```cpp
////////////////////////////////////////////////////////////////////////////////
//
// Класс для работы с панелью свойств
//
////////////////////////////////////////////////////////////////////////////////
#include "StdAfx.h"

#include "Resource.h"
#include "cPropMen.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern int LibMessage(LPCTSTR str, int flags = MB_OK);
extern void ClosePropertyManager(int mes);              /// Удаление пользовательской панели
extern void buildProfileObject();
extern void buildConditionsObject();

extern IPropertyManagerPtr propMng;
extern MenuProfile* menu_profile;
extern MenuConditions* menu_conditions;
extern enum MenuType* menu_type;

/// PropertyManagerEvent - обработчик событий от документа
PropertyManagerEvent::PropertyManagerEvent(LPDISPATCH manager) :
    ABaseEvent(manager, DIID_ksPropertyManagerNotify)
{
    Advise();
}

///
PropertyManagerEvent::~PropertyManagerEvent()
{
}

/// Карта сообщений
BEGIN_EVENTSINK_MAP(PropertyManagerEvent, ABaseEvent)
    /// 1 - prButtonClick - Нажатие кнопки.
    ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prButtonClick, ButtonClick, VTS_I4)
    /// 2 - prChangeControlValue - Событие изменения значения контрола
    ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prChangeControlValue, ChangeControlValue, VTS_DISPATCH)
    /// 3 - prControlCommand - Нажатие кнопки контрола
    ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prControlCommand, ControlCommand, VTS_DISPATCH VTS_I4)
    /// 4 - prButtonUpdate - Обновление состояния кнопки
    ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prButtonUpdate, ButtonUpdate, VTS_I4 VTS_PI4 VTS_PBOOL)
    /// 5 - prProcessActivate - Активизация процесса
```

```
50      ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prProcessActivate, ↵
           ProcessActivate, VTS_NONE)
51      /// 6 - prProcessDeactivate - Деактивизация процесса
52      ON_EVENT(PropertyManagerEvent, (unsigned int)-1,                    ↵
          prProcessDeactivate, ProcessDeactivate, VTS_NONE)
53      /// 7 - prCommandHelp - Вызов справки
54      ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prCommandHelp,     ↵
          CommandHelp, VTS_I4)
55      /// 8 - prSelectItem - Селектирование элемента в списке
56      ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prSelectItem,      ↵
          SelectItem, VTS_NONE)
57      /// 9 - prCheckItem - Выбор элемента в списке
58      ON_EVENT(PropertyManagerEvent, (unsigned int)-1, prCheckItem,       ↵
          CheckItem, VTS_NONE)
59  END_EVENTSINK_MAP()
60
61  /// prChangeControlValue - Событие изменения значения контрола
62  afx_msg BOOL PropertyManagerEvent::ChangeControlValue(LPDISPATCH iCtrl) ↵
        {
63      if (iCtrl) {
64          IPropertyControlPtr contol = iCtrl;
65          long id = contol->Id;
66          if (id == MenuProfile::Elements::e_hasHole) {
67              menu_profile->checkHasHole();
68              propMng->UpdateTabs();
69          }
70          else if (id == MenuProfile::Elements::e_cutRoundingType) {
71              menu_profile->checkEvolutionRoundingType();
72              propMng->UpdateTabs();
73          }
74          else if (id == MenuConditions::Elements::e_hasHole) {
75              menu_conditions->checkHasHole();
76              propMng->UpdateTabs();
77          }
78          else if (id == MenuConditions::Elements::e_angleEtaAuto) {
79              menu_conditions->checkAngleEtaAuto();
80              propMng->UpdateTabs();
81          }
82      }
83      return TRUE;
84  }
85
86  /// Получение файла помощи
87  void OpenHelp(int Id) {
88      char gaykaHlp[260] = "";
89      //if ( GetFullName( LIB_HELP, gaykaHlp, 260 ) )
90          //ksOpenHelpFile( gaykaHlp, HELP_CONTEXT,   Id );
91  }
92
93  /// prChangeControlValue - Событие изменения значения контрола
94  afx_msg BOOL PropertyManagerEvent::ButtonClick(long buttonID) {
95      if (buttonID == pbHelp) {
96          OpenHelp(1);
97      }
98      else if (buttonID == pbEsc) {
```

```
 99          ClosePropertyManager(0);
100      }
101      else if ((*menu_type == type_profile) && (buttonID == pbEnter)){
102          buildProfileObject();
103      }
104      else if ((*menu_type == type_conditions) && (buttonID == pbEnter)){
105          buildConditionsObject();
106      }
107      else if (buttonID == MenuProfile::Elements::e_button){
108          buildProfileObject();
109      }
110      else if (buttonID == MenuConditions::Elements::e_button){
111          buildConditionsObject();
112      }
113      else {
114          ClosePropertyManager(0);
115      }
116      return TRUE;
117  }
118
119  /// prControlCommand Нажатие кнопки контрола
120  afx_msg BOOL PropertyManagerEvent::ControlCommand(LPDISPATCH ctrl, long ↵
       buttonID) {
121      if (buttonID == MenuProfile::Elements::e_button) {
122          buildProfileObject();
123      }
124      else if (buttonID == MenuConditions::Elements::e_button) {
125          buildConditionsObject();
126      }
127      return TRUE;
128  }
129
130  /// prButtonUpdate - Установка состояния кнопки спецпанели.
131  afx_msg BOOL PropertyManagerEvent::ButtonUpdate(long buttonID, long*    ↵
       check, VARIANT_BOOL* _enable) {
132      if (check)
133          *check = false;
134      if (_enable)
135          *_enable = TRUE;
136
137      return TRUE;
138  }
139
140  /// prProcessActibate - Начало процесса.
141  afx_msg BOOL PropertyManagerEvent::ProcessActivate() {
142      return TRUE;
143  }
144
145  /// prProcessDeactivate - Завершение процесса.
146  afx_msg BOOL PropertyManagerEvent::ProcessDeactivate() {
147      return TRUE;
148  }
149
150  /// Вызов справки
151  afx_msg BOOL PropertyManagerEvent::CommandHelp(long buttonID) {
152      OpenHelp(1);
153      return TRUE;
```

```
154 }
155
156 /// Селектирование элемента в списке
157 afx_msg BOOL PropertyManagerEvent::SelectItem() {
158     return TRUE;
159 }
160
161 /// Выбор элемента в списке
162 afx_msg BOOL PropertyManagerEvent::CheckItem() {
163     return TRUE;
164 }
```