

Минимальные требования к Windows-приложению, реализующему визуализацию поиска экстремума двумерной функции

1. В класс поиска добавляете еще дек (deque) или список (list) из векторов, чтобы хранить там траекторию. Соответственно, на каждом шаге добавляете туда точку.
2. Программа должна уметь рисовать карту выбранной функции
3. По щелчку мышки над картой функции должно задаваться начальное значение поиска и запускаться сам поиск с установленными параметрами. После того как поиск остановится, траектория должна изображаться на фоне карты функции. Начальная и конечная точки выделены цветом. Координаты последней точки вместе со значением функции должны выводиться рядом с картой.
4. В параметрах можно: выбирать функцию (раскрывающийся список Combo box), выбирать метод (переключатель Radio Button), задавать область поиска и параметры поиска/критерия остановки.

В приложении у вас будет три новых элемента. Ниже помещены комментарии.

Как рисовать

Рисование (формирование картинки) идет в памяти, а затем все целиком выводится на экран. Это существенно быстрее, чем рисовать на экране попиксельно.

CClientDC dc(this); //получение контекста устройства (если в OnDraw, то контекст получать не надо – он передается через параметры, &dc -> pDC)

```
CBitmap bit;  
CDC dc2;  
  
bit.CreateCompatibleBitmap(&dc, width, height);  
  
dc2.CreateCompatibleDC(&dc);  
dc2.SelectObject( &bit );  
  
//Далее рисуете на dc2 как обычно. Например:  
dc2.FillSolidRect( 0, 0, width, height, RGB( 255,0,0) );  
//...  
//Затем выводите содержимое на экран:  
dc.BitBlt( x_left, y_top, width, height , &dc2, 0,0, SRCCOPY );
```

Как работать с Combo Box

Во-первых, когда создаете Combo-box, лучше убрать там установку, чтобы элементы сортировались по алфавиту (Property Sort = false). Размер раскрывающего списка можно поменять, щелкнув по стрелочке (треугольничку) вниз на комбо-боксе и растянув окно вниз.

Содержимое раскрывающего списка можно задать в Property Data, перечислив строки через точку с запятой без пробелов bbb;aaa;ccc;.

Но удобнее это делать в программе. В диалоге (при его создании лучше базовый класс поменять на CDialog), где находится этот элемент, вы связываете с ним не только обычную переменную (int или CString), но и control-переменную (CComboBox) . А затем добавляете строки с помощью функции AddString в OnInitDialog. Если функции OnInitDialog нет, то выделяете в Class View класс диалога, в свойствах (окне Properties) нажимаете на кнопку с изображением ластика (override), в появившемся списке функций находите OnInitDialog и делаете двойной щелчок по имени.

```

BOOL CAboutDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // TODO: Add extra initialization here
    combo_control.AddString("555");
    return TRUE; // return TRUE unless you set the focus to a control
}

```

Узнать результат выбора можно так:

```

CAboutDlg aboutDlg;
aboutDlg.DoModal();
if (aboutDlg.combo_val==1) AfxMessageBox("ooo"); // если связали int-
переменную
AfxMessageBox(aboutDlg.combo_string); // если связали CString-переменную.

```

Как реагировать на щелчок мышкой

Выделяете имя класса, который будет реагировать на клик. В свойствах (окне Properties) нажимаете на кнопку справа от изображения молнии, с подсказкой Messages. Там выбираете нужный идентификатор (например, если вы хотите реагировать на нажатие (DOWN) левой (L) кнопки (BUTTON) мыши, то идентификатор будет выглядеть так: WM_LBUTTONDOWN. Раскрываете окошко справа и выбираете <add>OnLButtonDown. Теперь у вас есть функция, которая вызовется при нажатии левой кнопки мыши. Можно сделать тоже самое для создания реакции на двойной щелчок (может быть, вам покажется, что задавать начальную точку двойным щелчком уместнее).

В функции OnLButtonDown есть параметры: nflag характеризует состояние кнопок мыши в момент щелчка левой кнопкой, point (класса CPoint из MFC) – координаты положения курсора мыши в момент щелчка относительно левого верхнего угла окна (названия полей класса CPoint – x и y).

=====

Прилагаются два примера (достаточно произвольных), на графику (Graph) и на комбо-бокс (Combo). В Graph используется реакция на таймер. Идея его использования такая: где-то вы его создаете (SetTimer), где-то «убиваете» (KillTimer), а внутри перегруженной (вы ее сами не создаете) функции OnTimer, которая вызывается автоматически по сигналам таймера, пишете те действия, которые должны происходить по этим сигналам (через 2 секунды в данном случае).

```

void CMainFrame::OnStartTimer()
{
    m_nTimer = SetTimer(1, 2000, 0);
}

void CMainFrame::OnStopTimer()
{
    KillTimer(m_nTimer);
}

void CMainFrame::OnTimer(UINT nIDEvent)
{
    MessageBeep(0xFFFFFFFF); // Beep - Ваша реакция на сигналы таймера

    // Call base class handler.
    CMDIFrameWnd::OnTimer(nIDEvent);
}

```