08]. An example of modern control theory is the modern-day jet fighter aircraft. The practice of implementing such concepts is called control engineering, or “the discipline that applies control theory to design systems with predictable behaviors" [wiki04]. It should be noted that modern control theory and control engineering as used in "satellites, spacecraft, automobiles, chemical processing plants, and manufacturing rely heavily on the software that is used to implement them" [HWV01]. Heck continues by quoting Boeing and Honeywell engineers who claim that the ratio of software development and control system design is as much as 5:1 [HWV01]. It is because industrial systems are so often designed and built in-house, third-party graphic user interface design is least likely to least be applied here.

2.3: Commercial Control Systems

The following section is a bit more in depth in order to give the reader an idea of the functionality that could be potentially managed by screens developed from the work presented in later chapters. Discussion of commercial audio and video systems, found extensively in this market, will instead be discussed in later sections so focus can be brought upon larger elements coalescing in this sector of the economy.

Functionality provided by a "computerized, intelligent network of electronic devices, designed to monitor and control the mechanical and lighting systems" [wiki11]. in a building is called a Building Automation System (BAS). As a type of distributed control system, a building automation system reduces building energy and maintenance costs. Building Management System (BMS) [wiki10] is a term coined in the 1970s for categorization of complex electronic devices developed for the purposes of managing critical building services by means of reliable RS-485 low voltage control networks tied to one or more building automation systems. The responsibility of the BMS is to control, monitor, and optimize the mechanical, electrical, and plumbing equipment for comfort, safety and efficiency (Fig 2.4).

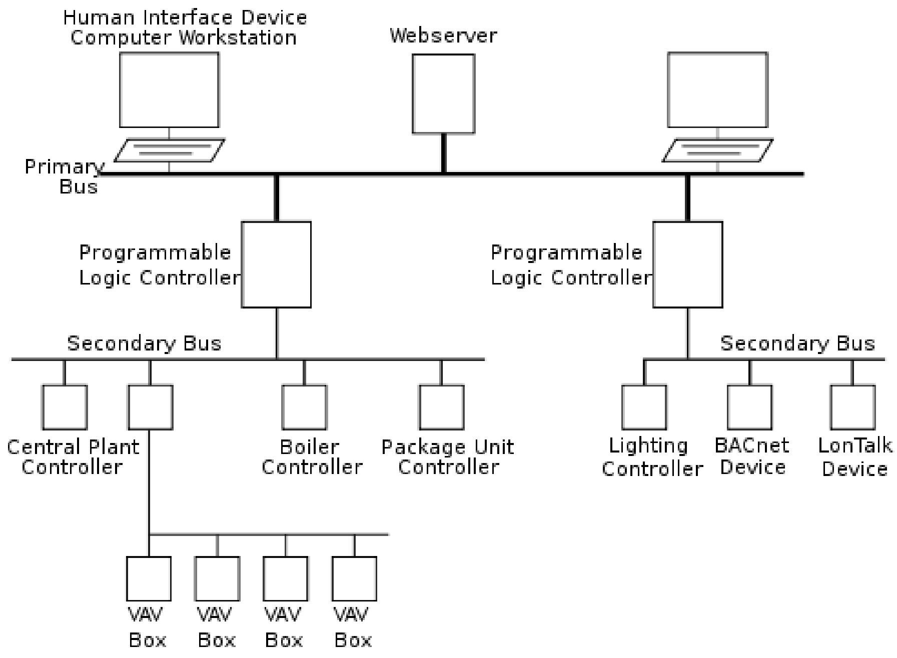


Figure 2.4: Block Diagram of Building Automation System

Recent developments by Cisco Systems and Johnson Controls now provide a way to transmit this data over a typical IT network. With risk managers free of looming concerns like security and critical functionality during network outages, the market is accelerating quickly towards a converged model comprised of integrated networks. As such, Cisco Systems has introduced a term of their own for a top level categorization of BMS called Connected Real Estate (CRE) [cisc02]. In addition to the functions of a BMS, CRE aims to include “IP telephony, unified communications, data center managements, physical security, digital signage, and many more specialized applications” [MCP01].

The data gathered to verify efficiency claims relies on a sophisticated graphic user interface design as noted in [CM01]. This is but one benefit of designing appropriate graphical user interfaces for this environment. Since a BMS can represent over two-thirds of the buildings energy usage, proper configuration of such a system is paramount. Conversely, it is thought that as much as a fifth of the total building energy cost can be attributed to the building management system if not properly configured. If this figure is accurate, this could account for as much as 8% of total energy usage in the United States, which is why the most important prerequisite for achieving Green Building status is its efficient functional integration of a BMS. According to requirements of LEED and other Green Building initiatives, such integration will also create better air quality and water efficiency, not to mention less maintenance required due to the lower life cycle costs of building equipment [loni01].

Furthermore, larger enterprise-level companies may further reduce labor and maintenance costs by the assignment of a single staff to monitor and optimize several BMSs from one remote operations center [loni01]. Here dynamic graphical user interfaces are essential, and it is not unusual that one or more are required. Among the many important items to monitor are the electric power distribution and energy consumption, alarms and faults, magnetic card and access, security and observations, burglar and fire, and lifts and elevator systems.

In order to achieve the energy savings above, building lighting must be included. With occupancy sensors, outdoor photo-sensors, and dimming, cost savings can easily top 50% while extending the life of the bulb, and reducing landfill waste [dain01] (Fig

2.5). Building automation systems further reduce energy costs by controlling when air handlers use chilled or heated water and optimize the mix between return and outside

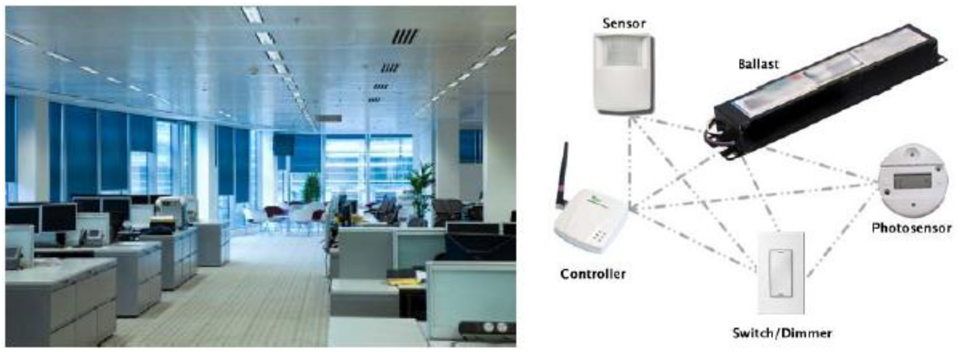


Figure 2.5: Left: Connected Real Estate with Automated Lighting, and Right: Lighting System Interconnection Diagram [dain01]

air for maximum efficiency. Data is also received from the heating, ventilation, and air-conditioning, trace heat, and plumbing system.

Connected Real Estate can also help prevent catastrophic events by means of alarms and notification capabilities. When an alarm is detected, it can notify personnel through an audible alarm and the graphical user interface. Commonly monitored devices include temperature gauges, refrigerant levels, pumps, actuators, and valves, along with ways to monitor carbon monoxide, carbon dioxide, relative humidity and other gases. Often, as in a brief power outage, buildings of this complexity may chalk up a few hundred alarms almost instantly. A well-programmed graphical user interface program would not only treat them in hierarchical fashion, but would also cascade the alerts by retriggering those that still need tending so all alarms can be adequately managed.

Building automation systems are often also used to augment building access control, security, and life safety. In the case of access control, turnstiles and access doors, as well as other security subsystems such as closed-circuit television (CCTV), a BAS can provide several additional data points on which to operate. Fire alarm systems can be designed to override the building automation system and initialize an emergency related program to minimize further outbreak. As of this writing, Cisco, AMX, and Crestron are introducing products that would allow the life safety system to override the digital signage content in a building and thus assist in mass notification in this case or for some other disturbance [schn01].

2.4: Residential Control Systems

Home automation [wiki12] can be quite similar to commercial building management systems, though on a reduced scale in size and critical function. In fact, [loni01] mentions graphical user interface requirements for home automation in mostly the same terms as they do for the BMS they specify. Incidentally, this market would be most likely be the first adopters of the immediate results of the work presented here, and that fact will be reflected in the prototypes presented in subsequent chapters.

It is becoming increasingly more common, particularly among new home construction, to pre-wire it as if it is to become a smart home [smar01], as much of the automation system in a retrofit environment is often using wireless or mesh technology, which is considerably more expensive to implement. Interior design, particularly those designed for upscale urban settings, now find it a requirement to include home automation options in their client designs, due to demand and global trends [java01].

Control options in the home automation market that are not present in CRE could include changing the color of lighting by means of LED fixtures, or involve the additional control of natural lighting, such as window draperies, standard shades, LCD shades, and awnings. Other popular features include audiovisual switching and distribution for multi-zone audio and video, integrated intercom, simulation of presence, medical alert, and also assistance to daily living, as explored in [VDC01].

Other perhaps more novel or eccentric automation (Fig 2.6) is also found here such as houseplant watering, pet feeding and watering, presets for entertaining guests, and the use of domestic robots. Using special hardware, almost any device can be

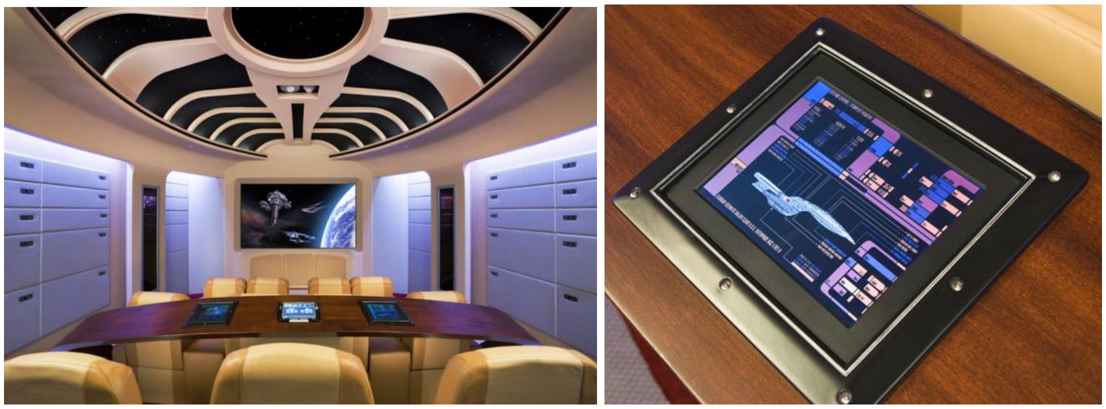


Figure 2.6: Automated Home Theater GUI Example [guif01]

monitored and controlled, such as coffee pots, garage doors, pool and spa amenities, and inventory monitoring via RFID.

2.5: Control Systems Hardware Manufacturers

AMX

The famous Apple origination story starts out in a garage. AMX, not to be outdone, started out in a garage…with a garage door controller. Since AMX was the first of its kind, so was the integrated control system industry born.

AMX designs and manufactures hardware and software used for room automation in boardrooms, auditoriums, teleconferencing systems, museums, and home theaters, as well as the set of "The Matrix," or wherever users desire touchscreens and traditional remotes to control a complex system of devices easily and intuitively. "Other common uses include entertainment systems, industrial command and control centers, security systems, hotels and restaurants" [wiki13].

AMX has developed a complete line of software tools for programming their products. Most notable is Netlinx Studio, which is set up as an IDE exclusively for configuring and programming for either the legacy Axcess system, or the next generation Netlinx systems. A separate application, Touch Panel Design 4, allows the design of touch panel interfaces.

> "Programming an AMX control system,…involves a rare combination of skills in computer science, programming, user interface design, control systems programming, event-driven programming, audio visual systems technology, customer management, scope management and project management" [wiki13].

Crestron

Crestron Electronics creates similar hardware and software, but implements both in substantially different ways when compared to AMX. Over the years, however, Crestron has become the de facto industry leader, although certain sectors such as government and military are still almost exclusively AMX.

Crestron allocates a significant portion of annual revenues for product research and development, which has allowed them a more diverse line of products and in some cases lower cost offerings. They also have successfully entered other vertical markets while their competition has not. Such success has allowed Crestron to bring over 1000 integrated products to market including media servers and distribution, signal processing, control systems, lighting, climate and shade control, touchpanels, keypads, and handheld remotes [wiki14]. Although AMX is known to be ranked as the best place to work in the US, Crestron now employs three to four times more employees.

Not unlike the Cisco/Johnson Controls strategic partnering mentioned earlier, Crestron has reached out to many audiovisual component manufacturers to facilitate integration with their equipment. The Crestron Integrated Partner Program [cres05] is a coalition of more than 400 manufacturers that are committed to deliver seamless integrated solutions with Crestron products. These programs help promote market development, and support Crestron resellers by proxy.

2.6: Advances In Touchscreen Technology

Touchscreen technology has been available since the late 1970s, but generally has been confined to processing a single touch at a time. Until recently, this technology also required production facilities to make the special materials used in their construction. Differing types of touch surfaces may use resistance, capacitance, or surface acoustic wave touch surface technology. When implemented with these materials, the touch surface is generally restricted to small surface areas.

Recently, a new low cost alternative was demonstrated that not only was much cheaper to construct, but also could process multiple touches at a time and act accordingly. In addition, these devices do not require industrial fabrication facilities like their predecessors and could be used in much larger scales. These new systems rely on infrared optics, and inspired web-savvy tinkerers and seasoned veterans alike [SHMP01]. Fig. 2.6 shows how these approaches work.

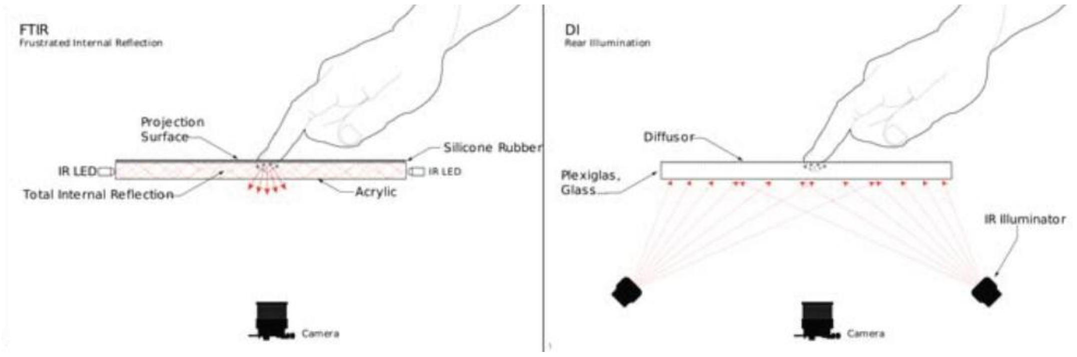


Figure 2.6: General Setup of Left: FTIR System, and Right: DI system [SHMP01]

Frustrated internal reflection (FTIR) beams infrared light through the touch medium. When pressed, the medium refracts the light, allowing some to escape the medium, where it is captured by a camera. The camera’s image can then be processed to reveal the location on which the surface medium was pressed. Diffuse Illumination (DI) is a similar technique; however, in this method the touch surface acts as a diffusor of infrared light shining on the back side of the medium. When the medium is pressed, that area is no longer diffuse, and image processing similar to FTIR allows for the location of the touch to be revealed (Fig 2.7).

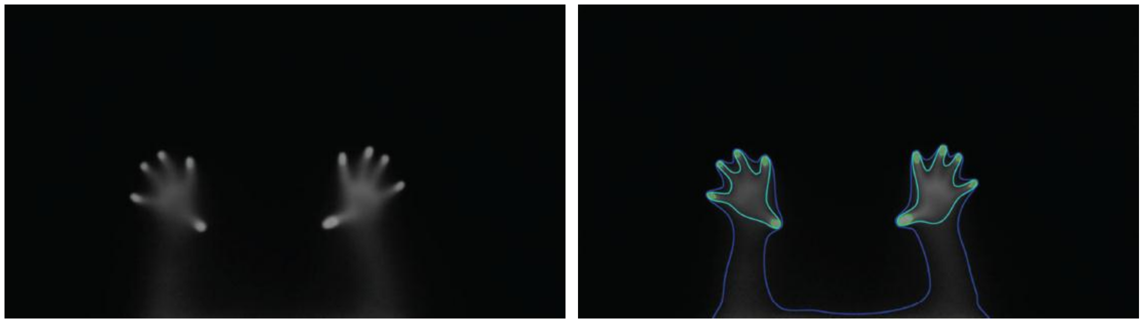


Figure 2.7: Left: IR Camera Image, and Right: Image Data Extracted [THFS01]

Current development of consumer products that incorporate this technology are Microsoft's Surface, a medium-sized table based on DI, the Apple iPhone, and the Apple iPad. In addition, small to mid-sized LCD multi-touch displays based on capacitive sensing are also available from regular hardware vendors, though much more limited in the amount of processing that can be done to attain the multi-touch data" [micr01] [THFS01].

Figure 2.8: Example Implementation of Microsoft's Touch

"Research is already ongoing with multi-touch surfaces such as the iPad also providing orientation data pertaining to acceleration and rotational attributes, such as pitch, roll and yaw" [BCTW01]. One such interaction technique implemented on an iPhone/iPod Touch was used for navigation tasks in a CAVE virtual environment. "We performed a pilot study to measure the control accuracy and to observe how human subjects respond to the interaction technique on the iPhone and iPod Touch devices" [KGMQ01]. In the mining industry, mining equipment is surrounded by various sensor networks that can provide feedback to the interactive display, allowing use of the equipment remotely and safely [BCTW01].

2.7 Graphical User Interface Development

Development

Generally speaking, issues addressed in GUI development usually concern perception, memory, learning, and problem solving. Aesthetics and ergonomics are covered only briefly. Closely held beliefs on the topic often involve such interactive systems, requiring a commitment and understanding of the entire process by the developer in order to be effective. Building a complete system and then spreading the interface over it "like peanut butter" [LRB01] rarely yields useful results.

Avoiding such a destiny requires the involvement of not only the GUI designer, but also programmers, quality control personnel, documentation and training personnel, and others [LRB01]. These assertions leveled in [LRB01] appear to be backed up by data showing that half of the defects are a problem with the code they are interfacing with, be it data manipulation, sequential processing problems or otherwise. In addition, as much as a one in five chance exists that using the GUI will result in a system crash. In order to avoid such errors, graphical user interface design should involve the entire development team as early as possible [BRM01].

Design

Early concepts for the results of the work presented here focus on the procedural nature of camera movement, and how a 3D animated touchscreen GUI might function. In the early phases of this project, which included the concept of a virtual reality experience on an Apple iPad, rendering iterations of every possible camera movement from multiple start and end points seemed like an interesting problem to solve. The related work presented below is similar, except that the animated object moves about the scene procedurally. A video of the result of this work can be found at [KL02], and is notable for its time. [KL01] notes that a concept used in artificial intelligence research was necessary to mitigate the factors of event-driven animation where operational sequences were required. This allowed the computer to decide exactly how to proceed from the initial animation state to the goal animation state.

A necessary tool for the animated interface designer should keep track of the requirements built into each start/goal sequence and verify the integrity thereof. In the conceptual AI planning flow chart (Fig 2.10), this function is performed by the Player Animation Controller. Such a concept is certain to play a factor in the work presented here, and has been noted for its usefulness.

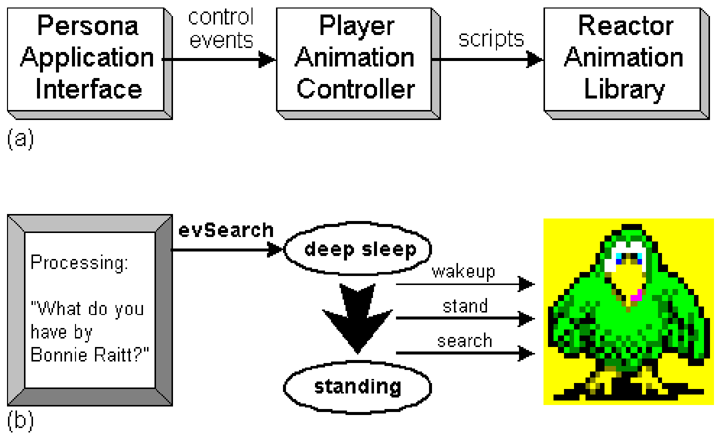

Persona Application Interface
control events
Player Animation Controller
scripts
Reactor Animation Library
(a)
Processing:
"What do you have by Bonnie Raitt?"
evSearch
deep sleep
wakeup
stand
search
standing
(b)

Figure 2.10: (a) AI Planning Flow Chart, and (b) Example Implementation [KL01]

Another eventuality is the concern with not yet (or partially) executed commands sent to the animation controller when a new one is received, and deciding which takes precedence. This was solved only by resolving all possible start states to all goal states as expressed in event definitions. As such, the instruction set behaves much like a state machine in the end but retains the flexibility of adjusting, manipulating, and expanding the code further at another time.

The emphasis on event handling as a requirement to break out of static interfaces is developed further in work by [JW01] who, while concerned with such issues, is more concerned with the notion of flow to generate positive affect. Flow is defined as “a state of concentration, deep enjoyment, and total absorption in an activity” [JW01]. Though brought from the study of games to the study of ergonomics, the author argues that such a concept can “inform the development of non-leisure software for positive effect” [JW01]. Another notable concept presented in this work is a specific example in which game theory contravenes non-leisure software applications. The author notes that games “often provide minimal information to the user, they employ context-dependent commands, and they allow the user to make a variety of errors" [JW01]. That is to say if we can make learning by error fun, then we stand more of a chance of success that the customer not only uses the GUI we designed, but also stands a better chance to use it more fully. Other work as shown in [GQMC01] seems to bear this out empirically.

The emphasis on flow and affect as a requirement to break out of static interfaces is developed further in work by [CU01] investigating other cognitive benefits to the user. As such, work presented in [CU01] seeks not to startle the user with unnatural motion of digital absolute or linear state, but to relax and amuse the user with analog motion blur techniques and cartoon animation. Such an approach allows moving objects to move more comprehensibly if attention to timing, transient detail, dissolves, and anticipatory/contrary motion are made. [CU01] argues that the cognitive burden of deciphering the interface can be reduced if flashes and sudden changes are eliminated. The author concludes with "the animation doesn't have to be slow, or distracting, or silly; on the contrary, with careful tuning, cartoon animation can turn the user interface into an understandable, engaging, and pleasurable experience."

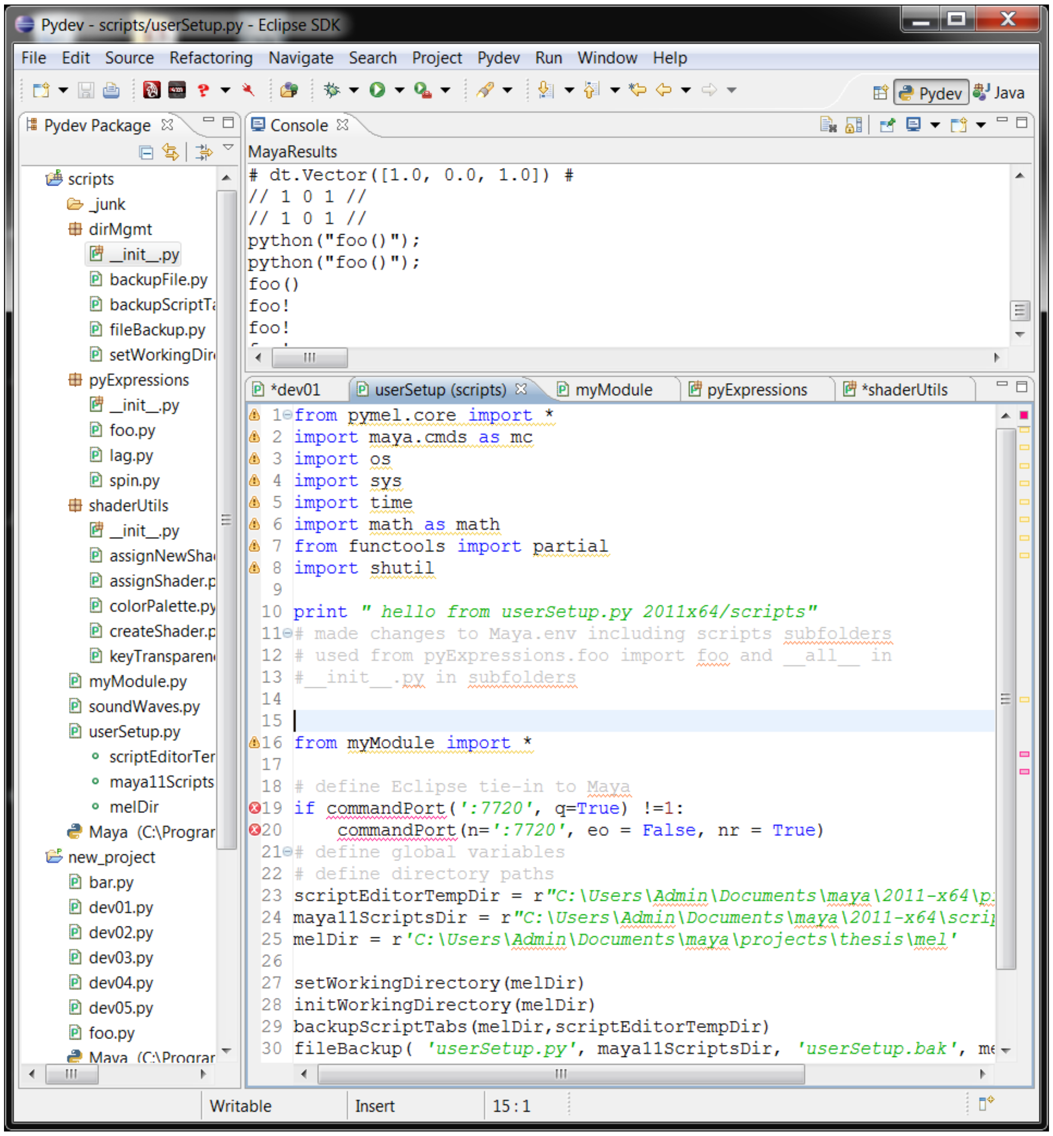
CHAPTER THREE

IMPLEMENTATION

3.1 Building an Integrated Development Environment

The implemented system, though developed primarily as a proof of concept, shows promise. Scripts were generated in MEL, Python, and Pymel, and some examples will be shown later in this chapter. MEL scripting proved most useful for situations that required quick processing. Otherwise, it was more of a hindrance, particularly in building UI windows in Maya, which were used to aid in internal development, not for end user graphical interfaces. Fast renders and calculations were not mission-critical for the project at hand, but ease of comprehending the code and organizing a large number of scripts were.

Multiple tabs in Maya's script editor are certainly an improved feature in the later releases, although developing scripts in the script editor proved to be somewhat unsafe due to concerns over data loss. An external IDE was therefore required for work at this level. IDLE, Wing, and EditPlus IDEs were considered, but Eclipse was ultimately chosen. As an open source alternative running under the Java environment, Eclipse has a modular design that subsumes the power of a myriad of plugins. It also supports all of the major operating systems platforms and has a comfortable workspace (Fig 3.1).



```
# dt.Vector([1.0, 0.0, 1.0]) #
// 1 0 1 //
// 1 0 1 //
python("foo()");
python("foo()");
foo()
foo!
foo!
```

```
from pymel.core import *
import maya.cmds as mc
import os
import sys
import time
import math as math
from functools import partial
import shutil

print " hello from userSetup.py 2011x64/scripts"
# made changes to Maya.env including scripts subfolders
# used from pyExpressions.foo import foo and __all__ in
#__init__.py in subfolders


from myModule import *

# define Eclipse tie-in to Maya
if commandPort(':7720', q=True) !=1:
    commandPort(n=':7720', eo = False, nr = True)
# define global variables
# define directory paths
scriptEditorTempDir = r"C:\Users\Admin\Documents\maya\2011-x64\p
maya11ScriptsDir = r"C:\Users\Admin\Documents\maya\2011-x64\scri
melDir = r'C:\Users\Admin\Documents\maya\projects\thesis\mel'

setWorkingDirectory(melDir)
initWorkingDirectory(melDir)
backupScriptTabs(melDir,scriptEditorTempDir)
fileBackup( 'userSetup.py', maya11ScriptsDir, 'userSetup.bak', me
```

Figure 3.1: A Screenshot of the Project Workspace in the Eclipse IDE

Complimentary plugins to the Eclipse IDE include a plugin that talks with Mayapy and subsequently the script editor (com.myplugin.eclipseMayaEditor_1.0.0), and provides syntax highlighting, for all three available scripting languages.

Eclipse presents a wide variety of tools useful for script development. Also available are open source repositories and version trackers such as Bazaar, Mercurial, and

Git that have plugins available for this IDE, which could prove useful if work expands beyond a single developer. Trac and Redmine are examples of bug-tracking software that is freely available for Eclipse, and there are several wiki environments for help documentation publishing that are available as well (e.g., Moin Moin and Zwiki).

3.2 Prototype Development

Several early prototypes were developed, and the first expressions were written. A lamp for lighting controls and a ceiling fan for climate controls were used with primitive GUIs built in Maya to simulate off, warm up, on, cool down, and off states. The ability to manage all different sorts of lights, fans, speaker zones, and other assets representing other control categories, quickly became a concern. From a scripting perspective, importing them to the proper spatial location within the scene, for example, or managing the appropriate assets for control system emulation, brought the focus back to learning more powerful ways of performing such tasks with Python. Pymel was tapped for its strengths in Maya GUIs, working with node attributes, and making sophisticated lines of code easier to read. With Pymel 1.0.0, now shipped with Maya and included in the help documentation, an even more potent scripting language for Maya is available for the deployment of the developer UI.

The help documentation tightly couples Pymel and Python, allowing the discovery of many useful concepts, including functions and methods for passing arguments between them, and ways to gather data from Maya GUI attributes. Concepts explored in this manner spurred improvements in the way Python scripts were sourced and organized, which allowed code to be used, re-used, and tracked more easily. Fortunately, in rewriting the core scripts in this new hierarchy, important overarching code development concepts were incorporated such that the initial design could more realistically be achieved.

3.3 Conceptual Refinement

As these development guidelines were advanced, it became more apparent the kinds of questions that should be answered as preliminary steps to achieving the initial design concept. Making an audiovisual GUI fluid and fun to use will test many of the necessary parts and provide results thereof.

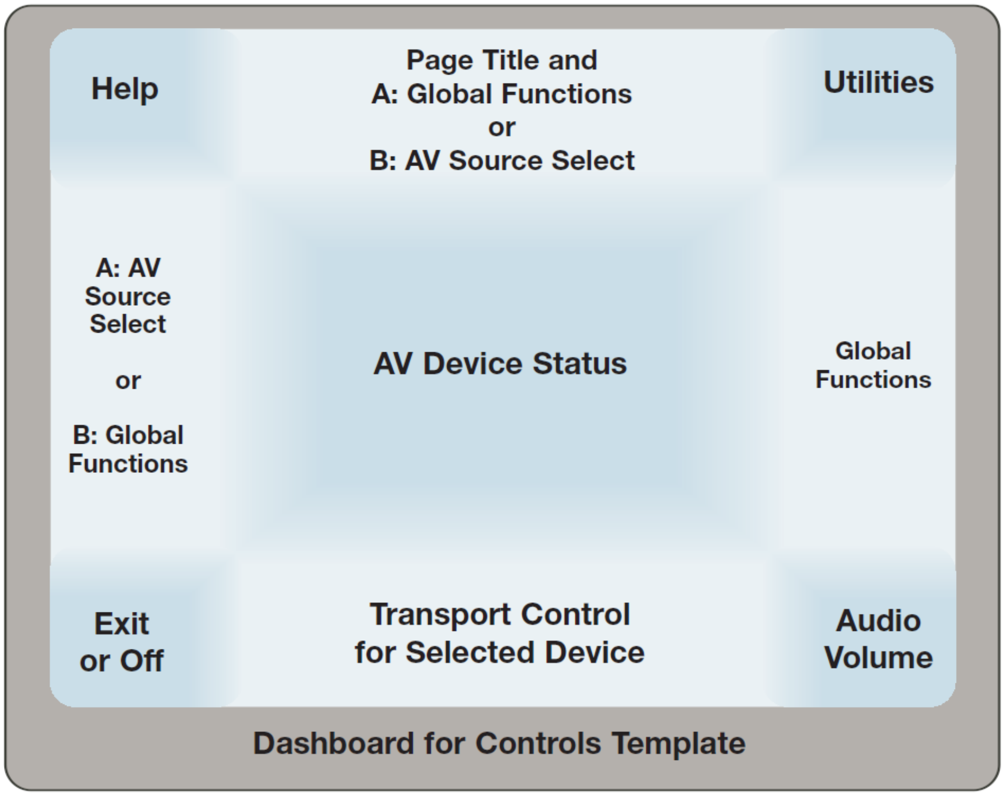


Figure 3.2: Dashboard for Controls Template [BCFG01]

By focusing on concepts of application that are more similar in nature to what is currently used as an industry standard, a basis is constructed to provide solid data points from which to strive and measure against, though concepts investigated are markedly different from current practice and are not without possible limitations.

Key among such limitations is how many frames per second the multi-touch device can handle. This issue is important in selecting from icon-based menus, and simulating browsing through them in a multi-touch fashion by spinning through them quickly. The ideal implementation would incorporate a variable frame rate applied to an image sequence such that multi-touch input would feel more interactive. For example, consider a spinning selector wheel. Allowing for a range in playback speed of 24–72 frames per second could assist in the feeling of immersion or potentially that of cartoon animation. Another potential problem could be multiple select menus or otherwise animated items operating at the same time.

Though each frame in an image sequence is stored locally on the device, they are triggered remotely by the control processor. Care would need to be taken in how this transaction occurred over time and over a wireless connection. Depending on the panel device's use, there is potential for the need of sending hundreds of events per second in order to trigger two or three sets of image frames as they are updated simultaneously.

CHAPTER FOUR

RESULTS

4.1 Standard Approaches

Though there are ways to use image sequences in Crestron's VT-ProE and AMX's TPD4 programs, it is a very cumbersome process. Thankfully, third party offerings allow the GUI designer to sidestep this problem while simultaneously granting us more general flexibility in how we tie-in to the control system. Currently there are no known GUI editors that are tailored for what we are striving to accomplish, though a few at least support multiple brands. This is an ideal place to start prototyping, however, in order to get the best support possible for a multi-touch device and to get the most out of the design. Another benefit is the ability to test the design on as many platforms as possible as early as possible, to avoid any pitfalls during deployment.

Our results will be tested on an Apple iPad (Fig 4.1) once we are in the production phase, due to its larger multi-touch surface and 9.7-inch screen. Though the iPhone (Fig 4.2) or iPod Touch will provide a nice range of product to support in the future, for now, proofs of concept are the main focus. To get a visual idea of what the end goal might look like, however, the following few examples present a cross-section of what is in use. First, the most up-to-date are shown in Fig. 4.1 and Fig. 4.2. Both are third-party designs that cost from several hundred to several thousand dollars apiece. Low level designs are shown in Fig. 4.3. These are developed from the GUI design software provided by some of the larger hardware manufacturers.

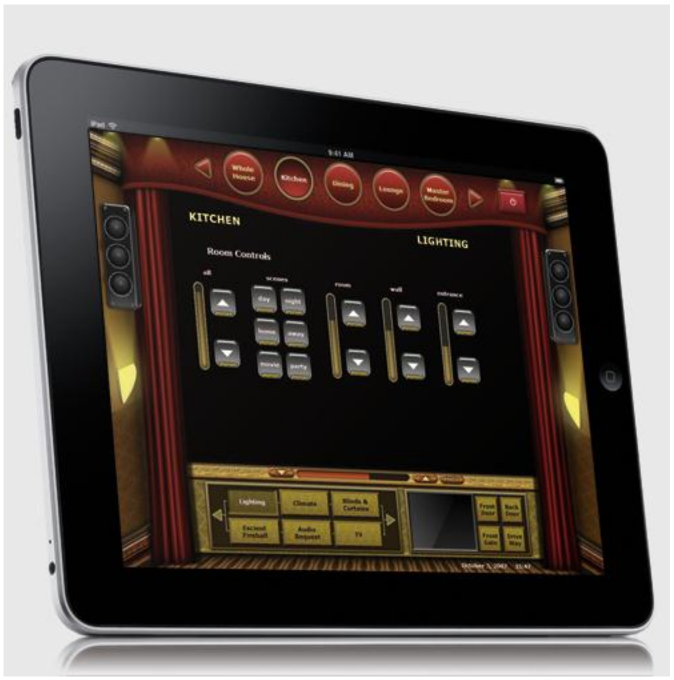


Fig 4.1 Apple iPad With CommandFusion Design [comm02]

Fig 4.2 Apple iPhone With CommandFusion Design [comm02]

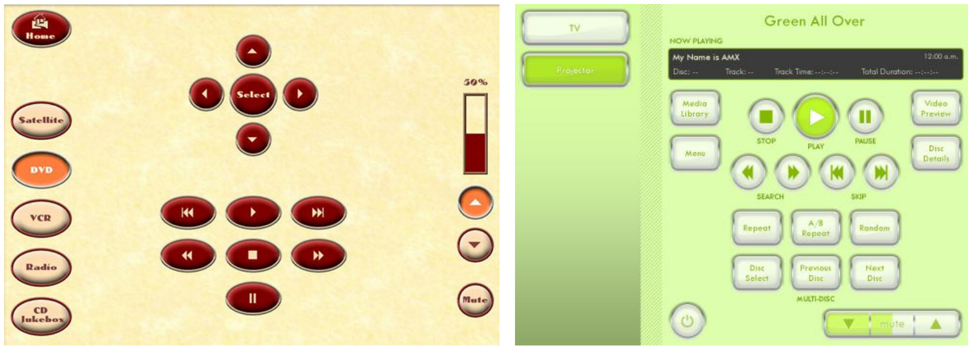
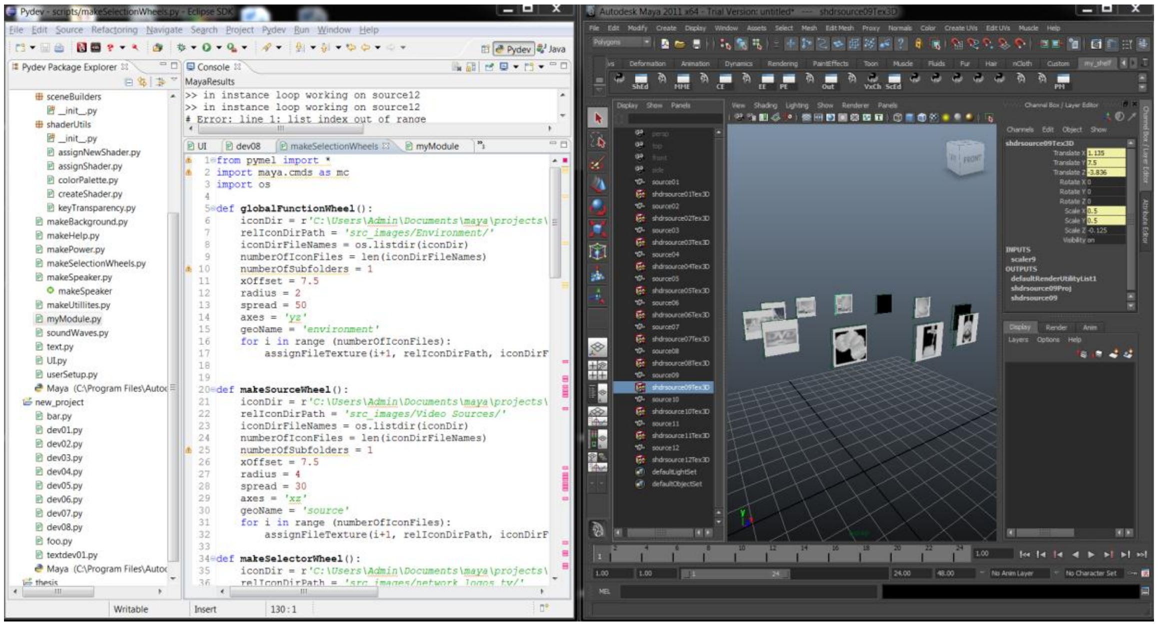


Figure 4.3 Left: VT-ProE Style Design, and Right: TPD4 Style Design [cres01] [amx04]

4.2 Python Scripting



```
>> in instance loop working on source12
>> in instance loop working on source12
# Error: line 1: list index out of range
```

```python
from pymel import *
import maya.cmds as mc
import os

def globalFunctionWheel():
    iconDir = r'C:\Users\Admin\Documents\maya\projects\
    relIconDirPath = 'src_images/Environment/'
    iconDirFileNames = os.listdir(iconDir)
    numberOfIconFiles = len(iconDirFileNames)
    numberOfSubfolders = 1
    xOffset = 7.5
    radius = 2
    spread = 50
    axes = 'yz'
    geoName = 'environment'
    for i in range (numberOfIconFiles):
        assignFileTexture(i+1, relIconDirPath, iconDirF


def makeSourceWheel():
    iconDir = r'C:\Users\Admin\Documents\maya\projects\
    relIconDirPath = 'src_images/Video Sources/'
    iconDirFileNames = os.listdir(iconDir)
    numberOfIconFiles = len(iconDirFileNames)
    numberOfSubfolders = 1
    xOffset = 7.5
    radius = 4
    spread = 30
    axes = 'xz'
    geoName = 'source'
    for i in range (numberOfIconFiles):
        assignFileTexture(i+1, relIconDirPath, iconDirF

def makeSelectorWheel():
    iconDir = r'C:\Users\Admin\Documents\maya\projects\
    relIconDirPath = 'src_images/network_logos_tv/'
```

Fig. 4.4 Creating a Source Selection Wheel

One of the principal ways to implement the selection process on a multi-touch screen is to use its ability to track finger movements and turn them into vectors. By dragging a finger across an area of grouped source icons, the user cannot only quickly make a selection, but also have persistent vision of others in the group that can be selected.

The script for this process (Fig. 4.4) makes use of Python's powerful "os" module, whereby we gather information from the system to count the number of sources by finding the number of icons in the source selection images folder. This information is passed into our main constructor function along with other useful parameters such as the button group radius, the two axes to build them around, and the spread distance between the buttons around the perimeter. Another important Python task is gathering all the names of the icon images and calling them one by one each time a new button is built. Using a new shader, projection node, and file input node (Fig. 4.5) for every button graphic allows for many interesting effects much later in the design process.

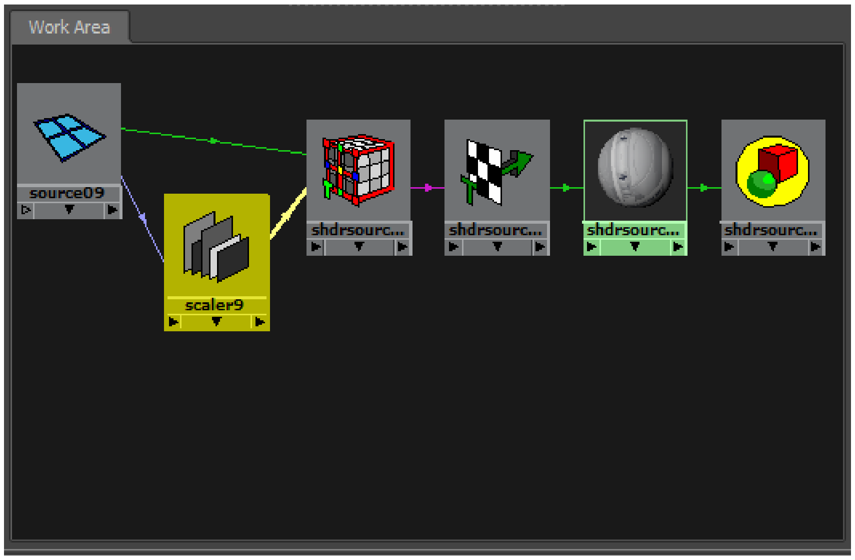
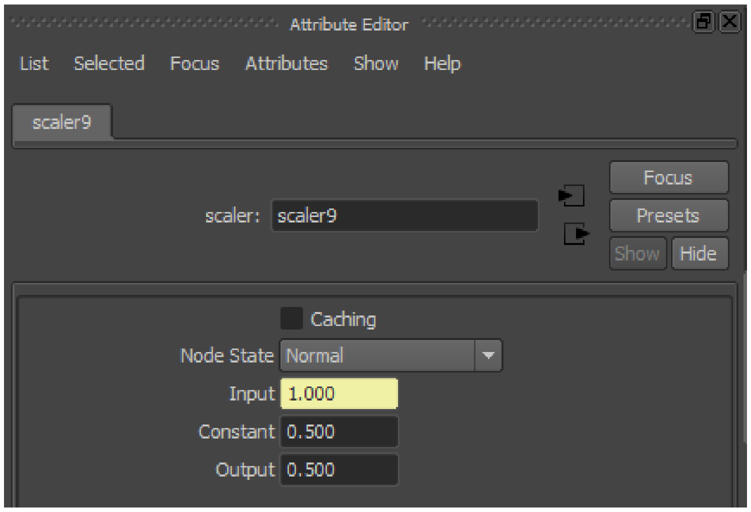


Figure 4.5: A Python Script-Generated Shader Network



scaler: scaler9

Caching

Node State Normal

Input 1.000

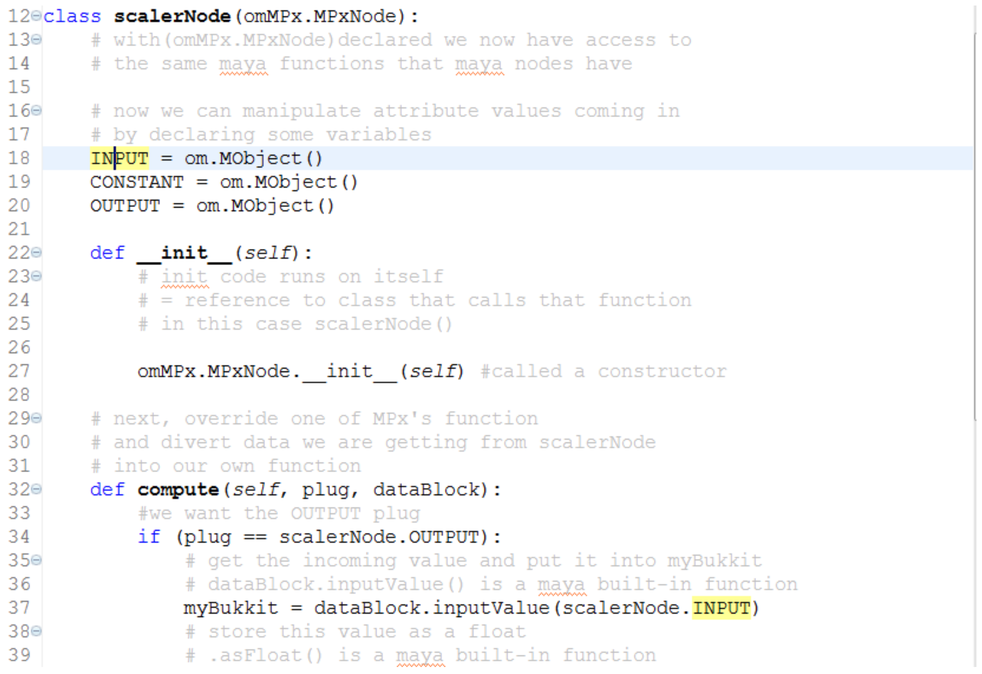
Constant 0.500

Output 0.500

Figure 4.6: Custom Node Attribute Fields

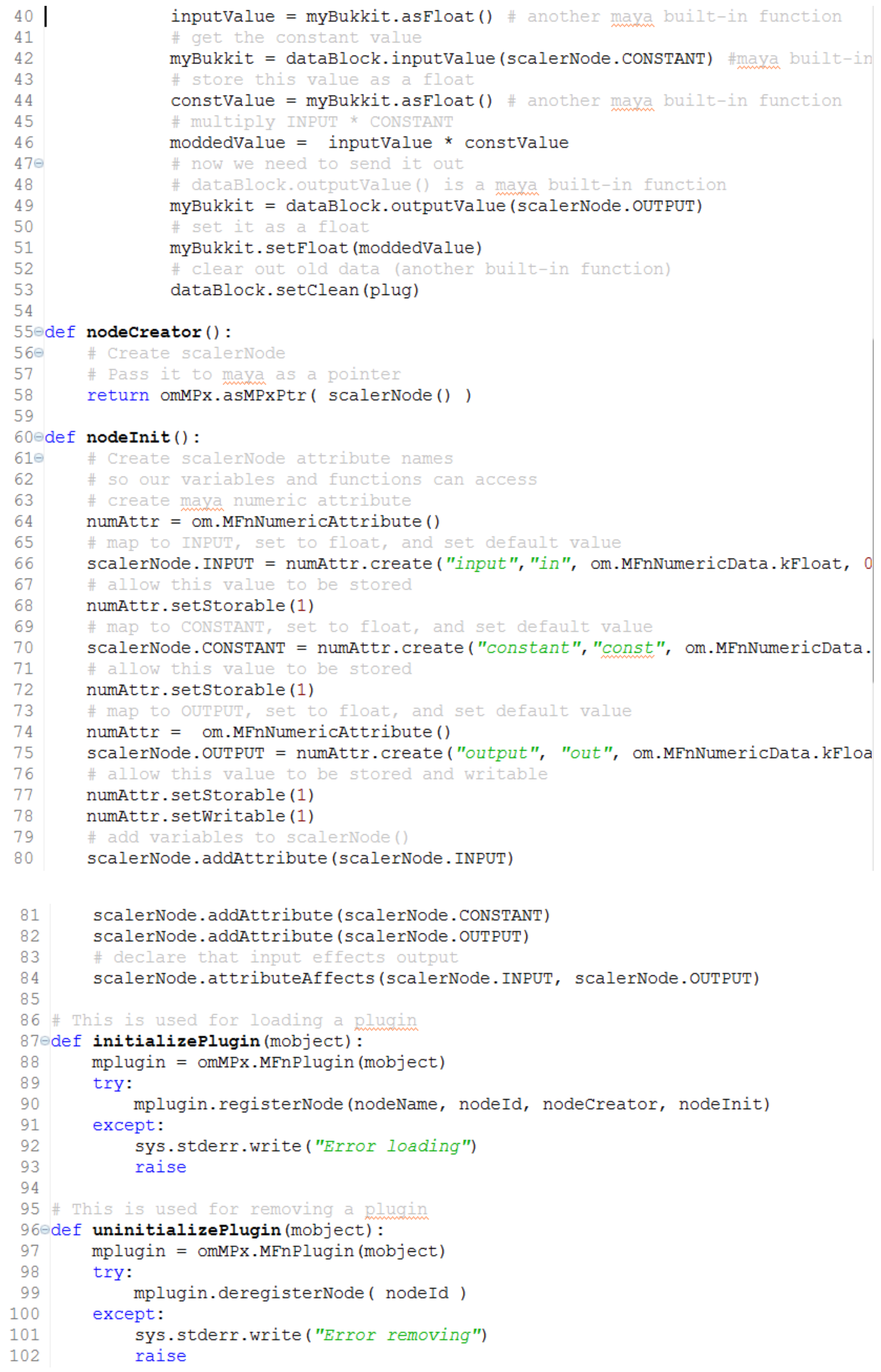
4.3 Python API Scripting

Also on display is a custom-built Maya node called “scaler” (Figs. 4.7 and 4.8), which was built using the Maya Python API. Previously, only C++ programmers could implement the API to great effect, but Python makes this important facet of Maya much more approachable. In the example above, we take scale data from the geometry’s transform node, and feed it into the texture placement node by first routing it through the scaler node. Inside, the data is modified to ensure that the placement node is always the correct size, no matter how large or how small the geometry becomes (Fig. 4.6). Each button goes through this entire process, and has been tested to function beyond 275 icons in one directory. This should prove useful when selecting from satellite radio channels, hundreds of cable or direc-TV channels, or even from a digital music collection.



```
class scalerNode(omMPx.MPxNode):
    # with(omMPx.MPxNode)declared we now have access to
    # the same maya functions that maya nodes have

    # now we can manipulate attribute values coming in
    # by declaring some variables
    INPUT = om.MObject()
    CONSTANT = om.MObject()
    OUTPUT = om.MObject()

    def __init__(self):
        # init code runs on itself
        # = reference to class that calls that function
        # in this case scalerNode()

        omMPx.MPxNode.__init__(self) #called a constructor

    # next, override one of MPx's function
    # and divert data we are getting from scalerNode
    # into our own function
    def compute(self, plug, dataBlock):
        #we want the OUTPUT plug
        if (plug == scalerNode.OUTPUT):
            # get the incoming value and put it into myBukkit
            # dataBlock.inputValue() is a maya built-in function
            myBukkit = dataBlock.inputValue(scalerNode.INPUT)
            # store this value as a float
            # .asFloat() is a maya built-in function
```

Figure 4.7: scaler.py Maya Node Script (Part 1)



```
            inputValue = myBukkit.asFloat() # another maya built-in function
            # get the constant value
            myBukkit = dataBlock.inputValue(scalerNode.CONSTANT) #maya built-in
            # store this value as a float
            constValue = myBukkit.asFloat() # another maya built-in function
            # multiply INPUT * CONSTANT
            moddedValue =  inputValue * constValue
            # now we need to send it out
            # dataBlock.outputValue() is a maya built-in function
            myBukkit = dataBlock.outputValue(scalerNode.OUTPUT)
            # set it as a float
            myBukkit.setFloat(moddedValue)
            # clear out old data (another built-in function)
            dataBlock.setClean(plug)

def nodeCreator():
    # Create scalerNode
    # Pass it to maya as a pointer
    return omMPx.asMPxPtr( scalerNode() )

def nodeInit():
    # Create scalerNode attribute names
    # so our variables and functions can access
    # create maya numeric attribute
    numAttr = om.MFnNumericAttribute()
    # map to INPUT, set to float, and set default value
    scalerNode.INPUT = numAttr.create("input","in", om.MFnNumericData.kFloat, 0
    # allow this value to be stored
    numAttr.setStorable(1)
    # map to CONSTANT, set to float, and set default value
    scalerNode.CONSTANT = numAttr.create("constant","const", om.MFnNumericData.
    # allow this value to be stored
    numAttr.setStorable(1)
    # map to OUTPUT, set to float, and set default value
    numAttr =  om.MFnNumericAttribute()
    scalerNode.OUTPUT = numAttr.create("output", "out", om.MFnNumericData.kFloa
    # allow this value to be stored and writable
    numAttr.setStorable(1)
    numAttr.setWritable(1)
    # add variables to scalerNode()
    scalerNode.addAttribute(scalerNode.INPUT)
    scalerNode.addAttribute(scalerNode.CONSTANT)
    scalerNode.addAttribute(scalerNode.OUTPUT)
    # declare that input effects output
    scalerNode.attributeAffects(scalerNode.INPUT, scalerNode.OUTPUT)

# This is used for loading a plugin
def initializePlugin(mobject):
    mplugin = omMPx.MFnPlugin(mobject)
    try:
        mplugin.registerNode(nodeName, nodeId, nodeCreator, nodeInit)
    except:
        sys.stderr.write("Error loading")
        raise

# This is used for removing a plugin
def uninitializePlugin(mobject):
    mplugin = omMPx.MFnPlugin(mobject)
    try:
        mplugin.deregisterNode( nodeId )
    except:
        sys.stderr.write("Error removing")
        raise
```

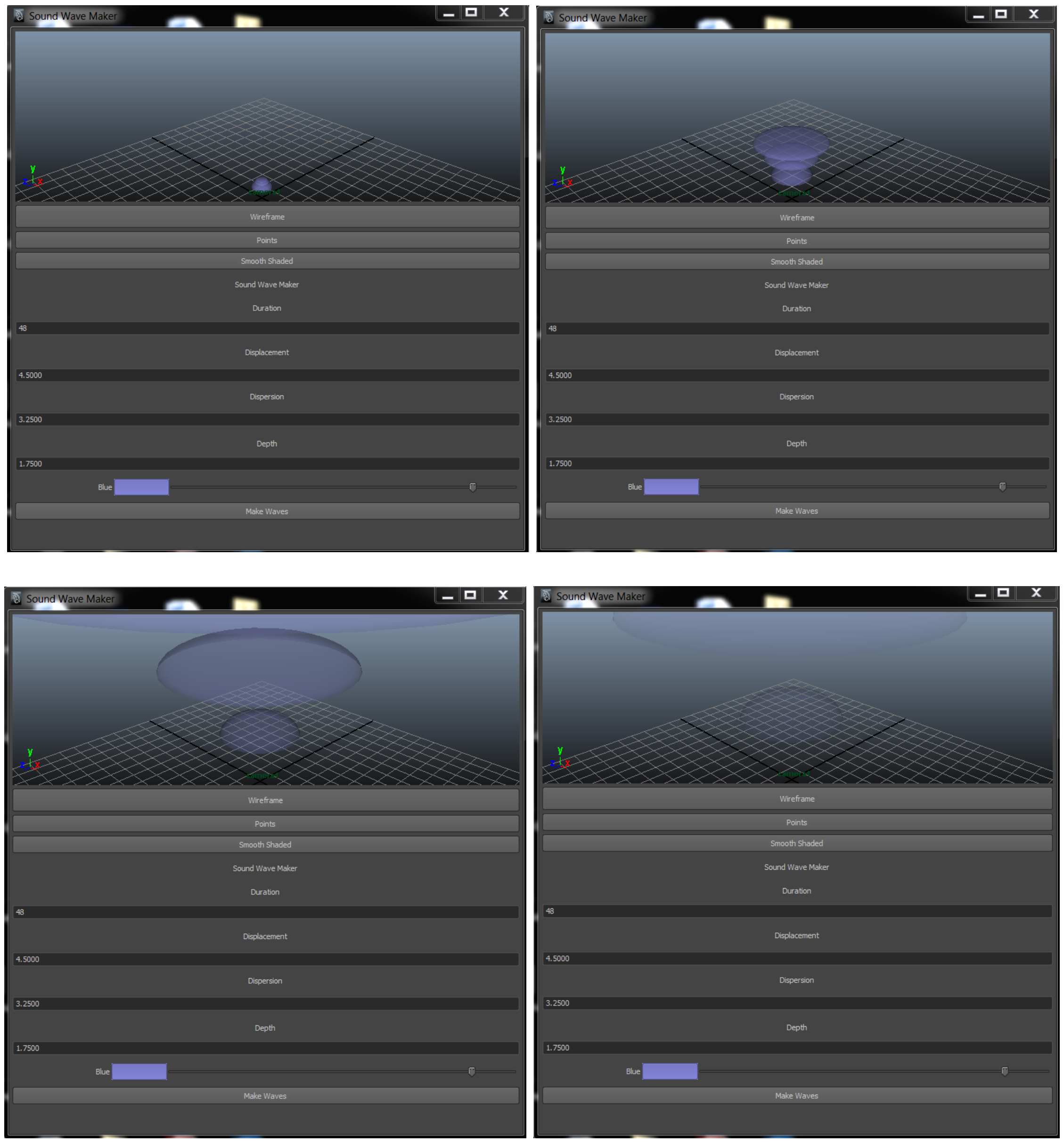
Figure 4.8: scaler.py Maya Node Script (Part 2)

Aside from handling a large amount of shader and geometry data in short order, the script was written such that part of it can become a GUI, allowing for as many differing types of selector wheels to be constructed as needed.

4.4 Pymel GUI Window Scripting

An example of a Maya GUI for manipulating scripts was created when developing environmental elements. In these scripts, focus was placed on gathering data from GUIs, automatically setting keyframe animation, and developing initial shader assignment tools. With commands from the Pymel command set, a window was created that allowed a camera view, some shading options, important attribute adjustment fields, and a color slider (Fig. 4.9). When the “Make Waves” button at the bottom of the GUI is pressed, integer, float, and vector values are sent to a separate function for processing. Once there, the time slider updates and keyframes are set so that the user can search for the correct animation length, shape factors over time, and shader color. At any time, the user can select the parented geometry stack from the outliner, delete, and start over. The number of waves in the animation depend on how many times the “Make Waves” button is pressed before saving out and closing the file or starting over using the above method.

Several scripts work together in order to create this workspace: UI.py, soundWaves.py, createShader.py, assignShader.py, keyTransparency.py, and newShaderColor.py. Each takes in values from other functions and work in concert to achieve the desired effect.



Duration
48
Displacement
4.5000
Dispersion
3.2500
Depth
1.7500

Figure 4.9: Sound Wave Maker GUI

4.5 Python Expressions

Finally, expressions play a large role moving forward. Without them, it would be difficult to animate from a scripting approach. Though MEL may be at its best here by keeping the playback and rendering times down, Python can be used in expressions, boiling down many lines of code in the expression editor by calling a function. The two expressions that were prototyped for this work are called “spin” and “lag.” Both will play an important role in our selection sets of buttons, as they will determine their animation as they get spun around from multi-touch input. The implementation goes as follows: the initial button created has the “spin” expression on it. It is put into the correct spot, and then the next button is made. The second button and all subsequent buttons follow the first one around by means of the “lag” expression. At every frame, the “spin” expression grabs its x and y coordinates from an array, and the “lag” function assumes the previous coordinates from the button ahead in the queue.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

In the final calculus, great care was taken to leverage production-level computer capabilities to address the needs of the audiovisual industry. The result was an environment in which easy-to-use, intuitive, and entertaining multi-touch graphics designs could be developed. Such an endeavor is more feasible in today's environment given the near ubiquitous nature of multi-touch display devices.

Emerging technology and industry trends show the value of such an endeavor. As discussed, various types of control systems benefit from the research results given here, specifically commercial and residential market sectors. Relevant statistics, facts and figures were provided, ranging from AV/IT convergence to standard guidelines for the construction of audiovisual touchpanels, facilitating a vision of near-future graphical user interfaces for control systems, as well as their immediate practical applications.

A major contribution of this work is the creation of a graphics lab from which to create designs as an AMX /Crestron programmer or for potential future business development, which is already generating interest among some industry professionals. By positing that graphics design is no longer up-to-date given current hardware capabilities, an argument can be made for its usefulness in many existing systems that could be up-fitted immediately with such modern graphics via multi-touch display products. The owners of said systems often genuinely want original content for themselves, and are willing to pay a premium for it.

Currently 44% of American homes have 6 remotes in them [OD01], not to mention laptops, PDAs and other gadgets that our current lifestyles embrace. It is the intent that research in this and similar fields will help those who wish to pick one platform for as many technological tasks as possible. The result of such a design principle, made possible by AV/IT convergence, would theoretically leave only one device to charge, one device to carry, and one device to manage. In achieving this goal, the result would be a single device to simplify life's daily demands in an ever more wired world.

A chief improvement to the work presented here would be a way to modify and customize script attributes and other assets for use in new or existing projects without the need of understanding Maya, Python, or Pymel. [APS01] offers a compelling argument here, but aside from laying the most basic of groundwork by selecting a scripting language that has such support in the greater programming community, this goal was beyond the scope of the current work. A program named Qt is a possible candidate for future directions.

Another important part of GUI creation indirectly targeted is the audio portion of the design and the resultant sound effects. A re-examining of the "Peedy the Parrot" [KL02] video with closer attention paid to the audio component reveals how much more effective the graphical user interface experience could be, particularly considering current audio production standards.

Overall this research brought together background in the audiovisual industry, computer programming techniques, and consideration of artistic principles as applied to computer graphics, though many challenges remain.

The actual problem addressed is real and present, and the development of ideas presented here may at the very least serve as a set of options, if not guidelines, when developing control systems graphical user interfaces for the audiovisual industry. Other recent developments in technology will doubtlessly demand refinement of the ideas discussed herein, making additional study not only possible, but likely. [SH01] describes other interfaces such as the “Wiimote” tested for use beyond the Nintendo platform for 3D UI purposes [BCFH01]. Other developments in the field of ubiquitous computing are certain to offer plenty of similar opportunity via work developed in augmented reality [BS01]. Even without these amazing new approaches to user interface design, the goal of this thesis is to provide pertinent information that would make further solutions to the specific problems raised here a worthwhile effort.

WORKS CITED

[amx02] AMX/ Crestron tie-ins to Apple product, http://www.amx.com/ui/apple.asp, August 2010.

[amx04] AMX Touch Panel Design 4 (TPD4), http://www.amx.com/products/TPDesign.asp, August 2010.

[APS01] M. Auer, J. Pölz, and S. Biffl, "End-User Development in a Graphical User Interface Setting," ICEIS, 2009, May 2010.

[BS01] István Barakonyi and Dieter Schmalstieg, "Augmented Reality Agents for User Interface Adaptation," Computer Animation and Virtual Worlds Vol/19 No.1, 2008, Hoboken NJ, USA, 2008.

[BCTW01] T. P. Bednarz, C. Caris, J. Thompson, C. Wesner, and M. Dunn, "Human-Computer Interaction Experiments," 2010 24th IEEE International Conference on Advanced Information Networking and Applications, 2010, Perth, Australia, April 2010.

[BCFH01] D. A. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa, and W. Stuerzlinger, "3D User Interfaces: New Directions and Perspectives," IEEE Annals of the History of Computing, 2008, Washington DC, USA, 2008.

[BCFG01] G. Bronson, T. Cape, A. Faunce, G. Maderic, R. Nimtz, Jr., H. Nunes, R. Remington, and D. Silberstein, "Dashboard for Control Design Guide Template," 2005, Fairfax VA, USA, April 2005.

[BRM01] P. A. Brooks, B. P. Robinson, A. M. Memon, "An Initial Characterization of Industrial Graphical User Interface Systems," 2009 International Conference on Software Testing Verification and Validation, 2009, April 2004.

[CU01] B. Chang and D. Ungar, "Animation: From Cartoons to the User Interface," Sun Microsystems, Inc. Technical Report: TR-95-33, 1995, Mountain View CA, USA, 1995.

[CM01] S. Chien and A. Mahdavi, “Implementation of a User Interface Model for Systems Control in Buildings,” Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, 2009, San Diego CA, USA, July 2009.

[cisc02] Cisco Connected Real Estate, http://newsroom.cisco.com/dlls/2005/whitepaper.pdf, August 2010.

[cisc03] Cisco Systems List of Acquisitions, http://www.cisco.com/web/about/doing_business/corporate_development/acquisitions/ac_year/about_cisco_acquisition_years_list.html, August 2010.

[comm02] Cross Platform Intermediary example, http://www.commandfusion.com/controlsystems, August 2010.

[cres01] Crestron website, www.crestron.com, August 2010.

[dain01] Daintree Networks, “Wireless lighting control saves money and makes sense,” Daintree Networks, 2009, Mountain View CA, USA, 2009.

[GQMC01] G. Golovchinsky, P. Qvarfordt, B. van Melle, S. Carter, and T. Dunnigan, “DICE: Designing Conference Rooms for Usability,” Conference on Human Factors in Computing Systems, 2009, Boston MA, USA, 2009.

[HWV01] B. S. Heck, L. M. Wills, and G. J. Vachtsevanos, “Software Technology for Implementing Reusable, Distributed Control Systems,” Applications of Intelligent Control to Engineering Systems Vol 39, 2009, June 2009.

[HLC01] D. H. Huang, Y. Z. Liang, and W. K. Chiou, “The Practices of Usability Analysis to Wireless Facility Controller for Conference Room,” Proceedings of the 12th international conference on Human-computer interaction, 2007, 2007.

[java01] Interior Design Trends – AV Smart Home Inclusion, http://javabali.info/trend/new-trend-interior-smart-home-decorating-ideas.html, August 2010.

[JW01] D. Johnson and J. Wiles, “Effective Affective User Interface Design in Games,” Ergonomics,” Volume 46, Issue 13 & 14, 2003, October 2003.

[KGMQ01] J. Kim, D. Gracanin, K. Matkovic, and F. Quek, “iPhone/iPod Touch as Input Devices for Navigation in Immersive Virtual Environments,” 2009 IEEE Virtual Reality Conference, 2009, Lafayette LA, USA, March 2009.

[KL01] D. Kurl and D. T. Ling, “Planning-Based Control of Interface Animation,” Proceedings of CHI ’95 1995, Redmond WA, April 1995.

[KL02] Peedy the Parrot, http://kurlander.net/DJ/Videos/PeedyVideo.shtml

[LRB01] C. Lewis, J. Rieman and J. Bluestein, “Task-Centered User Interface Design,” eBook, 2008, Boulder CO, USA, February 2008.

[loni01] Staff Assignment – Central Operations Center for Multiple Campus Buildings, http://www.lonix.com/specifications/IBMS_specification.pdf, August 2010.

[MCP01] J. Martocci, D. Chute, and V. Pothamsetty, “Building Automation System Over IP (BAS/IP) Design and Implementation Guide,” Johnson Controls Network and Information Technology Considerations Technical Bulletin, 2008, Milwaukee WI, USA, August 2008.

[MLC01] J. Meskens, K. Luyten, and K. Coninx, “Shortening User Interface Design Iterations Through Realtime Visualisation of Design Actions on the Target Device,” 2009 IEEE Symposium on Visual Languages and Human-Centric Computing, 2009, Corvallis OR, USA, September 2009.

[micr01] Microsoft Surface, http://www.microsoft.com/surface, August 2010.

[MMM01] T. L. T. Mohamed, R. H. A. Mohamed, and Z. Mohamed, “Development of Auto Tuning PID Controller Using Graphical User Interface (GUI),” 2010 Second International Conference on Computer Engineering and Applications, 2010, Bali Island, Indonesia, March 2010.

[MHP01] B. Myers, S. E. Hudson, and R. Pausch, “Past, Present, and Future of User Interface Software Tools,” ACM Transactions on Computer-Human Interaction (TOCHI) - Special Issue on Human-Computer Interaction in the New Millennium, Part 1, Volume 7 Issue 1, 2000, March 2000.

[OD01] O. Omojokun, P. Dewan, “Automatic Generation of Device User-Interfaces,” PERCOM '07 Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications, 2007, 2007.

[nsca01] AMX/ Crestron tie-ins to Apple product, http://www.nsca.org/Portals/0/Documents/IndustryNews/Projects/20100408-Projects-AMX.pdf, August 2010.

[rave01] AMX/ Crestron tie-ins to Apple product, http://ravepubs.com/index.php?option=com_content&view=article&id=2287:both-amx-and-crestron-announce-ipad-apps-to-turn-ipad-into-touch-screen-homeav-interface-&catid=48:media-recording-distribution-a-control&Itemid=94, August 2010.

[schn01] Schneider Electric, “Wireless Controller Networks for Building Automation,” Schneider Electric, North Andover MA, USA, June 2006.

[SHMP01] J. Schöning, J. Hook, N. Motamedi, P. Olivier, F. Echtler, P. Brandl, L. Muller, F. Daiber, O. Hilliges, M. Loechtefeld, T. Roth, D. Schmidt, and U. von Zadow, “Building Interactive Multi-Touch Surfaces,” Journal of Graphics, GPU, & Game Tools, 2010, April 2010.

[SH01] T. Shiratori and J. K. Hodgins, “Accelerometer-Based User Interfaces for the Control of a Physically Simulated Character,” ACM Transactions on Graphics Vol 27 I.5, 2008, New York NY, USA, December 2008.

[smar01] Smart Home Pre-Wire without component purchase, http://www.smarthouse.com.au/Automation/Sound/P9F9R3W2, August 2010.

[THFS01] J. Teichert, M. Herrlich, B. Walther-Franks, L. Schwarten, S. Feige, M. Krause, and R. Malaka, “Advancing Large Interactive Surfaces for Use in the Real World,” Advances in Human-Computer Interaction Volume 2010, 2010 Bremen, Germany, 2010

[VDC01] D. Vergnes, S. Giroux, and D. Chamberland-Tremblay, “Interactive Assistant for Activities of Daily Living,” From Smart Homes to Smart Care, 2005, Sherbrooke, Canada, 2005.

[wiki01] Wikipedia Entry: Microsoft Paint, http://en.wikipedia.org/wiki/Paint(software), August 2010.

[wiki03] Wikipedia Entry: Control Systems, http://en.wikipedia.org/wiki/Control_systems, August 2010.

[wiki04] Wikipedia Entry: Control Engineering, http://en.wikipedia.org/wiki/Control_engineering, August 2010.

[wiki05] Wikipedia Entry: Programmable Logic, http://en.wikipedia.org/wiki/Programmable_logic, August 2010.

[wiki06] Wikipedia Entry: Programmable Automation Controller, http://en.wikipedia.org/wiki/Programmable_automation_controller, August 2010.

[wiki07] Wikipedia Entry: Control Theory, http://en.wikipedia.org/wiki/Control_theory, August 2010.

[wiki08] Wikipedia Entry: State Variable, http://en.wikipedia.org/wiki/State_variable, August 2010.

[wiki09] Wikipedia Entry: PID Controller, http://en.wikipedia.org/wiki/PID_controller, August 2010.

[wiki10] Wikipedia Entry: Building Management System, http://en.wikipedia.org/wiki/Building_Management_System, August 2010.

[wiki11] Wikipedia Entry: Building Automation System, http://en.wikipedia.org/wiki/Building_Automation_Systems, August 2010.

[wiki12] Wikipedia Entry: Home Automation, http://en.wikipedia.org/wiki/Home_automation, August 2010.

[wiki13] Wikipedia Entry: AMX, http://en.wikipedia.org/wiki/AMX_LLC, August 2010.

[wiki14] Wikipedia Entry: Crestron, http://en.wikipedia/wiki/Crestron, August 2010.