Разработка программ с графическим интерфейсом

Тема 8.

Интерфейс программы и событийное программирование

Лекция 8.1.

Интерфейс Windows-программ.

Вопросы:

1. Понятие интерфейса программы.

2. Компоненты интерфейса.

3. IDE (интегрирующая среда разработки)

1. Понятие интерфейса программы

Пользовательский интерфейс **– совокупность информационной модели предметной области, средств и способов взаимодействия пользователя с этой моделью, а также компонентов, обеспечивающих формирование модели в процессе работы программы.**

Информационная модель – условное представление предметной области, формируемое с помощью визуальных и звуковых объектов.

Средства и способы взаимодействия с информационной моделью определяются аппаратными и программными средствами, имеющихся в распоряжении пользователя, характером решаемых задач. Чтобы решить задачу на ЭВМ пользователь должен знать правила работы. Для аппаратных средств правила воспринимаются как вполне однозначные и понятные (например, CD не пытаются поместить гнездо для ГМД).

Сложнее с программными средствами:

- для программ трудно сформулировать объективные требования по составу и компоновке органов управления;

- перечень возможностей значительно шире, а состав меняется значительно быстрее, чем аппаратные средства.

Часто программы, равноценные по назначению, существенно различаются по организации взаимодействия с пользователем (интерфейсы не лучше или хуже, а просто различные).

Качество пользовательского интерфейса является одним из основных свойств программ. Необходимо уметь разрабатывать хороший интерфейс.

Интерфейс следует разрабатывать, ориентируясь на интересы пользователя. Свойства хорошего интерфейса:

***Естественность*. Обеспечивает привычные для пользователя способы работы с программой. Результаты обработки и сообщения программы не требуют дополнительных пояснений, обозначения и термины соответствуют предметной области. Использование знакомых пользователю образов и метафор (рабочий стол, папка, документ и др.).**

***Согласованность*. Обеспечивает перенос имеющихся представлений на новую программу (визуальное представление, имена команд, поведение визуальных элементов). Согласованность в пределах программы и операционной среды.**

***Дружественность*. Разрешение выполнения допустимых команд, предотвращение нежелательных последствий, предупреждение об опасных ситуациях, возможность отмены выполненного действия.**

***Наличие обратной связи*. Действия пользователя своевременно подтверждаются визуально или звуком.**

***Простота и полнота*. Обеспечивает легкость освоения и использования, доступность всех функциональных возможностей. Возможные подходы: отображение на экране информации, минимально необходимой для выполнения шага задания; отсутствие многословных команд и сообщений; группирование элементов интерфейса по их логической взаимосвязи; последовательное раскрытие диалоговых окон или меню.**

***Гибкость*. Учет уровня квалификации пользователя.**

***Эстетическая привлекательность*. Привлекательная цветовая гамма, возможность ее изменения по вкусу пользователя, грамотность текста, использование удобочитаемых шрифтов.**

Цвет **является сильным средством воздействия на психику человек. Неудачное цветовое решение приводит к быстрому утомлению пользователя, к частым ошибкам. Удачно подобранные цветовые решения, осмысленные цветовые акценты создают комфортные условия работы. С помощью цвета можно привлечь внимание, отобразить различные состояния объекта. Восприятие цвета индивидуально (некоторые люди не воспринимают цвета), в разных странах существуют различные традиции в применении цвета.**

Не следует злоупотреблять ярким цветом, например, сочетаниями яркого красного на зеленом или черном фоне, не желательно применять в значительном объеме фиолетовый цвет. Для фона лучше применять нейтральные цвета (светло-серый, салатный и т.п.). Лучше использовать базовый набор цветов, применение палитры из 16 или 32 миллионов цветов может замедлить работу приложения.

Стандартизованный **графический интерфейс пользователя (GUI – Graphical User Interface):**

- использование единой рабочей среды в виде рабочего стола. Объекты задачи представлены на столе в виде графических образов – пиктограмм и окон. Действия выполняются не только с помощью команд, а в основном путем прямого манипулирования объектами;

- многооконность. Позволяет получить доступ к нескольким источникам информации одновременно;

- объектно-ориентированный подход. Первичными являются обрабатываемые данные, а не средства обработки;

- применение средств неклавиатурного ввода информации;

- узнаваемый (использование стандартных элементов с узаконенными названиями, свойствами, отношениями);

- «работа с тем, что видишь»;

- «что видишь, то и получишь».

2. Компоненты интерфейса

Существуют две модели интерфейса приложений:
- интерфейс с одним документом SDI;
- интерфейс с множеством документов MDI.

Интерфейс SDI не обязательно содержит только одно окно, в нужные моменты можно создавать вторичные окна, например, для открытия файлов. Эту модель интерфейса следует считать основной.

В приложении MDI имеется родительское окно и ряд дочерних окон. Приложения MDI рекомендуется использовать, если дочерние окна содержат идентичные объекты, например текст. Применять эту модель интерфейса не рекомендуется.

IDE
(интегрирующая среда разработки)

Что это такое?

- **Интегрированная среда разработки** *integrated development environment, IDE — класс ПО, обеспечивающий организацию процесса разработки ПО через объединение основных необходимых для этого компонентов за общим “фасадом” (пользовательским интерфейсом).*

Среда разработки включает в себя:

- *Текстовый редактор*
- *Транслятор (компилятор и/или интерпретатор*
- *Средства автоматизации сборки*
- *Отладчик (дебаггер).*

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов — для использования при объектно-ориентированной разработке ПО. ИСР обычно предназначены для нескольких языков программирования — такие как IntelliJ IDEA, NetBeans, Eclipse, Xcode, Microsoft Visual Studio, но есть и IDE для одного определённого языка программирования — как, например, Visual Basic, Delphi, Dev-C++ и т.д.

Обзор

- *Использование ИСР для разработки программного обеспечения является прямой противоположностью способу, в котором используются несвязанные инструменты, такие как текстовый редактор, компилятор, и т. п. Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами. Это позволяет разработчику сделать меньше действий для переключения различных режимов, в отличие от дискретных программ разработки. Однако так как ИСР является сложным программным комплексом, то среда разработки сможет качественно ускорить процесс разработки ПО лишь после специального обучения.*
- *ИСР обычно представляет собой единственную программу, в которой проводится вся разработка. Она, как правило, содержит много функций для создания, изменения, компилирования, развертывания и отладки программного обеспечения. Цель интегрированной среды заключается в том, чтобы объединить различные утилиты в одном модуле, который позволит абстрагироваться от выполнения вспомогательных задач, тем самым позволяя программисту сосредоточиться на решении собственно алгоритмической задачи и избежать потерь времени при выполнении типичных технических действий (например, вызове компилятора). Таким образом, повышается производительность труда разработчика. Например, ИСР позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках.*

История

- *Первые ИСР были созданы для работы через консоль или терминал, которые сами по себе были новинкой: до того программы создавались на бумаге, вводились в машину с помощью предварительно подготовленных бумажных носителей (перфокарт, перфолент) и т. д.*
- *Dartmouth BASIC был первым языком, который был создан с ИСР, и был также первым, который был разработан для использования в консоли или терминале. Эта ИСР (часть Dartmouth Time Sharing System) управлялась при помощи команд, поэтому существенно отличалась от более поздних, управляемых с помощью меню и горячих клавиш, и тем более графических ИСР, распространённых в XXI веке. Однако она позволяла править исходный код, управлять файлами, компилировать, отлаживать и выполнять программы способом, принципиально подобным современным ИСР.*