МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Основы проектирования графического интерфейса компьютерных систем

Электронный учебно-методический комплекс
по дисциплине в LMS Moodle

Работа выполнена по мероприятию блока 1 «Совершенствование образовательной деятельности» Программы развития СГАУ на 2009 – 2018 годы по проекту «Разработка магистерской программы «Программное обеспечение мобильных устройств» по направлению 230100.68 – Информатика и вычислительная техника»
Соглашение № 1/12 от 3.06.2013 г.

САМАРА
2013

УДК 004.451
О 753

Автор-составитель: **Климентьев Константин Евгеньевич**

Основы проектирования графического интерфейса компьютерных систем

[Электронный ресурс] : электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle / Мин-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост. К.Е. Климентьев - Электрон. текстовые и граф. дан. - Самара, 2013. – 1 эл. опт. диск (CD-ROM).

В состав электронного учебно-методического комплекса входят:

1. ОПГИКС.Курс лекций.pdf
2. ОПГИКС.Методические указания и задания к лабораторным работам. pdf
3. ОПГИКС.Вопросы к зачету.pdf
4. Рабочая программа - Основы проектирования графического интерфейса компьютерных систем.pdf

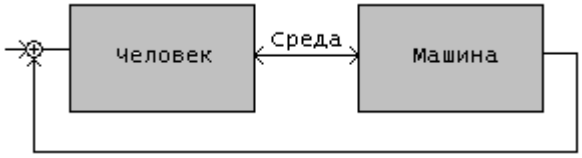
УМКД «Основы проектирования графического интерфейса компьютерных систем » предназначен для студентов факультета информатики, обучающихся по направлению подготовки магистров 230100.68 «Информатика и вычислительная техника».УМКД разработан на кафедре программных систем.

УМКД разработан на кафедре Информационных систем и технологий.

М2.В.ДВ.2.2. -
Основы проектирования графического интерфейса компьютерных систем
(курс лекций)

Составитель: к.т.н., доц. К.Е. Климентьев

Центральными понятиями являются термины «человеко-машинное взаимодействие» – Human-computer interaction (HCI) и «человеко-машинный интерфейс – Human-machine interface (HMI)».



Факторы:

- эргономичность HCI;
- окружение HCI (средства взаимодействия, гипермедиа и Web, средства связи);
- разработка и развитие систем, ориентированных на пользователя;
- модели пользователя (восприятие, моторика, мышление, взаимодействие, организация работы, адаптация к многообразию);
- принципы разработки удобных пользовательских HCI;
- критерии и проверка легкости использования.

Эргономика (от греч. érgon — работа и nómos — закон), научная дисциплина, комплексно изучающая человека (группу людей) в конкретных условиях его (их) деятельности в современном производстве.

Функциономика — раздел эргономики, исследующий алгоритмы действия человека-оператора в эргатических системах, включает:

- разработку тренажёров, моделирующих установок и испытательных стендов для определения эргономических показателей;
- исследование тренируемости и утомляемости для данного вида деятельности и условий восприятия информации;
- разработка методик функционирования человека-оператора в контуре управления;
- разработка методов контроля психо-физиологического состояния человека оператора.

Эргономическое обеспечение АРМ – создание благоприятных условий для использования АРМ при высокой производительности труда и низкой утомляемости пользователей.

Юзабилити (англ. usability — досл. «пользуемость», «возможность быть используемым») какой-то вещи — это степень её удобства в использовании.

Раздел эргономики, посвящённый разработке программного обеспечения. Международный стандарт ISO 9241-11: «степень, в которой продукт может быть использован определенными пользователями при определенном контексте использования для достижения определенных целей с должной эффективностью, продуктивностью и удовлетворенностью».

Юзабилити относительна. Примеры «хорошей» и «плохой» (в разных ситуациях) клавитатуры.

Интерфейс (interface) - это средства взаимодействия, средства связи, сопряжения, согласования.

Выделяют следующие виды интерфейса:

- аппаратный (физический) интерфейс - на уровне электронных компонентов;
- программный интерфейс - комплекс правил и соглашений о стыковке программных модулей;
- интерфейс пользователя - набор средств диалога, взаимодействия программы (машины) с человеком.

Интерфе́йс по́льзователя — совокупность средств, при помощи которых пользователь общается с различными устройствами, чаще всего — с компьютером или бытовой техникой, включает:

- средства отображения информации, отображаемую информацию, форматы и коды;
- командные режимы, язык «пользователь — интерфейс»;
- устройства и технологии ввода данных;
- диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;
- поддержку принятия решений в конкретной предметной области;
- порядок использования программы и документацию на неё.

Интерфейс графический – вид диалогового взаимодействия пользователя с ПК, при котором используются различные графические изображения (значки, пиктограммы, иконки, рисунки) объектов на экране (пример: OS Windows).

Интерфейс командный – вид диалогового взаимодействия пользователя с ПК, при котором используются различные команды, набираемые на клавиатуре и отображаемые на экране (пример: OS MSDOS).

Интерфейс командной строки — разновидность консольного интерфейса человека и компьютера, в котором инструкции компьютеру даются только путём ввода с клавиатуры текстовых строк (команд).

Интерфейс многооконный – вид диалогового взаимодействия пользователя с ПК, при котором каждой программе или данным отводится своя прямоугольная область (окно) на экране.

Любой интерфейс, независимо от сферы его применения, имеет пять основных характеристик:

- производительность пользователей
- количество человеческих ошибок
- скорость обучения работе с системой
- субъективное удовлетворение пользователей
- способность сохранения пользователями навыков работы с системой в течение длительного времени.

Физиологические основы HCI.

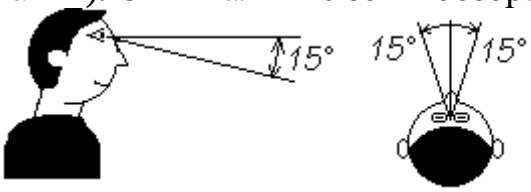
Важность адекватности интерфейса: пример с катастрофой Боинга-737 под Пермью в сентябре 1998 г. Прямая и обратная индикация на авиагоризонте – как правильно?

Для некоторых видов интерфейсов есть стандарты, например:

- ANSI/IEEE 1023-1988/2004 «Guide for the Application of Human Factors Engineering to Systems, Equipment and Facilities of Nuclear Power Generating Stations»;
- ГОСТ 21958-76 СЧМ. Зал и кабины операторов. Взаимное расположение рабочих мест. Общие эргономические требования;

- ГОСТ 22269-76 СЧМ. Рабочее место оператора. Взаимное расположение элементов рабочего места. Общие эргономические требования;
- ГОСТ 23000-78 СЧМ. Пульты управления. Общие эргономические требования;
- ГОСТ 21480-76 СЧМ. Мнемосхемы. Общие эргономические требования;
- ГОСТ 21829-76 СЧМ. Кодирование зрительной информации. Общие эргономические требования;
- ГОСТ 22902-78 СЧМ. Отсчетные устройства индикаторов визуальных. Общие эргономические требования;
- ГОСТ 21752-76 СЧМ. Маховики управления и штурвалы. Общие эргономические требования;
- ГОСТ 21753-76 СЧМ. Рычаги управления. Общие эргономические требования;
- ГОСТ 21786-76 СЧМ. Сигнализаторы звуковые неречевых сообщений. Общие эргономические требования;
- ГОСТ 22613-77 СЧМ. Выключатели и переключатели поворотные. Общие эргономические требования;
- ГОСТ 22614-77 СЧМ. Выключатели и переключатели клавишные и кнопочные. Общие эргономические требования;
- ГОСТ 22615-77 СЧМ. Выключатели и переключатели типа "Тумблер".

Общие эргономические требования. В соответствии с требованиями, изложенными в этих стандартах, можно сформировать ряд практических советов по построению человеко-машинных интерфейсов. Значительная часть этих требований уже учтена в инструментальных средах программирования систем автоматизации – кнопки, тумблеры, светодиоды, шкалы, условные обозначения объектов и прочие компоненты, из которых формируются мнемосхемы и лицевые панели, имеют характеристики, соответствующие требованиям стандарта, но все же окончательное слово в этом вопросе остается за проектировщиком. Элементы группируются либо по функциональному принципу (когда рядом размещаются элементы, совместно используемые при выполнении какой-либо задачи), либо по последовательному принципу (в порядке использования). Оптимальные зоны обзора:



При групповом размещении индикаторов необходимо выполнять следующие правила:

- при наличии в группе шести или более элементов располагать их в виде двух параллельных рядов (вертикальных или горизонтальных);
- не делать более 5-6 горизонтальных или вертикальных рядов;

- при наличии на экране более 25-30 одинаковых элементов компоновать их в 2-3 зрительно отличимые группы.

Мнемосхемы предназначаются для выполнения следующих функций:

- наглядного отображения функционально-технической схемы управляемого объекта и информации о его состоянии в объеме, необходимом для выполнения оператором возложенных на него функций;
- отображения связей и характера взаимодействия управляемого объекта с другими объектами и внешней средой;
- сигнализации обо всех существенных нарушениях в работе объекта;
- обеспечения быстрого выявления возможности локализации и ликвидации неисправностей.

Мнемосхема должна содержать только те элементы, которые необходимы оператору для контроля и управления объектом. Отдельные элементы и группы элементов, наиболее существенные для решения этой задачи, на мнемосхеме должны выделяться размерами, формой, цветом или другими способами. При компоновке мнемосхемы должно быть обеспечено пространственное соответствие между расположением элементов на мнемосхеме и расположением органов управления на пульте оператора. Допускается размещение на поле мнемосхемы приборов контроля и органов управления, которые при этом не должны закрывать от оператора другие элементы мнемосхемы. При компоновке мнемосхем должны учитываться привычные ассоциации оператора. Соединительные линии на мнемосхеме должны быть сплошными, простой конфигурации, минимальной длины и иметь наименьшее число пересечений. Следует избегать большого числа параллельных линий, расположенных рядом. Составными элементами мнемосхем являются мнемознаки – буквы, цифры, характерные условные обозначения, стилизованные изображения компонентов управляемого объекта и т.п. Комплекс мнемознаков, используемых на одной мнемосхеме, должен быть разработан как единый алфавит. Необходимо, чтобы алфавит мнемознаков был максимально коротким, а различительные признаки мнемознаков были четкими. Мнемознаки сходных по функциям объектов должны быть максимально унифицированы. Форма мнемознака должна соответствовать основным функциональным или технологическим признакам отображаемого объекта. Допускается брать за основу конструктивную форму объекта или его условное обозначение, принятое в технической документации. Размеры мнемознака должны обеспечивать оператору наиболее однозначное зрительное восприятие. Вспомогательные элементы и линии не должны пересекать контур мнемознака или каким-либо другим способом затруднять его восприятие. Яркостный контракт между мнемознаками и фоном мнемосхемы должен быть не менее 65%. Сигналы об изменениях состояния объекта (включен-отключен, открыт-закрыт) должны различаться особенно четко цветом, формой или другими признаками. Специальные сигналы (предупредительные, аварийные, неплановой смены состояния и т.п.) должны отличаться большей

интенсивностью (на 30-40%) по сравнению с сигналами нормального режима или быть прерывистыми (с частотой мигания 3-5 Гц и длительностью сигнала не менее 0.05 с). Допускается совместное применение обоих способов.

При проектировании и реализации человеко-машинных интерфейсов нередко возникает необходимость использовать для передачи информации оператору не только форму знаков, но и их цвет, взаимное расположение, движение отдельных элементов относительно друг друга и т.п.

Объекты и их характеристики делят на классификационные группировки, а для этого устанавливают признаки сходства и различия объектов, распределяют их по значимости и определяют основание деления. Вид алфавита кода выбирают с учетом характера передаваемой информации и задач, решаемых оператором, опираясь на системы знаний, закрепленных в опыте человека. В зависимости от характера и объема передаваемой информации устанавливают целесообразность использования одномерного или многомерного кода. Многомерным алфавитом кода считается алфавит, построенный путем объединения в одном знаке нескольких видов алфавита, например – формы, яркости, цвета и размера. Основание выбранного кода определяют исходя из количества кодируемых объектов и их характеристик согласно табл.

Вид кодирования	Количество кодируемых параметров
Размер	5
Пространственная ориентация	8
Длина линии	6
Ориентация линии	4
Количество точек	5
Яркость	4
Цвета	11
Частота мельканий	4

При передаче информации о нескольких признаках объекта используют многомерное кодирование. В структуре многомерного кода могут быть использованы сочетания различных видов алфавита: формы и цвета; формы, цвета и пространственной ориентации; размера, яркости и частоты мельканий.

Возможно использование самостоятельно сконструированных кодовых знаков. При этом основной классификационный признак объекта должен кодироваться контуром. Знак должен быть хорошо различим (иметь достаточный угловой размер и яркость) и представлять собой замкнутую фигуру. В алфавите должно быть установлено оптимальное количественное соотношение признаков знака и признаков объекта. В состав знака могут входить как основные, так и дополнительные детали. Дополнительные детали не должны пересекать или искажать контур знака (исключение могут составлять знаки, выражающие отмену или запрещение каких-либо действий, окончание их и т.п.). При конструировании знаков следует отдавать предпочтение внутренним деталям перед наружными. Детали кодовых знаков должны быть унифицированы. В качестве опознавательных признаков знаков в пределах одного алфавита нельзя использовать:

- число элементов в знаке (исключение могут составить знаки, обозначающие признак множественности без точной количественной характеристики, например, отображающие понятия «мало-много, «одиночный-групповой»);
- отличие знаков по признаку позитив-негатив;
- отличие знаков по признаку прямое-зеркальное отражение (за исключением случаев, когда это необходимо для отображения пространственной ориентации или направленности по принципу «вверх-вниз», «влево-вправо», «вперед-назад» и т.п.).

В алфавитах используют знаки симметричной формы с единообразием ориентации: контуры знаков должны быть по возможности ориентированы в соответствии с основными пространственными осями – горизонтали и вертикали.

Разрешено также использование цветовых алфавитов. В таком алфавите следует отдавать предпочтение зеленому, красному, голубому, желтому и фиолетовому цветам. Общее число используемых цветов может быть увеличено, если обозначения меняются не только по цветовому тону, но и по яркости. Знаки алфавита должны быть хорошо различимы при точном опознании цвета. Цветовой код применяют при освещении белым цветом, поскольку видимый цвет зависит от общего освещения. Допустимая яркость цветных знаков, кд/м^2: минимальная – 10; рекомендуемая – 170; для отраженного света – 30-70. Оптимальная угловая величина цветового знака 35-45’. Для знаков алфавита используют цвета в соответствии с табл.

Категория информации	Рекомендуемый цвет кода	
	Основной	Дополнительный
Предупреждающая информация, которая носит осведомительный характер, содержит сведения об общей обстановке (исключая аварийную) и рекомендации для принятия мер	Желтый	Белый
Предписывающая информация, которая носит командный характер, требует или разрешает выполнение строго определенных действий	Зеленый	Синий
Запрещающая информация, которая носит запретительный или ограничительный характер, указывает на неисправность или неготовность	Красный	Оранжевый

Для выделения особо важной информации внутри алфавита (например, для информации, требующей немедленного принятия решения) применяют дополнительный цвет. Для кодирования информации, содержащей сообщение о том, что произошло одно из двух («да» или «нет») равновероятных событий, могут быть использованы красный и синий цвета.

Низкоуровневые средства создания интерфейса.

Модель GDI (MS Windows).

В отличие от X Window, графический интерфейс Microsoft Windows создавался в качестве стандартного и единственно возможного для этой операционной системы, и наоборот - ее архитектура проектировалась в середине 1980-х годов специально для поддержки этого интерфейса. Близким аналогом графического интерфейса ранних версий MS Windows является Presentation Manager операционной системы OS/2.

Вплоть до версий 3.X MS Windows являлась внешней оболочкой, расширяющей возможности MS-DOS. Все современные версии Windows являются полноценными операционными системами. В настоящее время принято различать две «линии» развития MS Windows: системы для индивидуальных пользователей (Windows 95/98/ME) и «корпоративные» системы (Windows NT/2000/XP/2003). Несмотря на архитектурные различия этих классов операционных систем, их графические интерфейсы концептуально и функционально идентичны.

Объектная архитектура Windows. Несмотря на то, что в среде Windows могут функционировать программы, не использующие оконного ввода-вывода, тем не менее, они составляют незначительное меньшинство от всех Windows-программ. Типичное приложение Windows функционирует в рамках возможностей, предоставляемых графическим пользовательским интерфейсом.

Графический интерфейс Windows построен по объектной технологии. В операционной системе используется много различных объектов, каждый из которых принадлежит какому-то классу, но один из них играет ключевую роль; это – «окно». С точки зрения пользователя, окно – это прямоугольная область экрана, с которой ассоциирована некоторая Windows-программа. Одной программе может быть поставлено в соответствие несколько окон, но одно из них всегда считается «главным». На экране могут одновременно располагаться окна нескольких приложений, одно из них является «активным».

Рассмотрим принципы организации диалога в Windows. С каждым окном может быть связано некоторое количество компонентов диалогового взаимодействия – меню, кнопок, полей ввода и т.п. В каждый определенный момент времени актуальными являются диалоговые компоненты только «активного» окна. Весь клавиатурный ввод направляется в окно, обладающее «фокусом ввода». Напротив, мышь воздействует на диалоговые компоненты того окна, на котором располагается его курсор.

Все объекты Windows в процессе работы обмениваются сообщениями. В отличие от X Window, в Windows отсутствует разделение объектов на «клиенты» и «серверы». Обмениваться сообщениями могут даже отдельные части одного приложения, например, главная программа и ее окно. Обмен

сообщениями возможен как в синхронном, так и в асинхронном режиме. Сообщение – это совокупность:

- номера, идентифицирующего тип сообщения;
- набора данных, связанных с сообщением.

В Windows определены свыше тысячи стандартных сообщений, например: WM_CREATE посылается окну при его создании; WM_COMMAND посылается окну при операциях с каким-нибудь диалоговым элементом этого окна (например, с меню или кнопкой); WM_SETFOCUS посылается окну в тот момент, когда к нему переходит «фокус ввода»; BM_SETSTATE посылается органу управления (например, кнопке) для того, чтобы оно принудительно изменило свое состояние, и т.п.

Общая структура Windows-приложения. Типичное Windows-приложение начинает свою работу со следующих действий):

- описывает и регистрирует новый класс окна, а именно: 1) указывает параметры окна; 2) указывает ресурсы (меню, изображения, пиктограммы и т.п.), связанные с окном; 3) указывает процедуру, которая будет обрабатывать сообщения, поступающие в окно;
- отображает окно на экране (послав ему сообщение WM_CREATE);
- организует цикл приема и обработки сообщений, поступающих в приложение.

Также в приложении должны быть организованы процедуры, обрабатывающие сообщения, поступающие в окно и диалоговые органы управления. Адрес процедуры обработки оконных сообщений является одним из атрибутов объекта «окно». Этот адрес используется в качестве параметра при описании и регистрации нового класса окна.

Оконная процедура может получать множество различных сообщений. Большинство сообщений могут быть обработаны автоматически. Например, обычно не требуют самостоятельной обработки сообщения, свидетельствующие о закрытии, перемещении, свертывании окна. Но для Windows-программ, производящих вывод в окно, важно обрабатывать сообщение WM_PAINT с номером 0x000F. Это сообщение автоматически передается в окно в тот момент, когда требуется перерисовка изображения (например, если окно было передвинуто или часть его была перекрыта другим окном). Целесообразно производить весь вывод изображений в окно по приходу именно этого сообщения.

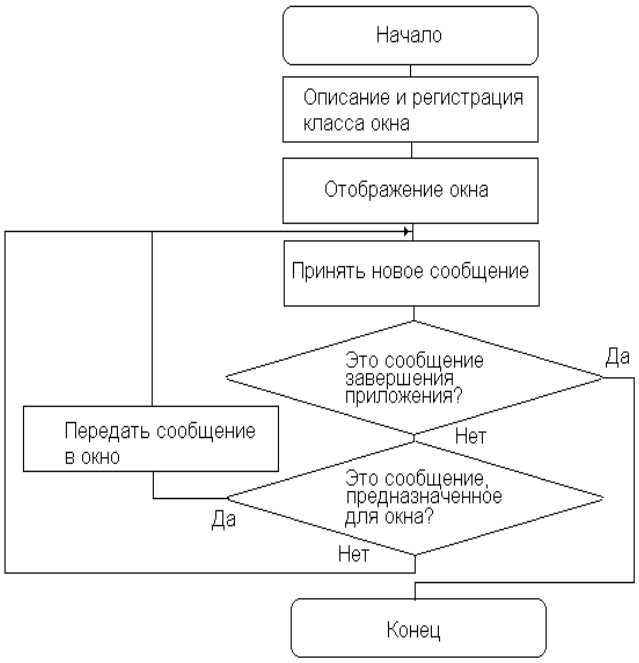


Рис. 6.5. Главная процедура Windows-приложения

Контекст графического устройства. С каждым физическим или виртуальным устройством, на которое Windows-программа может выводить изображение, связан контекст устройства. В качестве такого устройства могут выступать: окно, принтер, файл с изображением (метафайл), область памяти. Контекст устройства – это структура данных, однозначно описывающая параметры устройства. Работа программы, выводящей изображение на устройство с помощью контекста, аналогична работе с файлом:

- сначала контекст нужно создать или открыть (обычно, при помощи функции GetDC());
- затем идентификатор контекста передается в функции, выводящие изображение на устройства, в качестве параметра;
- после вывода изображения контекст надо закрыть (обычно, при помощи функции ReleaseDC()).

Графические примитивы Win API. Windows содержит стандартные динамические библиотеки, реализующие программный интерфейс для доступа к системе драйверов - Win API (Application Program Interface – интерфейс прикладного программирования). В ранних 16-разрядных версиях Windows существовало четкое разделение Win API на три части:

- KERNEL – отвечала за базовые средства операционной системы, включающие работу с файлами, с памятью, запуск и завершение процессов и т.п.;
- USER – отвечала за работу с клавиатурой, мышью, параллельным и последовательным портами и т.п.;
- GDI - отвечала за графический интерфейс.

В более современных 32-разрядных версиях структурное разделение исчезло, осталось только функциональное разделение API на три подсистемы.

GDI позволяет отображать следующие базовые графические элементы: пиксел, прямую и ломаные линии, кривые Безье, эллипс и его сектора, прямоугольник, прямоугольник со скругленными углами, произвольный многоугольник, текст. Линии могут быть непрерывными, либо иметь регулярную структуру (за эти характеристики отвечает свойство «перо»). Замкнутые фигуры могут быть закрашены каким-либо цветом, либо заполнены неким узором (за эти характеристики отвечает свойство «кисть»):

- `SetPixel(h, x, y, цвет)` – ставит точку указанным цветом (типа `COLORREF`) в указанной позиции;
- `COLORREF RGB(r, g, b)` – генерирует цвет из трех составляющих;
- `COLORREF GetPixel(h, x, y)` – возвращает цвет указанной точки;
- `LineTo(h, x, y)` – стороит линию от текущей позиции до указанной;
- `MoveToEx(h, x, y, &p)` – изменяет текущую позицию и возвращает параметры текущей точки в переменной `COLOR *p`;
- `long GetCurrentPosition()` – возвращает текущую позицию (x в младшем слове, y – в старшем слове);
- `Polygon(h, МассивВершин, КоличествоВершин)` – отображает многоугольник с вершинами, описанными в массиве типа `POINT;`

`TextOut(h, x, y, Строка, ДлинаСтроки)` – выводит строку текста.

Основной недостаток стандартных средств отображения графики в Win API – невысокая скорость вывода изображений на экран. Для ускорения этого процесса нередко построение изображения выполняют с использованием контекста оперативной памяти, а затем быстро копируют его в окно.

В общем, быстродействия базовых графических средств Win API вполне хватает для офисных приложений и прочих программ, не использующих динамичной смены изображений на экране.

Пример программы:

```
#include <windows.h>
#include <mem.h>

LRESULT CALLBACK MyProc
(HWND hWnd, UINT message, UINT wParam, LONG lParam){
  HDC h;
  switch(message) {
    case WM_DESTROY :
     PostQuitMessage (0);break;
    case WM_PAINT :
     h = GetDC(hWnd);
     /************** !!! ГРАФИКА ЗДЕСЬ  !!! *****************/
     ReleaseDC(hWnd, h); break;
    default :
     return (DefWindowProc(hWnd,message,wParam,lParam));
    }
   return 0L;
  }

int PASCAL WinMain
(HANDLE hIns, HANDLE hPrevIns, LPSTR lpCmdLine, int nCmdShow) {
  HWND hWnd;
  MSG m;
```

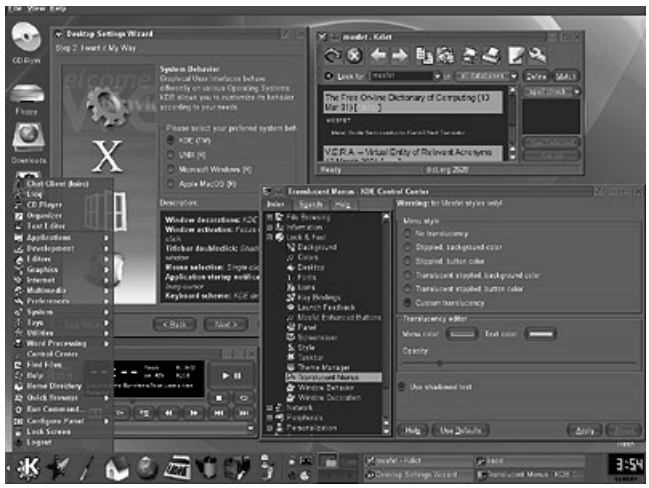
```
WNDCLASS wc;
BOOL result;

Setmem(wc, sizeof(wc), 0);
wc.style = CS_HREDRAW|CS_VREDRAW|WS_CLIPCHILDREN|WS_CLIPSIBLINGS;
wc.lpfnWndProc = MyProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = hIns;
wc.hIcon = LoadIcon(NULL,IDI_APPLICATION);
wc.hCursor = LoadCursor(NULL,IDC_ARROW) ;
wc.lpszMenuName = (LPSTR) NULL;
wc.lpszClassName = (LPSTR) "My";
wc.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);

result=RegisterClass(&wc);

hWnd=CreateWindow("My",
         "This is my simple example",
         WS_OVERLAPPEDWINDOW,
         CW_USEDEFAULT,
         CW_USEDEFAULT,
         CW_USEDEFAULT,
         CW_USEDEFAULT,
         NULL,NULL,
         hIns, NULL);

ShowWindow(hWnd,nCmdShow);
PostMessage(hWnd, WM_PAINT, 0, 0);

result = GetMessage(&m, NULL, NULL, NULL);
while (result)
 {
  TranslateMessage(&m);
  DispatchMessage(&m);
  result = GetMessage(&m, NULL, NULL, NULL);
 }

   return m.wParam; }
```

Модель METRO (MS Windows).

Metro— внутреннее кодовое название дизайнерского языка компании Microsoft, ориентированного на типографское оформление интерфейса пользователя. Изначально этот язык был разработан Майкрософт для использования в Windows Phone, также использовался при создании пользовательского интерфейса в Microsoft Encarta и MSN 2.0, а также Zune и Windows Media Center. Позже на основе Metro был построен интерфейс Windows Phone, веб-сайта Microsoft, Xbox 360 и Windows.

Стиль Metro основан на принципах дизайна швейцарского стиля. Основными принципами Metro являются акцент на хорошей типографике и крупный текст, который сразу бросается в глаза. Microsoft разработала Metro специально для укрепления группы общих задач для ускорения использования. Это достигается за счёт исключения лишней графики и вместо этого опоры на фактическое содержание, для функционирования в качестве основного пользовательского интерфейса. Большую роль играет анимация.

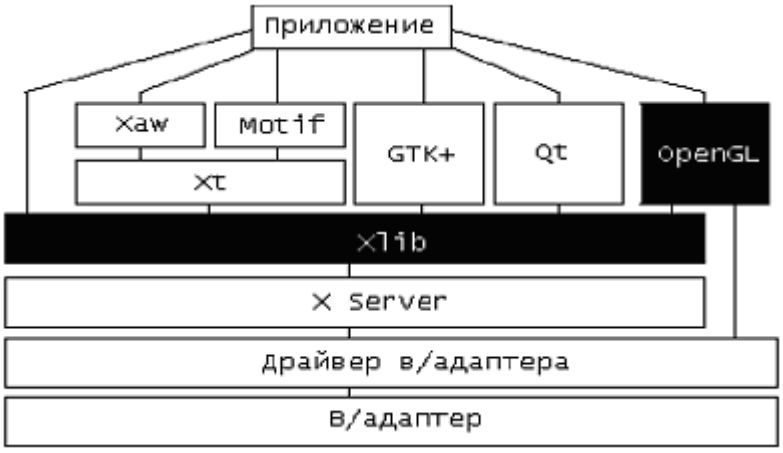
Модель X Window (UNIX).



X Window – это универсальный графический интерфейс, используемый в UNIX-подобных операционных системах (Linux, Free BSD, QNX, OS-9 и т.п.). X Window не является официальным стандартом для этих операционных систем, поскольку по умолчанию в качестве пользовательского интерфейса в них используется командная строка.

Тем не менее, если речь идет о графике в UNIX-подобных системах, то достойных альтернатив для X Window практически нет.

Первый вариант X Window был создан в 1984 году в MIT - Массачусетском технологическом институте (США).



X Window базируется на концепции «клиент-сервер». В роли «X-сервера» при этом выступает программное ядро графической системы, взаимодействующее с драйвером видеоконтроллера и способное предоставлять любым другим программным компонентам две абстрактных сущности: 1) образ видеопамяти; 2) информацию о системных событиях, связанных с клавиатурой и устройствами позиционирования. В роли «X-клиента» выступают любые

программные компоненты, в том числе и расположенные на удаленных компьютерах.

Процесс взаимодействия между «X-сервером» и «X-клиентом» осуществляется с помощью специального сетевого протокола - так называемого «X-протокола» и заключается в возбуждении событий. Протокол поддерживает всего 4 базовых типа событий: пришел запрос, получен ответ, получено оповещение о событии и получено сообщение об ошибке. Возбуждение событий происходит в асинхронном режиме.

Набор поддерживаемых X Window базовых операций вывода графики невелик - в него входит рисование пиксела, линии, текста, растрового изображения и ограниченного контуром или закрашенного многоугольника. Для программирования приложений, поддерживающих X Window, служит сервисная библиотека Xlib. Существуют более высокоуровневые библиотеки, обеспечивающие многообразные возможности по отображению графики (например, Motif). На основе X Window строятся популярные оконные оболочки KDE, GNOME и т.п.

Пример X Window.

```
// gcc myx.cpp -L/usr/X11R6/Lib -lX11
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

#include <unistd.h>
#define WIN_WIDTH   640
#define WIN_HEIGHT  480

unsigned long GetColor( Display* dis, char* color_name )  {    Colormap cmap;
     XColor near_color, true_color;
     cmap = DefaultColormap( dis, 0 );
 XAllocNamedColor( dis, cmap, color_name, &near_color, &true_color );
return( near_color.pixel );}

int main( void ) {
 Display* dis;
 Window win;
 XSetWindowAttributes att;
 GC gc;
 XEvent ev;
 int t;
 dis = XOpenDisplay( NULL );
 win = XCreateSimpleWindow( dis, RootWindow(dis,0), 100,
                 100,WIN_WIDTH, WIN_HEIGHT, 5,
                 WhitePixel(dis,0),  BlackPixel(dis,0) );
att.backing_store = WhenMapped;    XChangeWindowAttributes( dis, win,
CWBackingStore, &att );
XSelectInput( dis, win, ExposureMask );
XMapWindow( dis, win );
do { XNextEvent( dis, &ev); } while( ev.type != Expose );    gc = XCreateGC(
dis, DefaultRootWindow(dis), 0, 0 );    XSetFunction( dis, gc, GXxor );
XSetForeground( dis, gc, BlackPixel(dis,0)^GetColor( dis, "blue"));
XFillRectangle( dis, win, gc, 10, 10, 60, 60 );    XSetForeground(dis, gc,
BlackPixel(dis,0)^GetColor( dis, "red"));
XFillArc( dis, win, gc, 70, 70, 50, 50, 0, 360*64);        for (t=0;t<10;t++)
sleep(1);
```

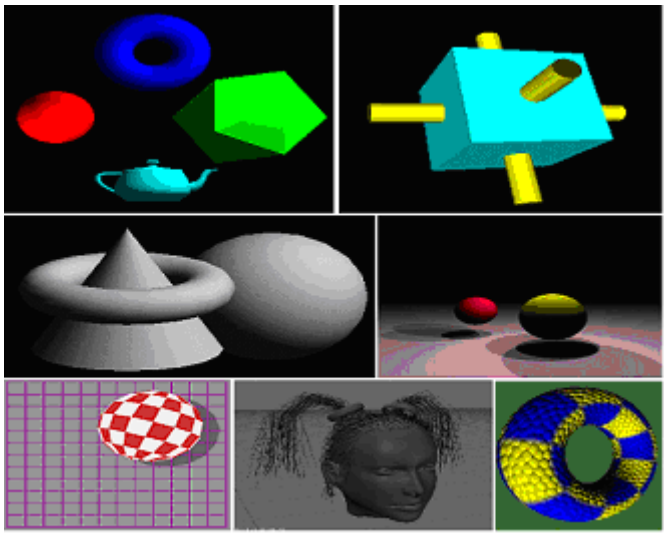
```
XDestroyWindow( dis , win );  XCloseDisplay( dis );   return(0);}
```

Высокоуровневые средства создания интерфейса.

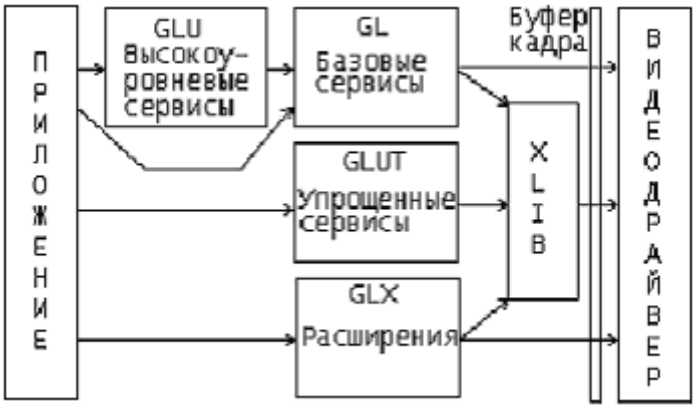
Межплатформенная среда OpenGL.

OpenGL - это оптимизированная, высокопроизводительная графическая библиотека. Стандарт OpenGL был утвержден в 1992 г. Он основан на библиотеке IRIS GL, разработанной компанией Silicon Graphics. OpenGL является переносимой библиотекой: существуют ее реализации для Windows и различных клонов Unix. Варианты этой библиотеки по умолчанию включены в такие среды программирования, как MS Visual Studio, Borland C/C++ Builder, Borland Delphi и пр. Интерфейс OpenGL реализован в виде набора процедур, к которым можно обращаться из прикладной программы. Процедуры OpenGL поддерживают полный комплект программных средств для формирования двух- и трехмерных изображений. В отличие от DirectDraw, библиотека OpenGL не использует никаких абстрактных концепций и оперирует классическими понятиями компьютерной графики: линия, поверхность, полигон, матрица преобразования, источник освещения, и т.п.

Примеры изображения OpenGL.



Структура библиотек OpenGL.



Для подключения библиотек OpenGL необходимо включить в текст программы строки:

```
#include <gl\gl.h>
#include <gl\glu.h>
#include <gl\glux.h>
```

Целесообразно оформить фрагмент рисования в виде трех процедур:

1. Инициализация OpenGL

```
MyInit(HWND hWnd)
{
 HDC h;
 HGLRC hgl;
 PIXELFORMATDESCRIPTOR p;
 int ip;

 memset(&p, 0, sizeof(PIXELFORMATDESCRIPTOR));
 p.nSize = sizeof(PIXELFORMATDESCRIPTOR);
 p.nVersion = 1; p.cColorBits = 24; p.cDepthBits = 32;
 p.dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL;
 p.iPixelType = PFD_TYPE_RGBA; p.iLayerType = PFD_MAIN_PLANE;
 h = GetDC(hWnd);
 ip =  ChoosePixelFormat(h, &p);
 SetPixelFormat(h, ip, &p);
 hgl = wglCreateContext(h);
 if (hgl) wglMakeCurrent(h, hgl);
}
```

2. Рисование средствами OpenGL

```
MyDraw(HWND hWnd)
{
}
```

3. Завершение работы с OpenGL

```
MyDisable(HWND hWnd)
{
 HGLRC hgl;
 HDC h;
```

```
 hgl = wglGetCurrentContext();
 if (hgl)
 {
  h = wglGetCurrentDC();
  wglMakeCurrent(NULL, NULL);
  ReleaseDC(hWnd, h);
  wglDeleteContext(hgl);
 }
}
```

Вызов этих процедур осуществляется в разных ветвях оконной процедуры:

```
LRESULT CALLBACK MyProc
                 (HWND hWnd, UINT message, UINT wParam, LONG lParam)
 {
  switch(message) {

    case WM_DESTROY :
         MyDisable(hWnd);
       PostQuitMessage (0);
         break;

    case WM_CREATE :
         MyInit(hWnd);
         break;

    case WM_PAINT :
        MyDraw(hWnd);
        break;

    default :
     return (DefWindowProc(hWnd,message,wParam,lParam));
    }
   return 0L;
  }
```

Простые фигуры. Изображение строится из ряда отдельных элементов, таких как линии, многоугольники и т.п., заданные множеством вершин. Вывод на экран одного элемента должен быть заключен между вызовами функций `glBegin(ТипЭлемента)` и `glEnd()`, а после вывода последней фигуры необходимо вызвать функцию `glFinish()`.

```
glBegin(. . .)
. . .
glEnd()
glBegin(. . .)
. . .
glEnd()
glFinish()
```

Допустимы следующие константы “ТипЭлемента” для вызова функции `glBegin()`:

- `GL_POINTS` – отдельные точки;
- `GL_LINES` – каждая пара вершин задает отрезок прямой;
- `GL_LINE_LOOP` – замкнутая линия;
- `GL_LINE_STRIP` – линия из нескольких связанных отрезков;
- `GL_TRIANGLES` – каждая тройка вершин образует треугольник;
- `GL_TRIANGLE_STRIP` – связанные треугольники с общей начальной вершиной;
- `GL_QUADS` – каждые четыре вершины образуют четырехугольник;
- `GL_QUAD_STRIP` – связанные четырехугольники; `GL_POLYGON` – один выпуклый многоугольник.

Координаты точек задаются функциями `glVertex2f(x,y)` – в двумерном случае; `glVertex3f(x,y,z)` – в трехмерном случае. Цвет элемента задается функциями `glColor2f(r, g, b)` и `glColor3f(r, g, b)`, причем значения интенсивности цвета задаются в долях от максимального значения в виде вещественных чисел в интервале от 0.0 до 1.0.

Пример фрагмента программы, рисующей цветные линии.

```
glLineWidth(10.0f);   // Толщина линий

glColor3f(1., 0, 0);  // Красный цвет
glBegin(GL_LINES);
glVertex2f(-0.5, 0);
glVertex2f(0.5, 0);
glEnd();

glColor3f(0, 0, 1.0); // Синий цвет
glBegin(GL_LINES);
glVertex2f(0, -0.5);
glVertex2f(0, 0.5);
glEnd();

glColor3f(0, 1.0, 0);  // Зеленый цвет
glBegin(GL_LINES);
glVertex2f(-0.5, -0.5);
glVertex2f(0.5, 0.5);
glEnd();

glColor3f(1.0, 1.0, 0); // Желтый цвет
glBegin(GL_LINES);
glVertex2f(-0.5, 0.5);
glVertex2f(0.5, -0.5);
glEnd();
```

Объемные фигуры. Объемная фигура идентифицируется переменной типа `GLUquadricObj`. Операторы построения изображения заключаются между функциями `gluNewQuadric()` и `gluDeleteQuadric()`.

```
GLUquadricObj *q;
q = gluNewQuadric();
. . .
gluDeleteQuadric(q);
```

Примеры функций для построения объемных фигур:

- `gluCylinder( GluquadricObj *q, int Радиус1, int Радиус2, int Высота, int Вдоль, int Поперек)` – рисует цилиндр (с разными радиусами торцов);
- `gluSphere(GluquadricObj *q, int Радиус, int Вдоль, int Поперек)` – рисует сферу;
- `gluDisk(GluquadricObj *q, int Радиус1, int Радиус2, int Вдоль, int Поперек)` – рисует диск или плоское кольцо с внешним и внутренним радиусами.

Фигуры изображаются в виде множества плоских граней, количество которых задается параметрами “Вдоль” и “Поперек”. Существует несколько стилей изображения объемной фигуры, например `GLU_FILL` – в виде закрашенных поверхностей (по умолчанию); `GLU_LINE` – в виде сетки; `GL_POINT` – в виде множества вершин. Этот режим можно изменить при помощи функции `qluQuadricStyle (GluquadricObj *q, int Стиль).`

По умолчанию все объемные фигуры строятся в точке с координатами (0,0,0) плашмя, поэтому для их перемещения и разворота используются преобразования координат.

Преобразования координат. Преобразования координат задаются в виде матриц однородных координат размером 4x4. Эти матрицы можно загрузить из массива вещественных чисел:

```
GLfloat m[16] = {. . .};
GlLoadMatrixf(m);
```

Кроме того, существуют функции для создания стандартных матриц:

- `glLoadIdentity()` – матрица тождественного преобразования;
- `glRotated(alfa, x, y, z)` – матрица поворота относительно вектора (x,y,z) на угол alfa (в градусах);
- `glScaled(x,y,z)` – масштабирование по осям;
- `glTranslated(x,y,z)` – сдвиг по осям.

Возможны три режима преобразований:

- `GL_MODELVIEW` – мировых координат в координаты проекции;
- `GL_PROJECTION` – координат проекции в экранные координаты;
- `GL_TEXTURE` - для матриц текстуры.

Текущий режим преобразования задается функцией `glMatrixMode(Режим)`. По умолчанию выполняется преобразование мировых координат в координаты проекции, т.е., фактически, трансформация в пространстве точки начала координат. Вид используемого преобразования задается функциями `glOrtho()` – аксонометрическая (по умолчанию); `gluPerspective()` – центральная. Функция glPushMatrix() позволяет сохранить в стеке позицию начала координат, а glPopMatrix() – восстановить ее из стека.

Пример фрагмента программы, выполняющей сдвиг изображения влево и вверх относительно центра окна и разворот в пространстве.

```
gluPerspective(50, 0.5, 1, 40);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslated(-0.2, 0.3, 0);  // Сдвиг по X и У
glRotated(-20, 1, 0, 0);     // Поворот 1
glRotated(-6, 0, 1, 0);      // Поворот 2
```

Модели освещения. В простейшем случае необходимо задать как минимум координаты 0-го (вообще их может быть несколько) источника света в виде массива из 4-х элементов:

```
GLfloat lp[4]={. . ., . . .,. . .,1};   // Массив координат лампочки
glLightfv(GL_LIGHT0, GL_POSITION, lp);
```

и поочередно разрешить различные режимы работы

```
glEnable(GL_LIGHT0);         // Разрешение освещения от 0-й лампочки
glEnable(GL_LIGHTING);       // Разрешить работу лампочки
glEnable(GL_COLOR_MATERIAL); // Разрешить цветное освещение
```

Пример фрагмента программы, рисующей фиолетовое бездонное ведро, освещенное слева-сверху.

```
GLUquadricObj *q;

q = gluNewQuadric();
glColor3f(1, 0., 1);
gluCylinder( q, 0.1, 0.2, 0.7, 12, 4);
gluDeleteQuadric(q);
```

Среда MS DirectX (Direct Draw и Direct3D)

Первая попытка фирмы Microsoft ускорить диалоговое взаимодействие пользователя с Windows была реализована в виде библиотеки WinG, ориентированной в основном на Windows 3.X. В 1995 году фирмой Microsoft была разработана (точнее, лицензирована у небольшой фирмы RenderMorfics) подсистема Reality Lab, которая послужила основой для продукта, известного сейчас под названием DirectX.

DirectX – это система библиотек, взаимодействующих с низкоуровневыми драйверами устройств ввода-вывода, а в отдельных случаях и самостоятельно

выполняющих роль монолитных драйверов. Благодаря этому обеспечивается высокая скорость доступа к аппаратным ресурсам.

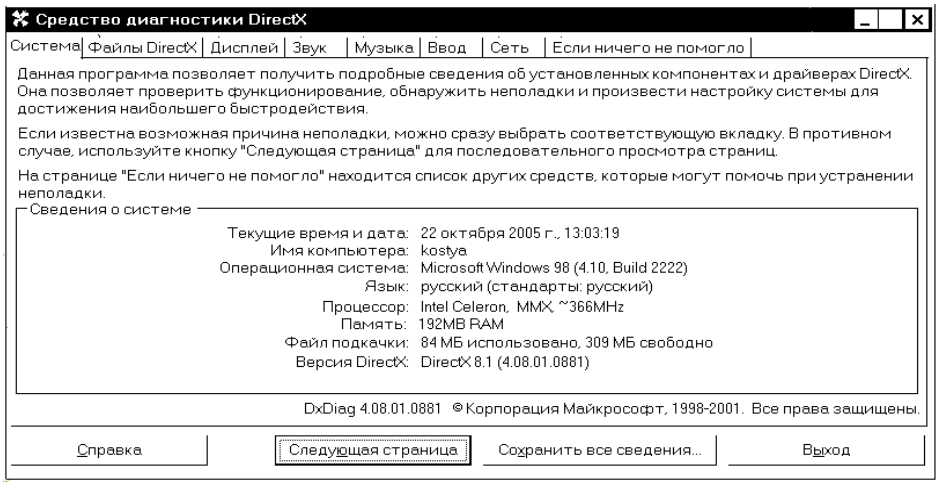
Составными частями DirectX являются:

- DirectDraw - отвечает за быстрый вывод, перемещение и масштабирование двухмерной графики, установку видеорежимов, взаимодействие с драйверами;
- DirectSound - отвечает за вывод и микширование звука;
- DirectInput - отвечает за ввод данных в систему в режиме реального времени через клавиатуру, мышь и джойстик;
- DirectPlay - независимый протокол для осуществления быстрой сетевой связи между компьютерами.
- Direct3D - отвечает за создание трехмерных графических изображений в режиме реального времени, базируется на DirectDraw.

DirectX постоянно расширяется и обновляется, регулярно выходят новые версии продукта. Это связано с прогрессом в области аппаратной обработки изображений (см. п. 7.1). Но все более поздние версии DirectX, как правило, включают в себя возможности, присутствовавшие в более ранних версиях.

DirectX – бесплатный продукт. Нередко он распространяется совместно с приложениями, использующими его возможности (например, с компьютерными играми). Некоторые версии DirectX включены в дистрибутив MS Windows по умолчанию, так, например, DirectX v3.0 является составной частью Windows 95 OSR2; DirectX v.6.0 включена в Windows 98; DirectX v7.1 интегрирована в Windows 98 SE, Windows ME и Windows 2000; DirectX v8.0 входит в состав Windows XP. Более новые и совершенные версии DirectX включаются в пакеты обновлений Windows - в так называемые «сервиспаки» (от англ. servicepack – сервисный набор). Узнать текущую версию DirectX, установленную на вашем компьютере, можно при помощи утилиты DxDiag, входящей в дистрибутив любой версии этой библиотеки.

Последнюю версию DirectX можно бесплатно скачать с сайта фирмы Microsoft (http://www.microsoft.com/directx). Оттуда же можно взять комплект средств разработки, так называемый DirectX SDK, ориентированный на Microsoft Visual C/C++. Однако, существуют варианты DirectX SDK (не самых последних версий), адаптированные энтузиастами для Borland C/C++ Builder и Borland Delphi.



Библиотека DirtectDraw обеспечивает прямой доступ к видеопамяти. Эта библиотека построена по правилам компонентной идеологии фирмы Microsoft (COM – Component Object Model), что облегчает ее взаимодействие с любыми другими продуктами Microsoft, но затрудняет программирование. DirectDraw способна выполнять операции трех типов:

- создание, уничтожение, модификацию, перемещение и копирование так называемых поверхностей – фрагментов видеопамяти;
- отсечение фрагментов поверхностей;
- чтение, модификацию и загрузку палитр.

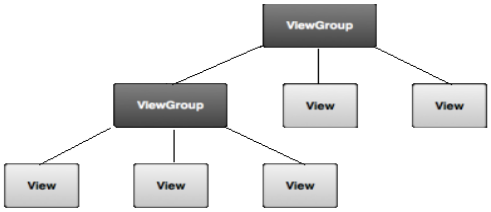
Обычно все базовые графические операции выполняются средствами процедур Win API, которым DirectDraw предоставляет контекст видеопамяти.

Библиотека Direct3D обеспечивает весь спектр высокоуровневых операций по созданию и преобразованию трехмерных полигональных изображений: проецирование, трансформации в пространстве, наложение текстур, удаление невидимых фрагментов, расчет освещения и т.п. При этом она активно пользуется аппаратными возможностями современных 3D-акселераторов.

Модель Google Android

Графический пользовательский интерфейс для Android приложения построен с использованием иерархии View и ViewGroup объектов. View объекты, как правило - это UI виджеты, такие как кнопки или текстовые поля и ViewGroup, а также невидимые контейнеры, которые определяют, как будут использовань child-элементы, например, в сетке или вертикальном списке.

Android предоставляет XML-словарь, который соответствует подклассам View и ViewGroup, поэтому вы можете определить свой пользовательский интерфейс в XML, используя иерархию элементов пользовательского интерфейса.



Пример интерфейса:

Модель Apple iOS

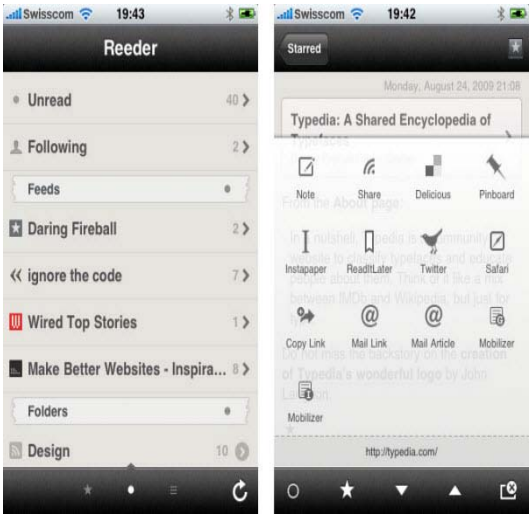
Apple на протяжении многих лет является ведущей компанией в мобильном мире с его iPhone и IPad. Они держат большую часть рынка мобильных платформ, и это является причиной того, что большинство клиентов хотят, чтобы их приложения были представлены в App Store от Apple, а это побуждает разработчиков учиться создавать iPhone-приложения.

Ниже — краткий перечень различных элементов пользовательского меню:

- Строка состояния (Status Bar) — отображает текущий уровень зарядки, 3G-связь, прием и многое другое. Рекомендуется всегда включать эти элементы.
- Панель навигации (Navigation Bar) — Дает пользователям возможность перемещаться между страницами. Часто сюда включают кнопки на левой стороне меню, чтобы пользователь мог вернуться к предыдущей странице.
- Панель инструментов (Toolbar) — появляется в нижней части iPhone приложений. Здесь будет несколько иконок, связанных с некоторыми функциями, такими как поделиться, загрузить, удалить и т.д.
- Меню вкладок (Tab Bar) — Очень похоже на панель инструментов, только теперь вы работаете с вкладками. Когда пользователь

щелкает по иконке вкладок, она будет автоматически выделена, и она будет подсвечиваться. Эта меню используется для переключения между окнами.

Разработка выполняется в среде Xbox на языке ObjC. Пример интерфейса:



ГОУВПО «Самарский аэрокосмический университет им. Акад. С.П. Королева»

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО КУРСУ «Основы проектирования графического интерфейса компьютерных систем»

Составитель: к.т.н., доц. К.Е. Климентьев

Самара 2013

Введение

Портирование (перенос) приложений из одной среды в другую существенно облегчается, если при создании приложений были использованы стандартные спецификации. Возможны следующие виды переноса:

- На уровне двоичного кода – благодаря использованию общей аппаратной платформы (например, x86) и стандартизованных библиотек-переходников (например, Win32 API).

- На уровне исходных тестов – благодаря использованию стандартизованных языков программирования (например, C или Java) и виртуальных машин или спецификаций (например, POSIX или JVM).

Однако, аппаратные различия и уникальные требования привели к созданию нескольких, слабо связанных Java-спецификаций:

- Java SE — Java Standard Edition, основное издание Java, содержит компиляторы, API, Java Runtime Environment; подходит для создания пользовательских приложений, в первую очередь — для настольных систем.
- Java EE — Java Enterprise Edition, представляет собой набор спецификаций для создания программного обеспечения уровня предприятия.
- Java ME — Java Micro Edition, создана для использования в устройствах, ограниченных по вычислительной мощности, например в мобильных телефонах, КПК, встроенных системах;
- JavaFX — технология, являющаяся следующим шагом в эволюции Java как Rich Client Platform; предназначена для создания графических интерфейсов корпоративных приложений и бизнеса.
- Java Card — технология предоставляет безопасную среду для приложений, работающих на смарт-картах и других устройствах с очень ограниченным объёмом памяти и возможностями обработки.

Преодолению данных несоответствий и посвящен настоящий курс лабораторных работ.

Термины:

Портирование - перенос программного обеспечения из одной системы в другую, результат этого действия называется порт. При портировании не обязательно использовать открытые исходные коды т.к. при портировании обычно используется враппер
Враппер (Wrapper) - программа которая преобразует машинный код для вызова функций API одной системы в другую.
Форк - создание новой программы базирующийся на старой для другой платформы.
Эмуляция - создание средств исполнения программы одной операционной системы в другой.
Интерпретация - исполнение команд средствами не входящими саму в программу.

Лабораторная работа 1. Портирование десктопного приложения под IPad.

Суть работы: провести перекомпиляцию приложения, написанного для Mac OS X, для работы на Ipad. Предварительно разработать и реализовать пользовательский интерфейс, максимально соответствующий оригинальному.
Инструменты:

- Среда Xbox;
- Компилятор ObjC.

Вариант 1. Программа «арифметический калькулятор».
Вариант 2. Программа «перекодировщик символов».
Вариант 3. Программа «расчет контрольных сумм и хеш-функций».
Вариант 4. Программа «шифрование/расшифрование данных методом RC4».

Лабораторная работа 2. Мобильный клиент для веб-сайта.

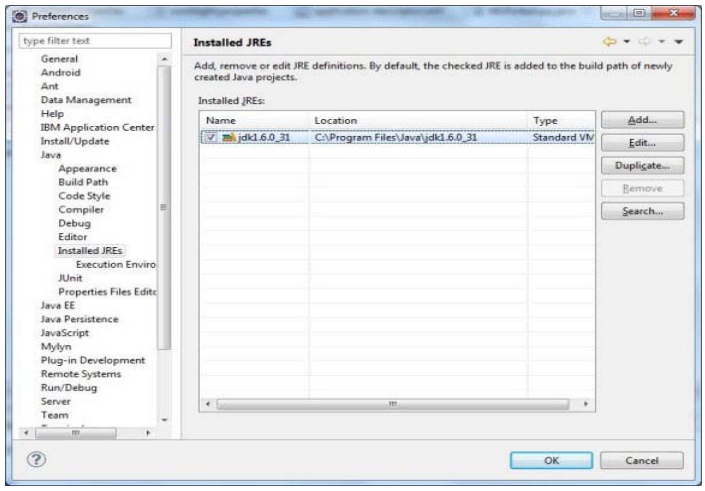
Суть работы: разработка мобильного клиента для контекстно-ориентированного считывания данных с требуемого Интетнет-сайта.

Чтобы построить описанный здесь пример гибридного мобильного приложения, понадобятся следующие правильно установленные и работающие программные продукты и компоненты.

- WebSphere Portal 8.0
- Eclipse
- Android SDK

После того как среда Eclipse установлена, нужно добавить Android SDK. Предварительным условием для работы Android SDK является наличие

Oracle Java JRE. Установив ее, настройте свой экземпляр Eclipse на использование JRE. Для этого выберите из меню Window > Preferences > Java > Installed JREs (рисунок 1).



- Worklight - Для этого примера требуется IBM Worklight 5.0 или более поздняя версия; с предшествующими версиями он работать не будет. Для целей разработки IBM Worklight Developer Edition 5.0 можно загрузить бесплатно. Подробности о том, как добавить в Eclipse плагин, содержатся в модулях по настройке (Setup) документации Начало работы с IBM Worklight.

Worklight устанавливает сервер в рамках среды Eclipse. Может потребоваться отредактировать файл eclipse.ini, заменив порт 8080 другим значением, например, 8085. При запуске Eclipse сервер Worklight запускается автоматически. В листинге 1 приведен пример содержания файла eclipse.ini с настройкой -Dworklight_port.

Листинг 1. eclipse.ini-startup

```
plugins/org.eclipse.equinox.launcher_1.2.0.v20110502.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.100.v20110502
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
--launcher.defaultAction
openFile
-vmargs
-Dworklight.port=8085
-Xms40m
-Xmx384m
```

- Клиент WebDAV

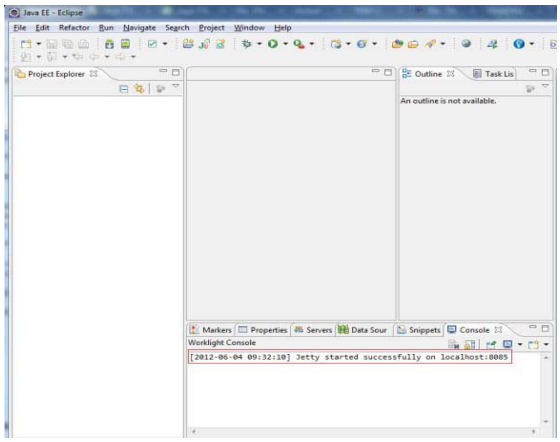
Чтобы изменить тему WebSphere Portal с помощью Worklight JavaScript, нужен клиент WebDAV. Этот пример разработан с помощью AnyClient. Вы можете использовать любой клиент WebDAV, который поддерживает

WebSphere Portal. Подробнее см. в документации Соединение с Portal WebDAV в версии 8.0.

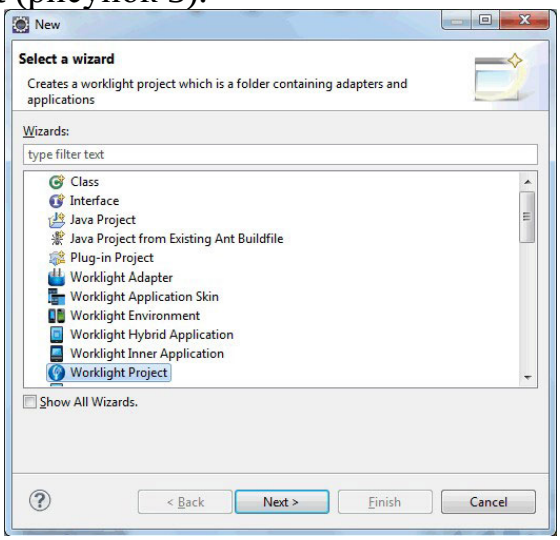
- Специальная тема

Прежде чем применять Worklight JavaScript со своей темой, нужно создать специальную тему. Скопируйте тему WebSphere Portal, чтобы ваша тема содержала все необходимые элементы, и fixpack не отменил внесенные изменения. Не изменяйте тему WebSphere Portal напрямую, потому что служба fixpacks может восстановить ее. Вместо этого следуйте инструкциям по созданию копии темы из из этой статьи.

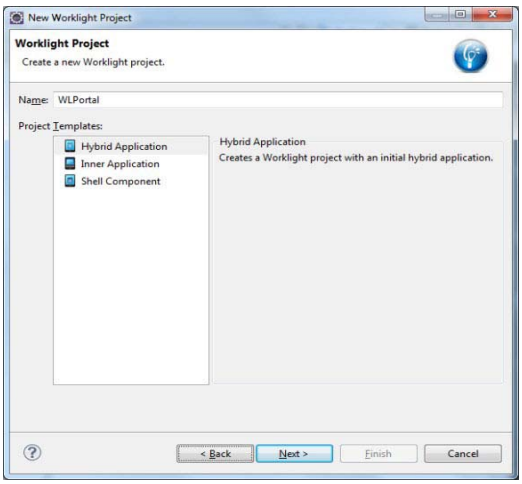
Создание приложения Worklight



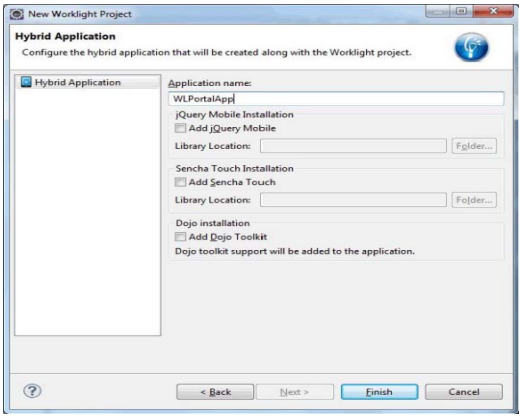
Чтобы приступить к созданию приложения, необходимо завести новый проект Worklight. В Eclipse выберите New > Other > Worklight Project и нажмите кнопку Next (рисунок 3).



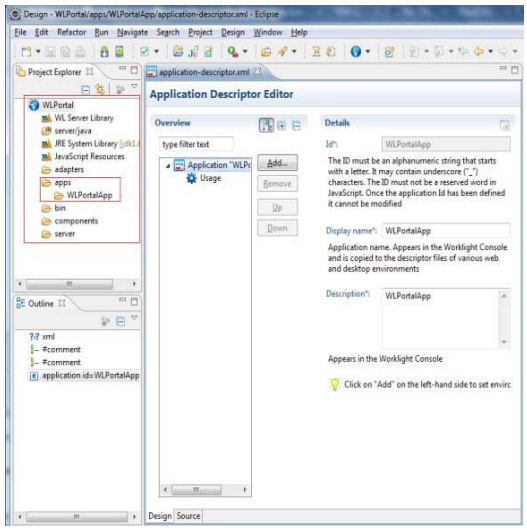
Присвойте проекту имя. На рисунке 4 проект называется WLPortal. Оставьте выбранный по умолчанию режим Hybrid Application и нажмите кнопку Next.



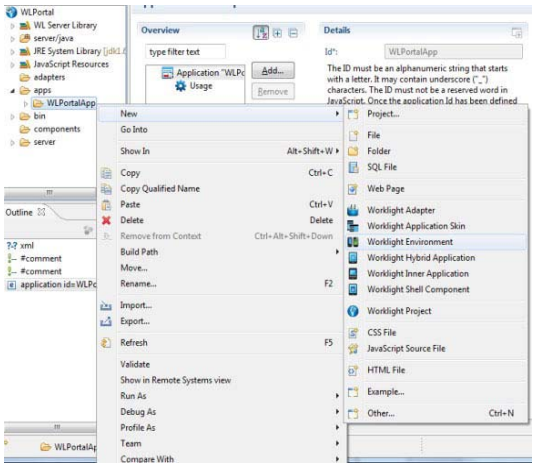
Присвойте имя гибридному приложению в проекте Worklight, в данном случае WLPortalApp. В этом примере нет пакета JavaScript, но его можно добавить на этой панели (рисунок 5).



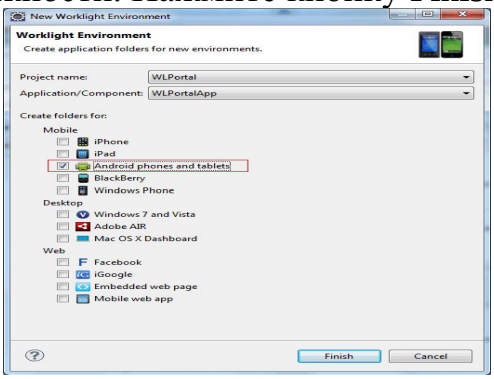
Нажмите кнопку Finish. Создаются артефакты проекта и приложения. Это может занять несколько минут. Затем вас могут попросить перейти в перспективу Design и просмотреть проект в окне Project Explorer (рисунок 6).



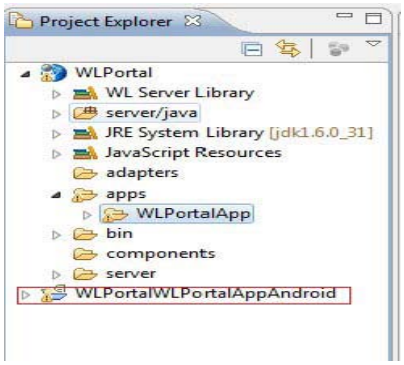
Далее, нужно создать среду Worklight для написания гибридного приложения. Щелкните правой кнопкой на WLPortalApp в папке приложения и выберите New > Worklight Environment (рисунок 7).



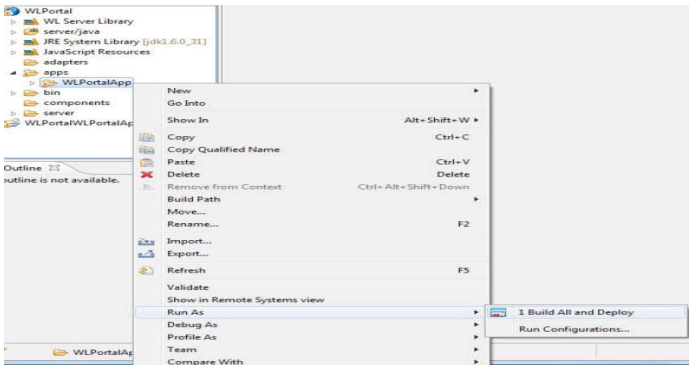
Появится панель New Worklight Environment (рисунок 8). Так как этот конкретный пример предназначен только для гибридного Android-приложения, выберите Android phones and tablets. Если необходимо создать среду для других операционных систем, здесь можно указать эти дополнительные возможности. Нажмите кнопку Finish.



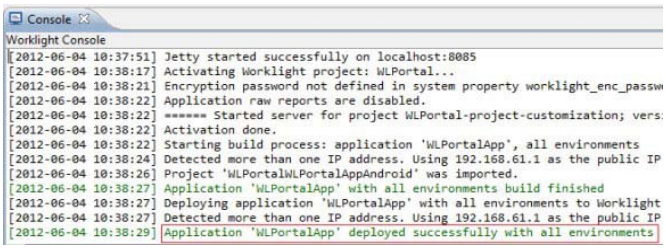
Worklight Studio создает и обновляет проект с "родным" приложением (рисунок 9). Имя "родного" проекта: Worklight Project/ WorklightApplication/Platform. Worklight Studio управляет жизненным циклом этого приложения, и вам не нужно вносить в него никаких изменений. Когда Web-приложение, первоначально созданное в проекте, готово и развернуто, "родное" приложение будет переписано с внесением всех изменений.



Теперь, когда приложение создано, его нужно скомпоновать и развернуть. Как видно на рисунке 10, команда Build All and Deploy означает, что "родное" приложение будет переустановлено с учетом изменений в Web-приложении. Команду Build All and Deploy можно выполнить, щелкнув правой кнопкой мыши на приложении и выбрав Run As > Build All and Deploy.

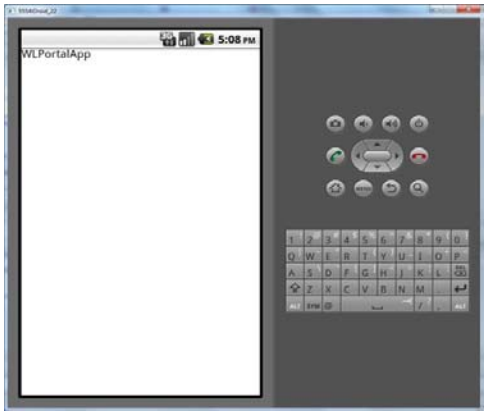


Когда начнется процесс сборки, за его ходом можно следить по индикатору на нижней правой панели состояния Eclipse. После завершения процесса на консоли Worklight должно появиться сообщение Application 'YourApp' deployed successfully with all environments (Приложение YourApp успешно развернуто во всех средах — рисунок 11).



Чтобы запустить Android-приложение, щелкните правой кнопкой мыши на проекте WLPortalWLPortalAppAndroid в обозревателе проектов и выберите Run As > Android Application. Это приведет к запуску эмулятора Android (если он еще не запущен) и загрузке экземпляра нового приложения. Возможно, чтобы увидеть приложение, вам придется открыть

первоначальный экран эмулятора. На рисунке 12 показано представление построенного приложения.



Аналогичным образом создаются и другие приложения. Задания на лабораторную работу.

Вариант 1. Мобильный клиент для просмотра форума на сайте www.ssau.ru
Вариант 2. Мобильный клиент для считывания новостей с www.mail.ru
Вариант 3. Мобильный клиент для считывания новостей с www.rambler.ru
Вариант 4. Мобильный клиент для просмотра курсов на virtual6.ssau.ru

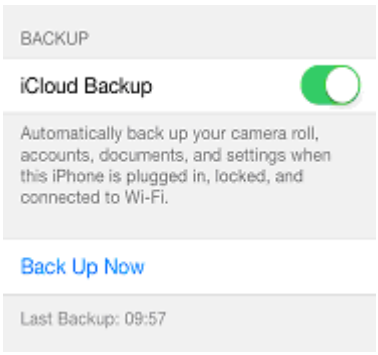
Лабораторные работы 3 и 4. «Портирование приложения с iPad на iPhone» и «Портирование iPhone приложения под Android.»

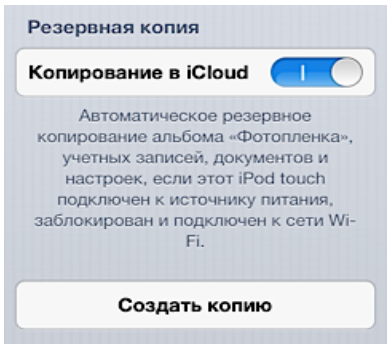
Суть работы: портирование приложений между разными программно-аппаратными платформами. При этом может потребоваться разработка нового интерфейса.

Выполните описанную ниже процедуру для iCloud, если на обоих устройствах установлена ОС iOS 5 или более поздней версии и у вас есть учетная запись iCloud. Если на одном или обоих устройствах используется более ранняя версия ОС iOS или у вас нет учетной записи iCloud, выполните описанную ниже процедуру для программы iTunes.
Использование службы iCloud для переноса информации с текущего устройства iOS на новое

Выполните следующие действия, если на обоих устройствах установлена ОС iOS 5 или более поздней версии и у вас есть учетная запись iCloud. Или выполните описанные ниже действия для программы iTunes.
Создайте резервную копию данных старого устройства в iCloud, открыв меню Настройки > iCloud > Хранилище и копии и включив параметр «Копирование в iCloud». Перед выполнением резервного копирования убедитесь, что ваше устройство подключено к сети Wi-Fi и источнику питания.

Примечание. Продолжительность резервного копирования и восстановления данных из резервной копии зависит от количества данных, содержащихся в резервной копии, и скорости подключения к Интернету.





iCloud создает резервную копию устройства один раз в сутки. Если необходимо создать резервную копию данных устройства перед миграцией, нажмите «Создать копию» на экране Настройки > iCloud > Хранилище и копии для просмотра состояния резервной копии. Чтобы гарантировать, что резервная копия содержит все данные, не начинайте миграцию до тех пор, пока резервное копирование не завершено. Дополнительную информацию о резервных копиях в iCloud см. в статье iCloud: обзор резервного копирования и восстановления.iOS 7 ОС iOS 6 и более ранних версий

По завершении резервного копирования данных старого устройства включите новое устройство и выполните действия в ассистенте настройки, выбрав язык и страну, и укажите, необходимо ли включить службы геолокации. При появлении запроса выберите сеть Wi-Fi.
При появлении приглашения к настройке выберите «Восстановить из копии iCloud», нажмите кнопку «Далее» и введите Apple ID и пароль.

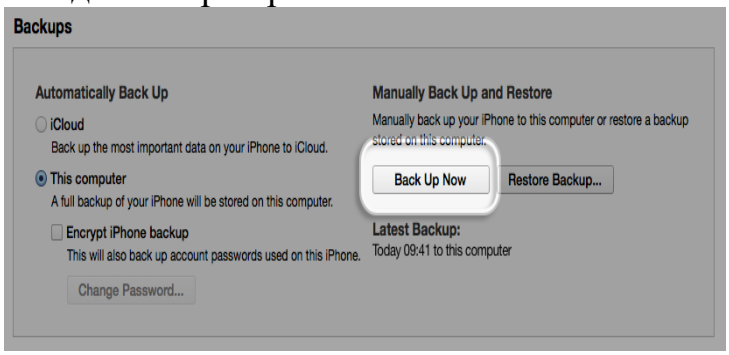
Выберите резервную копию старого устройства и нажмите «Восстановить». Дождитесь завершения восстановления данных из резервной копии.
Примечание. Продолжительность резервного копирования и восстановления данных из резервной копии зависит от количества данных, содержащихся в резервной копии, и скорости подключения к Интернету.

По завершении процесса восстановления устройство будет перезагружено. После этого оно будет готово к использованию. Для синхронизации данных, которые не содержатся в резервной копии (например, музыки, видеозаписей, приложений и т. д.), следует подключить устройство к программе iTunes. Проверить и изменить синхронизируемые типы данных можно на вкладках синхронизации в программе iTunes.

Примечание. При восстановлении резервной копии iCloud на новом устройстве потребуется еще раз ввести пароль для идентификатора Apple ID, учетной записи электронной почты и других учетных записей и автоответчика. Если вы забыли пароль для автоответчика, обратитесь к оператору.

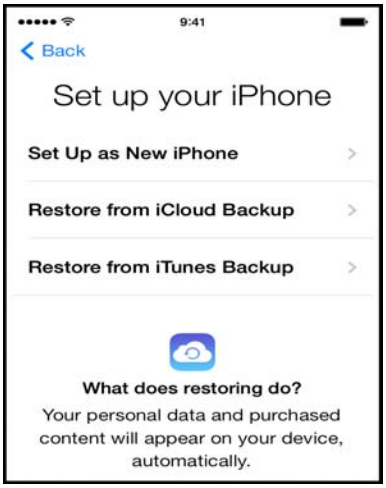
Использование программы iTunes для переноса информации с текущего устройства iOS на новое

Выполните следующие действия, если одно или оба ваших устройства используют операционную систему более ранней версии, чем iOS 5, или если у вас нет учетной записи iCloud.
Перед продолжением проверьте, что установлена последняя версия iTunes и последняя версия ОС iOS.
Создайте резервную копию данных старого устройства с помощью программы iTunes и перенесите данные с устройства на компьютер. См. дополнительные сведения о резервных копиях iTunes.

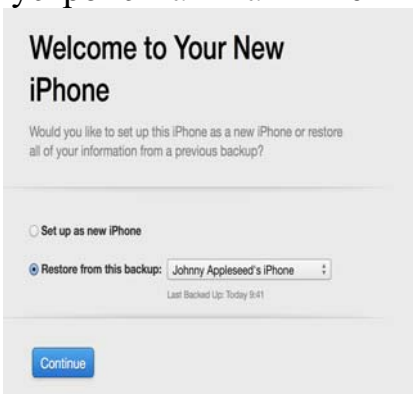


Если в устройство вставлена SIM-карта: с помощью инструмента для извлечения SIM-карты или скрепки извлеките SIM-карту из старого устройства и вставьте ее в новое. Если оператор предоставил вам новую SIM-карту, используйте ее.

Включите новое устройство и выполните действия в ассистенте настройки, выбрав язык и страну, и укажите, необходимо ли включить службы геолокации. При появлении запроса выберите сеть Wi-Fi.
Примечание. Если у вас возможности подключиться к Wi-Fi или к сотовой сети, вы также можете активировать услугу, нажав «Подключиться к iTunes».
При появлении запроса настроить устройство выберите «Восстановить из копии iTunes».



Подключите новое устройство к iTunes (на компьютере, который использовался для создания резервной копии данных, хранящихся на старом устройстве iPhone). В iTunes будет предложено выполнить восстановление из резервной копии или настроить устройство в качестве нового. Выберите резервную копию старого устройства и нажмите «Продолжить».



Примечание. В процессе восстановления старой резервной копии на новом устройстве iPhone можно заметить, что индикатор хода выполнения iTunes приостанавливается. В этом случае дождитесь завершения восстановления.
По завершении процесса восстановления устройство будет перезагружено. После этого оно будет готово к использованию. Для синхронизации данных, которые не содержатся в резервной копии (например, музыки, видеозаписей, приложений и т. д.), устройство должно быть подключено к iTunes. Проверить и изменить синхронизируемые типы данных можно на вкладках синхронизации в программе iTunes.
Примечание. После восстановления из незашифрованной резервной копии потребуется еще раз ввести пароль для Apple ID, учетной записи

электронной почты, а также других учетных записей и автоответчика. Если вы забыли пароль для автоответчика, обратитесь к оператору.

Вариант 1. Программа «арифметический калькулятор».
Вариант 2. Программа «перекодировщик символов».
Вариант 3. Программа «расчет контрольных сумм и хеш-функций».
Вариант 4. Программа «шифрование/расшифрование данных методом RC4».

Лабораторная работа 5. Разработка принципиальной схемы графического интерфейса веб- или настольного приложения по функциональным требованиям к приложению.

Суть работы: разработка интерфейса в PhotoShop и описание его в виде XML-спецификации.

Первый этап включает в себя планирование и эскизы. Рекомендуется набрасывать идеи для нескольких страниц (или экранов) вашего приложения. Просто нарисуйте прямоугольник, а может быть, 5 или 6, на листе бумаги, нарисуйте то, что вы хотите увидеть на каждой странице вашего приложения. Вы можете представлять экраны как разные страницы сайта. Каждый экран будет предлагать различные функции, такие как форма регистрации, список контактов или таблицу данных.

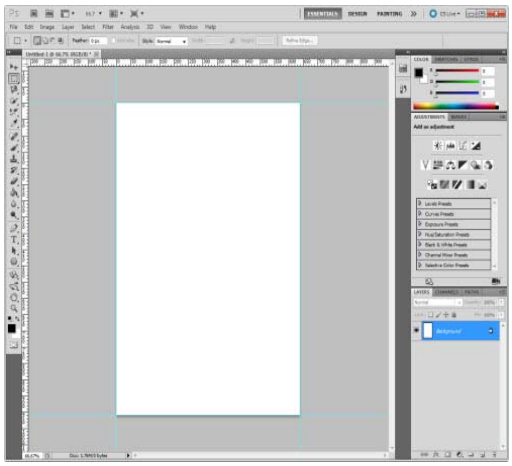
Элементы пользовательского меню:

- Строка состояния (Status Bar) — отображает текущий уровень зарядки, 3G-связь, прием и многое другое. Рекомендуется всегда включать эти элементы.
- Панель навигации (Navigation Bar) — Дает пользователям возможность перемещаться между страницами. Часто сюда включают кнопки на левой стороне меню, чтобы пользователь мог вернуться к предыдущей странице.
- Панель инструментов (Toolbar) — появляется в нижней части iPhone приложений. Здесь будет несколько иконок, связанных с некоторыми функциями, такими как поделиться, загрузить, удалить и т.д.
- Меню вкладок (Tab Bar) — Очень похоже на панель инструментов, только теперь вы работаете с вкладками. Когда пользователь щелкает по иконке вкладок, она будет автоматически выделена, и она будет подсвечиваться. Эта меню используется для переключения между окнами.

Дизайн макета в Photoshop

Уверен, что большинство из вас достаточно хорошо разбирается в Adobe Photoshop. Это ведущее программное обеспечение для создания графики для сайта, баннеров, логотипов, и мобильных макетов. Разработка графики для

сайта — довольно простой процесс, но все становится немного более сложным, когда речь заходит о дизайне iPhone приложений. Если вы хотите создать приложение, вы должны создать идеальный пиксельный дизайн макета с самого начала.



Для начала мы должны обсудить настройки в Photoshop. Так как мы разрабатываем для iPhone, нам необходимо рассмотреть два различных стиля дизайна. Стандартный дисплей iPhone составляет 320 x 480 пикселей. Однако iPhone 4 имеет новый дисплей «retina», который удваивает количество пикселей в пределах одного экрана. Таким образом, вы должны удвоить разрешение до 640 x 960 пикселей и создавать дизайн макетов и для этого стандарта.

Это означает, что вам также необходимо создать 2 набора иконок для ваших макетов. Первоначально иконки будут в 163 ppi, но вам нужно включить иконки с 326 ppi для iPhone 4. Иконки традиционно отмечаются @2x в конце названия их файлов, например «icon@2x.png» (Немного больше об этом в статье iPhone 4 Retina Display: Techniques and Workflow).
Теперь давайте оптимизировать наши новые настройки. Сначала мы должны отредактировать некоторые настройки, так что зайдите в Photoshop> Edit> Preferences> Guides, Grid and Slices. Мы установим линии сетки Gridline через каждые 20 пикселей с subdivisions на 2. При проектировании для дисплея iPhone 4, линии 2px будут означать 1 pts на экране. Вы должны это иметь в виду при масштабирования вашего приложения.

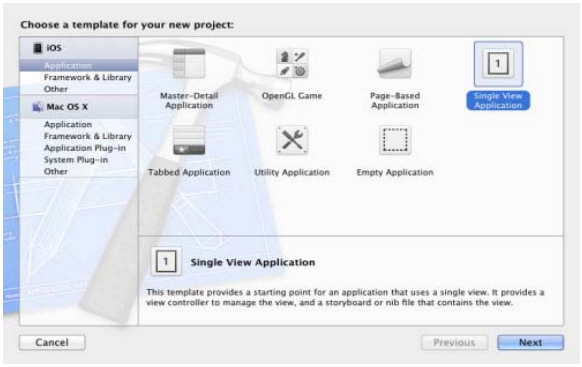
Как правило, легче создавать мои проекты с более высоким разрешением, а затем масштабировать их, но вы можете попробовать оба способа и посмотреть, что именно вам подходит лучше всего. Используйте настройки 640 x 960 пикселей при 326 ppi — лучше сохраните эти настройки, если вы предполагаете часто их использовать.
Использование шаблонных элементов

Теперь можно использовать Photoshop для создания идеального пиксельного макета, но это очень утомительная работа. Я рекомендую вам iPhone 4 GUI PSD от Teehan+Lax (см. Элементы интерфейса Mac, iPhone и iPad PSD).

Это огромный файл, в нем очень много элементов.
Совет: чтобы облегчить задачу, можно нажать V и активировать инструмент Move Tool, а затем выбрать в верхней панели Auto-Select, и выбрать Layer. С этими настройками вы можете нажать на любой элемент и Photoshop приведет вас к соответствующему слою.

Не бойтесь экспериментировать с макетом, вы даже можете создать прототип приложения с помощью макета. В зависимости от вашего приложения, вы можете включить множество функций, многие из которых вы сможете найти в этом файле PSD. Кроме того, можно перейти к слою этих элементов и отредактировать шрифт, градиент цвета, и другие стили дизайна. Главное — не изменяйте размер, так как все панели и элементы пользовательского интерфейса по умолчанию установлены стандартных размеров.
Разработка приложений в Xcode

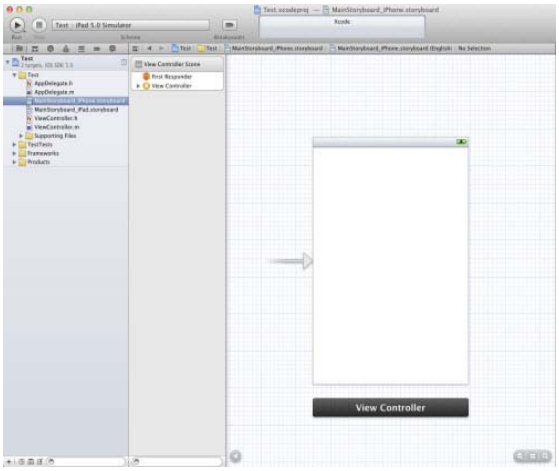
Инструмент разработки для IOS и Mac OS X программирования называется Xcode. Если вы работаете в OS X Lion, вы можете найти Xcode и все соответствующие пакеты бесплатно в Mac App Store.



После завершения установки, запустите Xcode, и появится экран приветствия. Отсюда вы можете загрузить старый проект или создать новый. Сейчас вы

должны нажать кнопку «Create a new Xcode project», и в новом окне вам будет предложено несколько вариантов. Выберите «Single View Application» и нажмите «Next». Вы можете назвать новое приложение, например, Test (желательно без пробелов), затем в поле «Company Identifier» введите любое слово, например, MyCompany, выберите папку и нажмите кнопку «Save».

Xcode создаст директорию и откроет новое окно для работы. Вы увидите список файлов, а папка, которая названа в честь вашего приложения, будет первой.



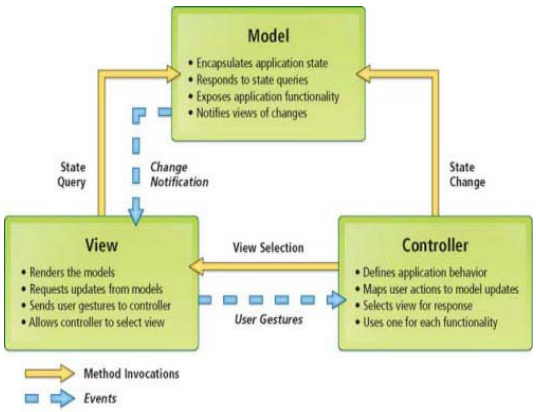
С новым Xcode 4,2 у нас есть два варианта дизайна элементов внешнего интерфейса. Классический xib/nib формат является стандартным для Mac OS X и приложений IOS, он требует от вас создавать каждый раз новый вид страницы.

Кроме того, вы увидите и файлы .h и .m. Это сокращенные имена файлов для заголовков и выполнения. Эти файлы находятся там, где вы пишете все Objective-C функции и переменные, необходимые для запуска вашего приложения.

Теперь нужно объяснить, как Xcode работает с MVC (Model, View, Controller), что является причиной того, что нам нужно 2 файла для каждого контроллера.

Программная иерархия MVC

Чтобы понять, как приложение работает, вам необходимо понять архитектуру его программирования. С Model, View, Controller (MVC) в качестве основы, Xcode может разделить все ваши дисплеи и коды интерфейса, исходя из вашей логики и функций обработки. На первый взгляд MVC может показаться запутанным, но это дело привычки.

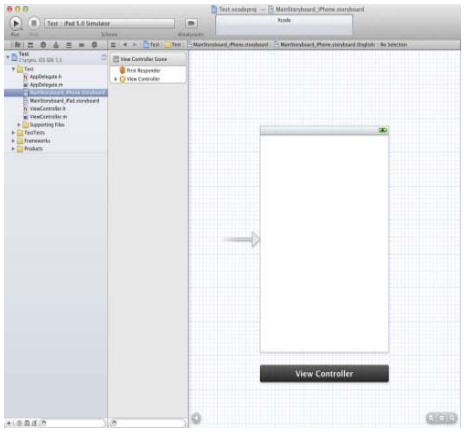


Чтобы было легче понять, разберём каждый объект:
Модель (Model) — Вмещает все ваши логические и основные данные. Это: переменные, подключения к внешним RSS-каналам или изображения, подробные функции и числовую информацию. Этот слой полностью отделяется от вашего визуального оформления, так что вы можете легко изменить вид дисплея, и у вас все равно останутся те же данные.
Вид (View) — экран или стиль отображения вашего приложения. Список таблиц, профиль страницы, статьи, аудио-плеер, видео плеер — все эти примеры экранов. Вы можете изменить свой стиль и удалять элементы, но вы будете работать с теми же данными в вашей модели.
Контроллер (Controller) — выступает в качестве посредника между ними. Вы подключаете объекты вида в ViewController, который передает информацию модели. Так что пользователь может нажать на кнопку, и зарегистрироваться в вашей модели. Затем выполнить выход из системы, и через тот же контроллер передать сообщение «вы успешно вышли из системы!»

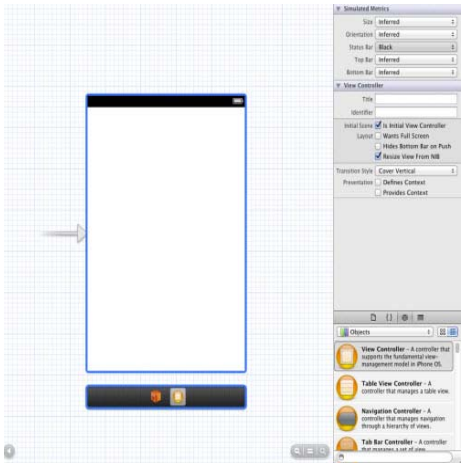
В основном ваша модель содержит всю информацию и функции, которые вам понадобятся для отображения на экране. Но модели не могут взаимодействовать с экраном, зато могут виды. Виды – это, в основном, все визуальные эффекты, и они могут только извлекать данные через ViewController. Контроллер на самом деле — это сложный способ передачи данных через интерфейс. Таким образом, вы можете обновлять дизайн, при этом не теряя какой-либо функциональности.
Обладая этими знаниями, вы не должны столкнуться с трудностями при попытке создать новое приложение. Как упоминалось ранее, Objective-C является основным языком программирования, который вы будете использовать для разработки приложений. Он построен на языке C, с обновленным синтаксисом и несколькими дополнительными парадигмами. Понадобится много времени, чтобы познакомиться с языком.
Дизайн вида с помощью iOS 5 Storyboards

Теперь, когда мы рассмотрели технические аспекты приложения, мы должны немного времени потратить на проектирование интерфейса. Я предполагаю, что вы выбрали «Storyboard» при создании проекта, что означает, что вы можете

найти файл MainStoryboard_iPhone.storyboard в папке, расположенной на левой стороне окна. Нажмите на файл, чтобы выбрать его и открыть.

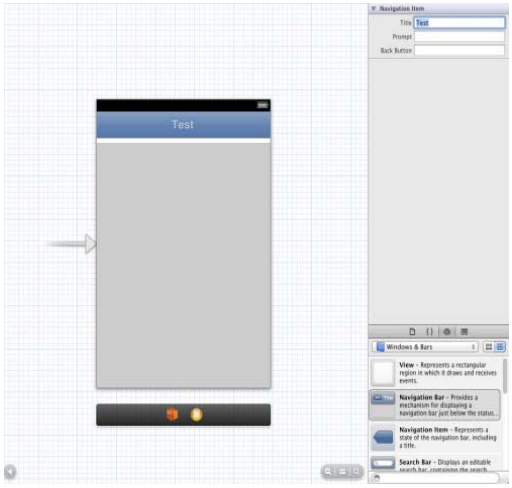


Новая боковая панель должна появиться непосредственно справа от папки. Это называется Document Outline, и это своего рода быстрый способ предварительного просмотра для проверки всех доступных видов в Storyboard.

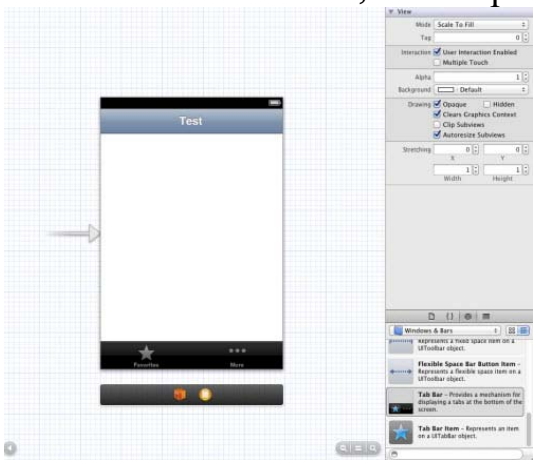


Мы хотим начать с добавления нескольких элементов страницы. Нам нужны два разных элемента: панель навигации Navigation Bar и панели вкладок Tab Bar. Перед тем, как создать их, зайдите в Attributes Inspector (View > Utilities > Show Attributes Inspector) в правой стороне окна, а затем посмотрите на Status Bar. По умолчанию он установлен в положение Inferred, который использует стандартный для iPhone цвет статуса, но вы также можете выбрать черный (Black) или прозрачный черный (Translucent Black), если они лучше подходят для вашего приложения.
Библиотека объектов

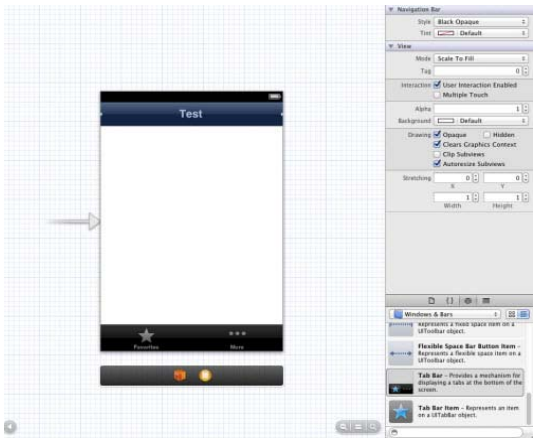
Если панель утилит на правой стороне окна не видна, ее можно включить, зайдя в View > Utilities > Show Utilities. На панели утилиты (Utilities) посмотрите на нижнюю панель под названием Object Library. Там есть выпадающее меню с пунктом «Objects» в начале списка. Если вы не смогли их найти, вы можете зайти в View > Utilities > Show Object Library.



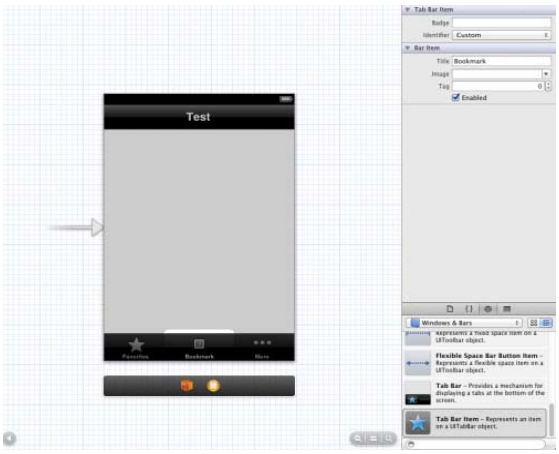
Из выпадающего меню Object Library, найдите и выберите Windows & Bars. Теперь нажмите на Navigation Bar (панель навигации), перетащите ее в окно просмотра и поместите непосредственно под черной строкой состояния Status Bar (это которая с индикатором аккумулятора). Мы можем настроить название и описание панели. Дважды щелкните на текст, где написано «Title», и вы увидите ярлык «Title» на панели утилит, где вы можете изменить название описание, например «Тест». Нажмите «Enter», посмотрите, что получится.



Опять в панели Windows & Bars прокрутите вниз, чтобы найти панели вкладок Tab Bar, а затем перетащите ее в окно просмотра и поместите ее в самом низу вашего приложения. По умолчанию эти два элемента выглядят отлично.



Теперь, может быть, вы хотите, чтобы градиент панели навигации сочетался с панелью вкладок в нижней части, и для этого вы можете нажать на панель навигации и посмотреть справа на панель Attributes в панели Utilities. Самая первая опция называется Style, который установлен по умолчанию. Измените стиль с дефолтного на Black Opaque, и ваши цвета будут сочетаться.



Давайте также добавим еще одну кнопку вкладок в нижней строке приложения. Наведите курсор мыши на панели Windows & Bars, и прокрутите вниз до Tab Bar Item, прямо под панелью вкладок. Перетащите его в ваше приложение, и поместите между 2 существующими кнопками на панель вкладок. Если вы дважды щелкните на эту кнопку, вы можете увидеть некоторые дополнительные опции в панели Utilities, здесь вы можете изменить изображение элемента и название. Например, в нашем примере название изменено на «Bookmark» для только что добавленного элемента панели вкладок.

Вариант 1. Программа «арифметический калькулятор».
Вариант 2. Программа «перекодировщик символов».
Вариант 3. Программа «расчет контрольных сумм и хеш-функций».
Вариант 4. Программа «шифрование/расшифрование данных методом RC4».

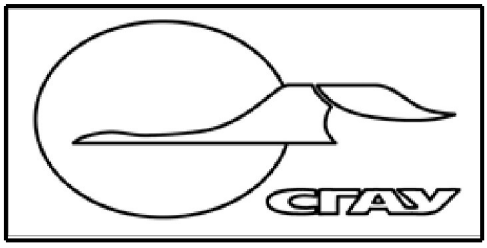
Контрольные вопросы к курсу
«Основы проектирования графического интерфейса компьютерных систем»

1.1. Каково назначение графического интерфейса устройства интерфейса устройства (GDI)?
1.2. Какие типы функций GDI Вы знаете?
1.3. Какие примитивы GDI Вы знаете?
1.4. Дайте характеристику метафайлам.
1.5. Что такое «сплайны Безье»? Каким образом их использовать?
1.6. Какие примитивы позволяют создавать функции API?
1.7. Дайте определение понятия «перо». Каким образом происходит создание, выбор и удаление перьев?
1.8. Что определяет режим рисования?
1.9. В чем особенность рисования полигонов?
1.10 Системы координат устройства. Чем отличаются физические и логические координаты?
1.11. Опишите процесс использования битовых образов.

2.1. В чем заключается принцип сетевой организации системы X Window? Что такое X-сервер ?
2.2. Как осуществляется в графической системе X Window взаимодействие прикладной программы с X-сервером ?
2.3. В чем особенность взаимодействия прикладной программы с X-сервером на локальной ЭВМ ?
2.4. Что такое окно с точки зрения системы X Window ?
2.5. На основе каких принципов организуется иерархия окон в системе X Window?
2.6. Что такое "ресурс приложения" в системе X Window ?
2.7. В чем общие черты и в чем различия графических сред Microsoft Windows и X Window ?
2.8. Расскажите о роли библиотеке "Xlib" в разработке программ X Window.
2.9. В чем заключается суть технологии программирования с управлением по событиям ?
2.10. Какие функции обеспечивают обработку событий (сообщений) в библиотеке Xlib ?
2.11. Что такое "механизм непринудительной многозадачности" ?

3.1. Перечислите основные элементы графического интерфейса Android
3.2. Перечислите основные элементы графического интерфейса iOS
3.3. Дайте общую характеристику графическим примитивам в Android
3.4. Дайте общую характеристику графическим примитивам в iOS
3.5. Охарактеризуйте общие принципы кросплатформенного портирования приложений

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)



СОГЛАСОВАНО

Управление образовательных программ

_____/ А.В. Дорошин /

"_____"_____ 20___ г.

УТВЕРЖДАЮ

Проректор по учебной работе

_____/ В.Н. Матвеев /

"____" _____ 20___ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование модуля (дисциплины)

М2.В.ДВ.2.2. Основы проект. графич. интерф. комп. систем

Цикл, в рамках которого происходит освоение модуля (дисциплины)	М2 - Профессиональный цикл
Часть цикла	М2.В.ДВ - Дисциплины по выбору

Код учебного плана

230100.68.2-13-56-3022

Факультет	Информатики
Кафедра	Инф. систем и технологий

Курс	5		
Семестр	9		
Лекции (СЛ)	18		
Семинарские и практические занятия (СП)	0		
Лабораторные занятия (СЛР)	36	Экзамен	
Контроль самостоятельной работы / Индивидуальные занятия (КСР / ИЗ)	0	Зачет	9
Самостоятельная работа (СРС)	54		
Всего (Всего с экзаменами)	108		

Наименование стандарта, на основании которого составлена рабочая программа:

230100.68 "Информатика и выч. техника"

Соответствие содержания рабочей программы, условий ее реализации, материально-технической и учебно-методической обеспеченности учебного процесса по дисциплине всем требованиям государственных стандартов подтверждаем.

Составители:

А. А. Калугин, К.Е. Климентьев _____
(подпись)

Заведующий кафедрой:

С.А. Прохоров _____
(подпись)

Рабочая программа обсуждена на заседании кафедры

Инф. систем и технологий

Протокол №___ от "____" _____ 20___ г.

Наличие основной литературы в фондах научно-технической библиотеки (НТБ) подтверждаем:

Директор НТБ _____ /_____/
(подпись) (расшифровка подписи)

Согласовано:

Декан _____ /_____/
(подпись) (расшифровка подписи)

1 Цели и задачи модуля (дисциплины), требования к уровню освоения содержания

1.1 Перечень развиваемых компетенций

Изучение дисциплины направлено на развитие следующих компетенций: ПК-3, ПК-5, ПК-11.

1.2 Цели и задачи изучения модуля (дисциплины)

Целью изучения дисциплины является получение студентами комплекса знаний и умений в области создания и использования эргономичных графических интерфейсов.
Задача курса состоит в изучении и практическом освоении студентами способов формирования, отображения, преобразования и хранения графической информации в мобильных системах.

1.3 Требования к уровню подготовки студента, завершившего изучение данного модуля (дисциплины)

Студенты должны:
- знать основные принципы построения видеоподсистем графических систем;
- решать задачи проектирования и реализации графических интефейсов.

1.4 Связь с предшествующими модулями (дисциплинами)

Изложение материала базируется на знании студентами следующих дисциплин: "Компьютерная графика"

1.5 Связь с последующими модулями (дисциплинами)

Дисциплина является предшествующей для выполнения квалификационной работы магистра.

2 Содержание рабочей программы (модуля)

Семестр 1		
СЛ 0,1667 18 часов 0,5001 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	Проблематика предмета: принципы Usability, назначение, цели и задачи, критерии, требования пользователя - 2 часа.
		Философия интерфейса: когнитивная психология и структурирование информации, оптимизация процессов взаимодействия пользователя с программной системой (optimal workflows), основы графического дизайна (цвет, контраст, типография, основы композиции) - 4 часа

		Типология программных систем и графических интерфейсов. Принципиальные отличия веб-приложений, приложений для настольных компьютеров и мобильных приложений. Основные «привычные»/«интуитивные» типы программных систем, мета-элементы графического интерфейса.
		Интерфейсы приложений для настольных компьютеров: парадигмы, принципы, примеры, отличительные особенности. Metro приложения - 2 часа
		Интерфейсы Веб-приложений: парадигмы, принципы, примеры, отличительные особенности. - 2 часа
		Интерфейсы приложений для iOS: Парадигмы, принципы, примеры, отличительные особенности. Приложения для IPhone и IPad, требования HIG. - 2 часа
		Интерфейсы приложений для Android: парадигмы, принципы, примеры, отличительные особенности. - 2 часа
СП 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СЛР 0,3333 36 часов 0,9999 ЗЕТ	Активные 0	
	Интерактивные 0	Портирование десктопного приложения под IPad. Раработка принципиальной схемы графического интерфейса - 2 часа
		Мобильный клиент для веб-сайта. Разработка принципиальной схемы графического интерфейса. - 2 часа
		Портирование приложения с iPad на iPhone – разработка принципиальной схемы графического интерфейса. - 2 часа
		Портирование iPhone приложения под Android. Разработка принципиальной схемы графического интерфейса - 2 часа

		Разработка принципиальной схемы графического интерфейса веб или настольного приложения по функциональным требованиям к приложению. - 4 часа
	Традиционные 0	
КСР 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СРС 0,5 54 часов 1,5 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	

3 Инновационные методы обучения

Лабораторные работы выполняются средствами вычислительной техники.

4 Технические средства и материальное обеспечение учебного процесса

Учебные комплекты в составе:
- устройство Mac Mini;
- клавиатура, мышь, ж/к монитор.
- операционная система Mac OS X.

5 Учебно-методическое обеспечение

5.1 Основная литература

1.Гультяев, Алексей Константинович. Проектирование и дизайн пользовательского интерфейса [Текст] / А. К. Гультяев, В. А. Машин. - СПб. : КОРОНА принт, 2000. - 349 с. Экземпляров всего:6

5.2 Дополнительная литература

2.Мунипов, Владимир Михайлович. Эргономика: человекоориентированное проектирование техники, программных средств и среды [Текст] : Учеб. для вузов / В. М. Мунипов, В. П. Зинченко. - М. : Логос, 2001. - 356 с. Экземпляров всего:8
1. Моргунов, Е. Б. Человеческие факторы в компьютерных системах [Текст] / Е. Б. Моргунов. - М. : ТОО "Тривола", 1994. Экземпляров всего:3

5.3 Электронные источники и интернет ресурсы

нет

5.4 Методические указания и рекомендации

нет