Автоматическое тестирование графического интерфейса интерактивных программных комплексов

Денисов Е.Ю., Волобой А.Г., Калугина И.А., ИПМ им. М.В. Келдыша РАН
eed@spp.keldysh.ru, voloboy@gin.keldysh.ru, qik@gin.keldysh.ru

Аннотация

В работе продемонстрирован выбор системы автоматизации действий с пользовательским интерфейсом программ в среде Windows для целей автоматического тестирования пользовательского интерфейса современных программных комплексов компьютерной графики и оптического моделирования. Специфика таких комплексов предъявляет особые требования к их тестированию.

1 Введение

Процесс создания современных программных систем требует автоматизации их разработки и сопровождения на всех этапах жизненного цикла программного продукта. Одним из важнейших этапов является тестирование. Сложность современных программных комплексов оптического моделирования, созданных нашим коллективом, достигла уровня, при котором классическое тестирование [Майерс, Баджетт, Сандлер, 2012] становится неэффективным, а часто и невозможным по ряду причин. Можно выделить следующие особенности тестирования таких комплексов:

Разнородные комплексы используют общие элементы пользовательского интерфейса;

Частые выпуски новых версий;

Ограниченные ресурсы и время.

В таких условиях абсолютно необходимым является комплексный подход к автоматизации тестирования. Предложенные методы позволяют добиться приемлемого качества тестирования, а значит и приемлемого качества выпускаемых программных комплексов. В основном используется два типа тестирования: тестирование на основе аналитических расчётов, обеспечивающее сравнение результатов работы системы с ожидаемыми, и так называемое регрессионное тестирование, обеспечивающее совпадение поведения новой реализации системы и её предыдущей реализации.

Требования к системе автоматического тестирования

Основываясь на накопленном опыте тестирования пользовательского интерфейса наших программных комплексов оптического моделирования, нами к системе автоматического тестирования были выдвинуты следующие требования:

- Способность распознавать элементы пользовательского интерфейса (окна, диалоги, кнопки, поля ввода, чекбоксы, слайдеры и т.д.) и производить над ними действия (ввод символов с клавиатуры, одиночное и двойное нажатие левой, правой и средней кнопками мыши, операции с колёсиком мыши, передвижение окон и диалогов, ввод и считывание значений);
- Поддержка продвинутого языка сценариев, позволяющего производить математические операции над числовыми величинами и операции над текстовыми строками;
- Возможность внесения изменений в имеющийся сценарий без полной его перезаписи;
- Поддержка работы с текстовыми файлами (чтение, запись);
- Отсутствие необходимости изменения кодов тестируемой программы (например, включение вспомогательных кодов для взаимодействия с внешним тестирующим пакетом);
- Способность оперировать относительными координатами (привязанными к окну тестируемой программы) вместо абсолютных (привязанных к экрану);
- Способность оперировать с выполняемыми процессами (по крайней мере, получение информации о них);
- Наличие операций с таймерами (вычисление разницы во времени между событиями, внесение определённой задержки между действиями, и т.д.).

2 Анализ существующих решений

Авторами были протестированы более семидесяти программных пакетов, позволяющих в том или ином виде записать и воспроизвести действия пользователя в интерактив-

ной программе. Среди них: [Test Automation FX], [TestComplete], [Squish], [Sikuli], [Atoma] и другие.

К сожалению, обнаруженные в большинстве пакетов недостатки не позволили полноценно использовать их для тестирования создаваемых программных комплексов. Среди этих недостатков:

- Запись действий пользователя в абсолютных координатах экрана (не позволяет воспроизводить записанную сессию на другом компьютере с другим разрешением экрана);
- Неспособность распознать UI элементы пакета Qt – основного средства разработки пользовательского интерфейса наших программных продуктов;
- Отсутствие возможности внесения изменений в записанную сессию;
- Отсутствие поддержки новейших версий Windows;
- Ошибки в работе, иногда приводящие к сбою всей OS.

Рассмотрим в качестве примера недостатки двух из вышеупомянутых продуктов.

2.1 Sikuli

Пакет автоматизации Sikuli является чисто графическим средством. Он оперирует с объектами пользовательского интерфейса как с изображениями. Для того, чтобы имитировать нажатие кнопки, необходимо создать (захватить) изображение этой кнопки и вставить его в скрипт. При этом качество распознавания изображения недостаточное. В случае наличия на экране двух объектов интерфейса с похожими надписями, например, кнопок с цифрами, Sikuli может нажать «не ту» кнопку. В случае, если на экране видны две одинаковые кнопки (например, кнопки «Close» в двух разных одновременно открытых диалоговых окнах), то не определено, какая именно из кнопок будет нажата, и нет никакой возможности уточнения. Кроме того, отсутствует обратная связь: нет возможности определить реакцию на воздействие на объекты интерфейса: определить, открылся ли необходимый диалог, и какие в нём есть объекты интерфейса, не представляется возможным.

2.2 Atoma

Пакет автоматизации Atoma, к сожалению, не распознаёт объекты пользовательского интерфейса, созданные с применением пакета Qt – основного пакета, используемого в наших программных комплексах для создания пользовательского интерфейса. Это делает невозможным взаимодействие тестирующей системы с объектами пользовательского интерфейса тестируемой программы.

Тем не менее, удалось подобрать один пакет, удовлетворяющий большинству условий. Это AutoIt – бесплатный пакет для автоматизации действий с пользовательским интерфейсом в Windows.

2.3 Autoit

AutoIt – это пакет автоматизации действий в пользовательском интерфейсе Windows, и не только. В его основе лежит язык сценариев, напоминающий BASIC. AutoIt использует комбинацию симуляции нажатий клавиш, движения мыши, операций с окнами и диалогами для автоматизации задач, недостижимой другими средствами (например, VBScript, SendKeys и т.д.). AutoIt – это небольшой по размеру самодостаточный пакет, работающий на всех версиях Windows и не требующий установки никаких дополнительных средств.

Изначально AutoIt был создан для автоматического конфигурирования тысяч компьютеров. Со временем он превратился в мощный язык сценариев, поддерживающий комплексные выражения, пользовательские функции, циклы и всё прочее, что могут ожидать специалисты от языка сценариев. Перечислим его основные особенности:

- Простой для изучения синтаксис, похожий на язык программирования BASIC;
- Имитация нажатий клавиш и движений мыши;
- Манипуляция окнами и процессами;
- Взаимодействие со всеми стандартными элементами пользовательского интерфейса Windows;
- Возможность создания собственных элементов пользовательского интерфейса;
- Поддержка технологии COM;
- Поддержка регулярных выражений;
- Прямой вызов функций из внешних и системных DLL;
- Совместимость с Windows XP / 2003 / Vista / 2008 / Windows 7 / 2008 R2 / Windows 8 / 2012 R2 / Windows 10;
- Поддержка Unicode строк;
- Поддержка 64-битной архитектуры;

- Сценарии могут быть скомпилированы в выполняемые модули;
- Подробное руководство пользователя.

AutoIt занимает мало места и не требует внешних DLL файлов и записей в системном реестре, что делает его безопасным для выполнения на серверных версиях Windows. В комплекте поставляется редактор скриптов с поддержкой цветового выделения конструкций языка.

Много усилий было направлено на оптимизацию функций симуляции нажатий клавиш и движения мыши с целью сделать их максимально точными и надёжными во всех поддерживаемых версиях Windows. Все функции, относящиеся к контролю мыши и клавиатуры, имеют широкие возможности для настройки скорости и функциональности их действий. Функции взаимодействия с окнами и диалогами позволяют, в том числе, передвигать, прятать, показывать, изменять размер, активировать, закрывать окна и диалоги. Нужные окна могут быть определены (найдены) по их заголовку, содержимому, размеру, положению, классу и даже внутренним дескрипторам Win32 API.

Кроме операций с имеющимися окнами, AutoIt имеет средства для создания собственных окон и диалогов – поддерживаются все стандартные элементы пользовательского интерфейса Windows.

3 Применённые решения

Так как одни и те же элементы пользовательского интерфейса используются в различных программных комплексах, оказалось целесообразным создать библиотеку функций на языке сценариев AutoIt для упрощённого выполнения некоторых часто используемых команд, таких как «открыть файл», «сохранить результат расчёта», и т.д. Вот так, например, в этом языке сценариев выглядит функция для загрузки так называемой сцены – модели данных, состоящей из многих объектов различных типов, связанных определёнными иерархическими связями (геометрические объекты, свойства их поверхностей, источники света, и т.д.):

```
  ; Function to open given scene
  Func SceneOpen($scene_name)
    ; Bring window of our program
to top
    WinActivate("Layouter")

    ; Send Ctrl+o
    Send("^o")
    ; Wait for "Open Scene" dia-
log to appear
    If  WinWaitActive("Open  Sce-
ne", "", $Timeout) = 0 Then
      Report("Dialog 'Open Scene'
not found")
      Exit(1)
    EndIf

    ; Type scene filename
    Send($scene_name)
    ; Click "Open" button
    ControlClick("[ACTIVE]",  "",
"Open")
    ; Make sure scene was loaded,
    ; its name must be presented
in window title
    CheckTitle($scene_name)
  EndFunc
```

Кроме часто используемых простых команд, в библиотеку включены также сложные последовательности действий, выполняющие определённую часто используемую операцию, например, присваивание выбранной поверхности геометрического объекта сцены указанное свойство (цвет, прозрачность, текстуру и т.д.). Рассмотрим соответствующий пример – присваивание указанному объекту сцены заданной текстуры с заданным разрешением по X и Y:

```
  ; Function to assign given tex-
ture with given size to given ob-
ject
  Func SetTexture($node,
$tex_name, $size)
    ; Double click to open prop-
erties window
    ControlClick("Scene Hierar-
chy", "", $node, "left", 2)
    If WinWaitActive("Surface
Properties", "", $Timeout) = 0
Then
      Report("Dialog 'Surface
Properties' not found")
      Exit(1)
    EndIf

    ; Click on "Texture" tab
```

```
    ControlClick("Surface Proper-
ties", "", "tab control", "left",
1, 230, 12)
    Sleep(1000)

    ; Click "Initial path"
    ControlClick("Surface Proper-
ties", "", "start_cmb")
    Send("{DOWN 5}{UP}{ENTER}")

    ; Load texture file
    ControlClick("Surface Proper-
ties", "", "Load texture")
    If WinWaitActive("Load tex-
ture file", "", $Timeout) = 0
Then
      Report("Dialog 'Load tex-
ture file' not found")
      Exit(1)
    EndIf
    Send($tex_name & "{ENTER}")

    ; Click "Image size" control
    ControlClick("Surface Proper-
ties", "", "image size x",
"left", 2)
    Send("100{TAB}");
    ; Pause for recalculation
    Sleep(2000)
Send("100{TAB}");
    ; Pause for recalculation
    Sleep(2000)

    ; Close dialogs
    ControlClick("Surface Proper-
ties", "", "OK")
    Sleep(1000)
  EndFunc
```

Тестирование с использованием пакета AutoIt происходит следующим образом: для каждой функциональности тестируемой программы создан сценарий, который выполняется, воздействуя на элементы пользовательского интерфейса этой программы. Результат выполнения сценария сравнивается с ожидаемым, и при их совпадении тест считается успешным. Таким результатом, как правило, является:

- Определённое состояние элементов интерфейса (необходимые диалоговые окна открыты, кнопки нажаты, текстовые поля имеют определённые значения) в ответ на интерактивные действия пользователя;
- Правильное изображение в окне программы после определённого набора действий (например, изменение мощности источника света должно приводить к изменению освещённости объектов сцены);
- Правильные численные значения проверяемых параметров (например, установка конкретного значения отражающей способности некоторого объекта должна приводить к известному, заранее посчитанному аналитически, количеству света, отражаемого этим объектом);
- Правильная, т.е. заранее ожидаемая реакция программы на определённые действия пользователя, как правильные, так и ошибочные (например, сообщение об ошибке в случае задания недопустимых параметров);
- Правильный результат расчёта освещённости как для всей сцены, так и для определённых её участков (как правило, чувствительных к программным ошибкам) – определяется сравнением финального изображения с ожидаемым; при разнице изображений свыше допустимой тест считается неуспешным;
- Отсутствие ошибочной реакции программы (например, такой, как аварийное завершение работы) в результате определённых действий.

Ниже приведён пример теста, сравнивающего изображение, выдаваемое тестируемой версией программы, с эталонным изображением, полученным ранее. Тестируется подсистема расчёта собственной яркости объектов сцены. Невооружённым глазом разница незаметна: нельзя сказать, являются ли изображения на рис. 1 и рис. 2 одинаковыми, или нет. Для анализа используется программа, вычисляющая разницу между двумя изображениями, и выдающая эту разницу в абсолютном виде и в виде изображения, удобном для восприятия.

Рис. 1. Эталонное изображение

Рис. 2. Изображение, полученное от тестируемой программы

Рис. 3. Разность изображений

Более яркие участки в этом разностном изображении соответствуют большей разнице в абсолютных значениях соответствующих пикселей исходных изображений. Абсолютное значение максимальной разницы между эталонным и реальным изображениями в данном случае составило 1.16%.

К сожалению, пакет AutoIt имеет и некоторые недостатки, специфичные для наших программных комплексов. В нём не полно-

стью реализована поддержка пакета GUI SDK «Qt», что приводит к некоторым неудобствам в работе. Например, отсутствует поддержка элементов меню (для вызова пунктов меню приходится использовать клавиатурные команды: Ctrl+o для открытия файла, и т.д.), нет возможности захвата (для дальнейшей обработки или записи в файл) изображения с области экрана, нет возможности узнать состояние некоторых элементов интерфейса (например, текущий элемент, выбранный в выпадающем списке).

В то же время AutoIt обладает рядом достоинств, которые позволяют существенно расширить область его применения. Его язык сценариев включает в себя наборы функций для:

- Получения и изменения значений переменных среды окружения;
- Чтения и записи текстовых и бинарных файлов;
- Воспроизведения мультимедиа файлов;
- Передачи данных по локальной сети;
- Поддержки COM;
- Управления процессами.

Кроме того, имеется мощнейший инструментарий для создания собственных элементов интерфейса, например, с целью интерактивного отображения процесса тестирования и управления им.

Кроме собственно тестирования, пакет AutoIt также может использоваться для демонстрации возможностей приложений, и в качестве интерактивного дополнения к инструкции по эксплуатации приложения.

4 Заключение

Пакет AutoIt успешно используется для тестирования и создания презентаций возможностей программ, входящих в состав программного комплекса компьютерной графики и оптического моделирования Lumicept [Жданов, Потемин, Галактионов, Барладян, Востряков, Шапиро, 2011]. Он позволяет значительно сократить время и снизить трудозатраты на выпуск, тестирование и сопровождение новых версий. Так, например, проверка сорока тестов проходит в полностью автоматическом режиме меньше чем за час, в то время как ручная проверка этих же тестов потребовала бы по меньшей мере день работы, учитывая тот факт, что прежде чем приступить к конкретному тесту, человек должен вспомнить (или прочитать), что именно и как необходимо тестировать в данном тесте.

Благодарности

Работа выполнена при частичной поддержке РФФИ, гранты № 16-01-00552, 18-01-00569.

Список литературы

Г. Майерс, Т. Баджетт, К. Сандлер. Искусство тестирования программ, 3-е издание - М.: "Диалектика", 2012.

Test Automation FX
URL: http://www.testautomationfx.com (дата обращения: 19.02.2018)

TestComplete
URL: http://smartbear.com/products/qa-tools/automated-testing-tools (дата обращения: 19.02.2018)

Squish
URL: http://www.froglogic.com/squish/ (дата обращения: 19.02.2018)

Sikuli
URL: http://www.sikuli.org/ (дата обращения: 19.02.2018)

Atoma
URL: http://www.getautoma.com/ (дата обращения: 19.02.2018)

AutoIt
URL: http://www.autoitscript.com/ (дата обращения: 19.02.2018)

Жданов Д.Д., Потемин И.С., Галактионов В.А., Барладян Б.Х., Востряков К.А., Шапиро Л.З. // Спектральная трассировка лучей в задачах построения фотореалистичных изображений "Программирование", 2011, № 5, с. 13-26.