ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Пользовательский интерфейс

Цель занятия: закрепить теоретические знания по разработке пользовательского интерфейса; получить практические навыки по проектированию и разработке пользовательского интерфейса.

Теоретические сведения

Существуют три совершенно различные модели пользовательского интерфейса: модель программиста, модель пользователя и программная модель. Программист, разрабатывая пользовательский интерфейс, исходит из того, управление какими операциями ему необходимо реализовать в пользовательском интерфейсе, и как это осуществить, не затрачивая ни существенных ресурсов компьютера, ни своих сил и времени. Его интересуют функциональность, эффективность, технологичность, внутренняя стройность и другие не связанные с удобством пользователя характеристики программного обеспечения. Именно поэтому большинство интерфейсов существующих программ вызывают серьёзные нарекания пользователей.

С точки зрения здравого смысла хорошим следует считать интерфейс, при работе с которым пользователь получает именно то, что он ожидал.

Представление пользователя о функциях интерфейса можно описать в виде пользовательской модели интерфейса.

Пользовательская модель интерфейса - это совокупность обобщённых представлений конкретного пользователя или некоторой группы пользователей о процессах, происходящих во время работы программы или программной системы. Эта модель базируется на особенностях опыта конкретных пользователей, который характеризуется:

- уровнем подготовки в предметной области разрабатываемого программного обеспечения;
- интуитивными моделями выполнения операций в этой предметной области;
- уровнем подготовки в области владения компьютером;
- устоявшимися стереотипами работы с компьютером.

Для построения пользовательской модели необходимо изучить перечисленные выше особенности опыта предполагаемых

пользователей программного обеспечения. С этой целью используют опросы, тесты и даже фиксируют последовательность действий, осуществляемых в процессе выполнения некоторых операций, на плёнку.

Приведение в соответствие моделей пользователя и программиста, а также построение на их базе программной модели (рис. 1) интерфейса задача не тривиальная. Причём, чем сложнее автоматизируемая предметная область, тем сложнее оказывается построить программную модель интерфейса, учитывающую особенности пользовательской модели и не требующую слишком больших затрат как в процессе разработки, так и во время работы. С этой точки зрения объектные интерфейсы кажутся наиболее перспективными, так как в их основе лежит именно отображение объектов предметной области, которыми оперируют пользователи. Хотя на настоящий момент времени их реализация достаточно трудоёмка.

Рисунок 1 – Процесс проектирования пользовательского интерфейса

При создании программной модели интерфейса также следует иметь в виду, что изменять пользовательскую модель непросто.

Повышение профессионального уровня пользователей и их подготовки в области владения компьютером в компетенцию разработчиков программного обеспечения не входит, хотя часто грамотно построенный интерфейс, который адекватно отображает сущность происходящих процессов, способствует росту квалификации пользователей.

Интуитивные модели выполнения операций в предметной области должны стать основой для разработки интерфейса, а потому в большинстве случаев их необходимо не менять, а уточнять и совершенствовать. Именно нежелание или невозможность следования интуитивным моделям выполнения операций приводит к созданию искусственных надуманных интерфейсов, которые негативно воспринимаются пользователями.

Критерии оценки интерфейса пользователем. Многочисленные опросы и обследования, проводимые ведущими фирмами по разработке

программного обеспечения, показали, что основными критериями оценки интерфейсов пользователем являются:

- простота освоения и запоминания операций системы – конкретно оценивают время освоения и продолжительность сохранения информации в памяти;

- скорость достижения результатов при использовании системы определяется количеством вводимых или выбираемых мышью команд и настроек;

- субъективная удовлетворённость при эксплуатации системы удобство работы, утомляемость и т. д.).

Типы интерфейсов

По аналогии с процедурным и объектным подходом к программированию различают процедурно-ориентированный и объектно-ориентированный подходы к разработке интерфейсов (рис. 2).

Различают процедурно – ориентированные интерфейсы трёх типов: «примитивные», меню и со свободной навигацией.

Примитивным называют интерфейс, который организует взаимодействие с пользователем в консольном режиме. Обычно такой интерфейс реализует конкретный сценарий работы программного обеспечения, например, ввод данных - решение задачи - вывод результата.

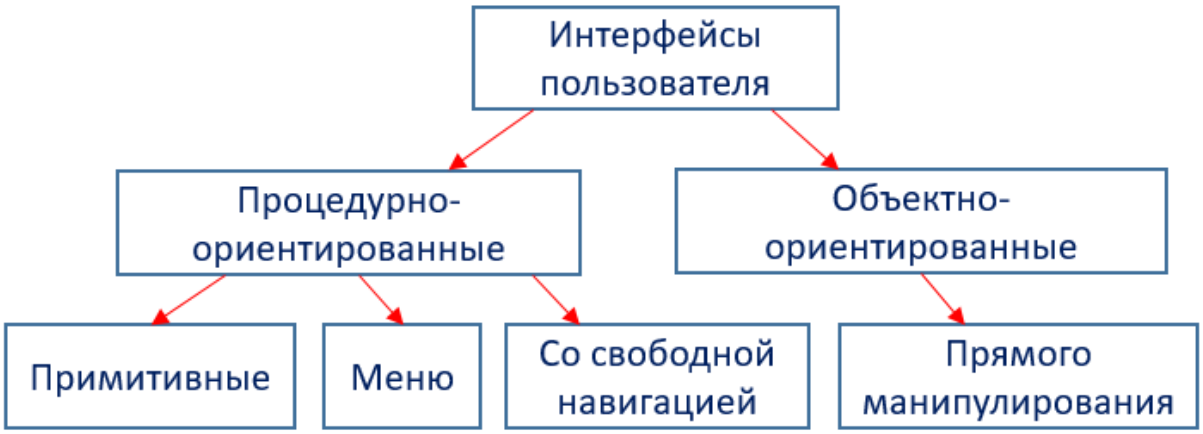


Рисунок 2 – Типы интерфейсов

Единственное отклонение от последовательного процесса, которое обеспечивается данным интерфейсом, заключается в организации цикла для обработки нескольких наборов данных. Подобные интерфейсы в настоящее время используют только в процессе обучения программированию или в тех случаях, когда вся программа реализует одну функцию, например, в некоторых системных утилитах.

Интерфейс-меню в отличие от примитивного интерфейса позволяет пользователю выбирать необходимые операции из специального списка, выводимого ему программой. Эти интерфейсы предполагают реализацию множества сценариев работы, последовательность действий в которых определяется пользователем. Различают одноуровневые и иерархические меню. Первые используют для сравнительно простого управления вычислительным процессом, когда вариантов немного (не более 5-7), и они включают операции одного типа, например, «Создать», «Открыть», «Закрыть» и т. п. Вторые - при большом количестве вариантов или их очевидных различиях, например, операции с файлами и операции с данными, хранящимися в этих файлах. Интерфейсы данного типа несложно реализовать в рамках структурного подхода к программированию.

Интерфейсы-меню в настоящее время используют редко и только для сравнительно простого программного обеспечения или в разработках, которые должны быть выполнены по структурной технологии и без использования специальных библиотек.

Интерфейсы со свободной навигацией также называют графическими пользовательскими интерфейсами (GUI - Graphic User Interface) или интерфейсами WYSIWYG (What You See Is What You Get – что видишь, то и получишь, т. е., что пользователь видит на экране, то

он и получит при печати). Эти названия подчёркивают, что интерфейсы данного типа ориентированы на использование экрана в графическом режиме с высокой разрешающей способностью.

Графические интерфейсы поддерживают концепцию интерактивного взаимодействия с программным обеспечением, осуществляя визуальную обратную связь с пользователем и возможность прямого манипулирования объектами и информацией на экране. Кроме того, интерфейсы данного типа поддерживают концепцию совместимости программ, позволяя перемещать между ними информацию.

В отличие от интерфейса-меню интерфейс со свободной навигацией обеспечивает возможность осуществления любых допустимых в конкретном состоянии операций, доступ к которым возможен через различные интерфейсные компоненты. Например, окна программ, реализующих интерфейс Windows, обычно содержат:

- меню различных типов: ниспадающее, кнопочное, контекстное;
- разного рода компоненты ввода данных.

Причём выбор следующей операции в меню осуществляется как мышью, так и с помощью клавиатуры.

Реализуют интерфейсы со свободной навигацией, используя событийное программирование и объектно-ориентированные библиотеки, что предполагает применение визуальных сред разработки программного обеспечения.

Объектно-ориентированные интерфейсы пока представлены только интерфейсом прямого манипулирования. Этот тип интерфейса предполагает, что взаимодействие пользователя с программным обеспечением осуществляется посредством выбора и перемещения пиктограмм, соответствующих объектам предметной области. Для реализации таких интерфейсов также используют событийное программирование и объектно-ориентированные библиотеки.

Задание для выполнения

1. Разработать графический интерфейс обучающе-контролирующего приложения. Приложение должно быть разработано в среде PascalABC.NET.
2. Сдать и защитить работу.

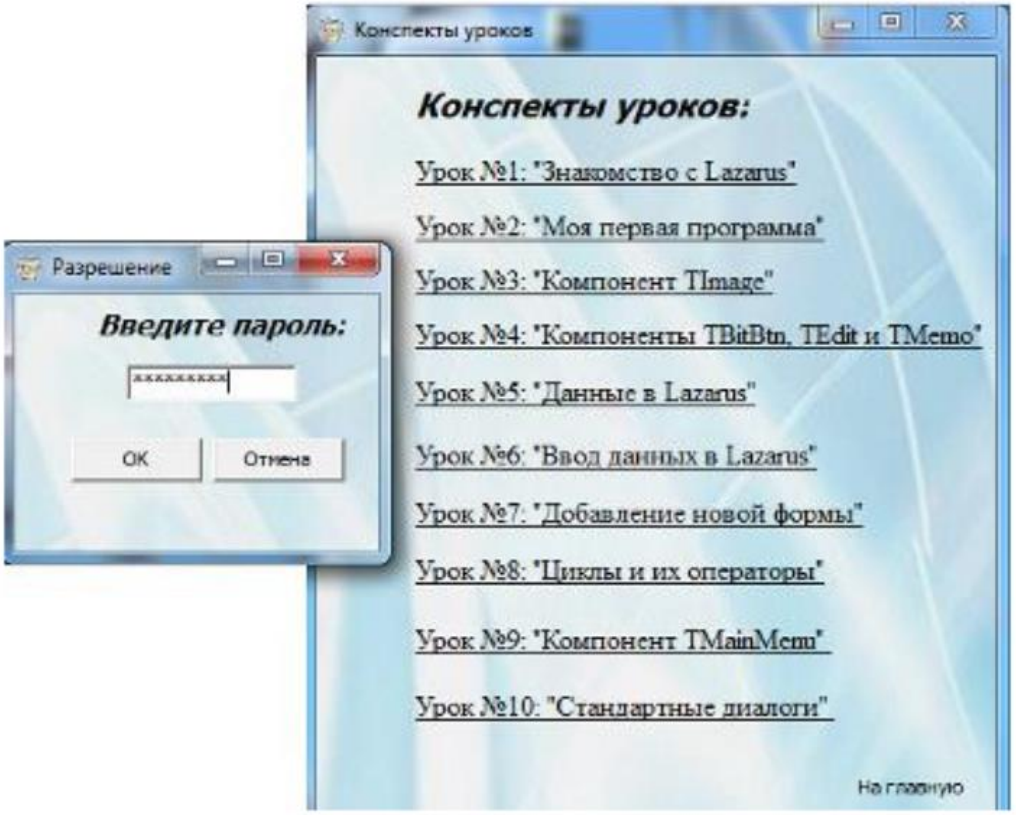


Рисунок 3 – Пример обучающе-контролирующего приложения «Конспекты уроков»

Рисунок 4 – Интерфейс со свободной навигацией

Контрольные вопросы

1. Какие типы интерфейсов являются основными в наше время?
2. Перечислите психофизические особенности человека, которые необходимо учитывать при проектировании интерфейсов.
3. Какие ограничения накладывают на интерфейс психофизические особенности человека?

При подготовке ответов на контрольные вопросы можно воспользоваться и материалами лекций.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Диалоги в графическом пользовательском интерфейсе

Цель занятия: закрепить теоретические знания по разработке пользовательского интерфейса; получить практические навыки по реализации диалогов в графическом пользовательском интерфейсе.

Теоретические сведения

Диалог - это процесс обмена информацией между пользователем и программной системой, осуществляемый через интерактивный терминал и по определённым правилам.

Различают тип диалога и его форму.

Типы диалога. Тип диалога определяет, кто из «собеседников» управляет процессом обмена информацией. Соответственно различают два типа диалога: управляемые программой и управляемые пользователем.

Диалог, управляемый программой, предусматривает наличие жесткого, линейного или древовидного, т. е. включающего возможные альтернативные варианты, сценария диалога, заложенного в программное обеспечение. Такой диалог обычно сопровождают большим количеством подсказок, которые уточняют, какую информацию необходимо вводить на каждом шаге.

Диалог, управляемый пользователем, подразумевает, что сценарий диалога зависит от пользователя, который применяет систему для выполнения необходимых ему операций. При этом система обеспечивает возможность реализации различных пользовательских сценариев.

Формы диалога. Никакой диалог невозможен, если не существует языка, понятного «собеседникам». Описание языка, на котором ведётся диалог, включает определение его синтаксиса - правил, определяющих допустимые конструкции (слова, предложения) языка или его форму, и семантики - правил, определяющих смысл синтаксически корректных конструкций языка или его содержание. В зависимости от вида используемых в конкретном случае синтаксиса и семантики различают три формы диалога:

- фразовую,
- директивную,
- табличную.

1. Фразовая форма предполагает «общение» с пользователем на естественном языке или его подмножестве. Содержание диалога в данной форме составляют повелительные, повествовательные и вопросительные предложения и ответы на вопросы. Общение может осуществляться в свободном формате, но возможна и фиксация отдельных фраз. Организация диалога на естественном языке на современном уровне - задача не решённая, так как естественный язык крайне сложен и пока не удаётся в достаточной степени формализовать его синтаксис и семантику.

Чаще всего используют диалоги, предполагающие односложные ответы, например:

Программа: Введите свой возраст (полных лет):

Пользователь: 48.

В этом случае программа содержит ограниченное описание как синтаксиса, так и семантики используемого ограниченно-естественного языка. Для данного примера достаточно определить синтаксис понятия «целое положительное число» и наложить ограничение на значение числа.

Однако существует некоторый опыт создания интерфейсов на базе ограниченного подмножества предложений естественного языка в основном для интеллектуальных систем. Синтаксис и семантика языков диалога, реализуемых в таких интерфейсах, достаточно сложны. При обработке фраз в этих случаях оперируют понятием словоформа.

Словоформа - отрезок текста между двумя соседними пробелами или знаками препинания. Обработка словоформ вне связи с контекстом называется морфологическим анализом.

Интерфейс, реализующий фразовую форму диалога, должен: преобразовывать сообщения из естественно-языковой формы в форму внутреннего представления и обратно, выполнять анализ и синтез сообщений пользователя и системы, отслеживать и запоминать пройдённую часть диалога.

Основными недостатками фразовой формы при использовании подмножества естественного языка являются:

- большие затраты ресурсов;
- отсутствие гарантии однозначной интерпретации формулировок;
- необходимость ввода длинных грамматически правильных фраз.

Основное достоинство фразовой формы состоит в относительно свободном общении с системой.

2. Директивная форма предполагает использование команд директив) специально разработанного формального языка. Командой в этом случае называют предложение этого языка, описывающее комбинированные данные, которые включают идентификатор инициируемого процесса и, при необходимости, данные для него.

Команду можно вводить:

- в виде строки текста, специально разработанного формата, например, команды MS DOS, которые вводятся в командной строке;
- нажатием некоторой комбинации клавиш клавиатуры, например, комбинации «быстрого доступа» современных Windows-приложений;
- посредством манипулирования мышью, например, «перетаскиванием» пиктограмм;
- комбинацией второго и третьего способов.

Основными достоинствами директивной формы являются:

- сравнительно небольшой объем вводимой информации;
- гибкость - возможности выбора операции в данном случае ограничены только набором допустимых команд;
- ориентация на диалог, управляемый пользователем;
- использование минимальной области экрана или неиспользование ее вообще;
- возможность совмещения с другими формами.

Недостатки директивной формы:

- практическое отсутствие подсказок на экране, что требует запоминания вводимых команд и их синтаксиса;
- почти полное отсутствие обратной связи о состоянии инициированных процессов;
- необходимость навыков ввода текстовой информации или манипуляций мышью;
- отсутствие возможности настройки пользователем.

Исследования показали, что директивная форма удобна для пользователя-профессионала, который обычно быстро запоминает синтаксис часто используемых команды или комбинации клавиш.

Основные достоинства формы (гибкость и хорошие временные характеристики) проявляются в этом случае особенно ярко.

3. Табличная форма предполагает, что пользователь выбирает

ответ из предложенных программой. Язык диалога для табличной формы имеет простейший синтаксис и однозначную семантику, что достаточно легко реализовать. Удобна эта форма и для пользователя, так

как выбрать всегда проще, чем вспомнить, что особенно существенно для пользователя-непрофессионала или пользователя, редко использующего конкретное программное обеспечение. Однако применение табличной формы возможно не всегда: ее можно использовать только, если множество возможных ответов на конкретный вопрос конечно. Причём, если количество возможных ответов велико (более 20), то применение табличной формы может оказаться нецелесообразным.

Достоинствами табличной формы являются:

- наличие подсказки, что уменьшает нагрузку на память пользователя, так как данная форма ориентирована не на запоминание, а на узнавание;
- сокращение количества ошибок ввода: пользователь не вводит информацию, а указывает на неё;
- сокращение времени обучения пользователя;
- возможность совмещения с другими формами;
- в некоторых случаях возможность настройки пользователем.

К недостаткам данной формы относят.

- необходимость наличия навыков навигации по экрану;
- использование сравнительно большой площади экрана для изображения визуальных компонентов;
- интенсивное использование ресурсов компьютера, связанное с необходимостью постоянного обновления информации на экране.

Следует иметь в виду, что типы и формы диалога выбирают независимо друг от друга: любая форма применима для обоих типов диалогов (рис. 1).

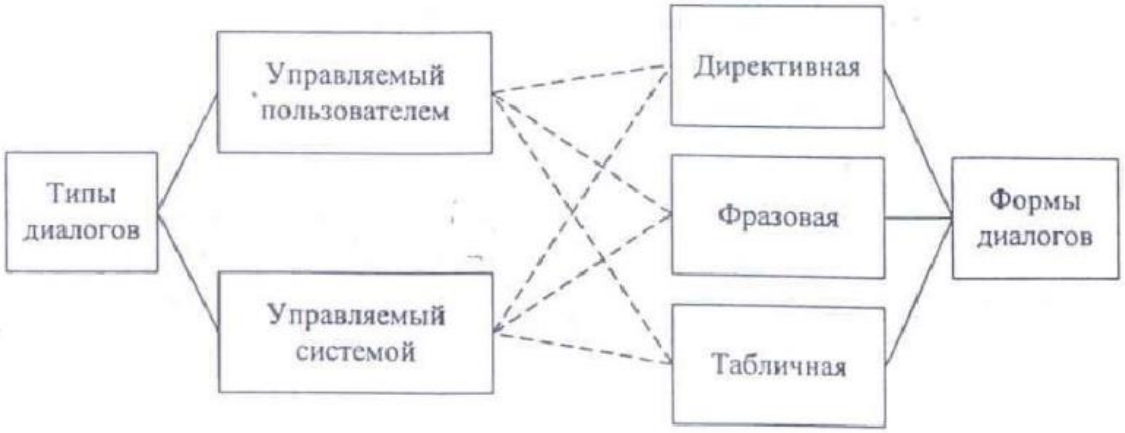


Рисунок 1 – Соответствие типов диалогов и его форм

Разработка диалогов. Процесс проектирования и реализации диалогов можно разделить на следующие стадии:

- определение множества необходимых диалогов, их основных сообщений и возможных сценариев — проектирование абстрактных диалогов;

- определение типа и формы каждого диалога, а также синтаксиса и семантики используемых языков - проектирование конкретных диалогов;

- выбор основных и дополнительных устройств и проектирование процессов ввода-вывода для каждого диалога, а также уточнение передаваемых сообщений – проектирование технических диалогов.

В основу абстрактных диалогов должна закладываться идеология технологического процесса, для автоматизации которого предназначается программный продукт. Именно анализируя составляющие автоматизируемого технологического процесса, разработчик определяет сценарии диалогов, которые должны быть предусмотрены в программном обеспечении.

Кроме сценариев, при проектировании абстрактных диалогов используют диаграммы состояний интерфейса или графы диалога.

Граф диалога — ориентированный взвешенный граф, каждой вершине которого сопоставлена конкретная картинка на экране (кадр) или определённое состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий. В качестве весов дуг указывают условия переходов из состояния в состояние и операции, выполняемые во время перехода.

Таким образом, каждый маршрут на графе соответствует возможному варианту диалога. Причём представление диалога в виде графа в зависимости от стадии разработки может выполняться с разной степенью детализации. По сути граф диалога - это граф состоянии конечного автомата, моделирующего поведение программного обеспечения при воздействиях пользователя. Для представления таких графов уже были введены две нотации: нотация диаграмм состояний структурного подхода к разработке и нотация диаграмм состояний UML.

Причём нотация UML является более мошной, так как позволяет использовать обобщённые состояния. Поэтому, чтобы не вводить новую

нотацию для представления графа диалога, будем использовать обозначения UML.

Как правило, сложное программное обеспечение с развитым пользовательским интерфейсом использует диалоги обоих типов: управляемые пользователем и управляемые системой.

Реализация диалогов, управляемых пользователем. Для реализации диалогов, управляемых пользователем, применяют меню различных видов: основное, панели инструментов, контекстные и кнопочные, т. е. сформированные из отдельных кнопок. Как альтернативу меню целесообразно использовать директивную форму диалога, поставив в соответствие основным командам определенные комбинации клавиш.

Кроме того, целесообразно предусмотреть возможность управления меню клавиатурой, что особенно важно, если большую часть времени работы с системой пользователь вводит текст или данные, т. е. взаимодействует с клавиатурой.

Меню.

Меню проектируют на основе графов диалогов разрабатываемого программного обеспечения. При этом, если число операций не превышает 5, то обычно используют кнопки. Если число операций не более 9-10, то - одноуровневое меню. И, наконец, если число реализуемых операций более 10, то используют «ниспадающее» двухуровневое иерархическое меню.

Ниспадающее меню. Первый уровень иерархического меню должен содержать имена основных групп операций. Традиционно первым является пункт Файл, вторым - Правка, третьим - Вид, а последним - Справка. Такое распределение пунктов специфично для программ обработки данных, размещённых в файлах, например, текстовых и графических редакторов. В последнее время с таким распределением пунктов возникают проблемы, так как большинство программ уже не работает с данными традиционным способом.

Количество уровней иерархического меню не должно превышать 2-3, так как при большем числе уровней требуемую операцию будет сложно искать. Причём желательно, чтобы число операций в окне меню не превышало 7-8, по той же причине.

Если число операций превышает 70-80, то возникает проблема, как построить наглядное меню с таким большим числом операций.

Интересное решение было предложено разработчиками Microsoft Word.

Они реализовали адаптивное иерархическое меню, где содержимое окна меню второго уровня постоянно меняется, отображая только те операции, которые использует пользователь. Если пользователь не находит нужной операции, то через несколько секунд или при нажатии специальной кнопки Word демонстрирует окно меню полностью.

Панель инструментов. На панель инструментов помещают пиктограммы часто используемых операций. Если множество таких операций существенно зависит от специфики выполняемых с разрабатываемым программным обеспечением работ, то целесообразно

обеспечить пользователю возможность формирования панелей инструментов по собственному усмотрению. В качестве примера можно посмотреть, как реализована операция настройки (Сервис\Настройка) Microsoft Word.

Контекстные меню. Контекстные меню включают операции, вероятность обращения к которым из данной зоны окна приложения с точки зрения разработчика максимальна. В процессе тестирования «удобства использования» содержание контекстного меню может уточняться. Так же, как и в случае основного меню, нежелательно, если число операций этого меню превышает 6-8. Причём, чтобы облегчить пользователю поиск нужной операции целесообразно операции контекстного меню горизонтальными линиями делить на группы.

Реализация диалогов, управляемых системой. Для реализации диалогов, управляемых системой, обычно используют диалоговые окна.

Причём, если число настраиваемых в процессе диалога элементов невелико, и диалогу соответствует последовательный сценарий, то проектируют одно диалоговое окно, включающее все необходимые компоненты. Такое окно часто называют формой. Если же диалог имеет сильно разветвлённую структуру, в которой следующий вопрос зависит от уже полученных ответов, или число настраиваемых в процессе диалога элементов велико, то для каждого шага диалога проектируют своё диалоговое окно.

Проектирование форм заключается в выборе необходимых компонентов интерфейса и размещении их в пределах диалогового окна.

Если количество компонентов более 4-5, то целесообразно их визуально разделить, используя рамки.

Проектирование последовательностей диалоговых окон. Как уже упоминалось выше, в основе диалогов, управляемых системой, лежит жёстко или не жёстко заданный сценарий. Именно этот сценарий

должен быть реализован последовательностью диалоговых окон. Независимо от степени жёсткости сценария при проектировании такой последовательности необходимо предусмотреть возможность возврата на предыдущий шаг.

Задание для самостоятельного выполнения

1. Разработать диаграмму вариантов использования для обучающе-контролирующей программы (см. материалы лекции).

2. Построить таблицу «Вариант использования одного из диалогов», а также детализировать один из диалогов обучающе-контролирующей программы.

3. Сдать и защитить работу.

Пример таблицы «Вариант использования диалога»

Действие исполнителя	Отклик системы
1. Пользователь инициирует новое задание	*2. Система регистрирует новое задание и предлагает список типов задач*
3. Пользователь выбирает тип задачи	*4. Система регистрирует тип задачи и предлагает список способов задания данных*
5.Пользователь выбирает способ задания данных: *а) Если выбран ввод с клавиатуры, см. раздел Ввод данных* *б) Если выбран ввод из базы данных, см. раздел Выбор данных из базы*	*6.Система регистрирует данные и предлагает список алгоритмов решения*
7.Пользователь выбирает алгоритм	*8.Система регистрирует алгоритм и предлагает начать решение*
9.Пользователь инициирует процесс решения	*10. Система проверяет полноту определения задания и запускает подпрограмму решения задачи*
11.Пользователь ожидает	*12. Система демонстрирует пользователю результаты и предлагает сохранить их в базе данных*
13. Пользователь анализирует результаты и выбирает, сохранять их в базе или нет	*14.Если выбрано сохранение данных, то система выполняет запись данных задания в базу* *15.Система переходит в состояние ожидания*

Контрольные вопросы

1. Что понимают под термином «диалог»?
2. Сколько диалогов может реализовывать программное обеспечение?
3. Назовите основные типы диалога и его формы.
4. Какие модели используют для описания диалогов?
5. Что служит исходными данными для проектирования диалогов?
6. В каких ситуациях граф диалога имеет вид цепи или дерева?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Интеллектуальные элементы пользовательских интерфейсов

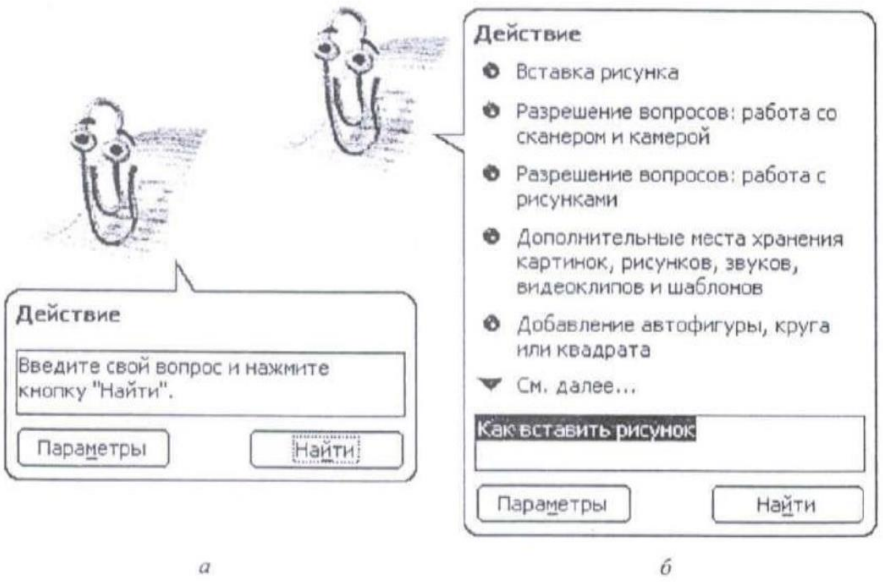
Цель занятия: закрепить теоретические знания по разработке пользовательского интерфейса; получить практические навыки по внедрению интеллектуальных элементов пользовательских интерфейсов.

Теоретические сведения

В последние годы появилось много новых перспективных элементов пользовательских интерфейсов, в основном привносящих в интерфейс элементы искусственного интеллекта, что проявляется в их названиях: Мастер, Советчик, Агент. Сделано множество попыток создания социализированного пользовательского интерфейса. В основе такого интерфейса лежит идея создания персонифицированного, т. е. «имеющего личность», интерфейса. Развлекающие программы, такие как Cats (Кошки) и Dogs (Собаки), реализующие достаточно сложное поведение домашних животных в разных ситуациях, показывают, что технически это вполне решаемая задача. Однако в этой области существуют психологические проблемы. В качестве примера вспомним, что даже «безобидный» Советчик Microsoft Office, рассмотренный ниже, вызывает у многих пользователей резко отрицательную реакцию. Пока попытки создания такой «личности» успеха не имели.

Советчики. Советчики представляют собой форму подсказки.

Обычно их можно вызвать с помощью меню справки, командной строки окна или из всплывающего меню. Советчики помогают пользователям в выполнении конкретных задач, но только, если пользователь представляет, что ему нужно сделать. Например, пользователь, работающий в Microsoft Word, собирается вставить в документ рисунок, но не знает как. Он активизирует Помощника-Скрепку и вводит вопрос в специальное поле (рис. 6.1, а). Справочная система анализирует вопрос и формирует список тем, косвенно связанных с интересующей пользователя, в расчёте, что пользователь сам выберет нужную справку (рис. 1, б).



а *б*

Рисунок 1 – Помощник-Скрепка MS Word2000:
а – запрос; б – список связанных тем

Программа Мастер. Программу-мастер используют для выполнения общераспространённых, но редко выполняемых отдельным пользователем задач, таких, как установка программ или оборудования. Выполнение подобных действий требует от пользователя принятия сложных взаимосвязанных решений, последовательность которых и диктует программа-мастер. Интеллектуальные Мастера способны на каждом шаге демонстрировать в окне просмотра результаты ответов пользователя на предыдущие вопросы, помогая последнему сориентироваться в ситуации.

Мастер реализует последовательный или древовидный сценарий диалога, поэтому его целесообразно использовать для решения хорошо структурированных, последовательных задач. При этом необходимо:

- предоставить пользователю возможность возврата на предыдущий шаг;

- предусмотреть возможность отмены работы Мастера;

- нумеровать шаги и сообщать пользователю количество шагов Мастера, особенное, если таких шагов больше трёх;

- пояснять пользователю каждый шаг;

- по возможности демонстрировать результат уже выполненных операций на каждом шаге.

Программные агенты. Наибольший интерес на настоящий момент представляют программные агенты, используемые для выполнения рутинной работы. Такой программный агент является элементом программного обеспечения, которому пользователь может передать часть своих обязанностей. Основными функциями Агентов-Помощников являются: наблюдение, поиск и управление. Различают:

- программы-агенты, настраиваемые на выполнение указанных задач;

- программы-агенты, способные обучаться, например, фиксируя действия пользователя (по типу магнитофона).

Создание агентов последнего типа, например, доступно через механизм макросов Microsoft Office.

Большинство интересных и достаточно сложных программных агентов в настоящее время «живёт» в Интернете, где и можно найти последнюю информацию по данной теме.

Задания для выполнения

1. Внедрить интеллектуальный элемент в пользовательский интерфейс (элемент на Ваше усмотрение).

2. Сдать и защитить работу.

Контрольные вопросы

1. Какие интеллектуальные компоненты пользовательских интерфейсов существуют в настоящее время?

2. Каковы их основные назначения?

3. В каких случаях их целесообразно применять?