Оглавление

Введение

За последнее время технологии разработки компьютерной техники начали расти высокими темпами. В настоящее время практически каждый человек использует для своих нужд различное программное обеспечение. В связи с этим у разработчиков программного обеспечения возникает необходимость обучения возможностям и функциям программного продукта пользователей, далеких от области информационных технологий. Реализация такой обучающей инфраструктуры, чтобы пользователь мог взаимодействовать с продуктом без затруднений на уровне полной его функциональности, является сложной задачей для разработчиков.

TRIK Studio − это среда визуального программирования роботов [1], которая применяется, в частности, в учреждениях среднего образования для преподавания основ информатики и кибернетики. Так как основной целевой аудиторией TRIK Studio являются школьные учителя и ученики (зачастую младших классов), обучающая инфраструктура среды должна соответствовать требованиям низкого порога вхождения [4]. Поэтому справка как основной метод обучения возможностям программы не является эффективным решением проблемы.

Можно выделить такие способы устройства обучающей инфраструктуры [11, 13] без живого учителя: справка, видео-уроки и обучение, встроенное в среду. Наиболее интересным решением является обучающая инфраструктура, встроенная в саму среду.

Хорошим примером организации обучающей инфраструктуры стали компьютерные игры: из всех классов программных продуктов они имеют наиболее низкий порог вхождения. В компьютерных играх обучение

встроено в программный продукт, оно неотделимо от самого игрового процесса. Таким образом, за несколько минут усваиваются основы, и дальнейшее изучение проходит самостоятельно. Пользователи компьютерных игр не занимаются поиском нужной информации в справке, и обучение слабо обособленно от самой игры, поэтому проходит почти с таким же интересом, как и полноценная игра.

Такой эффект достигается демонстрацией в интерфейсе самой игры простых действий, которые пользователь впоследствии должен повторить. Очевидным способом реализации такого механизма является некоторый язык управления действиями пользователя, потому что это позволяет задавать любые сценарии поведения среды под внешним воздействием. Таким образом, мы можем программно описать те действия, которыми сможет, в последствии, применить пользователя, решая свою задачу в среде.

Примечательно, что такой подход к обучению пользователей годится не только для компьютерных игр, но и для практически любого класса коробочного программного обеспечения, в частности, для среды визуального программирования TRIK Studio.

Постановка задачи

Целью данной работы стало исследование концепций построения обучающей инфраструктуры в программных средах и применение полученных знаний в реализации средств для создания обучающих демонстраций непосредственно пользователем в среде визуального программирования роботов TRIK Studio — среды обучения основам программирования и кибернетики.

Реализуемая технология должна предоставить инструмент, который предоставит возможность более опытному пользователю без труда разработать комплекс обучающих программ для обучения своих коллег.

Глава 1. Обзор существующих решений

1.1. Актуальность задачи

На текущий момент среда TRIK Studio предоставляет лишь громоздкая справка, описывающая возможности среды. В рамках курсовой работы третьего курса [6] был реализован интерфейс программирования приложений (API) к пользовательскому интерфейсу среды, с помощью которого возможно задать последовательность пользовательских действий и воспроизвести ее. По причине постоянно растущей функциональности данной среды, а также регулярного притока новых пользователей, возросла необходимость автоматизации процесса генерации пользовательских сценариев.

1.2. Видео-уроки

Хорошим примером реализации обучающей инфраструктуры можно считать видео-уроки: они наглядно иллюстрируют решение некоторой задачи в программной среде, сопровождаемое некоторыми подсказками. Большинство видео-уроков содержат вербальные комментарии автора, однако возможно и встраивать подсказки и в само видео, с помощью распознавания элементов интерфейса и их подсветки и даже делать его пошаговым, разделяя урок на значимые части [8, 12, 14].

Однако, такой способ обучения возможностям среды имеет ряд недостатков. Во-первых, пользователю приходится потратить почти столько же времени на просмотр, сколько и потребовалось автору, потому что поставленную цель зачастую можно решить быстрее другими способами и не делая пауз для комментариев. Во-вторых, видео-уроки требуют выйти за пределы среды, то есть попытка повторить действия в среде вынуждает

отвлечься от видео или досмотреть урок до конца, только после чего уже применять полученные знание. И в-третьих данный способ обучения сильно привязан к конкретному графическому интерфейсу, то есть при сильном его изменении требуется перезапись видео-урока.

1.3. Макросы

Разработчики программного обеспечения зачастую предоставляют некоторый интерфейс программирования приложений для более продвинутых пользователей. Он позволяет определять сценарии поведения среды - макросы, которые позволяют сократить временные затраты на рутинные последовательности пользовательских действий, описав их на некотором языке программирования.

Эта технология используется во множестве программных продуктов, предназначенных для различных предметных областей, например, в текстовых или графических редакторах, таких как Notepad++ и CorelDraw. Также ярким представителем можно считать язык VBA (Visual Basic for Application), представленный Microsoft в линейке Microsoft Office.

Основным применением макросов считается запись рутинных действий для их последующего мгновенного повторения, однако функции записи и последующего воспроизведения пользовательских действий открывают возможности для тестирования графического интерфейса, а также для организации обучающей инфраструктуры с помощью создания демонстраций.

Одной из главных проблем макроязыков можно считать потребность в навыках программирования, для создания макрокоманд.

1.4. Программирование по демонстрации

Попыткой решения данной задачи можно считать “программирование по демонстрации” [7]. Эта технология позволяет записывать пользовательские действия без программирования их сценариев. Данный метод основан на машинном обучении и извлечении полезной информации о повторяющихся последовательностях пользовательских действий, в то время, как пользователь взаимодействует со средой. Наиболее часто повторяемые последовательности переводятся в макросы автоматически и предоставляются пользователю для использования. Этот метод позволяет решить проблему создания нужных макросов, однако абсолютно неприменим для организации обучающей инфраструктуры или тестирования пользовательского интерфейса.

1.5. Встраиваемая обучающая инфраструктура

На данный момент, также, существуют готовые обучающие инфраструктуры, которые интегрируются в среду [10]. Данная технология позволяет избавится от одного из основных недостатков видео-уроков: отделенности обучения от программной среды. Это позволяет пользователю сосредоточится на уроке и усваивать его материал, непосредственно взаимодействуя с программных продуктом, что делает урок интерактивным.

Однако и этот метод имеет ряд недостатков. Во-первых, потребность в интеграции в программную среду, что вынуждает разработчиков изучать предлагаемую обучающую инфраструктуру и ее программные интерфейсы. Во-вторых, технология предлагает язык описания пользовательских сценариев, который хоть и проще в освоении, чем макрокоманды, однако имеет ряд схожих недостатков: описание сценариев на текстовом языке

программирования, отсутствует процесс записи. Таким образом, данная технология применима лишь к программному обеспечению с простым и понятным пользовательским интерфейсом, так как с усложнением интерфейса растет количество необходимых для описания возможностей демонстраций.

QReal, несмотря на свою целевую аудиторию, является средой разработки, поэтому данная среда имеет довольно большое количество инструментов интерфейса. Данная особенность целевой среды потребует создания широкой обучающей инфраструктуры.

1.5. Обучающая инфраструктура в программной среде QReal

Обучение посредством обучающих демонстраций в программной среде можно реализовать с помощью макросов, расширенных функциями, несущими характер “подсказки” для пользователя. Данное решение наглядно представит пользователю действия и их результат, то есть состояние среды после этих действий.

DSM-платформа QReal предоставляет инструменты для создания визуальных языков, поэтому реализация макросов в качестве визуального языка позволит частично решить их проблему записи, так как пользователям придется исправлять не код на текстовом языке, а соответствующую диаграмму на языке описания пользовательских сценариев, описывающую некоторую обучающую демонстрацию. Также, вовлечение пользователей в их создание позволит обеспечить более широкое покрытие множества возможностей среды.

Полученная технология должна предоставить возможности для взаимодействия с произвольными элементами пользовательского интерфейса

(кнопки, выпадающие списки, полосы прокрутки и т.д.), а также с инструментами интерфейса QReal (сцена, палитра и т.д.). Важным фактором является то, что использование данной технологии подразумевается не только разработчиками, но и пользователями, поэтому она должна как можно меньше опираться на внутреннюю структуру среды.

Глава 2. Реализация

2.1 Подход к решению задачи

TRIK Studio реализована на основе DSM-платформы QReal [5], которая позволяет создавать предметно-ориентированные визуальные языки программирования. Было бы полезно создать обучающую инфраструктуру для всей DSM-платформы, тогда все системы созданные на ее базе (такие как TRIK Studio) унаследуют эту технологию [1].

Практически любую предметную область можно формализовать и описать соответствующим предметно-ориентированным языком программирования. DSM-платформа QReal предоставляет нужные инструменты для создания таких языков, поэтому было принято решение перенести реализацию обучающей структуры в нее и в данном случае предметной областью станет программирование последовательностей пользовательских действий.

Данная работа предполагает создание языка на базе платформы QReal, а также реализацию записи и трансляции действий пользователя в этот язык. Полученная технология должна позволить создание обучающих демонстраций с помощью программирования последовательностей пользовательских действий на реализованном визуальном языке.

2.2. API для скриптования действий пользователя

Первым шагом на пути создания языка описания пользовательских сценариев стала реализация API на скриптовом языке, а конкретно QtScript. Данная работа была сделана в рамках курсовой работы третьего курса.

Qt содержит интерпретатор этого языка сценариев, а он, в свою очередь, позволяет взаимодействовать с объектами, описанными с помощью средств этого инструментария. Сам интерфейс программирования приложений предоставляет функции имитирующие действия пользователя со средой, ее пользовательским интерфейсом.

Так как основными способами взаимодействия пользователя со средой являются механические манипуляторы, такие как компьютерная мышь и клавиатура, были реализованы виртуальный курсор и клавиатура.

Так как многие специфичные элементы управления пользовательского интерфейса, как стандартные (выпадающие списки, полосы прокрутки и т.д.), так и особенные для DSM-платформы QReal и ее плагинов (палитра, сцена, редактор свойств), требуют индивидуального подхода, API содержит части, отвечающие за них. Например, paletteAPI содержит функции взаимодействия виртуального курсора с палитрой. Каждая такая функция разбивается на последовательность более примитивных: передвижение курсора, нажатие клавиш мыши и клавиатуры.

Помимо этого реализованный API содержит фасад к пользовательской части системы, а также несколько вспомогательных функций, необходимых при скриптовании действий пользователя. Также присутствует возможность отправки всплывающих подсказок и стрелок, указывающих на нужные элементы пользовательского интерфейса, чтобы из воспроизведения цепочки пользовательских действий получить полноценную и автономную обучающую демонстрацию.

2.3. Описание языка

Средства QReal позволяют создавать визуальные языки, поэтому следующим этапом данной работы стала реализация визуального языка описания пользовательских сценариев с помощью этой DSM-платформы.

Диаграммы на этом языке (рис. 1) будут транслироваться в язык QtScript (рис. 2) и интерпретироваться средствами среды, поэтому функциональные возможности должны покрывать стандартные конструкции языка сценариев, а также должны присутствовать блоки, схожие с функциями API (рис. 3):

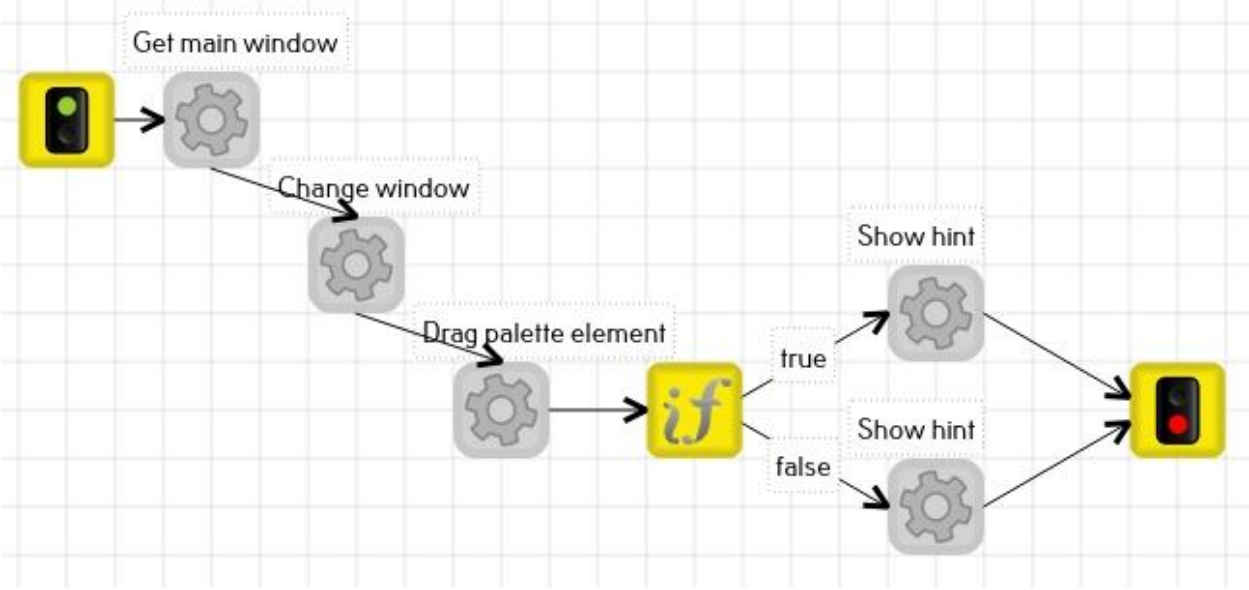

Рис. 1 сценарий создания элемента на сцене перетаскиванием его из палитры

```
var x;

var main = function()
{
        api.changeWindow(MainWindow);
        var x = api.palette().dragPaletteElement(
                        "qrm:/RobotsMetamodel/RobotsDiagram/TrikV6EnginesForward"
                        , 1000, 100, 100);
        if (x) {
                api.hints().addHint("Element now on scene.", 1000, MainWindow);
        } else {
                api.hints().addHint("Element not found!", 1000, MainWindow);
        }
        return;
}
```

Рис. 2 код на языке QtScript с использованием API, для диаграммы на рис. 1

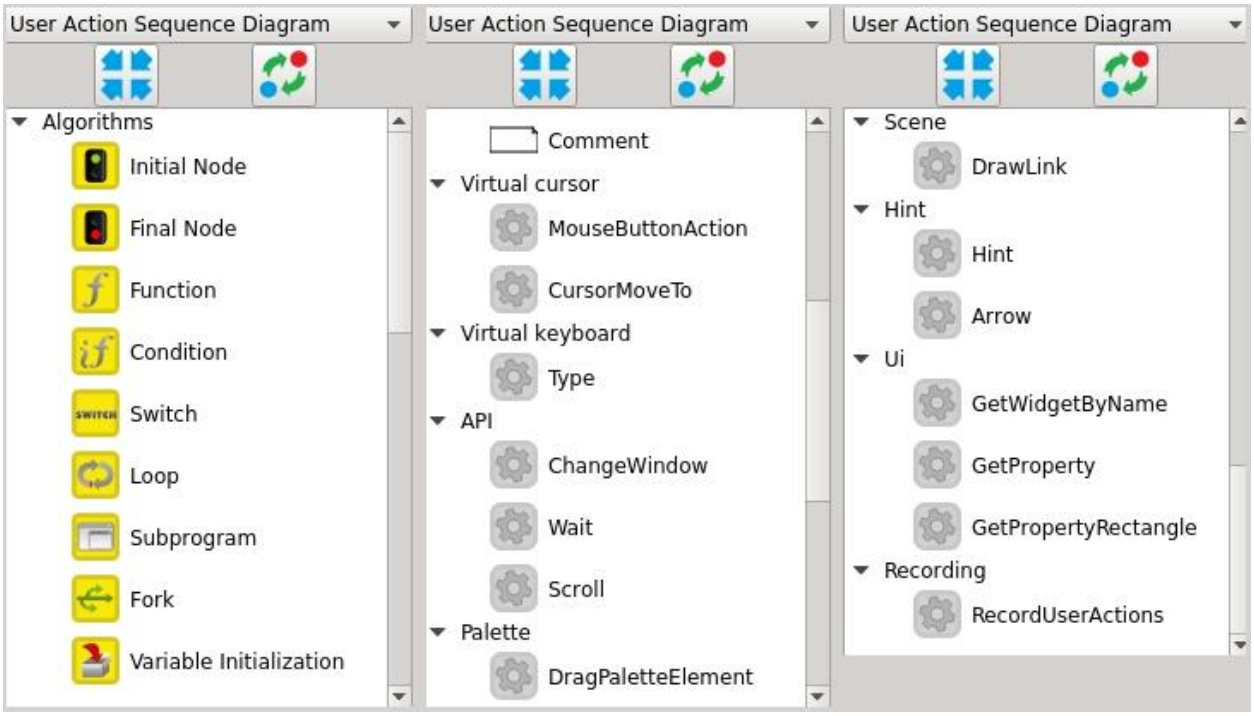

Рис. 3 блоки языка описания пользовательских сценариев

- Блоки, описывающие стандартные конструкции языка (блок цикла, блок условного оператора и т.д.);

- Блоки управления виртуальным курсором (эмуляция событий компьютерной мыши);
- Блок управления виртуальной клавиатурой (эмуляция событий клавиатуры);
- Блоки для работы с палитрой и сценой;
- Вспомогательные блоки (например функция wait, которая позволяет временно остановить поток управления, пока графический интерфейс претерпевает изменения в связи с предыдущими действиями);
- Блоки управления для сложных элементов пользовательского интерфейса (выпадающие списки, полосы прокрутки и т.д.).
- Блоки, позволяющие получать ссылки на необходимые элементы интерфейса по их типу, имени и родителю;
- Блок записи.

Данные блоки были описаны с помощью реализованного редактора языка. Большинство блоков содержат свойства, характеризующие цель (некоторый элемент пользовательского интерфейса) и длительность (в миллисекундах), за которое данное действие будет воспроизведено. Исключением являются блоки, которые находят некоторый элемент пользовательского интерфейса. Свойства этих блоков это строки, описывающие тип и имя искомого объекта или ссылка на родителя, найденного ранее иным способом, и позиция в его списке детей данного элемента.

Для трансляции реализованного языка в QtScript был реализован генератор кода. Он переводит каждый блок код обращения к функции API, где свойства подставляются на место аргументов, или некоторую синтаксическую конструкцию языка QtScript. Платформа QReal содержит

средства для создания генераторов кода [3], однако они расчитаны на роботические модели, поэтому потребовалось переопределение основных элементов языка, реализующих стандартные конструкции.

По причине того что некоторые функции API содержат перегрузки, были реализованы проверки заполненности нужного свойства. В зависимости от его пустоты, генерируется код обращения к функции с соответствующей перегрузкой.

Полученный язык позволяет создавать диаграммы, описывающие последовательности пользовательских действий в платформе QReal.

2.4. Запись действий пользователя

Одним из элементов описанного в данной работе языка стал блок записи пользовательских действий. Данный элемент позволяет записывать произвольные последовательности действий пользователя с системой QReal и восстанавливать их для последующего воспроизведения. Таким образом, пользователь сможет записывать свои действия в среде, не применяя навыков программирования, а просто выполнив их; представленный в данной работе инструмент самостоятельно подготовит их для последующего воспроизведения.

Блок записи фиксирует все значимые события, спровоцированные пользователем (например нажатие клавиши компьютерной мыши или ввода текста) и сериализует их в формат XML (рис. 4).

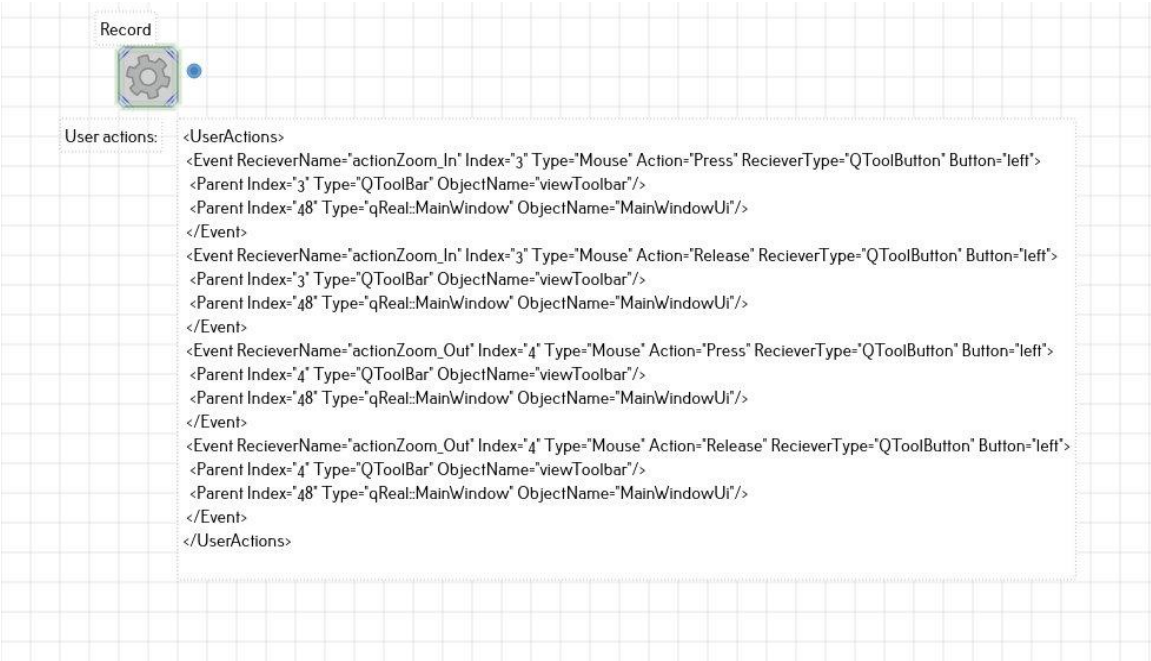

Рис. 4 блок записи с цепочкой записанных событий в его свойстве.

Основной проблемой в данной задаче является восстановление ссылки на элемент пользовательского интерфейса, получивший текущее событие. Поэтому вместе с описанием события записываются также и все “родители” получателя, вместе с их именами, типами и позицией в списке “детей”. Помимо этого, многие события, направленные на элементы пользовательского интерфейса требуют особого подхода, например событие, получателем которого является сцена требует фиксации позиции в ее системе координат, а событие с выпадающим списком в роли получателя требует записи выбранного пункта. Некоторые элементы пользовательского интерфейса вообще создаются динамически и требуют особого подхода.

Рассмотрев ряд данных проблем, было принято решение разработать набор эвристик [8] для записи значимых событий, так как не было

обнаружено приемлемых путей для решения данной задачи в общем случае. Таким образом, были рассмотрены большинство элементов пользовательского интерфейса и разработаны индивидуальные подходы к фиксации событий, направленных на них.

2.5. Восстановление записанных действий пользователя

Следующим этапом реализации данного инструмента стал процесс восстановления цепочки пользовательских действий из полученного ранее XML-файла, содержащего значимые события. Так как при записи использовался индивидуальный подход к элементам интерфейса, то и при их восстановлении потребуется соответствующий набор эвристик.

Например, для восстановления ссылки на элемент генерируется инициализация трех массивов (массив типов, имен и индекс в “родительском” массиве “детей” данного элемента интерфейса) и цикл, на каждой итерации которого производится поиск следующего объекта в цепочке “родителей” по доступным параметрам (индексу или типу и имени, если по некоторой причине запись индекса не удалась). На последней итерации будет получена ссылка на искомый элемент пользовательского интерфейса (рис. 5).

```
var Index = [-1, -1, 1];
var Type = ["qReal::MainWindow", "QDockWidget", "qReal::gui::PaletteTree", "QComboBox"];
var ObjectName = ["MainWindowUi", "paletteDock", "paletteTree", ""];
var widget = api.ui().widget("qReal::MainWindow", "MainWindowUi");
for (var i = 1; i <= 3; i ++)
{
        if (Index[i - 1] == -1) {
                widget = api.ui().widget(Type[i], ObjectName[i], widget);
        } else {
                widget = api.ui().widgetByIndex(Index[i - 1], widget);
        }
}
api.cursor().moveTo(widget, 500);
```

Рис. 5 код на языке QtScript с использованием API, который находит элемент графического интерфейса

Зачастую, значимым элементом пользовательского интерфейса является не сам получатель события, а один из его родителей. Например, полосы прокрутки и выпадающие списки. В случае взаимодействия с ними получателем события становится некоторая область прокрутки, которая остается неизменной, поэтому необходимо рассмотреть всех родителей получателя, после чего выбрать из них первый значимый и генерировать код, изменяющий его состояние.

Также необходимо учитывать, что при определенных действиях может сменится активное окно, а так как виртуальный курсор это тоже элемент графического интерфейса, то ему необходимо сменить родителя в данном случае.

Помимо этого были реализованы функции для генерации примитивных действий: перетаскивание элемента палитры на сцену, события мыши (движения, нажатия клавиш) и т.д. Данные функции напрямую подставляют записанные свойства события в качестве аргументов функций API.

Функции, описанные в API и расширенные стандартными конструкциями языка QtScript, позволяют описать, с их помощью, практически произвольную последовательность действий пользователя. Набор реализованных эвристик позволил генерировать из цепочки низкоуровневых событий QtScript-код (рис. 6).

```
var mainWindow = api.ui().mainWindow();
api.changeWindow(mainWindow);
var Index = [-1, -1, 1];
var Type = ["qReal::MainWindow", "QDockWidget", "qReal::gui::PaletteTree", "QComboBox"];
var ObjectName = ["MainWindowUi", "paletteDock", "paletteTree", ""];
var widget = api.ui().widget("qReal::MainWindow", "MainWindowUi");
for (var i = 1; i <= 3; i ++)
{
        if (Index[i - 1] == -1) {
                widget = api.ui().widget(Type[i], ObjectName[i], widget);
        } else {
                widget = api.ui().widgetByIndex(Index[i - 1], widget);
        }
}
api.cursor().moveTo(widget, 500);
api.cursor().leftButtonPress(widget);
api.wait(100);
api.pickComboBoxItem(widget, 1, 1000);
```

Рис. 6 пример кода на языке QtScript с использованием API

Глава 3. Апробация

Для апробации полученной технологии был создан набор обучающих демонстраций, который будет использоваться для реального обучения пользователей на мастер-классах, в школах и дома. Примеры реализованы на полученном языке в виде диаграмм на платформе QReal и по ним сгенерирован соответствующих код на языке QtScript.

Показанная на рисунке 7 диаграмма исполняет обучающую демонстрацию в TRIK Studio, которая описывает все основные элементы ее управления (рис. 8) и создает с их помощью примитивную диаграмму из нескольких блоков (движение робота по прямой в течение некоторого времени). В процессе демонстрации пользователю периодически показываются подсказки о проводимых действиях со средой (рис. 9).

Апробация показала, что реализованный язык позволяет создавать обучающие демонстрации в платформе QReal и ее плагинах. Однако, его использование сопряжено с потребностью в некоторых знаниях о внутреннем устройстве среды. Так как процесс записи справляется с большинством наиболее используемых пользовательских сценариев, то наиболее полезным для пользователей станет этот блок и блоки “подсказок”. С другой стороны, более продвинутые пользователи смогут создавать содержательные диаграммы на данном языке, изучив генерируемый код.

Github: https://github.com/DmitriyChernov/qreal/tree/tutorial-userAction

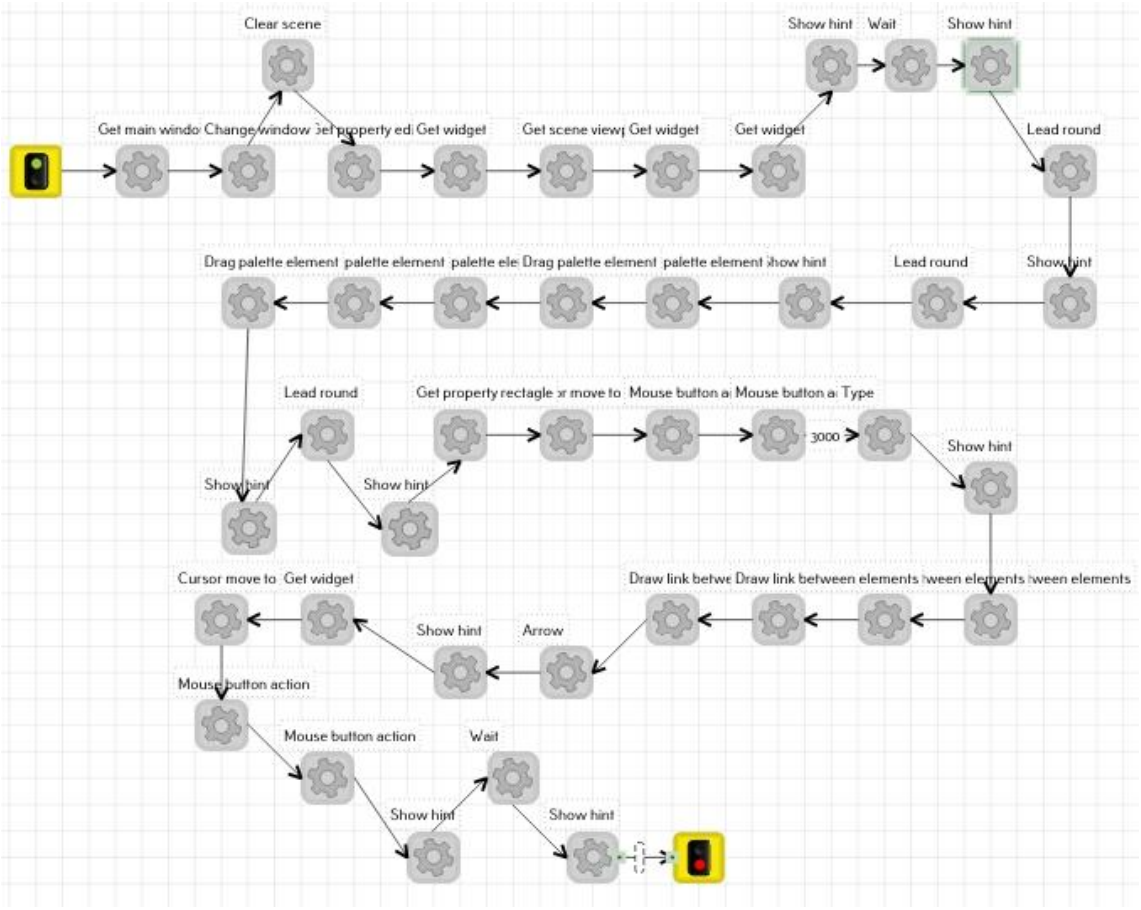

Рис. 7 диаграмма, выполняющая демонстрацию создания простейшей программы в TRIK Studio

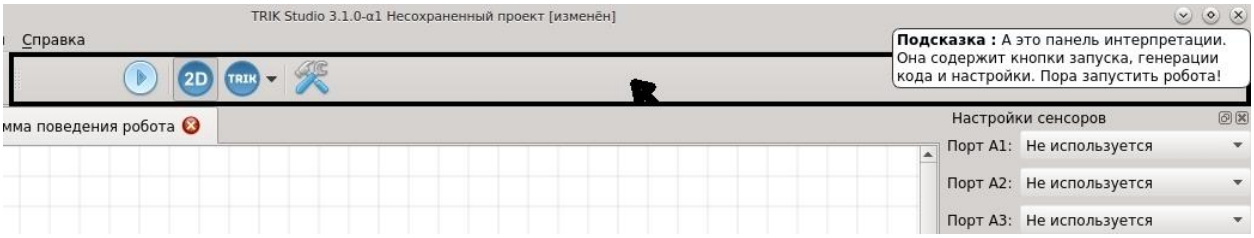

Рис. 8 демонстрация основных элементов управления TRIK Studio

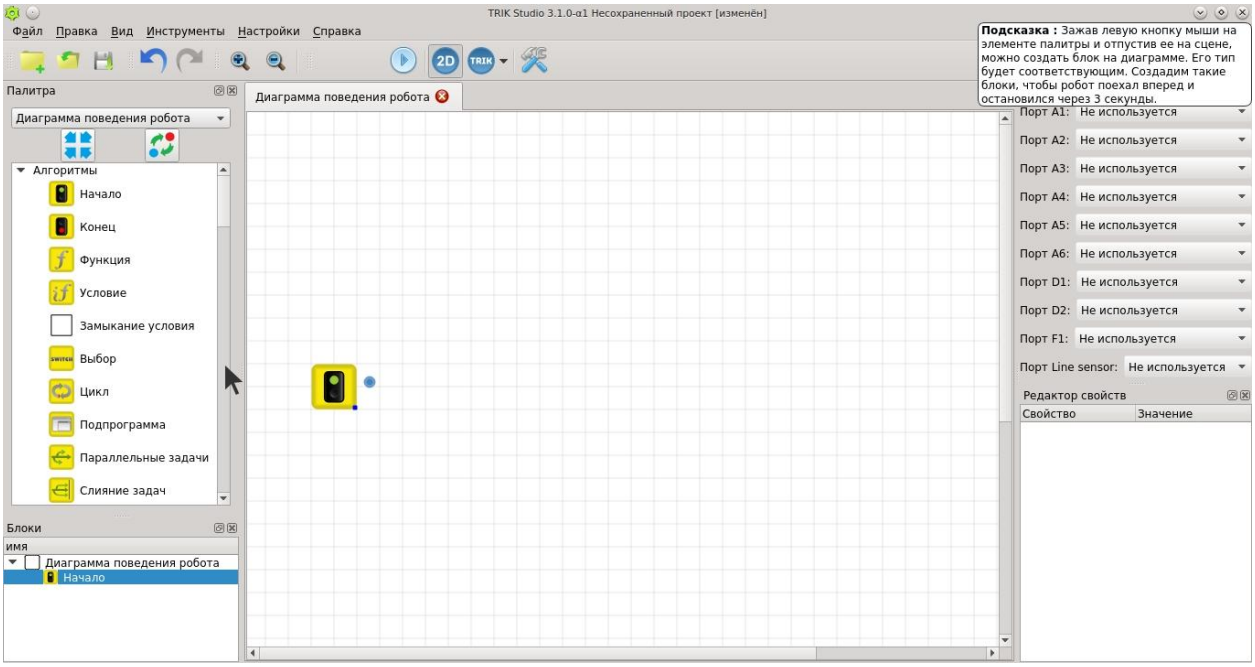

Рис. 9 всплывающие подсказки, для описания производимых действий

Заключение

В рамках данной работы было проведено исследование концепций обучения и скриптования действий пользователя в различных программных средах и разработана модель обучающей инфраструктуры в среде TRIK Studio.

Был реализован язык описания пользовательских действий и расширен инструментом, позволяющим записывать цепочки действий пользователя.

Полученная технология позволит создавать обучающие демонстрации в DSM-платформе QReal, а также будет полезна при написании тестов для пользовательского интерфейса среды.

Данная технология была апробирована путем создания набора обучающих демонстраций.

Список литературы

1. Кузенкова А.С., Дерипаска А.О., Таран К.С., Подкопаев А.В., Литвинов Ю.В., Брыксин Т.А., Средства быстрой разработки предметно-ориентированных решений в metaCASE-средстве QReal // Научно-технические ведомости СПбГПУ, Информатика, телекоммуникации, управление. Вып. 4 (128). СПб.: Изд-во Политехнического Университета. 2011, С. 142-145.
2. Литвинов Ю.В., Кириленко Я.А., TRIK Studio: среда обучения программированию с применением роботов // V Всероссийская конференция «Современное технологическое обучение: от компьютера к роботу» (сборник тезисов), СПб., ЗАО «Полиграфическое предприятие № 3», 2015, С. 5-7 https://robofinist.ru/uploads/2015/Thesis_2015.pdf
3. Подкопаев А.В., Брыксин Т.А., Средства описания генераторов кода для предметно-ориентированных решений в metaCASE-средстве QReal // Список-2012: Материалы всероссийской научной конференции по проблемам информатики. 25-27 апр. 2012г., Санкт-Петербург. — СПб.: Изд-во ВВМ, 2012. С. 49-55
4. Соковикова Н.А., Usability в проекте QReal:Robots // Список-2012: Материалы всероссийской научной конференции по проблемам информатики. 25-27 апр. 2012г., Санкт-Петербург. — СПб.: Изд-во ВВМ, 2012. С. 66-69. https://github.com/qreal/qreal/wiki/SPISOK-2012-Sokovikova.pdf
5. Терехов А.Н., Брыксин Т.А., Литвинов Ю.В., Среда визуального программирования роботов QReal:Robots // III Всероссийская конференция «Современное технологическое обучение:от компьютера

к роботу» (сборник тезисов), СПб., 2013, С. 1-4 (электронная публикация, http://www.239.ru/userfiles/file/Abstract_CompRobot2013.pdf

6. Чернов Д.В., Разработка средств для создания обучающих демонстраций в среде QReal:Robots, http://se.math.spbu.ru/SE/YearlyProjects/2014/YearlyProjects/2014/371/371-Chernov-report.pdf
7. Atsushi Sugiura, Yoshiyuki Koseki, Simplifying macro definition in programming by demonstration // UIST Proceedings of the 9th annual ACM symposium on User interface software and technology, 1996, pp 173-182
8. Ben Lafreniere , Tovi Grossman, George Fitzmaurice, Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations // CHI`13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013, pp 1779-1788
9. Cheng-Yao Wang, Wei-Chen Chu, Hou-Ren Chen, Chun-Yen Hsu, Mike Y. Chen, EverTutor: Automatically Creating Interactive Guided Tutorials on Smartphones by User Demonstration // CHI`13 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013, pp 4027-4036
10. Eleanor O'Rourke, Erik Andersen, Sumit Gulwani, Zoran Popovic´, A Framework for Automatically Generating Interactive Instructional Scaffolding // CHI`15 Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, pp 1545-1554
11. Francesmary Modugno, Albert T. Corbett, Brad A. Myers, Graphical Representation of Programs in a Demonstrational Visual Shell—An Empirical Evaluation // ACM Transactions on Computer-Human Interaction (TOCHI), 1997, pp 276-308

12. Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, Björn Hartmann, MixT: automatic generation of step-by-step mixed media tutorials // UIST '12 Proceedings of the 25th annual ACM symposium on User interface software and technology, 2012, pp 93-102
13. Rahul Agarwal, Stephen H. Edwards, and Manuel A. Pérez-Quiñones, Designing an Adaptive Learning Module to Teach Software Testing // ACM SIGCSE Bulletin, pp 259-263
14. Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, Michael F. Cohen, Pause-and-Play: Automatically Linking Screencast Video Tutorials with Applications // UIST`11 Proceedings of the 24th annual ACM symposium on User interface software and technology, 2011, pp 135-144