Overview of visual UI testing

Visual testing is a form of regression test that ensures that screens that were correct have not changed unexpectedly.

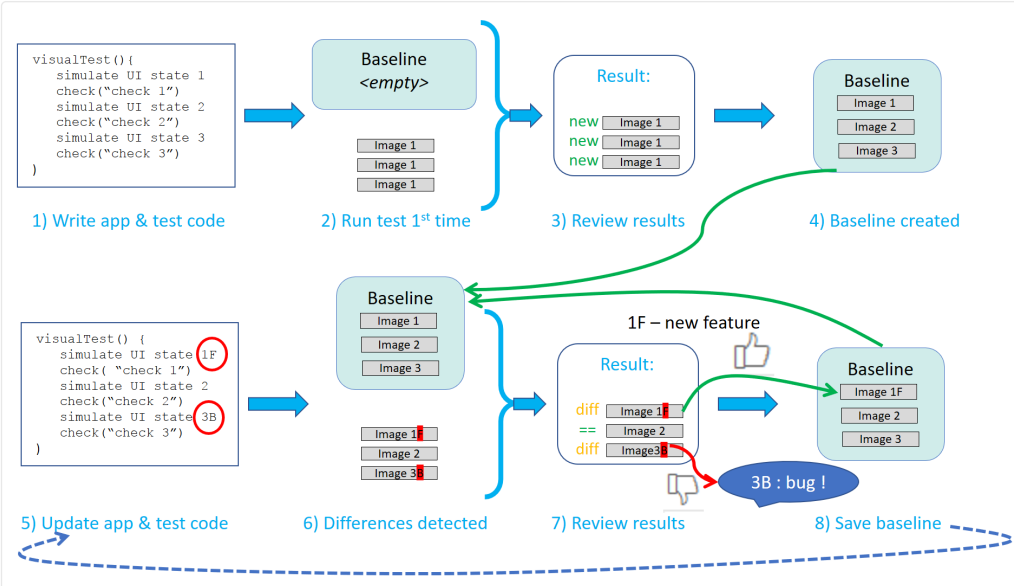
Visual testing of an application is done by running an application, and saving screen snapshots at key points called checkpoints. These screenshots are then compared to previously stored baseline images, and any visually significant differences are reported.

Implementing visual UI testing typically involves four basic steps:

1. Write a test that exercises your Application UI by sending simulated mouse and keyboard events in order enter various states, and capturing a screenshot in each of these states.
   a. Sending simulated mouse and keyboard events in order to enter various states
   b. Capturing a screenshot in each of these states.
2. Compare the captured screenshots to previously captured baseline images.
3. Review the resulting differences and :
   a. Identify cases where differences were caused by a new feature that does not appear in the baseline image and accept the new screenshot so that from this point forward it is used as the new baseline image for that checkpoint.
   b. Identify cases where differences indicate a bug that needs to be fixed, report the issue and reject the image - meaning that the baseline image is not updated and remains as is.
4. Save the baseline updates, so that they are used for the next test run.

The very first time a test is run, there are no baseline images, so the screenshots that are captured are adop as the baseline images. On subsequent runs, the flow is as described above.

This process is illustrated in the diagram below and the description that follows it.



The figure shows two phases - the upper row represents the first run and the second row represents subsequent runs which may be done multiple times (hence the arrow from step 8 to step 5).The numbering the sections below corresponds to the numbering in the figure above.

First run

2. Then you run the test for the 1st time.
3. Since there is no existing baseline, all the checkpoints are reported as "new" steps.
4. All the images captured during this run are automatically stored, and are then used as the baseline images in future tests that are run on this baseline.

Subsequent runs

5. You make some changes to the application and / or test code.
6. You run the test again and the differences are detected. In this example the differences are mark as 1F and 3B in the figure.

7a. In the first case (1F), you review the results and decide that the difference is due to the addition c a new feature. You accept the image, and once the baseline is saved, the newly captured image w be used as the new baseline image.

7b. In the second case you decide that the difference results from a bug. You report the bug and reje the image. As such the current baseline image should be retained.

8. Finally you save the baseline and then it will be used as the basis for comparison in the next test run, which now contains the new baseline image, 1F.