Implementation Strategies for Microsoft XNA Game Development in Academic Laboratory Environments

A Whitepaper describing the rationale, deployment, and operation of the Game Design & Development Laboratory at the Rochester Institute of Technology

Andrew M Phelps
Founding Director & Associate Professor
Game Design & Development
Golisano College of Computing & Information Sciences
Rochester Institute of Technology
Rochester, NY, 14623
http://games.rit.edu/ http://andysgi.rit.edu/

Christopher A Egert
Assistant Professor
Game Design & Development
Golisano College of Computing & Information Sciences
Rochester Institute of Technology
Rochester, NY, 14623
http://www.it.rit.edu/~cae/

Gus D Weber
Academic Developer Evangelist
Microsoft
100 Corporate Woods, Suite 240
Rochester, NY 14623
http://www.microsoft.com/

Abstract

This paper describes implementing an XNA Games Studio (GS) based laboratory at the Rochester Institute of Technology, with specific attention paid to administrative and installation issues as they relate to academic environments. This is in contrast to most of the generally available material on XNA GS, which focuses on non-academic installations. The experiences related in this paper should be transferrable to other institutions, and it is our hope that this information provides a platform of useful material in incorporating XNA GS/XBOX 360 systems at other academic institutions.

1. Introduction

The Game Design and Development (GD&D) program, a new area of study within the B. Thomas Golisano College of Computing and Information Sciences at the Rochester Institute of Technology (RIT), is growing at an astonishing rate since its inception three years ago. Due to this growth in the program, the facilities footprint associated with student support is rapidly expanding. RIT entered the domain of game programming in 2000 [11, 29], with an initial course in graphics for game development. Over the next several years, careful planning led to the development of academic concentrations, the Masters in Game Design and Development, and the Bachelors in Game Design and Development. Most recently, the program has proposed minors for computing as well as non-computing students throughout the institute. The original expectation of the GD&D faculty was to educate a handful of graduate students and a single "cohort" of 30 undergraduates per year. However, the current reality is that the programs have grown to nearly 25 graduate students and an entering undergraduate class of over a hundred students per year. These programs are a strategic focus of the Institute at this time, and have accrued a national reputation of quality: faculty are regularly sought out to speak on the topic of games education, academic program development, and other similar topics [8, 13-15, 25, 28, 32-36].

In seeking to provide an adequate working environment for students studying topics in the design and development of interactive entertainment, a number of issues became apparent. Fundamental areas of study include, but are not limited to, computer science, programmatic design, visual asset creation, software design, testing, human-computer interaction, game design, history of electronic entertainment, interactive media, web design, as well as additional relevant fields. As the needs of the programs escalate, a number of challenges have presented themselves relative to the creation of a suitable laboratory space. This paper is an attempt to describe our process in creating (and now administering) our new flagship facility: the Game Design and Development Laboratory at RIT.

The curriculum for our program is well documented, both online [30, 31], and as a formal program with the New York State Education Department Office of College and University Evaluation [27]. This program of study, by its very nature, pushes the boundaries of both hardware and software in a manner very similar to the commercial games industry. The latest and greatest is the "coin of the realm", new packages are released almost monthly, updates almost daily, and the state-of-the-art is constantly evolving. As part of the educational process, students actively explore such change, embracing fundamental concepts of analysis, adaptation, and dissemination. They explore the changing technology landscape noting how such differences impact the experiences they create. This model clashes with traditional academic computing environments, which tend to "flow" around the academic calendar: updates occur over the summer, installations are put off during the semester, and upgrades occur when grant funding or development monies are accumulated, or when a particular facility is updated by the central administration. Such upgrade and administration paths do not map to the traditional cycles of development environments found in the production of commercial work. These issues are compounded by the fact that many systems administrators that work in academic settings are likely to be unfamiliar with game development as a practice, and may not recognize the needs and best practices of the field at first glance.

2. Desired Laboratory Experience, Functionality, and Community

2.1 Laboratory Goals and Curriculum Requirements

The major focus of our laboratory facility is to provide a platform for game development for both personal computers and console systems. To accomplish this, we use both Intel [16] and AMD [4] processors running Microsoft Windows Vista [22] as a development platform. This same platform also doubles as our personal computer test system. For console development, we chose the XBOX360 [23] due to cost as well as ease-of-use of the XNA GS development platform [24]. In fact, XNA GS allows us to simultaneously develop games for both platforms, albeit with some marked differences in approach in certain instances.

In addition, the platform selection also allows us to support both casual and hard-core development experiences: as students progress through their studies they are subject to a curricular design identified as *scaffolding*, meaning that the curriculum will cycle through topics over and over again, using concepts introduced at an earlier stages of the learning process to motivate difficult and deeper concepts. Scaffolding reinforces prior learning, and when used effectively, can be leveraged to provide more self-direction in terms of guiding academic inquiry. To this end, the development environment must accomodate the following scheme: 1) beginning students explore XNA GS at a small scale,2) intermediate students create entire engines from scratch using DirectX and C++, and 3) advanced students explore XBOX 360 multi-threading with XNA GS and compare it to Windows threading in a variety of environments.

Another important educational mission is to allow students to experiences the differences when developing for personal computing hardware vs. consoles. For personal computers, we felt it was criticalthat the hardware in the lab ***not*** be identical: every year a different subset of the hardware is upgraded to ensure that there is a constant mix of processor types and speeds, various flavors of video cards, monitors, and peripherals from multiple vendors. This parallels the expectations of home computer game players: games must run anywhere on any reasonably configured system. To accomodate this experience, RIT partnered with Alienware [3] (a Dell subsidiary specializing in high performance gaming equipment) to meet its hardware needs based upon configurability, performance, compatibility and service-level agreements, although there is no reason that suitable machines could not be purchased from another vendor or built from scratch.

Another stated goal of the lab was to ensure that it introduced students to professional development tools. In our curriculum, we use both Autodesk Maya [7] and Autodesk 3ds MAX [6], as well as the Adobe Master Collection [2]. These packages have stringent hardware requirements, which were taken into account when selecting the laboratory hardware. In addition, we use several of the Microsoft development envrionments (discussed later in Section 4), which also have stated system requirements, but these tend to be less demanding than the graphics packages in general.

2.2 Administrative Goals for the GD&D Lab

From an administrative point of vew, it was our goal with the new laboratory facility to reduce the overall cost of the lab on an ongoing basis, and to specifically reduce the cost of deployment and staff overhead. The GD&D Laboratory marks the second attempt by RIT at creating a facility for game development – our first facility, the Entertainment Technology Laboratory, was created 3 years ago but was quickly outgrown due to the growth of the academic programs. However several lessons were learned during the development and deployment of the previous lab, which are summarized briefly here:

1) Whenever possible, create a rotational scheme for hardware and software upgrades to balance costs across multiple years rather than having "upgrade years" and "stagnant years". This was in contrast to some local budgeting practices within our institution, but was considered critical to maintaining currency, and from a budget planning perspective. Such planning also made it easier to provide access to a variety of computing configurations based upon parameters such as processor, graphics card, specialty peripherals, and software package choices.

2) Spend time on image deployment techniques and test the redundancy of such a system. Focus in this area (described later in this paper) allowed us to reduce the overall staff-time associated with laboratory setup, and utilize personnel elsewhere for other projects. Likewise, it also makes hardware rotation easier.

3) Plan for hardware failure. This goes without saying, but is often overlooked. In deploying a lab of 64 workstations, each with a computer, dual-monitors, XBOX 360, and associated peripherals, it is a certainty that not all of the hardware would arrive intact. Furthermore, despite proper precautions and purchasing decisions, game development seems to produce greater rates of hardware failure than standard laboratory use. Heat dissipation is always a challenge due to long periods of workstation use as well as high demands placed upon the graphics processing hardware.The XBOX 360s are also subject to hardware failure due to their intensive use. Planning for spare machines, service, and RMA procedures was a critical component to keeping our lab operational.

4) Provide multiple room and workspace configuration options. In learning from our first laboratory design, we specifically chose open spaces in which workstations could be reconfigured with relative ease. It is virtually impossible, in an academic setting, to foresee what projects might be undertaken in just a few years, particularly in an area where the curriculum and faculty interest is simply exploding. Thus, a reconfiguration capabilities were considered paramount by our faculty and students.

5) Provide a secure computer environment without foregoing usability, flexibility, and performance. Students in a development setting require access to a number of settings and files on a workstation that might normally be "locked down" in a more traditional lab environment. Likewise, simply letting students "do everything" on a machine without any form of security scheme was clearly both unwarranted and a gross violation of our institutional policies on computer use [37]. It was a goal of the new design to create a system that was flexible enough to meet the needs of the curriculum as a development space, while still providing added security given that it is utilized by hundreds of students per year.

2.3 Providing a Secure, Yet Flexible, Computing Environment and User Experience

As noted previously, a major goal of the new laboratory was a more secure environment, while at the same time providing increased functionality from our prior working environments. Within the department, we have historically offered two radically different approaches to student use of computing equipment: machines that are tightly controlled in which students have very little privilege, but the labs are publicly available to networks and resources, and machines that offer students full administrative rights, but are segmented away from the network and periodically wiped. For the purposes of providing a game development lab, we chose to "split the difference" between the two extremes, providing some login and monitoring capability, as well as some privileged access such that users could customize their working environment and software settings. Because of this mixed approach to account privilege, we enacted a much more rigorous security scheme than we might normally implement in a segmented laboratory.

To accomplish this, we used a variety of configuration and security options. User accounts at RIT are generally generated in one of two main campus systems: 1) the campus wide LDAP account system, or 2) the campus wide Active Directory system, which inherits from the LDAP implementation. Our department, in turn, runs a second Active Directory server, specifically for our own student body, as our computing resources are not available to students outside our immediate academic programs. In theory, this is ideal for any lab setup, as it means that Windows clients can simply log into the domain, and students can customize their own accounts at will. In practice, in a laboratory setting, this has proven to be somewhat problematic since as students customize their settings and accounts, their profile is re-downloaded to each machine they access at the time of login and must be synchronized as they finish their session. Gaming students, in particular, utilize resources several magnitudes of order greater than general students, especially when considering the content accompanying their game engines. With campus network congestion, and improper use of a student desktop (i.e. leaving large files in profile-based locations), significant issues can occur. We also encountered specific issues with Visual Studio 2005 and 2008. In particular, we had conflicts with configuration settings, additional libraries and headers (DirectX, physics packages, and other libraries/APIs), profiles, and differentiating global changes to all accounts and personal preferences for user accounts. Problems manifested as either settings that would not survive imaging or as unusually long startup times in which Visual Studio always believed it was running for the first time. Although such issues are not insurmountable, they are often difficult to trace and follow by non-gaming system administrators.

Because of these issues associated with AD/Domain accounts, *we chose to implement a single shared "base" account for all students.* This account is configured to automatically log on when the machine boots. Students are then forced to log on individually to the machine using MyLogon [17], a tool that allows Windows authentication without a joining the domain as well as scripted logon against a shared local account. In this fashion, we ensure that users are verified account holders to use the laboratory, and have monitoring access for which user is using a machine during any applicable timestamp, but have the luxury of pre-configuring one account for software use that can be a part of the default image that is replicated to all laboratory machines.

In terms of providing the best "balance" between usability and security, we chose to give the base shared account local user permissions at a Power User level. We experimented significantly with providing users with local Administrator accounts, but have thus far found it to be unnecessary. Power User permissions, while hidden from the default account types in Windows Vista, still exist and are exceedingly viable for our particular setup. We found several packages (most notably Adobe Director and a few elements of the Adobe CS3 Master Collection as well as some versions of Autodesk 3D Studio MAX) that would correctly operate at a Power User level rather than with Administrative access. Unfortunately this necessitated the deactivation of User Account Control within the Windows Vista security environment. These packages, with UAC active, would trigger a prompt requiring a login with administrative rights, but would operate correctly with UAC disabled. We assume that this is an incompatibility with the UAC software hook rather than a true request for administrative rights, as the packages operate as expected when run from accounts with reduced access.

This scheme of quasi-administrative rights is balanced by the use of Microsoft Windows SteadyState [21] technology on each of the workstations. SteadyState is a package that, once installed on a workstation, allows no further modification of a given partition, keeping a 'shadow' of any changes that are made to the drive and then 'snapping back' to the default image upon reboot. By incorporating SteadyState, changes made by students to the lab machines are simply wiped away at reboot – and the machines are once again pristine for the next user. Thus, the use of the globally shared base account is, for all intents and purposes, locked to a completely static environment.

In order to allow customization and user-created file access, the machines are configured to provide two additional partitions in addition to the system partition that is locked with SteadyState. The first is a small (40GB-50GB) working partition that is on the local drive of the machine, mounted as P:\ for public use. This partition is wiped upon user request at boot up, and is intended as the default local working area (the shared user account redirects all references to P:\ such as pictures, music, software default save locations, etc.). The other partition provided to students upon login is the H:\ drive, their home partition, which is mounted at logon as a Samba share on the departmental storage area network.

2.4 The Necessity of Multiple Hardware Sets

Another major goal of the lab, which was alluded to previously, was the support of multiple hardware sets. In terms of our curriculum, this is of paramount importance, as it provides students the necessary environment to explore the differences between vendor implementations and approaches. It is also critical from the viewpoint of creating a portfolio of work, as it allows students to properly test their work against a wide range of hardware – since they never know precisely what a prospective employer might have, or what the presentation environment will be at conferences and trade-shows.

The hardware in the lab was specifically chosen such that it incorporated both Intel [16] and AMD [4] processors at various speeds, and with differing architectures. Likewise, both nVidia [26] and ATI [5] graphics cards were used, with a variety of different models, some in SLI and CrossFire configurations, and some left as dual-card configurations. Each station also consisted of an XBOX360 for console development, and dual monitors for increased screen real-estate (the second monitor uses switchable input to provide a view of both the extended Windows desktop as well as the 360). The individual layout for a single laboratory workstation is shown in Figure 1:

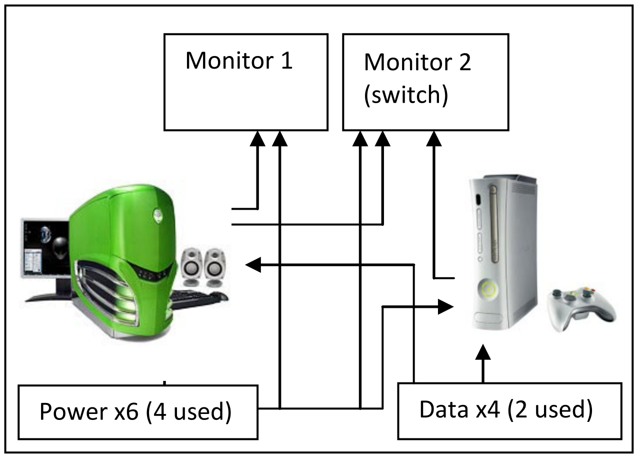


Figure 1: A single workstation (one of 64 such stations overall) within the Game Design & Development Laboratory. Note that each workstation provides both desktop and console development capability, as well as extra power and data jacks for expansion.

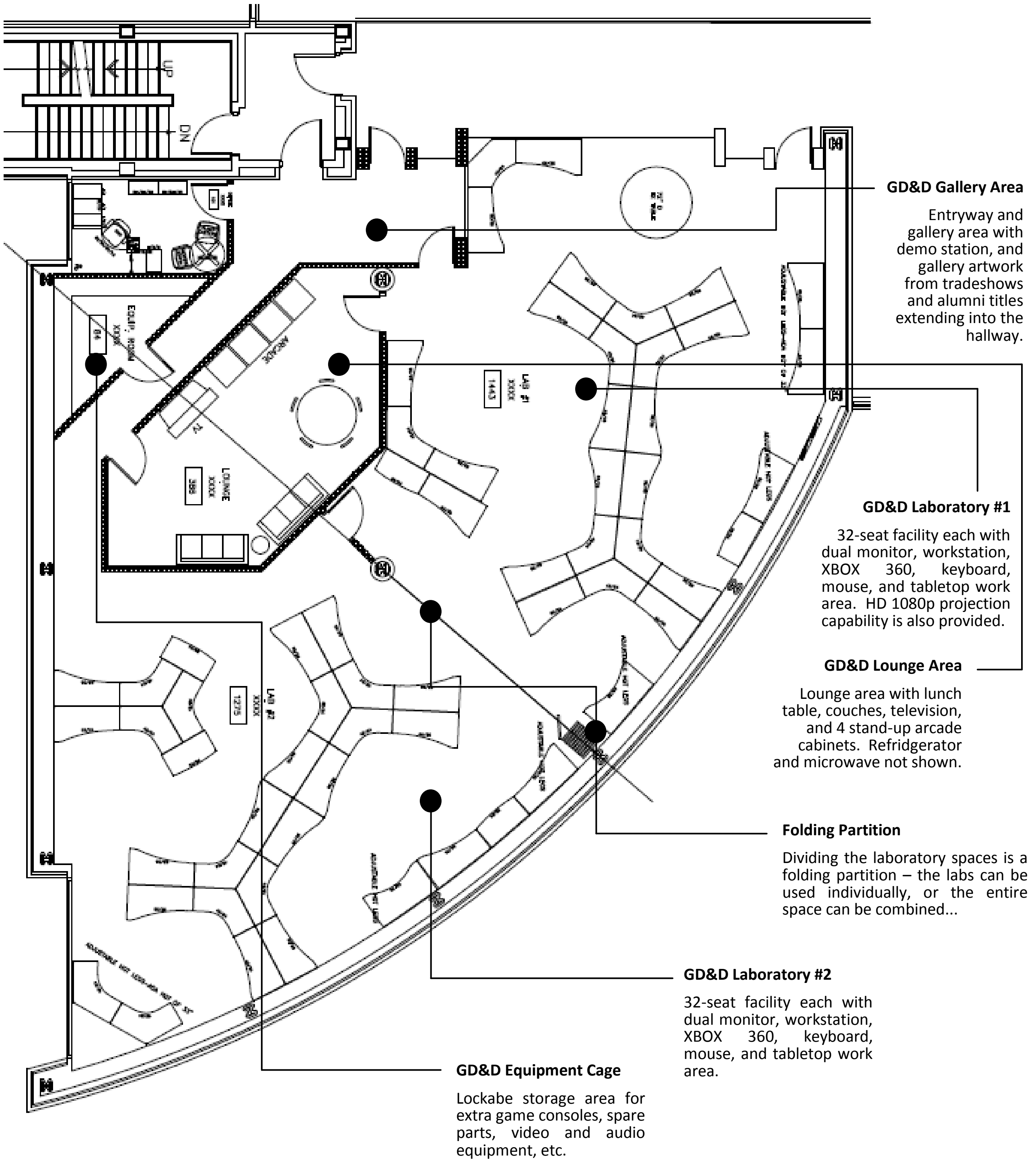
Given this mixed hardware environment, it was a goal of the systems administration group to be able to deploy a single image to the entire lab, despite the various differing system components. We were able to achieve this using a combination of pre-deployment setup work, and a series of post-image-deployment scripts. This is described in greater detail in section 3.3-3.5. It was a further goal to then be able to monitor the use of these various hardware sets and, through analysis, construct a detailed model of how students were using the lab for their development and testing work.

2.5 Creating a Community of Development

The final criteria taken into consideration in the design of the lab was the notion that the lab should not only provide for a suitable work environment in terms of the hardware and software involved, it should provide a focal point for the entire community that surrounds and is formed by our degree programs. Planning for these "community aspects" of the laboratory design centered around two differing aspect: the first being the support of group-work and student development teams, not just individual students, and the second being the acknowledgement and support of student life and social norms as they relate to the laboratory environment.

The first topic, the support of student teams, has been a critical issue in the success of our laboratory environments, and the design of the GD&D Laboratory is a direct implementation of lessons learned when we constructed our previous facility. In several places in our curriculum, students work in teams of anywhere from 3-7 students, and occasionally in larger groups. Given that this most often the size of a development team, desks and workstations were arranged in such a way that there were groupings of stations that supported close collaboration by groups of this size, with shared centralized areas and access to whiteboards, additional table space, and even floor-space as necessary. We also did our best to provide modular capability to the workstations and furniture such that they can be reconfigured with relative ease should a particular team or project need a specific setup.

The second form of community support – the generalized support of the community within the lab space – was the source of some of the less typical elements within the lab. The two elements found in the design of the lab that are in direct support of this goal are the folding wall that divides the larger working area, and the inclusion of a student lounge that is separate, but well integrated, to the overall space. The folding wall allowed us to have the best of two alternative scenarios: at certain points of the day when classes are in the lab, we have two separate work environments, each with a 32-seat capacity. This represents one "section" within the program – any given course taught in the laboratory has a maximum cap of 30 students. Being able to simultaneously have two courses taught side-by-side was critical from a scheduling viewpoint.



GD&D Gallery Area

Entryway and gallery area with demo station, and gallery artwork from tradeshows and alumni titles extending into the hallway.

GD&D Laboratory #1

32-seat facility each with dual monitor, workstation, XBOX 360, keyboard, mouse, and tabletop work area. HD 1080p projection capability is also provided.

GD&D Lounge Area

Lounge area with lunch table, couches, television, and 4 stand-up arcade cabinets. Refridgerator and microwave not shown.

Folding Partition

Dividing the laboratory spaces is a folding partition – the labs can be used individually, or the entire space can be combined...

GD&D Laboratory #2

32-seat facility each with dual monitor, workstation, XBOX 360, keyboard, mouse, and tabletop work area.

GD&D Equipment Cage

Lockabe storage area for extra game consoles, spare parts, video and audio equipment, etc.

Figure 2: The layout and overall design of the Game Design & Development Laboratory at RIT, with major areas identified by function, and notation of specialized features.

NOTE: drawing not necessarily to scale.

Having two smaller 32-seat labs, however, was not ideal from having a singular space which the program could "call home". Despite some additional cost, it was considered paramount that larger events could also be supported in the lab – from collaborative courses that worked together (i.e. a game engine development course scheduled side-by-side with an asset creation and animation course) to large events such as LAN parties and other student-run enterprises. (It is both important and not to be overlooked that these "informal" uses of the lab tend to cement the student experience – use of the lab by the RIT Game Developer's Club and the RIT Electronic Gaming Society for LAN parties, "development days," and general get-togethers have had significant impact on the overall incorporation of the lab into the student experience). Thus, by incorporating a folding partition, the main lab can operate both as two small spaces or one larger, single environment.

The other non-traditional element found in our laboratory is the games lounge, which forms the core of the social identity of the space. The lounge is a place where students can “crash” – eat lunch, check email, play a game. But it is also a place where student teams can dissect existing games, discuss plans for projects away from their workstations, and generally socialize in connection with, but not directly within, the working environment. The lounge is constantly utilized, and provides a casual, irreverent atmosphere that consistently draws students, graduate assistants, and faculty together. There are televisions, game consoles, couches, lunch tables, and arcade cabinets all available for student use, and the lounge also acts as a pass-through environment for all of the various areas that comprise the entire facility.

Finally, the lab also contains a “gallery” area that displays posters and box-art of games that alumni have worked on, as well as various posters from events and workshops in which the Game Design & Development program was represented (annual artwork from the Game Developer’s Conference is one such example). There is also a demo-box in the entry-way to entice anyone entering the facility to check out the latest creations that were wrought in the lab. An overall diagram of the entire facility, with labels describing the major areas and traffic flow, is provided in Figure 2 on the previous page.

3 A Vista-Based Laboratory for Game Development

The design and development of the image and image distribution system for the Game Design & Development laboratory form the core of the entire technological footprint of the lab. To briefly summarize from previous sections, our goals for the image were:

1. To deploy the image in a completely automated fashion from the initial “push” of the image onto the lab hardware all the way through a user-ready machine.
2. To develop and support the image on multiple hardware configurations.
3. To strike a balance between security and customizable usability: specifically to give users “Power User” status at the local level, but to enable drive protection and overall management techniques to respond quickly to security threats.
4. To automatically provide access to an additional public partition on each local machine, and network access to various student resources on the departmental storage area network.

The techniques we researched and developed in support of each goal are presented in the following sections.

3.3 Image creation

The first major change in the formation of the lab image was the very nature of the tools used to create the base operating system image. In the previous incarnations of the lab, under Windows XP, we used a tool called Universal Imaging Utility [9],which allowed systems administrators to add hardware drivers to the image pre-deployment. For our facility, this approach had numerous problems, false starts, and failed elements, due primarily to the newness of several hardware devices, difficulties in detecting certain hardware devices, and initial incompatibilities between the product and service pack 2 for Windows XP. In addition, since our chosen operating system for the new laboratory was Windows Vista Ultimate, we were not sure if the software would present additional challenges so a new approach was both needed and encouraged.

Vista is, in and of itself, an interesting operating system in terms of deployment – the Vista DVD is a live version of the operating system, complete with a localized Hardware Abstraction Layer. The HAL, however is an incredible tool, as any installation of Vista, once it sees a piece of hardware, will “remember” that piece of hardware in the driver cache even if it is subsequently removed from the local system. In this sense, the driver-cache is a “sticky” environment – and this formed the core of our new image development strategy.

Our approach was to create a “crash lab” prior to installing the main lab, which was simply a temporary room with one machine that represented each configuration that would be found in the main lab. In total, there were approximately ten different configurations, and as such the crash lab contained ten machines. We then installed Windows Vista Ultimate onto the first machine from DVD. Following this, the image from the first machine was captured onto a USB hard drive using the CloneZilla [10] drive cloning tool. The image was then deployed to the second machine, again using CloneZilla. When the second machine boots, Vista will detect any additional hardware present, adding the drivers to the cache. The image is then copied and deployed to the third machine, and so on until the image has been

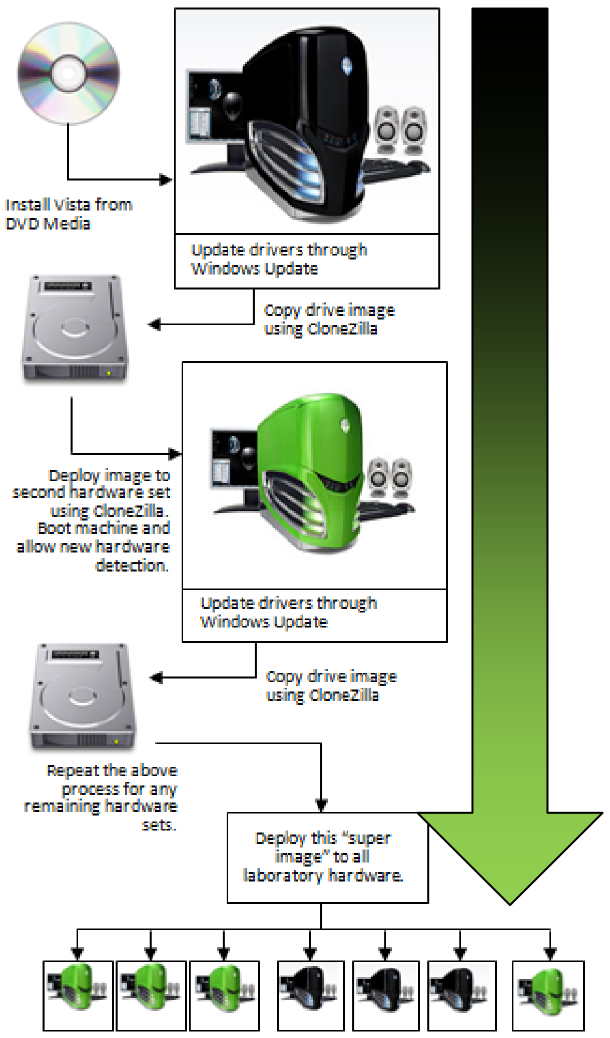

Install Vista from DVD Media
Update drivers through Windows Update
Copy drive image using CloneZilla
Deploy image to second hardware set using CloneZilla. Boot machine and allow new hardware detection.
Update drivers through Windows Update
Copy drive image using CloneZilla
Repeat the above process for any remaining hardware sets.
Deploy this "super image" to all laboratory hardware.

Figure 3: An overview of the imaging creation process using Vista's driver caching mechanism to create a "super image" for deployment on multiple hardware sets.

deployed to all of the machines. This resulting "super image" was then deployed to a machine for software installation. An overview of this process is depicted in Figure 3.

NOTE: this process was done with a fresh copy of Windows Vista Ultimate, and was done in a time-scale that allowed for Vista to remain non-activated. Had the OS been activated, it would have had to perform authentication checking as the hardware changes between machines were significant. At the end of this process, and after software installation, the image on the final machine was patched, licensed, and copied to the image server for distribution.

At any time, this process can be repeated by simply deploying the image to a test machine with new hardware, and then copying the image back to the server. At most, it would necessitate a possible deactivation/activation loop for the Vista license management environment.

NOTE: Several of the license issues associated here are due to the Vista Ultimate product SKU lacks support for site licensing. We also tested the use of an MSDNAA site license of Vista Business, and had significantly fewer issues in this regard, but wanted the richer features of the Ultimate installs.

It should come as some surprise to most systems administrators that Vista is in fact this resilient in terms of detecting and booting on hardware from a base image created on a different machine. The hardware changes were significant including switching brands and chipsets of processor, number of cores, amount of RAM memory, graphics card vendor, motherboard design, etc. We were pleasantly surprised that this was possible given the newly designed core of Windows Vista.This approach would not have been possible using older versions of the operating system. Over the course of our testing, we automated this process across our test network to avoid carrying the USB drive from machine to machine, but the underlying process was the same: allow Vista to "touch" each type of machine and pool together a "super image" for deployment.

Following this process, we were ready to deploy the image, and our post-deployment scripts, to each machine in the lab. The last thing added to the image prior to deployment are a set of scripts that will execute on the first boot of the image, which we placed in a folder labeled 'deployed' on what would eventually be the C:\ drive of the target machine. A detailed look at these scripts and their function is presented in section 3.5, and the code for the scripts themselves is presented in Appendix B.

3.4 Image Deployment via Multicast

The next step in creating the lab was the image deployment phase. For this stage, all hardware had been installed in the lab and physically connected to the network. Each machine in the laboratory was configured to boot from LAN using the nVidia PXE boot agent. (Similar configurations are possible using most LAN aware boot systems in the BIOS of the various motherboard manufacturers). First, the image/DHCP server was rebooted into its "image server on" state. Next, all machines in the lab were then booted to a power-on state, and booted via the network to a version of Linux that is shoved to each machine via a centralized Linux server (which in this case is our imaging server). Each machine joins a multicast session, and the server is configured to wait until each machine in the lab joins the multicast group before sending any further information beyond the files needed to boot the machine.

Once all of the machines are in the same multicast group, the server executes a push of the entire image. We tested this both using uncompressed images (in our case the final installation of the OS and all needed software was approximately 70GB), as well as images compressed using gzip. In terms of overall time, this was roughly a wash – the time reduction in sending less information was eaten up by the time spent compressing and uncompressing the image on either side of the network push.

NOTE: Our laboratory infrastructure was connected via gigabit Ethernet at every stage – all client machines had 1G interfaces, and the switch for the lab had 1G capability for all connected ports. Despite this, the maximum throughput for the push topped out at about 250Mbs – due primarily to the overhead of writing the image packets to the local disk. Our tool of choice here was UDP Cast [38] with ntfsclone [18], although we also tested CloneZilla in this context as well. Although Clonezilla provided a more user-friendly environment, the constraints of our network topology as well as limitations due to the institute network infrastructure made it impossible to correctly deploy Clonezilla without temporary loss of network access while changing configurations.

Once the entire laboratory image is pushed, byte by byte, to each client, the server switches into a pure DHCP mode and does not respond to requests to join a multicast session. The clients are delayed a few minutes and then reboot. Since the server has now switched modes, the boot-from-LAN option recognizes a local device option and the machines boot from disk, which is now a localized copy of the deployment image. The scripts mentioned earlier that were placed on the C:\deployed directory are now run automatically to configure the client machines. The source scripts for the multicast push, server, and client configuration are all contained in Appendix A of this paper.

3.5 Automated Image Configuration (a.k.a. Dante Reboots His Computer Six Times)

Now that the image has been deployed, there are six completely automated stages of configuration that each machine completes. These are all accomplished via script, and after the sixth and final reboot, the script directory is placed in a finalized state, leaving behind a workstation that is ready for student use. The source code to each of these scripts is presented in Appendix B.

The first of these scripts performs an important, but simple, operation. The script is written such that the first thing the computer does upon waking up is to wait for the hardware detection phase to complete: remember that while the "super image" on the machine has seen all of the hardware, it has only a 1-in-10 chance of being the hardware that the image was booted on prior to deployment. In addition, simple changes such as different USB ports for mouse and keyboard will cause windows to probe for new devices. Thus, it is almost always the case that extra time here is needed. Additionally, this script can be configured to add additional drivers to the machines as needed. Finally, the script uses a tool called WSNAME [39], which sets a unique machine name based upon IP address as well as a workgroup name. All machines have identifiable names (in our case we used names such as "GDD-LAB-01 through GDD-LAB-64"). The script then reboots the machine prior to proceeding to phase 2 in order for the hardware detection and system name change to take effect.

Phase 2 of the scripting setup performs, again, a simple and yet highly critical function. The purpose of this phase is to generate a unique system identification configuration for each workstation. To accomplish this, we use the NewSID [20] tool, which is available (but not officially supported) by Microsoft. Although there is still heated debate as to whether this tool works for Vista, we have found that with our software set and configuration, this tool performs flawlessly. This phase can take a significant amount of time, upwards of 30-40 minutes per machine. Most of the time in the script is simply delayed time waiting for NewSID complete. Once the system has a new system identifier, it is rebooted, again via script, and then enters Phase 3 of the scripting installation process.

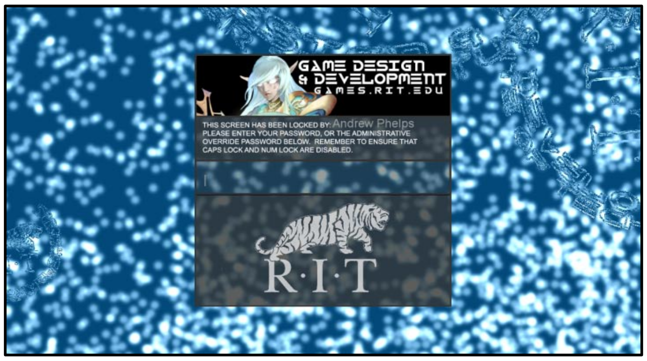
The purpose of Phase 3 is to properly license Windows Vista for each client machine. In our testing with Windows Business client images, this is trivial as RIT, like most MSDNAA partners, simply manages a site license server, and pointing any client machine to the server resolves the issue. Since we were using Vista Ultimate, however, things are a bit more involved. Each version of Ultimate has a unique product key, and these keys must be used as an argument for the `slmgr` tool to properly license and activate the operating system. We created a manifest that identified each client in the laboratory by fixed IP address (which are in turn assigned by MAC address from our DHCP configuration). We then scripted this phase to check the manifest, retrieve the appropriate individualized key, and then use this key to individually authorize each license of Windows Ultimate. Once authorized, these licenses can be re-imaged onto the same machines without issue, but cannot be moved to alternate hardware without notifying Microsoft and resetting the authorization key. Once Windows authorization and Genuine Advantage have been enabled, the machine is rebooted via script in order for these changes to take effect, and the machine proceeds to Phase 4 of the scripting setup process.

Phase 4 of the scripting process performs several key steps, which are unrelated to one another but are batched together for convenience. The first task at this stage is that the script synchronizes the machine to a Net-time server. Second, a script is run that expands the current C:\ partition to the size of the drive in the machine minus 40-50GB. The script then uses the remaining 40-50GB to create and mount a partition at P:\ labeled "Public". This is the public drive that was alluded to earlier in section 2.3. Finally, the script uses a tool called Display Changer [1] to enable both monitors connected to the computer, as well as to ensure that both monitors are running at the maximum possible resolution and refresh rate. These small "user tweaks" can seem trivial, but are part and parcel to providing a well-used environment, particularly since the main drive will be locked away from customization in the next stage. Any system-level tweak not performed here will have to be performed by a user every time he/she logs into the workstation. Finally after all of these tweaks and mods are performed, the script reboots the machine to ensure that all changes have taken place.

Phase 5 of the scripting process makes the last two (very important) changes to the system: first, it installs and configures the myLogon system, which replaces the standard Windows logon procedure as discussed previously. The details of this are straightforward, the only caveat being that customizations are added such that when a user logs on via MyLogon, a network drive is mounted as H:\ to the departmental storage area network account for the user that logs in. This is relatively simple in that MyLogon is essentially an Active Directory domain-based authentication procedure anyway, and so using the same information to mount a Samba share is straightforward.

The other task that is performed in Phase 5, after all other tasks have been completed, is to enable SteadyState drive protection. Up to and until this time, the machine has been in a state in which the scripts were running with administrative rights, and the drive was modifiable. Once enabled, however, SteadyState considers the drive to be "set" – any further changes that occur to the C:\ partition are seen as temporary, and the C:\ partition is reverted to this default state upon reboot. The script enables SteadyState protection, which requires the final reboot of the system. The script then deletes the contents of the C:\deployed directory prior to SteadyState switching on when the workstation restarts.

At this point, the machines are now ready for student use, all of the configuration scripts have run and all setup is complete. The machines are licensed, and the drives are "locked". A final tool that was created and deployed to each workstation was a screen-lock tool that serves the place of a screensaver in a more traditional logon environment. Because students are sharing a base Windows account, locking the machine through CRTL+ALT+DEL would lock this account, and students do not have the logon credentials to re-enter the workstation. Because of this pitfall, workstation lock, logout, and user switching are disabled through group policy settings. But we wanted to provide a way in which students could lock their individualized machines when they were away from the lab for short periods, and so we created an application to do just that. This application uses their Active Directory account information to authenticate the user against the same server that the MyLogon system uses for domain authentication, and thus users need only their one username/password combination. The application also silently removes the "Start Task Manager" functionality from the Vista logon screen, and returns this functionality when the machine is unlocked, by toggling a key in the registry hive. Finally, it was a goal of the lab that this application have an appropriate level of "sparkle" since it would be seen often, and so it was itself written in XNA GS/C#. A screengrab of the lock application is provided in Figure 4.



THIS SCREEN HAS BEEN LOCKED BY: Andrew Phelps
PLEASE ENTER YOUR PASSWORD, OR THE ADMINISTRATIVE OVERRIDE PASSWORD BELOW. REMEMBER TO ENSURE THAT CAPS LOCK AND NUM LOCK ARE DISABLED.

Figure 4: The C#-based screen lock application developed for the Game Design & Development Laboratory at RIT.

4 Supporting XNA GS in the Laboratory Environment

4.1 Multiple flavors of XNA GS

At the time of this writing, there are several existing versions of XNA Game Studio. For a variety of reasons, we support all releases of XNA, not just the latest version, as follows:

First, the versions of XNA GS currently available at this time are the original XNA Game Studio Express 1.0r (the refresh release), XNA Game Studio 2.0, and XNA Game Studio 3.0 CTP. As each of these versions was released, a variety of books, training demos, web-based tutorials, and other materials were released. Because this material is critical to student success, and because XNA GS is used in introductory coursework where students could not yet reasonably be expected to port a given sample to the latest version, the faculty felt it was in the best interest to support all versions available. Likewise, RIT has internally developed a number of samples and demonstratioin programs, and does not have the resources to keep every program at the latest release level. As there have been fairly significant changes during each release, both in terms of programmatic structure as well as hardware targets, it is simply more effective to support all of the versions.

4.2 XNA Installation Overview

After significant testing, we have found the most optimimum install path to be as follows:

1. Install Visual Studio 2005 C# Express.
2. Install Visual Studio 2005 Professional and associated MSDN release.
3. Install Visual Studio 2005 SP1 from the full downloadable .exe (i.e. NOT through Windows Update as this can hang).
   NOTE: this step must be repeated, as the first install will target Visual Studio, and the second install will target the express package.
4. Install XNA Game Studio Express 1.0r.
5. Install XNA Game Studio 2.0.
6. Run Windows Update and apply all patches
   NOTE: this step requires several reboots. Re-run Windows Update after each successful completion until there are no further critical updates.
7. Install Visual Studio 2008 Professional.
8. Install XNA Game Studio 3.0 CTP.
9. Install the DirectX SDK (latest versioin).
10. Install Visual Studio 2008 SP1 and MSDN for SP1 from the downloadable .iso images. (Installing from the iso images took approximately 1/5th the time of using the web-based installer in our testing, which had no relationship to actual network speed).
11. Run Windows Update and apply all patches
   NOTE: this step requires several reboots. Re-run Windows Update after each successful completion until there are no further critical updates.

4.3 XNA Installation Caveats and Workarounds

In each of the above installations, several caveats are applicable as noted below:

1. For Visual Studio installations, we always used a 'full' install – all features were installed to disk, nothing is set to run from a networked location or on an 'as needed' basis.
2. When installing MSDN (2005/2008), a full install is used. The help system is configured to use web-based help first, and disk-based help only if the network is unavailable. (This is mildly ironic since if the network is unavailable a user could not log on to the workstation in the first place).
3. During the installations of XNA Game Studio, the option to allow the installer to modify the Windows Firewall was selected. HOWEVER, this is insufficient, as noted in section 5.6.
4. Following the installation of the DirectX SDK, Visual Studio path variables must be re-configured to use the $DX_SDK variable rather than the hard-coded path. For example, instead of using "C:\Program Files\Microsoft DirectX SDK (August 2008)\include", the include path within Visual Studio should ideally be "$(DXSDK_DIR)include". The installers set the DXSDK_DIR environment variable, but often do not use it when setting paths inside visual studio.
5. The default project locations within Visual Studio need to be changed to the appropriate directories. For an Active Directory installation, these should be pointed to an area within the user account (but not within LOCAL as they are configured by default). For our installations, these were configured to point to the auto-mounted P:\ drive (i.e. the [P]ublic partition).

5 Network Configuration, Security, and Account Management

5.1 Laboratory Network Configuration

As noted previously, the overall setup for the network configuration of the lab is both (a) relatively simple, and (b) highly dictated by the requirements for the multi-cast image push. The lab is locked to a single subnet, and this subnet is specifically NOT shared with any other environment on campus. This allows a full broadcast of the multi-cast image without the need for the switch to do anything complicated with regard to segmenting the network. (We did have need to put a blocking call on the switch to avoid forwarding multicast traffic to any other portion of the RIT network).

The specifics of the DHCP setup for the lab are relatively straightforward. The switch itself is configured to send DHCP requests to a specific port, and our imaging/DHCP server is connected to the switch at that location. The pool of addresses that are valid run the entire range of the subnet: the first 64 addresses are reserved for the workstations in the lab – 129.XX.XX.1 – 129.XX.XX.64, and are assigned via a hardcoded table based on MAC address. The server address is also a fixed IP, and clients are configured to use this address at all times. This does create a single point of failure, but there are several techniques and possibilities to provide failsafe measures for a single DHCP server address, not the least of which is simply redundant architecture server hardware. The remaining addresses in the lab are assigned dynamically via DHCP, and are thus highly configurable, allowing not only the XBOX360 connections but student laptops, lounge machines, and game consoles and systems of several varieties.

5.2 Firewall Requirements and Configuration for XNA GS and XBOX360 Communication

The installation of XNA Game Studio (as noted previously) will correctly configure the local windows firewall. This was insufficient, however, for our prior lab, *as the firewall settings were overwritten as an inherited security policy from the domain controller*. Thus they had to be added manually. Likewise there are secondary and tertiary firewalls on the switching and routing equipment that comprise our campus infrastructure – and exceptions on these had to be registered with our university network administrators.

Specifically, ports 80, 88, and 3074 had to be opened, with UDP capability on 88 and 3074 in addition to TCP traffic. Additionally, both 80 and 3074 had to be configured with "port triggering" as well.

The other problem encountered initially in setting all of the XBOX 360 stations on the laboratory network was that the default security scheme for RIT networks is that the machine is authenticated against a list of "allowed" MAC addresses in order to receive a valid IP (i.e. one that does not force the user to a redirected registration page – which is not viewable on an XBOX). This can be circumvented with ease since the console allows the MAC address to be specified, but this was not an ideal situation. Eventually, after much trial and testing, we decided to go with the scheme described in the previous section that locked the first 64 addresses assigned to the MAC addresses of the 64 workstations in the lab, and allowed the other addresses in the lab to by dynamically assigned. This means that a given XBOX may not have the same address at all times, but will always receive a valid address from our DHCP server. Because we are handling DHCP locally on our subnet, we can disable the MAC address security check for specific ports, namely those to which the 360s are connected.

5.3 Account Management and Network Integration

Most of our setup and deployment issues surrounding accounts were covered earlier, or are direct implementations of a basic Active Directory scheme (with the exception of linking to the AD mounts from the SAMBA server, which is a topic unto itself and covered online in great detail).

A final note about accounts that is worth mentioning is the use of the Creators Club accounts that allow students to deploy work to the XBOX360. We were originally significantly concerned about these accounts as they are transferrable from XBOX to XBOX, and linked to XBOX Live Gold Accounts. Questions abounded about "how will we keep student A from using the account at home?" or "Suppose student A logs in on XBOX A, and then student B uses the same logon information for XBOX B?" In our previous lab, we had created a series of 25 Silver Accounts, and linked them to 25 Creators Club accounts that we were given as a part of an award as a winner of the 2007 Microsoft XNA Games Studio Express Innovation Award [12]. We were concerned about how these accounts would be able to be used in the new lab, what would happen when they expired, etc. We tested several alternatives about keeping accounts locked to USB keys, having an account-use checkout system, and other possible scenarios.

In the end, all of this questioning and testing was for naught. Microsoft made available last year a Creators Club account to any student at a recognized school through the DreamSpark program [19], and further extended these benefits to students covered through campus wide agreements under the Microsoft Developer's Network Academic Alliance (MSDNAA). Thus, any student at RIT that wants a Creator's Club account can create one by ordering through our MSDNAA portal, and so we simply require all GD&D majors to create such an account. Any XBOX they use in the lab can be temporarily "theirs" just by logging into XBOX live. This has the downside in that they have to re-connect it to the visual studio environment, but other than this slight inconvenience, the benefits far exceed this minor delay.

6 Conclusion & Future Work

Through a combination of tools, techniques, and testing, we were able to achieve our primary goals for providing a suitable workplace for our student body. The Laboratory can be deployed (and re-deployed) through an almost entirely unattended process, and provides what we feel is a suitable balance between usable computing infrastructure and security. Most important, with all that we have learned from this implementation, we are better prepared to take further steps to augment the current implementation.

Already, we are exploring ways in which to mount not only storage as a windows share, but to provide group-based accounts within and extended from the AD that link to SVN repositories for shared work and support for group projects. This in turn will likely be linked to Visual Studio Team Foundation Server, Sharepoint servers, and other such tools. The steps taken in this document provide only the basis of an automated system – much more could be done to create truly unique and immersive environments that best support student work in this exciting area. The authors would greatly welcome any discussion or further thoughts along these lines, and can be reached using the contact information at the start of the document. Furthermore, we wish you well in your own endeavors supporting game development laboratories!

References

1. 12Noon.com. *Display Changer 4.0*. 2008 [cited 2008 July 31]; Available from: http://www.12noon.com/displaychanger.htm.

2. Adobe Corporation. *Adobe Creative Suite 3 Master Collection*. 2008 [cited 2008 September 15]; Available from: http://www.adobe.com/products/creativesuite/mastercollection/?xNav=MC.

3. Alienware Corporation. *Alienware Corporation - Custom-Built Gaming Desktops and Notebooks*. 2008 [cited 2008 September 15]; Available from: http://www.alienware.com.

4. AMD. *AMD Processors - Product Information*. 2008 [cited 2008 September 16]; Available from: http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118,00.html.

5. AMD. *ATI Home Page*. 2008 [cited 2008 September 16]; Available from: http://ati.amd.com.

6. Autodesk. *Autodesk 3ds Max*. 2008 [cited 2008 September 15]; Available from: http://usa.autodesk.com/adsk/servlet/index?id=5659302&siteID=123112.

7. Autodesk. *Autodesk Maya*. 2008 [cited 2008 September 15]; Available from: http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018.

8. Bierre, K., et al., *Motivating OOP by Blowing Things Up: An Exercise in Cooperation and Competition in an Introductory Java Programming Course*, in *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*. 2006, ACM Press: Houston, TX. p. 354-358.

9. Big Bang LLC. *Universal Imaging Utility*. 2008 [cited 2008 September 15]; Available from: http://www.uiu4you.com/uiu_description.html.

10. Clonezilla. *Clonezilla Home Page*. 2008 [cited 2008 June 1]; Available from: http://www.clonezilla.org.

11. Deutsch, C.H., *TECHNOLOGY; Some Colleges Take Games Seriously*, in *New York Times*. April 1, 2002: New York, NY.

12. Downs, K., *RIT's Game Design and Development Program Wins Microsoft Research Award*, in *University News*. 2007.

13. Egert, C., et al., *Hello, M.U.P.P.E.T.S.: Using a 3D Collaborative Virtual Environment to Motivate Fundemental Object-Oriented Learning*, in *Companion to the 21st Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*. 2006, ACM Press: Portland, OR. p. 881-886.

14. Egert, C., S. Jacobs, and A. Phelps, *Bridging the Gap: Balancing Faculty Expectations and Student Realities in Computer Gaming Courses*, in *FuturePlay*. 2007: Toronto, Ontario, Canada. p. 201-204.

15. Egert, C., P. Ventura, and A. Decker, *Putting the "Fun" Back in Fundamentals: Using Games to Teach Object-Oriented Design Early*, in *American Society for Engineering Education St. Lawrence Section Conference*. 2005: Binghamton, NY.

16. Intel Corporation. *Intel Processors*. 2008 [cited 2008 September 16]; Available from: http://www.intel.com/products/processor/index.htm?iid=prod+prod_processor.

17. IWR Consultancy. *MyLogon - Alternative Network-Logon Applet for Windows*. 2008 [cited 2008 June 15]; Available from: http://iwrconsultancy.co.uk/mylogon/.

18. linux-ntfs.org. *NTFS Clone*. 2008 [cited 2008 August 15]; Available from: http://www.linux-ntfs.org/doku.php?id=ntfsclone.

19. Microsoft Corporation. *Microsoft Dreakspark*. 2008 [cited 2008 September 15]; Available from: https://downloads.channel8.msdn.com/.

20. Microsoft Corporation. *NewSID*. 2008 [cited 2008 June 1]; Available from: http://technet.microsoft.com/en-us/sysinternals/bb897418.aspx.

21. Microsoft Corporation. *Windows SteadyState*. 2008 [cited 2008 July 12]; Available from: http://www.microsoft.com/windows/products/winfamily/sharedaccess/default.mspx.

22. Microsoft Corporation. *Windows Vista Home Page*. 2008 [cited 2008 September 16]; Available from: http://www.microsoft.com/windows/windows-vista/default.aspx.

23. Microsoft Corporation. *XBox.com - XBox 360*. 2008 [cited 2008 September 16]; Available from: http://www.xbox.com/en-US/hardware/?WT.svl=nav.

24. Microsoft Corporation. *XNA Development Center*. 2008 [cited 2008 September 15]; Available from: http://msdn.microsoft.com/en-us/xna/default.aspx.

25. Nordlinger, J. and A. Phelps, *Gaming for Computer Science Instruction*, in *Microsoft Research Faculty Summit*. 2006: Redmond, WA.

26. nVidia. *nVidia Home Page*. 2008 [cited 2008 September 16]; Available from: http://www.nvidia.com/page/home.html.

27. NYS Department of Education. *NYSED Office of College and University Evaluation*. 2008 [cited 2008 September 16]; Available from: http://www.highered.nysed.gov/ocue/.

28. Phelps, A., *Social Aspects of Game Related Software*, in *Microsoft Research Social Software Symposium*. 2006: Redmond, WA.

29. Phelps, A. *AndyWorld 11.0 - Curriculum Vitae*. 2008 [cited 2008 September 16]; Available from: http://www.it.rit.edu/~amp/vitae.html.

30. Phelps, A. *Degree Program: Bachelors in Game Design and Development*. 2008 [cited 2008 September 16]; Available from: http://games.rit.edu/degree_programs/bachelors_in_game_design_development/.

31. Phelps, A. *Degree Programs: Masters in Game Design and Development*. 2008 [cited 2008 September 16]; Available from: http://games.rit.edu/degree_programs/masters_in_game_design_development/.

32. Phelps, A. and C. Egert. *Director Online: A Balrog in the Browser*. 2005 [cited 2008 Spetember 16]; Available from: http://director-online.com/buildArticle.php?id=1160.

33. Phelps, A. and C. Egert, *Educational Practices for Technology Students in Entertainment Domains*, in *American Society for Engineering Education St. Lawrence Section Conference*. 2005: Binghamton, NY.

34. Phelps, A., C. Egert, and K. Bierre, *MUPPETS: Multi-User Programming Pedagogy for Enhancing Traditional Study: An Environment for both Upper and Lower Division Students*, in *Proceedings of the 35th Annual Frontiers in Education Conference*. 2005: Indianapolis, IN. p. S2H8-S2H15.

35. Phelps, A., C. Egert, and K. Bierre, *Games First Pedagogy: Using Games and Virtual Worlds to Enhance Programming Education.* Journal of Game Development, 2006. 1(4): p. 45-64.

36. Phelps, A., et al., *An Open-Source CVE for Programming Education: A Case Study*, in *Workshop at the 32nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 2005: Los Angeles, CA.

37. Rochester Institute of Technology. *RIT Information Security*. 2008 [cited 2008 September 15]; Available from: http://security.rit.edu.

38. UDP Cast. *UDP Cast*. 2008 [cited 2008 August 8]; Available from: http://udpcast.linux.lu.

39. WSName. *WSName Home Page*. 2008 [cited 2008 August 15]; Available from: http://mystuff.clarke.co.nz/MyStuff/wsname.asp.

Acknowledgements

The authors would like to thank Dave Mitchell and Chris Satchell, Microsoft XNA, Dave Luehmann, Microsoft Games Studios, John Nordlinger, Microsoft Research, Gary Scarbrough, Chad Weeden, Jim Leone, and Virginia Gross-Abbey, RIT Information Technology, Kim Shearer, RIT B. Thomas Golisano College of Computing & Information Sciences, Dave Thelen, Greg Gardner and Andy Goud, RIT Information Technology Services, and the faculty of the RIT Game Design & Development program for their continued advice, support, and dedication. We could not have created this outstanding facility without you!

Appendix A: Image Server Deployment Scripts

Sample dhcpd.conf file from the linux distribution server

```
option domain-name "gdd.rit.edu";
option domain-name-servers 1xx.xx.xx.xx, 1xx.xx.xx.xx;
option routers 1xx.xx.xx.xx;
option ntp-servers 1xx.xx.xx.xx;
option netbios-name-servers 1xx.xx.xx.xx;
ddns-update-style none;
default-lease-time 86400;
max-lease-time 172800;
allow booting;
allow bootp;
next-server 1xx.xx.xx.249;
filename "pxelinux.0";

subnet 1XX.XX.XX.0 netmask 255.255.255.0 {
  range dynamic-bootp 1XX.XX.XX.XX 1XX.XX.XX.XXX;

  host gdd01 {
    # green alienware, dual 8800GTX, Intel Dual-Core
    hardware ethernet 00:00:00:00:00:00;
    fixed-address 1XX.XX.XX.1;
    option host-name "gdd01";
  }
  host gdd02 {
    # short black, ATI XFire, AMD quad core
    # old etl-24 in previous lab
    hardware ethernet 00:00:00:00:00:00;
    fixed-address 1XX.XX.XX.2;
    option host-name "gdd02";
  }
  # … several hosts (3-64) not listed here for brevity…
  #an example non-workstation fixed host on the lab network
  host gddFixedAddressMachine1 {
    hardware ethernet 00:00:00:00:00:00;
    fixed-address 1XX.XX.XX.240;
    option host-name "gddFixedAddressMachine";
  }
}
```

Sample deployment scripts from PXE server – sample deploy all

```
default interactive
timeout 5
prompt 1
label capture
    kernel rescuecd
    append initrd=initram.igz dodhcp ar_source=http://1xx.xx.xx.xx9/gdd/ autoruns=1
boothttp=http://1XX.XX.XX.249/sysrcd.dat cdroot setkmap=us noapic
label deploy
    kernel rescuecd
    append initrd=initram.igz dodhcp ar_source=http://1xx.xx.xx.249/gdd/ autoruns=2
boothttp=http://1XX.XX.XX.249/sysrcd.dat cdroot setkmap=us noapic
label interactive
    kernel rescuecd
    append initrd=initram.igz dodhcp boothttp=http://1xx.xx.xx.249/sysrcd.dat cdroot
setkmap=us noapic
label bootfromdisk
    localboot 0x80
```

Sample deployment scripts from the PXE server (continued) – script for 81151C

```
default bootfromdisk
timeout 5
prompt 1
label deploy
    kernel rescuecd
    append initrd=initram.igz dodhcp ar_source=http://1xx.xx.xx.249/gdd/ autoruns=4
boothttp=http://1xx.xx.xx.249/sysrcd.dat cdroot setkmap=us noapic
label interactive
    kernel rescuecd
    append initrd=initram.igz dodhcp boothttp=http://1xx.xx.xx.249/sysrcd.dat cdroot
setkmap=us noapic
label bootfromdisk
    localboot 0x80
```

Sample deployment scripts from PXE server (continued) – presender.sh references 81151C above

```
#!/bin/sh
cp /tftpboot/pxelinux.cfg/deploy /tftpboot/pxelinux.cfg/81151C
```

Sample deployment scripts from PXE server (continued) – sender.sh references 81151C above

```
#!/bin/sh
#udp-sender -f vistaunc.img --max-bitrate 50m
udp-sender --rexmit-hello-interval 1000 --max-bitrate 300m --ttl 2 -f vistaunc.img
#udp-sender --rexmit-hello-interval 1000 --max-bitrate 300m --ttl 2 -f
vistaimage.img.gz
sleep 30
cp /tftpboot/pxelinux.cfg/bootfromdisk /tftpboot/pxelinux.cfg/81151C
```

Sample reciever scripts from multicast image clone – autorun0

```
#!/bin/bash
#mkntfs -Q /dev/sda1
#mount -t nfs cartman:/home/clone /mnt/windows
#gunzip -c /mnt/windows/2520final.img.gz | ntfsclone -r -O /dev/sda1 -
```

Sample reciever scripts from multicast image clone – autorun1

```
#!/bin/bash
mount -t nfs 1xx.xx.xx.249:/images /mnt/custom
cd /mnt/custom
dd if=/dev/sda of=mbr.img bs=512 count=1
ntfsclone -s -O vistaunc.img /dev/sda1
#ntfsclone -s -o - /dev/sda1 | gzip -c > vistaimage.img.gz
sleep 10
poweroff
```

Sample reciever scripts from multicast image clone – autorun2

```
#!/bin/bash
#(sleep and poweroff commands omitted)
mount -t nfs 1xx.xx.xx.249:/images /mnt/custom
cd /mnt/custom
udp-sender -f vistaimage.gz
```

Sample reciever scripts from multicast image clone – autorun3

```
#!/bin/bash
#(sleep and poweroff commands omitted)
mount -t nfs 1xx.xx.xx.249:/images /mnt/custom
cd /mnt/custom
dd if=mbr.img of=/dev/sda
ntfsclone -r -O /dev/sda1 vistaunc.img
```

Sample reciever scripts from multicast image clone – autorun4

```
#!/bin/bash
mount -t nfs 1xx.xx.xx.249:/images /mnt/custom
cd /mnt/custom
sleep 5
dd if=mbr.img of=/dev/sda
sleep 5
# udp-receiver --ttl 2 -p "gunzip -c" | ntfsclone -r -O /dev/sda1 -
udp-receiver --ttl 2 -p "ntfsclone -r -O /dev/sda1 -"
sleep 120
reboot
```

Sample SAMBA configuration file for SAMBA server (accessed by clients through automounted drive H:\)

```
[global]
      netbios name = SERVERNAME
      workgroup = GDD
      wins support = yes
      server string = GDD Samba Server (Samba %v)
      security = user
      encrypt passwords = yes
      enable privileges = yes
      domain master = yes
      domain logons = yes
      local master = yes
      preferred master = yes
      os level = 33
      log level = 1
      max log size = 1000
      log file = /var/log/samba/log.%m
      # hosts deny = ALL
      hosts allow = 127.0.0.1 127.0.0.2 1xx.xx.xx.x 1xx.xx.xx.
      interfaces = eth0 lo
      bind interfaces only = yes
      lanman auth = no
      ntlm auth = no
      client NTLMv2 auth = yes
      client lanman auth = no
      client plaintext auth = no

      logon home = \\SERVERNAME\%U
      logon path = \\SERVERNAME\profiles\%u
      logon drive = H:
        # logon script = foobar.bat
      add machine script = /usr/sbin/useradd  -c Machine -d /var/lib/nobody -s
                                                          /bin/false %m
      printing = cups
      printcap name = cups
      printcap cache time = 750
      cups options = raw
      usershare allow guests = No

[homes]
      comment = Home Directory for %U
      # valid users = %S, %D%w%S
      valid users = %U
      browseable = No
      read only = No
      create mask = 0600
      directory mask = 0700
      inherit acls = Yes
```

```
[profiles]
      path = /srv/samba/profiles
      comment = Network Profiles Service
      valid users = %U
      read only = No
      browseable = no
      store dos attributes = Yes
      create mask = 0600
      directory mask = 0700
      profile acls = yes
      csc policy = disable

[profiles.v2]
      copy = profiles

[netlogon]
      # path = /var/lib/samba/netlogon
      path = /srv/samba/netlogon
      comment = Network Logon Service
      readonly = yes
      browseable = no
      write list = @gddadmin

[updaterdir]
      path = /srv/samba/updater
      comment = Updater Drive
      read only = yes
      valid users = %U

[printers]
      comment = All Printers
      path = /var/tmp
      printable = Yes
      create mask = 0600
      browseable = no

[print$]
      comment = /var/lib/samba/drivers
      path = /var/lib/samba/drivers
      write list = @gddadmin
      force group = gddadmin
      create mask = 0664
      directory mask = 0775
```

myLogon net logon script – gdd.ini

```
[RunBefore]

[Mappings]
; P is already mapped to PUBLIC
H: = \\1XX.XX.XX.XX\%user%

[Run]

[RunWait]

SetUser = WSCRIPT C:\DELIVERED\SINED\SETUSERNAME.VBS %user%
DelPub = WSCRIPT C:\DELIVERED\SINED\DELPUB.VBS
```

Sample myLogon.ini configuration file

```
[Global]

; User Items

Username=
LogonNetwork=GDD Workgroup
vpn=Direct Connection

; Interface Items:
ShowProgress = 1
Debug = 0
PurgeConnections = 1
ShareCleanup = 1
InterfaceStyle=FullFeatured
AutoUpdateRegistry=0
TimeSync=1

; ras/vpn:
vpnUsername=
vpnPassword=

; Shell Integration:
SecureMode =1
SelfRepair = 1

; Passwords
AcceptLastUsed = 1
AllowNullPassword = 0
StandaloneOverrides=0
Standalone = (a large unique number)
AdminOverride= (a large unique number)

; Registry Items for Shell Integration Mode

; Advised Changes
RestrictTaskMan =1
HideUserCPL =1
NoWelcomeScreen =1
AdminShareCheck =1
NoXPSharedFolders =1
WarnOfPasswordExpiry=1
PreventPasswordExpiry=1

; Optional Changes
NoScreenSaverLock =1
NoWindowsKey =1
NoCDAutoRun =1

;Kiosk Mode
kioskkey = (a large pseudo-random number)
kioskapp =notepad.exe
kioskcloseaction =Shutdown
kiosknetmode =Standalone
kioskuser =

; "" assumes username/password of "kiosk" -
; - which account should have very restricted priveleges.

; Username Syntax-Checking:
; uminspaces = 0
; umaxspaces = 99
; umindots = 0
; umaxdots = 99
; uminats = 0
```

Sample myLogon.ini configuration file for client workstations (continued)

```
; umaxats = 99
; uminunderscores =0
; umaxunderscores = 99
; uminhyphens = 0
; umaxhyphens = 99
; umincaps = 0
; umaxcaps = 99
; uminlength = 1
; umaxlength = 99


; Networks:
DefaultNetwork=
; Set this if the first is not the default network. Otherwise, first is assumed
default.

[GDD Workgroup]

NetworkComment =
LogonServer = server.domain.name
LogonShare = netlogon
LogonDomain = DOMAIN
LogonScript = gdd.ini

[SERVERNAME]

NetworkComment =
LogonServer = server.domain.name
LogonShare = netlogon
LogonDomain = GDD
LogonScript = netlogon.ini
```

Appendix B: Client Configuration Scripts

Client configuration batch script – dispatch.bat
(NOTE: This script relies on a fair number of other tools, some of which are custom, and many of which are a part of the SYSINTERNALS suite from Microsoft)

```
@ECHO OFF

REM DISPATCH.BAT

REM -------------------------------------------------------------------------
REM SET UP DEFAULTS
REM -------------------------------------------------------------------------

SET MACHID=NONE
SET VISTATYPE=R
SET MACADDR=12345
SET LICKEY=-1
SET SCRNTYPE=0
SET FIRSTMAC=12345

REM -------------------------------------------------------------------------
REM GENERATE MAC ADDRESSES FOR THE MACHINE
REM -------------------------------------------------------------------------

GETMAC /FO CSV /NH > C:\DELIVERED\SINED\MACADDR.TXT

REM GO THROUGH THE MAC ADDRESSES AND FIND IN THE MANIFEST

IF NOT EXIST C:\DELIVERED\SINED\MANIFEST.TXT GOTO :STAGE01

FOR /F "tokens=1,2 delims=," %%i IN (C:\DELIVERED\SINED\MACADDR.TXT) DO CALL
:PROCESSMAC %%i %%j

GOTO :STAGE01

:PROCESSMAC
   IF %FIRSTMAC% == 12345 SET FIRSTMAC=%~1

   FOR /F "tokens=1,2,3,4,5 delims=," %%i IN (C:\DELIVERED\SINED\MANIFEST.TXT) DO CALL
:PROCESSMAN %~1 %%i %%j %%k %%l %%m
   GOTO :EOF

:PROCESSMAN
   IF NOT %~1 ==  %~4 GOTO :EOF
   SET MACHID=%~2
   SET VISTATYPE=%~3
   SET MACADDR=%~4
   SET LICKEY=%~5
   SET SCRNTYPE=%~6
   GOTO :EOF

REM -------------------------------------------------------------------------
REM STAGE 01 - Wait for machine to normalize and then reboot
REM            Also, change the name of the machine, wg, and disk label
REM -------------------------------------------------------------------------

:STAGE01
   IF %MACHID% == NONE SET MACADDR=%FIRSTMAC%
   IF %MACHID% == NONE SET
MACHID=IT%MACADDR:~3,2%%MACADDR:~6,2%%MACADDR:~9,2%%MACADDR:~12,2%%MACADDR:~15,2%

   REM For Debugging...
   REM ECHO Machine ID  : %MACHID%
   REM ECHO Vista Type  : %VISTATYPE%
   REM ECHO Mac Address : %MACADDR%
   REM ECHO License Key : %LICKEY%
   REM ECHO Screen Type : %SCRNTYPE%
```

Client configuration batch script – dispatch.bat (continued)

```
IF EXIST C:\DELIVERED\SINED\STAGE01.STA GOTO :STAGE02
   CSCRIPT //NoLogo C:\DELIVERED\SINED\STAGEBG.VBS "Stage 1 - Normalizing Device
Drivers and Renaming Machine/WG/Drive ... Please Wait"

   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 300

   C:\DELIVERED\SEALED\WSNAME.EXE /WG:INFOTECH /SDL
/RDF:"C:\DELIVERED\SINED\MACHMAP.TXT" /DFK:$IP

   ECHO Stage 01 > C:\DELIVERED\SINED\STAGE01.STA

   SHUTDOWN /T 20 /R /F
   ECHO Stage 01 Complete
   GOTO :EOF

REM -------------------------------------------------------------------------
REM STAGE 02 - Handle SID Change
REM -------------------------------------------------------------------------

:STAGE02

   IF EXIST C:\DELIVERED\SINED\STAGE02.STA GOTO :STAGE03
   ECHO Stage 02 > C:\DELIVERED\SINED\STAGE02.STA

   CSCRIPT //NoLogo C:\DELIVERED\SINED\STAGEBG.VBS "Stage 2 - Changing SID ... Please
Wait"
   REG ADD "HKCU\Software\SysInternals\NewSID" /v EulaAccepted /t REG_DWORD /d
0x00000001 /f

   C:\Delivered\Sealed\newsid.exe /a

   REM THIS SHOULD BLOCK WAITING FOR NEWSID TO COMPLETE!

   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 7200

   SHUTDOWN /T 30 /R /F
   GOTO :EOF

REM -------------------------------------------------------------------------
REM STAGE 03 - Handle Activation (KMS for Business, Manifest for Ultimate)
REM -------------------------------------------------------------------------

:STAGE03
   IF EXIST C:\DELIVERED\SINED\STAGE03.STA GOTO :STAGE04
   ECHO Stage 03 > C:\DELIVERED\SINED\STAGE03.STA

   CSCRIPT //NoLogo C:\DELIVERED\SINED\STAGEBG.VBS "Stage 3 - Licensing Vista ...
Please Wait"

   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 30
   IF %VISTATYPE% == U CSCRIPT //NoLogo C:\WINDOWS\SYSTEM32\SLMGR.VBS -ipk %LICKEY%
   IF %VISTATYPE% == R CSCRIPT //NoLogo C:\WINDOWS\SYSTEM32\SLMGR.VBS -rearm
   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 120
   IF NOT %VISTATYPE% == R CSCRIPT //NoLogo C:\WINDOWS\SYSTEM32\SLMGR.VBS -ato
   SHUTDOWN /T 120 /R /F
   GOTO :EOF

REM -------------------------------------------------------------------------
REM STAGE 04 - NET TIME to point at Porsche, Lexus, and Beetle
REM            Select the proper monitor resolution
REM            Expand Disk Size and create public partition
REM            Change SID and Machine Name
REM -------------------------------------------------------------------------

:STAGE04

   IF EXIST C:\DELIVERED\SINED\STAGE04.STA GOTO :STAGE05
   ECHO Stage 04 > C:\DELIVERED\SINED\STAGE04.STA
   CSCRIPT //NoLogo C:\DELIVERED\SINED\STAGEBG.VBS "Stage 4 - NETTIME, Screen Res,
Drive Expand and P: Partition ... Please Wait"
```

Client configuration batch script – dispatch.bat (continued)

```
CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 30

   NET TIME /SETSNTP:"server1.domain.name  server2.domain.name  server3.domain.name "
   IF %SCRNTYPE% == 0 GOTO :ST00
   IF %SCRNTYPE% == 1 GOTO :ST01
   IF %SCRNTYPE% == 2 GOTO :ST02
   IF %SCRNTYPE% == 3 GOTO :ST03
   IF %SCRNTYPE% == 4 GOTO :ST04
   IF %SCRNTYPE% == 5 GOTO :ST05
   IF %SCRNTYPE% == 6 GOTO :ST06
   IF %SCRNTYPE% == 7 GOTO :ST07
   IF %SCRNTYPE% == 8 GOTO :ST08
   IF %SCRNTYPE% == 9 GOTO :ST09
   IF %SCRNTYPE% == 10 GOTO :ST10
   IF %SCRNTYPE% == 11 GOTO :ST11
   IF %SCRNTYPE% == 12 GOTO :ST12
   IF %SCRNTYPE% == 13 GOTO :ST13
   IF %SCRNTYPE% == 14 GOTO :ST14
   IF %SCRNTYPE% == 15 GOTO :ST15
   IF %SCRNTYPE% == 16 GOTO :ST16

:STDONE
   CSCRIPT //NoLogo C:\DELIVERED\SINED\DSKPROBE.VBS 40 >
C:\DELIVERED\SINED\DISKPART.INP
   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 10
   DISKPART /S C:\DELIVERED\SINED\DISKPART.INP
   SHUTDOWN /T 30 /R /F
   GOTO :EOF

REM ----------------------------------------------------------------------
REM STAGE 05 - Activate MyLogon and SteadyState
REM ----------------------------------------------------------------------

:STAGE05
   IF EXIST C:\DELIVERED\SINED\STAGE05.STA GOTO :STAGE06
   ECHO Stage 05 > C:\DELIVERED\SINED\STAGE05.STA

   CSCRIPT //NoLogo C:\DELIVERED\SINED\STAGEBG.VBS "Stage 5 - Enabling MyLogon and
SteadyState ... Please Wait"

   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 30

   REM Configure Keys for MyLogon (Active)

   REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v Shell /t
REG_SZ /d C:\Windows\MyLogon\Explorer.exe /f
   REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v
ShutdownFlags /t REG_DWORD /d 0x00000027 /f
   REG ADD "HKCU\Software\MyLogon" /v LastUsed /t REG_SZ /d
2193161362023453472146141226086051133 /f
   REG ADD "HKCU\Software\MyLogon" /v LogonStatus /t REG_SZ /d authenticated /f
   REG ADD "HKCU\Software\MyLogon" /v securemode /t REG_SZ /d 1 /f

   REM Configure Keys if Standard Logon (Inactive)

   REM REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v
AutoAdminLogon /t REG_SZ /d 0 /f
   REM REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v
DefaultUserName /t REG_SZ /d "" /f
   REM REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v
DefaultPassword /t REG_SZ /d "" /f
   REM REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v
DefaultDomainName /t REG_SZ /d "" /f
   REM REG ADD "HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon" /v
ForceAutoLogon /t REG_DWORD /d 0 /f

   REM Configure Keys for SteadyState

   REM REG ADD "HKLM\Software\DIGM"
   REM REG ADD "HKLM\Software\DIGM" /v NetworkUpdaterPath /t REG_SZ /d
C:\DELIVERED\SINED\ /f

   REM Disable UAC
```

Client configuration batch script – dispatch.bat (continued)

```
REG ADD "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System" /v EnableLUA
/t REG_DWORD /d 0 /f

   REM Remove Administrators Rights and Substitute Power Users
   NET localgroup "Power Users" %username% /ADD
   NET localgroup "Administrators" %username% /DELETE

   REM Delete all sensitive files
   DEL C:\DELIVERED\SINED\MANIFEST.TXT

   REM Activate SteadyState

:LOOPWSS
   "C:\Program Files\Windows SteadyState\SCTUI.EXE" /EnableWDPAndReboot
   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 5
   GOTO :LOOPWSS

   REM Code should NEVER GET TO THE NEXT TWO LINES UNLESS STEADYSTATE FAILS!
   CSCRIPT //NoLogo C:\DELIVERED\SINED\SLEEP.VBS 60
   SHUTDOWN /T 120 /R /F

   GOTO :EOF

:STAGE06
   ECHO Stage 06 Complete
   GOTO :EOF

REM -------------------------------------------------------------------------
REM Monitor Configurations
REM -------------------------------------------------------------------------

:ST00
   C:\DELIVERED\SEALED\DCCMD -monitor="\\.\DISPLAY1" -width=max -height=max -depth=max
-refresh=max
   C:\DELIVERED\SEALED\DCCMD -monitor="\\.\DISPLAY2" -secondary -right -more
   C:\DELIVERED\SEALED\DCCMD -apply
   C:\DELIVERED\SEALED\DCCMD -monitor="\\.\DISPLAY2" -width=max -height=max -depth=max
-refresh=max
   GOTO :STDONE

:ST01
   C:\DELIVERED\SEALED\DCCMD -monitor="\\.\DISPLAY1" -width=max -height=max -depth=max
-refresh=max
   C:\DELIVERED\SEALED\DCCMD -monitor="\\.\DISPLAY2" -secondary -right -more
   C:\DELIVERED\SEALED\DCCMD -apply
   C:\DELIVERED\SEALED\DCCMD -monitor="\\.\DISPLAY2" -width=max -height=max -depth=max
-refresh=max
   GOTO :STDONE
REM Remaining Monitor Configuration Blocks deleted for brevity, similar to above
```

Client Update Script – updater.bat - checks for new script set on server

```
REM UPDATER.BAT

REM Attach to the network share for updates
NET USE \\(server.domain.name)\updaterdir /USER:updater

REM See if the update manifest exists
IF NOT EXIST \\(server.domain.name)\updaterdir\update.txt GOTO :EOF

REM Copy the Manifest Locally
COPY /Y \\(server.domain.name)\updaterdir\update.txt
C:\Delivered\Sined\updates\update.txt

REM Scan through the Manifest and find what needs to be updated
FOR /F "tokens=1,2,3,4 delims=," %%i IN (C:\Delivered\Sined\updates\update.txt) DO CALL
:PROCESSUPDATE %%i %%j %%k %%l

GOTO :EOF
```

```
REM ------------------------------------------------------------------------
REM Process update function
REM ------------------------------------------------------------------------

:PROCESSUPDATE
   IF EXIST "C:\Delivered\Sined\updates\manifests\%~2" GOTO :EOF

   IF NOT EXIST "\\(server.domain.name)\updaterdir\%~3" GOTO :EOF
   COPY "\\(server.domain.name)\updaterdir\%~3" "C:\Delivered\Sined\updates\temp\%~3"

   "C:\Delivered\Sined\updates\temp\%~3" %~4
   ECHO "Installed " > "C:\Delivered\Sined\updates\manifests\%~2"

   DEL /Q "C:\Delivered\Sined\updates\temp\%~3"
```

Custom Diskprobe Utility – diskprobe.vbs

```
' DSKPROBE.VBS

' Routine that checks the disk size to see how far we can extend it
' Also determines if we have enough room for a public partition

' Parameter - Target Public Drive Size in GB

Set args = WScript.Arguments
publicsizegb = args(0)

WScript.Echo("SELECT DISK 0")
WScript.Echo("SELECT PARTITION 1")


' Get Disk Information

Set objWMIService =
GetObject("winmgmts:{impersonationlevel=impersonate}!\\.\root\cimv2")
Set colDiskDrives = objWMIService.ExecQuery("SELECT * FROM Win32_DiskDrive")


For Each objDiskDrive in colDiskDrives
    if (objDiskDrive.Index = 0) then
       disksizebytes = objDiskDrive.Size
    end if
Next


' Now get Partition information

Set colDiskPartitions = objWMIService.ExecQuery("SELECT * FROM Win32_DiskPartition")

For Each objPartition in colDiskPartitions
    if ( (objPartition.DiskIndex = 0) And (objPartition.Index = 0) ) then
       partsizebytes = objPartition.Size
    end if
Next


' Now figure out rest

publicsizebytes = publicsizegb * 1024 * 1024 * 1024

extendsizebytes = disksizebytes - partsizebytes - publicsizebytes

if (extendsizebytes > (1 * (1024 * 1024 * 1024)) ) then
   WScript.Echo("EXTEND SIZE=" & ((extendsizebytes / 1024 / 1024) \ 1))
end if


WScript.Echo("CREATE PARTITION PRIMARY")
WScript.Echo("FORMAT FS=NTFS LABEL=""PUBLIC"" QUICK")
WScript.Echo("ASSIGN LETTER=P")
```

Appendix C: List of Installed Packages on Rich Vista Image for GD&D Lab

Software Listing for Entire Rich Image for GD&D Laboratory Workstations 2008	
McAfee Virusscan Agent	Condor (RIT Internal Parallel Experiment)
Microsoft Office 2007 Business	
Microsoft Visio 2007	VMWare Workstation 6
Microsoft Project 2007	
	Firefox 3
Adobe Flash CS3	Opera
Adobe Director 11	Web development extension for Firefox
Quicktime Pro	Firebug Extension for Firefox
Real Player	JSView Extension for Firefox
Winamp	FireFTP Extension for Firefox
Sound Forge XP	Yahoo Widget Engine
Audacity	Copy Converter Widget for Yahoo
Nvidia SDK	
Nvidia CG Toolkit	Visual Studio .NET 2005
NVShaderPerf	Visual Studio .NET 2008
PhotoShop DDS Plugins	DirectX SDK
Tortoise CVS	Java SDK
Tortoise SVN	Java JRE
Apache Ant	Java Documentation
WinMerge	Java Web Services Developer Pack (JWSDP)
Wings3D	BlueJ
Cortona VRML	NetBeans
Flex Builder	Bloodshed C++
XNA 2.0	UltraEdit
XNA 3.0 CTP	Jedit
Flex Builder 3	Jgrasp
Irfanview	Crimson Editor
Adobe Flash Player	BasicX
Adobe Flash Player Debugger	Eclipse
Adobe Air Player	Eclipse SDK Packages
Maya 2008	Eclipse PHP Plugin
3DSMax 2008	Java IO Files (Serial/Parallel)
Panda DirectX Exporter for 3DSMAX	Qt
Alienbrain Client	
	MySql
WinZip	ERWin
7-Zip	Adam Ineractive Anatomy
WinRAR	Matlab
SmartFTP	Dchip.org
FileZilla	R
WinSCP	R Bioconductor for R
Putty	
Adobe Acrobat Reader	Morphon XML Editor
Adobe SVG Viewer	oXygen XML Editor
Nero Express	Adobe Photoshop CS3
WinDVD Suite	Adobe Illustrator CS3
Ulead DVD Movie Factory	Adobe Dreamweaver CS3
CygwinX	Adobe Fireworks CS3
Arduino IDE	Adobe Premiere CS3
Processing	Adobe After Effects CS3
Control P5 Library for Processing	Adobe Encore CS3
Minim Library for Processing	Adobe Soundbooth CS3

Appendix D: System Preparation Checksheet for GD&D Lab Workstations (Pre-Push Image Installation)

Sample System Preparation Guide for Pre-Image Master Workstation

1) Change **ITAdmin/GDDAdmin** password
2) Create **WSSAdmin** account
3) Give **WSSAdmin** Administrators group privilege
4) Install **MyLogon** program from WSSAdmin, Run As Administrator
5) Install **SteadyState** program from WSSAdmin, Run As Admnistrator
6) Install **drbl-winroll** program from WSSAdmin (DO NOT SET UP SID/SSH SERVER MODE!)
7) Change naming pattern for machines if necessary (IP or MAC pseudo-regex – see instructions)
8) Copy CD **artwork\gdddesktop.bmp C:\Windows**
9) Copy contents of CD **Delivered** Directory to **C:\**
10) Copy CD **MyLogon\myLogon.ini** to Directory **C:\Windows\MyLogon**
11) **[ETLAB]** Copy **MyLogon\banner.gif** to Directory **C:\Windows\MyLogon**
12) **[ITLAB]** Copy **MyLogon\oldbanner.gif** to Directory **C:\Windows\MyLogon**
13) Copy CD **Scripts** to **C:\Program Files\Windows SteadyState\Scripts**
14) Copy CD **XML** to **C:\Program Files\Windows SteadyState\XML**
15) Copy CD **GDD_XNA_Screen_SaverObfusated\GDD_XNA_Screen_Saver\GDD_XNA_Screen_Saver\bin\x86\Release\GDD_XNA_Screen_Saver.exe** to **C:\Program Files\ScreenLock\GDD_XNA_Screen_Saver.exe**
16) Make desktop shortcut to ScreenSaver and label it **Lock Computer**
17) Run SteadyState, disable all of its restrictions (6 set, will be one when done)
18) Set SteadyState to do updates at 3:00 AM
19) Set SteadyState to use **DIGMMSIUpdater.vbs** as an alternative updater script
20) From Administrator Command Prompt, run **netplwiz**
    a. Uncheck "Users must enter a username/password to use this computer"
    b. Select **ITAdmin/GDDAdmin** as the default account
    c. Enter the password if prompted
21) **[DONE] Start → Control Panel → System**
    a. Select **System Protection** from left panel
    b. Select **System Protection** tab
    c. Uncheck **C:** drive
22) **[DONE]** Open **Computer** and select **Organize → Folder and Search Options**
    a. Select **View** Tab
    b. Uncheck "Hide Extensions for known types"
23) **[DONE]** Right click on start button, select **Properties**
    a. Select **Start Menu** tab
    b. Next to the **Start Menu** radio button, click **Customize…**
    c. Uncheck "Highlight newly installed programs"
    d. Check "Printers"
    e. Check "Run Command"
    f. Look at pulldowns and make sure they make sense
24) Control Panel → Security Center
    a. Click on "Change the way Security Center Alerts Me"
    b. Select "Don't notify me but display the icon"

Sample System Preparation Guide for Pre-Image Master Workstation (continued)

25) Change GPEDIT.MSC Settings
    a. Computer Configuration
        i. Administrative Templates → System → Power Management
            1. **Enable "Select An Active Power Plan"**
            2. **Select "High Performance"**
        ii. Administrative Templates → Windows Components → Autoplay Policies
            1. **Enable "Turn off Autoplay"**
            2. **Select "CD ROM and Removable"**
        iii. Administrative Templates → System → Power Management → Button Settings
            1. **Enable "Select the Power Button Action (Plugged In)"**
            2. **Select "Take No Action"**
            3. **Enable "Select the Start Menu Power Button Action (Plugged In)"**
            4. **Select "Shut Down"**
        iv. Administrative Templates → System → Power Management → Sleep Settings
            1. **Disable "Require a Password when a computer wakes (Plugged In)"**
        v. Administrative Templates → System → Power Management →Video and Display Settings
            1. **Enable "Turn off the display (Plugged In)"**
            2. **Set time for 1800 seconds (30 minutes)**
        vi. Windows Settings → Scripts (Startup/Shutdown)
            1. **Select Startup**
            2. **Path "C:\Delivered\Sined\delusername.bat"**
    b. User Configuration
        i. Administrative Templates → System → CTRL-ALT-DEL options
            1. **Enable "Remove Change Password"**
            2. **Enable "Remove Lock Computer"**
            3. **Enable "Remove Log Off"**
        ii. Administrative Templates → System → Scripts
            1. **Enable "Run login Scripts visible"**
        iii. Administrative Templates → Desktop → Desktop
            1. **Enable "Desktop Wallpaper"**
            2. Path **C:\Windows\gdddesktop.bmp**
        iv. Administrative Templates → Control Panel → Display
            1. **Disable "Screen Saver"**
        v. Windows Settings → Scripts (Logon/Logoff)
            1. **Select Logon**
            2. **Path "C:\Delivered\Sined\dispatch.bat"**
            3. **Select Logoff**
            4. **Path "C:\Delivered\Sined\reboot.bat"**

26) CD into C:\Delivered\Sined

27) armmach.bat