2. http://hilscher.com
3. Alessandro Rubini, Jonathan Corbet. Linux Device Drivers. Second Edition. O'Reilly, 2001, 564p.
4. Олифер В.Г. , Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. Издание 4-ое. СП-б: Питер, 2010, 918с.

СОЗДАНИЕ ИНСТРУМЕНТОВ РАСШИРЕННОГО ПРЕДСТАВЛЕНИЯ ДЕЙСТВИЙ ОПЕРАТОРА В ПОЛЬЗОВАТЕЛЬСКОМ ИНТЕРФЕЙСЕ WINDOWS

Козак Николай Владимирович, Плаксин Александр Михайлович
РФ, г. Москва, МГТУ «Станкин»
alexander-hoy@yandex.ru

Важную роль в процессе разработки, как программного продукта, так и программного комплекса играет процесс тестирования приложений. Тестирование выявляет логические ошибки, недостатки программы, проверяет и подтверждает стабильность работы.

Этап «тестирование» можно разделить на две части:

- тестирование в процессе разработки (применяется для выявления ошибок);
- визуальное тестирование (применяется для утверждения факта работоспособности программы, перед сдачей ее заказчику).

Ручное тестирование пользовательского интерфейса проводится тестировщиком-оператором, который руководствуется в своей работе описанием тестовых примеров в виде набора сценариев. Каждый сценарий включает в себя перечисление последовательности действий, которые должен выполнить оператор, и описание важных для анализа результатов тестирования ответных реакций системы, отражаемых в пользовательском интерфейсе. Типичная форма записи сценария для проведения ручного тестирования - таблица, в которой в одной колонке описаны действия (шаги сценария), в другой - ожидаемая реакция системы, а третья предназначена для записи того, совпала ли ожидаемая реакция системы с реальной и перечисления несовпадений.

Ручное тестирование пользовательского интерфейса удобно тем, что контроль корректности интерфейса проводится человеком, т.е. основным "потребителем" данной части программной системы. К тому же при чисто косметических изменениях в интерфейсах системы, не отраженных в требованиях (например, при перемещении кнопок управления на 10 пикселей влево), анализ успешности прохождения теста будет выполняться не по формальным признакам, а согласно человеческому восприятию.

При этом ручное тестирование имеет и существенный недостаток - для его проведения требуются значительные человеческие и временные ресурсы. Особенно сильно этот недостаток проявляется при проведении регрессионного тестирования и вообще любого повторного тестирования - на каждой итерации повторного тестирования пользовательского интерфейса требуется участие тестировщика-оператора. В связи с этим в последнее десятилетие получили распространение средства автоматизации тестирования пользовательского интерфейса, снижающие нагрузку на тестировщика-оператора.

Хотя на первый взгляд ручное тестирование – довольно простая операция, при неправильном применении она может привести к проблемам. Группы проектирования быстро выясняют, насколько сложно работать с многочисленными электронными таблицами и другими средствами тестирования и хранения результатов. Они отмечают большой объем повторной работы, связанный со многими используемыми сценариями тестирования, что обусловлено недостатком повторного использования и модульностью. И постоянно приходится следить за тем, чтобы свести к минимуму ошибки персонала и противоречивость результатов. Эти факторы сказываются на конечной цели ручного тестирования - на качестве оценки приложений и выявлении ошибок.

Существует множество средств для упрощения, автоматизации операций и управления процессом ручного тестирования, например можно использовать IBM® Rational® Manual Tester которое облегчает выполнение следующих задач:

- Ясного и краткого документирования процесса выполнения ручного тестирования;
- Возможности общего использования содержимого тестирования в нескольких тестах;

- Помощи проводящим тестирование в выполнении обычных операций, подверженных ошибкам - ввод и проверка данных;
- Точного выполнения и записи результатов ручного тестирования.

В работе «Визуальное тестирование» ставится следующая задача – разработать программу, которая наглядно предоставляла бы такие возможности, как слежение за процессом ввода разработчика, позиционированием курсора, а также индикация нажатия системных клавиш. То есть программа должна уметь перехватывать сообщения от клавиатуры и мыши тестера, выводить их на дисплее, чтобы далее можно было произвести с помощью сторонних программных продуктов видеозахват или показать работу тестера в реальном времени большой аудитории через проектор и аналогичные средства вывода информации.

Рис. 1 – Схема порядка работы

В программе перехвата сообщений ввода используется тесная работа с WinApi32, с помощью методов которой и реализуется перехват, а также вывод сообщений о нажатых тестером клавишах.

Такой подход к тестированию приложений дает неоспоримый плюс, т.к. заказчик видит визуализацию работы своей программы, а так же при этом он видит все действия, которые были произведены.

Как можно увидеть на представленном снимке с экрана, программа выводит блок General возле указателя мыши, содержащий информацию о координатах положения курсора, а также о нажатии системных клавиш (Ctrl, Alt, Shift), блок Keyboard, находящийся над блоком General и содержащий информацию о нажатых клавишах, блок Mouse, находящийся под блоком General и содержащий информацию о нажатии (отжатии) кнопок мыши.

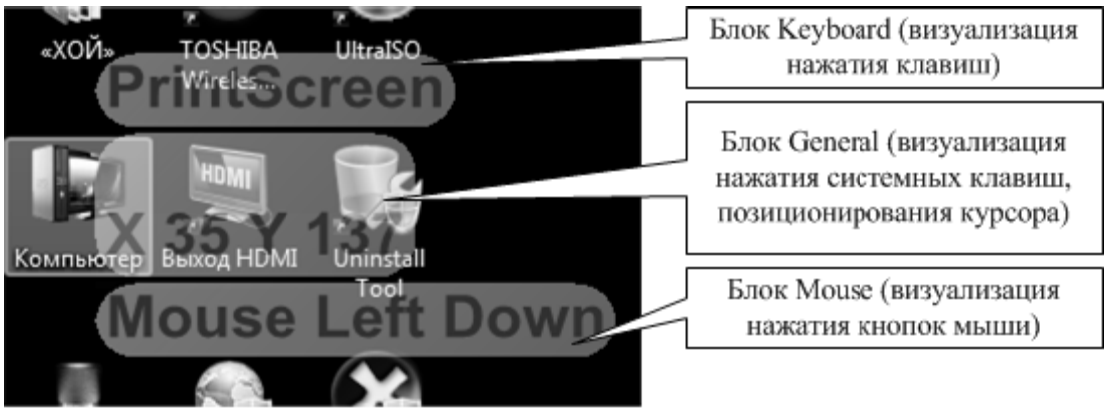

Рис. 2 – Скрин работы программы

Внутренние настройки программы позволяют включать/отключать блоки General, Mouse и Keyboard, изменять размер, стиль, цвет выводимого шрифта, прозрачность и цвет фона блоков, устанавливать время незатухания блоков (реализовано для блоков Keyboard и Mouse) по отдельности.

Также можно отметить, что этим настройки не ограничиваются. Пользователь сможет подобрать оптимальную настройку вывода, под конкретный случай презентации.

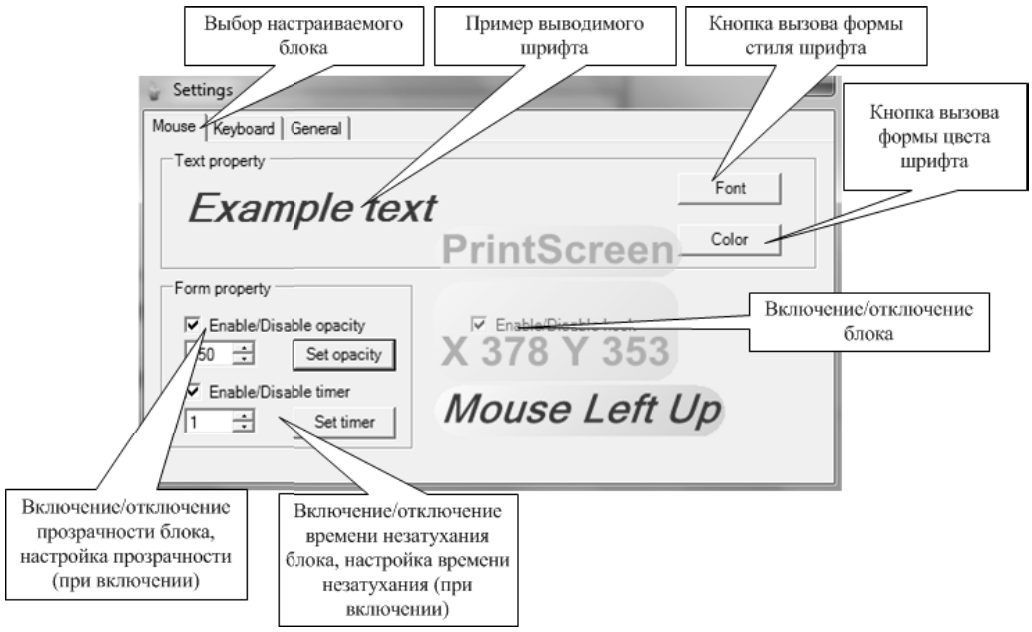

Рис. 3 – Диалоговое окно настроек Settings

Программа поддерживает работу в видеорежиме Dual Screen, как Clone, так и Wide режимы, поэтому при одновременном использовании монитора и проектора не возникнет проблем с выводом сообщений.

Программа работает в фоновом режиме. Все управление осуществляется с помощью иконки программы в системном трее.

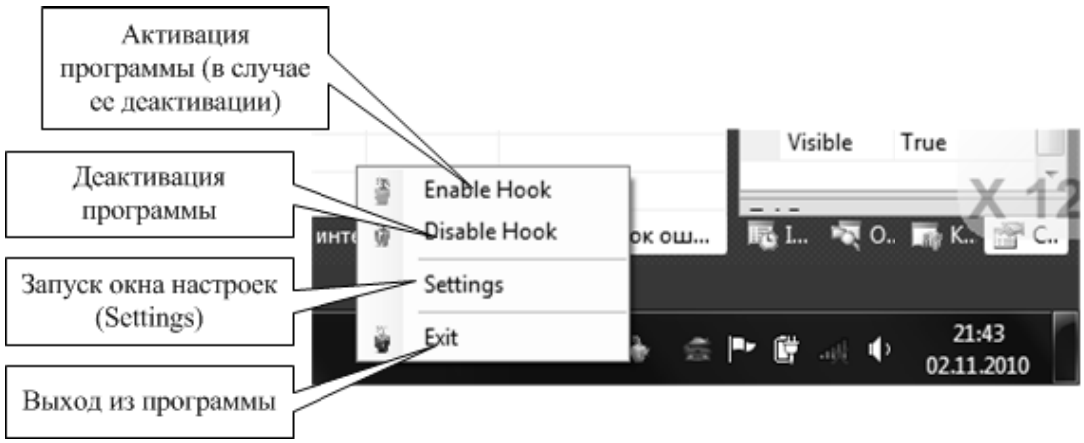

Рис. 4 – Активация иконки в системном трее

Благодаря этому окно программы не будет мешаться при проведении презентации\тестирования.

Стороннее использование программы может применяться в образовательных целях. Лектору не обязательно вслух озвучивать все действия, производимые с изучаемым программным продуктом, все действия лектора будут отображаться в реальном времени на дисплее или другом видеоустройстве вывода.

Интерес к такому подходу тестирования\образования заметно повысился в последнее время, так как:

- во-первых, производится много программных продуктов, заказчики которых требует предоставить работающую, отлаженную версию, что очень сложно сделать, если между исполнителем и заказчиком расстояние в несколько сотен километров;
- во-вторых, на рынке есть множество уже готовых программных продуктов, таких как, например, 3ds Max, AutoCad, Photoshop, при работе с которыми используется множество сочетаний клавиш.

Предложенный подход к тестированию и образовательной деятельности может оказаться наиболее удобным, так как при этом используется визуальное представление действий пользователя. Программа по перехвату сообщений ввода, безусловно, найдет свое место на рынке, так как она проста в управлении, настройке и использовании. Программа предоставляет такие возможности, которые окажутся полезными студентам, преподавателям и тестерам.

Библиографический список

1. Professional C# 2005 with .NET 3.0;
2. Christian Nagel, Bill Evjen, Jay Glynn, Morgan Skinner Karli Watson;
3. Willey Publishing, Inc;
4. Ресурс – http://msdn.microsoft.com/.

РЕАЛИЗАЦИЯ МОДЕЛИ ВЕНТИЛЬНОГО ДВИГАТЕЛЯ В СИСТЕМЕ MULTISIM

Кузовкин В. А. , Филатов В. В., Чумаева М. В.
РФ, Москва, ГОУ ВПО МГТУ «Станкин»
vfilatov45@mail.ru

К отличительным особенностям современного этапа развития электропривода следует отнести замену распространенных коллекторных двигателей постоянного тока (ДПТ) на бесконтактные двигатели с электронной системой коммутации и управления. На начальном этапе вентильным называли электропривод, в котором регулирование режима работы электродвигателя производилось с помощью управляемых электронных вентильных преобразователей электрической энергии: выпрямителя, импульсного регулятора, преобразователя частоты. Иными словами, к вентильным были отнесены электродвигатели, в которых ненадежный механический щеточно-коллекторный переключатель был заменен управляемым коммутатором на полупроводниковых элементах. В зависимости от конструктивных особенностей различают два основных вида вентильных двигателей: бесконтактные (бесколлекторные или бесщеточные) двигатели постоянного тока, называемые в англоязычной литературе "*brushless DC motors*" и бесконтактные двигатели