1 Google Application Testing

Using the code coverage utilities that we have adapted for Wine, we plan to identify portions of the Windows API that the Google applications are using but were not being covered by the Wine conformance test suite. Once we have identified untested modules, we can then fill these gaps in testing, and potentially reveal bugs or underdeveloped areas of code. Thus, we would improve the likelihood of Google applications eventually running seamlessly in Wine.

1.1 Application Testing Process

The following is the process we expect to follow when we begin Google application testing in earnest:

1. Compile patched Wine with GCOV code coverage profiling enabled.

2. Compile patched LCOV with differential code coverage functionality added.

3. Execute Wine conformance test using winetest.

4. Execute Google application via Wine.

5. Utilize LCOV to examine code coverage statistics generated by both executions. LCOV will then generate helpful HTML output regarding the differences in code coverage between the two executions.

6. Using LCOV results, identify functionality of the Windows API used by the Google application that remains untested by the Wine conformance test suite.

7. Write test files in C or Perl to test newly identified untested Windows API functionality.

8. If new tests reveal bugs, submit bug reports to the Wine project.

9. Restart process.
1.2 Target Google Applications

Google has revealed four potential target Windows applications for testing in Wine:

- **Picasa** - a program that indexes, manages and searches through images on a personal computer
- **Hello!** - a program that facilitates image uploading and posting on blogs
- **Google Desktop Search** - a program that can searches through common document file formats on a personal computer
- **Keyhole** - a program which provides geographic satellite maps and geolocation search

Thus far, Picasa has been the main focus of our testing efforts, though we have done cursory probes of the other three applications. Given our initial results with Picasa and the existence of an upcoming beta of the next version, it seems a reasonable candidate for further exploration, and as such, has been identified as our single focus for the near future. In the event that progress slows with Picasa, we can shift our focus to one of the other three applications.

1.3 Initial Results

We have done a cursory examination of the applications running in Wine, mainly with Picasa, Keyhole and Google Desktop Search. The applications we are using for testing are the public versions available for download on the Google website. Currently, we are working with Picasa 1.6, the beta of Google Desktop Search, Keyhole 2 LT and the current version of Hello. Of these applications, Hello is the only application that we have not attempted to execute in any fashion. Any and all anomalies we come across when trying to execute an application in Wine are going to be documented along with any workarounds we discover in the process.

1.3.1 Installation Difficulties

The installers for Picasa, Keyhole and Google Desktop Search each have different failures when being run in Wine. The installer for Picasa repeatedly
produces a dialog box with the text "Out of Memory" and an "Ok" button that dismisses the box, only to have it reproduced until the process is ended unnaturally. The installer for Keyhole generates a memory access violation and asks if the user would like to debug. The installer for Google Desktop Search produces a dialog box claiming that the user is attempting to install onto an incompatible version of Windows.

In the case of Picasa and Keyhole, it was enough to make a copy of the program’s installations and execute the executable with Wine. In the case of Google Desktop Search, a simple copy of directory and executable was not enough. We suspect that Wine might default to

1.3.2 Picasa Slideshow

While running Picasa, attempting to start a slideshow causes the application to hang. LCOV was used to profile Wine while running Picasa and trying to start a slideshow. When the differential code coverage analysis was run on the execution’s profiling output and the profiling output from winetest, it was revealed that 601 lines of code were being executed by Picasa and not the test suite. A majority of these lines appear to be in the commctrl.c, datetime.c and monthcal.c files, all part of the commctrl32.dll module. To follow through with our process, we plan to write or modify unit tests to expand testing coverage of the functionality in this module.

1.4 Future Work

Future work for the coming semester includes:

- Finalizing the LCOV patch that is to be submitted to the LCOV project.
- Further profiling of more components of Google applications (particularly Picasa).
- Writing unit tests for exposed gaps in test suite that are to be submitted to the Wine project.
- Submitting Wine bug reports if new tests expose errors.