МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«27» 12 2017 г.

ОПЕРАЦИОННОЕ ОКРУЖЕНИЕ СИСТЕМЫ VISUAL PROLOG

Методические указания по выполнению лабораторной работы для студентов направления подготовки 09.03.04 «Программная инженерия» по дисциплине «Функциональное и логическое программирование»

Курск 2017

2

УДК 004.652

Составители: В.Г. Белов, Т.М. Белова

Рецензент

Кандидат технических наук,доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Операционное окружение системы VisualProlog: методические указания по выполнению лабораторной работыпо дисциплине "Функциональное и логическое программирование" для студентов направления подготовки 09.03.04 "Программная инженерия" / Юго-Зап. гос. ун-т; сост.: В.Г. Белов, Т.М. Белова, – Курск, 2017. – 47 с.: ил. 48.

Изложена процедура работы с операционным окружением системы **VisualProlog**.

Материал предназначен для студентов направления подготовки 09.03.04 «Программная инженерия» специализации «Разработка программно-информационных систем»

Подписано в печать 24.12.17. Формат 60x84 1/16.
Усл. печ. л. 2,14. Уч.-изд. л. 2,2. Тираж 100 экз. Заказ 438. Бесплатно.
Юго-Западный государственный университет

305040, Курск, ул.50 лет Октября, 94.

СОДЕРЖАНИЕ

1. Цель лабораторной работы 1

Целью лабораторной работы является изучение основ функционирования Visual Prolog, включая порядок работы с системой меню, запуск программ, написанных на Visual Prolog, а также создание файлов с текстами программ с использованием редактора Visual Prolog.

2. Знакомство со средой Visual Prolog 1

Среда Visual Prolog представляет собой стандартную программную среду ОС Windows. Ярлык быстрого запуска среды обычно находится в главном меню в подменю Программы | Visual Prolog Personal Edition и называется VIP32.

Как и в среде Delphi, в Visual Prolog имеется понятие проекта. Каждый проект - это отдельная программа. Чтобы создать новый проект, необходимо в меню Project выбрать пункт New Project.... После этого появится окно Application Expert с несколькими страничками. В первой страничке нужно ввести имя проекта (поле ввода Project Name), имя главного файла проекта (Name of .VPR File) и директорию, где будут храниться файлы проекта (Base Directory). Затем необходимо нажать копку Create. После этого появится окно Project. Окно Application Expert для существующего проекта можно вызывать из главного меню **Options|Project (.VPR)|Application Expert**

2.1. Создание приложения "Hello World" 2

Пусть надо создать небольшой диалог с подсказкой “Hello World”.

Для этого нужно добавить пункт меню, при выборе которого будет активизирован предопределенный общий диалог dlg_Note.

Чтобы выполнить это, необходимо:

- добавить вход меню;
- добавить предложение, чтобы появилась реакция, когда пункт меню станет выбранным
- в этом предложении, сделать обращение к предикату dlg_Note.

2.1.1. Изменение меню в редакторе меню 3

Чтобы изменить меню, необходимо в окне Project нажать кнопку Menu в левой инструментальной панели.

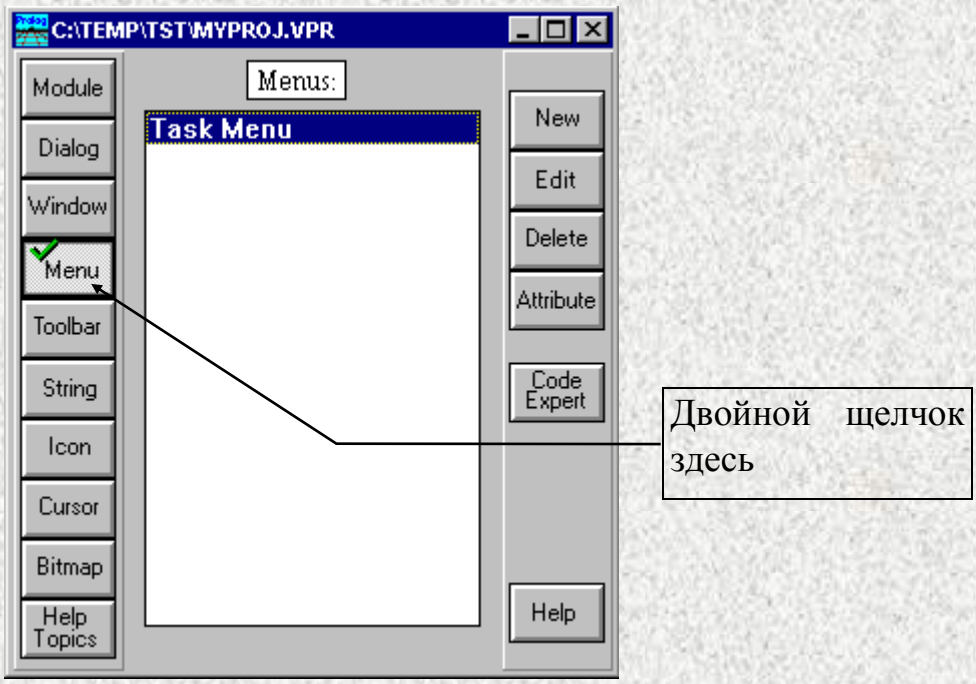

Рисунок 1 - Окно Project, показывающее меню

Двойной щелчок на Task Menu в окне Проекта, активизирует редактор меню.

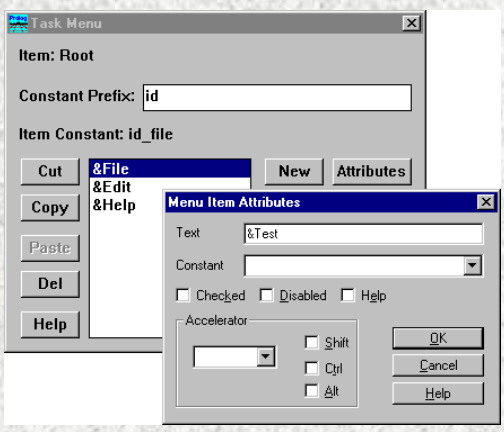

Рисунок 2 - Добавление пункта меню в редакторе меню.

Теперь нужно спуститься вниз к пункту меню Edit, и добавить новый, верхнего уровня, пункт меню с именем Test, нажатием кнопки New. Имя для входа меню автоматически получит по умолчанию константу 'id_test'. Символ '&', который является префиксом для некоторых пунктов меню появится как символ подчеркивания, когда покажется меню. Это используется, чтобы определить горячую клавишу.

Горячая клавиша может быть определена, набором символа в поле Acceralator и нажатием соответствующих комбинаций с клавишами Shift/Alt и Ctrl.

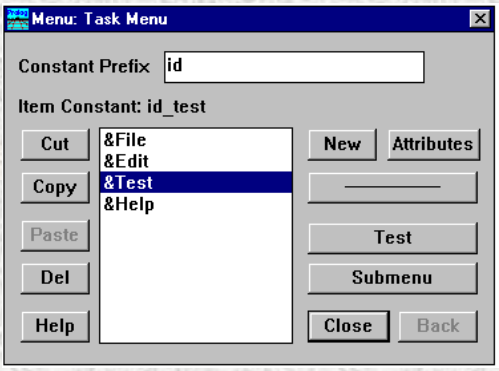

Рисунок 3 - Редактор меню после добавления пункта меню Test.

Для создания подменю с единственным элементом Hello World, который будет использоваться для сообщения Hello World необходимо нажать кнопку Submenu, которая следует за кнопкой New:

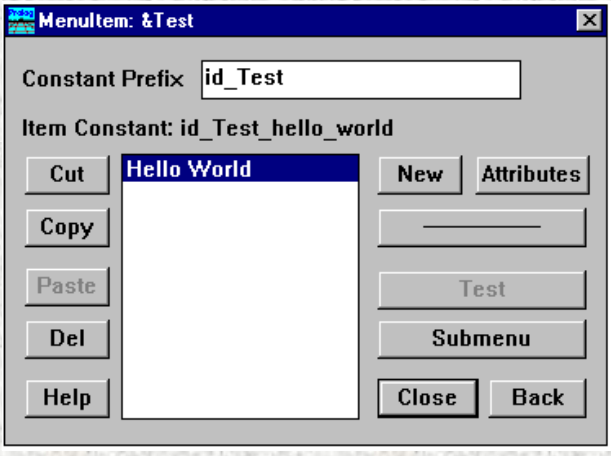

Рисунок 4 - Редактор меню в подменю.

Позже можно соединить вход меню с выполняемым кодом, используя постоянное значение, определяемое следующим образом. Нажатием кнопки Attribute, или двойным щелчком на выбранном пункте меню, выберем диалог **Menu Item Attributes**. Изменим автоматически сгенерированную константу на **id_hello**. Нажмите **OK**, чтобы подтвердить изменение.

Рисунок 5 - Атрибуты меню Editing.

В вышеупомянутом диалоге можно определить, что вход меню будет заблокирован (недоступен) или выбран (отмечен галочкой), или, если необходимо, может быть определена горячая клавиша для входа в меню.

Для проверки меню необходимо нажать кнопку Back, после чего она станет заблокированной, затем выбрать меню Test, чтобы предварительно просмотреть меню:

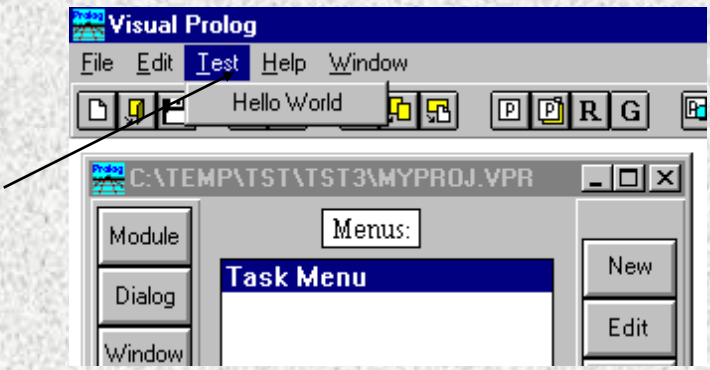

Рисунок 6 - Редактор меню в режиме теста

Чтобы выйти из режима теста, надо просто щелкнуть 2 раза мышью в любом окне за пределом меню, или снова нажать кнопку Test. После этого надо закрыть редактор меню, нажав кнопку Close.

2.1.2. Использование окна Dialog and Window Expert 3

Теперь нужно использовать текстовый редактор VP, чтобы добавить маленький фрагмент кода для создания диалога Hello World. Но в Visual Prolog лучше сделать, используя Code Experts.

Рекомендуем всегда использовать Dialog and Window Expert, чтобы добавить исходный текст Пролога для входа в меню. Сначала нужно выбрать Window из окна Project, а затем - окно Task, нажатием соответствующего элемента.

Рисунок 7 - Вызов Dialog and Window Expert из окна Project.

Затем нужно нажать на кнопку Code Expert, находящуюся справа (или горячую клавишу Ctrl + W) для вызова окна Dialog and Window Expert.

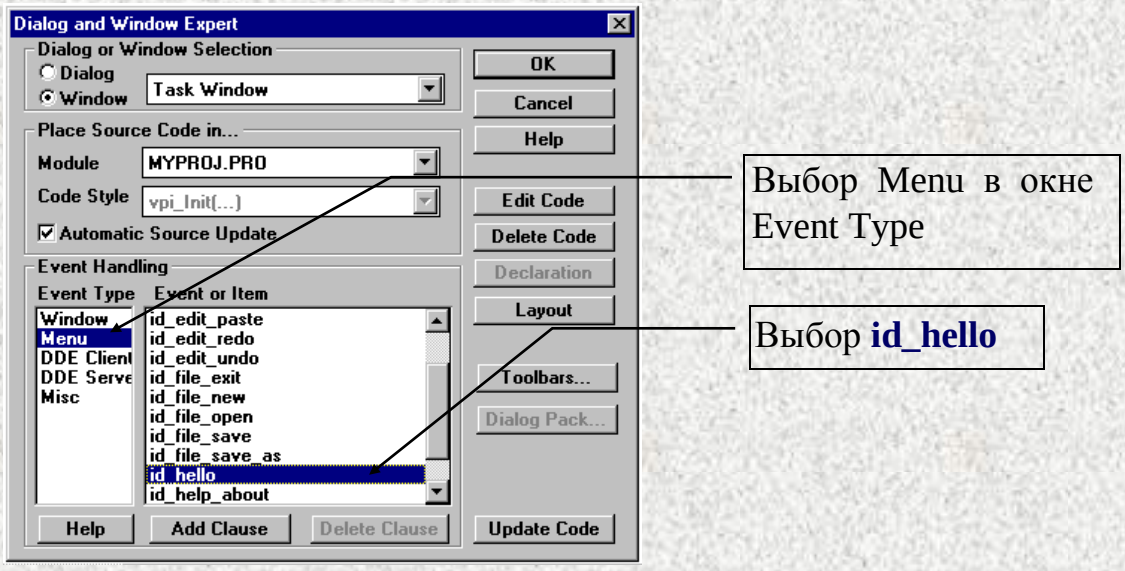

Рисунок 8 - Создание кода меню с помощью окна Dialog and Window Expert.

Когда появится окно Dialog and Window Expert, надо выполнить следующие шаги:

- выбрать пункт Menu из окна Event Type (слева);
- высветить пункт id_hello (или любой другой, который обозначен как константу меню) в окне Event or Item (можно использовать полосу прокрутки, чтобы найти его);
- нажать кнопку Add Clause, чтобы сгенерировать предложение пролога для события. (текст на кнопке Add Clause изменится на Edit Clause, когда код для события будет сгенерирован);
- нажать кнопку Edit Clause;
- нажатие кнопки Edit Clause приведет к появлению окна Editor. Обратите внимание, что следующее предложение теперь будет добавлено к предикату обработчика события для окна Task:

```
%BEGIN Task Window, id_hello
task_win_eh(_Win,e_Menu(id_hello,_ShiftCtlAlt),0):-! ,
!.
%END Task Window, id_hello
```

%BEGIN – %END комментарии используются окном Dialog and Window Expert, чтобы разместить код для компонентов интерфейса пользователя.

Переменная _Win содержит дескриптор окна, в данном случае – окна Task. Переменная _ShiftCtlAlt содержит флаг через который Вы можете получить комбинацию клавиш Shift и Ctrl, которые использовались, когда меню было активно. В обоих случаях, подчеркивание – синтаксическое соглашение, которое отменяет предупреждение компилятора для единичного использования имени переменной (как часто могло бы быть с неиспользуемыми переменными).

Чтобы добавить код, можно использовать редактор меню Insert. Для этого надо поместить курсор редактора в то место, где необходимо вставить код (в данном случае – в начало строки после первого вырезанного фрагмента), затем нажмем правую кнопку мыши, и выберем пункты Insert | Predicate Call | Window, Dialog or Toolbar.

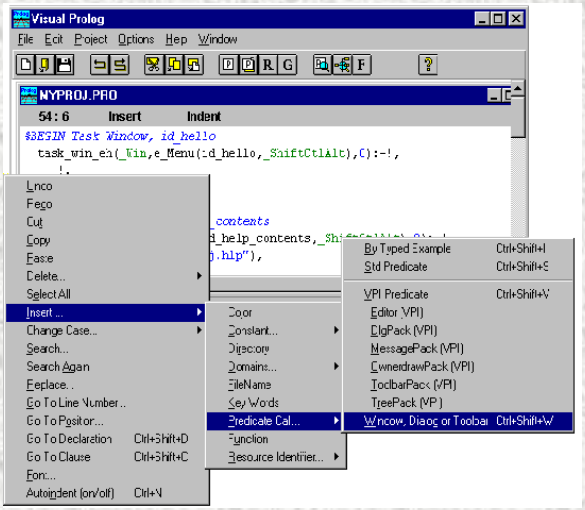

Рисунок 9 - Редактор меню вставки.

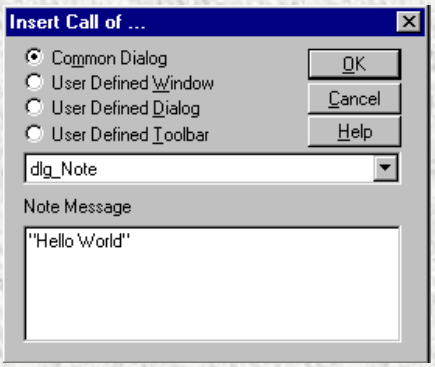

Рисунок 10 - Определение предиката для вставки.

В диалоге, который теперь появится, используйте кнопку списка , чтобы выбрать общий диалог dlg_Note и напечатайте строку "Hello World" (включая кавычки).

Вызов сообщения будет затем вставлен в источник, так, чтобы заключительный код для предложения выглядел так:

```
task_win_eh(_Win,e_Menu(id_hello,_ShiftCtlAlt),0):-!,
Title="Title",
dlg_Note(Title,"Hello World"),
!.
```

Для запуска программы нужно выбрать пункт меню Project | Run (или нажать кнопку),. Когда вход меню активен, появится следующий диалог, при выборе пункта меню Test | Hello:

Рисунок 11 - Диалоговое окно, которое появится в приложении.

2.2. Создание окна "Cross Window" 2

Этот пример покажет, как обрабатывать различные события в Visual Prolog.

2.1.1. Создание нового исходного модуля 3

Необходимо добавить исходный текст Пролога для окна Cross, и так как у него будет код, который является независимым от остальной части кода в проекте, поместим этот код в отдельный (новый) модуль. В Visual Prolog можно обрабатывать много окон и диалогов в том же самом исходном модуле – программист приложения сам решает, как организовать проект. В этом случае, создадим новый модуль. В окне Project нажмем кнопку New, в то время как выбрана кнопка Module.

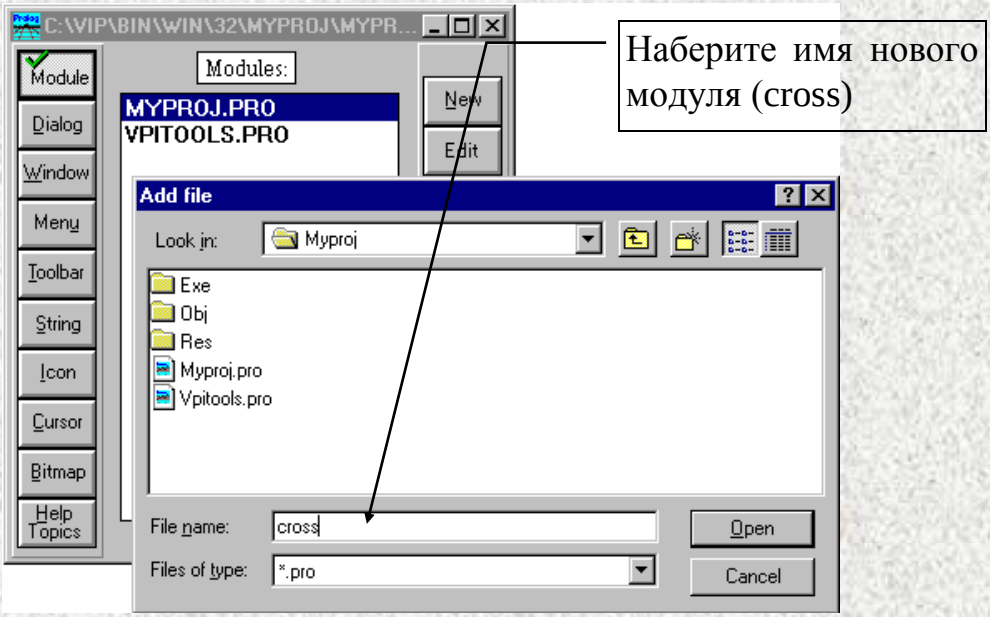

Рисунок 12 - Добавление нового исходного модуля.

После ввода имени нового модуля, нажмем OK, появится диалог, в котором можно указать должен ли .DOM и .PRE файл быть сгенерирован для модуля. Все экспортируемые глобальные объявления области для модуля хранятся в .DOM файлах, и все экспортируемые глобальные предикаты - в .PRE файлах.

Для реальных проектов, необходимо рассмотреть следующие пункты:

- не нужно включайте никакие .PRE файлы в MYPROJ.INC файл. Но можно включить .PRE файлы в каждый модуль - это даст возможность пройти компилятору через меньшее количество файлов;
- файл констант Resource (здесь MYPROJ.CON) должен быть включен в модуль, только если есть ссылки из этого модуля;
- файл констант Help (HLPTOPIC.CON) должен быть включен в модуль, только если есть ссылки из этого модуля.

.PRE файлы содержат определения предикатов Пролога и функции (они имеют функцию подобную .H файлам в Си), а .DOM файлы - об областях определяемых пользователем.

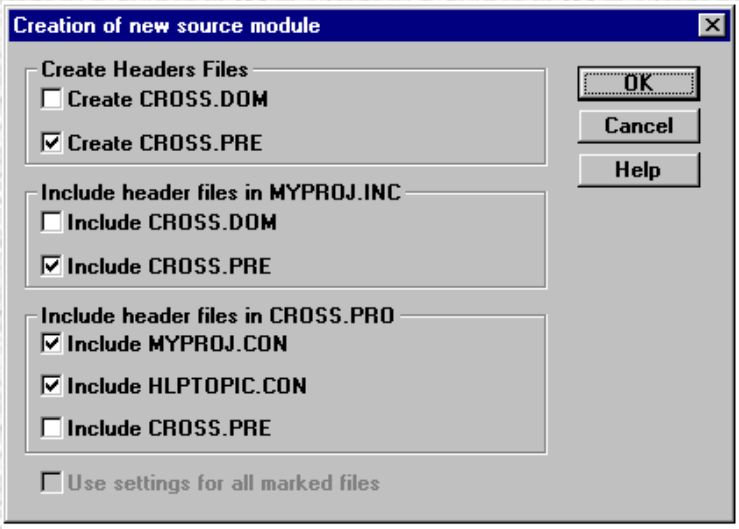

Рисунок 13 - Диалог атрибутов для модуля.

Нажатие OK принимает значения по умолчанию. Если просматривается только что сгенерированный исходный модуль, то заголовок, основанный на установках в Application Expert был сгенерирован, и что глобально подключаемый файл для проекта включен.

```
/*****************************************************
Copyright (c) Prolog Development Center
Project: MYPROJ
 FileName: CROSS.PRO
 Purpose: A little Demo
 Written by: Leo Jensen
 Comments:
*****************************************************/
include "myproj.inc"
include "myproj.con"
include "hlptopic.con"
```

Новый файл CROSS.PRE также был сгенерирован, и помещен в каталоге проектов. При просмотре глобально подключаемого файла, можно увидеть, что CROSS.PRE включен в конце этого файла:

```
include "cross.pre"
```

Dialog and Window Experts помещает глобальные объявления в .PRE файлы.

2.1.2. Создание нового окна 3

Сначала надо зарегистрировать новое окно в проекте. Для этого надо нажать кнопку Window с левой стороны в проекте, а затем кнопку New (вверху справа).

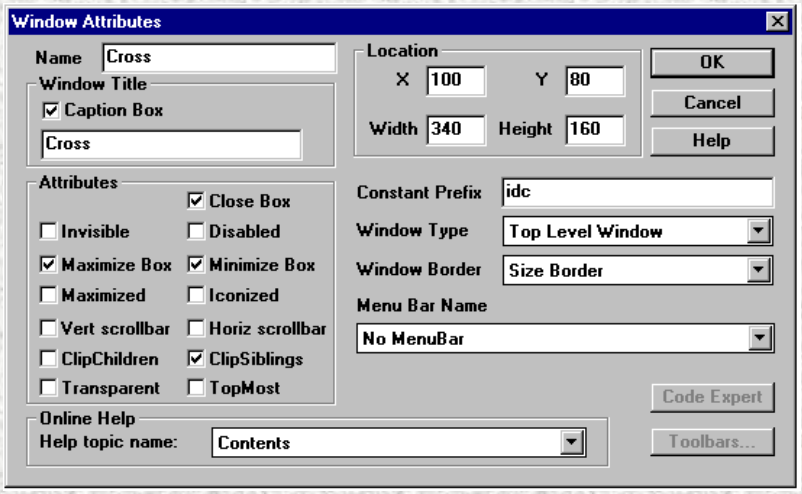

Рисунок 14 - Диалог Атрибутов Окна

Когда выбрана кнопка New, появляется вышеупомянутый диалог. Для этого окна надо только набрать имя 'Cross' и нажать OK.

Когда завершится диалог Window Attribute, автоматически появится Window Editor. Это используется, для помещения элементов управления в окна или для изменения размера окна.

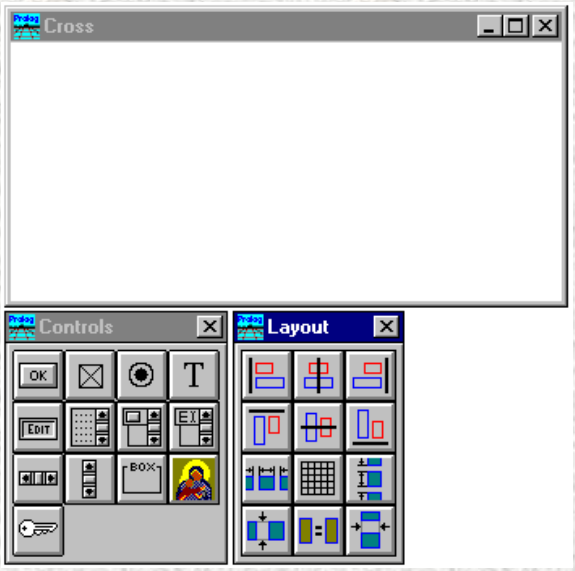

Рисунок 15 - Редактор окна.

2.1.3. Генерирование заданного по умолчанию кода для окна 3

Следующее, что необходимо сделать для окна Cross – сгенерировать заданный по умолчанию исходный код Пролога. Этот исходный код обеспечит:

- код, чтобы создать окно с определенными атрибутами размещения;
- предикат обработчика события, дающий возможность окну реагировать на события (сообщения), которые посылает система GUI (графического интерфейса пользователя) когда с окном что-то происходит.

Для активизации окне Dialog and Window Expert необходимо нажать кнопку Code Expert или Crtl + W, когда окно Cross высветится в окне Project.

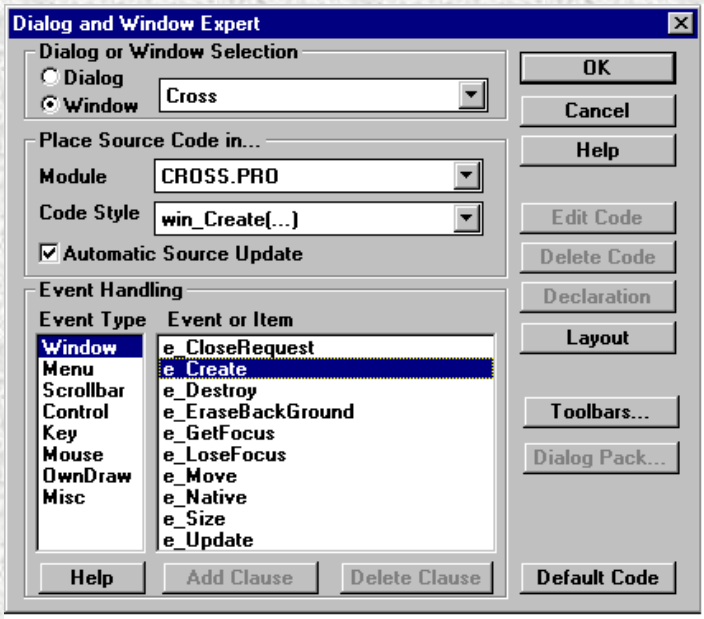

Рисунок 16 - Добавление заданного по умолчанию кода
в окне Dialog and Window Expert.

Когда вызывается окно Dialog and Window Expert, нужно выбрать модуль исходного кода Пролога, содержащий код для окна. Здесь выбрать модуль исходного текста CROSS.PRO, нажав кнопку списка. После этого нажать кнопку Default Code.

При вызове редактора (нажатием кнопки Edit Code), можно увидеть, что следующий код был добавлен в конце выбранного модуля:

```
%BEGIN_WIN Cross
/*************************************************************
Creation and event handling for window: Cross
*************************************************************/
CONSTANTS
%BEGIN Cross, CreateParms, 12:41:53-22.11.1995, Code automatically updated!
win_cross_WinType = w_TopLevel
win_cross_Flags = [wsf_SizeBorder,wsf_TitleBar,wsf_Maximize,
                   wsf_Minimize,wsf_Close,wsf_ClipSiblings]
win_cross_RCT = rct(100,80,440,240)
win_cross_Menu = no_menu
win_cross_Title = "Cross"
win_cross_Help = idh_contents
%END Cross, CreateParms
PREDICATES
win_cross_eh : EHANDLER
CLAUSES
win_cross_Create(Parent):-
win_Create(win_cross_WinType,win_cross_RCT,win_cross_Title,
win_cross_Menu,Parent,win_cross_Flags,win_cross_eh,0).
%BEGIN Cross, e_Create
win_cross_eh(_Win,e_Create(_),0):-!,
%BEGIN Cross, InitControls, 12:41:53-22.11.1995, Code automatically updated!
%END Cross, InitControls
%BEGIN Cross, ToolbarCreate, 12:41:53-22.11.1995, Code automatically updated!
%END Cross, ToolbarCreate
!.
%END Cross, e_Create
%MARK Cross, new events
%BEGIN Cross, e_Size
win_cross_eh(_Win,e_Size(_Width,_Height),0):-!,
IFDEF use_tbar
toolbar_Resize(_Win),
ENDDEF
!.
%END Cross, e_Size
%BEGIN Cross, e_Menu, Parent window
win_cross_eh(Win,e_Menu(ID,CAS),0):-!,
PARENT = win_GetParent(Win),
win_SendEvent(PARENT,e_Menu(ID,CAS)),
!.
%END Cross, e_Menu, Parent window
%END_WIN Cross
```

Система Visual Prolog добавляет различные комментарии к исходному тексту. Система использует эти комментарии, чтобы позже поместить код. Блок, окруженный надписью 'Code automatically updated!' означает, что система будет автоматически модифицировать эту часть кода, когда размещение или параметры изменятся в VDE. По этой причине нельзя добавлять свои собственные изменения для этих разделов, поскольку они будут перезаписаны в течение последующих модификаций с помощью Code Experts. Фактически имеется только несколько из этих разделов, которые будут модифицироваться автоматически. Остальную часть кода можно свободно модифицировать любым образом.

Чтобы какой-либо раздел кода, обновлялся автоматически, в окне Dialog and Window Expert нужно отключить опцию автоматического обновления исходного кода, либо закомментировать этот раздел, так что компилятор не будет его видеть.

Затем надо обеспечить активацию окна. Для этого нужно вставить один дополнительный пункт меню для этой цели и назвать его CrossWin. После этого необходимо использовать Menu и кнопку Edit в окне проекта чтобы редактировать меню Task. Когда появится Редактор Меню, надо выбрать элемент &Test и нажмать кнопку Submenu. После этого надо использовать кнопку New чтобы добавить вход для "CrossWin"

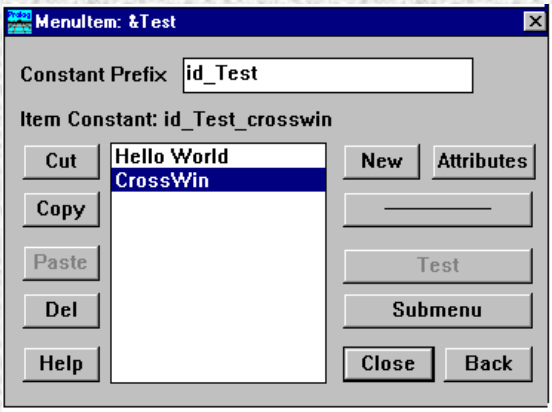

Рисунок 17 - Пункт меню, активизирующий окно Cross

Чтобы добавить функциональность для нового пункта меню надо активизировать Code Expert для окна Task (нажатием кнопки Window с левой стороны окна проекта, выбором Task Window и нажатием кнопки Code Expert), выделить Menu в секции Event Type и id_Test_Crosswin в секции Событие, а затем нажать на кнопку Add Clause.

Рисунок 18 - Добавление предложения в окне Dialog and Window expert.

После нажатия кнопки Add Clause надо нажать кнопку Edit Clause чтобы выбрать редактор. Следующее предложение было добавлено к обработчику события основного окна приложения (окно Task):

```
task_win_eh(_Win, e_Menu(id_Test_crosswin, _ShiftCtlAlt),0):-!,
!.
```

В начале второй строки нужно использовать меню, появляющееся по правой кнопки мыши, выбирая пункты Insert | Predicate Call | Window, Dialog or Toolbar (или нажимая Ctrl+Shift+W).

Появится следующий диалог:

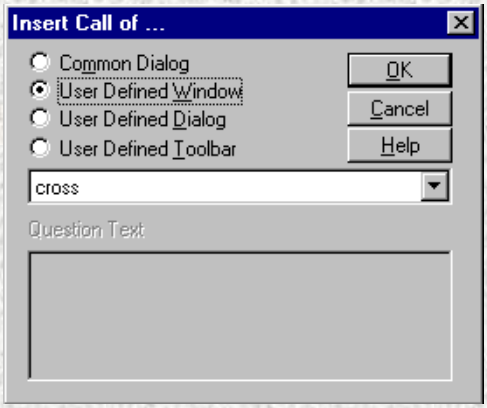

Рисунок 19 - Определение предиката, для вставки – окна, определенного пользователем.

После этого надо выбрать опцию User Defined Window и нажать OK. После этого будет вставлен предикат, вызывающий создание окна Cross.

```
task_win_eh(_WiN, e_Menu(iD_Test_Crosswin, _ShiftCtlAlt),0):-!,
win_cross_Create(_Win),
!.
```

Теперь можно запустить приложение и проверить, что выбор пункта меню Test | Cross-Win выдает небольшое окно с заголовком 'Cross'.

2.1.4. Проверка прихода события в окно 3

Теперь, необходимо попробовать провести небольшой эксперимент. Нужно поместить начало предиката обработчика события в модуле Cross, и напечатать новое первое предложение, которое распечатает событие и затем прервется. Это не будет иметь никаких эффектов на логике окна Cross, но в окне сообщения, можно будет теперь видеть все события, которые посланы обработчику события окон.

```
win_cross_eh(Win, Event, 0):- %              <-- Type these three lines
  write(Win, ":", Event),nl,
  fail.

%BEGIN Cross, e_Create
 win_cross_eh(_Win, e_Create(_),0):-!,
```

После этого можно запустить приложение и выполнить различные действия, такие как создание, перемещение и изменение размеров окна, печать, щелканье мышкой и закрытие окна.

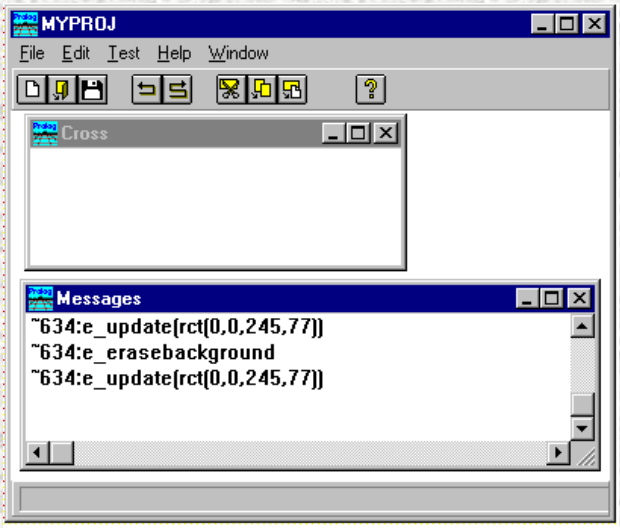

Рисунок 20 - Изучение событий.

Так теперь можно видеть, что события посылаются окну в различных обстоятельствах. Для реагирования на любые из таких событий можно добавить код (обработчик событий).

Пусть приложение должно быть всегда готово к перерисовке окна. Если окно было перекрыто и затем перенесено на передний план, его содержимое должно быть перерисовано.

Всякий раз, когда окно нуждается в перерисовке, обработчик события окна получает событие e_Update, и необходимо добавить код для прорисовки.

Пусть надо нарисовать диагонали окна. Чтобы сделать это надо войти в Dialog and Windows Expert и добавить предложение для события e_Update:

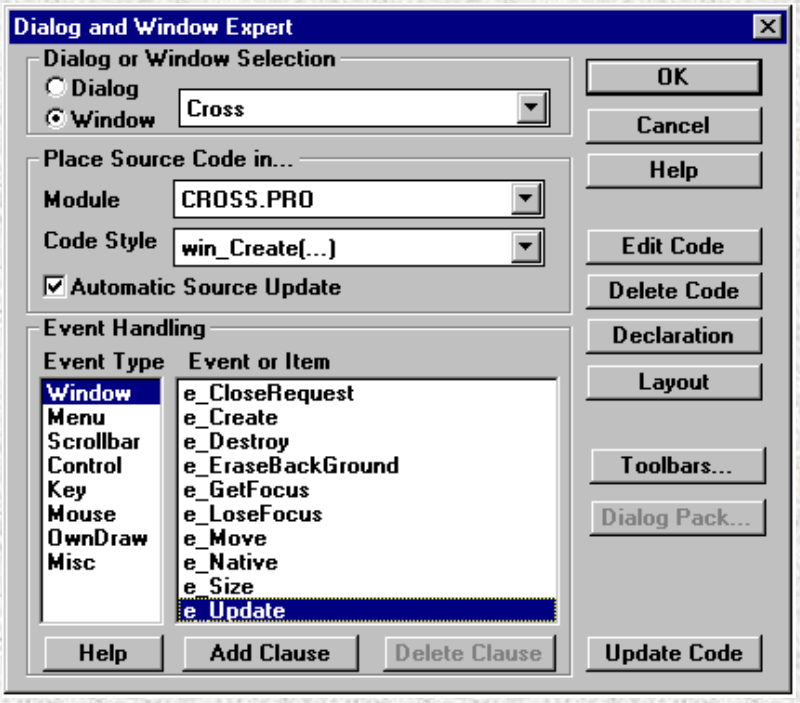

Рисунок 21 - Добавление предложения, для обработки события e_Update.

Чтобы добавить VPI вызов предиката, можно воспользоваться возможностью редактора делать вставку. Команда вставки всегда может быть найдена в меню редактора, но также

имеется комбинация горячих клавиш Shift+Ctrl+V, которая немедленно выдаст список предикатов VPI.

Чтобы нарисовать диагонали, сначала надо найти координаты нижнего правого угла. Чтобы сделать это, получим клиентский прямоугольник с помощью вызова функции win_GetClientRect; этот прямоугольник содержит в двух последних два параметрах ширину и высоту окна, так что код, мог бы выглядеть следующим образом:

```
win_cross_eh(_Win, e_Update(_),0):-!,
RCT = win_GetClientRect(_Win),
RCT = rct(_,_,R,B),
draw_Line(_Win, pnt(0,0),pnt(R,B)),
draw_Line(_Win, pnt(0,B),pnt(R,0)),
!.
```

Наберите здесь эти четыре строки: координаты относительно окна, верхний левый угол (0,0)

После этого можно запустить приложение и выбрать пункт меню Test | CrossWin - появится следующее окно:

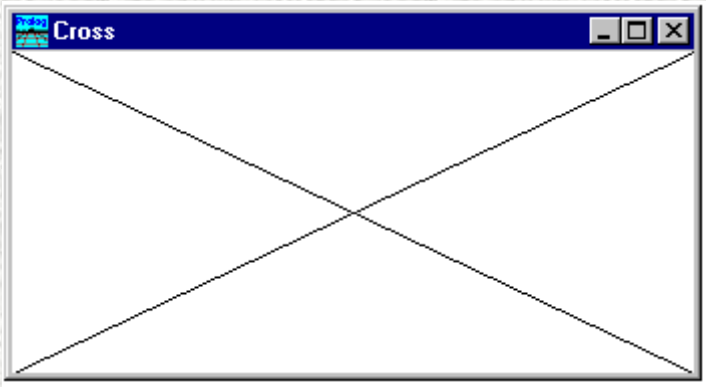

Рисунок 22 - Вид окна Cross.

Программа работает правильно, можно создавать множество экземпляров окна, перемещать окно вокруг, перекрывать его другим окном, и делать его снова видимым. Однако, если изменить размеры окна, то обнаруживается проблема. Окно не перерисовывается корректно. Перерисовывается только появившаяся часть окна. Так происходит потому, что система GUI (графического интерфейса пользователя) пытается рисовать как можно меньше, и для того, чтобы сделать это задается область отсечения, таким образом, что обновляется только новая часть окна. Это – правильное поведение для многих окон таких, как редакторы, изображения и т.д. Но в данном случае этого не достаточно, когда окно изменяет размеры, нам нужна перерисовка всего окна. Один из способов сделать это состоит в том, чтобы вызвать предикат win_Invalidate в событии e_Size. Для этого надо использовать Window and Dialog editor, нажать кнопку Edit Clause для события e_Size (там уже есть некоторый заданный по умолчанию код).

После вставки вызова win_Invalidate, код для события e_Size должен выглядеть так:

```
win_cross_eh(_Win, e_Size(_Width, _Height),0):-!,
win_Invalidate(_Win),
IFDEF use_tbar
toolbar_Resize(_Win),
ENDDEF
!.
```

Если теперь выполнить приложение, то окно будет теперь работает как требовалось.

2.1.5. Другие операции рисования 3

Чтобы проиллюстрировать другие предикаты, которые можно использовать для рисования, необходимо дописать следующий код к коду для события e_Update:

```
draw_Text(_Win, 55, 20, "This is a text in the Cross Window"),
draw_Ellipse(_Win, rct(30,50,60,100)),
draw_PolyGon(_Win,[pnt(130,130),pnt(180,130),pnt(145,100),
                                        pnt(150,150),pnt(170,110)]),
draw_Rect(_Win, rct(250,50,300,100)),
```

Существуют и другие примитивы рисования. Есть возможность установить шрифт, цвет, размер линии, стиль линии, шаблон для заливки и т.д., эти установки описаны в Руководстве VPI или в интерактивном справочнике.

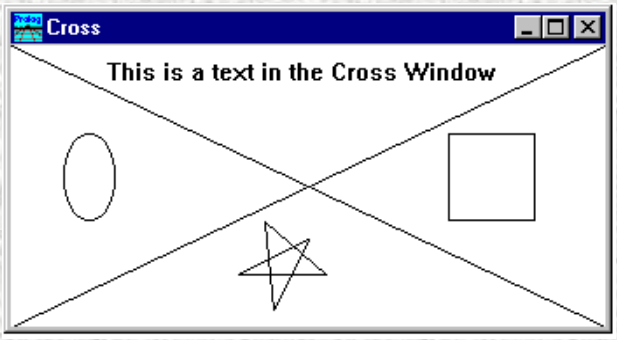

Рисунок 23 - Другие операции рисования в окне Cross.

2.2.Создание окна Sweep 2

Пусть над создать окно со следующими свойствами:

- Окно должно иметь собственное локальное меню.
- Окно должно иметь инструментальную панель справой стороны окна, которая может убираться.
- Мышь будет использоваться для рисования.

2.2.1. Создание модуля SWEEP.PRO 3

Вначале надо создать новый исходный модуль – так как окна полностью независимы, поместим их в отдельные модули. Для для создания модуля SWEEP.PRO надо выбрать модули, нажать кнопку New и ввести имя 'Sweep'. Исходный код для окна Sweep будет помещен в этот модуль.

2.2.2. Создание нового меню (перемещенное меню) 3

Чтобы создать новое меню, нужно выбрать Меню в окне Проекта и нажать кнопку New. Пусть новое меню будет называться 'Sweep menu', константа-идентификатор будет вставлена системой. Нажатие кнопки OK закроет диалог.

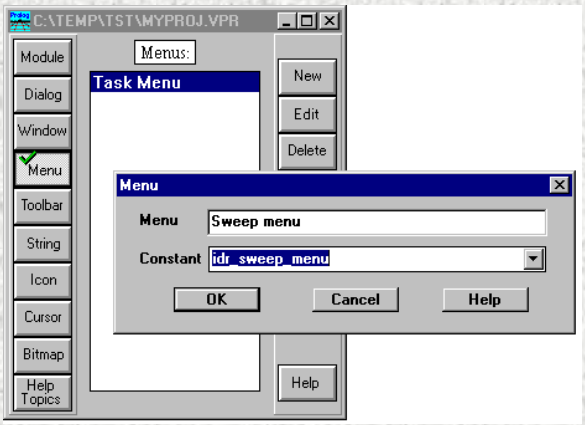

Рисунок 24 - Создание нового меню.

Когда имя "Sweep menu" появлится в списке меню, надо использовать Menu Editor, (активизируемый нажатием кнопки Edit) для добавления одного верхнего элемента меню с именем Sweep, (нажатием кнопки New).

Теперь надо использовать кнопки Submenu и New для создания подменю из одного элемента с названием Clear. По умолчанию задается идентификатор – id_Sweep_clear. Кнопка Attribute изменяет константу для элемента меню на id_clear, поскольку необходимо будет использовать эту константу позже.

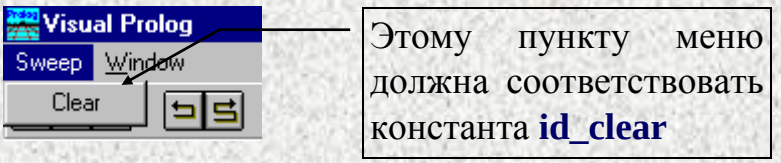

Рисунок 25 - новое меню в тестовом режиме.

2.2.3. Создание изображения 3

Чтобы создать новое изображение надо нажать на кнопку Bitmap окна Project, а затем на кнопку New. Теперь необходимо ввести имя для изображения (mybmp_norm) и желаемый размер. Если константа, идентифицирующая изображение не определена, она будет сгенерирована автоматически. Можно также ввести имя для .BMP файла (если ничего не задано, сгенерируется имя по умолчанию).

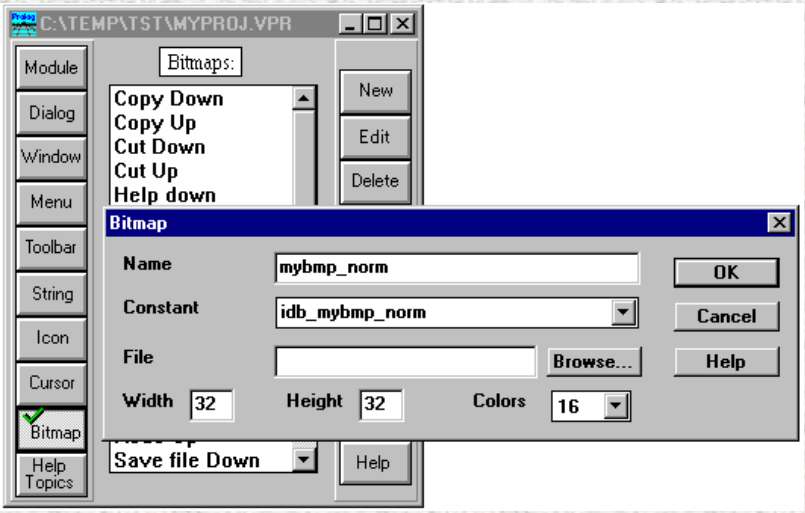

Рисунок 26 - Создание нового изображения из окна Project.

Теперь нужно дважды щелкнуть по только что зарегистрированному имени, чтобы вызвать редактор изображений (иконок, курсоров, изображений). Нарисуйте маленькую картинку, которую можно использовать на кнопке. Графический редактор очень прост в использовании, разработан специально для создания маленьких картинок и курсоров. Этот редактор не очень подходит для больших изображений. Для них придется использовать PAINT.EXE, PBRUSH.EXE или другие специализированные программы.

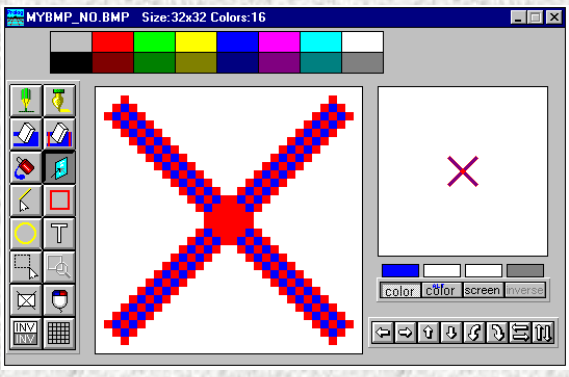

Рисунок 27 - Редактор изображений

Закончив рисовать специальную кнопку, необходимо закрыть графический редактор, и рисунок будет сохранен в файле.

Затем надо создать еще одно изображение (mybmp_down), которое будет использоваться когда кнопка находится в нажатом состоянии. Также нужно удостовериться, что оба

изображения имеют одинаковый размер (можно скопировать изображение mybmp_norm и использовали кнопку редактора – инверсия (invert)).

2.2.4. Создание инструментальной панели 3

Затем надо сформировать инструментальную панель. Для этого надо выбрать инструментальные панели в окне Project и нажать кнопку New. После этого появится новая инструментальная панель. Пусть она называется 'Sweep toolbar'

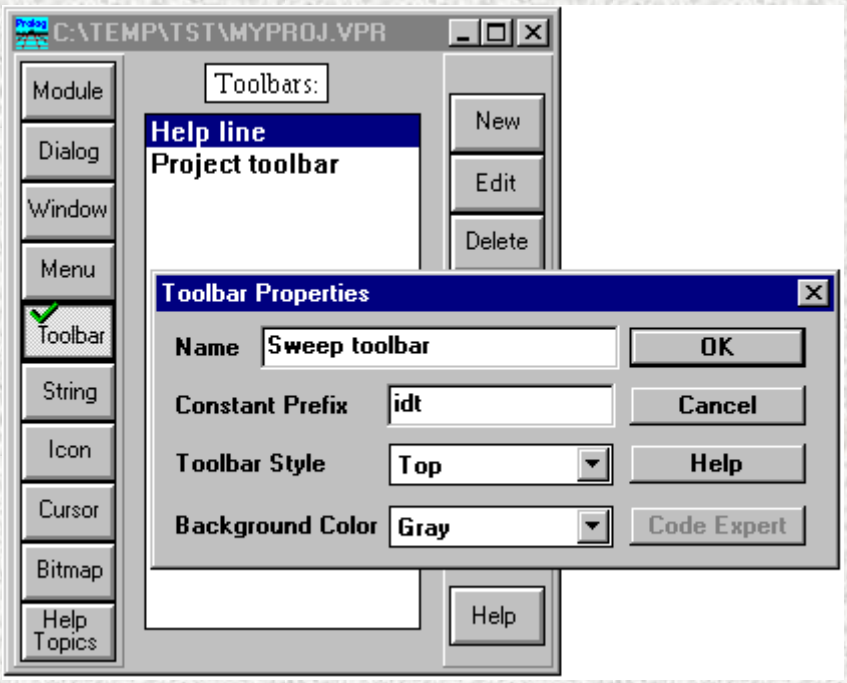

Рисунок 28 - Создание новой инструментальной панели из окна Project.

После нажатия кнопки OK имя новой инструментальной панели появится в списке, а на экране отобразится редактор инструментальной панели (Toolbar Editor) и его окно Control.

Существует пять типов элементов управления, которые можно разместить на инструментальной панели:

- кнопки, которые используются, чтобы вызвать действие (посылка команды меню);
- переключатели, для указания состояния;
- текстовое поле;
- текст, чувствительный к контексту, который обеспечивает объяснение других кнопок;
- выпадающий список, который может отображать список элементов.

Чтобы добавить кнопку в нашу инструментальную панель надо использовать новую кнопку (с изображением кнопки ОК ).

Рисунок 29 - Редактор инструментальной панели с верхней инструментальной панелью.

Чтобы установить свойства для новой кнопки, нужно дважды щелкнуть по ней. Для кнопки может быть задано три изображения, по одному на каждый из трех возможных состояний: отпущена, нажата и запрещена (недоступна).

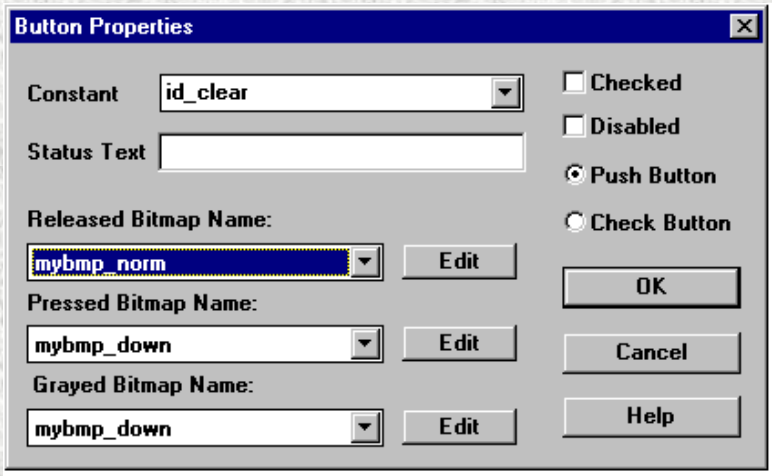

Рисунок 30 - Задание свойств кнопки.

Также необходимо ввести константу для кнопки, чтобы программа смогла реагировать, когда кнопка активизируется. Заданное по умолчанию поведение для инструментальной панели состоит в том, что она посылает то же самое событие, как если бы было активизировано меню. Это означает, что, если создается меню для окна, и вставляется код, чтобы обработать пункты меню, нет необходимости создавать дополнительный код, чтобы добавить кнопки панели, которые соответствуют входам меню. По умолчанию предлагается значение *idt_sweep_toolbar_1*. Сменим его на id_clear так, чтобы это соответствовало предыдущему входу меню.

Нужно обязательно удостовериться, что имена изображений соответствуют разработанным ранее изображениям.

Можно легко изменить стиль инструментальной панели, дважды щелкнув по ней и исправить свойство Toolbar Style на Right.

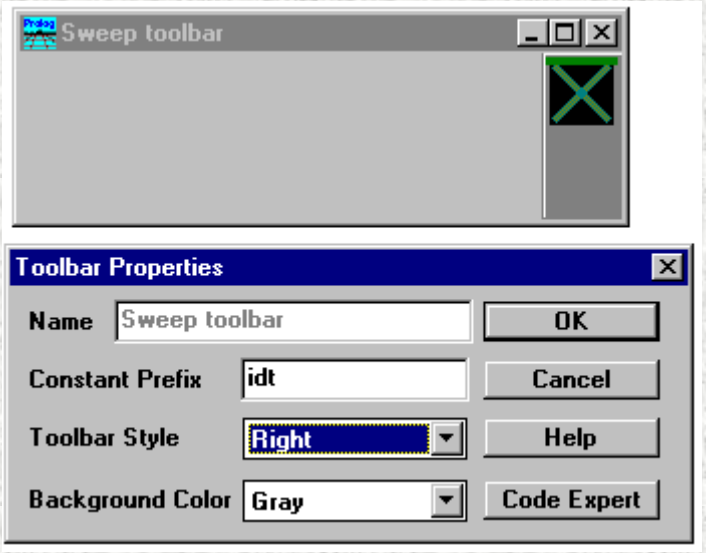

Рисунок 31 - Задание свойств инструментальной панели.

2.2.5. Создание окна 3

Затем надо создать окно. Действия здесь такие же, как и для других компонентов: нужно выбрать Window и нажать кнопку New.

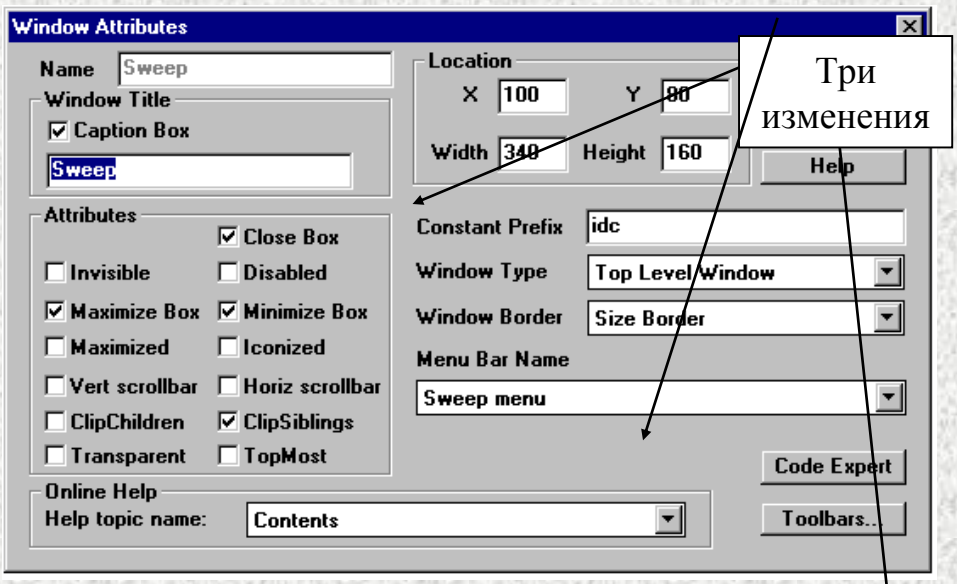

Рисунок 32 - Создание перемещенного окна.

Для окна Sweep нужно определить три свойства:

- Имя окна - Sweep.
- Меню - Sweep menu.
- Инструментальная панель - Sweep Toolbar.

Меню добавляется выбором имени из выпадающего списка Menu Bar Name.

Окно должно быть создано прежде, чем к нему присоединится инструментальная панель. Чтобы сделать это, нажмите кнопку OK, и появится Window Editor. чтобы появился диалог Window Attributes надо дважды щелкнуть по клиентской области окна. Нажав кнопку Toolbars, можно присоединить инструментальную панель Sweep toolbar так, как описано ниже.

2.2.6. Эксперт инструментальной панели (Toolbar Expert) 3

Чтобы добавить инструментальную панель надо вызвать Toolbar Expert нажатием кнопки **Toolbars**, из которого инструментальные панели могут быть добавлены к окну.

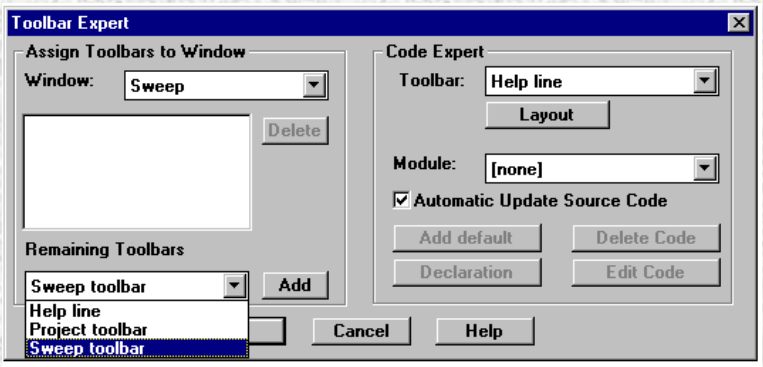

Рисунок 33 - Инструментальные панели Joining к окнам.

Диалог Toolbar Expert выполняет две различных операции: в левой части диалога он назначает инструментальные панели окну; в правой части диалога он позволяет добавить исходный код для управления инструментальной панелью. Как и с окнами, необходимо назначить модуль содержащий код инструментальной панели.

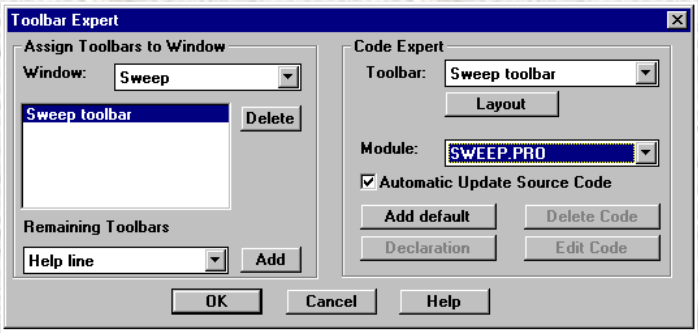

Рисунок 34 - Добавление кода для инструментальной панели.

После связывания 'Sweep toolbar' с окном нужно выбрать исходный модуль для него. Код инструментальной панели должен войти в модуль SWEEP.PRO.

Чтобы сгенерировать необходимый код , надо нажать кнопку Add default. В коде появится следующее:

```
%BEGIN_TLB Sweep toolbar, 17:13:20-17.11.1995, Code automatically updated!
/*****************************************************************
Creation of toolbar: Sweep toolbar
*****************************************************************/
CLAUSES
tb_sweep_toolbar_Create(_Parent):-
IFDEF use_tbar
toolbar_create(tb_right,0x808080,_Parent,
        [tb_ctrl(id_clear,pushb,idb_mybmp_norm,idb_mybmp_down,
                                        idb_mybmp_down,"",1,1)]),
ENDDEF
!.
%END_TLB Sweep toolbar
```

Теперь надо войти в Dialog and Window Expert и создать код для окна Sweep. Для этого нужно Используем Ctrl + W, чтобы просмотреть Dialog and Window Expert.

Рисунок 35 - Dialog and Window Expert создающий код для окна.

Сначала надо выбрать диалог или окно с которым будем работать (в данном случае – окно Sweep). Затем необходимо выбрать исходный модуль, содержащий код (SWEEP.PRO). После этого нажать кнопку Default Code, чтобы добавить базовый код для управления окном.

Теперь, когда окно создано, нужно его активизировать. Надо войти в редактор меню и добавить дополнительный пункт меню с именем Sweep.

Рисунок 36 - Меню Test с добавленным элементом Sweep menu.

Теперь надо войти в Dialog and Window Editor и добавить предложение для создания окна Sweep в момент, когда событие меню Test | Sweep приходит в окно Task.

Рисунок 37 - Добавление предложения для вызова окна Sweep.

Нажатие кнопок Add Clause и Edit Clause помещают в редактор соответствующее предложение, нажатие Shift + Ctrl + W выдаст диалог, где можно вставить вызов создания окна Sweep.

```
Task_win_eh (_Win, e_Menu (id_Test_sweep_window, _ShiftCtlAlt),) 0: -!,
Win_sweep_Create (_Win),
!.
```

Теперь можно выполнить приложение и проанализировать, что случится при активизации пункта меню Test | Sweep.

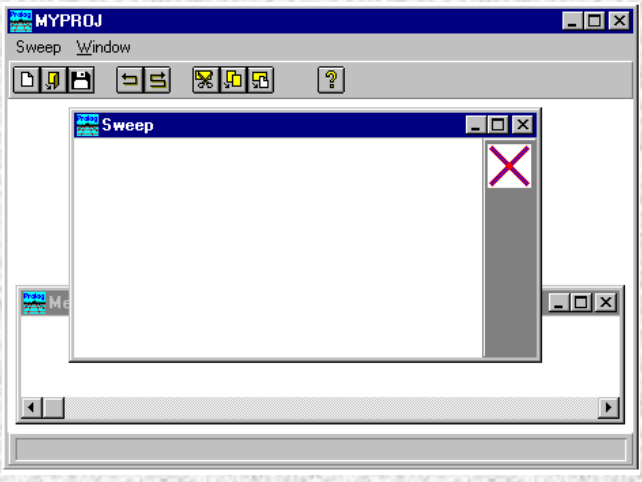

Рисунок 38 - Первый вызов окна Sweep.

В появившемся окне Sweep должна быть инструментальная панель.Когда появилось окно Sweep, меню в окне Task изменилось: к нему добавилось меню окна Sweep. Это – обычное поведение окон MDI системы MS WINDOWS – при изменении активного MDI окна, MDI контейнер добавляет к себе в меню, меню этого окна (поведение меню отлично для различных платформ GUI).

2.2.7. Рисование перемещением мыши 3

Пусть всякий раз, когда нажимается кнопка мыши, программа запоминает координаты точек, над которыми перемещается мышь (для этого, на период от нажатия до отпускания кнопки, захватывается событие **e_MouseMove**) и рисует линию от каждой точки до всех других.

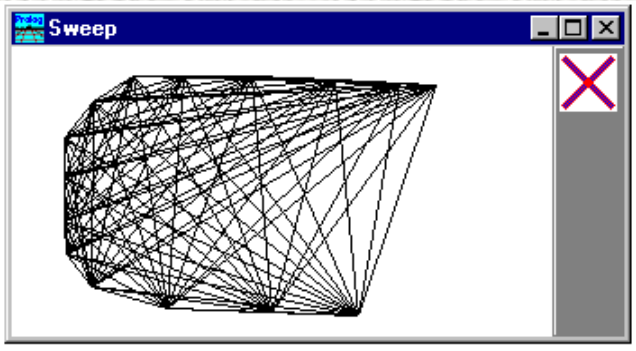

Рисунок 39 - Рисование в окне.

Сначала надо написать предикаты, которые могут рисовать линии от любой точки в списке к каждой из других точек в списке. Нужно напечатать следующие предложения наверху модуля SWEEP.PRO, они должны быть помещены сразу после оператора include.

```
/*************************************************************
    Handling pointlist's
*************************************************************/
PREDICATES
connect(WINDOW,pntlist)
drawlines(WINDOW,PNT,pntlist)
CLAUSES
connect(_,[]):-!.
connect(Win,[FROM | REST]):-
drawlines(Win,FROM,REST),
connect(Win,REST).
drawlines(_,_,[]):-!.
drawlines(Win,pnt(X,Y),[pnt(TOX,TOY) | REST]):-
draw_Line(Win,pnt(X,Y),pnt(TOX,TOY)),
drawlines(Win,pnt(X,Y),REST).
```

Рассмотрим логику обработки события. Нужна база данных для хранения координат мыши, назовем эту ее "point". Так как события *e_MouseMove* поступают независимо от изменения состояния кнопок мыши (событие *e_MouseMove* вызывается перемещением мыши), надо также зафиксировать их нажатие. Также нужно учесть возможность создание множества окон, в каждом из которых можно рисовать независимо. Это можно сделать запоминая дескриптор окна, гарантировано уникальный для каждого окна, вместе с другими данными, необходимыми для базы данных.

Напечатаем эти строки вначале модуля sweep.pro, после оператора include:

```
**********************************************************
    Факты для выполнения перемещения
**********************************************************/
DATABASE - apl
    nocopy point(WINDOW,PNT)
    mouse_isdown(WINDOW)
```

Для выполнения поставленной задачи надо обрабатывать следующий четыре события, код для которых должен быть добавлен в модуль SWEEP. PRO, используя Dialog и Window Expert.

2.2.7.1.Событие e_MouseDown 4

```
win_sweep_eh(_Win,e_MouseDown(_PNT,_ShiftCtlAlt,_Button),0):-!,
retractall(mouse_isdown(_Win)),
retractall(point(_Win,_)),
assert(mouse_isdown(_Win)),
win_Invalidate(_Win),
!.
```

Введите эти четыре строки между двумя ! знаками.

2.2.7.2.Событие e_MouseMove 4

```
win_sweep_eh(_Win,e_MouseMove(_PNT,_ShiftCtlAlt,_Button),0):-!,
mouse_isdown(_Win),
assert(point(_Win,_PNT)),
draw_Pixel(_Win,_PNT,color_Black),
!.
```

2.2.7.3.Событие e_MouseUp Event 4

```
win_sweep_eh(_Win,e_MouseUp(_PNT,_ShiftCtlAlt,_Button),0):-!,
retractall(mouse_isdown(_Win)),
win_Invalidate(_Win),
!.
```

2.2.7.4.Событие The e_Update Event 4

```
win_sweep_eh(_Win,e_Update(_Rct),0):-!,
win_Clear(_Win, color_White),
findall(X,point(_Win,X),List),
connect(_Win,List),
!.
```

2.2.8. Оперирование панелью инструментов 3

Пусть в программе обрабатывается нажатие кнопки Clear на панели инструментов. Когда нажата кнопка на панели инструментов, генерируется событие e_Menu. Нужно использовать Dialog иWindow Expert, чтобы добавить код для id_clear события меню:

```
win_sweep_eh(_Win,e_Menu(id_clear,_ShiftCtlAlt),0):-!,
    retractall(mouse_isdown(_Win)),
    retractall(point(_Win,_)),
    win_Clear(_Win, color_White),
    win_Invalidate(_Win),
    !.
```

Теперь можно открыть панель инструментов **Sweep** и проверить, что кнопка имеет постоянный идентификатор id_clear, который может быть выполнен, если выбирать пиктограмму панели инструментов (слева от окна **Project)** двойным нажатием **Sweep toolbar** и затем кнопки.

Вызов ***win_Clear*** - один из способов очистить окно.

2.2.9. Очищение 3

Если пользователь закрывает окно, должна проиизводится очистка фактов, которые были записаны для этого окна. Для этого имеется специальное событие ***e_Destroy***. ***e_Destroy*** – последнее событие, которое обрабатывает окно.

```
win_sweep_eh(_Win,e_Destroy,0):-!,
    retractall(mouse_isdown(_Win)),
    retractall(point(_Win,_)),
    !.
```

2.2.10. Изменение указателя мыши 3

Теперь нужно пробовать изменить курсор мыши для окна **Sweep**. Указатели зарегистрированы в окне Project, так что Вы должны выбрать пиктограмму Cursor на панели инструментов слева от окна **Project**, и нажать кнопку **New**, чтобы создать новый указатель с именем 'Sweep'.

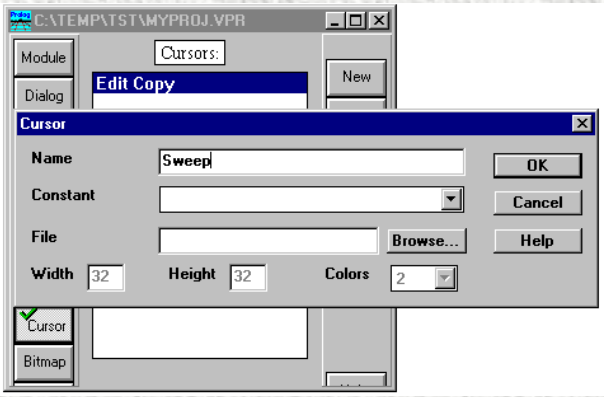

Рисунок 40 - Создание нового указателя

Когда имя нового указателя появилось в списке, надо щелкнуть на нем два раза, чтобы открыть редактор указателей.

Можно использовать четыре цвета для указателя: черный, белый, прозрачный (Transparent) (= экрана) и инверсный (Inverse). Попробуйте воспользоваться всеми четырьмя при создании курсора, чтобы в будущем знать, какие возможности существуют. Однако, указатель мыши обычно создается с использованием только цвета экрана (Screen) и инверсного (Inverse).

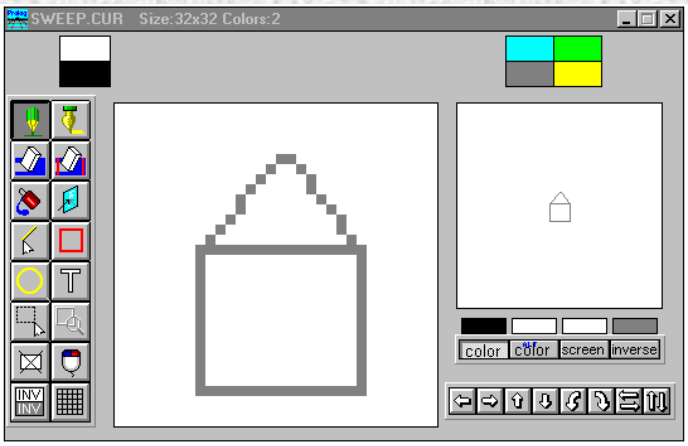

Рисунок 41 - Создание нового указателя

Чтобы увидеть, как выглядит созданный вами указатель, выберите **Tools | Test cursor**:

Рисунок 42 - Меню для проверки указателя

2.2.11. Установка горячей точки для указателя 3

Горячая точка - пиксель на указателе, где позиция нажатия мыши будет зарегистрирована. Для установки «горячей точки» нужно выбрать Set Cursor Hot Spot из меню **Edit**:

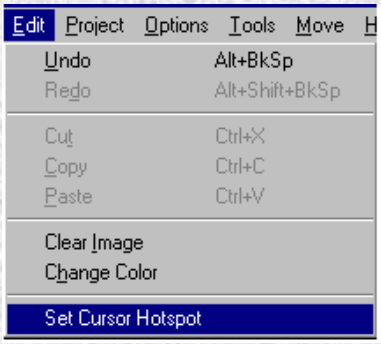

Рисунок 43 - Меню для установки «горячей точки» указателя мыши.

Когда этот пункт меню выбран, надо щелкнуть на картинке в том месте, где вы хотите установить «горячую точку»:

Вызвав ***cursor_Set,*** можно изменить указатель мыши для окна. window.

```
win_sweep_eh(_Win,e_Create(_),0):-!,
    cursor_Set(_Win, idc_sweep),
```

Для добавления вызова ***cursor_Set***, необходимо Нажав **Ctrl+W** сделайте активным окно Dialog и Window Expert, затем нажмите **Edit Clause** для ***e_Create*** события окна Sweep, нажать **Shift+Ctrl+V** , выделить второй аргумент (***ResId***) в вызове предиката ***cursor_Set***, затем выбрать **Insert | Resource identifier | Cursors** чтобы добавить имя указателя.

Теперь можно запустить приложение и посмотреть, как теперь себя ведет мышь.

2.2.12. Создание всплывающего меню 3

Во многих современных приложениях правая кнопка мыши используется для вызова всплывающего меню для каждого окна. Всплывающие меню разрабатываются в редакторе меню как и выпадающие меню. Вы должны создать новое (pop-up) меню с именем 'Sweep Pop-up' и определить следующие четыре пункта:

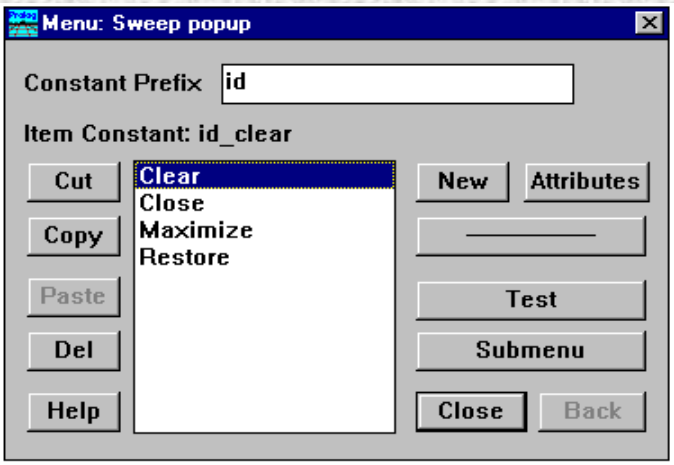

Рисунок 44 - Создание всплывающего меню.

Константа, используемая для **Clear**, должна быть та же, что определена для **Clear** в меню **Sweep** и в панели инструментов **Sweep** (id_clear).

Для добавления дополнительного clause для события ***e_MouseDown***, нужно использовать редактор как показано ниже. Этот clause проверяет, не была ли нажата правая кнопка мыши.

```
win_sweep_eh(_Win,e_MouseDown(PNT,_,mouse_button_right),0):-!,
menu_PopUp(_Win,res_menu(idr_sweep_popup),PNT, align_Right).
win_sweep_eh(_Win,e_MouseDown(_PNT,_, mouse_button_left),0):-!,
.........
```

Теперь можно запустить программу и использовать правую кнопку мыши.

Пункт меню **Clear** работает, потому что всплывающее меню генерирует такое же событие ***e_Menu*** как и выпадающие меню и панели инструментов.

Когда вы попытаетесь добавить код для трех новых пунктов меню, вы обнаружите, что их нет среди пунктов меню окна Sweep, как это зарегистрировано в Dialog и Window Expert! Это потому, что нельзя зарегистрировать тот факт, что всплывающее меню будет использоваться для окна. Но можно записать три clauses для этих событий вручную, так что соответствующее функциональное назначение может быть осуществлено. Для этого нужно добавить на соответствующее место в предикаты, обрабатывающие события, следующие три clauses:

```
win_sweep_eh(_Win,e_Menu(id_close,_ShiftCtlAlt),0):-!,
win_Destroy(_Win).
win_sweep_eh(_Win,e_Menu(id_maximize,_ShiftCtlAlt),0):-!,
win_SetState(_Win, [wsf_Maximized]).
win_sweep_eh(_Win,e_Menu(id_restore,_ShiftCtlAlt),0):-!,
win_SetState(_Win, [wsf_Restored]).
```

Кроме того, можно обнаружить, что нажатие правой кнопки мыши конфликтует с поведением прежде определенным обработчиком события ***e_MouseUp***. Если это так , необходимо изменить clause обработчика события ***e_MouseUp***, чтобы он реагировал только на нажатие левой кнопки мыши

```
win_sweep_eh(_Win,e_MouseUp(_PNT,_,mouse_button_left),0):-!,
```

или изменить порядок clauses.

2.3.Изменение цветов 3

При рисовании линии или пикселя используется выбранное текущее перо, имеющее толщину, стиль (solid or hollow) и цвет. Текущее состояние пера можно получить, вызвав ***win_GetPen,*** изменить свойства пера можно, вызвав ***win_SetPen***. Для выбора нового цвета можно использовать диалог ***dlg_ChooseColor,*** который возвращает новый цвет, выбранный

пользователем. Пусть надо добавить пункт меню Color, изменяющий цвет. Этому пункту меню будет соответствовать константа **id_color:**

```
win_sweep_eh(_Win,e_Menu(id_color,_ShiftCtlAlt),0):-!,
Pen=win_GetPen(_Win),
Pen = pen(Penwidth,PenStyle,Color),
NewColor = dlg_ChooseColor(Color),
win_SetPen(_Win,pen(Penwidth,PenStyle,NewColor)),
win_Invalidate(_Win).
```

2.4.Использование таймера – Clock Vindow 3

Пусть нужно создать окно, в котором выводится текущее время Для этого нужно создать новый исходный модуль CLOCK.PRO, добавить пункт **Clock** Window в меню **Task** подменю **Test**, создать новое окно CLOCK, добавить встроенный код для окна **Clock** в модуле CLOCK.PRO, добавить clause для окна **Task**, чтобы вывести окно Clock при выборе **Test | Clock Window**.

После этого можно запустить программу и посмотреть, создается ли окно при выборе **Test | Clock Window**.

Теперь нужно возвратиться к Dialog и Window Expert и добавить clause для события ***e_Update*** в окне **Clock**.

2.4.1. Использование Таймеров 3

Можно в любое время создавать новый таймер для окна при помощи вызова предиката ***timer_Set(Window, Milliseconds)***. Если предикат ***timer_Set*** вызван, через определенные интервалы времени будет происходить событие ***e_Timer***

2.4.2. Событие e_Update для Окна Часов 3

Пусть когда происходит событие обновления, время выводится в центре окна. Чтобы получить время надо воспользоваться предиком ***time.*** Сначала три целых, возвращенных как время, переводятся в строку предикатом format. Строка выводится в центре окна, для чего вызывается ***draw_TextInRect***, которому передается прямоугольник, в центре которого должен быть размещен текст.

```
win_clock_eh(_Win,e_Update(_),0):-!,
win_Clear(_Win,color_White),
RCT=win_GetClientRect(_Win),
time(Hours,Minutes,Seconds,_),
format(Str,"%02:%02:%02",Hours,Minutes,Seconds),
```

```
draw_TextInRect(_Win, RCT, Str, -1,[dtext_center,dtext_vcenter,
dtext_singleline]),
```

Событие e_Size для Окна Часов
Пусть окно перерисовывается, когда оно изменяет свой размер.

```
win_clock_eh(_Win,e_Size(_Width,_Height),0):-!,
win_Invalidate(_Win).
```

2.4.3. Событие e_Create для Окна Часов 3

В событии ***e_Create*** нужно запустить таймер. Величина интервала 1000 миллисекунд означает, что событие от таймера бутет поступать каждую секунду.

```
win_clock_eh(_Win,e_Create(_),0):-!,
_NewTimerId =timer_Set(_Win, 1000).
```

2.5.4. Событие e_Timer для Окна Часов 3

В событии e_Timer нужно перерисовывать окно. Оно заставит обновляться окно, если оно не минимизируется или не закрывается другим окном. Событие e_Timer находится в группе Misc. в Dialog and Window Expert.

```
win_clock_eh(_Win,e_Timer(_TimerId),0):-!,
win_Invalidate(_Win),
```

В реальном приложении нежелательно перерисовывать целое окно, поскольку это может занять слишком много времени.

Таймера являются ограниченным системным ресурсом. В реальном приложении идентификатор таймера, возвращаемый возвращенный ***timer_set*** должен быть сохранен для использования в событии ***e_destroy*** в качестве аргумента на ***timer_Kill/1***.

2.6. Окно с рисунком 2

Пусть нужно создать окно, в котором отображается рисунок. Рисунки в этом примере загружаются в файлы побитового отображения, и обрабатывать побитовые отображения легко используя VPI. Для этого надо создать новый исходный модуль PICTURE.PRO, добавить пункт меню **Picture** window в меню **Task** под меню **Test**, создать новое окно PICTURE, добавить встроенный код для окна **Picture** в модуле PICTURE.PRO, добавить clause для окна **Task**, чтобы вывести окно **Picture** при выборе **Test | Picture Window**.

После этого можно запустить программу и посмотреть, создается ли окно при выборе **Test | Picture Window**.

После этого нужно возвратиться к Dialog и Window Expert и добавить clause для события ***e_Update*** в окне **Clock**. Затем добавить следующий код в clause e ***e_Update***:

```
win_picture_eh(_Win,e_Update(_),0):-!,
/* Remember to use the correct path to the .BMP file */
Picture = pict_Load("C:\\VP\\BIN\\WIN\\PROLOG.BMP"),
ClntRCT=win_GetClientRect(_Win),
pict_GetSize(Picture, Width, Height, _Size),
pict_Draw(_Win,Picture, ClntRCT, rct(0,0, Width,Height),rop_SrcCopy),
pict_Destroy(Picture),
```

Этот clause нуждается в объяснении:

Сначала предикат ***pict_Load*** вызывается с именем файла, который содержит отображение. Предикат возвращает указатель на изображение, которое загружено в память. Можно использовать **Edit | Insert** меню, чтобы связать filename с любой bmp-файлом на диске

Можно выводить изображение в окне любого размера, так что необходимо знать размер как окна так и изображения.

Наконец вызывается предикат ***pict_Draw*** с указателями на окно и рисунок и с двумя размерами (окна и рисунка). Последний параметр в ***pict_Draw*** определяет как рисунок сочетается с пикселями уже в окне.В программе определено, что нужно скопировать изображение в окно (заменить существующие пикселы).

Наконец, уничтожается изображение, что оно не занимало место в памяти.

2.7. Создание окна, отображающего дерево 2

Пусть нужно создать окно, которое отображает древовидную структуру. Дерево рисуется инструментом, находящимся в каталоге VPI\INCLUDE, и это - один из файлов, которые могут быть включены в модуль VPITOOLS.PRO. Прежде, чем создать окно, отображающее дерево, нужно проверить, что в Application expert на страничке VPI Options опция Tree Tool доступна.

Теперь нужно выполнеить следующее: создать новый исходный модуль TREE.PRO, добавить пункт меню **Tree Window** в меню **Task** под меню **Test**, создать новое окно TREE, добавить встроенный код для окна **Picture** в модуле PICTURE.PRO, добавить код в модуль TREE.PRO, затем необходимо выбрать ***tree_Create*** как стиль генерации кода для Code Expert.

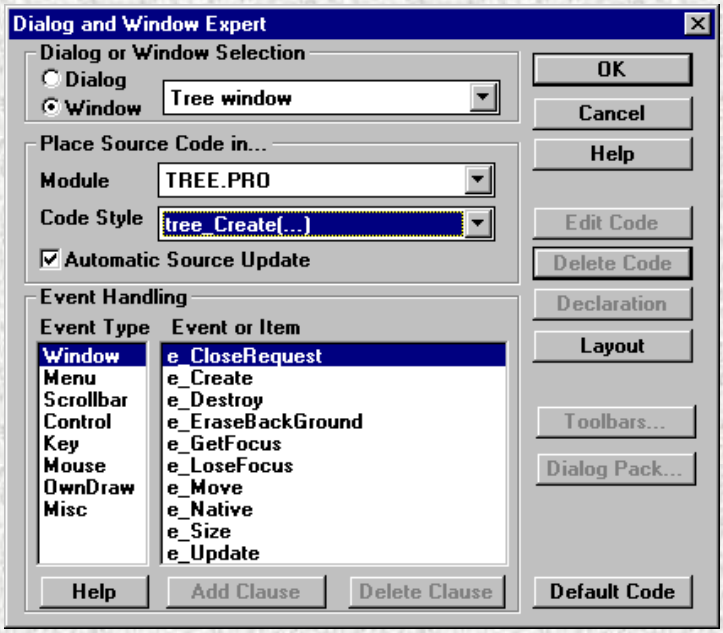

Рисунок 45 - Добавление кода по умолчанию для окна, отображающего дерево.

После этого нужно добавлять новый пункт в меню **Task** и создавать clause для вывода окна **Tree**.

В Прологе очень легкое создать дерево из различных структур данных, как например, списки файлов на жестком диске или записи в базе данных.

```
win_tree_window_Create(Parent):-
IFDEF use_tree
Tree = tree("0",unmarked,
 [tree("3",unmarked,[],0),
  tree("2",unmarked,
    [tree("4",unmarked,[],0)],0),
     tree("1",unmarked,[],0)
    ],0),
TreeDir = tree_dirright,
Font = font_Create(ff_Fixed,[],10),
TreeWinInfo = tree_WinInfo(Font,TreeDir,[]),
tree_Create(win_tree_window_WinType,win_tree_window_RCT,
    win_tree_window_Title,win_tree_window_Menu,Parent,
    win_tree_window_Flags,win_tree_window_eh,0,Tree,
    TreeWinInfo,[]),
ENDDEF
!.
```

Даже если окно Дерева является специальным типом окна, код имеет ту же структуру как код окна, которые Вы создаете сами: это создание, и стандартное событие-указатель.

Пустьбудет созможность реагировать на щелчок мыши на вершине. Для этого надо добавить следующий clause для события ***e_MouseUp*** .

```
win_tree_window_eh(_Win,e_MouseUp(_PNT,_ShiftCtlAlt,_Button),0):-!,
SelectorText=tree_GetSelectedNode(_Win),
dlg_Note("Node Selected",SelectorText),!.
```

2.8. Создание Window Editor 2

Есть очень гибкий API определенный для редактора VPI, так, что приложения Visual Prolog могут модифицировать и использовать этот редактор по желанию пользователя. Редактора поддерживает выделение цветом и области гипертекста, и эти можно использовать в ваших программах.

Для создания редактора необходимо создать новый исходный модуль EDWIN.PRO, создать новое окно EDITOR, определить код, который должен содержаться в модуле EDWIN, выбрать ***edit_Create*** как стиль генерации кода для Code Expert, добавить пункт меню **Edit Window** в меню **Test** и т.д.

После этого можно запустить приложение, открыть окно редактора и проверить, можно ли теперь отредактировать какой-нибудь текст, используя редактор.

При вызове создания окно редактора аргументы имеют установленные по умолчанию значения:

```
win_editor_Create(Parent):-
Text = "",
Font = font_Create(ff_Fixed,[],10),
ReadOnly = b_false, Indent = b_true, InitPos = 1,
edit_Create(win_editor_WinType,win_editor_RCT,win_editor_Title,
 win_editor_Menu,Parent,win_editor_Flags,Font,ReadOnly,
 Indent,Text,InitPos,win_editor_eh),
!.
```

Можно изменить этот код, например, чтобы загрузить определенный текст в редактор:

```
file_str("D:\\TEMP\\TST\\MYPROJ.CON" ,Text),
```

2.8.1. Щелчок по обозначению 3

Пусть надо написать предикат, позволяющий при двойном щелчке на слове выводить примечание к этому слову.

Для этого надо воспользоваться тремя предикатами: ***edit_SelectWord*** возвращает b_True, если есть выделенный текст, ***edit_GetSelection*** возвращает начальную и конечную позиции выделенного текста, и ***edit_GetText*** возвратит выбранный текст из редактора.

```
win_editor_eh(_Win,e_MouseDbl(_PNT,_ShiftCtlAlt,_Button),0):-!,
WordSelected = edit_SelectWord(_Win),
```

```
WordSelected = b_True,
edit_GetSelection(_Win,Pos1,Pos2),
Text = edit_GetText(_Win),
NoOfBytes = Pos2-Pos1,
substring(Text,Pos1,NoOfBytes,SubString),
format(Msg,"Got the token: >%<",SubString),
dlg_Note(Msg).
```

2.8.2. Вставка текста в редактор 3

Для вставки текста можно воспользоваться предикатом ***edit_PasteStr*** .

Пусть при наборе комментария '/*' завершение комментария '*/' будет вставляться автоматически. Чтобы сделать это, нужнопроверять вводимые символы. Когда пользователь нажимает клавишу, событие ***e_Char*** передается активному окну. Когда обнаружен символ '/', устанавливается флажок, если за ним следует символ '*', то вставляется завершение комментария '*/', в противном случае флаг обнуляется. Чтобы определить позицию курсора в редакторе, надо использовать предикат ***edit_GetPos,***. После этого пусть выделяется вставленные символы.

```
win_editor_eh(_Win,e_Char('/',_),0):-!,
not(commentflag(_Win)), assert(commentflag(_Win)),
fail.
win_editor_eh(_Win,e_Char('*',_),0):-!,
retract(commentflag(_Win)),
Pos = edit_GetPos(_Win),
edit_PasteStr( _Win, Pos, "* */" ),
Pos1 = Pos+1, Pos2 = Pos1 + 3,
edit_SetSelection(_Win, Pos1, Pos2),
NewPos = Pos + 1,
edit_GotoPos(_Win,NewPos),!.
win_editor_eh(_Win,e_Char(_,_),0):-!,
retract(commentflag(_Win)),
fail.
```

Две clauses, приведенные выше, которые просто манипулируют значением флага, также возвращают fail, так, что сам редактор сам может делать нормально обрабатывать эти сообщения (то есть, вставку символов, обновление окна, и т.п.), как если бы этих clauses не было.

Для того, чтобы определять базу данных использованную выше, включите следующий код в начало файла:

```
DATABASE - comment
commentflag(WINDOW)
```

2.8.3. Сохранение Текущего Текста 3

Текст в редакторе можно сохранить на диске, вызвав ***edit_GetText***, сопровождающийся вызовом ***file_str***.

2.9. Работа с буфером 2

Буфер используется для обмена информацией между приложениями. Приложения должны уметь помещать информацию в буфер и извлекать информацию из буфера. В Visual Prolog возможно вставлять в буфер текст, изображения и специальные форматы Пролога. Чтобы поместить в буфер текст и изображение, нужно просто вызывать ***cb_PutString*** или ***cb_PutPicture***, а извлечь их из буфера можно вызывая ***cb_GetString*** или ***cb_GetPicture***.

Пусть при нажатии клавиши 'c' в буфер копируется изображение окна Sweep, и возвращается текущее изображение из буфера при нажатии 'v'.

Прежде, чем изображение разместится в буфере, оно должно “захватываться” из окна в котором оно было нарисовано. Один из способов сделать это - использовать ***pict_Open***, затем выполняя операции рисования снова, так что они собираются в изображение, которое возвращается после вызова ***pict_Close***. Предикат ***sweep_Draw*** просто выполняет поиск всех точек, и затем называет предикат **connect** со списком точек, таким образом перерисовывая изображение, так чтобы копировалась последовательность между ***pict_Open*** и ***pict_Close***.

```
win_sweep_eh(_Win,e_Char('c',_),0):-!, % Copy window
RCT=win_GetClientRect(_Win),
PictWin = pict_Open(RCT),
findall(X,point(_Win,X),List),
connect(PictWin,List),
Picture = pict_Close(PictWin),
cb_PutPicture(Picture),
pict_Destroy(Picture).
win_sweep_eh(_Win,e_Char('v',_),0):-!, % Paste picture
Picture=cb_GetPicture(),
pict_Draw(_Win,Picture,pnt(0,0),rop_SrcCopy),
pict_Destroy(Picture).
```

Из-за того, что копируется побитовое изображение, а не список точек, вставленное изображение не будет перерисовано при вызове обработчика события ***e_Update***.

2.10. Печать 2

Печать инициируется вызовом ***print_StartJob*** и завершается ***print_EndJob***. Каждая страница должна “окружаться” парой ***print_StartPage*** - ***print_EndPage***. Все предикаты рисования работают как обычно, за исключением того это они должны передавать указатель окна распечатки как аргумент Window:

```
PRINTWIN=print_StartJob("Printing Sweep drawing"),
print_StartPage(PRINTWIN),
...Draw to the Printwindow
print_EndPage(PRINTWIN),
print_EndJob(PRINTWIN).
```

Обычно разрешение принтера значительно выше чем на экране монитора, так что если Вы просто скопируете ваш рисующий картинку код, вы получите скорей всего очень небольшие изображения.

Решение этой проблемы – преобразующие предикаты. GUI система может преобразовать координаты из одной системы в другую, масштабируя изображение. Масштабирование можно выполнить, вызывая предикаты ***set_MapMode*** и ***set_MapScale***.

2.11. Добавление элементов управления 2

Пусть надо добавить в окно с часами кнопку, после нажатия которой начинается отсчет времени либо время останавливается и Check box, чтобы определить, должно ли время отображаться или нет.

Для этого необходимо поместить эти элементы управления в окно. В окне Project нужно выбрать Window, выбрать нужное окно и нажать кнопку **Edit**.

Когда Window Editor открыт, можно нажать на элементе управления с панели инструментов **Controls**, и перетащить его в окно. Если выделить прямоугольник, элемент управления установится в выделенном прямоугольнике. Если отпустить кнопку мыши, активизируется диалог, в котором можно определить атрибуты для выбранного элемента управления. Важно ввести имя элемента управления.

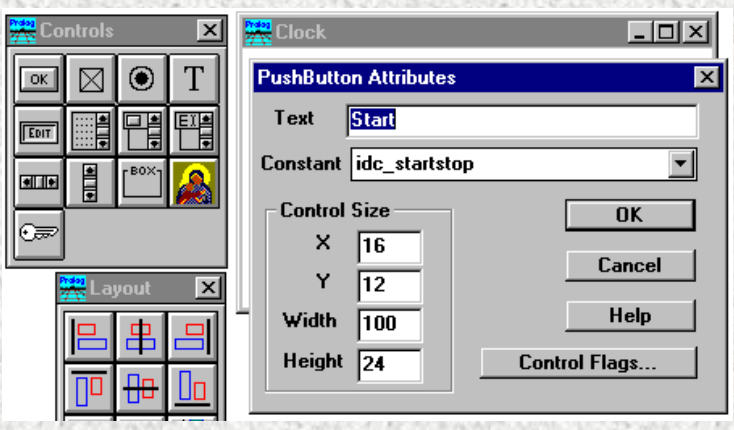

Рисунок 46 - Добавление элемента управления в Окно Часов.

Нужно создать два элемента управления со следующими названиями и константами:
Push button: Title = 'Start', Constant = 'idc_startstop'
Check box: Title = 'Show Time', Constant = 'idc_show_time'

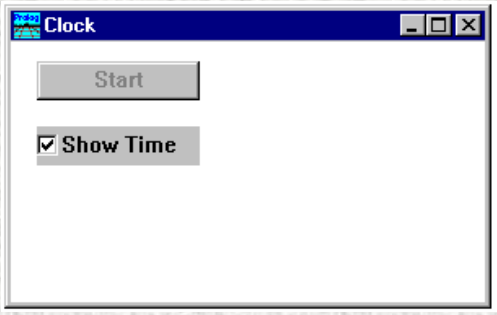

Рисунок 47 - Окно Clock с элементами управления.

В код для события ***e_Create*** для окна Clock были добавлены следующие строки:
win_CreateControl(wc_PushButton,rct(6,9,106,33),"Start / Stop",_Win, [ctl_Group,ctl_Tab-Stop], idc_startstop),
win_CreateControl(wc_CheckBox,rct(6,44,106,68),"Show Time", _Win,[ctl_Group,ctl_Tab-Stop, ctl_Auto],idc_show_date),

Чтобы наделить элементы управления функциональными возможностями, в окне Dialog and Window Expert в колонке Event Type нужно выделить 'Control'. Затем можно добавлять события для элементов управления в окне.

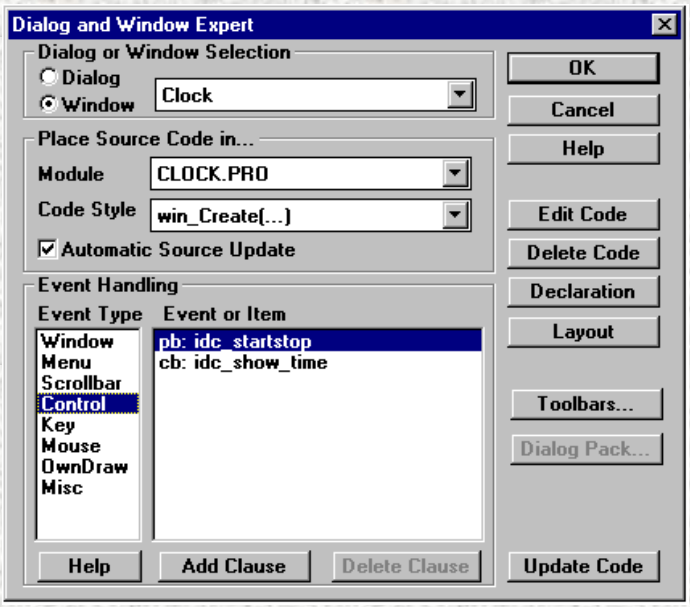

Рисунок 48 - Добавление событий для элементов управления .

Вначале надо объявить два предиката (***startTime*** и ***stopTimer***) и локальную базу данных (**clock**):

```
DATABASE - clock
timer(Window,LONG Id)
PREDICATES
startTimer(Window)
stopTimer(Window)
CLAUSES
startTimer(Win):-
TimerId = timer_Set(Win, 1000),
assert(timer(Win,TimerId)).
stopTimer(Win):-
retract(timer(Win,TimerId)),!,
timer_Kill(TimerId).
```

Теперь нужно модифицировать событие ***e_Create*** для окна Часов и удалить старый код для запуска таймера, так что таймер должен запускаться вручную после нажатием кнопки.

Теперь нужно выбрать clause события для кнопки (**Edit Clause**), и добавить следующий код для запуска и остановки таймера:

```
win_clock_eh(_Win,e_Control(idc_startstop,_,_CtrlWin,_CtlInfo),0):-
Title = win_GetText(_CtrlWin),
Title = "Start",
startTimer(_Win),
win_SetText(_CtrlWin,"Stop"),
!.
win_clock_eh(_Win,e_Control(idc_startstop,_,_CtrlWin,_CtlInfo),0):-!,
stopTimer(_Win),
win_SetText(_CtrlWin,"Start"),
!.
```

Код использует название кнопки, чтобы решить должен ли таймер запускаться или останавливаться. Вызывая ***win_GetText*** может получить текст кнопки, а вызывая ***win_SetText*** можно задать новый текст для кнопки.

Через каждую секунду мы хотим изменять наше окно часов в зависимости от состояния переключателя. Для этого мы используем предикат, ***win_IsChecked***, который ,передавая указатель в check box window, может возвратить логическое значение, указывающее, выбран ли check box или нет.

В clause события ***e_Update*** мы не имеем указателя на окно check box. Для того, чтобы получить указатель, нужно вызвать предикат ***win_GetCtrlHandle***, который, используя указатель на родительское окно и постоянную ID для элемента управления, возвращает указатель окна элемента управления.

```
win_clock_eh(_Win,e_Update(_),0):-  %Note: no Cut !
/* This must be the first e_Update clause */
_CtrlWin = win_GetCtlHandle(_Win, idc_show_time),
IsChecked = win_IsChecked(_CtrlWin),
IsChecked = b_False,!,
```

Нужно удалить первый "!" в вышеуказанном clause, а также вызов **timer_set** из обработчика события **e_Create**.

```
win_clock_eh(_Win,e_Create(_),0):-!,
_NewTimerId =timer_Set(_Win, 1000), %% <- Remove this line
```

Последнее изменение, которое необходимо внести – убедиться, что окно обновляется (перерисовывается) сразу же после изменения check box. Таким образом, при получении события от check box, нужно перерисовывать окно.

```
win_clock_eh(_Win,e_Control(idc_show_time,_,_CtrlWin,_CtlInfo),0):-!,
win_Invalidate(_Win),
```

В вышеуказанном примере учитываются специфические черты MS-Windows. Когда создается элемент управления, он имеет по умолчанию установленный флаг ***ctl_Auto***. Это означает, что check box автоматически будет изменять свое состояние, когда пользователь щелкает на нем мышью. На других платформах этот флаг не работает, в следствии чего необходимо добавлять дополнительный clause,вызывающий предикат ***win_Check***, чтобы изменить установку.

2.12. Другие предикаты и события, используемые в приложениях 2

draw_pixel(window W, pnt P, color C) - закрашивает пиксель в окне W с координатами P цветом C.

random (real R) - генерирует псевдослучайное число в интервале [0..1) и записывает его в R.

str_int(string S, int I) - конвертирует целое число в строку или наоборот.

str_real(string S, real I) - конвертирует действительное число в строку или наоборот.

str_char(string S, char C) - конвертирует символ C в строку S или наоборот.

dlg_note(string S) - выводит сообщение S.

date(integer Year, integer Month, integer Day, integer DayOfWeek) - возвращает или устанавливает год, месяц, день и день недели.

win_getactivewindow() - возвращает указатель на активное окно.

win_getparent(window W) - возвращает указатель на родительское окно для окна W.

Событие e_KeyDown возникает при нажатии любой клавиши. Первый параметр - код клавиши.

Событие e_getfocus возникает при появление фокуса ввода.

3. Задания для лабораторной работы 1

1. Написать программу, которая выдает код нажатой клавиши.

2. Написать программу, которая при двойном нажатии правой клавиши мыши мыши выводит на экран координату X курсора.
   Чтобы узнать координату X курсора, необходимо вставить в обработчик события e_MouseDbl строку
   _PNT=pnt(X,Y),

3. Написать программу, которая выдает нажатую клавишу.

4. Написать программу, которая закрывается при нажатии клавиши BackSpace. Код клавиши BackSpace - 315. Для закрытия приложения вставить строки в обработчик события
   W=win_getactivewindow(),
   Win=win_getparent(W),
   win_destroy(Win),

5. Написать программу, которая при нажатии Enter выдавала случайно число от 0 до 1. Код клавиши Enter - 13.

6. Написать программу, которая перед началом работы выводит “Здравствуйте ”, а после окончания - экран “До свидания”.

7. Написать программу, которая при нажатии любой клавиши выводит текущий год.

8. Написать программу, которая при нажатии любой клавиши выводит текущий месяц.

9. Написать программу, которая при нажатии любой клавиши выдает порядковый номер дня недели

10. Написать программу, которая при открытии проверяет день недели, и если это понедельник, то выдает сообщение “Началась рабочая неделя ”.

11. Написать программу, которая при закрытии проверяет день недели, и если это пятница, то выдает сообщение “Поздравляю с выходными!”.

12. Написать программу, которая при нажатии любой клавиши проверяет текущий месяц и, если это сентябрь, то выдает сообщение “Поздравляю с началом учебного года!”

13. Написать программу, которая при нажатии любой клавиши проверяет текущий месяц и, если это июнь, то выдает сообщение “Поздравляю с окончанием учебного года!”

14. Написать программу, которая первого числа любого месяца при нажатии любой клавиши выводит на экран “Сегодня начинается новый месяц”.

15. Написать программу, которая шестнадцатого числа любого месяца при нажатии любой клавиши выводит на экран “Уже прошла половина месяца”.