VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

School of Engineering and Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Web Interface to a Game Emulation Server

Jay Shepherd

Supervisors: Ian Welch & Stuart Marshall

Submitted in Partial fulfillment of the requirements for
Bachelor of Science with Honours in Computer Science.

Abstract

Digital artifacts are an important part of today's culture. However many digital artifact owners don't want their content to become part of a widely accessible archive for fear of losing control over their Intellectual Property. This could lead to future generations not having access to the historical artifacts of past generations. We propose a system that balances IP protection with accessibility and could be used by a trusted archive to protect these artifacts and make them available to the public. This will be a centrally managed and portable approach that also allows the ability to implement community features such as sharing game plays amongst users.

Table of Contents

1 Introduction

The aim of this project is to demonstrate the feasibility of creating a tool to support the archiving of digital artifacts by reducing IP-related barriers to their use. We will identify a set of requirements that support this goal and develop an approach to satisfy them. Finally we will implement a proof of concept to demonstrate the feasibility of the approach. The fundamental goal of this project is to allow people to use a piece of software without giving them the executable. Digital artifacts include software but also information about aspects of the software which will be referred to as Artifact Meta-data. Artifact Meta-data is information about an artifact such as author, purpose or company. In the context of executables it might be information such as manuals or locations of Easter Eggs[9] (hidden features). In the context of games it might be high scores, cheat codes, game guides or a recorded complicated gameplay. For the purposes of this project we are working within the domain of games as they pose a greater challenge, however this system is not limited to games.

1.1 Motivation

1.1.1 Problem

The preservation of digital artifacts is an important issue today as evidenced by government funding for projects such as New Zealand National Library's National Digital Heritage Archive[4]. The loss of such artifacts is a loss to popular computer and video game culture. There are many lessons about game play and designing games under unique constraints to be learned from forgotten games and many lessons that can be learned from other applications also. Even if purely for historical reasons it is important that all excellent creative works are preserved.

Preserving a cultural treasure like the Mona Lisa is important, but that preservation would mean nothing if no one could ever see the painting again. This is exactly the problem we face with digital artifacts. J Rothenberg once said "Old bit streams never die—they just become unreadable"[10]. Digital artifacts are made for a specific platform (made of hardware and software) and without this platform the artifact is unusable. Often digital artifacts are preserved but we face the problem of never being able to use them again, because we no longer have the original platform to run them on. Computing platforms are constantly evolving and as a result we often lose the ability to use applications for older platforms.

Various emulators have been created to allow artifacts for specific platforms to be executed on a newer platform, however none of them offer Intellectual Property protection and they often require lots of setup and maintenance. For the preservation of digital artifacts to be useful there needs to be an easy mechanism to emulate these platforms on a new platform and allow the digital artifacts to be used whilst protecting Intellectual Property.

Ideally, in a museum or archive setting we would allow visitors to access digital artifacts via these emulators. However, there are some difficulties in doing this that must be overcome. The first is practical, installing and maintaining multiple copies of emulators is time consuming and expensive, if we want to provide access on many machines. This is especially important as we would need to install different versions of the emulators to cope with diversity of platforms. The second relates to Intellectual Property and accessibility. We would like to allow visitors to access digital artifacts via their own devices or even from home. However, this would mean copying the digital artifacts onto their own machines and this would potentially violate New Zealand copyright law[5].

Furthermore this problem is not just limited to the preservation of software. Many people have lots of documents in formats that are now unsupported. If they don't have a machine that runs the original application, they are often left unable to use these documents. This project would allow people to use these old documents, as long as the original application is part of the archive.

1.1.2 Solution

A solution to the problem above is to use a thin-client approach with all execution occurring on a central server and only input and visual rendering occurring on the client. This server could be used as a trusted digital archive that provides access to use digital artifacts without allowing them to be copied. This system could be used by a government archive to preserve local content or as part of an international opt-in preservation project where intellectual property owners submit their work. The system would need to support access to software from a wide range of platforms to maximize preservation. Also it would need to provide an interface that is highly portable to maximize accessibility.

The preservation cause would also be supported by allowing users to interact with each other through an online community which would be used to build up Artifact Meta-data. The preservation of meta-data can be equally as important as preserving an application or game itself.

Thus the solution to this problem that we propose is a single system that can emulate multiple platform types, protect applications from being copied and provide the ability to use these applications over a highly portable medium that is not likely to be replaced in the near future. Should this medium be replaced the interface component could also be easily replaced to support a newer medium. This system must also be easy for a user to use an application, must require little initial setup and finally require no on-going maintenance.

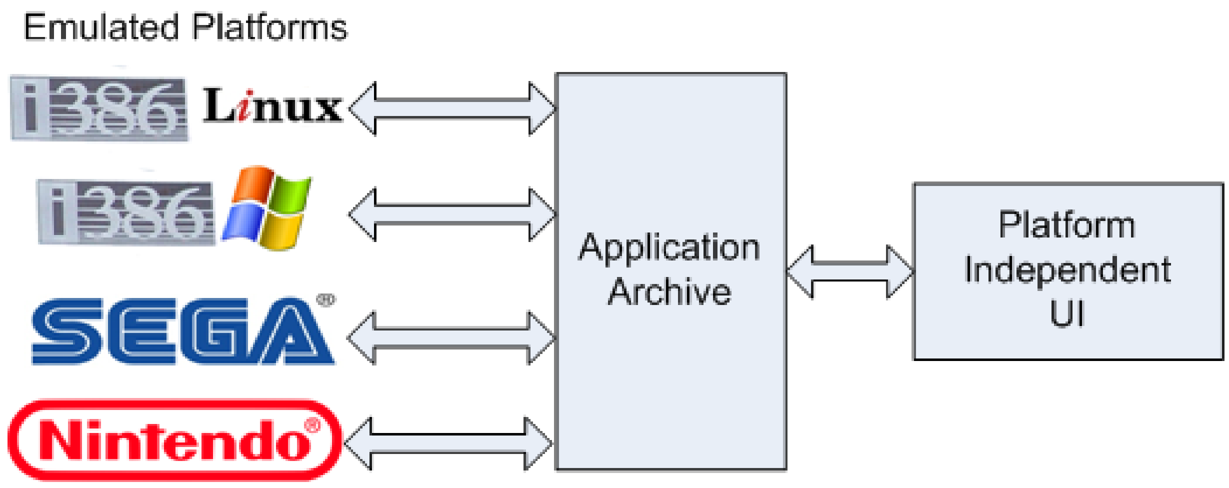


Figure 1: System design, Multiple platforms are supported through one common interface

1.2 Contributions

My first contribution to solving this problem was to develop an approach that overcomes the following issues:

- Emulation for various platforms in one single system.
- Platform independent client interface
- Intellectual property protection
- Support for meta-data and sharing game play amongst a community of enthusiasts.

My second contribution was to use this approach to develop and evaluate a proof of concept to demonstrate the feasibility of such an approach.

1.3 Methodology

This project followed a rapid iterative prototyping approach involving informal expert evaluations by myself and my supervisors on a 2 - 4 week basis. Regular evaluations lead to constant revisions of the architecture and design which were implemented in the next iteration of the prototype. Rapid prototyping suited this project as many of the requirements did not come to light until parts of the system had been implemented. Phase 1a involved assessing existing technologies such as emulators and virtualization systems to find the best system to run various applications. Phase 1b was where we came up with the design requirements and finalised the approach to the solution. Phase 2 covered the implementation, evaluation and requirements redesign of each iteration. Phase 3 involved a technical evaluation of the system which evaluated the requirements set out in Phase 2.

2 Background

In this chapter we will discuss the legal issues surrounding our preservation goals and the history of digital preservation attempts. We will identify why these attempts are not perfect solutions and how this project can solve their shortcomings. Finally we will discuss related technology that demonstrates the feasibility of each aspects of our chosen solution.

2.1 Legal Issues

Under New Zealand Law The National Library Act and its "National Library Requirement (Electronic Documents) Notice 2006"[6] classifies electronic documents as off-line documents. The National Library Act states that "Every Publisher of an off-line document must, at the publishers own expense, give the National Librarian 1 or more copies of the document. These exemptions to the Copyright Act of 1994 permit the National Library to store, copy and use any deposited document whether digital or otherwise.

However this does not permit the National Library to allow the distribution of media contained within an artifact such as images, sound or video. This artwork also carries intellectual property protection and so cannot be distributed by the National Archives. Unfortunately without this media an application or game cannot be used and so permission from the intellectual property owners is required to allow the artifact to be actually used.

Also most software licences are not designed for use in a multi-user server context, they are targeted at a single user and this makes the legalities surrounding this issue a grey area. Intellectual property concerns are a big issue for this project. For an interactive preservation system to be successful it must provide a wide range of digital artifacts to its users. Although intellectual property protection would be enforced by this system, intellectual property owners must opt-in and allow their work to be submitted to the system. This is the best way to ensure a legal system which could attract many intellectual property owners to want to submit their creative works.

2.2 History of Digital Preservation

2.2.1 Digital Reformatting

Digital reformatting is an approach taken by many preservation projects that simply converts a document in a certain format into a newer one. Some approaches build and uses document meta-data that describes a documents original structure and conversions to the new format. This approach has been used by the American Library of Congress to preserve their historical documents[1]. This approach works well for many situations but suffers from the problem that an entire collection must be constantly converted every time a new format becomes the norm and aspects of the original artefact are often lost in translation. Rather than constantly converting digital artifacts into new formats, the approach chosen for this project takes the opposite approach and just supports the original format in the environment it was designed for.

2.2.2 Emulators

One attempt to solve the issue of newer platforms loosing the ability to use applications from older platforms was to build emulators. An emulator is an application that emulates a specific platform on a newer platform[12]. This approach has worked well for a long time in the context of personal use. However emulators still suffer from the following issues.

- As platforms evolve we lose the ability to use applications built for older platforms. Since emulators are applications built for a specific platform they too suffer from this problem and must be ported to each platform that they will run on.
- Emulators require that the user have and install the executable for an application. Therefore emulators do not support intellectual property protection.
- Many emulators are difficult to setup and use making it difficult for the novice to use.
- Most emulators support only one specific platform.
- Emulators are aimed at an individual user and don't support wider aims such as archiving meta-data about an application or game.

2.2.3 Abandonware Communities

Abandonware is a term used for software that is no longer sold or supported[8]. Many communities have been built on the internet to talk about and trade abandonware. For example 'The Official Abandonware Ring'. The attitude that most members have is that if the company is no longer selling the product, then they should be able to use and redistribute it. The problem is that many intellectual property owners still refuse to let their property become part of these archives and still regard it as piracy. These archives usually take the approach of adding content and removing it when requested rather than gaining permission first.

These communities offer, emulators for using applications, tutorials for setting them up and the applications themselves. The main problem with this approach is the lack of intellectual property protection. However these communities demonstrate an interest in sharing gameplay experiences and information about games.

2.3 Related Technology

2.3.1 Remote ME

Remote ME[2] is a Java-based prototype for mobile-client / server communication that only requires a very thin mobile client. Delwadia et al [2]. investigated the feasibility of running games and applications remotely with a focus on how response times affect the users experience. This work tested two arcade games and came to the conclusion that game play is not affected with a minimum response time in the range of 75-150ms.

2.3.2 OnLive

OnLive[13] is a project that uses a thin client approach to remotely execute games on a server. It also protects intellectual property with this mechanism and uses web browser plugins as the client interface. It solves many of the issues this project is interested in and demonstrates the feasibility of the thin-client approach to gaming. However it does not address the issue of supporting a broad range of platforms as it is a closed commercial system that is targeted at new games. However it does show have It also suffers from the problem that the browser plugin must be ported to each browser and this reduces the portability of the system.

2.3.3 JPC

JPC [7] is an i386 emulator that is written in Java and can optionally run as a Java applet in a web browser. It supports preservation goals exceptionally well by providing emulation of digital artifacts on portable a medium. However it doesn't support intellectual property protection as execution is performed on the users machine. It has facility for a local network between virtual machines by running a virtual network hub. It will run any i386 based application for DOS, DSL Linux, Open BSD 4.3, Windows 3.0 and various other mini Linux distributions. JPC was written in Java and so will run anywhere with a JVM. According to the JPC website it is one of the fastest x86 emulators available and one of the few written in Java. It also has a built in snapshot feature allowing the emulators state to be saved or loaded at any time, which is a feature that would be required in an interactive preservation archive.

2.4 Project Feasibility

These projects show the feasibility of the approach planned for this project. The wide range of emulators available shows peoples interest in using applications and games from the past. Emulators themselves show the feasibility of running digital artifacts in their optimal environment on a new platform.

Remote ME and OnLive demonstrate the feasibility of a thin-client approach to gaming over the internet. OnLive has had a lot of success with this and is designed for new games with a higher resolution and more colours and therefore more data to transfer. Heritage applications with less data to transfer should work exceptionally well using this approach. It is interesting as it naturally supports intellectual property protection and would not require the client to install each application.

Abandonware communities demonstrate interest in preservation communities and show support for many features like sharing recorded games and meta-data. Their biggest downfall is a lack of intellectual property protection which this project aims to solve. Finally digital reformatting shows peoples interest in preserving documents stored in old formats. This project aims to use these ideas to better support digital preservation while avoiding the issues they faced.

JPC would be a good virtualization platform to use in an interactive preservation archive. A thin-client approach would be a good solution to the intellectual property protection goals of this project. Finally a community built around this system would make a good platform for preservation.

3 Requirements

To satisfy the goals of this project, the following issues need to be addressed.

1. Intellectual Property Protection
2. Portability
3. Broad Digital Artifact Platform Support
4. Performance
5. Persistent Sessions
6. Collaborative Gaming

3.1 Intellectual Property Protection

The first requirement is about providing protection from binaries being copied and subsequently redistributed. This is a significant barrier to archiving digital artifacts as losing distribution control can be a big concern for intellectual property owners. If this concern causes them to avoid submitting their work to a public archive such as a museum, the public looses out and the risk of the artifact being forgotten is increased. This system must be designed as an opt-in environment where intellectual property owners want to submit their artifacts. It could be also be used by organisations like the New Zealand National Archives who are legally entitled to a copy of any locally published works for archiving.

3.2 Portability

Another important issue is making sure that the interface to this system will run on a wide variety of platforms including mobile devices such as PDA's and phones. It needs to be implemented in such a way that it will continue to work as computing devices and platforms evolve. Porting an interface to every new operating system would quickly become tiresome. It will need to be built upon a platform common to all desktop computers and mobile devices. Maintaining a high degree of accessibility to digital artifacts is a major goal of this project. It should require little installation and no maintenance.

3.3 Broad Digital Artifact Platform Support

This requirement is about ensuring a wide range of platforms are supported to ensure maximum artifact coverage. As computing platforms evolve we often lose the ability to use old applications because of changes in system architectures and platforms. This system must be able to run a variety of emulators for a variety of platforms to maximize the coverage of the digital artifacts that it can support. This system needs to support every platform currently in existence and every platform not yet created. To achieve this the emulation parts of the system should be completely component based so that platform support can be easilly augmented into the system at any time.

3.4 Performance

Making sure the system is able to provide reasonable response times for the applications or games being used is an important goal. The required response time will vary from application to application but the system must provide a response time under 150ms. This threshold has been shown to be acceptable for various arcade games from research by Delwadia et al [2]. Users will not use the system if it can not perform at a relatively acceptable speed and so performance is a major criteria of this project.

3.5 Persistent Sessions

Requirement 5 is about making sure the system is able to save and store a users session so that they can leave the system and come back where they left off at any time. This must be independant of any save feature that an artifact may have internally as all artifacts must share a common method of saving sessions.

3.6 Collaborative Gaming

Finally requirement 6 is about making sure the system is accessible by multiple users simultaneously and allowing interaction between them. This interaction could come in the form of multiplayer games, watching another users live session and even cloning a users session so you can use it yourself. There should also be support for recording sessions as complicated processes could be demonstrated or complicated gameplays could be stored for 'brag rights'. There must also be support for meta-data to be stored along side an artifact.The preservation of this information is as important as storing the artifact itself and currently few archives preserve this information. This information could be submitted or generated by the users and helps to build interest in the artifacts being preserved.

4 Solution

In this section we will discuss the general architecture of the solution and how each component interacts with the system as a whole. We will discuss the community features in more detail and identify their impact on our preservation goals.

4.1 Architecture

Our solution to this problem was to construct a thin-client system that is comprised of three components: a virtualization server, an HTTP server and a web based interface.

4.1.1 Virtualization Server

To support requirement 3: "broad digital artifact platform support" and provide the ability to run applications designed for a range of platforms, virtual machine or emulator components will be required. The two main benefits of this approach are:
Isolation, Each virtual machine has its own emulated hardware and is completely isolated from all other virtual machines and the physical machine running them. This means that no amount of hacking could result in a security compromise through a virtual machine as the virtual machine cannot access any of the physical machines resources.
Custom Environment, The virtual machine can be customized and setup for the specific hardware requirements of the selected artifact. CPU, memory and operating system requirements can be matched exactly, offering the optimal environment for the artifact to run.

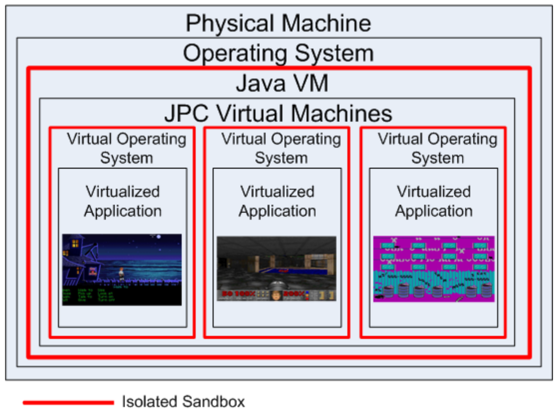


Figure 2: Server Environment, Showing system layers and isolation

To implement a multiuser system a virtualization server must be constructed to create, destroy, handle and provide access to multiple virtual machines. This is one of the most important parts of the system as virtual machine access times will directly affect both performance and saleability. It would need to provide the functionality to save and load the state of each virtual machine and also to save and destroy virtual machines from a disconnected session.

The virtualization software must be component based so that the virtualization server can support different types of virtual machines. This would give the system the ability to support different types of system architectures and would solve requirement 3: "broad digital artifact platform support".

4.1.2 HTTP Server and JavaScript Client Interface

For the interface component of this project we took a thin client approach using the web as the medium. The internet, HTTP, HTML and JavaScript all form a well established platform for online applications. Most consumer network capable devices with displays today can access the web. Devices from mobile phones to computers can all access the web and they do this from a variety of platforms.

By building the client interface in HTML and JavaScript and building an HTTP server into the system, the portability of this application today is maximized. This approach is also useful because the web will still be around for many years to come. Another advantage is that the use of this system would require zero installation as web browsers usually come preinstalled. One issue to consider when using JavaScript is that not all JavaScript processors interpret the language the same way and they don't all support the same functions. For this reason we decided to use a small subset of the language to ensure maximum compatibility. This implementation satisfies requirement 2: "portability" and also requirement 4: "ease of access".

Due to taking the thin client approach with all processing occurring on the server and only interface components transferred to the client, requirement 1:"intellectual property protection" is partly satisfied. It doesn't offer protection over the intellectual property of the images generated by the artifact such as art work in a game. The virtual machine is isolated and given no access of any kind to the outside world, so files cannot be transferred out of the environment. The only transactions from server to client are images representing frames from the virtual machine. The only transactions from client to server are key presses, mouse events and interface commands. Also there is potential for sound to be streamed to the client.

4.1.3 System Interaction

This section related to figure 3 on the following page and demonstrates the steps taken to start and use an artifact.

To start an application:

1. First the client makes an HTTP request.
2. Then they request and are sent the available applications list and HTML interface.
3. Then the they request and are sent the JavaScript interface.
4. Next the client sends a request to run a specific application.
5. This request is passed to the virtualization server.
6. The server then requests the configuration file for the specified application from the repository.
7. The repository returns the configuration file and the virtual machine is created using this file.
8. Next the virtual machine is stored in the VM store.
9. Then a response is sent back from the virtualization server.
10. Finally this response is relayed back to the client.

To interact with an application:

11. An HTTP frame request is sent from the client to the HTTP server with some user input.
12. This is then request is relayed to the virtualization server.
13. The clients virtual machine is then requested from the store.
14. To be returned to the virtualization servers frame processing method.
15. The client input is then applied to the clients virtual machine.
16. Followed by a frame being copied from the virtual machines VGA buffer.
17. The frame is then sent back to the HTTP server
18. Finally the frame is relayed back to the client, who renders it and proceeds back to step 11.

This method of transferring frames demonstrates a synchronised communication protocol as a frame is only returned after input has been applied, therefore the frame is guaranteed to be a response to the input. However the client and server are asynchronous because if the client paused or closed the interface, the server would continue to run their virtual machine.

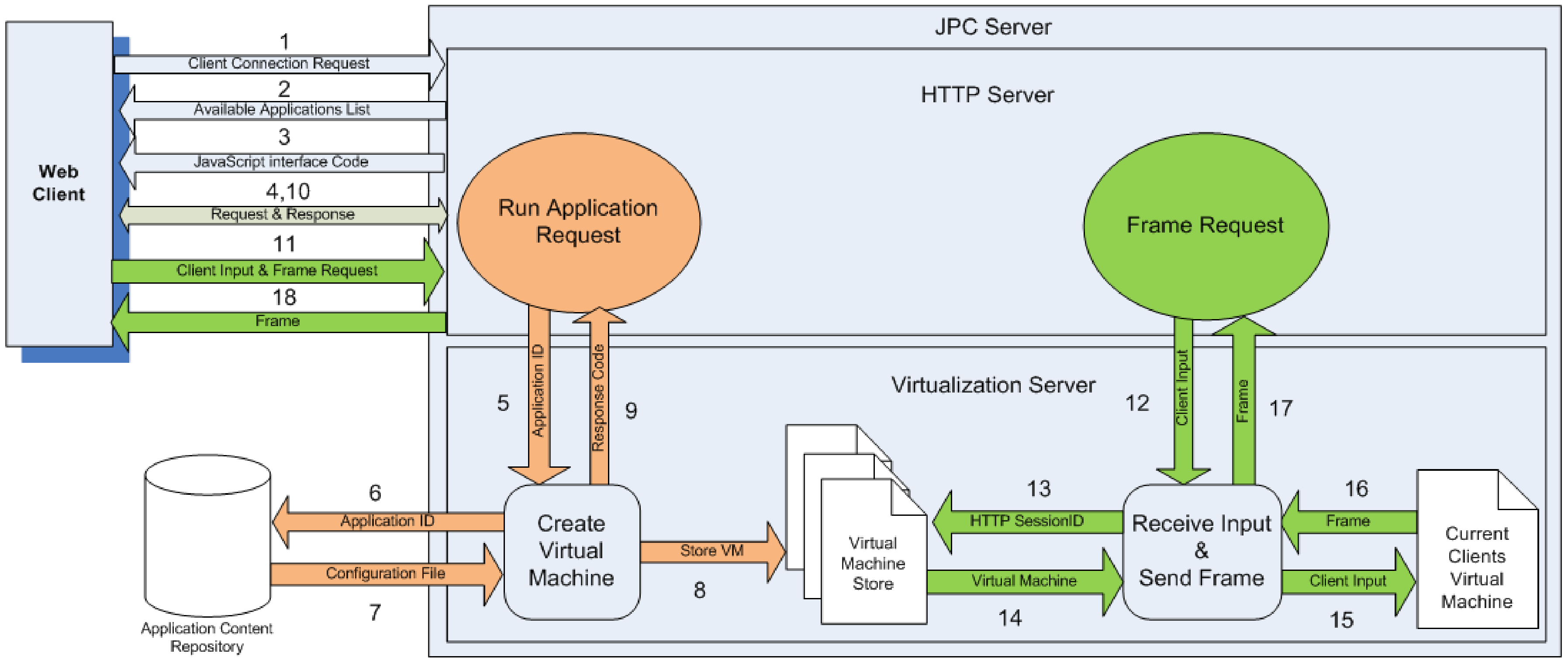


Figure 3: System Architecture Design

4.2 Community Features

To support a community around the artifacts we first looked into what features such a community might require. Abandonware websites were a good point of reference for this task. Typical community features such as chat rooms and forums would be implemented in a practical solution but are not required for this feasibility study. Also multiplayer support for some games was implemented, using a virtual Ethernet hub.

4.2.1 Archiving Meta-Data

Information such as author, company, high scores, Easter eggs, cheats, manuals and guides should all be stored alongside an application and these are also considered digital artifacts. This information is simply text organized into categories and could be added to the archive by the community. It would be stored in a meta-data repository which could take the form of an external SQL server.

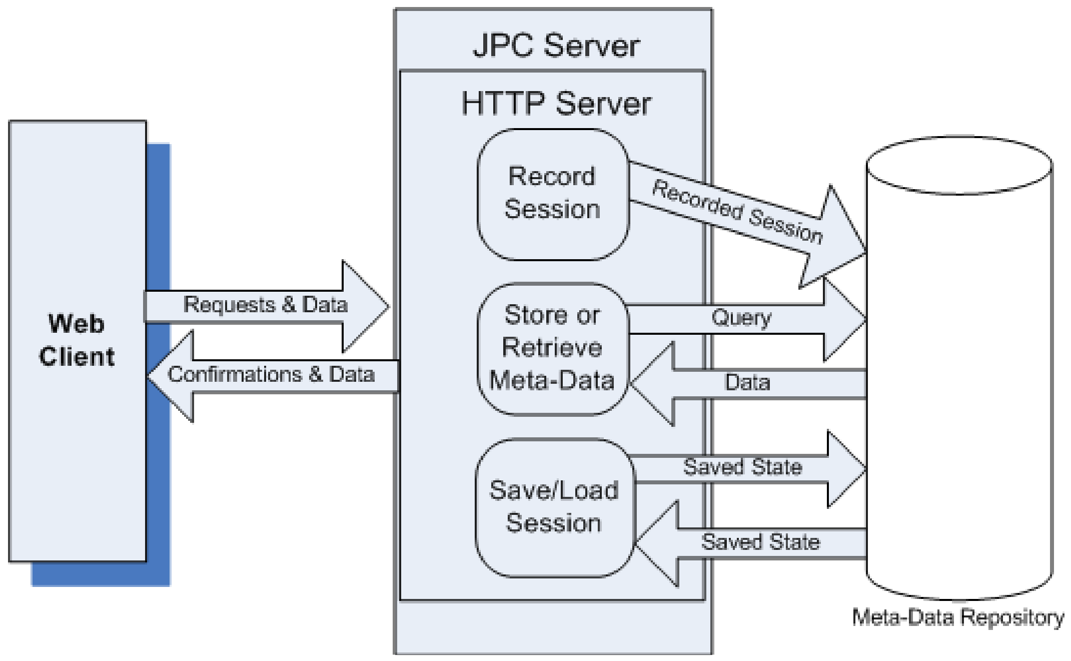


Figure 4: Meta-Data Storage Procedures

4.2.2 Watching Live Games

To further the community features we decided to implement a feature not found on existing gaming communities. The ability to watch a user play a live game has many potential uses. A user can see a list of online users and what applications they are using.

From this list they can select a user and watch their session. Possible uses range from showing off to giving tutorials.

4.2.3 Recording Games

Another practice common to gaming communities is to record gameplay and share it for 'Brag Rights'. Complicated gameplay's are important creative works and should also be preserved. To support this practice a recording system function was built into the prototype and the recording can be archived on the server alongside the application. The recording is also tagged with meta-data describing information like its author and a description. It could also be used as a tool for demonstrating how to use an application or how to access hidden features.

4.2.4 Cloning Games

Should a user be watching another users session and then want to take it over and interact with it themselves they can use the cloning feature. When user2 clicks the clone button in the online users list, user 1's session is saved and loaded into a new virtual machine ready for use by user 2. User 1 is able to continue using the session with only a small delay.

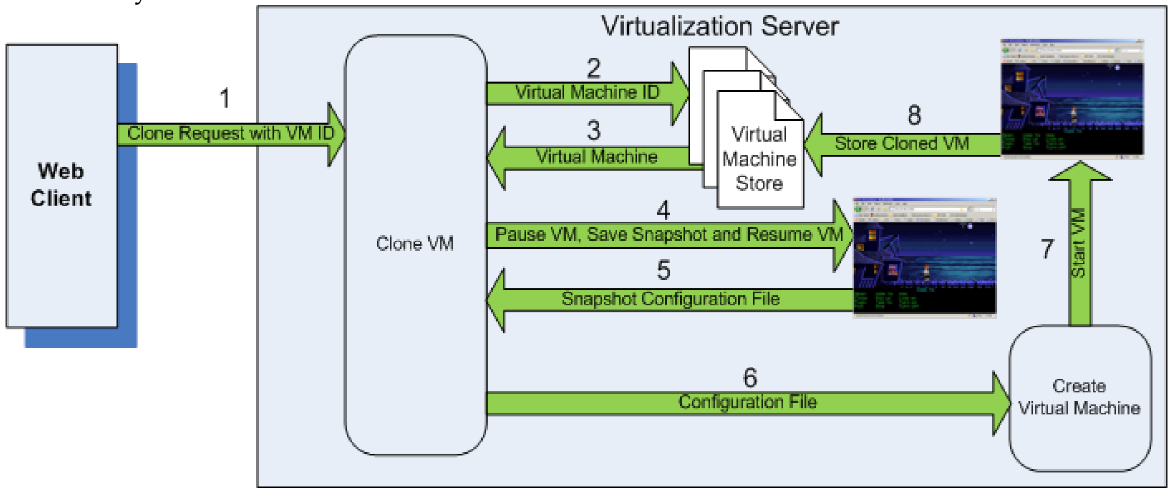


Figure 5: Steps involved in making a clone request

4.3 Summary

This design supports the goals of this project by providing a thin-client virtualization server that can use applications designed for a variety of platforms and also be accessed from a variety of platforms. It does this while protecting intellectual property and supports a preservation community. Its JavaScript, HTML and HTTP based interface make it highly portable and it uses only the most widely supported features of the JavaScript and HTML. For example no embedded objects such as Flash or Java Applets are used. Finally it also supports persistent sessions by saving the state of the virtual machine.

Requirement	Satisfied
Intellectual Property Protection	Yes
Portability	Yes
Broad Digital Artifact Platform Support	Yes
Performance	Yes
Persistent Sessions	Yes
Collaborative Gaming	Yes

Table 1: Design Requirements Summary

5 Prototype

This chapter describes the implementation of this design and the details of how it works. It also focuses on how well this implementation meets the requirements from chapter 3.

5.1 Server

The server component of this prototype was developed using JDK 1.6[3] and a component based architecture. It currently is able to use a wide variety of applications and games and achieves an average frame rate of 15 frames per second over a 54Mbps 802.11g wireless network. It has multiplayer support for the i386 platform through a virtual ethernet hub that connects the virtual machines, this also supports IPX which was tested with Doom[14].

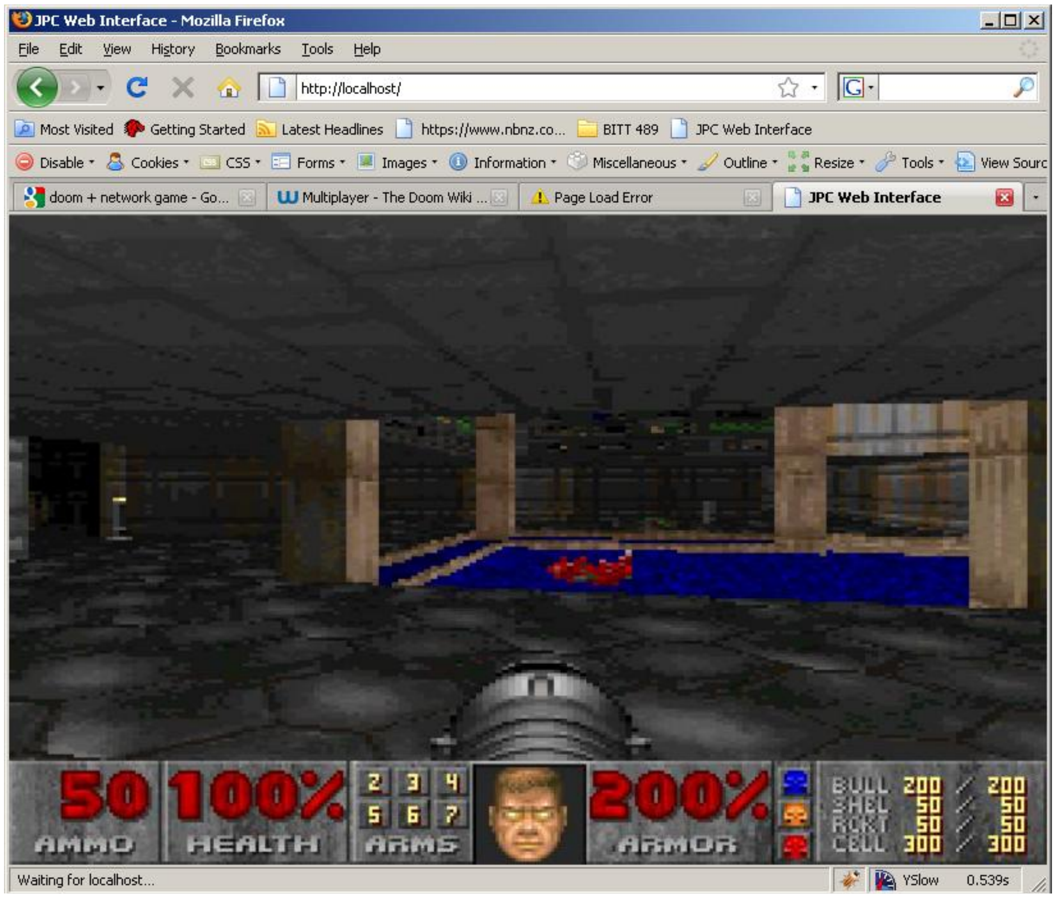
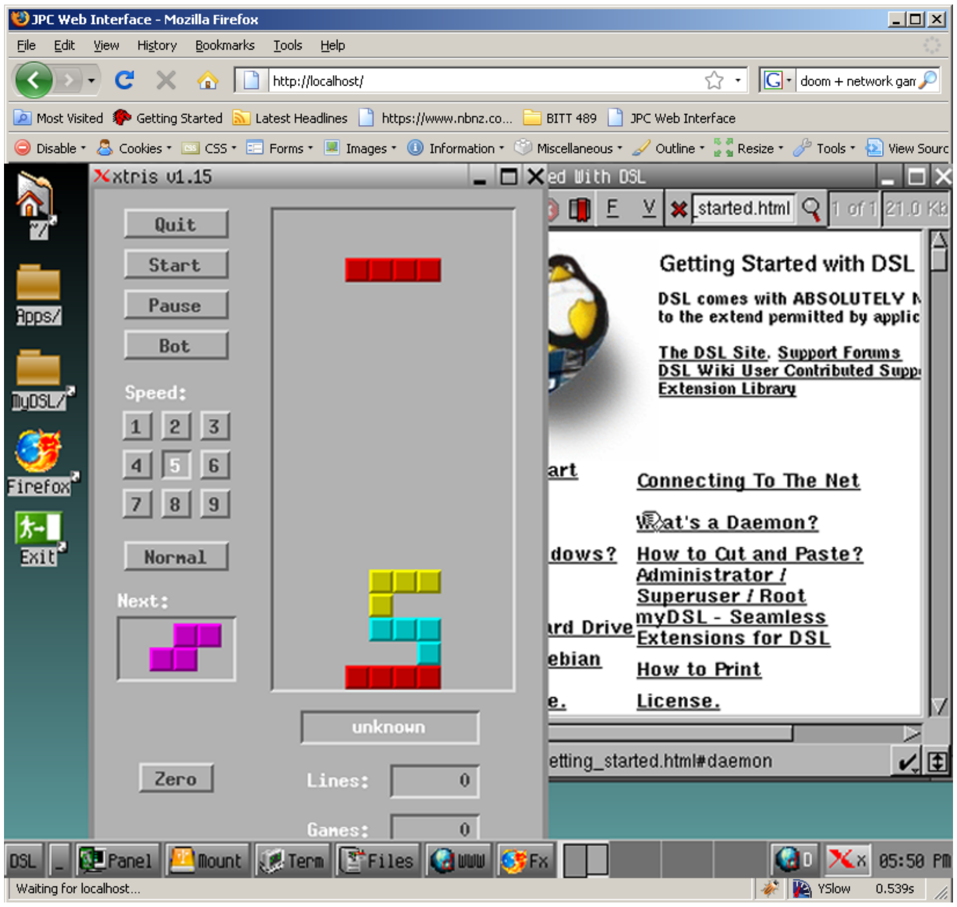

JPC Web Interface - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/
Most Visited Getting Started Latest Headlines https://www.nbnz.co... BITT 489 JPC Web Interface
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Sourc
doom + network game - Go... Multiplayer - The Doom Wiki ... Page Load Error JPC Web Interface
50 AMMO 100% HEALTH 2 3 4 5 6 7 ARMS 200% ARMOR
BULL 200 200 SHEL 50 50 RCKT 50 50 CELL 300 300
Waiting for localhost... YSlow 0.539s

Figure 6: Doom a first person shooter, demonstrating keyboard input.

The HTTP server component is extremely lightweight and consists of only the features and request handling required for processing the thin client interface. It uses HTTP sessions and Cookies to identify users and stores meta-data generated by users in a flat-file or external SQL server. When the client requests a frame from their virtual machine the HTTP server accesses the virtual machine and reads the VGA buffer. It then compresses the frame into a JPEG and sends it out to the client.

The virtualization server uses components to support a wide range of platforms. Currently the prototype only supports three platforms: i386 running DOS, i386 running Windows 3.0 and i386 running Linux. All three are supplied by JPC, a Java based i386 virtualization project from Oxford University. JPC was modified to suit the component based framework and to support multiple instances and clock speed variation. Configuration files are used to describe and build the optimal platform and environment an application requires and is also used to build the list of available applications for the client to select from.


JPC Web Interface - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/
doom + network gan
Most Visited Getting Started Latest Headlines https://www.nbnz.co... BITT 489 JPC Web Interface
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Sourc
xtris v1.15
Quit
Start
Pause
Bot
Speed:
1 2 3
4 5 6
7 8 9
Normal
Next:
unknown
Zero
Lines: 0
Games: 0
ed With DSL
started.html
1 of 1 21.0 Kb
Apps/
MyDSL/
Firefox
Exit
DSL Panel Mount Term Files WWW Fx
05:50 PM
Waiting for localhost...
YSlow 0.539s

Getting Started with DSL

DSL comes with ABSOLUTELY N
to the extend permitted by applic

The DSL Site, Support Forums
DSL Wiki User Contributed Supp
Extension Library

Connecting To The Net
What's a Daemon?
How to Cut and Paste?
Administrator / Superuser / Root
myDSL - Seamless Extensions for DSL
How to Print
License.

etting_started.html#daemon

Figure 7: Demonstrates platform support with DSL linux

5.2 Client

When the client first makes a connection request the HTTP server sends back the JavaScript interface. This interface is also extremely lightweight and consists of a small subset of the JavaScript language, this subset uses only the functions most widely supported by JavaScript engines. A JPEG image is used to represent a frame from the virtual machine as JPEG are common to web applications and require the least time to compress. Input is transferred by capturing keyboard and mouse events and storing them in a local buffer. When the next frame request is made the input buffer is encoded into a URL and this URL is used to update the source location of the frame. The server is sent the new input commands and sends back a new frame in one transaction.

This method comes with the benefit of synchronicity. Since input is sent with each frame request, the user is guaranteed a frame that is in response to the input. This design can be seen in figure 3. The decision to handle frame transactions in this manner was for the benefit of requirement 2: "portability" as it was the best way to perform this transaction using minimal JavaScript and no embedded objects such as Flash. Using minimal JavaScript and no embedded objects makes this interface exceptionally portable as almost all browsers support these functions. Mouse events are captured by listening for changes in the cursors position and simply transferring the delta to the server, thus reducing the size of the encoded URL. Once issue we encountered when designing this interface is the variation of codes used for keyboard presses across platforms. For these reasons we implemented a translation system that detects the platform in use and makes any translations required before storing them in the input buffer.

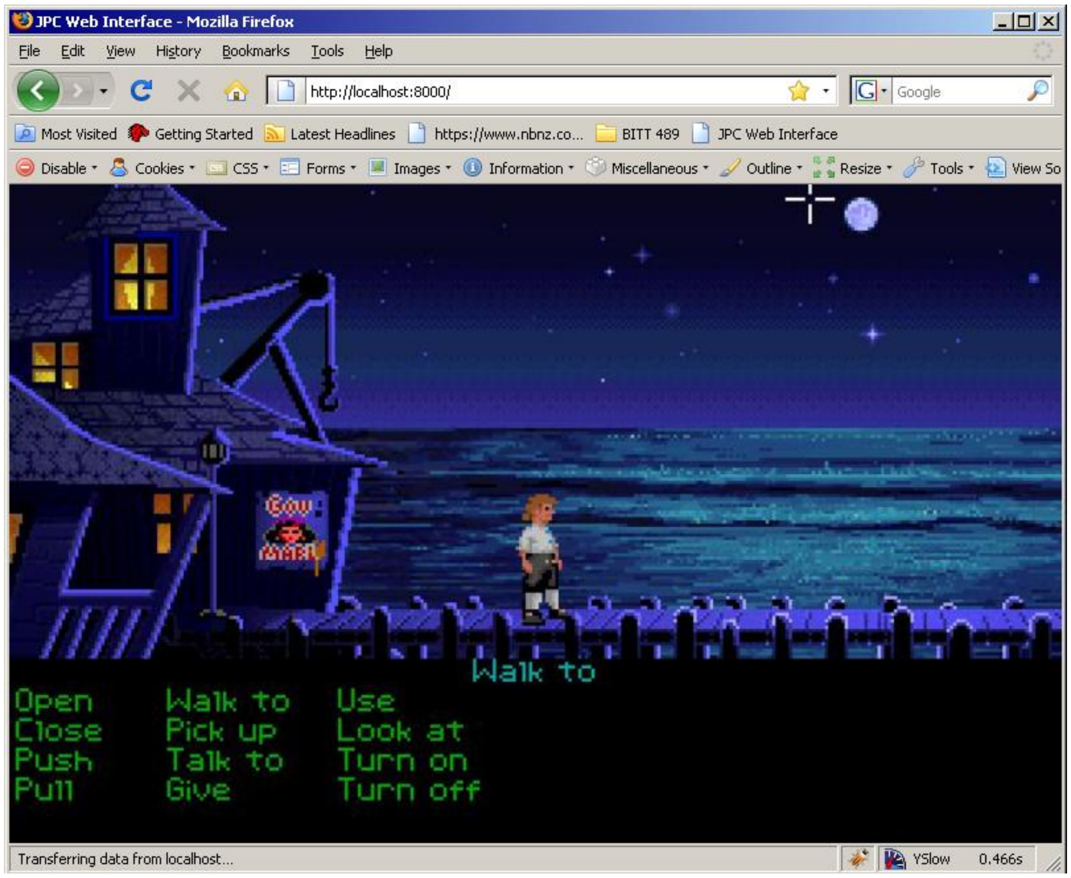

JPC Web Interface - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8000/
Most Visited Getting Started Latest Headlines https://www.nbnz.co... BITT 489 JPC Web Interface
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View So
Walk to
Open Walk to Use
Close Pick up Look at
Push Talk to Turn on
Pull Give Turn off
Transferring data from localhost...
YSlow 0.466s

Figure 8: The Secret of Monkey Island, demonstrating mouse input.

As a result of the thin-client interface, computation requirements are extremely low for the client device. This system can be used exceptionally well by mobile devices such as mobile phones and PDA's. Computation on these devices is often limited and lots of computation can be a big power drain. The interface for these devices required some special translation as touch screens on mobile phones make keyboard based applications unusable.

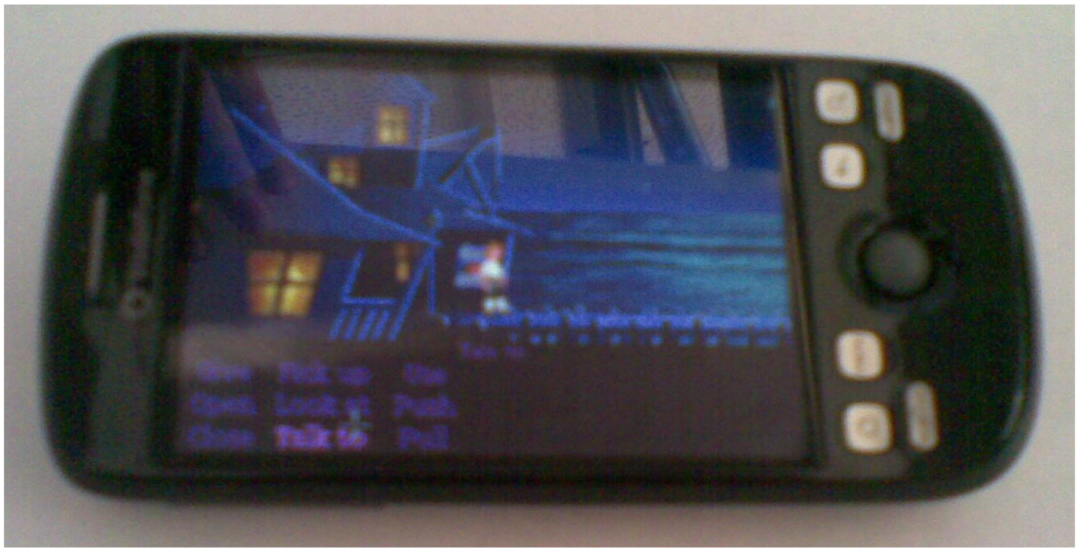


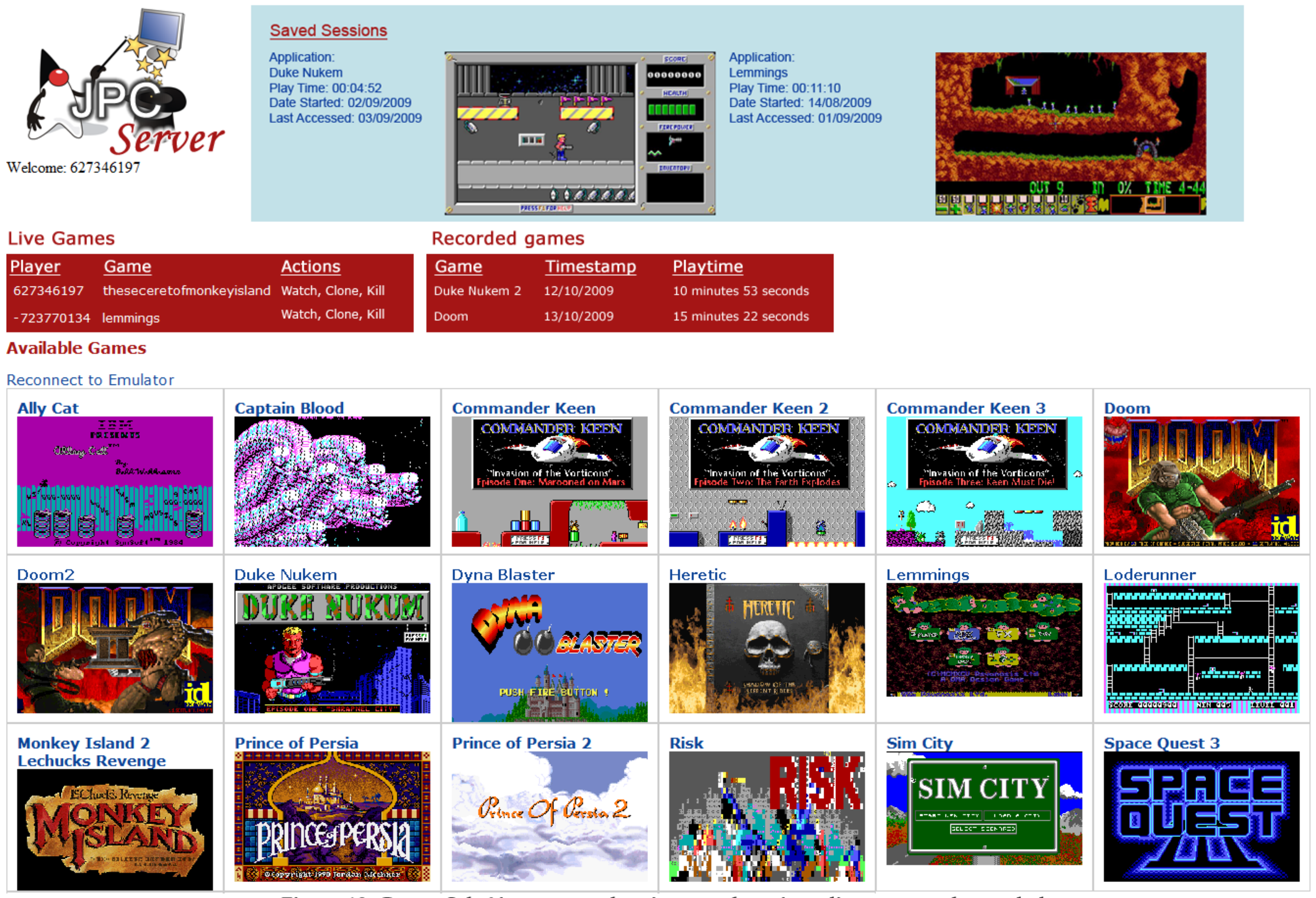
Figure 9: The Secret of Monkey Island, running through Android browser on an HTC Magic Smartphone

The community features such as the games selection list and saved sessions list can be seen in figure 9. At the top of the page is the list of saved sessions with a screenshot taken at the time the session was saved. Next is the live games list and recorded games list. The live games list shows what players are online and what game they are running. It also has buttons for watching or cloning each game. The recorded game list shows all the games that the user has recorded. Finally the bottom section of the screen shows the games selection list with screenshots defined when the game was first archived.

5.3 Supported Features

This prototype has the following features implemented.

- I386 platform support
- Platform independent client interface
- Mouse and keyboard support
- TCP and IPX local network support
- Live session viewing
- Session recording
- Session cloning
- The ability to save and load a session
- Intellectual property protection for execution code


Welcome: 627346197

Saved Sessions

Application:
Duke Nukem
Play Time: 00:04:52
Date Started: 02/09/2009
Last Accessed: 03/09/2009

Application:
Lemmings
Play Time: 00:11:10
Date Started: 14/08/2009
Last Accessed: 01/09/2009

Live Games

Player	Game	Actions
627346197	theseceretofmonkeyisland	Watch, Clone, Kill
-723770134	lemmings	Watch, Clone, Kill

Recorded games

Game	Timestamp	Playtime
Duke Nukem 2	12/10/2009	10 minutes 53 seconds
Doom	13/10/2009	15 minutes 22 seconds

Available Games

Reconnect to Emulator

Ally Cat	Captain Blood	Commander Keen	Commander Keen 2	Commander Keen 3	Doom
Doom2	Duke Nukem	Dyna Blaster	Heretic	Lemmings	Loderunner
Monkey Island 2 Lechucks Revenge	Prince of Persia	Prince of Persia 2	Risk	Sim City	Space Quest 3

Figure 10: Games Selection screen, showing saved sessions, live game and recorded games.

6 Evaluation

Evaluation of this prototype was based on the following requirements first defined in chapter 3.

1. Intellectual Property Protection
2. Portability
3. Broad Digital Artifact Platform Support
4. Performance
5. Persistent Sessions
6. Collaborative Gaming

6.1 Intellectual Property Protection

This prototype uses a thin-client approach to processing the archived applications. By this method the application is actually processed on the server and the client never gets access to the application's executable. This method simply transfers images and input commands between client and server and therefore restricts access to copying the executable through hacking the client application.

Another potential method of obtaining an artifact is by using the emulated operating system to transfer the artifact to a physical system. However since the emulated environment is isolated from all physical networks and the server itself, it is impossible to send files out of the emulated system. This leaves an attacker one last option, which is to show the binary data on screen inside the virtualized operating system and manually copy the data onto a physical machine. However this would be an extremely lengthy process and the simple answer is to remove any tools that could help perform such an action.

Based on these solutions this prototype can be deemed as offering intellectual property protection for archived execution code.

6.2 Portability

As the client interface component of this system is web-based it naturally is highly portable as many computing devices have web access. However there are subtle differences between browsers that can cause JavaScript to behave differently on different browsers. This part of the evaluation will be based on the prototype's performance across multiple platforms and devices.

	JavaScript Interface and Input Methods Supported	Resolution Supported
IE6	✓	✓
IE7	✓	✓
IE8	✓	✓
Firefox 3	✓	✓
Safari 4	✓	✓
Chrome Beta	✓	✓
Opera 10	✓	✓
Konqurer 3.5	✓	✓
Android OS Browser	✗ (Touch screen not supported, but a translation could be implemented)	✓
iPhone Browser	✗ (Touch screen not supported, but a translation could be implemented)	✓

Table 2: Browser Support

6.3 Broad Digital Artifact Platform Support

This project's primary goal is to demonstrate the feasibility of developing a system that solves the problems defined in chapter 3. Therefore this prototype does not need to support every platform for every application but it does need to demonstrate that it can support multiple platforms. In its current form it supports the i386 hardware running the DOS, Windows 3 or various mini Linux distributions. However the virtualization part of the prototype is component based and a developer could create virtualization components for other platforms at a later date.

Based on this ability to run multiple platforms and the component based platform support, this prototype can be deemed as supporting requirement 3: “Broad Digital Artifact Platform Support”.

6.4 Performance

This project’s success is largely judged on the performance of the system as this affects the usability of the application or game being used. If the system cannot provide a fast enough input-to-feedback response time the user would likely have issues using the application or game. To test the usability of the system we played 3 different types of games and measured performance by taking frame rate samples and calculating the average rate and standard deviation. The types of games we evaluated are:

- A game using the mouse as the primary input.
- A game using the keyboard as the primary input.
- A first person shooter requiring quick responses to enemies.

This test was be carried out on a Dell 3.0 GHZ Core 2 Duo with 1GB of RAM. It was performed over an 802.11g wireless network connection with 4 hops to the server. The sample size is 500 and each game was played from the beginning until the sample size was reached.

Game	**Average Frame Rate (fps)**	**Standard Deviation (fps)**
Monkey Island (mouse input)	22.3	5.6
Prince of Persia	18.6	7.6
Doom	18.4	6.2
Average=	19.8	6.5

Table 3: Performance Evaluation Results

This data shows that acceptable frame rates are achievable using this approach over a local network with 4 hops between client and server. Using the research performed by Delwadia et al [2] a response time of less than 150ms can be deemed enough to not effect gameplay. These results show an average frame rate of 19.8 fps across 3 different styles of game. We divide 1000 by the average frame rate to get a response time of 50.5ms. At a confidence level of 99%, the population mean will lie between 19.1 fps and 20.1 fps.

In summary this project satisfies requirement 4: “performance” as it has produced response times well within the threshold of at least 150ms.

6.4.1 Performance Issues

After evaluating the performance of these games we noticed that they were performing slower than expected. They were still quite playable but the frame rates did not match my calculations. Based on the average size of each frame over a 2Mbit network connection we found that frame transmission was accounting for a small amount of the response time. The response time of a frame request depends on 3 factors:

- The time required to access the virtual machine and retrieve a frame.
- The time required to compress the frame into a JPEG.
- The time required to transmit the frame to the client.

Upon further investigation we discovered that server was using 100% of its CPU which caused the slower response. The JPEG compression was expected to use a lot of resources but the virtualization turned out to be the problem. As the applications being virtualized were designed for old systems, their CPU usage was expected to be low. However it turned out that the virtualization and the JPEG compression were contending for CPU time which resulted in a slower client response. This problem was put down to the virtualization component being written in Java. Running a virtual machine inside the Java Virtual Machine will naturally lead to more resource usage than a virtual machine written in a lower level compiled language.

To tackle this problem we first looked into Java's Just In Time optimizations and found that they helped but didn't significantly affect response times. The next attempt to solve this problem was to try using Native Compilation, where Java bytecode is translated into the target platforms native machine code. This approach worked exceptionally well and boosted response times by 40% but still used more resources than we had expected. We came to the conclusion that JPC was not the best choice for virtualization in this project as resource usage has a big effect. JPC also comes with the problem that the clock speed is defined as a final value which makes setting clock speeds for specific games from a configuration file difficult. Unfortunately this value is read quite often in the code and results in slow performance if it is not final. This is because when Java performs its optimizations it will embed final values directly into the bytecode.

6.5 Persistent Sessions

This system has implemented a session saving system that works by saving the state of the virtual machine. Much like hibernation the state of the entire machine is saved and can be re-loaded on the fly. This offers the ability to save any application or game at any point even if the game doesn't support saving itself. This feature has added extra functionality to those games and shows that this prototype supports the saving and loading of sessions appropriately. This has been tested by accessing the system in a browser, starting a new game and saving it. Next the browser was closed, the system accessed again and finally the saved session loaded with gameplay continued.

6.6 Collaborative Gaming

This system was found to support 4 users at an acceptable speed on the machine defined above. It supports community features such as cloning live games, watching live games and recording games. It also supports multiplayer games and since the interface is web based in a real world situation the system could be easily extended to support chat room or forum features. It supports a meta-data repository that can be used to store information about features in an application or game. Based on these requirements this prototype successfully supports a preservation community with meta-data support.

6.7 Evaluation Requirements

Requirement	Requirement Satisfied
Intellectual Property Protection	✓
Portability	✓
Broad Digital Artifact Platform Support	✓
Performance	✓
Persistent Sessions	✓
Collaborative Gaming	✓

Table 4: Evaluation Results

In summary this prototype satisfies all the requirements specified in chapter 3 and successfully demonstrates the feasibility of this approach to digital preservation. This approach offers a highly accessible, multi-platform digital preservation system that protects intellectual property and supports a community of enthusiasts.

7 Conclusion

This project was designed to determine the feasibility of creating a tool to support the archiving of digital artifacts. It would do this by reducing IP-related barriers to their use and offer a legal method for digital archives to provide access to actually use archived software. Just preserving old bit streams is not enough, to truly preserve them in their original form the platform that they were designed for must be preserved too. The first task involved developing a set of requirements that support this goal and then to develop an approach to solving them. The result of this was a proof of concept named JPC Server which has shown that this approach can help digital preservation significantly.

JPC Server has demonstrated a centrally managed, community based preservation system that supports a wide range of digital artifacts and provides a platform independent interface to use them on. It has satisfied all the requirements defined in chapter 3 and is extremely easy to use. Its biggest downfalls are:

- Its lack of sound support
- Its lack of touch screen interface support, limiting its use on many mobile devices. This could easily be solved with a translation script.
- Its i386 emulation runs slower than necessary and should be optimized.

8 Future Work

I would like to further this project by looking into some of its shortcomings and finding a practical solution to them. The issues we have identified are the following:

- I need to investigate latency issues further and see how my implementation scales as more users are added.
- I would like to develop a methodology for taking a general emulator and formally document how someone would modify it for use with this system. At present the implementation only works with one virtualization platform (i386) however this part of the system is component based and components for virtualization of other platforms can be easily integrated.
- I would like to see the system running a newer operating system like Microsoft Windows XP and also running something with 3D acceleration. Recently there have been advances in virtualization that allow direct access to graphics cards, there is even support for SLI. This could potentially take the project to a new level as it could be used as a tool to demonstrate a companies latest software. Instead of trial software, a company could offer its customers access to the full version of an application through a website without fear of it being copied. Time limitations could be imposed and it would require zero installation and download time for the user.
- I would like to improve the user interface for the community features. The system needs
    - Authentication and management of user status.
    - More feedback to the user as to what the system is doing. Currently the system works well but is not very user friendly.
    - Removal of controls (such as Send ESC or Backspace) from the in-application page. This are only needed sometimes whilst using an application and so should be in a pop-up menu of some kind.
- I would like to implement sound support, by adding a streaming sound source into the web interface and turning the server into a streaming sound server.
- Finally we would like to look into optimizing the JPEG compression as an attempt to reduce latency. The bottlenecks in the system are the virtualization, the JPEG compression and the JPEG transmission. Applications for older platforms often used few colours and low resolution. Optimizing the JPEGS for specific platforms could lower compression and transmission times.

9 Acknowledgments

First I would like to thank my supervisors Stuart Marshall and Ian Welch who gave me help, advice and support throughout this project. This project would not have been the same without your input and ideas. Second I would like to thank my friends and family, who have been extremely supportive during this very busy year.

10 Bibliography

1. American Library of Congress, http://www.digitalpreservation.gov/
2. Delwadia, Vipul. Marshall, Stuart. Welch, Ian. Presenting RemoteME a thin-client approach to gaming: Remotely shooting asteroids on a mobile phone using a thin-client approach http://portal.acm.org/citation.cfm?doid=1577782.1577791
3. Java, JDK 1.6 http://java.sun.com/javase/reference/index.jsp
4. New Zealand National Library: Digital Heiritage Archive, Electronic Shepherd – http://www.natlib.govt.nz/about-us/current-initiatives/ndha "Evidence of government funding for digital preservation"
5. New Zealand Government: New Zealand Copy Right Law, http://www.med.govt.nz/templates/Page____7290.aspx
6. New Zealand Government: National Library Act, http://www.natlib.govt.nz/about-us/role-vision/the-legislation-that-governs-us
7. Oxford Universitys Particle Physics Department: JPC the pure Java x86 emulator http://www-jpc.physics.ox.ac.uk/
8. The Official Abandonware ring: http://www.abandonwarering.com
9. Robinett, Warren. Reference for Easter Eggs "Easter Egg" phrase was first defined in: Adventure as a Video Game. ISBN 0262195364
10. J. Rothenberg, "Ensuring the longevity of digital documents," Scientific American, vol. 272, no. 1, pp. 42-47, 1999.
11. Ward Larry, Virtualization using NVIDIA Graphics rendering: http://forums.techarena.in/monitor-video-cards/1151452.htm
12. Wikipedia: Emulators http://en.wikipedia.org/wiki/Emulator
13. Wikipedia: http://en.wikipedia.org/wiki/OnLive
14. Wikipedia: Doom http://en.wikipedia.org/wiki/Doom_(video_game)