Software migration applied to Commodore BASIC video games

Craig Harrington
Flinders University
Sturt Road
Bedford Park, SA 5042
craig.harrington@flinders.edu.au

Denise de Vries
Flinders University
Sturt Road
Bedford Park, SA 5042
+61 8 8201 3639
denise.devries@flinders.edu.au

ABSTRACT

Migration as a means of preserving legacy software is an under-utilised approach which offers an alternative perspective to the more commonly used paradigm of either system or application level emulation. A prototype system to translate legacy BASIC software for the Commodore VIC-20 computer to a combination of HTML5 and Javascript was developed and serves as a basis for ongoing work.

The preliminary system has been tested by performing an unassisted (automatic) translation of a Melbourne House title, "Grand Prix" (Ramshaw, 1983), to Javascript. This game was found to run at an acceptable frame rate within a web browser on a relatively recent Core2 Duo system. Future work will involve extending the system to work with a greater range of BASIC programs and optimising the generated Javascript for readability and performance. Eventually titles which include machine language subroutines and also games written purely in 6502 machine language will be supported using a similar framework.

Keywords

Software preservation, migration, translation, BASIC, digital heritage, videogame

INTRODUCTION

This work seeks to document issues involved in the development of a software tool for the migration of legacy video games to a contemporary language, suitable for running within a web browser. This work comprises a part of the "Play It Again" project, a multidisciplinary effort "conducting research into the largely unknown histories of 1980s game development in Australia and New Zealand, ensuring that local titles are documented, preserved and make it into national collections" (Swalwell & de Vries, 2013).

Emulation has been used extensively to provide access to legacy titles, with notable success, however it is not a panacea. Each new iteration of computer hardware requires the emulators to be rewritten (or emulated themselves), unless some solution along the lines of Rothenberg's (1999) Emulation Virtual Machine is employed.

The use of software migration as an alternative technique to provide access to legacy titles is less common, however may offer some benefit. For example, emulators typically require proprietary software, such as a copy of the operating system for the platform in question. This practice places the use of emulation in a legal grey area, where legal action is possible, although unlikely to be carried out in practice. Migration may circumvent this issue by translating the legacy operating system library calls to a modern equivalent, providing similar functionality.

The BASIC programming language was often the language of choice for homebrew software coded by home computer enthusiasts and hobbyists. Swalwell and de Vries (2013) echo Barnes' (1982) assertion that "whilst BASIC was sometimes maligned as a crude and even 'disgusting' language for amateurs, it may be that as a language, it survives the passage of time better than 'professional' languages". In particular, so-called "type-in" programs published in hobbyist magazines were quite often written in BASIC.

The Commodore VIC-20 system was chosen as the legacy platform of interest due to its relative simplicity and similarity to the more popular Commodore 64 platform. The version of the BASIC programming language used in each machine is identical. The VIC-20 contains a more primitive graphics chip, which lacks tile-based (or sprite) graphics, making it less complex to implement.

A prototype system for migration was constructed in Javascript, containing enough functionality to allow for the execution of BASIC programs. An HTML5 Canvas element provides a mechanism for character graphics to be displayed which were obtained by examining the data from a read-only memory (ROM) dump of the original machine.

A Javascript function library provides equivalents to various BASIC statements. Initially, the functions were tested by manually translating the legacy code from its original form to Javascript. Overall, the Javascript functions aim to replicate the functionality of the various BASIC statements but some differences are unavoidable. For example, keyboard input is event-driven in the browser environment rather than polled continuously, as in the original hardware (or an emulator).

A parser and lexer for Commodore BASIC V2 was constructed using the Python scripting language by extending an existing Dartmouth BASIC interpreter which was included as an example with the PLY lexer and parser package (Beazley, 2001). This involved adding in support for additional keywords, relational operators and also string and integer data types.

Another Python script processes the parsed program and outputs the equivalent Javascript function calls. Due to the absence of branching instructions in Javascript, a switch construct was used to achieve flow control, with a case statement for each line of the original source. This approach is analogous to that of the Sega Genesis Java translator (Delwadia, 2007).

Strictly, only the targets of branches are required to be separate cases in the switch statement, however, including more case statements than necessary allows for finer granularity in the timing of the translated program. Since BASIC programs are interpreted on the original hardware (or an emulator), calculating the precise number of CPU cycles (and hence, time) a given BASIC statement takes to execute is not trivial.

At the time of writing this abstract, the naive approach employed is to assume every line of the program takes the same amount of time to execute, irrespective of its content. The effect this has upon gameplay is negligible, being mitigated by the fact that BASIC programs tend to be less responsive to user input than titles written purely in machine code, in any case. This makes user evaluation slightly problematic since even titles running on original hardware may be judged poorly as a result.

Future work will involve extending the system to work with BASIC programs which include machine-language subroutines and also games written purely in machine language. The graphics routines may be rewritten to use libraries with better performance, such as WebGL. Sound capabilities have yet to be implemented. In addition, it is hoped that the unstructured BASIC code can be manipulated into a more structured form, taking advantage of the more modern control-flow mechanisms in the Javascript language, whilst retaining the semantics of the original code.

BIO

Craig Harrington holds a Bachelor of Science (with Honours) from Flinders University. He is currently studying full-time as a PhD candidate in the School of Computer Science, Engineering and Mathematics at Flinders University. His current research is focused upon the development of software tools to migrate legacy applications to contemporary languages, as a part of the "Play It Again" project.

Dr Denise de Vries has, since the early 1980s, developed commercial complex database systems for a variety of industry domains including local government, airline, public libraries, para-medical, diet and nutrition, commercial cleaning, automotive parts manufacturing, building and construction, inventory control, conference organisation and commercial photography. Based on this industry experience she researches techniques to preserve digital history and data semantics including techniques to deal with changes to information in a database such as structural change, semantic change and constraint change.

ACKNOWLEDGMENTS

The authors would like to thank the following organisations for their support: The Flinders University of South Australia, The University of Melbourne, Victoria University of Wellington, Australian Centre for the Moving Image, The New Zealand Film Archive and Berlin Computerspiele Museum.

"Play It Again: Creating a Playable History of Australasian Digital Games, for Industry, Community and Research Purposes" is supported under the Australian Research Councils (ARC) Linkage Projects funding scheme (project number LP120100218).

BIBLIOGRAPHY

Barnes, J. G. P. (1982) "The Development of Cultures in Programming Languages" in Ninth Australian Computer Conference (Sale, A.H.J. & Hawthorne, G., eds), ACS. pp. 9-21.

Beazley, D. M. (2001) PLY 3.4 (Python Lex-Yacc) software, URL: http://www.dabeaz.com/ply/ply.html.

Delwadia, Vipul (2007) "SGJ: Sega Genesis Java", Victoria University of Wellington, unpublished Honours thesis, available online, URL: http://eprints.rclis.org/16127/.

Ramshaw, C. (1982) VIC Innovative Computing, Melbourne House, Leighton Buzzard, UK.

Rothenberg, J. (1999) "Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation", CLIR reports, Council on Library and Information Resources.

Swalwell, Melanie (2009) "Towards the Preservation of Local Computer Game Software: Challenges, strategies, reflections", Convergence: The International Journal of Research into New Media Technologies, (special issue on cultural memory and digital preservation, ed. Will Straw and Jessica Santone), vol. 15, no. 3, Aug, pp. 263-279.

Swalwell, Melanie, Denise de Vries (2013) "Collecting Code: Challenges and strategies", SCAN: Journal of Media Arts Culture (the In/Visibilities of Code special issue), vol. 10, no. 2, available online, URL: http://scan.net.au/scn/journal/vol10number2/Melanie-Swalwell.html.