Dali Adaptor Automated Testing

July 11, 2014

Automated tests for the dali-adaptor project. These tests are callable from the TCT testing framework.

There are currently three executables built, each of which has its own source directory and contains several test cases. These are:

- `src/dali-adaptor` Uses a mocked platform abstraction, so doesn't test the abstraction implementations themselves. This currently tests the timer and key presses only.
- `src/dali-adaptor-internal` Tests some internal components of the abstraction implementations without attempting to have a live or mocked platform abstraction in place. This is the correct place for test cases which can test code in dali-adaptor in isolation. This suite is only built in debug mode and for development environments, not embedded device targets. Currently tied to the SLP platform abstraction directly. Noteworthy test cases:
  - `utc-Dali-InternalTests.cpp`: Runs suites of small tests of code that is not visible outside of its compilation units and so not possible to be tested using the usual mechanisms. See **Internal Tests** section below.
  - `utc-Dali-GifLoader.cpp`: Runs the gif loader syncronously and checks that the laoded bitmap matches a reference buffer on a pixel by pixel basis.
  - `utc-Dali-CommandLineOptions.cpp`: Tests whether commandline options of Dali are stripped out of various inputs.
- `src/dali-platform-abstraction` Uses a real live platform abstraction so can be used to test features exposed through the `Dali::Integration::PlatformAbstraction` abstract interface. Intended to test complex interactions operate correctly. For example:
  - Cancelation of a subset of a large number of in-flight requests.

– Correct reporting of successful and failed requests when large numbers of valid and invalid requests are issued in rapid sequence.

If a loader for an image type is added to the platform abstractions, a test case should be added to `src/dali-adaptor-internal`, styled after `utc-Dali-GifLoader.cpp`. An image of that type should also be added to the existing image test cases in `src/dali-platform-abstraction`.

If a new commandline option is added to Dali, `utc-Dali-CommandLineOptions.cpp` should be expanded with it.

Internal Tests

The platform abstraction supports a mechanism that is intended to allow the testing of code which is not visible outside its compilation unit such as small utility functions in unnamed namespaces.

To make use of it, simply include `portable/internal-test.h` and place a block of code like the one below outside function or class scope in any namespace of any `.cpp` file within the library. This example tests a hypothetical user-written `int mul(int, int)` function.

```
DALI_BEGIN_INTERNAL_TEST_NO_SUITE( MyTestName )

const int mulled = mul( mul ( 2, 3 ), mul( 5, 7 ) );
const int simpleMulled = 2 * 3 * 5 * 7;
DALI_INTERNAL_TEST_CHECK( mulled == simpleMulled );

DALI_END_INTERNAL_TEST( MyTestName )
```

Use the `DALI_BEGIN_INTERNAL_TEST` and `DALI_END_INTERNAL_TEST` macros at top and bottom of a block of test code. Most simple, isolated tests should use the variant `DALI_BEGIN_INTERNAL_TEST_NO_SUITE`. This ensures the test will be executed without writing any further boilerplate. If you use an existing suite name that is already executed from the automated tests, your new test case will also be run automatically with no further effort.

If you use a new test suite name with `DALI_BEGIN_INTERNAL_TEST`, you must also add a test case or cases to the testing framework to execute the suite. See existing examples here:

```
automated-tests/src/dali-adaptor-internal/utc-Dali-InternalTests.cpp
```

This feature can also be useful as a low-friction way to embed a small unit test immediately next to the code that it exercises, so they can be maintained side by side, even if the target code is actually callable from outside.

Building

Use the script `build.sh` in the root of `automated-tests/`. To build all three test executables, run the script without arguments:

```
./build.sh
```

To build just one executable, pass its name to the build script:

```
./build.sh dali-adaptor
```

```
./build.sh dali-adaptor-internal
```

```
./build.sh dali-platform-abstraction
```

Running

Use the `execute.sh` script. To execute the tests in all three executables, run the script with no arguments:

```
./execute.sh
```

To run just one executable, pass its name to the script:

```
./execute.sh dali-adaptor
```

```
./execute.sh dali-adaptor-internal
```

```
./execute.sh dali-platform-abstraction
```

To view the results of a test run, execute the following line:

```
firefox --new-window summary.xml
```