# Классы интерфейса 2012

Классы для работы с STL-файлами:

DDetail, Detail (Классы для прорисовки и изменения моделей):

void Draw(facet \*pFacet, int QntPlanes);

прорисовка детали без наложения текстуры

void Draw\_texture(facet \*pFacet, int QntPlanes, int \*PicSize);

прорисовка делали с наложение большой текстуры пиксель в пиксель

void Draw\_HMME(facet \*pFacet, int QntPlanes, int \*PicSize);

натягивание всей текстуры на объект

void SetRotate(float Rotete);

поворот фигуры по оси Z

void SetRotateX(float Rotete);

поворот фигуры по оси X

void SetScale(float ScX, float ScY, float ScZ);

изменение масштаба модели

void SetOffset(float X, float Y, float Z);

перенос центра модели в координаты (X,Y,Z)

STLRead (класс для чтения STL-файлов)

Класс для шрифта:

Для создания шрифта необходимо создать экземпляр класса CFonts. Пустой конструктор создаст шрифт с параметрами по умолчанию. Основные свойства шрифта: имя, размер, стиль (обычный, курсив, жирный, жирный курсив), цвет.

Прорисовка строки выполняется двумя функциями:

Void DrawString(CRectrect, constwchar\_t \*text, StringAlignmentOGLAligmentHor, StringAlignmentOGLAligmentVert);

Void DrawString(CRectrect, constwchar\_t \*text);,

Где rect– прямоугольник, в который выводится текст, третий и четвертый параметры – расположение текста (по умолчанию StrAlignmentNear).

Методы:

получить/изменитьразмер шрифта:

float GetSize() { return m\_param.Size; }

void SetSize(float newVal);

получить/изменить имя шрифта:

char\* GetName() { return m\_param.Name; }

void SetName(char\* newVal);

получить/изменить цвет шрифта:

COLORREF GetColor() { return m\_param.Color; }

Void SetColor(COLORREF newVal) { m\_param.Color = newVal; }

получить/изменить стиль шрифта:

FontsStyleGetStyle() { return m\_param.Style; }

Void SetStyle(FontsStylenewVal);

StringAlignmentOGL GetAligmentHorizontal() { return AligmentHorizontal; }

Void SetAligmentHorizontal(StringAlignmentOGLnewVal);

получить/изменить выравнивания по вертикали:

StringAlignmentOGL GetAligmentVertical() { returnAligmentVertical; }

Void SetAligmentVertical(StringAlignmentOGL newVal);

изменить шрифт:

FTFont\* GetFont() { return fonts; }

\*Перенос по строкам производится по буквам.

\*Если строка не помещается в заданный прямоугольник, для данной строки шрифт уменьшается.

## При работе с базовыми классами необходимо помнить:

Они ничего не должны знать о прорисовке элементов.

Базовые классы отвечают за реализацию «события», не связанного с рисованием. (m\_FunctionEvent)

Для элементов Button, ChanPanel переопределить виртуальные функции OnNotifyEvent, описанные в Custom\_ZETi12,

Например:

Virtual void OnNotifyEvent\_0(){}

m\_pButton2->m\_FunctionEvent = &CDialog\_ZETi12::OnNotifyEvent\_0;

Для элементов ComboBox, ChanSelector переопределить виртуальные функции OnNewLong, описанные в Custom\_ZETi12,

Например:

Virtual void OnNewLong\_0(longval){}

m\_pComboBox->m\_FunctionEvent = &CDialog\_ZETi12::OnNewLong\_0;

Для элементов Regulator, NumSelector, Slider, SimpleSlider переопределить виртуальные функции OnNewDouble, описанные в Custom\_ZETi12,

Например:

Virtual void OnNewDouble\_0(doubleval){}

m\_pSimpleSlider->m\_FunctionEvent = &CDialog\_ZETi12::OnNewDouble\_0;

Для всех остальных элементов данный функционал не реализован, при необходимости можно добавить его в Custom-классе данного компонента.

Необходимо, что бы в ZET- классе компонента было реализовано, что будет передаваться в качестве параметра при событии, и что считать событием. Для этого необходимо переопределить функции реакции, например, нажатие на кнопку или вращение колесика мыши, и указать в теле функции, что именно будет передаваться в функцию FunctionEvent.

На примере ComboBox:

Класс ComboBox:

Void CZComboBox::OnPickItem(long index)

{// оператор выбрал строку меню с индексом index

if ( m\_pDComboBox->GetIndexCur() != index)

{

m\_pDComboBox->SetIndexCur(index);

NewLong(index);

}

}

Класс CustomItem:

Void CCustomItem::NewLong(longval)

{

CWnd \*pParent = GetParent();

if (m\_pDClass->GetVisible() &&m\_pDClass->GetEnable() &&

m\_FunctionEvent&&pParent)

(((CCustom\_ZETi12\*)pParent)->\*m\_FunctionEvent)(val);

}

Теперь при выборе строки в меню номер этой строки будет передаваться в качестве входного параметра в функции m\_pComboBox->m\_FunctionEvent = &CDialog\_ZETi12::OnNewLong\_0;