ПРОГРАММИРОВАНИЕ II

Пользовательские интерфейсы

Литература

1. Торрес Р.Дж. *Практическое руководство по проектированию и разработке пользовательского интерфейса*. –М.: Вильямс, 2002.
2. Мандел Тео. *Разработка пользовательского интерфейса*. М.: ДМК Пресс, 2001.
3. Скопин И.Н. Разработка интерфейсов программных систем. // *Проблемы архитектуры, анализа и разработки программных систем* / Системная информатика - Вып.6. - Новосибирск: Наука, 1998.
4. Macintosh Human Interface Guidelines. Apple. http://developer.apple.com/techpubs/mac/HIGuidelines/
5. Microsoft Official Guidelines for User Interface Developers and Designers. Microsoft. http://msdn.microsoft.com/library/default.asp

Аспекты взаимодействия человека и машины

- Адекватные функциональные средства взаимодействия с пользователем.
- Средства установки, конфигурирования и корректного удаления программной системы.
- Система подсказок и документация.
- Интернационализация.
- Использование системы людьми с ограниченными возможностями.

Классификация управляющих средств пользовательских интерфейсов

классы	подклассы	примеры
Символьный	Командный	«вопрос-ответ»
		командная строка
Графический	Простой графический	Экранные формы
		Управляющие клавиши
	Истинно графический. Двухмерный	Меню
		Графические элементы управления
	Трехмерный	Конические деревья
		Трехмерные манипуляторы

ПИ стандарты. Нужно ли?

- Вы не знаете о существовании стандартов или не считаете нужным следовать им.
- Вы создаете произведение искусства / развлекательную программу.
- Вы создаете программу с экстраординарными требованиями и отступление от стандартов преследует определенные цели.
- Вы претендуете на создание нового индустриального стандарта.

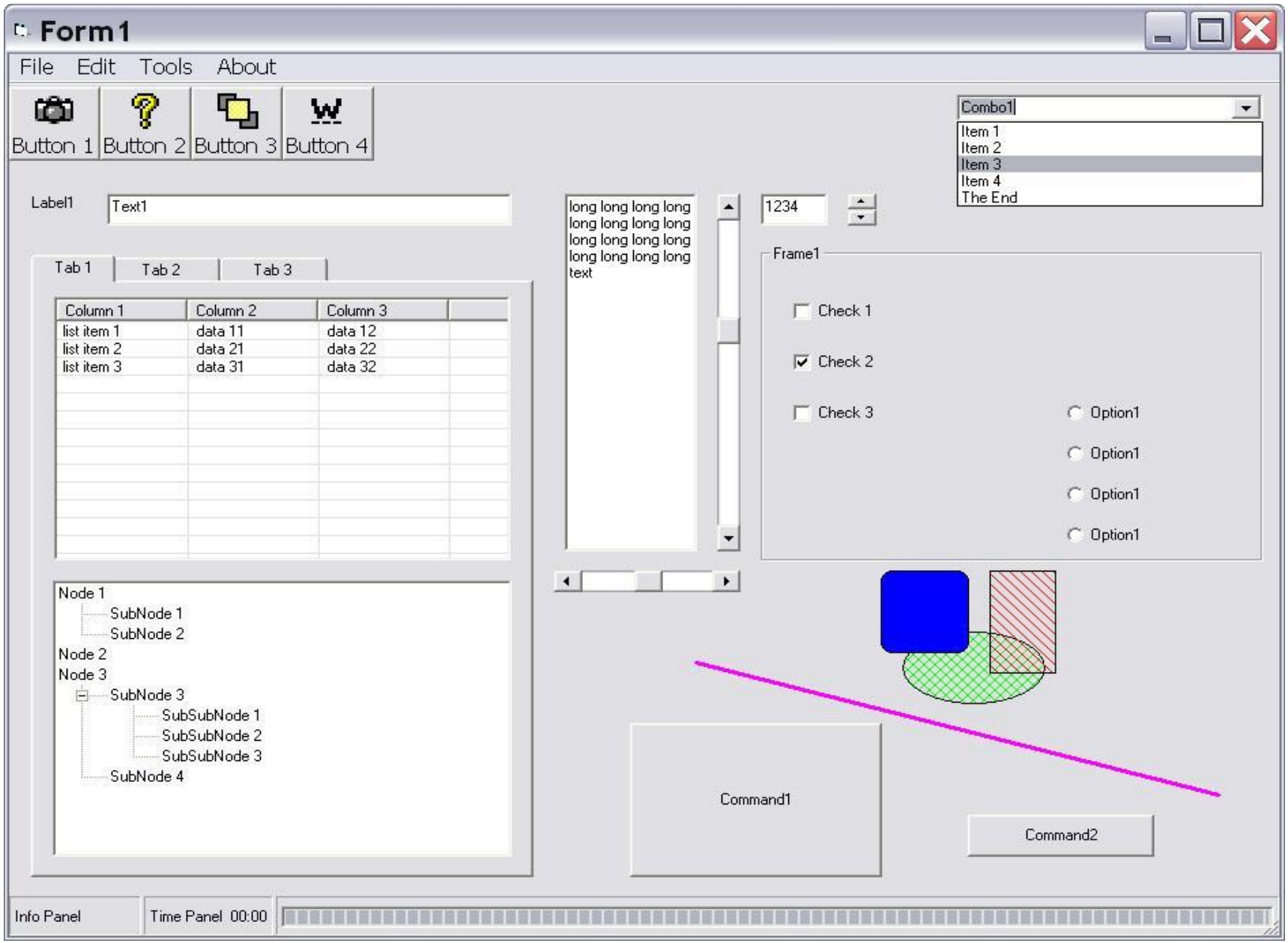
Интерфейсный стиль

- Интерфейсный стиль программного изделия – социально узнаваемый образ, который ассоциируется с этим проектом, продуктом и его составными частями.
- Дизайн интерфейса должен не противоречить, а подчеркивать общий стиль проекта.
- Компоненты дизайна не произвольны, а образуют некоторое стилевое единство.

Указания ISO 9241-10-98 об оценке и измерении usability ПИ

- эффективность (effectiveness) - влияния интерфейса на полноту и точность достижения пользователем целевых результатов;
- продуктивность (efficiency) или влияния интерфейса на производительность пользователя;
- степень (субъективной) удовлетворенности (satisfaction) конечного пользователя этим интерфейсом.

Основные элементы управления



Что нужно учитывать

- Использование стандартных элементов управления.
- Использование стандартных процедур размещения (layout engine) элементов управления?
- Следует придерживаться западной модели размещения информации.
- Доступ к функциональности должен быть обеспечен как с помощью клавиатуры, так и мыши.
- Информирование пользователя о длительно исполняемых действиях.
- Умолчания (default actions, Cancel, сохранение при закрытии приложения).
- Сообщение программы должны быть осмысленными.
- Использование системных цветов. Использование системных метрик. Неиспользование предположений о графическом режиме.
- Систематическая поддержка средств помощи (tooltips, context help, user manuals).
- ....

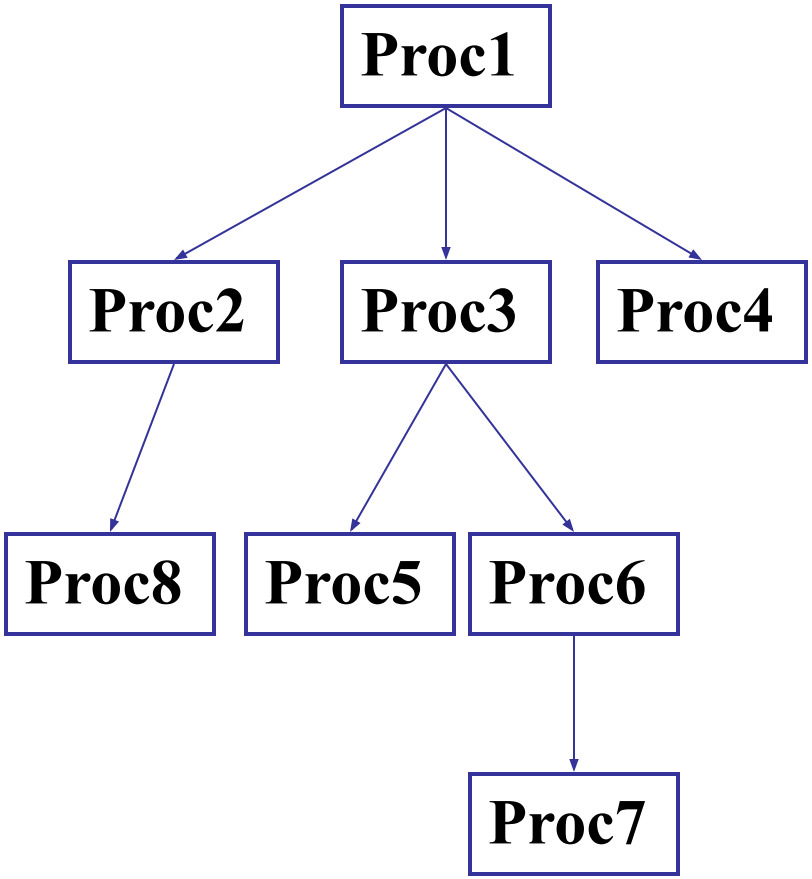
Некоторые примеры из CUA (IBM, около 1980 г.)

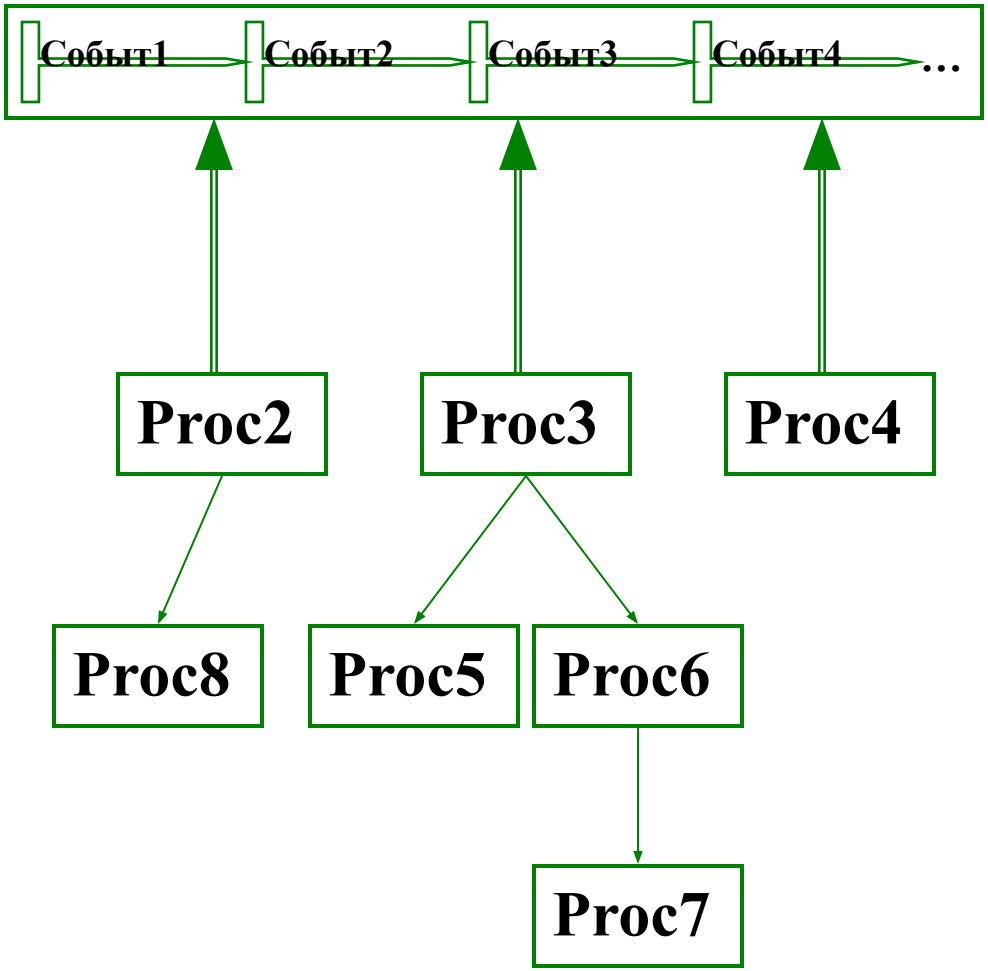
- любую операцию можно выполнить как мышью, так и клавиатурой;
- меню вызываются и скрываются клавишей F10 ;
- меню открываются нажатием клавиши Alt и подчёркнутой буквы в их названиях;
- команды меню, требующие уточнения параметров выполняемого действия, заканчиваются многоточием (…);
- параметры запрашиваются вторичными (диалоговыми) окнами;
- параметры сортируются по разделам с помощью вкладок;
- перемещение внутри полей в диалоговых окнах осуществляется клавишами управления курсором; между самими полями — клавишей Tab, а сочетанием Shift + Tab — в обратном направлении;
- в диалоговых окнах есть кнопка «Отмена», эквивалентная нажатию Esc , которая сбрасывает изменения, а также «ОК», эквивалентная нажатию Enter, которая принимает изменения;
- в программах есть встроенная справочная система, вызываемая из меню «Справка», расположенного в конце строки меню; контекстно-зависимая справка может вызываться клавишей F1;
- первое меню должно называться «Файл» и должно содержать операции по работе с файлами (создать, открыть, сохранить, сохранить как) и команду выхода; следующее меню «Правка» содержит команды отмены, повтора, вырезания, копирования, вставки и удаления;
- команда «Вырезать» выполняется нажатием Shift + Del , «Копировать» — Ctrl + Ins , а «Вставить» — Shift + Ins.

Сообщения

- Системные сообщения – данные, передаваемые о некотором событии, случившемся в системе (нажатие клавиши на клавиатуре, перемещение мыши или нажатие ее клавиши, изменение размеров окна, …), включают:
  - идентификатор окна, которому сообщение предназначено;
  - идентификатор сообщения;
  - параметры сообщения.
- Сообщения, определяемые приложениями. Event-driven programming – программирование приложений, управляемых событиями.

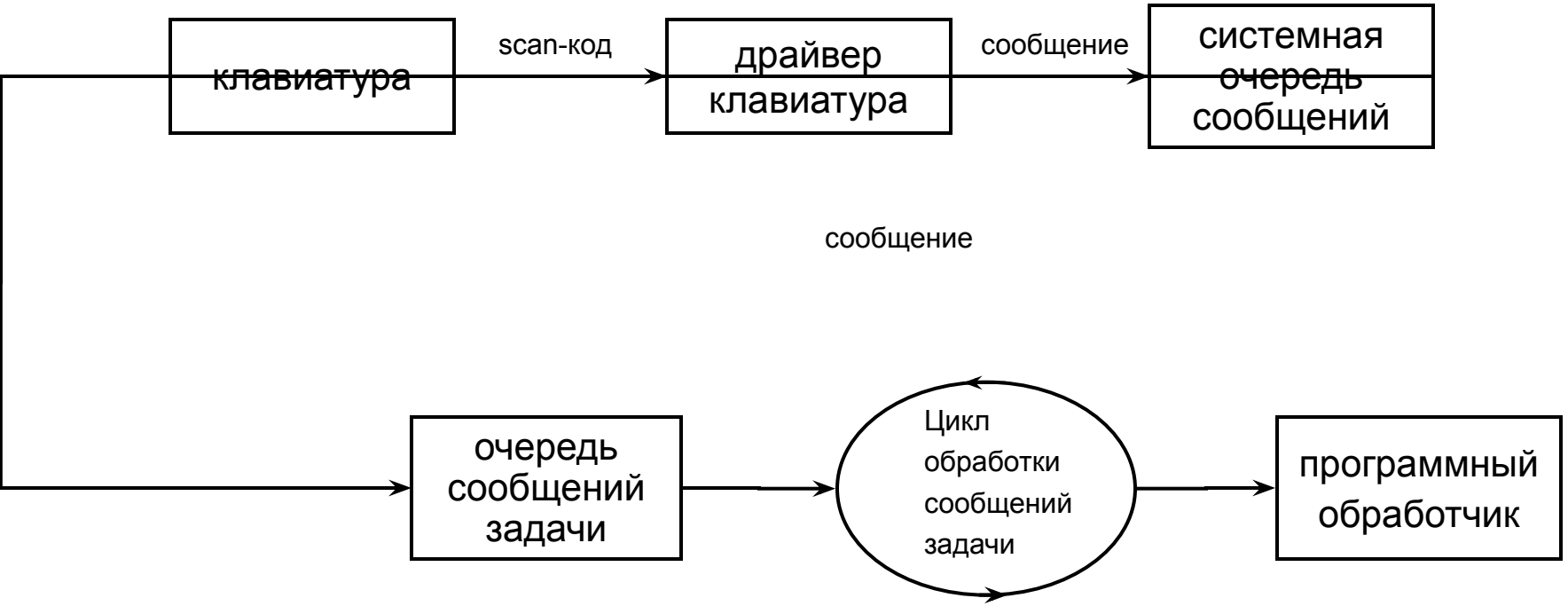
Event-driven programming





Сравнить E-D с C-B техникой.

Модель обработки событий



- Системная очередь сообщений – хранилище всех сообщений, полученных от клавиатуры или мыши и ожидающих передачи в очереди сообщений задач.
- Очередь сообщений задач – хранилище сообщений, уже идентифицированных как соответствующих задаче и ожидающих обработки в цикле обработки.
- В цикле обработке сообщения выбираются, параметры распаковываются и передаются в соответствующие програмные обработчики.

Некоторые события

- Initialize, Terminate, Resize, Paint, Load, Show, Hide, QueryUnload, Unload
- GotFocus, LostFocus
- Click, DblClick
- KeyDown, KeyUp, KeyPress
- MouseDown, MouseUp, MouseMove, MouseOver
- DragDrop, DragOver
- Change
- BeforeEdit, AfterEdit
- Expand (TreeView), Scroll (ScrollBar), ClickUp/ClickDown (UpDown)

Программные интерфейсы

```
BEGIN_MESSAGE_MAP(myFORM, CFormView)
    //{{AFX_MSG_MAP(myFORM)
    ...
    ON_BN_CLICKED(IDC_VeryImportantButton, OnVeryImportantButton)
    ON_NOTIFY(LVN_KEYDOWN, IDC_MyListObjects, OnKeydownMyListObjects)
    ...
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

...

void FORM::OnVeryImportantButton()
{
    // TODO: Add your control notification handler code here
}


void FORM::OnKeydownMyListObjects(NMHDR* pNMHDR, LRESULT* pResult)
{
    LV_KEYDOWN* pLVKeyDow = (LV_KEYDOWN*)pNMHDR;
    // TODO: Add your control notification handler code here
    *pResult = 0;
}
```

Приложение Windows Forms

- Создать проект типа Visual C++\CLR\Windows Forms Application.
- Добавить в проект новы элемент (форму) типа Visual C++\UI\Windows Form.
- Открыть Toolbox и установить на форму необходимые элементы управления.
  - Для упрощения процедур перерисовки использовался PictureBox.
- Определить необходимые события.

Получение доступа к Graphics

- Ссылку на объект `Graphics` можно получить через параметр типа `PaintEventArgs`, который передается в событие `Paint`, определенное для формы или элемента управления (ЭУ, control).

```
void Form1_Paint(Object^ sender, PaintEventArgs^ pe) {
   Graphics^ g = pe->Graphics;
   ...
}
```

- Метод `CreateGraphics`, имеющийся у форм или элементов управления, дает ссылку на объект `Graphics`. Этот объект представляет всю графическую поверхность формы или ЭУ.

```
Bitmap^ g = this->CreateGraphics();
```

- Всякий объект типа, являющийся наследником класса `Image`, предоставляет доступ к объекту `Graphics`.

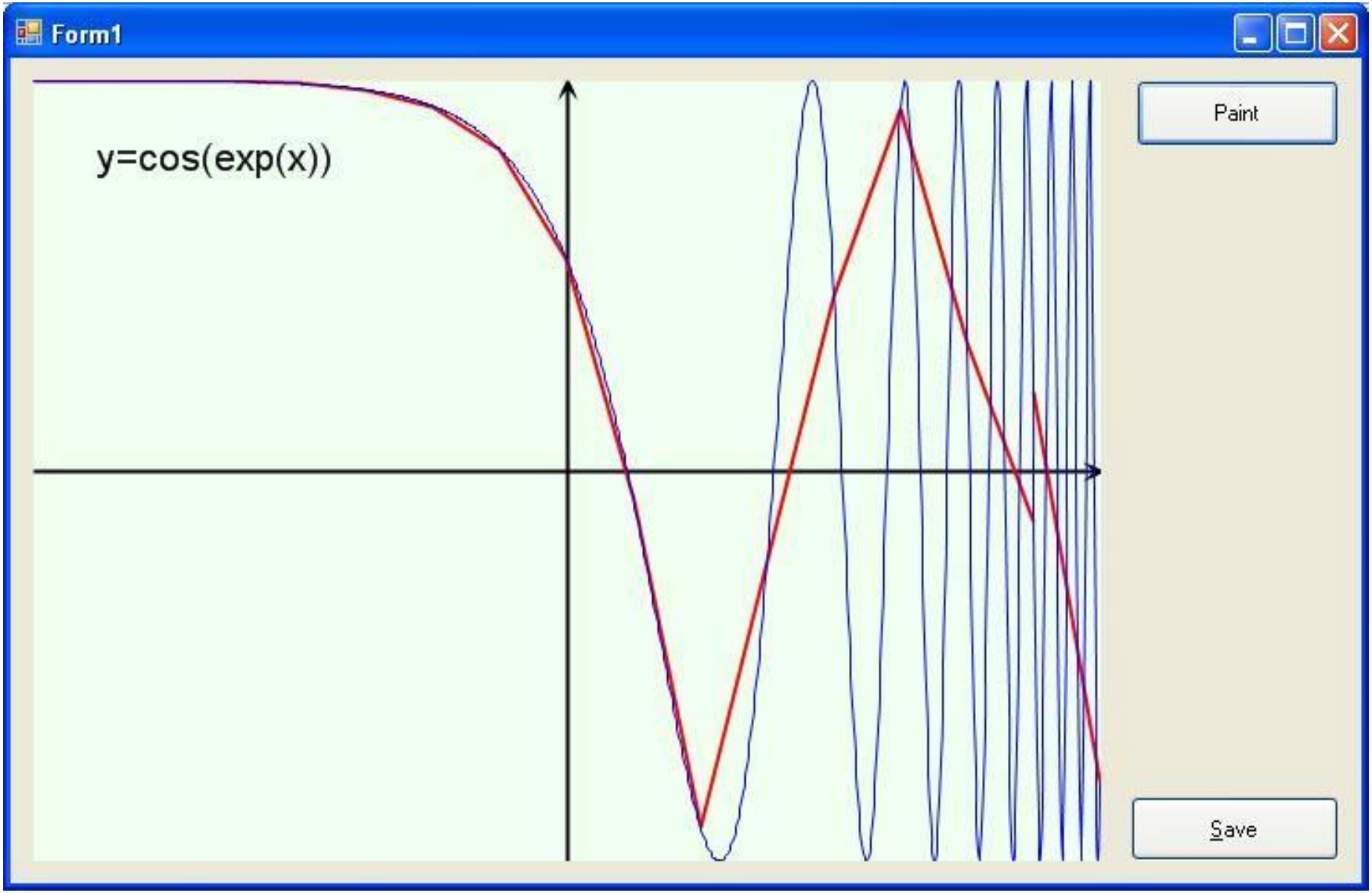
```
Bitmap^ myBmp = gcnew Bitmap("D:\\Pics\\myPic.bmp");
Graphics^ g = Graphics::FromImage(myBmp);
```

С использованием PictureBox

```
private:
System::Void Form1_Load(System::Object^ sender,
                        System::EventArgs^ e) {
   //...
   pbImage->Image=gcnew Bitmap(pbImage->Width,pbImage->Height);
}


private:
System::Void btnPaint_Click(System::Object^ sender,
                            System::EventArgs^ e) {
   Graphics^ gr = Graphics::FromImage(pbImage->Image);

   //...

   delete gr;
   pbImage->Refresh();
}
```

Разное

```
gr->SmoothingMode=Drawing2D::SmoothingMode::HighQuality;

SolidBrush^ br=gcnew SolidBrush(Color::Honeydew);
gr->FillRectangle(br,0,0,pbImage->Width,pbImage->Height);

Pen^ pn=gcnew Pen(Color::Black, 2);
gr->DrawLine(pn, 10, 100, 200, pbImage->Height);

array<Point>^ points={Point(0,0),Point(100,10),
                      Point(20,5),Point(305,100)};
gr->FillClosedCurve(gcnew SolidBrush(Color::Red), points);
gr->DrawCurve(pn, points);

gr->DrawString(«Какой-нибудь текст...",
               gcnew Drawing::Font("Arial",16), br, 30, 30);

delete pn;
delete br;
```


Form1
y=cos(exp(x))
Paint
Save