Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

В.С. Билык, В.С. Писарьков, А.А. Шалыто

Автоматная реализация иерархического меню

Объектно-ориентированное программирование с явным выделением состояний

Проектная документация

Проект создан в рамках
"Движения за открытую проектную документацию"
http://is.ifmo.ru

Санкт-Петербург
2003

1. Введение

Наиболее часто применяемым пользовательским интерфейсом является оконный интерфейс, в котором существенную роль играет **меню**. Существует множество различных методов реализации, как самого оконного интерфейса, так и меню. «Дружественность» и удобство такого интерфейса определяет его популярность среди пользователей. Однако традиционные подходы к разработке логики пользовательских интерфейсов, основанные на использовании флагов, недостаточно эффективны и приводят к трудно понимаемому коду.

Таким образом, весьма актуальна задача разработки эффективных методов реализации управления пользовательским меню.

В работе [1] предложен метод реализации конечных автоматов [2] в рамках объектно-ориентированного программирования. Особенность этого метода состоит в том, что автомат рассматривается как объект-оболочка для множества объектов-состояний.

В настоящей работе этот метод применяется для решения указанной выше задачи — реализации управления пользовательским меню.

2. Постановка задачи

Основная задача меню — упростить доступ пользователя к различным сервисным функциям программы. Меню состоит из частей, далее называемых **подменю**, образующих иерархию. Составной частью подменю является **пункт меню**, который либо может быть связан с внешней процедурой, либо вести к другому подменю, **вложенному** в данное. Упрощение доступа пользователя достигается за счет логической структуризации пунктов. Например, в текстовом процессоре логично разместить пункты «Найти…» и «Найти и заменить…» в одном подменю, так как сервисные функции программы, к которым они ведут, решают схожие задачи.

На рис. 1 изображено меню трехуровневой вложенности. При этом в пункт 2 первого подменю вложено второе подменю, в третий пункт которого, в свою очередь, вложено третье подменю.

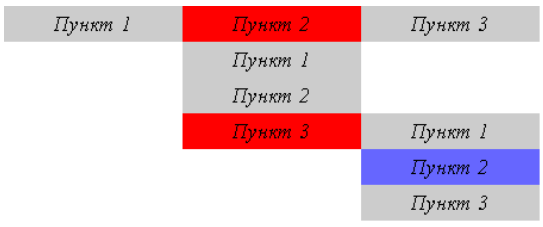

Рис. 1. Внешний вид меню

На рис. 2 для наглядности подменю выделены рамками, которые реально отсутствуют.

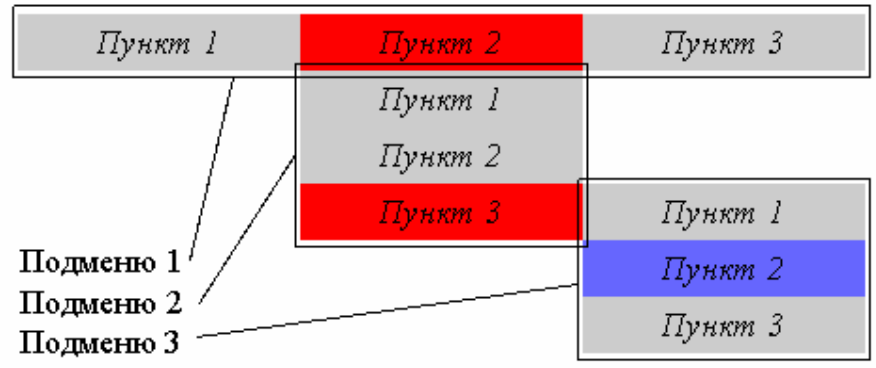

Рис. 2. Структура меню

Логическая структура меню приведена на рис. 3. Из этой структуры следует, что из некоторых пунктов подменю осуществляется вызов сервисных процедур, а с помощью других пунктов осуществляется переход к вложенным подменю.

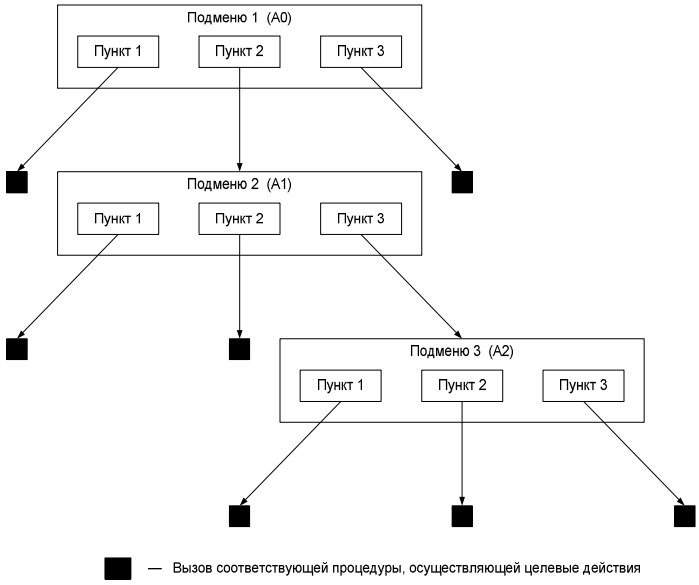

Рис. 3. Логическая структура меню

Меню с логической точки зрения имеет **древовидную структуру**, поэтому для того, чтобы выбрать определенный его пункт, пользователю, в общем случае, требуется последовательно проходя соответствующие пункты, спускаться вниз по дереву меню до необходимого подменю, содержащего требуемый пункт.

Подменю является независимой частью меню. Оно может находиться в следующих состояниях:

1. «Закрытое» («Closed»). В этом состоянии подменю не отображается.
2. «Ожидающее» («Waiting»). Подменю отображается, и все его пункты выглядят одинаково. На рис. 4 изображено подменю, находящееся в состоянии «Ожидающее».

Рис. 4. Подменю в состоянии «Ожидающее»

3. «Активное m» («Active m»). В этом состоянии подменю отображается, и подсвечен m-й пункт, который будем называть активным. На рис. 5 изображено подменю, находящееся в состоянии «Активное 2».

Рис. 5. Подменю в состоянии «Активное 2»

4. «Выбранное n» («Selected n»). В этом состоянии подменю отображается, и подсвечен n-й пункт, причем цветом, отличным от используемого в состоянии «Активное n». Этот пункт будем называть выбранным. На рис. 6 подменю находится в состоянии «Выбранное 2».

Рис. 6. Подменю в состоянии «Выбранное 2»

Число состояний «Активное m» и «Выбранное n» определяется количеством пунктов в данном подменю. Иначе говоря, m, n не превосходит N, где N — число пунктов в данном подменю.

Под **конфигурацией меню** будем понимать совокупность состояний всех его подменю. При этом считается, что для рассматриваемого меню определена **конфигурация по умолчанию —** конфигурация, в которой находится меню, когда пользователь с ним не работает. Например, используя введенные термины, можно сказать, что конфигурацией по умолчанию для стандартных меню в приложениях *Windows* является конфигурация, в которой «Подменю 1» находится в состоянии «Ожидающее», а все остальные — в состоянии «Закрытое».

Будем говорить, что подменю в данный момент **активно**, если оно находится в состояниях «Ожидающее» или «Активное m». В каждый момент времени только одно подменю может быть активным.

Работа пользователя с меню состоит в следующем.

1. Сначала меню находится в «конфигурации по умолчанию». Будем для определенности полагать, что в этой конфигурации активно лишь «Подменю 1» (рис. 2).

2. Когда пользователь не работает с конкретным подменю, оно находится в состоянии «Выбранное» (если пользователь уже прошел через это подменю), «Закрытое» или «Ожидающее».
3. В подменю, которое в данный момент активно, пользователь должен выбрать желаемый пункт. В то время, когда пользователь определяется с выбором, подменю может переходить из состояния «Активное i» в состояние «Активное j». В состоянии «Активное m» m-й пункт выделен другим цветом, что показывает пользователю, на каком именно пункте подменю он остановил свой выбор.
4. Когда пользователь определился с выбором и каким-либо образом сообщает об этом меню, то, в зависимости от того, куда указывает выбранный пункт (рис. 3), возможны два варианта:
   - если выбранный пункт ведет к подменю, вложенному в данное, то текущее подменю переходит из состояния «Активное m» в состояние «Выбранное m» (о чем свидетельствует выделение соответствующего пункта другим цветом) и перестает быть активным. Подменю, к которому ведет выбранный пункт, переходит из состояния «Закрытое» (его не было видно) в состояние «Ожидающее» (теперь оно отображается) и становится активным. Таким образом, осуществляется переход к вложенному подменю;
   - если выбранный пункт связан с внешней процедурой, реализующей целевые действия, то это означает, что пользователь осуществил свой *окончательный выбор* и закончил работу с меню. Управление передается упомянутой процедуре и меню принимает «конфигурацию по умолчанию».

Целью настоящей работы является создание программы, эффективно реализующей такое меню с использованием детерминированных конечных смешанных автоматов [2], которые в дальнейшем будем называть автоматами.

На рис. 1 в качестве примера был приведен внешний вид меню, состоящего из трех подменю, каждое из которых, в свою очередь, состоит из трех пунктов. Это меню будет в дальнейшем использоваться в качестве примера, который может быть обобщен на случай меню произвольной размерности. Под произвольной размерностью меню понимается произвольная сложность его логической структуры (рис. 3), но с обязательным требованием ее древовидности.

Отметим, что в работе [1] рассматривалось несколько реализаций автоматов на примере построения автомата для меню, состоящего всего из одного подменю. При этом удалось получить решение, не зависящее от количества элементов меню, которое и используется в настоящей работе.

При разработке использовался язык C++ и ОС *Windows*.

3. Краткое описание подхода

Каждое подменю будем рассматривать в виде отдельного программного объекта, объединяющего в себе **графическую оболочку,** отвечающую за внешний вид подменю на экране, и **логическую структуру,** представленную автоматом, которая определяет состояние подменю.

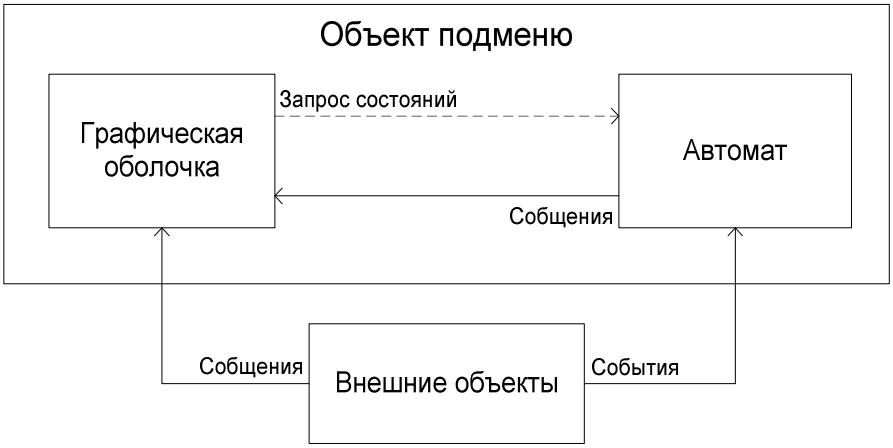

Рис. 7. Общая схема взаимодействия

В верхней части рис. 7 представлена схема взаимодействия этих двух составляющих подменю. Это взаимодействие осуществляется следующим образом. Графическая оболочка запрашивает у автомата его состояние и в соответствии с полученными данными отображает меню на экране. Автомат может переходить из состояния в состояние в зависимости от событий, которые автомату (через объект подменю) посылают внешние объекты. При этом на переходах автомат может отправлять графической оболочке сообщения для осуществления перерисовки меню, если это необходимо. Такие же сообщения графической оболочке могут посылать внешние объекты (например, операционная система).

Таким образом, отрисовка каждого подменю полностью определяется состоянием автомата, соответствующего данному подменю, независимо от других подменю.

Таким образом, мы разбили задачу на три подзадачи: реализация автоматов, реализация графической оболочки и объединение первых двух частей. Поэтому в дальнейшем для каждой из частей будут разработана соответствующая диаграмма классов.

Еще одна подзадача «Внешние объекты» (рис. 7) реализованная как взаимодействие пользователя с клавиатурой, не требует построения отдельной диаграммы классов в силу своей простоты.

4. Реализация автоматов

Основная особенность описываемого способа реализации меню — использование вложенных автоматов. Они взаимодействуют следующим образом: источник событий для меню (обработчик клавиатуры) посылает события *только* автомату A0, соответствующему "Подменю 1". Далее, когда "Подменю 1" переходит в состояние "Выбранное" этот автомат транслирует автомату A1 все посылаемые ему события. Последний ведет себя точно так же: когда "Подменю 2" переходит в состояние "Выбранное" автомат A1 транслирует все посылаемые ему события автомату A2. Этот автомат не содержит вложенных автоматов, и поэтому дальнейшей трансляции событий не происходит. Все автоматы отличаются друг от друга лишь *вызовом вложенных автоматов* и *выполнением целевых действий* на переходах. Поэтому указанный механизм трансляции событий вложенными автоматами позволяет одними и теми же событиями управлять каждым подменю. Более подробно это будет рассмотрено в разд. 3.4. Наличие в этих автоматах общей логики позволяет создать класс-предок, описывающий поведение автомата в целом, и унаследовать от него классы, реализующие поведение уже конкретных автоматов A0, A1 и A2. Представление автомата в виде класса и вопросы, связанные с наследованием, были рассмотрены в работе [1].

Будем полагать, что пользователь управляет меню *непосредственно* с помощью событий, происхождение которых произвольно.

На рис. 8 изображена схема наследования и взаимодействия вложенных автоматов.

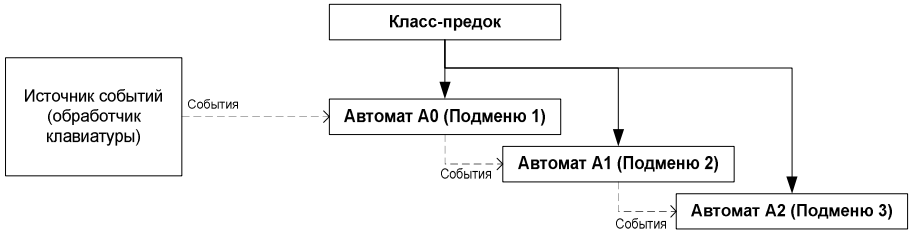

Рис. 8. Схема взаимодействия автоматов

4.1. Реализация «Подменю 1» (Автомат A0)

4.1.1. Схема связей

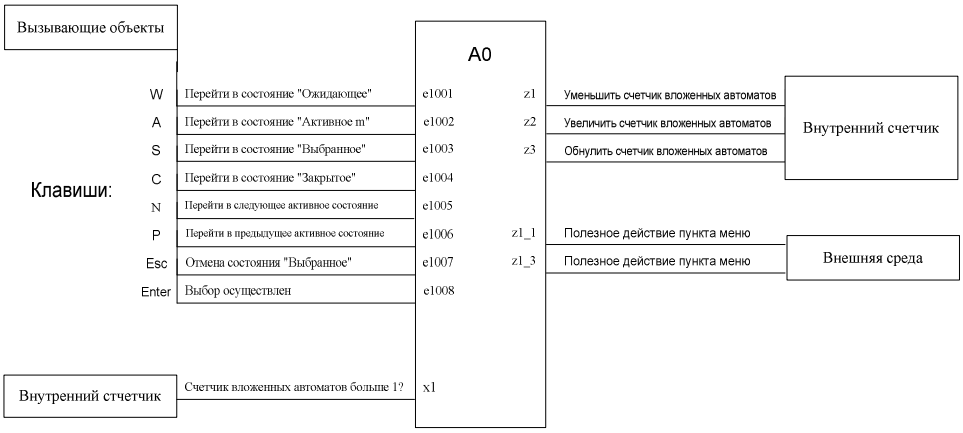

Рис. 9. Схема связей автомата A0

4.1.2. Граф переходов

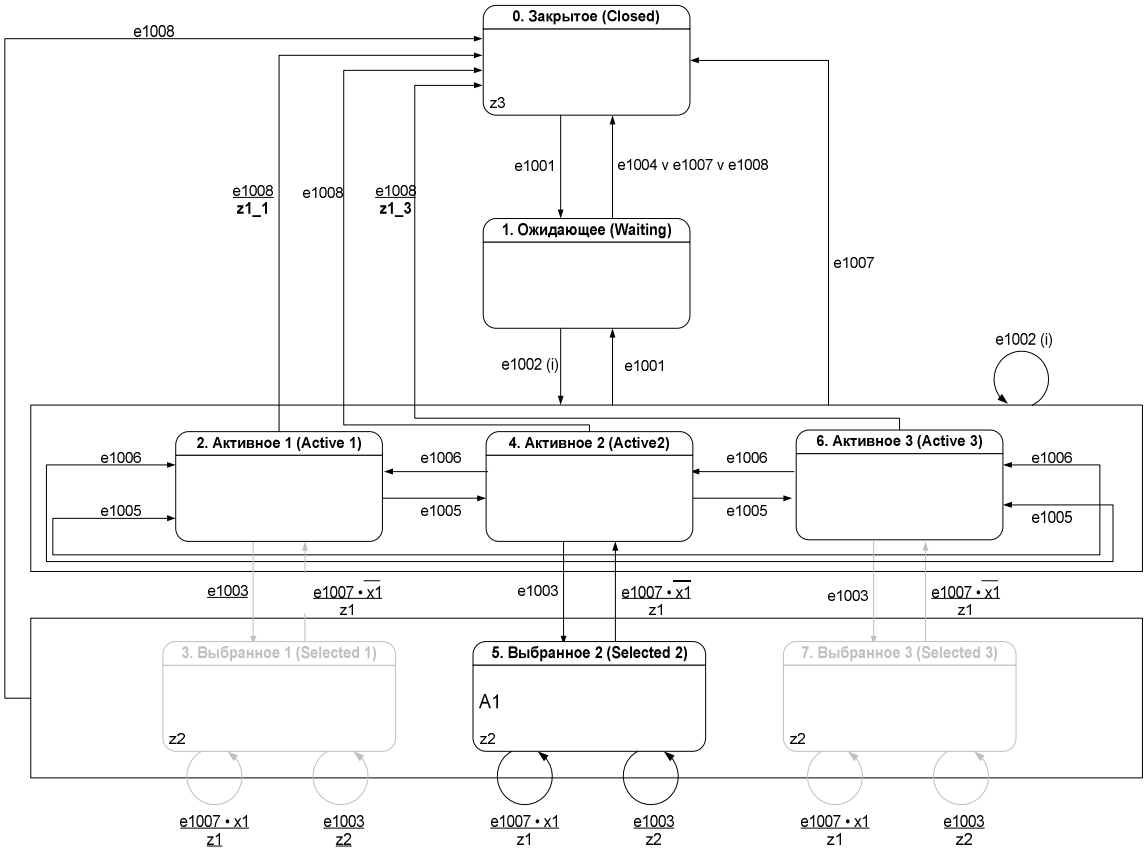

Рис. 10. Граф переходов автомата A0

4.2. Реализация «Подменю 2» (Автомат A1)

4.2.1. Схема связей

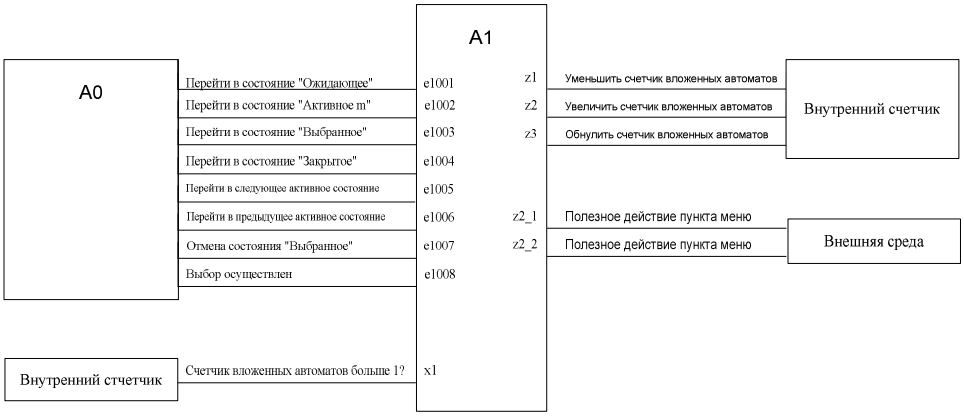

Рис. 11. Схема связей автомата A1

4.2.2. Граф переходов

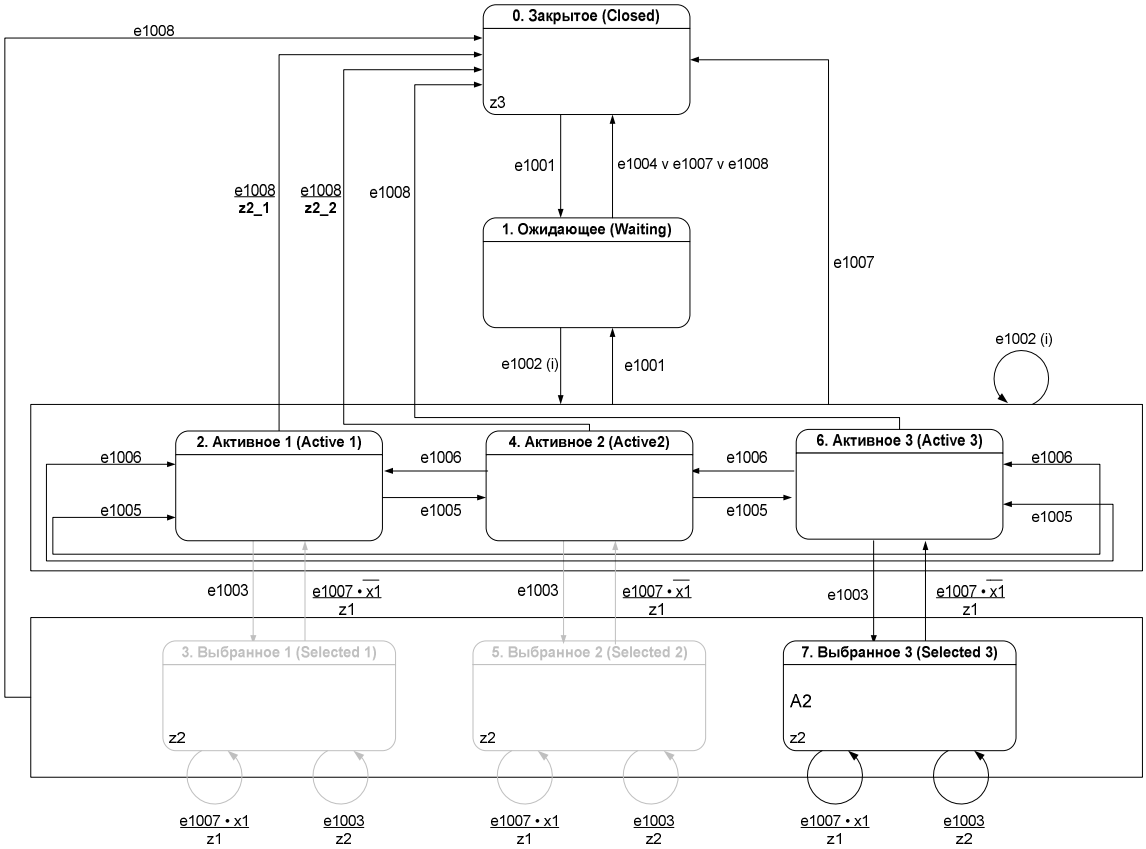

Рис. 12. Граф переходов автомата A1

4.3. Реализация «Подменю 3» (Автомат A2)

4.3.1. Схема связей

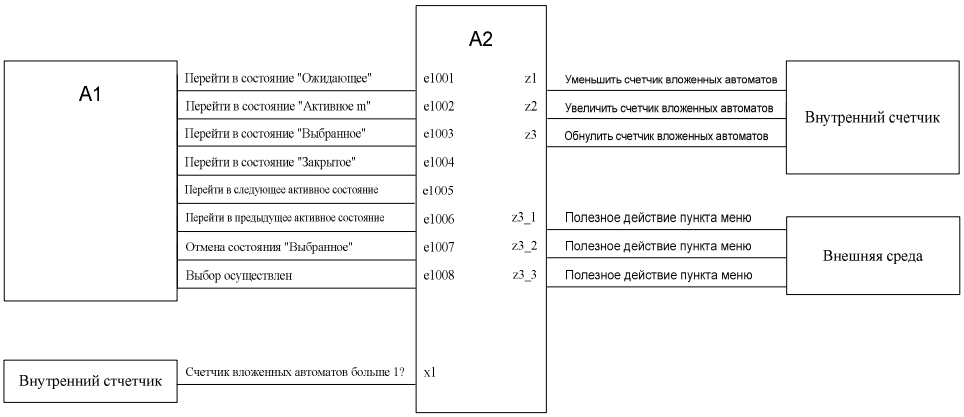

Рис. 13. Схема связей автомата A2

4.3.2. Граф переходов

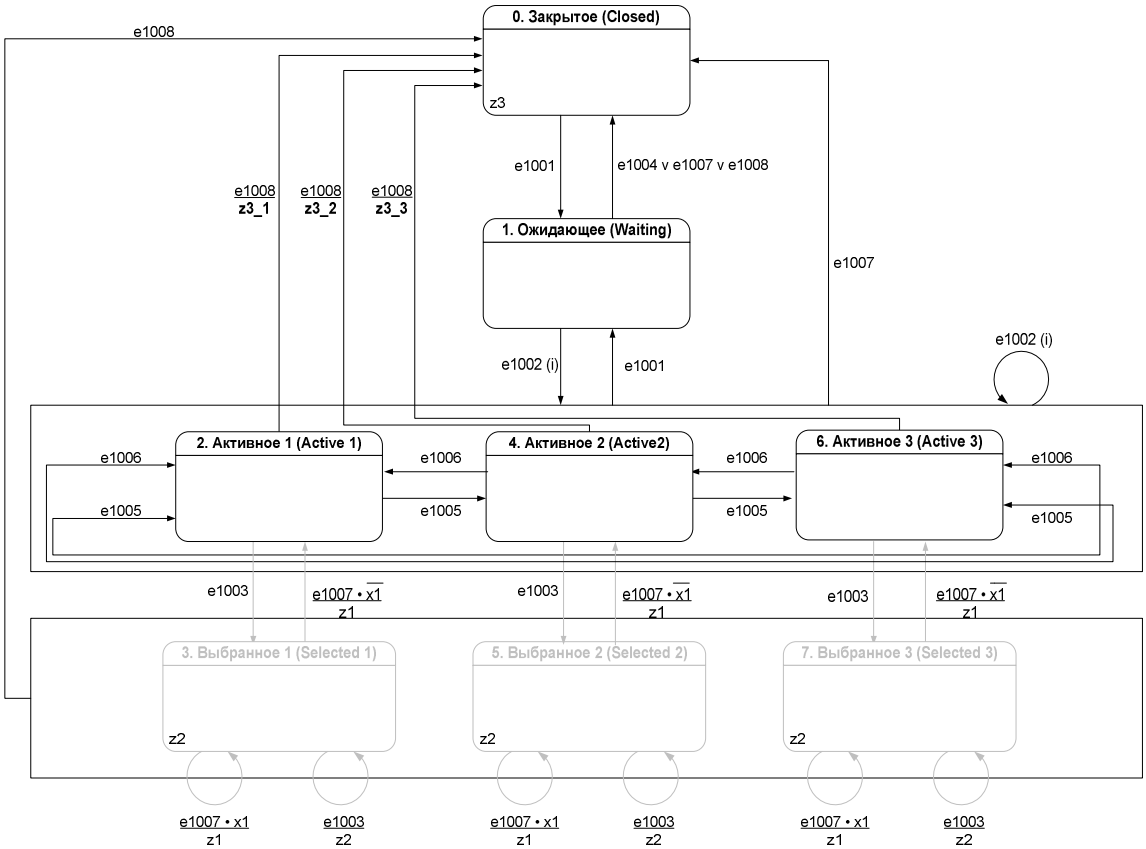

Рис. 14. Граф переходов автомата A2

4.4. Описание работы вложенных автоматов

Как уже было отмечено, все автоматы, реализующие логику подменю, отличаются друг от друга лишь *вызовом вложенных автоматов* и *выполнением целевых действий* на переходах. Механизм трансляции событий вложенным автоматам, упомянутый во введении разд. 3, позволяет с помощью одних и тех же событий управлять каждым подменю.

Для подробного рассмотрения работы этого механизма обратимся к логической структуре меню (рис. 3). Как отмечено выше, в каждый момент времени только одно подменю может быть активным. Это объясняется тем, что пользователь двигаясь по меню, каждый раз делает выбор среди пунктов того подменю, которое в данный момент *активно*.

Предположим, что в данный момент времени меню находится в следующей конфигурации: «Подменю 1» — в состоянии «Ожидающее», остальные подменю — в состоянии «Закрытое» (не отображаются). На рис. 15 представлена логическая структура меню в этой конфигурации, при этом активное подменю выделено серым цветом.

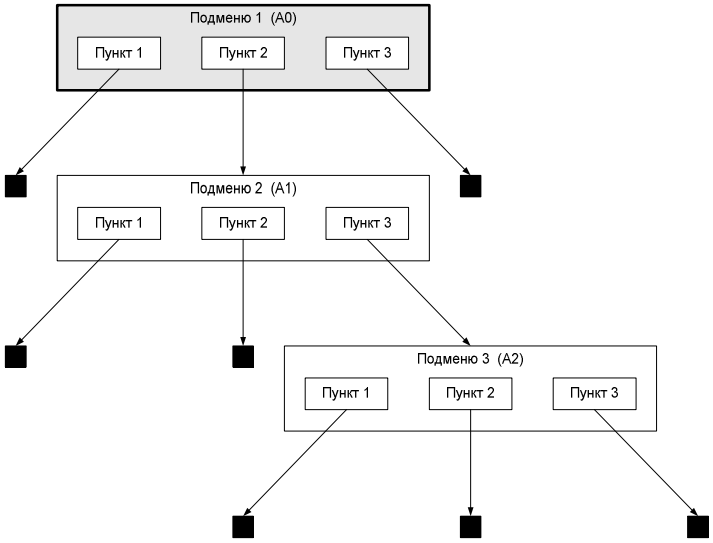

Рис. 15. «Подменю 1» активно и находится в состоянии «Ожидающее». Остальные подменю — в состояниях «Закрытое»

В этой ситуации получение событий *e1002* (Перейти в состояние «Активное m»), *e1005* (Перейти в следующее активное состояние) и *e1006* (Перейти в предыдущее активное состояние) автоматом A0 (рис. 10) обеспечивает возможность навигации в «Подменю 1». Под навигацией в подменю понимается процесс выбора пользователем одного из пунктов.

Автоматы A1 и A2 реагируют на эти события аналогично (рис. 12, 14). Поэтому для того, чтобы предоставить пользователю возможность навигации в других подменю, необходимо каким-либо образом направить указанные события к соответствующим автоматам. Однако из схемы взаимодействия автоматов (рис. 8) следует, что источник событий посылает события *только* автомату A0. Кажущаяся, на первый взгляд, невозможность управления вторым и третьим подменю разрешается следующим образом: в определенном состоянии автомат A0 транслирует все события автомату A1, который, в свою очередь, транслирует все события автомату A2.

Трансляция осуществляется с помощью вызова соответствующего вложенного автомата в некоторых состояниях «Выбранное k» (k – конкретный номер состояния). Эти состояния соответствуют пунктам подменю, ведущим к вложенным подменю. Например, в автомате A0 (рис. 10) вызов вложенных автоматов осуществляется только в одном состоянии — «Выбранное 2». При этом вызывается вложенный автомат A1. Это объясняется тем, что в соответствие с логической структурой меню (рис. 3) второй пункт «Подменю 1» ведет к вложенному «Подменю 2». Остальные пункты «Подменю 1» связаны с внешним процедурами, выполняющими целевые действия. В автомате A1 вызов вложенного автомата A2 осуществляется аналогично.

Напомним, что если в определенном состоянии автомата осуществляется вызов вложенного автомата, то любое событие, пришедшее первому автомату, будет *сначала* передано вложенному автомату, и *лишь затем* произойдет его обработка первым автоматом. Поэтому, если несколько последовательно вложенных автоматов получат событие, то его обработка соответствующими автоматами будет происходить в порядке обратном порядку передаче события. Например, если цепочка последовательно вложенных автоматов выглядит как A0 — A1 — A2, то событие, отправленное автомату A0, будет сначала обработано автоматом A2, затем — автоматом A1, и, наконец, автоматом A0.

Поясним, почему вызов вложенных автоматов происходит именно в состояниях «Выбранное m». В разд. 1 была описана схема работы пользователя с меню, из которой, в частности, следует, что в момент, когда пользователь выбрал в текущем подменю определенный пункт, ведущий к вложенному подменю, текущее подменю переходит в состояние «Выбранное m» (где m — номер соответствующего пункта). После этого вложенное подменю переходит из состояния «Закрытое» в состояние «Ожидающее». Благодаря этому оно появляется на экране и становится *активным*. При этом пользователь получает возможность перемещаться по меню.

Обобщим сказанное. В то время, когда пользователь перемещается по активному подменю, все подменю, через которые он уже прошел, находятся в состоянии «Выбранное k» (k – в каждом подменю разное). Этот факт имеет место также и в случае меню произвольной размерности.

На рис. 16 схематично изображено меню в ситуации, когда пользователь в «Подменю 1» выбрал второй пункт, а затем в «Подменю 2» выбрал третий пункт и теперь осуществляет навигацию в «Подменю 3», которое в данный момент активно.

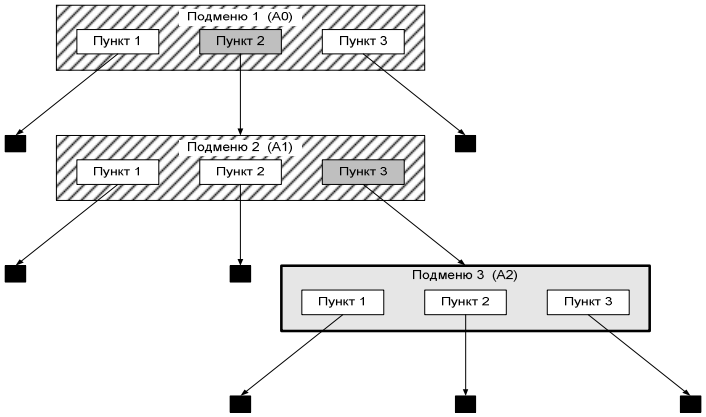

Рис. 16. «Подменю 3» активно (находится в состоянии «Ожидающее»), остальные подменю — в состояниях «Выбранное m»

Подменю, через которые пользователь уже прошел, на этом рисунке заштрихованы и находятся в состояниях «Выбранное m».

Для того, чтобы в рассматриваемой конфигурации меню пользователь мог осуществлять навигацию в «Подменю 3», необходимо обеспечить доставку событий автомату A2 от источника событий. Это обеспечивается по схеме (рис. 8).

Работа автоматов и трансляция событий во время навигации по «Подменю 3» отображена в следующем фрагменте файла протоколирования. При этом меню находится в конфигурации, изображенной на рис. 16. В этом фрагменте последовательно отправляются события *e1002, e1005, e1006* и осуществляется работа автоматов.

```
>> Key pressed: A (65)
A0          5 ( Selected 2          )  <- e1002
A1          7 ( Selected 3          )  <- e1002
A2          1 ( Waiting             )  <- e1002
A2          1 ( Waiting             )  ->    2 ( Active 1           )
A2          2 ( Active 1            )  finished (e1002)
A1          7 ( Selected 3          )  finished (e1002)
A0          5 ( Selected 2          )  finished (e1002)

>> Key pressed: N (78)
A0          5 ( Selected 2          )  <- e1005
A1          7 ( Selected 3          )  <- e1005
A2          2 ( Active 1            )  <- e1005
A2          2 ( Active 1            )  ->    4 ( Active 2           )
A2          4 ( Active 2            )  finished (e1005)
A1          7 ( Selected 3          )  finished (e1005)
A0          5 ( Selected 2          )  finished (e1005)

>> Key pressed: P (80)
A0          5 ( Selected 2          )  <- e1006
A1          7 ( Selected 3          )  <- e1006
A2          4 ( Active 2            )  <- e1006
A2          4 ( Active 2            )  ->    2 ( Active 1           )
A2          2 ( Active 1            )  finished (e1006)
A1          7 ( Selected 3          )  finished (e1006)
A0          5 ( Selected 2          )  finished (e1006)
```

Теперь поясним назначение событий ***e1007*** (*Отмена состояния «Выбранное»*) и ***e1008*** (*Выбор осуществлен*) на примере меню в конфигурации, изображенной на рис. 16.

Часто необходимо перейти из вложенного подменю в родительское. Это можно осуществить с помощью события ***e1007***. Заметим, что подменю, через которые пользователь уже прошел, находятся в состоянии «Выбранное m». В этих состояниях осуществляется трансляция событий активному подменю.

Отметим так же, что если любому автомату (A0, A1 или A2) находящемуся в состоянии «Ожидающее» или «Активное m» будет отправлено событие ***e1007***, то этот автомат перейдет в состояние «Закрытое» (рис. 10, 12, 14). Это свойство присуще всем автоматам. Таким образом, при получении события ***e1007*** активное подменю (оно всегда одно) переходит в состояние «Закрытое» и перестает быть активным.

Выше было отмечено, что при последовательной передаче событий от одного вложенного автомата к другому, их обработка будет происходить в обратном порядке. Таким образом, автомат A2 находящийся в состоянии «Ожидающее», первым обработает событие ***e1007*** и перейдет в состояние «Закрытое». Далее обработку этого события начнет автомат A1, который (рис. 12) перейдет из состояния «Выбранное 3» в состояние «Активное 3». Далее обработку события ***e1007*** начнет автомат A0, который *останется* в состоянии «Выбранное 2». Такое поведение достигается с помощью специального счетчика, который есть у каждого автомата. Этот счетчик увеличивается каждый раз, когда, находясь в состоянии «Выбранное m», автомат получает событие ***e1003*** (*Перейти в состояние «Выбранное»*). После этого, еще один автомат из цепочки вложенных автоматов переходит в состояние «Выбранное m» и *начинает транслировать события* следующему вложенному автомату.

Счетчик уменьшается каждый раз, когда в состоянии «Выбранное m» автомат получает событие ***e1007*** (*Отмена состояние «Выбранное»*). При этом автомат из цепочки вложенных автоматов, который последним перешел в состояние «Выбранное», переходит в состояние «Закрытое» и перестает транслировать события.

Переход из состояния «Выбранное m» в состояние «Активное m» осуществляется по событию ***e1007*** лишь в том случае, когда счетчик данного автомата не больше единицы (рис. 10, 12, 14). Это свойство присуще всем автоматам.

Вернемся к рассматриваемому примеру. После того, как событие ***e1007*** было последовательно обработано автоматами A2, A1 и A0, меню будет находиться в конфигурации, изображенной на рис. 17.

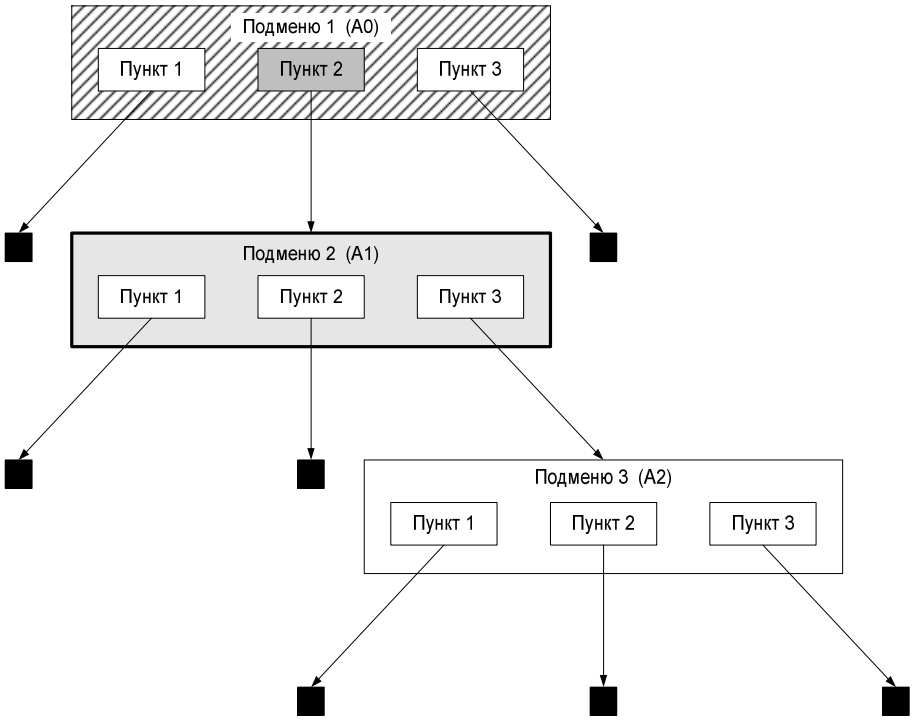

Рис. 17. «Подменю 2» находится в состоянии «Активное 3», «Подменю 1»— в состоянии «Выбранное 2», «Подменю 3» — в состоянии «Закрытое»

В этой конфигурации «Подменю 2» активно и в нем пользователь может осуществлять навигацию. Для большей понятности, рассмотрим по шагам, что произойдет, если автомат A0 еще раз получит событие *e1007*:

1. Автомат A0 передаст событие *e1007* автомату A1.
2. Автомат A1 получит событие *e1007* и будет осуществлен переход из состояния «Активное 3» в состояние «Закрытое». «Подменю 2» перестает быть активным.
3. Автомат A0 начнет обработку события *e1007* и, так как счетчик автомата A0 равен единице, будет осуществлен переход из состояния «Выбранное 2» в состояние «Активное 2». «Подменю 1» становится активным.

Работа автоматов и трансляция событий при получении двух событий *e1007* видна на следующем отрывке файла протоколирования программы:

```
>> Key pressed: • (27)
A0         5 ( Selected 2          )  <- e1007
A1         7 ( Selected 3          )  <- e1007
A2         1 ( Waiting             )  <- e1007
A2         1 ( Waiting             )  ->    0 ( Closed             )
A2         0 ( Closed              )  finished (e1007)
A1         7 ( Selected 3          )  ->    6 ( Active 3           )
A1         6 ( Active 3            )  finished (e1007)
A0         5 ( Selected 2          )  finished (e1007)

>> Key pressed: • (27)
A0         5 ( Selected 2          )  <- e1007
A1         6 ( Active 3            )  <- e1007
A1         6 ( Active 3            )  ->    0 ( Closed             )
A1         0 ( Closed              )  finished (e1007)
A0         5 ( Selected 2          )  ->    4 ( Active 2           )
A0         4 ( Active 2            )  finished (e1007)
```

В рассмотренном фрагменте протокола первые девять строк соответствуют обработке первого события *e1007*, а следующие семь строк – обработке второго события *e1007*.

Событие *e1008* (*Выбор осуществлен*) поступает в момент, когда пользователь осуществил свой *окончательный выбор*. Как уже было сказано в разд. 2, меню в этой ситуации передает управление внешней процедуре, реализующей целевые действия, и принимает «конфигурацию по умолчанию». Действительно, меню будет вести себя именно таким образом:

- событие *e1008* будет транслировано по цепочке активному подменю;
- это подменю начнет обработку события *e1008* и перейдет из состояния «Активное m» в состояние «Закрытое». При этом на переходе будет вызвана упомянутая выше внешняя процедура;
- остальные подменю один за другим начнут обработку пришедшего события и будут переходить из состояния «Выбранное m» в состояние «Закрытое».

4.5. Диаграмма классов

Диаграмма классов, реализующих автоматы, приведена на рис. 18. Отметим, что на этой и дальнейших диаграммах стрелки направлены от предков к потомкам.

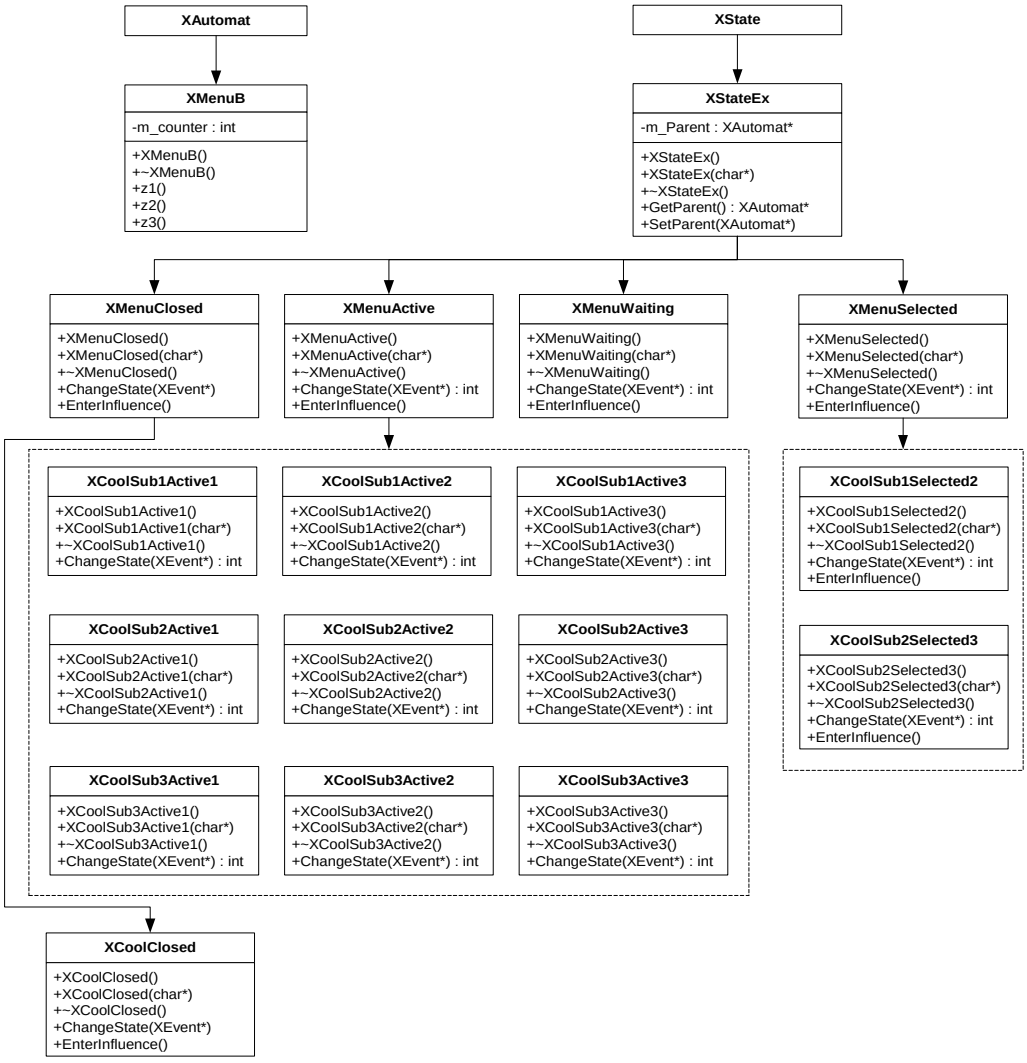

Рис. 18. Диаграмма классов, реализующих автоматы

Она состоит из двух частей. Первая часть реализует классы-оболочки автоматов (*XAutomat*), а вторая – классы-состояния (*XState*).

Опишем первую часть. Класс *XAutomat* является базовым. От этого класса унаследован класс *XMenuB*, который предоставляет необходимые средства для реализации поведения подменю. Эти средства являются универсальными для всех меню:

- переменная-счетчик вложенных автоматов (*m_counter*), о которой уже упоминалось в разд. 3.4;
- функция *x1*, проверяющая значение переменной *m_counter*;
- методы *z1*, *z2*, *z3*, изменяющие значение этой переменной. Эти методы увеличивают, уменьшают и сбрасывают указанную переменную.

Вторая часть диаграммы классов реализует классы-состояния. Класс *XState* является абстрактным базовым классом. В нем определены виртуальные методы *ChangeState и EnterInfluence*, отвечающие за переход в новое состояние и осуществление входного воздействия в состоянии [1].

Класс *XStateEx* добавляет к классу *XState* указатель на объект-оболочку (*m_Parent*), содержащий объект-состояние, и методы установки и получения его значения (методы *SetParent* и *GetParent*). Классы *XMenuClosed*, *XMenuWaiting*, *XMenuActive* и *XMenuSelected* унаследованы от класса *XStateEx*. Они реализуют методы *ChangeState* и *EnterInfluence*, определяя универсальное поведение подменю независимо от числа состояний автомата.

Теперь, если создать экземпляр класса *XMenuB*, добавить в него по экземпляру классов *XMenuClosed* и *XMenuWaiting*, а затем экземпляр *XMenuActive* и экземпляр *XMenuSelected*, то получим автомат, соответствующий простейшему меню, состоящему из одного пункта. Однако построенный автомат не будет выполнять никаких целевых действий, поскольку они не предусмотрены в реализациях классов *XMenuClosed*, *XMenuWaiting*, *XMenuActive* и *XMenuSelected*.

Для реализации целевых действий необходимо унаследоваться от классов *XMenuClosed*, *XMenuWaiting*, *XMenuActive*, *XMenuSelected* и переопределить методы *ChangeState*, *EnterInfluence*. Если в переопределенных методах вызывать методы базовых классов, которые реализуют универсальное поведение, а затем процедуры и функции, выполняющие целевые действия, то можно получить необходимые частные реализации классов-состояний.

Именно такой подход используется в реализации классов *XCoolClosed*, *XCoolSub1Active1*, *XCoolSub1Active2*, *XCoolSub1Active3*, *XCoolSub2Active1*, *XCoolSub2Active2*, *XCoolSub2Active3*, *XCoolSub3Active1*, *XCoolSub3Active2*, *XCoolSub3Active3*, *XCoolSub1Selected2* и *XCoolSub2Selected3*.

5. Графическая оболочка

5.1. Словесное описание

Графическая оболочка отвечает за отображение объекта-меню (отдельного подменю) на экране. Для ее реализации использовалась среда *Visual Studio 6.0* и ее библиотека классов *MFC*.

5.2. Диаграмма классов

Графическая оболочка представлена двумя классами *CWnd* и *XTmpWnd* (рис. 19, 20).

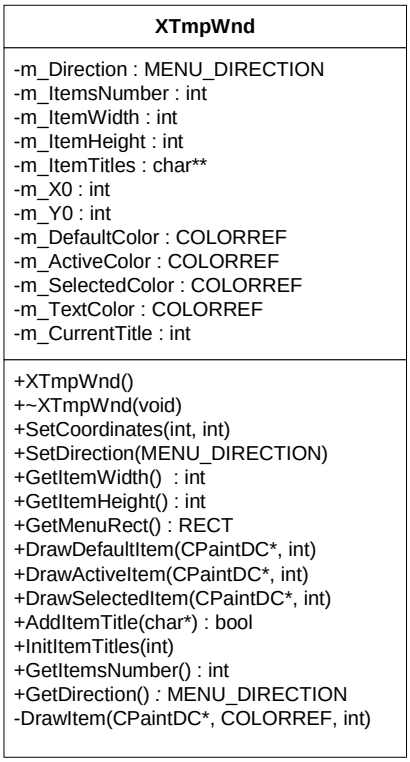

Рис. 19. Класс XTmpWnd

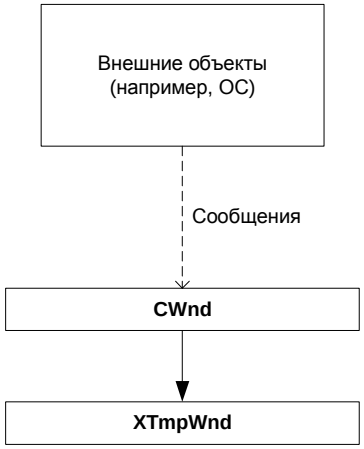

Рис. 20. Диаграмма классов графической оболочки

5.3. Класс CWnd

Класс *CWnd* является базовым в иерархии классов, реализующей графическую оболочку. Он представляет собой часть библиотеки *MFC* (базовый класс окна). *CWnd* отвечает за обработку всех сообщений, приходящих к окну (рис. 19), и предоставляет стандартные средства отображения окон в операционной системе *Windows*.

5.4. Класс XTmpWnd

Графическое представление всех подменю полностью определяется реализацией этого класса. Этот класс предоставляет все необходимые средства для его отображения: методы *DrawDefaultItem*, *DrawActiveItem* и *DrawSelectedItem* (рис. 20). Этот класс унаследован от класса *CWnd*.

6. Объединение автоматной и графической части

6.1. Словесное описание

В соответствии с подходом, описанным в разд. 2, объект подменю содержит в себе свойства графической оболочки (класс *XTmpWnd*) и автомата (*XMenuB*). Объект подменю представлен классом *XMenuWnd*, который может быть базовым для всех конкретных реализаций подменю для ОС *Windows*.

6.2. Схема наследования

На рис. 21 приведена схема множественного наследования, отражающая объединение автоматной и графической частей.

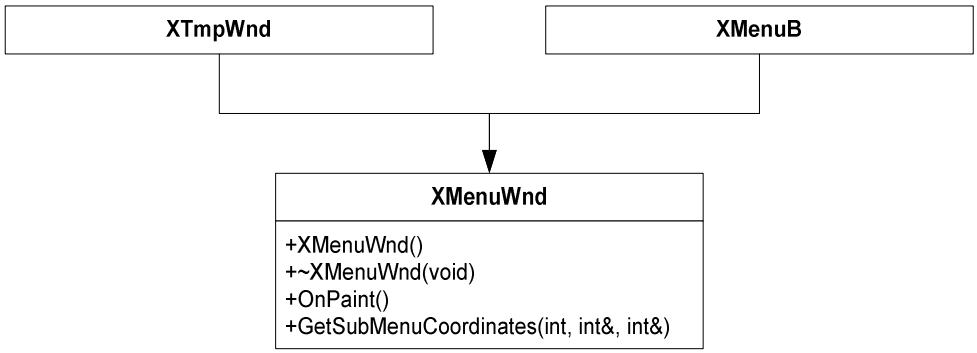

Рис. 21. Схема наследования

Класс *XMenuWnd* наследуется от классов *XTmpWnd* и *XMenuB*. Он объединяет в себе все атрибуты и методы обоих классов. В этом классе переопределяется обработчик *OnPaint* (сообщения *WM_PAINT*), отвечающий за перерисовку окна. Внутри этого обработчика для отображения каждого пункта подменю вызываются методы *DrawDefaultItem*, *DrawActiveItem* или *DrawSelectedItem* класса *XTmpWnd* в зависимости от того в каком состоянии находится автомат, соответствующий данному подменю.

Класс *XMenuWnd* по своей функциональности полностью соответствует схеме указанной на рис. 7. Запрос состояний производится вызовом метода *GetCurrentState*. Отправка события автомату осуществляется методом *ReceiveEvent*. Отправка сообщения графической оболочке выполняется методами, реализованными в классе *CWnd*.

Таким образом, для того, чтобы создать подменю, достаточно унаследоваться от класса *XMenuWnd*.

6.3. Диаграмма классов

На рис. 22 приведена диаграмма классов, реализующих подменю.

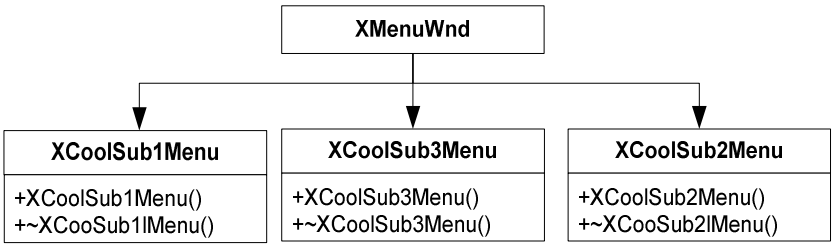

Рис. 22. Диаграмма классов, реализующих подменю

Классы *XCoolSub1Menu*, *XCoolSub2Menu* и *XCoolSub3Menu*, унаследованные от *XMenuWnd*, реализуют «Подменю 1», «Подменю 2» и «Подменю 3» соответственно (рис. 18, 22). В конструкторах этих классов «собираются» соответствующие им автоматы из требуемых объектов-состояний.

6.4. Управление меню с помощью клавиатуры

Управление меню с помощью клавиатуры реализовано по схеме, изображенной на рис. 23.

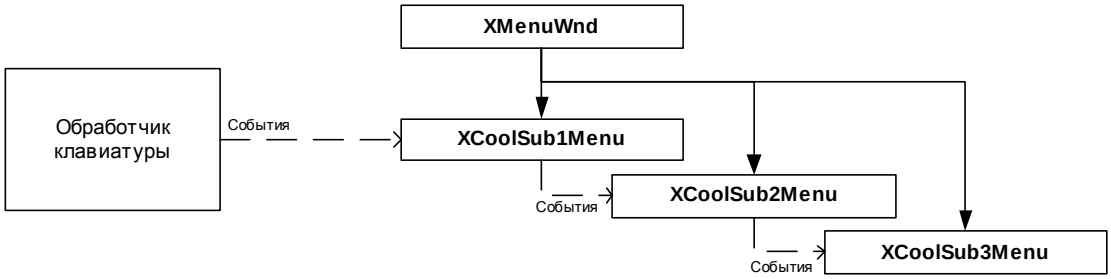

Рис. 23. Схема взаимодействия подменю

Обработчик клавиатуры в зависимости от нажатых клавиш посылает события автомату главного подменю, который при необходимости может транслировать их автоматам вложенных подменю. Соответствие клавиш и событий приведено в таблице.

Таблица. Соответствие клавиш и событий

Клавиша	**Событие**
W	e1001
A	e1002
S	e1003
C	e1004
N	e1005
P	e1006
Esc	e1007
Enter	e1008

7. Заключение

Первоначальной целью работы было создание метода, позволяющего легко и быстро создавать абстрактные (не привязанные к какой-либо системе пользовательского интерфейса) меню, поведение которых должны были реализовывать автоматы. Выбор на меню пал не случайно. Среди всех инструментов пользовательского интерфейса меню, видимо, наиболее распространены. Они используются практически во всех программах, начиная от компьютерных игр и заканчивая оболочками для мобильных телефонов. Такая распространенность применения меню и привела к идее их абстрактной реализации с целью создания метода их построения.

Состояниям, которые может принимать меню (разд. 2), естественно сопоставить состояния автомата. В этом случае для реализации вложенных подменю можно использовать вложенные автоматы. Это обосновывает целесообразность применения автоматов для решения рассматриваемой задачи.

Отметим, что хотя пунктов меню может быть любое количество, поведения от этого не изменяется. В этот момент стало ясно, что SWITCH-технология, как она описана в работах [2, 3], не удовлетворяет полностью поставленной задаче, так как количество пунктов меню, а, следовательно, количество состояний автомата может быть произвольным и заранее не известным.

В результате было принято решение модифицировать SWITCH-технологию так, чтобы она позволяла решать и такие задачи. Это побудило авторов написать работу [1], в которой описан новый метод реализации автоматов. С помощью этого метода и была решена поставленная задача. Предложенный подход позволяет также реализовывать автоматы, изменяющие свое поведение во время работы. Такие автоматы могут быть названы перестраиваемыми.

Литература

1. *Шалыто А.А., Билык В.С.*. Сравнение SWITCH-технологии и паттерна State. http://is.ifmo.ru (раздел «Проекты»).
2. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. 628 с.
3. *Шалыто А.А., Туккель Н.И.* Программирование с явным выделением состояний. //Мир ПК. 2001. №8. С. 116-121, №9. С. 132-138.
4. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д.*. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001. 368 с.

8. Приложение. Исходные тексты программы.

8.1. Файл automat.cpp

//Реализация базового класса-оболочки автомата.

```
#include "automat.h"
#include <memory.h>

/*---------------------------------[ XEvent ]---------------------------*/

XEvent::XEvent()
{
	m_Info = new char[4];
	for (int i = 0; i < 4; i++)
	{
		((char*)m_Info)[i] = 0;
	}
}

XEvent::XEvent(int n)
{
	m_Info = new char[n];
	for (int i = 0; i < n; i++)
	{
		((char*)m_Info)[i] = 0;
	}
}

XEvent::~XEvent()
{
	delete [] m_Info;
}

/*---------------------------------[ XState ]---------------------------*/

XState::XState()
{
	m_Title = 0;
	SetTitle("Untitled");
}

XState::XState(const char* title)
{
	m_Title = 0;
	SetTitle(title);
}

void XState::SetTitle(const char* title)
{
	if (m_Title)
	{
		delete [] m_Title;
	}
	int len = strlen(title);
	m_Title = new char[len + 1];
	strcpy(m_Title, title);
}

char* XState::GetTitle()
{
	return m_Title;
}

XState::~XState()
{
	delete [] m_Title;
}
/*---------------------------------[ XAutomat ]---------------------------*/

XAutomat::XAutomat(void)
{
	m_BeginLogging = m_TransLogging = m_EndLogging = 0;
	m_CurrentState = m_Available = m_StatesNumber = -1;
	m_ReportLog = 0;
	m_States = 0;
```

```
	SetTitle("Untitled");
}

XAutomat::XAutomat(int num)
{
	m_BeginLogging = m_TransLogging = m_EndLogging = 0;

	m_CurrentState = 0;
	m_StatesNumber = 0;
	m_Available = num;
	m_States = new XState*[m_Available];
	m_ReportLog = 0;
	SetTitle("Untitled");
}

XAutomat::XAutomat(const char* title, int num)
{
	m_BeginLogging = m_TransLogging = m_EndLogging = 1;

	m_CurrentState = 0;
	m_StatesNumber = 0;
	m_Available = num;
	m_States = new XState*[m_Available];
	m_Title = 0;
	SetTitle(title);
	m_ReportLog = 0;
}

XAutomat::~XAutomat()
{
	if (m_Title)
	{
		delete [] m_Title;
	}
	if (m_States)
	{
		delete [] m_States;
	}
}

void XAutomat::Automat(const XEvent &e)
{
	int OldState = m_CurrentState;

	LogBegin(m_CurrentState, e.m_Number);
	m_CurrentState = m_States[m_CurrentState]->ChangeState(e);

	if (OldState == m_CurrentState)
	{
		LogEnd(m_CurrentState, e.m_Number);
		return;
	}

	LogTrans(OldState, m_CurrentState);

	if ((m_CurrentState >= m_StatesNumber) || (m_CurrentState < 0))
	{
		LogError(m_CurrentState);
	}

	m_States[m_CurrentState]->EnterInfluence();
	LogEnd(m_CurrentState, e.m_Number);
}

void XAutomat::SetTitle(const char* title)
{
	if (m_Title)
	{
		delete [] m_Title;
	}
	int len = strlen(title);
	m_Title = new char[len + 1];
	strcpy(m_Title, title);
}

void XAutomat::ReceiveEvent(const XEvent &e)
{
	Automat(e);
}
```

```
int XAutomat::GetCurrentState(void)
{
	return m_CurrentState;
}

int XAutomat::GetStatesNumber(void)
{
	return m_StatesNumber;
}

XState* XAutomat::GetState(int num)
{
	if ((num < m_StatesNumber) & (num > -1))
	{
		return m_States[num];
	}
	else
	{
		return 0;
	}
}

void XAutomat::AddState(XState *s)
{
	if (m_Available == m_StatesNumber)
	{
		m_Available *= 2;
	 XState** tmp = new XState*[m_Available];
		memcpy((char*)tmp, ((const char*)m_States), sizeof(XState*) *  m_StatesNumber);
		delete [] m_States;
		m_States = tmp;
	}
	m_States[m_StatesNumber] = s;
	m_StatesNumber++;
}

bool XAutomat::InsertState(XState* s, int num)
{
	if (num >= (m_StatesNumber - 1))
	{
		return 0;
	}
	m_States[num] = s;
	return 1;
}

void XAutomat::InitLog(ofstream* fOut, bool begin, bool trans, bool end)
{
	m_ReportLog = fOut;
	m_BeginLogging = begin;
	m_TransLogging = trans;
	m_EndLogging = end;
	(*m_ReportLog) << m_Title << " was initialized" << endl;
}

void XAutomat::TerminateLog(void)
{
	(*m_ReportLog) << m_Title << " was terminated" << endl;
}

void XAutomat::LogBegin(int state, int event)
{
	if (!m_BeginLogging)
	{
		return;
	}
	(*m_ReportLog).width(8);
	(*m_ReportLog).setf(ios::left);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_Title;
	(*m_ReportLog).width(4);
	(*m_ReportLog).setf(ios::right);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << state << " ( ";
	(*m_ReportLog).unsetf(ios::right);
	(*m_ReportLog).width(20);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_States[state]->GetTitle() << ")  <- e" << event << endl;
}
```

```
void XAutomat::LogTrans(int OldState, int NewState)
{
	if (!m_TransLogging)
	{
		return;
	}
	(*m_ReportLog).width(8);
	(*m_ReportLog).setf(ios::left);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_Title;
	(*m_ReportLog).width(4);
	(*m_ReportLog).setf(ios::right);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << OldState << " ( ";
	(*m_ReportLog).unsetf(ios::right);
	(*m_ReportLog).width(20);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_States[OldState]->GetTitle() << ")  -> ";
	(*m_ReportLog).width(4);
	(*m_ReportLog).setf(ios::right);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << NewState << " ( ";
	(*m_ReportLog).unsetf(ios::right);
	(*m_ReportLog).width(20);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_States[NewState]->GetTitle() << ")"<< endl;
}

void XAutomat::LogEnd(int state, int event)
{
	if (!m_EndLogging)
	{
		return;
	}
	(*m_ReportLog).width(8);
	(*m_ReportLog).setf(ios::left);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_Title;
	(*m_ReportLog).width(4);
	(*m_ReportLog).setf(ios::right);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << state << " ( ";
	(*m_ReportLog).unsetf(ios::right);
	(*m_ReportLog).width(20);
	(*m_ReportLog).fill(' ');
	(*m_ReportLog) << m_States[state]->GetTitle() << ")  finished (e" << event << ")" << endl;


}

void XAutomat::LogError(int state)
{
	(*m_ReportLog) <<  m_Title << " have changed state to unknown (" << state << ")" << endl;
}
```

8.2. Файл ChildView.cpp

//Класс оболочки MFC, реализующий дополнительное окно, в котором происходит
//визуализация работы меню.

```
// ChildView.cpp : implementation of the CChildView class
//

#include "stdafx.h"
#include "vmenu.h"
#include "ChildView.h"

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

/////////////////////////////////////////////////////////////////////////////
// CChildView

CChildView::CChildView()
{
}

CChildView::~CChildView()
{
}


BEGIN_MESSAGE_MAP(CChildView,CWnd )
	//{{AFX_MSG_MAP(CChildView)
	ON_WM_PAINT()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CChildView message handlers

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
	if (!CWnd::PreCreateWindow(cs))
		return FALSE;

	cs.dwExStyle |= WS_EX_CLIENTEDGE;
	cs.style &= ~WS_BORDER;
	cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
		::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1), NULL);

	return TRUE;
}

void CChildView::OnPaint()
{
	CPaintDC dc(this); // device context for painting

	// TODO: Add your message handler code here

	// Do not call CWnd::OnPaint() for painting messages
}
```

8.3. Файл MainFrm.cpp

//Реализация вспомогательных классов оболочки MFC для визуализации работы меню.
//Класс главного окна. Конструирование и инициализация меню, обработка событий
//клавиатуры.

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "vmenu.h"

#include "MainFrm.h"

extern CVmenuApp theApp;
ofstream& log = theApp.fOut;

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

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	//{{AFX_MSG_MAP(CMainFrame)
	ON_WM_CREATE()
	ON_WM_SETFOCUS()
	ON_WM_KEYDOWN()
	ON_COMMAND(CHANGE_A, OnA)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,           // status line indicator
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
	// TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;
	// create a view to occupy the client area of the frame
	if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
		CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
	{
		TRACE0("Failed to create view window\n");
		return -1;
	}

	m_tmpWnd.InitLog(&theApp.fOut, true, true, true);
	m_tmpWnd.InitItemTitles(3);

	m_tmpWnd.InitItemTitles(3);
	m_tmpWnd.AddItemTitle("Пункт 1");
	m_tmpWnd.AddItemTitle("Пункт 2");
	m_tmpWnd.AddItemTitle("Пункт 3");
```

```
    m_tmpWnd.SetCoordinates(0, 0);

        if (!m_tmpWnd.Create(NULL, NULL, WS_CHILD,
                CRect(m_tmpWnd.GetMenuRect()), &m_wndView, 6666, NULL))
        {
                TRACE0("Failed to create my temporary window\n");
                return -1;
        }
        m_tmpWnd.Show();

        XEvent e(sizeof(int));

        e.m_Number = 1001;
        m_tmpWnd.ReceiveEvent(e);
        e.m_Number = 1002;
        *((int*)(e.m_Info)) = 1;
        m_tmpWnd.ReceiveEvent(e);

        m_tmpWnd.ShowWindow(theApp.m_nCmdShow);

/*****************************************************************************************
** Sub  Menu ****************************************************************************
*****************************************************************************************/

        m_tmpSub.SetDirection(MD_VERTICAL);
        m_tmpSub.InitLog(&theApp.fOut, true, true, true);
        m_tmpSub.InitItemTitles(3);

        int x, y;
        m_tmpWnd.GetSubMenuCoordinates(1, x, y);
        m_tmpSub.SetCoordinates(x, y);
        m_tmpSub.AddItemTitle("Пункт 1");
        m_tmpSub.AddItemTitle("Пункт 2");
        m_tmpSub.AddItemTitle("Пункт 3");


        if (!m_tmpSub.Create(NULL, NULL, WS_CHILD,
                CRect(m_tmpSub.GetMenuRect()), &m_wndView, 6667, NULL))
        {
                TRACE0("Failed to create my temporary window\n");
                return -1;
        }

        m_tmpSub.ShowWindow(theApp.m_nCmdShow);
        m_tmpSub.MoveWindow(0, 0, 0, 0);

/*****************************************************************************************
** Sub2 Menu ****************************************************************************
*****************************************************************************************/

        m_tmpSub2.SetDirection(MD_VERTICAL);
        m_tmpSub2.InitLog(&theApp.fOut, true, true, true);
        m_tmpSub2.InitItemTitles(3);

        m_tmpSub2.AddItemTitle("Пункт 1");
        m_tmpSub2.AddItemTitle("Пункт 2");
        m_tmpSub2.AddItemTitle("Пункт 3");

        m_tmpSub.GetSubMenuCoordinates(3, x, y);
    m_tmpSub2.SetCoordinates(x, y);
        if (!m_tmpSub2.Create(NULL, NULL, WS_CHILD,
                CRect(m_tmpSub2.GetMenuRect()), &m_wndView, 6667, NULL))
        {
                TRACE0("Failed to create my temporary window\n");
                return -1;
        }

        m_tmpSub2.ShowWindow(theApp.m_nCmdShow);
        m_tmpSub2.MoveWindow(0, 0, 0, 0);

        return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        if( !CFrameWnd::PreCreateWindow(cs) )
                return FALSE;

        cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
```

```
	cs.lpszClass = AfxRegisterWndClass(0);
	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
void CMainFrame::OnSetFocus(CWnd* pOldWnd)
{
	// forward focus to the view window
	// m_wndView.SetFocus();
}

BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo)
{
	// let the view have first crack at the command
	if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
		return TRUE;

	// otherwise, do default handling
	return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}


void CMainFrame::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{

	char tmp[2];
	tmp[0] = (char)nChar;
	tmp[1] = 0;

	log << endl << ">> Key pressed: " << tmp << " (" << nChar << ")" << endl;
	bool ok = 1;

	XEvent e(sizeof(int));
	if (nChar == 0x41)      // key A
	{
		e.m_Number = 1002;
		*((int*)(e.m_Info)) = 1;
	}
	else if (nChar == 0x57) // key W
	{
		e.m_Number = 1001;
	}
	else if (nChar == 0x43) // key C
	{
		e.m_Number = 1004;
	}
	else if (nChar == 0x4e) // key N
	{
		e.m_Number = 1005;
	}
	else if (nChar == 0x50) // key P
	{
		e.m_Number = 1006;
	}
	else if (nChar == 0x53) // key S
	{
		e.m_Number = 1003;
	}
	else if (nChar == VK_ESCAPE)
	{
		e.m_Number = 1007;
	}
	else if (nChar == VK_RETURN)
```

```
	{
		e.m_Number = 1008;
	} else {
		ok = 0;
	}

	if (ok) {
		m_tmpWnd.ReceiveEvent(e);
		m_tmpWnd.InvalidateRect(NULL);
		m_tmpSub.InvalidateRect(NULL);
		m_tmpSub2.InvalidateRect(NULL);
	}

	CFrameWnd::OnKeyDown(nChar, nRepCnt, nFlags);
}

void CMainFrame::OnA()
{
	int num = m_tmpSub2.GetItemsNumber();
	num++;

	char* str1 = new char[20];

	sprintf(str1, "Active %d", num);
	XCoolSub2Active3* sub2act4 = new XCoolSub2Active3(str1);
	sprintf(str1, "Selected %d", num);
	XMenuSelected* sub2sel4 = new XMenuSelected(str1);
	sub2act4->SetParent(&m_tmpSub2);
	sub2sel4->SetParent(&m_tmpSub2);
	m_tmpSub2.AddState(sub2act4);
	m_tmpSub2.AddState(sub2sel4);

	m_tmpSub2.ClearItemTitles();
	m_tmpSub2.InitItemTitles(num);

	for (int i = 0; i < num; i++)
	{
		sprintf(str1, "Пункт %d", (i+1));
		m_tmpSub2.AddItemTitle(str1);
	}

	m_tmpSub2.Show();
}
```

8.4. Файл vmenu.cpp

//Главный файл проекта. Реализация вспомогательных классов оболочки MFC.
//Инициализация и деинициализация лога.

```
// vmenu.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "vmenu.h"

#include "MainFrm.h"

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

/////////////////////////////////////////////////////////////////////////////
// CVmenuApp

BEGIN_MESSAGE_MAP(CVmenuApp, CWinApp)
	//{{AFX_MSG_MAP(CVmenuApp)
	ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CVmenuApp construction

CVmenuApp::CVmenuApp()
{
	fOut.open("log.log", ios::out);
}


CVmenuApp::~CVmenuApp()
{
	fOut.close();
}
/////////////////////////////////////////////////////////////////////////////
// The one and only CVmenuApp object

CVmenuApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CVmenuApp initialization

BOOL CVmenuApp::InitInstance()
{
	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	// Change the registry key under which our settings are stored.
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization.
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));


	// To create the main window, this code creates a new frame window
	// object and then sets it as the application's main window object.

	CMainFrame* pFrame = new CMainFrame;
	m_pMainWnd = pFrame;
```

```
	// create and load the frame with its resources

	pFrame->LoadFrame(IDR_MAINFRAME,
		WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, NULL,
		NULL);



	// The one and only window has been initialized, so show and update it.
	pFrame->ShowWindow(SW_SHOW);
	pFrame->UpdateWindow();

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CVmenuApp message handlers




/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
		// No message handlers
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CVmenuApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CVmenuApp message handlers
```

8.5. Файл XCustomMenu.cpp

//Содержит реализацию классов-состояний, унаследованных от базового
//класса-состояния.

```
#include <fstream>
#include "xcustommenu.h"

#include "mainfrm.h"
#include "vmenu.h"

extern CVmenuApp theApp;
ofstream& fOut2 = theApp.fOut;

XCoolActive1* act1;
XCoolActive2* act2;
XCoolActive3* act3;

XMenuSelected* sel1;
XCoolSelected2* sel2;
XMenuSelected* sel3;


void z10(XMenuWnd* a)
{
	a->MoveWindow(0, 0, 0, 0);
}

void z20(XMenuWnd* a)
{
	a->MoveWindow(&(a->GetMenuRect()));
}


int XCoolClosed::ChangeState(const XEvent &e)
{
	int next = XMenuClosed::ChangeState(e);
	if (next == 1)
	{
		z20((XMenuWnd*)GetParent());
	}
	return next;
}

void XCoolClosed::EnterInfluence()
{
	z10((XMenuWnd*)GetParent());

	XMenuClosed::EnterInfluence();
}


XCoolSub1Menu::XCoolSub1Menu(void)
{

	act1 = new XCoolActive1("Active 1");
	act2 = new XCoolActive2("Active 2");
	act3 = new XCoolActive3("Active 3");

	sel1 = new XMenuSelected("Selected 1");
	sel2 = new XCoolSelected2("Selected 2");
	sel3 = new XMenuSelected("Selected 3");

	SetTitle("A0");
	XCoolClosed* cls = new XCoolClosed("Closed");
	InsertState(cls, 0);
	((XStateEx*)(GetState(0)))->SetParent(this);
	((XStateEx*)(GetState(1)))->SetParent(this);
	((XStateEx*)(GetState(1)))->SetTitle("Waiting");
	act1->SetParent(this);
	act2->SetParent(this);
	act3->SetParent(this);
	sel1->SetParent(this);
	sel2->SetParent(this);
	sel3->SetParent(this);
	AddState(act1);
	AddState(sel1);
```

```
	AddState(act2);
	AddState(sel2);
	AddState(act3);
	AddState(sel3);
}

XCoolSub1Menu::~XCoolSub1Menu(void)
{
	TerminateLog();
	delete act1;
	delete act2;
	delete act3;
	delete sel1;
	delete sel2;
	delete sel3;
}


void z1_1()
{
	fOut2 << "action in z1_1" << endl;
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z1_1()!", "Menu action!",
MB_OK);
}

void z1_2()
{
	fOut2 << "action in z1_2" << endl;
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z1_2()!", "Menu action!",
MB_OK);
}

void z1_3()
{
	fOut2 << "action in z1_3" << endl;
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z1_3()!", "Menu action!",
MB_OK);
}


int XCoolActive1::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z1_1();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}

int XCoolActive2::ChangeState(const XEvent &e)
{
	return XMenuActive::ChangeState(e);
}

int XCoolSelected2::ChangeState(const XEvent &e)
{
	// Nested automat call here
	((CMainFrame*)(theApp.m_pMainWnd))->m_tmpSub.ReceiveEvent(e);

	return XMenuSelected::ChangeState(e);
}

int XCoolActive3::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z1_3();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}

void XCoolSelected2::EnterInfluence()
{
	XEvent e;
	e.m_Number = 0;
	// Nested automat initialization
	((CMainFrame*)(theApp.m_pMainWnd))->m_tmpSub.ReceiveEvent(e);
```

```
	XMenuSelected::EnterInfluence();
}
/*****************************************************************************************
** Sub Menu ******************************************************************************
*****************************************************************************************/

XCoolSubActive1* subact1;
XCoolSubActive2* subact2;
XCoolSubActive3* subact3;

XMenuSelected* subsel1;
XMenuSelected* subsel2;
XCoolSubSelected3* subsel3;


XCoolSub2Menu::XCoolSub2Menu(void)
{

	subact1 = new XCoolSubActive1("Active 1");
	subact2 = new XCoolSubActive2("Active 2");
	subact3 = new XCoolSubActive3("Active 3");

	subsel1 = new XMenuSelected("Selected 1");
	subsel2 = new XMenuSelected("Selected 2");
	subsel3 = new XCoolSubSelected3("Selected 3");

	SetTitle("A1");
	XCoolClosed* cls = new XCoolClosed("Closed");
	InsertState(cls, 0);
	((XStateEx*)(GetState(0)))->SetParent(this);
	((XStateEx*)(GetState(1)))->SetParent(this);
	((XStateEx*)(GetState(1)))->SetTitle("Waiting");
	subact1->SetParent(this);
	subact2->SetParent(this);
	subact3->SetParent(this);
	subsel1->SetParent(this);
	subsel2->SetParent(this);
	subsel3->SetParent(this);
	AddState(subact1);
	AddState(subsel1);
	AddState(subact2);
	AddState(subsel2);
	AddState(subact3);
	AddState(subsel3);
}

XCoolSub2Menu::~XCoolSub2Menu(void)
{
	TerminateLog();
	delete subact1;
	delete subact2;
	delete subact3;
	delete subsel1;
	delete subsel2;
	delete subsel3;
}


void z2_1()
{
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z2_1()!", "Menu action!",
MB_OK);
	fOut2 << "action in z2_1" << endl;
}

void z2_2()
{
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z2_2()!", "Menu action!",
MB_OK);
	fOut2 << "action in z2_2" << endl;
}

void z2_3()
{
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z2_3()!", "Menu action!",
MB_OK);
	fOut2 << "action in z2_3" << endl;
}
```

```
int XCoolSubActive1::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z2_1();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}

int XCoolSubActive2::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z2_2();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}

int XCoolSubActive3::ChangeState(const XEvent &e)
{
	return XMenuActive::ChangeState(e);
}

int XCoolSubSelected3::ChangeState(const XEvent &e)
{
	// Nested automat call here
	((CMainFrame*)(theApp.m_pMainWnd))->m_tmpSub2.ReceiveEvent(e);

	return XMenuSelected::ChangeState(e);
}

void XCoolSubSelected3::EnterInfluence()
{
	XEvent e;
	e.m_Number = 0;
	// Nested automat initialization
	((CMainFrame*)(theApp.m_pMainWnd))->m_tmpSub2.ReceiveEvent(e);

	XMenuSelected::EnterInfluence();
}

/*****************************************************************************************
** Sub2 Menu ******************************************************************************
*****************************************************************************************/
XCoolSub2Active1* sub2act1;
XCoolSub2Active2* sub2act2;
XCoolSub2Active3* sub2act3;

XMenuSelected* sub2sel1;
XMenuSelected* sub2sel2;
XMenuSelected* sub2sel3;


XCoolSub3Menu::XCoolSub3Menu(void)
{

	sub2act1 = new XCoolSub2Active1("Active 1");
	sub2act2 = new XCoolSub2Active2("Active 2");
	sub2act3 = new XCoolSub2Active3("Active 3");

	sub2sel1 = new XMenuSelected("Selected 1");
	sub2sel2 = new XMenuSelected("Selected 2");
	sub2sel3 = new XMenuSelected("Selected 3");

	SetTitle("A2");
	XCoolClosed* cls = new XCoolClosed("Closed");
	InsertState(cls, 0);
	((XStateEx*)(GetState(0)))->SetParent(this);
	((XStateEx*)(GetState(1)))->SetParent(this);
	((XStateEx*)(GetState(1)))->SetTitle("Waiting");
	sub2act1->SetParent(this);
	sub2act2->SetParent(this);
	sub2act3->SetParent(this);
	sub2sel1->SetParent(this);
	sub2sel2->SetParent(this);
	sub2sel3->SetParent(this);
	AddState(sub2act1);
	AddState(sub2sel1);
```

```
	AddState(sub2act2);
	AddState(sub2sel2);
	AddState(sub2act3);
	AddState(sub2sel3);
}

XCoolSub3Menu::~XCoolSub3Menu(void)
{
	TerminateLog();
	delete sub2act1;
	delete sub2act2;
	delete sub2act3;
	delete sub2sel1;
	delete sub2sel2;
	delete sub2sel3;
}


void z3_1()
{
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z3_1()!", "Menu action!",
MB_OK);
	fOut2 << "action in z3_1" << endl;
}

void z3_2()
{
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z3_2()!", "Menu action!",
MB_OK);
	fOut2 << "action in z3_2" << endl;
}

void z3_3()
{
	MessageBox(*((CMainFrame*)(theApp.m_pMainWnd)), "Action in z3_3()!", "Menu action!",
MB_OK);
	fOut2 << "action in z3_3" << endl;
}


int XCoolSub2Active1::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z3_1();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}

int XCoolSub2Active2::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z3_2();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}

int XCoolSub2Active3::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1008)
	{
		z3_3();
		return 0;
	}
	return XMenuActive::ChangeState(e);
}
```

8.6. Файл XMenuB.cpp

//Реализация класса, предоставляющего необходимые средства для реализации поведения //подменю.

```
#include "xmenub.h"

XMenuB::XMenuB(void) :  XAutomat("Untitled Menu", 2)
{
	XMenuClosed* closed = new XMenuClosed[1];
	XMenuWaiting* waiting = new XMenuWaiting[1];
	AddState(closed);
	AddState(waiting);

	m_counter = 0;
}

XMenuB::XMenuB(const char* clTitle, const char* wtTitle, int num) : XAutomat("Untitled Menu",
num)
{
	XMenuClosed* closed = new XMenuClosed[1];
	closed->SetTitle(clTitle);
	XMenuWaiting* waiting = new XMenuWaiting[1];
	waiting->SetTitle(wtTitle);
	AddState(closed);
	AddState(waiting);
}

XMenuB::~XMenuB(void)
{
}


XMenuClosed::XMenuClosed(void) : XStateEx("Untitled MenuClosed")
{
}

XMenuClosed::XMenuClosed(const char* title) : XStateEx(title)
{
}


XMenuClosed::~XMenuClosed(void)
{
}

int XMenuClosed::ChangeState(const XEvent &e)
{
	if (e.m_Number == 1001) return 1;
	return 0;
}

void XMenuClosed::EnterInfluence()
{
((XMenuB*)GetParent())->z3();
}


XMenuWaiting::XMenuWaiting(void) : XStateEx("Untitled MenuWaiting")
{
}

XMenuWaiting::XMenuWaiting(const char* title) : XStateEx(title)
{
}


XMenuWaiting::~XMenuWaiting(void)
{
}

int XMenuWaiting::ChangeState(const XEvent &e)
{
	switch (e.m_Number)
	{
	case 1004:
		return 0;
```

```
	case 1008:
		return 0;
	case 1007:
		return 0;
	case 1002:
		return (*((int*)(e.m_Info))) * 2;
	default:
		return 1;
	}
}

void XMenuWaiting::EnterInfluence()
{
}


XMenuSelected::XMenuSelected(void) : XStateEx("Untitled MenuSelected")
{
}

XMenuSelected::XMenuSelected(const char* title) : XStateEx(title)
{
}

XMenuSelected::~XMenuSelected(void)
{
}

int XMenuSelected::ChangeState(const XEvent &e)
{
	XMenuB* parent = (XMenuB*)GetParent();
	switch (e.m_Number)
	{
	case 1007:
		if (!parent->x1())
		{
			parent->z1();
			return parent->GetCurrentState() - 1;
		}
		if (parent->x1()){
			parent->z1();
			return GetParent()->GetCurrentState();
		}

	case 1003:
		parent->z2();
		return GetParent()->GetCurrentState();

	case 1008:
		return 0;

	default:
		return GetParent()->GetCurrentState();
		break;
	}
}

void XMenuSelected::EnterInfluence()
{
((XMenuB*)GetParent())->z2();
}


XMenuActive::XMenuActive(void) : XStateEx("Untitled MenuActive")
{
}

XMenuActive::XMenuActive(const char* title) : XStateEx(title)
{
}


XMenuActive::~XMenuActive(void)
{
}

int XMenuActive::ChangeState(const XEvent &e)
{
	XMenuB* tmp = (XMenuB*)GetParent();
	int curr = tmp->GetCurrentState();
```

```
        switch (e.m_Number)
        {
        case 1001:
                return 1;
        case 1002:
                return (*((int*)(e.m_Info))) * 2;
        case 1005:
                if (GetParent()->GetState(curr + 2) != 0)
                {
                        return curr + 2;
                }
                else
                {
                        return 2;
                }
                break;
        case 1006:
                if (curr > 2)
                {
                        return curr - 2;
                }
                else
                {
                        return GetParent()->GetStatesNumber() - 2;
                }

        case 1003:
                return GetParent()->GetCurrentState() + 1;
                break;
        case 1007:
                return 0;
                break;
        case 1008:
                // Action here (useful menu item action) depends on
                // implementation and might be redefined in a child classes.
                return 0;
        default:
                return GetParent()->GetCurrentState();
                break;

        }

}

void XMenuActive::EnterInfluence()
{
}

void XMenuB::z1()
{
        m_counter--;
}

void XMenuB::z2()
{
        m_counter++;
}

void XMenuB::z3()
{
        m_counter = 0;
}

bool XMenuB::x1()
{
        return m_counter > 1;
}
```

8.7. Файл xmenuwnd.cpp

//Реализация класса, объединяющего автоматную и графическую часть меню.

```
#include "StdAfx.h"
#include "XMenuWnd.h"

XMenuWnd::XMenuWnd(void)
{
}

XMenuWnd::~XMenuWnd(void)
{
}

BEGIN_MESSAGE_MAP(XMenuWnd, CWnd)
	ON_WM_PAINT()
END_MESSAGE_MAP()

void XMenuWnd::OnPaint()
{
	CPaintDC dc(this);

	dc.SelectObject(GetFont());

	int num = GetItemsNumber();

	int curr = (GetCurrentState() / 2) - 1;

	if (GetCurrentState() == 0)
	{
		return;
	}
	int act = GetCurrentState() % 2;
	for (int i = 0; i < num; i++)
	{
		if (i == curr)
		{
			if (act == 1)
			{
				DrawSelectedItem(&dc, i);
			}
			else
			{
				DrawActiveItem(&dc, i);
			}
		}
		else
		{
			DrawDefaultItem(&dc, i);
		}
	}
}

void XMenuWnd::GetSubMenuCoordinates(int num, int& x, int& y)
{
	RECT r = GetMenuRect();
	int w = GetItemWidth();
	int h = GetItemHeight();
	int x0 = r.left;
	int y0 = r.bottom;
	if (GetDirection() == MD_HORIZONTAL)
	{
		x = x0 + num * w;
		y = y0;
	}
	else
	{
		x = x0 + w;
		y = num * h;
	}
}
```

8.8. Файл XTmpWnd.cpp

//Реализация класса - графической оболочки.

```
#include "StdAfx.h"
#include "xtmpwnd.h"

IMPLEMENT_DYNAMIC(XTmpWnd, CWnd)
XTmpWnd::XTmpWnd(void)
{
	m_Direction = MD_HORIZONTAL;
	m_ItemsNumber = 0;
	m_ItemWidth = 150;
	m_ItemHeight = 30;

	m_ItemTitles = 0;

	m_X0 = m_Y0 = 0;

    m_DefaultColor = 0xc0c0c0;
    m_ActiveColor = 0xe77e7e;
	m_SelectedColor = 0x0000ff;
	m_TextColor = 0;
	m_CurrentTitle = 0;

	VERIFY(m_Font.CreateFont(
			20,                        // nHeight
			0,                         // nWidth
			0,                         // nEscapement
			0,                         // nOrientation
			FW_NORMAL,                 // nWeight
			TRUE,                     // bItalic
			FALSE,                     // bUnderline
			0,                         // cStrikeOut
			DEFAULT_CHARSET,              // nCharSet
			OUT_DEFAULT_PRECIS,        // nOutPrecision
			CLIP_DEFAULT_PRECIS,       // nClipPrecision
			DEFAULT_QUALITY,           // nQuality
			DEFAULT_PITCH | FF_SWISS,  // nPitchAndFamily
			"Times"));                 // lpszFacename
}

XTmpWnd::~XTmpWnd(void)
{
	ClearItemTitles();
	m_Font.DeleteObject();
}

void XTmpWnd::SetCoordinates(int x, int y)
{
	m_X0 = x;
	m_Y0 = y;
}

RECT XTmpWnd::GetMenuRect(void)
{
	RECT r;
	r.left  = m_X0;
	r.top   = m_Y0;
	if (m_Direction == MD_HORIZONTAL)
	{
		r.right = r.left + m_ItemsNumber * m_ItemWidth;
		r.bottom = r.top + m_ItemHeight;
	}
	else if (m_Direction == MD_VERTICAL)
	{
		r.right = r.left + m_ItemWidth;
		r.bottom = r.top + m_ItemsNumber * m_ItemHeight;
	}
	return r;
}

void XTmpWnd::GetSubMenuCoordinates(int &x, int &y, int num)
{
	if (m_Direction == MD_HORIZONTAL)
	{
		x = m_X0 + num * m_ItemWidth;
		y = m_Y0 + m_ItemHeight;
```

```
	}
	else if (m_Direction == MD_VERTICAL)
	{
		x = m_X0 + m_ItemWidth;
		y = m_Y0 + num * m_ItemHeight;
	}
}


void XTmpWnd::DrawDefaultItem(CPaintDC* dc, int num)
{
	DrawItem(dc, m_DefaultColor, num);
}

void XTmpWnd::DrawActiveItem(CPaintDC* dc, int num)
{
	DrawItem(dc, m_ActiveColor, num);
}

void XTmpWnd::DrawSelectedItem(CPaintDC* dc, int num)
{
	DrawItem(dc, m_SelectedColor, num);
}

void XTmpWnd::DrawItem(CPaintDC* dc, COLORREF c, int num)
{
	int x0, y0, x1, y1;
	if (m_Direction == MD_HORIZONTAL)
	{
		x0 = m_ItemWidth * num;
		y0 = 0;
	}
	else
	{
		x0 = 0;
		y0 = m_ItemHeight * num;
	}
	x1 = x0 + m_ItemWidth;
	y1 = y0 + m_ItemHeight;

	dc->FillSolidRect(x0, y0, m_ItemWidth, m_ItemHeight, c);

	RECT r;
	r.left = x0;
	r.top = y0;
	r.right = x1;
	r.bottom = y1;

	dc->DrawText(m_ItemTitles[num], -1, &r, DT_SINGLELINE | DT_VCENTER | DT_CENTER);
}

bool XTmpWnd::AddItemTitle(char* title)
{
	int len = strlen(title);
	if (m_CurrentTitle < (m_ItemsNumber))
	{
		m_ItemTitles[m_CurrentTitle] = new char[len + 1];
	 strcpy(m_ItemTitles[m_CurrentTitle], title);
		m_CurrentTitle++;
		return 1;
	}
	return 0;
}

bool XTmpWnd::InsertItemTitle(char* title, int num)
{
	int len = strlen(title);
	if (num < (m_ItemsNumber))
	{
		if (m_ItemTitles[num])
		{
			delete [] m_ItemTitles[num];
		}
		m_ItemTitles[num] = new char[len + 1];
	 strcpy(m_ItemTitles[num], title);
		return 1;
	}
	return 0;
}
```

```
void XTmpWnd::Show()
{
	if (m_Direction == MD_HORIZONTAL)
	{
		MoveWindow(m_X0, m_Y0, m_ItemsNumber * m_ItemWidth, m_ItemHeight);
	}
	else if (m_Direction == MD_VERTICAL)
	{
		MoveWindow(m_X0, m_Y0, m_ItemWidth, m_ItemsNumber * m_ItemHeight);
	}
}

void XTmpWnd::Hide()
{
	MoveWindow(m_X0, m_Y0, 0, 0);
}

void XTmpWnd::InitItemTitles(int num)
{
	m_ItemsNumber = num;
	m_ItemTitles = new char*[m_ItemsNumber];
	for (int i = 0; i < m_ItemsNumber; i++)
	{
		m_ItemTitles[i] = 0;
	}
}

void XTmpWnd::ClearItemTitles()
{
	if (m_ItemTitles)
	{
		for (int i = 0; i < m_ItemsNumber; i++)
		{
			delete [] m_ItemTitles[i];
		}
		delete [] m_ItemTitles;
	}
	m_ItemsNumber = 0;
	m_CurrentTitle = 0;
}

BEGIN_MESSAGE_MAP(XTmpWnd, CWnd)

END_MESSAGE_MAP()
```

Далее следуют заголовочные файлы для приведенных выше файлов.

8.9. Файл automat.h

```
/*==================================[ REDEFINITION DEFENCE ]=================*/
#ifndef       AUTOMAT_H
#define       AUTOMAT_H


#pragma once

#include <fstream>
using namespace std;

/*---------------------------------[ XEvent ]--------------------------*/
class XEvent
{
public:

/*================================[ PUBLIC CLASS ATRIBUTES ]=================*/

       int           m_Number;
       void*   m_Info;

/*================================[ PUBLIC    CLASS METHODS ]===============*/

       XEvent();
       XEvent(int n);
       ~XEvent();
};

/*---------------------------------[ XState ]--------------------------*/

class XState
{
/*================================[ PRIVATE CLASS ATRIBUTES ]=================*/

private:

       char* m_Title;

/*================================[ PUBLIC    CLASS METHODS ]===============*/

public:

       XState();
       XState(const char* title);

    void SetTitle(const char* title);
       char* GetTitle();

       virtual int ChangeState(const XEvent& e) = 0;
       virtual void EnterInfluence() = 0;
       ~XState();
};

/*---------------------------------[ XAutomat ]--------------------------*/

class XAutomat
{
/*================================[ PRIVATE CLASS ATRIBUTES ]=================*/

private:

       char*         m_Title;                    // automat's title

       XState**      m_States;                   // array of states
       int                  m_CurrentState;             // current state of automat
       int                  m_StatesNumber;             // number of states in automat
       int                  m_Available;          // number reserved states

       ofstream*     m_ReportLog;          // pointer to log file
       bool          m_BeginLogging;             // if true activate LogBegin
       bool          m_TransLogging;             // if true activate LogTrans
       bool          m_EndLogging;         // if true activate LogEnd
```

```
/*==================================[ PRIVATE   CLASS METHODS ]===============*/

private:

        void Automat(const XEvent& e);              // automat procedure

        void LogBegin(int state, int event);
        void LogTrans(int OldState, int NewState);
        void LogEnd(int state, int event);

        void LogError(int state);

/*==================================[ PUBLIC    CLASS METHODS ]===============*/

public:

        XAutomat();
        XAutomat(int num);
        XAutomat(const char* title, int num);

        ~XAutomat();

        void SetTitle(const char* title);

        void ReceiveEvent(const XEvent &e);  // interface function for other objects

        int GetCurrentState(void);

        int GetStatesNumber(void);

        XState* GetState(int num);

        void AddState(XState* e);            // add state to the end of automat

        // insert state into defined place
        bool InsertState(XState* e, int num);

        // initalize logging
        void InitLog(ofstream* fOut, bool begin, bool trans, bool end);

        // final automat's word
        void TerminateLog(void);

};

/*==================================[ END REDEFINITION DEFENCE ]==============*/
#endif         AUTOMAT_H
```

8.10. Файл ChildView.h

```
// ChildView.h : interface of the CChildView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_CHILDVIEW_H__CB5D033E_ED41_4CDA_9478_C3E4C420F28A__INCLUDED_)
#define AFX_CHILDVIEW_H__CB5D033E_ED41_4CDA_9478_C3E4C420F28A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CChildView window

class CChildView : public CWnd
{
// Construction
public:
	CChildView();

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CChildView)
	protected:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CChildView();

	// Generated message map functions
protected:
	//{{AFX_MSG(CChildView)
	afx_msg void OnPaint();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_CHILDVIEW_H__CB5D033E_ED41_4CDA_9478_C3E4C420F28A__INCLUDED_)
```

8.11. Файл MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__F3BFA83B_3177_4DD1_BF4B_D79584600F5A__INCLUDED_)
#define AFX_MAINFRM_H__F3BFA83B_3177_4DD1_BF4B_D79584600F5A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ChildView.h"
#include "xcustommenu.h"

class CMainFrame : public CFrameWnd
{

public:
	CMainFrame();
protected:
	DECLARE_DYNAMIC(CMainFrame)

// Attributes
public:
	XCoolSub1Menu  m_tmpWnd;
	XCoolSub2Menu  m_tmpSub;
	XCoolSub3Menu  m_tmpSub2;

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CMainFrame)
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
	virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
pHandlerInfo);
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CMainFrame();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
//	CStatusBar  m_wndStatusBar;
//	CToolBar    m_wndToolBar;
	CChildView    m_wndView;

// Generated message map functions
protected:
	//{{AFX_MSG(CMainFrame)
	afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
	afx_msg void OnSetFocus(CWnd *pOldWnd);
	afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
	afx_msg void OnA();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_MAINFRM_H__F3BFA83B_3177_4DD1_BF4B_D79584600F5A__INCLUDED_)
```

8.12. Файл vmenu.h

```
// vmenu.h : main header file for the VMENU application
//

#if !defined(AFX_VMENU_H__C8EBCD84_9639_4209_9DBF_AA66E24CEBDC__INCLUDED_)
#define AFX_VMENU_H__C8EBCD84_9639_4209_9DBF_AA66E24CEBDC__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
#include <fstream>
using namespace std;

/////////////////////////////////////////////////////////////////////////////
// CVmenuApp:
// See vmenu.cpp for the implementation of this class
//

class CVmenuApp : public CWinApp
{
public:
	CVmenuApp();

	ofstream      fOut;

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CVmenuApp)
	public:
	virtual BOOL InitInstance();
	//}}AFX_VIRTUAL

// Implementation

public:
	~CVmenuApp();
	//{{AFX_MSG(CVmenuApp)
	afx_msg void OnAppAbout();
		// NOTE - the ClassWizard will add and remove member functions here.
		//    DO NOT EDIT what you see in these blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_VMENU_H__C8EBCD84_9639_4209_9DBF_AA66E24CEBDC__INCLUDED_)
```

8.13. Файл XCustomMenuh

```
#ifndef XCustomMenu_H
#define XCustomMenu_H

#include "xmenuwnd.h"

void z10(XMenuWnd* a);
void z20(XMenuWnd* a);

void z1_1();
void z1_2();
void z1_3();
void z2_1();
void z2_2();
void z2_3();

/*****************************************************************************************
** Main Menu *****************************************************************************
*****************************************************************************************/

class XCoolSub1Menu : public XMenuWnd
{
public:
	XCoolSub1Menu();
	~XCoolSub1Menu();
};


class XCoolClosed :
	public XMenuClosed
{
public:
	XCoolClosed(void) {}
	XCoolClosed(const char* title) : XMenuClosed(title){}
	~XCoolClosed(void){}

	virtual int ChangeState(const XEvent &e);
	virtual void EnterInfluence();
};

class XCoolActive1 : public XMenuActive
{
public:
	XCoolActive1() {}
	XCoolActive1(const char* title) : XMenuActive(title) {}

	virtual int ChangeState(const XEvent& e);
	~XCoolActive1() {}
};

class XCoolActive2 : public XMenuActive
{
public:
	XCoolActive2() {}
	XCoolActive2(const char* title) : XMenuActive(title) {}

	virtual int ChangeState(const XEvent& e);
	~XCoolActive2() {}
};

class XCoolActive3 : public XMenuActive
{
public:
	XCoolActive3() {}
	XCoolActive3(const char* title) : XMenuActive(title) {}

	virtual int ChangeState(const XEvent& e);
	~XCoolActive3() {}
};

class XCoolSelected2 : public XMenuSelected
{
public:
	XCoolSelected2() {};
	XCoolSelected2(const char* title) : XMenuSelected(title) {}

	virtual int ChangeState(const XEvent& e);
	virtual void EnterInfluence();
```

```
        ~XCoolSelected2() {}
};

/*****************************************************************************************
** Sub  Menu ****************************************************************************
*****************************************************************************************/

class XCoolSub2Menu : public XMenuWnd//XMenuB
{
public:
        XCoolSub2Menu();
        ~XCoolSub2Menu();
};

class XCoolSubActive1 : public XMenuActive
{
public:
        XCoolSubActive1() {}
        XCoolSubActive1(const char* title) : XMenuActive(title) {}

        virtual int ChangeState(const XEvent& e);
        ~XCoolSubActive1() {}
};

class XCoolSubActive2 : public XMenuActive
{
public:
        XCoolSubActive2() {}
        XCoolSubActive2(const char* title) : XMenuActive(title) {}

        virtual int ChangeState(const XEvent& e);
        ~XCoolSubActive2() {}
};

class XCoolSubActive3 : public XMenuActive
{
public:
        XCoolSubActive3() {}
        XCoolSubActive3(const char* title) : XMenuActive(title) {}

        virtual int ChangeState(const XEvent& e);
        ~XCoolSubActive3() {}
};

class XCoolSubSelected3 : public XMenuSelected
{
public:
        XCoolSubSelected3() {}
        XCoolSubSelected3(const char* title) : XMenuSelected(title) {}

        virtual int ChangeState(const XEvent& e);
        virtual void EnterInfluence();
        ~XCoolSubSelected3() {}
};

/*****************************************************************************************
** Sub2 Menu ****************************************************************************
*****************************************************************************************/
class XCoolSub3Menu : public XMenuWnd
{
public:
        XCoolSub3Menu();
        ~XCoolSub3Menu();
};

class XCoolSub2Active1 : public XMenuActive
{
public:
        XCoolSub2Active1() {}
        XCoolSub2Active1(const char* title) : XMenuActive(title) {}

        virtual int ChangeState(const XEvent& e);
        ~XCoolSub2Active1() {}
};

class XCoolSub2Active2 : public XMenuActive
{
public:
        XCoolSub2Active2() {}
        XCoolSub2Active2(const char* title) : XMenuActive(title) {}
```

```
        virtual int ChangeState(const XEvent& e);
        ~XCoolSub2Active2() {}
};

class XCoolSub2Active3 : public XMenuActive
{
public:
        XCoolSub2Active3() {}
        XCoolSub2Active3(const char* title) : XMenuActive(title) {}

        virtual int ChangeState(const XEvent& e);
        ~XCoolSub2Active3() {}
};

#endif
```

8.14. Файл XMenuB.h

```
#pragma once

#ifndef       X_MENU_B_H
#define       X_MENU_B_H

#include    "automat.h"

class XMenuB :
       public XAutomat
{
public:
       int m_counter;
public:
       XMenuB(void);
       XMenuB(const char* closed, const char* waiting, int num);
       ~XMenuB(void);

       void z1();
       void z2();
       void z3();
       bool x1();
};

class XStateEx :
       public XState
{
private:
       XAutomat* m_Parent;
public:
       XStateEx(void) : XState("Untitled StateEx") { m_Parent = 0;}
       XStateEx(const char* title) : XState(title) { m_Parent = 0;}
       ~XStateEx(void) {}

       inline XAutomat* GetParent() {return m_Parent;}
       inline void SetParent(XAutomat* aP) {m_Parent = aP;}
};

class XMenuClosed :
       public XStateEx
{
public:
       XMenuClosed(void);
       XMenuClosed(const char*);

       ~XMenuClosed(void);

       virtual int ChangeState(const XEvent &e);
       virtual void EnterInfluence();
};

class XMenuWaiting :
       public XStateEx
{
public:
       XMenuWaiting(void);
       XMenuWaiting(const char*);
       ~XMenuWaiting(void);

       virtual int ChangeState(const XEvent &e);
       virtual void EnterInfluence();
};

class XMenuSelected :
       public XStateEx
{
public:
       XMenuSelected(void);
       XMenuSelected(const char*);
       ~XMenuSelected(void);

       virtual int ChangeState(const XEvent &e);
       virtual void EnterInfluence();
};

class XMenuActive :
       public XStateEx
{
```

```
public:
        XMenuActive(void);
        XMenuActive(const char*);
        ~XMenuActive(void);

        virtual int ChangeState(const XEvent &e);
        virtual void EnterInfluence();
};

#endif         X_MENU_B_H
```

8.15. Файл menuwnd.h

```
#include "xtmpwnd.h"
#include "xmenub.h"

class XMenuWnd : public XTmpWnd, public XMenuB
{
public:
	XMenuWnd(void);
	~XMenuWnd(void);

	void GetSubMenuCoordinates(int num, int& x, int& y);
	DECLARE_MESSAGE_MAP()
	afx_msg void OnPaint();
};
```

8.16. Файл XTmpWnd.h

```
#pragma once
#include "afxwin.h"

enum MENU_DIRECTION
{
	MD_VERTICAL  = false,
	MD_HORIZONTAL = true
};

class XTmpWnd :
	public CWnd
{
	DECLARE_DYNAMIC(XTmpWnd)

private:
	MENU_DIRECTION m_Direction;

	int			m_ItemsNumber;
	int			m_ItemWidth;
	int			m_ItemHeight;

	char**		m_ItemTitles;

	int			m_X0;
	int			m_Y0;

	COLORREF	m_DefaultColor;
	COLORREF	m_ActiveColor;
	COLORREF	m_SelectedColor;
	COLORREF	m_TextColor;

	CFont		m_Font;

	int			m_CurrentTitle;

public:
	XTmpWnd(void);
	~XTmpWnd(void);

	void SetCoordinates(int x, int y);
	inline void SetDirection(MENU_DIRECTION dir) {m_Direction = dir;}

	inline int GetItemWidth(void) {return m_ItemWidth;}
	inline int GetItemHeight(void) {return m_ItemHeight;}
	inline CFont* GetFont(void) {return &m_Font;}
	void GetSubMenuCoordinates(int &x, int &y, int num);

	RECT GetMenuRect(void);

	virtual void DrawDefaultItem(CPaintDC* dc, int num);
	virtual void DrawActiveItem(CPaintDC* dc, int num);
	virtual void DrawSelectedItem(CPaintDC* dc, int num);

	bool AddItemTitle(char* title);
	bool InsertItemTitle(char* title, int num);
	void InitItemTitles(int num);
	void ClearItemTitles();

	inline int GetItemsNumber(void){return m_ItemsNumber;}
	inline MENU_DIRECTION GetDirection(void) {return m_Direction;}

	void Show();
	void Hide();

private:
	void DrawItem(CPaintDC* dc, COLORREF c, int num);

protected:
	DECLARE_MESSAGE_MAP()
};
```