Displaying Early Software Online: Challenges and Outcomes
(Summer Scholar Report, Feb 2015)
by Sarah McKenzie

Why Emulate?

Software from the early personal computer era is of interest because it is a historical record of what was being done with computers at the time. People developed their own software for home or business use on 8-bit computers all over the world, and these homebrewed software solutions are important parts of the historical record, as well as examples of early software development. It is easy to convey the functionality of an application, since the application is likely similar to ones in use now, but emulation also lets people see what early software looked like and how it handled, and allows people to compare these early works to current applications.

Games are harder to demonstrate through images and description. There are styles of game and design choices that are simply not used any more, and some of the limitations of the old systems are not apparent to people who have never used them. For example, the limited number of colours available on a ZX Spectrum might seem to be a design choice. Emulation offers a way for people to experience a game made for older platforms without having to set up difficult to find computers and let members of the public handle hard to replace equipment.

It is particularly important for the early software produced in New Zealand and Australia. Though some early software made in these countries is well known, other examples are obscure or hard to find. As time goes one more and more people will not know about the early home computer software that was available, and information available tends to be about the most well-known, usually northern hemisphere, products. Emulation lets people experience the games for themselves, and increases awareness of early software. This helps to preserve software from the early days of the home computer era and provide insight beyond a textual explanation.

Some of the games that have been selected for display online for this project are particularly interesting. The Hobbit, by Philip Mitchell and Veronika Megler, seems to be a simple text adventure with graphics where the player takes on the role of Bilbo Baggins on the way to the Lonely Mountain. The reality is that the non-player characters such as Gandalf and Thorin are agents that can move freely in the world and do not follow a set path.

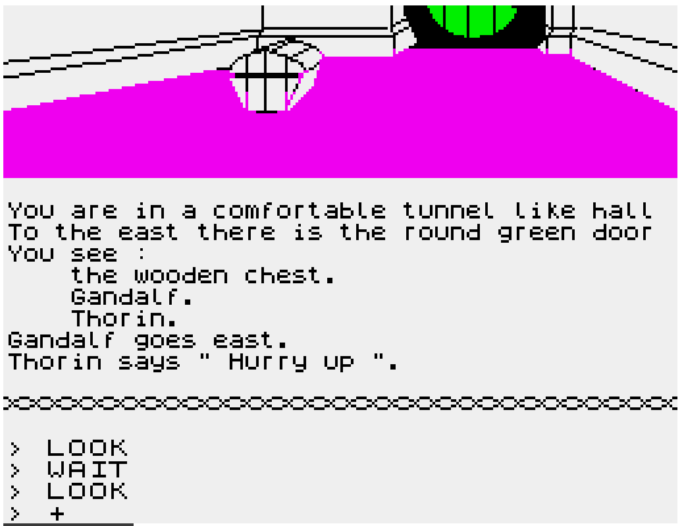

Illustration 1: Gandalf is tired of waiting for the player, and has wandered off into the wilderness. (screenshot from Spectrum version)

This is a fairly radical idea in a text adventure, and is more complex than most people would give a game published in 1982 credit for. The game's cassette inlay claimed it was possible for the characters to fight and kill enemies that player had not met yet. This is something that would surprise people who have no prior knowledge of the game, and it is certainly interesting to see the characters coming and going from a location while trying to accomplish tasks.

Another game that is fascinating is The Way of the Exploding Fist, by Gregg Barnett. It is an early fighting game that features two characters in a series of karate bouts. The game features excellent animations, and is quite fast paced.

Illustration 2: As can be seen here, a static image conveys no understanding of the excellent animations and exciting gameplay. (screenshot from Spectrum version)

So, emulation provides insight into early software that simple description fails to convey. But for emulation to be an accessible method for educating people about early software, it must be accessible. Traditionally emulators have been applications run form a user's own computer. This is acceptable for the individual who has the time and inclination to set up a system, but does not help people who simply wish to see the software working, and do not possess the desire to figure out how the program works – or the means to download an application in a controlled environment. Thus, browser based emulators are more useful for this purpose.

Games Considered for this Project

Title	Platform	Publisher
The Hobbit	Spectrum/C64	Melbourne House
The Way of the Exploding Fist	C64	Melbourne House
Halloween Harry	Microbee	Honeysoft
Emu Joust	Microbee	Honeysoft
The Search for King Solomon's Mines	C64	Softgold
Laser Hawk	Atari	Red Rat Software
Necromancer II	C64	Type in Listing

Burglar Bill	Sega SC3000	Poseidon
Fire King	C64	SSG
Raft Away River	C64/Microbee	Jacaranda

Some of these are well known and popular, and can be found displayed in emulators online, while others are not easy to find copies of online.

Emulators

An emulator is a piece of software that runs the operating system of an old computer, as a virtual machine. This means it can contain a virtual recreation of the old computer's hardware and also the software that ran it – older computers that did not boot to a desktop environment still have an operating system, it is the system that allocates memory and controls the hardware components. To us today there is a considerable distance between the operating system we interact with and the lower level system, but for 8-bit computers it was often possible to interact directly with the memory allocation from the top level software that was running at the time. For example, some systems let people interrupt the execution of a game and enter commands, two of which were very useful: PEEK, which displayed the contents of memory locations, and POKE which inserted a value into a memory location. Using these let players of games add lives, adjust their scores, or modify the game code to prevent collision detection, and then resume play. An emulator that runs a proper virtual machine will allow people to do this.

Emulators are usually specialised software, written to be identical in performance to the older machine. It is possible to have a generalised program that emulates several different machines that can be chosen as needed. The question is then, is it better to use a specialised emulator for each machine, or a generalised one.

Emulators can be written to run locally on a user's own computer, or be embedded in a web browser so that people can experience the software from the system being emulated without having to go to any trouble installing programs and finding copies of the old software they wish to try out. The majority of emulators online are focussed on games, rather than applications for productivity. There is little interest in an emulated word processor (however, it is always possible that some stubborn people may still be using old software to write documents).

Identifying Emulators

It is difficult and time consuming to write an emulator for one system, let alone several. However, some enthusiasts create them on their own, and provide them for use free of charge online.

Web search is a primary method in finding emulators. The retro computing communities often collate lists in articles or forum posts which have links to many examples of emulators (in the course of this research a link to an emulated version of ENIAC was found, but sadly the link was dead). There is particular interest in browser-based emulators since they are easier to use for people who want a "quick fix". The downside is that they often do not support users running whatever they wish on them. Unfortunately the community managed lists are often not maintained, and emulators that have been taken offline are not removed from the lists.

Fortunately, the sheer volume of emulators that have been made over the years means there are usually some emulations of a given platform available to test for suitability.

Emulators and Software

Emulators do not load software directly from a physical medium such as tape or disk. Instead they use images of the data on the physical medium (a *tape image* or a *disk image*) to load software. In some cases this has been superseded by use of a *memory snapshot* which contains a dump of the memory of a system, usually when the game has first been loaded. Memory snapshots are particularly common for the Spectrum.

Loading Software

To provide users with the experience of using an early computer, an emulator can be built to provide the experience of loading software from a virtual drive. The desktop application Steem (an emulated Atari ST – a 16-bit computer which is not part of the current work) does this: It has a menu outside the emulation where users select which disks are in which drive, and then a virtual power button turns the emulated computer on and loads the disk from the virtual ST's A drive. This is an interesting exercise for the user, but the actual loading is unnoticeable – the emulator runs fast, and it is emulating disk load time, which is reasonably quick. A simulated cassette load would be educational for users the first time, but quickly put them off exploring further examples of early software if there was no means to bypass this. In an ideal world the emulator would also offer a way to load the emulated system without any software, to get to the BASIC prompt and let people type in their own code.

There are two models used for most emulators on the web. The first is to try and simulate the experience of loading software from a disk or cassette (*User Selection*). The user picks the disk image or memory snapshot, and the emulator loads it. This is not a direct analogy for the original loading methods: If a memory snapshot is loaded then it immediately works (a memory snapshot is just the contents of the virtual machine's RAM at the time it was taken, and can simply be assigned to the virtual RAM directly). This is a useful method, as it shows that old computers were general purpose machines which could load different pieces of software.

Another method of loading would be to have the software embedded with the emulator (*Embedded Software*), so that a single instance of the emulator has the software running. For example, the emulator called Hob has a separate page for each game it demonstrates, and when that page is loaded that one game is presented without any user intervention. This is a useful way to demonstrate individual pieces of software and removes the need for users to do anything before being presented with the emulated software. An example of this is provided by the author of Qaop (Bobrowski, 2011) using the *param* tag to provide the emulator with the file to load.

```
HTML example:
-------------------------------------------------------------------------------
<applet name=qaop archive=qaop.jar code=Qaop width=320 height=256>
<param name=load value=demo.tap>
<param name=ay value=1>
</applet>
-------------------------------------------------------------------------------
```

Illustration 3: The param tag provides parameters for the applet tag it is enclosed in. Here, the emulator will load a tape image called demo.tap

Emulator Testing

A variety of emulators were tested, however the availability of javascript emulators for some 8-bit platforms is minimal. In particular, there are many 8-bit Atari emulators but almost none that run via the web – almost all are standalone applications for use on a local machine, requiring download

and possibly some work to run beyond simply double clicking a file (as such they were dismissed immediately).

Evaluating emulators for suitability is difficult, since it can require a working knowledge of the given systems. Martignoni et al, (2013) show that it is possible to do automated testing of emulated hardware, but this is a low-level testing for accuracy. To test the accuracy of an emulator that will be used for running games it is useful to have an idea of whether or not the game itself is running properly

Assuming it is possible to acquire the old hardware in working order and have the software to run on each platform, some proposals for approaches to evaluate emulators are:

Framerate Comparison: To do this, one would take an original system, and link it up to a video capture system. Play a game on the original system and record the images. For each emulator, replicate the sequence of events while capturing video (possible to do locally on a modern PC). Analyse the frames from each video feed to see if they match. In theory this should offer a way to see that the emulation is performing the same as the original.

The problems with framerate comparison beyond finding an original system and software is getting the same series of actions in a game. It may not be possible to replicate the exact sequence of commands in the same rate for someone who is not experienced. Mitigating this by using short sequences instead of a long run of playing may allow for replication between the original platform and the emulator. Some games are not suited to replication (e.g. games with random elements – Space Invaders and Galaxians features random behaviour by the enemies) and so games with fixed sequences in the game world where the player can replicate their commands are better (e.g. Boulder Dash, Manic Miner).

Domain Expert: A domain expert is someone who has a reasonably good to expert level of experience and knowledge of the subject. For each system being emulated, the domain expert would have to play some games they are very familiar with and fill in a report on how well they think the emulation matches the real system. Ideally they would be able to re-familiarise themselves with an original system before sampling the emulators.

An initial problem with this the different controls some emulators use. For example, Spectrum games were all designed for keyboard controls, but Atari games were all designed for joystick – modern PCs do not require joysticks and so emulators map keys to the joystick controls, which can change how the games handle. Domain experts might have a bias towards the old system, making them overly critical of the emulators. Given the long time period it can be hard to find people who have maintained their skills (though some games are so simple it just takes a few warm-up rounds to get back to the level of skill required). Another problem is that it can be hard to find people who are experts in the software that will be demonstrated. Another issue is that there are not many people who are experts in using more than one system: The cost of computers in the 1980s was very high and no-one would have bought more than one model of computer. It may be possible to find people who have a reasonable level of expertise in one system and a passing knowledge of others that lets them evaluate systems.

General Problems with Evaluation: In all cases there are barriers to a good evaluation of emulators. In some cases an emulator is able to run some games well, and others badly. This can be an artefact of running a virtual machine simulating the hardware – some older software was built on a particular system and may rely on tricks that work with that system's hardware. This problem existed with the original computers, for example porting a game from one 8-bit platform to another could lead to the port being slightly different despite the hardware capabilities being relatively

similar. This could even happen when a computer game was moved from the European market to the United States, due to differences in screen refresh rates, as Archer Maclean noted with the bootleg copy of his game Dropzone (Hague, 1997). Since this is a known problem from the early days of the home computer era, it may be enough to simply find the best possible fit for the combination of system being emulated and software to run on the emulator and explain that it's as close a fit as possible, and lay out the hardware and software limitations – these limitations are part of the history of early software and so having them present is unfortunate, as opposed to being a disaster.

Evaluation involving human assessment is tricky, since there is always the chance that people will reject any emulation due to feelings of nostalgia. Conversely, reading retro computing forums to see which emulators people like may not be useful – people are unlikely to criticise a working emulator for being less than perfect, since the difficulties are known to these communities, and there is no desire to alienate people who work for free to provide others with the means to enjoy old software.

Subsequently, it is assumed that a "good enough" approach is suitable, since it is not possible to find perfection without a dedicated team of programmers with first hand knowledge of a system to develop an emulator.

Method

For the purpose of this research, it was assumed that the researcher's knowledge of 1980's computer games would serve as a good starting point to evaluate emulators. It is possible to find web based emulators that run the games this project wishes to put on display (The Melbourne House version of The Hobbit is very popular).

Emulators were identified through web search and reading retro computing forums and blogs, and for each emulator some games the researcher was familiar with were tested to see how well they played. A limitation in this process was the difficulty in testing some emulated systems using keyboard controls (up, down, left, right, and fire) when the original games were played using a joystick – the design of the games often relies on that input paradigm and thus do not play as well using keyboard commands. On the other hand, some systems (e.g. the Spectrum) had many products available which assumed keyboard controls would be the default used by most consumers.

Consideration has been given to whether or not the emulator is open source, how the emulator loads software, and any particular notes of interest. Performance on games was considered first on any that the emulator had that fit the needs of this project, and then games that are particularly well-known or interesting.

Findings

The following table describes the best emulators that were found during this research.

Emulator	Location	Platform	Evaluation
Hob	www.twinbee.org/hob/	ZX Spectrum	No sound, uses SNA and Z80 memory snapshot files to load the games. Tested using The Hobbit (runs well, draws the screen reasonably quickly), Way of the Exploding

			Fist (seems slow but may be an artefact of the original game). Hob is available for use freely and has a variety of options for using, such as using their hosted version or getting a copy. Loading: Embedded Software. Hob also includes a version running the BASIC prompt: http://www.twinbee.org/hob/play.php?snap=basic
j64	www.live-id.org/j64/	C64	Creates a security alert when running due to not meeting the Java 7 security requirements. This can be circumvented by adding the site to the Java exception list but this is not something all users will be willing or able to do.
jsA8E	http://atariage.com/forums/topic/224709-jsa8e-javascript-atari-800-xl-emulator-online/ (link is to a forum post providing links to various emulated games)	Atari 8-bit	A Javascript, browser based version of the author's standalone emulator. Loading: Embedded Software Tested using: Montezuma's Revenge (plays well), River Raid (jerky, slows down considerably on plane refuelling), Dropzone (agonisingly slow), Drol (crashed on the second level, version here lacked colour). Occasionally crashes, has long load times, and struggles with multiple animations happening at once. No sound.
Javascript Commodore Emulator	www.kingsquare.nl/jsc64	C64	Some problems loading on some systems/browsers. Open source. Instructions for use are provided on the emulator's homepage.
JSMESS	http://www.archiveteam.org/index.php?title=Javascript_Mess	All (theoretically)	Javascript port of MESS (Multi Emulator Super System). Only in beta and only has some demos running. Would be ideal for this project in a final version.
JSSpeccy	jsspeccy.zxdemo.org/	ZX Spectrum	Open source emulator written in Javascript, with good instructions

			provided for setup on Github. Has limitations on multicolour display in emulation. Testing using: The Hobbit (sometimes doesn't recognise keypresses when typing fast), Jet Set Willy (visible screen tearing when moving), Ant Attack (incredibly slow). Can run as a 48k or 128k Spectrum. Loading: User Selection is shown on the page, but it is possible to make the emulator automatically load a specific tape file. Appears to be possible to set up the emulator to require people to go through the Spectrum commands to load the tape file into the virtual Spectrum.
Nanowasp	http://www.nanowasp.org/	Microbee	Javascript emulator, but requires using a makefile to compile. Loading: Ostensibly simulates User Selection, via a menu. Each piece of software is stored separately but the name of each file must be hard-coded into the emulator before compiling. Tested using: Towers of Hanoi, Laser Blazer, Breakout – all run well, but lack of
Qaop	wizard.ae.krakow.pl/~jb/qaop/ (Java version) http://torinak.com/qaop/games (Javascript port)	ZX Spectrum	Open source, written in Java and embedded in a page. Released under GNU GPL Licence. Tested using: Jet Set Willy (sound stutters, emulator particularly slow), Starquake (runs fast). Loading: The Java version features User Sleection. The Javascript port uses Embedded Software. Binaries and source are provided for the Java version. Good documentation. Handles memory snapshots and tape files. Java version runs into security issues on some machines.

Emulators Selected

Despite the disappointing turnout for some platforms, there are an abundance of Spectrum emulators. Some of these are simple to use and offer permission to use them as an embedded copy of the emulator. C64 emulators tend to be unruly and not work consistently, if at all. Atari emulators are almost always desktop versions that are not suitable for this project. The Sega SC3000 was not represented at all. The sole Microbee emulator was found to be very good, but difficult to set up.

Some emulators that did not work threw security errors. Though is is possible to disable the security settings on a local machine, this is not a step it is reasonable to expect users to take.

Atari 8-Bit: Only one working browsr-based Atari emulator was found, and jsA8E has some major problems. The slowness of the emulator is noticeable, with some games that were fast on the original Atari being difficult to play due to the slowdown. Load times were unpleasantly slow, and the crashing was unfortunate. JsA8E is also not open source, and so contacting the developer would be required to gain a copy of the source code if there was to be any work done on improving the emulator.

Commodore 64: There were no good C64 emulators, with all the ones viewed either being very slow or running poorly on different platforms. Further searching for compatible emulators will be required.

Microbee: Nanowasp is the only real option for an emulated Microbee. Building a copy of this requires downloading the source code repository and running a makefile to compile the Javascript files into a usable configuration (Javascript has no way to include the contents of one Javascript file in another, and development is easier when there is not one huge file). However, it was found during this research that the source repository on Github, an internet based version control system, does not compile properly if the repository is downloaded as a ZIP file, despite other Javascript projects working fine.

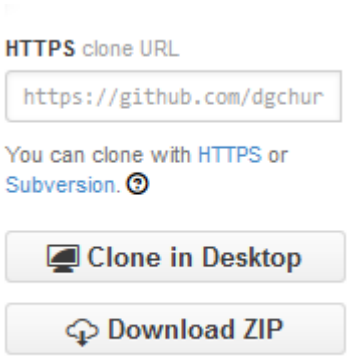

Illustration 4: Github's "Download ZIP" option did not provide a usable copy of the source code in this case.

Some research showed that there was a problem with the makefile checking the Git repository history during compile, which meant that the project did not compile due to the ZIP option not providing those files. Cloning the repository from the command line did reconstruct the source ocontrol history and allow the emulator to compile. Unfortunately, Once this was done the emulator was not testable on a local version (which should be possible with a Javascript based site).

However, the version online shows that it is a good option for emulating the Microbee. The compiled version was provided in case it was possible to make it work by putting it on a web server.

SC3000: It was not possible to find a working SC3000 emulator.

Spectrum: For the Spectrum, Hob is a good choice since the author offers the ability to embed the emulator, and offers to provide a copy to anyone who asks. The emulator itself has an emulated copy of The Hobbit, which the emulator was created to run (at: http://www.twinbee.org/hob/play.php?snap=hobbit), and this game runs well in Hob. Hob is limited by not having proper sound, but this should not matter with The Hobbit. Hob also has a version running a BASIC prompt (at: http://www.twinbee.org/hob/play.php?snap=basic).

For further research on a good Spectrum emulator that involves emulator development, Qaop is the option to take. It has sound, and allows numerous features. Qaop is a slow and clunky emulation of a Spectrum, but the open source nature of the project means it can possibly be improved.

Further Research

Finding a working emulator for some systems was difficult, and so it is a priority to find one for the C64, Atari, and SC3000.

Modifying the source code for open source emulator projects may be feasible as a goal, but it may be hard to find people with the expertise to do so. This is an expansion of the project that may be difficult to do, but might provide better results.

JSMESS would be ideal and should be followed for future developments – as a single package emulating any platform it is the most efficient solution, however it did not seem to be at a usable stage yet.

References

Bobrowski, J., 2011. *Qaop – ZX Spectrum emulator* [online] Available at: <http://wizard.ae.krakow.pl/~jb/qaop/> [Accessed February 28, 2015]

Hague, J., 1997. Archer MacLean in *Halcyon Days: Interviews with Classic Computer and Video Game Programmers*. [online] Available at: <http://web.archive.org/web/20150206175750/http://www.dadgum.com/halcyon/BOOK/MACLEAN.HTM> [Accessed February 28, 2015]

Martignoni, L. et al., 2013. A methodology for testing CPU emulators. *ACM Transactions on Software Engineering and Methodology*, 22(4), pp.1–26. Available at: http://dl.acm.org.helicon.vuw.ac.nz/citation.cfm?id=2522920.2522922 [Accessed February 28, 2015].